ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA MULTIPLATAFORMA DE MONITORIZACIÓN Y ADMINISTRACIÓN DE RED, CON INTERFAZ WEB PARA EL USUARIO Y UTILIZANDO EL PROTOCOLO SNMPv3

PROYECTO PREVIO A LA OBTENCIÓN DEL TITULO DE INGENIERO EN ELECTRÓNICA Y REDES DE LA INFORMACIÓN

FAUSTO VINICIO CASTAÑEDA VILLARREAL
Email: faustcast.cracker@hotmail.com

DIRECTOR: ING. XAVIER CALDERÓN, MSc.
Email: xavieralex_calderon@hotmail.com

Quito, febrero 2011

DECLARACIÓN

Yo, Fausto Vinicio Castañeda Villarreal, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

_____

Fausto Vinicio Castañeda Villarreal

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Fausto Vinicio Castañeda Villarreal, bajo mi supervisión.

_____

MSc. Calderón Hinojosa Xavier Alexander

DIRECTOR DE PROYECTO

AGRADECIMIENTOS

Agradezco primeramente a Dios Todopoderoso y a la Santísima Virgen María por su fundamental ayuda e inspiración para realizar este proyecto, a mis padres por su apoyo incondicional, a mi director por su valiosa guía, y a todos quienes han aportado con sus ideas en el desarrollo del mismo.

Fausto V. Castañeda V.

DEDICATORIA

Dedico este proyecto a mi Dios Todopoderoso y a mi Madre, la Santísima Virgen María, porque ellos son la razón de todos mis triunfos y logros alcanzados. Y a mis padres, porque ellos son mi mayor ejemplo de superación, perseveracia, dedicación, amor y fé.

Fausto V. Castañeda V.

ÍNDICE GENERAL

DECLARACIÓN ..... *ii*

CERTIFICACIÓN ..... *iii*

AGRADECIMIENTOS ..... *iv*

DEDICATORIA ..... *v*

ÍNDICE GENERAL ..... *vi*

ÍNDICE DE FIGURAS ..... *x*

ÍNDICE DE TABLAS ..... *xvii*

ÍNDICE DE CÓDIGOS FUENTE ..... *xxi*

RESUMEN ..... *xxii*

PRESENTACIÓN ..... *xxiii*

CAPÍTULO 1. SUSTENTACIÓN TEÓRICA *1*

1.1 CONCEPTOS BÁSICOS DE REDES DE COMPUTADORAS **2**

1.2 INTRODUCCIÓN A LA ADMINISTRACIÓN DE REDES [2] **3**

1.2.1 DEFINICIÓN DE ADMINISTRACIÓN DE RED [3] 4

1.2.2 FUNCIONES DE ADMINISTRACIÓN DE RED 5

1.2.3 SISTEMAS DE ADMINISTRACIÓN DE RED [6] 5

1.3 FUNDAMENTOS DEL PROTOCOLO SNMP [4][7] **7**

1.3.1 COMPONENTES DE SNMP [7] 8

1.3.2 FUNCIONAMIENTO DE SNMP [4] 9

1.3.3 RELACIÓN DE SNMP CON LOS PROTOCOLOS TCP/IP [4] 10

1.3.4 MECANISMOS DE SEGURIDAD BÁSICA [4][10] 12

1.3.5 INFORMACIÓN DE ADMINISTRACIÓN SNMP [4][8] 13

1.3.6 OPERACIONES SNMP [8][10] 26

1.4 SNMPv3 [7] **34**

1.4.1 CARACTERÍSTICAS DE SNMPv3 [8] 35

1.4.2 ARQUITECTURA SNMP [7][8][11] 36
1.4.3 PROCESAMIENTO Y ENVÍO DE MENSAJES SNMP [14] 58
1.4.4 MODELO DE SEGURIDAD BASADA EN EL USUARIO (USM) [8][11] 72
1.4.5 MODELO DE CONTROL DE ACCESO BASADO EN VISTAS (VACM) [8][11] 92

1.5 XML 105

1.5.1 INTRODUCCIÓN A XML [21][22][10] 105
1.5.2 DOCUMENTO XML [10][21][23][24] 105
1.5.3 VALIDACIÓN XML [10][21][23][24] 109
1.5.4 OTROS LENGUAJES BASADOS EN XML 114
1.5.5 HERRAMIENTAS PARA EL DESARROLLO DE APLICACIONES XML [10] [32] 118
1.5.6 XML EN LA ADMISTRACIÓN DE REDES [10][33] 119

1.6 TECNOLOGÍAS Y HERRAMIENTAS DE DESARROLLO WEB 122

1.6.1 JAVA [36] 122
1.6.2 JSP Y JAVA SERVLETS [30][37] 124
1.6.3 TOMCAT [30] 134
1.6.4 IDE ECLIPSE WTP [41] 136
1.6.5 ADOBE DREAMWEAVER [42] 139

CAPÍTULO 2. DESARROLLO DEL SISTEMA DE ADMINISTRACIÓN DE RED 141

2.1 ANÁLISIS DE REQUERIMIENTOS DEL SISTEMA [35] 142

2.1.1 RECOPILACIÓN DE REQUERIMIENTOS 142
2.1.2 MODELO DE CASOS DE USO 147
2.1.3 MODELO DE ANÁLISIS [35] 167

2.2 DISEÑO DEL SISTEMA [35] 202

2.2.1 MODELO DE DISEÑO 202
2.2.2 MODELO DE DESPLIEGUE 252

2.3 IMPLEMENTACIÓN DEL SISTEMA [35] 254

2.3.1 ENTORNO DE DESARROLLO 254
2.3.2 MODELO DE IMPLEMENTACIÓN [35] 254
2.3.3 CREACIÓN DE LA APLICACIÓN WEB 256

CAPÍTULO 3. PRUEBAS Y RESULTADOS OBTENIDOS 270

3.1 CARACTERÍSTICAS DE LA APLICACIÓN 271

3.1.1 DEFINICIÓN 271
3.1.2 CARACTERÍSTICAS PRINCIPALES 271
3.2 PRUEBAS 279
3.2.1 INICIAR SESIÓN 282
3.2.2 CONFIGURAR PARÁMETROS 284
3.2.3 DESCUBRIR ELEMENTOS DE RED 289
3.2.4 VISUALIZAR INFORMACIÓN DE ELEMENTO DE RED 293
3.2.5 NAVEGAR EN GRUPO MIB Y EJECUTAR OPERACIÓN SNMPv3 300
3.2.6 VISUALIZAR NOTIFICACIONES 311
3.2.7 GESTIONAR MIBS 313
3.2.8 GESTIONAR USUARIOS 316
3.2.9 GESTIONAR VISTAS 323
3.2.10 GESTIONAR DERECHOS DE ACCESO 329
3.2.11 CAMBIAR PASSWORDS 336
3.3 ANÁLISIS DE RESULTADOS 343
3.3.1 RESTRICCIONES DE USO 343
3.3.2 MENSAJES DE CONFIRMACIÓN Y EJECUCIÓN 353
3.3.3 ENVÍO Y RECEPCIÓN DE PAQUETES 358
3.3.4 ANÁLISIS DE LOS RESULTADOS 375
CAPÍTULO 4. CONCLUSIONES Y RECOMENDACIONES 378
4.1 CONCLUSIONES 378
4.2 RECOMENDACIONES 381
CAPÍTULO 5. BIBLIOGRAFÍA 383
5.1 BIBLIOGRAFÍA 383
5.2 REFERENCIAS DE INTERNET 385
ANEXOS 388
ANEXO A. CLASES DE ANÁLISIS Y DISEÑO 389
ANEXO B. DIAGRAMAS DE ESQUEMAS DE ARCHIVOS XML 431
ANEXO C. EJEMPLOS DE CÓDIGO FUENTE DE LA APLICACIÓN. 440

ANEXO D. CONFIGURACIÓN INICIAL PARA SNMPv3 450

ANEXO E. MANUAL DE USUARIO 453

E.1. REQUISITOS [44][30] 454

E.2. INSTALACIÓN 455

E.3. UTILIZACIÓN 459

E.4. DESINSTALACIÓN 482

ANEXO. F EJEMPLO BÁSICO DE UTILIZACIÓN DE JAVA SERVER PAGES, JAVA SERVLETS Y TOMCAT. 483

F.1. DESCRIPCIÓN DEL EJEMPLO 484

F.2. ANÁLISIS DE COMPONENTES 485

F.3. ANÁLISIS DE ARCHIVOS DE CONFIGURACIÓN 488

ANEXO G. ANÁLISIS COMPARATIVO CON OTRAS HERRAMIENTAS DE ADMINISTRACIÓN DE RED [45] 491

ÍNDICE DE FIGURAS

FIGURA 1. 1 EJEMPLO DE ARQUITECTURA DE UN SISTEMA DE GESTIÓN DE RED. 7
FIGURA 1. 2 COMPONENTES DE SNMP. 9
FIGURA 1. 3 FUNCIONAMIENTO DE SNMP. 10
FIGURA 1. 4 MODELO DE COMUNICACIÓN TCP/IP Y SNMP. 12
FIGURA 1. 5 UNA PARTE DEL ÁRBOL JERÁRQUICO DE OBJETOS ORGANIZADOS SEGÚN SMIV1. 15
FIGURA 1. 6 DESPLIEGUE DEL SUB-ÁRBOL MIB-2. 16
FIGURA 1. 7 EXTENSIÓN DEL ÁRBOL JERÁRQUICO DEFINIDO POR SMIV2. 18
FIGURA 1. 8 EJEMPLO DE RECURSIVIDAD DE LA ESTRUCTURA TLV. 23
FIGURA 1. 9 DETALLE DEL CAMPO TIPO. 24
FIGURA 1. 10 FORMATO CORTO DEL CAMPO LONGITUD. 25
FIGURA 1. 11 FORMATO LARGO DEL CAMPO LONGITUD. 25
FIGURA 1. 12 FORMATO DE MENSAJE SNMP (SNMPV1 Y SNMPV2). 26
FIGURA 1. 13 FORMATO DE PDU PARA GETREQUEST, GETNEXTREQUEST, SETREQUEST, GETREPONSE, INFORMREQUEST Y REPORT. 26
FIGURA 1. 14 FORMATO DE PDU PARA GETBULKREQUEST 29
FIGURA 1. 15 FORMATO DE PDU TRAP EN SNMPV1. 31
FIGURA 1. 16 FORMATO DE PDU TRAP EN SNMPV2. 31
FIGURA 1. 17 ENTIDAD SNMP (RFC 3411). 37
FIGURA 1. 18 GESTOR SNMP TRADICIONAL. 41
FIGURA 1. 19 AGENTE SNMP TRADICIONAL. 42
FIGURA 1. 20 ESTRUCTURA DEL MENSAJE SNMPV3. 59
FIGURA 1. 21 DIAGRAMA DE FUNCIONAMIENTO PARA UN GENERADOR DE COMANDOS U ORIGINADOR DE NOTIFICACIONES. 65
FIGURA 1. 22 DIAGRAMA DE FUNCIONAMIENTO PARA UN CONTESTADOR DE COMANDOS O RECEPTOR DE NOTIFICACIONES. 69
FIGURA 1. 23 DIAGRAMA DE FLUJO PARA EL PROCESO DE SINCRONIZACIÓN. 79
FIGURA 1. 24 DIAGRAMA DE FLUJO DE VACM. 104
FIGURA 1. 25 DIAGRAMA DE TRADUCCIÓN Y COMPILACIÓN DE UNA PÁGINA JSP A SERVLET. 127
FIGURA 1. 26 MODELO DE ARQUITECTURA 1 PARA APLICACIONES JSP. 129
FIGURA 1. 27 MODELO DE ARQUITECTURA 2 PARA APLICACIONES JSP. 131
FIGURA 1. 28 ARQUITECTURA DE TOMCAT. 135
FIGURA 1. 29 ECLIPSE WEB TOOLS PLATAFORM PROJECT. 137

FIGURA 1. 30 ADOBE DREAMWEAVER. 139
FIGURA 2. 1 ACTORES 147
FIGURA 2. 2 DIAGRAMA DE CASOS DE USO DE LA APLICACIÓN. 152
FIGURA 2. 3 DIAGRAMA DE INTERACCIÓN DE CLASES DE ANÁLISIS DE: INICIO DE SESIÓN. 169
FIGURA 2. 4 DIAGRAMA DE INTERACCIÓN DE CLASES DE ANÁLISIS DE: CONFIGURAR PARÁMETROS. 170
FIGURA 2. 5 DIAGRAMA DE INTERACCIÓN DE CLASES DE ANÁLISIS DE: DESCUBRIR ELEMENTOS DE RED. 172
FIGURA 2. 6 DIAGRAMA DE INTERACCIÓN DE CLASES DE ANÁLISIS DE: VISUALIZAR INFORMACIÓN DE ELEMENTO DE RED. 174
FIGURA 2. 7 DIAGRAMA DE INTERACCIÓN DE CLASES DE ANÁLISIS DE: NAVEGAR EN UN GRUPO MIB. 175
FIGURA 2. 8 DIAGRAMA DE INTERACCIÓN DE CLASES DE ANÁLISIS DE: EJECUTAR OPERACIÓN SNMPV3. 176
FIGURA 2. 9 DIAGRAMA DE INTERACCIÓN DE CLASES DE ANÁLISIS DE: VISUALIZAR NOTIFICACIONES. 178
FIGURA 2. 10 DIAGRAMA DE INTERACCIÓN DE CLASES DE ANÁLISIS DE: GESTIONAR MIBS. 180
FIGURA 2. 11 DIAGRAMA DE INTERACCIÓN DE CLASES DE ANÁLISIS DE: GESTIONAR USUARIOS. 182
FIGURA 2. 12 DIAGRAMA DE INTERACCIÓN DE CLASES DE ANÁLISIS DE: GESTIONAR VISTAS. 184
FIGURA 2. 13 DIAGRAMA DE INTERACCIÓN DE CLASES DE ANÁLISIS DE: GESTIONAR DERECHOS DE ACCESO. 187
FIGURA 2. 14 DIAGRAMA DE INTERACCIÓN DE CLASES DE ANÁLISIS DE: CAMBIAR PASSWORDS. 189
FIGURA 2. 15 ESTRUCUTRA DE PAQUETE DE ANÁLISIS: GESTÓN DE SESIÓN. 197
FIGURA 2. 16 ESTRUCTURA DE PAQUETE DE ANÁLISIS: GESTÓN DE CONFIGURACIÓN. 197
FIGURA 2. 17 ESTRUCTURA DE PAQUETE DE ANÁLISIS: GESTIÓN DE ELEMENTOS. 198
FIGURA 2. 18 ESTRUCTURA DE PAQUETE DE ANÁLISIS: GESTIÓN DE MIBS. 198
FIGURA 2. 19 ESTRUCTURA DE PAQUETE DE ANÁLISIS: GESTIÓN DE NOTIFICACIONES. 199
FIGURA 2. 20 ESTRUCTURA DE PAQUETE DE ANÁLISIS: GESTIÓN DE CAMBIO DE PASSWORDS. 199
FIGURA 2. 21 ESTRUCTURA DE PAQUETE DE ANÁLISIS: GESTIÓN GENERAL. 200
FIGURA 2. 22 DEPENDENCIAS DE PAQUETES DE ANÁLSIS. 201
FIGURA 2. 23 DIAGRAMA DE CLASES DEL CASO DE USO: INICIAR SESIÓN. 204
FIGURA 2. 24 DIAGRAMA DE COLABORACIÓN DEL CASO DE USO: INICIAR SESIÓN. 204
FIGURA 2. 25 DIAGRAMA DE SECUENCIA DEL CASO DE USO: INICIAR SESIÓN. 205
FIGURA 2. 26 DIAGRAMA DE CLASES DEL CASO DE USO: CONFIGURAR PARÁMETROS. 206
FIGURA 2. 27 DIAGRAMA DE SECUENCIA DEL CASO DE USO: CONFIGURAR PARÁMETROS. 208
FIGURA 2. 28 DIAGRAMA DE CLASES DEL CASO DE USO: DESCUBRIR ELEMENTOS DE RED. 210
FIGURA 2. 29 DIAGRAMA DE COLABORACIÓN DEL CASO DE USO: DESCUBRIR ELEMENTOS DE RED. 211
FIGURA 2. 30 DIAGRAMA DE SECUENCIA DEL CASO DE USO: DESCUBRIR ELEMENTOS DE RED. 212
FIGURA 2. 31 DIAGRAMA DE CLASES DEL CASO DE USO: VISUALIZAR INFORMACIÓN DE ELEMENTO DE RED. 214
FIGURA 2. 32 DIAGRAMA DE COLABORACIÓN DEL CASO DE USO: VISUALIZAR INFORMACIÓN DE ELEMENTO DE RED. 215

FIGURA 2. 33 DIAGRAMA DE SECUENCIA DEL CASO DE USO: VISUALIZAR INFORMACIÓN DE ELEMENTO DE RED. 216
FIGURA 2. 34 DIAGRAMA DE CLASES DEL CASO DE USO: NAVEGAR EN UN GRUPO MIB. 218
FIGURA 2. 35 DIAGRAMA DE COLABORACIÓN DEL CASO DE USO: NAVEGAR EN UN GRUPO MIB. 219
FIGURA 2. 36 DIAGRAMA DE SECUENCIA DEL CASO DE USO: NAVEGAR EN UN GRUPO MIB. 220
FIGURA 2. 37 DIAGRAMA DE CLASES DEL CASO DE USO: EJECUTAR OPERACIÓN SNMPV3. 221
FIGURA 2. 38 DIAGRAMA DE COLABORACIÓN DEL CASO DE USO: EJECUTAR OPERACIÓN SNMPV3. 222
FIGURA 2. 39 DIAGRAMA DE SECUENCIA DEL CASO DE USO: EJECUTAR OPERACIÓN SNMPV3. 223
FIGURA 2. 40 DIAGRAMA DE CLASES DEL CASO DE USO: VISUALIZAR NOTIFICACIONES. 225
FIGURA 2. 41 DIAGRAMA DE COLABORACIÓN DEL CASO DE USO: VISUALIZAR NOTIFICACIONES. 225
FIGURA 2. 42 DIAGRAMA DE SECUENCIA DEL CASO DE USO: VISUALIZAR NOTIFICACIONES. 226
FIGURA 2. 43 DIAGRAMA DE CLASES DEL CASO DE USO: GESTIONAR MIBS. 228
FIGURA 2. 44 DIAGRAMA DE COLABORACIÓN DEL CASO DE USO: GESTIONAR MIBS. 228
FIGURA 2. 45 DIAGRAMA DE SECUENCIA DEL CASO DE USO: GESTIONAR MIBS. 229
FIGURA 2. 46 DIAGRAMA DE CLASES DEL CASO DE USO: GESTIONAR USUARIOS. 231
FIGURA 2. 47 DIAGRAMA DE COLABORACIÓN DEL CASO DE USO: GESTIONAR USUARIOS. 232
FIGURA 2. 48 DIAGRAMA DE SECUENCIA DEL CASO DE USO: GESTIONAR USUARIOS. 233
FIGURA 2. 49 DIAGRAMA DE CLASES DEL CASO DE USO: GESTIONAR VISTAS. 235
FIGURA 2. 50 DIAGRAMA DE COLABORACIÓN DEL CASO DE USO: GESTIONAR VISTAS. 236
FIGURA 2. 51 DIAGRAMA DE SECUENCIA DEL CASO DE USO: GESTIONAR VISTAS. 237
FIGURA 2. 52 DIAGRAMA DE CLASES DEL CASO DE USO: GESTIONAR DERECHOS DE ACCESO. 239
FIGURA 2. 53 DIAGRAMA DE COLABORACIÓN DEL CASO DE USO: GESTIONAR DERECHOS DE ACCESO. 240
FIGURA 2. 54 DIAGRAMA DE SECUENCIA DEL CASO DE USO: GESTIONAR DERECHOS DE ACCESO. 241
FIGURA 2. 55 DIAGRAMA DE CLASES DEL CASO DE USO: CAMBIAR PASSWORDS. 243
FIGURA 2. 56 DIAGRAMA DE COLABORACIÓN DEL CASO DE USO: CAMBIAR PASSWORDS. 244
FIGURA 2. 57 DIAGRAMA DE SECUENCIA DEL CASO DE USO: CAMBIAR PASSWORDS. 245
FIGURA 2. 58 EJEMPLO DE GRÁFICO DE CLASE DE DISEÑO. 247
FIGURA 2. 59 SUBSISTEMA DE DISEÑO IU DE ADMINISTRADOR LIMITADO/GENERAL. 249
FIGURA 2. 60 DISEÑO DE INTERFAZ GRÁFICA DE USUARIO PARA IUBIENVENIDA. 249
FIGURA 2. 61 DISEÑO DE INTERFAZ GRÁFICA DE USUARIO PARA IUINICIOSESION 250
FIGURA 2. 62 DISEÑO DE INTERFAZ GRÁFICA DE USUARIO PARA IUELEMENTOS Y IUGRUPOMIB. 250
FIGURA 2. 63 DISEÑO DE INTERFAZ GRÁFICA DE USUARIO PARA LAS CLASES DE INTERFAZ RESTANTES. 251
FIGURA 2. 64 DIAGRAMA DE DISTRIBUCIÓN DE LA APLICACIÓN. 253
FIGURA 2. 65 DISTRIBUCIÓN DE SUBSISTEMAS EN LOS NODOS. 253
FIGURA 2. 66 ESTRUCTURA DEL ARCHIVO XML PARA INFORMACIÓN DE RED. 267

FIGURA 3. 1 PÁGINA DE BIENVENIDA. 273
FIGURA 3. 2 PÁGINA DE INICIO DE SESIÓN. 273
FIGURA 3. 3 PÁGINA DE DESCUBRIMIENTO DE HOSTS. 274
FIGURA 3. 4 PÁGINA DE ELEMENTOS. 274
FIGURA 3. 5 MENÚ PRIMARIO. 275
FIGURA 3. 6 SUBMENÚ DEL ENLACE CONFIGURACIÓN. 276
FIGURA 3. 7 SUBMENÚ DEL ENLACE GESTIÓN GENERAL. 277
FIGURA 3. 8 SUBMENÚ DEL ENLACE DE USUARIO QUE HA INICIADO SESIÓN. 278
FIGURA 3. 9 TOPOLOGÍA LÓGICA DE RED USADA COMO AMBIENTE DE PRUEBAS. 280
FIGURA 3. 10 URL DE INGRESO A LA APLICACIÓN SWAR DEL SERVIDOR XP DESDE LA MÁQUINA CLIENTE. 282
FIGURA 3. 11 MENSAJE DE ERROR DE CERTIFICADO. 282
FIGURA 3. 12 PÁGINA DE BIENVENIDA DE LA APLICACIÓN. 283
FIGURA 3. 13 PÁGINA DE INICIO DE SESIÓN. 283
FIGURA 3. 14 PÁGINA PRINCIPAL DE LA APLICACIÓN. 284
FIGURA 3. 15 CUERPO DE LA PÁGINA CONFIGURACIÓN DE PARÁMETROS DE MENSAJES SNMPV3. 285
FIGURA 3. 16 CUERPO DE LA PÁGINA CONFIGURACIÓN DE PARÁMETROS DE SERVIDOR. 288
FIGURA 3. 17 CUERPO DE LA PÁGINA DE OPCIONES DE DESCUBRIMIENTO DE HOSTS. 290
FIGURA 3. 18 PÁGINA DE ESPERA. 291
FIGURA 3. 19 CUERPO DE LA PÁGINA DE RESULTADOS DE HOSTS ACTIVOS. 292
FIGURA 3. 20 PARTE DE LA PÁGINA DE ELEMENTOS DE RED ACTIVOS. 293
FIGURA 3. 21 EJEMPLO DE CONSULTA DE INFORMACIÓN DE RED. 294
FIGURA 3. 22 EJEMPLO DE INFORMACIÓN BÁSICA DE UN ELEMENTO DE RED. 295
FIGURA 3. 23 CONSULTA DE INFORMACIÓN DE SISTEMA DEL ELEMENTO 172.31.1.6. 297
FIGURA 3. 24 CONSULTA DE INFORMACIÓN DE DISPOSITIVOS DEL ELEMENTO 172.31.1.6. 297
FIGURA 3. 25 CONSULTA DE INFORMACIÓN DE INTERFACES DEL ELEMENTO 172.31.1.6. 298
FIGURA 3. 26 CONSULTA DE INFORMACIÓN DE ALMACENAMIENTO DEL ELEMENTO 172.31.1.6. 298
FIGURA 3. 27 CONSULTA DE INFORMACIÓN DE SOFTWARE DEL ELEMENTO 172.31.1.6. 299
FIGURA 3. 28 CONSULTA DE INFORMACIÓN DE PROCESOS DEL ELEMENTO 172.31.1.6. 299
FIGURA 3. 29 CUERPO DE LA PÁGINA DE NAVEGACIÓN EN UN GRUPO MIB Y EJECUCIÓN DE OPERACIÓN SNMPV3. 300
FIGURA 3. 30 PRESENTACIÓN DE PROPIEDADES DE UN NODO MIB DEL TIPO OBJETO. 301
FIGURA 3. 31 PRESENTACIÓN DE PROPIEDADES DE UN NODO MIB DEL TIPO OBJECT-IDENTITY. 302
FIGURA 3. 32 DIAGRAMA DE FLUJO PARA RESTRICCIONES DE OPERACIONES SNMPV3. 304
FIGURA 3. 33 EJECUCIÓN Y RESULTADO DE UNA OPERACIÓN *GET*. 305
FIGURA 3. 34 EJECUCIÓN Y RESULTADO DE UNA OPERACIÓN *GETNEXT*. 306

FIGURA 3. 35 EJECUCIÓN Y RESULTADO DE UNA OPERACIÓN *GETBULK*. 307
FIGURA 3. 36 EJECUCIÓN Y RESULTADO DE UNA OPERACIÓN *WALK*. 307
FIGURA 3. 37 OBTENCIÓN DEL VALOR ACTUAL DEL OBJETO SYSLOCATION PREVIO A LA OPERACIÓN *SET* 308
FIGURA 3. 38 EJECUCIÓN Y RESULTADO DE UNA OPERACIÓN *SET*. 309
FIGURA 3. 39 EJECUCIÓN DE UNA OPERACIÓN *TRAP*. 309
FIGURA 3. 40 CUERPO DE LA PÁGINA DE NOTIFICACIONES DEL SISTEMA SWAR UBICADO EN EL SERVIDOR LINUX. 310
FIGURA 3. 41 EJECUCIÓN Y RESULTADO DE UNA OPERACIÓN *INFORM*. 311
FIGURA 3. 42 CUERPO DE LA PÁGINA DE NOTIFICACIONES. 311
FIGURA 3. 43 INFORMACIÓN DE DETALLE DE UNA NOTIFICACIÓN. 312
FIGURA 3. 44 CUERPO DE LA PÁGINA DE GESTIÓN DE MIBS. 313
FIGURA 3. 45 VENTANA DE SELECCIÓN DEL ARCHIVO MIB A SUBIR. 314
FIGURA 3. 46 LISTA ACTUALIZADA DE GRUPOS MIBS Y MENSAJE DE ÉXITO DE LA OPERACIÓN. 315
FIGURA 3. 47 CUERPO DE LA PÁGINA DE GESTIÓN DE USUARIOS. 316
FIGURA 3. 48 CUERPO DE LA PÁGINA DE EXPORTACIÓN DE USUARIO. 318
FIGURA 3. 49 INFORME DE LOS RESULTADOS DE LA EXPORTACIÓN DE USUARIO. 319
FIGURA 3. 50 CUERPO DE LA PÁGINA DE ANULACIÓN DE USUARIO. 321
FIGURA 3. 51 RESULTADOS DEL PROCESO DE ANULACIÓN DE UN USUARIO. 322
FIGURA 3. 52 CUERPO DE LA PÁGINA DE GESTIÓN DE VISTAS. 323
FIGURA 3. 53 CUERPO DE LA PÁGINA DE EXPORTACIÓN DE VISTA. 325
FIGURA 3. 54 INFORME DE LOS RESULTADOS DE LA EXPORTACIÓN DE UNA VISTA. 326
FIGURA 3. 55 CUERPO DE LA PÁGINA DE ANULACIÓN DE VISTA. 327
FIGURA 3. 56 RESULTADOS DEL PROCESO DE ANULACIÓN DE UNA VISTA. 328
FIGURA 3. 57 CUERPO DE LA PÁGINA DE GESTIÓN DE DERECHOS DE ACCESO. 329
FIGURA 3. 58 CUERPO DE LA PÁGINA DE EXPORTACIÓN DE DERECHO DE ACCESO. 332
FIGURA 3. 59 INFORME DE LOS RESULTADOS DE LA EXPORTACIÓN DE UN DERECHO DE ACCESO. 333
FIGURA 3. 60 CUERPO DE LA PÁGINA DE ANULACIÓN DE DERECHO DE ACCESO. 334
FIGURA 3. 61 RESULTADOS DEL PROCESO DE ANULACIÓN DE UN DERECHO DE ACCESO. 335
FIGURA 3. 62 INICIO DE SESIÓN DEL NUEVO USUARIO. 336
FIGURA 3. 63 NAVEGACIÓN EN EL GRUPO MIB *SYSTEM* Y EJECUCIÓN DE UNA OPERACIÓN *GET*. 337
FIGURA 3. 64 NAVEGACIÓN EN EL GRUPO MIB *SNMP* Y EJECUCIÓN DE UNA OPERACIÓN *GETNEXT*. 337
FIGURA 3. 65 CUERPO DE LA PÁGINA DE CAMBIO DE PASSWORDS DE USUARIO. 339
FIGURA 3. 66 CUERPO DE LA PÁGINA DE EXPORTACIÓN DE CAMBIO DE PASSWORDS DE USUARIO. 340
FIGURA 3. 67 INFORME DE LOS RESULTADOS DE LA EXPORTACIÓN DE CAMBIO DE PASSWORDS DE USUARIO. 341
FIGURA 3. 68 EJEMPLO DE MENSAJE DE ADVERTENCIA EN LA PÁGINA DE INICIO DE SESIÓN. 344

FIGURA 3. 69 EJEMPLOS DE MENSAJES DE ADVERTENCIA EN LA PÁGINA DE CONFIGURACIÓN DE MENSAJES SNMPV3. 345
FIGURA 3. 70 EJEMPLOS DE MENSAJES DE ADVERTENCIA EN LA PÁGINA DE CONFIGURACIÓN DE SERVIDOR. . 346
FIGURA 3. 71 EJEMPLOS DE MENSAJES DE ADVERTENCIA EN LA PÁGINA DE DESCUBRIMIENTO DE HOSTS. 347
FIGURA 3. 72 EJEMPLOS DE MENSAJES DE ADVERTENCIA EN LA PÁGINA DE NAVEGACIÓN EN UN GRUPO MIB. 347
FIGURA 3. 73 EJEMPLOS DE MENSAJES DE ADVERTENCIA EN LA PÁGINA DE GESTIÓN DE USUARIOS. 348
FIGURA 3. 74 EJEMPLOS DE MENSAJES DE ADVERTENCIA EN LA PÁGINA DE GESTIÓN DE VISTAS. 349
FIGURA 3. 75 EJEMPLOS DE MENSAJES DE ADVERTENCIA EN LA PÁGINA DE GESTIÓN DE DERECHOS DE ACCESO. 350
FIGURA 3. 76 EJEMPLOS DE MENSAJES DE ADVERTENCIA EN LA PÁGINA DE CAMBIO DE PASSWORDS DE USUARIO. 351
FIGURA 3. 77 MENSAJE DE ADVERTENCIA POR INTENTO DE INGRESO A PÁGINAS DE USO EXCLUSIVO DEL ADMINISTRADOR GENERAL. 352
FIGURA 3. 78 EJEMPLO DE MENSAJE DE EXCEPCIÓN POR ALTERACIÓN INDEBIDA DEL ARCHIVO XML QUE GUARDA INFORMACIÓN DE PARÁMETROS DE MENSAJES SNMPV3 353
FIGURA 3. 79 EJEMPLO DE MENSAJE DE CONFIRMACIÓN. 354
FIGURA 3. 80 EJEMPLO DE MENSAJE DE EJECUCIÓN. 354
FIGURA 3. 81 EJEMPLO DE MENSAJE DE EJECUCIÓN LUEGO DE UNA OPERACIÓN *GETNEXT*. 356
FIGURA 3. 82 EJEMPLO DE MENSAJES DE EJECUCIÓN EN INFORME DE ANULACIÓN DE UNA CUENTA DE USUARIO. 357
FIGURA 3. 83 EJEMPLO DE MENSAJE DE EJECUCIÓN EN LA PÁGINA DE GESTIÓN DE MIBS. 357
FIGURA 3. 84 SECUENCIA DE EJECUCIÓN DE UN MENSAJE SNMPV3. 359
FIGURA 3. 85 PAQUETES CAPTURADOS EN LA EJECUCIÓN DE UNA OPERACIÓN SNMPV3. 360
FIGURA 3. 86 DETALLE DEL PRIMER MENSAJE DE SOLICITUD PARA DESCUBRIMIENTO USM. 361
FIGURA 3. 87 DETALLE DEL SEGUNDO MENSAJE DE REPORTE PARA DESCUBRIMIENTO USM. 362
FIGURA 3. 88 DETALLE DEL TERCER MENSAJE DE SOLICITUD PARA DESCUBRIMIENTO USM. 363
FIGURA 3. 89 DETALLE DEL CUARTO MENSAJE DE REPORTE PARA DESCUBRIMIENTO USM. 363
FIGURA 3. 90 COMPROBACIÓN DEL RESULTADO DE UNA OPERACIÓN *GETREQUEST*. 364
FIGURA 3. 91 COMPROBACIÓN DE LA PROTECCIÓN CONTRA REVELACIÓN. 365
FIGURA 3. 92 COMPROBACIÓN DEL RESULTADO DE UNA OPERACIÓN *GETNEXTREQUEST*. 366
FIGURA 3. 93 COMPROBACIÓN DEL RESULTADO DE UNA OPERACIÓN *GETBULKREQUEST*. 367
FIGURA 3. 94 COMPROBACIÓN DEL RESULTADO DE UNA OPERACIÓN *WALK*. 368
FIGURA 3. 95 COMPROBACIÓN DEL RESULTADO DE UNA OPERACIÓN *SETREQUEST*. 369
FIGURA 3. 96 CAPTURA DE UN MENSAJE *TRAP*. 370

FIGURA 3. 97 COMPROBACIÓN DEL RESULTADO DE UNA OPERACIÓN *INFORMREQUEST*..... 371
FIGURA 3. 98 COMPROBACIÓN DEL RESULTADO DE UNA EXPORTACIÓN DE CAMBIO DE PASSWORDS DE USUARIO. 372
FIGURA 3. 99 DETALLE DE UN MENSAJE DE RESPUESTA EN EL PROCESO DE EXPORTACIÓN DE CAMBIO DE PASSWORDS. 373
FIGURA A. 1 DIAGRAMA GENERAL DE CLASES DE DISEÑO. 430
FIGURA B. 1 ESTRUCTURA DE ARCHIVO XML PARA INFORMACIÓN DE DERECHO DE ACCESO. 432
FIGURA B. 2 ESTRUCTURA DE ARCHIVO XML PARA INFORMACIÓN DE ELEMENTO DE RED. 433
FIGURA B. 3 ESTRUCTURA DE ARCHIVO XML PARA INFORMACIÓN DE GRUPO VACM. 433
FIGURA B. 4 ESTRUCTURA DE ARCHIVO XML PARA INFORMACIÓN DE PARÁMETROS DE MENSAJE SNMPV3. .. 434
FIGURA B. 5 ESTRUCTURA DE ARCHIVO XML PARA INFORMACIÓN DE GRUPO MIB. 434
FIGURA B. 6 ESTRUCTURA DE ARCHIVO XML PARA INFORMACIÓN DE NODOS MIB. 435
FIGURA B. 7 ESTRUCTURA DE ARCHIVO XML PARA INFORMACIÓN DE NOTIFICACIÓN. 435
FIGURA B. 8 ESTRUCTURA DE ARCHIVO XML PARA INFORMACIÓN DE RED. 436
FIGURA B. 9 ESTRUCTURA DE ARCHIVO XML PARA INFORMACIÓN DE REGISTRO DE DERECHO DE ACCESO. 436
FIGURA B. 10 ESTRUCTURA DE ARCHIVO XML PARA INFORMACIÓN DE REGISTRO DE PASSWORDS. 437
FIGURA B. 11 ESTRUCTURA DE ARCHIVO XML PARA INFORMACIÓN DE REGISTRO DE USUARIO. 437
FIGURA B. 12 ESTRUCTURA DE ARCHIVO XML PARA INFORMACIÓN DE REGISTRO DE VISTA. 438
FIGURA B. 13 ESTRUCTURA DE ARCHIVO XML PARA INFORMACIÓN DE SERVIDOR. 438
FIGURA B. 14 ESTRUCTURA DE ARCHIVO XML PARA INFORMACIÓN DE USUARIO. 439
FIGURA B. 15 ESTRUCTURA DE ARCHIVO XML PARA INFORMACIÓN DE VISTA. 439
FIGURA E. 1 EJEMPLO DE VENTANA DEL DIRECTORIO DONDE SE DEBE UBICAR EL ARCHIVO SWAR.WAR. 455
FIGURA E. 2 EJEMPLO DE VENTANA DEL DIRECTORIO DONDE SE DEBE UBICAR EL ARCHIVO *LOCALHOST.JKS*... 456
FIGURA E. 3 EJEMPLO DE VENTANA DEL DIRECTORIO DONDE ESTÁ UBICADO EL ARCHIVO *SERVER.XML*..... 456
FIGURA E. 4 EJEMPLO DE CONFIGURACIÓN DEL ARCHIVO *SERVER.XML*. 457
FIGURA E. 5 EJEMPLO DE VENTANA DEL DIRECTORIO DONDE ESTÁ UBICADO EL ARCHIVO *WEB.XML*. 458
FIGURA E. 6 EJEMPLO DE CONFIGURACIÓN DEL ARCHIVO *WEB.XML*. 458
FIGURA E. 7 MENSAJE DE ERROR DE CERTIFICADO. 459
FIGURA E. 8 PÁGINA DE BIENVENIDA DE LA APLICACIÓN. 460
FIGURA E. 9 PÁGINA DE INICIO DE SESIÓN. 460
FIGURA E. 10 PRIMER TIPO DE INTERFAZ GRÁFICA DE LA APLICACIÓN. 461
FIGURA E. 11 SEGUNDO TIPO DE INTERFAZ GRÁFICA DE LA APLICACIÓN. 461
FIGURA F. 1 INGRESO DE DATOS EN LA PÁGINA. 484
FIGURA F. 2 PRESENTACIÓN DEL MENSAJE DE BIENVENIDA..... 484
FIGURA G. 1 EJEMPLO DE LA INTERFAZ GRÁFICA DE OPENNMS..... 492

FIGURA G. 2 EJEMPLO DE LA INTERFAZ GRÁFICA DE SOLARWINDS ORION NETWORK PERFOMANCE MANAGER. 493
FIGURA G. 3 EJEMPLO DE LA INTERFAZ GRÁFICA DE IPSWITCH WHATSUP GOLD PREMIUM. 494

ÍNDICE DE TABLAS

TABLA 1. 1 GRUPOS DE OBJETOS PERTENECIENTES AL SUB-ÁRBOL MIB-2. 17
TABLA 1. 2 TIPOS DE DATOS ABSTRACTOS DEFINIDOS POR ASN.1 Y ACEPTADOS POR SMIV1. 20
TABLA 1. 3 MARCAS DE LOS TIPOS DE DATOS. 21
TABLA 1. 4 TIPOS DE DATOS ABSTRACTOS DEFINIDOS POR SMIV2. 21
TABLA 1. 5 RELACIÓN DE LOS NOMBRES DE MARCA CON LOS BITS 8 Y 7. 24
TABLA 1. 6 LISTADO DE ERRORES DEFINIDOS PARA SNMPV1 Y SNMPV2. 28
TABLA 1. 7 LISTADO DE TRAPS PREDEFINIDOS. 33
TABLA 1. 8 RFCS PARA SNMPV3. 35
TABLA 1. 9 REFERENCIAS DE ENTIDAD PREDEFINIDAS. 108
TABLA 1. 10 TIPOS DE ATRIBUTOS. 112
TABLA 1. 11 LISTA DE VALORES POR DEFECTO. 112
TABLA 2. 1 MODELO DE DEFINICIÓN DE CASOS DE USO. 148
TABLA 2. 2 CASO DE USO INICIAR SESIÓN. 148
TABLA 2. 3 CASO DE USO CONFIGURAR PARÁMETROS. 148
TABLA 2. 4 CASO DE USO DESCUBRIR ELEMENTOS DE RED. 149
TABLA 2. 5 CASO DE USO VISUALIZAR INFORMACIÓN DE ELEMENTO DE RED. 149
TABLA 2. 6 CASO DE USO NAVEGAR EN GRUPO MIB. 149
TABLA 2. 7 CASO DE USO EJECUTAR OPERACIÓN SNMPV3. 150
TABLA 2. 8 CASO DE USO VISUALIZAR NOTIFICACIONES. 150
TABLA 2. 9 CASO DE USO GESTIONAR MIBS. 150
TABLA 2. 10 CASO DE USO GESTIONAR USUARIOS. 150
TABLA 2. 11 CASO DE USO GESTIONAR VISTAS. 151
TABLA 2. 12 CASO DE USO GESTIONAR DERECHOS DE ACCESO. 151
TABLA 2. 13 CASO DE USO CAMBIAR PASSWORDS. 151
TABLA 2. 14 MODELO DE ESPECIFICACIÓN DE CASOS DE USO. 156
TABLA 2. 15 ESPECIFICACIÓN DE INICIAR SESIÓN. 157
TABLA 2. 16 ESPECIFICACIÓN DE CONFIGURAR PARÁMETROS. 157
TABLA 2. 17 ESPEFICICACIÓN DE DESCUBRIR ELEMENTOS DE RED. 158

TABLA 2. 18 ESPECIFICACIÓN DE VISUALIZAR INFORMACIÓN DE ELEMENTO DE RED. 159
TABLA 2. 19 ESPECIFICACIÓN DE NAVEGAR EN ÁRBOL MIB. 159
TABLA 2. 20 ESPECIFICACIÓN DE EJECUTAR OPERACIÓN SNMPV3. 160
TABLA 2. 21 ESPECIFICACIÓN DE VISUALIZAR NOTIFICACIONES. 161
TABLA 2. 22 ESPECIFICACIÓN DE GESTIONAR MIBS. 161
TABLA 2. 23 ESPECIFICACIÓN DE GESTIONAR USUARIOS. 163
TABLA 2. 24 ESPECIFICACIÓN DE GESTIONAR VISTAS. 164
TABLA 2. 25 ESPECIFICACIÓN DE GESTIONAR DERECHOS DE ACCESO. 166
TABLA 2. 26 ESPECIFICACIÓN DE CAMBIAR PASSWORDS. 167
TABLA 2. 27 MODELO DE REALIZACIÓN DE CASO DE USO-ANÁLISIS. 168
TABLA 2. 28 REALIZACIÓN DE INICIAR SESIÓN. 168
TABLA 2. 29 REALIZACIÓN DE CONFIGURAR PARÁMETROS. 169
TABLA 2. 30 REALIZACIÓN DE DESCUBRIR ELEMENTOS DE RED. 171
TABLA 2. 31 REALIZACIÓN DE VISUALIZAR INFORMACIÓN DE ELEMENTO DE RED. 173
TABLA 2. 32 REALIZACIÓN DE NAVEGAR EN UN GRUPO MIB. 175
TABLA 2. 33 REALIZACIÓN DE EJECUTAR OPERACIÓN SNMPV3. 176
TABLA 2. 34 REALIZACIÓN DE VISUALIZAR NOTIFICACIONES. 177
TABLA 2. 35 REALIZACIÓN DE GESTIONAR MIBS. 179
TABLA 2. 36 REALIZACIÓN DE GESTIONAR USUARIOS. 181
TABLA 2. 37 REALIZACIÓN DE GESTIONAR VISTAS. 183
TABLA 2. 38 REALIZACIÓN DE GESTIONAR DERECHOS DE ACCESO. 186
TABLA 2. 39 REALIZACIÓN DE CAMBIAR PASSWORDS. 188
TABLA 2. 40 PAQUETES DE ANÁLISIS DE LA APLICACIÓN. 192
TABLA 2. 41 PAQUETES DE SERVICIO DE LA APLICACIÓN. 196
TABLA 2. 42 MODELO DE REALIZACIÓN DE CASO DE USO-DISEÑO. 202
TABLA 2. 43 REALIZACIÓN DE INICIAR SESIÓN. 203
TABLA 2. 44 REALIZACIÓN DE CONFIGURAR PARÁMETROS. 206
TABLA 2. 45 DIAGRAMA DE COLABORACIÓN DEL CASO DE USO: CONFIGURAR PARÁMETROS. 207
TABLA 2. 46 REALIZACIÓN DE DESCUBRIR ELEMENTOS DE RED. 209
TABLA 2. 47 REALIZACIÓN DE VISUALIZAR INFORMACIÓN DE ELEMENTO DE RED. 213
TABLA 2. 48 REALIZACIÓN DE NAVEGAR EN UN GRUPO MIB. 217
TABLA 2. 49 REALIZACIÓN DE EJECUTAR OPERACIÓN SNMPV3. 221
TABLA 2. 50 REALIZACIÓN DE VISUALIZAR NOTIFICACIONES. 224
TABLA 2. 51 REALIZACIÓN DE GESTIONAR MIBS. 227
TABLA 2. 52 REALIZACIÓN DE GESTIONAR USUARIOS. 230

TABLA 2. 53 REALIZACIÓN DE GESTIONAR VISTAS. 234
TABLA 2. 54 REALIZACIÓN DE GESTIONAR DERECHOS DE ACCESO. 238
TABLA 2. 55 REALIZACIÓN DE CAMBIAR PASSWORDS. 242
TABLA 2. 56 MODELO DE CLASE DE DISEÑO. 246
TABLA 2. 57 EJEMPLO DE DEFINICIÓN DE UNA CLASE DE DISEÑO. 246
TABLA 2. 58 SUBSISTEMAS DE DISEÑO PARA LA APLICACIÓN. 247
TABLA 3. 1 TABLA DE DIRECCIONAMIENTO PARA TOPOLOGÍA DE RED. 280
TABLA A. 1 CLASES DE ANÁLISIS. 405
TABLA A. 2 CLASE DERECHOACCESO. 406
TABLA A. 3 CLASE GESTORDERECHOACCESO. 406
TABLA A. 4 CLASE ELEMENTO. 407
TABLA A. 5 CLASE GESTORELEMENTO. 407
TABLA A. 6 CLASE GRUPOVACM. 408
TABLA A. 7 CLASE GESTORGRUPOVACM. 408
TABLA A. 8 CLASE SERVIDOR. 409
TABLA A. 9 CLASE GESTORSERVIDOR. 409
TABLA A. 10 CLASE MENSAJE. 410
TABLA A. 11 CLASE VINCULOVARIABLE. 410
TABLA A. 12 CLASE GESTORMENSAJE. 411
TABLA A. 13 CLASE MIB. 411
TABLA A. 14 CLASE GESTORMIB. 412
TABLA A. 15 CLASE NODOMIB. 412
TABLA A. 16 CLASE NODOTIPOOBJETO. 412
TABLA A. 17 CLASE GESTORNODOMIB. 413
TABLA A. 18 CLASE NOTIFICACION. 413
TABLA A. 19 CLASE GESTORNOTIFICACION. 414
TABLA A. 20 CLASE RECEPTORNOTIFICACIONES. 414
TABLA A. 21 CLASE RED. 415
TABLA A. 22 CLASE GESTORRED. 415
TABLA A. 23 CLASE REGISTRODERECHO. 415
TABLA A. 24 CLASE GESTORREGISTRODERECHO. 416
TABLA A. 25 CLASE REGISTROPASSWORDS. 416
TABLA A. 26 CLASE GESTORREGISTROPASSWORDS. 417
TABLA A. 27 CLASE REGISTROUSUARIO. 417
TABLA A. 28 CLASE GESTORREGISTROUSUARIO. 418

TABLA A. 29 CLASE REGISTROVISTA. 418
TABLA A. 30 CLASE GESTORREGISTROVISTA. 419
TABLA A. 31 CLASE USUARIO. 419
TABLA A. 32 CLASE GESTORUSUARIO. 420
TABLA A. 33 CLASE VISTA. 420
TABLA A. 34 CLASE GESTORVISTA. 421
TABLA A. 35 CLASE BUSCADORELEMENTOS. 421
TABLA A. 36 CLASE PINGUTIL. 422
TABLA A. 37 CLASE EXPORTADORDERECHO. 422
TABLA A. 38 CLASE ANULADORDERECHO. 423
TABLA A. 39 CLASE EXPORTADORUSUARIO. 424
TABLA A. 40 CLASE ANULADORUSUARIO. 424
TABLA A. 41 CLASE EXPORTADORVISTA. 425
TABLA A. 42 CLASE ANULADORVISTA. 425
TABLA A. 43 CLASE EXPORTADORPASSWORDS. 426
TABLA A. 44 CLASE KEYCHANGEMANAGER. 427
TABLA A. 45 CLASE GESTORXML. 427
TABLA A. 46 CLASE SWARSERVLET. 427
TABLA A. 47 CLASE SNMPENGINE. 428
TABLA A. 48 CLASE INFOHOSTTREEFACTORY. 429
TABLA A. 49 CLASE NAVMIBTREEFACTORY. 429
TABLA D. 1 CONFIGURACIÓN DEL USUARIO INITIAL EN EL MODELO USM. 451
TABLA D. 2 CONFIGURACIÓN DEL CONTEXTO SNMP EN EL MODELO VACM. 451
TABLA D. 3 CONFIGURACIÓN DEL GRUPO INITIAL EN EL MODELO VACM. 451
TABLA D. 4 CONFIGURACIÓN DE LA VISTA INTERNET EN EL MODELO VACM. 451
TABLA D. 5 CONFIGURACIÓN DE UN DERECHO DE ACCESO EN EL MODELO VACM. 452
TABLA E. 1 PRINCIPALES SECCIONES DE LA INTERFAZ GRÁFICA DE LA APLICACIÓN WEB. 462
TABLA E. 2 PRINCIPALES PÁGINAS QUE CONFORMAN LA APLICACIÓN. 465
TABLA E. 3 DESCRIPCIÓN DE PARÁMETROS DE MENSAJES SNMPV3. 467
TABLA E. 4 DESCRIPCIÓN DE PARÁMETROS DE SERVIDOR. 469
TABLA E. 5 DESCRIPCIÓN DE PARÁMETROS PARA DESCUBRIMIENTO DE ELEMENTOS. 470
TABLA E. 6 DESCRIPCIÓN DE LOS TEMAS DE CONSULTA DE INFORMACIÓN ADICIONAL. 473
TABLA E. 7 DESCRIPCIÓN DE LOS PARÁMETROS DE EJECUCIÓN DE MENSAJES SNMPV3. 475
TABLA E. 8 DESCRIPCIÓN DE LOS PARÁMETROS DE UNA CUENTA DE USUARIO. 477
TABLA E. 9 DESCRIPCIÓN DE LOS PARÁMETROS DE UNA VISTA MIB. 479

TABLA E. 10 DESCRIPCIÓN DE LOS PARÁMETROS DE UN DERECHO DE ACCESO. 480
TABLA E. 11 DESCRIPCIÓN DE LOS PARÁMETROS DE CAMBIO DE PASSWORDS DE USUARIO. 481
TABLA G. 1 TABLA DE COMPARACIÓN DE HERRAMIENTAS NMS. 496

ÍNDICE DE CÓDIGOS FUENTE

CÓDIGO FUENTE C. 1 PÁGINA CABECERA.JSP. 441
CÓDIGO FUENTE C. 2 PÁGINA PIEPAG.JSP. 442
CÓDIGO FUENTE C. 3 PÁGINA MENUIZQ.JSP. 442
CÓDIGO FUENTE C. 4 EJEMPLO DE INCLUSIÓN DE UNA PÁGINA JSP. 443
CÓDIGO FUENTE C. 5 MÉTODO DE SWARSERVLET PARA PROCESAMIENTO DE SOLICITUDES Y RETORNO DE RESPUESTAS. 443
CÓDIGO FUENTE C. 6 EJEMPLO DE UNA HOJA DE ESTILOS EN CASCADA PARA LA APLICACIÓN. 444
CÓDIGO FUENTE C. 7 DEFINICIÓN DE UNA CLASE DE ENTIDAD. 445
CÓDIGO FUENTE C. 8 MÉTODOS DE UNA CLASE DE CONTROL. 446
CÓDIGO FUENTE C. 9 VALIDACIÓN Y OBTENCIÓN DE DOCUMENTO XML. 446
CÓDIGO FUENTE C. 10 MÉTODO DE EJECUCIÓN DE UNA OPERACIÓN SNMPV3 POR PARTE DE MENSAJE. 447
CÓDIGO FUENTE C. 11 MÉTODO DE ENVÍO DE UN MENSAJE SNMPV3 POR PARTE DEL SNMPENGINE. 447
CÓDIGO FUENTE C. 12 ARCHIVO DE CONFIGURACIÓN WEB.XML 448
CÓDIGO FUENTE C. 13 ARCHIVO XSD PARA REDES. 448
CÓDIGO FUENTE C. 14 ARCHIVO XML PARA ALMACENAMIENTO DE REDES. 449
CÓDIGO FUENTE F. 1 CLASE CLIENTE 485
CÓDIGO FUENTE F. 2 CLASE EJEMPLOSERVLET. 486
CÓDIGO FUENTE F. 3 PÁGINA ADMINCLIENTE.JSP. 486
CÓDIGO FUENTE F. 4 ARCHIVO WEB.XML DE APLICACIÓN *EJEMPLO*. 489
CÓDIGO FUENTE F. 5 ARCHIVO DE CONFIGURACIÓN *SERVER.XML* DEL SERVIDOR TOMCAT. 490

RESUMEN

El presente trabajo trata acerca del diseño e implementación de un software de monitorización y administración de red que tiene: interfaz web, seguridades, capacidad de instalación en diferentes sistemas operativos, y que tiene al protocolo SNMPv3 como su principal componente. El trabajo está divido en cuatro capítulos, cada uno comprende un tema específico, éstos son: sustentación teórica, desarrollo del sistema, pruebas y resultados obtenidos, y finalmente conclusiones y recomendaciones.

El primer capítulo trata acerca de la sustentación teórica, sobre la cual se fundamenta el desarrollo de la aplicación web. En forma general, los temas comprendidos dentro de este capítulo son: conceptos básicos sobre redes de computadores, administración y gestión de redes, la arquitectura de un sistema de administración de red, fundamentos del protocolo SNMP, análisis del protocolo SNMPv3, teoría de la tecnología XML y sus derivados, y la relación de XML con la gestión de redes. Además, se hace un estudio del lenguaje de programación Java, enfocándose en las tecnologías y herramientas destinadas al desarrollo web.

El segundo capítulo trata acerca del desarrollo de la aplicación web. Para el efecto se siguen los procedimientos que define UML; es decir, se empieza con el análisis de requerimientos, luego se hace el diseño, y se finaliza con la implementación en sí.

El tercer capítulo trata acerca de las pruebas y los resultados obtenidos. En primer lugar se hace una breve descripción de las características de la aplicación. Luego, se realizan las pruebas de funcionamiento sobre una topología de red determinada. Y posteriormente se hacen los análsis a partir de la observación de dichos resultados.

En el último capítulo se presentan las conclusiones y las recomendaciones que se concibieron al desarrollar el software de administración de red.

PRESENTACIÓN

En una red informática, la comunicación que se produce entre un dispositivo y otro a simple vista parece sencilla pero no la es; por general los mensajes atraviesan por varias etapas como cables de cobre, cables de fibra óptica, enlaces microonda,etc.;y debido a esta travesía se generan varios problemas tales como: ruido, pérdida de conexión, pérdida de paquetes, etc. Estos problemas deben ser enfrentados con el firme propósito de garantizar que la información llegue a su destino de manera íntegra. Todo esto obliga a realizar actividades de administración de los diferentes elementos que componen la red.

Debido a que las infraestructuras de las redes cada vez se van haciendo más complejas, la administración de las mismas también. Por tal motivo, se crean nuevas y mejores herramientas que ayuden al administrador a ejecutar las funciones de gestión de redes, procurando que éstas no representen elevados costos de adquisición y mantenimiento. Ahora bien, existen en el mercado muchas soluciones de software que permiten realizar administración de redes; sin embargo, no todas brindan una solución efectiva y conveniente para el usuario. Por ejemplo, existen aplicaciones que no ofrecen las seguridades necesarias para proteger la información, hay otras que para operar requieren que previamente el usuario realice configuraciones complicadas, hay otras que se diseñan para ser instaladas en sistemas operativos específicos, u otras que son completas y fáciles de operar pero su precio es demasiado elevado.

Por todos los motivos citados anteriormente, agregando además el desarrollo progresivo que ha tenido el Internet, se tratan de diseñar soluciones software, que sean del tipo web ya que son las más usadas hoy en día, y que logren resolver los inconvenientes citados anteriormente, procurando siempre de mantener los costos a un nivel adecuado de tal manera que sean accesibles para el usuario. Por lo tanto, el presente proyecto presenta una solución de software que logra cubrir con la mayor parte de requerimientos anteriormente descritos.

CAPÍTULO 1. SUSTENTACIÓN TEÓRICA

Este capítulo contiene temas teóricos sobre los cuales se fundamenta el proyecto. Empieza con información relacionada a conceptos básicos de Redes de Computadores, luego una introducción a la Administración de Redes, consecuentemente se analiza fundamentos importantes del protocolo SNMP que son necesarios para posteriormente abordar el tema objetivo del proyecto, SNMPv3. Finalmente se estudia aspectos esenciales de XML y tecnologías Java para el desarrollo de páginas web.

Comenzando por conceptos básicos de Redes de Computadores, se trata acerca su definición y sus fines, tipos de redes de computadores, tecnologías de hardware de conectividad para formar redes, y una breve reseña de los componentes básicos (hardware y software) de una red informática corporativa.

Luego en la introducción a la Administración de Redes, se muestra su definición, las funciones de administración de red definidas por el modelo de gestión de Internet, y un resumen de la arquitectura de un sistema de administración de red.

En los fundamentos del protocolo SNMP se muestra su definición, sus componentes, su funcionamiento y su relación con los protocolos TCP/IP. Además, se trata acerca de la estructura de información de administración y las operaciones SNMP.

Respecto a SNMPv3 se estudian sus características, la arquitectura que define, los modelos de: procesamiento de mensajes, seguridad y control de acceso a la información; y referencias rápidas de los grupos MIB asociados a este protocolo.

Finalmente se aborda tópicos importantes de XML, tales como: características, tipos de documentos, lenguajes derivados, herramientas de desarrollo y su uso en la administración de redes. Además se tratan tecnologías que Java propone para el desarrollo de páginas web y herramientas de software relacionadas.

1.1 CONCEPTOS BÁSICOS DE REDES DE COMPUTADORAS

Una red informática es una agrupación de equipos, computadoras y otros dispositivos, que se comunican entre sí a través de medios de transmisión (cables de cobre, fibra óptica, ondas electromagnéticas, etc.) para intercambiar información (archivos), recursos (hardware) y servicios (correo electrónico, web, acceso a internet). [1]

En una red de computadoras, la información es convertida en señales[1] para luego ser transmitida a través de los medios de comunicación. Existen varias tecnologías para lograr la transmisión de datos de un equipo a otro dentro de la misma red: Ethernet, Token Ring y FDDI. Estas tecnologías permiten construir la unidad básica de redes de computadoras, Red LAN (*Local Area Network*). Una red LAN tiene un área de cobertura limitada tal como: una oficina, un campus, etc. Actualmente existen tecnologías LAN modernas las cuales probablemente sean las más utilizadas: Ethernet conmutada e IEEE802.11. En el caso de transferencia de información desde una red local a otra, existen las redes WAN. Las redes WAN se caracterizan por tener: menores tasas de transmisión, grandes coberturas geográficas y utilización de servicios de arrendamiento de enlaces de transmisión.

Para construir redes LAN y WAN se usan tecnologías de hardware de conectividad. Ejemplos de estas tecnologías son: hubs, switches, bridges y routers. Un hub o concentrador es un dispositivo que retransmite la señal de información que recibe por un puerto hacia los puertos restantes.

Un ruteador es un equipo que identifica la dirección de red destino de un paquete y en base a ello encuentra el mejor camino para reenviarlo a dicha dirección. Un router también puede realizar filtración de tráfico.

[1] Las señales pueden ser eléctricas u ópticas.

Un bridge es un dispositivo que permite conectar dos redes LAN distintas. Los puentes o bridges son menos costosos que los ruteadores debido a que la toma de decisión de enviar un paquete hacia su destino es sencilla, y también a que usan poca información de las cabeceras de los paquetes para realizar esta función.

Por otra parte, los switches son equipos que pueden administrar la conmutación de información entre computadores de una misma red así como de múltiples redes LAN. Un switch tiene las mismas funciones de filtraje y envío pero con la diferencia de que no usa direcciones de red (IP) sino direcciones físicas (MAC).

Por lo general, una red corporativa debe estar compuesta por ciertos elementos (hardware y software) como mínimo para ser realmente funcional. Entre los de hardware están: servidores, dispositivos de almacenamiento de información, hubs, estaciones de trabajo, tarjetas adaptadoras de red, cableado estructurado, switches, ruteadores, hardware a ser compartido (impresoras, escáneres, etc.). Entre los elementos software se tiene: programas de servidores, programas de clientes, sistemas operativos de red, aplicaciones de desarrollo (lenguajes de programación), herramientas de administración de red, software de seguridad. [2]

1.2 INTRODUCCIÓN A LA ADMINISTRACIÓN DE REDES [2]

En una red informática, la comunicación entre un dispositivo y otro no es tan simple como parece. Cuando se transmiten paquetes desde un equipo que se encuentra en una red hacia otro que se encuentra fuera de ella, por lo general atraviesan por varias etapas: cables de cobre, cables de fibra óptica, enlaces microonda, redes inalámbricas, ruteadores, conmutadores, etc. Debido a toda esta travesía, se generan varios problemas que se deben enfrentar: ruido, pérdida de conexión, pérdida de paquetes, colisiones de paquetes, incremento de tráfico, cuellos de botella, caída de nodos, etc. Por lo tanto, una solución a todo esto es realizar actividades de administración de red.

Para gestionar una red previamente se debe identificar a los nodos para luego realizar procesos de monitorización. Si la red es grande y compleja sería ilógico que el administrador hiciera estas tareas manualmente; por lo tanto, es razonable utilizar un sistema que se encargue de realizar todo esto de forma correcta y automática. Este sistema de denomina **Sistema de Administración de Red**, que inglés comúnmente se le conoce como *NMS (Network Management System – Sistema de Administración de Red).*

1.2.1 DEFINICIÓN DE ADMINISTRACIÓN DE RED [3]

Según T.Saydam*, la administración de red es aquello que involucra el desarrollo, integración y coordinación de hardware, software y elemento humano para monitorizar, probar, sondear, configurar, analizar, evaluar y controlar los recursos de la red para alcanzar requerimientos de tiempo real, desempeño operacional y calidad de servicio a un precio razonable*.[2]

En otras palabras, la administración de red es un concepto general que hace uso de varias herramientas, técnicas, y aplicaciones para ayudar al administrador en la gestión de los dispositivos, aplicaciones y redes. El objeto de todo esto es lograr un desempeño sostenible de la red a costos convenientes.

Existen varios modelos de gestión que definen funciones, protocolos y estructuras para el manejo de la información, de entre ellos el más utilizado ha sido el **modelo de gestión de Internet**.

El modelo de gestión de Internet se caracteriza porque la administración está basada en estructuras de información y protocolos sencillos, lo cual permite agilitar y facilitar la implementación de aplicaciones de administración de red.

[2] SAYDAM, Tuncay, http://202.113.75.224/newenglish/netman_intro.html, Acceso: 7 de diciembre 2009, 18:27.

1.2.2 FUNCIONES DE ADMINISTRACIÓN DE RED

1.2.2.1 Funciones de administración definidas por el modelo de gestión de Internet (TCP/IP) [4][5]

1.2.2.1.1 Monitorización de Red

Es el proceso de observar el comportamiento de la red y de sus nodos a través de la correspondiente información de administración. Esto se realiza con el fin de detectar fallas en la red y en los nodos. La monitorización involucra dos factores importantes: el tiempo de duración del monitoreo y el uso de los recursos de red. Mientras mayor sea el tiempo de monitoreo más efectiva será la detección de problemas, pero también habrá una mayor ocupación de los recursos lo cual perjudicará a otras tareas. Por lo tanto, deberá existir un balance para conseguir un desempeño aceptable del sistema.

1.2.2.1.2 Control de Red

Es el proceso de mejorar el comportamiento de la red cambiando o configurando adecuadamente los valores de ciertos parámetros y realizando acciones pertinentes en los elementos de red para solucionar fallas detectadas.

1.2.2.1.3 Seguridad de Red

La Seguridad de Red es la nueva función definida por el modelo Internet. Esta función comprende: proteger la información de administración y controlar el acceso a recursos de la red.

1.2.3 SISTEMAS DE ADMINISTRACIÓN DE RED [6]

Un sistema de administración de red, es decir un NMS, es una colección de herramientas que permiten realizar las funciones de monitorización, control y seguridad de la red.

1.2.3.1 Arquitectura de un sistema de administración de red

Un sistema de administración de red tiene como elementos: el nodo(s) administrador(es) y el/los nodo(s) administrable(s). Cada nodo, a más del sistema operativo, el software de comunicaciones y los servicios de nivel de aplicación, está compuesto por los siguientes módulos:

1.2.3.1.1 NME (Network Management Entity – Entidad de Administración de Red)

Este módulo está presente en el nodo administrable y administrador. Se lo conoce generalmente como **Agente** y realiza las siguientes tareas de administración de red:

- Recolección de estadísticas sobre actividades ocurridas en la red.
- Notificación de eventualidades (conocidas como ***traps***) al nodo administrador.
- Contestación a los comandos transmitidos por el nodo administrador. Estos comandos pueden ser: transmisión de los datos recolectados, modificación del valor de algún atributo, obtención de información de estado del nodo administrable, generación de tráfico para realizar pruebas de conexión, etc.

1.2.3.1.2 NMA (Network Management Application – Aplicación de Administración de Red)

Este módulo está presente únicamente en el nodo administrador. Es un software que ejerce las funciones de monitorización y control de red. Típicamente conocido como **Gestor de red**. Un NMA efectúa las siguientes actividades:

- Provisión de mecanismos de autenticación de usuarios para controlar el acceso a la administración de la red.
- Contestación a los comandos generados por los usuarios por medio del despliegue de resultados en pantalla.
- Transmisión de comandos a los agentes.

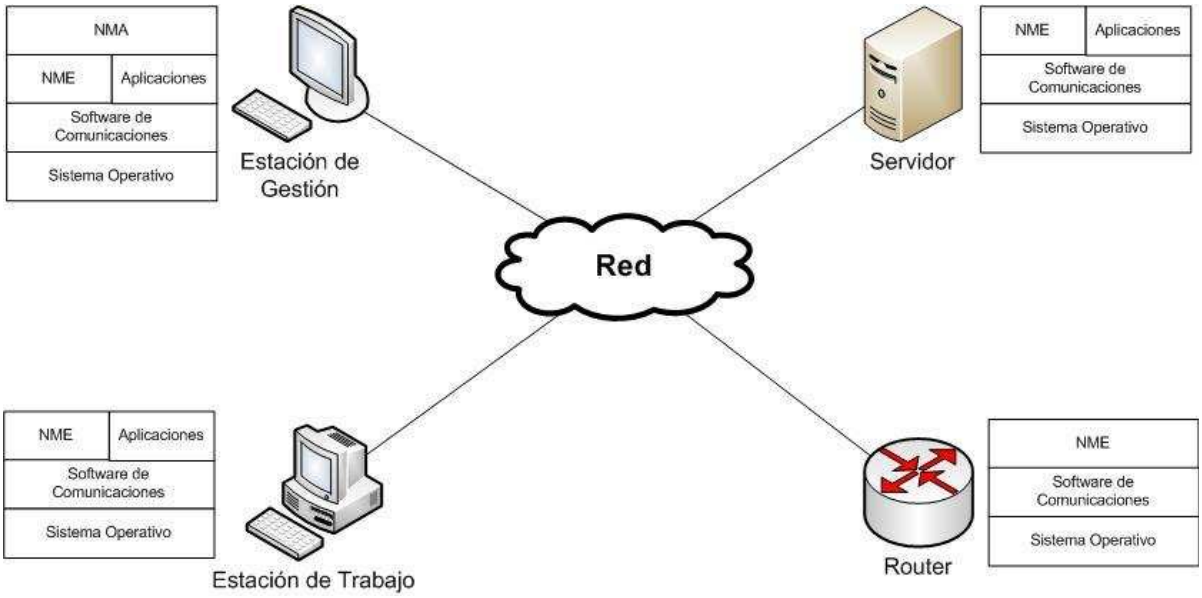


Figura 1. 1 Ejemplo de arquitectura de un sistema de gestión de red.[3]

1.3 FUNDAMENTOS DEL PROTOCOLO SNMP [4][7]

SNMP (*Simple Network Management Protocol – Protocolo Simple de Administración de Red*) es el protocolo de administración de red propuesto por el modelo de gestión Internet. SNMP pertenece al conjunto de protocolos TCP/IP. SNMP es un protocolo de capa aplicación que hace posible el intercambio de información de gestión entre dispositivos de red.

Existen varias versiones del protocolo: SNMPv1, SNMPv2, y SNMPv3. Las dos primeras siguen un mismo esquema (con mejores funcionalidades en la segunda versión) y dejan al descubierto grandes deficiencias como por ejemplo: poca o ninguna seguridad, inconvenientes en la transferencia de grandes volúmenes de información, y débiles mecanismos de autenticación y privacidad. SNMPv3 corrige estas deficiencias y además tiene soporte para todas las funcionalidades de las versiones anteriores.

[3] STALLINGS William; SNMP, SNMPv2, SNMPv3 and RMON 1 and 2, Tercera Edición, Figura 1-1 tomada de la pág.8. Acceso: 8 de diciembre 2009, 12:00.

Gracias al desarrollo de SNMPv3, las funciones de administración del modelo de gestión Internet se han extendido de tal modo que se ha agregado la funcionalidad de Seguridad de Red.

1.3.1 COMPONENTES DE SNMP [7]

1.3.1.1 Nodo administrable

Es cualquier dispositivo de red que tenga instalado el software agente (NME).

1.3.1.2 Nodo administrador

Representa a la estación de administración que tiene instalado el agente (NME) y la aplicación de administración (NMA).

1.3.1.3 Protocolo de administración

Es el protocolo que hace posible la comunicación de mensajes de gestión entre el nodo administrador y el nodo administrable. Sus funciones son:

- **Lectura**: Se refiere a la monitorización de la red.
- **Escritura**: Se refiere al control de la red.
- **Notificación**: Cuando un agente informa a la estación de administración sobre eventualidades sucedidas en la red.

1.3.1.4 Información de administración

Son datos de gestión (configuraciones locales, tráfico manejado, etc.) usados para las operaciones de monitorización y control de la red. Esta información se almacena en la base de información de administración (MIB, *Management Information Base* –

Base de Información de Administración). El agente es quién permite al nodo administrador acceder a la MIB.

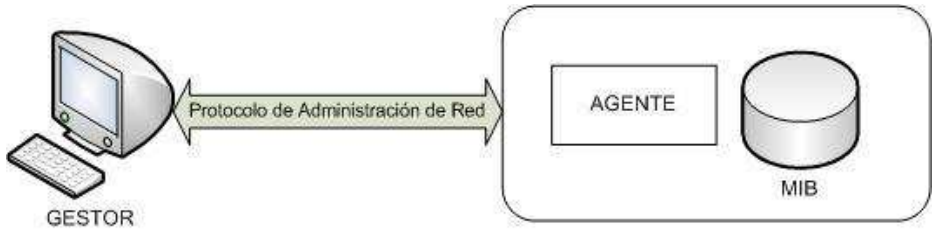


Figura 1. 2 Componentes de SNMP.

1.3.2 FUNCIONAMIENTO DE SNMP [4]

El funcionamiento del protocolo SNMP sigue un modelo de comunicación asíncrono de petición-respuesta. Esto significa que, el proceso de intercambio de mensajes puede ser iniciado por la estación de administración o por el agente (cuando no existe una petición previa por parte de la estación de administración).

En el caso de que la comunicación es iniciada por el nodo administrador, se secuencian los siguientes pasos:

1) La estación de administración envía un pedido de información al agente.
2) El agente realiza una autenticación del pedido y luego responde con la información solicitada.

En el caso de que la comunicación es iniciada por el nodo administrable, se secuencian los siguientes pasos:

1) El agente envía información de algún suceso imprevisto al nodo administrador sin que éste lo haya solicitado anticipadamente.
2) El nodo administrador hará las acciones pertinentes para resolver el problema y esto podría significar realizar reconfiguraciones a los nodos lo cual implicaría repetir el escenario anterior.

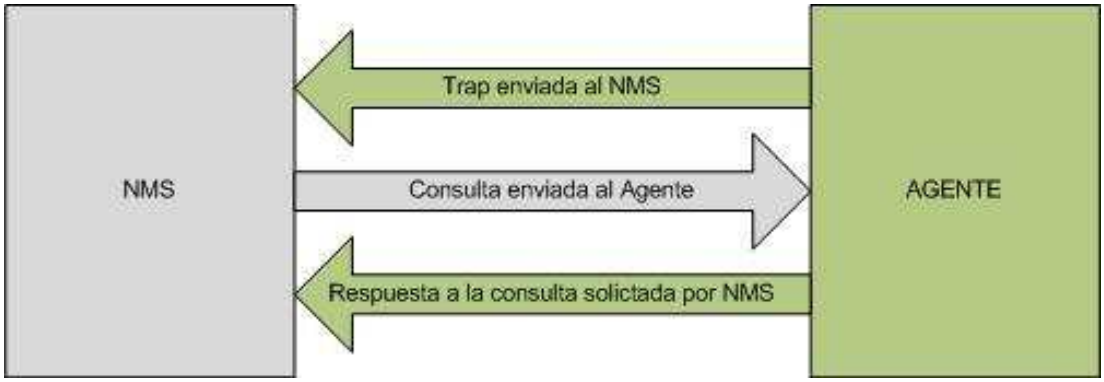


Figura 1. 3 Funcionamiento de SNMP.[4]

1.3.3 RELACIÓN DE SNMP CON LOS PROTOCOLOS TCP/IP [4]

SNMP utiliza el protocolo UDP (*User Datagram Protocol – Protocolo de Datagramas de Usuario*) para el transporte de datos entre administradores y agentes. La razón de la elección de este protocolo en lugar de TCP es debido a las características que posee: no orientado a conexión y no confiable; gracias a esto, SNMP puede ejercer sus operaciones en tiempo real y con mínimo impacto en el rendimiento de la red. En el caso de que se produzcan pérdida de paquetes por cualquier motivo, la corrección de errores estará a cargo de la aplicación de administración de red. Para detectar si existe pérdida de paquetes la aplicación puede basarse en el método de tiempo de espera (*timeout*). El NMS manda una solicitud UDP hacia el agente y espera por una respuesta, si el *timeout* ha superado el valor umbral, entonces el NMS asume que el paquete fue perdido y procede a realizar retransmisión de paquetes si así lo desea.
SNMP emplea el puerto UDP 161 para el envío de mensajes de petición y recepción de respuestas, y el puerto UDP 162 para recepción de *traps*.

En la figura 1.4 se muestra la pila de protocolos TCP/IP sobre la cual se realiza la comunicación SNMP. En cualquier caso si se da una notificación o un comando, suceden una serie de eventos en el stack de protocolos TCP/IP:

[4] DOUGLAS M., SCHMIDT K.; Essential SNMP, Segunda edición, Figura 1-1, Acceso: 10 de diciembre 2009, 18:00.

- **Aplicación:** En primer lugar, la aplicación SNMP (gestor o agente) decide la actividad a realizar; por ejemplo: enviar solicitudes, responder a las solicitudes mediante mensajes, o enviar notificaciones de algún suceso imprevisto. La capa aplicación se encarga de proveer al usuario final los servicios necesarios para llevar a cabo las actividades SNMP.

- **UDP:** El siguiente nivel, UDP, permite la comunicación host a host. La cabecera del datagrama UDP contiene información del puerto destino perteneciente al host al cual se le está enviando: una solicitud (puerto 161), una respuesta (puerto 161) o una notificación (puerto 162).

- **IP:** El protocolo IP (*Internet Protocol – Protocolo de Internet*) es el responsable de entregar el paquete SNMP hacia al host destino identificado por una dirección IP.

- **MAC:** En el último nivel, MAC (*Media Access Control – Control de Acceso al Medio*), los paquetes SNMP son puestos en la red física para que sean enrutados y puedan llegar a su destino final. La capa MAC es responsable de recibir los paquetes desde la red física y enviarlos al stack de protocolos para que puedan ser procesados por la capa aplicación.

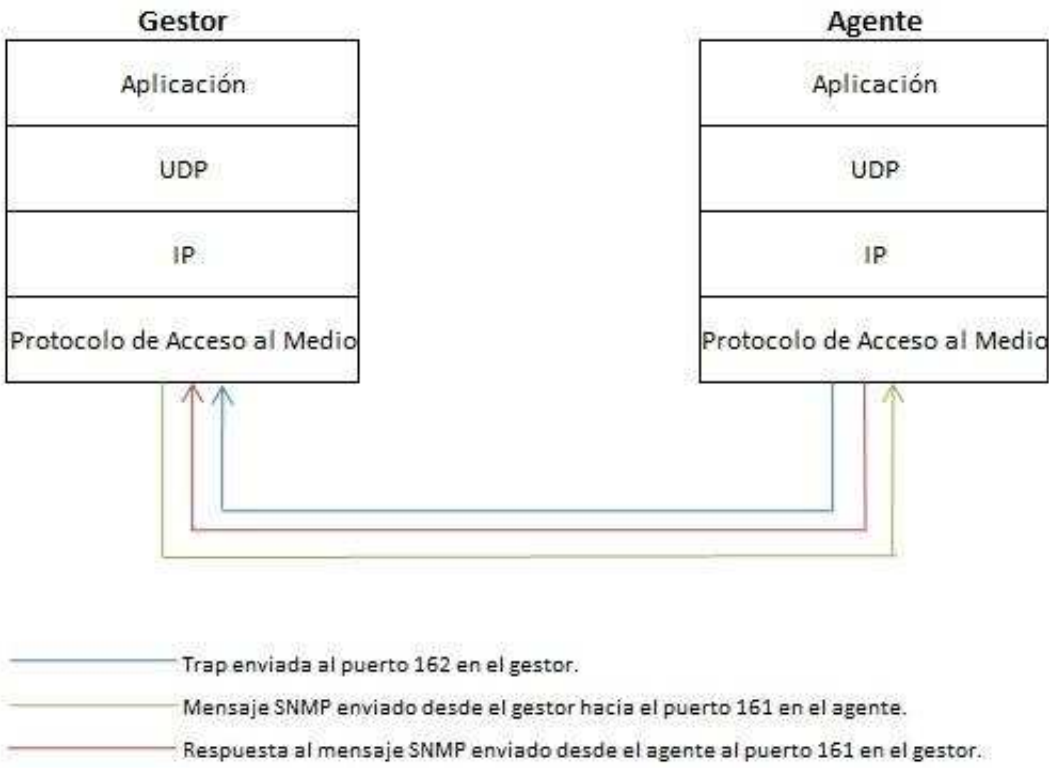


Figura 1. 4 Modelo de comunicación TCP/IP y SNMP.[5]

1.3.4 MECANISMOS DE SEGURIDAD BÁSICA [4][10]

En las versiones de SNMPv1 y SNMPv2 se manejaban mecanismos de seguridad no eficaces basados en el concepto de **Comunidades**, las cuales fueron implementadas para controlar el acceso a la información de administración de cada dispositivo por parte de una estación de administración de red.

Los **Nombres** de **Comunidad** o **Comunidades** son claves de acceso o *passwords*. Una comunidad a la vez constituye un método que permite establecer una relación de confianza entre un agente y una estación gestora de red. Una comunidad basa su funcionamiento en dos mecanismos: la autenticación y las políticas de acceso.

[5] DOUGLAS M., SCHMIDT K.; Essential SNMP, Segunda edición, Figura 2-1, Acceso: 11 de diciembre 2009, 14:00.

La **Autenticación** se refiere a la confirmación de la identidad de un emisor utilizando el nombre de la comunidad. Este mecanismo es vulnerable debido a que la clave es transportada en texto plano y sin encriptación.

Las **Políticas de Acceso** son restricciones de las cuales se ayuda el agente para proteger la información de administración de cada dispositivo. Existen varias categorías de acceso que pueden ser seleccionadas en base a nombres de comunidades. Las categorías de acceso pueden ser: **sólo lectura** (por defecto el nombre de comunidad es: *Public*), **escritura** (por defecto el nombre de comunidad es: *Private*) y notificación (*traps*). En las políticas de acceso también existe el concepto de **Vistas**, las cuales son colecciones de información tomadas de la base de datos (MIB) y que pueden ser accedidas por los gestores de red**.** Es posible definir varias vistas por cada nombre de comunidad.

1.3.5 INFORMACIÓN DE ADMINISTRACIÓN SNMP [4][8]

La MIB es una base de datos que contiene información de administración de los dispositivos gestionados. Cada elemento o recurso a administrar se representa como un objeto. La MIB es una agrupación estructurada de dichos objetos. Cada dispositivo de red mantiene una MIB que refleja el estado de los recursos gestionados en él. Una NMS puede monitorear y controlar los recursos de cada dispositivo, leyendo y modificando los valores de los objetos en la MIB. Para ello, una MIB debe cumplir con los siguientes objetivos que satisfacen los requerimientos de un sistema de administración de red:

- Los objetos utilizados para representar un recurso particular deben ser los mismos en todos los dispositivos de red.

- Es necesario usar un esquema común de la representación de la información para conseguir interoperabilidad.

El primer objetivo se alcanza mediante la definición de los objetos y su estructuración en la MIB. El segundo objetivo se alcanza mediante la definición de una estructura de la información de administración (SMI, *Structure of Management Information – Estructura de Información de Administración)*.

1.3.5.1 Estructura de Información de Administración (SMI)

La SMI define el marco general para la definición y creación de una MIB. La SMI también define los tipos de datos que pueden usarse en la MIB y especifica cómo se deben representar y nombrar los objetos dentro de la MIB. La versión primera de SMI (SMIv1, RFC1155) trata acerca de lo anterior, pero también existe una segunda versión (SMIv2, RFC 2578) que ofrece mejoras para SNMPv2.

El objetivo general de SMI es mantener la simplicidad y extensibilidad de la MIB, por lo tanto, la MIB solo puede guardar dos tipos de datos simples: **escalares** y **arreglos bidimensionales de escalares** (tablas).

Concretamente, la SMI proporciona métodos para:

- *Definir la estructura de una MIB particular*.
- *Definir objetos individuales, incluyendo la sintaxis y el valor de cada objeto*.
- *Codificar los valores de los objetos*.[6]

1.3.5.1.1 Estructura de la MIB

Todos los objetos gestionados están ordenados en una estructura de árbol jerárquico. Las hojas del árbol son los objetos gestionados. A cada objeto se le asocia una identificación única, el OID (*Object IDentifier – Objeto Identificador*). Un OID está formado por una serie de números enteros (llamados sub-identificadores y están basados en los nodos del árbol) separados por puntos (.). También existe la

[6] LESCHENNE, Sebastián; SALAZAR, Martín, Universidad Nacional de Rosario, Modelo de Gestión Internet SNMP-RMON, pág.3. Acceso: 11 de diciembre 2009, 18:00

alternativa de no usar números enteros sino los nombres de los nodos; esto facilita al usuario a ubicar un objeto dentro del árbol.

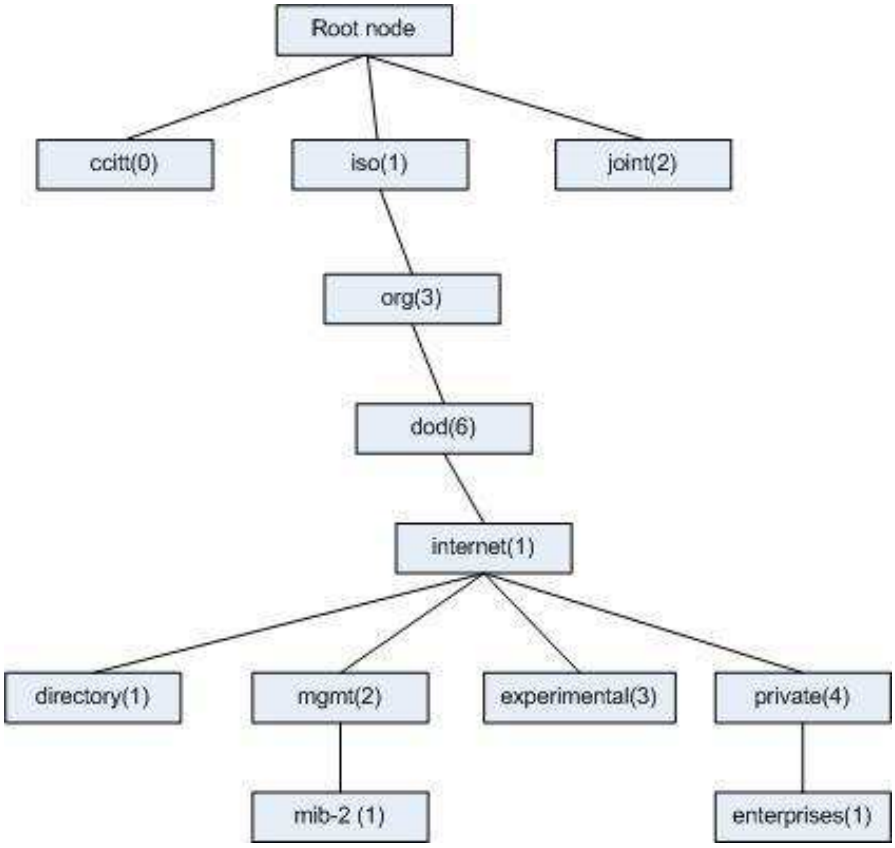


Figura 1. 5 Una parte del árbol jerárquico de objetos organizados según SMIv1.[7]

En la figura 1.5 se muestra el árbol de objetos organizados según SMIv1. El nodo de primer nivel se denomina *root* (*Root node – Nodo raíz*), los nodos que se derivan en otros (por ejemplo: *iso*) son sub-árboles y los nodos que no se derivan en otros (por ejemplo: *ccitt* y *joint*) son nodos hojas.

El sub-árbol *iso(1).org(3).dod(6).internet(1)* es el de mayor importancia debido a su relación con el modelo de gestión de Internet. A partir de este árbol existen varias ramificaciones: *directory* (reservado para uso futuro manejado por *OSI* directory-*X.500*), *management* o *mgmt* (define un conjunto estándar de objetos de la gestión

[7] DOUGLAS M., SCHMIDT K.; Essential SNMP, Segunda edición, Figura 2-2, Acceso: 12 de diciembre 2009, 10:00

del modelo de Internet), *experimental* (reservado para fines de prueba e investigación) y *private* (para objetos propietarios definidos por fabricantes).

El sub-árbol *mgmt* contiene las definiciones de los grupos MIB. Existen dos versiones de la MIB: *mib-1* y *mib-2*. La segunda es extensión de la primera. El grupo *mib-2* es el encargado de almacenar información fundamental para la configuración y solución de fallas. La figura 1.6 muestra el despliegue del sub-árbol *mgmt.*

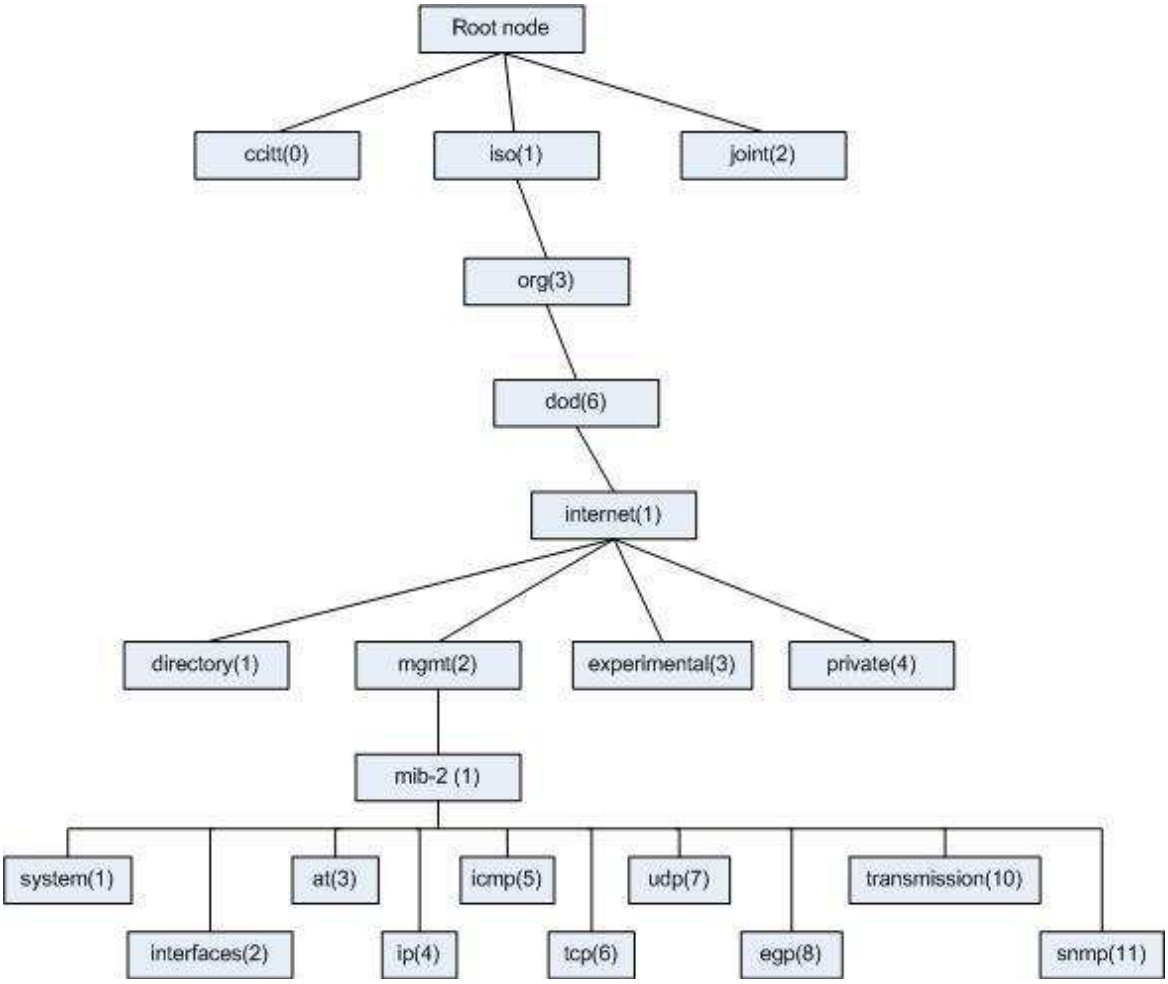


Figura 1. 6 Despliegue del sub-árbol mib-2.[8]

[8] VELÁSQUEZ, Andrés; Diseño e implementación de un módulo software para la monitorización de Elementos de una red informática utilizando el protocolo SNMP y el lenguaje XML, Figura 1.7, Acceso: 12 de diciembre 2009, 14:00.

Grupo	OID	Descripción
system	.1.3.6.1.2.1.1	Define una lista de objetos que se relacionan con la operación del sistema. Por ejemplo: tiempo de disponibilidad, nombre, contacto del sistema, etc.
interfaces	.1.3.6.1.2.1.2	Guarda el estado de cada interfaz de red en una entidad gestionada. Por ejemplo: que interfaz está activada y cuál desactivada, número de paquetes recibidos, con errores o descartados, etc.
at	.1.3.6.1.2.1.3	El grupo *address translation* trata acerca de todo lo relacionado a la traducción de direcciones de red a físicas.
ip	.1.3.6.1.2.1.4	Mantiene información de varios aspectos de IP, incluye enrutamiento IP.
icmp	.1.3.6.1.2.1.5	Sigue la pista de mensajes ICMP, por ejemplo paquetes ICMP con errores, descartados, etc.
tcp	.1.3.6.1.2.1.6	Mantiene información del estado de una conexión TCP.
udp	.1.3.6.1.2.1.7	Almacena información relacionada al protocolo UDP, por ejemplo: estadísticas de datagramas enviados o recibidos, entre otros.
egp	.1.3.6.1.2.1.8	Guarda estadísticas del protocolo EGP (Exterior Gateway Protocol) y las tablas EGP de nodos vecinos.
transmission	.1.3.6.1.2.1.10	Guarda información sobre los medios de transmisión.
snmp	.1.3.6.1.2.1.11	Mide el rendimiento de una implementación SNMP en un nodo administrado, supervisa el número de paquetes SNMP que han sido enviados y recibidos, etc.

Tabla 1. 1 Grupos de objetos pertenecientes al sub-árbol mib-2.

En la tabla 1.1 se describen de modo conceptual los grupos pertenecientes al sub-árbol *mib-2*.

Por otra parte, con el aparecimiento de SMIv2, se hizo una extensión al árbol MIB añadiendo la rama *snmpv2* bajo el subárbol *internet*. En la figura 1.7 se muestran los nodos añadidos. En el grupo *snmpModules* residen las MIBs para SNMPv3.

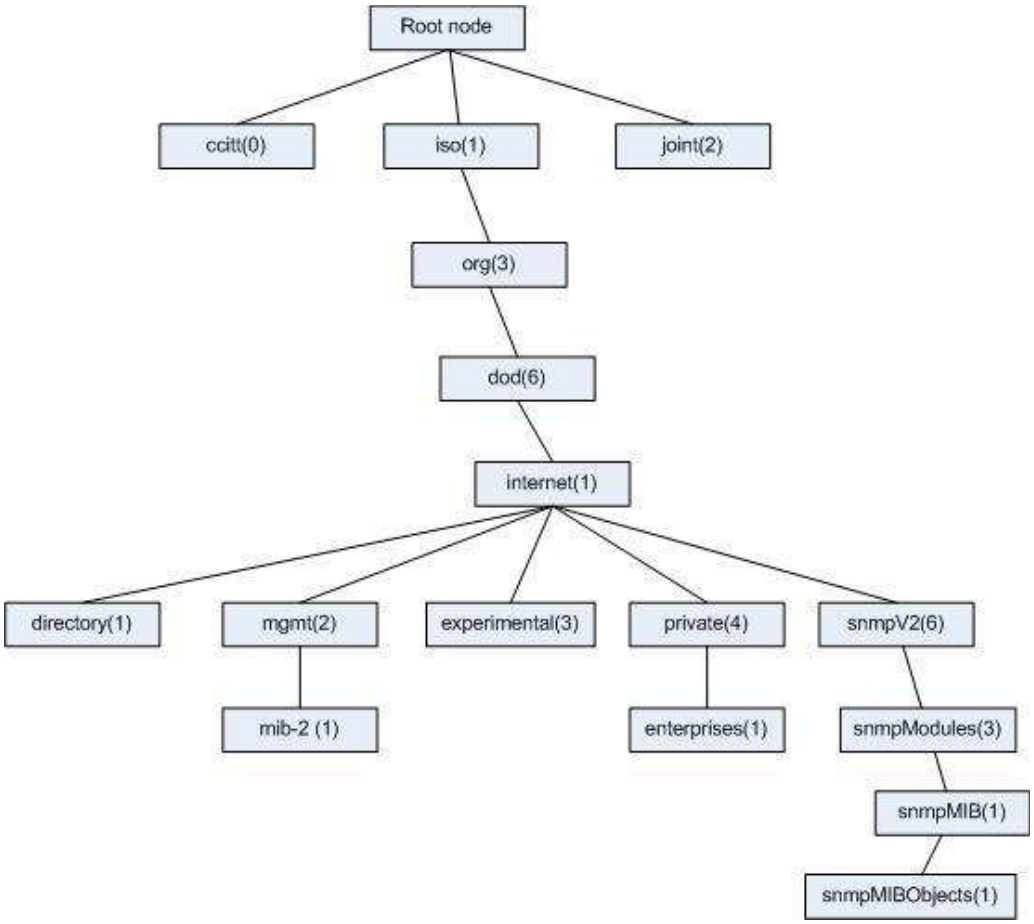


Figura 1. 7 Extensión del árbol jerárquico definido por SMIv2.

1.3.5.1.2 Sintaxis de Objetos

Cada objeto dentro de la MIB se define de una manera formal. La definición para un objeto MIB comprende lo siguiente: el tipo de dato, los estados permitidos, rango de valores, y la relación con otros objetos dentro de la MIB.

La notación **ASN.1 (*Abstract Syntax Notation One – Notación de Sintaxis Abstracta Uno*)** es un método que se usa para definir cada objeto y también para definir la estructura completa de la MIB. Lo más relevante de ASN.1 es su independencia de plataforma, ya que permite a dos computadores, de diferente hardware y software, comunicarse sin inconvenientes y sin preocuparse del ordenamiento de los bits.

SMIv1, utilizando un subconjunto de especificaciones de ASN.1, define los tipos de datos permitidos. Existen tres categorías para los tipos de datos:

- **Simples o primitivos:** Almacenan valores escalares (por ejemplo: un número o una cadena de texto).

- **Compuestos o estructurados:** Almacenan arreglos de valores (tablas).

- **Etiquetas o definidos:** Destinados a objetos específicos y están basados en una combinación de los anteriores.

TIPOS DE DATOS ABSTRACTOS		
	TIPO DE DATO	**DESCRIPCIÓN**
SIMPLES	INTEGER	Utilizado para representar números de hasta 32 bits. El número cero (0) no es posible representarlo con este tipo de dato.
	OCTET STRING	Usado para representar una cadena de cero o más octetos.
	OBJECT IDENTIFIER	Usado para representar la identificación de un objeto (OID).
	NULL	Empleado para la representación de ausencia de valor.
COMPUESTOS	SECUENCE	Para representar estructuras ordenadas que contienen datos simples. Se usa para el almacenamiento de una fila.
	SECUENCE OF	Para representar objetos que contienen datos del tipo *Sequence.* Se usa para almacenar las filas de una tabla.

DEFINIDOS	NetworkAddress	Para representar tipos de direcciones de red (IP, IPX, etc.).
	IpAddress	Usado para representar direcciones de red IPv4 de 32 bits.
	Counter	Empleado para representar contadores de números enteros positivos de 32 bits. Sólo incremento. El rango de valores es desde 0 hasta 2^{32}-1.
	Gauge	Similar a Counter pero con la diferencia que también soporta decremento.
	TimeTicks	Es un tipo de dato usado para medir tiempo. La resolución es del orden de centésimas de segundo. Es un número entero positivo de 32 bits.
	Opaque	Permite que cualquier otra codificación ASN.1 pueda incluirse dentro de un tipo de dato *Octet String.*

Tabla 1. 2 Tipos de datos abstractos definidos por ASN.1 y aceptados por SMIv1.

Según ASN.1, los tipos de datos están clasificados de modo más general y se les denomina **marcas.** Una marca está formada por un nombre de marca y un número de marca. Existen cuatro tipos de marcas:

- **Universal:** A esta marca pertenecen los tipos de datos Primitivos y los Estructurados. Estos tipos de datos son independientes de la aplicación y son de uso general.

- **Aplicación:** A esta marca pertenece el tipo de dato Definido. Estos tipos de datos son determinados por cada aplicación particular.

- **Contexto específico:** A esta marca pertenecen los tipos de datos definidos para aplicaciones particulares. Por ejemplo, las PDUs de SNMP.

- **Privada:** A esta marca pertenecen los tipos de datos propietarios.

Tipo de dato	Nombre de marca	Identificador de marca
INTEGER	UNIVERSAL	2
OCTET STRING	UNIVERSAL	4
OBJECT IDENTIFIER	UNIVERSAL	6
NULL	UNIVERSAL	5
SECUENCE	UNIVERSAL	16
SECUENCE OF	UNIVERSAL	16
IpAddress	APPLICATION	0
Counter	APPLICATION	1
Gauge	APPLICATION	2
TimeTicks	APPLICATION	3
Opaque	APPLICATION	4

Tabla 1. 3 Marcas de los tipos de datos.

Con SMIv2 se agregan nuevos tipos de datos, éstos se detallan en la tabla 1.4.

TIPO DE DATO	DESCRIPCIÓN
Integer32	La misma funcionalidad que de un tipo de dato INTEGER.
Counter32	La misma funcionalidad que de un tipo de dato Counter.
Gauge32	La misma funcionalidad que de un tipo de dato Gauge.
Unsigned32	Para representar valores decimales dentro del rango de 0 a 2^{32}-1, inclusive.
Counter64	Similar a Counter32, pero el máximo valor es 2^{64}-1.
BITS	Para la enumeración de bits no negativos.

Tabla 1. 4 Tipos de datos abstractos definidos por SMIv2.

1.3.5.1.3 Definición de Objetos

SNMP utiliza la macro OBJECT-TYPE (definida en SMIv1) la cual contiene un conjunto de notaciones que sirven para definir cualquier objeto. La macro está compuesta principalmente por las siguientes notaciones:

- **SYNTAX:** Especifica el tipo de datos que manejará el objeto. Esto puede ser: *Integer, Octect String, IpAddress*, etc.

- **ACCESS:** Especifica el nivel de permiso para manejar al objeto. Las opciones son: lectura, lectura y escritura, escritura o no accesible.

- **STATUS:** Define si la utilización del objeto es: *mandatory*, *optional*, *obsolete, deprecated.* El estado *obsolete* significa que los dispositivos gestionados no deberían usar más ese objeto. *Deprecated* significa que el objeto puede ser soportado, pero debería ser removido de la próxima versión de la MIB.

- **DESCRIPTION:** Una explicación textual del objeto.

Luego, SMIv2 agrega nuevas notaciones para la definición de objetos; éstas se describen a continuación:

- **UnitParts:** Una descripción textual de las unidades (por ejemplo: segundos, milisegundos, etc.) utilizadas para representar el objeto.

- **MAX-ACCESS:** Usado en SNMPv2. Permite que un objeto ofrezca mayores posibilidades de acceso a la información. Entre las posibles opciones se tiene: *read-only, read-write, read-create, not-accesible* y *accesible-for-notify.*

- **STATUS:** Esta notación ha sido extendida para permitir otros estados: *current, obsolete* y *deprecated.* El estado *current* en SNMPv2 significa lo mismo que *mandatory* en SNMPv1.

- **AUGMENTS:** Permite extender una tabla añadiendo una o más columnas, representado por algún otro objeto.

1.3.5.1.4 Codificación de los valores de los objetos

La codificación es el mecanismo a través del cual la información es traducida a símbolos para luego ser transmitida.

Los valores de los objetos son codificados a una cadena de octetos empleando las normas **BER (*Basic Encoding Rules - Reglas de Codificación Básica*)**. BER define cómo deben codificarse y decodificarse los valores de cada tipo ASN.1, de tal modo que puedan ser transmitidos a través de la red.

BER provee el formato TLV (*Type-Length-Value; Tipo-Longitud-Valor*) para realizar la codificación; esto significa, que cualquier valor ASN.1 puede codificarse como un conjunto de los siguientes elementos:

- **Tipo:** Indica el tipo de marca, el número de marca y el tipo de formato.

- **Longitud:** Indica la longitud (número de bytes) de la representación del valor.

- **Valor:** Representa el valor del tipo ASN.1 a manera de cadena de octetos.

La estructura TLV es recursiva; esto significa que para cualquier valor ASN.1 el componente valor de la TLV se codifica así mismo como una o más estructuras TLV.

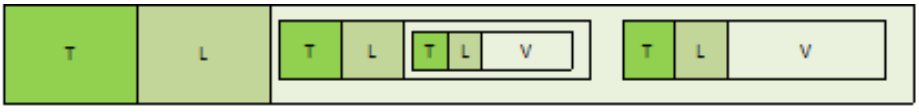


Figura 1. 8 Ejemplo de recursividad de la estructura TLV.[9]

[9] VELÁSQUEZ, Andrés; Diseño e implementación de un módulo software para la monitorización de Elementos de una red informática utilizando el protocolo SNMP y el lenguaje XML, Figura 1.9 tomada de la pág.24. Acceso: 11 de diciembre 2009, 15:00.

El campo tipo está conformado por 8 bits y tiene la siguiente forma:

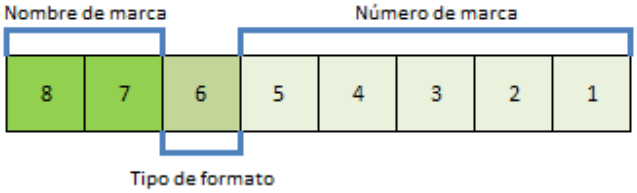


Figura 1. 9 Detalle del campo tipo.[10]

El nombre de marca está definido por los bits 8 y 7. Las combinaciones de estos dos bits representan a una clase o marca de datos. La tabla 1.5 muestra la relación de los nombres de marca con la respectiva combinación de dichos bits.

Bit 8	Bit 7	Nombre de Marca
0	0	Universal
0	1	Aplicación
1	0	Contexto específico
1	1	Privada

Tabla 1. 5 Relación de los Nombres de Marca con los bits 8 y 7.[11]

Luego, el bit 6 define el tipo de formato del campo valor, el cual puede ser: 0 (**primitivo,** por ejemplo si el valor es del tipo *Integer*) o 1 (**construido o estructurado**, por ejemplo si el valor es del tipo *sequence of*).

Los siguientes 5 bits se utilizan para representar el número de marca. Si el número de marca a representar en modo binario está entre 0 y 30, entonces los 5 bits son suficientes. Sin embargo, de ser el número mayor o igual a 31, estos cinco bits son colocados a 1 y el número de marca se representaría como la sumatoria de todos los

[10] VELÁSQUEZ, Andrés; Diseño e implementación de un módulo software para la monitorización de Elementos de una red informática utilizando el protocolo SNMP y el lenguaje XML, Figura 1.10 tomada de la pág.25. Acceso: 12 de diciembre 2009, 14:30.
[11] VELÁSQUEZ, Andrés; Diseño e implementación de un módulo software para la monitorización de Elementos de una red informática utilizando el protocolo SNMP y el lenguaje XML, Tabla 1.3 tomada de la pág.25. Acceso: 12 de diciembre 2009, 14:40.

octetos que han sido agregados. Estos octetos tendrían el bit más significativo en 1 a excepción del último octeto el cual lo tendría en 0.

El campo longitud tiene dos formatos: el **formato corto** y el **formato largo**. El formato corto se lo emplea cuando el número de bytes del campo valor es menor o igual a 127 octetos; si éste es el caso, el bit más significativo se lo setea a 0 y el resto de bits serán la longitud en sí. En cambio, el formato largo se lo emplea cuando el número de bytes del campo valor son mayores a 127; sí este es el caso, el bit más significativo se lo setea a 1 y los 7 bits restantes representan la cantidad de octetos que se van a adicionar. Los octetos agregados son los que van a especificar la longitud del campo valor. El número de octetos adicionales están limitados desde 1 hasta 126.

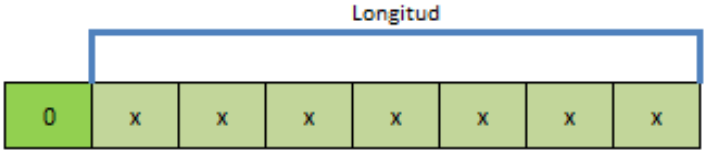


Figura 1. 10 Formato corto del campo longitud.[12]

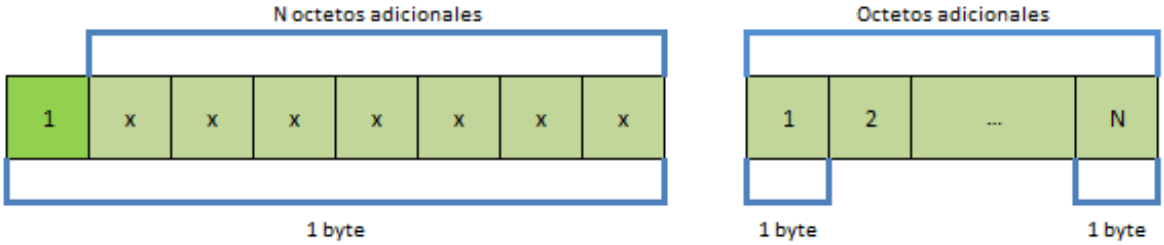


Figura 1. 11 Formato largo del campo longitud.[13]

Finalmente, el campo valor es el aquel que codifica el valor del objeto. En este campo se introducen los tipos de dato abstractos y los mensajes o PDU de SNMP. La forma de codificación del campo valor dependerá del tipo de dato a introducir.

[12] VELÁSQUEZ, Andrés; Diseño e implementación de un módulo software para la monitorización de Elementos de una red informática utilizando el protocolo SNMP y el lenguaje XML, Figura 1.11 tomada de la pág.26. Acceso: 12 de diciembre 2009, 17:00

[13] VELÁSQUEZ, Andrés; Diseño e implementación de un módulo software para la monitorización de Elementos de una red informática utilizando el protocolo SNMP y el lenguaje XML, Figura 1.12 tomada de la pág.26. Acceso: 12 de diciembre 2009, 17:10.

1.3.6 OPERACIONES SNMP [8][10]

En el modelo de gestión de Internet, el NMS y el agente se comunican mediante mensajes SNMP. Cada mensaje, para el caso de SNMPv1 y SNMPv2, incluye los siguientes campos:

- **Versión:** Número de versión del protocolo SNMP.

- **Comunidad:** Nombre de comunidad empleado para el procedimiento de autenticación.

- **PDU:** Contiene cualquiera de los tipos de PDU.

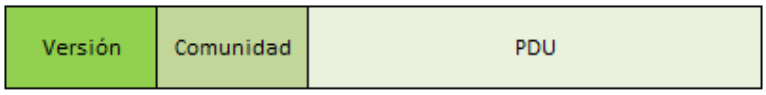


Figura 1. 12 Formato de mensaje SNMP (SNMPv1 y SNMPv2).[14]

1.3.6.1 Tipos de PDU

1.3.6.1.1 GetRequest

El mensaje SNMP que contiene esta PDU es iniciado por el NMS el cual solicita al agente el valor de algún objeto. El agente recibe este mensaje, lo procesa o no dependiendo de su capacidad de procesamiento, y envía una respuesta al NMS indicando el éxito (información solicitada) o fracaso (algún tipo de error) de la consulta. El formato de la PDU *GetRequest* es el mismo para las PDUs *GetNextRequest, SetRequest, GetReponse, InformRequest y Report* en todas las versiones de SNMP (SNMPv1, SNMPv2 y SNMPv3).

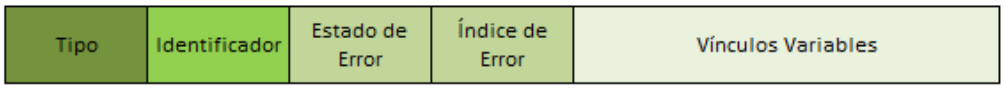


Figura 1. 13 Formato de PDU para GetRequest, GetNextRequest, SetRequest, GetReponse, InformRequest y Report.[15]

[14] STALLINGS William; SNMP, SNMPv2, SNMPv3 and RMON 1 and 2, Tercera Edición, Figura 7.3 (a) tomada de la pág.172. Acceso: 12 de diciembre 2009, 18:00.

- **Tipo:** Especifica el tipo de PDU SNMP.

- **Identificador:** Es un número asignado por el NMS que identifica de manera única un pedido y lo asocia con la respuesta del agente. Este número también ayuda al NMS a distinguir PDUs duplicadas debidas al servicio de transporte no confiable.

- **Estado de Error:** Es un número que indica el tipo de error producido.

- **Índice de Error:** Es un número que indica la variable (dentro de los vínculos variables) donde se ocasionó el error.

- **Vínculos Variables:** En este campo están enlistadas las instancias de los objetos cuyos valores son requeridos. La forma como se presenta es: nombre de la variable (instancia de objeto) y valor asignado.

LISTADO DE ERRORES			
	No.	**Nombre del Error**	**Descripción**
SNMPv1	0	noError	No se produjo error en el intercambio de mensajes SNMP.
	1	tooBig	El tamaño del PDU ha superado la capacidad máxima permitida.
	2	noSuchName	La variable de la cual se solicitó su valor no es válida.
	3	badValue	Codificación errónea en la variable del mensaje de PDU *SetRequest*.
	4	readOnly	El valor de la variable solicitada es sólo de lectura.
	5	genError	Para errores que no pertenecen a los anteriores.

[15] STALLINGS William; SNMP, SNMPv2, SNMPv3 and RMON 1 and 2, Tercera Edición, Figura 7.3 (c) tomada de la pág.172., Acceso: 13 de diciembre 2009, 15:30.

SNMPv2	6	noAccess	Cuando se trata de acceder a un objeto con *max-access* igual a *not-accesible.*
	7	wrongType	El valor a asignar a una variable no concuerda con el tipo de dato aceptado.
	8	wrongLength	El valor a asignar a una variable supera la longitud permitida.
	9	wrongEncoding	El valor contiene una codificación ASN.1 inconsistente.
	10	wrongValue	El valor no puede ser asignado a la variable bajo ninguna circunstancia.
	11	noCreation	Intento de cambiar el valor de una variable que no existe.
	12	inconsistentValue	El valor puede, bajo otras circunstancias, ser asignado a la variable pero actualmente es inconsistente.
	13	resourceUnavailable	El cambio de valor de una variable no ha sido realizado debido a que el recurso requerido no está disponible.
	14	commitFailed	Se utiliza cuando existen errores por fallas en las operaciones de escritura.
	15	undoFailed	Al existir fallas de escritura, el agente deja el último valor estable.
	16	authorizationError	Nombre de comunidad erróneo.
	17	notWritable	No es permitido realizar procesos de escritura sobre dicha variable.
	18	inconsistentName	Intento de cambiar el valor de una variable cuyo nombre es inconsistente.

Tabla 1. 6 Listado de errores definidos para SNMPv1 y SNMPv2.

1.3.6.1.2 GetNextRequest

La PDU *GetNextRequest* tiene el mismo patrón de intercambio y formato de la PDU *GetRequest*. La diferencia está en que el agente que responde al pedido observa en el campo *variable bindings* el nombre del objeto y retorna la instancia del objeto que sigue en orden lexicográfico. El orden lexicográfico es un método de ordenamiento, el cual hace una relación "≤" (menor igual que) con cada par de caracteres de los nombres de objetos y determina que objeto va continuación de otro. [9]

La PDU *GetNextRequest* le permite a una NMS descubrir dinámicamente la estructura de una vista MIB y el contenido de objetos tablas. El formato de la PDU *GetNextRequest* es la misma en todas las versiones de SNMP.

1.3.6.1.3 GetBulkRequest

Este tipo de PDU está disponible solo para SNMPv2 y SNMPv3. La PDU *GetBulkRequest* es un tipo de mensaje SNMP que facilita la recuperación de grandes cantidades de información con el mínimo intercambio de mensajes para el efecto. Esta operación SNMP sigue el mismo proceso de selección de la PDU *GetNextRequest*, es decir, selecciona la próxima instancia de objeto en orden lexicográfico. La diferencia radica en que *GetBulkRequest* permite especificar el número de sucesores lexicográficos a seleccionar. En la figura 1.14 se muestra el formato de la PDU *GetBulkRequest.*

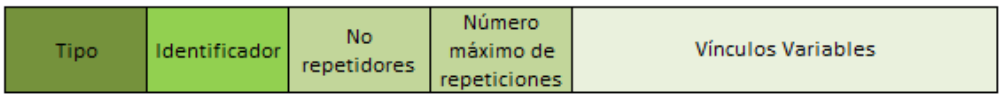


Figura 1. 14 Formato de PDU para GetBulkRequest .[16]

Dentro de los campos de la PDU *GetBulkRequest*, los más relevantes son:

[16] STALLINGS William; SNMP, SNMPv2, SNMPv3 and RMON 1 and 2, Tercera Edición, Figura 12.1 (c) tomada de la pág.368, Acceso: 15 de diciembre 2009, 12:20.

- **No repetidores:** Número de variables en el campo vínculos variables para las cuales debe ser retornado un único sucesor lexicográfico.

- **Número máximo de repeticiones:** Número de sucesores lexicográficos que deben retornarse para las restantes variables en el campo vínculos variables.

- **Vínculos Variables:** Contendrá una lista de variables que incluye aquellas que requieren un solo sucesor lexicográfico y aquellas que requieren de múltiples sucesores lexicográficos.

El número total de instancias de objetos que serán devueltos como respuesta a la solicitud *GetBulkRequest* obedece a la siguiente relación:

$$\#total\ de\ obj.recuperados = N + M * R$$

Donde:
N = número de variables (objetos escalares) para las cuales se requiere un solo sucesor lexicográfico.
R = número de variables (objetos tablas) para las cuales se requiere múltiples sucesores lexicográficos.
M = número de sucesores lexicográficos para las R variables.

1.3.6.1.4 SetRequest

La PDU *SetRequest* tiene el mismo procedimiento de intercambio y el mismo formato de la PDU *GetRequest.* Su función es la de escribir un valor en la instancia de un objeto especificado. Si la operación es exitosa, el agente envía una PDU *GetResponse* a la NMS con los nuevos valores que han sido actualizados; caso contrario se envía una PDU *GetResponse* especificando el tipo de error suscitado y la variable en donde ocurrió. El formato de la PDU *SetRequest* es la misma en todas las versiones de SNMP.

1.3.6.1.5 GetResponse

El mensaje SNMP que contiene la PDU *GetResponse* es enviado por el agente hacia el NMS debido a la petición solicitada por éste. Si la operación es exitosa el campo vínculos variables contendrá la información requerida; caso contrario, no se devolverá ningún valor (en el caso de SNMPv1 porque es operación atómica) o se devolverán sólo aquellos que se encuentran disponibles (en el caso de SNMPv2 y SNMPv3); además, se notificará el tipo de error y la variable en que sucedió. El formato de la PDU *GetResponse* es el mismo en todas las versiones de SNMP.

1.3.6.1.6 Trap

El mensaje SNMP que contiene la PDU *Trap* es emitido asincrónicamente por el agente hacia el NMS para avisar sobre algún suceso imprevisto que requiere ser atendido. La PDU *Trap* no genera una respuesta desde el otro lado (la estación administradora de red). El formato de la PDU *Trap* es distinto en SNMPv1 y en SNMPv2 (la PDU propiamente se denomina *Notification - Notificación*).

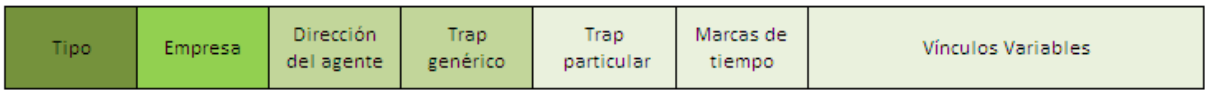


Figura 1. 15 Formato de PDU Trap en SNMPv1.[17]

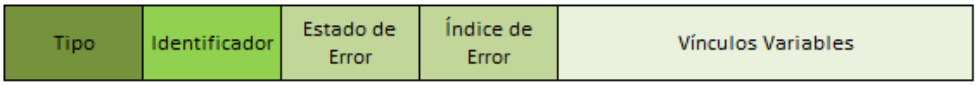


Figura 1. 16 Formato de PDU Trap en SNMPv2.[18]

El formato de la PDU Trap SNMPv1 tienen los siguientes campos:

- **Tipo:** Especifica el tipo de PDU SNMP.

[17] STALLINGS William; SNMP, SNMPv2, SNMPv3 and RMON 1 and 2, Tercera Edición, Figura 7.3 (d) tomada de la pág.174, Acceso: 16 de diciembre 2009, 15:00.
[18] STALLINGS William; SNMP, SNMPv2, SNMPv3 and RMON 1 and 2, Tercera Edición, Figura 12.1 (a) tomada de la pág.368, Acceso: 16 de diciembre 2009, 15:00.

- **Empresa:** Es un valor del tipo OID que identifica al fabricante del dispositivo de red y del agente.

- **Dirección del agente:** Dirección IP del agente que emitió el *trap*.

- **Trap genérico:** Es un número entero que identifica el tipo de *trap* predefinido.

- **Trap particular:** Es un código que especifica la naturaleza del *trap*.

- **Marcas de tiempo:** Es el tiempo transcurrido desde la última (re)inicialización del agente hasta la generación del primer *trap.*

- **Vínculos variables:** Este campo contiene información adicional relacionada con el *trap*.

LISTADO DE TRAPS PREDEFINIDOS		
Id.	**Nombre**	**Descripción**
0	coldStart	Es cuando el agente se reinicia inesperadamente debido a una falla grave. Pueden producirse alteraciones en la configuración del agente.
1	warmStart	Es cuando el agente se reinicia de modo normal. No se producen alteraciones en la configuración del agente.
2	linkDown	Informa falla en el enlace de comunicación del agente.
3	linkUp	Avisa recuperación del enlace antes caído.
4	authenticationFailure	Comunica que el agente recibió un mensaje SNMP indicando que ha fallado la autenticación.
5	egpNeighborLoss	Anuncia que un vecino EGP para quien el agente era un par EGP no está funcionando y la relación

		ya no existe.
6	enterpriseSpecific	Es cuando se produce un evento especial y el campo *Trap particular* lo especifica.

Tabla 1. 7 Listado de Traps predefinidos.

Dentro de los campos de la PDU *Trap* SNMPv2, el más relevante es **Vínculos Variables** y este campo contiene los siguientes valores:

- **sysUpTime**: Tiene una función similar al campo marcas de tiempo del *trap* SNMPv1.

- **snmpTrapOID**: Identifica el tipo de *trap*. Parte del grupo *trap* en la MIB *snmpv2.*

1.3.6.1.7 InformRequest

La PDU *InformRequest* está disponible sólo para SNMPv2 y SNMPv3. El mensaje SNMP que contiene esta PDU se envía entre dos NMS para proporcionarse información de gestión. Cuando el mensaje alcanza su destino y no tiene errores, la estación receptora confirma la llegada del mensaje enviado un mensaje *GetResponse* con los mismos valores (identificador y vínculos variables) del mensaje *InformRequest;* si el mensaje tiene errores se notifica el tipo de error. La PDU *InformRequest* tiene el mismo formato de la PDU *Notification.*

1.3.6.1.8 Report

La operación *Report* fue definida en la versión draft (borrador) de SNMPv2 pero jamás fue implementada; sin embargo, ahora es parte del estándar SNMPv3 y su función básicamente es permitir que los Motores SNMP puedan comunicarse unos a otros (principalmente para reportar problemas de procesamiento de mensajes SNMP). La PDU *Report* tiene el mismo formato de la PDU *GetRequest.*

1.4 SNMPv3 [7]

La seguridad ha sido una de las principales debilidades de SNMP en sus primeras versiones. La autenticación en SNMPv1 y SNMPv2 está basada en un password (nombre de comunidad) el cual es enviado en texto plano entre el sistema gestor y el agente. Esto representa un grave riesgo ya que el nombre de comunidad puede ser capturado, interpretado y utilizado para: obtener información de gestión desde los dispositivos de red, modificar parámetros de configuración o dar de baja a los dispositivos. Para solucionar esto se ha desarrollado SNMPv3.

SNMPv3 ha sido creado esencialmente para resolver la deficiente seguridad existente en versiones anteriores. La seguridad es el principal tema tratado en SNMPv3 y no existen otros cambios significativos en el protocolo; sin embargo, se han introducido nuevos conceptos, convenciones y terminologías que definen de manera precisa a los componentes ya conocidos de SNMP. Además, en SNMPv3 no se definen nuevas operaciones (tipos de PDU) y más bien soporta aquellas ya existentes.

En síntesis, SNMPv3 es un protocolo de gestión de red que define una nueva arquitectura que ofrece seguridad de acceso a los dispositivos por medio de una combinación de autenticación y encriptación de los mensajes.

Número	**Nombre**
RFC 3411	*Arquitectura para sistemas SNMP.*
RFC 3412	*Despacho y procesamiento de mensajes SNMPv3.*
RFC 3413	*Aplicaciones SNMPv3.*
RFC 3414	*USM (User-based Security Model - Modelo de Seguridad basada en el Usuario)*
RFC 3415	*VACM (View-based Access Control Model – Modelo de Control de Acceso basado en Vistas).*

RFC 3416	*Operaciones de protocolo para SNMPv2.*
RFC 3417	*Mapeo de transporte para SNMPv2*
RFC 3418	*MIBs para SNMPv2.*
RFC 2576	*Coexistencia entre versiones de SNMP.*
RFC 2570	*Introducción a SNMPv3.*
RFC 2786	*Administración de la llave Diffie-Hellman para USM.*

Tabla 1. 8 RFCs para SNMPv3.

1.4.1 CARACTERÍSTICAS DE SNMPv3 [8]

- Define una arquitectura y mecanismos seguridad basados en SNMPv2u y SNMPv2* (éstas son versiones posteriores a SNMPv2).

- La arquitectura está constituida por entidades que a su vez están conformadas por subsistemas los cuales interactúan entre sí para proporcionar servicios abstractos.

- Permite mover diferentes partes de la arquitectura pero dentro del estándar; por ejemplo, se pueden poner modelos de seguridad alternativos al determinado por el estándar.

- SNMPv3 ofrece protección contra las siguientes amenazas:
  - **Modificación de información:** Cuando una entidad no autorizada altera un mensaje ya en tránsito enviado por una entidad autorizada y puede realizar modificaciones indebidas en los valores de objetos MIB.
  - **Enmascaramiento:** Cuando una entidad asume la identidad de una entidad autorizada para realizar operaciones de gestión ilícitas.
  - **Modificación del flujo de mensajes:** Cuando los mensajes SNMP son reordenados, retrasados o reenviados con el fin de realizar operaciones de gestión no autorizadas.

- **Revelación:** Cuando una entidad observa los intercambios de mensajes entre un gestor y un agente, y aprende los valores de los objetos gestionados y la notificación de eventos.

- SNMPv3 no ofrece protección para las siguientes amenazas:
  - **Negación de Servicio:** Cuando una entidad impide el intercambio de mensajes entre un gestor y un agente.

  - **Análisis de Tráfico:** Cuando una entidad observa el patrón general de tráfico entre gestores y agentes.

- SNMPv3 provee configuración remota para los parámetros de los subsistemas que se encargan de la seguridad y restricción de acceso.

- SNMPv3 permite tener a disposición mayores funcionalidades para realizar configuraciones más complejas pero utilizando moderadamente los recursos, afín de seguir manteniendo la simplicidad del protocolo.

1.4.2 ARQUITECTURA SNMP [7][8][11]

En SNMPv3 no existe el concepto de gestores y agentes. Ambos, administradores y agentes, ahora se denominan **entidades SNMP**. Cada entidad está formada por un único **motor SNMP** y por una o más **aplicaciones SNMP**. El motor y las aplicaciones a su vez están formados por diferentes **subsistemas** o **módulos**. Los subsistemas a su vez están conformados por **modelos**. Los subsistemas interactúan entre sí mediante primitivas y parámetros abstractos con el fin de que una entidad pueda proveer un determinado servicio. Dependiendo de los tipos de módulos que conformen una entidad, ésta podrá actuar como: agente, gestor o una combinación de ambos.

La arquitectura SNMP tiene varias ventajas:

- Mejoramiento de los módulos sin alterar las especificaciones del protocolo.

- Permite el desarrollo de estrategias de coexistencia y transición con otros módulos.

- Definición del rol de una entidad en base a los módulos que tenga implementado.

1.4.2.1 Entidades SNMP

La figura 1.17 muestra un diagrama de bloques de una entidad SNMP.

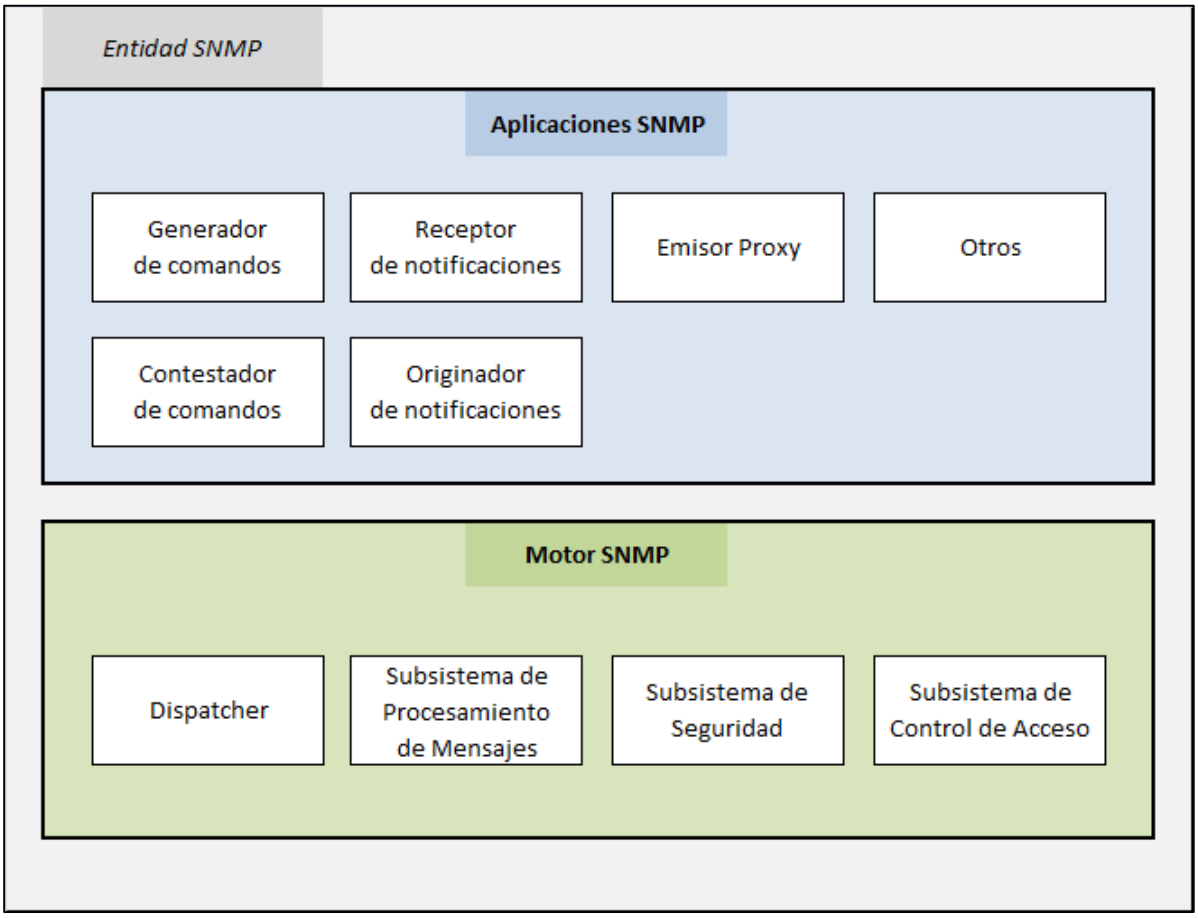


Figura 1. 17 Entidad SNMP (RFC 3411).[19]

[19] STALLINGS William; SNMP, SNMPv2, SNMPv3 and RMON 1 and 2, Tercera Edición, Figura 15.3 tomada de la pág.456, Acceso: 17 de enero 2010, 08:00.

1.4.2.1.1 Motor SNMP

El **motor SNMP** es la parte esencial de cualquier entidad SNMP y se encarga de proveer las siguientes funcionalidades:

- Envío y recepción de mensajes SNMP.
- Autenticación y encriptación/desencriptación de mensajes SNMP.
- Control de acceso a los objetos MIB.

Estas funciones se suministran como un servicio a una o más aplicaciones SNMP que se conformaron junto al motor SNMP para formar una entidad SNMP.

Un motor SNMP es identificado de una manera única e inconfundible por un **snmpEngineID** dentro de un dominio de gestión[20]. Este identificador tiene sintaxis de *Octet String* y es asignado con valores hexadecimales. Debido a que una entidad SNMP puede tener un sólo motor SNMP, entonces el *snmpEngineID* también permite identificar de manera única a una entidad SNMP.

Un Motor SNMP puede estar compuesto por los siguientes módulos:

- **Dispatcher:** Se encarga de enviar y recibir mensajes SNMP sobre la red. Este módulo determina la versión de cada mensaje recibido (es decir v1, v2 o v3), y si la versión es soportada, transfiere el mensaje al Subsistema de Procesamiento de Mensajes. Si un mensaje ya está creado y listo para ser enviado, el Subsistema de Procesamiento de Mensajes lo transmite al Dispatcher para que éste sea enviado sobre la red. El Dispatcher también realiza intercambios de PDUs entrantes o salientes entre Aplicaciones SNMP.

[20] **Dominio de gestión**: Es el conjunto de todos los agentes que informan al mismo gestor. http://inspeccionumvi11.iespana.es/ind12287.pdf

- **Subsistema de Procesamiento de Mensajes:** Su función es armar los mensajes para que sean enviados y extraer PDUs de los mensajes recibidos. Un Subsistema de Procesamiento de Mensajes puede contener varios modelos de procesamiento de mensajes para SNMPv1 (denominado v1MP), para SNMPv2 (denominado v2cMP), para SNMPv3 (denominado v3MP) y para otros protocolos de gestión.

- **Subsistema de Seguridad:** Proporciona servicios de autenticación y privacidad para los mensajes SNMP. Agrega un encabezado específico de seguridad en cada mensaje; este encabezado contiene información útil para los protocolos de autenticación y privacidad.

- **Subsistema de Control de Acceso:** Proporciona servicios de autorización para controlar el acceso a los objetos MIB, es decir determina qué objetos pueden ser accedidos y qué operaciones pueden ser ejecutadas en éstos.

1.4.2.1.2 Aplicaciones SNMP

Las **aplicaciones SNMP** son subsistemas que usan los servicios de un motor SNMP para llevar a cabo tareas específicas relacionadas al procesamiento de operaciones de información de gestión. Una entidad puede estar formada por una o más aplicaciones SNMP, éstas pueden ser las siguientes:

- **Generador de comandos:** Monitoriza y maneja los datos de administración de los dispositivos gestionados. Específicamente genera las PDUs: *GetRequest, GetNextRequest, GetBulkRequest* y *SetRequest*. Además, el Generador de Comandos recibe y procesa las respuestas (*GetResponse*) a los pedidos que ha generado. Esta aplicación es implementada en la entidad que actúa como NMS.

- **Contestador de comandos***:* Recibe las PDUs *GetRequest, GetNextRequest, GetBulkRequest* y *SetRequest,* realiza las acciones solicitadas, y utilizando el

control de acceso genera mensajes *GetResponse* para que sean enviados a la entidad solicitante (NMS). Esta aplicación es implementada en la entidad que actúa como agente SNMP.

- **Originador de notificaciones:** Monitoriza eventos y genera *Traps* (SNMPv1 o v2) o *InformRequest* para sean enviados al NMS. Esta aplicación puede implementarse tanto en una entidad gestora como una entidad agente.

- **Receptor de notificaciones:** Espera por la llegada de mensajes *Traps* e *InformRequest*, y genera un mensaje GetResponse para el mensaje *InformRequest.* Esta aplicación es implementada en una entidad gestora.

- **Reenviador Proxy:** Reenvía mensajes entre entidades SNMP. La implementación de esta aplicación es opcional. Un reenviador proxy soporta todos los comandos SNMP, incluído aquel que contiene la PDU *Report.*

1.4.2.1.3 Gestor SNMP

La figura 1.18 muestra el diagrama de bloques de un **Gestor SNMP** tradicional.

Una entidad SNMP que contenga una o más aplicaciones: generadoras de comandos, receptoras de notificaciones y originadoras de notificaciones (junto con un motor SNMP) se llama **Gestor SNMP**. Todas estas aplicaciones utilizan los servicios proporcionados por el motor SNMP y éste realiza dos operaciones completas:

- Recibe las PDUs salientes desde las aplicaciones SNMP, realiza la inserción de códigos de autenticación y cifrado, y luego arma los mensajes SNMP para posteriormente ser enviados a través de la red.

- Recibe los mensajes SNMP desde la capa transporte (UDP), realiza el procedimiento de autenticación y descifrado, y luego extrae las PDU para posteriormente ser entregadas a las aplicaciones pertinentes.

En un Gestor SNMP, el motor SNMP debe contener los siguientes módulos: Dispatcher, un Subsistema de Procesamiento de Mensajes y un Subsistema de Seguridad.

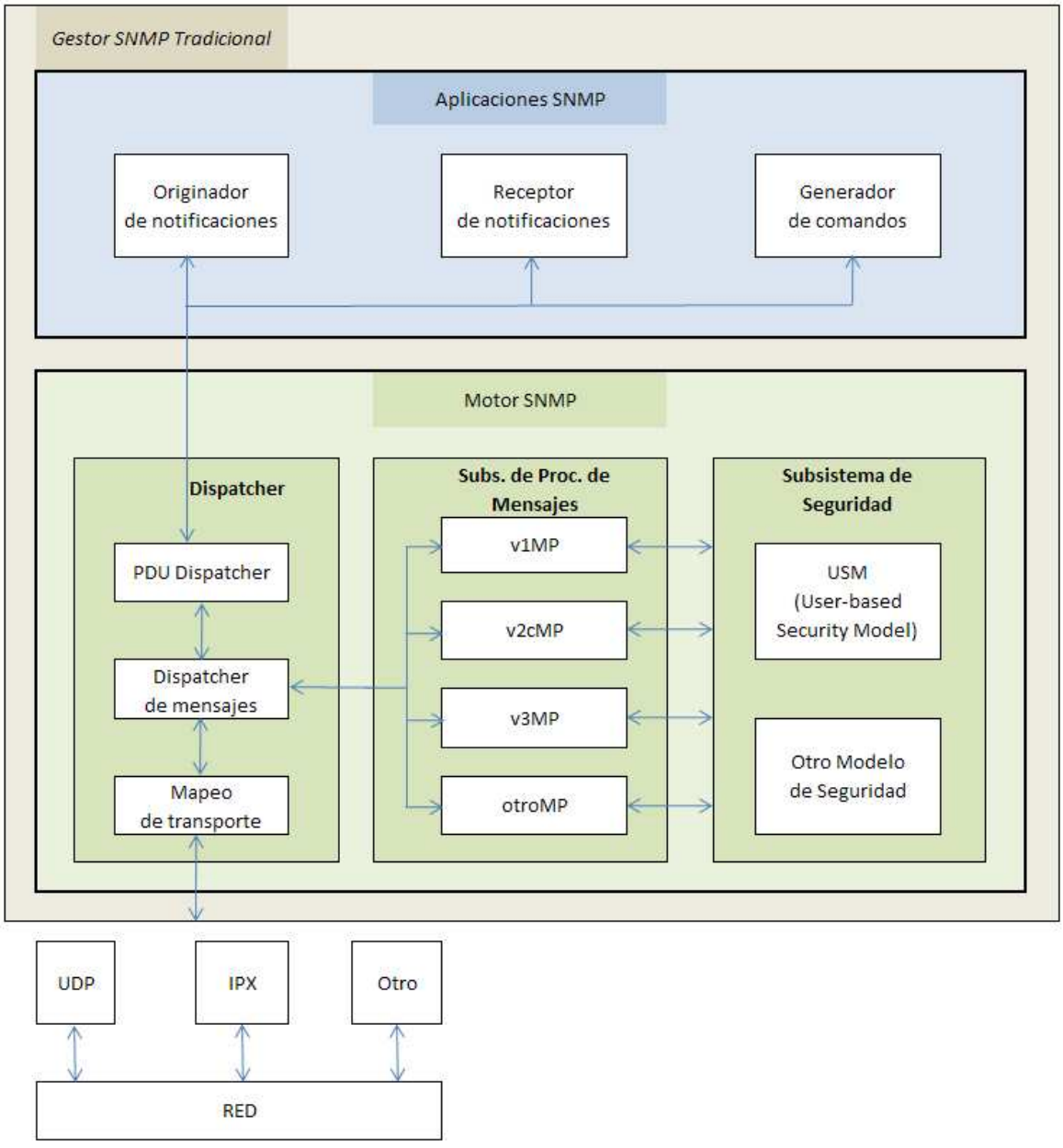


Figura 1. 18 Gestor SNMP Tradicional.[21]

[21] STALLINGS William; SNMP, SNMPv2, SNMPv3 and RMON 1 and 2, Tercera Edición, Figura 15.4 tomada de la pág.458, Acceso: 19 de enero 2010, 10:00.

1.4.2.1.4 Agente SNMP

La figura 1.19 muestra el diagrama de bloques de un *Agente SNMP* tradicional.

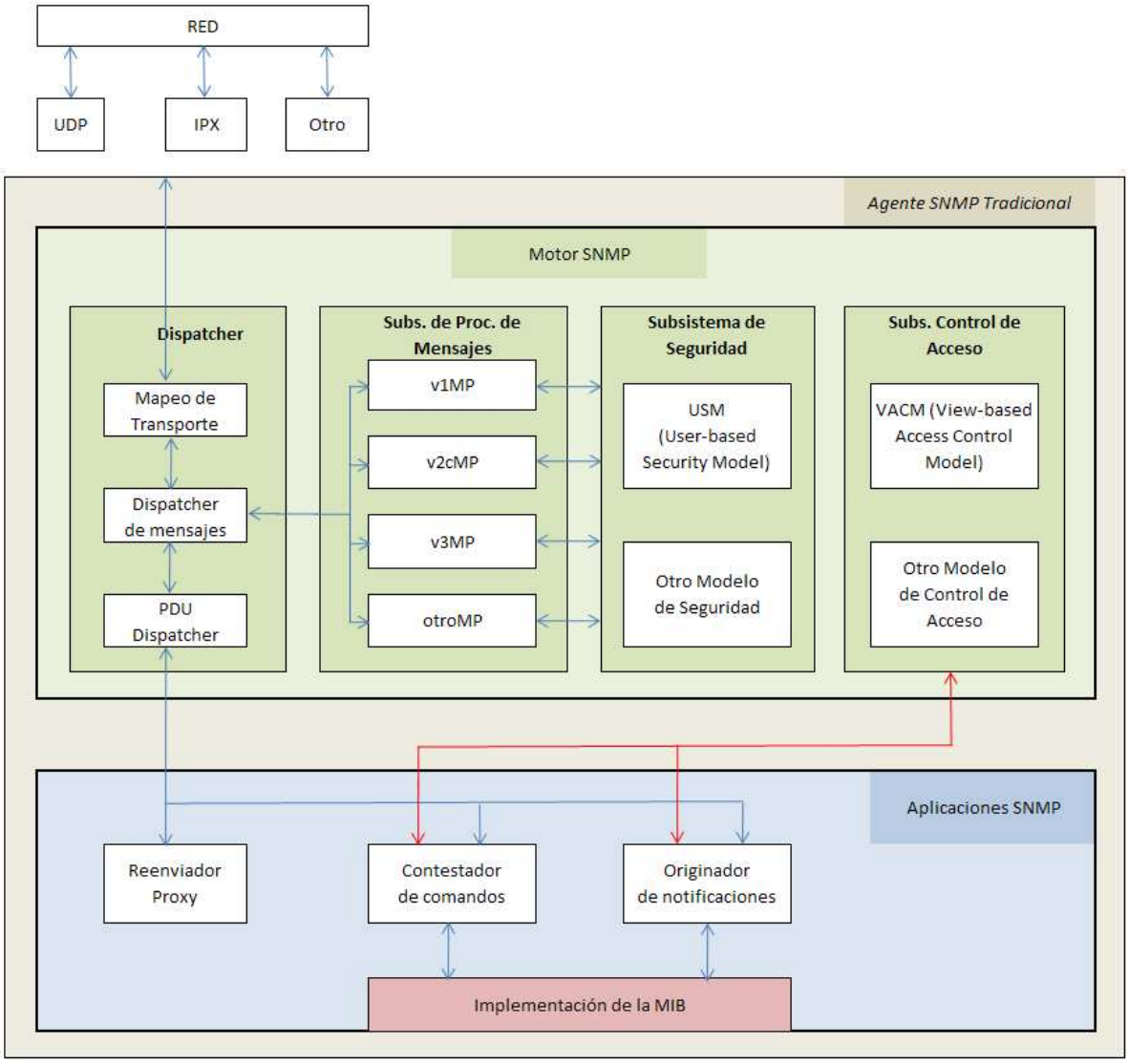


Figura 1. 19 Agente SNMP Tradicional.[22]

Una entidad SNMP que contenga una o más aplicaciones: contestadoras de comandos, originadoras de notificaciones y opcionalmente reenviadores Proxy (junto con un motor SNMP) se llama **Agente SNMP**.

[22] STALLINGS William; SNMP, SNMPv2, SNMPv3 and RMON 1 and 2, Tercera Edición, Figura 15.5 tomada de la pág.460, Acceso: 20 de enero 2010, 14:00.

El motor SNMP de un Agente SNMP contiene todos los subsistemas del motor de un Gestor SNMP más un Subsistema de Control de Acceso. Este subsistema provee servicios de autorización para controlar el ingreso a las MIBs al realizar procesos de lectura y escritura de los objetos gestionados. Estos servicios se ejecutan en base al contenido de las PDUs.

1.4.2.2 Terminología SNMPv3

Existen varios términos en SNMPv3 que son indispensables para el estudio en detalle de la arquitectura SNMP. A continuación:

- **Principal:** Es la entidad en nombre de quién se procesan los servicios (autenticación, privacidad y control de acceso). Generalmente, un Principal es un gestor SNMP.

- **securityName:** Es un nombre legible que identifica a un Principal.

- **snmpEngineID:** Es un identificador único de un motor SNMP y de la entidad SNMP que contiene dicho motor.

- **Contexto SNMP:** Es una colección de información de administración dentro de la implementación de la MIB que puede ser accedida por una entidad.

- **contextEngineID:** Es un identificador único de una entidad SNMP que puede realizar una instancia de un contexto SNMP. En la práctica, esta entidad es un Agente SNMP y el *contextEngineID* identifica a una aplicación (por ejemplo, una aplicación contestadora de comandos).

- **contextName:** Es un término usado para nombrar un contexto SNMP. Este nombre identifica a un contexto de manera única dentro de una entidad.

- **scopedPDU:** Es un bloque de datos que contiene un *contextEngineID*, un *contextName* y una PDU.

- **snmpMessageProcessingModel:** Es un número que identifica a un modelo de procesamiento del Subsistema de Procesamiento de Mensajes. Los valores posibles son: 0 (SNMPv1), 1 (SNMPv2c), 2 (SNMPv2u, SNMPv2*), 3 (SNMPv3).

- **snmpSecurityModel:** Es un número que identifica a un modelo de seguridad del Subsistema de Seguridad. Los posibles valores son: 0 (cualquier modelo), 1 (SNMPv1), 2 (SNMPv2c), 3 (SNMPv3, USM).

- **snmpSecurityLevel:** Determina el nivel de seguridad con el cual un mensaje SNMP ha sido enviado o con el cual ciertas operaciones están siendo procesadas. Los posibles valores son: *noAuthNoPriv* (sin autenticación y sin privacidad), *authNoPriv* (con autenticación pero sin privacidad), *authPriv* (con autenticación y privacidad).

1.4.2.3 Interfaces de servicios abstractos

Los servicios entre subsistemas de una entidad se definen por medio de **primitivas** y **parámetros**. Una primitiva representa a una función a realizar o un servicio a solicitar, y los parámetros representan datos e información de control. Una interfaz definida por primitivas y parámetros se denomina **Interfaz de Servicios Abstractos.**

En SNMPv3 se especifican tres tipos de valores que intervienen en un servicio:

- **Valor de entrada:** es un parámetro que se pasa como argumento dentro de una primitiva desde el subsistema llamante al subsistema llamado.

- **Valor de salida:** es un parámetro que se pasa como argumento dentro de una primitiva desde el subsistema llamado al subsistema llamante.

- **Valor primitivo:** es un código asignado a la primitiva en conjunto a la respuesta retornada.

El formato que tienen las primitivas es el siguiente:

$$function_code = request_service(IN\ w\ \ IN\ x\ \ OUT\ y\ \ OUT\ z)$$

Donde:

- request_service Nombre de la primitiva.
- w, x Valores de entrada.
- y, z Valores de salida.
- function_code Valor primitivo.

1.4.2.3.1 Primitivas del Dispatcher

Estas primitivas definen la comunicación entre el Dispatcher y las Aplicaciones SNMP. Las primitivas del Dispatcher son las siguientes:

- **sendPDU:** Utilizada por el Generador de Comandos o el Originador de Notificaciones para solicitar al Dispatcher la preparación de un mensaje de solicitud o de notificación para que sea enviado hacia otra entidad SNMP. Esta primitiva devuelve el valor primitivo *statusInformation.* Si el Dispatcher pudo preparar el mensaje, *statusInformation* será asignado con el valor del parámetro *sendPduHandle,* el cual es utilizado por la aplicación para relacionar la PDU de respuesta con la PDU de petición. Si el Dispatcher no pudo preparar el mensaje, el *statusInformation* tendrá un valor de error.

- **processResponsePDU:** Usada por el Dispatcher para enviar una PDU entrante de respuesta a la aplicación adecuada para que sea procesada. La primitiva posee el parámetro de entrada *sendPduHandle* el cual permite identificar a qué pedido corresponde la respuesta entrante.

- **processPDU:** Usada por el Dispatcher para enviar una PDU entrante de solicitud o de notificación a la aplicación pertinente.

- **returnResponsePDU:** Usada por el Contestador de Comandos para enviar una PDU de Respuesta al Dispatcher.

- **registerContextEngineID:** Usada por el Contestador de Comandos para registrar su identidad y poder manipular ciertos tipos de PDU para un contexto SNMP específico.

- **unregisterConextEngineID:** Usada por el Contestador de Comandos para quitar el registro efectuado con la primitiva *registerContextEngineID*.

1.4.2.3.2 Primitivas del Subsistema de Procesamiento de Mensajes.

Estas primitivas definen la comunicación entre el Dispatcher y el Subsistema de Procesamiento de Mensajes. Las primitivas son las siguientes:

- **prepareOutgoingMessage:** Utilizado por el Dispatcher para solicitar al Subsistema de Procesamiento de Mensajes que realice el armado de un mensaje SNMP de salida de pedido o notificación. El Dispatcher proporciona la PDU como uno de los parámetros de entrada, y si la acción es exitosa, recibe un mensaje que incluye la PDU proporcionada como parámetro de salida.

- **prepareDataElements:** Utilizado por el Dispatcher para solicitar al Subsistema de Procesamiento de Mensajes que realice la extracción de la PDU de un mensaje SNMP entrante. El Dispatcher incluye al mensaje entrante como parámetro de entrada y recibe la PDU como parámetro de salida.

- **prepareResponseMessage:** Utilizado por el Dispatcher para solicitar al Subsistema de Procesamiento de Mensajes que realice el armado de un mensaje SNMP de salida de respuesta.

1.4.2.3.3 Primitivas del Subsistema de Seguridad.

Estas primitivas definen la comunicación entre el Subsistema de Procesamiento de Mensajes y el Subsistema de Seguridad. Las primitivas son las siguientes:

- **generateRequestMessage:** Usada por el Subsistema de Procesamiento de Mensajes para solicitar al Subsistema de Seguridad la generación de las seguridades (encriptación y/o autenticación) de un mensaje saliente de pedido o notificación. El Subsistema de Procesamiento de Mensajes proporciona el encabezado y una *scopedPDU* como parámetros de entrada, y recibe un mensaje seguro (encriptado y/o autenticado) y parámetros de seguridad asociados como parámetros de salida.

- **processIncomingMessage:** Usado por el Subsistema de Procesamiento de Mensajes para solicitar al Subsistema de Seguridad el procesamiento del descifrado y/o autenticación de un mensaje entrante. El Subsistema de Procesamiento de Mensajes incluye al mensaje entrante como parámetro de entrada, y recibe una PDU y el parámetro *securityStateReference* como algunos de los parámetros de salida. El parámetro *securityStateReference* contiene información de estado del mensaje que se usará en una posible respuesta.

- **generateResponseMessage:** Usada por el Subsistema de Procesamiento de Mensajes para solicitar al Subsistema de Seguridad la generación de las seguridades (encriptación y/o autenticación) de un mensaje saliente de respuesta. Los parámetros de entrada y de salida son los mismos que se incluyen en la primitiva *generateRequestMessage* pero además se agrega el

parámetro de entrada *securityStateReference,* el cual ya se presentó en la primitiva *processIncomingMessage.*

1.4.2.3.4 Primitivas del Subsistema de Control de Acceso.

Existe una única primitiva, **isAccessAllowed**, la cual define la comunicación entre el Subsistema de Control de Acceso y las Aplicaciones SNMP. Esta primitiva es usada por las Aplicaciones SNMP para solicitar al Subsistema de Control de Acceso la verificación del permiso de acceso a la implementación de la MIB. La primitiva devuelve un valor primitivo *statusInformation* el cual indica si se permite el acceso.

1.4.2.4 Convenciones Textuales para Arquitectura SNMP [12]

Las Conveniones Textuales están definidas en el RFC 3411[23]. Las **Convenciones Textuales** son tipos de datos Definidos que tienen significado más preciso para representar los elementos pertenecientes a una entidad SNMP. Estos objetos son:

- **snmpEngineID**

ESTADO: current.

SINTAXIS: OCTET STRING (SIZE (1...32)).

DESCRIPCIÓN: Es un identificador único de un motor SNMP. El valor para este objeto no puede ser todos ceros (00H), ni todos unos (FFH), ni cadena vacía.
El valor inicial para este objeto puede ser configurado mediante consola de administración o a través de una función algorítmica. La función algorítmica consiste en lo siguiente:

1. El primer bit indica la composición del resto de datos:
    a. 0 Significa que el resto de datos han sido definidos mediante métodos anteriores a SNMPv3. (Ver paso 2.)

[23] RFC 3411: Arquitecture for SNMP Management Frameworks. RFC publicado en diciembre del 2002.

   b. 1 Significa que el resto de datos han sido definidos mediante la arquitectura vigente en SNMPv3. (Ver paso 3).

2. El snmpEngineID tiene una longitud de 12 octetos. Los cuatro primeros octetos representan al número de identificación del agente de una determinada empresa privada. Los ocho octetos restantes son definidos por métodos específicos de dicha empresa.

3. El snmpEngineID tiene una longitud variable. Los cuatro primetos octetos tienen la misma función indicada en el paso anterior. El quinto octeto indica el formato de los octetos restantes. Los posibles valores para el quinto octeto son:
   a. 0 Reservado.
   b. 1 Dirección IPv4 (4 octetos). Indica que los siguientes 4 octetos deberán representar una dirección IPv4.
   c. 2 Dirección IPv6 (16 octetos). Indica que los siguientes 16 octetos deberán representar una dirección IPv6.
   d. 3 Dirección MAC (6 octetos). Indica que los siguientes 6 octetos representan a una dirección física.
   e. 4 Textual, asignados administrativamente. La palabra textual se traduce a octetos. La longitud máxima es 27 octetos.
   f. 5 Octetos, asignados administrativamente. La longitud máxima 27 octetos.
   g. 6 – 127 Reservado.
   h. 127 -255 Definidos por la empresa. Longitud máxima 27 octetos.

Ejemplo: 80:00:13:70:01:c0:a8:00:64

El primer octeto hexadecimal (80) transformado a binario sería: 1000 0000; aquí se puede notar que el primer bit es 1 y por lo tanto significa que el resto de datos han sido definidos mediante la arquitectura vigente en SNMPv3.

Luego, los cuatro primeros octetos hexadecimales (80:00:13:70) equivalen en decimal a 4976 (sin contar el primer bit del primer octeto el cual ya fue utilizado en el paso anterior). Este número es el identificador del agente de la empresa privada *AGENT++* de *Frank Fock*. Estos números son asignados por IANA[24].

El quinto octeto hexadecimal (01) equivale en decimal a 1 e indica que los octetos restantes representan a la dirección IPv4 del host que alberga a la entidad SNMP. Entonces, los cuatro últimos octetos hexadecimales (c0:a8:00:64) representan en decimal a la dirección IP 192.168.0.100.

- **snmpSecurityModel**

ESTADO: current.

SINTAXIS: INTEGER (0...2147483647).

DESCRIPCIÓN: Es un identificador único de un modelo de seguridad (securityModel). La identificación es a nivel de Subsistema de Seguridad (porque pueden existir varios modelos de seguridad dentro de un mismo subsistema).

Los posibles valores que pueden ser asignados a este objeto son:

a. 0 Reservado.
b. 1-255 Modelos estándar de seguridad gestionados por IANA.
c. >255 Modelos propietarios de seguridad de empresas particulares. En este caso el snmpSecurityModel se define del siguiente modo:

*EnterpriseID*256 + modelo de seguridad dentro de la empresa*.

[24] IANA (Internet Assigned Numbers Authority – Autoridad de Asignación de Números de Internet). Es el organismo que se encarga de la coordinación a nivel mundial del direccionamiento IP, asignación de identificadores dentro del grupo MIB *enteprises*, asignación de servidores DNS de primer nivel y de otros recursos del protocolo IP.

Ejemplo:

El cuarto modelo de seguridad definido por la empresa *NxNetworks* cuyo identificador[25] de empresa es 1 sería representado por el número 259.

El esquema de designación de valores al snmpSecuirtyModel define como máximo 255 tipos de modelos estándar de seguridad y 255 tipos de modelos propietarios de seguridad por empresa.

Dentro de los modelos estándar de seguridad existen algunos que se usan con SNMP o tienen soporte de definición de objetos MIB, éstos son: 0 (reservado para *any*), 1 (reservado para SNMPv1), 2 (reservado para SNMPv2c) y 3 (USM).

- **snmpMessageProcessingModel**

ESTADO: current.

SINTAXIS: INTEGER (0...2147483647).

DESCRIPCIÓN: Es un identificador único de un modelo de procesamiento de mensajes. La identificación es a nivel de Subsistema de Procesamiento de Mensajes.

Los posibles valores que pueden ser asignados a este objeto son:

a. 0-255 Modelos estándar de procesamiento de mensajes gestionados por IANA.
b. >255 Modelos propietarios de procesamiento de mensajes de empresas particulares. En este caso el snmpMessageProcessingModel se define del siguiente modo:

[25] La lista de identificadores para empresas está publicada en la siguiente dirección: http://www.iana.org/assignments/enterprise-numbers

*EnterpriseID*256 + modelo de procesamiento de mensajes dentro de la empresa*.

Ejemplo:
El tercer modelo de procesamiento de mensajes definido por la empresa *NxNetworks* cuyo identificador de empresa es 1 sería representado por el número 258.

El esquema de designación de valores al snmpMessageProcessingModel define como máximo 255 tipos de modelos estándar de procesamiento de mensajes y 255 tipos de modelos propietarios de procesamiento de mensajes por empresa.

Dentro de los modelos estándar de procesamiento de mensajes existen algunos que se usan con SNMP, éstos son: 0 (reservado para SNMPv1), 1 (reservado para SNMPv2c), 2 (reservado para SNMPv2u y SNMPv2) y 3 (reservado para SNMPv3).

- **snmpSecurityLevel**

ESTADO: current.

SINTAXIS: INTEGER {noAuthNoPriv (1), authNoPriv (2), authPriv (3)}

DESCRIPCIÓN: Indica el nivel de seguridad con el cual se envían mensajes SNMP o con el cual las operaciones están siendo procesadas. La jerarquía de niveles de seguridad desde la más alta a la más baja es la siguiente: authPriv, authNoPriv y noAuthNoPriv.

- **snmpAdminString.**

DISPLAY-HINT[26]: "255a".

[26] DISPLAY-HINT: Es una cláusula que indica un ejemplo como debería mostrarse el valor de una instancia de un objeto con la sintaxis especificada para el mismo.

ESTADO: current.

SINTAXIS: OCTET STRING (SIZE (0...255)).

DESCRIPCIÓN: Es un objeto que contiene información administrativa en formato legible para el ser humano.

1.4.2.5 MIB para Arquitectura SNMP [12]

La MIB para Arquitectura SNMP está definida en el RFC 3411 y se denomina **snmpFrameworkMIB** cuyo objeto identificador es: *iso(1).org(3).dod(6).internet(1).snmpv2(6).snmpModules(3).snmpFrameworkMIB(10)*. Esta MIB contiene objetos que sirven para definir la arquitectura SNMPv3. Se deriva principalmente en los siguientes objetos:

1.4.2.5.1 Grupo snmpEngine.

El grupo MIB **snmpEngine** cuyo objeto identificador es*: snmpFrameworkMIB(10).snmpFrameworkMIBObjects(2).snmpEngine(1),* posee objetos que contienen información de: identificación del motor, inicio/reinicio del motor y tamaño máximo aceptado del mensaje. Estos objetos son:

- **snmpEngineID.**
  OID: s*nmpFrameworkMIB(10).snmpFrameworkMIBObjects(2).snmpEngine(1). snmpEngineID(1).*

  SINTAXIS: SnmpEngineID.

  ACCESO: read-only.

  ESTADO: current.

DESCRIPCIÓN: Es un identificador único de un motor SNMP.

- **snmpEngineBoots:**
  OID: s*nmpFrameworkMIB(10).snmpFrameworkMIBObjects(2).snmpEngine(1). snmpEngineBoots(2).*

  SINTAXIS: INTEGER.

  ACCESO: read-only.

  ESTADO: current.

  DESCRIPCIÓN: Este objeto indica las veces que el motor se ha inicializado o reinicializado desde la última configuración del *snmpEngineID*.

- **snmpEngineTime:**
  OID: s*nmpFrameworkMIB(10).snmpFrameworkMIBObjects(2).snmpEngine(1). snmpEngineTime(3).*

  SINTAXIS: INTEGER.

  ACCESO: read-only.

  ESTADO: current.

  DESCRIPCIÓN: Indica el tiempo transcurrido en segundos desde el último valor de *snmpEngineBoots* hasta un nuevo cambio.

- **snmpEngineMaxMessageSize:**

OID: s*nmpFrameworkMIB(10).snmpFrameworkMIBObjects(2).snmpEngine(1). snmpEngineMaxMessageSize(4).*
SINTAXIS: INTEGER.

ACCESO: read-only.

ESTADO: current.

DESCRIPCIÓN: Indica el máximo tamaño en octetos del mensaje SNMPv3 que este motor puede enviarlo o recibirlo y procesarlo.

1.4.2.6 MIBs para aplicaciones SNMPv3 [8][13]

1.4.2.6.1 Management Target MIB

La MIB **snmpTargetMIB**, definida en el RFC3413[27] y cuyo OID es: *iso(1).org(3).dod(6).internet(1).snmpv2(6).snmpModules(3).snmpTargetMIB(12),* contiene objetos para la configuración remota de parámetros usados por una entidad SNMP (que tenga originadores de notificaciones o reenviadores proxy) para la generación y envío de mensajes hacia otra entidad SNMP (que tenga receptores de notificaciones y/o reenviadores proxy), es decir desde la entidad agente hacia la entidad gestor SNMP. Dichos parámetros se relacionan a dos clases de información:

- **Información de destino**. Esta información consiste del dominio de transporte y de la dirección de transporte.

- **Parámetros de mensaje SNMP.** Esta información consiste del modelo del procesamiento de mensajes, modelo de seguridad, nivel de seguridad y el nombre de seguridad.

[27] RFC 3413: Simple Network Management Protocol (SNMP) Applications. RFC publicado en diciembre del 2002.

Los principales objetos de esta MIB son:

- **snmpTargetAddrTable.**

OID: *snmpTargetMIB(12).snmpTargetObjects(1).snmpTargetAddrTable(2)*

SINTAXIS: SECUENCE OF.

ACCESO: not-accessible.

ESTADO: current.

DESCRIPCIÓN: Es una tabla de direcciones de transporte a ser usados en la generación de mensajes SNMP.

- **snmpTargetParamsTable.**

OID: *snmpTargetMIB(12).snmpTargetObjects(1).snmpTargetParamsTable (3)*

SINTAXIS: SECUENCE OF.

ACCESO: not-accessible.

ESTADO: current.

DESCRIPCIÓN: Es una tabla de información de gestión a ser usada en la generación de mensajes SNMP.

1.4.2.6.2 Notification MIB

La MIB **snmpNotificationMIB,** definida en el RFC3413 y cuyo OID es: *iso(1).org(3).dod(6).internet(1).snmpv2(6).snmpModules(3).snmpNotificationMIB(13),* contiene objetos para la configuración remota de los parámetros usados por una

entidad SNMP para la generación de notificaciones. Este grupo MIB consiste de los siguientes objetos:

- **snmpNotifyTable.**

OID: *snmpNotificationMIB(13).snmpNotifyObjects(1).snmpNotifyTable(1)*

SINTAXIS: SECUENCE OF.

ACCESO: not-accessible.

ESTADO: current.

DESCRIPCIÓN: Cada entrada de esta tabla selecciona una o más entradas en snmpTargetAddrTable para ser usadas en la generación de notificaciones.

- **snmpNotifyFilterProfileTable.**

OID: *snmpNotificationMIB(13).snmpNotifyObjects(1).snmpNotifyFilterProfileTable(2)*

SINTAXIS: SECUENCE OF.

ACCESO: not-accessible.

ESTADO: current.

DESCRIPCIÓN: Esta tabla es usada para asociar un conjunto de filtros de notificaciones con un destino de gestión particular. Un destino de gestión puede ser una entidad gestor SNMP.

- **snmpNotifyFilterTable.**

OID: *snmpNotificationMIB(13).snmpNotifyObjects(1).snmpNotifyFilterTable(3)*

SINTAXIS: SECUENCE OF.

ACCESO: not-accessible.
ESTADO: current.

DESCRIPCIÓN: Esta tabla que contiene filtros usados para limitar el número de notificaciones generados para un destino de gestión particular.

1.4.2.6.3 Proxy MIB

La MIB **snmpProxyMIB,** definida en el RFC3413 y cuyo OID es: *iso(1).org(3).dod(6).internet(1).snmpv2(6).snmpModules(3).snmpProxyMIB(14),* contiene objetos para la configuración remota de los parámetros usados por una entidad SNMP para las operaciones de reenvío de mensajes SNMP.

1.4.3 PROCESAMIENTO Y ENVÍO DE MENSAJES SNMP [14]

En el RFC 3412 (*Despacho y procesamiento de mensajes SNMPv3*) se define, entre otras cosas, el formato de un mensaje SNMPv3, y los procedimientos para la generación y procesamiento de mensajes SNMP.

Los procedimientos que se siguen para generar mensajes salientes o para procesar mensajes entrantes están definidos en los RFCs 3413 y 3412. Estos procedimientos incluyen interacciones entre: aplicaciones SNMP y el Dispatcher, e interacciones entre los diferentes subsistemas dentro de un motor SNMP; todos expresados en términos de primitivas.

1.4.3.1 Formato del mensaje SNMPv3 [8][11]

La figura 1.20 se muestra la estructura del mensaje SNMPv3; los campos sombreados con color verde son generados y procesados por el Subsistema de Procesamiento de Mensajes.

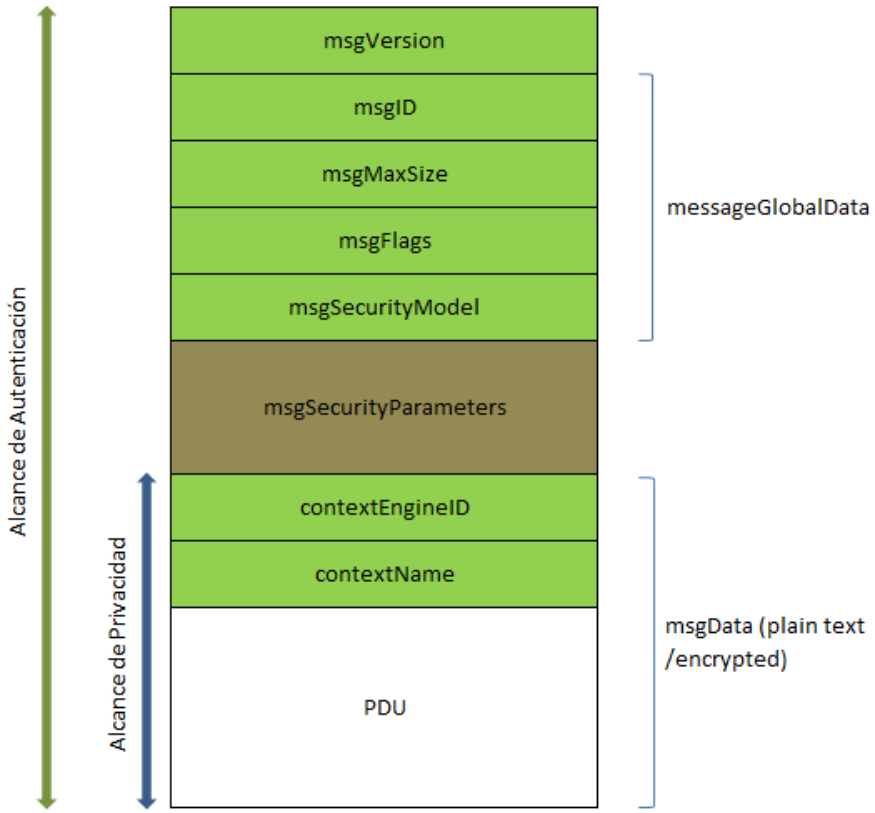


Figura 1. 20 Estructura del mensaje SNMPv3.[28]

El mensaje incluye los siguientes campos:

- **msgVersion:** Está seteado a 3 (SNMPv3).
- **messageGlobalData:** Es un bloque que contiene información útil para efectuar el procesamiento de mensajes. Este bloque está conformado por los siguientes campos:
  - **msgID:** Es un identificador que permite relacionar los mensajes de petición con los de respuesta. Su rango va desde 0 a $[2^{31} - 1]$.
  - **msgMaxSize:** Este campo expresa en octetos el máximo tamaño del mensaje que puede recibir la entidad SNMP que emitió el mensaje actual. Su rango va desde 484 a $[2^{31} - 1]$.

[28] STALLINGS William; SNMP, SNMPv2, SNMPv3 and RMON 1 and 2, Tercera Edición, Figura 16.1 tomada de la pág.490, Acceso: 30 de enero 2010, 15:00.

- **msgFlags:** Es un octeto que contiene tres banderas en los bits menos significativos. Las banderas son: **reportableFlag, privFlag** y **authFlag**. La bandera *reportableFlag* sirve para determinar cuándo enviar (1) ó no (0) una PDU *Report* a la entidad emisora del mensaje, esta bandera debe estar seteada a 1 en las PDUs *Get, Set e Inform,* y debe estar seteada a 0 en las PDUs *Response, Trap y Report.* Las banderas *privFlag* y *authFlag* sirven para indicar el nivel de seguridad aplicado al mensaje; se tienen las siguientes combinaciones en el orden (*privFlag* | *authFlag*): 00 (*noAuthNoPriv*), 01 (authNoPriv), 10 (Reservado) y 11 (*authPriv*).

- **msgSecurityModel:** Sirve para indicar qué tipo de modelo de seguridad fue usado por el remitente del mensaje y por lo tanto qué modelo de seguridad debe ser usado por el receptor para procesar el mensaje. Su rango va desde 0 a $[2^{31} - 1]$ pero existen valores reservados: 0 (cualquiera), 1 (SNMPv1), 2 (SNMPv2c) y 3 (USM).

- **msgSecurityParameters:** Es un bloque que contiene campos definidos y generados por el Subsistema de Seguridad.

- **msgData:** Este bloque contiene información útil para la aplicación SNMP al efectuar el procesamiento de una PDU. Este bloque tiene dos tipos de representación: *plaintext* (denominado como *scopedPDU*) o *encryptedPDU* (es el valor *scopedPDU* pero encriptado).

- **scopedPDU:** Este bloque contiene los siguientes campos:
  - **contextEngineID:** En los mensajes entrantes, este campo indica a qué aplicación se entregará la PDU. En los mensajes salientes, este campo es establecido al valor proveído por la aplicación cuando ésta emite una solicitud para enviar un mensaje.

- **contextName:** Es un octeto que identifica a un contexto SNMP que tiene relación con la información de administración contenida en la PDU del mensaje.

- **data:** Es una PDU. El modelo de procesamiento de mensajes SNMPv3 (v3MP) especifica que la PDU debe ser alguna de las definidas en el RFC 3416 (Operaciones de SNMPv2).

1.4.3.2 Procesamiento de un Mensaje por el Generador de Comandos u Originador de Notificaciones [8][11]

1.4.3.2.1 Preparación de un mensaje saliente de pedido

1) El Generador de Comandos usa la primitiva *sendPdu* para enviar una PDU saliente de pedido al Dispatcher e incluye los siguientes parámetros de entrada: destino deseado (*transportDomain* y *transportAddress*), parámetros de seguridad (*securityName, securityModel y securityLevel*), el *contextEngineID,* el *contextName* y la PDU. El Dispatcher asigna un valor único al parámetro *sendPduHandle* para esta PDU. El parámetro *transportDomain* representa al protocolo de transporte y de red usados, por ejemplo UDP/IPv4. El parámetro *transportAddress* representa la dirección de transporte usada, por ejemplo una dirección IP y un puerto UDP.

2) El Dispatcher revisa la versión de la PDU y determina qué modelo de procesamiento de mensajes (v1MP, v2cMP, v3MP u otro) debe ser elegido.

3) El Dispatcher pide al Subsistema de Procesamiento de Mensajes la preparación de un mensaje de salida a través de la primitiva *prepareOutgoingMessage*. El Dispatcher incluye en la primitiva los parámetros de entrada dados por el Generador de Comandos y el *sendPduHandle* que él estableció para dicha PDU.

4) El Subsistema de Procesamiento de Mensajes prepara el mensaje de salida que se solicitó; primero genera una *scopedPDU* y luego prepara el *msgGlobalData* (asigna un valor único al campo *msgID* y setea las banderas *msgFlags* según el valor del parámetro *securityLevel* proporcionado por el Generador de Comandos).

5) El Subsistema de Procesamiento de Mensajes pide al Subsistema de Seguridad la generación de las seguridades del mensaje ya armado a través de la primitiva *generateRequestMessage.* En esta primitiva se incluyen el mensaje preparado previamente como parámetro de entrada.

6) El Subsistema de Seguridad llena los campos correspondientes al bloque *msgSecurityParameters* y devuelve el mensaje generado al Subsistema de Procesamiento de Mensajes.

7) El Subsistema de Procesamiento de Mensajes almacena los valores de los parámetros (*sendPduHandle, securityModel, securityName, securityLevel, contextEngineID* y *contextName*) provenientes de la primitiva *prepareOutgoingMessage* y el *msgID,* estos valores se usarán para validar los mensajes de respuesta que posteriormente vendrán.

8) El Subsistema de Procesamiento de Mensajes retorna al Dispatcher el mensaje preparado como parámetro de salida de la primitiva *prepareOutgoingMessage.*

9) El Dispatcher envía al Generador de Comandos el *sendPduHandle* como valor primitivo de la primitiva *sendPdu*. El Generador de Comandos guarda dicho valor para poder relacionar el mensaje de respuesta a recibirse con el mensaje de petición enviado.

10) El Dispatcher entrega el mensaje preparado a la capa transporte para que sea enviado a través de la red con destino especificado en *transportDomain* y *transportAddress.*

1.4.3.2.2 Recepción y procesamiento de un mensaje entrante de respuesta.

1) El mensaje de respuesta, con información de dirección y dominio origen (*transportDomain, transportAddress*), es recibido por el Dispatcher desde la red.

2) El Dispatcher solicita al Subsistema de Procesamiento de Mensajes que realice la extracción de la PDU del mensaje de respuesta entrante a través de la primitiva *prepareDataElements.* En esta primitiva se incluyen los parámetros de entrada: *transportDomain, transportAddress* y el mensaje recibido.

3) El Subsistema de Procesamiento de Mensaje realiza la extracción de los campos del bloque *MessageGlobalData* y el *msgID*. Verifica si el modelo de seguridad (*msgSecurityModel*) es soportado y si las banderas (*authFlag* y *privFlag*) del campo *msgFlags* tienen un combinación válida; de no ser así entonces se descarta el mensaje y se envía una indicación de error.

4) El Subsistema de Procesamiento de Mensajes solicita al Subsistema de Seguridad el procesamiento de seguridad (descifrado y/o autenticación) del mensaje de respuesta entrante mediante la primitiva *processIncomingMessage.* El parámetro de entrada *securityLevel* de esta primitiva es establecido a partir de los valores de las banderas *authFlag* y *privFlag.*

5) El Subsistema de Seguridad procesa el bloque *msgSecurityParameters* y devuelve al Subsistema de Procesamiento de Mensajes la *scopedPdu* y el valor primitivo *statusInformation* de la primitiva *processIncomingMessage.*

6) El Subsistema de Procesamiento de Mensajes usa el *msgID* del mensaje de respuesta recibido para encontrar la información correspondiente al mensaje de pedido relacionado. Luego, se realiza una comparación entre los parámetros del mensaje de respuesta y los parámetros guardados del mensaje de pedido. Los parámetros comparados son: *securityModel, securityName, securityLevel, contextEngine* y *contextName.* Si todos coinciden, el valor guardado *sendPduHandle* se recupera y se pasa como uno de los parámetros de salida (también se incluyen la PDU, el *contextEngine* y el *contextName*) de la primitiva *prepareDataElements* al Dispatcher.

7) El Dispatcher determina qué aplicación está esperando la llegada de una PDU de respuesta asociada con este *sendPduHandle*.

8) El Dispatcher envía la PDU de respuesta al Generador de Comandos apropiado mediante la primitiva *processResponsePdu.* En esta primitiva se incluyen los parámetros de entrada: *sendPduHandle, contextName, contextEngineID* y la *PDU*. La aplicación usa el valor del parámetro *sendPduHandle* para identificar a qué pedido corresponde la PDU entrante.

9) El Generador de Comandos examina los valores de la primitiva *processResponsePdu* y si algunos de los parámetros no concuerda, entonces se descarta la respuesta. Dentro de los parámetros, si el *statusInformation* indica falla en el pedido, se realiza alguna acción dependiente de la aplicación, como por ejemplo reenviar el pedido o notificar al administrador.

10) El Generador de Comandos examina los campos de la PDU de respuesta, especialmente el Identificador de solicitud. Si este valor no corresponde al especificado en el pedido, se descarta la respuesta.

11) Si los dos pasos anteriores son exitosos, se ejecuta una acción dependiente de la aplicación.

Figura 1. 21 Diagrama de funcionamiento para un Generador de Comandos u Originador de Notificaciones.[29]

[29] STALLINGS William; SNMP, SNMPv2, SNMPv3 and RMON 1 and 2, Tercera Edición, Figura 16.3 tomada de la pág.493, Acceso: 10 de febrero 2010, 17:00.

La figura 1.21 muestra el procedimiento que se sigue para generar mensajes salientes de pedido o para procesar mensajes entrantes de respuesta por parte de un Generador de Comandos o un Originador de Notificaciones.

Para el caso de una aplicación Originadora de Notificaciones se sigue el mismo procedimiento usado en una aplicación Generadora de Comandos. Si se tiene que enviar una PDU *informRequest* se utilizan las primitivas *sendPdu* y *processResponsePdu;* en cambio para enviar una PDU *Trap* sólo se usa la primitiva *sendPdu.*

1.4.3.3 Procesamiento de un Mensaje por el Contestador de Comandos o Receptor de Notificaciones [8][11]

1.4.3.3.1 Recepción de un mensaje de pedido.

1) El Contestador de Comandos empieza registrándose contra el Dispatcher usando la primitiva *registerContextEngineID.* El fin de esto es poder asociarse con el motor SNMP y poder manipular ciertas PDUs para un Contexto SNMP. Una vez que la aplicación esté registrada, todos los mensajes recibidos que contengan la combinación registrada de *contextEngineID* y *pduType* (tipo de PDU reconocida y aceptada) se envían al Contestador de Comandos que soporta esa combinación. Si la aplicación desea desasociarse del motor SNMP, utiliza la primitiva *unregisterContextEngineID.*

2) Se secuencian los mismos pasos 1, 2, 3, 4 y 5 utilizados en la recepción de un mensaje de respuesta por parte de un Generador de Comandos.

3) El Subsistema de Procesamiento de Mensajes guarda la siguiente información relacionada al mensaje de pedido entrante: *msgVersion, msgID, msgMaxSize, msgFlags, securityModel, securityLevel, securityName, securityStateReference, contextEngineID, contextName,*

maxResponseSizeScopedPdu (Este parámetro indica el tamaño máximo de la PDU de respuesta), *transportDomain* y *transportAddress.*

4) El Subsistema de Procesamiento de Mensajes envía al Dispatcher los siguientes parámetros de salida incluidos en la primitiva *prepareDataElements*: *stateReference*, *contextEngineID, contextName, sendPduHandle* (sin valor) y la PDU de petición.

5) El Dispatcher define qué aplicación debe recibir esta PDU en base a los valores de los parámetros *contextEngineID* y *pduType*.

6) El Dispatcher envía la PDU de petición al Contestador de Comandos apropiado mediante la primitiva *processPdu.* Los parámetros de entrada que se incluyen en esta primitiva son: *contextEngineID, contextName, stateReference* y parámetros de seguridad.

1.4.3.3.2 Preparación de un mensaje de respuesta saliente

1) El Contestador de Comandos examina el contenido de la PDU de pedido. El tipo de PDU debe concordar con alguno de aquellos tipos registrados previamente por la aplicación.

2) El Contestador de Comandos solicita la verificación del permiso para realizar la operación de gestión solicitada por la PDU al Subsistema de Control de Acceso mediante la primitiva *isAccessAllowed*.

3) El Subsistema de Control de Acceso responderá a la solicitud de la aplicación, indicando si se permite el acceso o no, mediante el valor primitivo (*statusInformation*) devuelto por la primitiva *isAccessAllowed.*

4) Si el acceso está permitido, el Contestador de Comandos realiza la operación solicitada y prepara una PDU de respuesta. Caso contrario, la aplicación prepara una PDU de respuesta que indicará la falla.

5) El Contestador de Comandos envía la PDU de respuesta al Dispatcher mediante la primitiva *returnResponsePdu* incluyendo los parámetros de entrada: *stateReference*, *contextEngineID* y *contextName.*

6) El Dispatcher pide al Subsistema de Procesamiento de Mensajes a través de la primitiva *prepareResponseMessage* realizar el armado de un mensaje SNMP de respuesta de salida. En esta primitiva se incluyen los parámetros de entrada: *stateReference, contextEngineID, contextName* y la PDU proporcionada por la aplicación.

7) El Subsistema de Procesamiento de mensajes solicita al Subsistema de Seguridad a través de la primitiva *generateResponseMessage* generar las seguridades (cifrado y/o autenticación) para el mensaje de respuesta saliente. El Subsistema de Seguridad llena el bloque *msgSecurityParameters* y devuelve el mensaje preparado al Subsistema de Procesamiento de Mensajes.

8) El Subsistema de Procesamiento de Mensajes responde al Dispatcher con el mensaje preparado y con los parámetros de salida *transportDomain* y *transportAddress.*

9) El Dispatcher entrega el mensaje preparado a la capa transporte para que sea enviado a través de la red con destino especificado en *transportDomain* y *transportAddress.*

Figura 1. 22 Diagrama de funcionamiento para un Contestador de Comandos o Receptor de Notificaciones.[30]

[30] STALLINGS William; SNMP, SNMPv2, SNMPv3 and RMON 1 and 2, Tercera Edición, Figura 16.4 tomada de la pág.496, Acceso: 15 de febrero 2010, 19:00.

Para el caso de una aplicación Receptora de Notificaciones se sigue el mismo procedimiento usado en una aplicación Contestadora de Comandos. El Receptor de Notificaciones primero debe registrarse para recibir las PDUs *InformRequest* y/o *Trap*. Ambos tipos de PDUs se reciben mediante la primitiva *proccessPdu.* Para responder a una PDU *InformRequest* se usa la primitiva *returnResponsePdu*.

La figura 1.22 muestra el procedimiento que se sigue para procesar mensajes entrantes de pedido y para generar mensajes salientes de respuesta por parte de un Contestador de Comandos o un Receptor de Notificaciones.

1.4.3.4 MIBs para procesamiento y envío de mensajes SNMP [14]

El grupo MIB **snmpMPDMIB,** definido en el RFC3412[31] y cuyo OID es: *iso(1).org(3).dod(6).internet(1).snmpv2(6).snmpModules(3).snmpMPDMIB(11),* contiene objetos útiles para el procesamiento y envío de mensajes SNMP. Este grupo principalmente se deriva en:

1.4.3.4.1 snmpMPDStats

El grupo **snmpMPDStats**, cuyo OID es: *snmpMPDMIB(11).snmpMPDMIBObjects(2).snmpMPDStats(1),* contiene objetos que almacenan estadísticas relacionadas a los mensajes SNMP. Estos objetos son:

- **snmpUnknownSecurityModels:**
  OID: *snmpMPDMIB(11).snmpMPDMIBObjects(2).snmpMPDStats(1). snmpUnknownSecurityModels(1).*

  SINTAXIS: Counter32.

  ACCESO: read-only.

[31] RFC 3412: Message Processing and Dispatching for the Simple Network Management Protocol. RFC publicado en diciembre del 2002.

ESTADO: current.

DESCRIPCIÓN: Es un contador de mensajes recibidos que fueron descartados debido a que referenciaban un modelo de seguridad (*securityModel*) desconocido o no soportado por el motor SNMP.

- **snmpInvalidMsgs:**
  OID: *snmpMPDMIB(11).snmpMPDMIBObjects(2).snmpMPDStats(1). snmpInvalidMsgs(2).*

  SINTAXIS: Counter32.

  ACCESO: read-only.

  ESTADO: current.

  DESCRIPCIÓN: Es un contador de mensajes recibidos que fueron descartados debido a que presentaban errores en los campos de formato del mensaje.

- **snmpUnknownPDUHandlers:**
  OID: *snmpMPDMIB(11).snmpMPDMIBObjects(2).snmpMPDStats(1). snmpUnknownPDUHandlers(3).*

  SINTAXIS: Counter32.

  ACCESO: read-only.

  ESTADO: current.

DESCRIPCIÓN: Es un contador de mensajes recibidos que fueron descartados debido a que la PDU contenida en el mensaje no pudo ser pasada a una aplicación responsable que maneje de tales tipos de PDU; la razón de esto puede ser de que no existía una aplicación que se haya registrado con una combinación apropiada de *contextEngine* y *pduType*.

1.4.4 MODELO DE SEGURIDAD BASADA EN EL USUARIO (USM) [8][11]

USM está definido en el RFC 3414[32] y describe elementos, procedimientos y MIBs para la provisión de adecuados niveles de seguridad a los mensajes SNMP.

1.4.4.1 Motor SNMP Autorizado

Es uno de los motores que intervienen en el intercambio de mensajes; éste es responsable de la referencia del tiempo (reloj), de los identificadores únicos de motor (*snmpEngineID*) en cada mensaje y de la gestión de localización de claves. Para determinar qué motor es el autorizado, se siguen las siguientes reglas:

- Cuando el mensaje es de pedido, el motor que lo recibió es el autorizado.
- Cuando el mensaje es de respuesta, *Trap* o reporte, el motor que lo envió es el autorizado.

Por lo general, el motor de la entidad Agente es el motor autorizado; y cada motor no autorizado (el motor de la entidad Gestor) debe mantener un tabla que contenga los tiempos e identificadores de los motores autorizados con los cuales se comunica.

1.4.4.2 Formato del bloque msgSecurityParameters

El bloque *msgSecurityParameters* del mensaje SNMPv3 está compuesto por los siguientes campos:

[32] RFC 3414: User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3). RFC publicado en diciembre del 2002.

- **msgAuthoritativeEngineID:** Representa el *snmpEngineID* del motor SNMP autorizado.

- **msgAuthoritativeEngineBoots:** Representa el *snmpEngineBoots* del motor SNMP autorizado.

- **msgAuthoritativeEngineTime:** Representa el *snmpEngineTime* del motor SNMP autorizado.

- **msgUserName:** Especifica el usuario *(Principal*) en nombre de quién se está realizando el intercambio de mensajes de administración.

- **msgAuthenticationParameters:** Son parámetros definidos por el protocolo de autenticación en uso.

- **msgPrivacyParameters:** Son parámetros definidos por el protocolo de privacidad en uso.

1.4.4.3 Autenticación [8] [15][16]

Provee integridad de los datos y autenticación del origen de los datos. La integridad de los datos se refiere a que éstos estén libres de alteraciones o modificaciones inapropiadas. La autenticación se refiere a la confirmación de la identidad de la entidad SNMP de la cual se recibió el mensaje.

La autenticación se logra mediante una función criptográfica de autenticación que requiere de una clave de autenticación (*authKey*). El motor SNMP almacena para cada usuario (local o remoto) una *authKey*. Un usuario local es un principal (asociado a un motor SNMP local) que tiene permisos para realizar operaciones de administración de red. Mientras que, un usuario remoto es un principal (asociado a un motor SNMP remoto) con quien está permitido entablar intercambio de

información. El alcance de la autenticación es de todo el mensaje SNMP. Las funciones de autenticación soportadas son: HMAC-MD5-96 y HMAC-SHA-96.

1.4.4.3.1 Protocolo HMAC-MD5-96

Este protocolo utiliza el código de autenticación HMAC y la función hash[33] MD5. HMAC (*Hash-based Message Authentication Code – Código de Autenticación de Mensaje basado en Hash*) es un mecanismo para autenticar mensajes, que se basa en el uso de una llave secreta y en el uso de funciones hash. La llave secreta sirve para el cálculo y verificación de los valores de autenticación de los mensajes. HMAC es usado entre dos entidades SNMP que comparten una llave secreta con el fin de validar la información intercambiada entre ellas. La función hash MD5 (*Message Digest Algorithm – Algoritmo de Resumen de Mensaje*) sirve para convertir un dato de longitud cualquiera a un dato de longitud pequeña fija de 128 bits (16 octetos).

En síntesis, el protocolo HMAC-MD5-96 trabaja de la siguiente la siguiente manera:

- El mensaje SNMP entero ingresa al algoritmo HMAC.
- HMAC utiliza la llave secreta *authKey* y la función hash MD5 para a partir del mensaje generar un resumen de longitud de 16 octetos que posteriormente se trunca a 12 octetos (96 bits).
- Este valor de 12 octetos es un código de autenticación que se inserta en el campo *msgAuthenticationParameters* del mensaje SNMPv3.

1.4.4.3.2 Protocolo HMAC-SHA-96

Este protocolo también utiliza el código de autenticación HMAC y la función hash SHA-1. La función hash SHA-1 (*Secure-Hash Algorithm - Algoritmo de Hash Seguro*) produce un resumen de mensaje de longitud 160 bits (20 bytes).

---

[33] **Hash** es una función para generar claves que representen de manera casi única a un archivo. http://es.wikipedia.org/wiki/Hash.

MD5 y SHA-1 tienen notables diferencias. MD5 realiza una ejecución más rápida de algoritmo que SHA-1; sin embargo, a pesar de que SHA-1 es más lento, genera un resumen de mensaje de mayor longitud lo cual hace que el algoritmo sea más robusto contra ataques de fuerza bruta. En ambos casos, los dos algoritmos son no reversibles; es decir, no es posible encontrar el mensaje original a partir del resumen y llave secreta.

El protocolo HMAC-SHA-96 trabaja de manera similar como el protocolo anterior, con la diferencia de que genera un resumen de mensaje de longitud de 20 octetos que luego se trunca a 12 octetos.

1.4.4.4 Privacidad [8][16][17][18]

Provee protección contra la revelación de datos del mensaje a entidades no autorizadas. Esto se logra mediante un algoritmo de cifrado que requiere de una clave privada (*privKey*). El motor SNMP almacena para cada usuario (local o remoto) una *privKey*. La porción del mensaje SNMP que requiere encriptación es la *scopedPdu* (ver figura 1.20). Las funciones de encriptación soportadas son: DES y AES.

1.4.4.4.1 Protocolo DES

DES (*Data Encryption Standard – Cifrado Estándar de Datos*) es un algoritmo de encriptación simétrico; es decir, utiliza la misma llave tanto para encriptar como para desencriptar. DES tiene varios modos de operación, de entre ellos CBC (*Cipher Block Chaining – Encadenamiento de Bloques de Cifrado*) es escogido por USM para ser usado. Este algoritmo requiere de una llave secreta de 16 bytes (128 bits). El modo CBC requiere de unos 8 bytes (64 bits) iniciales (denominado vector de inicialización) para comenzar a ejecutar el algoritmo de cifrado.

El mecanismo que emplea USM para la creación de la llave secreta y el vector de inicialización están basados en dos valores ya obtenidos: un valor secreto (*privKey*) y un valor timeliness (*snmpEngineBoots*). El valor secreto se comparte entre un usuario (Principal) y un motor SNMP autorizado.

Los primeros 8 octetos de *privKey* se utilizan como llave secreta para DES. Debido a que DES verdaderamente requiere de 56 bits, el bit menos significativo de cada octeto es ignorado.

El vector de inicialización, denominado como IV (*Initialization Vector – Vector de Inicialización*) se genera de la siguiente manera:

- De la *privKey* se toman los últimos 8 bytes para usarlos como un pre-IV.
- Para asegurar que se utilicen dos IV distintos para diferentes entradas de texto cifrados bajo la misma clave, se crea un octeto *salt value. Salt value* se crea a partir de la concatenación de *snmpEngineBoots* y un valor entero de 64 bits dado por el protocolo de encriptación.
- Se realiza una operación XOR bit a bit entre el pre-IV y el octeto *salt value*, el resultado es el vector de inicialización.

El octeto *salt value* es insertado en el campo *msgPrivacyParameters* del mensaje SNMPv3 con el fin de que la entidad receptora pueda generar el vector de inicialización adecuado y pueda descifrar el mensaje.

1.4.4.4.2 Protocolo AES

AES (*Advanced Encryption Standard – Cifrado Estándar Avanzado*) es un algoritmo de encriptación simétrico diseñado con características de: resistencia mejorada contra ataques, fácil implementación, eficiencia y diseño escalable. En el RFC 3826 se define el funcionamiento de AES en USM. El modo de funcionamiento seleccionado es CFM (*Cipher Feedback Mode – Modo de Cifrado Retroalimentado*).

Este algoritmo soporta longitudes de llave de 128, 192 y 256 bits, pero la de 128 bits es la escogida para USM.

De la misma forma que en DES, se emplea un mecanismo similar para generar la llave secreta y el vector de inicialización. Para generar la llave secreta se toman los primeros 128 bits de *privKey*. Mientras que, para generar el vector de inicialización de 128 bits se realiza una concatenación de *snmpEngineBoots*, *snmpEngineTime* (pertenecientes al motor SNMP autorizado) y de un entero local de 64 bits. El entero local es inicializado a un valor aleatorio al reiniciarse el motor SNMP. Este entero local es insertado en el campo *msgPrivacyParameters* del mensaje SNMP con el fin de que la entidad receptora pueda generar el vector de inicialización y pueda desencriptar el mensaje.

1.4.4.5 Puntualidad (Timeliness) [8][11]

Brinda protección contra la recepción de mensajes que lleguen tarde y contra el reenvío y redireccionamiento no autorizados de mensajes. Para emplear el módulo *Timeliness* es indispensable haber aplicado antes un servicio de autenticación al mensaje. Los mecanismos de puntualidad son: administración de relojes autorizados, sincronización y verificación de puntualidad.

1.4.4.5.1 Administración de relojes autorizados.

Los motores SNMP autorizados deben administrar los objetos *snmpEngineBoots* y *snmpEngineTime* los cuales indican su tiempo local. La gestión de relojes consiste en lo siguiente:

- Si *snmpEngineTime* alcanza su máximo valor, se produce un incremento en el valor del *snmpEngineBoots* y comienza nuevamente desde 0.
- Si un motor autorizado no logra identificar su último valor del *snmpEngineBoots*, lo setea al valor umbral, es decir $[2^{31} - 1]$.

- Si el *snmpEngineBoots* alcanza su máximo valor, éste se bloquea y se produce un error de autenticación. Por lo tanto, se requiere una reconfiguración manual del sistema que consiste en cambiar el *snmpEngineID* o las claves de los protocolos de autenticación y privacidad de todos los usuarios conocidos por este motor.

1.4.4.5.2 Sincronización.

Para que un motor no autorizado pueda sincronizarse con los motores autorizados, almacena copias de las siguientes variables que pertenecen a cada motor autorizado: *snmpEngineBoots*, *snmpEngineTime* y *latestReceivedEngineTime.* La variable *latestReceivedEngineTime* representa el valor más alto de *msgAuthoritativeEngineTime* del motor autorizado.

Para que un motor no autorizado reciba por primera vez las variables de sincronización se sigue un mecanismo de Descubrimiento[34] de motores autorizados. Luego, por cada mensaje SNMP recibido, el motor no autorizado lee los valores y actualiza las variables.

Para proceder a realizar la actualización de las variables, previamente se realiza comparaciones entre las variables guardadas localmente (*snmpEngineBoots*, *snmpEngineTime*, y *latestReceivedEngineTime*) contra las variables recibidas en los mensajes SNMP (*msgAuthoritativeEngineBoots, msgAuthoritativeEngineTime, y msgAuthoritativeEngineTime*)[35], respectivamente. Si el valor de la variable recibida es mayor o igual a la almacenada, entonces se realiza la actualización; caso contrario, no.

La figura 1.23 muestra un diagrama de flujo donde se resume el proceso de sincronización de un motor no autorizado.

---

[34] El mecanismo de Descubrimiento de motores autorizados se lo trata más adelante.
[35] Estos campos pertencen al bloque msgSecurityParameters del mensaje SNMPv3. Ver ítem 1.4.4.2.

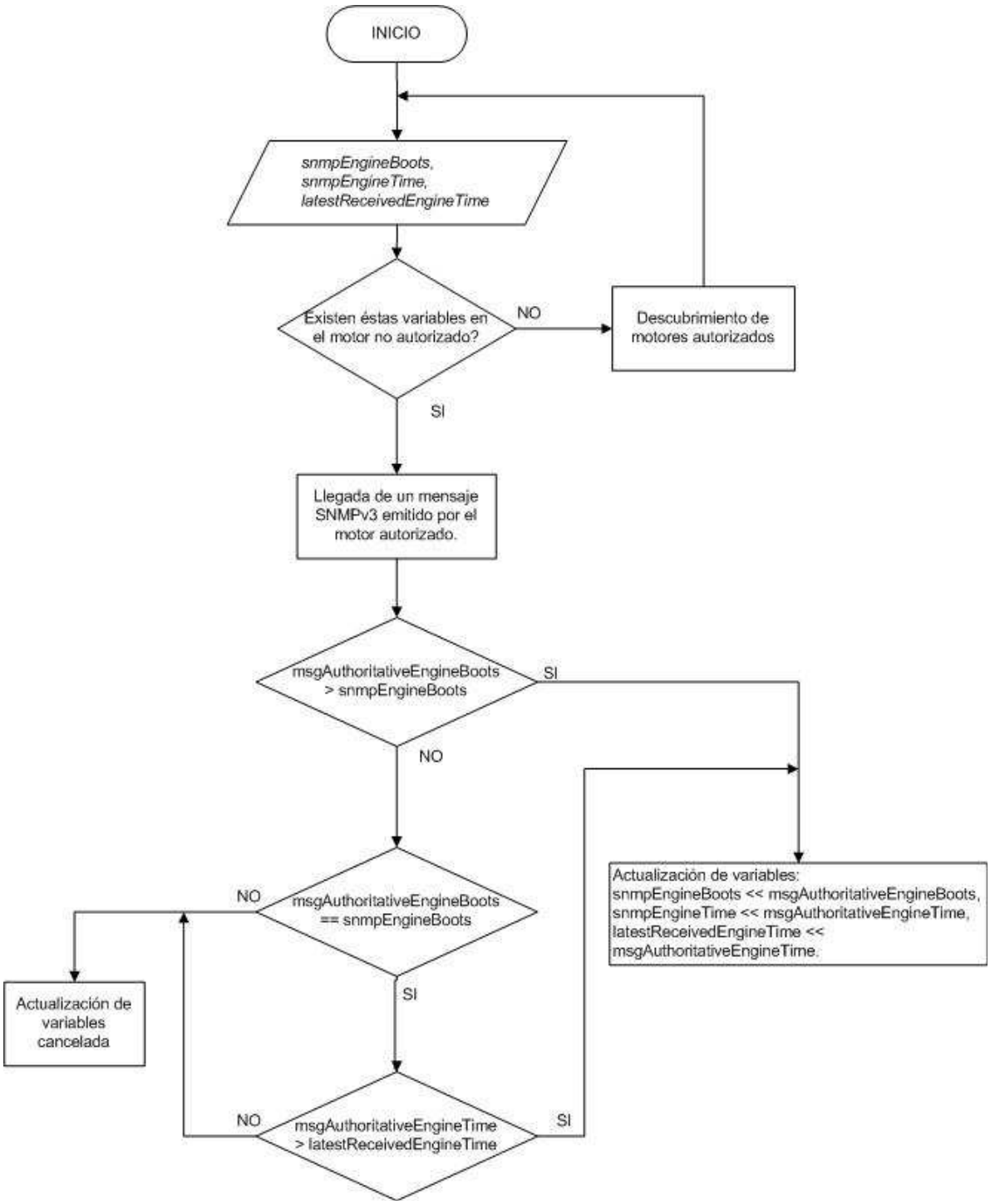


Figura 1. 23 Diagrama de flujo para el proceso de sincronización.

1.4.4.5.3 Verificación de puntualidad.

Para chequear la puntualidad de un mensaje se utiliza el concepto de ventana de tiempo. Una ventana es un intervalo de tiempo en el cual si un mensaje llega dentro de dicho intervalo, el mensaje es considerado auténtico.

Si el receptor del mensaje es un motor autorizado, considera que el mensaje llegó fuera de la ventana de tiempo si se cumple alguna de las siguientes condiciones:

- El *snmpEngineBoots* está trabado en su máximo valor, es decir [2^{31}-1].
- EL *msgAuthoritativeEngineBoots* y *snmpEngineBoots* no tienen los mismos valores.
- El valor del campo *msgAuthoritativeEngineTime* difiere de *snmpEngineTime* por más de 150 segundos.[36]

Si el receptor del mensaje es un motor no autorizado, considera que el mensaje llegó fuera de la ventana de tiempo si se cumple alguna de las siguientes condiciones:

- El *snmpEngineBoots* está trabado en su máximo valor.
- EL *msgAuthoritativeEngineBoots* tiene un valor menor al *snmpEngineBoots*.
- El valor de *msgAuthoritativeEngineTime* difiere de *snmpEngineTime* por más de 150 segundos.

Si el mensaje es considerado fuera de ventana de tiempo, se lo denota como no auténtico y se devuelve un error (*notInTimeWindow*) al subsistema solicitante.

1.4.4.6 Descubrimiento [8][11]

Especifica mecanismos para entablar conversaciones entre motores SNMP. Los motores no autorizados previamente deben conocer el *snmpEngineID* de los motores autorizados para luego proceder con la conversación. Los mecanismos de descubrimiento pueden ser los siguientes:

[36] LESCHENNE, Sebastián; SALAZAR, Martín, Universidad Nacional de Rosario, Modelo de Gestión Internet SNMP-RMON, pág.82. Acceso: 2 de marzo 2010, 17:00

- El motor no autorizado envía un mensaje *GetRequest* al motor autorizado. Este mensaje incluye el valor *noAuthNoPriv* en el parámetro *securityLevel,* las banderas msgFlags seteadas a *Reportable,* un valor nulo en *msgUserName* y un valor nulo en *msgAuthoritativeEngineID.* Luego, el motor autorizado responderá con un mensaje *Report* el cual incluye su valor de *snmpEngineID* en el campo *msgAuthoritativeEngineID*.
- El motor no autorizado ya tiene conocimiento del motor autorizado pero además requiere sincronizarse con él. El motor autorizado envía un mensaje *GetRequest* que incluye el valor de *snmpEngineID* en el campo *msgAuthoritativeEngineID*, y valores nulos en *msgAuthoritativeEngineBoots* y *msgAuthoritativeEngineTime*. Luego, el motor autorizado responderá con un mensaje *Report* el cual incluye sus valores de *snmpEngineBoots* y *snmpEngineTime*.

1.4.4.7 Gestión de claves [8][11]

La gestión de claves define procedimientos para la generación, localización y actualización de las mismas. El principal[37] es el que se encarga de la gestión de las claves. Cada principal debe mantener una única clave de autenticación y una de privacidad.

1.4.4.7.1 Generación de claves

La generación de claves se realiza mediante el algoritmo *Password-Key*. Este algoritmo requiere de uno o dos *passwords,* dados desde el sistema de gestión, para crear dos claves de 16 o 20 octetos. Esto depende si se desea utilizar el mismo password para generar las claves de autenticación y encriptación, o utilizar un password diferente para cada clave. El procedimiento es el siguiente:

- Se forma una cadena de texto de longitud 2^{20} repitiendo el password. Esta cadena de texto se denomina *digest0*.

[37] **Principal:** Es la entidad en nombre de quién se procesan los servicios. Generalmente, un principal es un gestor SNMP. Ver ítem 1.4.2.2.

- Para crear una llave de 16 octetos se aplica la función MD5 al *digest0* y se obtiene la clave de usuario (también se denomina *digest1*). En cambio, para crear una llave de 20 octetos se aplica la función SHA-1.

1.4.4.7.2 Localización de claves

La localización de claves es el proceso de generación de varias claves diferentes a partir de la clave de usuario previamente obtenida (*digest1*). Cada clave producida se denomina *clave localizada*. La clave localizada es la que verdaderamente se comparte entre un principal en un motor no autorizado y un motor autorizado (agente). La clave localizada puede ser *authKey* (para el caso de autenticación) o *privKey* (para el caso de privacidad). El procedimiento de localización de claves es el siguiente:

- Se forma una cadena de texto con: *digest1,* el *snmpEngineID* del motor autorizado y nuevamente *digest1.* Esta cadena de texto se le denomina *digest2*.
- Se aplica las funciones MD5 o SHA-1 sobre *digest2*. El resultado es la clave localizada.

1.4.4.7.3 Actualización de claves

El fin de la actualización de claves es mantener la seguridad y confiabilidad del sistema. El cambio de claves se realiza desde el sistema de gestión. En primer lugar se cambia el *password* para generar una nueva clave de usuario. Luego se recalcula las claves localizadas por cada agente con el que se mantiene comunicación y finalmente se comunica a cada agente del cambio para que la actualicen.

Para efectuar las actualizaciones se utiliza el objeto MIB *KeyChange.* El sistema gestor, al momento de realizar la actualización, modifica este objeto para que luego sea utilizado por los agentes y actualicen la clave.

1.4.4.8 Convenciones Textuales para USM [19]

- **KeyChange**

ESTADO: current.

SINTAXIS: OCTET STRING

DESCRIPCIÓN: Es un objeto que se utiliza para el cambio de llaves de los protocolos de autenticación y privacidad.

El valor de este objeto se obtiene utilizando un protocolo (HMAC-MD5-96 o HMAC-SHA-96), una llave secreta, y un algoritmo hash (MD5 o SHA-1) que producen una salida de determinada longitud de octetos (*KeyChange*). Dicho valor de salida es el que se envía hacia una entidad agente para efectuar el cambio de passwords.

1.4.4.9 MIBs para USM [19]

El grupo MIB **snmpUsmMIB,** definido en el RFC3414[38] y cuyo OID es: *iso(1).org(3).dod(6).internet(1).snmpv2(6).snmpModules(3).snmpUsmMIB(15),* contiene todos los objetos empleados por USM para procesar las seguridades de un mensaje entrante o saliente. Este grupo principalmente se divide en los siguientes objetos:

1.4.4.9.1 snmpAuthProtocols

Este grupo es importado desde la MIB *snmpFrameworkMIB.*[39] Se emplea para la identificación de los protocolos de autenticación. Su objeto identificador es:

[38] RFC 3414: User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3). RFC publicado en diciembre de 2002.

[39] La MIB snmpFrameworkMIB es tratada en el literal **1.4.2.5 "**MIB para Arquitectura SNMP**".**

iso(1).org(3).dod(6).internet(1).snmpV2(6).snmpModules(3).snmpFrameworkMIB(10). snmpFrameworkAdmin(1).snmpAuthProtocols(1). Se divide en:

- **usmNoAuthProtocol:**
  OID: *snmpAuthProtocols(1).usmNoAuthProtocol(1).*

  TIPO: OBJECT-IDENTITY[40]

  ESTADO: current.

  DESCRIPCIÓN: Es un objeto identificador de ausencia de protocolo de autenticación.

- **usmHMACMD5AuthProtocol:**
  OID: *snmpAuthProtocols(1).usmHMACMD5AuthProtocol(2).*

  TIPO: OBJECT-IDENTITY

  ESTADO: current.

  DESCRIPCIÓN: Es un objeto identificador del protocolo de autenticación HMAC-MD5-96.

- **usmHMACSHAAuthProtocol:**
  OID: *snmpAuthProtocols(1).usmHMACSHAAuthProtocol(3).*

  TIPO: OBJECT-IDENTITY

  ESTADO: current.

[40] OBJECT-IDENTITY es una macro perteneciente a SMIv2 que se utiliza para definir información sobre la asignación de un identificador de objeto MIB.

DESCRIPCIÓN: Es un objeto identificador del protocolo de autenticación HMAC-SHA-96.

1.4.4.9.2 snmpPrivProtocols

Este grupo es importado desde la MIB *snmpFrameworkMIB.* Se emplea para la identificación de los protocolos de privacidad. Su objeto identificador es: *iso(1).org(3).dod(6).internet(1).snmpV2(6).snmpModules(3).snmpFrameworkMIB(10). snmpFrameworkAdmin(1).snmpPrivProtocols(2)*. Se divide en:

- **usmNoPrivProtocol:**
  OID: *snmpPrivProtocols(2).usmNoPrivProtocol(1).*

  TIPO: OBJECT-IDENTITY
  ESTADO: current.

  DESCRIPCIÓN: Es un objeto identificador de ausencia de protocolo de privacidad.

- **usmDESPrivProtocol:**
  OID: *snmpPrivProtocols(2).usmDESPrivProtocol(2).*

  TIPO: OBJECT-IDENTITY

  ESTADO: current.

  DESCRIPCIÓN: Es un objeto identificador del protocolo de privacidad DES.

- **usmAesCfb128Protocol**[41]**:**
  OID: *snmpPrivProtocols(2).usmAesCfb128Protocol(4).*

  TIPO: OBJECT-IDENTITY

  ESTADO: current.

  DESCRIPCIÓN: Es un objeto identificador del protocolo de privacidad AES.

1.4.4.9.3 usmUser

Este grupo MIB, cuyo OID es: *snmpUsmMIB(15).usmMIBObjects(1).usmUser(2),* contiene información relacionada a los principales locales y remotos. Está formado por los objetos: **usmUserSpinLock** y **usmUserTable**. El *usmUserSpinLock,* cuyo OID es: *snmpUsmMIB(15).usmMIBObjects(1).usmUser(2).usmUserSpinLock(1),* es un objeto escalar que hace posible la cooperación de las aplicaciones Generadoras de Comandos para coordinar el cambio de claves a través del uso de la tabla *usmUserTable*. La tabla *usmUserTable,* cuyo OID es: *snmpUsmMIB(15).usmMIBObjects(1).usmUser(2).usmUserTable(2),* contiene información de los Principales válidos configurados en un motor SNMP. Esta tabla está conformada por los siguientes objetos columnas:

- **usmUserEngineID:**
  OID: *usmUser(2).usmUserTable(2).usmUserEntry(1).usmUserEngineID(1).*

  SINTAXIS: snmpEngineID (OCTET STRING).

  ACCESO: not-accessible.

[41] Este nodo MIB está definido en el RFC 3826: The Advanced Encryption Standard (AES) Cipher Algorithm in the SNMP User-based Security Model.

ESTADO: current.

DESCRIPCIÓN: Identificador de un motor SNMP autorizado.

- **usmUserName:**
  OID: *usmUser(2).usmUserTable(2).usmUserEntry(1).usmUserName(2).*

  SINTAXIS: snmpAdminString (OCTET STRING).

  ACCESO: not-accessible.

  ESTADO: current.

  DESCRIPCIÓN: Nombre que identifica a un usuario (Principal) con un formato dependiente del Modelo de Seguridad.

- **usmUserSecurityName:**
  OID: *usmUser(2).usmUserTable(2).usmUserEntry(1).usmUserSecurityName (3).*

  SINTAXIS: snmpAdminString (OCTET STRING).

  ACCESO: read-only.

  ESTADO: current.

  DESCRIPCIÓN: Nombre que identifica un usuario pero con un formato independiente del Modelo de Seguridad. Por defecto es *usmUserName.*

- **usmUserCloneForm:**
  OID: *usmUser(2).usmUserTable(2).usmUserEntry(1).usmUserCloneForm(4).*

SINTAXIS: RowPointer (OBJECT IDENTIFIER).

ACCESO: read-create.

ESTADO: current.

DESCRIPCIÓN: Es un puntero que hace referencia a otra fila de la cual se copian los parámetros de autenticación y privacidad de un Principal en particular.

- **usmUserAuthProtocol:**
  OID: *usmUser(2).usmUserTable(2).usmUserEntry(1).usmUserAuthProtocol(5).*

  SINTAXIS: AutonomousType (OBJECT IDENTIFIER).

  ACCESO: read-create.
  ESTADO: current.

  DESCRIPCIÓN: Es un indicador que especifica si el mensaje enviado en nombre de este usuario necesita autenticación (define el protocolo a usarse) o no.

- **usmUserAuthKeyChange:**
  OID: *usmUser(2).usmUserTable(2).usmUserEntry(1).usmUserAuthKeyChange(6).*

  SINTAXIS: KeyChange (OCTET STRING).

  ACCESO: read-create.

  ESTADO: current.

DESCRIPCIÓN: Es un octeto que al ser modificado, una función *one-way* modifica también la clave de autenticación usada por este usuario.

- **usmUserOwnAuthKeyChange:**
  OID: *usmUser(2).usmUserTable(2).usmUserEntry(1).usmUserOwnAuthKeyChange(7).*

  SINTAXIS: KeyChange (OCTET STRING).

  ACCESO: read-create.

  ESTADO: current.

  DESCRIPCIÓN: Se comporta igual que el objeto anterior siempre y cuando el *usmUserName* del solicitante sea el mismo que el *usmUserName* referenciado en esta fila.

- **usmUserPrivProtocol:**
  OID: *usmUser(2).usmUserTable(2).usmUserEntry(1).usmUserPrivProtocol(8).*

  SINTAXIS: AutonomousType (OBJECT IDENTIFIER).

  ACCESO: read-create.

  ESTADO: current.

  DESCRIPCIÓN: Es un indicador que especifica si el mensaje enviado en nombre de este usuario necesita privacidad (define el protocolo a usarse) o no.

- **usmUserPrivKeyChange:**
  OID: *usmUser(2).usmUserTable(2).usmUserEntry(1).usmUserPrivKeyChange(9).*

SINTAXIS: KeyChange (OCTET STRING).

ACCESO: read-create.

ESTADO: current.

DESCRIPCIÓN: Es un octeto que al ser modificado, una función *one-way* modifica también la clave de privacidad usada por este usuario.

- **usmUserOwnPrivKeyChange:**
  OID: *usmUser(2).usmUserTable(2).usmUserEntry(1).usmUserOwnPrivKeyChange(10).*

  SINTAXIS: KeyChange (OCTET STRING).

  ACCESO: read-create.
  ESTADO: current.

  DESCRIPCIÓN: Se comporta igual que el objeto anterior siempre y cuando el *usmUserName* del solicitante sea el mismo que el *usmUserName* referenciado en esta fila.

- **usmUserPublic:**
  OID: *usmUser(2).usmUserTable(2).usmUserEntry(1).usmUserPublic(11).*

  SINTAXIS: OCTET STRING.

  ACCESO: read-create.

  ESTADO: current.

DESCRIPCIÓN: Es un valor escrito como parte de algún procedimiento de cambio de clave de autenticación o cifrado perteneciente a un usuario. Este valor sirve para la comprobación del cambio*.*

- **usmUserStorageType:**
  OID: *usmUser(2).usmUserTable(2).usmUserEntry(1).usmUserStorageType(12).*

  SINTAXIS: StorageType (INTEGER) {other(1), volatile(2), nonVolatile(3), permanent(4), readOnly(5)}.

  ACCESO: read-create.

  ESTADO: current.

  DESCRIPCIÓN: Indica el tipo de almacenamiento para la fila actual.

- **usmUserStatus:**
  OID: *usmUser(2).usmUserTable(2).usmUserEntry(1).usmUserStatus(13).*

  SINTAXIS: RowStatus (INTEGER) {active(1), notInService(2), notReady(3), createAndGo(4), createAndWait(5), destroy(6)}.

  ACCESO: read-create.

  ESTADO: current.

  DESCRIPCIÓN: Indica el estado de la fila actual.

1.4.5 MODELO DE CONTROL DE ACCESO BASADO EN VISTAS (VACM) [8][11]

VACM (*View-based Access Control Model – Modelo de Control de Acceso basado en Vistas*) está definido en el RFC 3415[42]. Las funciones principales de este modelo son: controlar el acceso hacia un objeto MIB a un principal remoto, definir un grupo MIB destinado a las políticas de control de acceso y hacer posible la configuración remota.

El modelo VACM utiliza los campos *msgFlags*, *msgSecurityModel* y *scopedPdu* del mensaje SNMPv3 para determinar el tipo de acceso que tiene el mensaje. Si el acceso no es permitido a dicho tipo de solicitud, entonces de envía una notificación de error al principal.

1.4.5.1 Elementos del modelo VACM [8][11]

1.4.5.1.1 Grupos

Un Grupo es un conjunto de principales a quienes se les asigna algunos derechos de acceso a determinados objetos de la MIB. Cada principal está representado por una dupla <*securityModel, securityName*>. Un grupo es identificado por un *groupName*. En un agente, un principal debe pertenecer a un solo grupo.

1.4.5.1.2 Nivel de seguridad

Es el grado de seguridad con el cual un mensaje SNMP ha sido enviado o con el cual ciertas operaciones están siendo procesadas. El nivel de seguridad es importante en la definición de derechos de acceso, de tal manera que, si el nivel de seguridad es *noAuthNoPriv,* entonces los derechos de acceso pueden ser sólo de lectura. Si el nivel de seguridad es *authPriv* o *authNoPriv*, entonces los derechos de acceso pueden ser de escritura.

[42] RFC 3415: View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP). RFC publicado en diciembre de 2002.

1.4.5.1.3 Contextos

Un Contexto es una colección de instancias de objetos que tiene asignado diferentes derechos de acceso y que puede ser accedido por una entidad SNMP (por ejemplo, una aplicación contestadora de comandos). Una instancia de objeto puede pertenecer a más de un contexto SNMP. Una entidad SNMP (identificada por un *contextEngineID*) puede acceder a más de un contexto SNMP. Por tanto, para identificar la instancia de un objeto individual se requiere el *contextName* (es el identificador del contexto), *contextEngineID*, el tipo de objeto y la instancia. El *contextName* por defecto es una cadena vacía.

1.4.5.1.4 Vistas MIB

Una Vista MIB es una agrupación de subárboles de objetos. Cada subárbol puede estar incluido o excluido de la vista. Una vista sirve para controlar el acceso hacia determinados objetos a un Grupo específico. Una vista está relacionada con cada objeto identificador (OID) de cada subárbol junto a su respectiva máscara. La máscara indica qué subidentificadores del subárbol asociado son de interés en la definición de la vista MIB.

1.4.5.1.5 Derechos de acceso

Son las normas de acceso que se asignan a un contexto SNMP. La determinación de un derecho de acceso depende de los siguientes factores:

- El gestor SNMP (principal) que está realizando el pedido de acceso.
- El nivel de seguridad del mensaje de pedido.
- El modelo de seguridad usado para el procesamiento del mensaje de pedido.
- El contexto MIB referenciado en el mensaje de pedido.
- La instancia de objeto específica que se ha solicitado.
- El tipo de acceso solicitado (lectura, escritura o notificación).

1.4.5.2 MIBs para VACM [20]

El grupo **snmpVacmMIB**, definido en el RFC3415[43] y cuyo OID es: *iso(1).org(3).dod(6).internet(1).snmpV2(6).snmpModules(3).snmpVacmMIB(16),* contiene los objetos usados por VACM para el efectuar el control de acceso. Este grupo se deriva en las siguientes tablas:

1.4.5.2.1 vacmContextTable

La tabla **vacmContextTable,** cuyo OID es: *snmpVacmMIB(16).vacmMIBObjects(1).vacmContextTable(1)*, almacena todos los contextos locales disponibles. Esta tabla es utilizada por las aplicaciones Generadoras de Comandos para configurar la tabla *vacmAccessTable*.[44] Cada fila de esta tabla contiene un:

- **vacmContextName**:
  OID: *vacmContextTable(1).vacmContextEntry(1).vacmContextName(1).*

  SINTAXIS: snmpAdminString (OCTET STRING).

  ACCESO: read-only.

  ESTADO: current.

  DESCRIPCIÓN: Nombre textual de un contexto (*contextName*).

1.4.5.2.2 vacmSecurityToGroupTable

La tabla **vacmSecurityToGroupTable,** cuyo OID es: *snmpVacmMIB(16).vacmMIBObjects(1).vacmSecurityToGroupTable(2),* almacena

[43] RFC3415: View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP). RFC publicado en diciembre de 2002.
[44] Esta tabla se verá posteriormente.

información de los grupos. Esta tabla contiene filas conformadas por las siguientes columnas:

- **vacmSecurityModel:**
  OID: *vacmSecurityToGroupTable (2).vacmSecurityToGroupEntry(1). vacmSecurityModel(1).*

  SINTAXIS: snmpSecurityModel (INTEGER).

  ACCESO: not-accessible.

  ESTADO: current.

  DESCRIPCIÓN: Modelo de seguridad en uso (*securityModel*).

- **vacmSecurityName:**
  OID: *vacmSecurityToGroupTable (2).vacmSecurityToGroupEntry(1). vacmSecurityName(2).*

  SINTAXIS: snmpAdminString (OCTET STRING).

  ACCESO: not-accessible.
  ESTADO: current.

  DESCRIPCIÓN: Es el *securityName*. Si el modelo de seguridad usado es USM, *securityName* y *userName* son idénticos.

- **vacmGroupName:**
  OID: *vacmSecurityToGroupTable (2).vacmSecurityToGroupEntry(1). vacmGroupName(3).*

  SINTAXIS: snmpAdminString (OCTET STRING).

ACCESO: read-create.

ESTADO: current.

DESCRIPCIÓN: Nombre textual del grupo (*groupName*).

- **vacmSecurityToGroupStorageType:**
  OID: *vacmSecurityToGroupTable (2).vacmSecurityToGroupEntry(1). vacmSecurityToGroupStorageType(4).*

  SINTAXIS: StorageType (INTEGER) {other(1), volatile(2), nonVolatile(3), permanent(4), readOnly(5)}.

  ACCESO: read-create.

  ESTADO: current.

  DESCRIPCIÓN: Indica el tipo de almacenamiento para la fila actual.

- **vacmSecurityToGroupStatus:**
  OID: *vacmSecurityToGroupTable (2).vacmSecurityToGroupEntry(1). vacmSecurityToGroupStatus(5).*

  SINTAXIS: RowStatus (INTEGER) {active(1), notInService(2), notReady(3), createAndGo(4), createAndWait(5), destroy(6)}.

  ACCESO: read-create.

  ESTADO: current.

  DESCRIPCIÓN: Indica el estado de la fila actual.

1.4.5.2.3 vacmAccessTable

La tabla **vacmAccessTable,** cuyo OID es: *snmpVacmMIB(16).vacmMIBObjects(1).vacmAccessTable(4),* almacena los derechos de acceso definidas para cada grupo. Esta tabla contiene filas conformadas por las siguientes columnas:

- **vacmAccessContextPrefix:**
  OID: *vacmAccessTable(4).vacmAccessEntry(1).vacmAccessContextPrefix(1).*

  SINTAXIS: snmpAdminString (OCTET STRING).

  ACCESO: not-accessible.

  ESTADO: current.

  DESCRIPCIÓN: Puede ser un prefijo o nombre completo del contexto (*contextName*) al cual se va acceder.

- **vacmAccessSecurityModel:**
  OID: *vacmAccessTable(4).vacmAccessEntry(1).vacmAccessSecurityModel(2).*

  SINTAXIS: snmpSecurityModel (INTEGER).

  ACCESO: not-accessible.

  ESTADO: current.

  DESCRIPCIÓN: Modelo de seguridad que debe ser usado para obtener acceso.

- **vacmAccessSecutityLevel:**
  OID: *vacmAccessTable(4).vacmAccessEntry(1).vacmAccessSecutityLevel(3).*

  SINTAXIS: snmpSecurityLevel (INTEGER).

  ACCESO: not-accessible.

  ESTADO: current.

  DESCRIPCIÓN: Indica el nivel de seguridad mínimo que debe ser usado para obtener acceso.

- **vacmAccessContextMatch:**
  OID: *vacmAccessTable(4).vacmAccessEntry(1).vacmAccessContextMatch(4).*

  SINTAXIS: INTEGER {exact(1), prefix(2)}
  ACCESO: read-create.

  ESTADO: current.

  DESCRIPCIÓN: Es un factor que determina que valor contendrá *vacmAccessContextPrefix.* El valor del factor puede ser: *exact* o *prefix.* Si es *exact,* el valor de *vacmAccessContextPrefix* será el mismo que *contextName*. Si es *prefix*, el valor de *vacmAccessContextPrefix* será un prefijo (algunos de los primeros octetos) de *contextName.*

- **vacmAccessReadViewName:**
  OID: *vacmAccessTable(4).vacmAccessEntry(1).vacmAccessReadViewName(5).*

  SINTAXIS: snmpAdminString (OCTET STRING).

ACCESO: read-create.

ESTADO: current.

DESCRIPCIÓN: Nombre de la vista MIB (*viewName*) sobre la cual se autoriza accesos sólo de lectura.

- **vacmAccessWriteViewName:**
  OID: *vacmAccessTable(4).vacmAccessEntry(1).vacmAccessWriteViewName(6).*

  SINTAXIS: snmpAdminString (OCTET STRING).

  ACCESO: read-create.

  ESTADO: current.

  DESCRIPCIÓN: Nombre de la vista MIB sobre la cual se autoriza accesos de escritura.

- **vacmAccessNotifyViewName:**
  OID: *vacmAccessTable(4).vacmAccessEntry(1).vacmAccessNotifyViewName(7).*

  SINTAXIS: snmpAdminString (OCTET STRING).

  ACCESO: read-create.

  ESTADO: current.

  DESCRIPCIÓN: Nombre de la vista MIB sobre la cual se autoriza accesos de notificación.

- **vacmAccessStorageType:**
  OID: *vacmAccessTable(4).vacmAccessEntry(1).vacmAccessStorageType(8).*

  SINTAXIS: StorageType (INTEGER) {other(1), volatile(2), nonVolatile(3), permanent(4), readOnly(5)}.

  ACCESO: read-create.

  ESTADO: current.

  DESCRIPCIÓN: Indica el tipo de almacenamiento para la fila actual.

- **vacmAccessStatus:**
  OID: *vacmAccessTable(4).vacmAccessEntry(1).vacmAccessStatus(8).*

  SINTAXIS: RowStatus (INTEGER) {active(1), notInService(2), notReady(3), createAndGo(4), createAndWait(5), destroy(6)}.

  ACCESO: read-create.

  ESTADO: current.

  DESCRIPCIÓN: Indica el estado de la fila actual.

1.4.5.2.4 vacmMIBViews

El grupo **vacmMIBViews,** cuyo OID es: *snmpVacmMIB(16).vacmMIBObjects(1).vacmMIBViews(5),* está compuesto por el objeto escalar **vacmViewSpinLock** (OID: *vacmMIBViews(5).vacmViewSpinLock(1))*) y el objeto tabla **vacmViewTreeFamilyTable** (OID: *vacmMIBViews(5).vacmViewTreeFamilyTable(2)*). El objeto *vacmViewSpinLock* es

usado por las aplicaciones Generadoras de Comandos para coordinar el uso del comando *SetRequest* en la modificación de las vistas MIB. La tabla *vacmViewTreeFamilyTable* almacena las vistas MIB. Esta tabla contiene filas conformadas por las siguientes columnas:

- **vacmViewTreeFamilyViewName:**
  OID: *vacmViewTreeFamilyTable(2).vacmViewTreeFamilyEntry(1). vacmViewTreeFamilyViewName(1).*

  SINTAXIS: snmpAdminString (OCTET STRING).

  ACCESO: not-accessible.

  ESTADO: current.

  DESCRIPCIÓN: Nombre (*viewName*) de la vista MIB.

- **vacmViewTreeFamilySubtree:**
  OID: *vacmViewTreeFamilyTable(2).vacmViewTreeFamilyEntry(1). vacmViewTreeFamilySubtree(2).*

  SINTAXIS: OBJECT IDENTIFIER.

  ACCESO: not-accessible.

  ESTADO: current.

  DESCRIPCIÓN: Es el OID de un subárbol.

- **vacmViewTreeFamilyMask:**
  OID: *vacmViewTreeFamilyTable(2).vacmViewTreeFamilyEntry(1). vacmViewTreeFamilyMask(3).*

SINTAXIS: OCTET STRING.

ACCESO: read-create.

ESTADO: current.

DESCRIPCIÓN: Representa el bit de máscara; ésta en combinación con *vacmViewTreeFamilySubtree* definen los subárboles para armar la vista MIB.

- **vacmViewTreeFamilyType:**
  OID: *vacmViewTreeFamilyTable(2).vacmViewTreeFamilyEntry(1). vacmViewTreeFamilyType(4).*

  SINTAXIS: INTEGER {included(1), excluded(2)}.

  ACCESO: read-create.

  ESTADO: current.

  DESCRIPCIÓN: Indica que si los subárboles definidos mediante *vacmViewTreeFamilySubtree* y *vacmViewTreeFamilyMask* deben ser incluidos o no en la vista MIB.

- **vacmViewTreeFamilyStorageType:**
  OID: *vacmViewTreeFamilyTable(2).vacmViewTreeFamilyEntry(1). vacmViewTreeFamilyStorageType(5).*

  SINTAXIS: StorageType (INTEGER) {other(1), volatile(2), nonVolatile(3), permanent(4), readOnly(5)}.

  ACCESO: read-create.

ESTADO: current.

DESCRIPCIÓN: Indica el tipo de almacenamiento para la fila actual.

- **vacmViewTreeFamilyStatus:**
  OID: *vacmViewTreeFamilyTable(2).vacmViewTreeFamilyEntry(1). vacmViewTreeFamilyStatus(6).*

  SINTAXIS: RowStatus (INTEGER) {active(1), notInService(2), notReady(3), createAndGo(4), createAndWait(5), destroy(6)}.

  ACCESO: read-create.

  ESTADO: current.

  DESCRIPCIÓN: Indica el estado de la fila actual.

La figura 1.24 muestra el diagrama de flujo que representa cómo se relacionan las variables y las tablas VACM para tomar una decisión sobre permitir el acceso o no.

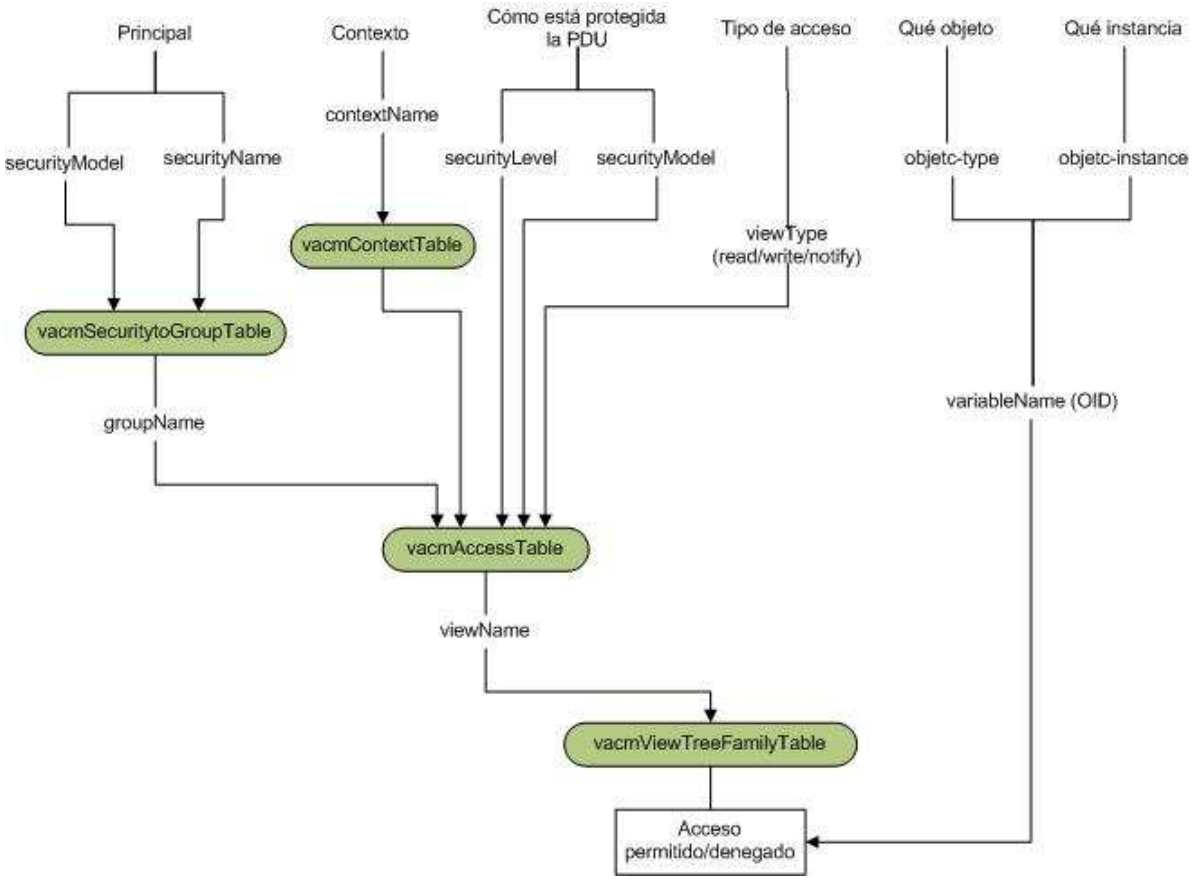


Figura 1. 24 Diagrama de flujo de VACM.[45]

[45] STALLINGS William; SNMP, SNMPv2, SNMPv3 and RMON 1 and 2, Tercera Edición, Figura 17.2 tomada de la pág.532, Acceso: 5 de abril 2010, 17:00.

1.5 XML

1.5.1 INTRODUCCIÓN A XML [21][22][10]

XML (*eXtensible Markup Language*) es un lenguaje extensible de etiquetas. XML es una recomendación de w3C[46] que fue diseñada para la estructuración, almacenamiento y transporte de información; además, permite definir otros lenguajes de uso específico y dispone de varias herramientas de desarrollo.

Una etiqueta o marca es una palabra limitada por símbolos “<” y “>”. Las etiquetas sirven para representar la información contenida entre un par de ellas (etiqueta de inicio y etiqueta de fin). XML no establece etiquetas predefinidas, éstas son definidas por el usuario. Las etiquetas en conjunto permiten codificar un documento completo.

XML tiene múltiples aplicaciones, entre las más comunes están las siguientes:

- Sirve para separar la parte de datos de un documento HTML dinámico. Esto significa que los datos se almacenan en un archivo XML aparte y se evita modificar el documento HTML cada vez que se produzca un cambio de datos.
- Ayuda a simplificar la compartición y el transporte de datos. Esto se debe gracias a que XML guarda la información en texto plano de tal modo que es compatible con cualquier tipo de aplicación.
- Facilita la actualización o expansión de sistemas mediante la agregación de etiquetas determinadas.
- Incrementa la disponibilidad de la información almacenada.
- Permite la creación de nuevos lenguajes para usos específicos.

1.5.2 DOCUMENTO XML [10][21][23][24]

Un documento XML es un fichero cuya información está codificada con el lenguaje XML y está estructurada en forma de árbol jerárquico.

[46] W3C, World Wide Web Consortium. Es una asociación internacional que produce recomendaciones para la web.

1.5.2.1 Partes de un documento XML

1.5.2.1.1 Declaración XML

En la declaración XML se definen la versión XML y la codificación de texto a usarse.

```
<?xml versión=”1.0” encoding=”ISO-8859-1”?>
```

Por ejemplo, esta línea significa que va a usar la versión 1.0 de XML y los datos se van a escribir con caracteres de tipo ISO-8859-1 (Latin-1/West European).[47]

1.5.2.1.2 Elementos

Un elemento XML es todo lo va entre una etiqueta de inicio y una etiqueta de fin. Un elemento puede contener otros elementos y éstos a su vez pueden tener atributos.

```
<librería>
  <libro categoría=”WEB”>
    <titulo>Aprendiendo XML<\titulo>
  <\libro>
<\librería>
```

En este ejemplo, el elemento `<librería>` está formado por el elemento `<libro>`, y el de `<libro>` por `<título>`. El elemento `<título>` está conformado por los caracteres “`Aprendiendo XML`”. Además, `<libro>` contiene un atributo llamado “categoría” cuyo valor es “`WEB`”.

Existe también el concepto de **elemento root**. Éste permite establecer puntos de referencia para los analizadores o aplicaciones XML del inicio y final de un documento XML. En el ejemplo anterior el elemento root sería `<librería>`.

[47] ISO-8859-1 es un estándar de codificación de caracteres basado en ASCII, 8 bits, y de alfabeto Latino.

1.5.2.1.3 Atributos

Proveen información adicional acerca de un elemento. Esta información ayuda a la aplicación en el procesamiento del elemento. Los atributos se ubican en la etiqueta de inicio de un elemento y se expresan en el formato: *nombre_atributo="valor".* En el ejemplo anterior, el nombre del atributo es categoría y su valor es `"WEB"`.

El nombre de un atributo puede aparecer una sola vez en una etiqueta de inicio. Además, pueden existir varios pares atributo/valor en una etiqueta sin importar el orden en que estén ubicados ya que el resultado no cambia.

1.5.2.1.4 Datos de carácter

Los datos de carácter representan el contenido de un elemento XML.

1.5.2.1.5 Sector CDATA

Es un término especial que provoca al analizador XML ignorar la información contenida en dicho término. Se utiliza para representar datos que contienen caracteres especiales (por ejemplo &, <,>, etc). Los caracteres especiales al ser leídos por un analizador XML cumplen alguna función específica; sin embargo, cuando estos caracteres están incluidos en un sector CDATA son interpretados como información común. Un sector CDATA comienza con `"<![CDATA["` y finaliza con `"]]>"`.

1.5.2.1.6 Referencias de Entidad

Es un tipo de dato usado en XML para referenciar alguna información. La referencia de entidad consiste de un nombre único de entidad precedido por un ampersand (&) y seguido por un punto y coma (;), es decir: *"&nombre_entidad;".* Las entidades son variables usadas para definir accesos rápidos a caracteres especiales o texto común.

Cuando un analizador XML observa una referencia de entidad, reemplaza esta referencia con el valor asignado a la entidad.
Existen referencias de entidades predefinidas que tratan acerca de caracteres especiales empleados en XML. La tabla 1.9 muestra estas referencias de entidad.

Referencia de entidad	Dato	Descripción
<	<	menor que
>	>	mayor que
&	&	ampersand
'	‘	apóstrofe
"	“	comillas dobles

Tabla 1. 9 Referencias de entidad predefinidas.[48]

1.5.2.1.7 Instrucciones de Procesamiento

Son comandos que hacen que la aplicación realice tareas específicas. Estas instrucciones están delimitadas por las etiquetas "`<?`" y "`?>`". Por ejemplo:

```
<?xml-stylesheet href="XSL/JavaXML.html.xsl" type="text/xsl"?>
```

Donde, `xml-stylesheet` es el objetivo o la acción de la instrucción, y el resto son los datos sobre los cuales se va a realizar la operación.

1.5.2.1.8 Comentarios

Un comentario es información que la considera como tal, siempre y cuando esté contenida entre las etiquetas: "`<!--`" y "`-->`".

[48] RAY, Erik; Learning XML, Editorial O'Reilly, Tabla tomada del ítem 2.5.1 “Character Entities”. Acceso: 11 abril 2010, 14:00.

1.5.2.1.9 Namespaces

Un *Namespace* XML es un método para evitar conflictos de nombres de elementos. En un documento XML pueden existir elementos con el mismo nombre pero distinto contenido, y por tanto necesitan una identificación única mediante su propio nombre y un *namespace* URI. [49]

El espacio de nombres consiste en la agregación de un prefijo y un atributo "*xmlns"* en la etiqueta de inicio de un elemento. Este atributo define la URI asignada al prefijo.

```
<h: table xmlns:h="http://www.w3.org/TR/html14"> </h:table>
```

En el ejemplo, el elemento es `<table>`, el prefijo es `"h"` y la URI asignada a este prefijo es `"http://www.w3.org/TR/html14"`.

1.5.3 VALIDACIÓN XML [10][21][23][24]

Según XMLv1.0 existen dos tipos de documentos aceptados: los documentos XML bien formados y los documentos XML válidos.

1.5.3.1 Documentos XML bien formados.

Un documento de este tipo tiene una sintaxis XML correcta. Las **reglas de sintaxis** que siguen los documentos bien formados son las siguientes:

- Los documentos XML deben tener un único elemento root.
- Todos los elementos del documento XML deben tener su respectiva etiqueta de cierre.
- Las etiquetas XML son sensibles a letras mayúsculas y minúsculas.
- Los elementos XML deben estar correctamente anidados, es decir no debe existir solapamiento entre etiquetas de diferentes elementos.

[49] Un URI (Uniform Resource Identifier – Identificador Uniforme del Recurso) es una cadena de caracteres que identifica un recurso de Internet. El ejemplo más claro de un URI es un URL (Uniform Resource Locator - Localizador Uniforme de Recursos) el cual identifica una dirección de dominio de Internet.

- Los valores de atributos de un elemento deben ir entre comillas dobles.

1.5.3.2 Documentos XML válidos. [25][10]

Un documento XML válido es un documento que, además de ser un documento bien formado, sigue las reglas de **DTD (*Document Type Definition – Definición de Tipo de Documento*)**. El propósito general de DTD es definir la estructura de un documento XML siguiendo un formato preestablecido de declaraciones. Las declaraciones son especificaciones de qué elementos, atributos, entidades o notaciones van a formar parte del documento y los valores que éstos pueden contener.

1.5.3.2.1 Declaración DTD

Esta declaración sirve para indicar que el documento es un documento XML válido. Existen dos tipos de declaraciones:

- **Declaración DTD interna:** La declaración DTD va en el mismo documento XML. Tiene la siguiente forma:

  ```
  <!DOCTYPE elemento_root [declaraciones_de_elementos]>
  ```

  `!DOCTYPE` define el elemento principal del documento.

- **Declaración DTD externa:** La declaración DTD va en un fichero de extensión .dtd externo (referenciado por `"URI/URL"`). En el documento XML se escribe la siguiente sintaxis el cual indica que la declaración DTD es externa:

  ```
  <!DOCTYPE elemento_root SYSTEM "URI/URL">
  ```

1.5.3.2.2 Declaraciones de elementos

Para la declaración de un elemento se tiene la siguiente sintaxis:

```
<!ELEMENT nombre_del_elemento (contenido_del_elemento)>
ó

<!ELEMENT nombre_del_elemento categoría)>
```

El contenido del elemento puede ser: elementos hijos, el parámetro `#PCDATA` (indica que el contenido debe ser sólo texto) o una combinación de ambos.
Referente a categoría puede ser uno de los siguientes valores: `ANY` (el contenido puede ser otros elementos y/o solo texto) o `EMPTY` (contenido vacío).

1.5.3.2.3 Declaraciones de atributos

Para la declaración de un atributo se tiene la siguiente sintaxis:

```
<!ATTLIST nombre_elemento nombre_atributo tipo_atributo valor_por_defecto>
```

Tipo de atributo	**Descripción**
CDATA	El valor del atributo será una cadena de caracteres.
ID	El valor del atributo será un identificador único correspondiente al elemento.
IDREF	El valor del atributo será el identificador único correspondiente a otro elemento.
IDREFS	El valor del atributo será una lista de otros identificadores.
NMTOKEN	El valor del atributo será una cadena de caracteres válidos.
NMTOKENS	El valor del atributo será una lista de cadena de caracteres válidos.
ENTITY	El valor del atributo será una entidad.

ENTITIES	El valor del atributo será una lista de entidades.
NOTATION	El valor del atributo será el nombre de una notación.

Tabla 1. 10 Tipos de atributos.[50]

Valor por defecto	**Significado**
#REQUIRED	El atributo obligatoriamente debe tener un valor.
#IMPLIED	El atributo puede o no tener un valor.
#FIXED	El atributo tendrá el valor definido en la declaración de atributos.

Tabla 1. 11 Lista de valores por defecto.[51]

1.5.3.2.4 Declaraciones de entidades

Se declaran dos tipos generales de entidades: entidades analizadas y entidades no analizadas.

Las **entidades analizadas** son aquellas cuyo contenido se utiliza para reemplazar texto XML válido. Las entidades analizadas son invocadas a través de referencias de entidad. Este tipo de entidades se divide en: entidades internas, entidades externas y entidades de parámetros.

- **Declaraciones de entidades internas:** La entidad interna es aquella cuyo valor asignado está definido en la misma declaración. La declaración tiene la siguiente sintaxis:

```
<!ENTITY nombre_entidad "valor_asignado_a_entidad">
```

- **Declaraciones de entidades externas:** La entidad externa es aquella cuyo valor asignado está definido en una dirección de dominio "URI/URL". La declaración tiene la siguiente sintaxis:

[50] W3SCHOOLS, DTD - Attributes, http://www.w3schools.com/dtd/dtd_attributes.asp. Acceso: 17 abril 2010, 16:00.
[51] W3SCHOOLS, DTD - Attributes, http://www.w3schools.com/dtd/dtd_attributes.asp. Acceso: 17 abril 2010, 16:20

```
<!ENTITY nombre_entidad SYSTEM "URI/URL">
```

- **Declaraciones de entidades de parámetros:** La entidad de parámetros es aquella que sirve para reemplazar texto sólo en declaraciones DTD y no en documentos XML. La declaración tiene la siguiente sintaxis:

  ```
  <!ENTITY %nombre_entidad "valor_asignado_a_entidad">
  ```

  El símbolo "%" va tanto en la declaración de la entidad como en la referencia de la misma: `"%nombre_entidad;"`.

Las **entidades no analizadas** son aquellas cuyo contenido puede ser cualquier texto no necesariamente XML. Las entidades no analizadas son invocadas a través de su nombre el cual está definido como valor del atributo ENTITY o ENTITIES.

- **Declaraciones de entidades no analizadas:** Tiene la siguiente sintaxis:

  ```
  <!ENTITY nombre_entidad SYSTEM "URI/URL" NDATA nombre_notación>
  ```

  `NDATA` define la notación asociada a la entidad no analizada.

1.5.3.2.5 Declaraciones de notaciones

Las **notaciones** sirven para indicar el tipo de archivo del valor asignado (URL) a una entidad no analizada. Las notaciones se invocan a través de su nombre. La declaración tiene la siguiente sintaxis:

```
<!NOTATION nombre_notación PUBLIC "tipo_archivo">
```

`PUBLIC` define el tipo de archivo.

1.5.4 OTROS LENGUAJES BASADOS EN XML

1.5.4.1 XSD [26]

XSD (*XML Schema Definition – Definición de Esquema XML*) o Esquema XML es un lenguaje que describe estructura de documentos XML de mejor manera que DTD.

A más de definir de manera más precisa elementos y atributos, XSD tiene las siguientes características:

- Sigue las reglas de sintaxis XML.
- Trabaja con *namespaces*.
- Tiene soporte para tipos de datos, esto es: definición de tipos de datos aceptados para contenido de documentos XML, validación de integridad de datos, restricciones en los valores de datos, definición de formato de datos, y conversión de tipos de datos.
- XSD permite trabajar con datos desde bases de datos.
- XSD es extensible para futuras adiciones.
- Las declaraciones XSD que definen la estructura de un documento están contenidas en un fichero .xsd externo, el mismo que es referenciado mediante direcciones URL desde el documento XML.

1.5.4.2 DOM [10][27]

DOM (*Document Object Model – Modelo de Objetos de Documento*) es una plataforma y lenguaje que permite acceder y manipular documentos tales como XML y HTML mediante definición de objetos y métodos.

DOM está divido en tres partes:

- **Core DOM:** Es un modelo estándar para cualquier documento estructurado.
- **XML DOM:** Es un modelo estándar para documentos XML.
- **HTML DOM:** Es un modelo estándar para documentos HTML.

1.5.4.2.1 XML DOM

XML DOM es un modelo de objetos estándar para documentos XML. Su función principal es representar un documento XML como un conjunto de objetos que tienen propiedades y métodos. Gracias a esto es posible acceder a la información de un documento y manejarla.

Según XML DOM, todo lo que existe en documento XML es un nodo, es decir existen: nodo documento, nodo elemento, el contenido de texto de un elemento es un nodo, nodo atributo y nodo comentario. Un nodo puede contener otros nodos. Un documento XML es representado como un árbol estructurado de nodos. Por lo tanto, todos los nodos pueden ser accedidos a través del árbol para realizar sobre éstos tareas de edición, creación o eliminación.

XML DOM dispone de una interfaz de programación de aplicaciones (API DOM) que está definida por las propiedades y métodos de los objetos los cuales permiten acceder y modificar los nodos.

Para convertir un documento XML a árbol de nodos se utiliza un analizador XML DOM.

1.5.4.3 SAX [10][28]

SAX (*Simple API for XML – API Simple para XML*) es una interfaz sencilla que permite acceder y manejar documentos XML por medio de eventos. Un evento es un suceso que al ocurrir provoca la generación de una respuesta por parte del sistema. Por ejemplo, un evento en XML sería cuando el analizador XML/SAX encuentra una etiqueta de inicio e informa a la aplicación XML para que invoque algún método que realice alguna acción específica.

SAX no crea un árbol de nodos a partir del documento XML. SAX no crea un árbol de nodos para procesar la información; la información es directamente pasada a la aplicación XML desde el documento XML, esto implica un ahorro de consumo de procesamiento y memoria.

Entre las actividades que se puede realizar SAX están: modificar un documento XML haciendo cambios pequeños (por ejemplo, renombrar elementos), leer datos para construir una estructura de datos más compleja, buscar un elemento dentro de un documento XML proporcionando una palabra clave, etc.

Existen cosas que no se puede realizar con SAX por ejemplo: reordenar elementos en un documento XML, validar un documento XML, resolver referencias cruzadas entre elementos, etc.

1.5.4.4 XSL [10][29][30]

XSL (*eXtensible Stylesheet Language – Lenguaje Extensible de Hojas de Estilo*) es un lenguaje extensible de hojas de estilo que permite definir la presentación del contenido de documentos XML.

XSL se divide en: XSLT, XSLT-FO y XPath.

1.5.4.4.1 XSLT

XSLT (*XSL Transformations – Transformaciones XSL*) es un lenguaje que permite transformar documentos XML en otro tipo de documentos no necesariamente de la familia XML, por ejemplo HTML, PDF, etc.

Para efectuar las transformaciones se usa una o más plantillas predefinidas (indican las partes que se deben convertir) que son ingresadas en conjunto con el documento fuente hacia un procesador XSLT. Las partes del documento fuente que coincidan

con la(s) plantilla(s) serán las que se deba transformar para el obtener el tipo de documento deseado.

1.5.4.4.2 XSL-FO

XSL-FO (*XSL Formatting Objects – XSL para Formato de Objetos*) es un lenguaje que permite dar formato al contenido de documentos XML para que se los pueda presentar en pantalla, papel u otro tipo de medios. La aplicación más común de XSL-FO es para presentar el contenido de un documento XML en un archivo PDF.

XSL-FO define archivos de extensión .fo, .fob ó .xml los cuales son documentos XML que contienen la información de salida y/o los formatos que se va a aplicar. Un documento XSL-FO se puede generar directamente de tal modo que contenga los datos y formatos, ó partir de un documento XML mediante XSLT utilizando una hoja de estilos adecuada; en este último caso sólo los formatos están incluidos en el documento XSL-FO.

Para realizar el formateo de los datos XSL-FO define un conjunto de objetos de formato que representan páginas, tablas, párrafos, títulos, gráficos, etc.

1.5.4.4.3 XPath

XPath es un leguaje que permite buscar información dentro de un documento XML. También, XPath permite definir qué partes de un documento XML son las que se deben transformar para obtener un nuevo documento usando XSLT, XPath contiene librerías de funciones estándar y utiliza expresiones de ruta para navegar dentro de documentos XML.

Las expresiones de ruta son direcciones de ubicación de nodos (elementos, atributos, comentarios, *namespaces*, etc.) mediante los cuales se puede encontrar

cualquier información en un documento XML. XPath trata al documento XML como un árbol de nodos.

Las funciones estándar están relacionadas a: comparaciones de valores de caracteres, valores numéricos, valores de fecha y tiempo; manipulación de nodos; etc.

1.5.4.5 XQuery [10][31]

XQuery es un lenguaje para consultar datos XML, no solo en documentos XML sino también en todo aquello que pueda contener información XML, por ejemplo bases de datos.

XQuery está construido con expresiones de ruta XPath para realizar las operaciones de consulta. Por tanto, XQuery también considera a un documento XML como un árbol estructurado de nodos.

Dentro de las aplicaciones más comunes de XQuery están: extracción de información para uso en servicios web, generación de reportes de resumen, búsqueda de información relevante en documentos web, etc.

1.5.5 HERRAMIENTAS PARA EL DESARROLLO DE APLICACIONES XML [10] [32]

1.5.5.1 Editores XML

Un Editor XML es una herramienta que permite desarrollar o editar documentos XML. Un Editor XML puede ser cualquier editor de texto convencional (por ejemplo, Notepad); sin embargo, se recomienda que se use uno profesional. Un editor XML profesional tiene funcionalidades que permiten crear documentos libres de errores de sintaxis, permiten validar documentos XML, y permiten forzar al usuario al desarrollo de estructuras XML válidas.

Existen varias clases de editores XML profesionales, unos ofrecen mejores prestaciones en cuanto a representación gráfica (en forma de árbol) del documento, mientras que otros, sobre una representación normal del documento ofrecen funcionalidades sencillas pero útiles. También existen editores que permiten construir documentos XML bien formados o documentos XML válidos. Como ejemplos de editores XML profesionales están: XML Notepad, Altova XMLSpy, Arbortext, XML Studio, Eclipse, etc.

1.5.5.2 Editores DTD

Un Editor DTD es un editor XML que permite crear o modificar documentos XML válidos siguiendo la estructura de declaraciones de contenido especificado por DTD. Algunos ejemplos de Editores DTD: TDTD, Altova XMLSpy DTD Editor, Exchanger XML Lite, EZDTD, etc.

1.5.5.3 Analizadores XML

Son herramientas que sirven para chequear si la sintaxis y estructura de un documento XML son correctas; es decir si un documento XML es un documento bien formado. También existen analizadores de documentos válidos que se basan en las reglas de sintaxis XML y declaraciones DTD para efectuar la validación. Ejemplos de analizadores XML: MSXML, LTXML, GNU JAXP, Piccolo, etc.

1.5.6 XML EN LA ADMISTRACIÓN DE REDES [10][33]

El propósito de incluir XML en la administración de redes es superar algunos inconvenientes que tiene SNMP pese a las mejoras que presenta su última versión. Éstos inconvenientes son:

- Inexistencia de formatos estándar para el procesamiento y almacenamiento de información de gestión.
- Ineficiencia en la transferencia de datos de gran tamaño.

- Dificultad en la creación de ciclos eficientes de desarrollo de software debido a que SNMP es una tecnología de bajo nivel.

XML tiene la capacidad de resolver estos problemas debido a que:

- XML es un lenguaje diseñado para realizar estructuración bien definida de la información.
- XML permite almacenar y transportar grandes cantidades de información.
- XML dispone de varios lenguajes y herramientas de alto nivel totalmente compatibles entre sí.

La mejor forma como se podría incluir XML sería a manera de complemento al protocolo de administración ya existente. Esto significa, que no es necesario reemplazar todo el sistema basado en SNMP por uno basado en XML. La tecnología XML podría ir en el lado de la estación de gestión con lo cual se evitaría elevados costos por concepto de migración total de tecnología.

1.5.6.1 XML para representar información de administración

SNMP define la codificación de información de administración para uso interno del protocolo pero no para almacenamiento y procesamiento. XML puede codificar la información para todos éstos requerimientos. Además, teniendo la información de administración representada en formato XML, se puede realizar transferencias de grandes cantidades de datos en un solo archivo y se puede facilitar las operaciones de depuración (programador) y procesamiento (máquina) de la información.

1.5.6.2 HTTPS para trasporte seguro de la información de administración [34]

HTTPS (*HyperText Transfer Protocol Secure – Protocolo Seguro de Transferencia de Hipertexto*) puede ser usado para transferir de forma segura información de administración (incluso si ésta tiene formato XML). Esto hace posible realizar

configuraciones remotas de dispositivos de manera protegida, y sin que la información de administración que viaja sea revelada a individuos no autorizados.

HTTPS es una combinación de HTTP y SSL/TLS. TLS (*Transport Layer Security – Seguridad en la Capa Transporte*) y su antecesor (*Secure Sockets Layer – Capa de Sockets Segura*) son protocolos criptográficos que proveen seguridad a nivel de capa transporte. TLS y SSL cifran segmentos cuyos datos provienen de la capa aplicación y de esta manera se garantiza la seguridad en la transmisión extremo a extremo en la capa transporte.

1.5.6.3 DOM y SAX para acceder a la información de administración

Una vez que la información de administración esté representada en formato XML, ésta puede ser fácilmente accedida y manipulada por la aplicación de administración de red utilizando DOM y SAX.

1.5.6.4 XSL para procesar información de administración

XSLT puede utilizarse para convertir la información de administración que está en formato XML a otro tipo de formato, por ejemplo: páginas web HTML, documentos PDF, documentos en texto plano, etc. XPath puede utilizarse para filtrar ciertos datos con el objetivo de lograr estas conversiones.

1.5.6.5 XSD para definir estructura de información de administración

Debido a que XSD define de manera precisa la estructura y contenido de documentos XML, se lo puede utilizar para convertir las MIBs SMI a formato XML. Mediante esta conversión se puede mejorar la integridad de los datos de gestión, es decir que las operaciones de edición ejecutadas sobre los datos sean adecuadas.

1.6 TECNOLOGÍAS Y HERRAMIENTAS DE DESARROLLO WEB

En este tema se tratan algunas tecnologías que Java propone para el desarrollo de páginas web, éstas son: *JavaServer Pages* (JSP), *Java Servlets* y *Tomcat*. Además, se tratan las siguientes herramientas de desarrollo: AdobeDreamweaver e IDE Eclipse.

A continuación, una breve descripción de Java y las tecnologías mencionadas.

1.6.1 JAVA [36]

Java es un lenguaje de programación desarrollado por *Sun Microsystems* (la cual ahora es sucursal de *Oracle Corporation*). Principalmente se caracteriza por ser un lenguaje de programación orientado a objetos, robusto, seguro, independiente de la arquitectura, de alto rendimiento, mutitarea, dinámico, portable y de código abierto.

La mayoría de tecnologías de Java han sido desarrolladas bajo licencia GNU GPL[52] lo cual facilita la adiquisición de las mismas para el desarrollo de aplicaciones de cualquier tipo, especialmente web.

1.6.1.1 El Entorno de desarrollo de Java.

Existen varios programas que permiten desarrollar código Java. Los más comunes son: JDK e IDE.

JDK (*Java Development Kit – Herramienta de Desarrollo Java*) es distribuído gratituamente por la compañía Oracle. JDK es un conjunto de programas y librerías que permiten desarrollar, compilar y ejecutar programas en Java. JRE (*Java Runtime Environment – Ambiente de Ejecución de Java*) es una versión simplificada de JDK la cual permite sólo ejecutar código Java (no permite compilar).

[52] GNU General Public License (GNU GPL o simplemente GPL) es la licencia de software libre más usada en todo el mundo.

Los IDEs (*Integrated Development Environment - Ambiente de Desarrollo Integrado*) son entornos de desarrollo integrados; es decir, un mismo programa permite escribir el código Java, compilarlo y ejecutarlo sin tener que cambiar de aplicación. Los IDEs permiten desarrollar aplicaciones de forma rápida ya que a menudo incorporan librerías como componentes ya desarrollados, los cuales se incoporan al proyecto en desarrollo.
Generalmente, los IDEs utilizan a JDK para realizar sus funciones.

1.6.1.2 El compilador de Java

El compilador de Java es una de las herramientas incluídas en el JDK. El compilador se denomina *javac.exe*. Este compilador realiza un análisis de sintaxis del código escrito en los ficheros fuente (archivos con extensión .java) y si no encuentra errores entonces el código genera los ficheros compilados (archivos con extensión .class), caso contrario muestra las líneas de código que contienen errores.

1.6.1.3 JVM (Java Virtual Machine)

JVM es el intérprete de Java. JVM es quien interpreta y ejecuta los *bytecodes* (ficheros compilados con extensión .class) convirtiéndolos en código particular de la CPU utilizada.

Un JRE contiene: la JVM, librerías de clases y otros ficheros. La JVM es quien realmente ejecuta el programa utilizando librerías y demás ficheros que JRE provee.

En Windows JVM realmente se denomina *java.exe*, mientras que en Linux, solamente *java*.

1.6.1.4 PATH y CLASSPATH

PATH y *CLASSPATH* son variables de entorno que indican las rutas de directorios de herramientas de compilación (javac.exe) y ejecución (java.exe), y directorios de

clases, librerías Java (el API de Java) y clases de usuario, respectivamente. Una variable de entorno de un sistema operativo es un valor dinámico cargado en memoria el cual es accedido simultáneamente por varios procesos. Estas variables de entorno son comúnmente usadas en el desarrollo y ejecución de aplicaciones Java.

1.6.2 JSP Y JAVA SERVLETS [30][37]

JavaServer Pages (JSP) y Java Servlets son tecnologías de Java EE[53] que permiten producir contenido web dinámico.

Ambas tecnologías son diferentes pero complementarias. Java Servlets, el primero en aparecer, fue definido como una extensión a un servidor web para producir contenido web dinámico. JSP, en cambio, es una tecnología que apareció luego, capaz de producir el mismo contenido al igual que los Servlets. Sin embargo, la manera en como generan el contenido dinámico es aquello que los diferencia: los Servlets incorporan contenido web dentro de código Java, mientras que los JSPs incorporan código Java en el contenido web.

1.6.2.1 Java Servlets

Como se dijo anteriormente, Java Servlet es un elemento agregado a un servidor web cuya tarea consiste en procesar las solicitudes y dinámicamente constriur respuestas apropiadas. En la práctica, tales respuestas suelen ser devueltas en forma de HTML o XML y son el resultado de una petición HTTP de un usuario vía navegador web.

Un servlet es simplemente una clase Java que hereda comportamiento de la interfaz *javax.servlet.Servlet.* Esta interfaz forma parte del API Java destinado para servlets: *javax.servlet*.

[53] Java Plataform, Enterprise Edition o Java EE es una plataforma de programación para desarrollar y ejecutar software de aplicaciones en lenguaje Java con arquitectura distribuída.

El Java Servlet API está contenido en el paquete *javax.servlet,* el cual define interacciones entre un contenedor web y los servlets. En este paquete existen varias clases e interfaces, entre ellas se destaca la clase *javax.servlet.http.HttpServlet,* la cual permite crear un servlet HTTP, el mismo que se caracteriza por generar respuestas en formato HTML.

Los servlets son compilados y ubicados dentro de un *servlet container* (contenedor de servlets o también denominado contenedor web), el cual es una aplicación Java que gestiona el ciclo de vida de un servlet y maneja comunicaciones a nivel de sockets. El contenedor de servlets puede venir integrado en un servidor web Java o puede ser utilizado como un producto independiente que se integra con un servidor web externo. Un ejemplo de contenedor de servlets es *Apache Jakarta Tomcat*, el cual puede ser usado como servidor web y contenedor de servlets, o puede ser usado solo como contenedor de servlets que se integra a un servidor web Apache.

Los servlets son utilizados comúmente para devolver contenido basado en texto tales como HTML o XML, y también para devolver datos binarios tales como imágenes u objetos Java serializados los cuales son usados para generar respuestas de contenido dinámico.

1.6.2.1.1 Ciclo de vida de un servlet [38]

El ciclo de vida de un servlet consiste de los siguientes pasos:

1) La clase servlet es cargada por el contenedor web durante su fase de arranque.

2) El contenedor web llama al constructor sin argumentos de la clase servlet.

3) El contenedor inicializa al servlet. El método invocado para el efecto es *init( ).* Este método debe ser invocado tan sólo una vez.

4) Después de la inicialización, el servlet puede atender solicitudes de clientes. Cada solicitud es atendida por un subproceso por separado. El contenedor web invoca al método *service( )* para cada solicitud. El método *service( )* determina la clase de la petición que fue realizada y la envía al método apropiado del servlet para que maneje la petición y forme una respuesta adecuada. En el caso de un servlet HTTP, estos métodos pueden ser los siguientes: *doGet( ), doPost( ), doPut( ),* entre otros.

5) Finalmente, el contenedor web invoca al método *destroy( ).* Este método provoca que el servlet suspenda su servicio. Este método también debe ser invocado una sola vez al igual que *init( ).*

1.6.2.2 JavaServer Pages (JSP)

JavaServer Pages (JSP), como se dijo anteriomente, es una tecnología Java que permite generar contenido web dinámico en forma de documentos HTML, XML u otro tipo; al igual que los servlets.

Una página JSP es un archivo de texto que contiene etiquetas (generalmente HTML). Dentro de estas etiquetas hay elementos JSP especiales, los cuales son utilizados para proporcionar la lógica de procesamiento que permite producir contenido dinámico en cada petición del usuario.

Además, dentro de una página JSP es posible insertar código Java, el cual colabora con la lógica de procesamiento de solicitudes; esta característica de JSP comúnmente se denomina ***scriptlets*.**

La tecnología JSP es una extensión de la tecnología Servlet, y por lo tanto una página JSP tiene la capacidad de ser convertida en servlet con el fin de proporcionar

lógica de procesamiento de solicitudes. El servlet producto de esta conversión se denomina **JSP implementation servlet** (servlet de aplicación JSP).

La siguiente figura muestra como el contenedor web traduce y compila una página JSP a servlet de aplicación, el cual es utilizado para procesar todas las solicitudes.

Cuando una solicitud llega, inicialmente ésta es manejada por el servidor web; luego, el servidor web delega la solicitud al contenedor web. El contenedor web traduce los contenidos de la página JSP y la compila para obtener el servlet de aplicación. El servlet de aplicación es utilizado para procesar la solicitud. El contenedor web también tiene la capacidad de verificar si el contenido de la página JSP ha variado, y si ese es el caso realiza una retraducción y recompilación de la misma.

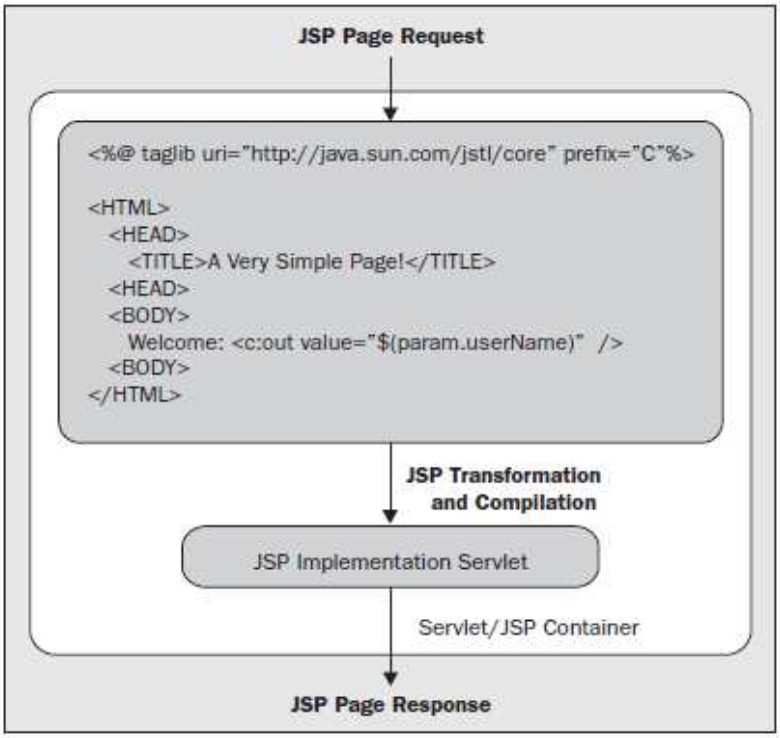


Figura 1. 25 Diagrama de traducción y compilación de una página JSP a servlet.[54]

[54] BROWN S., DALTON S., JEPP D., JOHNSON D., LI S., RAIBLE M.; Pro JSP 2, Cuarta Edición, Figura 1-1 tomada de la pág.36, Acceso: 2 de octubre 2010, 17:00.

1.6.2.2.1 Ciclo de vida de un JSP [39]

El ciclo de vida de un JSP consta de los siguientes pasos:

1) El contenedor web examina la página JSP y verifica que la sintaxis esté correcta. Si la página no tiene errores, entonces se la traduce a una clase Java de implementación (archivo con extensión .java) y luego se la compila obteniendo el servlet de implementación (archivo con extensión .class).

2) El contenedor web carga el servlet de implementación creando una instancia del mismo y lo inicializa invocando el método de inicialización *jspInit( ).*

3) Después de la inicialización, el servlet puede atender solicitudes de clientes. Cada solicitud es atendida por un subproceso por separado. El contenedor web invoca al método *_jspService( )* para cada solicitud. El método *_jspService( )* contiene todas las definciones de *scriptlets*, las cuales realizan el manejo de peticiones y creación de respuestas.

4) Finalmente, el contenedor web invoca al método *jspDestroy( ).* Este método provoca que el servlet suspenda su servicio. Este método generalmente se llama cuando el contenedor web está apagándose o reiniciándose.

1.6.2.3 Arquitecturas de aplicaciones JSP[30][37]

La inserción de código Java en HTML abre la posibilidad de construir páginas web dinámicas; sin embargo, no es la opción más eficiente y eficaz. El problema surge al desarrollar aplicaciones web complejas, el código se torna difícil de mantener y no permite ejercer un trabajo colaborativo entre diferentes áreas de una empresa sobre el mismo proyecto. La solución que existe a estos problemas son los modelos de arquitectura de aplicaciones JSP.

1.6.2.3.1 Modelo de arquitectura 1

Este modelo consiste en que cada página JSP se encargue ella misma de atender las solicitudes y de generar las respuestas. Estas páginas usan una forma de **modelo** para representar la lógica de negocio de la aplicación. Las clases Java denominadas **JavaBeans**[55] son las que se utilizan para modelar la lógica del negocio**.** Luego, los JavaBeans son usados dentro de páginas JSP para presentar las respuestas obtenidas. La siguiente figura muestra cómo todo el procesamiento de solicitudes y generación de respuestas se dan en un solo nivel de la aplicación.

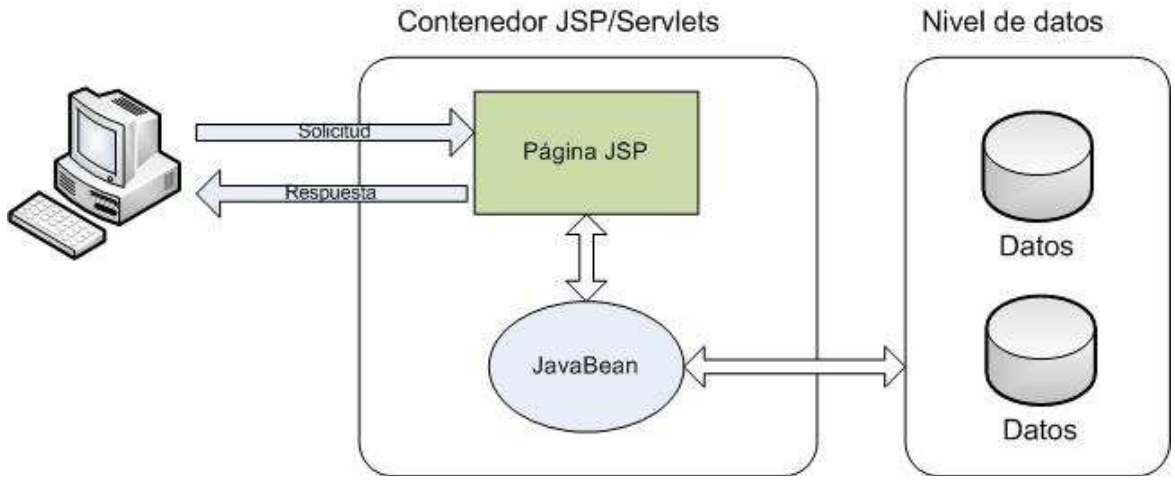


Figura 1. 26 Modelo de arquitectura 1 para aplicaciones JSP.[56]

Este modelo presenta los siguientes problemas:

- **Problemas de mantenimiento:** Debido a que una página JSP es la única encargada de atender solicitudes de clientes, entonces la estructura de la lógica de procesamiento de la aplicación estará incluída dentro de la misma página, esto implica que la página contendrá cientos de líneas de código y esto obviamiente complicará las operaciones de mantenimiento.

[55] Un JavaBean es un componente desarrollado en Java el cual puede ser reutilizado y manipulado visualmente, por ejemplo desde una página JSP. http://en.wikipedia.org/wiki/JavaBean
[56] BROWN S., DALTON S., JEPP D., JOHNSON D., LI S., RAIBLE M.; Pro JSP 2, Cuarta Edición, Figura 1-3 tomada de la pág.9, Acceso: 3 de octubre 2010, 18:00.

- **Problemas de reutilización:** Debido a que toda lógica de procesamiento está embedida en una página JSP en forma de *scriptlets*, no es posible reutilizar funciones comunes.

- **Problemas de seguridad:** Debido a que una página JSP incorpora la lógica de procesamiento para atender todas las solicitudes de usuario, es posible que cualquier acción que requiera que el usuario haya iniciado sesión, termine exponiendo información sensible.

Por lo tanto, el modelo de arquitectura 1 es adecuado sólo para aplicaciones web sencillas, ya que la lógica de procesamiento que contendrán las páginas JSP de la misma manera será simple y todavía posible de mantener.

1.6.2.3.2 Modelo de arquitectura 2 (Modelo-Vista-Controlador)

El modelo de ariquitectura 2 es una aplicación del diseño **MVC** (Modelo-Vista-Controlador). Este modelo refuerza la separación de la lógica de procesamiento de la aplicación de la presentación de la misma.

En este modelo de arquitectura la parte del **modelo** está representada por JavaBeans, la parte de la **vista** está representada por páginas JSP, y la parte del **controlador** está representado por un servlet. El modelo es aquel que forma los datos, la vista es la que presenta los datos, y el controlador es aquel que maneja intercambio de información entre el modelo y la vista. De una forma detallada, el servlet es aquel que: atiende todas las solicitudes de usuario, maneja la lógica de negocio, e instancia los Javabeans; mientras que la página JSP es aquella que: obtiene información de los Javabeans, da formato a la respuesta y la presenta al usuario; y finalmente los Javabeans son aquellos que modelan la información.

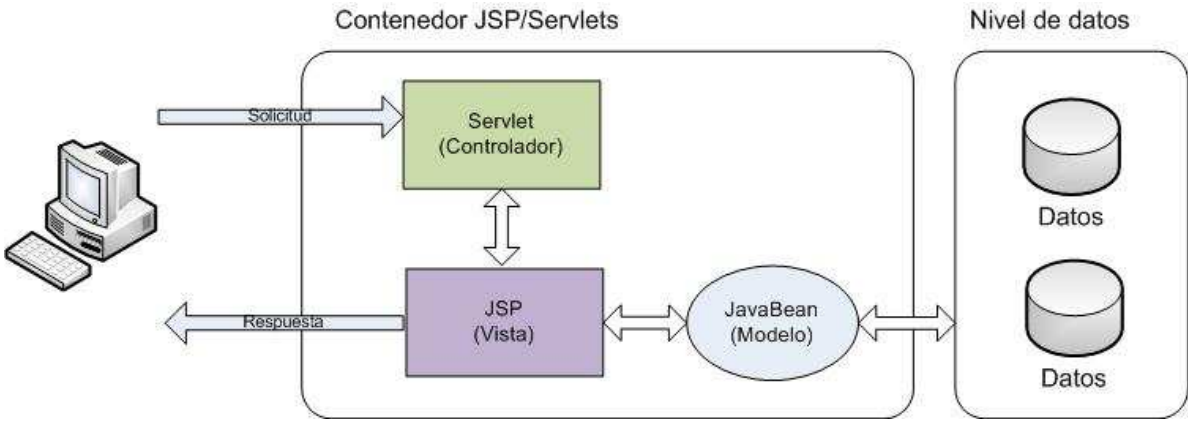


Figura 1. 27 Modelo de arquitectura 2 para aplicaciones JSP.[57]

El modelo de arquitectura 2 permite resolver los problemas que surgían al utilizar el modelo de arquitectura 1; por lo tanto, este modelo es recomendable para construir aplicaciones web complejas.

Otro punto a favor de este modelo es que toda la lógica de procesamiento es centralizada (en el servlet) lo cual ayuda a proveer una solución basada en componentes, los mismos que pueden ser reutilizados y extendidos, contribuyendo así en la flexibilidad, fácil mantenimiento y alto desempeño de la aplicación.

El diseño del Sistema de Monitorización y Administración de Red está basado en el modelo de arquitectura 2.

1.6.2.4 Estructura básica de una aplicación web [40]

Una aplicación web puede ser una colección de los siguientes ítems:

- Servlets.
- Páginas JSP.
- Clases.
- Documentos web estáticos (HTML, imágenes, sonido, etc).

[57] BROWN S., DALTON S., JEPP D., JOHNSON D., LI S., RAIBLE M.; Pro JSP 2, Cuarta Edición, Figura 1-4 tomada de la pág.11, Acceso: 4 de octubre 2010, 15:00.

- JavaBeans.
- Información meta-descriptiva. Esta información indica dependencias (librerías) de la aplicación.

La especificación Java Servlet 2.5 [58] ha definido una estructura especial que todas las aplicaciones web deben seguir para que el contenedor de servlets/JSPs sepa donde encontrar los recursos que componen la aplicación. La estructura especial se refiere a una estructura jerárquica de directorios.

La estructura jerárquica de directorios tiene un documento root dentro del cual se ubican los demás componentes de la aplicación (páginas JSP y demás documentos web estáticos). Por ejemplo, para una aplicación web con un *context path* (nombre que identifica un documento root) denominado */catalogo*, un archivo *index.html* ubicado bajo el elemento root puede ser solicitado mediante la ruta */catalogo/index.html*.

Dentro del documento root también se encuentra un directorio especial llamado *WEB-INF*. En este directorio se ubican recursos de la aplicación que no deberían ser accedidos directamente por un cliente desde un *browser*. El único que puede acceder al contenido de este directorio es el servlet. Los contenidos del directorio *WEB-INF* son los siguientes:

- **/WEB-INF/web.xml:** Es el descriptor de despliegue de la aplicación. Este es un archivo xml que contiene información de: parámetros iniciales de configuración de la aplicación, definiciones y asignaciones de servlets/JSP, parámetros de configuración de servlets, lista de archivos de bienvenida, lista de páginas de error, aspectos de seguridad, entre otros.

[58] La especificación Java Servlet es el documento oficial publicado por Sun Microsystems. Este documento contiene toda la teoría relacionada a los servlets. La útima versión estable es la 2.5.

- **/WEB-INF/classes:** Es el directorio para ubicar servlets y clases (archivos .class). El cargador de clases de la aplicación busca en este directorio para cargar las clases.

- **/WEB-INF/lib/*.jar:** Es el directorio para almacenar archivos JAR (Java Archive – Archivo Java). Estos archivos contienen librerías (pertenecientes a projectos externos) que son reutilizadas por la aplicación.

Otro directorio importante que se ubica en el documento root es el *META-INF.* Dentro de este directorio se encuentra el archivo /**META-INF/MANIFEST.MF**. En este archivo se enlistan los paquetes que son necesarios para el funcionamiento de la aplicación. Este archivo es usado por el contenedor web para verificar dependencia de paquetes. Este archivo generalmente se utiliza cuando la aplicación web se construye con formato WAR.

Si todos los paquetes o librerías están incluídas en el directorio /WEB-INF/lib de la aplicación, entonces no es necesario declarar estas dependencias en el archivo MANIFEST.MF y se lo deja vacío.

1.6.2.5 Formatos de aplicaciones web [37]

La mayoría de contenedores web admiten que las aplicaciones web hayan sido desarrolladas en cualquiera de las siguientes formas:

- **Formato de directorio expandido:** La aplicación web en su estructura predefinida es simplemente copiada al directorio de implementación del contenedor.

- **Formato de archivo WAR (Web Archive – Archivo Java):** La aplicación en su estructura predefinida es comprimida (formato WAR) y luego es copiada al directorio de de implementación del contenedor.

1.6.3 TOMCAT [30]

Tomcat esencialmente es: un servidor web y un contenedor de servlets. Además, Tomcat es un proyecto de la Fundación Apache Software.

1.6.3.1 Arquitectura de Tomcat

La estructura de un servidor Tomcat está definida por el archivo *server.xml.* Este archivo está ubicado en el directorio *conf* de la carpeta de instalación de Tomcat. Concretamente, el archivo *server.xml* define los siguientes componentes:

- **Context:** Representa a un elemento del grupo de componentes de Tomcat denominados *containers.* Este elemento representa a una aplicación web.

- **Connector:** Es un componente asociado a puertos TCP que maneja las comunicaciones entre aplicaciones web y clientes (browsers). A este conector HTTP generalmente se le conoce con el nombre de *Coyote.* Este componente hace que Tomcat sea considerado como servidor web.

- **Host:** Representa a un host virtual. Un host virtual es la asociación de un nombre de dominio con el servidor. Un host virtual puede contener varios *Context* (aplicaciones web). El host virtual que viene por defecto se denomina *localhost.*

- **Engine:** Es un elemento contenedor de *Hosts.* El nombre por defecto asignado al *Engine* es *Catalina.* El Engine Catalina maneja todas solicitudes HTTP recibidas, las reenvía al *Host* y *Context* adecuados, y devuelve las respuestas correspondientes. Catalina representa al contenedor de servlets. Catalina contiene al host *localhost*.

- **Service:** Es el componente que asocia uno o más *Connector* a un *Engine.* Para el Engine Catalina el *Service* se denomina de la misma forma, *Catalina*.

- **Server:** Representa una instancia de Tomcat. Contiene uno o más elementos del tipo *Service.*

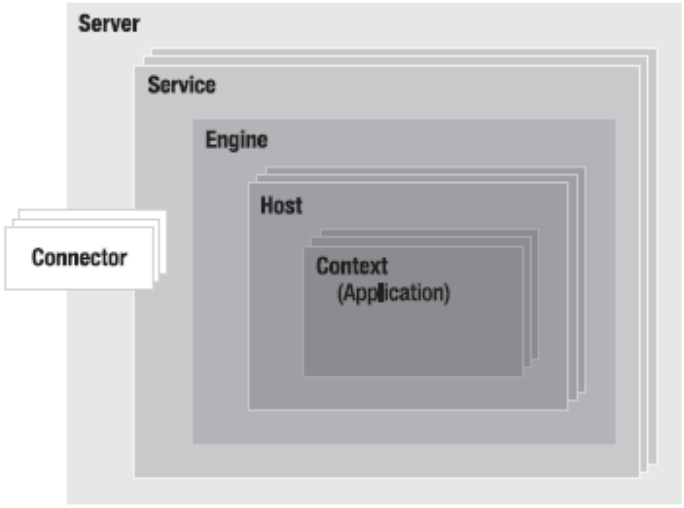


Figura 1. 28 Arquitectura de Tomcat.[59]

1.6.3.2 Estructura de directorios de Tomcat

La carpeta de instalación de Tomcat contiene los siguientes directorios:

- **bin:** Este directorio contiene el archivo ejecutable de Tomcat.

- **conf:** Este directorio contiene los archivos de configuración que se aplican a todas las aplicaciones web. El archivo *server.xml* se ubica en este directorio.

- **lib:** Este directorio contiene librerías .jar comunes a todas las aplicaciones web.

[59] ZAMBON G., SEKLER M.; Beginning JSP, JSF and Tomcat Web Development from Novice to Professional, Figura 7-1 tomada de la pág.213, Acceso: 5 de octubre 2010, 12:30.

- **logs:** En este directorio se almacenan los archivos de *log* (registros de eventos).

- **temp:** En este directorio se guardan archivos temporales.

- **webapps:** Es el directorio base de aplicaciones web de *localhost.* El contenido de cualquier subdirectorio dentro de este directorio es accesible a través del browser. Por ejemplo, una página "index.jsp" contenida dentro de la carpeta "ejemplo" es accesible desde el browser mediante la dirección URL:

  ```
  http://localhost:8080/ejemplo/index.jsp
  ```

- **webapps/ROOT:** Es la carpeta por defecto de aplicación de *localhost.* El nombre asignado a este directorio es una cadena vacía, y por lo tanto para acceder a cualquier recurso que esté ubicado bajo este directorio solo se debe escribir el nombre correspondiente. Por ejemplo, una página "index.jsp" contenida dentro de la ROOT es accesible desde el browser mediante la dirección URL:

  ```
  http://localhost:8080/index.jsp
  ```

- **work:** Es el directorio donde se almacenan las traducciones de páginas JSP a ficheros fuente Java (archivos con extensión .java).

1.6.4 IDE ECLIPSE WTP [41]

Es un entorno de desarrollo de software multilenguaje que integra un entorno de desarrollado integrado (IDE) y un sistema extensible de *plugins*[60].

Eclipse está constituído con plataforma modular. Esto significa que permite agregar capacidades adicionales y plugins. Cada plugin contiene código que provee al IDE

[60] Un plugin es un conjunto de componentes software que agregan una nueva funcionalidad a un programa más grande. http://en.wikipedia.org/wiki/Plug-in_(computing)

alguna funcionalidad extra. Existe un proyecto que extiende las capacidades de Eclipse para el desarrollo de aplicaciones web, el proyecto se denomina **Eclipse Web Tools Plataform Project.**

El proyecto WTP (*Web Tools Plataform*) provee al Eclipse herramientas para desarrollo de aplicaciones Web y Java EE. Esto incluye: editores de código para una variedad de lenguajes, wizards para simplificar el desarrollo de aplicaciones, y herramientas y APIs para soporte de desarrollo, ejecución y depuración de aplicaciones.

Figura 1. 29 Eclipse Web Tools Plataform Project.[61]

1.6.4.1 Características

Entre las características más importantes de Eclipse WTP están:

- WTP cubre una amplia gama de tecnologías, éstas son: XML, CSS, HTML, JavaScript, JSP, XSL, DTD, XSD, Servers, Servlets, Web Services y Java EE.

- Capacidad de agregación de servidores/contenedores web externos al IDE, tales como: Tomcat de Apache, WebSphere de IBM, JBoss, OC4J de Oracle, entre otros.

[61] Ícono de Eclipse WTP, http://www.eclipse.org/webtools/, Acceso: 6 de octubre 2010, 10:30.

- Permite realizar operaciones de control y monitorización de los servidores agregados.

- Permite realizar operaciones de ejecución y depuración de las aplicaciones web en los servidores agregados.

- Facilita la creación de aplicaciones web mediante *wizards* cumpliendo las recomendaciones de las especificaciones de Java Servlets y JSP.

- Permite cambiar el formato común de las aplicaciones web a formato comprimido (WAR). Esto ortoga capacidad de portabilidad a la aplicación.

1.6.4.2 Aplicaciones

Dentro del marco del proyecto, Eclipse WTP es utilizado para:

- Creación, edición y compilación de páginas JSP y HTML.

- Creación y compilación de Servlets.

- Creación y compilación de clases Java.

- Creación y edición de archivos XML, XSD y DTD.

- Creación y edición de archivos JavaScript y CSS.

- Realización de pruebas de ejecución y depuración de la aplicación web.

1.6.5 ADOBE DREAMWEAVER [42]

Dreamweaver es software creado por la compañía Adobe el cual permite diseñar, implementar y mantener aplicaciones web estándar. La última versión estable de Dreamweaver es la *CS 5*.

Figura 1. 30 Adobe Dreamweaver.[62]

1.6.5.1 Características

Entre las características más importantes de Dreamweaver están:

- Soporte de una gran variedad de tecnologías, entre las cuales están: HTML, XSLT, CSS, JavaScript, ActionScript, XML, ASP JavaScript, ASP VBScript, ASP.NET C#, ASP.NET VB, ColdFusion, JSP y PHP.

- Diseño de páginas web en modo visual y en código. Esto agilita el desarrollo de páginas web de modo eficaz.

- Provisión de herramientas que garantizan compatibilidad en los browsers. Cuando se crea una aplicación web, no todos los browsers la interpretan y

[62] Ícono Adobe Dreamweaver, http://2.bp.blogspot.com/_J61AwSJqXd8/S0P0pJBYh7I/AAAAAAAAF-s/5XFF4mFa1YY/s1600/Adobe+Dreamweaver+CS3.jpg.png, Acceso: 6 de octubre 2010, 11:30.

muestran su contenido de la misma manera; por lo tanto, Dreamweaver pone a disposición herramientas que permiten tratar estos problemas de compatibilidad para generar mejores resultados en la aplicación.

- Facilita la integración de archivos multimedia en las aplicaciones web.

- Provisión de asistencia inteligente en la edición de código. Esto ayuda al usuario a minizar errores de sintaxis.

- Permite realizar operaciones de prueba de la aplicación web en varios tipos de browser comerciales: Internet Explorer, Firefox Mozilla, Opera, Safari, etc.

1.6.5.2 Aplicaciones

Dentro del marco del proyecto, Adobe Dreamweaver es utilizado para:

- Diseño y modelaje previo de la interfaz gráfica de la aplicación, es decir de las páginas JSP.

- Comprobación previa de aplicación de hojas de estilo en cascada (CSS) en interfaces de la aplicación.

CAPÍTULO 2. DESARROLLO DEL SISTEMA DE ADMINISTRACIÓN DE RED

En este capítulo se tratan los pasos involucrados en el desarrollo del sistema de administración de red; éstos son: análisis de requerimientos, diseño, e implementación del sistema. El desarrollo del sistema de administración de red está realizado en base a las recomendaciones de UML (*Unified Modeling Language*).

En el análisis de requerimientos se identifica lo que la aplicación va a hacer. Para lograr este objetivo se realiza la recopilación de requerimientos y luego la utilización de dos herramientas: el modelo de casos de uso y el modelo de análisis. La primera herramienta hace una representación de los requerimentos de usuario, y define comportamientos y funcionalidades del sistema. La segunda herramienta hace refinación de los requerimientos anteriomente identificados, lo cual da soporte al siguiente paso que es el diseño del sistema.

En la fase de diseño se define la estructura del sistema para que satisfaga los requerimientos identificados en la fase anterior. Para lograr este objetivo se utilzan dos herramientas: el modelo de diseño y el modelo de despliegue. La primera herramienta define la estructuración física de la aplicación. La segunda herramienta permite describir la distribución física del sistema (hardware).

En la fase de implementación se muestran detalles de la representación física del sistema y de su creación. En esta fase se explica como los diferentes componentes de la fase de diseño se representan en forma de archivos de código fuente, librerías, paquetes, etc.

2.1 ANÁLISIS DE REQUERIMIENTOS DEL SISTEMA [35]

En el análisis de requerimientos se determinan las funcionalidades que la aplicación debe proveer para satisfacer las necesidades de los usuarios. Para lograr esto se hace una recopilación de los requerimientos y luego se utiliza dos tipos de herramientas: el modelo de casos de uso y el modelo de análisis.

2.1.1 RECOPILACIÓN DE REQUERIMIENTOS

Este proyecto tiene como fin desarrollar una aplicación que permita monitorizar elementos de red y controlar la configuración de los mismos, esto a través de consultas y manipulaciones de los valores de los objetos MIB, respectivamente. Además, la aplicación debe proveer los mecanismos de seguridad necesarios para efectuar estas operaciones en modo seguro.

Por lo tanto, en la recopilación de requerimientos se identifican los usuarios que van a interactuar con ésta aplicación y los requisitos que éstos demandan de la misma.

2.1.1.1 Usuarios

Todas las actividades que se realizan en la administración de redes tienen como fin mantener un desempeño óptimo de la red, y por lo general estas actividades están a cargo de él o los administradores de red. Los administradores de red utilizan herramientas de hardware y software para cumplir con estas responsabilidades. Sin embargo, los administradores de red no deben tener las mismas responsabilidades; es decir, existen funciones críticas en la gestión de redes que deberían ser ejecutadas sólo por un administrador general y otras funciones no críticas que podrían ser ejecutadas por cualquier administrador. La razón de esto sería evitar conflictos en el manejo de información de administración de los elementos de red y restringir el acceso a ciertos datos.

La razón de que se consideran distintos tipos de administradores es por el protocolo SNMPv3, éste soporta la creación de usuarios agrupados en grupos que tienen difererentes permisos sobre los objetos MIB; y por tanto es justificable clasificar a los usuarios administradores de red en lo que respecta al proyecto.

Entonces, se identifican dos tipos de usuarios de la aplicación: **Administrador Limitado y Administrador General de Red**.

2.1.1.2 Requisitos funcionales

Los requisitos funcionales especifican los servicios que debe proporcionar el sistema para satisfacer las necesidades de los usuarios.

Si bien es cierto, la monitorización, control y seguridad de red son los requerimientos principales, éstos no permiten identificar claramente que es lo que la aplicación debe hacer para satisfacerlos. Por lo tanto, es necesaria una especificación de requisitos funcionales de los usuarios con mayor detalle.

Para el caso del usuario Administrador General de Red, se tiene los siguientes requisitos funcionales:

- **Inicio de sesión:** La aplicación debe solicitar un nombre de usuario y su respectivo password para poder acceder a la misma. Esta información del usuario será aquella que ha sido registrada en la MIB de cada elemento de red. El password será aquel requerido por el protocolo de autenticación de SNMPv3.

- **Configuración de parámetros:** La configuración debe mostrar al usuario valores de los parámetros de mensajes SNMPv3 y parámetros del servidor; también la aplicación debe permitir la actualización de estos parámetros si el

usuario lo requiere. Estos parámetros sirven para efectuar otras operaciones dentro de la aplicación.

- **Descubrimiento de elementos de red:** El descubrimiento se refiere a la búsqueda de elementos de red existentes dentro de un rango determinado de direcciones IP. Esta búsqueda también implica una clasificación de los elementos en base a la red a la que pertenecen y en base a su disponibilidad del servicio SNMPv3.

- **Visualización de información de administración de un elemento de red:** Este requerimiento se refiere a que la aplicación muestre información de administración de un dispositivo de red. La información importante que interesa conocer del elemento es relacionada a su identificación, es decir: nombre, dirección IP, dirección física, tipo de elemento (si es un computador personal, un router, un switch, etc.) y un ícono representativo del elemento. Además, también interesa conocer información adicional útil, como por ejemplo: características del elemento (sistema), dipositivos, interfaces de red, almacenamiento, software y procesos.

- **Navegación en un grupo MIB:** La aplicación debe muestrar la estructura jerárquica (en forma de árbol) de un grupo MIB específico para que el usuario pueda navegar a través de él, y además muestre las características de cada nodo MIB seleccionado.

- **Ejecución de una operación SNMPv3:** La aplicación debe emitir un tipo mensaje SNMPv3 hacia un determinado elemento de red. El mensaje puede ser del tipo *Get, Set*, *Trap* o *Inform.* La disponibilidad de un tipo de mensaje estará restringida por el tipo de acceso del objeto MIB.

- **Visualización de notificaciones recibidas:** La aplicación debe permitir mostrar y eliminar notificaciones (*Trap* y/o *Inform)* recibidas. Esta información

se actualiza en cada petición del usuario. La disponibilidad de este requerimiento funcional dependerá si existen derechos de acceso que asignen vistas de notificación al grupo al cual pertenece el usuario.

- **Gestión de MIBs:** La aplicación debe permitir la agregación y eliminación de archivos MIB según requiera el usuario. Los archivos MIB son fuente de información utilizada por otros casos de uso.

- **Gestión de cuentas de usuarios:** Este requerimiento se refiere a que la aplicación permita crear y eliminar cuentas de usuarios en la aplicación y en cada elemento de red registrado.

- **Gestión de vistas:** Este requerimiento se refiere a que la aplicación permita crear y eliminar vistas en la aplicación y en cada elemento de red registrado.

- **Gestión de derechos de acceso:** Este requerimiento se refiere a que la aplicación permita crear y eliminar derechos de acceso en la aplicación y en cada elemento de red registrado.

- **Cambio de passwords de usuario**: Este requerimiento se refiere a que la aplicación permita cambiar passwords de usuario de los protocolos de autenticación y/o privacidad en la aplicación y en cada elemento de red registrado.

Para el caso del Administrador Limitado, los requerimientos funcionales son los mismos que para el anterior a excepción de: Gestión de MIBs, Gestión cuentas de usuarios, Gestión de vistas y Gestión de derechos de acceso. La razón es dichos requerimientos funcionales controlan la estabilidad de todo el sistema de administración de red y por lo tanto únicamente deben ser utilizados por un ente central (Administrador general) que los configure adecuadamente.

2.1.1.3 Requisitos no funcionales

Los requisitos no funcionales definen las propiedades y restricciones de la aplicación referidas a las tecnologías utilizadas para su implementación.

Según los objetivos del proyecto, los requisitos no funcionales son los siguientes:

- **Utilización del protocolo SNMPv3 para ejecutar funciones de administración de red:** Únicamente SNMPv3 será el protocolo de administración de red escogido para la aplicación y deberá ser utilizado como: mecanismo de recolección de información en la monitorización de red, mecanismo de establecimientos de valores de objetos MIB en el control de red, mecanismo de envío y recepción de notificaciones, y mecanismo de autenticación, privacidad y control de acceso en la seguridad de red.

- **Utilización del lenguaje de programación Java:** La aplicación deberá ser implementada mediante el lenguaje de programación Java, para que ésta pueda ser instalada en cualquier plataforma que soporte JVM[63] (Concretamente en dos computadores, uno con sistema operativo Linux y otro con Windows XP).[64]

- **Manejo de la aplicación a través de una interfaz web segura:** La aplicación deberá ser del tipo web de manera que la aplicación pueda ser utilizada desde un navegador de Internet de propósito general y además cuente con seguridades criptográficas.

- **Manejo de información por medio de XML:** La información que manejará la aplicación deberá estar representada como archivos XML de tal modo que

[63] JVM (Java Virtual Machine) es un programa disponible para diversas plataformas el cual interpreta y ejecuta instrucciones de un programa desarrollado en Java.

[64] Esto está especificado en el alcance del plan del proyecto.

permita realizar operaciones de almacenamiento y procesamiento de la información.

- **Utilización de IPv4 como protocolo de red:** La aplicación deberá soportar como protocolo de red únicamente a IPv4[65].

2.1.2 MODELO DE CASOS DE USO

El modelo de casos de uso representa los distintos requerimientos que hacen los usuarios al sistema, especificando funcionalidades y comportamiento del sistema al interactuar con usuarios u otros sistemas. Dichas funcionalidades son conocidas como **Casos de Uso**, mientras los que provocan su ejecución se denominan **Actores.** Casos de uso y actores interactúan entre si mediante **Relaciones.**

Para modelar los casos de uso se parte de los usuarios y requisitos funcionales ya descritos.

2.1.2.1 Actores

Un actor es un conjunto de personas o sistemas externos los cuales cumplen un papel determinado e interactúan con el sistema. En base a los usuarios ya identificados anteriormente se definen los siguientes actores: **Administrador Limitado** y **Administrador General.**

Figura 2. 1 Actores

[65] Esto está especificado en el alcance del plan del proyecto.

2.1.2.2 Casos de uso

Un caso de uso representa lo que hace el sistema y como se relaciona con su entorno; es decir, un caso de uso representa a un requisito funcional. Por lo tanto, para definir los casos de uso se require de la inclusión de actores y requisitos funcionales. La definición de casos de uso se realiza utilizando el siguiente formato:

Requisito: Requisito funcional.
Caso de uso: Nombre del caso de uso.
Actor: Actor que participa en el caso de uso.
Descripción: Definición del caso de uso.

Tabla 2. 1 Modelo de definición de casos de uso.

Las definiciones de los casos de uso de la aplicación son las siguientes:

Requisito: Inicio de sesión.
Caso de uso: Iniciar sesión.
Actores: Administrador General y Administrador Limitado.
Descripción: Los actores utilizan este caso de uso para ingresar a la aplicación. La aplicación comprueba el nombre de usuario y password de autenticación.

Tabla 2. 2 Caso de uso Iniciar sesión.

Requisito: Configuración de parámetros.
Caso de uso: Configurar parámetros.
Actores: Administrador General y Administrador Limitado.
Descripción: Los actores utilizan este caso de uso para cambiar los valores de parámetros de mensajes SNMPv3 y parámetros del servidor, los cuales son necesarios para poder ejecutar otras operaciones de la aplicación.

Tabla 2. 3 Caso de uso Configurar parámetros.

Requisito: Descubrimiento de elementos de red.
Caso de uso: Descubrir elementos de red.
Actores: Administrador General y Administrador Limitado.
Descripción: Los actores utilizan este caso de uso para encontrar elementos de red dentro de un rango de direcciones IP. De los elementos encontrados se indica aquellos que tienen activado el servicio SNMPv3 y los que no.

Tabla 2. 4 Caso de uso Descubrir elementos de red.

Requisito: Visualización de información de un elemento de red.
Caso de uso: Visualizar información de elemento de red.
Actores: Administrador General y Administrador Limitado.
Descripción: Los actores utilizan este caso de uso para observar información de administración de un determinado elemento de red. Cada vez que el actor interactúa con este caso de uso se actualiza la información.

Tabla 2. 5 Caso de uso Visualizar información de elemento de red.

Requisito: Navegación en un grupo MIB.
Caso de uso: Navegar en grupo MIB.
Actor(es): Administrador General y Administrador Limitado.
Descripción: Los actores utilizan este caso de uso para navegar a través de la estructura en forma de árbol de un determinado grupo MIB y observar las características correspondientes de cada nodo MIB seleccionado.

Tabla 2. 6 Caso de uso Navegar en grupo MIB.

Requisito: Ejecución de una operación SNMPv3.
Caso de uso: Ejecutar operación SNMPv3.
Actores: Administrador General y Administrador Limitado.

Descripción: Los actores utilizan este caso de uso para emitir un tipo de mensaje SNMPv3 hacia un elemento de red. La disponibilidad de un tipo de mensaje está restringida por el tipo de acceso del objeto.

Tabla 2. 7 Caso de uso Ejecutar operación SNMPv3.

Requisito: Visualización de notificaciones recibidas.
Caso de uso: Visualizar notificaciones.
Actores: Administrador General y Administrador Limitado.
Descripción: Los actores utilizan este caso de uso para observar y/o eliminar las notificaciones (*Traps* o *Informs*) recibidas de los elementos de red registrados. Esta información se actualiza en cada petición del usuario.

Tabla 2. 8 Caso de uso Visualizar notificaciones.

Requisito: Gestión de MIBs.
Caso de uso: Gestionar MIBs.
Actor: Administrador General.
Descripción: El administrador general utiliza este caso de uso para agregar y/o eliminar archivos MIBs según éste los requiera. Estos archivos son fuente de información que se utiliza en otros casos de uso.

Tabla 2. 9 Caso de uso Gestionar MIBs.

Requisito: Gestión de cuentas de usuarios.
Caso de uso: Gestionar usuarios.
Actor: Administrador General.
Descripción: El administrador general utiliza este caso de uso para crear y eliminar cuentas de usuarios. Las cuentas de usuarios son registradas en la aplicación y en la MIB de cada elemento.

Tabla 2. 10 Caso de uso Gestionar usuarios.

Requisito: Gestión de vistas.
Caso de uso: Gestionar vistas.
Actor: Administrador General.
Descripción: El administrador general utiliza este caso de uso para crear y eliminar vistas. Las vistas son registradas en la aplicación y en la MIB de cada elemento.

Tabla 2. 11 Caso de uso Gestionar vistas.

Requisito: Gestión de derechos de acceso.
Caso de uso: Gestionar derechos de acceso.
Actor: Administrador General.
Descripción: El administrador general utiliza este caso de uso para crear y eliminar derechos de acceso. Los derechos de acceso son registrados en la aplicación y en la MIB de cada elemento.

Tabla 2. 12 Caso de uso Gestionar derechos de acceso.

Requisito: Cambio de passwords de usuario
Caso de uso: Cambiar passwords.
Actores: Administrador General y Administrador Limitado.
Descripción: Los actores utilizan este caso de uso para cambiar passwords de los protocolos de autenticación y privacidad. Los cambios serán registrados en la aplicación y en la MIB de cada elemento.

Tabla 2. 13 Caso de uso Cambiar passwords.

2.1.2.3 Diagrama de casos de uso

En los diagramas de casos de uso se representan a los actores, los casos de uso y las diferentes interacciones entre ambos, es decir sus relaciones.

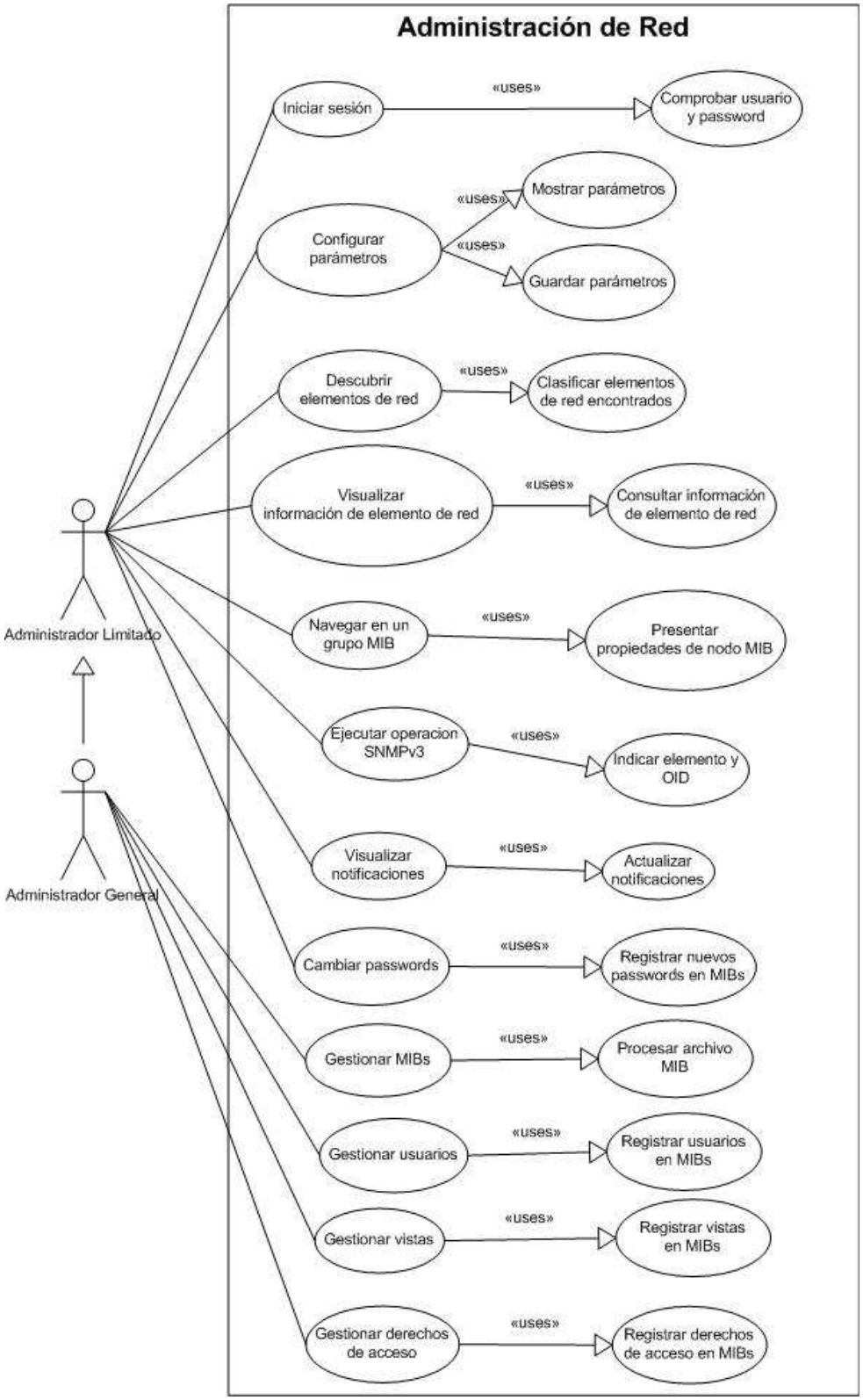


Figura 2. 2 Diagrama de casos de uso de la aplicación.

En la figura 2.2 se presenta al diagrama de casos de uso de la aplicación. En este diagrama también se muestran subcasos de uso los cuales sirven para abstraer más el comportamiento de la aplicación frente a las peticiones de los usuarios. Estos subcasos de uso se obtienen a partir de la descripción de los requisitos funcionales y de la definición de los casos de uso.

La flecha que va desde el Administrador General hacia el Administrador Limitado significa que es una relación de generalización; es decir, Administrador General hereda el comportamiento de Administrador Limitado pero además posee otras funcionalidades que lo diferencian de éste (es decir: gestión de MIBs, de cuentas de usuarios, de vistas y de derechos de acceso).

2.1.2.4 Especificación de requerimientos

La especificación de requerimientos es una descripción básica de la secuencia ideal de los casos de uso y de la interacción de los usuarios con ellos. La especficación de requerimientos sirve para definir las funcionalidades que debe cumplir la aplicación.

La especificación de requerimientos de la aplicación es la siguiente:

1) **Iniciar sesión:** Inicialmente el administrador general usa este caso de uso para ingresar a la aplicación. Si el nombre de usuario y password son correctos se permite el ingreso.

2) **Configurar parámetros:** Una vez que el administrador general haya ingresado a la aplicación, usa este caso de uso para conocer los valores de parámetros de mensajes SNMPv3 y parámetros del servidor, y si se desea cambia dichos valores.

3) **Descubrir elementos de red:** Luego, el administrador general utiliza este caso de uso para descubrir elementos de red que se encuentran activos, y

para clasificarlos por red a la que pertenecen y por disponibilidad del servicio SNMPv3.

4) **Visualizar información de elemento de red:** Consecuentemente, el administrador general utiliza este caso de uso para observar información de administración de un elemento de red seleccionado. El administrador general solicita consultar dicha información.

5) **Navegar en un grupo MIB:** Posteriormente, el administrador general emplea este caso de uso para navegar a través de la estructura en forma de árbol de un determinado grupo MIB y observa las características de cada nodo MIB seleccionado.

6) **Ejecutar operación SNMPv3:** Al momento de seleccionar un nodo MIB desde el caso de uso anterior, el administrador general utiliza el presente caso de uso para transmitir un mensaje SNMPv3 hacia un determinado elemento de red con el fin de recuperar (*Get*), establecer (*Set*) o transmitir (*Trap/Inform*) información de tal nodo MIB.

7) **Visualizar notificaciones:** Luego, el administrador general utiliza este caso de uso para observar notificaciones recibidas de los elementos de red registrados y borrar aquellas que este considere. El administrador general puede solicitar actualización de esta información.

8) **Gestionar MIBs:** Luego, el administrador general usa este caso de uso para agregar nuevos archivos MIBs a la aplicación o para eliminar aquellos archivos MIB que se consideren innecesarios. Estos archivos son procesados para proveer información útil a otros casos de uso.

9) **Gestionar usuarios:** A continuación, el administrador general utiliza este caso de uso para observar cuentas de usuarios existentes, crear nuevas o

eliminar otras (cuentas del tipo administrador limitado). Cada cuenta de usuario es registrada en la aplicación y en los agentes de los elementos registrados.

10) **Gestionar vistas:** Posteriormente, el administrador general utiliza este caso de uso para observar vistas existentes, crear nuevas o eliminar otras. Cualquier cambio será efectuado tanto en la aplicación como en los agentes.

11) **Gestionar derechos de acceso:** A continuación, el administrador general utiliza este caso de uso para observar derechos de acceso existentes, crear nuevos o eliminar otros. Cualquier cambio será efectuado tanto en la aplicación como en los agentes.

12) **Cambiar passwords:** Posteriormente, si el administrador general lo desea utiliza este caso de uso para cambiar sus passwords de autenticación y/o privacidad. Cualquier cambio será efectuado tanto en la aplicación como en los agentes.

En el caso de que el actor sea el administrador limitado, las especificaciones de requerimientos son similiares a excepción de: gestión de MIBs, gestión de usuarios, gestión de vistas y gestión de derechos de acceso.

2.1.2.5 Especificación de casos de uso

La especificación de casos de uso consiste en la descripción de la secuencia de interacciones entre actor y aplicación que se producen dentro de un caso de uso. La especificación de casos de uso se realiza siguiendo el siguiente formato:

Caso de uso: Nombre del caso de uso.
Precondición: Estado del entorno de la aplicación para que sea factible iniciar la ejecución del caso de uso.

Flujo básico: Secuencia de interacciones normales entre el usuario y la aplicación al realizar el caso de uso.
Flujo alternativo: Interacciones entre el usuario y la aplicación cuando no sigue la secuencia básica.
Postcondición: Estado del entorno de la aplicación para que sea factible finalizar el caso de uso.

Tabla 2. 14 Modelo de especificación de casos de uso.

Al igual que en las especificaciones de requimientos, en las especificaciones de caso de uso el primer actor que debe interactuar debe ser Administrador General. Luego de que éste haya creado las cuentas de usuarios, vistas, derechos de acceso y haya agregado los archivos MIBs necesarios, entonces un Administrador Limitado podrá utilizar la aplicación. Por lo tanto, las especficaciones de los casos de uso de la aplicación son las siguientes:

Caso de uso: Iniciar sesión.
Precondición: La cuenta de usuario debe estar creada en la MIB de cada dispositivo de red y en la aplicación. Si es la primera vez que se usa la aplicación, la cuenta del administrador general debe estar ya creada por defecto.
Flujo básico: 1) Al momento de ejecutar la aplicación, el actor interactúa con este caso de uso para poder ingresar a la aplicación. 2) La aplicación muestra la página de Bienvenida la cual tiene un enlace para ingresar a la aplicación. 3) El actor hace clic en el botón Ingresar y accede a la página Inicio Sesión. 4) El actor ingresa su nombre de usuario y su password (el mismo especificado para el protocolo de autenticación) y luego hace clic en el botón Iniciar Sesión. 5) La aplicación comprueba estos valores y si son correctos retorna al usuario la página Principal la cual contiene enlaces a las demás páginas.

Flujo alternativo: En el paso 4 si el resultado de la comprobación es erróneo la aplicación retorna un mensaje de error en la página Inicio Sesión; el actor puede volver a escribir su nombre y password ó puede salir de la aplicación.
Postcondición: La instancia del caso de uso finaliza cuando: se hace posible el ingreso a la aplicación al mostrarse la página Principal o se cierra la aplicación.

Tabla 2. 15 Especificación de Iniciar sesión.

Caso de uso: Configurar parámetros.
Precondición: Los archivos que guardan información de los valores de parámetros de configuración están disponibiles.
Flujo básico: 1) El actor invoca este caso de uso para precisar valores de los parámetros de configuración de acuerdo a sus necesidades. 2) La aplicación muestra al actor los valores actuales de dichos parámetros. 3) El actor observa los valores y si desea realiza cambios. 4) Una vez establecidos los valores de los parámetros la aplicación guarda estos valores en los archivos que almacenan este tipo de información.
Flujo alternativo: En el paso 3 el actor puede cancelar los cambios realizados o establecerlos por defecto.
Postcondición: La instancia del caso de uso finaliza cuando se guardan los valores de los parámetros.

Tabla 2. 16 Especificación de Configurar parámetros.

Caso de uso: Descubrir elementos de red.
Precondición: La interfaz de red está activa.
Flujo básico: 1) El actor invoca este caso de uso para descubrir elementos de red. 2) La aplicación presenta al actor opciones para realizar el descubrimiento de elementos de red; éstas son: especificación de la red sobre la cual se desea obtener sus elementos (identificada por su dirección de red), máscara de

subred y definición de rango de direcciones IP. 3) El actor ingresa la dirección de red, la máscara de subred y el rango de direcciones IP dentro de esa red. 4) La aplicación emite mensajes de red a todo el rango de direcciones IP delimitadas en el paso anterior. 5) La aplicación recibe respuestas de los elementos que se encuentran activos. 6) La aplicación emite mensajes SNMPv3 a los elementos activos para identificar a aquellos que tienen capacidad de ser gestionados mediante este protocolo y los que no. 7) La aplicación guarda temporalmente los resultados y los muestra al actor. 8) La aplicación da la opción al actor de guardar los resultados obtenidos o descartarlos.
Flujo alternativo: En el paso 5 si los elementos de red no responden a los mensajes de red enviados por la aplicación se finaliza el caso de uso.
Postcondición: La instancia del caso de uso finaliza cuando: los elementos no responden a los mensajes de red, la conexión de red se corta, el actor guarda o descarta los resultados.

Tabla 2. 17 Espeficicación de Descubrir elementos de red.

Caso de uso: Visualizar información de elemento de red.
Precondición: Está disponible información guardada de los elementos de red identificados en el caso de uso anterior.
Flujo básico: 1) El actor invoca este caso de uso para observar información de administración de un determinado elemento de red. 2) La aplicación presenta al actor los elementos activos en forma de árbol, agrupados por dirección de red y agrupados por disponibilidad del servicio SNMPv3. 3) El actor navega por el árbol y selecciona un elemento. 4) La aplicación muestra información básica del elemento.

5) El actor observa esta información y solicita consultar información de administración adicional. 6) La aplicación emite mensajes SNMPv3 al elemento seleccionado para obtener la información solicitada. 7) La aplicación recibe respuestas y las presenta al usuario.
Flujo alternativo: En el paso 6 no se puede emitir mensajes SNMPv3 debido los siguientes casos: la conexión de red está caída, el agente del elemento no responde, o el usuario no tiene permisos de lectura sobre las MIBs requeridas.
Postcondición: La instancia del caso de uso termina cuando se muestra la información de gestión del elemento seleccionado.

Tabla 2. 18 Especificación de Visualizar información de elemento de red.

Caso de uso: Navegar en un grupo MIB.
Precondición: Está disponible el archivo MIB requerido.
Flujo básico: 1) El actor invoca este caso de uso para navegar a través de un grupo MIB y observar las propiedades del nodo MIB seleccionado. 2) La aplicación muestra un listado de grupos MIB disponibles, el usuario escoge uno de ellos, la aplicación recupera y lo muestra al actor. 3) El actor navega a través del grupo MIB y selecciona un nodo MIB. 4) La aplicación recupera información de las propiedades del objeto seleccionado y las muestra al actor.
Postcondición: La instancia del caso de uso termina cuando se muestran las propiedades del objeto seleccionado.

Tabla 2. 19 Especificación de Navegar en árbol MIB.

Caso de uso: Ejecutar operación SNMPv3.
Precondición: El elemento de red destino está activo.
Flujo básico: 1) El actor invoca este caso de uso para ejecutar una operación SNMPv3 hacia

determinado elemento de red. 2) La aplicación presenta al actor el listado de elementos de red activos y el grupo MIB mostrado en el caso de uso anterior. 3) El actor escoge el elemento de red (es decir, la dirección IP) y selecciona el OID desde el árbol MIB. 4) La aplicación muestra las diversas operaciones SNMPv3 que se pueden enviar para el nodo MIB seleccionado. Las operaciones pueden ser: *Get, GetNext, GetBulk, Set*, *Trap* o *Inform*. La disponibilidad de cada operación está restringida por el tipo de acceso del nodo MIB. 5) El actor escoge una operación SNMPv3. 6) La aplicación emite el tipo de mensaje SNMPv3 seleccionado hacia el elemento destino. 7) La aplicación recibe respuestas del elemento y las presenta al actor.
Flujo alternativo: En el paso 3 el actor puede proporcionar una dirección IP diferente o un OID diferente ingresándolos manualmente. En el paso 7 si no se recibe respuesta del dispositivo de red se indica fallo en la ejecución de la operación, a menos de que sea un mensaje del tipo *Trap*.
Postcondición: La instancia del caso de uso termina cuando se muestran las respuestas recibidas o cuando el agente del elemento destino no responde.

Tabla 2. 20 Especificación de Ejecutar operación SNMPv3.

Caso de uso: Visualizar notificaciones.
Precondición: La aplicación se encuentra en estado de escucha para receptar notificaciones. Está disponible el archivo de almacenamiento de notificaciones.
Flujo básico: 1) El actor invoca este caso de uso para visualizar notificaciones recibidas. 2) La aplicación recupera información de notificaciones y las muestra al actor. 3) El actor observa las notificaciones y solicita ver el detalle de una. 4) La aplicación recupera información de detalle de la notificación seleccionada y la muestra al actor.

5) El actor observa el detalle y solicita actualización para visualizar notificaciones recientes. 6) La aplicación recupera información de las nuevas notificaciones recibas y las muestra al actor.
Postcondición: La instancia del caso de uso termina cuando se muestran las notificaciones recibidas o cuando se muestra el detalle de una.

Tabla 2. 21 Especificación de Visualizar notificaciones.

Caso de uso: Gestionar MIBs.
Precondición: El archivo que guarda información del listado de grupos MIB está disponible.
Flujo básico: 1) El administrador general invoca este caso de uso para agregar nuevos archivos MIB a la aplicación. 2) El administrador general solicita mostrar el listado de los archivos MIB-XML[66] existentes. 3) La aplicación recupera y muestra el listado al adminstrador general. 4) El administrador general observa los archivos existentes y solicita subir uno nuevo. 5) El actor proporciona el archivo MIB, la aplicación recibe el archivo, ésta lo convierte a formato XML y agrega una nueva entrada en la lista de grupos MIBs. 6) La aplicación muestra al administrador general la lista actualizada de grupos MIB.
Flujo alternativo: En el paso 4 es posible eliminar un archivo MIB-XML en lugar de subirlo. La aplicación elimina el archivo y actualiza la lista de grupos MIB disponibles.
Postcondición: La instancia del caso de uso termina cuando se muestra el listado actualizado de grupos MIB.

Tabla 2. 22 Especificación de Gestionar MIBs.

[66] Los archivos MIB-XML son el resultado de la transformación de un archivo MIB común a un archivo MIB con formato XML; ésto obecede a los requerimientos no funcionales especificados en el ítem 2.1.1.3.

Caso de uso: Gestionar usuarios.

Precondición: La cuenta de usuario de administrador general está previamente creada en la aplicación y en los agentes de los elementos gestionados. Además, en cada uno de los agentes están creados: vistas, grupo y derechos de acceso relacionados al administrador general. El servicio SNMPv3 debe estar activo.

Flujo básico:

1) El administrador general invoca este caso de uso para crear cuentas de usuarios (basados en el modelo de seguridad USM de SNMPv3) en la aplicación y en las MIBs de los agentes de los elementos registrados.
2) El administrador general solicita mostrar el listado de cuentas de usuario existentes.
3) La aplicación recupera y muestra al administrador general el listado.
4) El administrador general observa las cuentas de usuarios existentes y solicita crear una nueva cuenta.
5) La aplicación muestra los atributos que se deben establecer valores para crear un nuevo usuario.
6) El administrador general establece valores a tales atributos. Los atributos son: nombre de seguridad, password de autenticación (esta clave también es usada para iniciar sesión), password de privacidad, grupo VACM y tipo de administrador (general o limitado).
7) La aplicación crea localmente la cuenta y la muestra al administrador general.
8) El administrador general pide exportar la nueva cuenta a los elementos de red deseados.
9) La aplicación muestra el listado de elementos para que sean escogidos.
10) El administrador general escoge los elementos y acepta iniciar la exportación.
11) La aplicación envía mensajes SNMPv3 de establecimiento en nombre del administrador general hacia los elementos de red elegidos.
12) La aplicación recibe respuestas desde los dispositivos de red y presenta al actor un informe de la operación.

Flujo alternativo: En el paso 7 es posible realizar el proceso de exportación de una

cuenta de usuario ya existente. Además, en el paso 7 es posible también que el actor desee no exportar sino anular una cuenta de usuario ya existente; para lo cual de la misma manera selecciona los elementos de red deseados y solicita iniciar la anulación. Luego de esto se procede a eliminar la cuenta de usuario en la aplicación. Antes del paso 10 el actor puede solicitar una presentación del informe de registros de exportación de la cuenta de usuario selecionada para verificar el estado de operaciones anteriores y en base a ello tomar decisiones sobre qué elementos se debe seleccionar para efectuar la exportación o anulación de la cuenta de usuario seleccionada.
Postcondición: La instancia del caso de uso termina cuando se muestra el informe de la operación de exportación o anulación de cuenta de usuario.

Tabla 2. 23 Especificación de Gestionar usuarios.

Caso de uso: Gestionar vistas.
Precondición: La cuenta de usuario de administrador general está previamente creada en la aplicación y en los agentes de los elementos gestionados. Además, en cada uno de los agentes están creados: vistas, grupo y derechos de acceso relacionados al administrador general. El servicio SNMPv3 debe estar activo.
Flujo básico: 1) El administrador general invoca este caso de uso para crear vistas en la aplicación y en las MIBs de los agentes de los elementos registrados. 2) El administrador general solicita mostrar el listado de vistas existentes. 3) La aplicación recupera y muestra al adminstrador general el listado. 4) El administrador general observa las vistas existentes y solicita crear una nueva vista. 5) La aplicación muestra los atributos que se deben establecer valores para crear una nueva vista. 6) El administrador general establece valores a tales atributos. Los atributos son: nombre de la vista, subárbol, máscara y tipo (included o excluded).

7) La aplicación crea localmente la vista y la muestra al administrador general. 8) El administrador general pide exportar la nueva vista a los elementos de red deseados. 9) La aplicación muestra el listado de elementos para que sean escogidos. 10) El administrador general escoge los elementos y acepta iniciar la exportación. 11) La aplicación envía mensajes SNMPv3 de establecimiento en nombre del administrador general hacia los elementos de red elegidos. 12) La aplicación recibe respuestas desde los elementos de red y presenta al actor un informe de la operación.
Flujo alternativo: En el paso 7 es posible realizar el proceso de exportación de una vista ya existente. Además, en el paso 7 es posible también que el actor desee no exportar sino anular una vista ya existente; para lo cual de la misma manera selecciona los elementos de red deseados y solicita iniciar la anulación. Luego de esto se procede a eliminar la vista en la aplicación. Antes del paso 10 el actor puede solicitar una presentación del informe de registros de exportación de la vista selecionada para verificar el estado de operaciones anteriores y en base a ello tomar decisiones sobre qué elementos se debe seleccionar para efectuar la exportación o anulación de la vista seleccionada.
Postcondición: La instancia del caso de uso termina cuando se muestra el informe de la operación de exportación de la vista.

Tabla 2. 24 Especificación de Gestionar vistas.

Caso de uso: Gestionar derechos de acceso.
Precondición: La cuenta de usuario de administrador general está previamente creada en la aplicación y en los agentes de los elementos gestionados. Además, en cada uno de los agentes están creados: vistas, grupo y derechos de acceso relacionados al administrador general. El servicio SNMPv3 debe estar activo.
Flujo básico: 1) El administrador general invoca este caso de uso para crear derechos de

acceso en la aplicación y en las MIBs de los agentes de los elementos registrados.

2) El administrador general solicita mostrar el listado de derechos de acceso existentes.
3) La aplicación recupera y muestra al adminstrador general el listado.
4) El administrador general observa los derechos de acceso existentes y solicita crear un nuevo derecho de acceso.
5) La aplicación muestra los atributos que se deben establecer valores para crear un nuevo derecho de acceso.
6) El administrador general establece valores a tales atributos. Los atributos son: grupo VACM, prefijo de contexto, nivel de seguridad mínimo, match de contexto (*exact* o *prefix*), vista de lectura, vista de escritura y vista de notificación.
7) La aplicación crea localmente el derecho de acceso y lo muestra al administrador general.
8) El administrador general pide exportar el nuevo derecho de acceso a los elementos de red deseados.
9) La aplicación muestra el listado de elementos para que sean escogidos.
10) El administrador general escoge los elementos y acepta iniciar la exportación.
11) La aplicación envía mensajes SNMPv3 de establecimiento en nombre del administrador general hacia los elementos de red elegidos.
12) La aplicación recibe respuestas desde los elementos de red y presenta al actor un informe de la operación.

Flujo alternativo: En el paso 7 es posible realizar el proceso de exportación de un derecho de acceso ya existente. Además, en el paso 7 es posible también que el actor desee no exportar sino anular un derecho de acceso ya existente; para lo cual de la misma manera selecciona los elementos de red deseados y solicita iniciar la anulación. Luego de esto se procede a eliminar el derecho de acceso en la aplicación.

Antes del paso 10 el actor puede solicitar una presentación del informe de registros de exportación del derecho de acceso selecionado para verificar el estado de operaciones anteriores y en base a ello tomar decisiones sobre qué elementos se debe seleccionar para efectuar la exportación o anulación del derecho de acceso seleccionado.
Postcondición: La instancia del caso de uso termina cuando se muestra el informe de la operación de exportación del derecho de acceso.

Tabla 2. 25 Especificación de Gestionar derechos de acceso.

Caso de uso: Cambiar passwords.
Precondición: La cuenta de usuario está previamente creada en la aplicación y en los agentes de los elementos gestionados. El servicio SNMPv3 debe estar activo.
Flujo básico: 1) El actor (cualquiera) invoca este caso de uso para cambiar sus passwords y exportar estos cambios hacia las MIBs de los agentes de los elementos registrados. 2) El actor ingresa los passwords actuales de autenticación y privacidad, también ingresa dos veces[67] los nuevos passwords de autenticación y privacidad. 3) La aplicación guarda localmente los cambios de passwords. 4) El actor pide exportar los cambios de passwords a los elementos de red deseados. 5) La aplicación muestra el listado de elementos para que sean escogidos. 6) El actor escoge los elementos y acepta iniciar la exportación. 7) La aplicación envía mensajes SNMPv3 de establecimiento en nombre del actor hacia los elementos de red elegidos. 8) La aplicación recibe respuestas desde los elementos de red y presenta al actor un informe de la operación.
Flujo alternativo: En el paso 2 si el usuario ingresa incorrectamente los passwords actuales de autenticación y/o privacidad, la aplicación retorna un mensaje de error.

[67] La razón de que ingrese dos veces el nuevo password es para asegurar que el actor esté familiarizado con su nuevo password.

En el paso 4 es posible realizar el proceso de exportación de cambios de passwords pero utilizando passwords anteriores. Esto sucede cuando en una exportación previa no se seleccionaron algunos elementos de red y por lo tanto estos entienden passwords anteriores para realizar cualquier transacción SNMP. Antes del paso 6 el actor puede solicitar una presentación del informe de registros de exportación de cambio de passwords para verificar el estado de operaciones anteriores y en base a ello tomar decisiones sobre qué elementos se debe seleccionar para efectuar la exportación de cambio de passwords.
Postcondición: La instancia del caso de uso termina cuando se muestra el informe de la operación de exportación de cambio de passwords o cuando se ingresa incorrectamente los passwords.

Tabla 2. 26 Especificación de Cambiar passwords.

2.1.3 MODELO DE ANÁLISIS [35]

El modelo de análisis es un nivel de diseño intermedio entre la capa de recopilación de requerimientos y la de diseño. Permite refinar el modelo de casos de uso lo cual ayuda a una mejor descripción de los requerimientos funcionales. Es útil para identificar responsabilidades y requisitos sobre los objetos que participan en los casos de uso.

El modelo de análisis tiene los siguientes componentes: realización de casos de uso-análisis, clases de análisis y paquetes de análisis.

2.1.3.1 Realización de casos de uso-análisis

La realización de casos de uso-análisis es una descripción de cómo se ejecuta un caso de uso determinado en términos de clases de análisis y sus interacciones. La realización de un caso de uso-análisis contiene las clases de análisis, la descripción

textual de la secuencia de sucesos, y el diagrama de interacción de clases de análisis. La siguiente tabla presenta el modelo para la realización de casos de uso-análisis.

Caso de uso: Nombre del caso de uso.
Clases de análisis: Clases de análisis que intervienen en el caso de uso.
Secuencia: Descripción textual de la secuencia de sucesos.
Diagrama: Diagrama de interacción de clases de análisis.[68]

Tabla 2. 27 Modelo de realización de caso de uso-análisis.

Caso de uso: Iniciar sesión.
Clases de análisis: IUBienvenida, IUPrincipal, IUInicioSesion, GestorUsuario, Usuario y SwarServlet.
Secuencia: Administrador Limitado/General solicita el ingreso a la aplicación a través de IUBienvenida pulsando el botón Ingresar (1)[69]. SwarServlet muestra la página IUInicioSesion (2). En IUInicioSesion el actor ingresa sus datos y presiona el botón de Iniciar Sesión (3). SwarServlet valida estos datos para lo cual solicita a GestorUsuario recuperar información de dicho usuario (4-7).[70] Si la validación resulta existosa, entonces SwarServlet muestra la página IUPrincipal la cual contiene enlaces a los servicios que ofrece la aplicación. IUPrincipal presenta el nombre del actor que ha iniciado sesión (8). Ver la siguiente figura.

Tabla 2. 28 Realización de Iniciar sesión.

[68] El diagrama será excluído de la tabla del modelo con el fin de mejorar la presentación de la misma.
[69] Este número hace referencia a la identificación de la interacción mostrada en el diagrama.
[70] Significa que se producen en secuencia las interacciones desde la 4 hasta la 7.

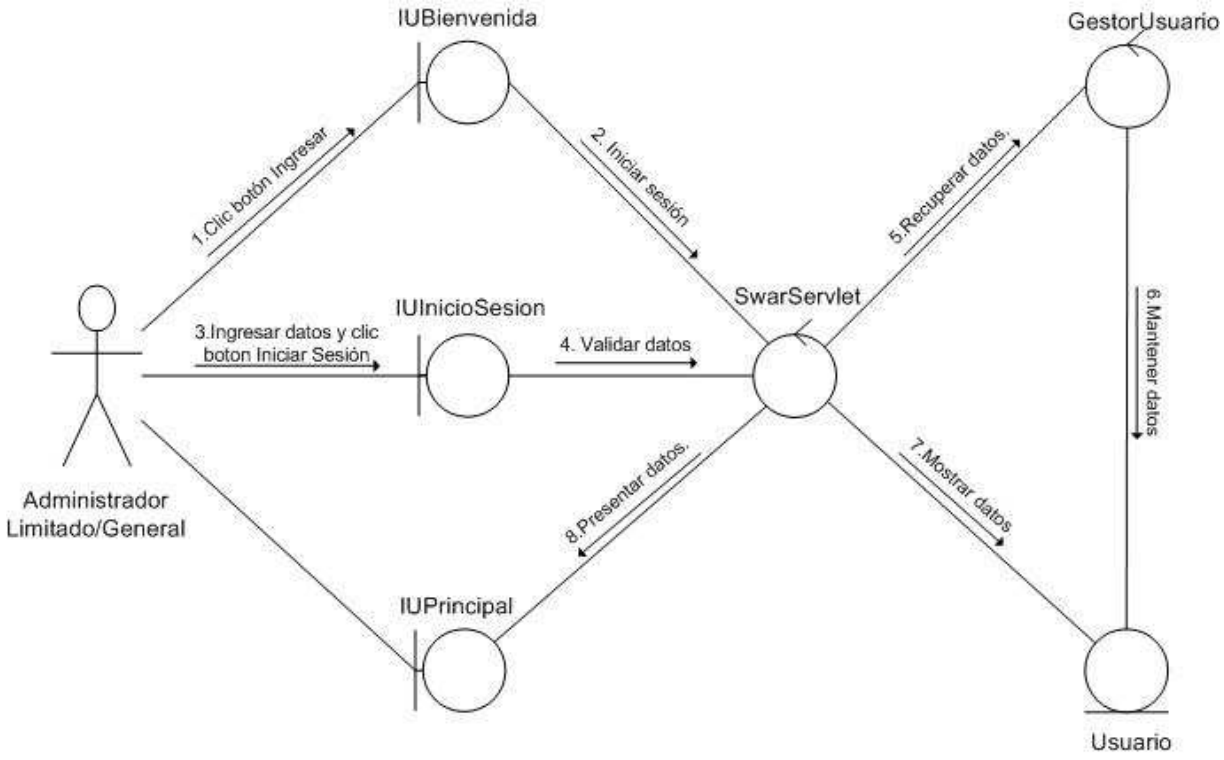


Figura 2. 3 Diagrama de interacción de clases de análisis de: Inicio de sesión.

Caso de uso: Configurar parámetros.
Clases de análisis: IUPrincipal, IUConfigMensajes, IUConfigServidor, GestorMensaje, Mensaje, GestorServidor, Servidor, SwarServlet y SnmpEngine.
Secuencia: Administrador Limitado/General solicita desde IUPrincipal mostrar la página de configuración de parámetros de Mensajes o de Servidor (1). SwarServlet usa a GestorServidor y GestorMensaje para recuperar los parámetros actuales (2-5). SwarServlet asigna a IUConfigMensajes/IUConfigServidor presentar los parámetros obtenidos (6). Luego, el actor solicita editar y guardar los valores de los parámetros (7). SwarServlet utiliza a GestorMensaje y GestorServidor para establecer los nuevos valores y se los exhiben al usuario (8- 12). SnmpEngine guarda en sí mismo los valores de parámetros.

Tabla 2. 29 Realización de Configurar parámetros.

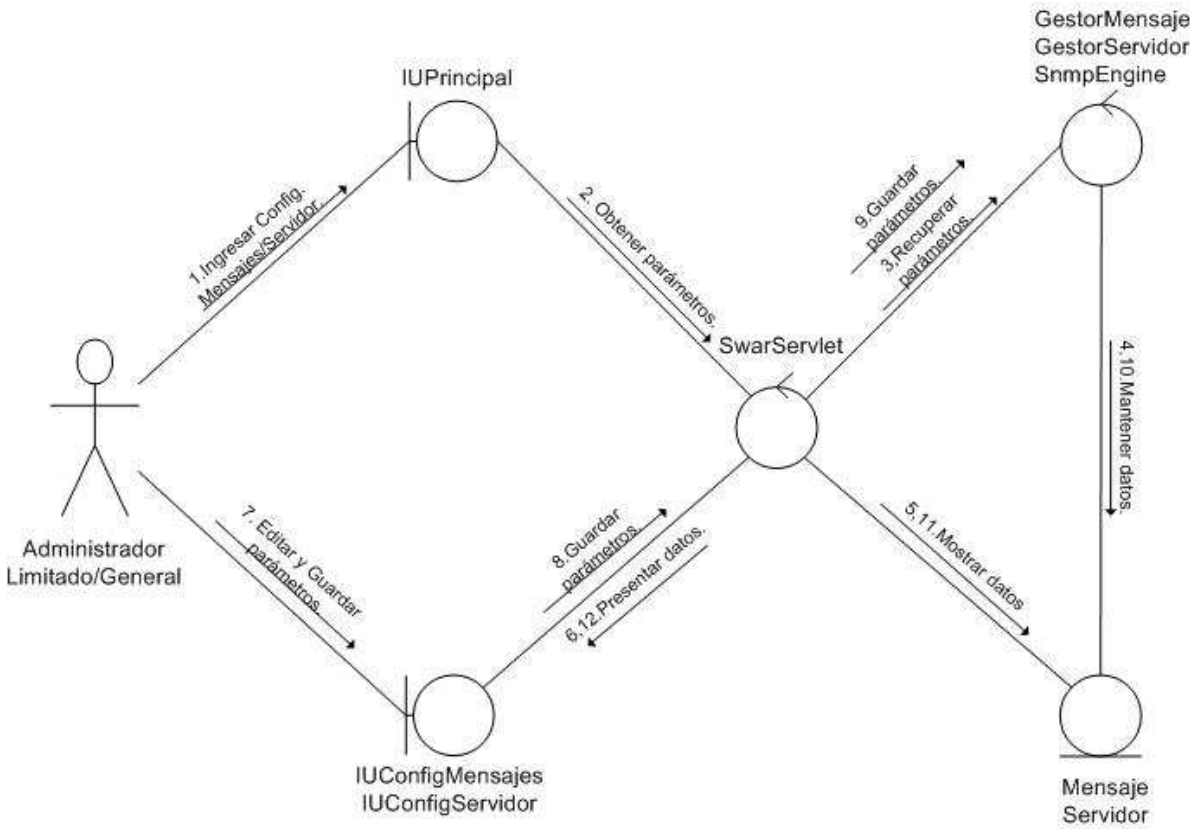


Figura 2. 4 Diagrama de interacción de clases de análisis de: Configurar parámetros.

En la figura anterior, se puede apreciar que existen clases que tienen dos o más nombres (por ejemplo, las clases: IUConfigMensajes y IUConfigServidor); esto representa que hay dos o más clases que siguen la(s) misma(s) interacción(es) con otra clase (en caso del ejemplo, la clase SwarServlet).

Caso de uso: Descubrir elementos de red.
Clases de análisis: IUPrincipal, IUDescubHosts, IUDescubHostsResult, GestorElemento, GestorServidor, GestorMensaje, GestorRed, BuscadorElementos, Elemento, Servidor, Mensaje, Red, SwarServlet, PingUtil y SnmpEngine.
Secuencia: Administrador Limitado/General solicita desde IUPrincipal mostrar la página de descubrimiento de elementos (1). SwarServlet utiliza a GestorServidor para recuperar información actual de los valores de parámetros de configuración de servidor (2-6). Luego, el actor ingresa el rango de direcciones IP y hace clic en Iniciar (7). SwarServlet utiliza a GestorServidor, GestorMensaje y a GestorRed para recuperar información necesaria para efecuar la operación de descubrimiento (8-14). Después, SwarServlet designa a BuscadorElementos la tarea de descubrir hosts en la red (15). BuscadorElementos utiliza a PingUtil para identificar elementos activos (16) y almacena los resultados temporales en Elemento (17). Luego, BuscadorElementos utiliza a Mensaje para efectuar operaciones SNMP (18). Mensaje utiliza a SnmpEngine para enviar el mensaje SNMPv3 hacia los elementos activos (19-20). Luego, IUDescubHostsResult presenta los resultados obtenidos (21-22). Finalmente, el actor solicita guardar los resultados (23). SwarServlet utiliza a GestorElemento y a GestorRed para efectuar esta actividad (24-28).

Tabla 2. 30 Realización de Descubrir elementos de red.

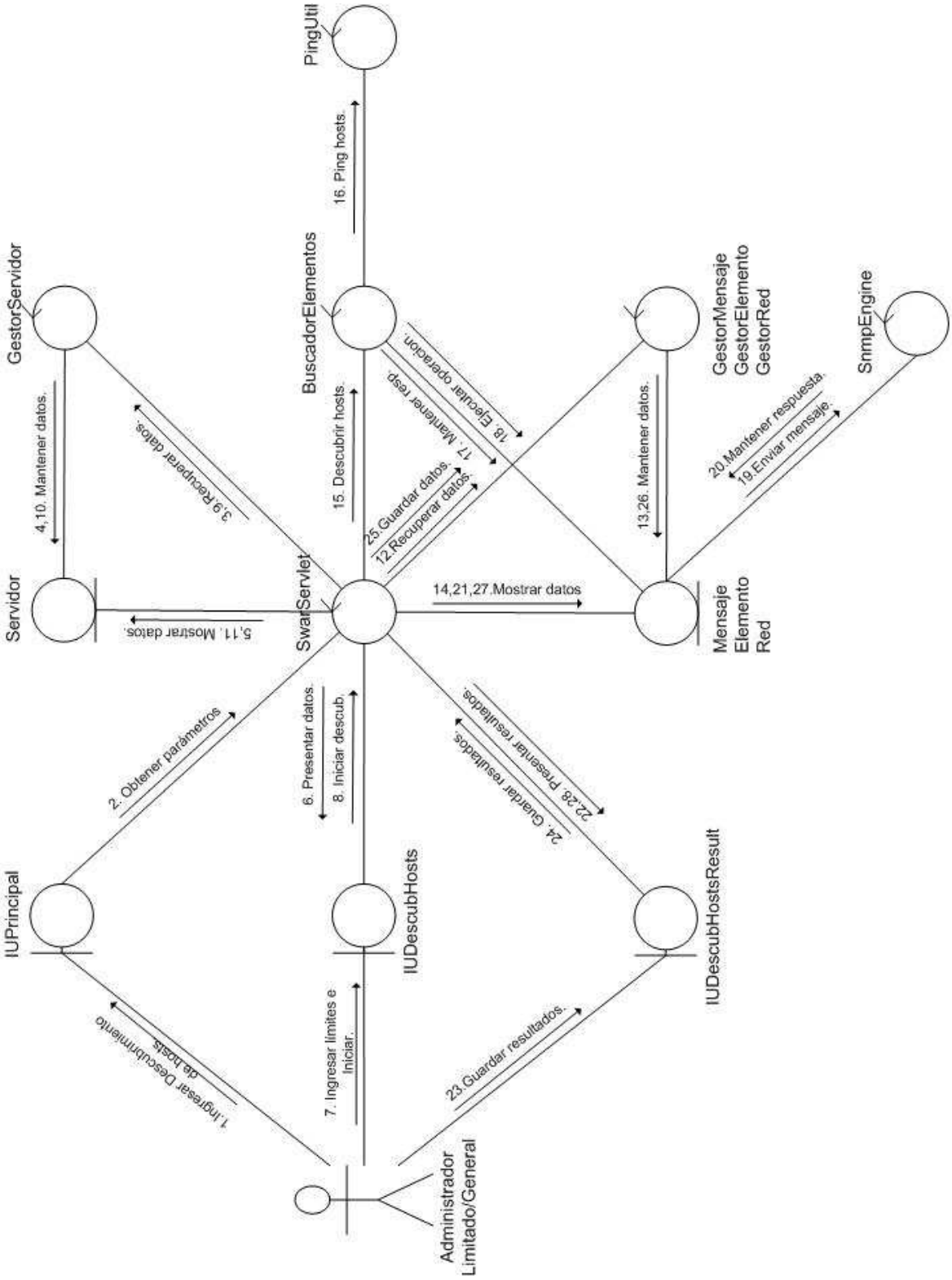


Figura 2. 5 Diagrama de interacción de clases de análisis de: Descubrir elementos de red.

Caso de uso: Visualizar información de elemento de red.
Clases de análisis: IUPrincipal**,** IUElementos, IUInfoElemento, Elemento, Red, Mensaje, GestorElemento, GestorRed, GestorMensaje, SwarServlet y InfoHostTreeFactory.
Secuencia: Administrador Limitado/General solicita desde IUPrincipal mostrar la página de elementos (1). SwarServlet utiliza a GestorElemento y a GestorRed para obtener a elementos y redes registrados (2-5). Luego SwarServlet utiliza a InfoHostTreeFactory para formar el árbol de elementos, organizados por red a la que pertenecen y por capacidad de gestión con el protocolo SNMPv3 (6-7). Luego, el actor solicita mostrar información básica de un elemento de red seleccionado a través de IUElementos (8). SwarServlet utiliza a GestorElemento para presentar dicha información en IUInfoElemento (9-13). Finalmente, el actor solicita mostrar información adicional de un elemento de red a través de IUInfoElemento (14). SwarServlet utiliza a GestorMensaje para obtener Mensaje y éste ejecuta operaciones SNMP para recuperar la información solicitada. Los resultados se presentan en IUInfoElemento (15-22).

Tabla 2. 31 Realización de Visualizar información de elemento de red.

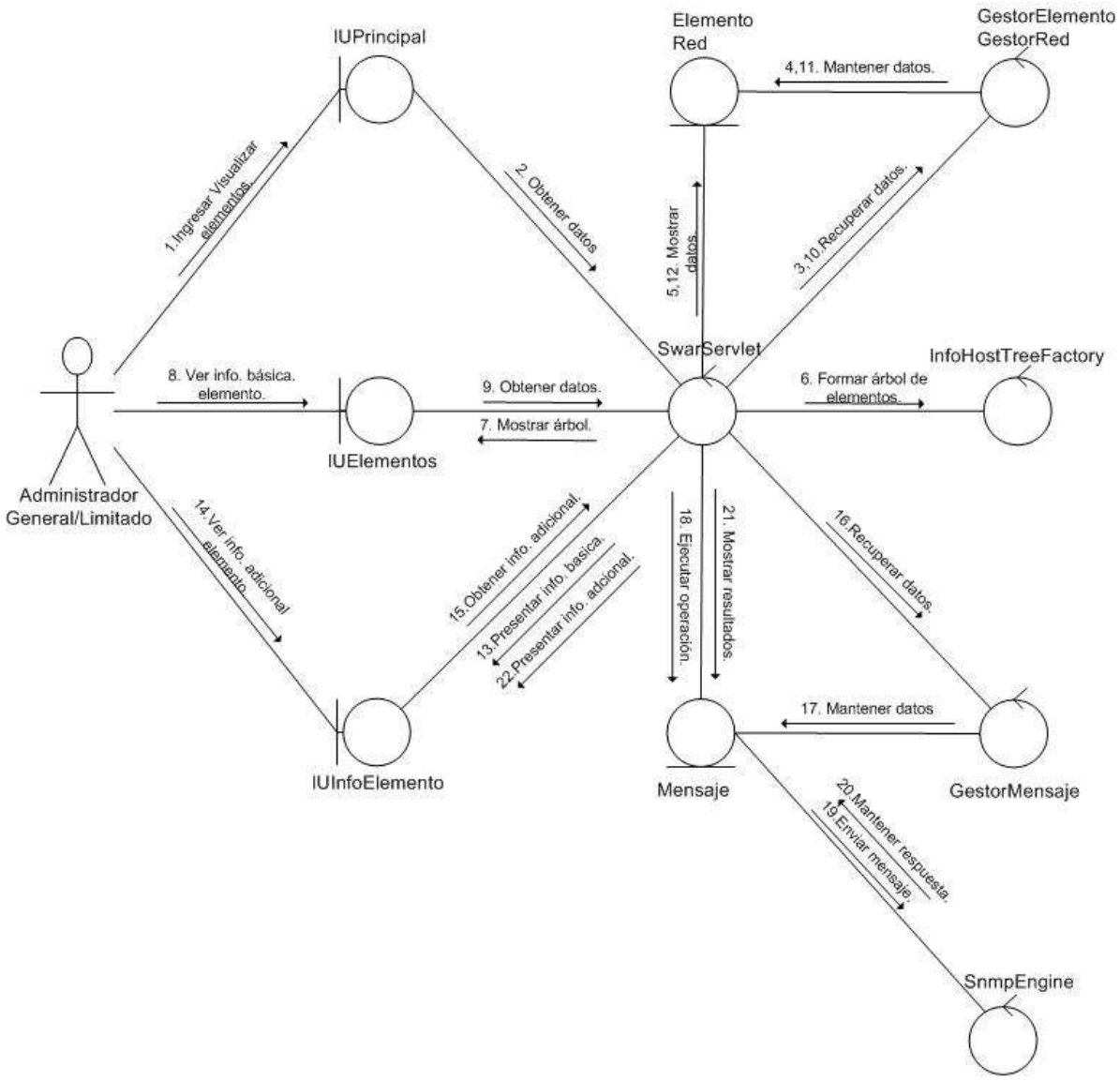


Figura 2. 6 Diagrama de interacción de clases de análisis de: Visualizar información de elemento de red.

Caso de uso: Navegar en un grupo MIB.
Clases de análisis: IUPrincipal, IUGrupoMIB, GestorMIB, MIB, GestorNodoMIB, NodoMIB, SwarServlet y NavMIBTreeFactory.
Secuencia: Administrador Limitado/General solicita desde IUPrincipal mostrar la página IUGrupoMIB (1). SwarServlet utiliza a GestorMIB para recuperar la lista de grupos MIBs y la presenta en IUGrupoMIB (2-6). El actor selecciona el nombre de un grupo y solicita cargar el árbol de dicho grupo (7). SwarServlet utiliza a GestorMIB y GestorNodoMIB para recuperar la información necesaria para formar el árbol (8-14). Luego, SwarServlet asigna a NavMIBTreeFactory la tarea de formar el árbol MIB y lo presenta en IUGrupoMIB (15-16). Luego, administrador limitado/general solicita presentar las propiedades del nodo MIB seleccionado en el grupo MIB actual (17). SwarServlet utiliza a GestorNodoMIB para obtener dicha información y la exhibe al usuario (18-22).

Tabla 2. 32 Realización de Navegar en un grupo MIB.

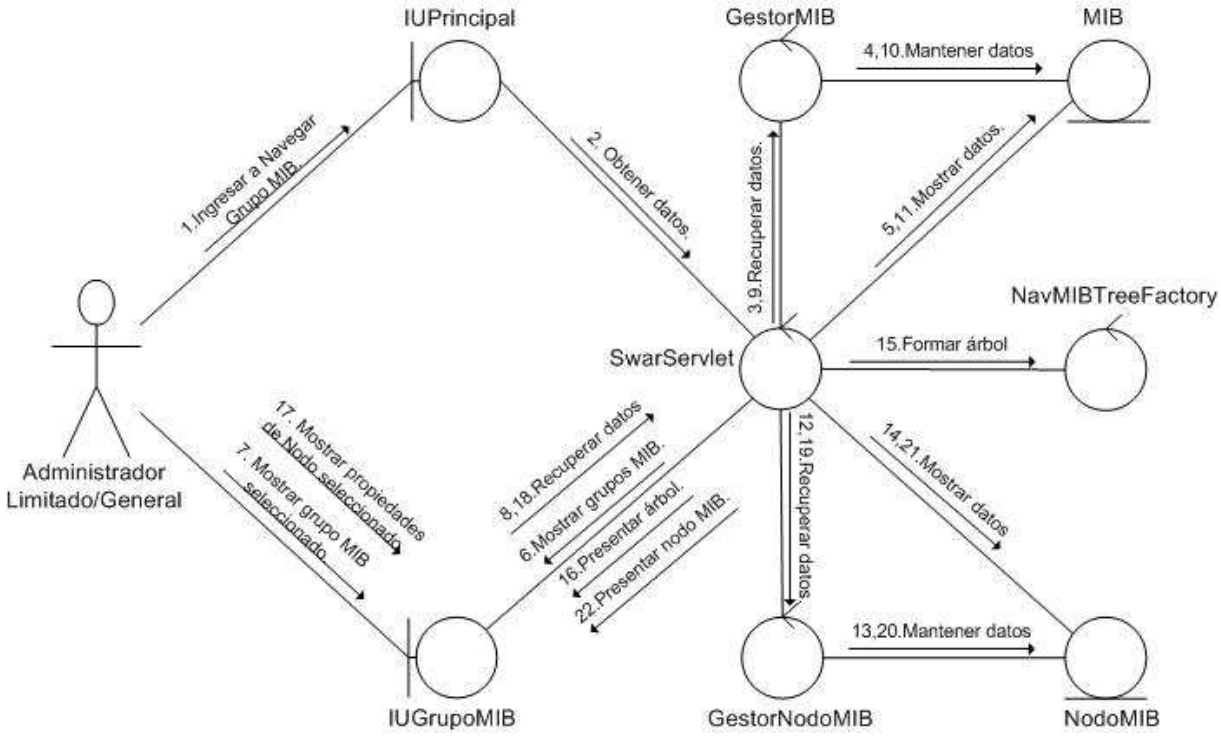


Figura 2. 7 Diagrama de interacción de clases de análisis de: Navegar en un grupo MIB.

Caso de uso: Ejecutar operación SNMPv3.
Clases de análisis: IUGrupoMIB, Elemento, GestorElemento, Mensaje, GestorMensaje, SwarServlet y SnmpEngine.
Secuencia: Administrador limitado/general solicita ejecutar una operación SNMPv3. SwarServlet utiliza a GestorElemento para presentar al actor los elementos de red existentes (1-4). Luego, el actor selecciona un elemento y solicita iniciar la ejecución de una operación SNMP (5). SwarServlet utiliza a GestorMensaje para obtener Mensaje (6-8). SwarServlet pide a Mensaje ejecutar la operación solicitada (9). Mensaje utiliza a SnmpEngine para envar el mensaje SNMP (10-11). Finalmente, el resultado de la operación se muestra en IUGrupoMIB (12-13).

Tabla 2. 33 Realización de Ejecutar operación SNMPv3.

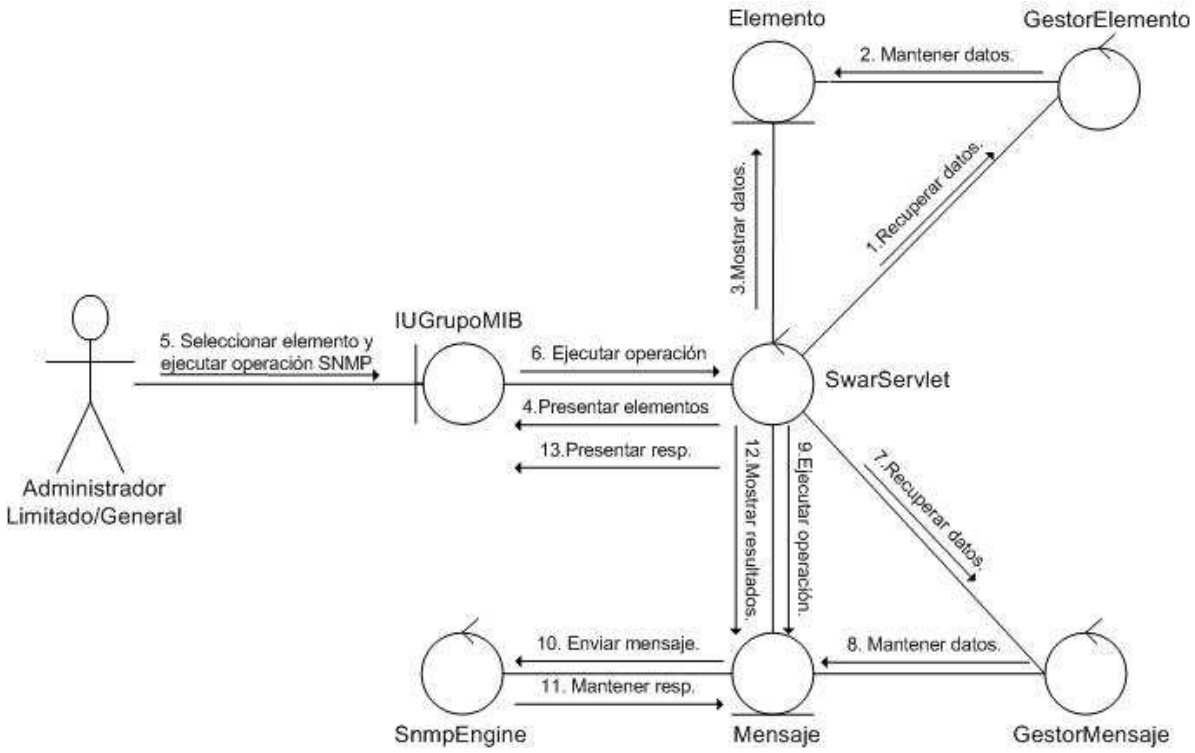


Figura 2. 8 Diagrama de interacción de clases de análisis de: Ejecutar operación SNMPv3.

Caso de uso: Visualizar notificaciones.
Clases de análisis: IUPrincipal, IUNotificaciones, GestorNotificacion, Notificacion, SnmpEngine y ReceptorNotificaciones.
Secuencia: SwarServlet pide a SnmpEngine escuchar la llegada de notificaciones (1). SnmpEngine utiliza a ReceptorNotificaciones para capturar las notificaciones llegadas (2). ReceptorNotifcaciones pide a GestorNotificacion guardar la notificación capturada (3-4). Luego, Administrador limitado/general solicita visualizar las notificaciones recibidas (5). SwarServlet utiliza a GestorNotificacion para obtener las notificaciones recientes y pide a IUNotificaciones que las presente (6-10). Consecuentemente, el actor solicita actualizar para observer nuevas notificaciones (11). SwarServlet usa a GestorNotificacion para ello (12-16). Finalmente, el actor pide mostrar información detallada de una notificación (17). SwarServlet usa a GestorNotificacion para efectuar esta actividad (18-22).

Tabla 2. 34 Realización de Visualizar notificaciones.

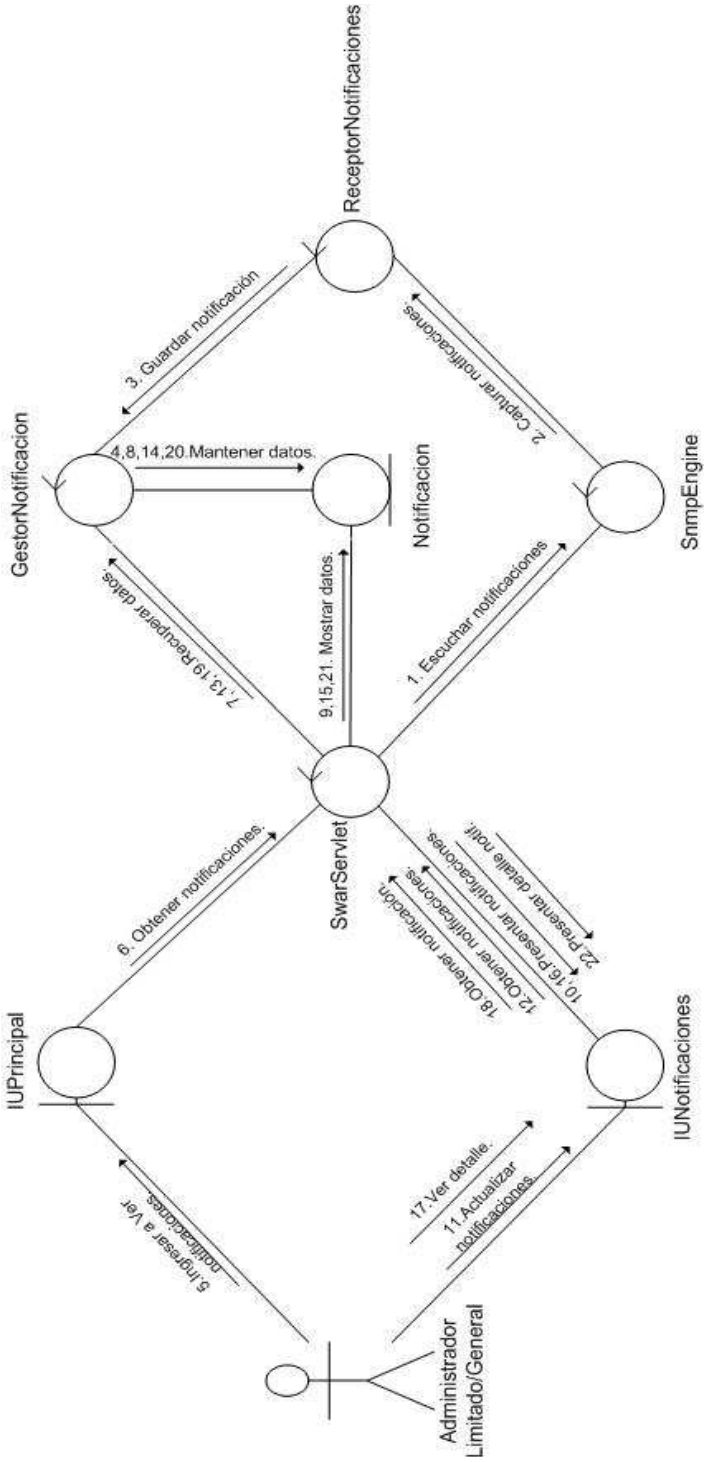


Figura 2. 9 Diagrama de interacción de clases de análisis de: Visualizar notificaciones.

Caso de uso: Gestionar MIBs.
Clases de análisis: IUPrincipal, IUGestionMIBs, SwarServlet, GestorMib, GestorNodoMib, NodoMib y Mib.
Secuencia: Administrador General solicita desde IUPrincipal mostrar la página IUGestionMIBs (1). SwarServlet pide a GestorMIB recuperar la lista de grupos MIB y se la presenta al actor en IUGestionMIBs (2-6). Luego, administrador general solicita subir un nuevo archivo MIB a la aplicación (7). SwarServlet recibe el archivo, lo lee, extrae información y la almacena en MIB y en NodoMIB (8-10). Posterioremente, SwarServlet solicita a GestorNodoMIB la creación de un archivo MIB-XML a partir de los datos anteriormente almacenados (11). SwarServlet utiliza a GestorMIB para agregar una nueva entrada a la lista de grupos MIB disponibles y se presenta al actor la lista actualizada (12-15). Finalmente, administrador general puede pedir que se elimine un archivo MIB (16). SwarServlet utiliza a GestorNodoMIB para eliminar el archivo y a GestorMIB para eliminar el grupo MIB (17-19). SwarServlet utiliza a GestorMIB para recuperar la lista actualizada de grupos MIB y se la presenta al actor (20-23).

Tabla 2. 35 Realización de Gestionar MIBs.

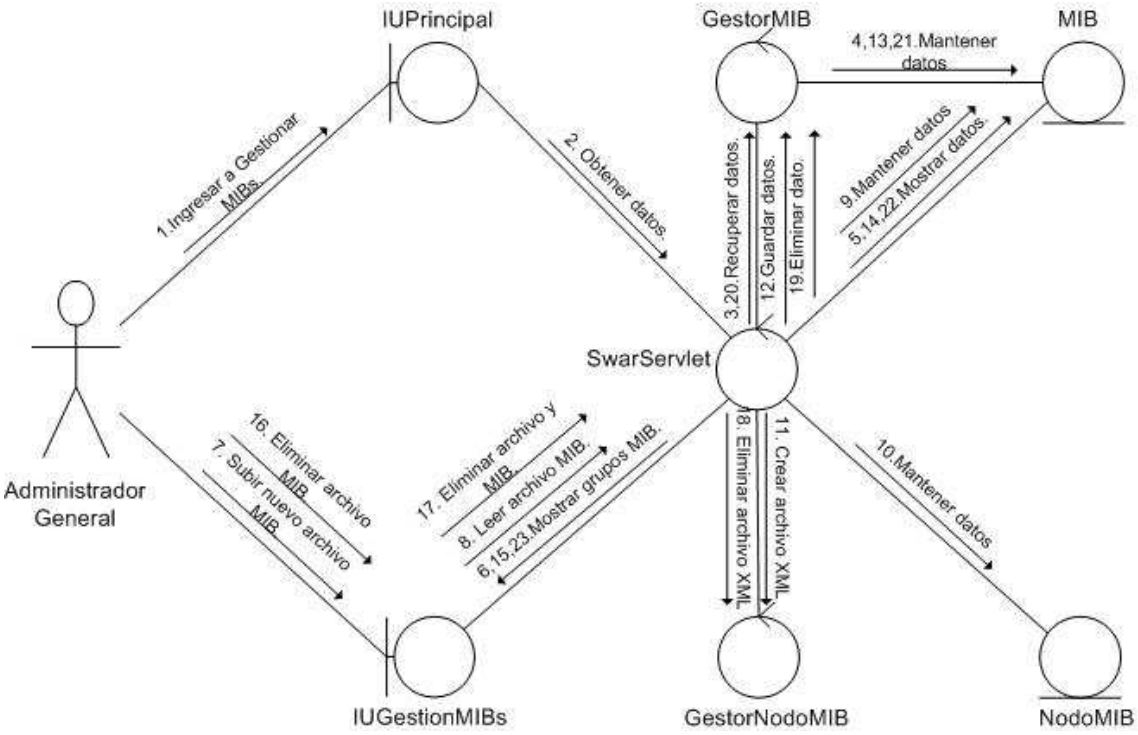


Figura 2. 10 Diagrama de interacción de clases de análisis de: Gestionar MIBs.

Caso de uso: Gestionar usuarios.
Clases de análisis: IUGestionUsuarios, IUExpAnuUsuario, GestorUsuario, Usuario, GestorRed, Red GestorElemento, Elemento, GestorMensaje, Mensaje, GestorGrupoVacm, GrupoVacm, GestorRegistroUsuario, RegistroUsuario, ExportadorUsuario, AnuladorUsuario, SnmpEngine y KeyChangeManager.
Secuencia: Administrador General solicita desde IUPrincipal mostrar la página IUGestionUsuarios (1). SwarServlet pide a GestorUsuario recuperar la lista de usuarios existentes y se la presenta al actor en IUGestionUsuarios (2-6). Administrador general observa la lista de usuarios y solicita guardar uno nuevo (7). SwarServlet utiliza a GestorUsuario y a GestorGrupoVacm para guardar el nuevo usuario y el nuevo grupo, respectivamente (8-11). SwarServlet pide a SnmpEngine agregar el nuevo usuario en su modelo de seguridad USM (12). IUGestionUsuarios presenta la lista de usuarios actualizada (13).

Luego, administrador general pide realizar la exportación del usuario seleccionado (14). SwarServlet pide a GestorUsuario, GestorGrupoVacm, GestorRed y GestorElemento recuperar la información necesaria para realizar la exportación (15-21). IUExpAnuUsuario muestra la información recuperada (22). El actor selecciona los elementos y acepta iniciar la exportación (23). SwarServlet utiliza a GestorMensaje recuperar los parámetros de SNMPv3 (24-27). Luego, SwarServlet asigna a ExportadorUsuario la tarea en si (28-33). ExportadorUsuario guarda los resultados temporalmente en RegistroUsuario (34). Luego, ExportadorUsuario pide a GestorRegistroUsuario guardar los resultados definitivos (35-36). SwarServlet solicita a IUExpAnuUsuario presentar los resultados finales (37-38).

Opcionalmente, administrador general puede pedir que se realice la anulación de un usuario en cada elemento de red para luego eliminarlo de la aplicación (39). Para ello, SwarServlet pide a GestorUsuario, GestorGrupoVacm, GestorRed y GestorElemento recuperar la información necesaria para realizar la anulación (40-46). IUExpAnuUsuario muestra la información recuperada (47). El actor selecciona los elementos y acepta iniciar la anulación (48). SwarServlet utiliza a GestorMensaje recuperar los parámetros de SNMPv3 (49-52). Luego, SwarServlet asigna a AnuladorUsuario la tarea en si (53-58). AnuladorUsuario pide a GestorRegistroUsuario eliminar los registros de usuarios ya anulados (59). Finalmente, SwarServlet solicita a IUExpAnuUsuario mostrar los resultados de la anulación (60-61).

Tabla 2. 36 Realización de Gestionar usuarios.

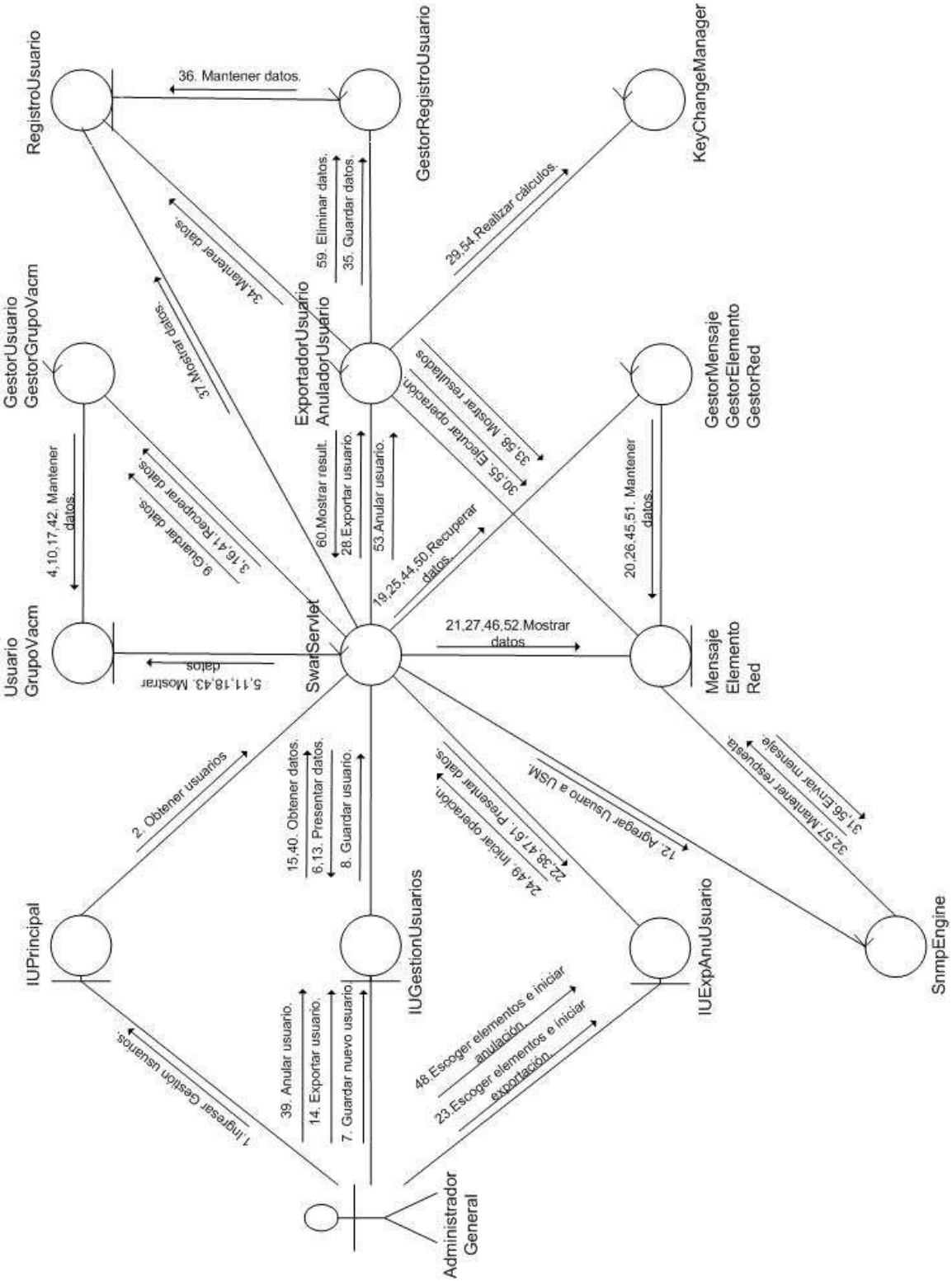


Figura 2. 11 Diagrama de interacción de clases de análisis de: Gestionar usuarios.

Caso de uso: Gestionar vistas.
Clases de análisis: IUGestionVistas, IUExpAnuVista, GestorVista, Vista, GestorRed, Red, GestorElemento, Elemento, GestorMensaje, Mensaje, GestorRegistroVista, RegistroVista, ExportadorVista, AnuladorVista y SnmpEngine.
Secuencia: Administrador General solicita desde IUPrincipal mostrar la página IUGestionVistas (1). SwarServlet pide a GestorVista recuperar la lista de vistas existentes y se la presenta al actor en IUGestionVistas (2-6). Administrador general observa la lista de vistas y solicita guardar una nueva (7). SwarServlet utiliza a GestorVista para guardar la nueva vista (8-11). IUGestionVistas presenta la lista de vistas actualizada (12). Luego, administrador general pide realizar la exportación de la vista seleccionada (13). SwarServlet pide a GestorVista, GestorRed y GestorElemento recuperar la información necesaria para realizar la exportación (14-20). IUExpAnuVista muestra la información recuperada (21). El actor selecciona los elementos y acepta iniciar la exportación (22). SwarServlet utiliza a GestorMensaje recuperar los parámetros de SNMPv3 (23-26). Luego, SwarServlet asigna a ExportadorVista la tarea en si (27-31). ExportadorVista guarda los resultados temporalmente en RegistroVista (32). Luego, ExportadorVista pide a GestorRegistroVista guardar los resultados definitivos (33-34). SwarServlet solicita a IUExpAnuVista presentar los resultados finales (35-36). Opcionalmente, administrador general puede pedir que se realice la anulación de una vista en cada elemento de red para luego eliminarlo de la aplicación (37). Para ello, SwarServlet pide a GestorVista, GestorRed y GestorElemento recuperar la información necesaria para realizar la anulación (38-44). IUExpAnuVista muestra la información recuperada (45). El actor selecciona los elementos y acepta iniciar la anulación (46). SwarServlet utiliza a GestorMensaje recuperar los parámetros de SNMPv3 (47-50). Luego, SwarServlet asigna a AnuladorVista la tarea en si (51-55). AnuladorVista pide a GestorRegistroVista eliminar los registros de las vistas ya anuladas (56). Finalmente, SwarServlet solicita a IUExpAnuVista mostrar los resultados de la anulación (57-58).

Tabla 2. 37 Realización de Gestionar vistas.

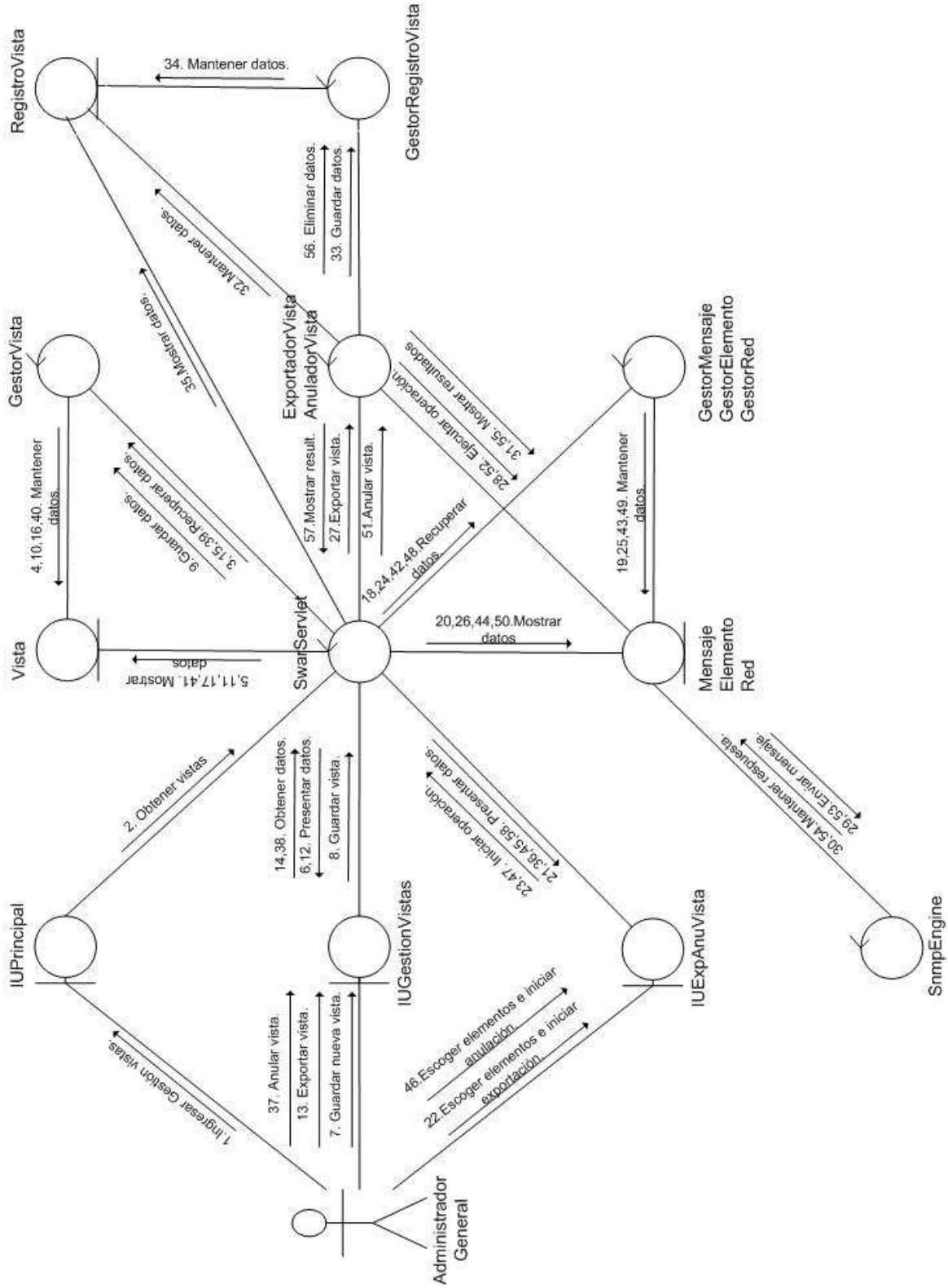


Figura 2. 12 Diagrama de interacción de clases de análisis de: Gestionar vistas.

Caso de uso: Gestionar derechos de acceso.
Clases de análisis: IUGestionDerechos, IUExpAnuDerecho, GestorDerechoAcceso, DerechoAcceso, GestorRed, Red, GestorElemento, Elemento, GestorGrupoVacm, GrupoVacm, GestorVista, Vista, GestorMensaje, Mensaje, GestorRegistroDerecho, RegistroDerecho, ExportadorDerecho, AnuladorDerecho y SnmpEngine.
Secuencia: Administrador general solicita desde IUPrincipal mostrar la página IUGestionDerechos (1). SwarServlet pide a GestorDerecho recuperar la lista de derechos existentes, a GestorGrupoVacm recuperar los grupos y a GestorVista recuperar las vistas; puesto que un derecho de acceso está relacionado con un grupo VACM y con tres vistas (escritura, lectura y notificación). IUGestionDerechos presenta al actor la lista de derechos de acceso (2-6). Administrador general observa la lista de derechos y solicita guardar uno nuevo (7). SwarServlet utiliza a GestorDerecho para guardar el nuevo derecho (8-11). IUGestionDerechos presenta la lista de derechos actualizada (12). Luego, administrador general pide realizar la exportación del derecho de acceso seleccionado (13). SwarServlet pide a GestorDerecho, GestorGrupoVacm, GestorVista, GestorRed y GestorElemento recuperar la información necesaria para realizar la exportación (14-20). IUExpAnuDerecho muestra la información recuperada (21). El actor selecciona los elementos y acepta iniciar la exportación (22). SwarServlet utiliza a GestorMensaje recuperar los parámetros de SNMPv3 (23-26). Luego, SwarServlet asigna a ExportadorDerecho la tarea en si (27-31). ExportadorDerecho guarda los resultados temporalmente en RegistroDerecho (32). Luego, ExportadorDerecho pide a GestorRegistroDerecho guardar los resultados definitivos (33-34). SwarServlet solicita a IUExpAnuDerecho presentar los resultados finales (35-36). Opcionalmente, administrador general puede pedir que se realice la anulación de un derecho de acceso en cada elemento de red para luego eliminarlo de la aplicación

(37). Para ello, SwarServlet pide a GestorDerecho, GestorGrupoVacm, GestorVista, GestorRed y GestorElemento recuperar la información necesaria para realizar la anulación (38-44). IUExpAnuDerecho muestra la información recuperada (45). El actor selecciona los elementos y acepta iniciar la anulación (46). SwarServlet utiliza a GestorMensaje recuperar los parámetros de SNMPv3 (47-50). Luego, SwarServlet asigna a AnuladorDerecho la tarea en si (51-55). AnuladorDerecho pide a GestorRegistroDerecho eliminar los registros de los derechos ya anulados (56). Finalmente, SwarServlet solicita a IUExpAnuDerecho mostrar los resultados de la anulación (57-58).

Tabla 2. 38 Realización de Gestionar derechos de acceso.

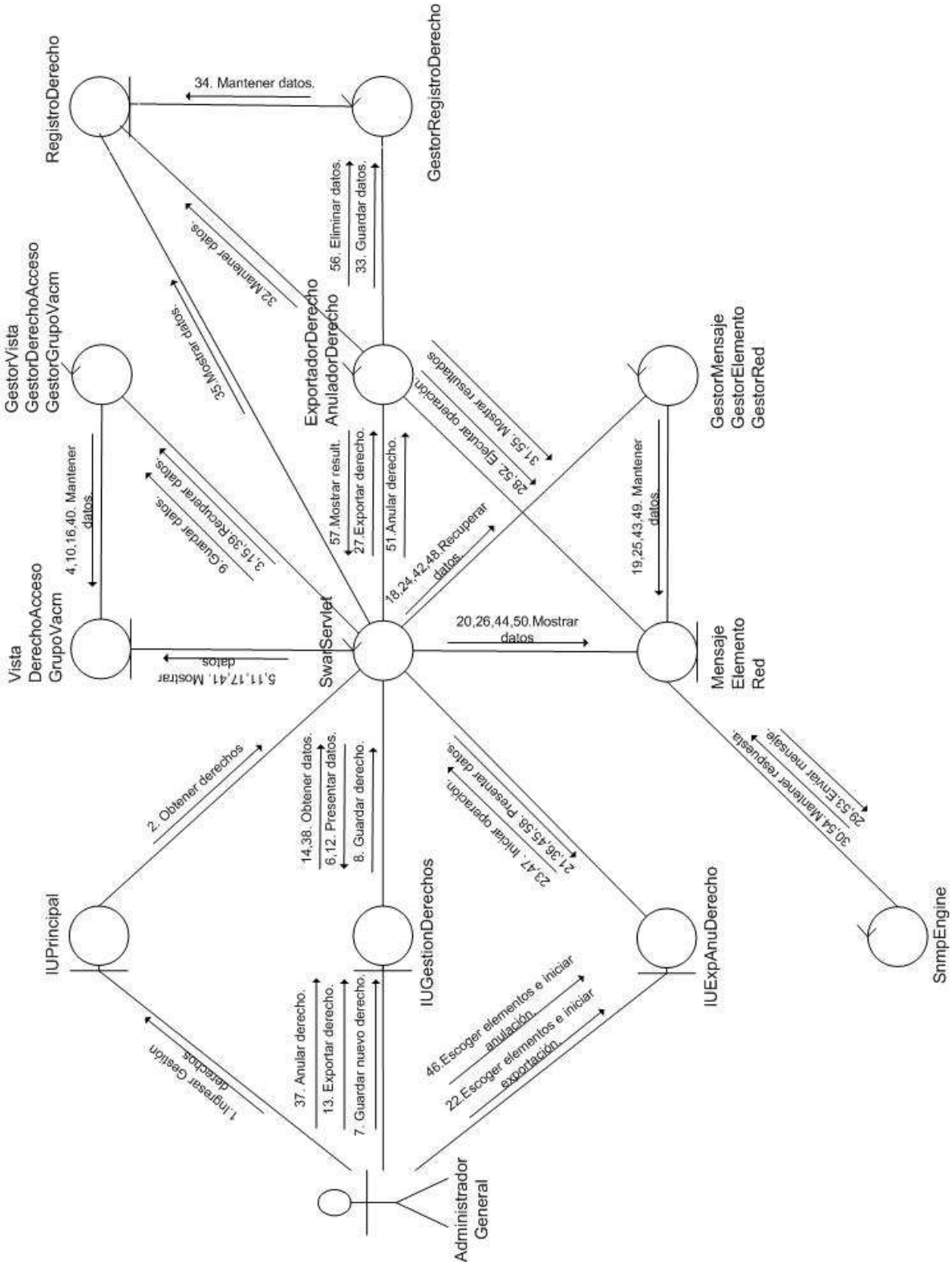


Figura 2. 13 Diagrama de interacción de clases de análisis de: Gestionar derechos de acceso.

Caso de uso: Cambiar passwords.
Clases de análisis: IUCambioPasswords, IUExpCambioPwds, GestorUsuario, Usuario, GestorRed, Red, GestorElemento, Elemento, GestorMensaje, Mensaje, GestorRegistroPasswords, RegistroPasswords, ExportadorPasswords, SnmpEngine y KeyChangeManager.
Secuencia: Administrador limitado/general solicita desde IUPrincipal mostrar la página IUCambioPasswords (1). SwarServlet pide a IUCambioPasswords mostrar el formulario (2-3). El actor ingresa los actuales y los nuevos passwords y luego solicita a la aplicación que realice el cambio (4). SwarServlet ordena a GestorUsuario a actualizar los passwords (5-7). Luego, el actor pide realizar la exportación del cambio de passwords (8). SwarServlet pide a GestorUsuario, GestorRed y GestorElemento recuperar la información necesaria para realizar la exportación (9-15). IUExpCambioPwds muestra la información recuperada (16). El actor selecciona los elementos y acepta iniciar la exportación (17). SwarServlet utiliza a GestorMensaje recuperar los parámetros de SNMPv3 (18-21). Luego, SwarServlet asigna a ExportadorPasswords la tarea en si (22). ExportadorPasswords utiliza a KeyChangeManager para realizar los cálculos requeridos para obtener los valores que se necesitan para establecer el cambio de passwords en la MIB de cada elemento seleccionado (23). ExportadorPasswords usa a Mensaje para ejecutar operaciones SNMP de establecimiento (24-27) y guarda temporalmente los resultados de todo el proceso en RegistroPasswords (28). Luego, ExportadorPasswords pide a GestorRegistroPasswords guardar los resultados definitivos (29-30). SwarServlet solicita a IUExpAnuPasswords presentar los resultados finales (31-32). Finalmente, el actor ordena a la aplicación activar los cambios en SnmpEngine (33). SwarServlet pide a GestorUsuario recuperar los datos necesarios para la activación (34-37). SwarServlet ordena a SnmpEngine a actualizar los passwords con la información obtenida (38).

Tabla 2. 39 Realización de Cambiar passwords.

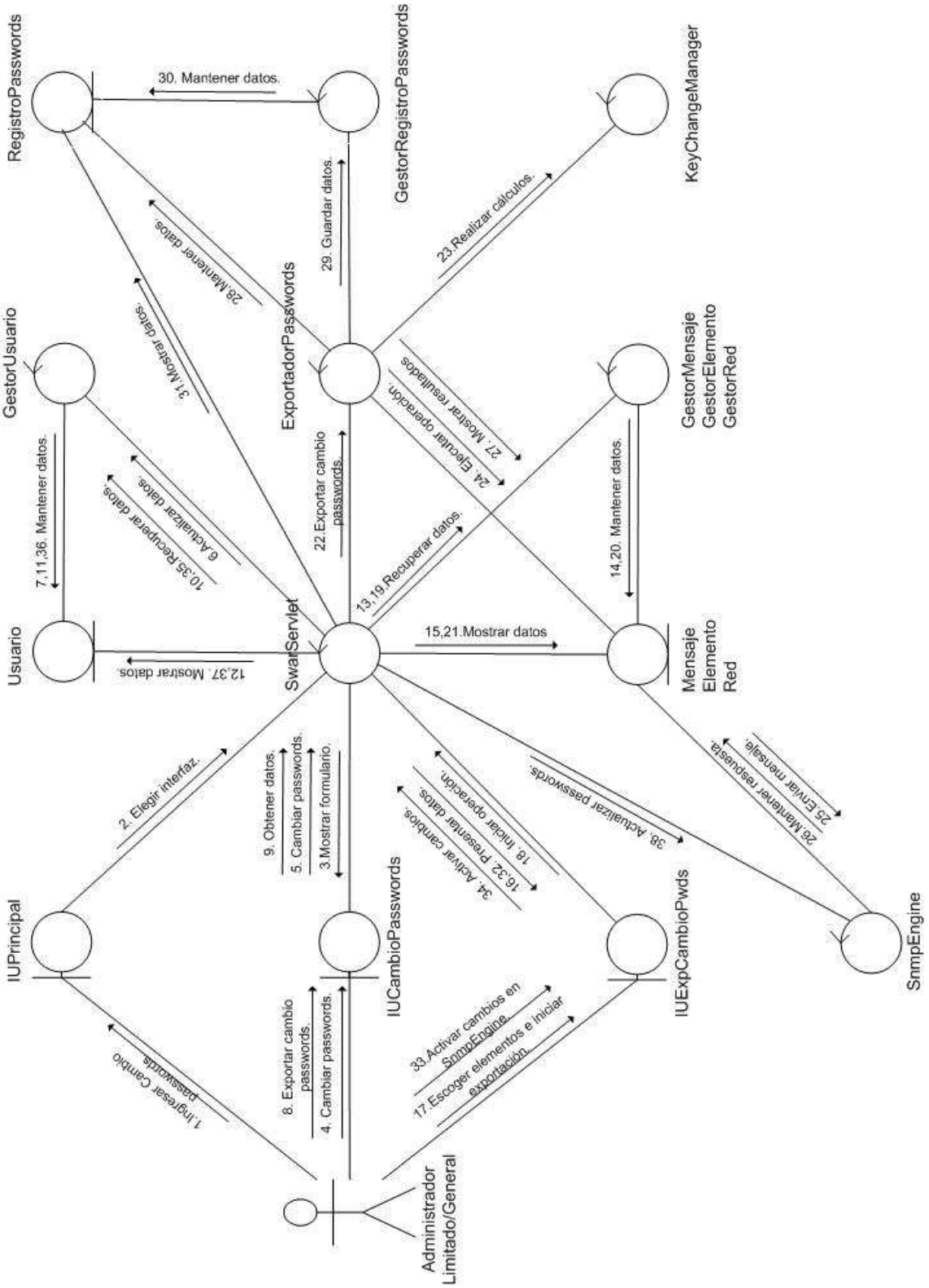


Figura 2. 14 Diagrama de interacción de clases de análisis de: Cambiar passwords.

2.1.3.2 Clases de análisis

Las clases de análisis son abstracciones de los elementos que intervienen en las ejecuciones de los casos de uso. Hay tres tipos de clases de análisis: clases de interfaz, clases de entidad y clases de control.

- **Clases de interfaz:** Estas clases modelan la interacción entre el actor y la aplicación. Generalmente una clase de interaz representa abstracciones de ventanas, formularios, paneles, entre otros, por donde el usuario solicita la ejecución de un requerimiento y recibe respuesta del sistema a través del mismo medio.

- **Clases de entidad.** Estas clases representan los datos almacenados y comportamiento de objetos que participan en los casos de uso. Sirven para representar la estructuración lógica de la información.

- **Clases de control**. Estas clases representan transferencia de información entre objetos (instancias de clases interfaz y entidad). Se encargan de ejecutar procesos relacionados con la lógica del negocio no presente en las clases de entidad. Además, pueden crear, modificar y eliminar datos.

A continuación se muestran las clases de análisis indentificadas a partir del modelo de casos de uso de la aplicación:

- **Clases de interfaz:** IUBienvenida, IUPrincipal, IUInicioSesion, IUConfigMensajes, IUConfigServidor, IUDescubHosts, IUDescubHostsResult, IUElementos, IUInfoElemento, IUGrupoMIB, IUNotificaciones, IUGestionMIBs, IUGestionUsuarios, IUExpAnuUsuario, IUGestionVistas, IUExpAnuVista, IUGestionDerechos, IUExpAnuDerecho, IUCambioPasswords y IUExpCambioPwds.

- **Clases de entidad:** Usuario, Mensaje, MIB, Elemento, Red, Servidor, NodoMIB, Notificacion, GrupoVacm, RegistroUsuario, Vista, RegistroVista, DerechoAcceso, RegistroDerecho y RegistroPasswords.

- **Clases de control:** SwarServlet, GestorMensaje, GestorMIB, GestorElemento, GestorRed, BuscadorElementos, PingUtil, SnmpEngine, ReceptorNotificiones, InfoHostTreeFactory, NavMIBTreeFactory, GestorNodoMIB, GestorNotificacion, GestorUsuario, GestorServidor, GestorGrupoVacm, GestorRegistroUsuario, ExportadorUsuario, AnuladorUsuario, GestorVista, GestorRegistroVista, ExportadorVista, AnuladorVista, GestorDerechoAcceso, GestorRegistroDerecho, ExportadorDerecho, AnuladorDerecho, GestorRegistroPasswords, ExportadorPasswords y KeyChangeManager.

En la sección A de ANEXOS se muestra la definición de cada una de las clases de análisis. Cada definición contiene responsabilidades y atributos pertenecientes a una clase en particular.

2.1.3.3 Paquetes de análisis

Los paquetes de análisis son formas compactas de organización del modelo de análisis. Una forma de identificar paquetes de análisis es en base a la asignación de casos de uso a un paquete en particular. Los criterios para asignar casos de uso a un paquete son:

- Agrupar casos de uso que dan soporte a un determinado proceso dentro del sistema.

- Agrupar casos de uso que dan soporte a un determinado actor.

- Agrupar casos de uso que estén relacionados con relaciones de generalización y de extensión.

A continuación, los paquetes de análisis identificados para la aplicación:

Paquetes de análisis	**Descripción**
Gestión de sesión.	Contiene el caso de uso relacionado al ingreso de un usuario a la aplicación: Iniciar Sesión.
Gestión de configuración.	Contiene el caso de uso referido a la configuración de parámetros de funcionamiento de la aplicación: Configurar parámetros.
Gestión de elementos.	Contiene los casos de uso relacionados al manejo de información de elementos de red: Descubrir elementos de red, Visualizar información de elemento de red y Ejecutar operación SNMPv3.
Gestión de MIBs.	Contiene el caso de uso relacionado al manejo de información de grupos MIB: Navegar en un grupo MIB.
Gestión de notificaciones.	Contiene el caso de uso relacionado al manejo de información de notificaciones recibidas: Visualizar notificaciones.
Gestión general.	Contiene los casos de uso relacionados al manejo de información crítica de la aplicación por parte del administrador general; éstos son: Gestión de MIBs, Gestión de usuarios, Gestión de vistas y Gestión de derechos de acceso.
Gestión de cambio de passwords.	Contiene el caso de uso relacionado al manejo de passwords de un usuario: Cambiar passwords.

Tabla 2. 40 Paquetes de análisis de la aplicación.

Consecuente a los paquetes de análisis es necesario identificar **los paquetes de servicios**. Un **servicio**[71] es un conjunto de acciones relacionadas funcionalmente que se utiliza en varios casos de uso. La razón de los paquetes de servicio es debido a que existen clases de análisis que participan en varios paquetes de análisis. Por tanto, un **paquete de servicio** es un agrupamiento de clases de análisis relacionadas funcionalmente que contribuyen a la consecución de un servicio.

Paquetes de servicio	**Clases**	**Descripción**
Host	- Servidor. - GestorServidor.	Brinda servicios relacionados al manejo de información del equipo que alberga la aplicación.
Elementos	- Elemento. - GestorElemento.	Brinda servicios relacionados al manejo de información de elementos de red.
Usuarios	- Usuario. - GestorUsuario.	Brinda servicios relacionados al manejo de información de usuarios.
MIBs	- MIB. - GestorMIB.	Brinda servicios relacionados al manejo de información de grupos MIB.
Nodos	- NodoMIB. - GestorNodoMIB.	Brinda servicios relacionados al manejo de información de objetos MIB y de creación de archivos MIB-XML.
Notificaciones	- Notificacion. - GestorNotificacion.	Brinda servicios relacionados al manejo de

[71] Este concepto de servicio pertenece al Modelo de Análisis.

	- ReceptorNotificaciones.	información de notificaciones y a la recepción de las mismas.
Redes	- Red. - GestorRed.	Brinda servicios relacionados al manejo de información de redes.
Mensajes	- Mensaje. - GestorMensaje.	Brinda servicios relacionados al manejo de información de parámetros de mensajes SNMPv3 y de la ejecución de comandos *Get, Set, Trap e Inform*.
Grupos	- GrupoVacm. - GestorGrupoVacm.	Brinda servicios relacionados al manejo de información de grupos Vacm.
Vistas	- Vista. - GestorVista.	Brinda servicios relacionados al manejo de información de vistas.
Derechos	- DerechoAcceso. - GestorDerechoAcceso.	Brinda servicios relacionados al manejo de información de derechos de acceso.
Registros de usuarios	- RegistroUsuario. - GestorRegistroUsuario.	Brinda servicios relacionados al manejo de información de registros de usuarios. Cada registro asocia a un usuario con un elemento de red.
Registros de vistas	- RegistroVista.	Brinda servicios

	- GestorRegistroVista.	relacionados al manejo de información de registros de vistas. Cada registro asocia a una vista con un elemento de red.
Registros de derechos de acceso	- RegistroDerecho. - GestorRegistroDerecho.	Brinda servicios relacionados al manejo de información de registros de derechos de acceso. Cada registro asocia a un derecho de acceso con un elemento de red.
Registros de cambio de passwords	- RegistroPasswords. - GestorRegistroPasswords.	Brinda servicios relacionados al manejo de información de registros de cambio de passwords. Cada registro asocia a un usuario que ha cambiado sus passwords con un elemento de red.
Servlet	- SwarServlet.	Brinda servicios de atención a todos los requerimientos web efectuados por el usuario.
SNMP	- SnmpEngine.	Brinda servicios de: envío, recepción, encriptación y desencriptación de mensajes SNMP.
KCM	- KeyChangeManager	Brinda servicios de cálculo de valores utilizados para

		cambiar passwords de autenticación y privacidad de un usuario.
Buscador hosts	- BuscadorElementos. - PingUtil.	Brinda servicio de identificación de elementos de red activos. Además verifica diponibilidad del servicio SNMPv3 en elementos de red.
ExpAnuDerecho	- AnuladorDerecho. - ExportadorDerecho.	Brinda servicio de exportación/ anulación de un derecho de acceso hacia/en elementos de red.
ExpAnuUsuario	- ExportadorUsuario. - AnuladorUsuario.	Brinda servicio de exportación/ anulación de una cuenta de usuario hacia/en elementos de red.
ExpAnuVista	- ExportadorVista. - AnuladorVista.	Brinda servicio de exportación/ anulación de una vista hacia/en elementos de red.
ExpPasswords	- ExportadorPasswords.	Brinda servicio de exportación de cambio de passwords de un usuario hacia elementos de red.
Tree	- InfoHostTreeFactory. - NavMIBTreeFactory.	Brinda servicios de generación de árboles de elementos de redes y de nodos MIB.

Tabla 2. 41 Paquetes de servicio de la aplicación.

Los paquetes de servicios pueden incluirse dentro de los paquetes de análisis. El criterio de inclusión es agregar aquellos paquetes de servicio que contienen las clases de análisis que permiten llevar a cabo los casos de uso que conforman un paquete de análisis. A continuación se muestra la estructura de los paquetes de análisis:

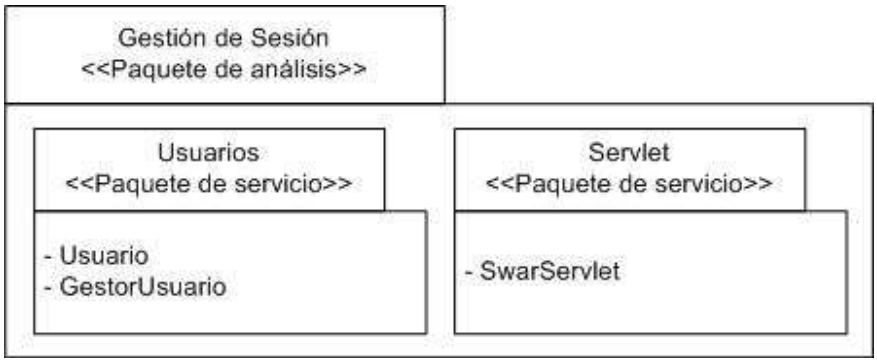


Figura 2. 15 Estrucutra de paquete de análisis: Gestón de Sesión.

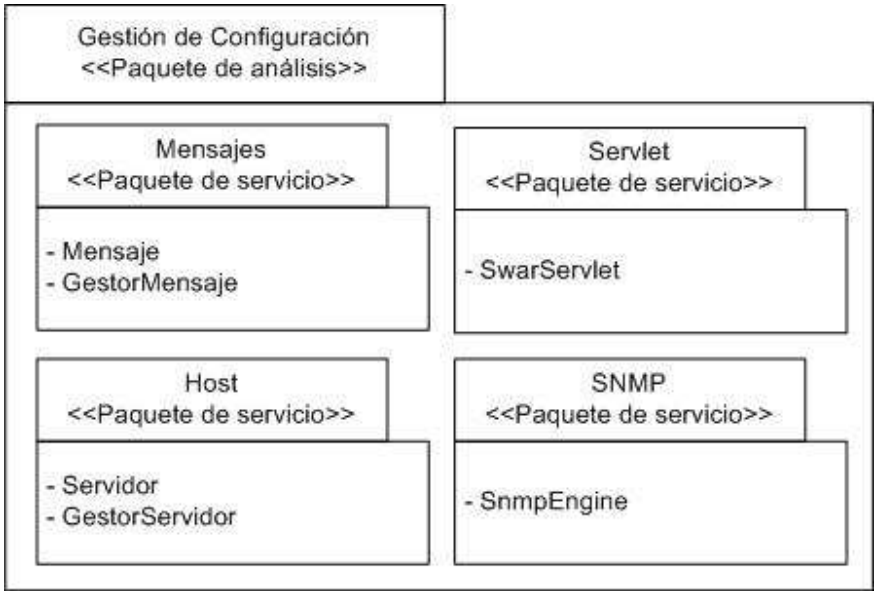


Figura 2. 16 Estructura de paquete de análisis: Gestón de Configuración.

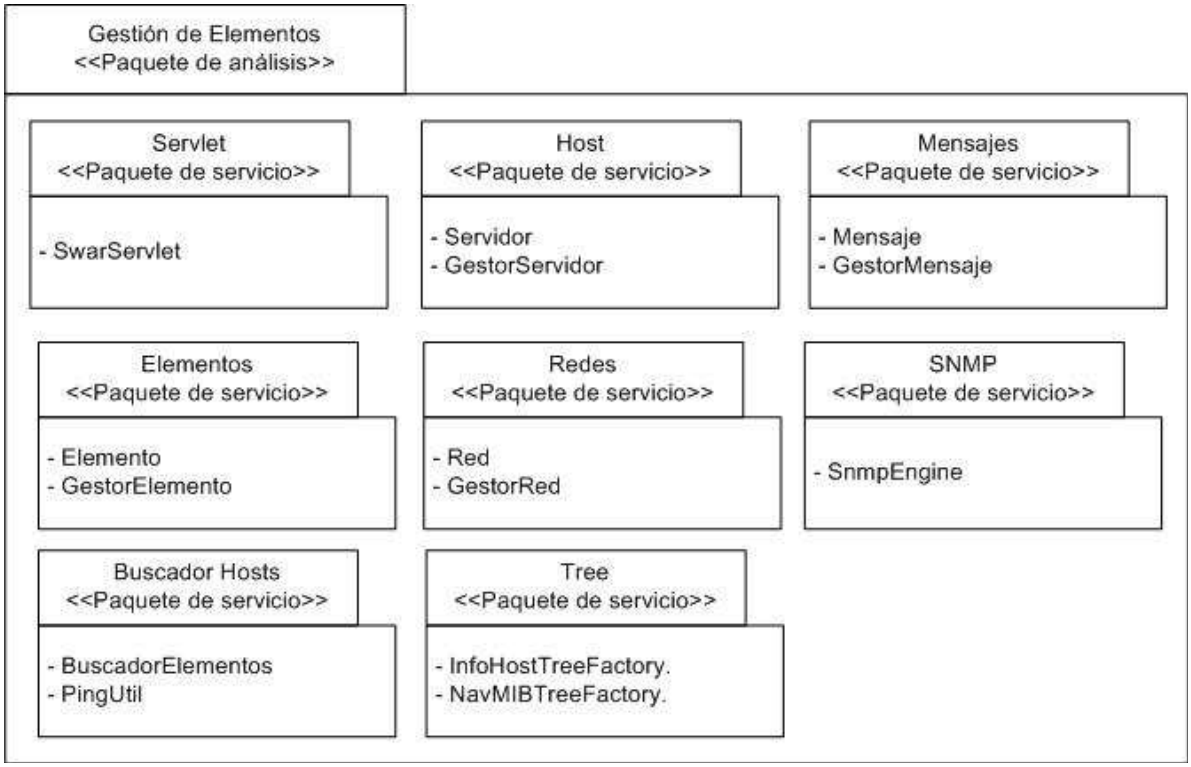


Figura 2. 17 Estructura de paquete de análisis: Gestión de Elementos.

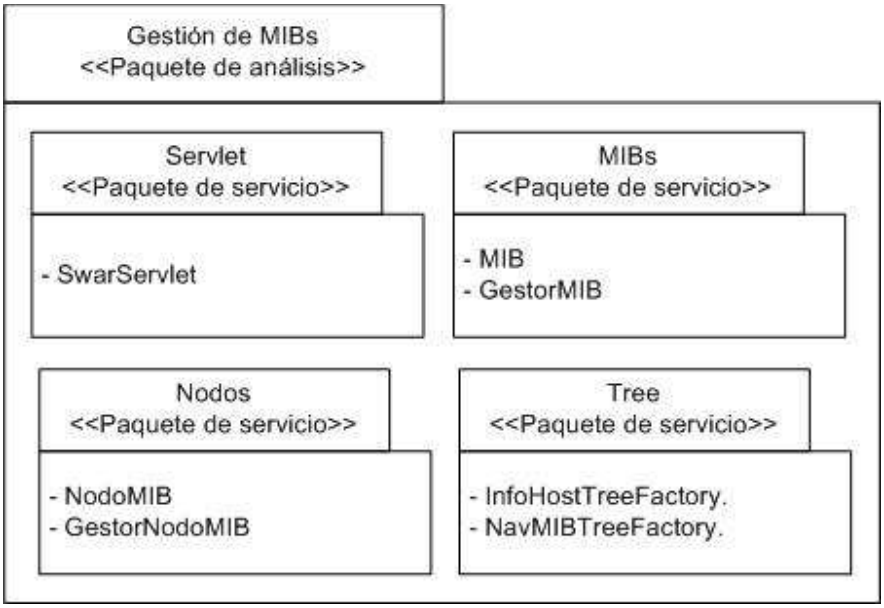


Figura 2. 18 Estructura de paquete de análisis: Gestión de MIBs.

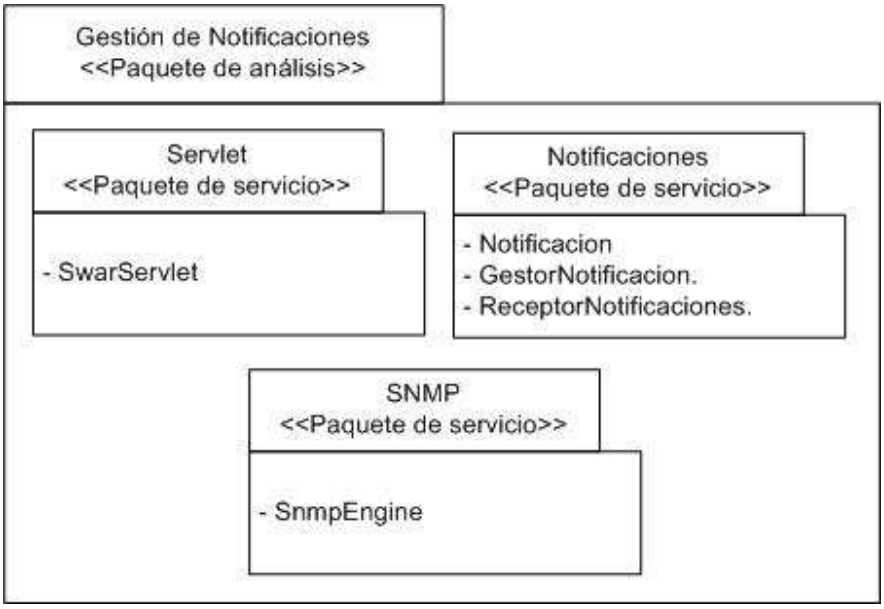


Figura 2. 19 Estructura de paquete de análisis: Gestión de Notificaciones.

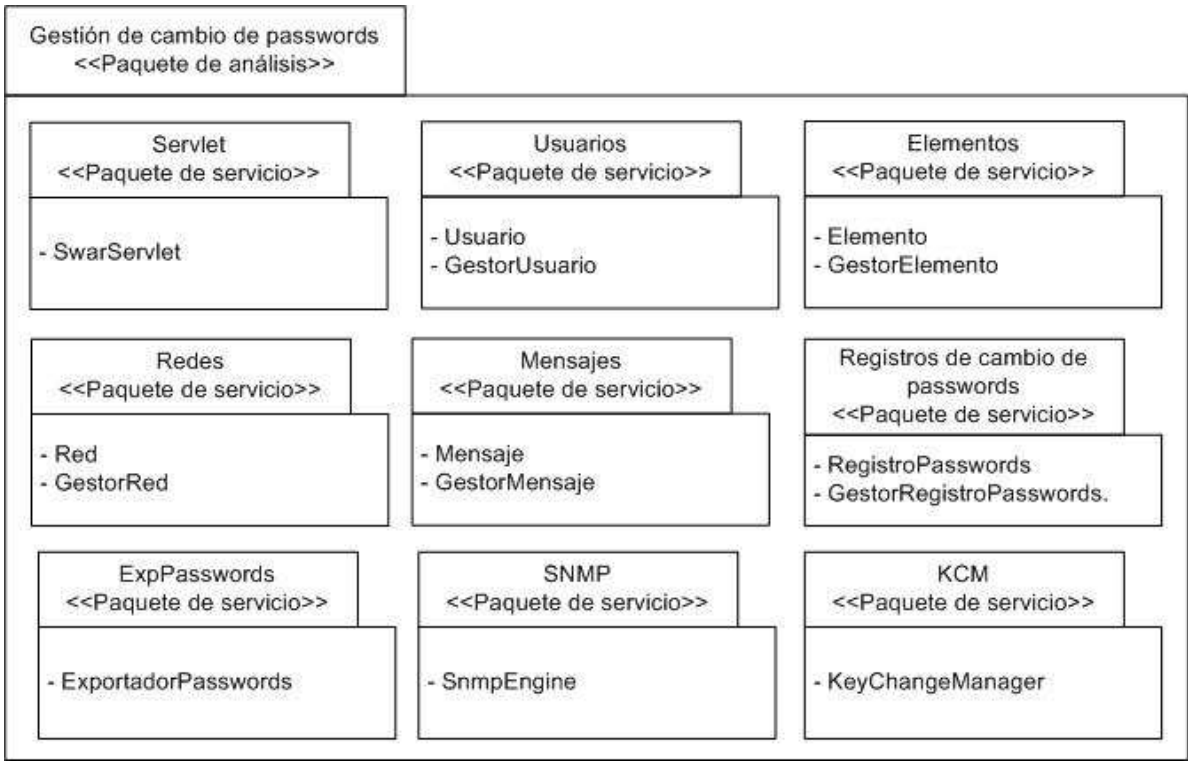


Figura 2. 20 Estructura de paquete de análisis: Gestión de cambio de passwords.

Figura 2. 21 Estructura de paquete de análisis: Gestión General.

Los paquetes de análisis identificados anteriormente también poseen interrelaciones de dependiencia. Esto se debe a que un paquete de servicio es usado en más de un paquete de análisis. A continuación se muestra las relaciones de dependencia entre paquetes de análisis:

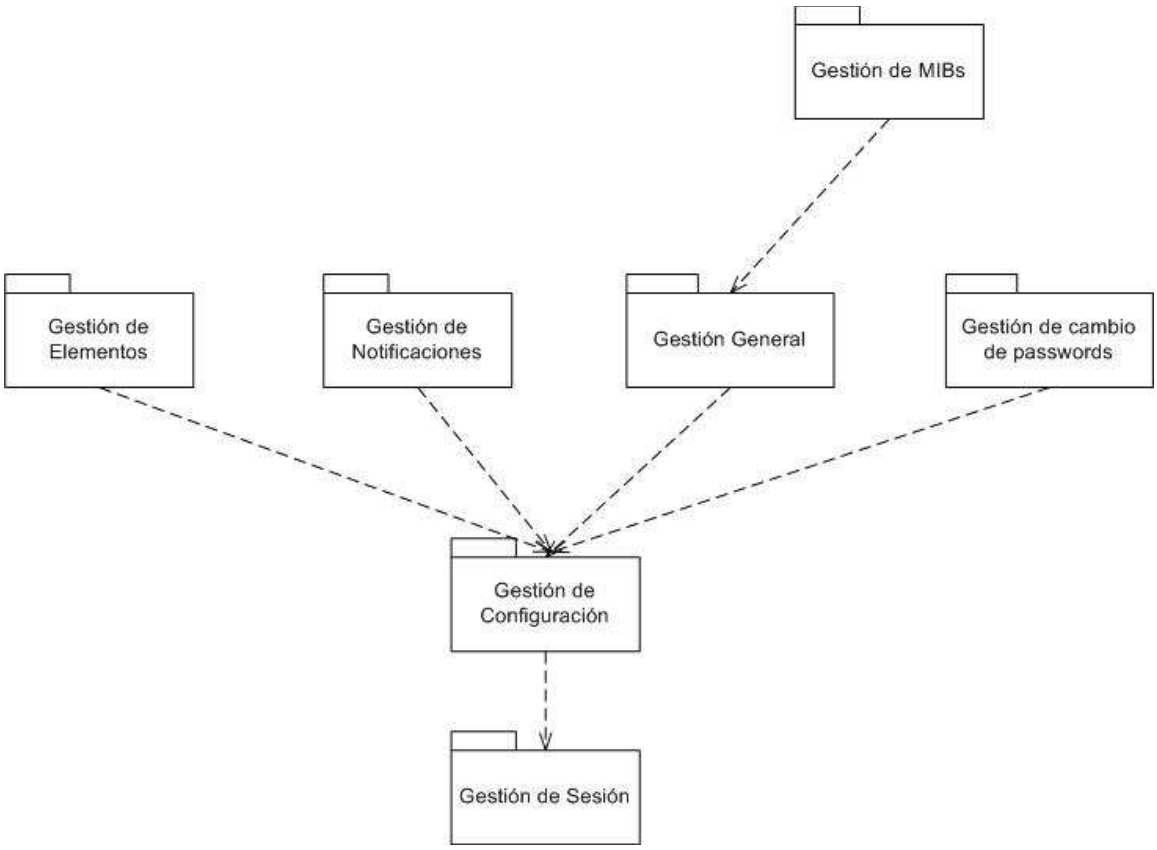


Figura 2. 22 Dependencias de paquetes de análsis.

En la figura anterior, el paquete Gestión de Configuración depende del paquete Gestión de Sesión porque este último contiene los paquetes de servicio básicos para ingresar a la aplicación y registrar el nombre de usuario con el cual se van a ejecutar las demás funciones de la misma. Luego, los paquetes de análisis: Gestión de Elementos, Gestión de Notificaciones, Gestión General y Gestión de cambio de passwords dependen del paquete Gestión de Configuración; la razón es Gestión de Configuración maneja los valores de parámetros de Servidor y Mensaje los cuales son utilizados para proporcionar servicios de dichos paquetes. Finalmente, el paquete Gestión de MIBs depende de Gestión General debido a que éste maneja, además de otras cosas, la disponiblidad de archivos MIB-XML.

2.2 DISEÑO DEL SISTEMA [35]

El Diseño es el procedimiento mediante el cual se da forma al sistema para que cumpla con todos los requisitos. El diseño incluye: modelo de diseño y modelo de despliegue.

2.2.1 MODELO DE DISEÑO

El **modelo de diseño** es la descripción de la estructuración física de la aplicación, y por lo tanto es indispensable para llevar a cabo la implementación de la misma. Los componentes del modelo de diseño son: realización de casos de uso-diseño, clases de diseño, diseño de la interfaz gráfica y subsistemas de diseño.

2.2.1.1 Realización de casos de uso-diseño

La realización de casos de uso-diseño es una descripción de la realización de un caso de uso determinado en términos de clases de diseño y sus objetos. Para ello se toma como entrada la realización de casos de uso-análisis. La realización de casos de uso-diseño tiene los siguientes componentes: diagrama de clases, diagrama de colaboración, diagrama secuencia y descripción. La siguiente tabla muestra el formato para la realización de casos de uso-diseño:

Caso de uso: Nombre del caso de uso.
Diagrama de clases[72]: Gráfico de clases para el caso de uso.
Diagrama de colaboración: Gráfico de colaboración de clases del caso de uso.
Diagrama de secuencia: Gráfico de secuencia de las clases del caso de uso.
Descripción: Descripción textual de la realización del caso de uso-diseño.

Tabla 2. 42 Modelo de realización de caso de uso-diseño.

[72] Estos diagramas de clases únicamente representarán las relaciones entre clases, no atributos y métodos. La razón de esto es la insuficiencia de espacio. Más adelante se mostrán gráficos con toda la información incluída. Además, el diagrama será excluído de la tabla del modelo con el fin de mejorar la presentación de la misma.

A continuación se muestran la realización de caso de uso - diseño para cada caso de uso de la aplicación:

Caso de uso: Iniciar sesión.
Descripción: Administrador limitado/general desea ingresar a la aplicación, para ello en IUBienvenida hace clic en el botón Ingresar y se muestra IUInicioSesion. Adminstrador limitado/general ingresa los datos en IUInicioSesion y hace clic en el botón Iniciar Sesión. IUInicioSesion utiliza a SwarServlet para validar la información que se ingresó. SwarServlet pide a GestorUsuario obtener información de dicho usuario. Usuario mantiene la información obtenida. SwarServlet realiza las comprobaciones pertinentes, y si el resultado es correcto, pide la presentación de IUPrincipal mostrando el nombre del usuario.

Tabla 2. 43 Realización de Iniciar Sesión.

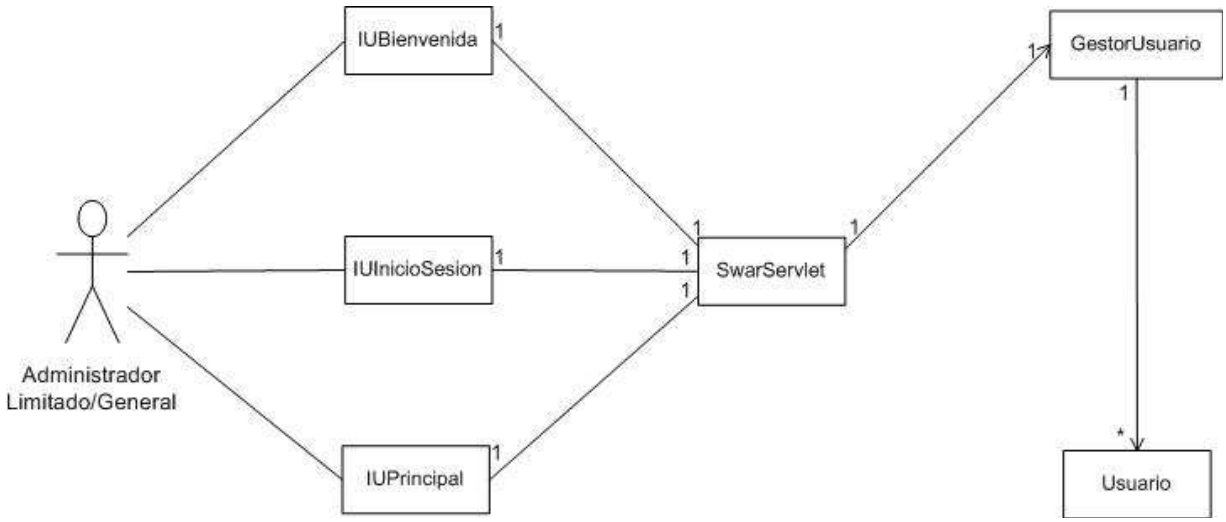


Figura 2. 23 Diagrama de clases del caso de uso: Iniciar Sesión.

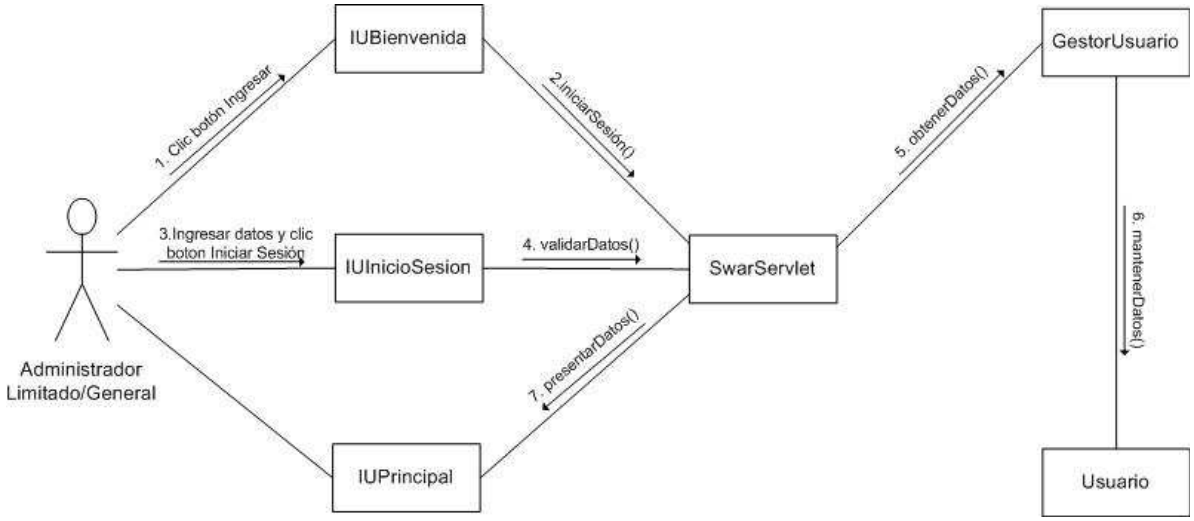


Figura 2. 24 Diagrama de colaboración del caso de uso: Iniciar Sesión.

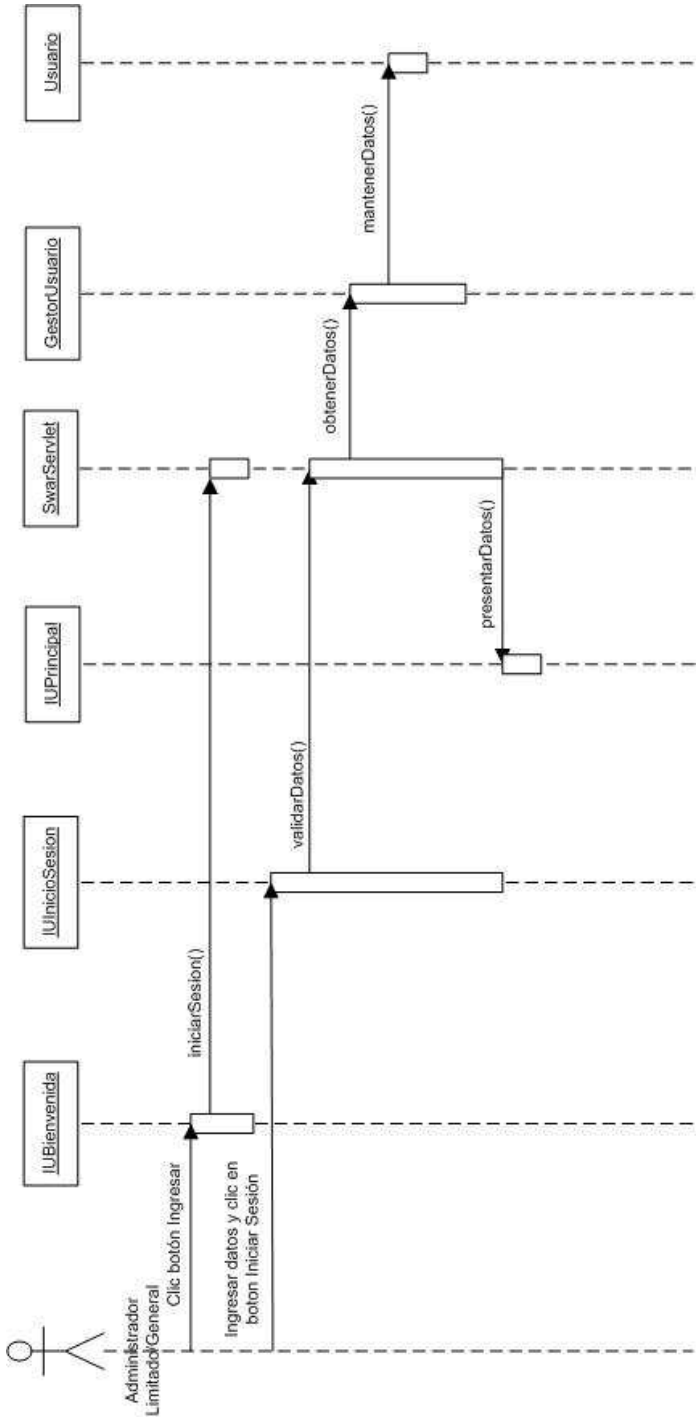


Figura 2. 25 Diagrama de secuencia del caso de uso: Iniciar Sesión.

Caso de uso: Configurar parámetros.
Descripción: Administrador limitado/general a través de IUPrincipal solicita ingresar a configurar mensajes. SwarServlet solicita a GestorMensaje recuperar los parámetros del mensaje SNMPv3 y los resultados obtenidos son presentados en IUConfigMensajes. Luego, el actor pide editar y guardar los parámetros de mensajes desde IUConfigMensajes. SwarServlet solicita a GestorMensaje actualizar los parámetros del mensaje SNMPv3 con los nuevos valores y éstos son presentados en IUConfigMensajes. Después, administrador limitado/general a través de IUPrincipal solicita ingresar a configurar servidor. SwarServlet solicita a GestorServidor recuperar los parámetros de servidor y los resultados obtenidos son presentados en IUConfigServidor. Luego, el actor pide editar y guardar los parámetros de servidor desde IUConfigServidor. SwarServlet solicita a GestorServidor actualizar los parámetros de servidor con los nuevos valores. Además, SwarServlet solicita a SnmpEngine a actualizar sus propios parámetros. Los nuevos valores son presentados en IUConfigServidor.

Tabla 2. 44 Realización de Configurar parámetros.

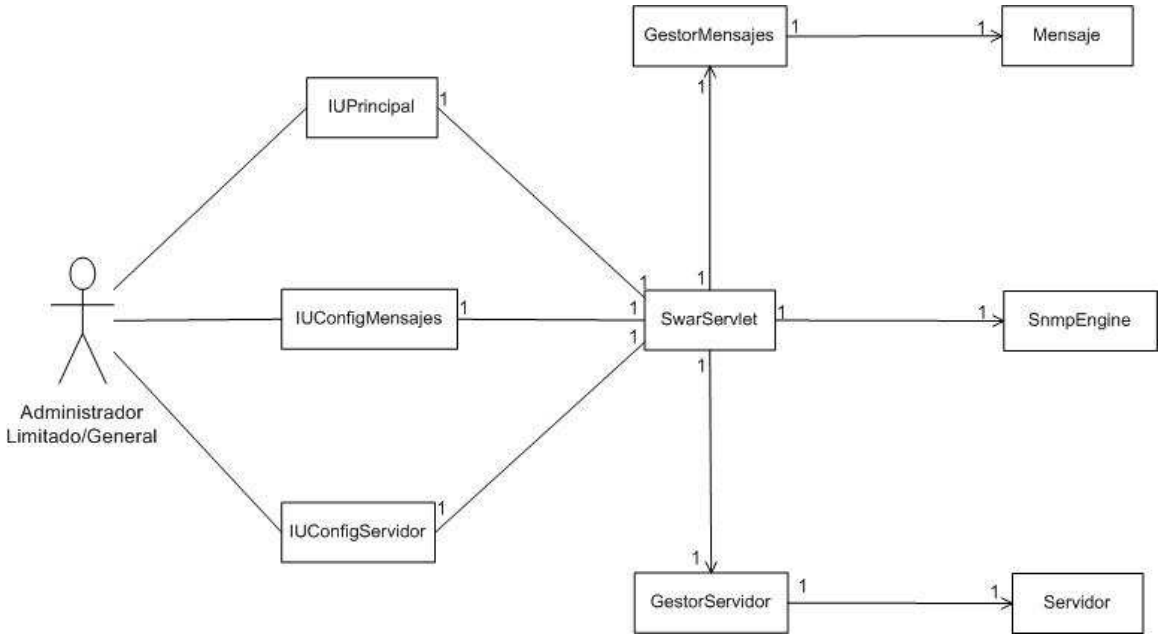


Figura 2. 26 Diagrama de clases del caso de uso: Configurar Parámetros.

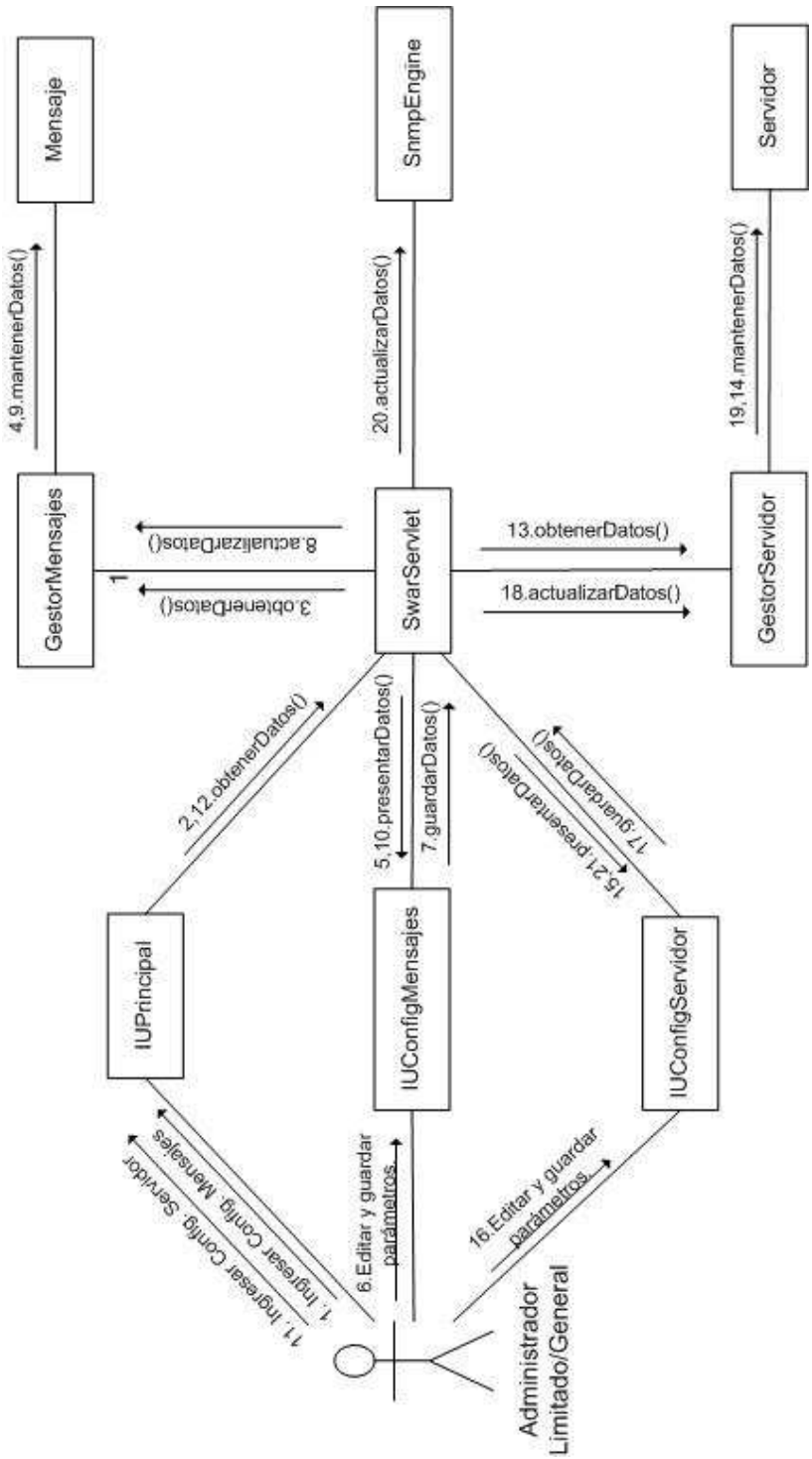


Tabla 2. 45 Diagrama de colaboración del caso de uso: Configurar Parámetros.

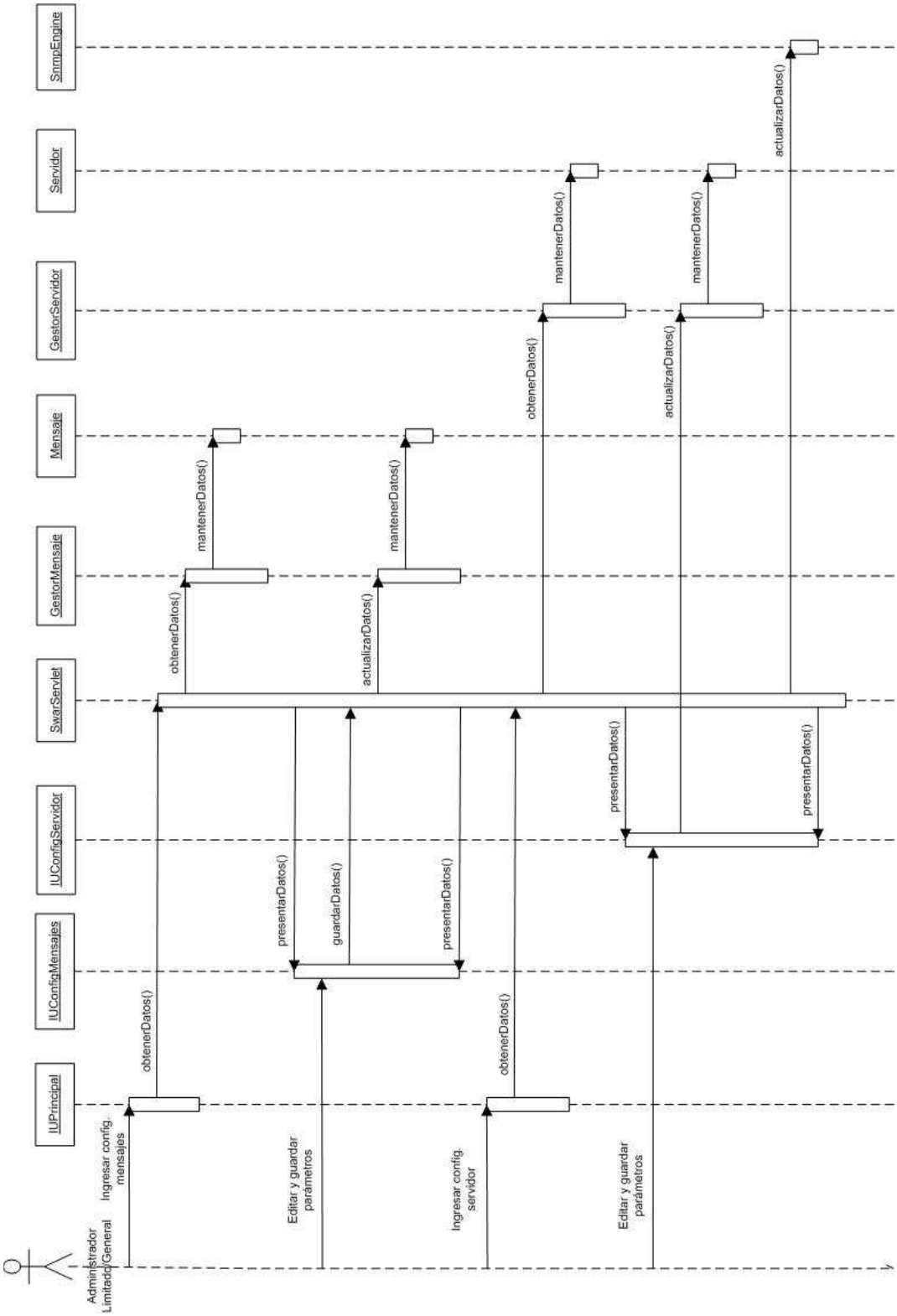


Figura 2. 27 Diagrama de secuencia del caso de uso: Configurar Parámetros.

Caso de uso: Descubrir elementos de red.
Descripción: Administrador Limitado/General solicita desde IUPrincipal ir a descubrimiento de elementos. SwarServlet utiliza a GestorServidor para recuperar información actual del servidor y se la presenta en IUDescubHosts. Luego, el actor ingresa el rango límite de direcciones IP y acepta inciar la operación desde IUDescubHosts. SwarServlet utiliza a GestorServidor, GestorMensaje y a GestorRed para recuperar información requerida por BuscadorElementos para que pueda realizar la tarea de descubrir hosts en la red. BuscadorElementos utiliza a PingUtil para identificar elementos activos y pide a Elementos mantener los resultados. Luego, BuscadorElementos utiliza a Mensaje para efectuar operaciones SNMP. Mensaje utiliza a SnmpEngine para enviar el mensaje SNMPv3 hacia los elementos activos y las respuestas se mantienen en VinculoVariable. Luego, IUDescubHostsResult presenta los resultados obtenidos. Finalmente, el actor solicita guardar los resultados. SwarServlet utiliza a GestorElemento y a GestorRed para efectuar esta actividad y nuevamente se presentan los resultados en IUDescubHostsResult.

Tabla 2. 46 Realización de Descubrir elementos de red.

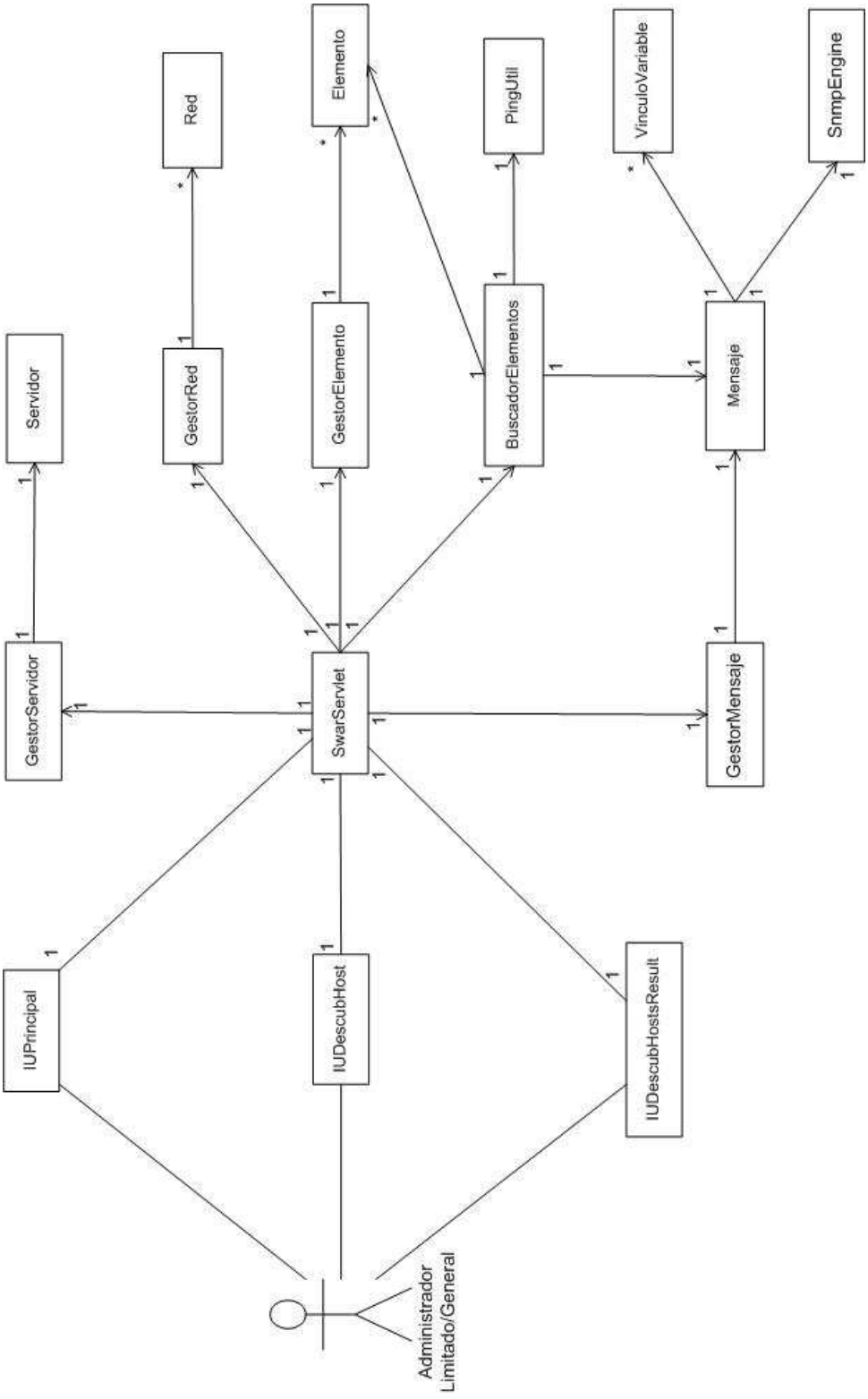


Figura 2. 28 Diagrama de clases del caso de uso: Descubrir elementos de red.

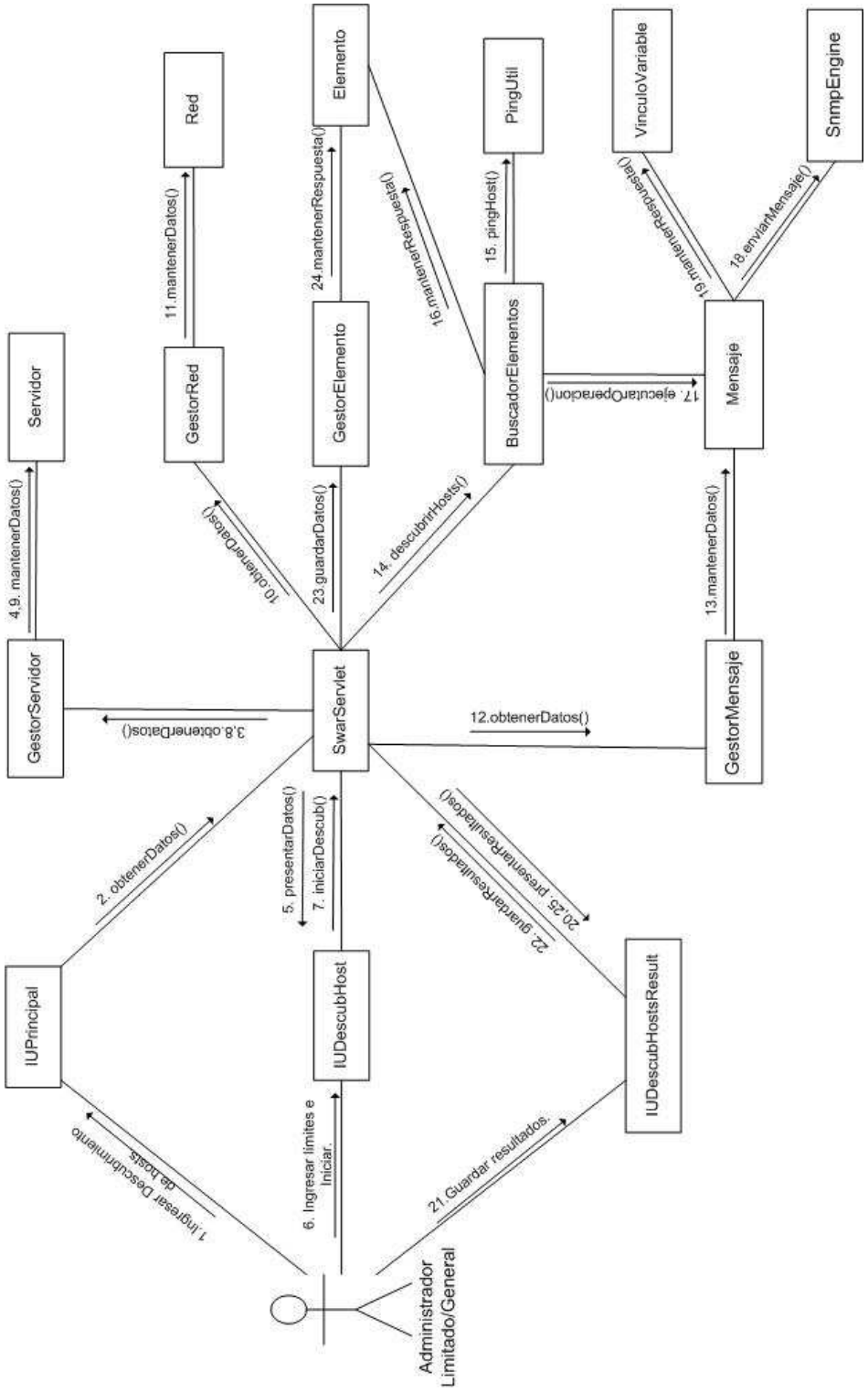


Figura 2. 29 Diagrama de colaboración del caso de uso: Descubrir elementos de red.

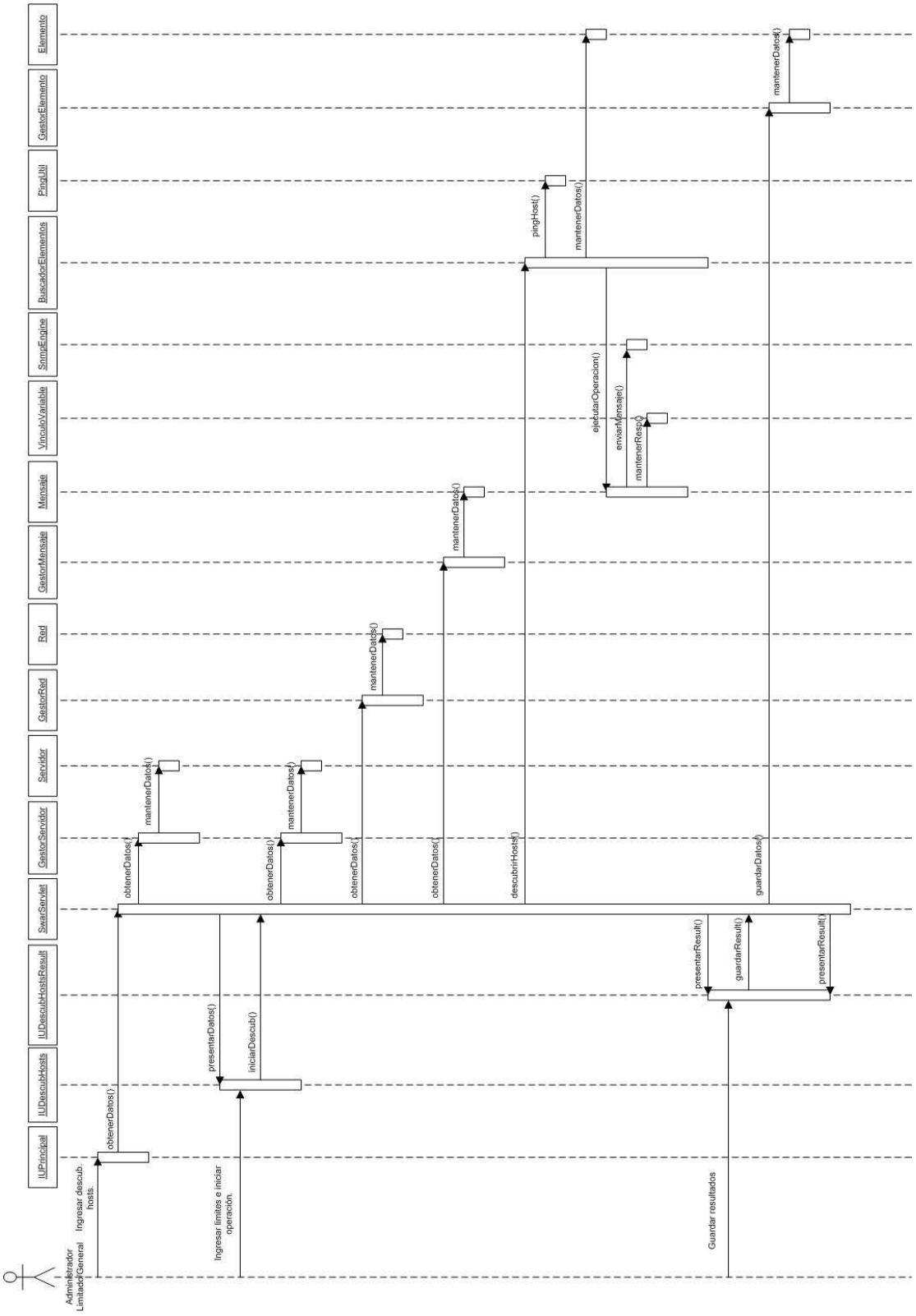


Figura 2. 30 Diagrama de secuencia del caso de uso: Descubrir elementos de red.

Caso de uso: Visualizar información de elemento de red.
Descripción: Administrador Limitado/General solicita desde IUPrincipal mostrar la página de elementos. SwarServlet utiliza a GestorElemento y a GestorRed para obtener elementos y redes registrados que InfoHostTreeFactory requiere para formar el árbol de elementos. El árbol formado se muestra en IUElementos. Luego, el actor solicita mostrar información básica de un elemento de red seleccionado desde IUElementos. SwarServlet utiliza a GestorElemento para recuperar dicha información. Los resultados se presentan en IUInfoElemento. Finalmente, el actor solicita mostrar información adicional de un elemento de red desde IUInfoElemento. SwarServlet utiliza a GestorMensaje para obtener Mensaje y SwarServlet lo utiliza para ejecutar operaciones SNMP necesarias para conseguir la información solicitada. Los resultados se presentan en IUInfoElemento.

Tabla 2. 47 Realización de Visualizar información de elemento de red.

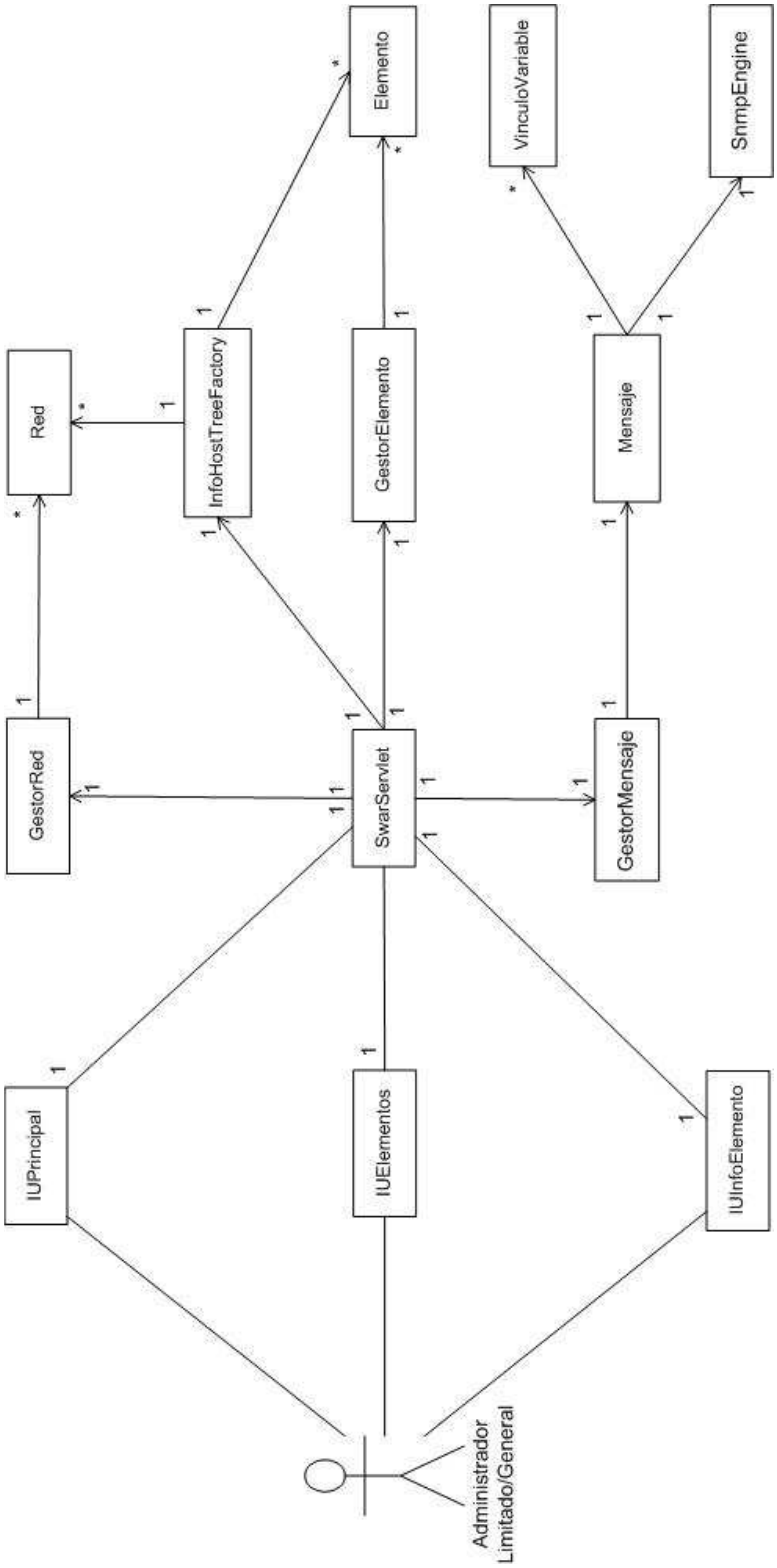


Figura 2. 31 Diagrama de clases del caso de uso: Visualizar información de elemento de red.

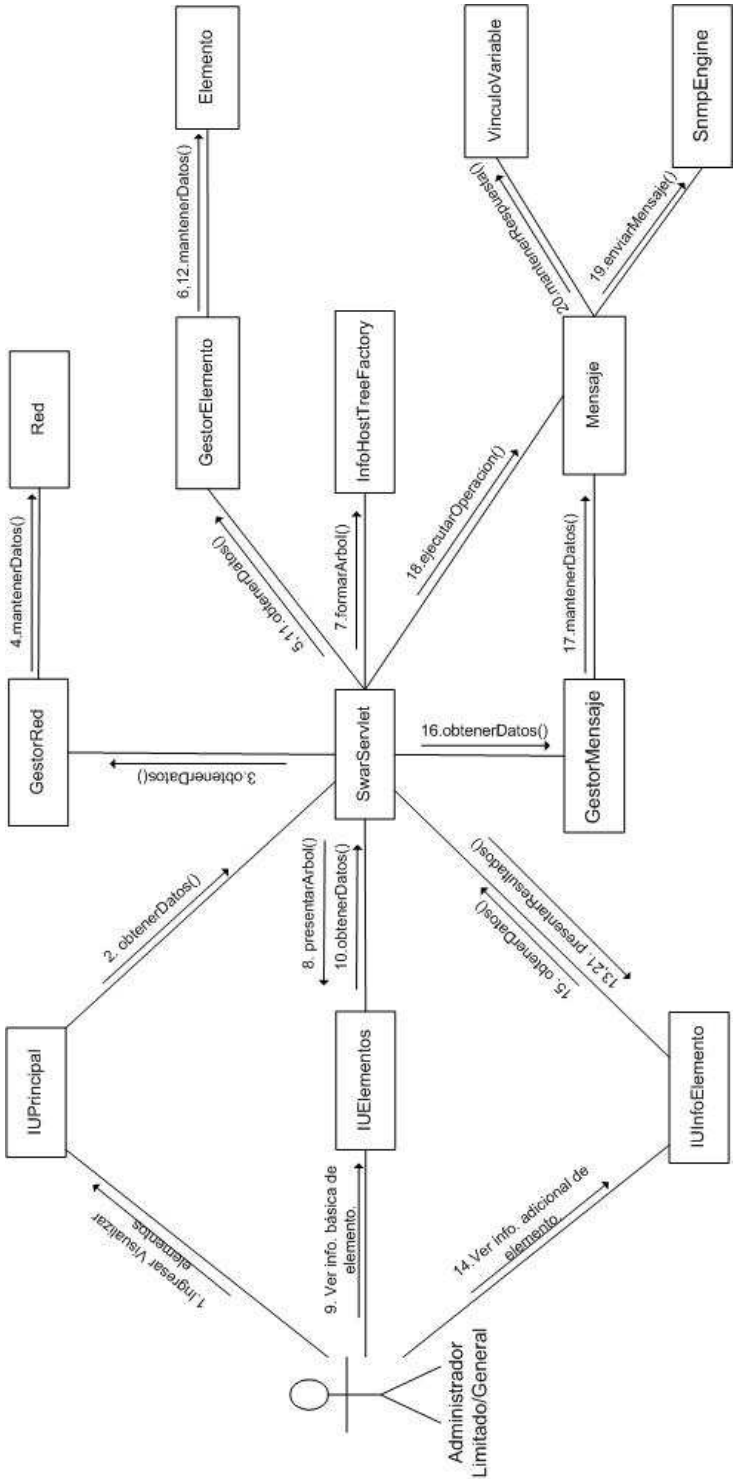


Figura 2. 32 Diagrama de colaboración del caso de uso: Visualizar información de elemento de red.

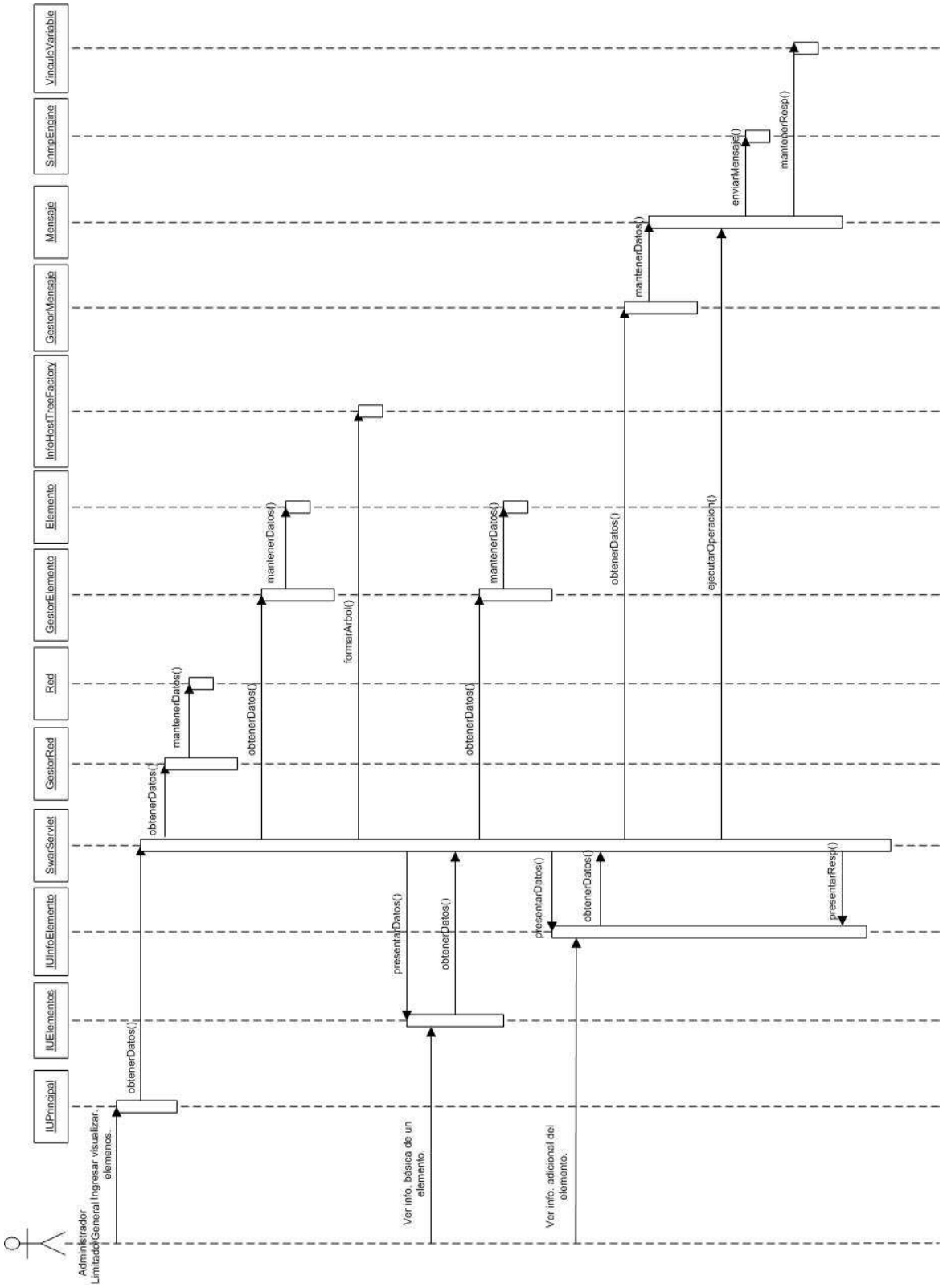


Figura 2. 33 Diagrama de secuencia del caso de uso: Visualizar información de elemento de red.

Caso de uso: Navegar en un grupo MIB.
Descripción: Administrador Limitado/General solicita desde IUPrincipal mostrar la página IUGrupoMIB. SwarServlet utiliza a GestorMIB para recuperar la lista de grupos MIBs y la presenta en IUGrupoMIB. El actor selecciona el nombre de un grupo y solicita cargar el árbol de dicho grupo. SwarServlet utiliza a GestorMIB y GestorNodoMIB para recuperar la información requerida por NavMIBTreeFactory para formar el árbol MIB. El árbol formado es presentado en IUGrupoMIB. Luego, administrador limitado/general solicita presentar las propiedades del nodo MIB seleccionado. SwarServlet utiliza a GestorNodoMIB para obtener dicha información y la muestra en IUGrupoMIB.

Tabla 2. 48 Realización de Navegar en un grupo MIB.

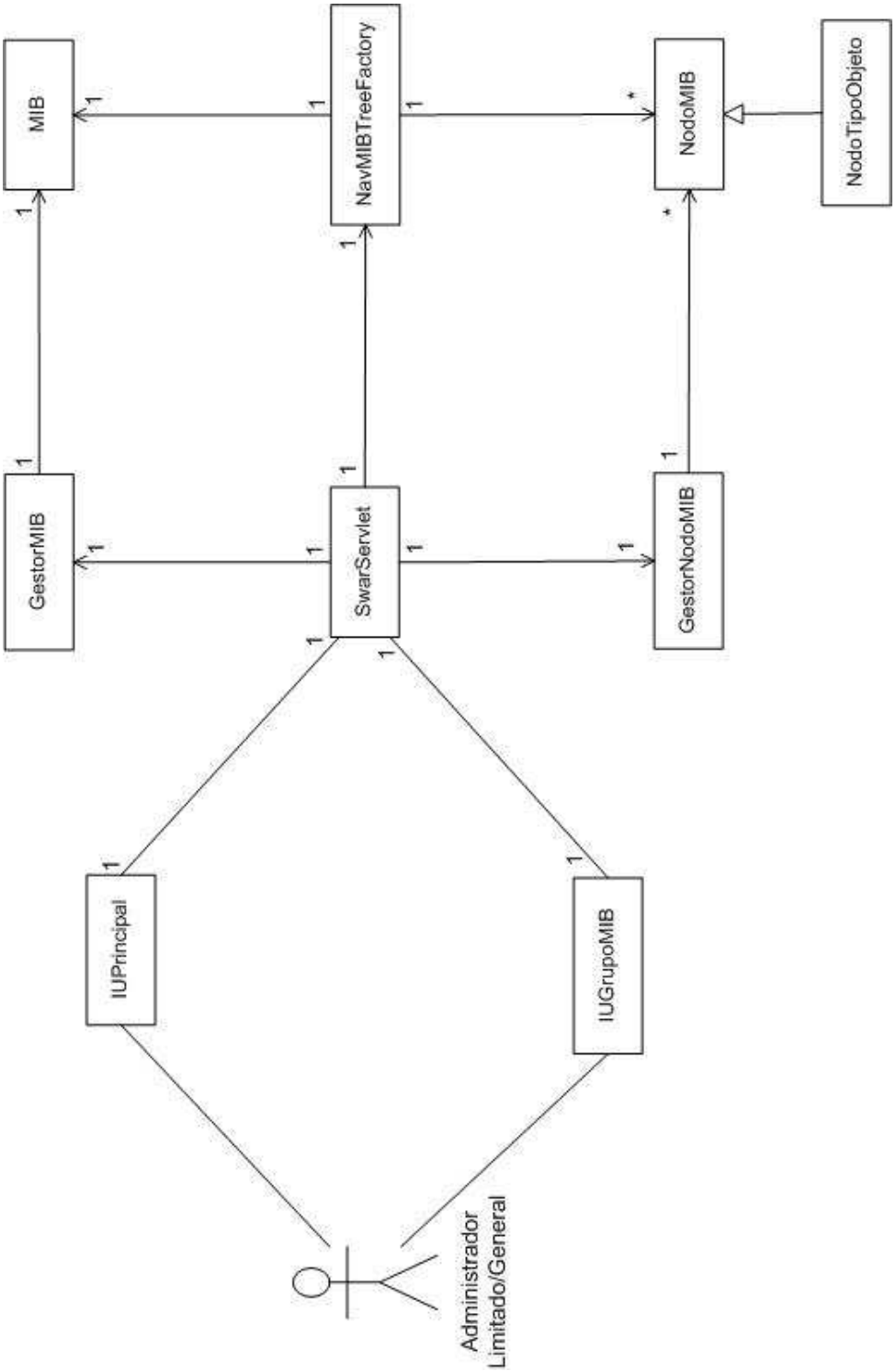


Figura 2. 34 Diagrama de clases del caso de uso: Navegar en un grupo MIB.

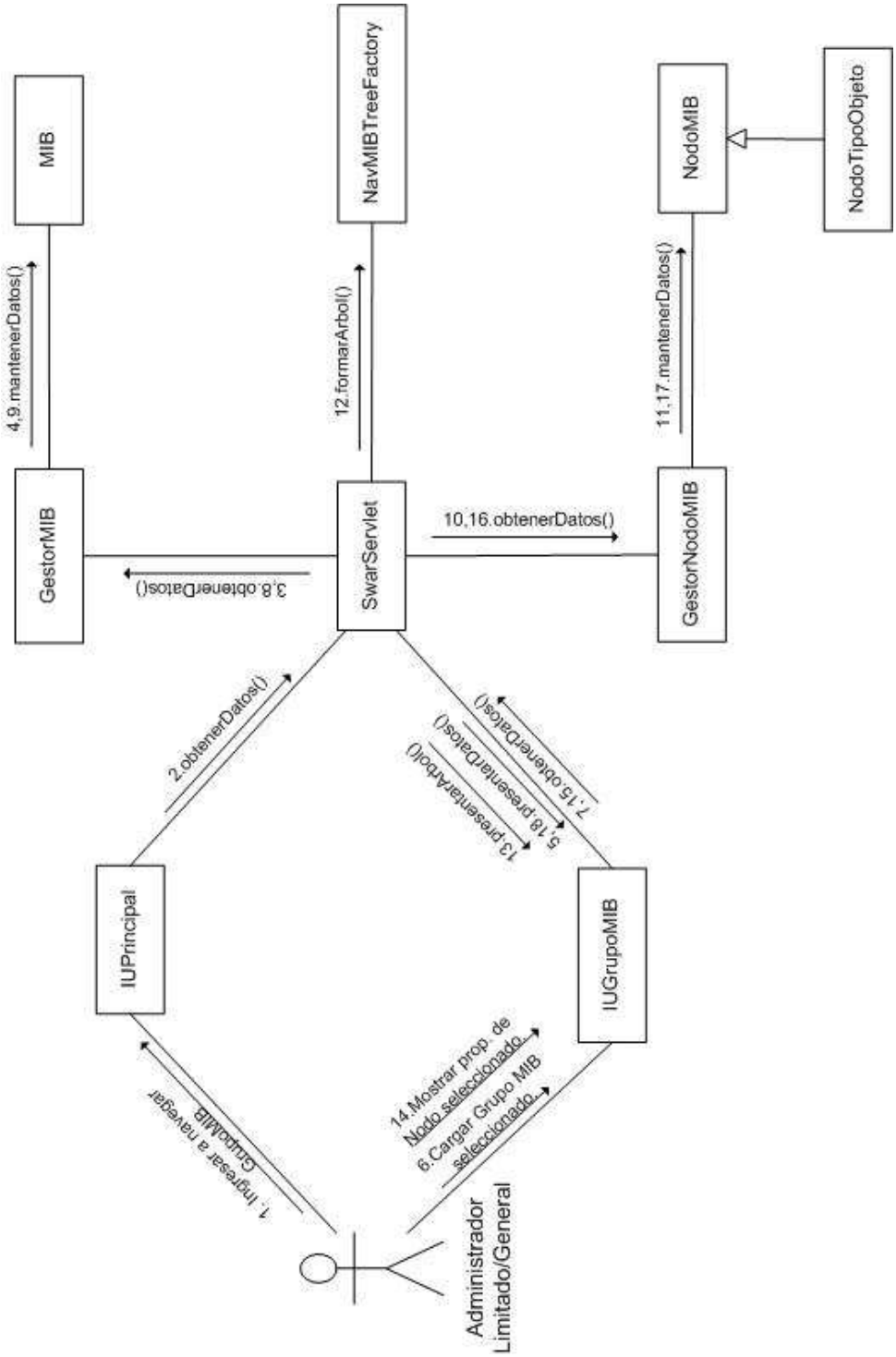


Figura 2. 35 Diagrama de colaboración del caso de uso: Navegar en un grupo MIB.

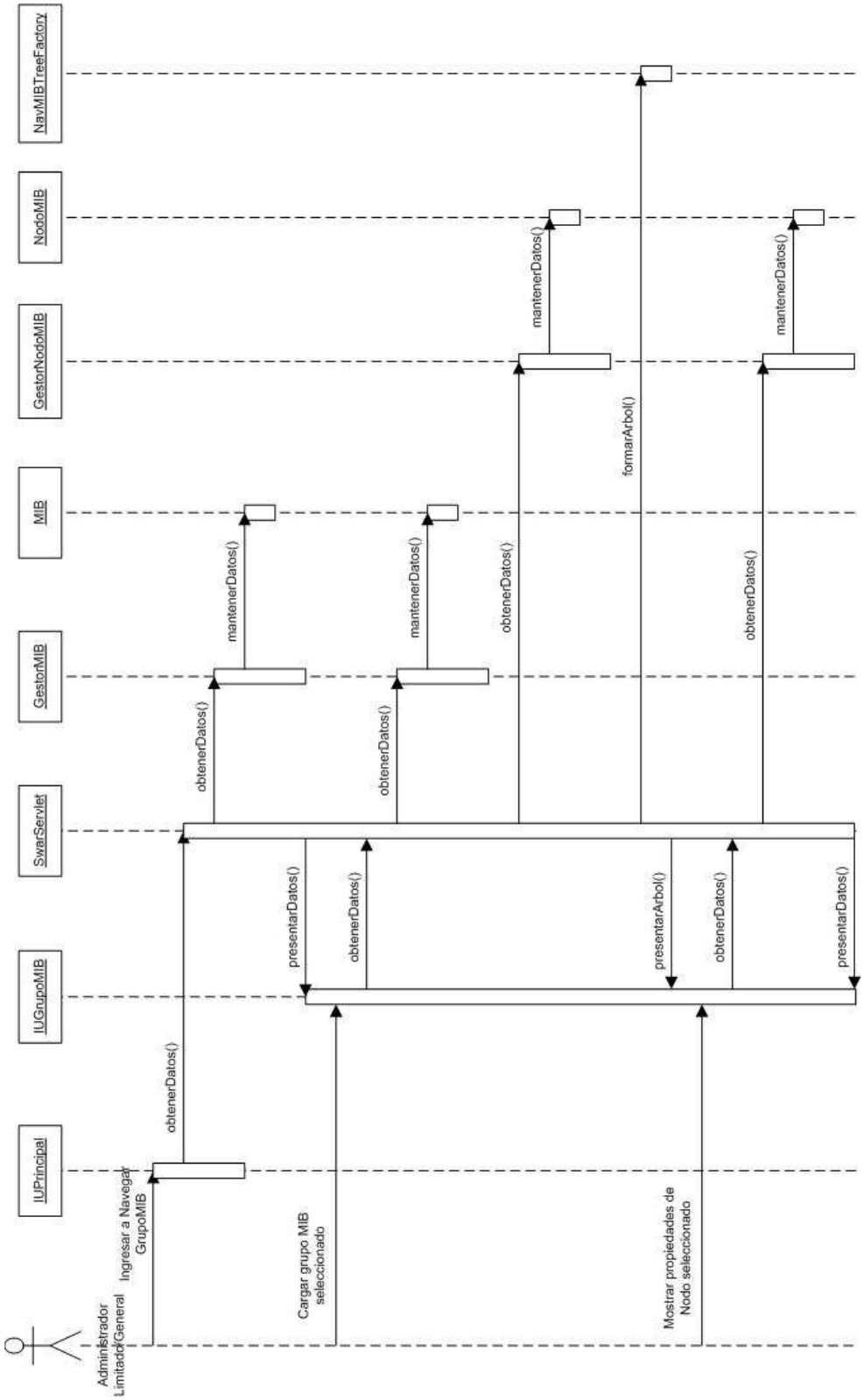


Figura 2. 36 Diagrama de secuencia del caso de uso: Navegar en un grupo MIB.

Caso de uso: Ejecutar operación SNMPv3.
Descripción: SwarServlet pide a GestorElemento obtener la lista de elementos registrados y ésta es presentada en IUGrupoMIB. Luego, administrador limitado/general solicita a través de IUGrupoMIB ejecutar una operación SNMP hacia el elemento seleccionado de la lista. SwarServlet pide a GestorMensaje obtener los parámetros de Mensaje. SwarServlet pide a Mensaje ejecutar el comando. Mensaje pide a su vez a SnmpEngine enviar el mensaje hacia el elemento destino. Las respuestas son almacenadas en VinculoVariable. Los resultados obtenidos son mostrados en IUGrupoMIB.

Tabla 2. 49 Realización de Ejecutar operación SNMPv3.

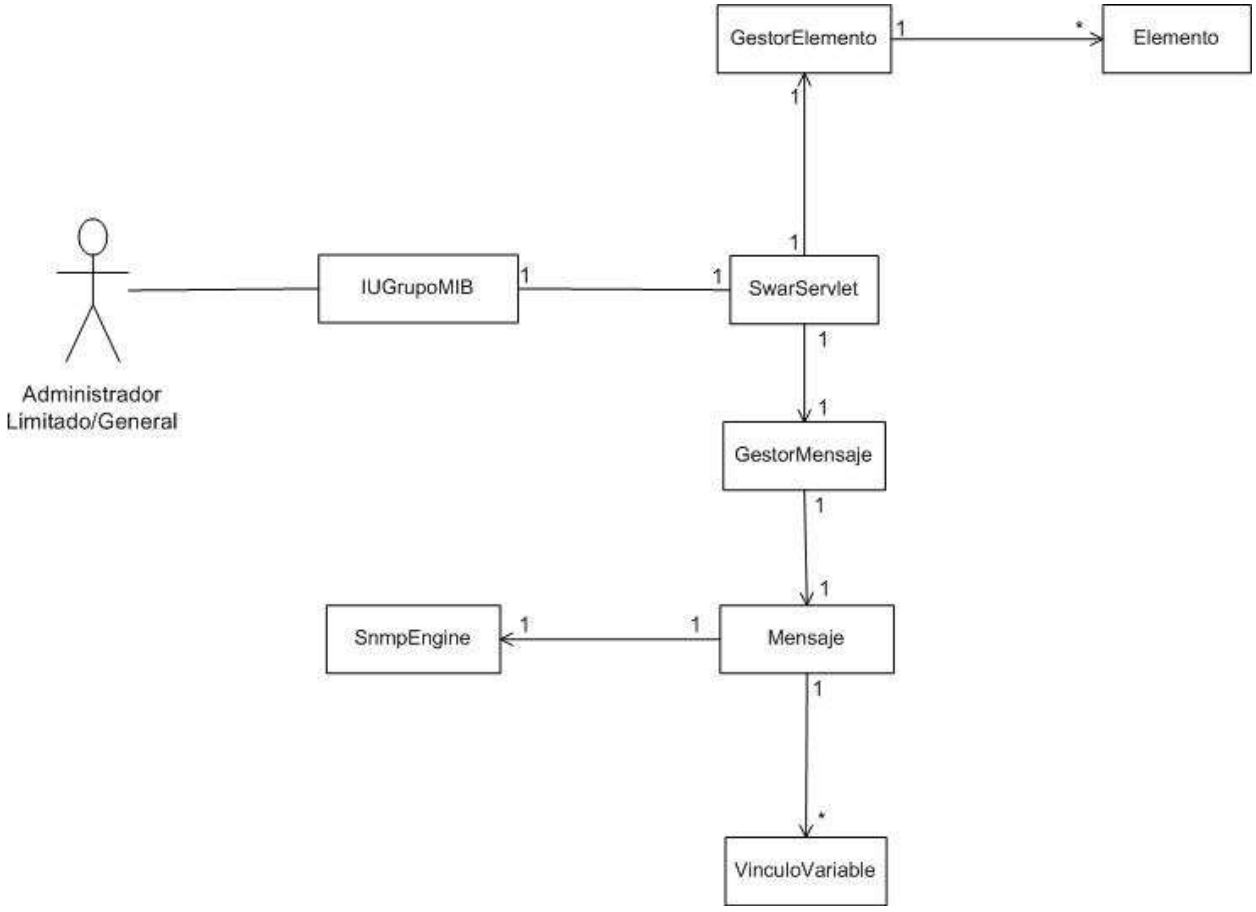


Figura 2. 37 Diagrama de clases del caso de uso: Ejecutar operación SNMPv3.

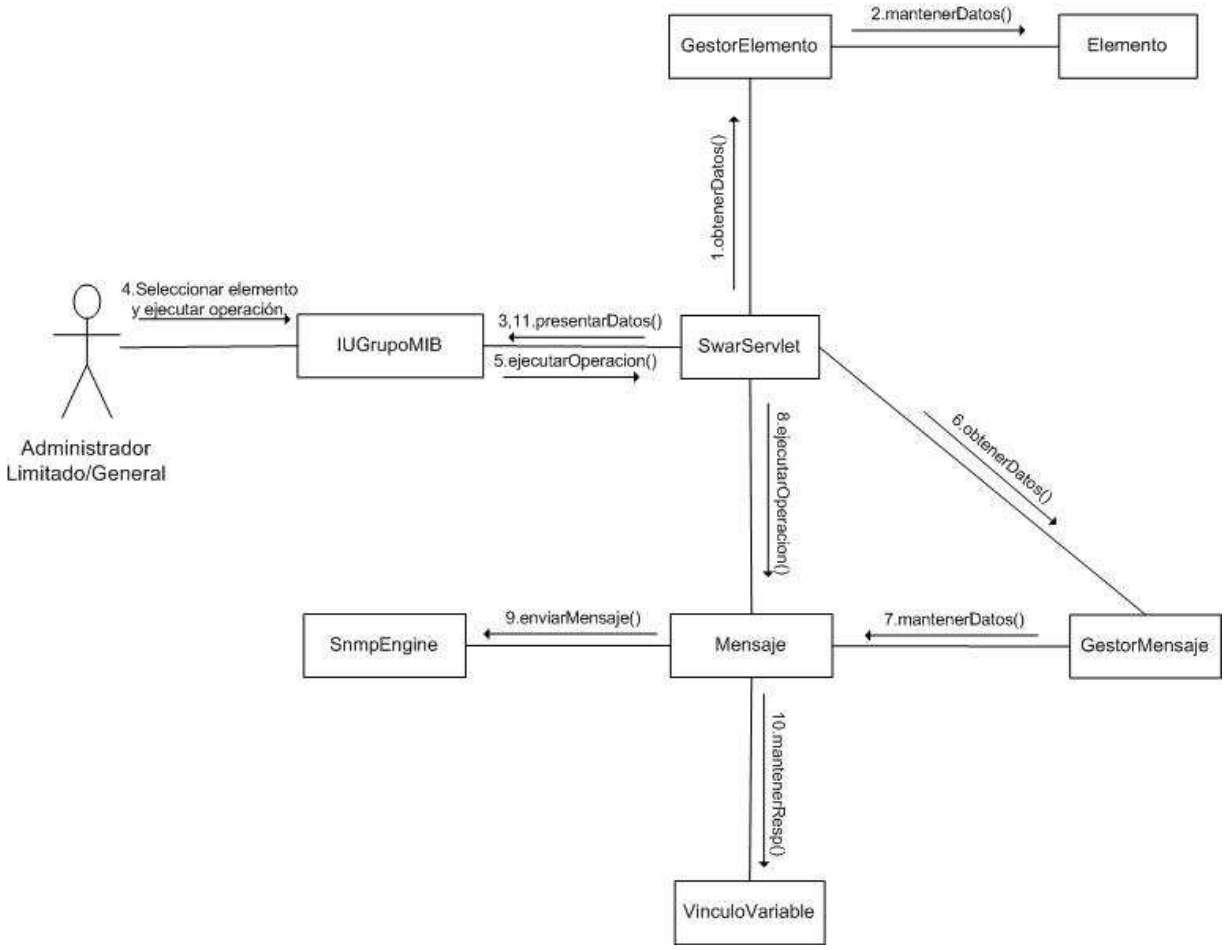


Figura 2. 38 Diagrama de colaboración del caso de uso: Ejecutar operación SNMPv3.

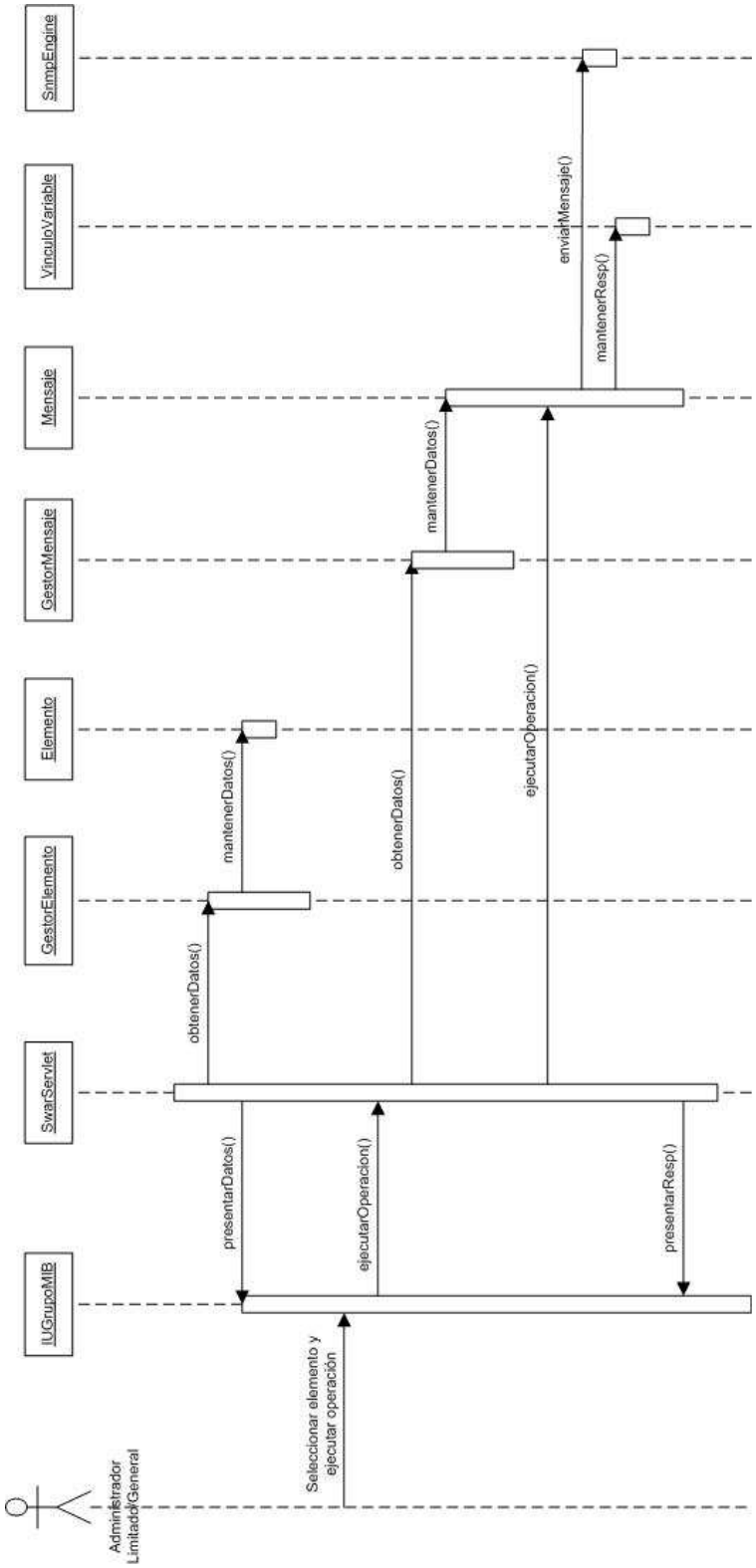


Figura 2. 39 Diagrama de secuencia del caso de uso: Ejecutar operación SNMPv3.

Caso de uso: Visualizar notificaciones.
Descripción: SwarServlet pide a SnmpEngine escuchar la llegada de notificaciones. SnmpEngine utiliza a ReceptorNotificaciones para capturar las notificaciones que llegan. ReceptorNotifcaciones pide a GestorNotificacion guardar la notificación capturada. Consecuentemente, administrador limitado/general solicita visualizar las notificaciones recibidas; para lo cual SwarServlet utiliza a GestorNotificacion para obtener las notificaciones recientes y pide a IUNotificaciones que las presente. Luego, el actor solicita actualizar para observar nuevas notificaciones. SwarServlet usa a GestorNotificacion para ello y se muestran los resultados en IUNotificaciones. Finalmente, el actor pide mostrar información detallada de una notificación. SwarServlet usa a GestorNotificacion para efectuar esta actividad.

Tabla 2. 50 Realización de Visualizar notificaciones.

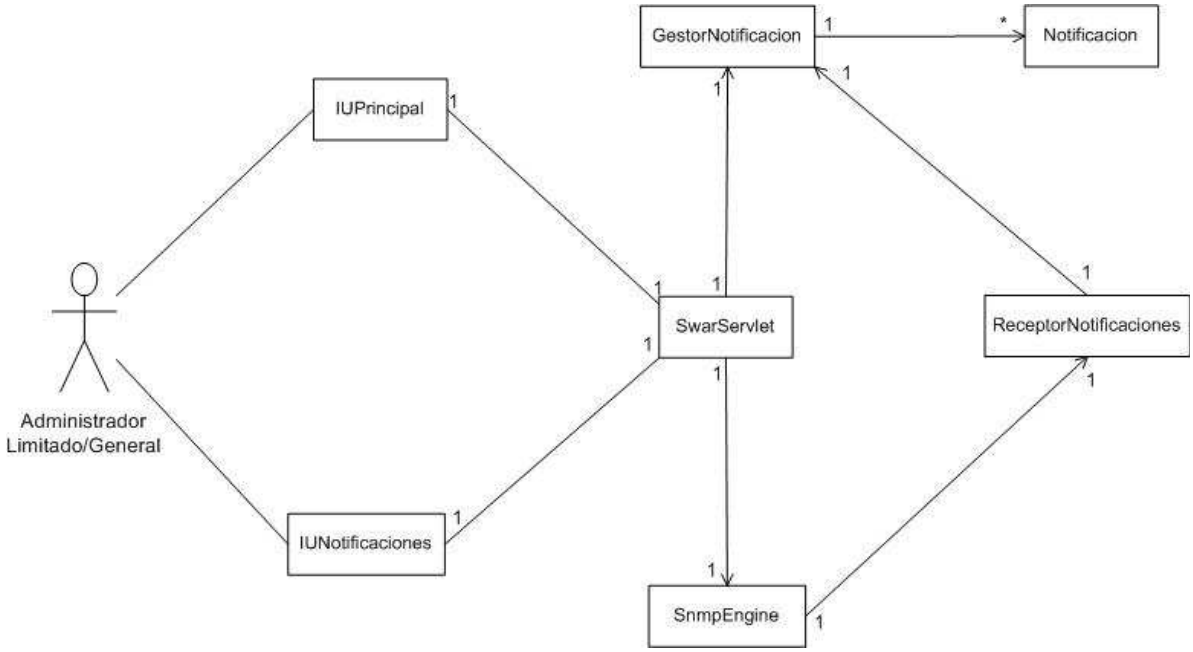


Figura 2. 40 Diagrama de clases del caso de uso: Visualizar notificaciones.

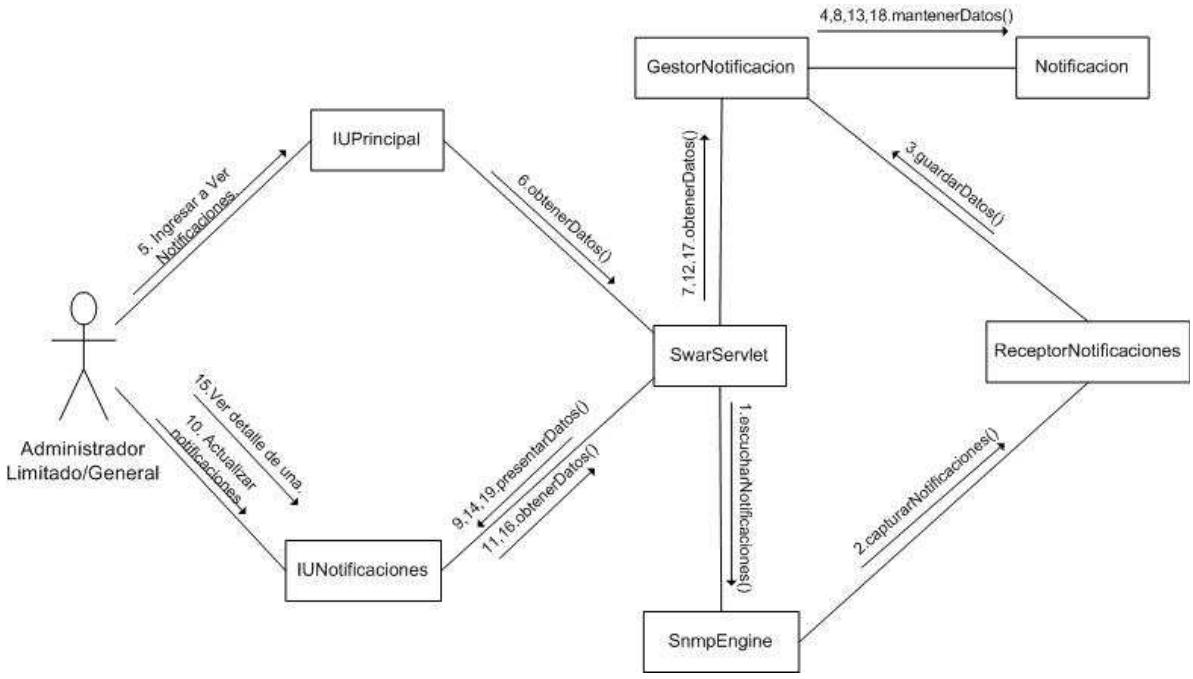


Figura 2. 41 Diagrama de colaboración del caso de uso: Visualizar notificaciones.

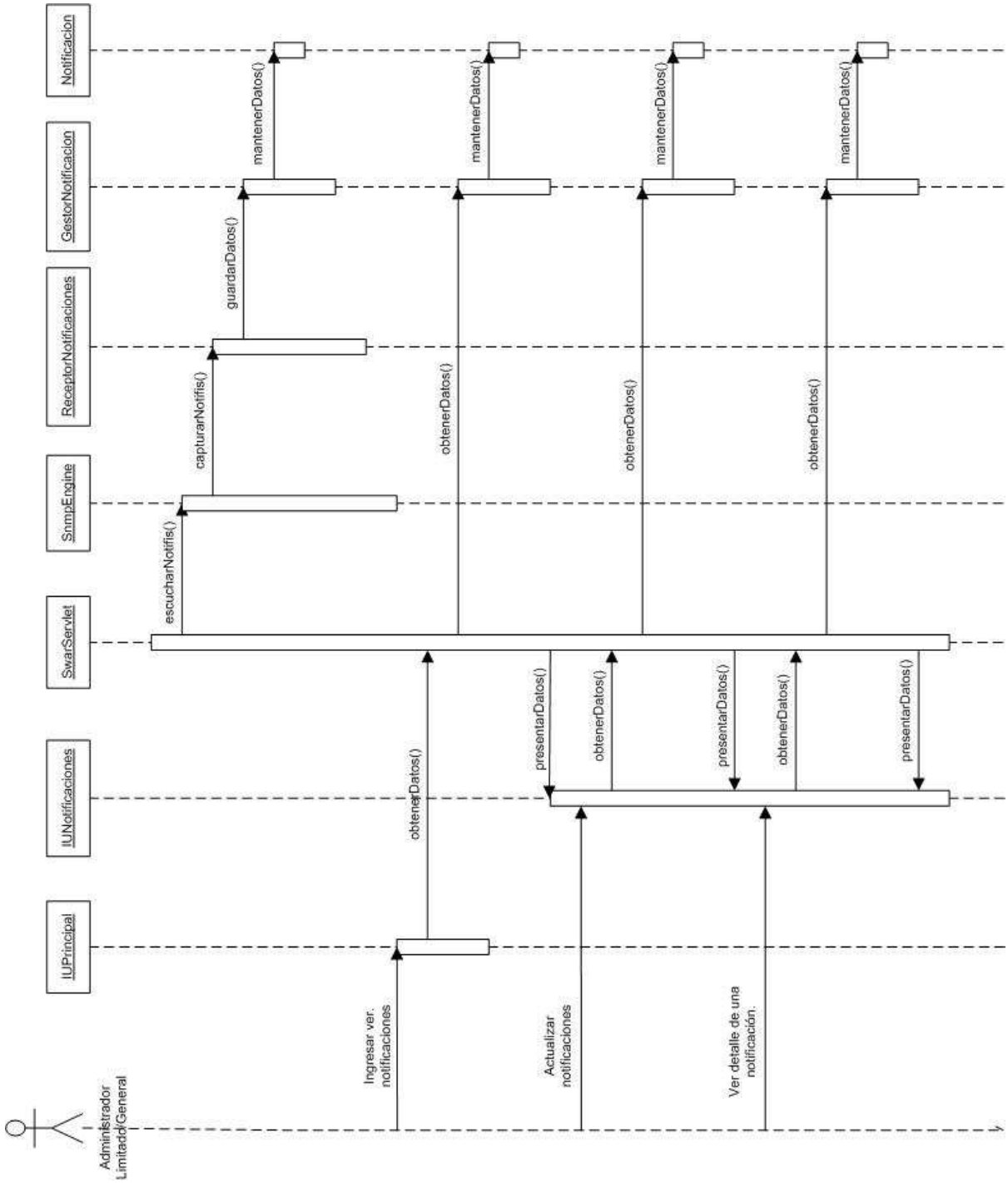


Figura 2. 42 Diagrama de secuencia del caso de uso: Visualizar notificaciones.

Caso de uso: Gestionar MIBs.
Descripción: Administrador General solicita desde IUPrincipal mostrar la página IUGestionMIBs. SwarServlet pide a GestorMIB recuperar la lista de grupos MIB y se la presenta al actor en IUGestionMIBs. Luego, administrador general solicita subir un nuevo archivo MIB a la aplicación. SwarServlet recibe el archivo, lo lee, extrae información y la almacena en MIB y en NodoMIB. SwarServlet solicita a GestorNodoMIB la creación de un archivo MIB-XML a partir de los datos anteriormente almacenados. SwarServlet utiliza a GestorMIB para agregar una nueva entrada a la lista de grupos MIB disponibles y se la presenta al actor. Finalmente, administrador general puede pedir que se elimine un archivo MIB. SwarServlet utiliza a GestorNodoMIB para eliminar el archivo MIB-XML y a GestorMIB para eliminar el grupo MIB. SwarServlet pide a GestorMIB para recuperar la lista actualizada de grupos MIB y se la presenta al actor.

Tabla 2. 51 Realización de Gestionar MIBs.

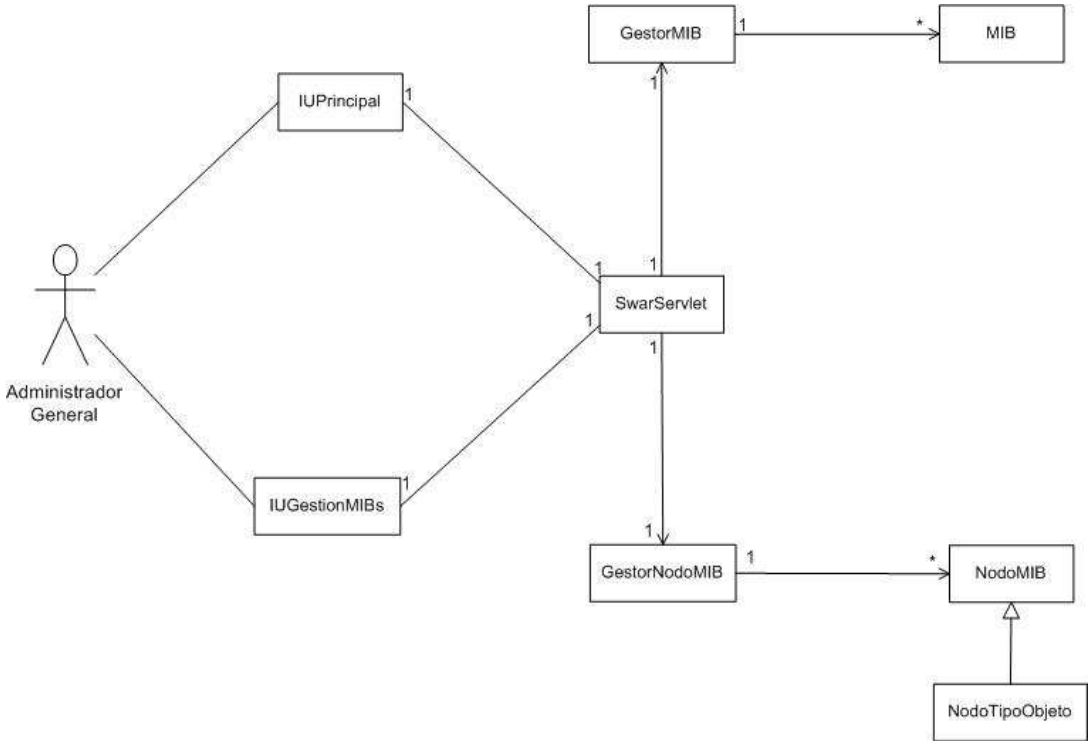


Figura 2. 43 Diagrama de clases del caso de uso: Gestionar MIBs.

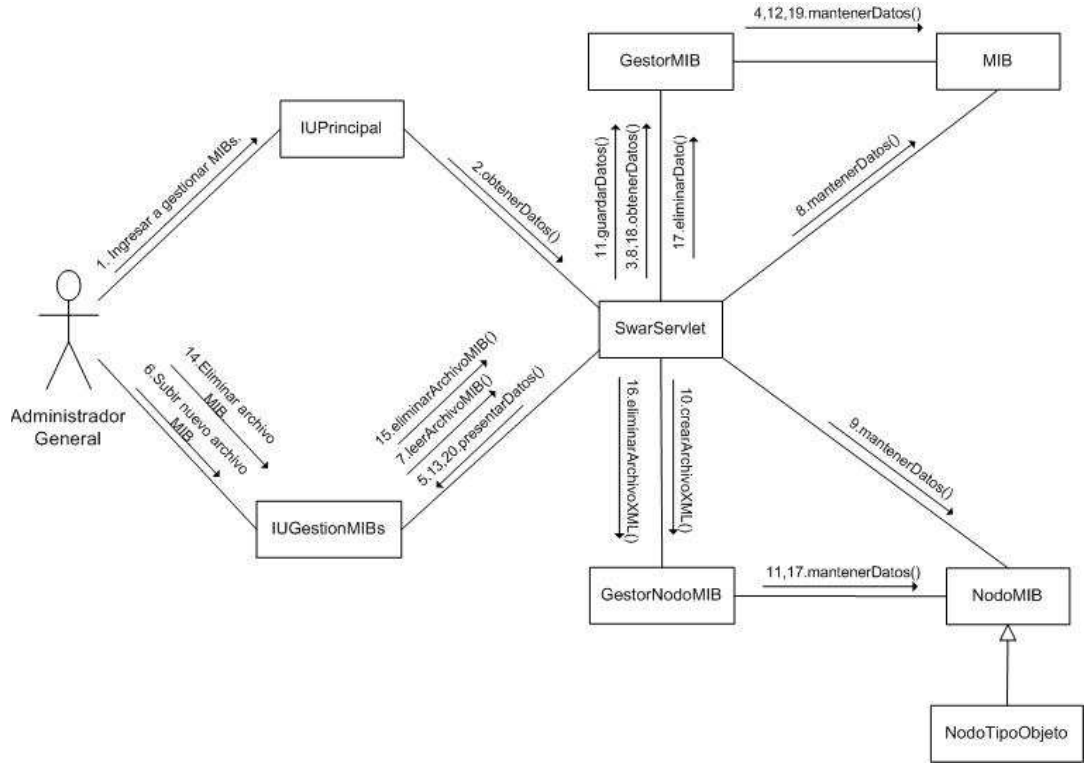


Figura 2. 44 Diagrama de colaboración del caso de uso: Gestionar MIBs.

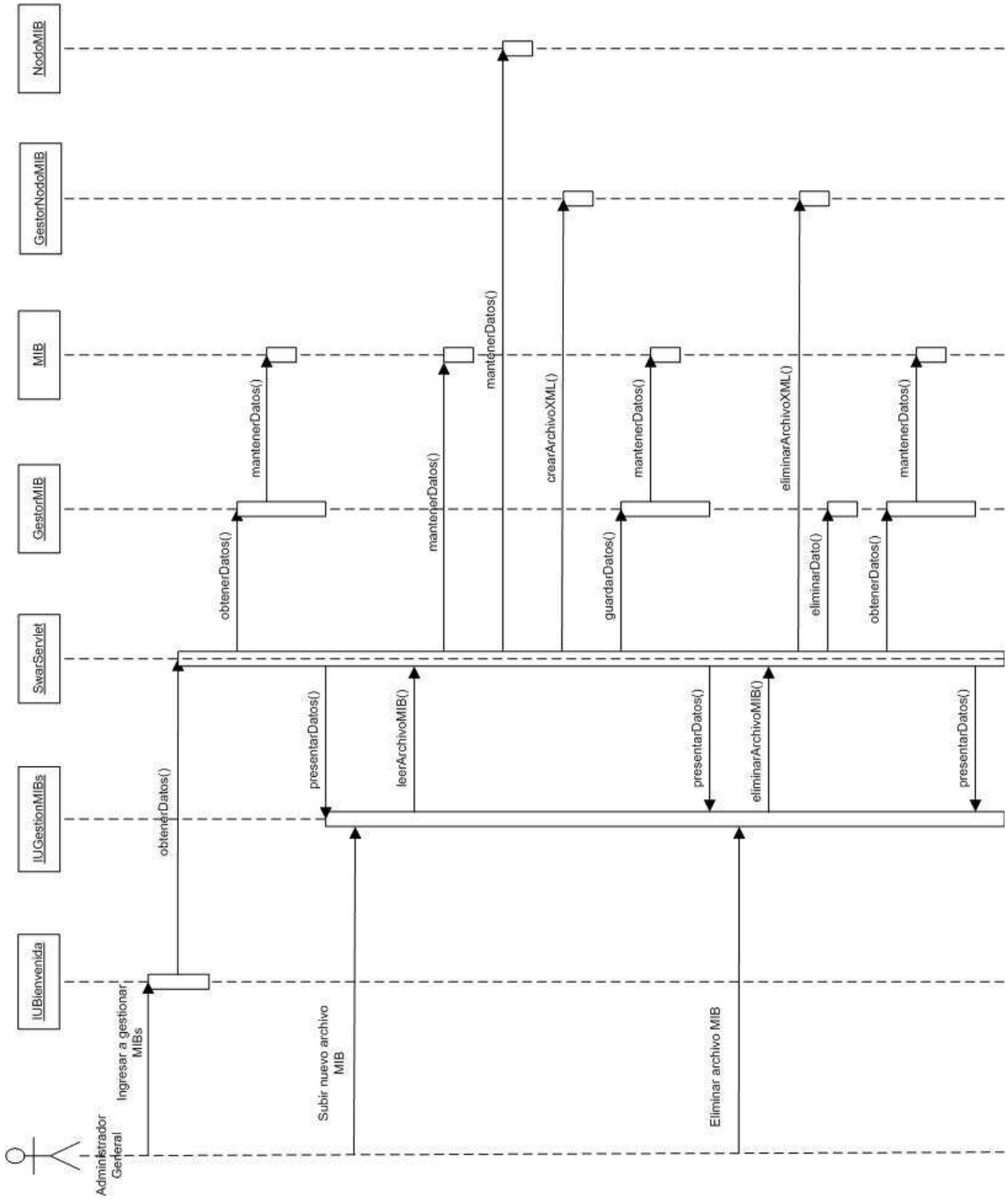


Figura 2. 45 Diagrama de secuencia del caso de uso: Gestionar MIBs.

Caso de uso: Gestionar usuarios.
Descripción: Administrador General solicita desde IUPrincipal mostrar la página IUGestionUsuarios. SwarServlet pide a GestorUsuario recuperar la lista de usuarios existentes y se la presenta al actor en IUGestionUsuarios. Administrador general solicita guardar un nuevo usuario. SwarServlet utiliza a GestorUsuario para guardar el nuevo usuario. SwarServlet pide a SnmpEngine agregar el nuevo usuario en su modelo de seguridad USM. IUGestionUsuarios presenta la lista de usuarios actualizada. Luego, administrador general pide realizar la exportación del usuario seleccionado SwarServlet pide a GestorUsuario, GestorGrupoVacm, GestorRed y GestorElemento recuperar la información necesaria para realizar la exportación. Desde IUExpAnuUsuario el actor selecciona los elementos y acepta iniciar la exportación. SwarServlet utiliza a GestorMensaje recuperar los parámetros de SNMPv3. Luego, SwarServlet asigna a ExportadorUsuario la tarea en si. ExportadorUsuario guarda los resultados temporalmente en RegistroUsuario. Luego, ExportadorUsuario pide a GestorRegistroUsuario guardar los resultados definitivos. Los resultados finales se presentan en IUExpAnuUsuario. Consecuentemente, administrador general solicita se realice la anulación de un usuario en cada elemento de red para luego eliminarlo de la aplicación. Para ello, SwarServlet pide a GestorUsuario, GestorGrupoVacm, GestorRed y GestorElemento recuperar la información necesaria para realizar la anulación. IUExpAnuUsuario muestra la información recuperada. El actor selecciona los elementos y acepta iniciar la anulación. SwarServlet utiliza a GestorMensaje recuperar los parámetros de SNMPv3. Luego, SwarServlet asigna a AnuladorUsuario la tarea en si. AnuladorUsuario pide a GestorRegistroUsuario eliminar los registros de usuarios ya anulados. Finalmente, SwarServlet solicita a IUExpAnuUsuario mostrar los resultados de la anulación.

Tabla 2. 52 Realización de Gestionar usuarios.

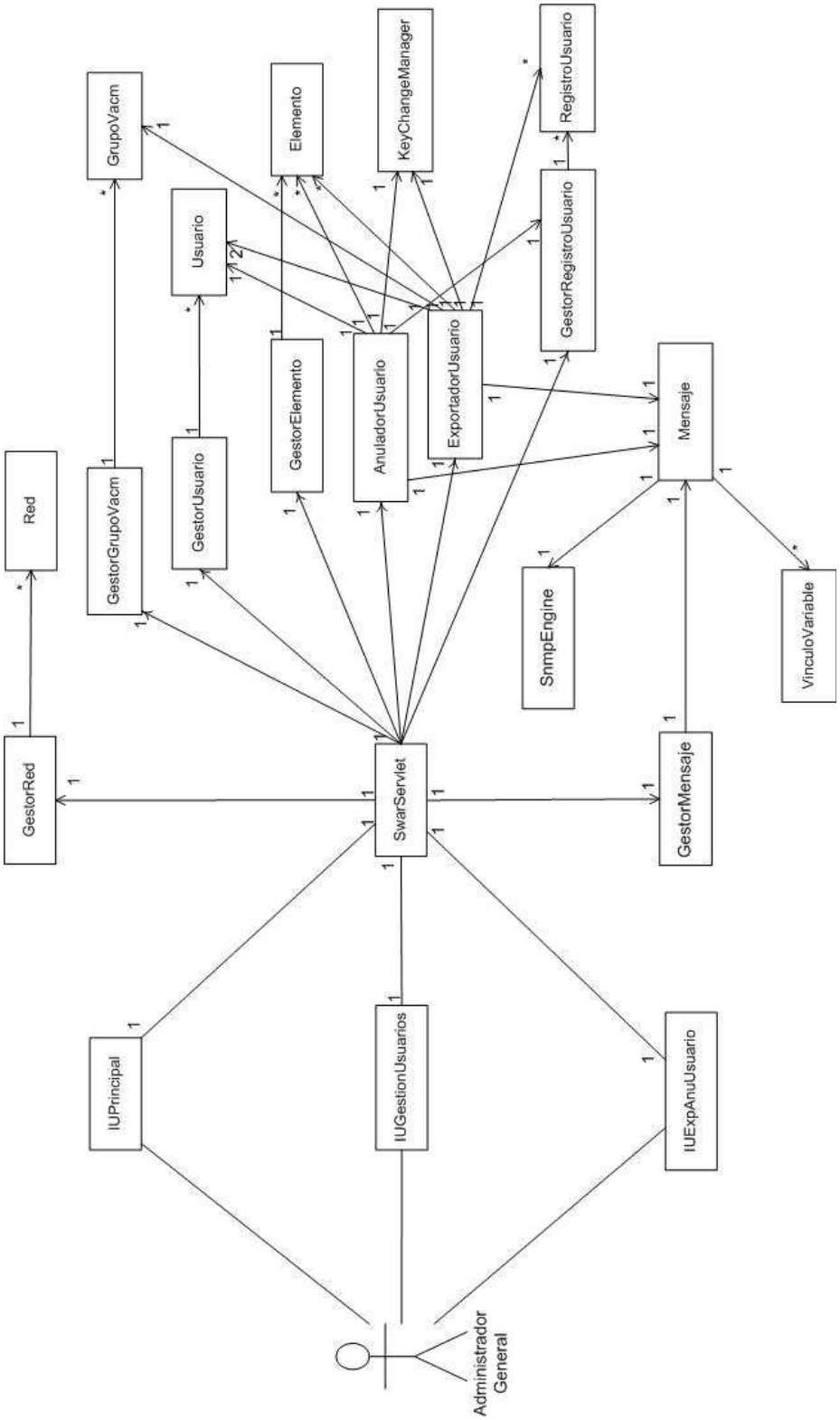


Figura 2. 46 Diagrama de clases del caso de uso: Gestionar Usuarios.

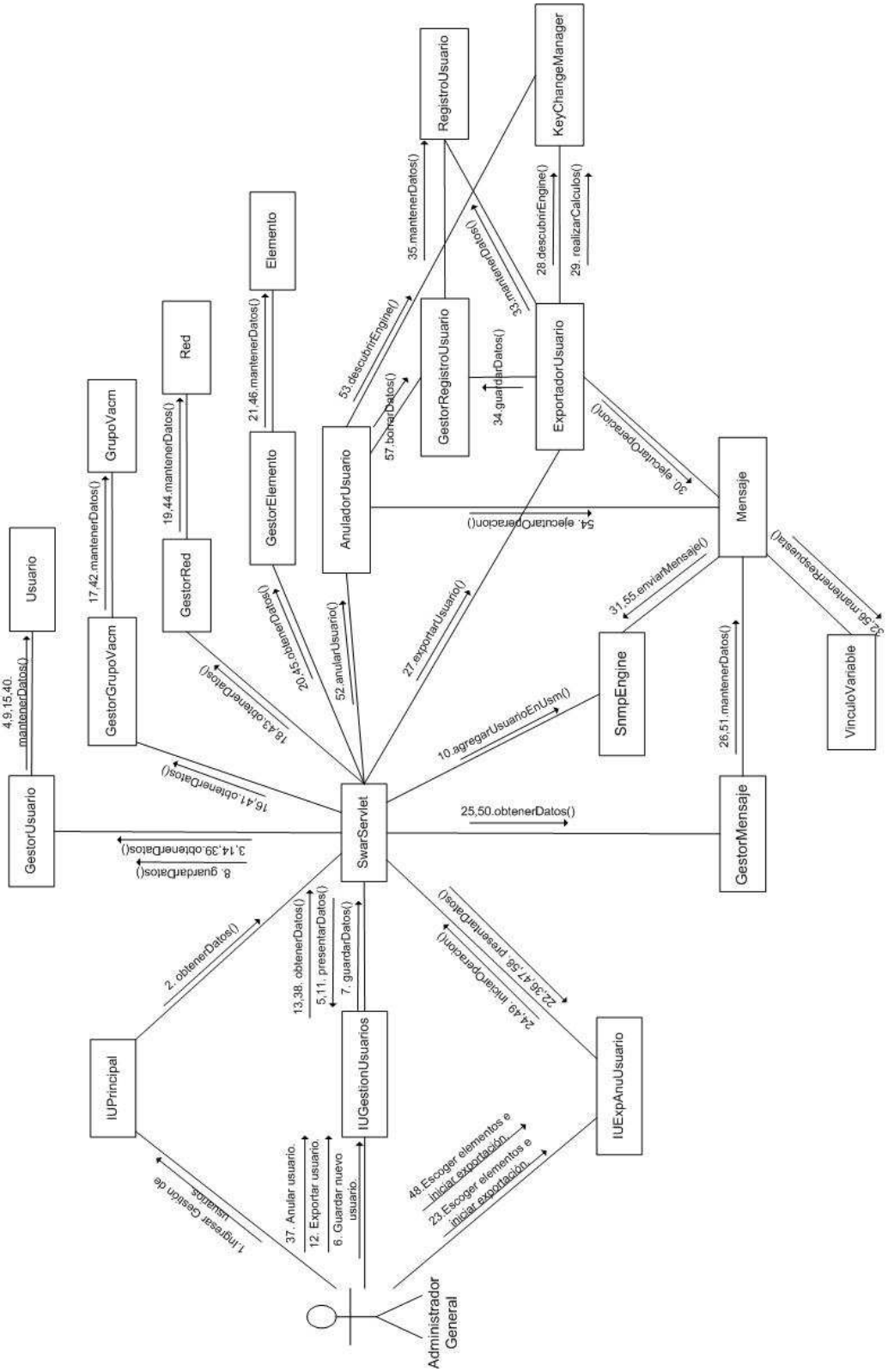


Figura 2. 47 Diagrama de colaboración del caso de uso: Gestionar Usuarios.

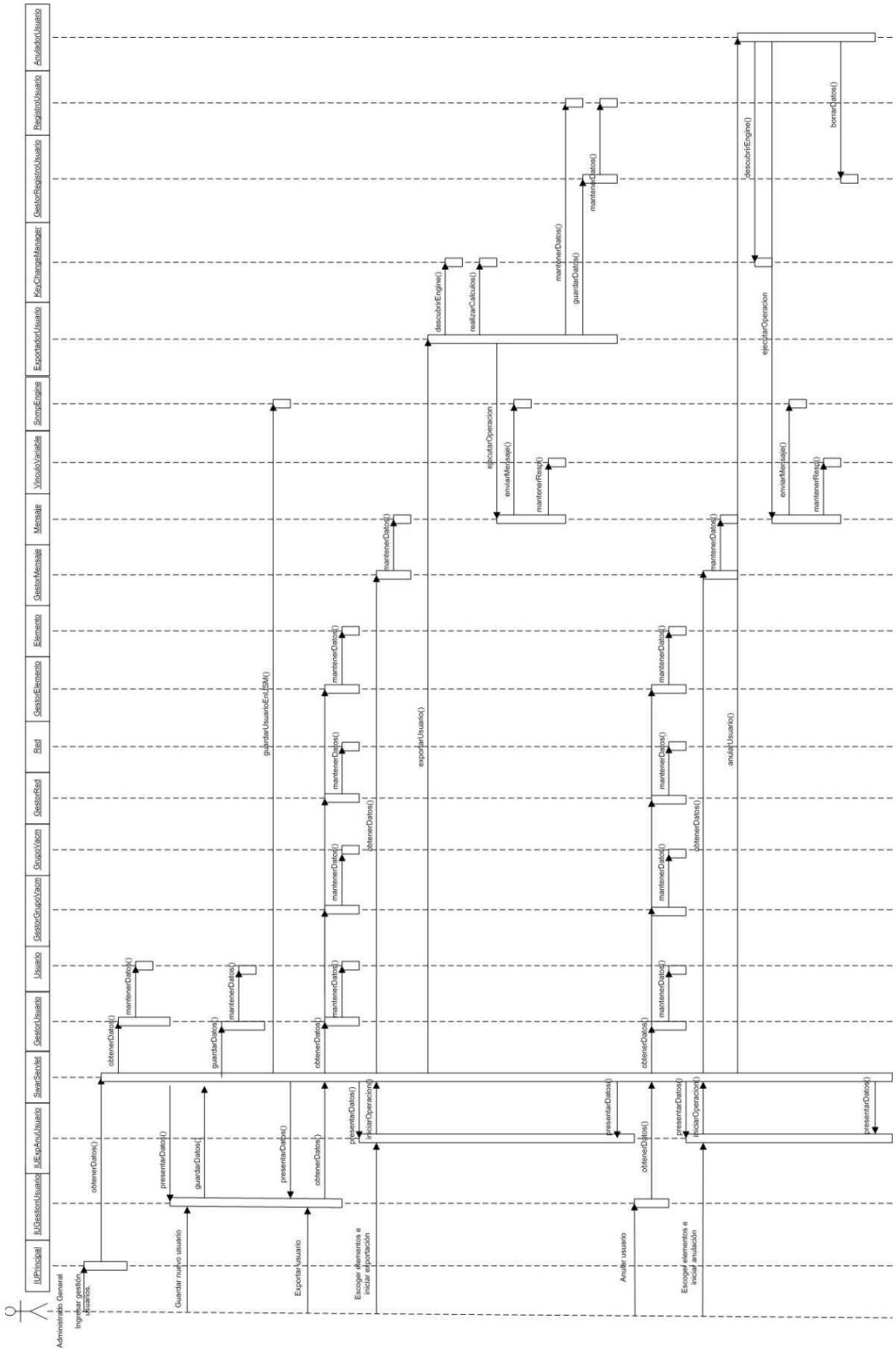


Figura 2. 48 Diagrama de secuencia del caso de uso: Gestionar Usuarios.

Caso de uso: Gestionar vistas.
Descripción: Administrador General solicita desde IUPrincipal mostrar la página IUGestionVistas. SwarServlet pide a GestorVista recuperar la lista de vistas existentes y se la presenta al actor. Administrador general solicita guardar una nueva vista. SwarServlet utiliza a GestorVista para guardar la nueva vista. IUGestionVistas presenta la lista de vistas actualizada. Administrador general pide realizar la exportación de la vista seleccionada. SwarServlet pide a GestorVista, GestorRed y GestorElemento recuperar la información necesaria para realizar la exportación. IUExpAnuVista muestra la información recuperada. El actor selecciona los elementos y acepta iniciar la exportación. SwarServlet utiliza a GestorMensaje recuperar los parámetros de mensaje SNMPv3. Luego, SwarServlet asigna a ExportadorVista la tarea en si. ExportadorVista guarda los resultados temporalmente en RegistroVist. Luego, ExportadorVista pide a GestorRegistroVista guardar los resultados definitivos. SwarServlet solicita a IUExpAnuVista presentar los resultados finales. Luego, administrador general solicita la anulación de una vista en cada elemento de red para luego eliminarlo de la aplicación. Para ello, SwarServlet pide a GestorVista, GestorRed y GestorElemento recuperar la información necesaria para realizar la anulación. IUExpAnuVista muestra la información recuperada. El actor selecciona los elementos y acepta iniciar la anulación. SwarServlet utiliza a GestorMensaje recuperar los parámetros de SNMPv3. Luego, SwarServlet asigna a AnuladorVista la tarea en si. AnuladorVista pide a GestorRegistroVista eliminar los registros de las vistas ya anuladas. Finalmente, SwarServlet solicita a IUExpAnuVista mostrar los resultados de la anulación.

Tabla 2. 53 Realización de Gestionar vistas.

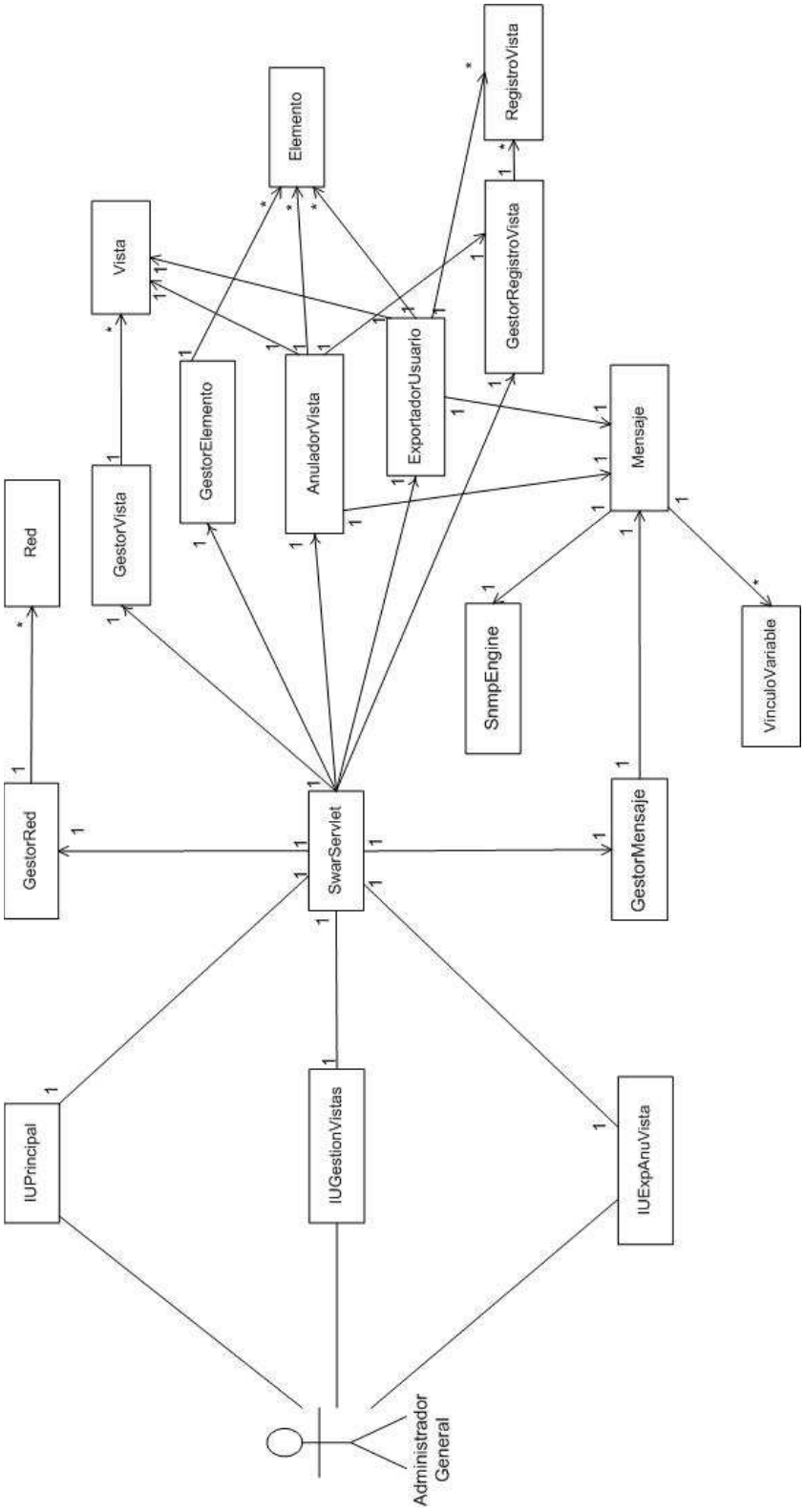


Figura 2. 49 Diagrama de clases del caso de uso: Gestionar vistas.

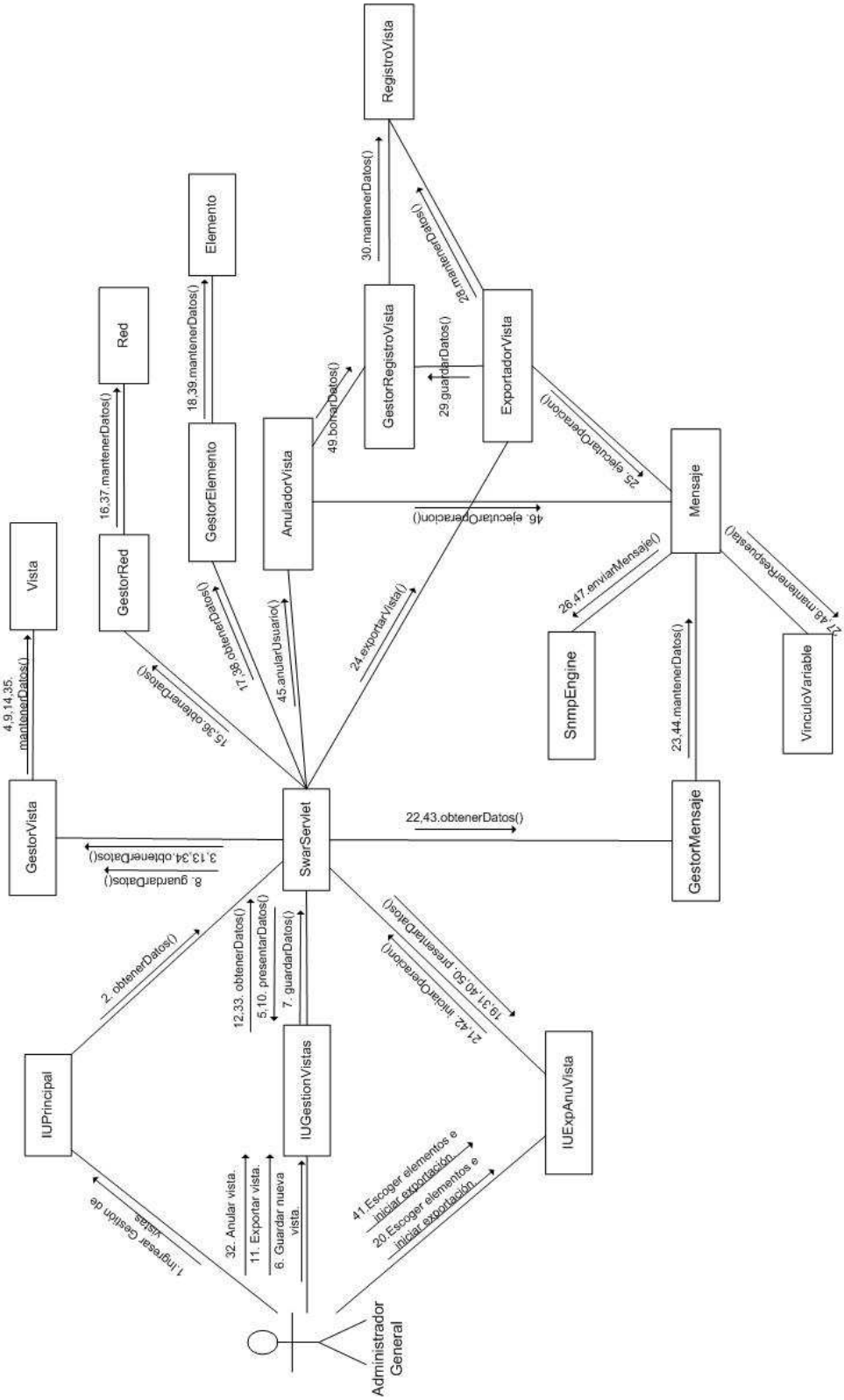


Figura 2. 50 Diagrama de colaboración del caso de uso: Gestionar vistas.

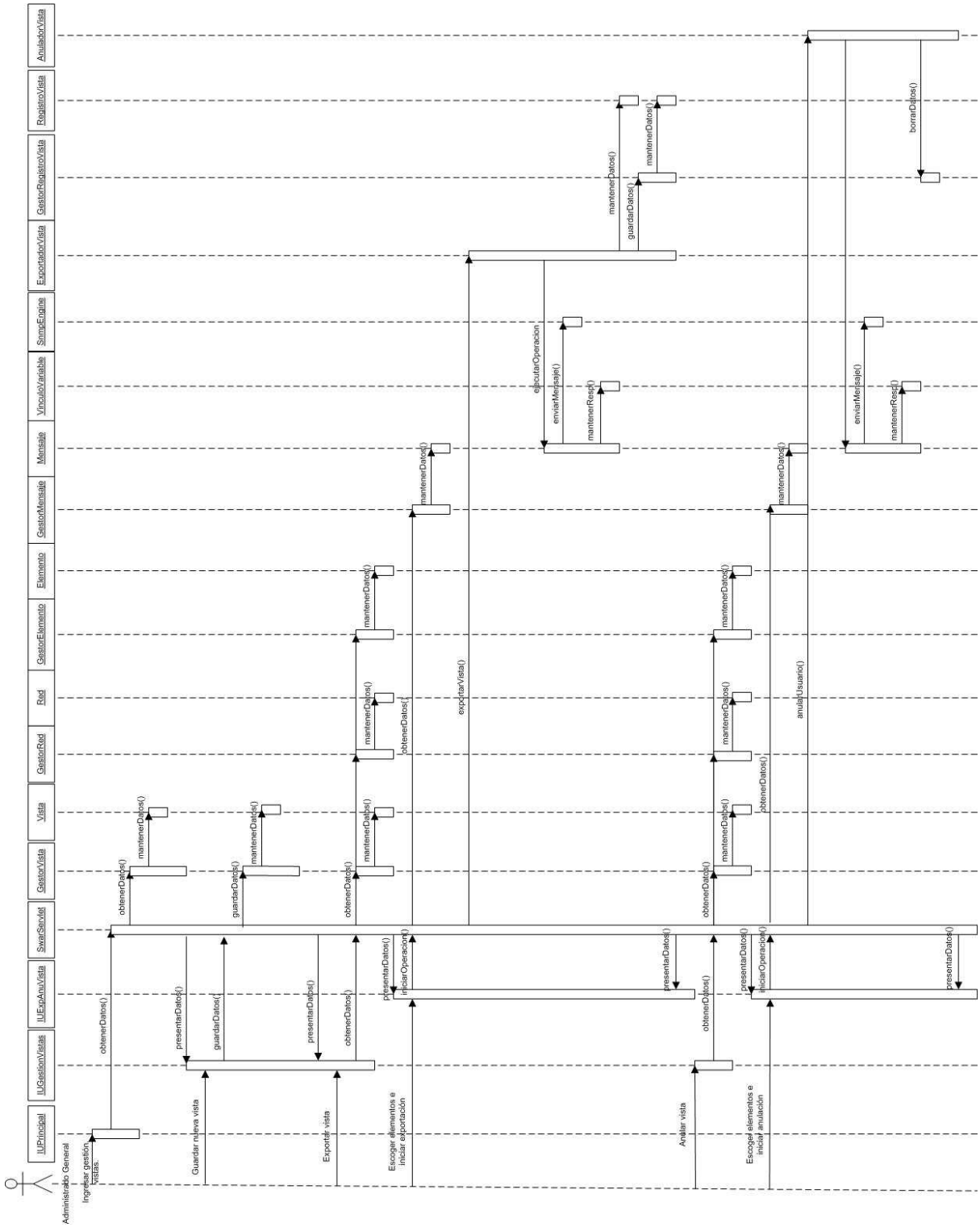


Figura 2. 51 Diagrama de secuencia del caso de uso: Gestionar vistas.

Caso de uso: Gestionar derechos de acceso.
Descripción: Administrador general solicita desde IUPrincipal mostrar la página IUGestionDerechos. SwarServlet pide a GestorDerecho recuperar la lista de derechos existentes, a GestorGrupoVacm recuperar los grupos y a GestorVista recuperar las vistas. IUGestionDerechos presenta al actor la lista de derechos de acceso. Administrador general observa la lista de derechos y solicita guardar uno nuevo. SwarServlet utiliza a GestorDerecho para guardar el nuevo derecho. IUGestionDerechos presenta la lista de derechos actualizada. Luego, administrador general pide realizar la exportación del derecho de acceso seleccionado. SwarServlet pide a GestorDerecho, GestorGrupoVacm, GestorVista, GestorRed y GestorElemento recuperar la información necesaria para realizar la exportación. IUExpAnuDerecho muestra la información recuperada. El actor selecciona los elementos y acepta iniciar la exportación. SwarServlet utiliza a GestorMensaje recuperar los parámetros de SNMPv3. Luego, SwarServlet asigna a ExportadorDerecho la tarea en si. ExportadorDerecho guarda los resultados temporalmente en RegistroDerecho. Luego, ExportadorDerecho pide a GestorRegistroDerecho guardar los resultados definitivos. SwarServlet solicita a IUExpAnuDerecho presentar los resultados finales. Opcionalmente, administrador general puede pedir que se realice la anulación de un derecho de acceso en cada elemento de red para luego eliminarlo de la aplicación. Para ello, SwarServlet pide a GestorDerecho, GestorGrupoVacm, GestorVista, GestorRed y GestorElemento recuperar la información necesaria para realizar la anulación. IUExpAnuDerecho muestra la información recuperada. El actor selecciona los elementos y acepta iniciar la anulación. SwarServlet utiliza a GestorMensaje recuperar los parámetros de SNMPv3. Luego, SwarServlet asigna a AnuladorDerecho la tarea en si. AnuladorDerecho pide a GestorRegistroDerecho eliminar los registros de los derechos ya anulados. Finalmente, SwarServlet solicita a IUExpAnuDerecho mostrar los resultados de la anulación.

Tabla 2. 54 Realización de Gestionar derechos de acceso.

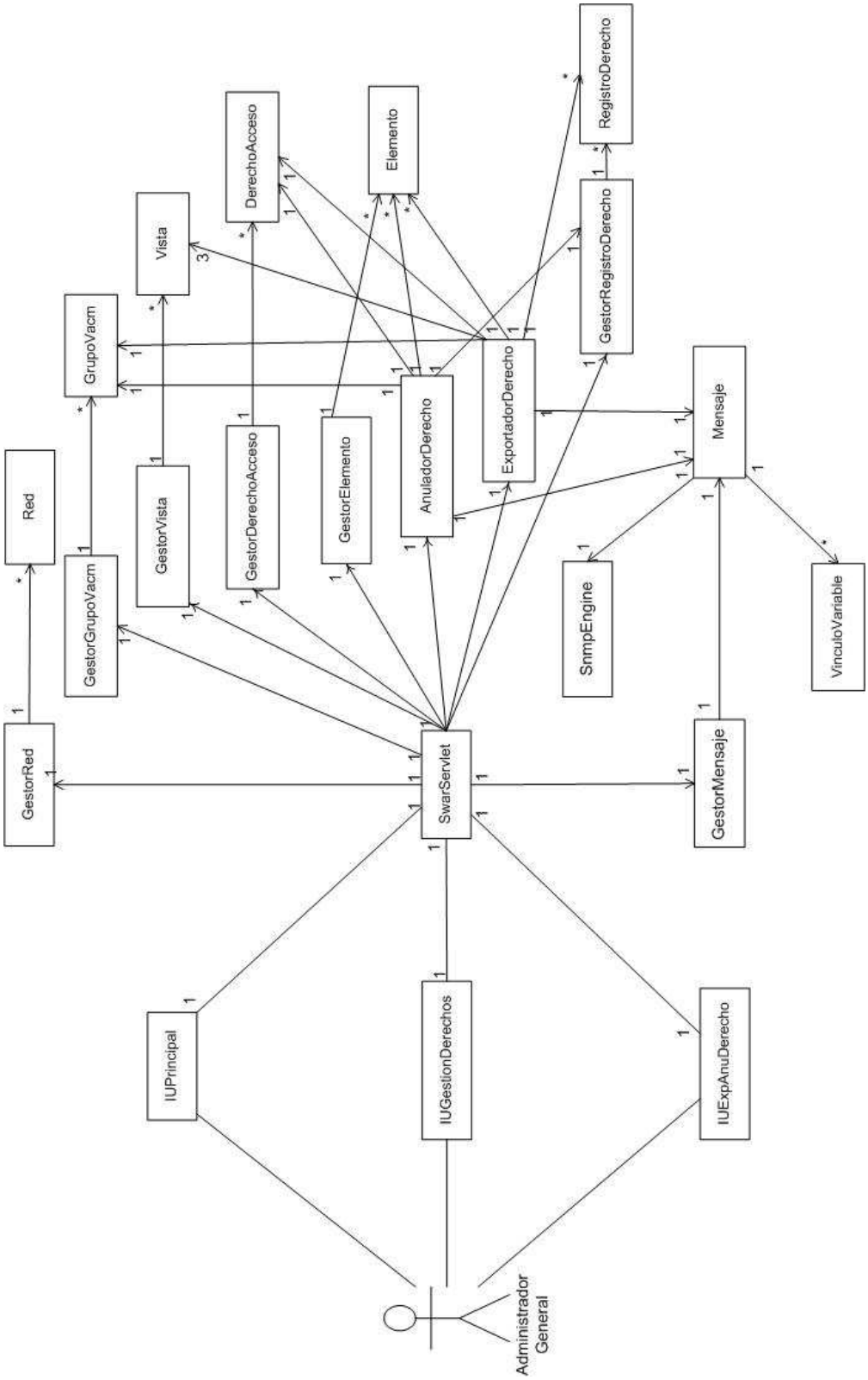


Figura 2. 52 Diagrama de clases del caso de uso: Gestionar derechos de acceso.

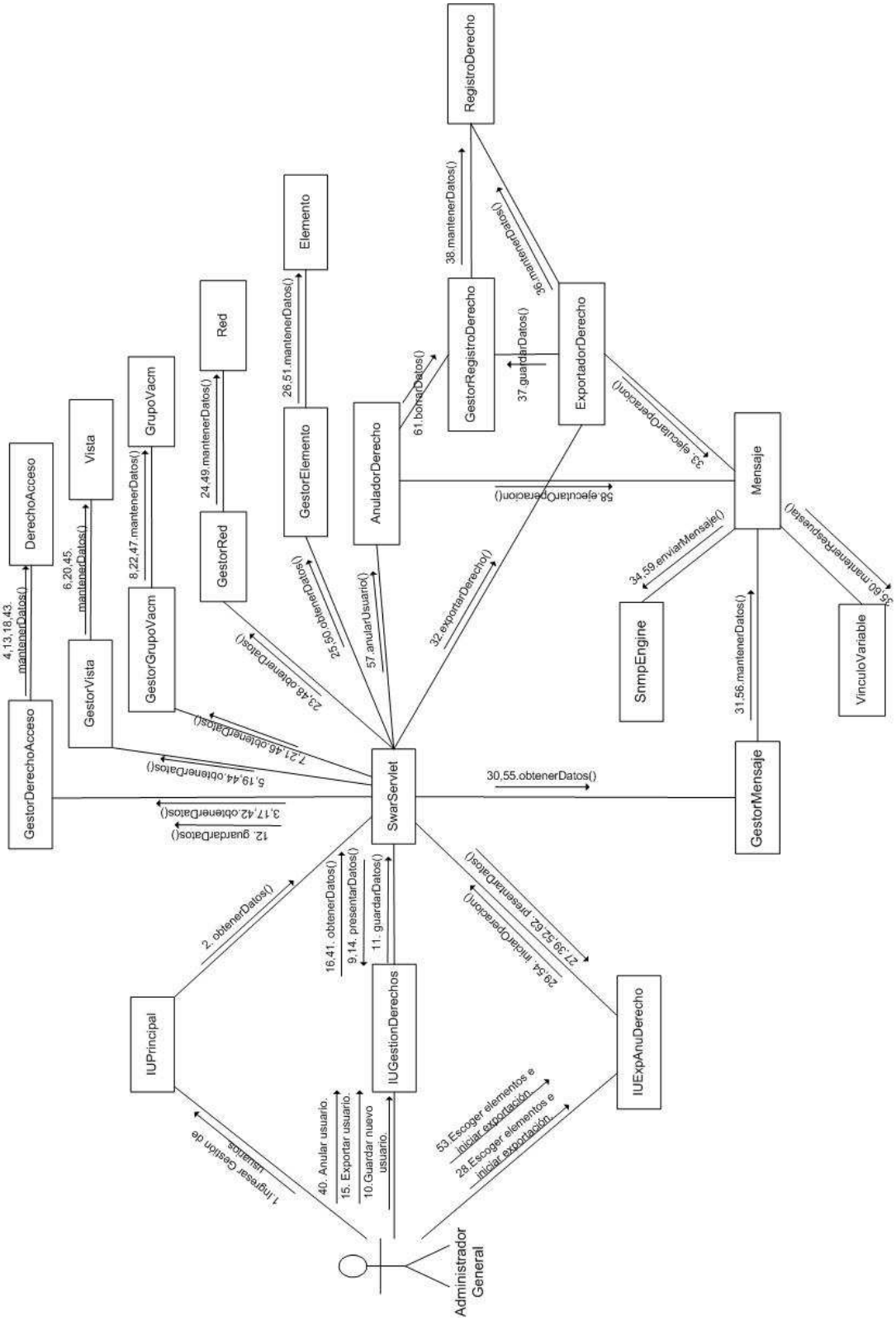


Figura 2. 53 Diagrama de colaboración del caso de uso: Gestionar derechos de acceso.

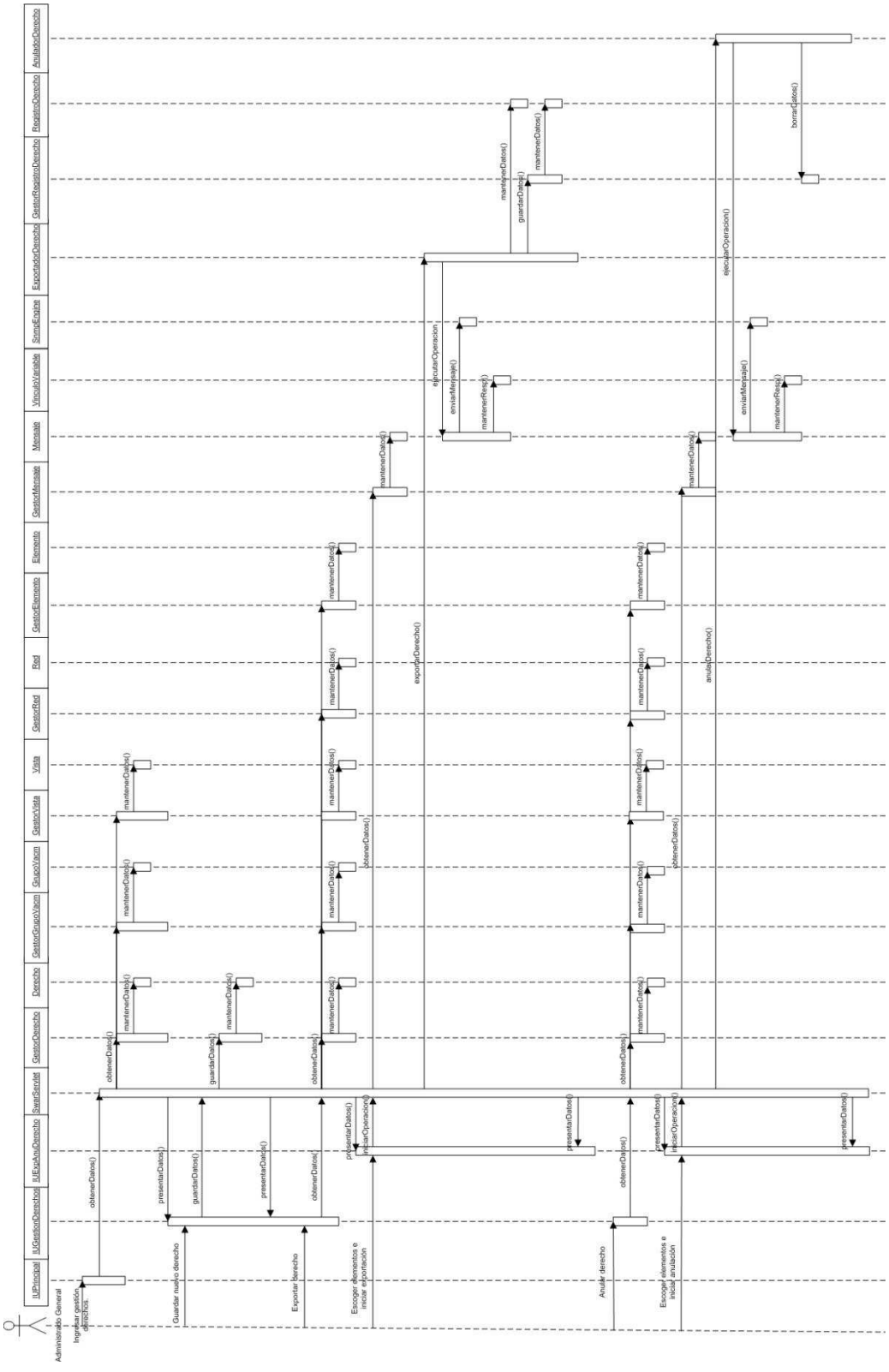


Figura 2. 54 Diagrama de secuencia del caso de uso: Gestionar derechos de acceso.

Caso de uso: Cambiar passwords.
Descripción: Administrador limitado/general solicita desde IUPrincipal mostrar la página IUCambioPasswords. SwarServlet pide a IUCambioPasswords mostrar el formulario. El actor ingresa los actuales y los nuevos passwords y luego solicita a la aplicación que realice el cambio. SwarServlet ordena a GestorUsuario a actualizar los passwords. Luego, el actor pide realizar la exportación del cambio de passwords. SwarServlet pide a GestorUsuario, GestorRed y GestorElemento recuperar la información necesaria para realizar la exportación. IUExpCambioPwds muestra la información recuperada. El actor selecciona los elementos y acepta iniciar la exportación. SwarServlet utiliza a GestorMensaje recuperar los parámetros de SNMPv3 (18-21). Luego, SwarServlet asigna a ExportadorPasswords la tarea en si. ExportadorPasswords utiliza a KeyChangeManager para realizar los cálculos requeridos para obtener los valores que se necesitan para establecer el cambio de passwords en la MIB de cada elemento seleccionado. ExportadorPasswords usa a Mensaje para ejecutar operaciones SNMP de establecimiento y guarda temporalmente los resultados de todo el proceso en RegistroPasswords. Luego, ExportadorPasswords pide a GestorRegistroPasswords guardar los resultados definitivos. SwarServlet solicita a IUExpAnuPasswords presentar los resultados finales. Finalmente, el actor ordena a la aplicación activar los cambios en SnmpEngine. SwarServlet pide a GestorUsuario recuperar los datos necesarios para la activación. SwarServlet ordena a SnmpEngine a actualizar los passwords.

Tabla 2. 55 Realización de Cambiar passwords.

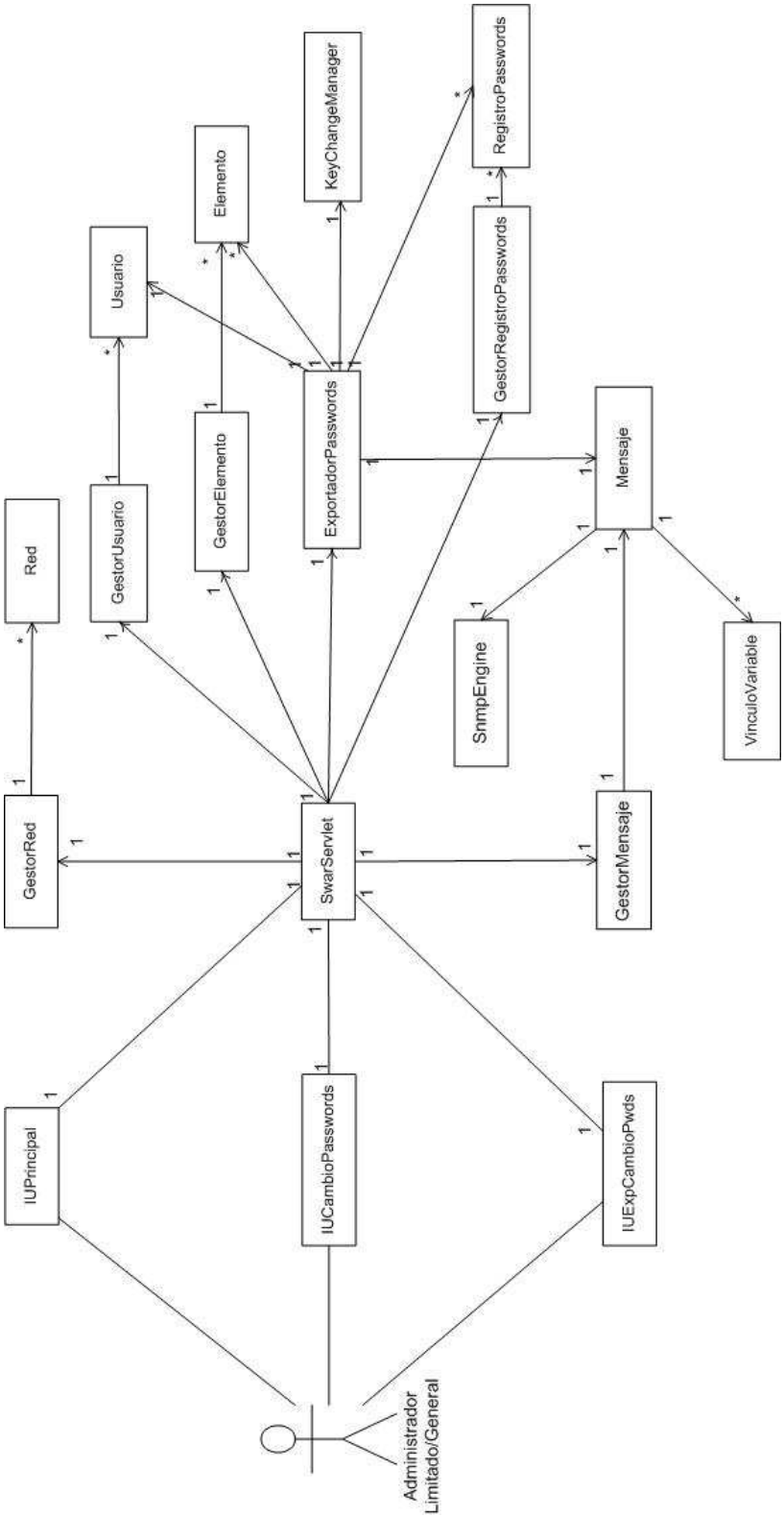


Figura 2. 55 Diagrama de clases del caso de uso: Cambiar passwords.

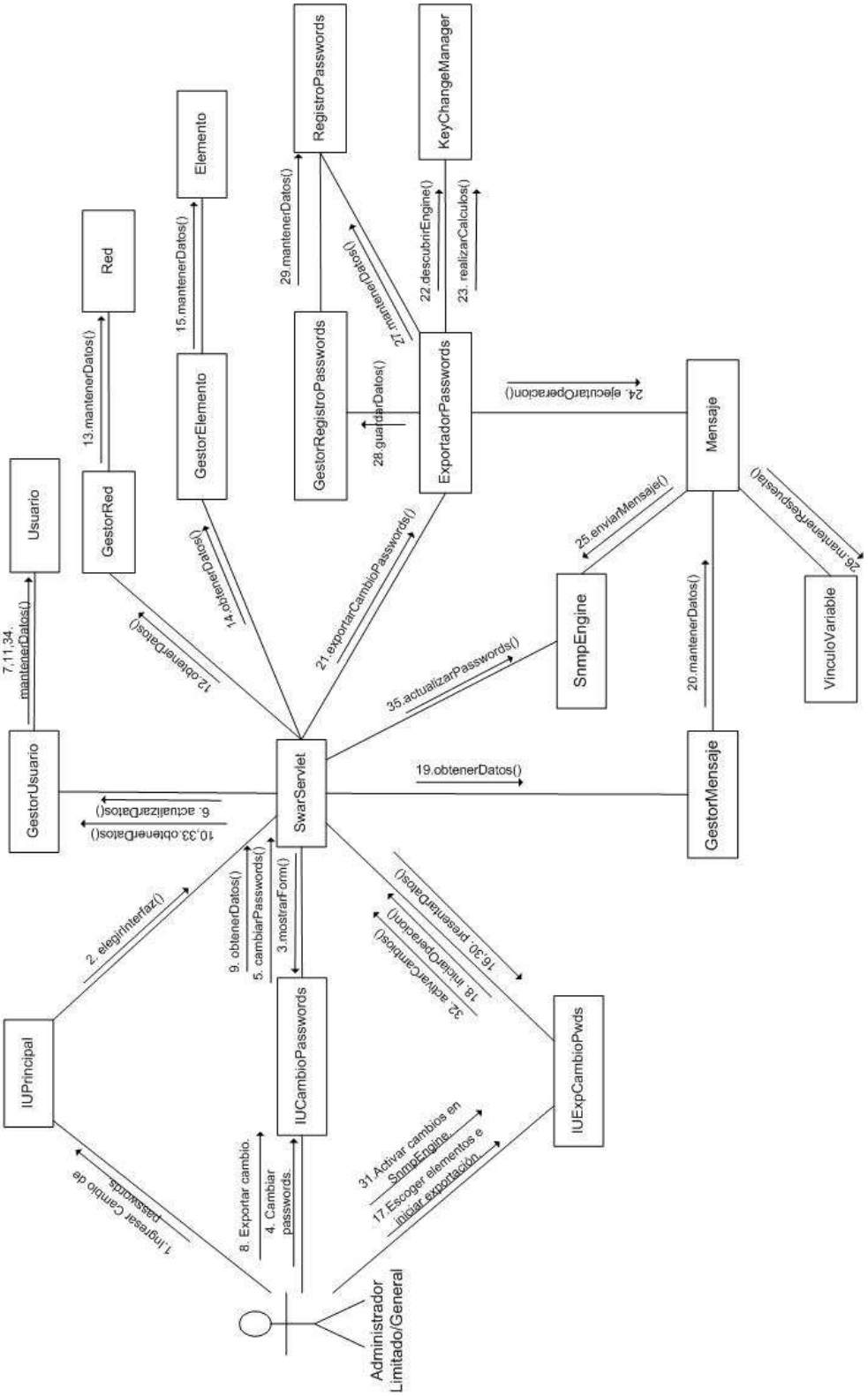


Figura 2. 56 Diagrama de colaboración del caso de uso: Cambiar passwords.

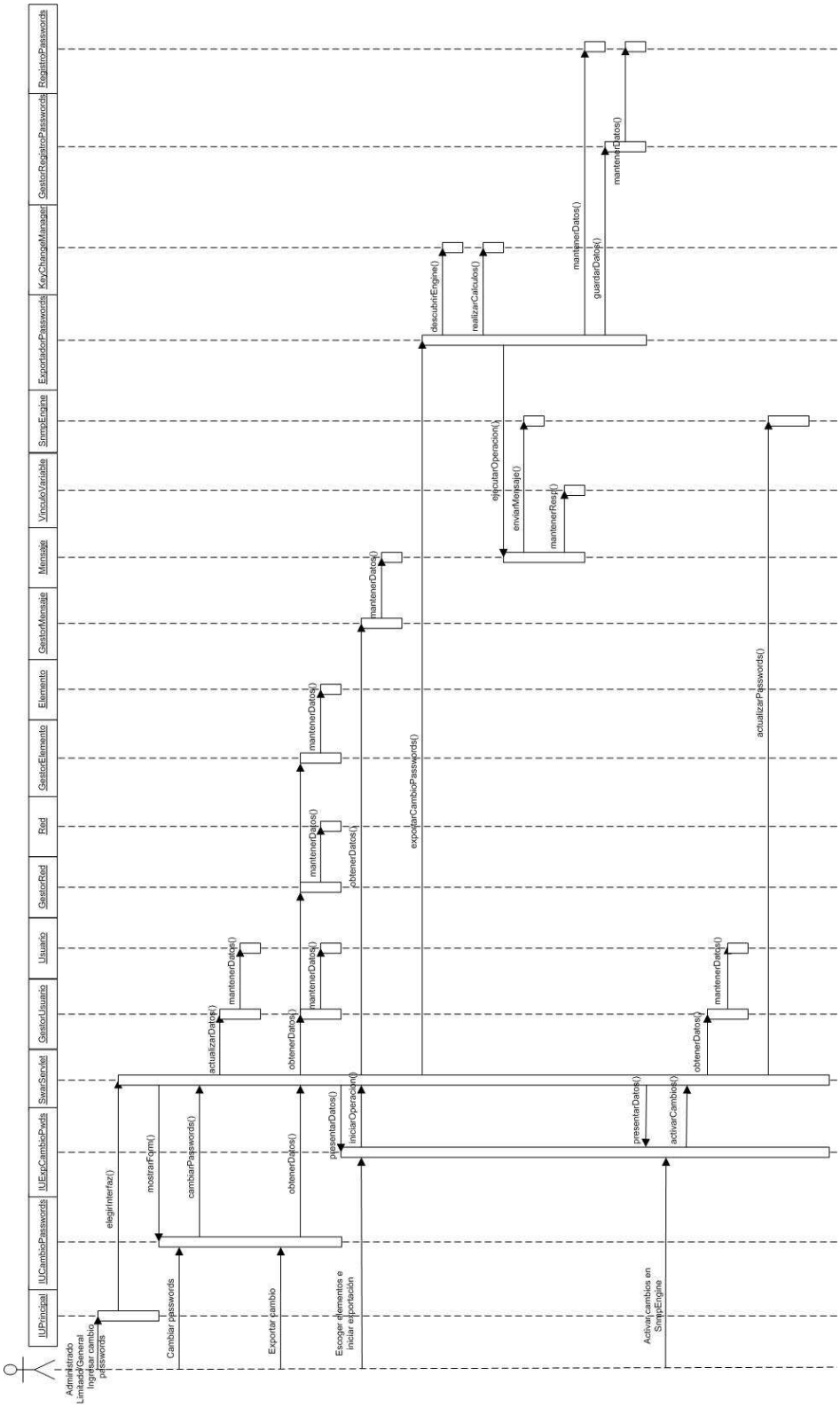


Figura 2. 57 Diagrama de secuencia del caso de uso: Cambiar passwords.

2.2.1.2 Clases de diseño

La definición de clases de diseño comprende el establecimiento de atributos, métodos y relaciones de acuerdo al papel que éstas cumplen dentro de la realización de casos de uso. En la siguiente tabla se muestra el formato de definición de clase de diseño y luego un ejemplo de definición de clase. La definición de las clases restantes se la realiza en la sección A de ANEXOS.

Clase: Nombre de la clase.
Atributos: Atributos de la clase.
Métodos: Métodos y funciones de la clase.

Tabla 2. 56 Modelo de clase de diseño.

Clase: Usuario.
Atributos: o securityName: Nombre de seguridad de usuario. o protocolAuth: Protocolo de autenticación. o protocolPriv: Protocolo de privacidad. o passwordAuth: Password actual asociado al protocolo de autenticación. o passwordPriv: Password actual asociado al protocolo de privacidad. o oldPasswordAuth: Password anterior asociado al protocolo de autenticación. o oldPasswordPriv: Password anterior asociado al protocolo de privacidad. o idGrupoVACM: Identificador de Grupo VACM al cual pertenece el usuario. o tipoAdministrador: Clasificación de administrador: limitado o general.
Métodos: o Usuario: Constructor de la clase Usuario. Crea un objeto Usuario.

Tabla 2. 57 Ejemplo de definición de una clase de diseño.

Figura 2. 58 Ejemplo de gráfico de clase de diseño.

2.2.1.3 Subsistemas de diseño

Los subsistemas de diseño son una forma de organizar los componentes del modelo de diseño con la finalidad de facilitar el manejo y mantenimiento del sistema. La definición de los susbsistemas de diseño parte desde los paquetes de análisis. La siguiente tabla muestra los subsistemas de diseño para la aplicación:

Subsistema de diseño	Descripción
Gestión de sesión.	Maneja información relacionada al inicio de sesión.
Gestión de configuración.	Maneja información relacionada a la configuración de parámetros de mensajes SNMPv3 y parámetros de servidor.
Gestión de elementos.	Maneja información relacionada a los elementos de red.
Gestión de MIBs.	Maneja información relacionada a grupos MIB.
Gestión de notificaciones.	Maneja información relacionada a notificaciones recibidas.
Gestión general.	Maneja información de usuarios, vistas, MIBs y derechos de acceso.
Gestión de cambio de passwords.	Maneja información relacionada a cambios de passwords del usuario.

Tabla 2. 58 Subsistemas de diseño para la aplicación.

También se definen **subsistemas de servicio**. Cuando existen clases de diseño que participan en más de un subsistema de diseño, entonces se crean los subsistemas de servicio. Los subsistemas de servicio se originan desde los paquetes de servicio y por lo tanto tienen estructura idéntica, así también como sus relaciones de dependencia. Ver desde la Figura 2.15 hasta la Figura 2.22.

2.2.1.4 Diseño de la interfaz gráfica

La interfaz gráfica del usuario es el componente de diseño a través del cual administrador limitado/general puede interactuar con la aplicación.

Las clases de análisis: IUBienvenida, IUPrincipal, IUInicioSesion, IUConfigMensajes, IUConfigServidor, IUDescubHosts, IUDescubHostsResult, IUElementos, IUInfoElemento, IUGrupoMIB, IUNotificaciones, IUGestionMIBs, IUGestionUsuarios, IUExpAnuUsuario, IUGestionVistas, IUExpAnuVista, IUGestionDerechos, IUExpAnuDerecho, IUCambioPasswords e IUExpCambioPwds, constituyen la base de definición de las interfaces de la aplicación.

Consecuentente, se define el subsistema de diseño que incluye a las interfaces de usuario. Se identifica un subsistema de diseño: IU de Administrador limitado/general, donde las clases: IUGestionMIBs, IUGestionUsuarios, IUExpAnuUsuario, IUGestionVistas, IUExpAnuVista, IUGestionDerechos y IUExpAnuDerecho son manejadas únicamente por administrador general. La figura siguiente muestra la estrutuctura de este subsistema:

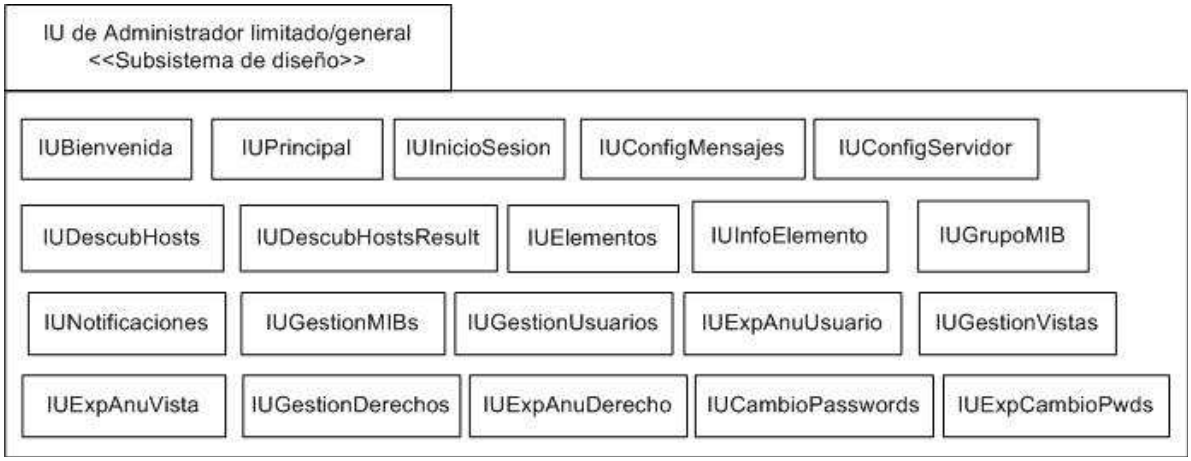


Figura 2. 59 Subsistema de diseño IU de Administrador limitado/general.

Dado que uno de los requerimientos no funcionales era que la interfaz de la aplicación sea web, entonces el diseño de los componentes gráficos corresponderá al diseño estructural gráfico de páginas web. Las siguientes figuras detallan las estructuras de las interfaces gráficas.

Figura 2. 60 Diseño de interfaz gráfica de usuario para IUBienvenida.

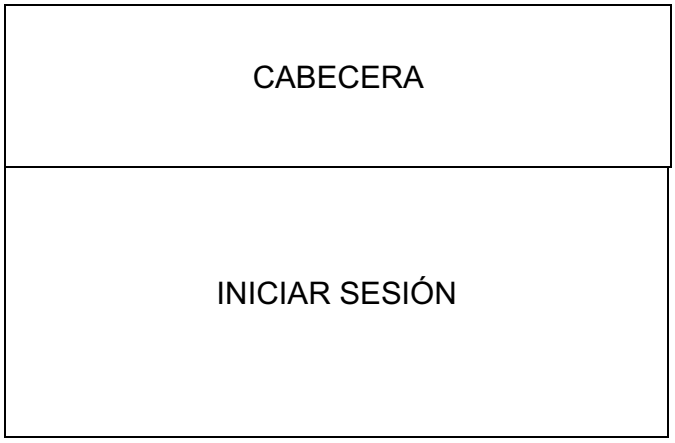


Figura 2. 61 Diseño de interfaz gráfica de usuario para IUInicioSesion

Figura 2. 62 Diseño de interfaz gráfica de usuario para IUElementos y IUGrupoMIB.

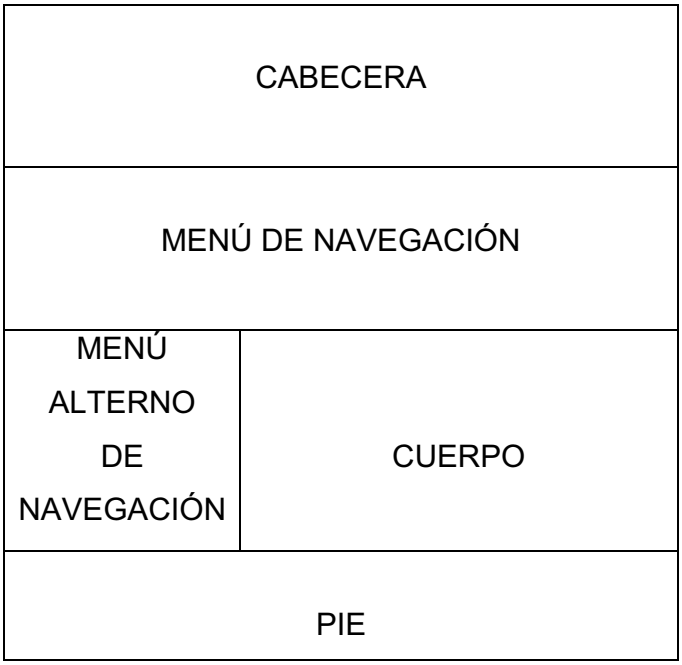


Figura 2. 63 Diseño de interfaz gráfica de usuario para las clases de interfaz restantes.

En la cabecera se ubicará el nombre de la aplicación, un logo representativo y una imagen de fondo.

En el menú de navegación se ubicarán todos los enlaces para acceder a las diferentes páginas que conforman la aplicación web.

En el cuerpo se ubicarán diferentes elementos como botones, cuadros de texto, tablas, listas desplegables, etiquetas e imágenes que variarán de acuerdo a cada página.

En el pie también se establecerán enlaces a algunas páginas y un enlace que permita subir al inicio de la página en cuestión.

En la Figura 2.61, la parte de Iniciar Sesión contendrá dos cuadros de texto para ingresar el nombre de usuario y el password de autenticación y un botón Iniciar sesión para enviar esta información a la aplicación y la valide.

En la Figura 2.62, la sección árbol mostrará: el árbol de elementos y redes en el caso de la clase IUElementos; mientras que en el caso de la clase IUGrupoMIB, mostrará el árbol de nodos del grupo MIB seleccionado.

En la Figura 2.63, la sección menú alterno de navegación contendrá enlaces a las páginas: IUConfigServidor, IUElementos y a una página de ayuda. La ubicación de este menú ofrece al usuario rapidez de acceso a tales páginas, las cuales posiblemente serán las más visitadas por el usuario.

2.2.2 MODELO DE DESPLIEGUE

El modelo de despliegue es un modelo de objetos que describe la distribución física del sistema sobre nodos de cómputo. Un nodo generalmente representa un procesador o un dispositivo de hardware similar.

Según los requerimientos del proyecto, la aplicación será albergada en dos estaciones de trabajo diferentes (una con sistema operativo Linux y otra con Windows) y serán comandadas desde una tercera estación de trabajo. Por lo tanto, la aplicación será del tipo cliente-servidor. La siguiente figura muestra el diagrama de despliegue de la aplicación.

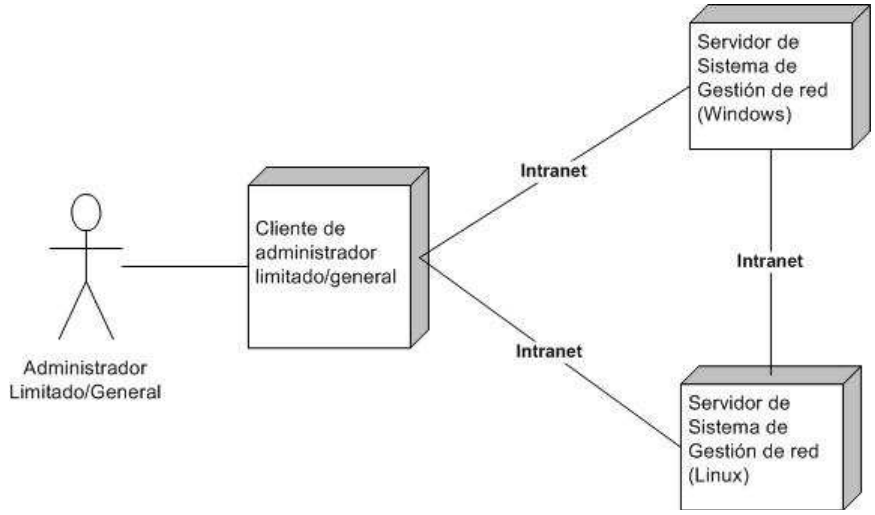


Figura 2. 64 Diagrama de distribución de la aplicación.

La figura siguiente muestra la distribución de los subsistemas de la aplicación sobre los diferentes nodos:

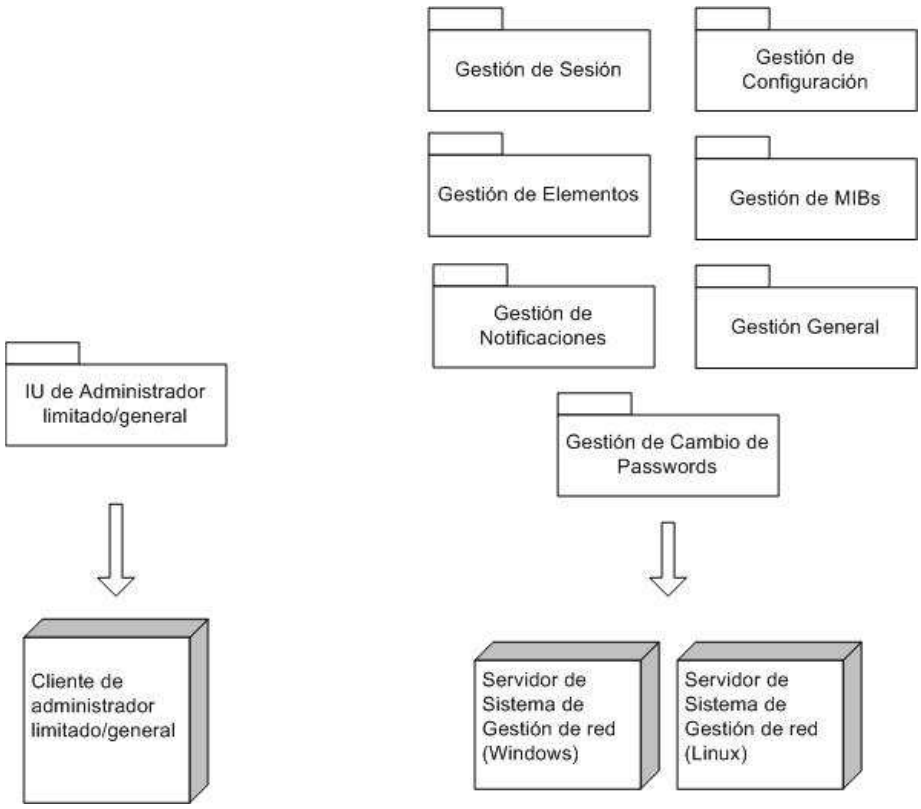


Figura 2. 65 Distribución de subsistemas en los nodos.

2.3 IMPLEMENTACIÓN DEL SISTEMA [35]

La implementación es realizar los componentes del sistema en base a los resultados obtenidos del diseño. Un componente es la representación física de los elementos de un modelo, como son las clases en el modelo de diseño. Ejemplos de componentes son: archivos de código fuente, *scripts*, etc. Previo a la implementación del sistema, se estudia el entorno de desarrollo Java[73] en el cual de desarrollará la aplicación.

2.3.1 ENTORNO DE DESARROLLO

La aplicación se realiza utilizando el entorno de desarrollo Java. Debido a que la aplicación es el tipo web, se utilizan algunas de las tecnologías que Java propone para la realización de ese tipo de aplicaciones; concretamente: *JavaServer Pages* (JSP), *Java Servlets* y *Tomcat*.[74]

Además, como herramientas de desarrollo se utilizan: *IDE Eclipse* y *Adobe Dreamweaver*. La primera herramienta se utiliza para la realización de todos los subsistemas de diseño y la segunda para el modelaje de interfaces gráficas.

En el primer capítulo se hace una breve descripción de Java y las tecnologías mencionadas.

2.3.2 MODELO DE IMPLEMENTACIÓN [35]

El modelo de implementación detalla la representación física de los elementos del modelo de diseño. El modelo de implementación está formado por: componentes y subsistemas de implementación.

[73] La justificación de la elección del entorno de desarrollo Java está basada en el requerimiento no funcional tratado en el ítem 2.1.1.3.
[74] En la sección F de ANEXOS se muestra un ejemplo básico de utilización de todos estos elementos.

2.3.2.1 Componentes

Como se dijo anteriormente, el componente es la representación física de un elemento del modelo de diseño, como por ejemplo una clase de diseño. En el marco de la aplicación, cada clase de diseño sería representada por un archivo de código fuente. Como ejemplos concretos de componentes de la aplicación se tiene:

- SwarServlet.java
- Usuario.java
- GestorRegistroDerecho.java
- ExpAnuVista.jsp

Cabe recalcar que es normal que se creen más componentes que clases de diseño. La razón de esto es optimizar los procesos y técnicas de codificación; por ejemplo, en la clase de diseño SwarServlet se pueden implementar diferentes componentes para cada de caso de uso que permitan organizar de mejor manera el procesamiento de solicitudes.

2.3.2.2 Subsistemas de implementación

Los subsistemas de implementación son la representación física de los subsistemas de diseño. Además, esta relación se extiende a los subsistemas de servicio los cuales están contenidos dentro de los subsistemas de diseño; por lo tanto, los subsistemas de servicio representan la unidad básica de organización de la aplicación.

En el marco de la aplicación, los subsistemas de servicio están representados por paquetes[75] que contienen componentes. Los paquetes identificados para la aplicación son:

[75] Un paquete o *package* es el mecanismo de utiliza Java para organizar sus archivos de código fuente.

- Host.
- Elementos.
- Usuarios.
- MIBs.
- Nodos.
- Notificaciones.
- Redes.
- Mensajes.
- Grupos.
- Vistas.
- Derechos.
- Registros de usuarios.
- Registros de vistas.
- Registros de derechos de acceso.
- Registros de cambio de passwords.
- Servlet.
- SNMP.
- KCM.
- Buscador hosts.
- ExpAnuDerecho.
- ExpAnuUsuario.
- ExpAnuVista.
- ExpPasswords.
- Tree.

2.3.3 CREACIÓN DE LA APLICACIÓN WEB

Este tema aborda información detallada acerca de la implementación física (codificación) de los componentes del modelo de implementación. Se lo ha dividido de la siguiente manera: modelo de arquitectura web, estructura de la aplicación web,

páginas web, archivos de código fuente, archivos de configuración, conectividad a datos y librerías utilizadas.

2.3.3.1 Modelo de arquitectura web

El modelo de arquitectura adoptado para la creación de la aplicación web es el MVC ó modelo de arquitectura 2; donde:

- Las vistas (páginas JSP) están representadas por todas las clases de diseño de interfaz de usuario que contiene el susbistema de diseño IU de Administrador limitado/general. Ver Figura 2.59.

- El controlador principal (servlet) está representado por la clase de diseño SwarServlet.

- La parte del modelo (java beans) está representado por las clases de diseño: Usuario, Mensaje, MIB, Elemento, Red, Servidor, NodoMIB, Notificacion, GrupoVacm, RegistroUsuario, Vista, RegistroVista, DerechoAcceso, RegistroDerecho y RegistroPasswords.

- Las clases de diseño restantes representan a controladores secundarios que trabajan para el controlador principal; éstas son: GestorMensaje, GestorMIB, GestorElemento, GestorRed, BuscadorElementos, PingUtil, SnmpEngine, ReceptorNotificiones, InfoHostTreeFactory, NavMIBTreeFactory, GestorNodoMIB, GestorNotificacion, GestorUsuario, GestorServidor, GestorGrupoVacm, GestorRegistroUsuario, ExportadorUsuario, AnuladorUsuario, GestorVista, GestorRegistroVista, ExportadorVista, AnuladorVista, GestorDerechoAcceso, GestorRegistroDerecho, ExportadorDerecho, AnuladorDerecho, GestorRegistroPasswords, ExportadorPasswords y KeyChangeManager.

La razón de la elección del modelo MVC es la similitud que existe entre este modelo y el modelo de análisis y el modelo de diseño, desarrollados anteriormente.

2.3.3.2 Estructura de la aplicación web

Basándose en las especificaciones de Java Servlet 2.5, la estructura de la aplicación web está definida de la siguiente forma:

- **swar:** Es el directorio global de toda la aplicación.

- **swar/src/:** Es el directorio para ubicar los paquetes que contendrán a los archivos de código fuente.

- **swar/WebContent/:** Es el documento root de la aplicación web. Aquí de ubican todos los recursos web de la aplicación (imágenes, archivos javascript, archivos css, etc).

- **swar/WebContent/WEB-INF/web.xml:** Es el descriptor de despliegue de la aplicación.

- **swar/WebContent/WEB-INF/classes:** Es el directorio para ubicar servlets y clases compiladas (archivos .class).

- **swar/WebContent/WEB-INF/lib/*.jar:** Es el directorio para almacenar las librerías (archivos JAR) principales que necesita la aplicación.

- **swar/WebContent/WEB-INF/jsp/:** Es el directorio para ubicar las páginas JSP. Debido a que el directorio /WEB-INF es accesible únicamente desde el servlet, esto obliga al usuario a realizar cualquier solcitud al servlet y no permite acceder a las páginas directamente. Esto ayuda a minimizar errores de procesamiento de solicitudes y a establecer mecanismos de seguridad.

- **swar/WebContext/WEB-INF/model/:** Es el directorio para ubicar los archivos de almacenamiento de datos de la aplicación.

- **swar/WebContent/META-INF/MANIFEST.MF:** Es el archivo de información meta-descriptiva de la aplicación.

2.3.3.3 Páginas web

Las clases de diseño de interfaz de usuario se representan físcamente como páginas web (archivos con extensión .jsp).

2.3.3.3.1 Elementos y controles utilizados

Las páginas JSP utilizan los elementos estándar de HTML y otros elementos especiales.

Para pasar información al servidor se utilizan los elementos *HTML Forms* (formularios). Un elemento del tipo *form* (representado por la etiqueta: <form>) puede contener varios elementos del tipo *input* (representado por la etiqueta: <input>) tales como:

- Cajas de texto, representados por el tipo *text.*
- Casillas de verificación, representados por el tipo *checkbox.*
- Botones, representados por los tipos: *radio*, *submit* y *button*.
- Listas de selección, representados por el tipo *select*.
- Cajas de ingreso de passwords, representados por el tipo *password.*
- Cargadores de archivos, representados por el tipo *file.*

Para presentar información organizada a manera de árbol jerárquico se utiliza el elemento especial *sp:JSPTree*[76] (representado por la etiqueta: <sp:JSPTree>). Este

[76] Este elemento fue desarrollado en el proyecto JSP Tree Project por KAMENSKY Vladislav, http://jsptree.sourceforge.net/

elemento trabaja en conjunto con las clases de diseño *InfoHostTreeFactory* y *NavMIBTreeFactory*. Estas clases de diseño organizan la información que reciben y el elemento *sp:JSPTree* presenta la información organizada. Este elemento se utiliza en los casos que se solicita ver: elementos y redes registrados, y nodos de un determinado grupo MIB.

Para la presentación de resultados principalmente se utilizan los elementos *HTMLTables* (tablas). Un elemento del tipo *table* (representado por la etiqueta: <table>) está conformado por elementos del tipo *row* (fila, representado por la etiqueta: <tr>). A su vez, un elemento *row* está conformado por elementos del tipo *data cell* (celda, representado por la etiqueta: <td>) en los cuales va la información en sí. Un elemento *row* también puede contener el elemento *header* (cabecera, representado por la etiqueta: <th>) en el cual generalmente van títulos de las tablas.

2.3.3.3.2 Implementación de las páginas web

Las páginas web se implementan según el diseño de las interfaces web desarrollado anteriormente (ver figuras 2.60 – 2.63). Por lo tanto, a más de las clases de interfaz identificadas previamente se agregan otras páginas web de acuerdo a la siguiente especificación:

- **Cabecera.jsp:** Esta página web contiene las secciones de cabecera y menú de navegación.

- **PiePag.jsp:** Esta página web contiene la sección pie**.**

- **Menulzq.jsp:** Esta página web contiene la sección menú alterno de navegación.

Las secciones restantes como: mensaje de bienvenida, enlace a iniciar sesión, iniciar sesión, árbol y cuerpo están contenidas en las páginas JSP restantes.

Las razones de esta división de la interfaz en otras páginas JSP son las siguientes:

- **Reutilización de código**. Casi todas las interfaces de usuario tienen en común las secciones cabecera y pie de página, por lo tanto estas secciones van representadas en páginas JSP separadas y son invocadas por el servlet o por otra página JSP.

- **Inclusión de páginas JSP**: La tecnología JSP tiene la característica de que una página JSP puede incluir en su interior otra página JSP, o de que un servlet puede juntar páginas JSP y mostrarlas al usuario. Para lograr esto hay a disposición: un elemento especial de JSP denominado *jsp:include* (representado por la etiqueta: <jsp:include>) y un método usado en el servlet denominado *include()*.

El código fuente C.1[77] muestra la codificación de la página Cabecera.jsp. Las etiquetas iniciales sirven para declarar el tipo de lenguaje a utilizar (en este caso Java), el tipo de codificación de la página web, la importación de librerías de etiquetas específicas (en este caso JSTL), el tipo de documento para efectuar la validación de la página, el título de la página web (la cadena de caractereres: ${titulo}), y las referencias a los archivos CSS y JavaScript.

JSTL (*JavaServer Pages Standar Tag Library*) es una librería de etiquetas que permiten realizar varias acciones dinámicas en páginas web, entre las cuales están: control de flujo (por ejemplo las sentencias: *if*, *for* e*ach*, etc.), manipulación de variables, manejo de documentos XML, etc. [30]

Los carateres “${”y “}” son símbolos pertenecientes al lenguaje denominado JSP-EL (*JSP Expression Language).* JSP-EL es un lenguaje de símbolos que permiten implementar lógica de negocio al igual que los *scriptlets* (código Java incrustado en

[77] Los fragmentos de códigos fuente están la sección C de ANEXOS.

páginas JSP) pero con mayor simplicidad. JSTL trabaja en conjunto con JSP-EL para acceder y manipular objetos en páginas JSP de forma sencilla y eficiente. [30]

A continuación, debajo del comentario “Sección de Cabecera” se encuentra una etiqueta identificada con el nombre *header_pan* (panel de cabecera). Dentro de esta etiqueta se encuentran más elementos , y su función es ubicar en ellos a los diferentes elementos de cabecera, los cuales básicamente son: el logo del programa y la imagen de fondo.

Luego, debajo del comentario “Sección de Navegación” se encuentra una etiqueta identificada con el nombre *header_nav_menu.* Igualmente, dentro de esta etiqueta existen más elementos diferentes, y su objetivo es formar la estructura de una barra de menú horizontal.

El código fuente C.2[78] muestra la codificación de la página PiePag.jsp. La etiqueta identificada como *footer_pan* representa al contenedor de pie de página y contiene dos elementos : *logo_nms* y *footer_nav_menu*. El primero sirve como contenedor de una imagen de logo y el segundo como contenedor de otro menú de navegación.

El código fuente C.3 muestra la codificación de la página MenuIzq.jsp. La etiqueta identificada como *left_pan* representa al contenedor de menú izquierdo y en interior contiene otro elemento identificado como *left_nav_menu*. En este elemento se ubican los enlaces de navegación de acceso rápido.

El código fuente C.4 muestra un ejemplo de inclusión de una página JSP dentro de otra página JSP. La etiqueta <jsp:include> hace posible que la página MenuIzq.jsp sea insertada dentro de la página actual. La página actual es DescubHost.jsp; ésta también contiene la sección cuerpo y está representada por el elemento

[78] Los fragmentos de códigos fuente están la sección C de ANEXOS.

identificado como *right_pan*; en este elemento se ubica el título de la página y el contenido del cuerpo en sí. Este mecanismo de inclusión de la página MenuIzq.jsp se usa en las demás páginas web.

El código fuente C.5 muestra el método del servlet SwarServlet para procesamiento de solicitudes web. Todas las solicitudes de usuario generadas desde las páginas web son atendidas por este método (El método doGet() invoca al mismo método doPost(); se lo diseñó así con el fin de centralizar el procesamiento de solicitudes). A cada solicitud se le analiza la información de la dirección o *path;* y en base a ello el servlet instancia la clase[79] adecuada. El servlet reenvía la solicitud a la clase instanciada para que se realice el procesamiento. La clase instanciada procesa la solicitud, genera las respuestas y las almacena en los JavaBeans correspondientes. La clase instanciada devuelve el nombre de la página web elegida para presentar los resultados. Finalmente, El servlet encarga de juntar y enviar las páginas web: Cabecera.jsp, PiePag.jsp, y la página JSP elegida (en este caso, DescubHost.jsp), mediante el método include().

La manera cómo se presentan los diferentes elementos dentro de una página web está definida mediante archivos CSS (Cascading Style Sheets). Las hojas de estilo en cascada contienen reglas que asignan valores a propiedades (color, fuente, tamaño de fuente, margen, posición, etc.) de los elementos pertenecientes a una página web. Una regla CSS está formada por dos partes principales: un selector y una o más declaraciones. El selector es el elemento HTML y las declaraciones son expresiones de propiedad y valor. El selector puede ser: la etiqueta del elemento, el identificador del elemento, o la clase del elemento. [43]

El código fuente C.6 muestra ejemplos de reglas CSS utilizadas en la aplicación. La primera regla contiene un selector de etiqueta. El selector de esta etiqueta es *body* y las declaraciones definen: el margen, tipo de letra, color de fondo, etc. Este tipo de

[79] Esta clase es un componente adicional a los definidos en el modelo de diseño. Esta clase solo le sirve al SwarServlet para organizar de mejor manera el procesamiento de solicitudes de usuario.

regla se caracteriza porque su aplicación se realiza en todos los elementos que tienen la misma etiqueta. La segunda regla contiene un selector de identificador. El selector es *#header_pan* y las declaraciones definen: el ancho, el alto, la posición, etc. Este tipo de regla se caracteriza porque su aplicación se realiza únicamente en un solo elemento de la página, es decir aquel referenciado por el identificador. Y la tercera regla contiene un selector de clase. El selector es *h2.title* y las declaraciones definen: tipo de letra, relleno, margen, color de fuente, color de fondo, etc. Este tipo de regla se caracteriza porque su aplicación se realiza en aquellos elementos que pertenecen a la misma clase, es decir aquellos cuyo atributo *class* (en este caso *title*) es el mismo.

2.3.3.4 Archivos de código fuente

Los archivos de código fuente son la representación física de las clases de diseño. Estos archivos están escritos en lenguaje Java. Los archivos contienen atributos y métodos que se encargan de la lógica de procesamiento de la aplicación. Las clases de control y de entidad que eran del modelo de análisis y que luego se consolidaron en el modelo de diseño son las que se representan en estos archivos. A continuación se describen los archivos de código fuente de algunas clases de diseño.

El código fuente C.7 muestra la codificación de una clase de entidad. Este archivo es la representación de la clase Red. En este archivo a más de atributos y métodos tiene una relación de herencia hacia la interfaz *Serializable*. Esto se realiza con el fin de que esta clase pueda considerarse como JavaBean y de esa manera pueda ser usado en una página JSP para generar el contenido web dinámico.

El código fuente C.8 muestra algunos métodos de una clase de control. Estos métodos pertenecen a la clase GestorRed y se encargan de manipular información de una red. Dentro del contexto de estos métodos como paso incial se realiza la obtención de datos extraídos desde un archivo XML. Los métodos que manejan archivos XML son heredados de la clase GestorXML. Los métodos de la clase

GestorXML como paso importante realizan la validación del archivo XML frente a un archivo XSD (Ver código fuente C.9). Luego de que se realiza la validación y la obtención de datos, cada método maneja la información según convenga. Por ejemplo, en el caso del método *obtenerRed()*, la información es filtrada mediante funciones XPath y se devuelve únicamente un objeto del tipo Red. En el caso del método *agregarRed()*, nueva información es agregada, para lo cual se utiliza como entrada un objeto Red, el cual luego es traducido en términos de componentes de un documento XML. En el caso del método *eliminarRed()*, para eliminar información se utiliza como entrada un objeto Red cuyos datos son usados para identificar que partes del documento XML se deben borrar. Luego de que se realiza la manipulación de información, se realiza la operación de escritura en el archivo XML si el método así lo requiere. Al ejecutar la operación de escritura se guardan definitivamente los cambios realizados. Todos los pasos de manejo de información indicados para la clase GestorRed se repiten para otras clases de control.

El código fuente C.10 muestra el método de ejecución de operaciones SNMP el cual es usado como mecanismo de recolección de información de administración. Este método pertenece a la clase de diseño Mensaje. El tipo de operación a ejecutar depende del tipo del PDU (ScopedPDU) del mensaje. Dentro de este método lo primero que se hace es crear el header y el scopedPDU. Luego se pide al SnmpEngine que envíe el mensaje creado. La respuesta recibida es procesada por la función *procesarPDUResponse()* la cual extrae los resultados y los almacena en clases del tipo VinculoVariable.

El código fuente C.11 muestra el método de envío de mensaje SNMPv3. Este método pertenece a la clase de diseño SnmpEngine. En este método existe una clase de mapeo de transporte (*transportComandGen*) y una clase de sesión SNMP (*session*[80]). La primera maneja la comunicación a nivel de capa transporte utilizando

[80] Este clase para el envío de mensajes SNMP invoca a la primitiva sendPDU() perteneciente al Dispatcher. Para mayor información visitar http://www.snmp4j.org/doc/index.html.

UDP, mientras que la segunda se encarga de enviar el mensaje formado en el método anterior y de recibir una respuesta.

2.3.3.5 Archivos de configuración

Los únicos archivos de configuración de la aplicación son *web.xml* y *MANIFEST.MF*, de los cuales sólo *web.xml* requiere de una configuración específica y a *MANIFEST.MF* se lo deja vacío.

El código fuente C.12 muestra algunas partes importantes de la configuración del archivo *web.xml*. Las primeras líneas especifican: declaración del documento XML, espacios de nombre (*namespaces*) relacionados a aplicaciones web Java, y la dirección del archivo XSD con el cual se valida este archivo de configuración. Luego, las líneas siguientes definen características de la aplicación en sí, tales como: nombre de la aplicación (etiqueta <display-name>), parámetros de configuración del servlet (etiqueta <servlet>), asignación de dirección al servlet (etiqueta <servlet-mapping>), lista de archivos de bienvenida que se presentan al iniciar la aplicación (etiqueta <welcome-file-list>) y datos de parámetros globales a ser usados en la aplicación (etiqueta <context-param>).

Otra etiqueta importante es <init-param> la cual está contenida dentro de la etiqueta <servlet>. Dentro de esta etiqueta se especifica la dirección donde están ubicados los archivos XML de almacenamiento de datos de la aplicación.

2.3.3.6 Conectividad a datos

Los datos de aplicación se guardan en archivos XML. Esto cumple con el requerimiento no funcional especificado en el ítem 2.1.1.3. Para poder extraer información de un archivo XML se requiere previamente que éste sea validado, y por lo tanto para efectuar la validación se definen los archivos XSD. Los archivos XSD básicamente permiten: definir la estructura de un archivo XML, definir tipos de datos,

y aplicar restricciones a los valores que se almacenan. La estructura de un archivo XSD está basada en los atributos de una clase de entidad. Los códigos fuente C.7 y C.13 muestran dicha similitud. El primer código fuente corresponde a la definición de la clase Red, mientras que el segundo corresponde a la definición de la estructura del archivo XML donde se va almacenar información de redes.

La siguiente figura muestra el esquema que representa la estructura del archivo XML que almacena información de red. En este esquema se identifica un elemento raíz (redes) el cual actúa como una tabla de una base de datos. El siguiente elemento (red) es un conjunto de celdas (fila) que contiene información de una red. Los elementos siguientes contenidos en el elemento red son columnas para cada fila que indican el valor que se guarda en cada celda. Además, en este diagrama se especifica una restricción (*constraint*) la cual indica que el valor *id* de una red debe único en todo el rango del elemento raíz. El código fuente C.14 muestra la implementación de este esquema en el archivo XML correspondiente.

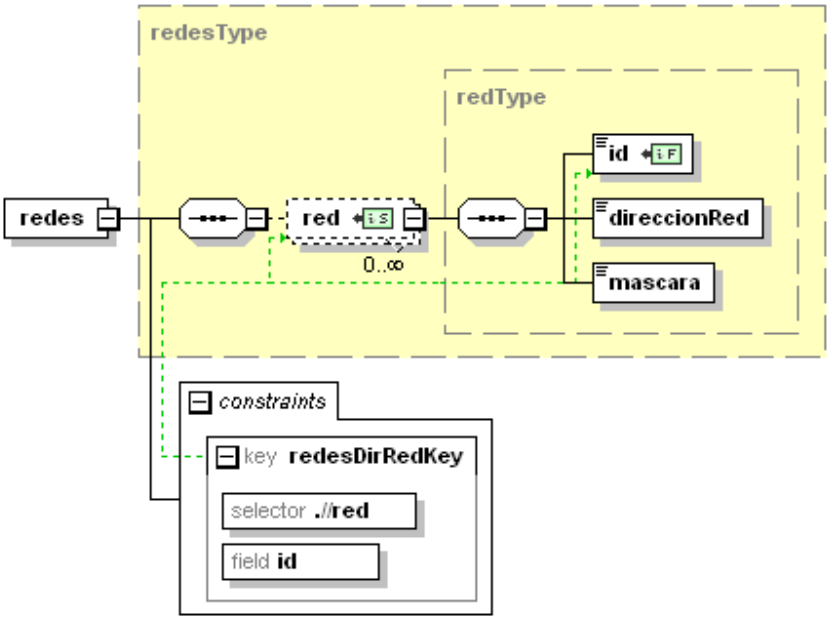


Figura 2. 66 Estructura del archivo XML para información de red.[81]

[81] Esta figura fue realizada usando la herramienta XMLSpy de Altova.

Los esquemas que definen las estructuras de los demás archivos XML están la sección B de ANEXOS.

2.3.3.7 Librerías utilizadas

Las principales librerías Java que se han utilizado para el desarrollo de la aplicación son:

- **javax.servlet:** Usado para implementación y manejo de servlets.
- **javax.xml:** Usado para operaciones de lectura, escritura y validación de archivos XML.
- **java.net:** Usado para colaborar en las operaciones de descubrimiento de elementos de red activos.
- **java.io:** Usado en operaciones de entrada y salida de datos; específicamente, aquellas operaciones que se involucran en: comunicaciones de elementos de red y archivos del sistema.
- **java.util:** Usado para dar soporte (tipo de datos especiales como arreglos) a todas las funciones de la aplicación.
- **org.apache.xpath:** Usado para operaciones de consulta dentro de documentos XML.
- **org.w3c.dom:** Usado para manejo de información de documentos XML utilizando el método DOM.
- **jpcap:** Usado para operaciones de descubrimiento de elementos de red activos.

- **org.snmp4j:** Usado para implementación de la arquitectura SNMPv3, es decir implementación del motor SNMP (SnmpEngine), de la aplicación generadora de comandos (Mensaje) y de la aplicación receptora de notificaciones (ReceptorNotificaciones).

- **org.xml.sax:** Usado para colaborar en las operaciones de validación de archivos XML.

- **net.percederberg.mibble:** Usado para operaciones de lectura y extracción de información de archivos MIB.

- **net.sf.jsptree:** Usado para la estructuración y presentación de árboles jerarquicos en páginas JSP.

- **com.oreilly.servlet.multipart:** Usado para hacer posible la subida de archivos MIB a la aplicación.

- **jstl:** Esta librería de etiquetas es usada para realizar varias acciones dinámicas en páginas JSP.

CAPÍTULO 3. PRUEBAS Y RESULTADOS OBTENIDOS

En este capítulo se tratan las pruebas de funcionamiento de la aplicación y el análisis de los resultados obtenidos. Además, previo a los dos temas mencionados se hace un estudio general sobre las características de la aplicación.

Antes de realizarse las pruebas de funcionamiento, se hace un tratado de las características de la aplicación. En este tema se trata acerca de: la definición de la aplicación, características principales, funcionalidades principales que presta, y una descripción general de su interfaz gráfica.

Luego, se hacen las pruebas de funcionamiento de la aplicación para verificar si se cumplen los requisitos funcionales y no funcionales tratados en el capítulo anterior. Las pruebas se las realiza en una red de área local bajo condiciones ideales, y la secuencia como se las van realizando está en concordancia con la secuencia de la realización de cada caso de uso; por lo tanto, se hace una prueba de funcionamiento por cada caso de uso.

La prueba que se realiza para cada caso de uso está compuesta por: una descripción detallada de las páginas web que conforman el caso de uso, una guía de cómo utilizar los controles de cada página, y una exposición de los resultados obtenidos.

Consecuentemente, se realiza el análisis de los resultados. Este en este tema se aborda aspectos como: restricciones de la aplicación cuando se la ejecuta en condiciones no ideales o con errores del usuario, mensajes que presenta la aplicación cuando se originan resultados no deseados, análisis de envío y recepción de paquetes utilizando la herramienta *Wireshark*, y finalmente un análisis general de todos los resultados obtenidos verificando si se han cumplido los requisitos del usuario y describiendo las ventajas que se tiene al utilizar esta aplicación.

3.1 CARACTERÍSTICAS DE LA APLICACIÓN

3.1.1 DEFINICIÓN

La aplicación elaborada tiene por nombre SWAR, son siglas que significan: Sistema Web de Administración de Red. SWAR es una aplicación web que permite ejercer operaciones de monitorización y control de los elementos de una red.

Para acceder a la aplicación, en un navegador se escribe a la siguiente dirección URL:

```
https://[direccionIP]/swar
```

En donde:

- *https*[82] es el protocolo seguro de transferencia de hipertexto.
- *direccionIP* es la dirección IP del equipo donde está corriendo la aplicación.
- *swar* es el documento raíz de la aplicación.

Esta forma de acceder a la aplicación utiliza al host virtual *localhost* del servidor Tomcat, el cual fue explicado en el modelo de implementación de la aplicación.

3.1.2 CARACTERÍSTICAS PRINCIPALES

- Utiliza el protocolo SNMPv3.

- Es una aplicación desarrollada en Java, por lo tanto puede ejecutarse en una variedad de sistemas operativos.

- Es portable. Los datos de la aplicación son almacenados en archivos XML.

- Es extensible. Permite subir archivos MIB estándar y propietarios.

[82] El protocolo HTTPS permite satisfacer el requimiento no funcional: **"Manejo de la aplicación a través de una interfaz web segura"** especificado en el ítem 2.1.1.3.

- Dispone de seguridad mejorada. Utiliza el subsistema de seguridad basada en usuario (USM), el protocolo HTTPS, y un control de ingreso a la aplicación.

3.1.2.1 Funcionalidades principales

- Monitorización de red.
- Control de configuración de los elementos de red.
- Detección de eventos ocurridos en la red mediante recepción de notificaciones.
- Seguridad de red, esto es: protección de la información de la administración mediante el subsistema USM y mediante el protocolo HTTPS.
- Control de ingreso a la aplicación.
- Manejo de cuentas de usuarios USM.
- Manejo de elementos del subsistema VACM. Esto comprende: Grupos VACM, Vistas MIB y Derechos de Acceso.
- Exportación y anulación de usuarios grupos VACM, vistas, y derechos de acceso hacia elementos de red.
- Manejo de passwords de usuario, tanto a nivel del servidor como a nivel de cada elemento de red.

3.1.2.2 Descripción de la interfaz gráfica

Las figuras que se muestran a continuación muestran ejemplos de las interfaces gráficas de la aplicación, en las cuales se indican las partes más representativas de la aplicación.

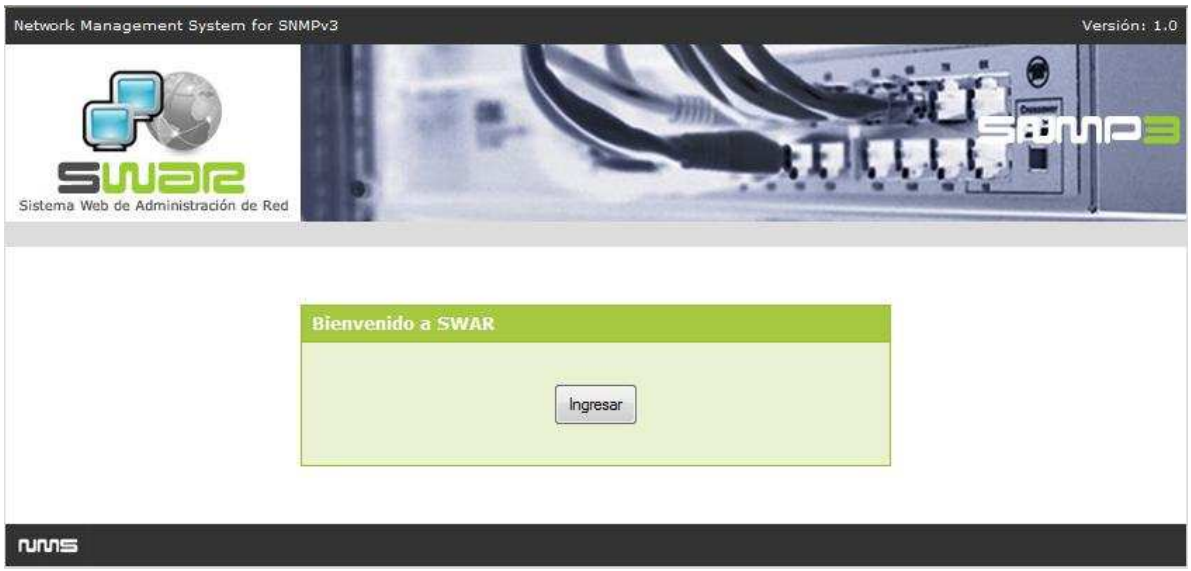


Figura 3. 1 Página de Bienvenida.

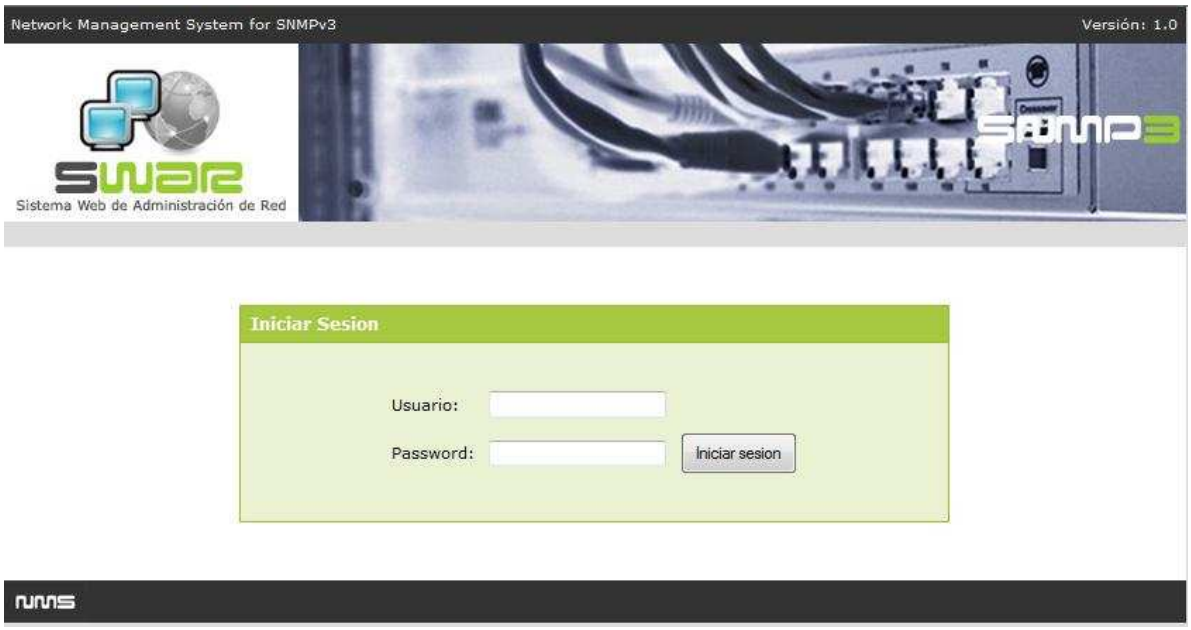


Figura 3. 2 Página de Inicio de Sesión.

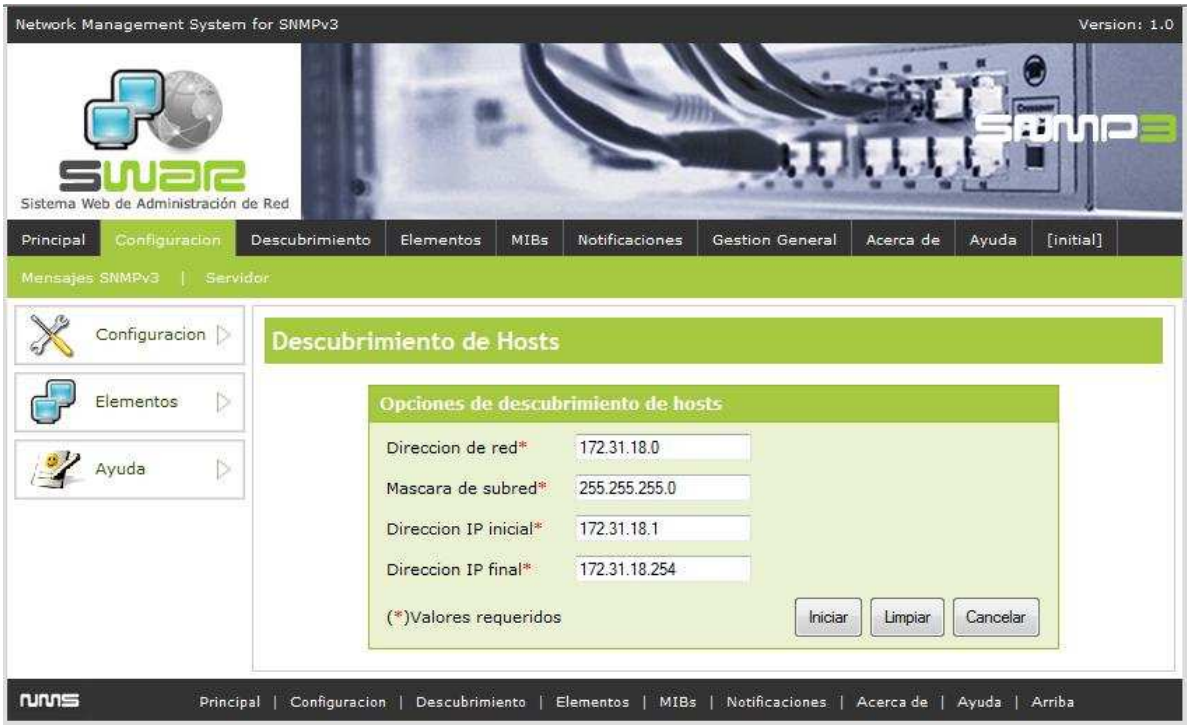


Figura 3. 3 Página de Descubrimiento de Hosts.

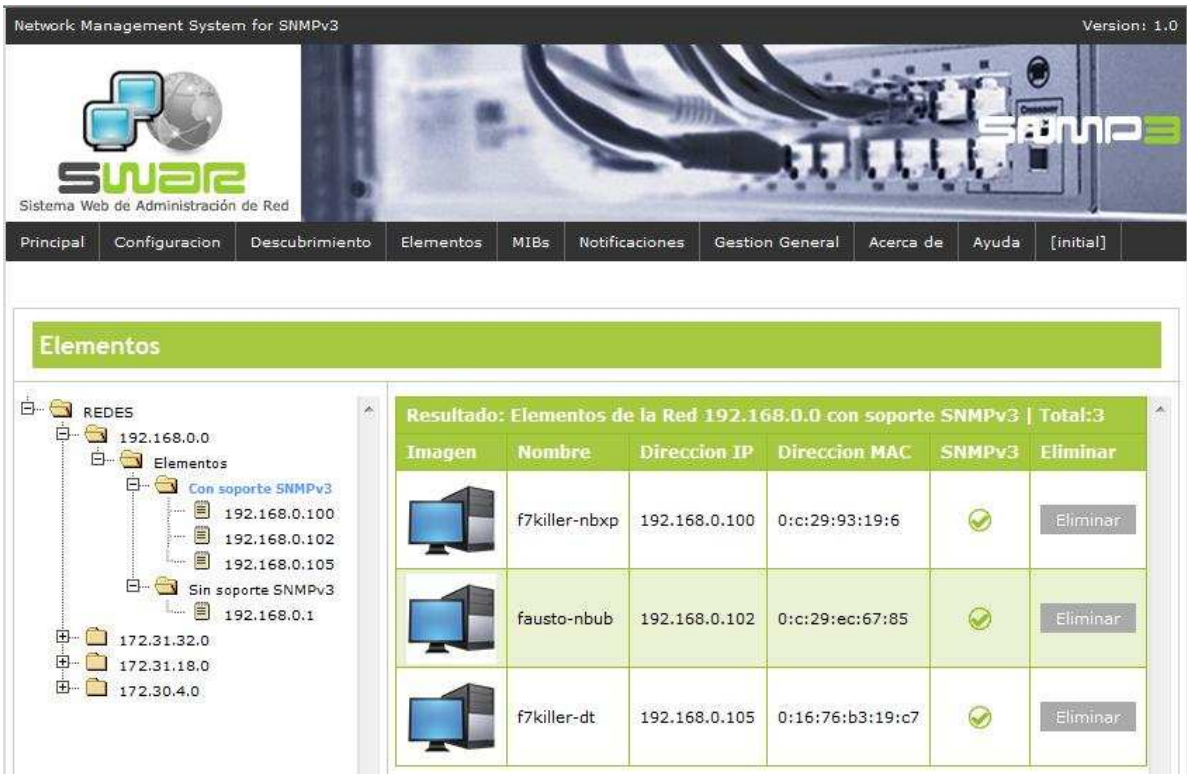


Figura 3. 4 Página de Elementos.

La figura 3.1 representa a la página de bienvenida de la aplicación. En esta página se pueden identificar la sección *Cabecera*, la sección *Mensaje de Bienvenida* y la sección *Enlace a Inicio de sesión*; éstas y todas las secciones fueron definidas en el diseño de interfaz gráfica.

La figura 3.2 representa a la página de inicio de sesión. En esta página se pueden identificar la sección *Cabecera* y la sección de *Inicio de sesión*.

La figura 3.3 representa a la página de descubrimiento de hosts. En esta página se pueden identificar la sección *Cabecera*, la sección *Menú de Navegación*, la sección *Menú Alterno de Navegación*, la sección *Cuerpo*, y la sección *Pie*. Estas secciones se repiten para el resto de páginas, en las cuales solo la sección *Cuerpo* es la que varía su contenido según la función y el contenido del resto de secciones se mantienen iguales.

La figura 3.4 representa a la página de elementos de red registrados. Esta página tiene las mismas secciones que la página anterior a excepción de la sección *Menú Alterno de Navegación*, en su lugar está la sección *Árbol*. Esta sección también se encuentra en la página de grupos MIB.

La sección *Cabecera* es la parte superior de cada página. Está formado por tres componentes: un panel negro que contiene el nombre del tipo de aplicación y la versión, un ícono representativo de la aplicación, y una imagen de fondo.

La sección principal que permite acceder a las páginas de la aplicación es *Menú de Navegación*. Este menú está compuesto por un menú primario y por submenús. El menú primario es el de color negro, mientras que los submenús son de color verde.

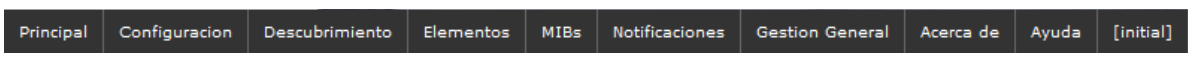


Figura 3. 5 Menú primario.

En el menú primario existen 10 enlaces a las diferentes funciones de la aplicación. La figura 3.5 muestra el menú primario.

El enlace *Principal* apunta hacia la página principal de la aplicación. Esta página contiene un menú de enlaces pero con íconos y descripciones breves de los mismos.

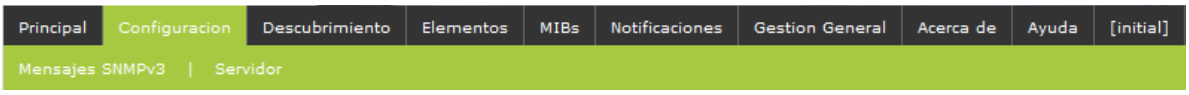


Figura 3. 6 Submenú del enlace Configuración.

El enlace *Configuración* (ver figura 3.6) contiene un submenú de dos enlaces: *Mensajes SNMPv3* y *Servidor*. El primero accede a la página de configuración de parámetros de Mensajes SNMPv3, la cual muestra los valores predeterminados y permite modificarlos. El segundo accede a la página de configuración de parámetros de Servidor, la cual también muestra los valores predeterminados y permite modificarlos.

El enlace *Descubrimiento* accede a la página de descubrimiento de elementos de red. Esta página permite realizar el descubrimiento de elementos activos a partir de una dirección de red e intervalo de direcciones IP dados. Los resultados son presentados en otra página del mismo caso de uso, en la cual se puede solicitar que se guarden o descarten los resultados obtenidos.

El enlace *Elementos* accede a la página donde se muestran los elementos de red registrados, representados en un árbol jerárquico, y organizados por red a la que pertenecen y por capacidad de gestión con el protocolo SNMPv3. También se muestra información según el nodo que se seleccione en el árbol. Además, dentro de esta página están los enlaces que invocan a la página que muestra información detallada de un determinado elemento de red.

El enlace *MIBs* accede a la página donde se muestran: la estructura en forma de árbol de un grupo MIB seleccionado, y la información del nodo MIB seleccionado. Además, en esta página se puede ejecutar los diferentes comandos SNMPv3 hacia un determinado elemento de red en función del nodo MIB seleccionado.

El enlace *Notificaciones* accede a la página donde se muestran las notificaciones que se han recibido. Además, en esta página se muestra información detalla de la notificación que se seleccione.

Figura 3. 7 Submenú del enlace Gestión General.

El enlace *Gestión General* (ver figura 3.7) contiene un submenú de cuatro enlaces: *MIBs, Usuarios, Vistas* y *Derechos de Acceso.* El enlace *MIBs* accede a la página que permite subir y eliminar archivos MIBs. El enlace *Usuarios* accede a la página que permite crear y eliminar usuarios en el servidor, y además en esta página están los enlaces que apuntan a las páginas que permiten exportar y anular cuentas de usuarios hacia elementos de red. El enlace *Vistas* accede a la página que permite crear y eliminar vistas MIB en el servidor, y además en esta página están los enlaces que apuntan a las páginas que permiten exportar y anular vistas MIB hacia elementos de red. El enlace *Derechos de Acceso* accede a la página que permite crear y eliminar derechos de acceso en el servidor, y además en esta página están los enlaces que apuntan a las páginas que permiten exportar y anular derechos de acceso hacia elementos de red. Cabe recalcar que el enlace *Gestión General* solo aparece si el usuario que ha iniciado sesión es del tipo Administrador General.

El enlace *Acerca de* accede a la página que muestra información de la aplicación, como su nombre, la versión, los créditos, el autor, etc.

En enlace *Ayuda* accede a la página que contiene información guía para manejar la aplicación.

Figura 3. 8 Submenú del enlace de usuario que ha iniciado sesión.

En enlace *[initial]* (ver figura 3.8) es dinámico ya que en él aparece el nombre del usuario que ha iniciado sesión y en este caso el nombre del usuario es *initial*. Este enlace tiene un submenú de dos enlaces: *Cambiar Passwords* y *Cerrar Sesión.* En el primer enlace accede a la página que permite realizar el cambio de passwords del usuario que ha iniciado sesión; además en esta página está el enlace que apunta a la página que permite exportar el cambio de passwords hacia los elementos de red. El segundo enlace permite cerrar la sesión del usuario.

La sección *Menú Alterno de Navegación* contiene tres enlaces: *Configuración* del servidor, *Elementos* y *Ayuda*. Este menú es móvil, se desplaza a lo largo de la página que esté ese momento. El objetivo de este menú es agilitar el acceso rápido a estas páginas que son consideradas como las más visitadas.

La sección *Pie* también tiene un menú de navegación, el cual es útil cuando la página en la que se está navegando es muy extensa.

La descripción detallada de cada página se la tratará en los ítems referidos a las pruebas de funcionamiento de la aplicación.

3.1.2.3 Datos y archivos

La aplicación contiene como recursos: archivos XML y archivos XSD. Estos archivos sirven a modo de base de datos y en ellos se almacena toda la información que maneja el sistema.

Además, la aplicación contiene: archivos de configuración *web.xml* (para la aplicación) y *server.xml* (para el servidor Tomcat), librerías necesarias para el correcto funcionamiento de la aplicación, archivos de hojas de estilo CSS, archivos de JavaScript utilizados para validar datos de entrada, páginas web JSP y otros componentes característicos una aplicación web desarrollada en Java.

3.2 PRUEBAS

Este tema describe cómo se prueban las diferentes funciones de la aplicación en un ambiente de pruebas determinado.

La secuencia como se van realizando las pruebas está en concordancia con la secuencia de la realización de cada caso de uso; por lo tanto, se hace una prueba de funcionamiento por cada caso de uso.

Las condiciones en que se realizan las pruebas son ideales. Esto significa que: el estado del entorno de la aplicación es factible para iniciar la ejecución del caso de uso, no existe errores de manejo de la aplicación por parte del actor, y no existe errores de ninguna capa (arquitectura TCP/IP) en la topología de red considerada como ambiente de pruebas.

El ambiente de pruebas corresponde a la topología de red que se muestra en la figura 3.9. El plan de direccionamiento para esta topología está detallado en el tabla 3.1.

A todos los host se les realiza una configuración inicial del protocolo SNMPv3, esto es efectuado como paso previo a la ejecución de la aplicación. Las tablas que indican detalles de la configuración SNMPv3 están en la sección D de Anexos. En síntesis, estas configuraciones lo que hacen es crear la cuenta de administrador general (denominado *initial*), crear la vista MIB (denominada *internet*) usada para

operaciones de escritura, lectura y notificación, crear el grupo VACM (denominado *initial*), crear el contexto de información de gestión, y crear el derecho de acceso que relaciona al grupo VACM y a las vistas MIB.

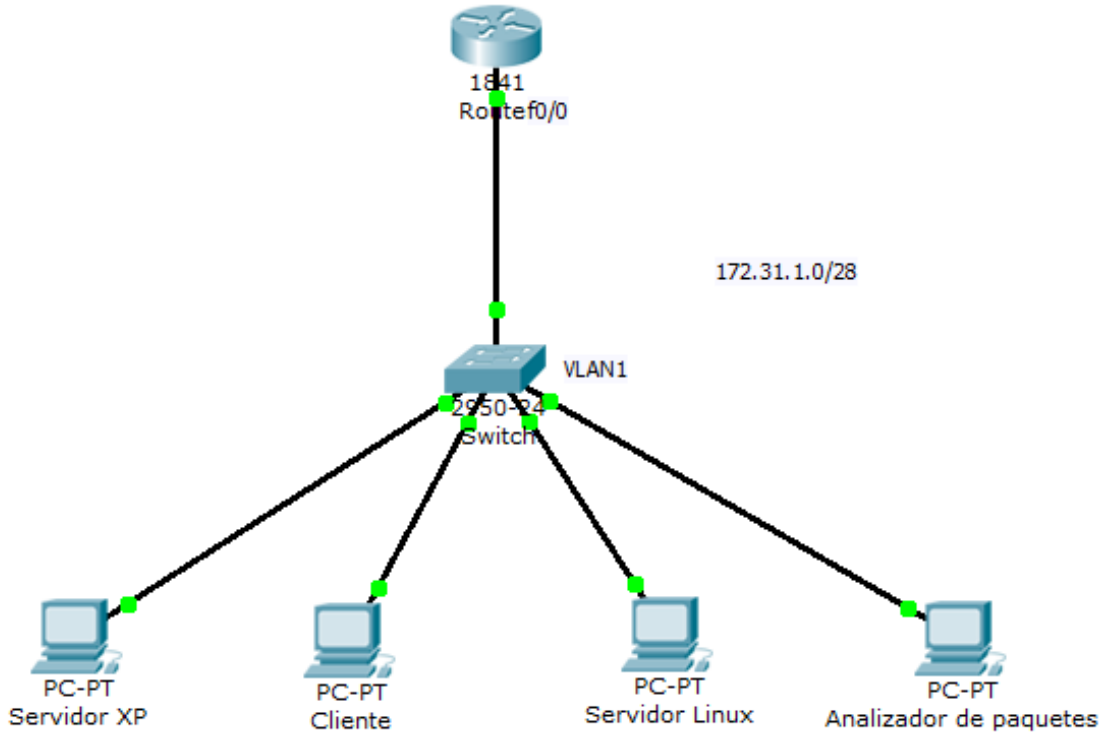


Figura 3. 9 Topología lógica de red usada como ambiente de pruebas.

Dispositivo	Nombre	Interfaz de red	Dirección IP	Máscara de subred	Gateway
Router	R1	f0/0	172.31.1.1	255.255.255.240	No aplicable
Switch	S1	VLAN1	172.31.1.2	255.255.255.240	172.31.1.1
Servidor XP	fausto-nbxp	NIC	172.31.1.3	255.255.255.240	172.31.1.1
Servidor Linux	fausto-nbub	NIC	172.31.1.4	255.255.255.240	172.31.1.1
Cliente	fausto-vaio	NIC	172.31.1.5	255.255.255.240	172.31.1.1
Analizador de paquetes	fausto-nbfd	NIC	172.31.1.6	255.255.255.240	172.31.1.1

Tabla 3. 1 Tabla de direccionamiento para topología de red.

En la topología de red se puede identificar a dos servidores: Servidor XP y Servidor Linux (con distribución Ubuntu desktop). Ambos servidores tienen instalado el servidor web Tomcat, y en cada uno de ellos está contenida una aplicación SWAR. Cada aplicación SWAR es accedida vía navegador web (Internet Explorer 8.0) desde la máquina Cliente.

El computador Cliente tiene instalado el sistema operativo Windows 7, pero no tiene instalado el agente SNMPv3; la razón de esto es porque el Cliente solo sirve para manejar ambos sistemas SWAR.

El computador analizador de paquetes sí tiene instalado y configurado el agente SNMPv3 y sirve como estación que captura paquetes SNMP, los cuales serán analizados próximamente. El analizador de paquetes tiene instalado el sistema operativo Linux de la distribución Fedora.

El router es un *Cisco* de la serie 1841, mientras que el switch es *Cisco Catalyst* de la serie 3560.

La comunicación que se produce en el sistema es en dos vías, el computador Cliente envía solicitudes de información de administración hacia cualquiera de los dos servidores utilizando HTTPS, y luego éstos reenvían dichas solicitudes a los hosts destino empleando SNMPv3. Por tanto, la comunicación en el sistema está protegida de principio a fin.

Por motivo de simplificar la explicación, se mostrarán las pruebas solamente de la aplicación SWAR ubicada en el Servidor XP. Solo en los casos de uso: *Ejecución de Operación SNMPv3* y *Visualizar Notificaciones* se tratará también a la aplicación SWAR ubicada en el *Servidor Linux*, esto se debe a que ambas aplicaciones SWAR interactúan entre sí mediante mensajes SNMPv3 (*Trap* e *Inform*).

3.2.1 INICIAR SESIÓN

Para acceder a SWAR localizado en el Servidor XP se digita en Internet Explorer de la máquina Cliente la dirección URL que indica la figura siguiente:

Figura 3. 10 URL de ingreso a la aplicación SWAR del Servidor XP desde la máquina Cliente.

Luego, el navegador muestra un mensaje de error de certificado. Esto se debe a que se configuró el servidor Tomcat con un certificado auto firmado para levantar el protocolo HTTPS. Utilizar un certificado auto firmado es suficiente para la ejecución de pruebas; sin embargo, ya en un ambiente de producción no debería utilizar este tipo de certificado sino uno firmado por una autoridad de certificados reconocida. Por lo tanto, para seguir en el proceso de acceso a la aplicación se escoge la opción no recomendada *Continuar a este sitio web.*

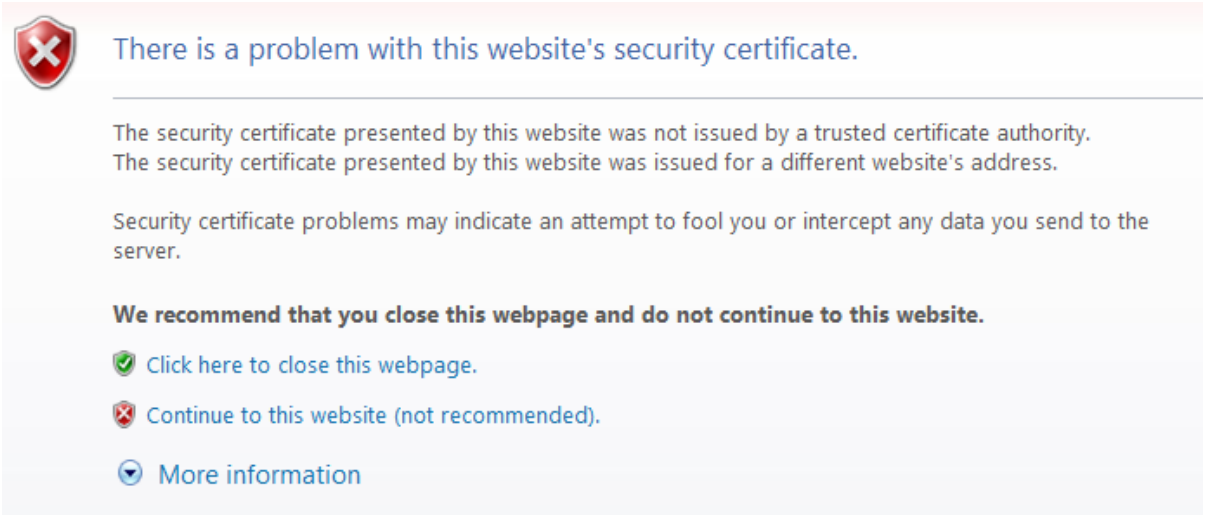


Figura 3. 11 Mensaje de error de certificado.

Posteriormente, se muestra la página de bienvenida a la aplicación, la cual contiene el enlace de ingreso. Ver figura 3.12.

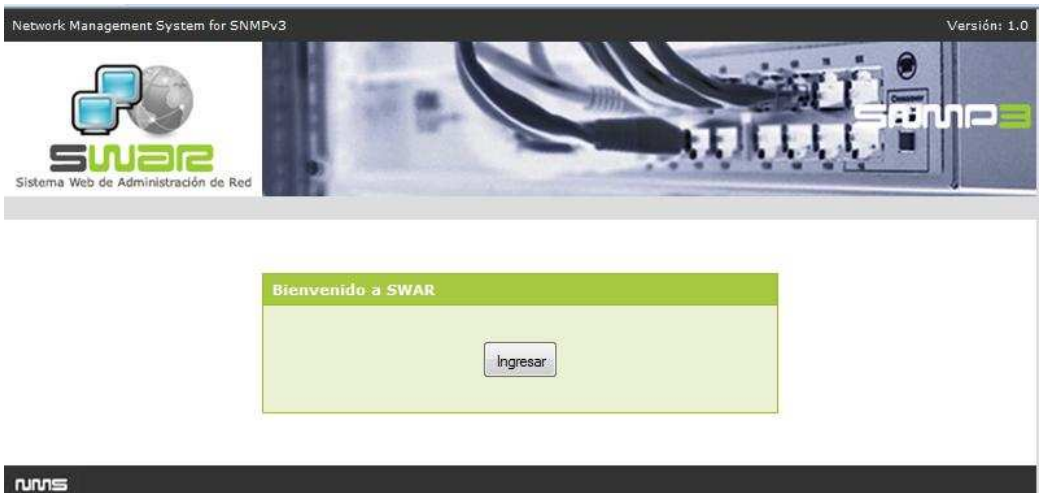


Figura 3. 12 Página de bienvenida de la aplicación.

La aplicación muestra la página de inicio de sesión. El usuario ingresa su nombre (en este caso *initial*[83]) y password de autenticación de manera correcta, entonces se presenta la página principal de la aplicación. La página principal contiene enlaces de ingreso a las demás funciones de la aplicación. Ver figuras 3.13 y 3.14.

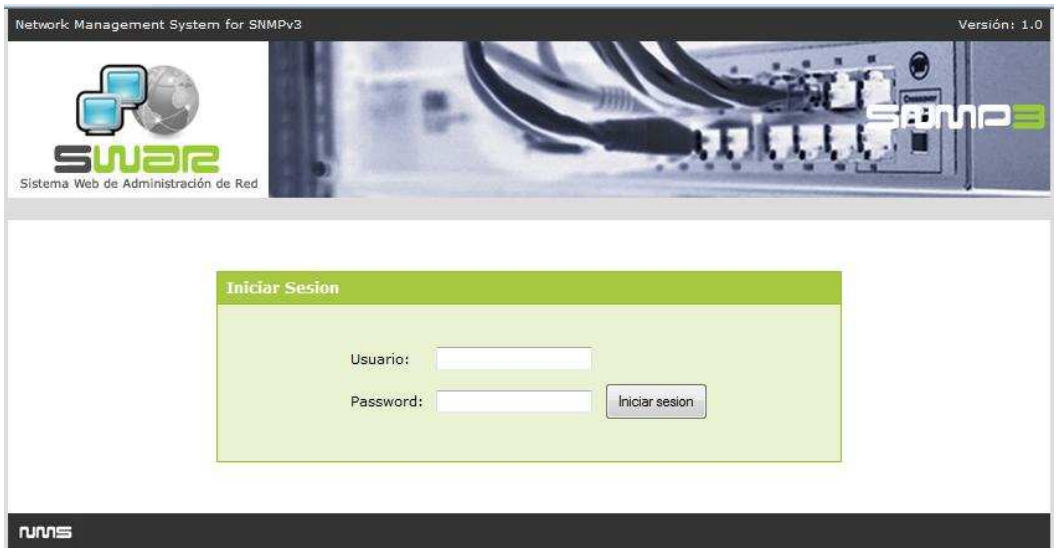


Figura 3. 13 Página de Inicio de Sesión.

[83] *initial* es el nombre de usuario del tipo Administrador General, el cual fue preconfigurado manualmente en cada agente de cada elemento de red.

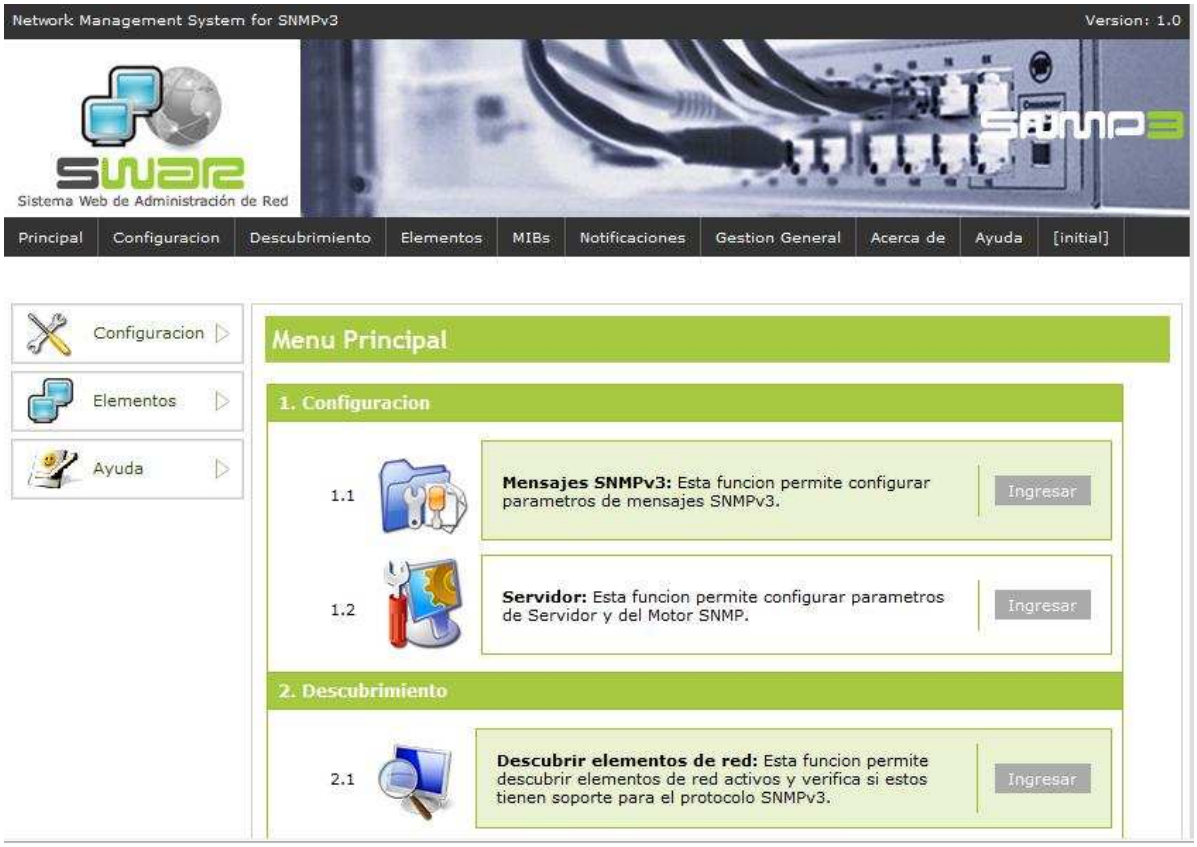


Figura 3. 14 Página Principal de la aplicación.

3.2.2 CONFIGURAR PARÁMETROS

Este caso de uso está divido en dos partes: Configuración de parámetros de Mensajes SNMPv3 y Configuración de parámetros de Servidor. En ambos casos existen valores por defecto preestablecidos, y por tanto, cuando el usuario ingresa a la página de configuración ya sea de mensajes o de servidor, la aplicación muestra dichos valores.

La siguiente figura muestra el cuerpo de la página configuración de parámetros de mensajes SNMPv3. Para acceder a esta página se lo hace desde el enlace: *Configuración>Mensajes SNMPv3* (ver figura 3.6).

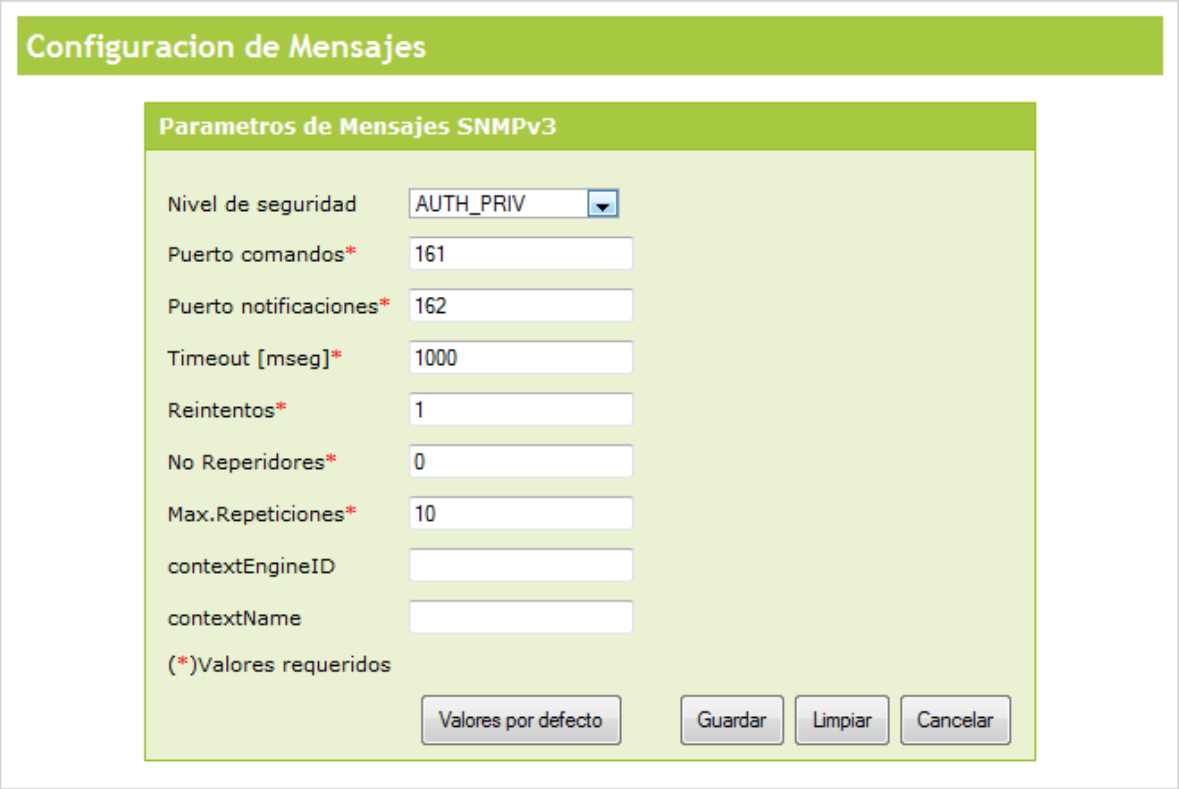


Figura 3. 15 Cuerpo de la página configuración de parámetros de mensajes SNMPv3.

En el formulario *Parámetros de Mensajes SNMPv3* se tienen los siguientes campos:

- **Nivel de seguridad:** Define el nivel de seguridad para el envío de mensajes SNMP. Tiene tres valores posibles: NOAUTH_NOPRIV (sin autenticación ni privacidad), AUTH_NOPRIV (con autenticación pero sin privacidad) y AUTH_PRIV (con autenticación y privacidad). El valor por defecto depende del usuario que haya iniciado sesión. Por ejemplo, si el usuario solo tiene password de autenticación y no de privacidad, entonces el nivel de seguridad por defecto será establecido a AUTH_NOPRIV. Sin embargo, para el ambiente de pruebas este parámetro es establecido al valor AUTH_NOPRIV; la razón de esto es mantener compatibilidad con los equipos de conectividad ya que éstos solo soportan autenticación.

- **Puerto de comandos:** Define el puerto destino al cual se enviarán los comandos de solicitud. El valor por defecto es 161.

- **Puerto de notificaciones:** Define el puerto destino al cual se enviarán los comandos de notificación. El valor por defecto es 162.

- **Timeout:** Define el intervalo de tiempo de espera en microsegundos. Si se manda un mensaje SNMP y no se recibe respuesta dentro de aquel intervalo de tiempo, entonces se considera como perdido al mensaje y se realiza uno o más reintentos de transmisión. El valor por defecto es 1000 milisegundos (1 segundo).

- **Reintentos:** Define el número de retransmisiones de un mensaje SNMP luego de que haya vencido el tiempo de espera (*timeout*). El valor por defecto es 1.

- **No Repetidores y Máximo de Repeticiones:** Definen el número de objetos escalares y el número de objetos tabla, respectivamente. Estos parámetros son usados cuando se envía un comando *GetBulkRequest*. Los valores por defecto son 0 y 10, respectivamente.

- **ContextEngineID:** Define el identificador del motor autorizado de la entidad agente a la cual va dirigido el mensaje de solicitud. El valor por defecto es una cadena vacía. Esto se debe a que la aplicación realiza automáticamente la consulta del valor de este parámetro. Para mayor información acerca de este identificador revisar el ítem 1.4.2.2 *Terminología SNMPv3.*

- **ContextName:** Define el nombre del contexto SNMP al cual se desea acceder mediante el mensaje. El valor por defecto es una cadena vacía.

Una vez que se hayan modificado los valores de los parámetros se pulsa el botón *Guardar* para almacenarlos definitivamente.

En el caso de que se hayan cometido errores en el ingreso de datos y no se haya pulsado el botón *Guardar,* entonces se puede pulsar el botón *Cancelar* para deshacer los cambios y volver a mostrar los valores anteriormente almacenados. Si se desea borrar completamente las cajas de texto se pulsa el botón *Limpiar.*

Si se desea volver a los valores por defecto se pulsa el botón *Valores por defecto* y luego se pulsa el botón *Guardar* para hacer que los cambios tengan efecto.

Los valores de parámetros de mensajes SNMPv3 son usados en los siguientes casos de uso: descubrir elementos de red, visualizar información de elemento de red, ejecutar operación SNMPv3, gestionar usuarios, gestionar vistas, gestionar derechos de acceso y cambiar passwords.

En el segundo caso de uso, configurar parámetros de servidor, se tiene el cuerpo de página tal como muestra la siguiente figura. Estos parámetros están relacionados a características del equipo donde reside la aplicación y a características del Motor SNMP (*SnmpEngine*). En el formulario *Parámetros de Servidor* se tiene los siguientes campos:

- **Nombre:** Muestra el nombre textual del equipo que hospeda la aplicación. Este campo no se puede editar.

- **Dirección IP:** Define la interfaz de red (identificada por la dirección IP) con la cual se desea trabajar en el envío de mensajes. El valor por defecto es la dirección IP de la interfaz de red activa.

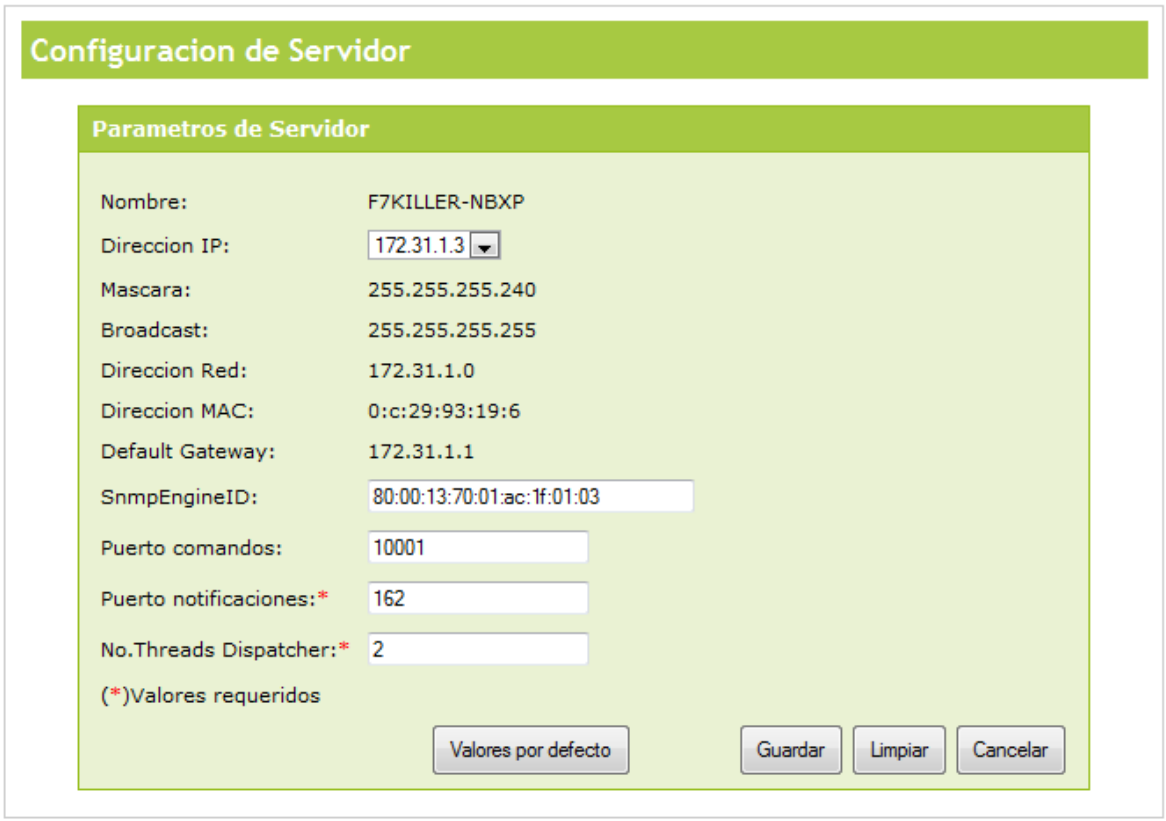


Figura 3. 16 Cuerpo de la página configuración de parámetros de servidor.

- **Máscara, Broadcast, Dirección de Red, Dirección MAC y Default Gateway:** Estos campos están relacionados al de *Dirección IP*. Estos no son editables y se establecen automáticamente después de pulsar el botón *Guardar*.

- **SnmpEngineID:** Define el identificador del Motor SNMP de la aplicación. El valor por defecto es establecido de acuerdo a la dirección IP elegida y se actualiza automáticamente luego de pulsar el botón *Guardar*. Para mayor información acerca de este identificador revisar el ítem 1.4.2.4 *Convenciones Textuales para Arquitectura SNMP.*

- **Puerto de comandos:** Define el puerto origen desde el cual se enviarán los comandos de solicitud. El valor por defecto es 10001, es un valor de puerto no bien conocido.

- **Puerto de notificaciones:** Define el puerto de escucha para la recepción de notificaciones. El valor por defecto es 162.

- **No. Threads Dispatcher:** Define el número de hilos de procesamiento para el Dispatcher. Esto permite que el motor SNMP pueda recibir, procesar y enviar mensajes SNMP simultáneamente con un mejor rendimiento. El valor por defecto es 2.

Los botones: *Guardar, Cancelar, Limpiar,* y *Valores por defecto* tienen funcionalidades semejantes al del formulario anterior.

Para acceder a la página de configuración de servidor se lo puede hacer desde el enlace: *Configuración>Servidor* (ver figura 3.6).

3.2.3 DESCUBRIR ELEMENTOS DE RED

Luego de que se hayan configurado los parámetros de la aplicación se procede al descubrimiento de elementos de red. Este caso de uso está conformado por dos páginas: la primera donde se ingresa la red objetivo y el rango de direcciones IP, y la segunda donde muestran los resultados obtenidos. Para acceder a la primera página se lo puede hacer desde el enlace *Descubrimiento* (ver figura 3.5). La segunda página es mostrada automáticamente por la aplicación una vez de que el proceso de descubrimiento haya finalizado. La siguiente figura muestra el cuerpo de la primera página previamente indicada.

Figura 3. 17 Cuerpo de la página de opciones de descubrimiento de hosts.

En el formulario *Opciones de descubrimiento de hosts* se tiene los siguientes campos: dirección de red, máscara de subred, dirección IP inicial y dirección IP final. Por defecto todos los campos son establecidos según la dirección IP configurada en el caso de uso anterior.

En el caso de la figura anterior la dirección de red, la máscara de subred y la dirección IP inicial fueron establecidas por defecto. Lo único que se modificó fue la dirección IP final de 172.31.1.14[84] a 172.31.1.6, esto se realizó con el fin de reducir el rango de direcciones IP y simplificar el proceso de descubrimiento de hosts. Si los datos establecidos por defecto no son de interés, se puede cambiarlos pulsando el botón *Limpiar* y después se ingresan manualmente los nuevos valores.

Luego, para comenzar el proceso de descubrimiento se hace clic en el botón Iniciar. La aplicación muestra temporalmente una página que contiene un mensaje de espera; esta página permanece hasta que el proceso de descubrimiento haya finalizado.

[84] Este valor representa a la última dirección IP útil de la red 172.31.1.0/28

Figura 3. 18 Página de espera.

Es importante indicar que la aplicación utiliza dos tipos de mensajes para efectuar el proceso de descubrimiento de hosts: ICMP ó ARP, y SNMP.

Los mensajes ICMP y ARP son usados para identificar que elementos de red se encuentran activos. Los mensajes ARP son usados si la red destino es la red local, mientras que los mensajes ICMP (del tipo *echo-request*) son usados si la red destino no es la red local. En el caso del ambiente de pruebas los mensajes ARP son los elegidos para este proceso.

Los mensajes SNMP son usados para determinar si los elementos activos tienen activado el protocolo SNMPv3, para determinar el tipo de elemento (router, switch, computador, etc.), y para determinar si estos elementos son capaces de ser gestionados por la aplicación.

Red destino	Mascara	Rango de direcciones	
172.31.1.0	255.255.255.240	172.31.1.1	172.31.1.6

Resultados de hosts activos

No.	Imagen	Tipo	Nombre	Direccion IP	Direccion MAC	SNMPv3
1		router	R1	172.31.1.1	0:1c:58:1a:41:1e	✓
2		switch	S1	172.31.1.2	0:64:40:2c:8f:40	✓
3		computer	f7killer-nbxp	172.31.1.3	0:c:29:93:19:6	✓
4		computer	fausto-nbub	172.31.1.4	0:c:29:ec:67:85	✓
5		unknown	desconocido	172.31.1.5	0:1d:72:45:fe:5e	✗
6		computer	fausto-nbfd	172.31.1.6	0:c:29:4f:68:73	✓

Reemplazar si host ya existe ☐ Guardar Descartar

Figura 3. 19 Cuerpo de la página de resultados de hosts activos.

La figura anterior muestra el cuerpo de la página de resultados de elementos de red activos; en él se encuentran dos tablas: la primera que indica los valores que se ingresaron en la página de opciones de descubrimiento de host y la segunda que indica los resultados en sí. La segunda tabla está conformada por varias columnas de la cuales la columna *SNMPv3* indica si el elemento soporta dicho protocolo y es capaz de ser gestionado por la aplicación. La columna *No.* sirve para enumerar las filas. Al final de la segunda tabla también existen tres controles. El botón *Guardar* sirve para almacenar definitivamente los resultados obtenidos en el archivo XML destinado para ello. El botón *Descartar* sirve para borrar los resultados obtenidos. El otro control es una caja de verificación, la cual sirve para indicar a la aplicación que al dar la orden de guardar los resultados, actualice la información de los elementos que

constan en la tabla y que también ya existen en el archivo XML; este control por defecto está desactivado.

3.2.4 VISUALIZAR INFORMACIÓN DE ELEMENTO DE RED

Luego de que se haya realizado el descubrimiento de hosts activos se procede a visualizar información de dichos hosts. Este caso de uso está conformado por dos páginas: la primera que muestra el listado de hosts a manera de árbol jerárquico y clasificados por dirección de red y por capacidad de soporte del protocolo SNMPv3; y la segunda página que presenta información de administración del elemento de red seleccionado desde la primera página.

La siguiente figura muestra parte de la página de listado de hosts. Para acceder a esta página se lo puede hacer desde el enlace *Elementos* (ver figura 3.5).

Figura 3. 20 Parte de la página de elementos de red activos.

En la figura anterior se puede apreciar dos secciones: la sección izquierda que indica el listado de hosts en forma de árbol, y la sección derecha que indica una tabla de información acerca del nodo seleccionado desde el árbol. En el caso de la figura, el

nodo seleccionado fue *Con soporte SNMPv3* (está resaltado con color celeste). Al ser seleccionado este nodo se hizo una petición a la aplicación que devuelva el listado de aquellos elementos de red que soportan el protocolo SNMPv3 y que pertenecen a la red 172.31.1.0/28; esta información se presenta en la tabla ubicada en la sección derecha. Esta tabla está conformada por un título y seis columnas. El título muestra el tema de la consulta (en este caso *Elementos de la Red 172.31.1.0 con soporte SNMPv3*) y muestra el número de filas devueltas. Las seis columnas corresponden a los siguientes campos: imagen representativa del elemento, nombre, dirección IP, dirección MAC, indicador de soporte SNMPv3 y un enlace para eliminar al elemento de la lista.

La siguiente figura muestra otro ejemplo de consulta, pero en este caso el nodo seleccionado fue la red 172.31.1.0. Los resultados de la consulta muestran información tal como: identificador de la red, dirección IP, máscara y un enlace para eliminar la red y todos los elementos que pertenecen a ella.

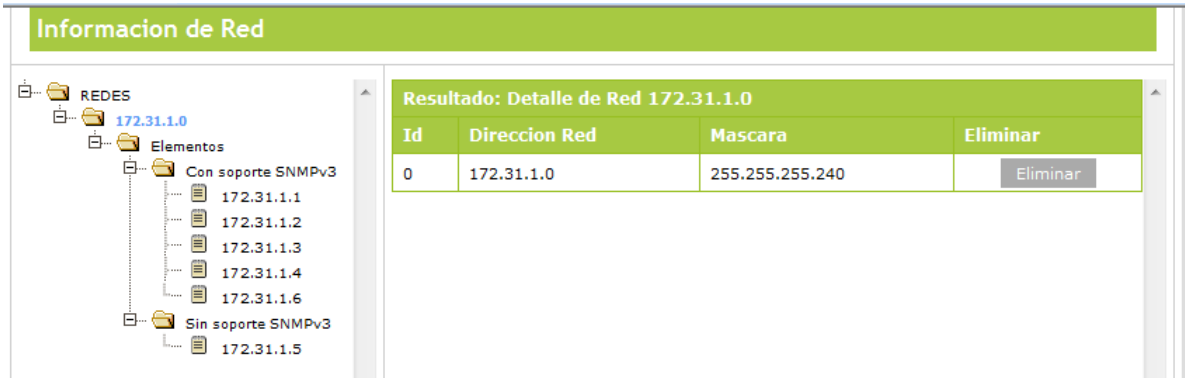


Figura 3. 21 Ejemplo de consulta de información de red.

La siguiente figura muestra la información básica de un elemento de red. Para acceder a esta página se lo puede hacer desde el campo dirección IP ubicado en la sección derecha de la página de listado de hosts o aquel que está ubicado en la tabla de resultados de consultas de la sección izquierda (ver figura 3.20). En este caso la dirección IP del elemento seleccionado es 172.31.1.6 (el computador analizador de

paquetes). La información básica comprende: nombre del dispositivo, dirección IP, dirección MAC, tipo de dispositivo y una imagen representativa del elemento.

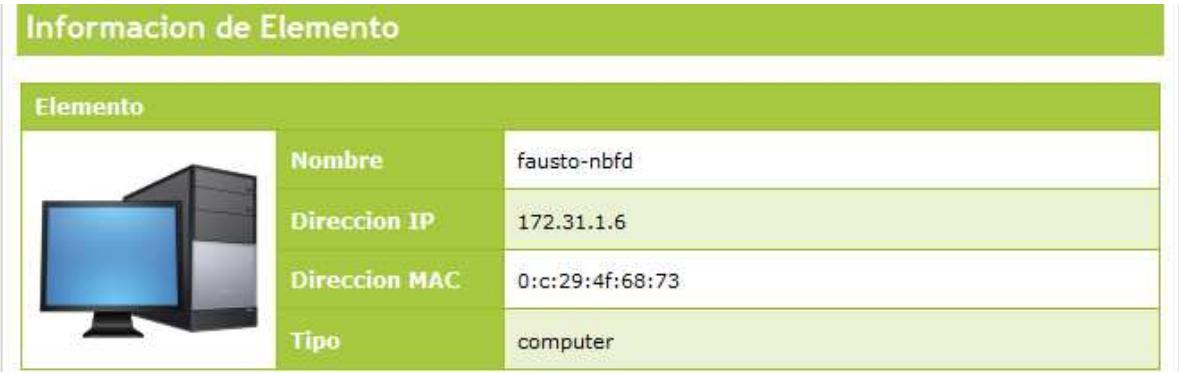


Figura 3. 22 Ejemplo de información básica de un elemento de red.

Para obtener información adicional de un elemento de red se dispone de un formulario que permite consultar según el tema. Este formulario está ubicado a continuación de la tabla de información básica de un elemento de red. Los temas de consulta son los siguientes:

- **Información de Sistema:** La información de sistema comprende datos generales del elemento de red; éstos son: descripción del elemento, ubicación física, contacto, tiempo transcurrido desde el encendido del elemento (*Uptime*), número de usuarios creados que pueden iniciar sesión en el elemento, número máximo de procesos que el elemento puede soportar, y cantidad de memoria RAM (en KBytes) que tiene el elemento.

- **Información de Dispositivos:** Esta información trata acerca de los dispositivos hardware instalados en el elemento de red. Los datos para cada dispositivo son: índice (sirve como identificador del dispositivo), descripción, número de errores ocurridos, y estado actual.

- **Información de Interfaces:** Esta información trata acerca de las tarjetas de red instaladas en el elemento. Los datos para cada tarjeta de red son: índice (sirve como identificador de la tarjeta), tipo, descripción, velocidad (bps), dirección MAC, y estado actual.

- **Información de Almacenamiento:** Esta información trata acerca de los dispositivos de almacenamiento (por ejemplo discos duros, memoria RAM, etc.) instalados en el elemento de red. Los datos para cada dispositivo de almacenamiento son: índice, tipo, descripción, tamaño total (KB), y tamaño usado (KB) y tamaño usado (en porcentaje).

- **Información de Software:** Esta información trata acerca del software instalado en el elemento de red. Los datos para cada software son: índice, nombre, fecha de instalación y tipo.

- **Información de Procesos:** Esta información se trata de los procesos que están corriendo en el elemento de red. Los datos para cada proceso son: índice, nombre, tipo, estado actual, tiempo de procesamiento CPU (centi-segundos) y cantidad de memoria RAM (KBytes) que consume.

Las siguientes figuras muestran ejemplos de los tipos de información que se pueden consultar para un elemento de red. En este caso el elemento consultado es el computador analizador de paquetes que tiene la dirección IP: 172.31.1.6.

Figura 3. 23 Consulta de información de sistema del elemento 172.31.1.6.

Figura 3. 24 Consulta de información de dispositivos del elemento 172.31.1.6.

Elemento

Nombre	fausto-nbfd
Direccion IP	172.31.1.6
Direccion MAC	0:c:29:b1:c0:27
Tipo	computer

Informacion adicional: Interfaces Consultar

Informacion de Interfaces | Total:2

Indice	Tipo	Descripcion	Velocidad (bps)	Direccion MAC	Estado
2	Ethernet CSMA/CD	eth0	10000000	00:0c:29:b1:c0:27	Up
1	Software Loopback	lo	10000000		Up

Figura 3. 25 Consulta de información de interfaces del elemento 172.31.1.6.

En la figura 3.24 se puede apreciar que para algunos dispositivos existen valores para desconocidos, esto se debe a que los objetos MIBs que proporcionan dicha información no tienen establecido ningún valor.

Elemento

Nombre	fausto-nbfd
Direccion IP	172.31.1.6
Direccion MAC	0:c:29:b1:c0:27
Tipo	computer

Informacion adicional: Almacenamiento Consultar

Informacion de Almacenamiento | Total:9

Indice	Tipo	Descripcion	Tamano(KB)	Usado(KB)	Usado(%)
3	Memoria Virtual	Virtual memory	2074576.0	400980.0	19%
1	Memoria RAM	Physical memory	1026004.0	400980.0	39%
10	Memoria Virtual	Swap space	1048572.0	0.0	0%
7	Otro	Cached memory	214428.0	214428.0	100%
6	Otro	Memory buffers	1026004.0	23412.0	2%
32	Disco Duro no removible	/boot	495844.0	47776.0	9%
31	Disco Duro no removible	/	8773016.0	5226516.0	59%
33	Disco Duro no removible	/sys/fs/fuse/connections	0.0	0.0	0%
34	Disco Duro no removible	/home/F7killer/.gvfs	0.0	0.0	0%

Figura 3. 26 Consulta de información de almacenamiento del elemento 172.31.1.6.

Elemento

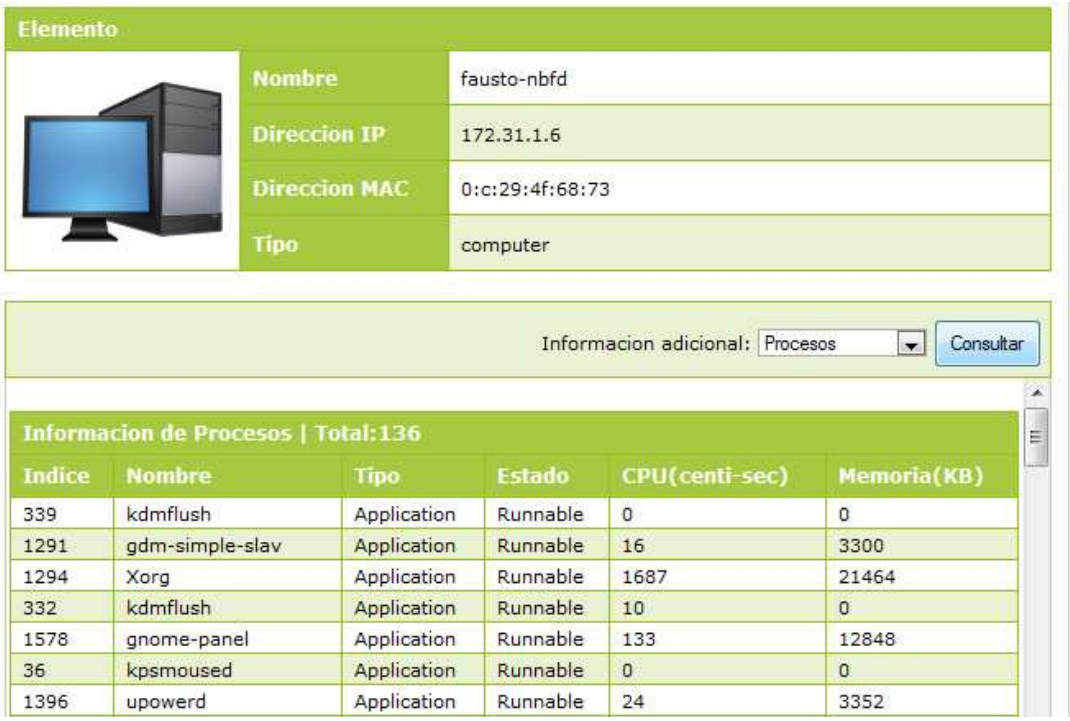
Nombre	fausto-nbfd
Direccion IP	172.31.1.6
Direccion MAC	0:c:29:b1:c0:27
Tipo	computer

Informacion adicional: Software Consultar

Informacion de Software Instalado | Total:1091

Indice	Nombre	Fecha instalacion	Tipo
338	cryptopp-5.6.1-1.fc14	2010-10-22, 13:17:3	Operating System
339	libICE-1.0.6-2.fc13	2010-10-22, 13:16:10	Operating System
941	xorg-x11-drv-cirrus-1.3.2-3.fc14	2010-10-22, 13:21:49	Application
942	groff-1.20.1-2.fc14	2010-10-22, 13:17:43	Application
940	libconfig-1.4.5-1.fc14	2010-10-22, 13:16:33	Operating System

Figura 3. 27 Consulta de información de software del elemento 172.31.1.6.



Elemento

Nombre	fausto-nbfd
Direccion IP	172.31.1.6
Direccion MAC	0:c:29:4f:68:73
Tipo	computer

Informacion adicional: Procesos Consultar

Informacion de Procesos | Total:136

Indice	Nombre	Tipo	Estado	CPU(centi-sec)	Memoria(KB)
339	kdmflush	Application	Runnable	0	0
1291	gdm-simple-slav	Application	Runnable	16	3300
1294	Xorg	Application	Runnable	1687	21464
332	kdmflush	Application	Runnable	10	0
1578	gnome-panel	Application	Runnable	133	12848
36	kpsmoused	Application	Runnable	0	0
1396	upowerd	Application	Runnable	24	3352

Figura 3. 28 Consulta de información de procesos del elemento 172.31.1.6.

3.2.5 NAVEGAR EN GRUPO MIB Y EJECUTAR OPERACIÓN SNMPv3

Ambos casos de uso están incluidos en una sola página la cual puede ser accedida desde el enlace *MIBs* (ver figura 3.5). Esta página tiene por título: *Grupo MIB*. La siguiente figura muestra el cuerpo de ésta página.

El cuerpo de la página *Grupo MIB* está divida en cinco secciones que se describen a continuación:

1) **Sección de árbol MIB:** Esta sección muestra la estructura en forma de árbol del grupo MIB seleccionado desde la sección dos. En este caso el grupo MIB representado tiene por nombre *SNMPv2-MIB.*

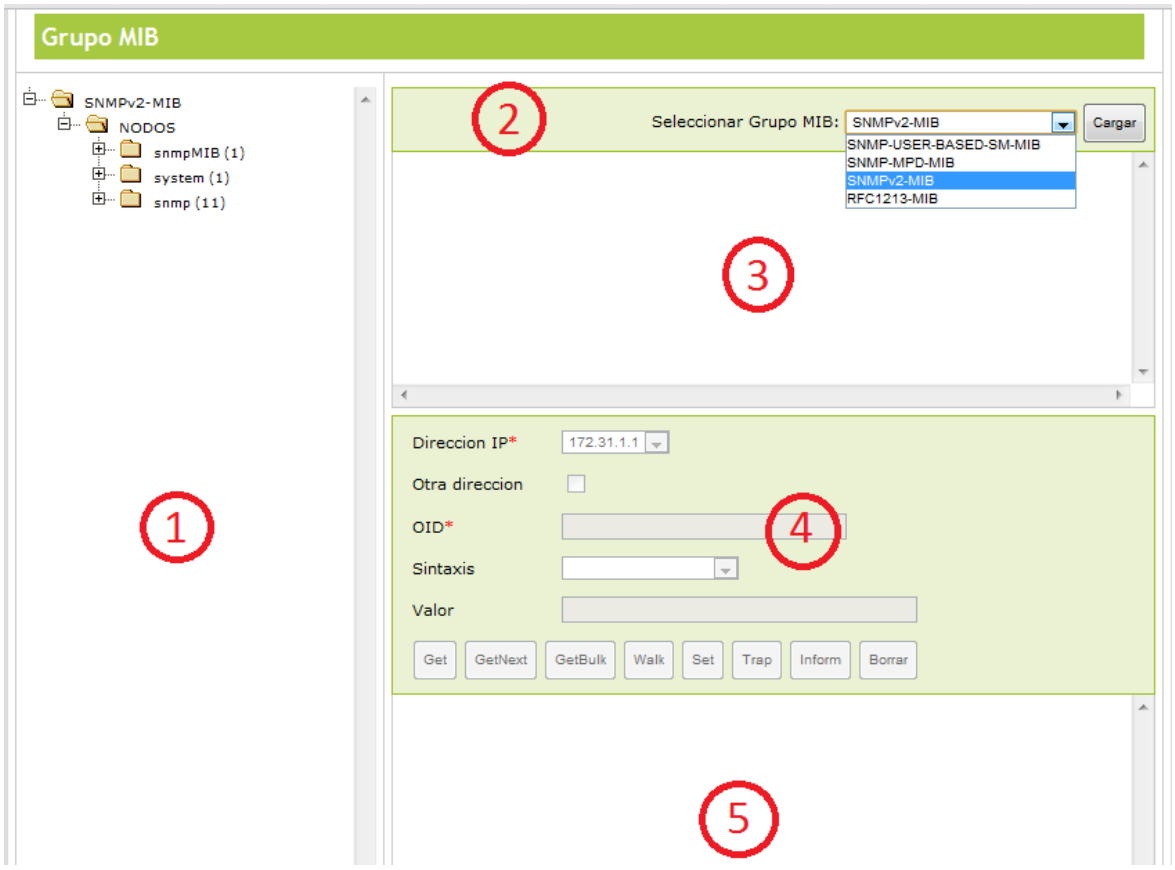


Figura 3. 29 Cuerpo de la página de navegación en un grupo MIB y ejecución de operación SNMPv3.

2) **Sección de selección de grupo MIB:** Esta sección está conformada por un formulario que permite seleccionar un grupo MIB desde la lista desplegable de grupos MIB disponibles. Una vez seleccionado el grupo MIB se pulsa el botón *Cargar* para cargar el grupo MIB en la aplicación y éste sea mostrado en la sección de árbol MIB. El manejo de la lista de grupos MIBs disponibles se la maneja desde la página de Gestión de MIBs la cual será tratada más adelante.

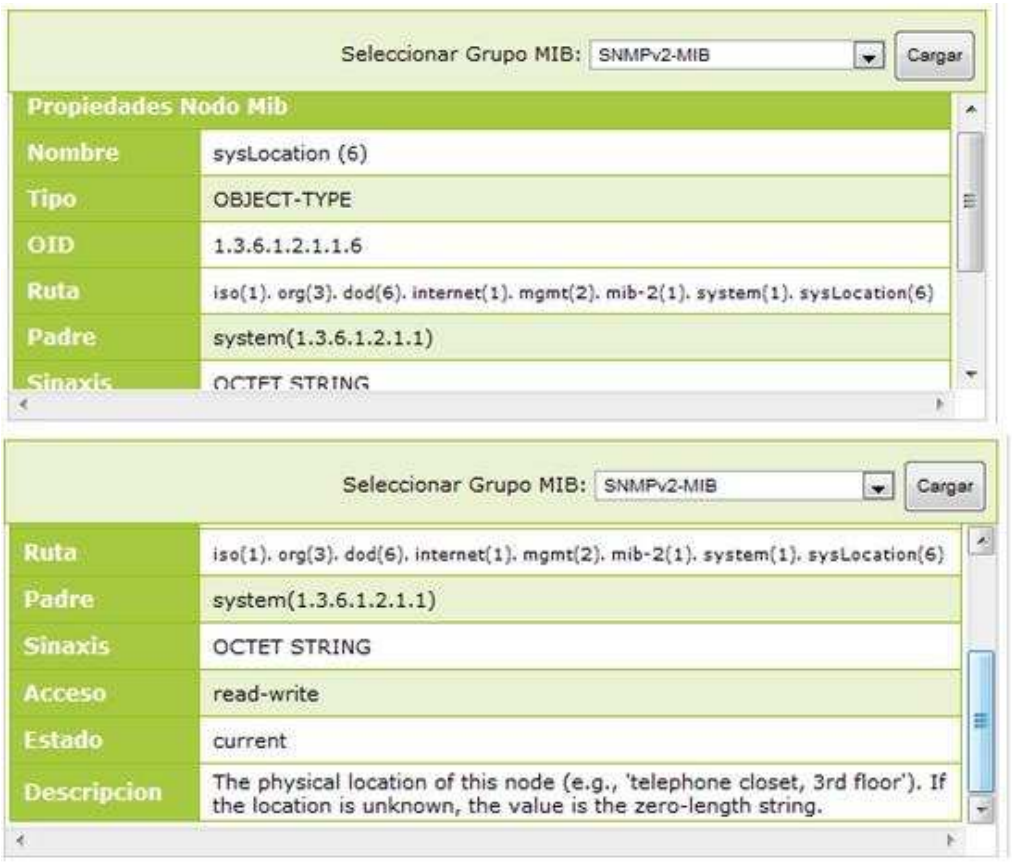


Figura 3. 30 Presentación de propiedades de un nodo MIB del tipo objeto.

3) **Sección de propiedades de nodo MIB:** Esta sección muestra las propiedades del nodo MIB seleccionado desde el árbol MIB. Estas propiedades son: nombre, tipo, objeto identificador, ruta, nodo padre, sintaxis, acceso, estado y descripción. La disponibilidad de cada propiedad depende del tipo de nodo seleccionado. Por ejemplo, si el nodo seleccionado es del tipo

objeto (OBJECT-TYPE85) entonces todas las propiedades están disponibles; y si el objeto seleccionado es de otro tipo, por ejemplo OBJECT-IDENTITY, las únicas propiedades que no se muestran son: sintaxis, acceso y estado. Ver las figuras 3.30 y 3.31.

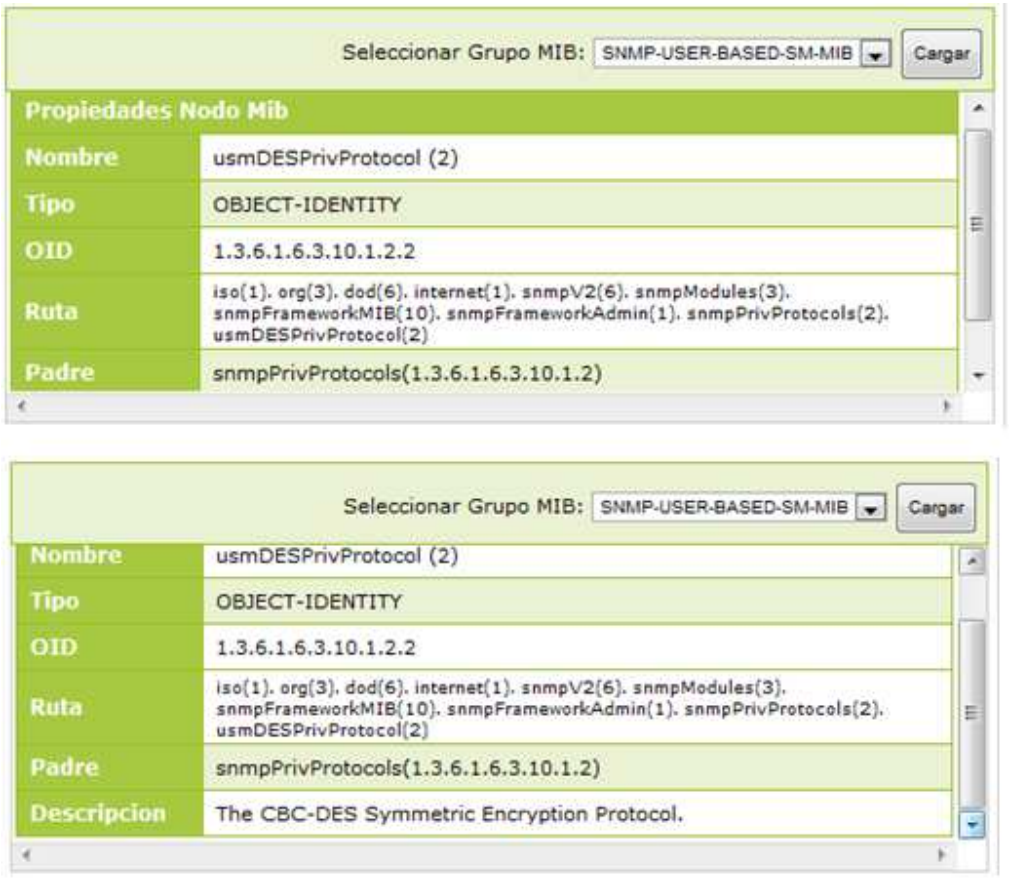


Figura 3. 31 Presentación de propiedades de un nodo MIB del tipo OBJECT-IDENTITY.

En las figuras anteriores se puede apreciar que el valor del campo *Nombre* tiene un número entre paréntesis, este número corresponde al último dígito del OID que identifica al nodo MIB; esto sucede también en el árbol MIB y en las tablas de respuestas.

[85] Los nodos MIB del tipo OBJECT-TYPE son los únicos de interés para la aplicación ya que éstos son los que realmente manejan información de administración. Nodos MIB de otros tipos tales como: MODULE-IDENTITY, OBJECT-IDENTITY, NOTIFICATION-TYPE, etc., solo sirven de soporte para los nodos de tipo objeto y para la MIB en sí.

4) **Sección de ejecución de operación SNMPv3:** Esta sección está conformada por un formulario que permite enviar casi todos los tipos de mensajes SNMPv3 hacia un determinado elemento de red.

5) **Sección de resultados:** Esta sección muestra los resultados de la operación SNMPv3 seleccionada.

Para ejecutar una operación SNMPv3 hacia un elemento de red se tiene el formulario de la sección 4 (ver figura 3.29). Este formulario contiene los siguientes campos:

- **Dirección IP:** Es una lista desplegable que contiene las direcciones IP de todos los elementos de red registrados. El usuario selecciona la dirección IP del elemento al cual va dirigido el mensaje.

- **Otra dirección:** Esta es una casilla de verificación que sirve para habilitar una caja de texto en el cual se puede ingresar una dirección IP distinta a las anteriores.

- **OID:** Es una caja de texto en el cual se especifica el objeto identificador del nodo MIB destino. El valor de este campo es establecido automáticamente cuando se selecciona un nodo del árbol MIB; sin embargo, el valor también puede ser establecido de forma manual.

- **Sintaxis:** Es una lista desplegable que contiene tipos de datos de objetos MIB. El valor es establecido automáticamente cuando se selecciona un nodo del árbol MIB.

- **Valor:** Es una caja de texto donde se ingresa el valor para el objeto MIB destino. Este campo también muestra el resultado de una operación efectuada.

- **Botones Get, GetNext, GetBulk, Walk, Set, Trap e Inform:** Son botones que ejecutan la operación según el tipo de PDU.

- **Botón Borrar:** Eliminar los resultados obtenidos de la última operación realizada.

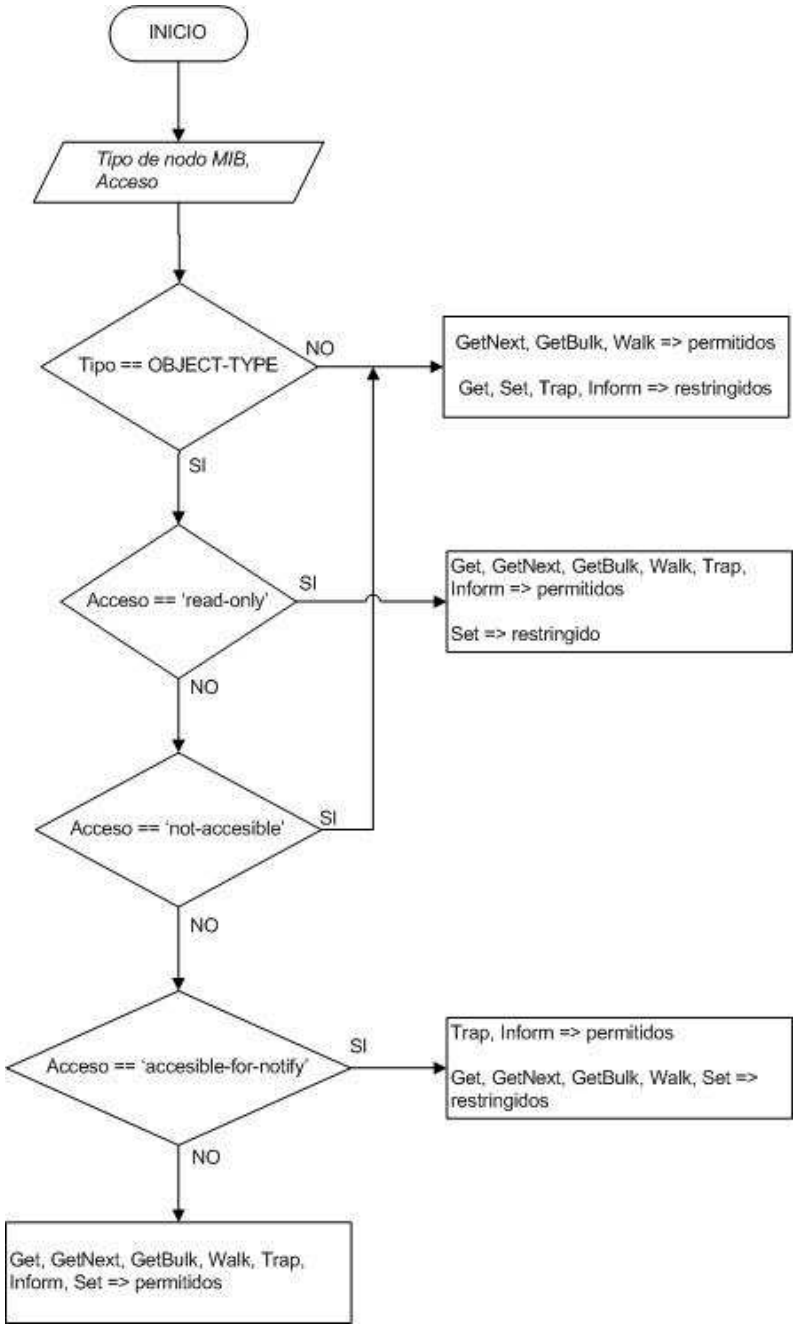


Figura 3. 32 Diagrama de flujo para restricciones de operaciones SNMPv3.

La aplicación ejecuta una lógica de restricciones de operaciones SNMPv3 al momento de seleccionarse un nodo MIB. Esta lógica considera dos parámetros: el tipo de acceso y el tipo de nodo MIB. Cuando se produce una restricción, ésta se manifiesta bloqueando el (los) botón(es) relacionado(s). La figura anterior muestra el diagrama de flujo de la lógica de restricciones.

Cuando se ejecuta una operación SNMP los resultados son mostrados en una tabla que tiene las siguientes columnas: nombre del objeto MIB, objeto identificador de la instancia del objeto, sintaxis del objeto y valor de la instancia del objeto. Además la tabla tiene un encabezado que muestra el nombre de la operación solicitada y el número de filas devueltas. La tabla de resultados se la ubica debajo del formulario de ejecución de operaciones SNMP.

La siguiente figura muestra el resultado de la ejecución de una operación *Get.* En este caso el elemento destino es el router R1 cuya dirección IP es 172.31.1.1, y el objeto solicitado es *sysDescr*, el cual pertenece al grupo MIB *system.*

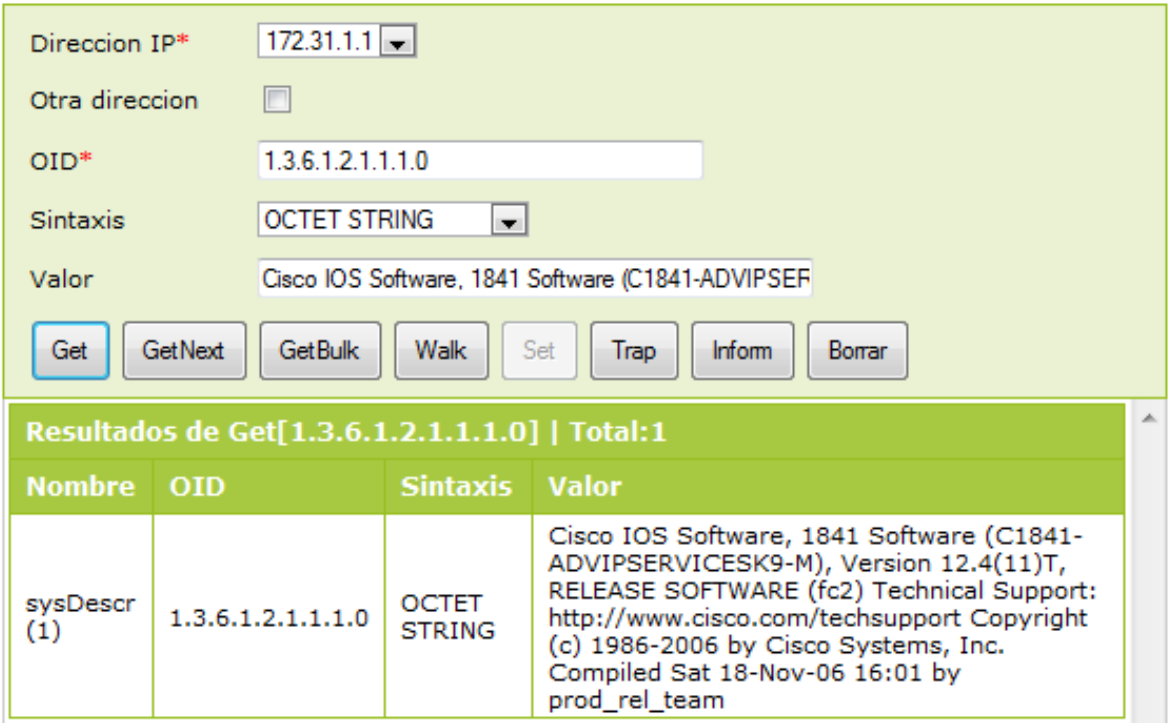


Figura 3. 33 Ejecución y resultado de una operación *Get.*

La siguiente figura muestra el resultado de la ejecución de una operación *GetNext.* En este caso el elemento destino es el switch S1 cuya dirección IP es 172.31.1.2, y el objeto solicitado es *sysName*, el cual pertenece al grupo MIB *system.* Como se puede apreciar en la imagen el OID mostrado en la tabla de resultados no corresponde al OID solicitado en el formulario; esto se debe a que este tipo de operación devuelve la siguiente instancia de objeto desde el objeto solicitado.

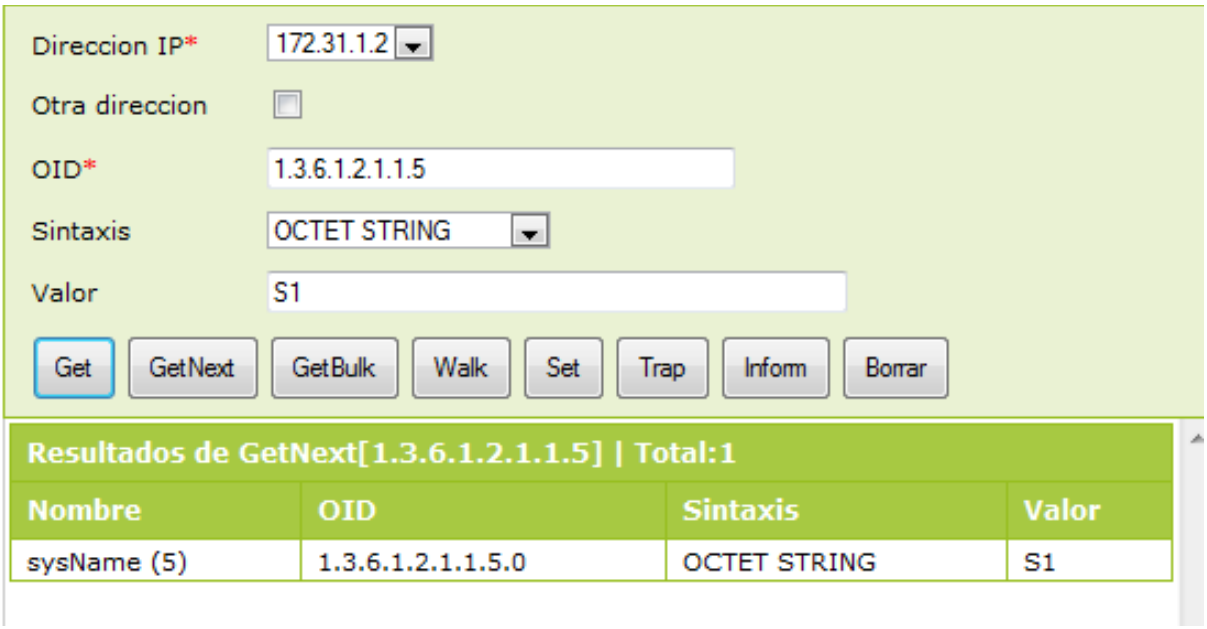


Figura 3. 34 Ejecución y resultado de una operación *GetNext*.

La siguiente figura muestra el resultado de la ejecución de una operación *GetBulk.* En este caso el elemento destino es el servidor Linux cuya dirección IP es 172.31.1.4, y el objeto solicitado es *system*. Como se puede apreciar en la figura el número de filas devueltas son 10, lo se comprueba con la configuración de parámetros de mensajes SNMPv3 realizado anteriormente.

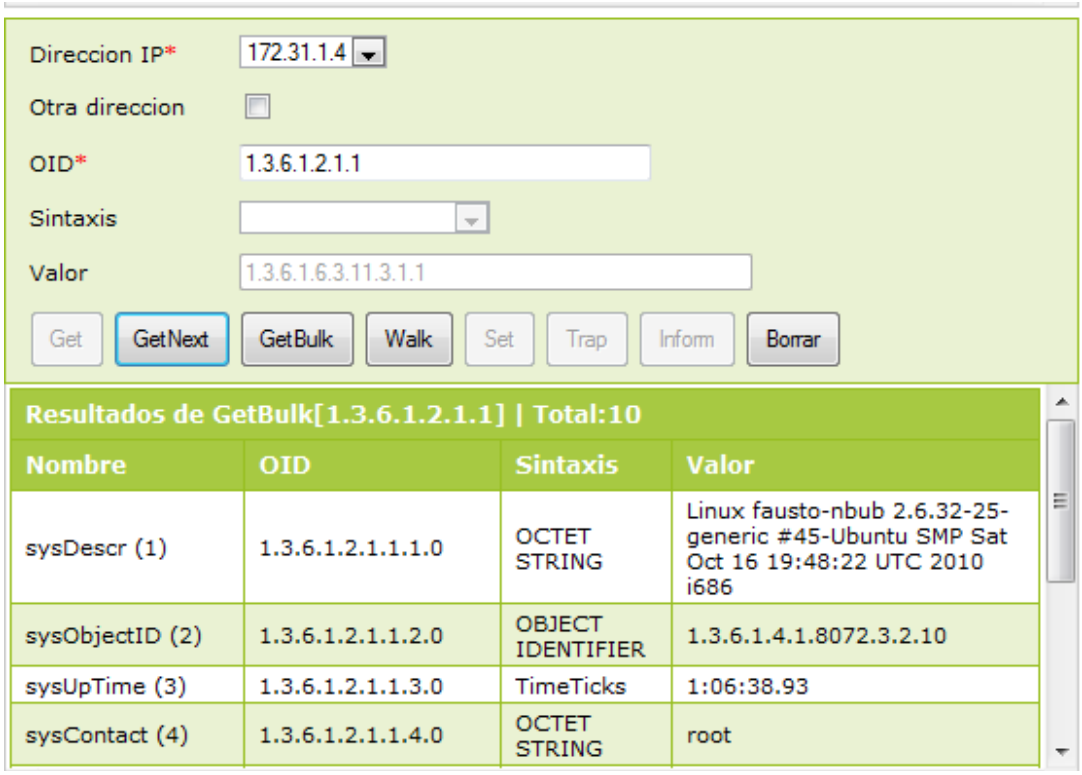


Figura 3. 35 Ejecución y resultado de una operación *GetBulk*.

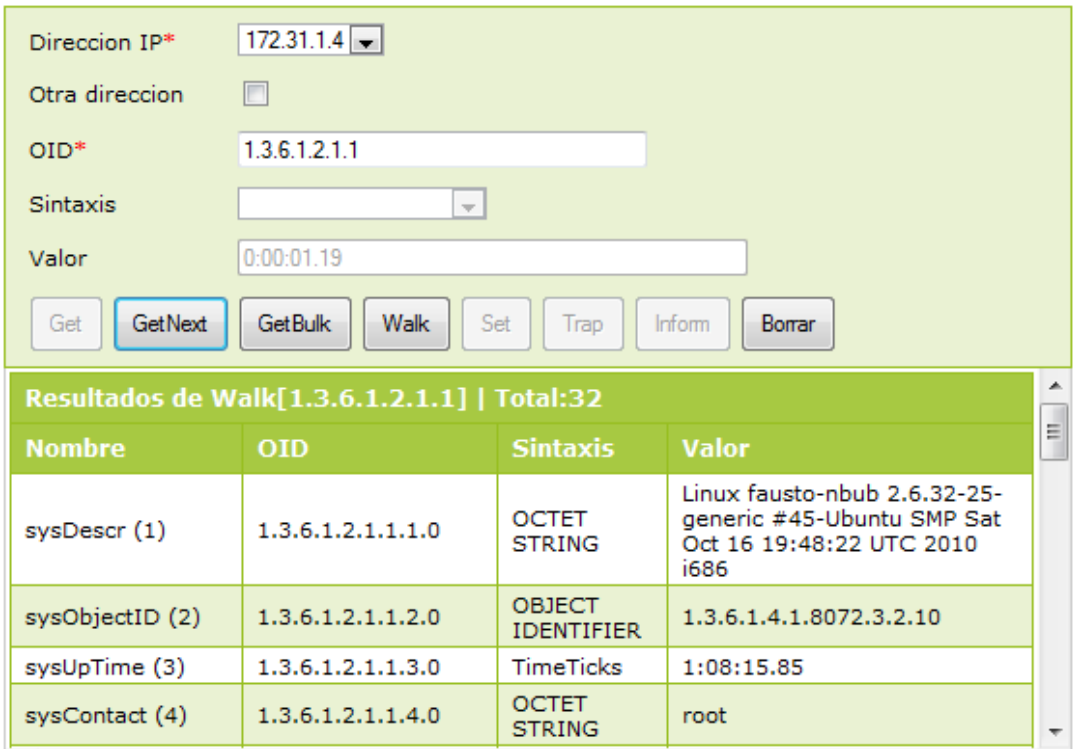


Figura 3. 36 Ejecución y resultado de una operación *Walk*.

La figura anterior muestra el resultado de la ejecución de una operación *Walk.* En este caso el elemento destino es el mismo servidor Linux cuya dirección IP es 172.31.1.4, y el objeto solicitado es también *system*. Como se puede apreciar en el título de la tabla de respuestas, el número de filas devueltas es 32; esto se debe a que este tipo de operación utiliza al mensaje *GetNext* para recorrer todo el grupo *system*.

Las dos figuras siguientes muestran un ejemplo de la ejecución de una operación *Set.* En este caso el elemento destino es el computador analizador de paquetes cuya dirección IP es 172.31.1.4, y el objeto solicitado es también *sysLocation,* el cual pertenece al grupo MIB *system*.

En la primera figura se realiza una operación *Get* para conocer el valor actual del objeto *sysLocation.* Como se pude apreciar en esta figura, el valor es uno establecido por defecto. La idea es cambiar este valor a otro, por ejemplo: *Escuela Politécnica Nacional*. Ver figura 3.37.

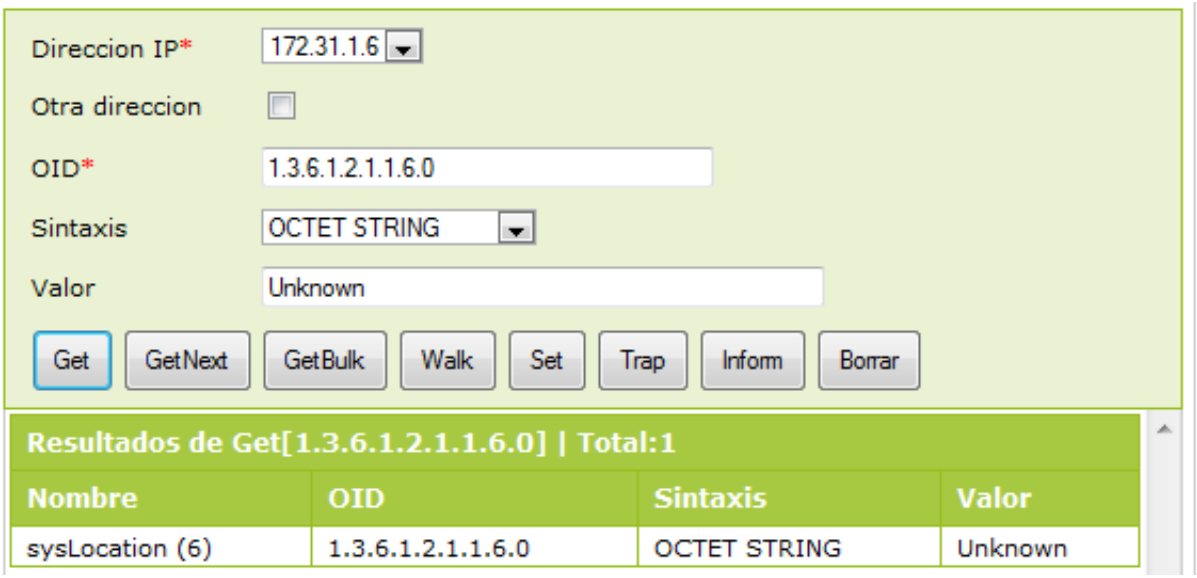


Figura 3. 37 Obtención del valor actual del objeto sysLocation previo a la operación *Set* .

En la segunda figura se realiza la operación Set para cambiar el valor establecido por defecto. Como se puede apreciar en la figura, el valor *Escuela Politécnica Nacional* es ingresado manualmente. El valor de sintaxis es establecido automáticamente desde el resultado de la operación anterior. En la tabla de resultados se puede apreciar el nuevo valor asignado al objeto *sysLocation.* Ver figura 3.38.

Figura 3. 38 Ejecución y resultado de una operación *Set*.

La siguiente figura muestra el resultado de la ejecución de una operación *Trap*. En este caso el elemento destino es el servidor Linux cuya dirección IP es 172.31.1.4, y el objeto utilizado es *sysContact* del grupo MIB *system*.

Figura 3. 39 Ejecución de una operación *Trap*.

Como se puede apreciar en la figura anterior, no se muestra una tabla de resultados sino tan solo un mensaje que indica el éxito del envío de la PDU *Trap*. Esto se debe a que este tipo de PDU no provoca que se retorne una respuesta por parte del elemento destino; sin embargo, para comprobar que la trap ha llegado a su destino se muestra la siguiente figura.

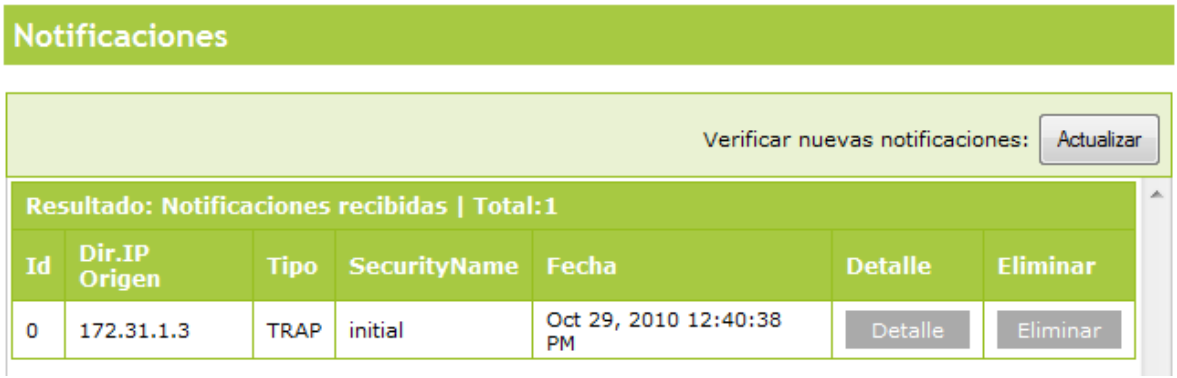


Figura 3. 40 Cuerpo de la página de notificaciones del sistema SWAR ubicado en el servidor Linux.

La figura anterior corresponde al cuerpo de la página de notificaciones recibidas del sistema SWAR ubicado en el Servidor Linux. En él se puede apreciar la *trap* enviada por el servidor XP que tiene dirección IP 172.31.1.3. La página de notificaciones será analizada con profundidad más adelante.

La siguiente figura muestra el resultado de la ejecución de una operación *Inform*. En este caso el elemento destino es el mismo servidor Linux cuya dirección IP es 172.31.1.4, y el objeto utilizado es *sysName* del grupo MIB *system*. Este tipo de operación, a diferencia de *Trap*, si provoca que se retorne una respuesta por parte del elemento destino. En esta figura se puede apreciar que la tabla de respuestas contiene tres filas correspondientes a los siguientes objetos: *sysUpTime*, *snmpTrapOID,* y *sysName.* El valor del primer objeto, *sysUpTime,* es el tiempo que ha transcurrido desde la última (re)inicialización del agente de este elemento. El valor del segundo objeto, *snmpTrapOID,* indica qué objeto MIB es utilizado como información de la notificación, y en este caso es el valor de OID del objeto *sysName.*

Finalmente, el tercer objeto, *sysName,* indica el valor de información de la notificación.

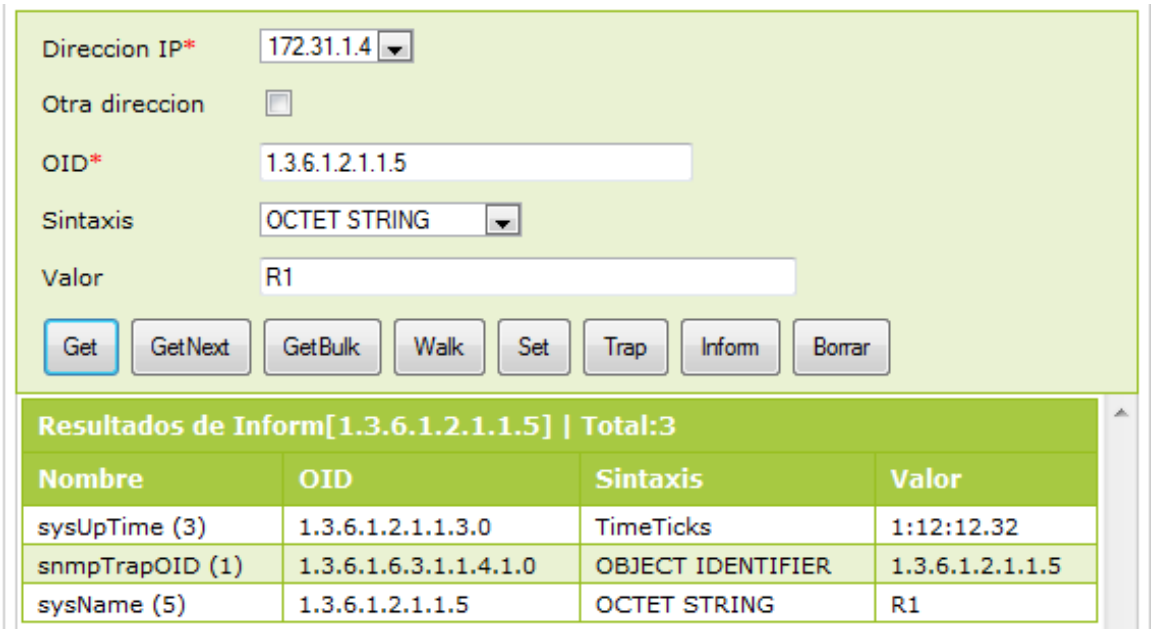


Figura 3. 41 Ejecución y resultado de una operación *Inform*.

3.2.6 VISUALIZAR NOTIFICACIONES

Este caso de uso está conformado por una página que muestra el listado de las notificaciones recibidas, las cuales han sido enviadas por los elementos de red registrados. La siguiente figura muestra parte de dicha página. Para acceder a ésta página se lo puede hacer desde el enlace *Notificaciones* (ver figura 3.5).

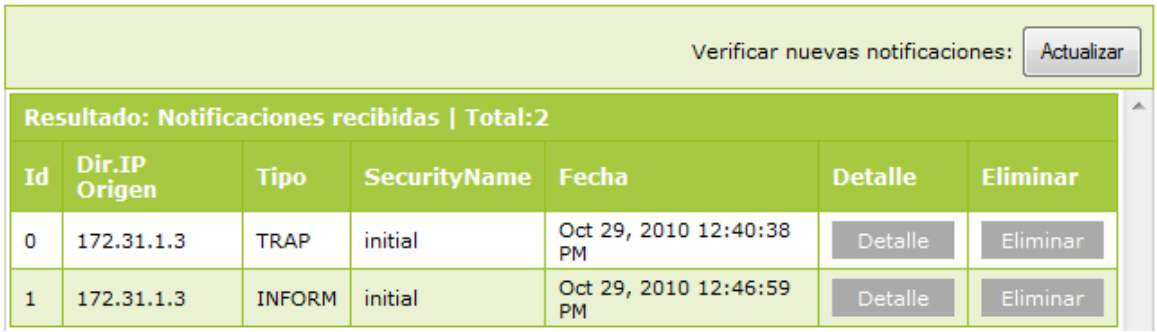


Figura 3. 42 Cuerpo de la página de notificaciones.

Cabe aclarar que la página de notificaciones mostrada en la figura anterior pertenece al sistema SWAR ubicado en el servidor Linux y las notificaciones corresponden a los mensajes *Trap* e *Inform* enviados previamente en el caso de uso anterior.

Como se puede apreciar en la figura anterior existe un formulario con un botón *Actualizar.* Este botón se utiliza para recuperar la última actualización de la lista de notificaciones. Además, existe también una tabla que presenta a la lista de notificaciones; esta tabla está compuesta por un título y siete columnas. El título muestra el número de notificaciones recibidas. Las siete columnas corresponden a lo siguiente: identificador de la notificación (*Id*), tipo de notificación, nombre del usuario (*securityName*) a quien va dirigida la notificación, fecha de recepción, un enlace de detalle para mostrar mayor información, y un enlace de eliminar para borrar la notificación.

Cuando se hace clic en el enlace *Detalle,* la aplicación muestra mayor información de la notificación. Esta información de detalle es presentada en otra tabla ubicada debajo de la lista de notificaciones recibidas. Ver la siguiente figura.

Detalle de Notificacion			
Id	0		
Dir.IP Origen	172.31.1.3		
Tipo	TRAP		
SecurityName	initial		
Nivel Seguridad	AUTH_NOPRIV		
Fecha	Friday, October 29, 2010 12:40:38 PM		
Vinculos Variables | Total:3			
Nombre	**OID**	**Sintaxis**	**Valor**
sysUpTime (3)	1.3.6.1.2.1.1.3.0	TimeTicks	1:05:50.90
snmpTrapOID (1)	1.3.6.1.6.3.1.1.4.1.0	OBJECT IDENTIFIER	1.3.6.1.2.1.1.4
sysContact (4)	1.3.6.1.2.1.1.4	OCTET STRING	Administrador

Figura 3. 43 Información de detalle de una notificación.

En la figura anterior se puede apreciar una tabla que muestra mayor información de una notificación. En este caso la información corresponde a la primera notificación de la lista (ver figura 3.42). La información adicional que muestra es la siguiente: nivel de seguridad con el que se envío la notificación, fecha de recepción (incluye el día de la semana), y vínculos variables. Los vínculos variables generalmente son tres. El primero corresponde al objeto *sysUpTime*, el cual es el tiempo que ha transcurrido desde la última (re)inicialización del agente del elemento que ha enviado la notificación. El segundo corresponde al objeto *snmpTrapOID,* el cual indica qué objeto MIB es utilizado como información de la notificación, y en este caso es el valor de OID del objeto *sysContact.* Y finalmente, el tercero corresponde al objeto *sysContact,* el cual indica el valor de información de la notificación, en este caso es la cadena de texto *Administrador*.

3.2.7 GESTIONAR MIBS

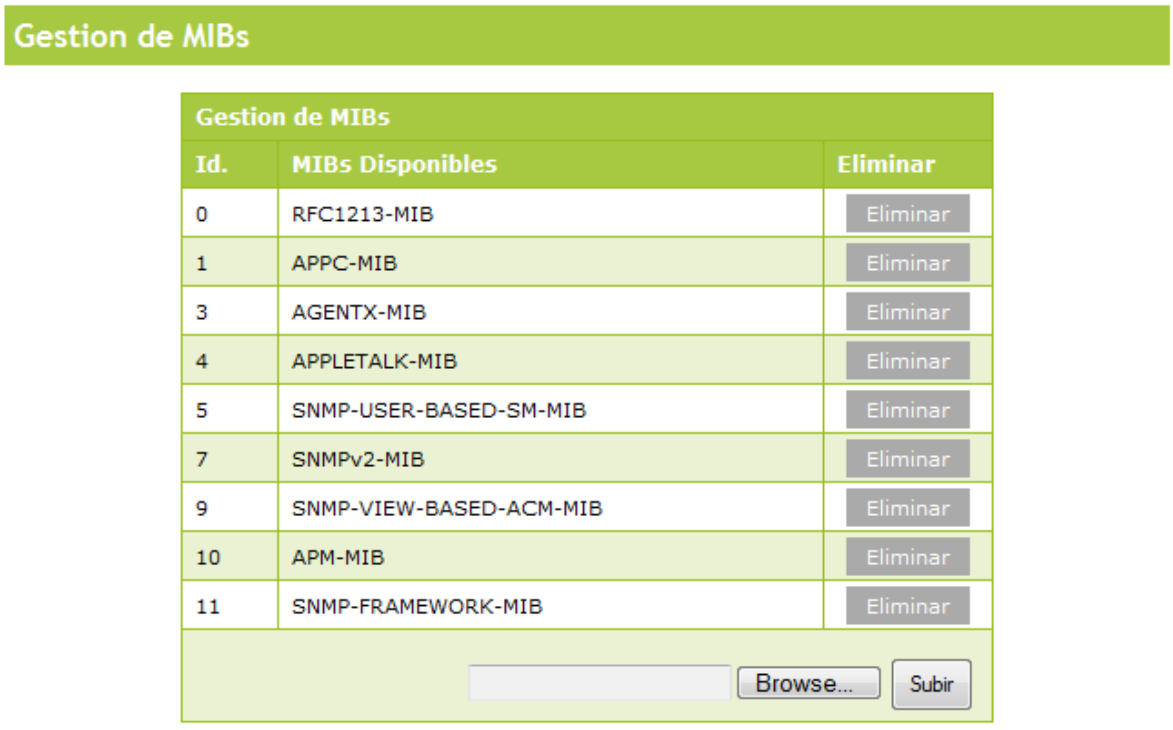


Figura 3. 44 Cuerpo de la página de gestión de MIBs.

Este caso de uso está conformado por una página que muestra el listado de grupos MIBs disponibles y que además permite subir un nuevo archivo MIB a la aplicación. La figura anterior muestra el cuerpo de dicha página. Para acceder a esta página se lo hace desde el enlace *Gestión General* > *MIBs* (ver figura 3.7). Sólo el administrador general tiene permiso de acceder a esta página.

En la figura se puede apreciar que existe una tabla cuyas filas son los grupos MIBs disponibles. Esta lista de nombres de grupos MIB es la que aparece en la sección de selección de grupo MIB de la página *Nodos MIBs* (ver figura 3.29). La tabla está conformada por tres columnas: identificador (*Id.*) del grupo MIB, nombre del grupo MIB, y un enlace para eliminar el grupo MIB de la lista.

Para subir un nuevo archivo MIB, se pulsa el botón *Browse (Explorar)* y aparece la ventana de exploración de archivos, en la cual se indica la ruta del nuevo archivo MIB (en este caso es el archivo MIB *HOST-RESOURCES-MIB*); y luego se pulsa el botón *Subir.* La aplicación recibe el archivo, lo procesa, lo transforma a uno de formato XML, lo agrega a la lista de grupos MIBs disponibles, y luego muestra un mensaje de éxito de la operación. Ver las figuras siguientes.

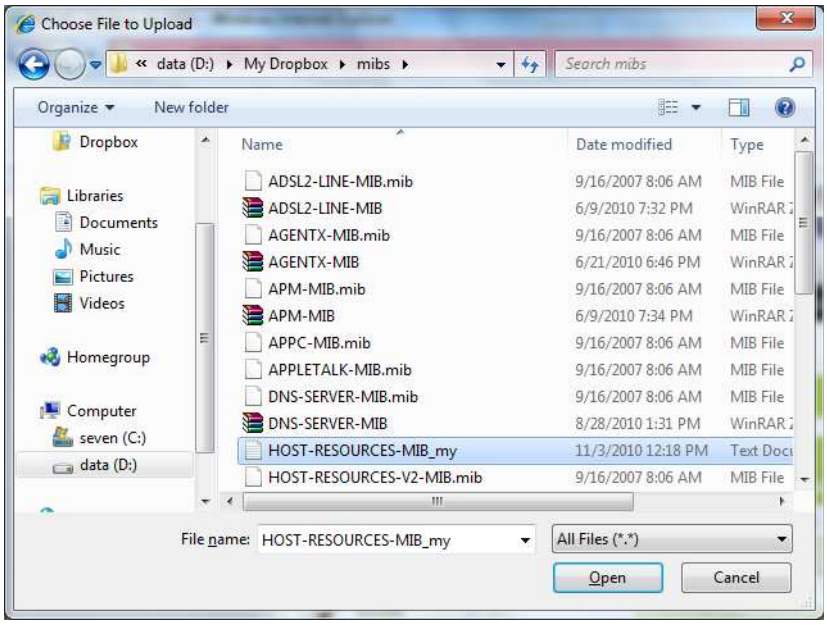


Figura 3. 45 Ventana de selección del archivo MIB a subir.

Figura 3. 46 Lista actualizada de grupos MIBs y mensaje de éxito de la operación.

El archivo MIB que se desea subir debe tener el formato y sintaxis adecuados para que sea reconocido por el sistema al momento de subirlo. Generalmente la extensión de este archivo debe ser *.mib,* o puede ser también un archivo de texto simple (puede ser de extensión *.txt*). En todo caso, si el formato del archivo es inadecuado o si el archivo contiene errores de sintaxis entonces la aplicación muestra un mensaje de error. Los errores de sintaxis significan que el archivo MIB no ha sido correctamente estructurado.

3.2.8 GESTIONAR USUARIOS

Este caso de uso está conformado por tres páginas: la primera que permite crear y eliminar cuentas de usuarios localmente en la aplicación, la segunda que permite exportar cuentas de usuarios hacia determinados elementos de red, y la tercera que permite anular cuentas de usuarios en determinados elementos de red. Para acceder a la primera página se lo hace desde el enlace *Gestión General* > *Usuarios* (ver figura 3.7). La segunda y tercera páginas son accedidas desde la primera página. Sólo el administrador general tiene permiso de acceder a estas páginas.

Figura 3. 47 Cuerpo de la página de gestión de usuarios.

La figura anterior muestra la primera página que conforma este caso de uso. Esta página permite crear y eliminar cuentas de usuarios en el repositorio local de la aplicación. En la figura se puede apreciar que existe un formulario y una tabla de usuarios existentes. La tabla que muestra la lista de usuarios existentes está conformada por un título y cuatro columnas. El título indica el número de usuarios

existentes, mientras que las cuatro columnas corresponden a: número de fila, nombre de usuario (*securityName*), tipo de administrador (sólo dos valores posibles: general o limitado), y nombre del grupo VACM al cual pertenece el usuario. El formulario *Parámetros de Usuario* (ver figura anterior) está conformado por los siguientes campos:

- **SecurityName:** Define el nombre de seguridad del usuario.

- **Password autenticación:** Define el password de autenticación del usuario. Este password no debe ser una cadena vacía ya que de tal modo el nuevo usuario no podrá iniciar sesión en el sistema.

- **Password privacidad:** Define el password de privacidad del usuario. Este password si puede ser una cadena vacía, y si ese fue el caso los mensajes SNMP enviados en nombre de este usuario como máximo nivel de seguridad debería ser *AUTH_NOPRIV* (ver figura 3.15).

- **Grupo VACM:** Es una lista desplegable de los nombres de los grupos VACM existentes. El nombre seleccionado determina que el usuario pertenecerá a ese grupo.

- **Otro grupo:** Es una casilla de verificación que habilita una caja de texto para definir un nuevo grupo VACM al cual pertenecerá el nuevo usuario. Este es un control alternativo al anterior.

- **Tipo de administrador:** Define el tipo de administrador del usuario.

Para crear localmente una nueva cuenta de usuario se debe pulsar el botón *Nuevo* (esto provoca que los campos del formulario se habiliten), se ingresan los valores correspondientes en cada uno de los campos, y finalmente se graban los datos pulsando el botón *Guardar.*

El botón *Limpiar* sirve para vaciar completamente los campos del formulario, y el botón Cancelar sirve para deshacer los cambios realizados.

Para realizar el proceso de exportación de una cuenta de usuario hacia determinados elementos de red se hace clic en el nombre del usuario (en este caso es el usuario *admin2*) que está ubicado en la tabla de usuarios existentes (ver figura 3.47) y luego se pulsa el botón *Exportar.* Este botón indica a la aplicación la cuenta de usuario que se desea exportar y presenta la página de exportación de usuario. Ver la figura siguiente.

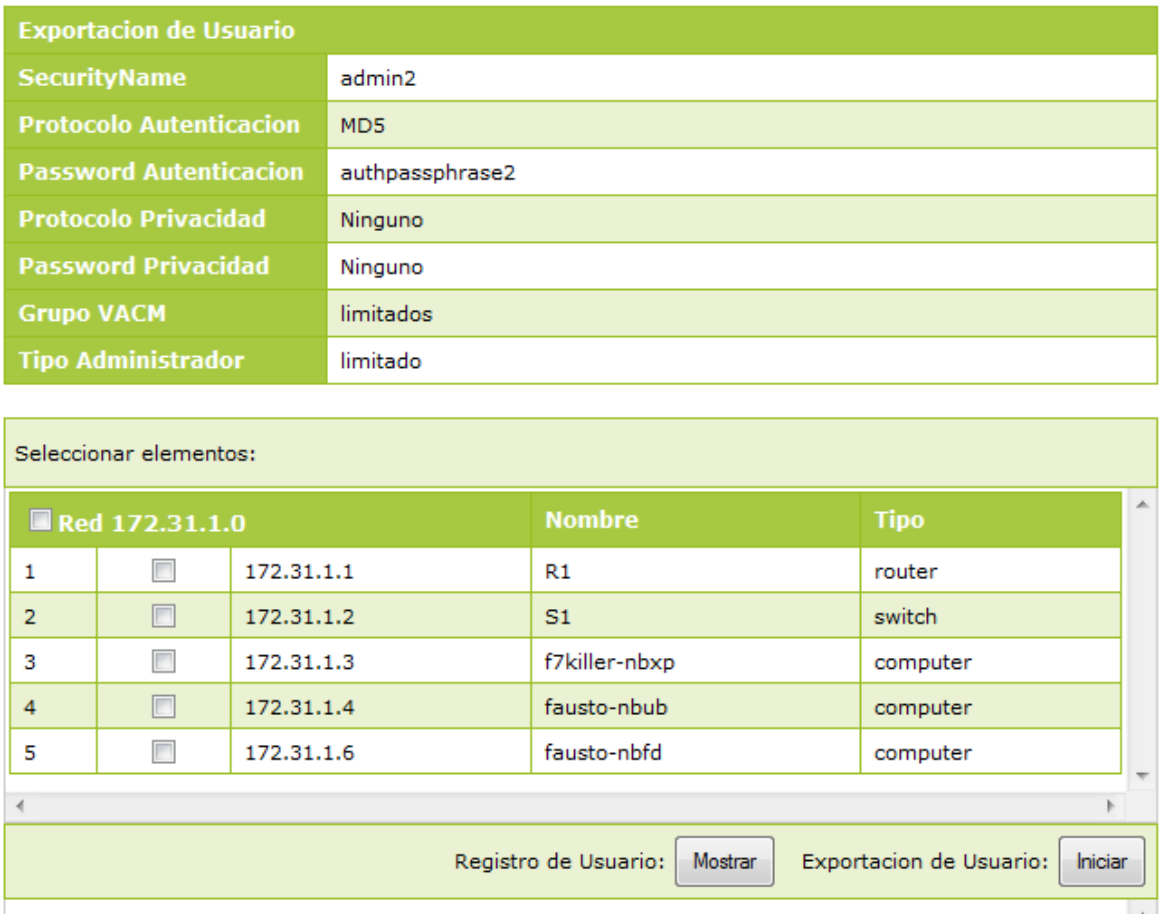


Figura 3. 48 Cuerpo de la página de exportación de usuario.

En la figura anterior se puede apreciar el cuerpo de la página de exportación de usuario. El cuerpo de la página está divida en tres secciones: la superior que contiene información de las características del usuario que se desea exportar, la intermedia que contiene un formulario que permite seleccionar los elementos destinos, y la inferior que contiene un formulario y un espacio dedicado a la presentación del informe de la exportación.

Entonces, se seleccionan los elementos de red a los cuales se desea exportar (también es posible seleccionar redes enteras haciendo clic en la casilla de la red) y luego se pulsa el botón *Iniciar.* En ese momento la aplicación comienza con el proceso de exportación. La exportación está basada en el envío secuencial[86] de mensajes SNMP de recuperación y establecimiento de valores a objetos MIB determinados. Estos objetos MIB pertenecen al grupo MIB *snmpUsmMIB*[87].

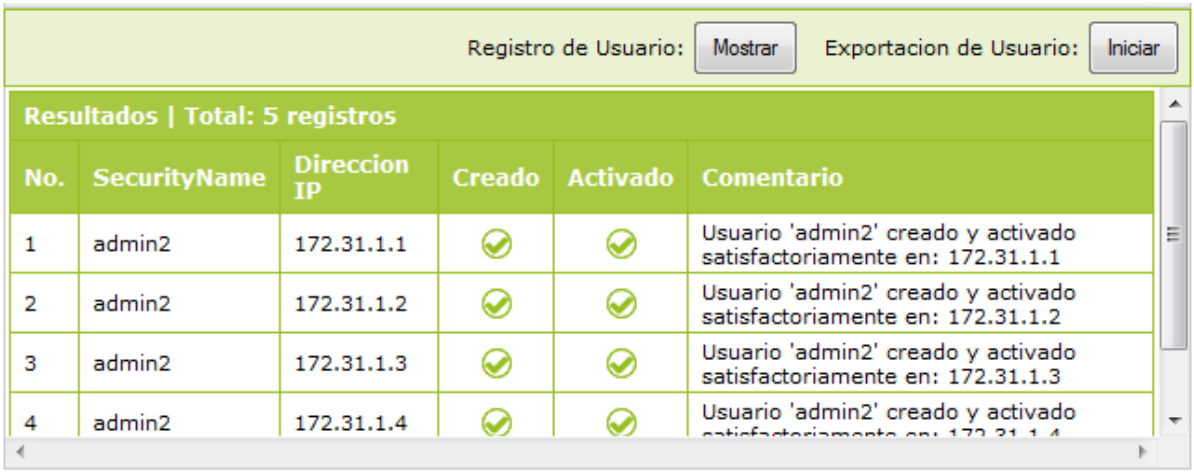


Figura 3. 49 Informe de los resultados de la exportación de usuario.

Una vez finalizado el proceso se muestra el informe de los resultados de la exportación (ver figura anterior). Este informe básicamente es una tabla que está

[86] El envío secuencial de mensajes SNMP está basada en la recomendación del RFC 3414, *User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3),* para crear usuarios USM. Para mayor información ver las págs.39 - 41 del RFC 3414.

[87] Para mayor información revisar el ítem 1.4.4.9 “MIBs para USM”.

conformada por un título y seis columnas. El título muestra el número de filas devueltas. Cada fila corresponde a un *Registro de Usuario.* Un registro de usuario es conjunto de datos que relaciona la existencia de una cuenta usuario en un elemento de red. Un registro de usuario está definido por las seis columnas de la siguiente manera: número de fila (sirve como identificador del registro), nombre del usuario (*securityName*), dirección IP del elemento, indicador del estado de la creación del usuario, indicador del estado de la activación del usuario, y un comentario.

Los indicadores de creación y activación son importantes para conocer si la nueva cuenta de usuario está lista para ser utilizada. Sí y solamente sí ambos indicadores tienen el valor de visto, entonces la cuenta de usuario puede ser utilizada. Si alguno de los indicadores tiene una "X", significa que ha ocurrido un error en el proceso y el campo de comentario se encarga de indicar lo sucedido.

El botón *Mostrar* (ver figura 3.49) sirve para presentar los registros del usuario seleccionado. Esta función es útil para revisar los elementos en los cuales la cuenta de usuario ya está creada y activada, y en base a ello tomar la decisión sobre qué elementos se deben escoger para realizar una nueva exportación o anulación.

Para realizar el proceso de anulación de una cuenta de usuario en determinados elementos de red se hace clic en el nombre del usuario (en este caso es el usuario *admin2*) que está ubicado en la tabla de usuarios existentes (ver figura 3.47) y luego se pulsa el botón *Anular.* Este botón indica a la aplicación la cuenta de usuario que se desea anular y presenta la página de anulación de usuario.

En la figura siguiente se puede apreciar el cuerpo de la página de anulación de usuario. El cuerpo de la página es similar al de la página de exportación, con la diferencia de que, en la sección inferior se presenta una tabla con los resultados de la anulación.

Figura 3. 50 Cuerpo de la página de anulación de usuario.

Entonces, se seleccionan los elementos de red a los cuales se desea anular (también es posible seleccionar redes enteras haciendo clic en la casilla de la red) y luego se pulsa el botón *Iniciar.* En ese momento la aplicación comienza con el proceso de anulación. El anulación consiste en la eliminación de filas de la tabla *usmUserTable*[88] perteneciente al grupo MIB *snmpUsmMIB*.

Una vez finalizado el proceso se muestra los resultados de la anulación (ver figura siguiente). Los resultados se presentan también en una tabla que está conformada por un título y tres columnas. El título muestra el número de filas devueltas. Las tres

[88] Para mayor información revisar el ítem 1.4.4.9 “MIBs para USM”.

columnas corresponden a lo siguiente: número de fila, indicador del resultado de la anulación, y una información de detalle del proceso de anulación.

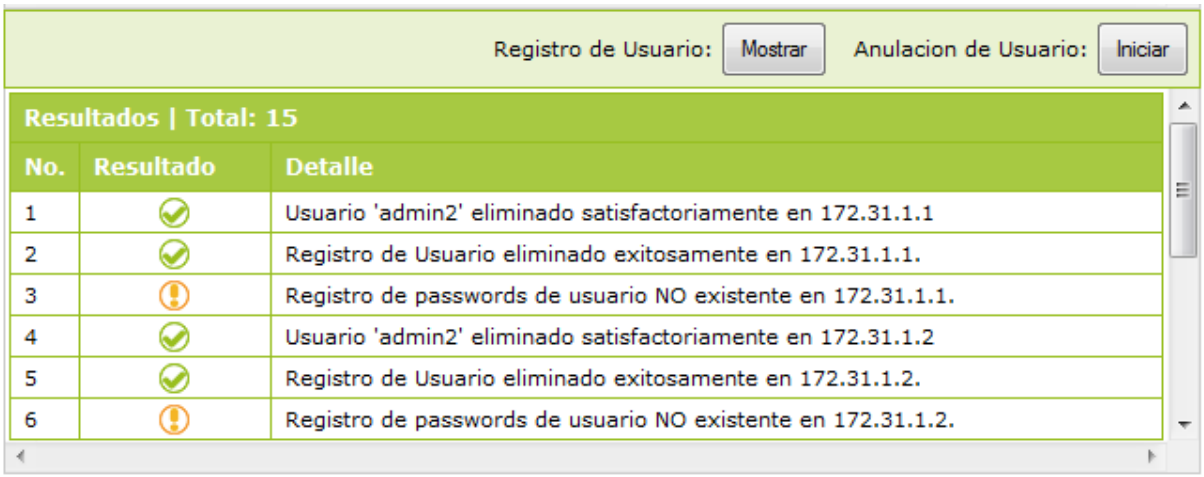


Figura 3. 51 Resultados del proceso de anulación de un usuario.

Las filas de la tabla de resultados de anulación, a diferencia de las filas de la tabla de resultados de exportación, sólo son informativas; es decir, sólo sirven para presentar mensajes de éxito o fracaso de la operación de anulación y por lo tanto esta información no es almacenada en los repositorios del sistema.

Otro punto importante que se debe destacar del proceso de anulación es que también elimina registros de usuario y registros de passwords de usuario (éstos se verán más adelante) relacionados. Esto se realiza con el fin de mantener consistencia en los datos que maneja la aplicación.

En la figura anterior se puede apreciar que existen filas con un indicador de admiración (!), esto significa que no se ha producido la eliminación de la fila de algún registro (en este caso registro de passwords de usuario) debido a que simplemente no existe. Un registro de passwords de usuario solamente existe en el momento que se haya realizado una exportación de cambio de claves, lo cual se verá más adelante.

3.2.9 GESTIONAR VISTAS

Este caso de uso está conformado por tres páginas: la primera que permite crear y eliminar vistas MIB localmente en la aplicación, la segunda que permite exportar vistas MIB hacia determinados elementos de red, y la tercera que permite anular vistas MIB en determinados elementos de red. Para acceder a la primera página se lo hace desde el enlace *Gestión General* > *Vistas* (ver figura 3.7). La segunda y tercera páginas son accedidas desde la primera página. Sólo el administrador general tiene permiso de acceder a estas páginas.

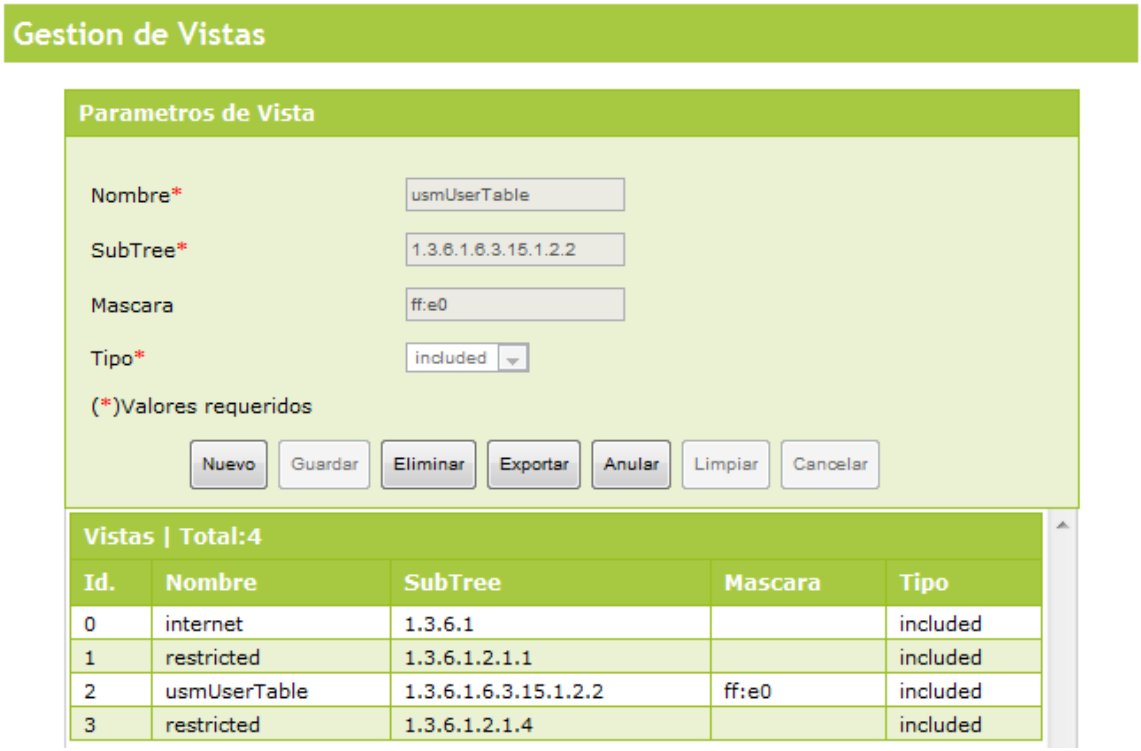


Figura 3. 52 Cuerpo de la página de gestión de vistas.

La figura anterior muestra la primera página que conforma este caso de uso. Esta página permite crear y eliminar vistas MIB en el repositorio local de la aplicación. En la figura se puede apreciar que existe un formulario y una tabla de vistas existentes. La tabla que muestra la lista de vistas MIB existentes está conformada por un título y

cinco columnas. El título indica el número de vistas existentes, mientras que las cinco columnas corresponden a: número de fila (sirve como identificador de la vista), nombre de la vista, sub-árbol (*subtree*), máscara y tipo (sólo dos valores posibles: *included* o *excluded*).

Es posible que varias vistas tengan el mismo nombre, esto sucede cuando se necesita precisar los objetos MIB a los cuales se permite el acceso.

El formulario *Parámetros de Vista* (ver figura anterior) está conformado por los siguientes campos:

- **Nombre:** Define el nombre de la vista.

- **SubTree:** Define el sub-árbol que conforma la vista.

- **Máscara:** Define la máscara asociada al sub-árbol. El valor la máscara debe ser expresado en octetos hexadecimales y separados por dos puntos. Estos octetos expresados en binarios deben corresponder con cada número del valor OID del sub-árbol. Si el valor de un bit de la máscara es 1, entonces indica que el correspondiente OID del sub-árbol debe emparejar con el OID que se solicita en el mensaje SNMP. En cambio, si el valor de un bit de la máscara es 0, entonces indica que el OID correspondiente, que se solicita en el mensaje SNMP, puede ser cualquiera. Si el valor de la máscara es una cadena vacía, entonces la máscara equivale a todos los bits igual a 1.

- **Tipo:** Define si la combinación resultante entre el sub-árbol y la máscara debe ser incluida o excluida de la vista.

Para crear localmente una nueva vista se debe pulsar el botón *Nuevo* (esto provoca que los campos del formulario se habiliten), se ingresan los valores correspondientes

en cada uno de los campos, y finalmente se graban los datos pulsando el botón *Guardar.*

El botón *Limpiar* sirve para vaciar completamente los campos del formulario, y el botón Cancelar sirve para deshacer los cambios realizados.

Para realizar el proceso de exportación de una vista MIB hacia determinados elementos de red se hace clic en el nombre de la vista (en este caso es la vista *restricted*) que está ubicada en la tabla de vistas existentes (ver figura 3.52) y luego se pulsa el botón *Exportar.* Este botón indica a la aplicación la vista que se desea exportar y presenta la página de exportación de vista. Ver la figura siguiente.

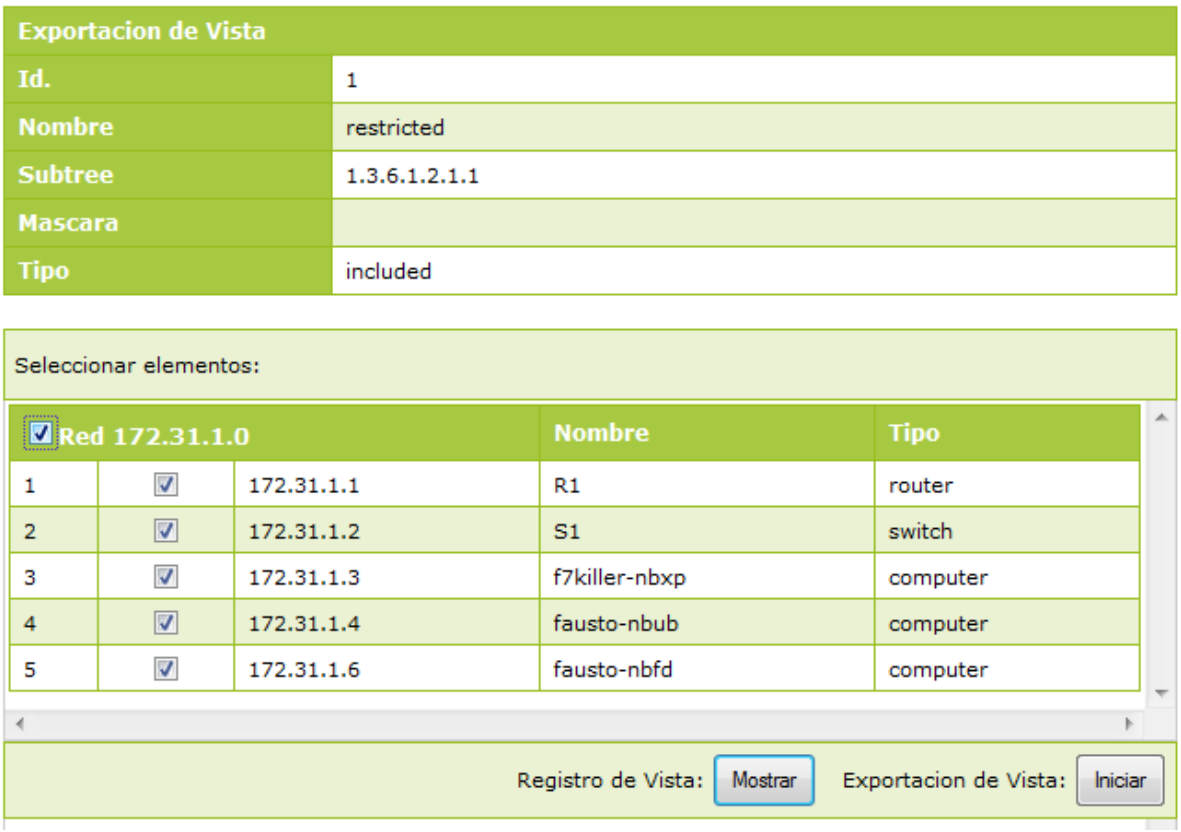


Figura 3. 53 Cuerpo de la página de exportación de vista.

En la figura anterior se puede apreciar el cuerpo de la página de exportación de vista. El cuerpo de la página está divida en tres secciones: la superior que contiene

información de las características de la vista que se desea exportar, la intermedia que contiene un formulario que permite seleccionar los elementos destinos, y la inferior que contiene un formulario y un espacio dedicado a la presentación del informe de la exportación.

Entonces, se seleccionan los elementos de red a los cuales se desea exportar (también es posible seleccionar redes enteras haciendo clic en la casilla de la red) y luego se pulsa el botón *Iniciar.* En ese momento la aplicación comienza con el proceso de exportación. La exportación está basada en el envío de mensajes SNMP de recuperación y establecimiento de valores a objetos MIB determinados. Estos objetos MIB pertenecen a la tabla *vacmViewTreeFamilyTable*, y ésta a su vez pertenece al grupo MIB *snmpVacmMIB*[89].

Figura 3. 54 Informe de los resultados de la exportación de una vista.

Una vez finalizado el proceso se muestra el informe de los resultados de la exportación (ver figura anterior). Este informe básicamente es una tabla que está conformada por un título y cinco columnas. El título muestra el número de filas devueltas. Cada fila corresponde a un *Registro de Vista*. Un registro de vista es conjunto de datos que relaciona la existencia de una vista MIB en un elemento de red. Un registro de vista está definido por las cinco columnas de la siguiente manera: número de fila (sirve como identificador del registro), número de identificación de la

[89] Para mayor información revisar el ítem 1.4.5.2 “MIBs para VACM”.

vista, dirección IP del elemento, indicador del estado de la creación de la vista, y un comentario. El indicador de creación sirve para conocer si la nueva vista está lista para ser utilizada.

El botón *Mostrar* (ver figura 3.54) sirve para presentar los registros de la vista MIB seleccionada. Esta función es útil para revisar los elementos en los cuales la vista ya está creada, y en base a ello tomar la decisión sobre qué elementos se deben escoger para realizar una nueva exportación o anulación.

Para realizar el proceso de anulación de una vista MIB en determinados elementos de red se hace clic en el nombre de la vista (en este caso es la vista *restricted* con id. 1) que está ubicada en la tabla de vistas existentes (ver figura 3.52) y luego se pulsa el botón *Anular.* Este botón indica a la aplicación la vista MIB que se desea anular y presenta la página de anulación de vista. Ver la figura siguiente.

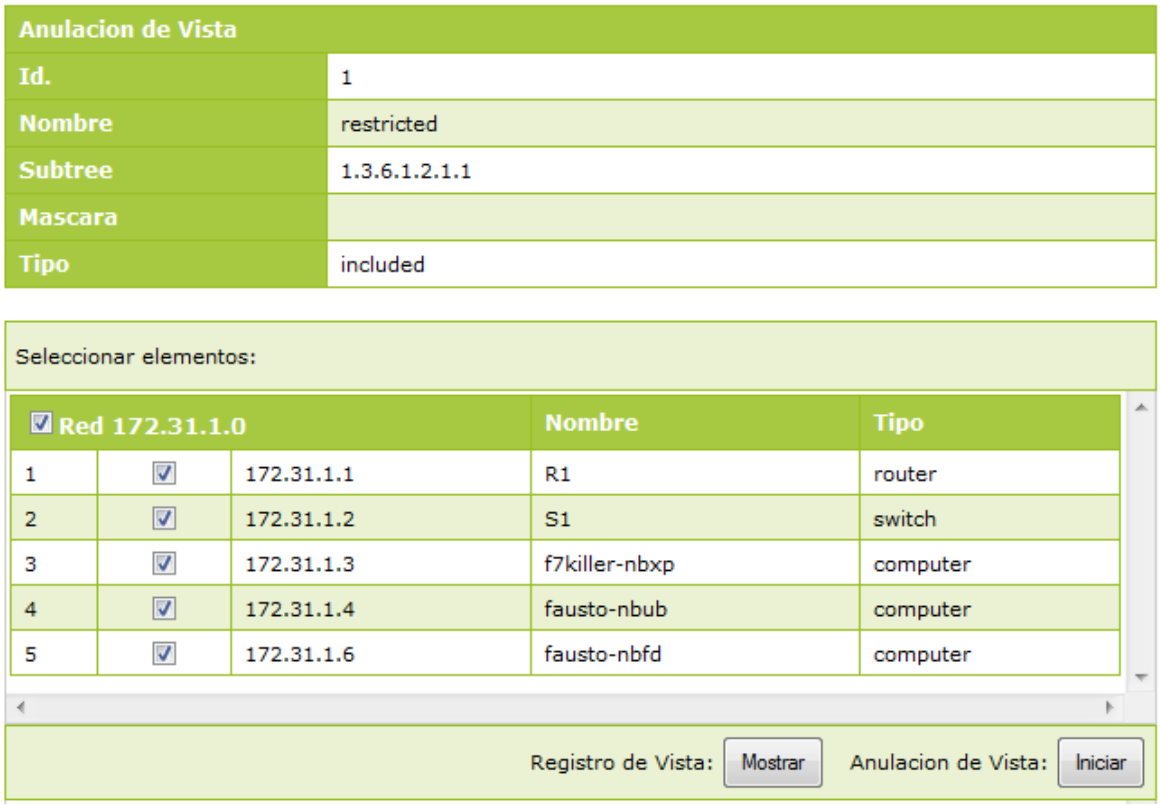


Figura 3. 55 Cuerpo de la página de anulación de vista.

En la figura anterior se puede apreciar el cuerpo de la página de anulación de vista. El cuerpo de la página es similar al de la página de exportación, con la diferencia de que, en la sección inferior se presenta una tabla con los resultados de la anulación.

Entonces, se seleccionan los elementos de red a los cuales se desea anular (también es posible seleccionar redes enteras haciendo clic en la casilla de la red) y luego se pulsa el botón *Iniciar.* En ese momento la aplicación comienza con el proceso de anulación. La anulación consiste en la eliminación de filas de la tabla *vacmViewTreeFamilyTable.*

Una vez finalizado el proceso se muestra los resultados de la anulación (ver figura siguiente). Los resultados se presentan también en una tabla que está conformada por un título y tres columnas. El título muestra el número de filas devueltas. Las tres columnas corresponden a lo siguiente: número de fila, indicador del resultado de la anulación, y una información de detalle del proceso de anulación.

Figura 3. 56 Resultados del proceso de anulación de una vista.

Las filas de la tabla de resultados de anulación, a diferencia de las filas de la tabla de resultados de exportación, sólo son informativas; es decir, sólo sirven para presentar mensajes de éxito o fracaso de la operación de anulación y por lo tanto esta información no es almacenada en los repositorios del sistema.

Otro punto importante que se debe destacar del proceso de anulación es que también elimina registros de vista. Esto se realiza con el fin de mantener consistencia en los datos que maneja la aplicación.

3.2.10 GESTIONAR DERECHOS DE ACCESO

Este caso de uso está conformado por tres páginas: la primera que permite crear y eliminar derechos de acceso localmente en la aplicación, la segunda que permite exportar derechos de acceso hacia determinados elementos de red, y la tercera que permite anular derechos de acceso en determinados elementos de red. Para acceder a la primera página se lo hace desde el enlace *Gestión General* > *Derecho de Acceso* (ver figura 3.7). La segunda y tercera páginas son accedidas desde la primera página. Sólo el administrador general tiene permiso de acceder a estas páginas.

Figura 3. 57 Cuerpo de la página de gestión de derechos de acceso.

La figura anterior muestra la primera página que conforma este caso de uso. Esta página permite crear y eliminar derechos de acceso en el repositorio local de la aplicación. En la figura se puede apreciar que existe un formulario y una tabla de derechos de acceso existentes. La tabla que muestra la lista de derechos de acceso existentes está conformada por un título y ocho columnas. El título indica el número de derechos de acceso existentes, mientras que las ocho columnas corresponden a: identificador del derecho de acceso, nombre del grupo VACM, prefijo de contexto, nivel de seguridad, *match* (emparejador) de contexto, nombre de la vista de lectura, nombre de la vista de escritura, y nombre de la vista de notificación.

El formulario *Parámetros de Derechos de Acceso* (ver figura anterior) está conformado por los siguientes campos:

- **Grupo VACM:** Define el nombre del grupo VACM al cual se le permite acceder a las vistas de lectura, escritura y notificación.

- **Prefijo de contexto:** Define el prefijo de contexto con el cual el nombre de contexto (*contextName*) del mensaje SNMPv3 debe emparejar parcial o totalmente (esto lo define el *match* de contexto) para que se permita el acceso. El valor por defecto es una cadena vacía, lo cual empareja exactamente con la cadena vacía del nombre de contexto (que igual es una cadena vacía) por defecto. Revisar campo *contextName* (nombre de contexto) de la figura 3.16.

- **Nivel de seguridad:** Define el nivel de seguridad mínimo para que se permita el acceso a las vistas de lectura, escritura y notificación.

- **Match de contexto:** Define si el nombre de contexto debe emparejar parcial o totalmente contra el prefijo de contexto para que se permita el acceso. Los valores posibles son: *exact (*totalmente*)* y *prefix* (parcialmente).

- **Vista de lectura:** Define el nombre de la vista para la cual se permite acceso sólo para lectura.

- **Vista de escritura:** Define el nombre de la vista para la cual se permite acceso para escritura.

- **Vista de notificación:** Define el nombre de la vista para la cual se permite acceso para notificación.

En el caso de que existan varias vistas almacenadas en el sistema que tengan el mismo nombre, entonces los valores de los campos anteriores (vistas de: lectura, escritura y notificación) harán referencia a todo el conjunto de vistas que tengan el mismo nombre.

Para crear localmente un derecho de acceso se debe pulsar el botón *Nuevo* (esto provoca que los campos del formulario se habiliten), se ingresan los valores correspondientes en cada uno de los campos, y finalmente se graban los datos pulsando el botón *Guardar.*

El botón *Limpiar* sirve para vaciar completamente los campos del formulario, y el botón Cancelar sirve para deshacer los cambios realizados.

Para realizar el proceso de exportación de un derecho de acceso hacia determinados elementos de red se hace clic en el nombre del grupo VACM (en este caso es el grupo VACM *limitados*) que está ubicado en la tabla de derechos de acceso existentes (ver figura 3.67) y luego se pulsa el botón *Exportar.* Este botón indica a la aplicación el derecho de acceso que desea exportar y presenta la página de exportación de derecho de acceso.

En la figura siguiente se puede apreciar el cuerpo de la página de exportación de derecho de acceso. El cuerpo de la página está divida en tres secciones: la superior

que contiene información de las características del derecho de acceso que se desea exportar, la intermedia que contiene un formulario que permite seleccionar los elementos destinos, y la inferior que contiene un formulario y un espacio dedicado a la presentación del informe de la exportación.

Figura 3. 58 Cuerpo de la página de exportación de derecho de acceso.

Entonces, se seleccionan los elementos de red a los cuales se desea exportar (también es posible seleccionar redes enteras haciendo clic en la casilla de la red) y luego se pulsa el botón *Iniciar.* En ese momento la aplicación comienza con el proceso de exportación. La exportación está basada en el envío de mensajes SNMP de recuperación y establecimiento de valores a objetos MIB determinados. Estos

objetos MIB pertenecen a la tabla *vacmAccessTable*, y ésta a su vez pertenece al grupo MIB *snmpVacmMIB*[90].

Figura 3. 59 Informe de los resultados de la exportación de un derecho de acceso.

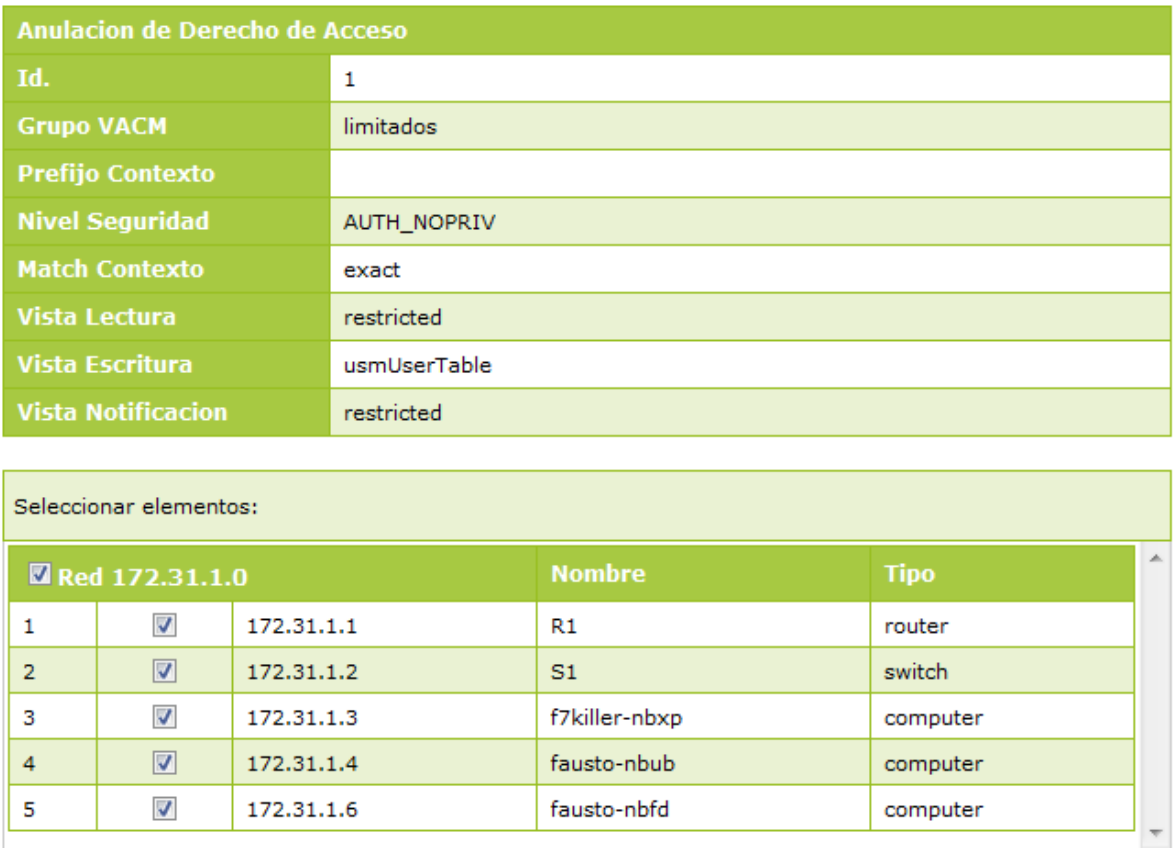
Una vez finalizado el proceso se muestra el informe de los resultados de la exportación (ver figura anterior). Este informe básicamente es una tabla que está conformada por un título y cinco columnas. El título muestra el número de filas devueltas. Cada fila corresponde a un *Registro de Derecho de Acceso.* Un registro de derecho de acceso es un conjunto de datos que relaciona la existencia de un derecho de acceso en un elemento de red. Un registro de derecho de acceso está definido por las cinco columnas de la siguiente manera: número de fila (sirve como identificador del registro), número de identificación del derecho de acceso, dirección IP del elemento, indicador del estado de la creación del derecho de acceso, y un comentario. El indicador de creación sirve para conocer si el nuevo derecho de acceso está listo para ser utilizado.

El botón *Mostrar* (ver figura 3.59) sirve para presentar los registros del derecho de acceso seleccionado. Esta función es útil para revisar los elementos en los cuales el

[90] Para mayor información revisar el ítem 1.4.5.2 “MIBs para VACM”.

derecho de acceso ya está creado, y en base a ello tomar la decisión sobre qué elementos se deben escoger para realizar una nueva exportación o anulación.

Para realizar el proceso de anulación de un derecho de acceso en determinados elementos de red se hace clic en el nombre del grupo VACM (en este caso es el grupo VACM *limitados*) que está ubicado en la tabla de derechos de acceso existentes (ver figura 3.57) y luego se pulsa el botón *Anular.* Este botón indica a la aplicación el derecho de acceso que se desea anular y presenta la página de anulación de derecho de acceso. Ver la figura siguiente.



Anulacion de Derecho de Acceso	
Id.	1
Grupo VACM	limitados
Prefijo Contexto	
Nivel Seguridad	AUTH_NOPRIV
Match Contexto	exact
Vista Lectura	restricted
Vista Escritura	usmUserTable
Vista Notificacion	restricted

Seleccionar elementos:

☑ Red 172.31.1.0			Nombre	Tipo
1	☑	172.31.1.1	R1	router
2	☑	172.31.1.2	S1	switch
3	☑	172.31.1.3	f7killer-nbxp	computer
4	☑	172.31.1.4	fausto-nbub	computer
5	☑	172.31.1.6	fausto-nbfd	computer

Figura 3. 60 Cuerpo de la página de anulación de derecho de acceso.

En la figura anterior se puede apreciar el cuerpo de la página de anulación de derecho de acceso. El cuerpo de la página es similar al de la página de exportación,

con la diferencia de que, en la sección inferior se presenta una tabla con los resultados de la anulación.

Entonces, se seleccionan los elementos de red a los cuales se desea anular (también es posible seleccionar redes enteras haciendo clic en la casilla de la red) y luego se pulsa el botón *Iniciar.* En ese momento la aplicación comienza con el proceso de anulación. La anulación consiste en la eliminación de filas de la tabla *vacmAccessTable.*

Una vez finalizado el proceso se muestra los resultados de la anulación (ver figura siguiente). Los resultados se presentan también en una tabla que está conformada por un título y tres columnas. El título muestra el número de filas devueltas. Las tres columnas corresponden a lo siguiente: número de fila, indicador del resultado de la anulación, y una información de detalle del proceso de anulación.

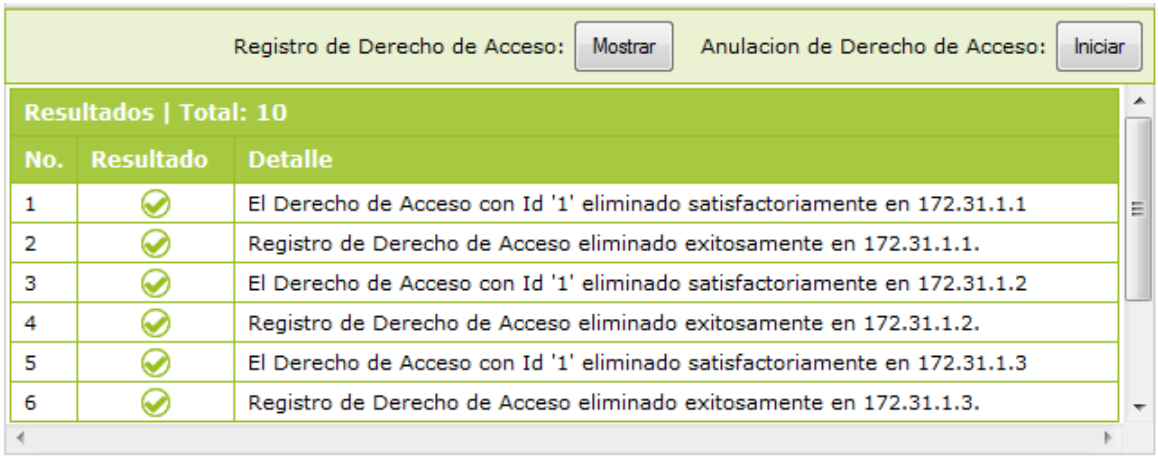


Figura 3. 61 Resultados del proceso de anulación de un derecho de acceso.

Las filas de la tabla de resultados de anulación, a diferencia de las filas de la tabla de resultados de exportación, sólo son informativas; es decir, sólo sirven para presentar mensajes de éxito o fracaso de la operación de anulación y por lo tanto esta información no es almacenada en los repositorios del sistema.

Otro punto importante que se debe destacar del proceso de anulación es que también elimina registros de derechos de acceso. Esto se realiza con el fin de mantener consistencia en los datos que maneja la aplicación.

3.2.11 CAMBIAR PASSWORDS

Previo a este caso de uso, se realiza la comprobación de la creación de la cuenta de usuario (cuyo nombre es *admin2*), de las vistas *restricted* y *usmUserTable*, y del derecho de acceso que asocia al grupo VACM *limitados* al cual este usuario pertenece. Revisar las figuras 3.47, 3.52 y 3.57.

La siguiente figura muestra el inicio de sesión de la nueva cuenta de usuario *admin2.* En el campo de password de ingresa el password del protocolo de autenticación con el que se creó el usuario.

Figura 3. 62 Inicio de sesión del nuevo usuario.

Luego, para realizar la comprobación de la vista *restricted* y del derecho de acceso del grupo VACM *limitados* se utilizan los casos de uso: navegar en grupo MIB y ejecutar operación SNMPv3. Observar las siguientes figuras.

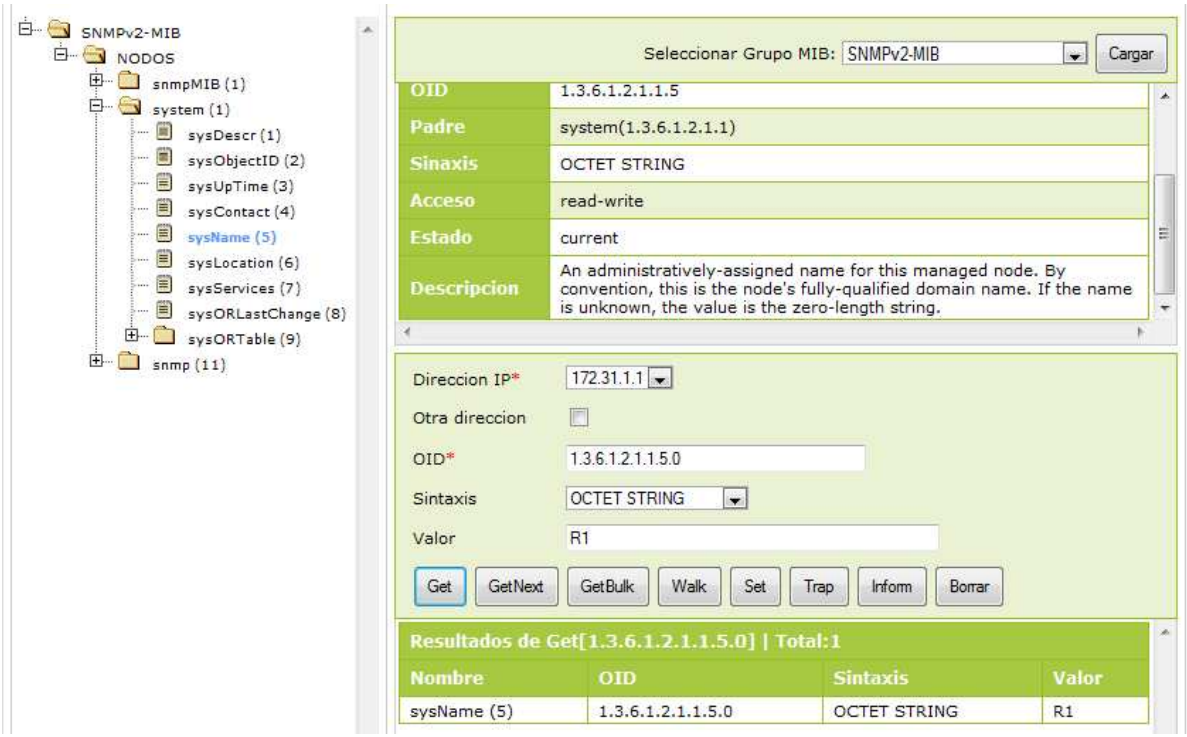


Figura 3. 63 Navegación en el grupo MIB *system* y ejecución de una operación *Get*.

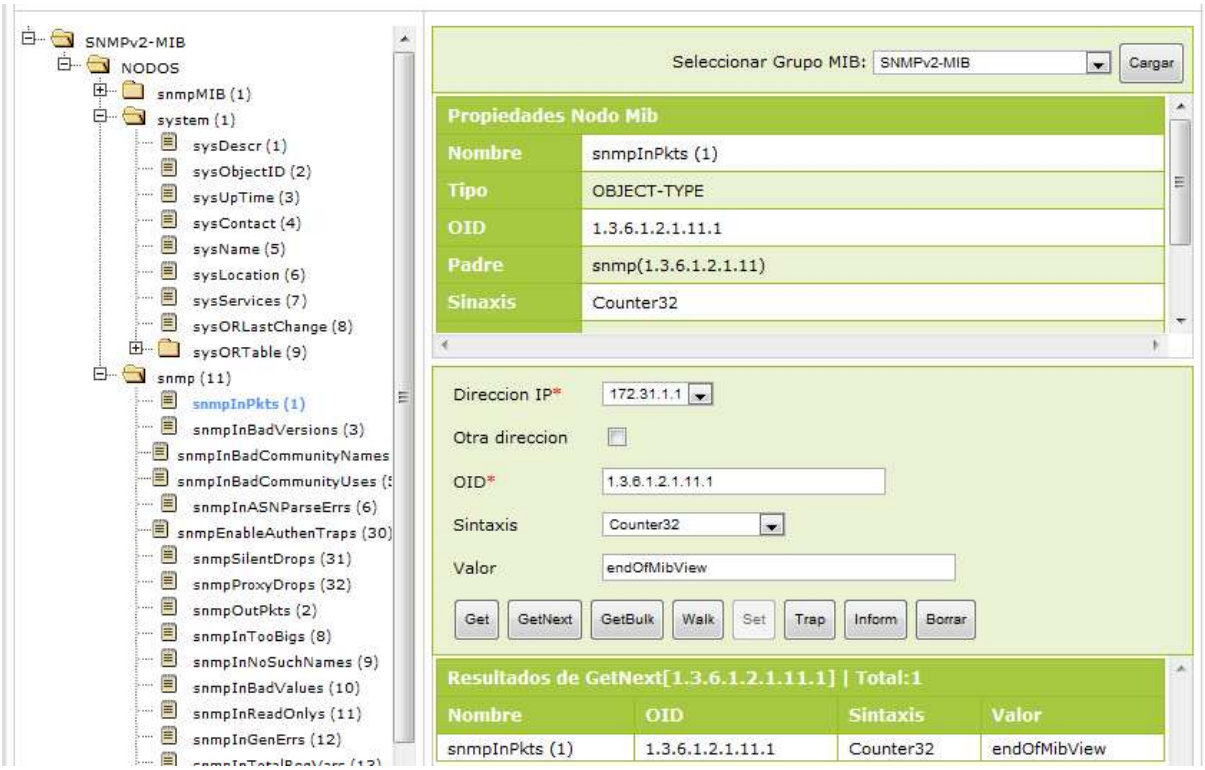


Figura 3. 64 Navegación en el grupo MIB *snmp* y ejecución de una operación *GetNext*.

Como se puede apreciar en la figura 3.63, se realiza una navegación sobre el grupo MIB *system*, se selecciona el nodo MIB *sysName* y se ejecuta la operación *Get*; lo cual da como resultado la devolución del valor de dicho objeto, el valor es *R1.*

Luego, en la figura 3.64 se realiza una navegación sobre el grupo MIB *snmp*, se selecciona el nodo MIB *snmpInPkts* y se ejecuta la operación *GetNext*; lo cual da como resultado un error en la devolución del valor de dicho objeto, el valor es *endOfMibView.* Esto significa que el OID especificado en el formulario está fuera de la vista de lectura para este usuario.

Como conclusión de los resultados anteriores, se comprueba el perfecto funcionamiento de la cuenta de usuario *admin2*, de la vista *restricted* y del derecho de acceso del grupo VACM *limitados.*

En lo que respecta al caso de uso *Cambiar passwords,* éste está conformado por dos páginas: la primera que permite cambiar los passwords localmente en la aplicación, y la segunda permite exportar los cambios de passwords hacia determinados elementos de red. Para acceder a la primera página se lo hace desde el enlace *[nombre de usuario]* > *Cambiar passwords* (ver figura 3.8). La segunda página es accedida desde la primera página.

La figura siguiente muestra la primera página que conforma este caso de uso. Esta página permite cambiar los passwords del usuario en repositorio local de la aplicación. En la figura se puede apreciar que existe un formulario denominado *Cambio de passwords,* el cual está conformado por los siguientes campos:

- **Protocolo de autenticación:** Indica el protocolo de autenticación definido para el usuario.

Figura 3. 65 Cuerpo de la página de cambio de passwords de usuario.

- **Protocolo de privacidad:** Indica el protocolo de privacidad definido para el usuario. Si el valor es ninguno, significa que no puede definir ningún password para privacidad.

- **Password actual:** Aquí se ingresa el password actual, tanto para el caso de autenticación como para el de privacidad.

- **Password nuevo:** Aquí se ingresa el nuevo password, tanto para el caso de autenticación como para el de privacidad.

- **Confirmar password:** aquí se ingresa otra vez el nuevo password, tanto para el caso de autenticación como para el de privacidad. Esto sirve para confirmar que el usuario conoce muy bien los nuevos passwords y así se evita errores de tipiado.

Para cambiar localmente los passwords de usuario se debe ingresar los valores correspondientes en cada uno de los campos, y finalmente se graban los cambios pulsando el botón *Cambiar.*

El botón *Limpiar* sirve para vaciar completamente los campos del formulario, y el botón Cancelar sirve para deshacer los cambios realizados.

Luego, para realizar el proceso de exportación de cambio de passwords de usuario hacia determinados elementos de red se pulsa el botón *Exportar.* Este botón indica a la aplicación qué usuario desea exportar sus cambios de passwords y presenta la página de exportación de cambio de passwords. Ver la figura siguiente.

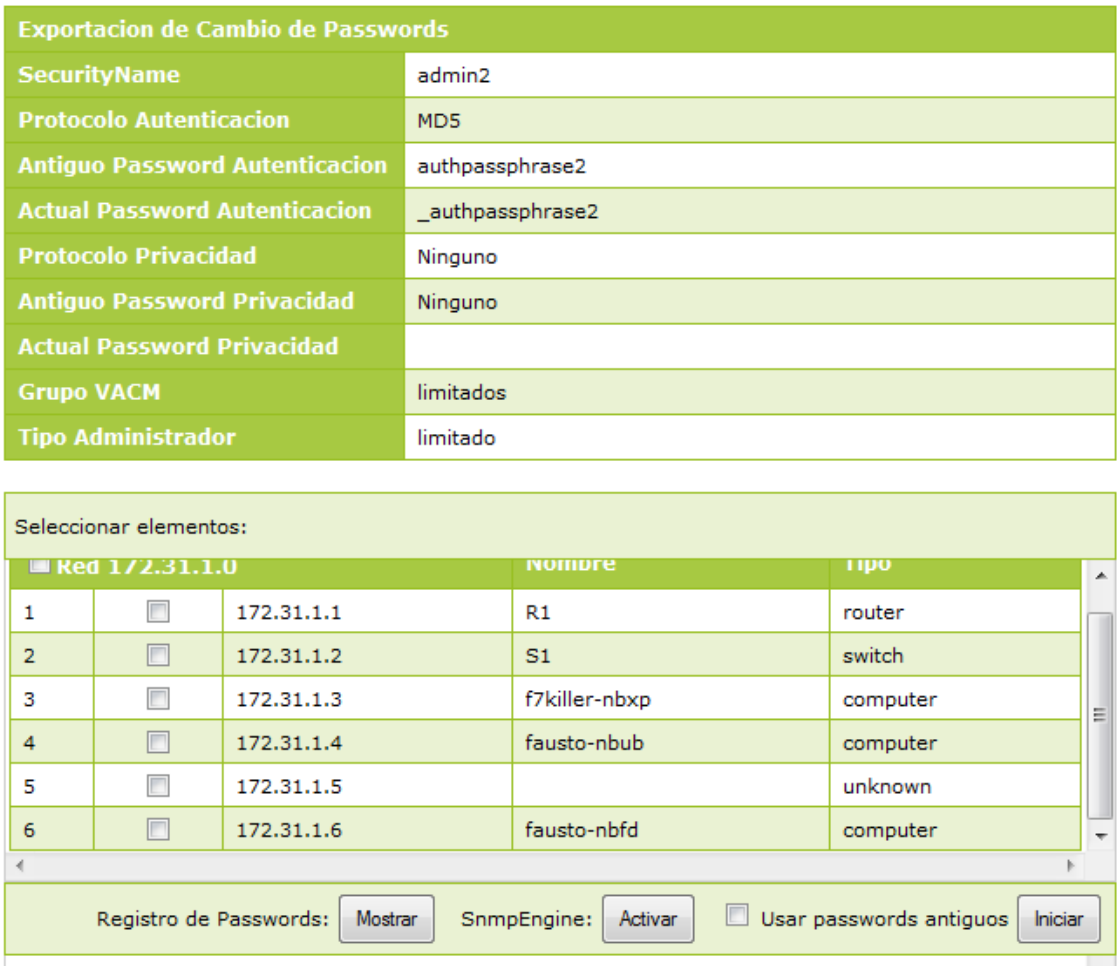


Figura 3. 66 Cuerpo de la página de exportación de cambio de passwords de usuario.

En la figura anterior se puede apreciar el cuerpo de la página de exportación de cambio de passwords de usuario. El cuerpo de la página está divida en tres secciones: la superior que contiene información de las características del usuario (incluidos los antiguos y nuevos passwords), la intermedia que contiene un formulario que permite seleccionar los elementos destinos, y la inferior que contiene un formulario y un espacio dedicado a la presentación del informe de la exportación.

Entonces, se seleccionan los elementos de red a los cuales se desea exportar (también es posible seleccionar redes enteras haciendo clic en la casilla de la red) y luego se pulsa el botón *Iniciar.* En ese momento la aplicación comienza con el proceso de exportación. La exportación está basada en el envío secuencial[91] de mensajes SNMP de recuperación y establecimiento de valores a objetos MIB determinados (específicamente, *usmUserOwnAuthKeyChange* y *usmUserOwnPrivKeyChange*). Estos objetos MIB pertenecen al grupo MIB *snmpUsmMIB*[92].

Figura 3. 67 Informe de los resultados de la exportación de cambio de passwords de usuario.

[91] El envío secuencial de mensajes SNMP está basada en la recomendación del RFC 3414, *User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3),* para cambiar passwords de autenticación y privacidad de un usuario USM. Para mayor información ver las págs.44 – 45, 47 - 48 del RFC 3414.

[92] Para mayor información revisar el ítem 1.4.4.9 “MIBs para USM”.

Una vez finalizado el proceso se muestra el informe de los resultados de la exportación (ver figura anterior). Este informe básicamente es una tabla que está conformada por un título y seis columnas. El título muestra el número de filas devueltas. Cada fila corresponde a un *Registro de Passwords.* Un registro de passwords es conjunto de datos que relaciona la realización de un cambio de passwords en un elemento de red. Un registro de passwords está definido por las seis columnas de la siguiente manera: número de fila (sirve como identificador del registro), nombre del usuario (*securityName*), dirección IP del elemento, indicador del estado del cambio de passwords, fecha de realización del cambio, y un comentario. El indicador de cambio sirve para conocer si los nuevos passwords han sido perfectamente actualizados en cada elemento de red.

El botón *Mostrar* (ver figura 3.67) sirve para presentar los registros de passwords del usuario. Esta función es útil para revisar los elementos en los cuales el cambio de passwords ya fue realizado, y en base a ello tomar la decisión sobre qué elementos se deben escoger para realizar una nueva exportación. También existe una casilla de verificación cuya etiqueta dice *Usar passwords antiguos,* la cual se la utiliza para igualar con las nuevas claves a aquellos los elementos que no hayan sido considerados o que hayan tenido problemas en una exportación anterior; estos elementos de red entienden únicamente passwords antiguos por lo tanto esta función resulta conveniente para actualizarlos con los nuevos passwords.

El botón *Activar* (ver figura 3.67) sirve para activar los nuevos passwords en el motor SNMP (*SnmpEngine*). Sólo de esta manera se le indica al motor que para mandar y recibir mensajes SNMP utilice los nuevos passwords. Si este botón no es activado, entonces se producen errores en la encriptación/desencriptación y autenticación de mensajes SNMP.

3.3 ANÁLISIS DE RESULTADOS

Las pruebas de funcionamiento realizadas anteriormente fueron hechas en un escenario ideal, es decir con todas las precauciones de no cometer errores para que la aplicación funcione adecuadamente. Sin embargo, pueden existir otros escenarios no ideales en los cuales existan errores de diferente tipo en el funcionamiento de la aplicación.

Por lo tanto, en este tema se aborda previamente aspectos como restricciones de uso de la aplicación, mensajes de confirmación y ejecución, y envío y recepción de paquetes, para posteriormente realizar el análisis de los resultados del funcionamiento de la aplicación.

3.3.1 RESTRICCIONES DE USO

Las restricciones de uso se refieren a la realización de una actividad por parte de la aplicación frente a información errónea proporcionada por parte del usuario. Generalmente la actividad es la presentación de un mensaje de advertencia del error cometido por el usuario.

Las restricciones de uso se presentan en los siguientes casos de uso:

- **Iniciar sesión:** Las restricciones se refieren cuando el usuario no ingresa su nombre o su password de autenticación en la página de inicio de sesión. Ambos datos deben ser ingresados.

  La siguiente figura muestra un ejemplo de mensaje de advertencia en el caso de que se incumpla las restricciones en la página de inicio de sesión.

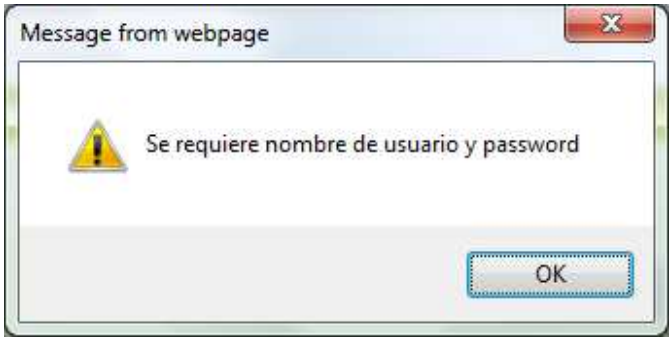


Figura 3. 68 Ejemplo de mensaje de advertencia en la página de inicio de sesión.

- **Configurar parámetros:** Las restricciones existen tanto en la página de configuración de mensajes SNMPv3 como en la página de configuración de servidor.

  En la página de configuración de mensajes (ver figura 3.15), las restricciones se aplica en algunos campos del formulario; por ejemplo valores para los campos: puerto de comandos, puerto de notificaciones, *timeout* (tiempo de espera), reintentos, no repetidores y máximo repeticiones, deben ser números enteros no negativos y no deben ser vacíos puesto que son requeridos. Los valores para los campos *contextEngineID* y *contextName* se los puede dejar vacíos (puesto que la aplicación, en el caso de *contextEngineID*, obtiene el valor automáticamente); sin embargo, en el caso de que se ingresen valores manualmente, el formato[93] del campo *contextEngineID* debe ser una secuencia de octetos hexadecimales separados por dos puntos (:), de longitud mínima 6 octetos y longitud máxima 32 octetos, mientras que el valor del campo *contextName* debe ser una cadena de caracteres.

  La siguiente figura muestra ejemplos de mensajes de advertencia en el caso de que se incumplan las restricciones en la página de configuración de mensajes SNMPv3.

[93] Para mayor información revisar en el ítem 1.4.2.4 "*Convenciones Textuales para Arquitectura SNMP*" el formato del identificador *snmpEngineID.*

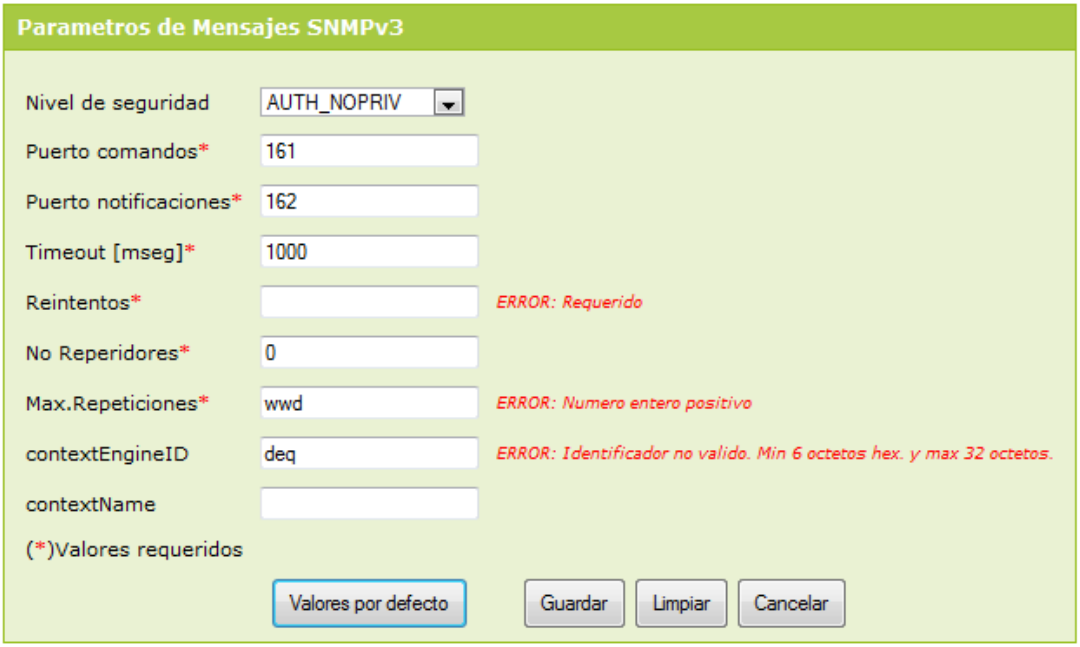


Figura 3. 69 Ejemplos de mensajes de advertencia en la página de configuración de mensajes SNMPv3.

En la página de configuración de servidor (ver figura 3.16), las restricciones se aplica en algunos campos del formulario; por ejemplo valores para los campos: puerto de comandos, puerto de notificaciones, y número de *threads para dispatcher*, deben ser números enteros no negativos y no deben ser vacíos puesto que son requeridos. El formato[94] para el valor para el campo *snmpEngineID* debe ser una secuencia de octetos hexadecimales separados por dos puntos (:), de longitud mínima 6 octetos y longitud máxima 32 octetos, y no se acepta que el valor de este campo sea una cadena vacía.

La siguiente figura muestra ejemplos de mensajes de advertencia en el caso de que se incumplan las restricciones en la página de configuración de servidor.

[94] Para mayor información revisar en el ítem 1.4.2.4 "*Convenciones Textuales para Arquitectura SNMP*" el formato del identificador *snmpEngineID.*

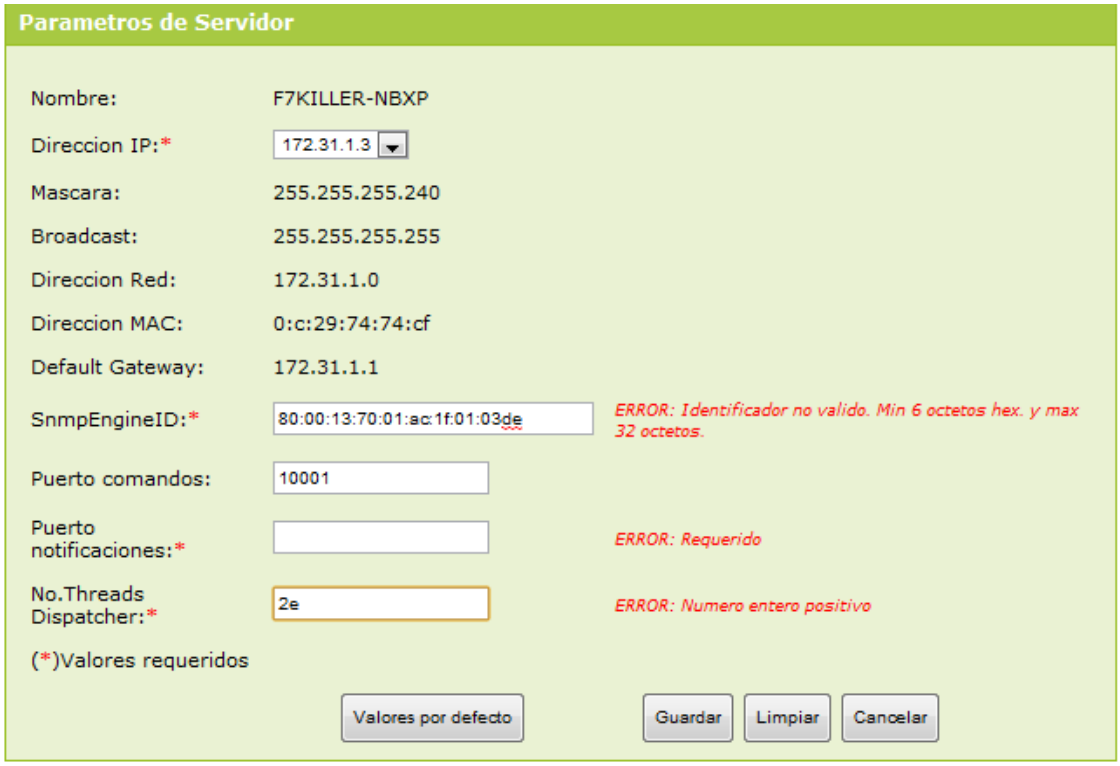


Figura 3. 70 Ejemplos de mensajes de advertencia en la página de configuración de servidor.

- **Descubrir de elementos de red:** Las restricciones existen en la página de descubrimiento de hosts (ver figura 3.17). Las restricciones se relacionan a los siguientes campos: dirección de red, máscara de subred, dirección IP inicial y dirección IP final. El valor del campo máscara de subred debe tener el formato correcto, es decir debe estar formato por cuatro números y separados por puntos; los únicos valores posibles para estos números son: 0, 128, 192, 224, 240, 248, 252, 254 y 255. Los valores para los campos restantes deben tener el formato de una dirección IP, es decir deben estar formados por cuatro números comprendidos entre 0 y 255 y separados por puntos.

  Otra restricción es el rango de direcciones IP, es decir la dirección IP final debe ser mayor que la dirección IP inicial. Por ejemplo, 172.31.1.6 (dirección IP inicial) y 172.31.1.1 (dirección IP final) son valores erróneos, los valores correctos son 172.31.1.1 (dirección IP inicial) y 172.31.1.6 (dirección IP final).

La siguiente figura muestra ejemplos de mensajes de advertencia en el caso de que se incumplan las restricciones en la página de descubrimiento del servidor.

Figura 3. 71 Ejemplos de mensajes de advertencia en la página de descubrimiento de hosts.

- **Ejecutar operación SNMP:** Las restricciones existen en la página de navegación en grupo MIB (ver figura 3.29). Las restricciones se aplican a los campos *dirección IP* y *OID.* El valor para el campo *dirección IP* debe tener el formato adecuado para una dirección IP (ver caso de uso anterior). El valor para el campo *OID* debe ser una secuencia de números enteros, separados cada uno por un punto (.). Por ejemplo, valores como .1.3.6.1 o 1.3.6.1.2 están permitidos, valores como 1..3.6.1 o 1.3.6.1.2. están prohibidos.

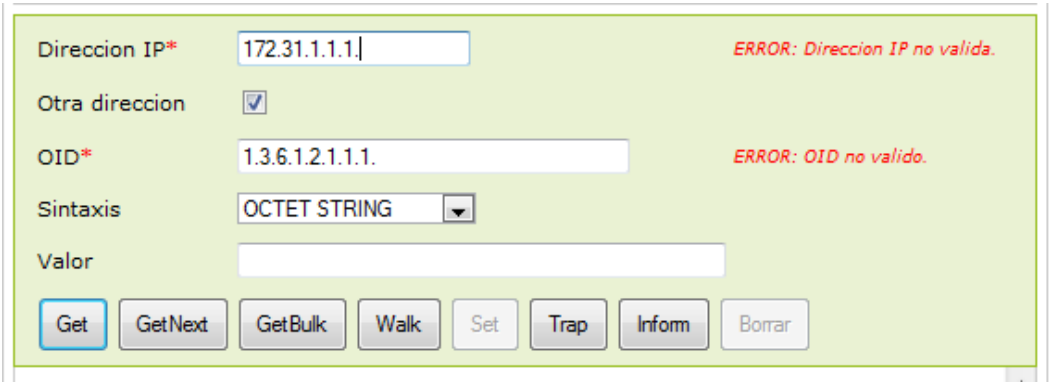


Figura 3. 72 Ejemplos de mensajes de advertencia en la página de navegación en un grupo MIB.

La figura anterior muestra ejemplos de mensajes de advertencia en el caso de que se incumplan las restricciones en la página de navegación en un grupo MIB.

Existe otra restricción que se aplica al campo *valor* al momento de ejecutar una operación *Set*, *Trap* o *Inform.* El valor que se ingresa en este campo debe concordar con el valor del campo *sintaxis* caso contrario se produce un error en el envío del mensaje SNMP.

- **Gestionar usuarios:** Las restricciones existen en la página de gestión de usuarios (ver figura 3.47). Las restricciones se refieren a los campos *securityName* (nombre de seguridad), password de autenticación, password de privacidad, grupo VACM y tipo de administrador. Todos los campos, a excepción de password de privacidad, son requeridos. Además, la longitud de los passwords, tanto para el de autenticación como para el de privacidad, deben tener una longitud mínima de ocho[95] caracteres.

Figura 3. 73 Ejemplos de mensajes de advertencia en la página de gestión de usuarios.

[95] La longitud mínima de ocho caracteres para los passwords de autenticación y privacidad es obtenida de la recomendación del RFC 3414, *User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3).* Para mayor información revisar la pág.73, ítem 11.2 *Defining Users.*

La figura anterior muestra ejemplos de mensajes de advertencia en el caso de que se incumplan las restricciones en la página de gestión de usuarios.

Existe también una restricción en la página de exportación y anulación de un usuario (ver figuras 3.48 y 3.50), la cual exige que para efectuar el proceso de exportación o anulación se requiere seleccionar al menos un elemento de red destino.

- **Gestionar vistas:** Las restricciones existen en la página de gestión de vistas (ver figura 3.52). Las restricciones se refieren a los campos nombre, *subtree* (sub-árbol), máscara de vista, y tipo (*included* o *excluded*). Todos los campos, a excepción de máscara de vista, son requeridos. Además, el formato del valor para el campo *subtree* debe ser del tipo OID.

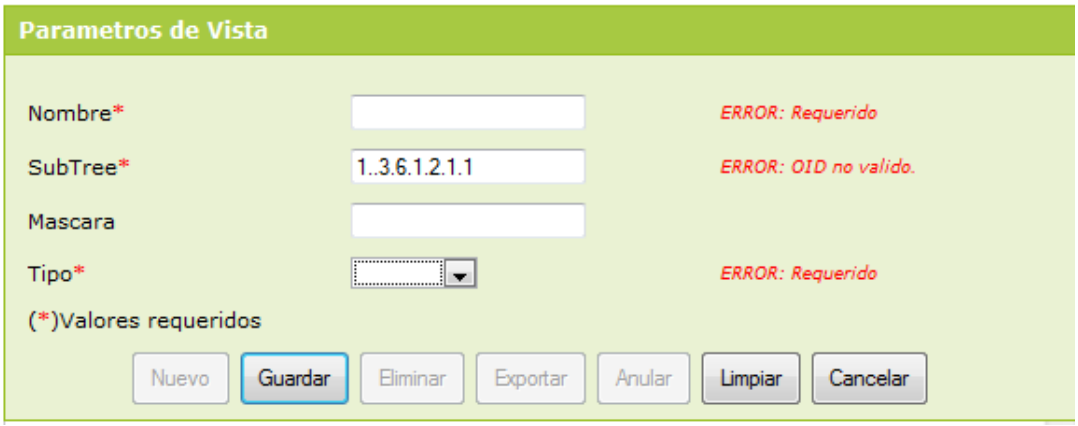


Figura 3. 74 Ejemplos de mensajes de advertencia en la página de gestión de vistas.

La figura anterior muestra ejemplos de mensajes de advertencia en el caso de que se incumplan las restricciones en la página de gestión de vistas.

Existe también una restricción en la página de exportación y anulación de una vista (ver figuras 3.53 y 3.55), la cual exige que para efectuar el proceso de

exportación o anulación se requiere seleccionar al menos un elemento de red destino.

- **Gestionar derechos de acceso:** Las restricciones existen en la página de gestión de derechos de acceso (ver figura 3.57). Las restricciones se refieren a los campos grupo VACM, prefijo de contexto, nivel de seguridad, *match* (emparejador) de contexto, vista de lectura, vista de escritura y vista de notificación. Todos los campos, a excepción del prefijo de contexto, son requeridos.

Figura 3. 75 Ejemplos de mensajes de advertencia en la página de gestión de derechos de acceso.

La figura anterior muestra ejemplos de mensajes de advertencia en el caso de que se incumplan las restricciones en la página de gestión de derechos de acceso.

Existe también una restricción en la página de exportación y anulación de una vista (ver figuras 3.58 y 3.60), la cual exige que para efectuar el proceso de exportación o anulación se requiere seleccionar al menos un elemento de red destino.

- **Cambiar passwords:** Las restricciones existen en la página de cambio de passwords de usuario (ver figura 3.65). Las restricciones se refieren a los campos passwords actual, password nuevo y confirmar password. Todos los campos son requeridos; además, los passwords deben tener longitud mínima de ocho caracteres.

Figura 3. 76 Ejemplos de mensajes de advertencia en la página de cambio de passwords de usuario.

La figura anterior muestra ejemplos de mensajes de advertencia en el caso de que se incumplan las restricciones en la página de cambio de passwords de usuario.

Existe también una restricción en la página de exportación del cambio de passwords de usuario (ver figura 3.66), la cual exige que para efectuar el proceso de exportación se debe seleccionar al menos un elemento de red destino.

Existen también otras clases de restricciones de uso. Por ejemplo, existe una restricción que se refiere a la interfaz de red del servidor, la cual debe estar activa para poder enviar mensajes sobre la red; esto es indispensable en ejecución de la mayoría de casos de uso.

En el caso de que el usuario que ha iniciado sesión sea del tipo administrador limitado, existe la restricción de que no puede ingresar a las páginas de gestión de usuarios, vistas y derechos de acceso. En caso de que se incumpla esta restricción, se presenta un mensaje de advertencia. Ver la figura siguiente.

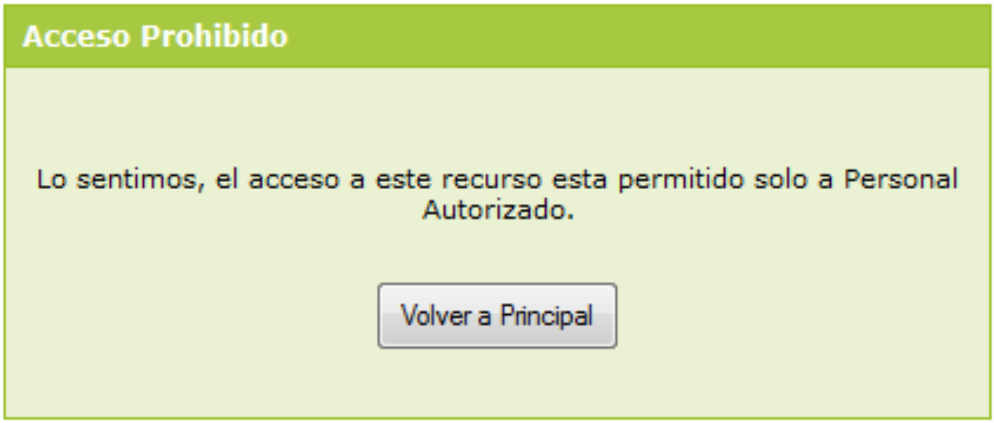


Figura 3. 77 Mensaje de advertencia por intento de ingreso a páginas de uso exclusivo del administrador general.

Existe también la restricción de que no se debe alterar el contenido de los archivos XML de almacenamiento de datos de la aplicación, tampoco se debe alterar la organización de directorios de dichos archivos. Esto puede ocasionar problemas en el funcionamiento de la aplicación.

Por ejemplo, si se altera un archivo XML con algún editor XML, se produce una excepción en el sistema y ésta es mostrada al usuario mediante un mensaje ubicado en una página de excepciones. Ver la figura siguiente.

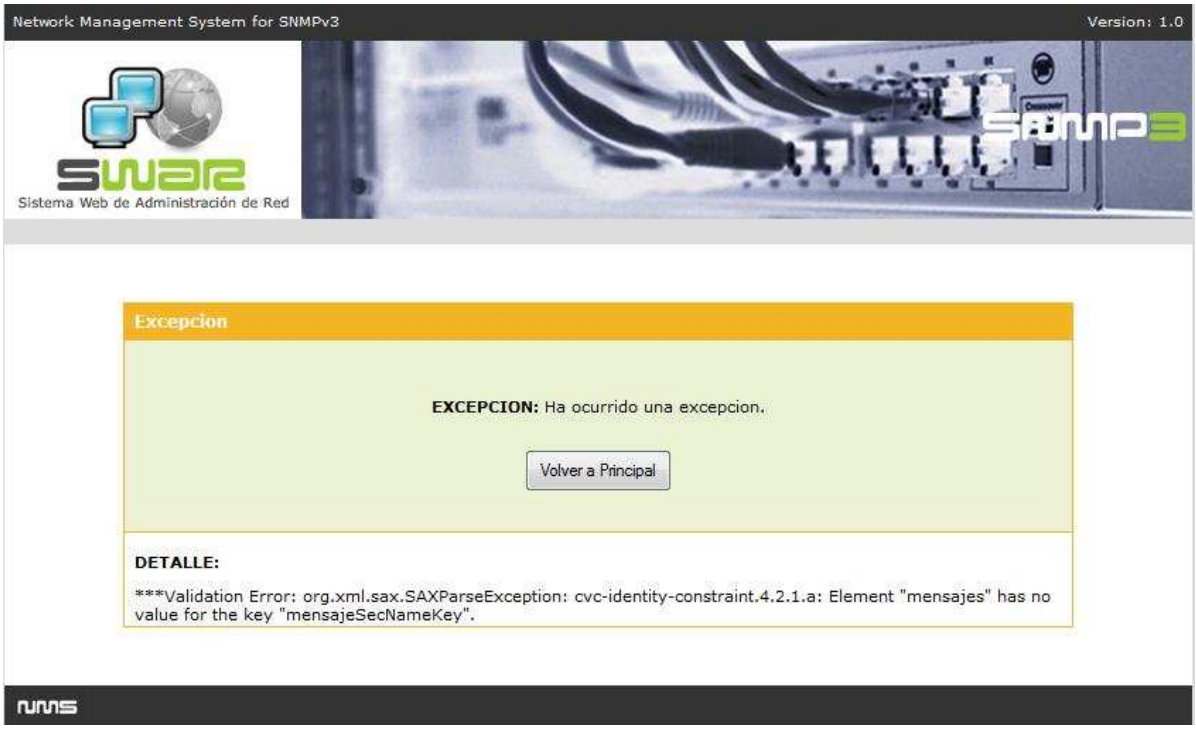


Figura 3. 78 Ejemplo de mensaje de excepción por alteración indebida del archivo XML que guarda información de parámetros de mensajes SNMPv3 .

3.3.2 MENSAJES DE CONFIRMACIÓN Y EJECUCIÓN

Los mensajes de ejecución muestran información concisa acerca del éxito o fracaso de alguna operación realizada. Los mensajes de confirmación son ventanas que solicitan ratificación del usuario para realizar una operación. Generalmente los mensajes de confirmación se presentan antes de los mensajes de ejecución.

Los mensajes de confirmación se presentan generalmente en los casos de uso en donde se manipula información del sistema (archivos XML), y en aquellos donde se maneja envío y recepción de mensajes sobre la red.

La siguiente figura muestra un ejemplo de mensaje de confirmación, el cual se trata de la ratificación para eliminar un elemento de la lista de elementos registrados. Los elementos registrados están almacenados en un archivo XML apropiado.

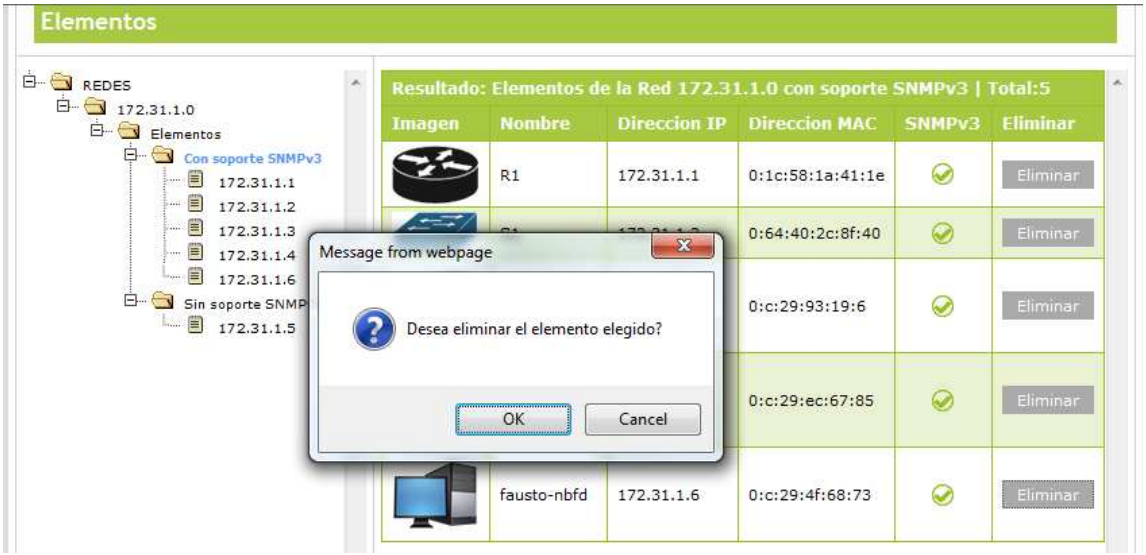


Figura 3. 79 Ejemplo de mensaje de confirmación.

Figura 3. 80 Ejemplo de mensaje de ejecución.

Los mensajes de ejecución también se presentan en los casos de uso donde se manipula información de los archivos XML, y donde se maneja envío y recepción de mensajes sobre la red. Si los mensajes que se envían por la red son del tipo SNMP, entonces el contenido de los mensajes de ejecución puede ser cualquiera de los tipos de errores definidos para SNMP (revisar tabla 1.6).

La figura anterior muestra un ejemplo de mensaje de ejecución después de realizarse una operación de consulta de información de sistema de un elemento de red. El mensaje indica que el tiempo de espera ha vencido y que la operación *Get* ha fallado. Las razones de que haya sucedido ésto pueden ser las siguientes:

- El elemento de red destino está inactivo.
- El servicio SNMPv3 del elemento de red no está ejecutándose.
- Existe incongruencia en la configuración de los parámetros del mensaje SNMPv3 con la configuración del agente del elemento destino.
- El elemento destino está ejecutando muchos procesos y demora en responder a la solicitud del mensaje SNMPv3.

La figura siguiente muestra otro ejemplo de mensaje de ejecución después de realizarse una operación *GetNext* desde la página de navegación en grupo MIB. El mensaje indica que se ha producido un error, el estado de error es *Authorization error (16)* (error de autorización cuyo identificador es el número 16) y el índice de error es 0. La razón más probable de que haya sucedido esto puede ser que el parámetro nivel de seguridad configurado para el mensaje SNMPv3 es inadecuado, es decir se requiere de un nivel más seguro para autorizar el acceso al objeto MIB (al menos AUTH_NOPRIV, por ejemplo).

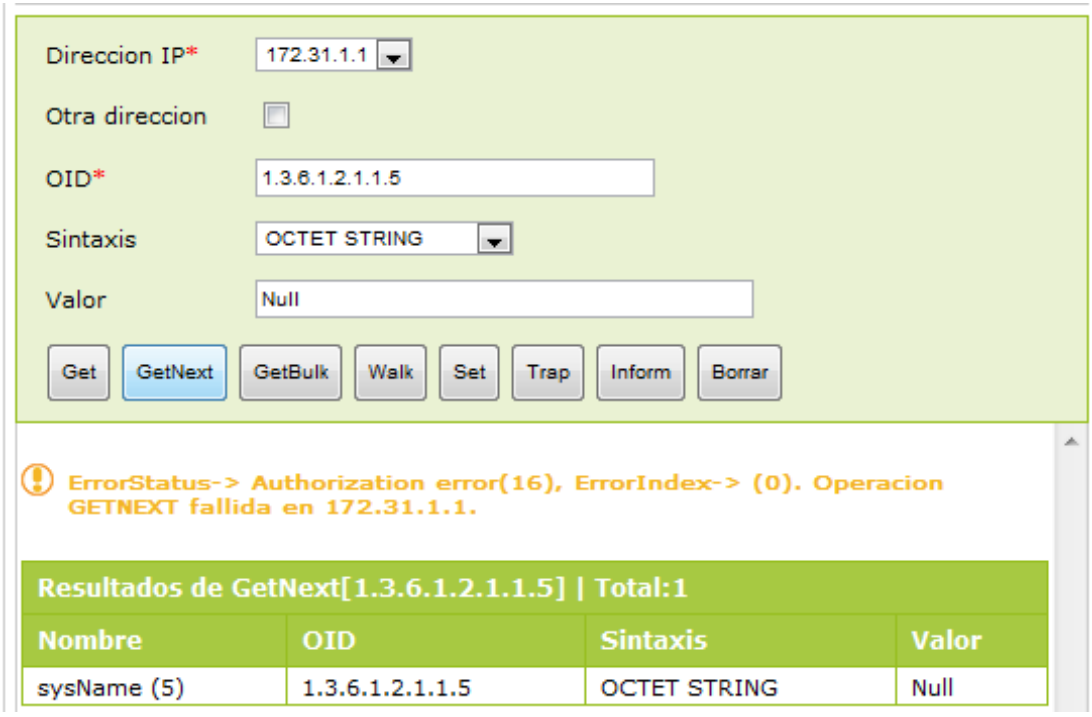


Figura 3. 81 Ejemplo de mensaje de ejecución luego de una operación *GetNext*.

En la figura anterior se puede apreciar que en la tabla de resultados el campo valor está asignado con *NULL;* este valor aparece cuando se produce un error del tipo SNMP (en este caso se produjo un error de autorización).

Existen también otro tipo de mensajes de ejecución, los cuales se presentan dentro del contenido de los informes de exportación y anulación de: una cuenta de usuario, una vista MIB, un derecho de acceso o un cambio de password de usuario. Generalmente la sección comentario o detalle de estos informes conforma el mensaje de ejecución.

Por ejemplo, la siguiente figura corresponde al informe de resultados de anulación de la cuenta de usuario *admin2.* En la figura se puede apreciar que en los registros 3 y 6 existen inconvenientes. La sección detalle explica posibles las razones de la ocurrencia del error. En este caso la explicación hace referencia a que no existen

registros de passwords de este usuario en dicho elemento de red para eliminarlos, y de esa manera mantener consistencia en los datos del sistema. Esto se debe a que el usuario *admin2*, antes de ser anulado, nunca ha exportado un cambio de passwords; únicamente cuando se realiza esta exportación, se crea un nuevo de registro de passwords.

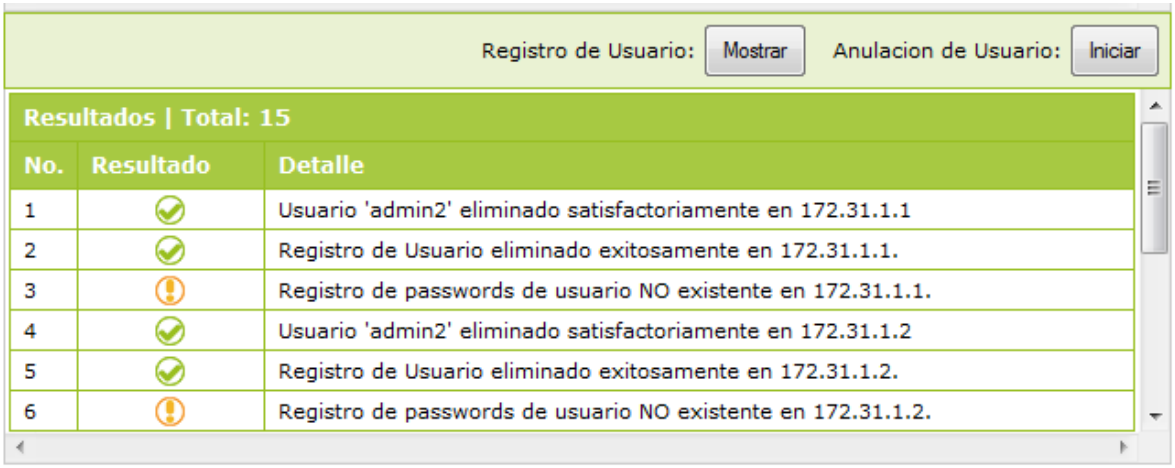


Figura 3. 82 Ejemplo de mensajes de ejecución en informe de anulación de una cuenta de usuario.

Figura 3. 83 Ejemplo de mensaje de ejecución en la página de gestión de MIBs.

Específicamente en el caso de uso de gestionar MIBs, existe un mensaje de ejecución especial que aparece al momento de subir y procesar el nuevo archivo MIB. La siguiente anterior muestra este tipo de mensaje. La posible causa de este tipo de error es porque el archivo que se subió tiene formato inadecuado o contiene errores de sintaxis.

3.3.3 ENVÍO Y RECEPCIÓN DE PAQUETES

Para verificar cómo se comunica la aplicación con los diferentes elementos de red se utiliza una herramienta de captura y análisis de paquetes. La herramienta utilizada para este propósito es *Wireshark* versión 1.4.1. Esta herramienta fue instalada en la máquina cuya dirección IP es 172.31.1.6 y cuyo sistema operativo es Linux de la distribución Fedora. Ver tabla 3.1.

Para simplificar la explicación de la comunicación, se capturarán los paquetes que se transmiten únicamente en los casos de uso *Ejecutar operación SNMPv3* y *Cambiar Passwords*. La razón de esto es que en dichos casos de uso se transmiten mensajes SNMPv3 con pocos vínculos variables lo cual facilita la interpretación de la captura de mensajes, y además se pueden apreciar todos los tipos de mensajes SNMP.

3.3.3.1 Ejecutar operación SNMPv3

La comunicación que se genera al ejecutar cualquier operación de solicitud SNMPv3 (a excepción del comando *Trap*) se desarrolla en cuatro pasos en los cuales participan: el Cliente (el navegador web desde donde se maneja la aplicación), el Servidor (donde está albergada la aplicación) y el elemento de red que se va a gestionar. De manera general, los pasos son los siguientes:

1) El Cliente envía al Servidor la petición de información de gestión de un elemento de red determinado. Esta petición es transportada en forma segura mediante el protocolo HTTPS.

2) El Servidor recibe los mensajes, los procesa e interpreta su contenido. Luego, el servidor forma el tipo de mensaje SNMPv3 de solicitud requerido y lo envía hacia el elemento de red destino.

3) El elemento de red destino recibe el mensaje, lo procesa, forma un mensaje SNMPv3 de respuesta y lo envía hacia el Servidor.

4) El Servidor recibe el mensaje de respuesta, procesa su contenido, formula una respuesta que sea legible para el usuario, y utilizando el protocolo HTTPS la envía hacia al Cliente.

Cabe indicar que previo a estos pasos, cuando recién se ingresa a la aplicación y se muestra la pantalla de bienvenida, se produce una comunicación inicial entre el Servidor y el Cliente. El objetivo de esta comunicación es negociar los parámetros relacionados al protocolo HTTPS. Una vez que el Servidor y el Cliente hayan llegado a un acuerdo, pueden intercambiar datos cifrados.

La siguiente figura muestra la secuencia de los pasos anteriormente citados.

Figura 3. 84 Secuencia de ejecución de un mensaje SNMPv3.

En el caso de que se envíe un mensaje *Trap,* sólo se cumplen los pasos 1 y 2. Esto se debe a la característica que tiene este mensaje Trap de no generar respuesta de retorno.

La siguiente figura muestra un ejemplo de captura de la secuencia de la ejecución de un mensaje SNMPv3. Los tres paquetes capturados corresponden al primer paso de la secuencia. Los siguientes seis paquetes (coloreados son celeste) corresponden a los pasos dos y tres. Finalmente, los paquetes restantes corresponden al paso cuatro. En la figura también se puede apreciar que hay paquetes cuyo protocolo es TLSv1 (*Transport Layer Security version 1 – Seguridad en la Capa Transporte versión 1*) en estos paquetes es en donde se transporta la información cifrada de HTML. Este protocolo forma parte del protocolo HTTPS. Para mayor información revisar el ítem 1.5.6.1.

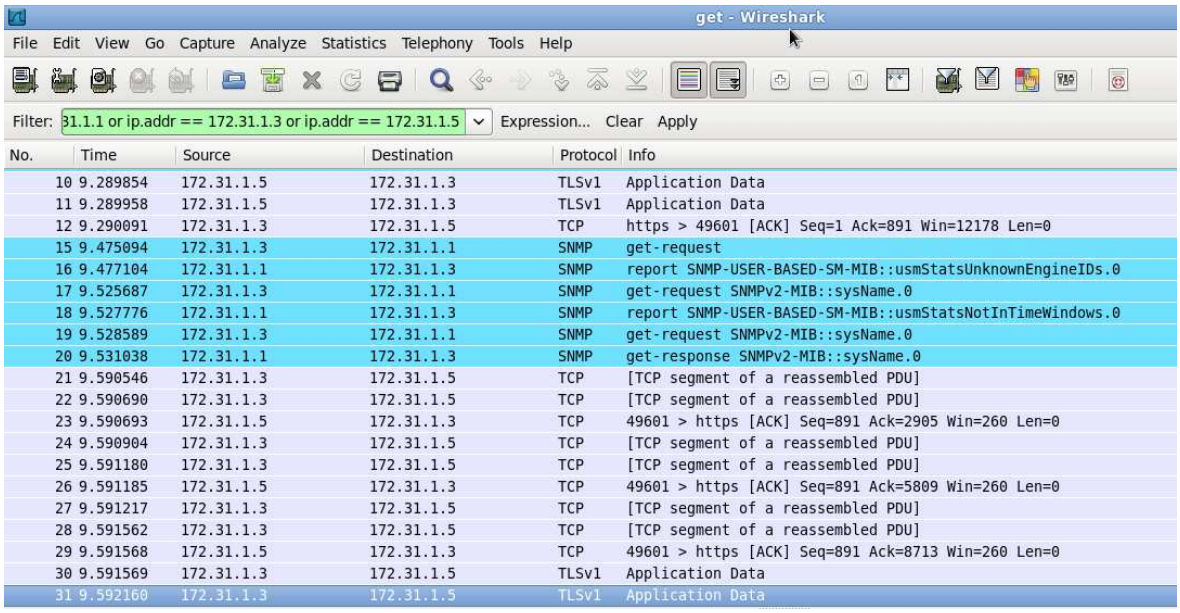


Figura 3. 85 Paquetes capturados en la ejecución de una operación SNMPv3.

A continuación se muestra la comprobación de los mensajes SNMPv3 que se han capturado mediante la herramienta Wireshark. Para poder interpretar los contenidos de cada mensaje, se ha configurado el nivel de seguridad a AUTH_NOPRIV (autenticación pero sin privacidad) desde la página de configuración de parámetros de mensaje SNMPv3.

3.3.3.1.1 GetRequest

Como se puede ver en la siguiente figura se han capturado seis mensajes SNMP al ejecutar la operación *GetRequest*. De estos paquetes, los cuatro primeros tienen relación con el mecanismo de descubrimiento[96] USM. Este mecanismo de descubrimiento se lo realiza cada vez que un motor no autorizado (en este caso, aquel que pertenece al servidor[97] de dirección IP 172.31.1.3) desea comunicarse por primera vez con un motor autorizado (en este caso, aquel que pertenece al agente del router cuya dirección IP es 172.31.1.1).

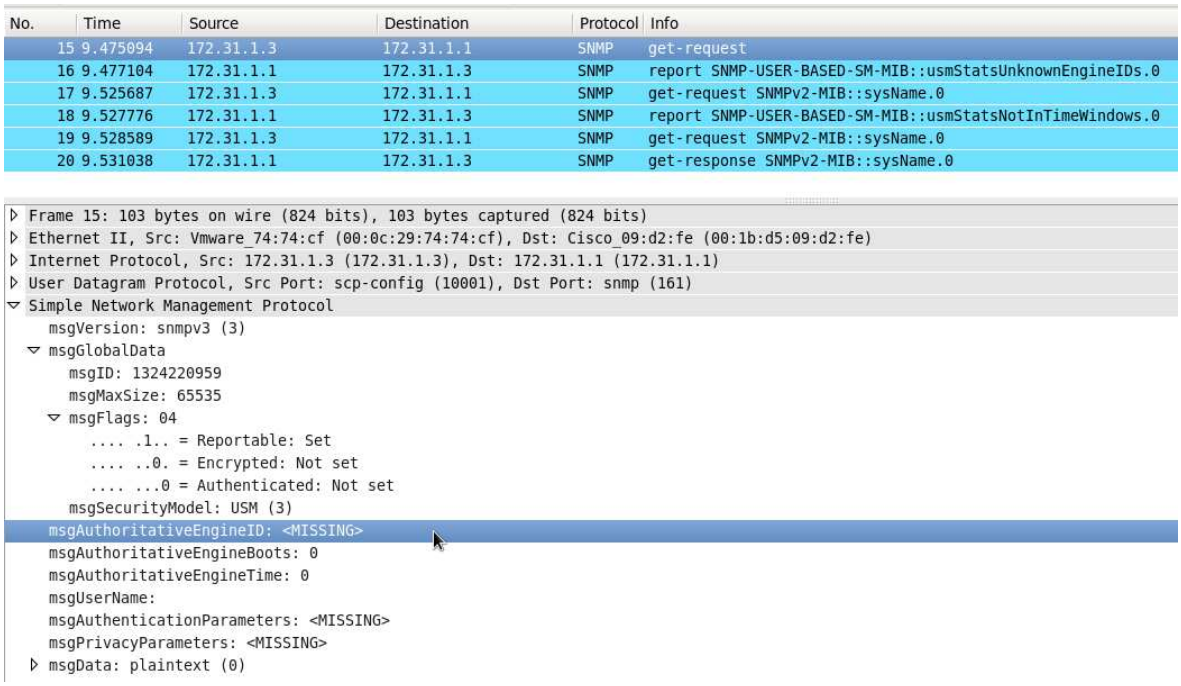


Figura 3. 86 Detalle del primer mensaje de solicitud para descubrimiento USM.

Como se puede apreciar en la figura anterior, el primer mensaje (resaltado con azul), es un mensaje de solicitud para consultar al motor autorizado su valor de *snmpEngineID* (observar que el campo *msgAuthoritativeEngineID* está en <MISSING>, es decir vacío).

---

[96] Para mayor información revisar el ítem 1.4.4.6.

[97] Revisar el diagrama de la topología de red mostrado en la figura 3.9.

En la siguiente figura se muestra el detalle del segundo mensaje (resaltado con azul) para descubrimiento USM. Este mensaje es del tipo *Report*[98]. Este mensaje incluye el valor de *snmpEngineID* en el campo *msgAuthoritativeEngineID.* Este valor es utilizado en adelante por el motor no autorizado para enviar mensajes de solicitud hacia dicho motor autorizado.

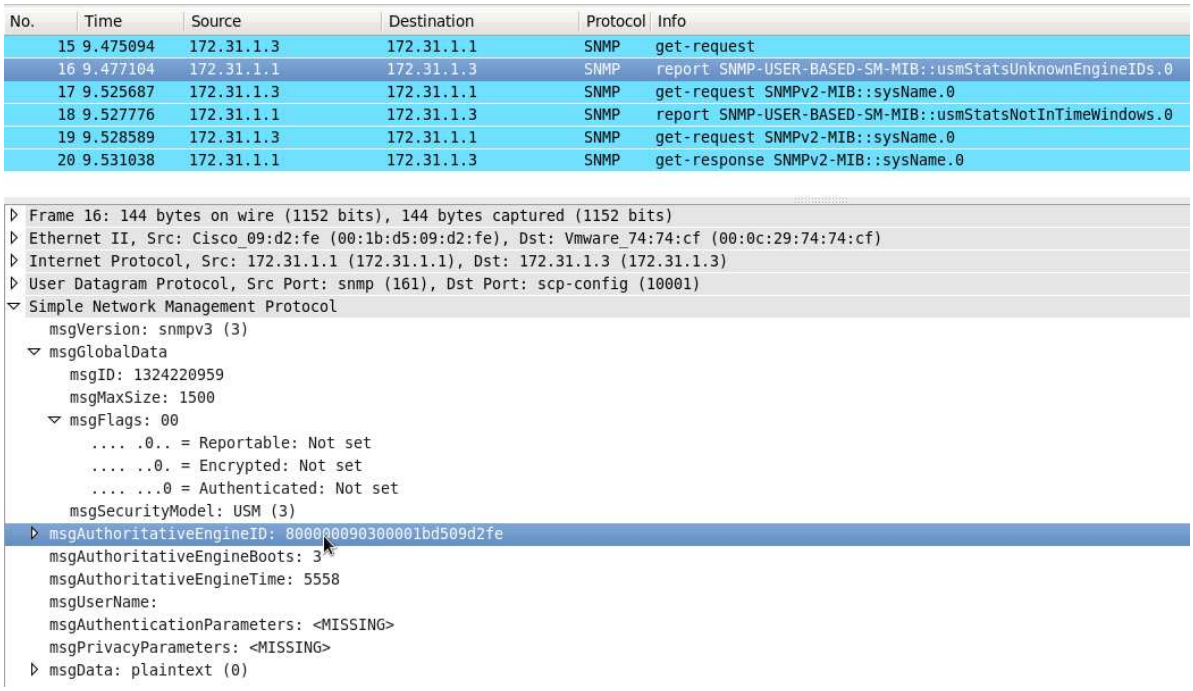


Figura 3. 87 Detalle del segundo mensaje de reporte para descubrimiento USM.

En la siguiente figura se muestra el detalle del tercer mensaje (resaltado con azul) para descubrimiento USM. En este mensaje se consulta los valores *msgAuthoritativeEngineBoots* y *msgAuthoritativeEngineTime (*están seteados en cero*)*, los cuales sirven para que el motor no autorizado pueda sincronizarse con el motor autorizado. Luego de dicha figura, se muestra otra en la cual se presenta el detalle del cuarto mensaje (resultado con azul) para descubrimiento USM. Este mensaje también es del tipo *Report*. Este mensaje incluye los valores de *snmpEngineBoots y snmpEngineTime* en los campos *msgAuthoritativeEngineBoots* y

[98] Para mayor información revisar el ítem 1.3.6.1.8.

msgAuthoritativeEngineTime, respectivamente. Luego, de recibido el mensaje *report,* el motor no autorizado se sincroniza con estos valores y está apto para enviar el mensaje *GetRequest* con el OID requerido.

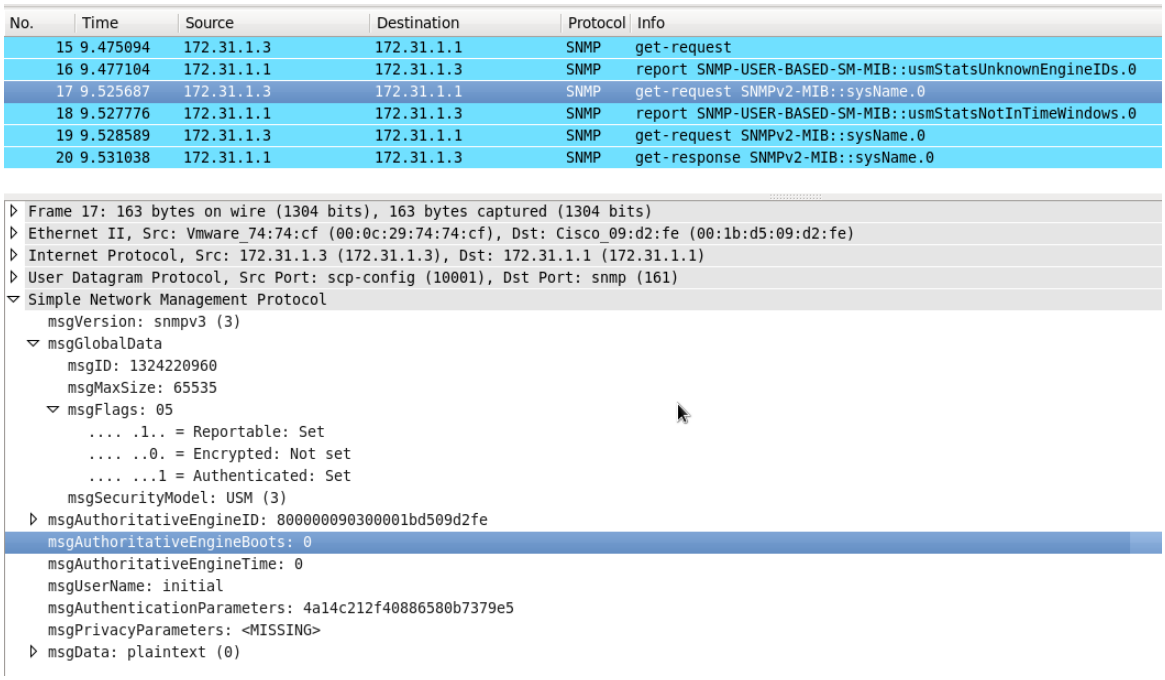


Figura 3. 88 Detalle del tercer mensaje de solicitud para descubrimiento USM.

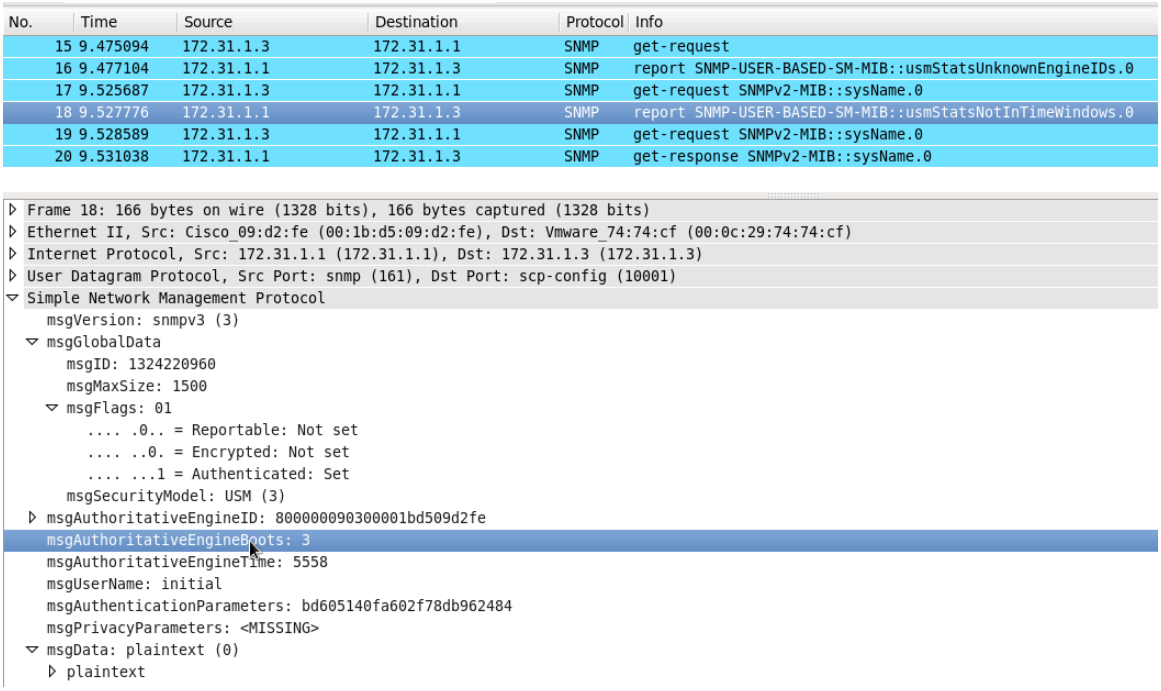
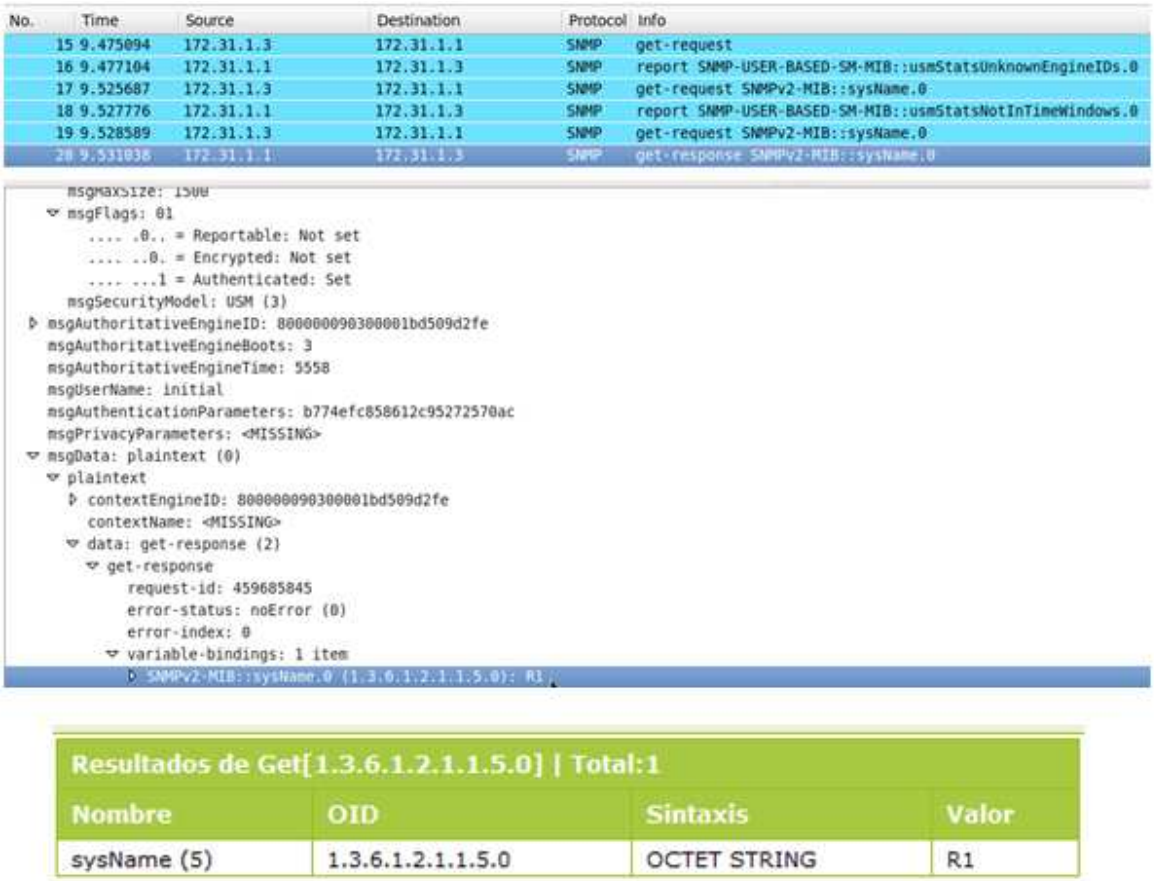


Figura 3. 89 Detalle del cuarto mensaje de reporte para descubrimiento USM.

Finalmente, en la siguiente figura se muestra el detalle del mensaje de respuesta a una petición *GetRequest.* También se muestra el resultado mostrado por la aplicación, esto permite comprobar la coincidencia de ambos resultados.



Resultados de Get[1.3.6.1.2.1.1.5.0] | Total:1

Nombre	OID	Sintaxis	Valor
sysName (5)	1.3.6.1.2.1.1.5.0	OCTET STRING	R1

Figura 3. 90 Comprobación del resultado de una operación *GetRequest*.

Con el fin de demostrar que SNMPv3 ofrece protección contra la amenaza de Revelación[99], se ha realizado otra prueba utilizando el mensaje *GetRequest* pero configurando el nivel de seguridad a AUTH_PRIV (autenticación y privacidad) desde la página de configuración de parámetros de mensaje SNMPv3. Esto se demuestra en la siguiente figura.

[99] Para mayor información revisar el ítem 1.4.1.

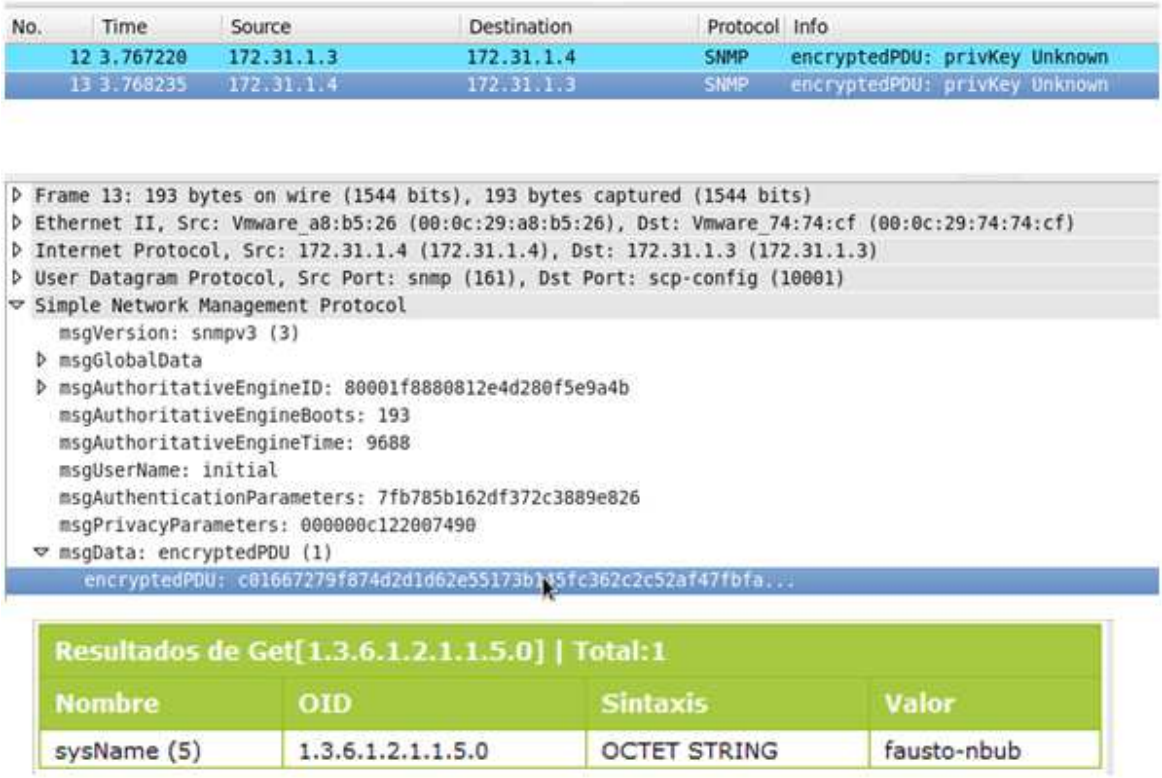


Figura 3. 91 Comprobación de la protección contra revelación.

Como se puede ver en la figura anterior, el valor en la tabla de resultados es legible para el usuario; pero el valor en la captura de paquetes (lo que está resaltado con azul), no. Esto representa protección contra revelación, es decir cuando un mensaje en tránsito es capturado, los valores del mensaje no pueden ser interpretados y aprendidos debido a que éstos están cifrados.

3.3.3.1.2 GetNextRequest

En la siguiente figura se muestra la comprobación de resultados al enviarse un mensaje *GetNextRequest.* Los que participan en esta comunicación son los mismos que participaron en la anterior comprobación, es decir el servidor y el router. Como se puede apreciar en la figura, ya no existen los mensajes de descubrimiento USM; por lo tanto, el motor no autorizado se sincroniza ahora con los valores que vienen en el mensaje de respuesta que se retorna por cada solicitud enviada.

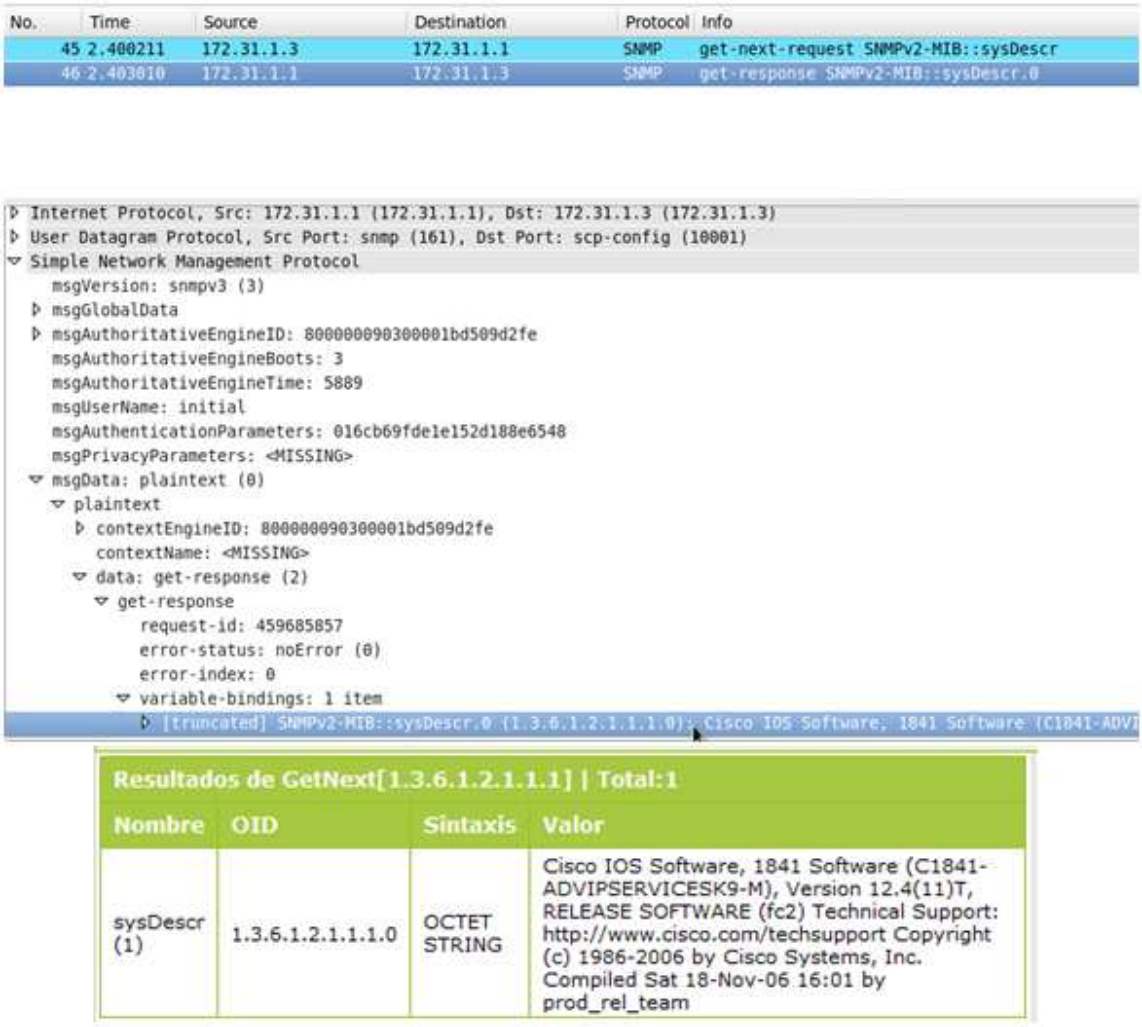


Figura 3. 92 Comprobación del resultado de una operación *GetNextRequest*.

3.3.3.1.3 GetBulkRequest

En la siguiente figura se muestra la comprobación de resultados al enviarse un mensaje *GetBulkRequest.* Los que participan en esta comunicación son el servidor y equipo que tiene la dirección IP 172.31.1.6 (el analizador de paquetes). Como se puede apreciar en la figura, en este caso si se produce un mecanismo de descubrimiento USM pues es la primera vez que el servidor se comunica con este elemento.

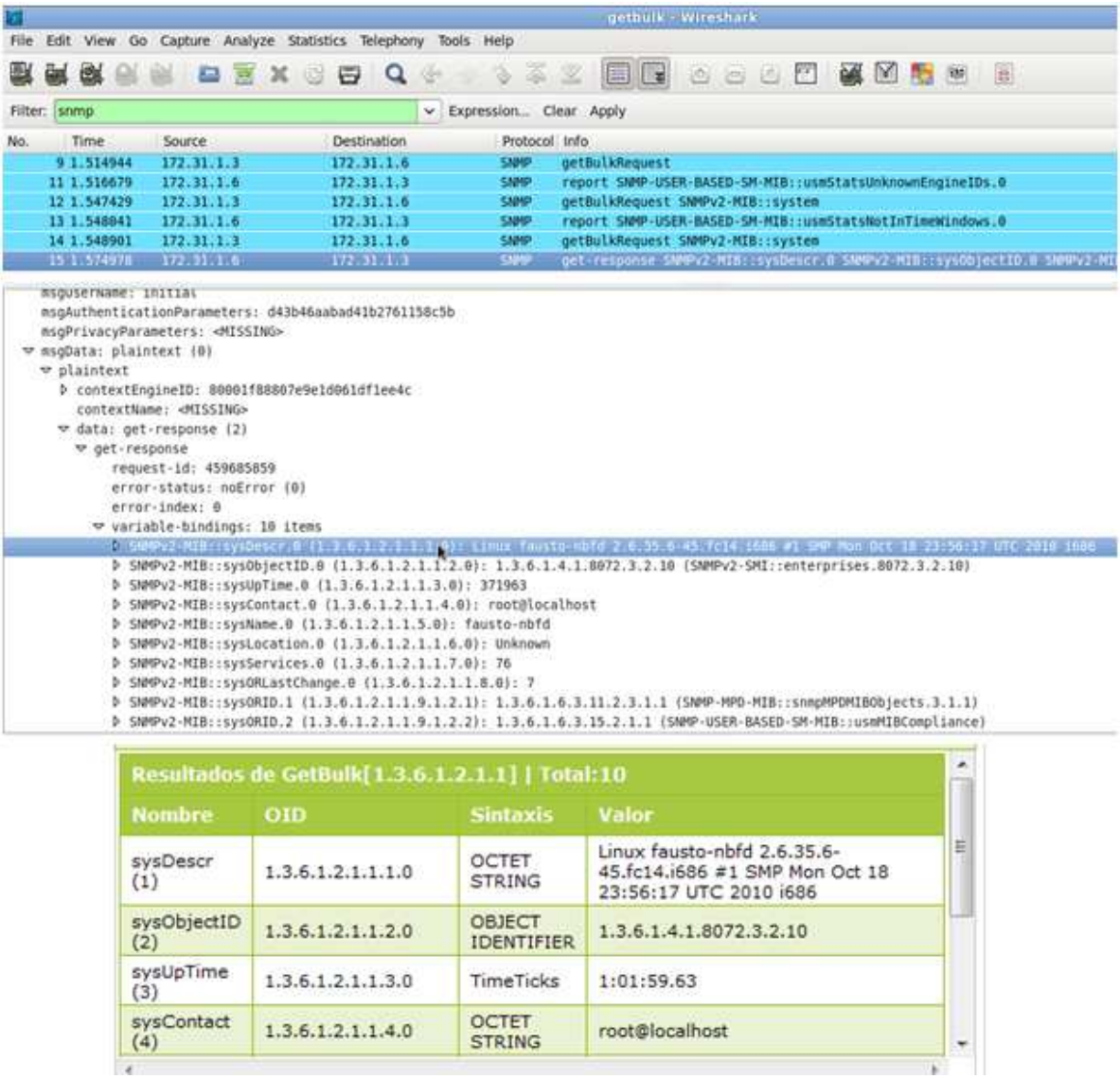


Figura 3. 93 Comprobación del resultado de una operación *GetBulkRequest*.

3.3.3.1.4 Walk

En la siguiente figura se muestra la comprobación de resultados al ejecutarse una operación *Walk*. Este tipo de operación utiliza al mensaje *GetNextRequest* para recorrer todas las instancias de objetos a partir de un identificador dado referente a un grupo MIB de nivel superior (en este caso es el grupo MIB *system*). Los que

participan en esta comunicación son el servidor y el equipo que tiene la dirección IP 172.31.1.4 (Servidor Linux).

No.	Time	Source	Destination	Protocol	Info
25	5.913364	172.31.1.3	172.31.1.4	SNMP	get-next-request SNMPv2-MIB::system
26	5.953189	172.31.1.4	172.31.1.3	SNMP	get-response SNMPv2-MIB::sysDescr.0
27	5.955115	172.31.1.3	172.31.1.4	SNMP	get-next-request SNMPv2-MIB::sysDescr.0
28	5.955730	172.31.1.4	172.31.1.3	SNMP	get-response SNMPv2-MIB::sysObjectID.0
29	5.956411	172.31.1.3	172.31.1.4	SNMP	get-next-request SNMPv2-MIB::sysObjectID.0
30	5.956814	172.31.1.4	172.31.1.3	SNMP	get-response SNMPv2-MIB::sysUpTime.0
31	5.957429	172.31.1.3	172.31.1.4	SNMP	get-next-request SNMPv2-MIB::sysUpTime.0
32	5.957804	172.31.1.4	172.31.1.3	SNMP	get-response SNMPv2-MIB::sysContact.0
33	5.958416	172.31.1.3	172.31.1.4	SNMP	get-next-request SNMPv2-MIB::sysContact.0
34	5.958771	172.31.1.4	172.31.1.3	SNMP	get-response SNMPv2-MIB::sysName.0
35	5.959368	172.31.1.3	172.31.1.4	SNMP	get-next-request SNMPv2-MIB::sysName.0
36	5.959746	172.31.1.4	172.31.1.3	SNMP	get-response SNMPv2-MIB::sysLocation.0
37	5.960923	172.31.1.3	172.31.1.4	SNMP	get-next-request SNMPv2-MIB::sysLocation.0
38	5.961348	172.31.1.4	172.31.1.3	SNMP	get-response SNMPv2-MIB::sysServices.0
39	5.962022	172.31.1.3	172.31.1.4	SNMP	get-next-request SNMPv2-MIB::sysServices.0
40	5.962497	172.31.1.4	172.31.1.3	SNMP	get-response SNMPv2-MIB::sysORLastChange.0
41	5.963743	172.31.1.3	172.31.1.4	SNMP	get-next-request SNMPv2-MIB::sysORLastChange.0
42	5.964333	172.31.1.4	172.31.1.3	SNMP	get-response SNMPv2-MIB::sysORID.1
43	5.964954	172.31.1.3	172.31.1.4	SNMP	get-next-request SNMPv2-MIB::sysORID.1
44	5.965355	172.31.1.4	172.31.1.3	SNMP	get-response SNMPv2-MIB::sysORID.2

Resultados de Walk[1.3.6.1.2.1.1] | Total:32

Nombre	OID	Sintaxis	Valor
sysDescr (1)	1.3.6.1.2.1.1.1.0	OCTET STRING	Linux fausto-nbub 2.6.32-25-generic #45-Ubuntu SMP Sat Oct 16 19:48:22 UTC 2010 i686
sysObjectID (2)	1.3.6.1.2.1.1.2.0	OBJECT IDENTIFIER	1.3.6.1.4.1.8072.3.2.10
sysUpTime (3)	1.3.6.1.2.1.1.3.0	TimeTicks	1:38:32.94
sysContact (4)	1.3.6.1.2.1.1.4.0	OCTET STRING	root

Figura 3. 94 Comprobación del resultado de una operación *Walk*.

3.3.3.1.5 SetRequest

En la siguiente figura se muestra la comprobación de resultados al enviarse un mensaje *SetRequest.* Los que participan en esta comunicación son el servidor y el

switch que tiene la dirección IP 172.31.1.2. Como se puede apreciar en la figura, a más del mensaje *Set* también existe un mensaje GetRequest y su correspondiente mensaje de respuesta. Estos se utilizan para informar al sistema de algún cambio en el agente que exija actualizar los datos en el repositorio local; por ejemplo si mediante el mensaje *Set* se cambió el nombre de host del elemento de red.

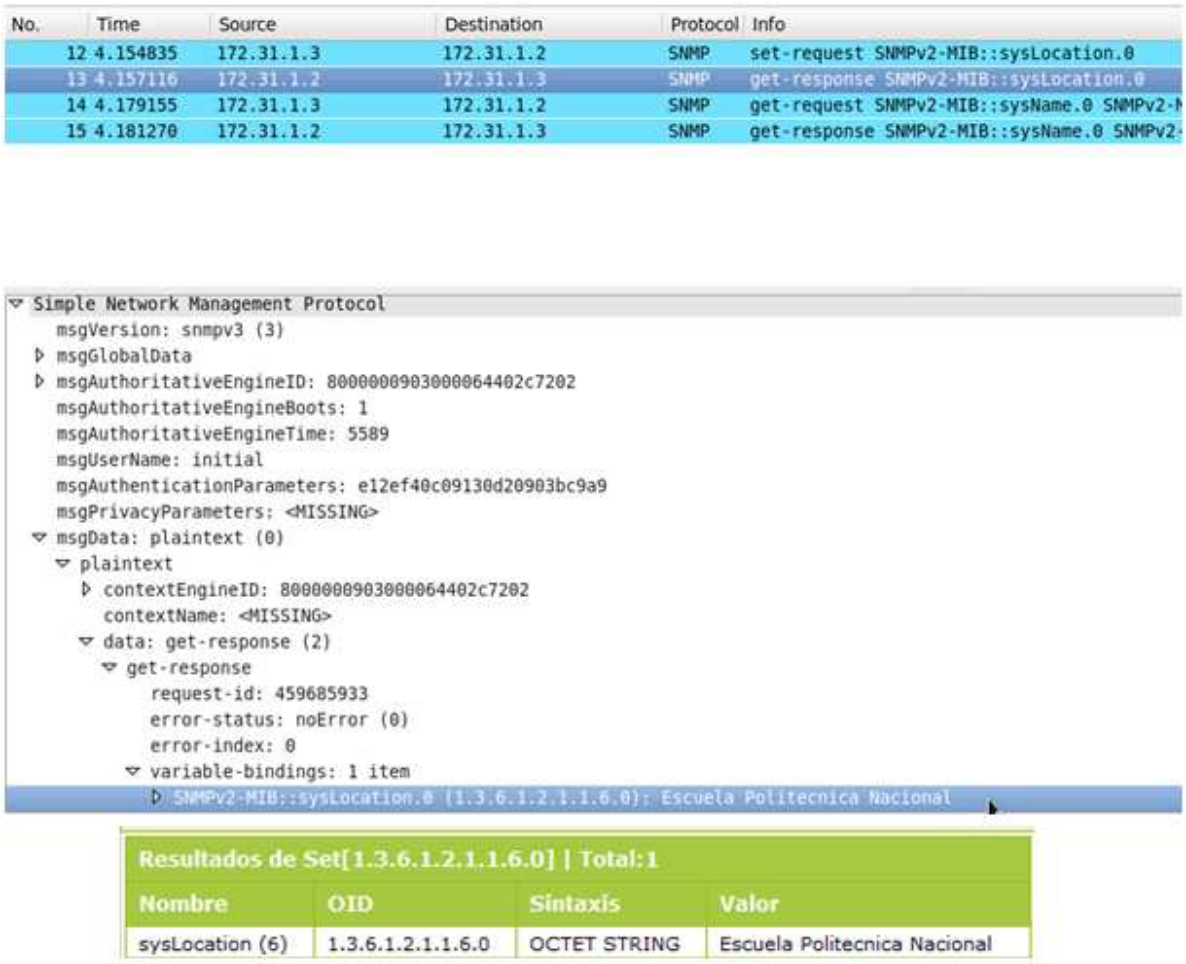


Figura 3. 95 Comprobación del resultado de una operación *SetRequest*.

3.3.3.1.6 Trap

En la siguiente figura se muestra la captura de un mensaje *Trap.* Los que participan en esta comunicación son el servidor y el equipo que tiene la dirección IP 172.31.1.4

(Servidor Linux). Como este tipo de mensaje no genera respuesta de retorno, entonces se muestra nada en la tabla de resultados de la aplicación.

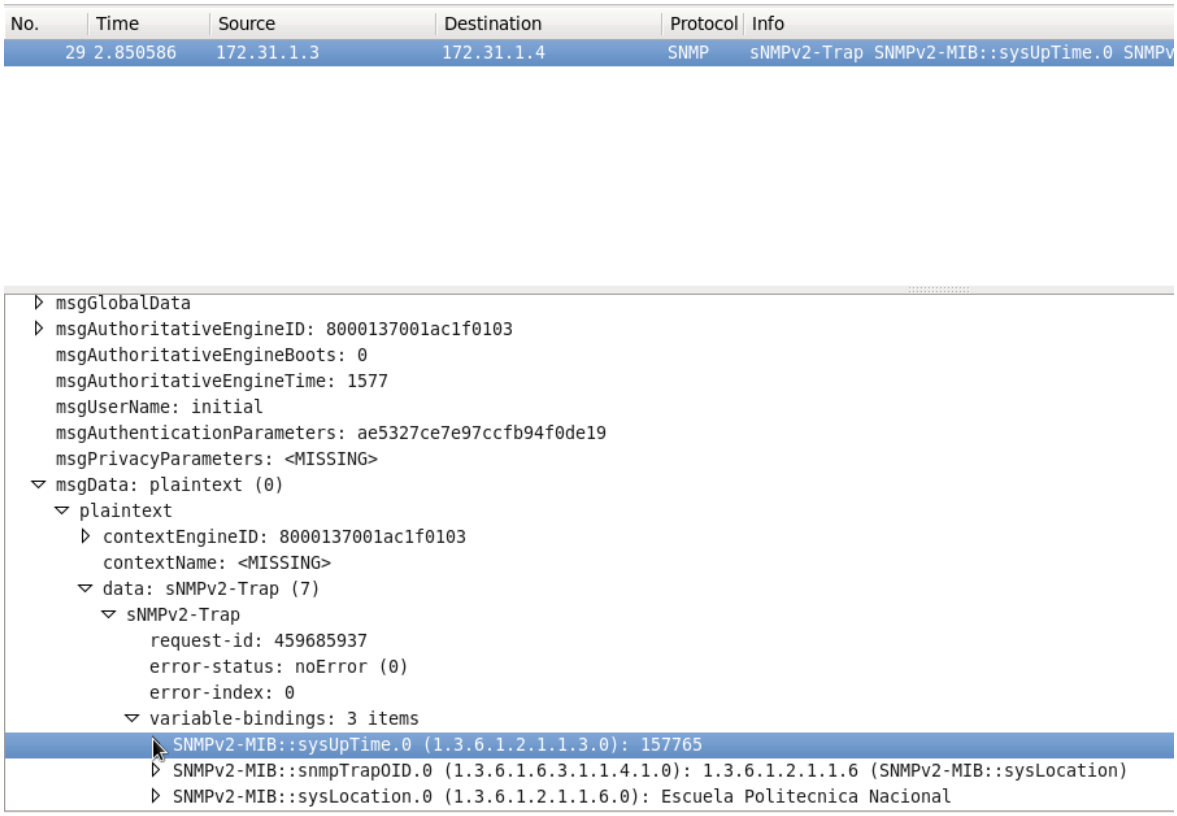


Figura 3. 96 Captura de un mensaje *Trap*.

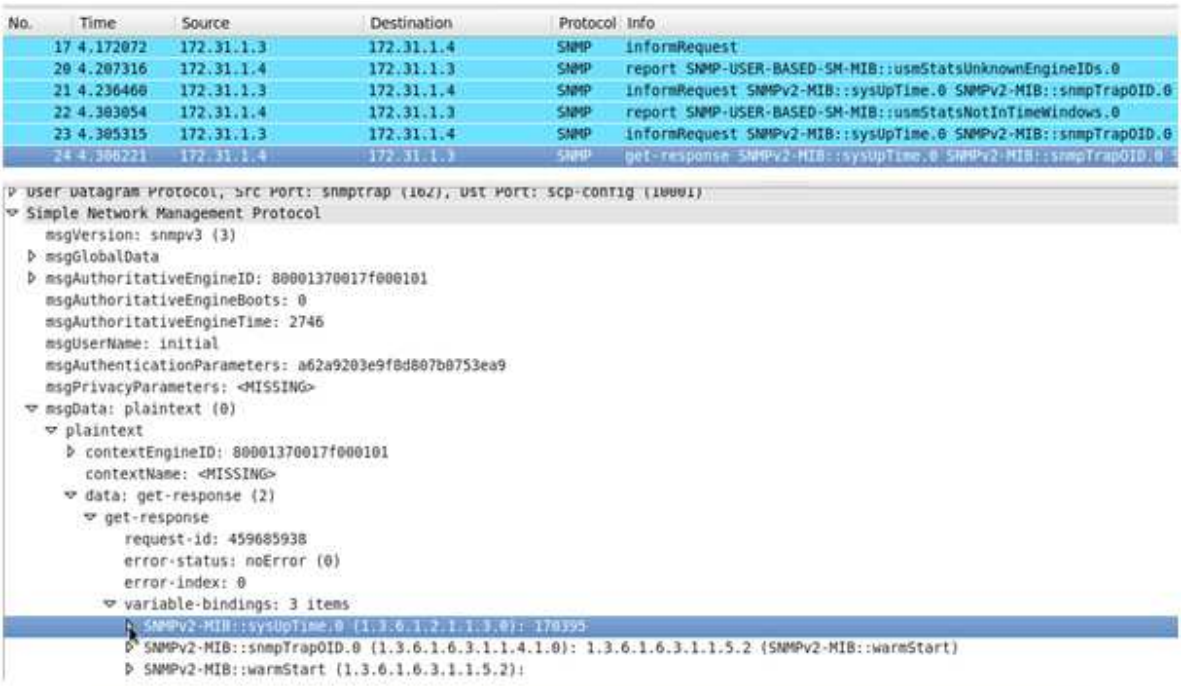
En la figura se puede apreciar que el mensaje *Trap* tiene tres vínculos variables, el primero representa las marcas de tiempo[100], el segundo identifica el tipo de trap (en este caso es el objeto *sysLocation* del grupo system) y el tercero muestra información adicional (en este caso el valor del objeto *sysLocation*, es decir: *Escuela Politécnica Nacional*).

Estos vínculos variables son los que caracterizan a una *Trap SNMPv2*.

[100] Para mayor información revisar el ítem 1.3.6.1.6.

3.3.3.1.7 InformRequest

En la siguiente figura se muestra la comprobación de resultados al enviarse un mensaje *InformRequest.* Los que participan en esta comunicación son el servidor y el equipo que tiene la dirección IP 172.31.1.4 (Servidor Linux). Este tipo de mensaje es similar al mensaje *Trap,* con la diferencia que sí genera respuesta de retorno.



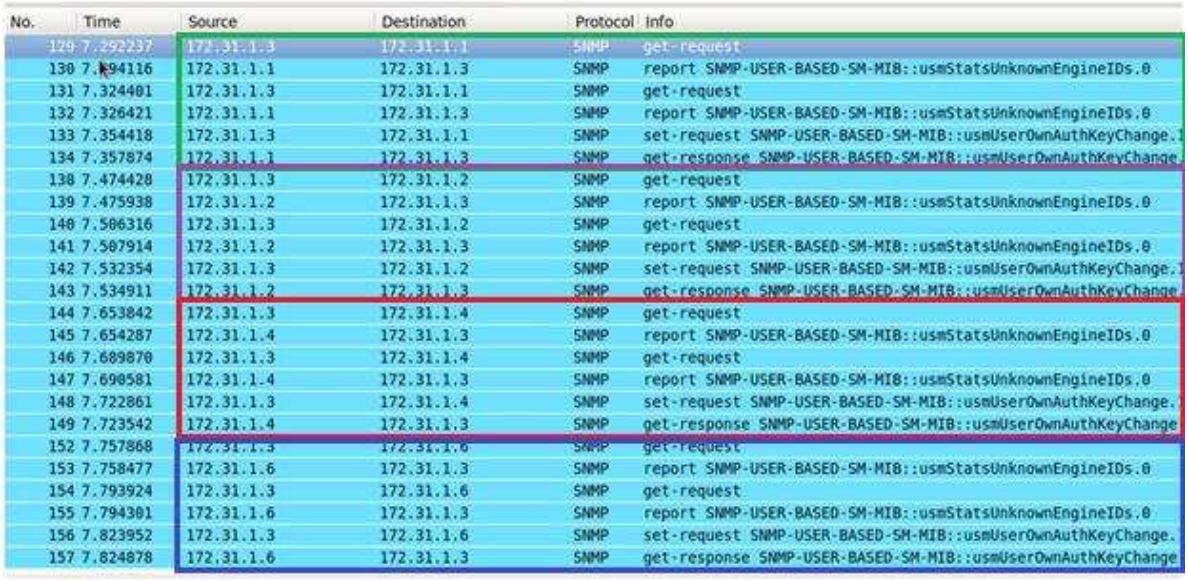
Resultados de Inform[1.3.6.1.6.3.1.1.5.2] | Total:3

Nombre	OID	Sintaxis	Valor
sysUpTime (3)	1.3.6.1.2.1.1.3.0	TimeTicks	0:28:23.95
snmpTrapOID (1)	1.3.6.1.6.3.1.1.4.1.0	OBJECT IDENTIFIER	1.3.6.1.6.3.1.1.5.2
warmStart (2)	1.3.6.1.6.3.1.1.5.2	NULL	Null

Figura 3. 97 Comprobación del resultado de una operación *InformRequest*.

3.3.3.2 Cambiar Passwords

La comunicación que se produce al exportar un cambio de passwords de un usuario hacia determinados elementos de red sigue la misma secuencia de cuatro pasos indicados anteriormente (ver figura 3.84).



No.	Time	Source	Destination	Protocol	Info
129	7.292237	172.31.1.3	172.31.1.1	SNMP	get-request
130	7.294116	172.31.1.1	172.31.1.3	SNMP	report SNMP-USER-BASED-SM-MIB::usmStatsUnknownEngineIDs.0
131	7.324401	172.31.1.3	172.31.1.1	SNMP	get-request
132	7.326421	172.31.1.1	172.31.1.3	SNMP	report SNMP-USER-BASED-SM-MIB::usmStatsUnknownEngineIDs.0
133	7.354418	172.31.1.3	172.31.1.1	SNMP	set-request SNMP-USER-BASED-SM-MIB::usmUserOwnAuthKeyChange.1
134	7.357874	172.31.1.1	172.31.1.3	SNMP	get-response SNMP-USER-BASED-SM-MIB::usmUserOwnAuthKeyChange.
138	7.474428	172.31.1.3	172.31.1.2	SNMP	get-request
139	7.475938	172.31.1.2	172.31.1.3	SNMP	report SNMP-USER-BASED-SM-MIB::usmStatsUnknownEngineIDs.0
140	7.506316	172.31.1.3	172.31.1.2	SNMP	get-request
141	7.507914	172.31.1.2	172.31.1.3	SNMP	report SNMP-USER-BASED-SM-MIB::usmStatsUnknownEngineIDs.0
142	7.532354	172.31.1.3	172.31.1.2	SNMP	set-request SNMP-USER-BASED-SM-MIB::usmUserOwnAuthKeyChange.1
143	7.534911	172.31.1.2	172.31.1.3	SNMP	get-response SNMP-USER-BASED-SM-MIB::usmUserOwnAuthKeyChange.
144	7.653842	172.31.1.3	172.31.1.4	SNMP	get-request
145	7.654287	172.31.1.4	172.31.1.3	SNMP	report SNMP-USER-BASED-SM-MIB::usmStatsUnknownEngineIDs.0
146	7.689870	172.31.1.3	172.31.1.4	SNMP	get-request
147	7.690581	172.31.1.4	172.31.1.3	SNMP	report SNMP-USER-BASED-SM-MIB::usmStatsUnknownEngineIDs.0
148	7.722861	172.31.1.3	172.31.1.4	SNMP	set-request SNMP-USER-BASED-SM-MIB::usmUserOwnAuthKeyChange.1
149	7.723542	172.31.1.4	172.31.1.3	SNMP	get-response SNMP-USER-BASED-SM-MIB::usmUserOwnAuthKeyChange
152	7.757868	172.31.1.3	172.31.1.6	SNMP	get-request
153	7.758477	172.31.1.6	172.31.1.3	SNMP	report SNMP-USER-BASED-SM-MIB::usmStatsUnknownEngineIDs.0
154	7.793924	172.31.1.3	172.31.1.6	SNMP	get-request
155	7.794301	172.31.1.6	172.31.1.3	SNMP	report SNMP-USER-BASED-SM-MIB::usmStatsUnknownEngineIDs.0
156	7.823952	172.31.1.3	172.31.1.6	SNMP	set-request SNMP-USER-BASED-SM-MIB::usmUserOwnAuthKeyChange.
157	7.824878	172.31.1.6	172.31.1.3	SNMP	get-response SNMP-USER-BASED-SM-MIB::usmUserOwnAuthKeyChange

Resultados | Total: 5 registros

No.	SecurityName	Direccion IP	Cambio	Fecha	Comentario
1	admin2	172.31.1.1	✓	Nov 26, 2010 12:51:00 PM	Cambio de passwords exitoso. Passwords vigentes: Auth='_authpassphrase2' Priv=''.
2	admin2	172.31.1.2	✓	Nov 26, 2010 12:51:00 PM	Cambio de passwords exitoso. Passwords vigentes: Auth='_authpassphrase2' Priv=''.
3	admin2	172.31.1.3	✓	Nov 26, 2010 12:51:00 PM	Cambio de passwords exitoso. Passwords vigentes: Auth='_authpassphrase2' Priv=''.
4	admin2	172.31.1.4	✓	Nov 26, 2010 12:51:00 PM	Cambio de passwords exitoso. Passwords vigentes: Auth='_authpassphrase2' Priv=''.
5	admin2	172.31.1.6	✓	Nov 26, 2010 12:51:00 PM	Cambio de passwords exitoso. Passwords vigentes: Auth='_authpassphrase2' Priv=''.

Figura 3. 98 Comprobación del resultado de una exportación de cambio de passwords de usuario.

Los mensajes SNMPv3 que se envían son del tipo S*etRequest*. Además, la comunicación que se genera es de uno a varios, es decir el servidor es el que se comunica con varios elementos de red según los que se hayan seleccionado. La figura anterior permite comprobar los resultados de la aplicación con las capturas de los paquetes de esto que se ha indicado.

Como se puede apreciar en la figura anterior, en la sección de capturas de paquetes se han encerrado en recuadros de colores a conjuntos de paquetes, esto se lo realizó con el fin de señalar cada comunicación que tiene el servidor con cada elemento de red. Además, se puede apreciar que el registro No.3 no consta en la sección de capturas; esto se debe a que esta comunicación se da a nivel local en el servidor.

La siguiente figura muestra el detalle de un mensaje de respuesta.

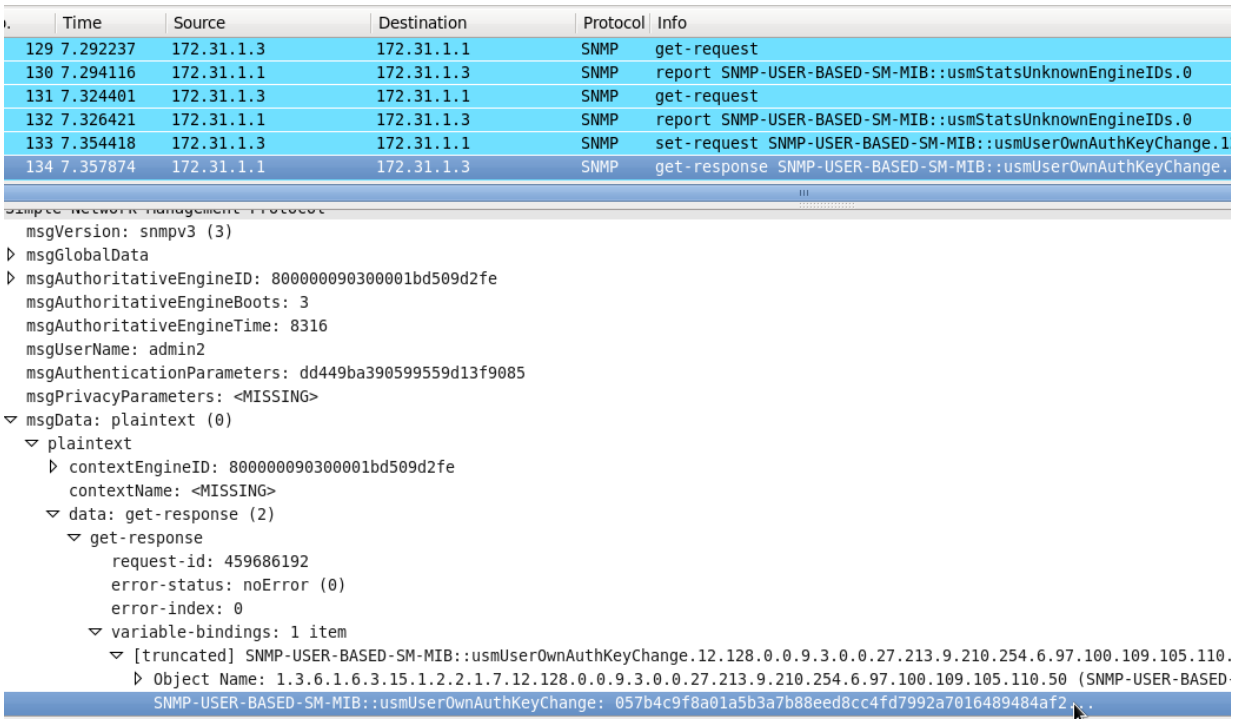


Figura 3. 99 Detalle de un mensaje de respuesta en el proceso de exportación de cambio de passwords.

Como se puede apreciar en la figura anterior, el objeto MIB que contiene este mensaje es el *usmUserOwnAuthKeyChange* y el nombre de seguridad es *admin2.* El valor asignado a este objeto (está resultado con azul) es del tipo *KeyChange*[101]*,* el cual fue calculado automáticamente por la aplicación.

El *usmUserOwnAuthKeyChange* es un objeto que sirve para modificar la clave de autenticación usada por el usuario (en este caso *admin2*). Teóricamente, para realizar operaciones de escritura sobre este objeto no se requiere de un control de acceso VACM, ya que únicamente es necesario que el nombre del usuario referenciado a la fila que se va escribir sea el mismo que el valor *msgUserName* del mensaje *SetRequest.* Sin embargo, en la realización de pruebas se ha comprobado que no todos los agentes cumplen esta teoría; y por lo tanto, fue necesario crear y exportar una vista de escritura para estos usuarios (ver la vista *usmUserTable* en la figura 3.52).

Si el usuario que va exportar su cambio de passwords soporta también privacidad; entonces, el objeto MIB *usmUserOwnPrivKeyChange* será también incluido dentro de los vínculos variables del mensaje *SetRequest.* Este objeto MIB tiene una función similar al objeto *usmUserOwnAuthKeyChange,* con la diferencia que es para cambiar la clave de privacidad.

[101] Para mayor información revisar el ítem 1.4.4.8.

3.3.4 ANÁLISIS DE LOS RESULTADOS

El análisis de resultados se enfoca en la verificación del cumplimiento de los requerimientos del usuario, de los objetivos del proyecto, y en la descripción de las ventajas que se tienen al usar la aplicación.

Haciendo un recordatorio de los objetivos del presente proyecto se tenía que se deseaba diseñar e implementar un software de monitorización y administración de red con seguridades mejoradas, con independencia de la arquitectura en donde se lo ejecute, con interfaz web y con el uso del protocolo SNMPv3. Según las pruebas realizadas confirman que han logrado alcanzar estos objetivos debido a que: la aplicación es un software de gestión de red, es del tipo web, utiliza SNMPv3 para ejecutar las funciones de gestión de red, la seguridad es implementada en base a la combinación del subsistema de seguridad USM de SNMPv3 con el protocolo HTTPS, y la aplicación puede ejecutarse en varias plataformas gracias al uso de la tecnología Java para su desarrollo.

En cuanto a los requisitos funcionales y no funcionales se refiere, las pruebas realizadas demuestran que todos éstos han sido cumplidos satisfactoriamente. Ya que la aplicación dispone de un control de ingreso a la misma mediante inicio de sesiones, permite configurar parámetros para el servidor y para los mensajes SNMPv3, permite descubrir elementos en una red dada, permite obtener información de administración de los elementos de red registrados, permite ver la estructura en forma de árbol de un grupo MIB, permite enviar todos los tipos de mensajes de SNMPv3, permite conocer fallas en la red mediante la recepción de notificaciones, permite gestionar archivos MIBs y elementos de los subsistemas USM y VACM, y permite cambiar los passwords de usuario en cada elemento de red. Además, para la implementación de la aplicación se utiliza el protocolo SNMPv3, la tecnología Java, el protocolo HTTPS, y XML (para almacenamiento de datos).

Como conclusión de los resultados obtenidos, se pueden citar las siguientes ventajas:

- La aplicación es del tipo web y puede instalarse en una variedad de sistemas operativos. Gracias a esto, el acceso a la aplicación puede realizarse desde una intranet o desde el Internet a través de un navegador web. Esto representa grandes beneficios para el administrador de la red en cuanto a trabajo a distancia se refiere. Además, dado que la aplicación fue implementada en Java se la puede instalar en la plataforma que se disponga lo cual implica reducción de costos.

- La aplicación dispone de las seguridades necesarias para transmitir información sensible. Dado que aplicación utiliza SNMPv3 y HTTPS, la información que se transmite desde el navegador web hasta el elemento administrable y viceversa, está protegida contra varias amenazas que atentan contra la integridad de la información de gestión.

- La aplicación es de fácil manejo. Debido a que la interfaz web ha sido diseñada para que el usuario se limite a dar clic en botones, enlaces, y en pocas ocasiones a escribir ciertos valores en cajas de texto o a seleccionar opciones desde listas desplegables, el manejo de la aplicación es sencilla y eficiente. Además, la aplicación maneja adecuadamente los errores de utilización y funcionamiento indicando el mensaje de aviso respectivo.

- Interfaz gráfica amigable y variedad de menús. Los detalles de la interfaz gráfica que posee la aplicación ayudan a que el usuario se sienta cómodo al utilizarla, y además, gracias a la variedad de menús, especialmente el de acceso rápido, el usuario puede acceder inmediatamente a las funciones que éste requiera desde la sección de página que éste se encuentre.

- La configuración remota potencializa la gestión de la red. La aplicación dispone de mecanismos de configuración remota (exportaciones y

anulaciones) de elementos de los subsistemas de USM y VACM pertenecientes a los dispositivos de red registrados. Como resultado de esto, el administrador general de la red puede crear y eliminar usuarios USM, vistas MIB, derechos de acceso, grupos VACM de manera remota, adecuada y rápida.

- Los usuarios USM pueden cambiar sus passwords fácilmente. Debido a SNMPv3 utiliza passwords de autenticación y privacidad los cuales pertenecen a cada usuario USM, es una buena política de seguridad cambiar los passwords periódicamente. Por lo tanto, la aplicación ofrece una función que permite al usuario actualizar sus passwords localmente en la aplicación y en todos los agentes de los elementos de red registrados. Esto es muy útil ya que evita el usuario cambie sus passwords de dispositivo en dispositivo lo cual repercute en desgaste de tiempo y en incremento de la tendencia al cometimiento de errores.

- La aplicación es portable. Gracias a que los datos que maneja la aplicación son almacenados en archivos XML en lugar de bases de datos comunes, se puede transportar fácilmente la aplicación y ejecutarla en otra plataforma. Los datos que contienen los archivos XML son manejados y procesados por procedimientos propios de la aplicación.

- Flexibilidad para agregar nuevos archivos MIB. Esta es una ventaja importante puesto que da la posibilidad de que el usuario suba nuevos archivos MIB, estándar o propietarios, según las necesidades del mismo. Cuando un archivo MIB es subido, éste es transformado a formato XML para que pueda ser utilizado por la aplicación para mostrar su forma de árbol jerárquico.

Por lo descrito anteriormente, se concluye que la aplicación ofrece mejores prestaciones que otros programas de gestión de red dedicados para sistemas operativos específicos.

CAPÍTULO 4. CONCLUSIONES Y RECOMENDACIONES

4.1 CONCLUSIONES

Las siguientes conclusiones han sido concebidas al desarrollar el presente proyecto:

- La utilización de una herramienta computacional facilita la ejecución de las funciones de gestión de redes. Considerando que hoy en día las redes de la información se tornan cada vez más complejas, la gestión de los recursos que conforman dichas redes también tiende a complicarse. Por lo tanto, hacer uso de una herramienta de gestión de redes como la desarrollada en este proyecto facilita en gran parte la realización de las funciones de monitorización y control de dichos recursos.

- La seguridad de la información de gestión se ve amenazada ante los avances tecnológicos. Dado que en la actualidad se desarrollan mejorados y potentes dispositivos de hardware (tales como: microprocesadores más rápidos e inteligentes, memorias RAM de frecuencia más alta y de mayor capacidad, discos duros de gran tamaño, etc.), éstos pueden ser utilizados para romper las seguridades que protegen a la información de gestión. Por lo tanto, debe haber un mejoramiento continuo de dichas seguridades, así como SNMPv3 que ofrece un mecanismo modular que permite para añadir nuevos y más confiables subsistemas de seguridad que ayuden a preservar eficazmente la información sensible.

- SNMPv3 es el protocolo de uso recomendado en la actualidad. Muchas empresas recomiendan el uso de SNMPv3 (por ejemplo Cisco) ya que este protocolo enfrenta la mayor parte de amenazas que existen en contra de la información de gestión. Y además, este protocolo sigue manteniendo la simplicidad que siempre lo ha caracterizado desde sus versiones anteriores.

Esta simplicidad del protocolo hace posible que la mayor parte equipos de una red lo soporten adecuadamente.

- El uso de una herramienta independiente de la arquitectura implica reducción de costos. Generalmente una empresa no está comprometida con un único proveedor de hardware y software, tienen diversidad de plataforma. Entonces, si por cada plataforma se tuviera un sistema de gestión, los costos de adquisición y mantenimiento se encarecerían significativamente. Por lo tanto, es razonable utilizar una herramienta como la presentada en el presente proyecto para que se la pueda ejecutar desde la arquitectura deseada y de esta manera reducir notablemente los costos.

- XML representa una tecnología complementaria para SNMP. Si bien es cierto, SNMP es el protocolo de gestión más utilizado debido a su simplicidad, no implica que sea el más eficaz; tiene ciertas deferencias como por ejemplo: ineficiencia en el transporte de grandes cantidades de datos, y ausencia de formatos estándar para el almacenamiento y procesamiento de la información de gestión. Es aquí donde entra la tecnología XML, satisface completamente estos requerimientos, no reemplazando totalmente a SNMP sino como un complemento; de esta manera se evitaría realizar una migración total de tecnología la cual representaría costes elevados.

- XML es un mecanismo alternativo al motor de bases de datos. Puesto que la tecnología XML, de entre de todas sus aplicaciones existentes puede utilizarse para almacenamiento de información, representa una alternativa al uso de las bases de datos. Además, existen varios componentes desarrollados para la tecnología XML como lenguajes de consulta (por ejemplo *XPath*, y *XQuery*), y lenguajes de estructuración de datos (por ejemplo *DTD y XSD*), que permiten manejar la información de manera similar como se lo hace en las bases de datos; con el agregado de que no se requiere conexión alguna para acceder y manipular la información almacenada.

- La configuración remota agilita las funciones de administración de la red. Considerando como ejemplo el presente proyecto, el cual permite configurar remotamente elementos de los subsistemas USM y VACM en cada dispositivo de red, se puede notar que existe una agilidad y facilidad para ejercer control sobre la red. Como beneficios de esto se tiene: mayor eficacia para gestionar los recursos de la red, ahorro de tiempo en la ejecución de las funciones, y minimización de errores cometidos por parte del administrador de la red.

- Java es una potente herramienta para el desarrollo de aplicaciones web. Java, a diferencia de otras tecnologías, posee una gran cantidad de proyectos de colaboración destinados al desarrollo de aplicaciones web, de entre los cuales JSP y Servlets son los más utilizados. Ambas tecnologías permiten generar contenido web dinámico; los Servlets incorporan contenido web dentro de código Java, mientras que los JSPs incorporan código Java en el contenido web. Estas tecnologías permiten al programador tener un control mayor sobre los recursos que conforman una aplicación web. Además, existen proyectos que desarrollan servidores web/contenedores de servlets que se ejecutan sobre Java (por ejemplo Tomcat), los cuales pueden ser ejecutados sobre una diversidad de sistemas operativos, y con la ventaja que son gratuitos y de código abierto.

- El Lenguaje de Desarrollo Unificado (UML, por sus siglas en inglés) es una guía general para el diseño e implementación de aplicaciones software. Dado que UML está conformado por un conjunto de modelos, cada uno de ellos guía paso a paso en el desarrollo de una aplicación, empezando por la recopilación de requerimientos y terminado en la implementación de la misma. Además, UML ofrece flexibilidad para adecuar los modelos según el tipo de aplicación que se desea realizar. Por ejemplo, en el caso del presente proyecto, el modelo de análisis no pertenece a UML pero se la agregado con la finalidad de comprender de mejor manera los requerimientos del usuario, y de encaminar al proyecto hacia un diseño MVC (modelo-vista-controlador), el

cual encaja adecuadamente con las tecnologías que Java propone para el desarrollo de aplicaciones web.

- Las aplicaciones del tipo web son las más preferidas por los usuarios. Gracias al vertiginoso avance del Internet, las aplicaciones web se han convertido en las más solicitadas por los usuarios, no sólo por información estática que éstas proporcionan sino que además permiten interacciones directas con el usuario y de manera remota. Es por tal motivo que se desarrollan aplicaciones web para satisfacer diversas necesidades de los usuarios; por ejemplo, si un usuario requiere configurar un equipo (por ejemplo un router inalámbrico), por lo menos éste tiene su aplicación web para efectuar dicha configuración. Y también, el presente proyecto cabe dentro de estos ejemplos, que para ejecutar las funciones de administración de red, se tiene la aplicación web que se ha desarrollado para este fin.

4.2 RECOMENDACIONES

Como recomendaciones se tienen las siguientes:

- Debido al continuo avance tecnológico, para la gestión de red es recomendable utilizar SNMPv3. Ya no se debería utilizar versiones anteriores de SNMP puesto que éstas no ofrecen las seguridades necesarias para manejar la información de gestión. Existen aplicaciones como analizadores de paquetes con los cuales se puede capturar los mensajes, y a partir de esto se puede obtener información sensible desde los dispositivos de red, modificar inadecuadamente parámetros de configuración o dar de baja a los dispositivos.

- XML es una tecnología de tal importancia cuyo conocimiento debería ser impartido. La mayoría de aplicaciones sobretodo del tipo web están utilizando la tecnología XML para diferentes fines, ya que a partir de ella se han

desarrollado una diversidad de tecnologías tales como XSD, DTD, XSLT etc.; por lo tanto, XML es digno de ser un tema de instrucción para saber utilizarlo y considerarlo en el desarrollo de diferentes proyectos.

- Promover un mecanismo centralizado que maneje la información que usa una aplicación de gestión cuando se tienen varias de ellas en una misma red. Esta es una recomendación para mejorar el desempeño de la aplicación del presente proyecto. Cada aplicación maneja un repositorio XML local en el cual se almacenan los datos que ésta utiliza; y cuando se tienen varias aplicaciones en una misma red, resulta un problema hacer que todos los repositorios tengan la misma información. Por lo tanto, resulta conveniente promover un mecanismo centralizado que ayude a solucionar este problema.

- Proponer mecanismos alternativos para el control de ingreso a la aplicación de gestión de red. Esta también es una recomendación para el mejoramiento de la aplicación del presente proyecto. El mecanismo de control de ingreso está basado en el nombre de usuario y su password; este no es un mecanismo efectivo puesto que puede ser vulnerado mediante ataques de fuerza bruta o palabras de diccionario. Entonces resulta conveniente proponer otros mecanismos más seguros para efectuar el control de ingreso.

- En SNMPv3, en lo posible utilizar los protocolos SHA-1 y AES. Hoy en día se ha comprobado que los protocolos DES y MD5 ya no son confiables puesto que ha logrado vulnerarlos debido la corta longitud de sus respectivas llaves. Los protocolos SHA-1 y AES tienen llaves de mayor longitud y están basados en algoritmos más complejos.

- Impulsar en el desarrollo de un agente SNMP que sea independiente de la plataforma. Esta idea se originó a partir del desarrollo del presente proyecto, el cual es un gestor SNMP independiente de la arquitectura. Java propone una infraestructura para lograr este objetivo.

CAPÍTULO 5. BIBLIOGRAFÍA

5.1 BIBLIOGRAFÍA

- [2] SRINIVA, Basa; NAVEEN, Ganji, Enhanced NMS Tool Architecture for Discovery and Monitoring of Nodes. Blekinge Institute of Technology. Enero 2008.

- [4] DOUGLAS, Mauro; SCHMIDT, Kevin, Essential SNMP. Segunda edición. O'Reilly. USA. 2005.

- [10] VELÁSQUEZ, Andrés; Diseño e implementación de un módulo software para la monitorización de Elementos de una red informática utilizando el protocolo SNMP y el lenguaje XML. Febrero 2009.

- [11] STALLINGS William; SNMP, SNMPv2, SNMPv3 and RMON 1 and 2. Tercera Edición. USA. Abril 2004.

- [12] RFC 3411, Arquitecture for SNMP Management Frameworks. Diciembre 2002.

- [13] RFC 3413, Simple Network Management Protocol (SNMP) Applications. Diciembre 2002.

- [14] RFC 3412, Message Processing and Dispatching for the Simple Network Management Protocol (SNMP). Diciembre 2002.

- [18] RFC 3826, The Advanced Encryption Standard (AES) Cipher Algorithm in the SNMP User-based Security Model. Junio 2004.

- [19] RFC 3414, User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3). Diciembre 2002.

- [20] RFC 3415, View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP). Diciembre 2002.

- [23] RAY, Erik, Learning XML. Segunda edición. O'Reilly. USA. 2003.

- [24] McLAUGHLIN, Brett; EDELSON, Justin, Java and XML. O'Reilly. USA. 2006.

- [30] ZAMBON G., SEKLER.M; Beginning JSP™, JSF™, and Tomcat Web Development From Novice to Professional. Apress. USA. 2007.

- [35] JACOBSON I., BOOCH G., RUMBAUGH J., El Proceso Unificado de Desarrollo de Software. Pearson. España. 2000.

- [36] GARCÍA J., RODRIGUEZ J., MINGO I., IMAZ I., BRAZÁLEZ A., LARZABAL A., CALLEJA J., GARCÍA J.; Aprenda Java como si estuviera en primero. San Sebastián. Enero 2000.

- [37] BROWN S., DALTON S., JEPP D., JOHNSON D., LI S., RAIBLE M.; Pro JSP 2. Cuarta edición. Apress. USA. 2005.

- [40] COWARD D., YOSHIDA Y.; Java™ Servlet Specification Version 2.4. Sun Microsystems. USA. 2003.

5.2 REFERENCIAS DE INTERNET

- [1] WIKIPEDIA, Red de computadoras, http://es.wikipedia.org/wiki/Red_de_computadoras.

- [3] LUQUE J.,GONZALO F., ESCUDERO J.,CARRASCO A., Universidad de Sevilla, España, TMN vs. SNMP-based Network Management Systems, http://personal.us.es/jluque/Congresos/1999%20Cigre%20Polonia-1.pdf.

- [5] VICTORIA, Patricia, Administración de Redes, http://www.monografias.com/trabajos43/administracion-redes/administracion-redes2.shtml#funciones. Febrero 2007.

- [6] WON-KI HONG J., Department of Computer Science and Engineering POSTECH, Overview of Network Management, http://dpnm.postech.ac.kr/cs637/lecture/nm-overview.ppt.

- [7] VALLES, Kirssy, DIRECTV Latinoamérica, SNMPv3, http://neutron.ing.ucv.ve/revista-e/No6/Valles%20Kirssy/SNMPV3/Snmpv3.htm.

- [8] LESCHENNE, Sebastián; SALAZAR, Martín, Universidad Nacional de Rosario, Modelo de Gestión Internet SNMP-RMON, www.eie.fceia.unr.edu.ar/~etapia/Redes2002pub/SNMP_RMON.PDF. 2002.

- [9] WIKIPEDIA, Orden Lexicográfico, http://es.wikipedia.org/wiki/Orden_lexicogr%C3%A1fico.

- [15] ANÓNIMO, The Secure Hash Algorithm Directory MD5, SHA-1 and HMAC Resources, http://www.secure-hash-algorithm-md5-sha-1.co.uk/

- [16] SÁNCHEZ G., HENAINE M., GÓMEZ V., Análisis de Comparativo de diferentes algoritmos de cifrado (encriptar), http://www.somece.org.mx/.../4_SanchezGuerreroLourdes_analisis2.pdf.

- [17] CISCO, AES and 3-DES Encryption Support for SNMP v3 Version, http://www.cisco.com/en/US/docs/ios/ios_xe/netmgmt/configuration/guide/nm_encrypt_snmp_sup_xe.pdf.

- [21] W3SCHOOLS, Tutorial XML, http://www.w3schools.com/xml/default.asp.

- [22] W3C, XML, http://www.w3.org/TR/REC-xml/

- [25] W3SCHOOLS, DTD Tutorial, http://www.w3schools.com/dtd/default.asp

- [26] W3SCHOOLS, XML Schema Tutorial, http://www.w3chools.com/schema/schema_intro.asp

- [27] W3SCHOOLS, XML DOM Tutorial, http://www.w3chools.com/dom/dom_intro.asp

- [28] ANÓNIMO, Simple API for XML (SAX) and the Document Object Model (DOM), http://www.dcs.bbk.ac.uk/~mick/academic/xml/dip/sax-dom.shtml

- [29] W3SCHOOLS, XSL Languages, http://www.w3schools.com/xsl/xsl_languages.asp

- [31] WSCHOOLS, XQuery Tutorial, http://www.w3schools.com/xquery/default.asp

- [32] GARSHOL, XML tools by category, http://www.garshol.priv.no/download/xmltools/cat_ix.html#SC_XML
- [33] KLIE T., STRAUS F., Integrating SNMP Agents with XML-based Management Systems, http://www.ibr.cs.tu-bs.de/papers/commag-2004.pdf
- [34] WIKIPEDIA, HTTP Secure, http://en.wikipedia.org/wiki/HTTP_Secure
- [38] WIKIPEDIA, Java Servlet, http://en.wikipedia.org/wiki/Java_Servlet
- [39] WIKIPEDIA, JavaServer Pages, http://es.wikipedia.org/wiki/JavaServer_Pages
- [41] ECLIPSE Foundation, Eclipse Web Tools Platform Project, http://www.eclipse.org/projects/project_summary.php?projectid=webtools
- [42] ADOBE, Dreamweaver Features, http://www.adobe.com/products/dreamweaver/features/
- [43] W3SCHOOLS, CSS Tutorial, http://www.w3schools.com/css/css_intro.asp
- [44] ORACLE, Java SE, http://www.oracle.com/technetwork/java/javase/system-configurations-135212.html
- [45] AARON Leskiw, Network Management Software Autumn Smackdown, http://www.networkmanagementsoftware.com/network-management-software-autumn-smackdown