ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE INGENIERÍA

CONSTRUCCIÓN DE UN BENCHMARK PARA REALIZAR UN ESTUDIO COMPARATIVO ENTE WEB SERVICES EN PLATAFORMAS JAVA Y .NET

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS INFORMÁTICOS Y DE COMPUTACIÓN

MÁRMOL PANAMÁ MIGUEL PATRICIO
PEÑAHERRERA PACHECO DAYSI DE LAS MERCEDES

DIRECTOR: Ing. Christian Suárez

Quito, Octubre 2006

DECLARACIÓN

Nosotros, Miguel Patricio Mármol Panamá y Daysi de las Mercedes Peñaherrera Pacheco, declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que hemos consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedemos nuestros 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.

Miguel Patricio Mármol Panamá

Daysi Peñaherrera Pacheco

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado Miguel Patricio Mármol Panamá y Daysi de las Mercedes Peñaherrera Pacheco, bajo mi supervisión.

(Ing. Christian Suárez)

DIRECTOR DE PROYECTO

AGRADECIMIENTO

A mis padres por el apoyo y el amor incondicional, y por ser un ejemplo de vida para mí.

A Daysi, quien con su paciencia y dedicación ayudó a cumplir esta meta.

A Lucy, Dani y Anantu por haberme acogido en momentos cruciales de mi vida.

A los amigos que siempre han estado en las buenas, en las no tan buenas y en las malas.

Para finalizar quiero agradecer a mis hermanos y al resto de mi familia.

MIGUEL

DEDICATORIA

A todas las personas que creen y confían en mí

MIGUEL

AGRADECIMIENTO

A mis padres por haberme dado la oportunidad de estudiar, por el apoyo que me han brindando a lo largo de la carrera y en especial en la realización de este proyecto.

De manera especial a Miguel por el trabajo y colaboración para poder alcanzar este objetivo.

A Javier por constituir una persona muy especial en mi vida, por su apoyo incondicional paciencia y especialmente por permitirme ver el mundo de manera diferente

Mi sincero agradecimiento a los grandes amigos por compartir su amistad y aquellos momentos inolvidables.

Por último a mis hermanos en especial a Doris por estar siempre a mi lado en los buenos y malos momentos y a mis sobrinitos por su ternura e inocencia.

DAYSI

DEDICATORIA

Dedico este proyecto a mi familia y en especial a mi madre por compartir este sueño conmigo y por su apoyo incondicional.

DAYSI

INDICE DE CONTENIDO

RESUMEN ..... **I**

PRESENTACIÓN ..... **III**

CAPITULO 1. ARQUITECTURA ORIENTADA A SERVICIOS **1**

1.1. PLATAFORMAS DE LA ARQUITECTURA ORIENTADA A SERVICIOS 1

1.1.1. FUNDAMENTOS DE LA ARQUITECTURA ORIENTADA A SERVICIOS 1

1.1.2. COMPONENTES DE LA ARQUITECTURA ORIENTADA A SERVICIOS 3

1.1.2.1. Servicio 3

1.1.2.2. Descripción del servicio 4

1.1.3. ROLES DE LA ARQUITECTURA ORIENTADA A SERVICIOS 4

1.1.3.1. Proveedor de servicios 4

1.1.3.2. Agente de servicios 4

1.1.3.3. Consumidor de servicios 4

1.1.4. PLATAFORMAS DE LA ARQUITECTURA ORIENTADA A SERVICIOS 5

1.1.4.1. BEA WebLogic 5

1.1.4.2. JBOSS 6

1.1.4.3. Tomcat 6

1.1.4.4. Microsoft .NET 7

1.1.4.5. Oracle Application Server y Developer Suite 7

1.2. INTRODUCCION A LOS WEB SERVICES 8

1.2.1. HISTORIA DE LOS WEB SERVICES 8

1.2.2. CUANDO USAR WEB SERVICES 8

1.2.3. VENTAJAS DE USAR WEB SERVICES 9

1.2.4. BENEFICIOS DE LOS WEB SERVICES SOBRE PROVEEDORES DE SERVICIO DE APLICACIONES 10

1.2.5. MODELO DE ENTREGA DE WEB SERVICES 11

1.2.6. ACUERDOS A NIVEL DE SERVICIO (SLAS) 11

1.2.7. COMPRANDO WEB SERVICES 12

1.2.8. PUBLICANDO WEB SERVICES 13

1.2.8.1. S2C Web Services 14

1.2.8.2. S2B Web Services 15

1.2.8.3. S2E Web Services 16

1.2.9. TECNOLOGÍAS UTILIZADAS EN LOS WEB SERVICES 17
1.2.9.1. SOAP 17
1.2.9.1.1. Alternativas de SOAP 19
1.2.9.2. WSDL 19
1.2.9.3. UDDI 20
1.2.9.4. Otras Tecnologías 22

CAPITULO 2. FUNDAMENTO TEÓRICO 23
2.1. WEB SERVICES IMPLEMENTADOS EN PLATAFORMA JAVA 23
2.1.1. ARQUITECTURA DE WEB SERVICES EN J2EE: JAXRPC 23
2.1.1.1. Servicio, Puertos y Bindings 24
2.1.2. ARQUITECTURA DE WEB SERVICES EN: AXIS 24
2.1.2.1. Handlers y el Message Path en Axis 25
2.1.2.1.1. Message Path en el Servidor 25
2.1.2.1.2. Message Path en el Cliente 27
2.1.2.2. Desarrollo del Web Service Usando WSDL2Java de Axis 27
2.1.2.2.1. Binding en la parte servidora: skeleton 28
2.1.2.2.2. Los Skeleton 28
2.2. WEB SERVICES IMPLEMENTADOS EN PLATAFORMA .NET 29
2.2.1. ARQUITECTURA DE WEB SERVICES: .NET 2003 29
2.2.2. ARQUITECTURA DE WEB SERVICES: VISUAL STUDIO .NET 2005 30
2.2.2.1. Infraestructura de Web Services 31
2.2.2.2. Desarrollo del Web Services usando visual Studio .NET 2005 31
2.2.2.3. Invocar un Web Service 32
2.2.3. DEFINICION 32
2.3. ESTADIGRAFOS 32
2.3.1. TEORIA DE ESTADIGRAFOS 32
2.3.1.1. Variables discretas y continuas 33
2.3.1.1.1. Variable discreta 33
2.3.1.1.2. Variables continúas 33
2.3.1.2. Las variables y el tiempo. 33
2.3.1.3. Estadígrafos 34
2.3.2. ESTADÍGRAFOS DE POSICIÓN O DE TENDENCIA CENTRAL 34
2.3.2.1. Media Aritmética. 34
2.3.2.2. Mediana 35

2.3.2.2.1. Fráctiles 35
2.3.2.3. Moda 35
2.3.2.4. Media Geométrica 36
2.3.3. ESTADIGRAFOS DE DISPERSIÓN 36
2.3.3.1. Rango 36
2.3.3.2. Desviación 36
2.3.3.2.1. Desviación media 37
2.3.3.2.2. Desviación mediana 37
2.3.3.3. Desviación cuadrática media o varianza 37
2.3.3.3.1. Desviación Típica o estándar 37

CAPITULO 3. DESARROLLO DEL BENCHMARK 38
3.1. CAPTURA DE REQUERIMIENTOS DEL BENCHMARK 38
3.1.1. INTRODUCCION 38
3.1.1.1. Respuesta Vacío 38
3.1.1.2. Respuesta Estructura 38
3.1.1.3. Respuesta Lista Enlazada 39
3.1.1.4. Respuesta Sintético 39
3.1.1.5. Respuesta Orden 39
3.1.2. REQUERIMIENTOS 39
3.1.2.1. Diagrama de Casos de Uso y Descripción 40
3.1.2.1.1. Diagrama de Casos de Uso 40
3.1.2.2. Glosario 44
3.2. ANÁLISIS DEL BENCHMARK 47
3.2.1. MODELO DE ANÁLISIS 47
3.2.1.1. Clases de Análisis 47
3.2.1.1.1. Diagrama de Clases 47
3.2.1.1.2. Gestor de agentes 48
3.2.1.1.3. Gestor de ejecución 49
3.2.1.1.4. Gestor de Monitoreo 50
3.2.1.1.5. IU Ejecutar Benchmark 50
3.2.1.1.6. IU Web Service 51
3.2.1.2. Paquetes de Análisis 51
3.2.1.2.1. Distribución de Clases y Paquetes de Análisis 52

3.2.1.2.2. Relaciones entre paquetes 52
3.2.1.3. Realización de Casos de Uso – Análisis 53
3.2.1.3.1. Caso de Uso Ejecutar Benchmark 53
3.2.1.3.2. Caso de Uso Evaluar web service 55
3.2.1.3.3. Caso de Uso Monitorear CPU y Memoria 57
3.2.1.3.4. Caso de Uso Obtener Resultados 59
3.3. DISEÑO DEL BENCHMARK 61
3.3.1. MODELO DE DISEÑO 61
3.3.1.1. Clases de Diseño 61
3.3.1.1.1. Diagrama de Clases 62
3.3.1.1.2. Gestor de agentes 62
3.3.1.1.3. Gestor de ejecución 64
3.3.1.1.4. Gestor de Monitoreo 64
3.3.1.1.5. I Web Service 65
3.3.1.1.6. IU Ejecutar Benchmark 65
3.3.2. REALIZACIÓN DE CASOS DE USO - DISEÑO 66
3.3.2.1. Caso de Uso Ejecutar Benchmark 66
3.3.2.1.1. Diagrama de Interacción 66
3.3.2.1.2. Flujo de Sucesos - Diseño 67
3.3.2.2. Caso de Uso Evaluar web service 68
3.3.2.2.1. Diagrama de Interacción 68
3.3.2.2.2. Flujo de Sucesos - Diseño 69
3.3.2.3. Caso de Uso Monitorear CPU y Memoria 70
3.3.2.3.1. Diagrama de Interacción 70
3.3.2.3.2. Flujo de Sucesos - Diseño 70
3.3.2.4. Caso de Uso Obtener Resultados 71
3.3.2.4.1. Diagrama de Interacción 71
3.3.2.4.2. Flujo de Sucesos – Diseño 72
3.3.3. SUSBSISTEMAS DE DISEÑO 72
3.3.3.1. Dependencias entre Subsistemas de Diseño 73
3.3.4. INTERFACES 73
3.3.4.1. Diagrama de Interfaces para los Subsistemas de Diseño 74
3.3.5. DESCRIPCIÓN DE LA ARQUITECTURA (VISTA DEL MODELO DE DISEÑO) 74
3.3.6. MODELO DE DESPLIEGUE 74

3.3.7. DESCRIPCIÓN DE LA ARQUITECTURA (VISTA DEL MODELO DE DESPLIEGUE) 76
3.4. IMPLEMENTACIÓN DEL BENCHMARK 76
3.4.1. DESCRIPCIÓN DE HERRAMIENTAS A UTILIZARSE PARA EL DESARROLLO DEL BENCHMARK 77
3.4.1.1. Selección de la herramienta de desarrollo 77
3.4.1.1.1. .NET 77
3.4.1.1.2. Java 1.5 79
3.4.2. DESCRIPCIÓN DE LA HERRAMIENTA DE DESARROLLO 81
3.4.3. CONSTRUCCIÓN DEL BENCHMARK 83
3.4.3.1. Estándares de programación 83
3.4.3.1.1. Idioma 83
3.4.3.1.2. Archivos 84
3.4.3.1.3. Comentarios 84
3.4.3.1.4. Convenciones de Nombramiento 85
3.4.3.2. Modelo de Implementación 86
3.4.3.3. Clases de Implementación 86
3.4.3.4. Componentes 89
3.4.3.4.1. Componente Address.java 89
3.4.3.4.2. Componente Agent.java 89
3.4.3.4.3. Componente Client.java 89
3.4.3.4.4. Componente CPU.resultados 89
3.4.3.4.5. Componente CPUMonitoringListener.java 90
3.4.3.4.6. Componente Item.java 90
3.4.3.4.7. Componente memoria.resultados 90
3.4.3.4.8. Componente MemoryMonitoringListener.java 90
3.4.3.4.9. Componente Node.java 90
3.4.3.4.10. Componente Order.java 90
3.4.3.4.11. Componente parametros.props 90
3.4.3.4.12. Componente probador.resultados 91
3.4.3.4.13. Componente Service.java 91
3.4.3.4.14. Componente ServiceLocator.java 91
3.4.3.4.15. Componente ServiceSoap.java 91
3.4.3.4.16. Componente ServiceSoap12Stub.java 91
3.4.3.4.17. Componente ServiceSoapStub.java 91

3.4.3.4.18. Componente Structure.java 92
3.4.3.4.19. Componente Synthetic.java 92
3.4.3.4.20. Componente Tester.java 92
3.4.3.4.21. Componente WSTestClient.java 92
3.4.3.5. Subsistemas de Implementación 92
3.4.3.5.1. Subsistema Benchmark 94
3.4.3.5.2. Subsistema Monitoreo 94
3.4.3.5.3. Subsistema Web Service 94
3.4.3.6. Descripción de la Arquitectura (Vista del modelo de Implementación) 95
3.4.3.7. Plan de integración de construcciones 95
3.4.3.7.1. Primera Iteración 95
3.4.3.7.2. Segunda Iteración 96
3.4.4. PRUEBAS DEL BENCHMARK 96
3.4.4.1. Modelo de Pruebas 97
3.4.4.1.1. Caso de Prueba Ejecutar Benchmark 97
3.4.4.1.2. Caso de Prueba Evaluar Web Service 98
3.4.4.1.3. Caso de Prueba Monitorear CPU y Memoria 100
3.4.4.1.4. Caso de Prueba Obtener Resultados 101
3.5. PLANIFICACIÓN DE PRUEBAS PARA COMPARACIÓN 103
3.6. APLICACIÓN DEL BENCHMARK 104
3.6.1. INSTALACIÓN 104
3.6.2. DETALLE DE LA INSTALACIÓN Y CONFIGURACIÓN 105
3.6.3. AMBIENTE DE PRUEBAS 105
3.6.3.1. Definición de escenarios para la realización de pruebas con los servidores de aplicaciones en las plataformas Java y .NET 105
3.6.3.2. Selección de Servidor de Aplicaciones para la plataforma Java. 107
3.6.3.2.1. Número de Peticiones Atendidas 107
3.6.3.2.2. Tiempo de Respuesta 109
3.6.3.2.3. Disponibilidad 110
3.7. MEDICIÓN DE ÍNDICES Y GENERACIÓN DE RESULTADOS 112

CAPITULO 4. ANÁLISIS DE RESULTADOS 114
4.1. ANÁLISIS DE RESULTADOS DEL WEB SERVICE IMPLEMENTADO EN LA PLATAFORMA JAVA.....115
4.1.1. ÍNDICE NÚMERO DE LLAMADAS ATENDIDAS POR EL WEB SERVICE EN EL SERVIDOR DE APLICACIONES JBOSS. 115
4.1.2. ÍNDICE TIEMPO DE RESPUESTA DEL WEB SERVICE EN EL SERVIDOR DE APLICACIONES JBOSS 117
4.1.3. ÍNDICE DISPONIBILIDAD DEL WEB SERVICE EN EL SERVIDOR DE APLICACIONES JBOSS 118
4.1.4. ÍNDICES DE DISPONIBILIDAD DE CPU DEL SERVIDOR DE APLICACIONES JBOSS .120
4.1.4.1. Índice USER_CPU del web service en el Servidor de Aplicaciones JBOSS..... 120
4.1.4.2. Índice IDLE_CPU del web service en el Servidor de Aplicaciones JBOSS..... 120
4.1.4.3. Índice TOTAL_CPU del web service en el Servidor de Aplicaciones JBOSS..... 121
4.1.5. INDICES DE DISPONIBILIDAD DE MEMORIA EN EL SERVIDOR DE APLICACIONES JBOSS 122
4.1.5.1. Índice USED_MEM por el web service en el Servidor de Aplicaciones JBOSS..... 122
4.2. ANÁLISIS DE RESULTADOS DEL WEB SERVICE IMPLEMENTADO EN LA PLATAFORMA .NET 124
4.2.1. ÍNDICE NÚMERO DE LLAMADAS ATENDIDAS POR EL WEB SERVICE EN EL SERVIDOR DE APLICACIONES IIS.....124
4.2.2. ÍNDICE TIEMPO DE RESPUESTA DEL WEB SERVICE EN EL SERVIDOR DE APLICACIONES IIS.....125
4.2.3. ÍNDICE DISPONIBILIDAD DEL WEB SERVICE EN EL SERVIDOR DE APLICACIONES IIS 127
4.2.4. ÍNDICES DE DISPONIBILIDAD DE CPU DEL SERVIDOR DE APLICACIONES JBOSS .128
4.2.4.1. Índice USER_CPU del web service en el Servidor de Aplicaciones IIS 128
4.2.4.2. Índice IDLE_CPU del web service en el Servidor de Aplicaciones IIS 129
4.2.4.3. Índice TOTAL_CPU para el web service en el Servidor de Aplicaciones IIS 130

4.2.5. INDICES DE DISPONIBILIDAD DE MEMORIA EN EL SERVIDOR DE APLICACIONES IIS 131
4.2.5.1. Índice USED_MEM por el web service en el Servidor de Aplicaciones IIS 131
4.3. COMPARACIÓN DE LOS RESULTADOS OBTENIDOS 133
4.3.1. ÍNDICE NÚMERO DE PETICIONES ATENDIDAS DEL WEB SERVICE 133
4.3.2. ÍNDICE TIEMPO DE RESPUESTA DEL WEB SERVICE 134
4.3.3. ÍNDICE DISPONIBILIDAD DEL WEB SERVICE 136
4.3.4. ÍNDICE DISPONIBILIDAD DE CPU UTILIZADO POR EL WEB SERVICE 138
4.3.5. ÍNDICE DISPONIBILIDAD DE MEMORIA UTILIZADO POR EL WEB SERVICE 138

CAPITULO 5. CONCLUSIONES Y RECOMENDACIONES 139
5.1. CONCLUSIONES 139
5.2. RECOMENDACIONES 141

BIBLIOGRAFÍA 143
GLOSARIO DE TÉRMINOS 145

INDICE DE TABLAS

Tabla 1-1 Tipos de Web Services 17
Tabla 2-1: Binding en la parte servidora Skeleton 28
Tabla 3-1 Caso de Uso Ejecutar Benchmark 41
Tabla 3-2 Caso de Uso Evaluar web service 42
Tabla 3-3 Caso de Uso Medir Porcentaje de Uso de Memoria 43
Tabla 3-4 Caso de Uso Obtener Resultados 44
Tabla 3-5 Glosario de Términos 44
Tabla 3-6 Lista de atributos de la clase Gestor de agentes 48
Tabla 3-7 Lista de atributos de la clase Gestor de ejecución 49
Tabla 3-8 Transición de las Clases de Análisis a las de Diseño 61
Tabla 3-9 Lista de atributos de la clase Gestor de agentes 62
Tabla 3-10 Métodos de la clase Gestor de agentes 63
Tabla 3-11 Lista de atributos de la clase Gestor de ejecución 64
Tabla 3-12 Métodos de la clase Gestor de ejecución 64
Tabla 3-13 Métodos de la clase Gestor de Monitoreo 64
Tabla 3-14 Métodos de la interfaz I Web Service 65
Tabla 3-15 Métodos de la interfaz IU Ejecutar Benchmark 66
Tabla 3-16 Interfaces definidas para los Subsistemas de Diseño 73
Tabla 3-17 Cuadro comparativo entre las herramientas de desarrollo 80
Tabla 3-18 Etiquetas para generación de documentación en Java 84
Tabla 3-19 Transición de las clases de diseño a las de implementación 87
Tabla 3-20 Procedimiento de prueba del caso de prueba Ejecutar Benchmark .. 97
Tabla 3-21: Resultados Obtenidos caso de prueba Ejecutar Benchmark 98
Tabla 3-22 Procedimiento de prueba caso de prueba Evaluar Web Service 99
Tabla 3-23: Resultados obtenidos caso de prueba Evaluar Web Service 99
Tabla 3-24 Procedimiento de prueba caso de Monitorear CPU y Memoria 100
Tabla 3-25 Resultados obtenidos caso de prueba Monitorear CPU y Memoria. 101
Tabla 3-26 Procedimiento de prueba caso de prueba Obtener Resultados 102
Tabla 3-27 Resultados obtenidos caso de prueba Obtener Resultados 103
Tabla 3-28 Configuraciones de Hardware 104
Tabla 3-29 Configuraciones de software 104

Tabla 3-30 Número de peticiones atendidas por los servidores de aplicaciones de la plataforma Java 107
Tabla 3-31 Tiempo de respuesta para los servidores de aplicaciones Java 109
Tabla 3-32 Disponibilidad para los servidores de aplicaciones Java 110
Tabla 4-1: Número de llamadas atendidas por el web service en el servidor de aplicaciones JBOSS 116
Tabla 4-2: Tiempo de Respuesta del web service en el servidor de aplicaciones JBOSS 117
Tabla 4-3: Disponibilidad del web service en el servidor de aplicaciones JBOSS118
Tabla 4-4: Número de llamadas atendidas por el web service en el servidor de aplicaciones IIS. 124
Tabla 4-5: Tiempo de Respuesta del web service en el Servidor de Aplicaciones IIS 126
Tabla 4-6: Disponibilidad del web service en el Servidor de Aplicaciones IIS ... 127
Tabla 4-7: Número de Peticiones Atendidas JBOSS - ISS 133
Tabla 4-8: Tiempo de Respuesta JBOSS – ISS 135
Tabla 4-9: Tiempo de Respuesta JBOSS – ISS 136

INDICE DE FIGURAS

Figura 1-1 Arquitectura Orientada a Servicios 2
Figura 1-2: Arquitectura SOAP 19
Figura 1-3: Web Services Description Language 20
Figura 1-4: Interacción de Protocolos 22
Figura 2-1: Arquitectura de Web Services en J2EE:JAXRPC 23
Figura 2-2: Servicios, puertos y bindings 24
Figura 2-3: Message Path en el Servidor 26
Figura 2-4: Message Path en el Cliente 27
Figura 2-5: Escenario web services .NET 30
Figura 2-6 Infraestructura de Servicios Web 31
Figura 3-1 Diagrama de Casos de Uso 40
Figura 3-2 Diagrama de Clases de Análisis 47
Figura 3-3 Paquetes de Análisis 51
Figura 3-4 Distribución de Clases y Paquetes de Análisis 52
Figura 3-5 Relaciones entre paquetes 53
Figura 3-6 Diagrama de Colaboración del Caso de Uso Ejecutar Benchmark 53
Figura 3-7 Diagrama de Colaboración del Caso de Uso Evaluar web service 55
Figura 3-8 Diagrama de Colaboración del Caso de Uso Monitorear CPU y Memoria 57
Figura 3-9 Diagrama de Colaboración del Caso de Uso Obtener Resultados 59
Figura 3-10 Diagrama de Clases de Diseño 62
Figura 3-11 Diagrama de Interacción del Caso de Uso Ejecutar Benchmark.. 66
Figura 3-12 Diagrama de Interacción del Caso de Uso Evaluar web service .. 68
Figura 3-13 Diagrama de Interacción del Caso de Uso Monitorear CPU y Memoria 70
Figura 3-14 Diagrama de Interacción del Caso de Uso Obtener Resultados .. 71
Figura 3-15 Dependencias entre Subsistemas de Diseño 73
Figura 3-16 Diagrama de Interfaces para los Subsistemas de Diseño 74
Figura 3-17 Diagrama de despliegue 75
Figura 3-18 Nodo Servidor 75
Figura 3-19 Nodo Cliente 76

Figura 3-20 Diagrama de clases de implementación..... 88
Figura 3-21 Diagrama de subsistemas de implementación..... 93
Figura 3-22 Comparación Número de llamadas entre servidores de aplicaciones Java..... 108
Figura 3-23 Comparación del tiempo de respuesta para los servidores de aplicaciones Java..... 109
Figura 3-24 Comparación de Disponibilidad de los servidores de aplicaciones Java..... 111
Figura 4-1: Número de Peticiones Atendidas por el web service en el Servidor de Aplicaciones JBOSS 116
Figura 4-2: Tiempo de Respuesta del web service en el Servidor de Aplicaciones JBOSS 118
Figura 4-3: Disponibilidad del web service en el Servidor de Aplicaciones JBOSS 119
Figura 4-4: Porcentaje USER_CPU del web service en el Servidor de Aplicaciones JBOSS 120
Figura 4-5: Porcentaje IDLE_CPU del web service en el Servidor de Aplicaciones JBOSS 121
Figura 4-6: Porcentaje TOTAL_CPU del web service en el Servidor de Aplicaciones JBOSS 122
Figura 4-7: Porcentaje USED_MEMORY por el web service en el Servidor de Aplicaciones JBOSS 123
Figura 4-8: Número de Peticiones Atendidas por el web service en el Servidor de Aplicaciones IIS..... 125
Figura 4-9: Tiempo de respuesta del web service en el Servidor de Aplicaciones IIS..... 126
Figura 4-10: Disponibilidad del web service en el Servidor de Aplicaciones IIS 128
Figura 4-11: Porcentaje USER_CPU del web service en el Servidor de Aplicaciones IIS..... 129
Figura 4-12: Porcentaje IDLE_CPU del web service en el Servidor de Aplicaciones IIS..... 130
Figura 4-13: Porcentaje TOTAL_CPU del web service en el Servidor de Aplicaciones IIS..... 131

Figura 4-14: Porcentaje USED_MEMORY por el web service en el Servidor de Aplicaciones IIS..... 132
Figura 4-15: Comparación Número de Peticiones Atendidas por los web services en los servidores de aplicaciones JBOSS y IIS..... 134
Figura 4-16: Comparación Tiempo de Respuesta de los web services en los servidores de aplicaciones JBOSS y IIS. 135
Figura 4-17: Comparación de Disponibilidad de los web services en los servidores de aplicaciones JBOSS y IIS. 137

RESUMEN

El presente trabajo describe en cada uno de los capítulos desarrollados los conceptos y definiciones necesarias para el análisis, diseño, implementación y pruebas de un Benchmark que permita evaluar web services desarrollados en plataformas Java y .NET respectivamente.

Inicialmente, se presenta el marco teórico a cerca de los conceptos sobre arquitectura orientada a objetos, roles de la misma, se mencionan algunas de las plataformas que utilizan arquitectura orientada a servicios, así como también se expone un estudio introductorio a los web services, describiendo la historia de los mismos, cuando se los debe utilizar y las tecnologías que éstos utilizan.

A continuación se presentan las formas de implementación de los web services en plataforma Java, al igual que en plataforma .NET, conceptos necesarios para tener un conocimiento a cerca de la teoría de benchmark y de estadígrafos

Posteriormente se define la captura de requerimientos, análisis y diseño de los servicios con los que contará el Benchmark que permitirá evaluar web services desarrollados en plataformas Java y .NET siguiendo la metodología RUP (Rational Unified Process), adicionalmente se detallan los parámetros que serán medidos, los mismos que permitirán evaluar el web service tanto en rendimiento como el uso de procesador y memoria que este haga, debido a que existen varias distribuciones para la plataforma Java ha sido necesario probar el web service alojado sobre varios servidores de aplicaciones, por lo tanto se exponen las pruebas realizadas con los mismos y una selección de entre ellos con la finalidad de obtener un solo servidor de aplicaciones para las pruebas entre las plataformas Java y .NET.

Una vez escogidos los servidores de aplicaciones, se definen las pruebas que van a ser realizadas utilizando el Benchmark, las mismas que nos permitirán

realizar la comparación del rendimiento del web service implementados en plataformas Java y .NET respectivamente.

Finalmente se presentan como resultado del desarrollo del Benchmark y los conocimientos adquiridos a lo largo de las investigaciones realizadas, las conclusiones y recomendaciones, con la finalidad de constituir un aporte significativo para el desarrollo de futuros proyectos de titulación que pretendan evaluar el rendimiento de web services.

PRESENTACIÓN

En los últimos años la mayoría de procesos de negocio han cambiado, necesitando características que anteriormente no se las tomaba en cuenta, tales como: flexibilidad, interconectividad y autonomía debido a las condiciones del mercado, a los nuevos modelos organizacionales y a los escenarios de uso de los sistemas de información. En este contexto, el Internet está cambiando la forma en la que se ofrecen los negocios y los servicios en la sociedad global, y en la que estos negocios interoperan. Esta tendencia nos lleva a sistemas de información conectados e integrados a través de la infraestructura que proporciona Internet. Internet introduce un nuevo entorno donde el software se va a ofrecer y acceder como servicio. Los web services proporcionan la plataforma tecnológica ideal para conseguir la completa integración de los procesos de negocio de una organización con diferentes organizaciones.

Los web services surgieron ante la necesidad de estandarizar la comunicación entre distintas plataformas y lenguajes de programación. Anteriormente se habían realizado intentos de crear estándares pero fracasaron o no tuvieron el suficiente éxito, algunos de ellos por ser dependientes de la implementación del vendedor.

Actualmente las aplicaciones de software se encuentran desarrolladas en diferentes plataformas por lo que se ha hecho necesaria la utilización de web services para comunicar las diferentes aplicaciones o componentes de éstas, de forma estándar a través de protocolos comunes y de manera independiente al lenguaje de programación, plataforma de implementación, formato de presentación o sistema operativo.

Los web services permiten que las aplicaciones compartan información y que además invoquen funciones de otras aplicaciones independientes de cómo se hayan creado las aplicaciones, cuál sea el sistema operativo o la plataforma en que se ejecutan y cuáles son los dispositivos utilizados para obtener acceso a ellas, de ahí la necesidad de evaluar el rendimiento de estos en diferentes

plataformas, para lo cual se pone a consideración el presente trabajo que incluye un estudio completo y desarrollo del diseño e implementación de un Benchmark que permitirá evaluar web services de similar funcionalidad, implementados en plataformas Java y .NET.

Los Autores

CAPITULO 1. ARQUITECTURA ORIENTADA A SERVICIOS

La Arquitectura Orientada a Servicios (en inglés Service Oriented Architecture SOA), es una manera de concebir sistemas distribuidos, que define la utilización de servicios para dar soporte a los requerimientos de software del usuario.

1.1. PLATAFORMAS DE LA ARQUITECTURA ORIENTADA A SERVICIOS

1.1.1. FUNDAMENTOS DE LA ARQUITECTURA ORIENTADA A SERVICIOS

La Arquitectura Orientada a Servicios dentro del espacio de las tecnologías de información es un estilo de arquitectura que promueve el uso de servicios como un mecanismo arquitectónico para la construcción de sistemas de información, con la peculiar característica “evolucionar y adaptarse fácilmente a los cambios futuros”[1].

La Arquitectura Orientada a Servicios promueve la creación de servicios, encargados de realizar alguna tarea u operación especifica del negocio, actuando sobre sus propios datos y conteniendo sus propias reglas. De tal forma que una vez escritos todos los servicios se pueda construir una aplicación que dirija el uso de todos estos, para integrar un proceso de negocio.

Esta arquitectura está caracterizada por las siguientes propiedades:

- Vista Lógica

El servicio es una abstracción (vista lógica) de los programas, base de datos, procesos de negocio, que llevan a cabo una operación del negocio.

- Orientación a mensajes

El servicio se define formalmente en términos de los mensajes intercambiados entre agentes proveedores y solicitantes, este mensaje no se basa en las

propiedades de los agentes, la arquitectura de estos agentes se abstrae en SOA, lo que permite que cualquier componente o programa se pueda incorporar a esta arquitectura.

- Orientación a la descripción

En la descripción se incluyen aquellos detalles que se exponen públicamente y son importantes para el uso del servicio. La semántica de un servicio debe documentarse a través de la descripción.

- Granularidad

"Los servicios tienen la tendencia de usar un número pequeño de operaciones con mensajes relativamente complejos"[2].

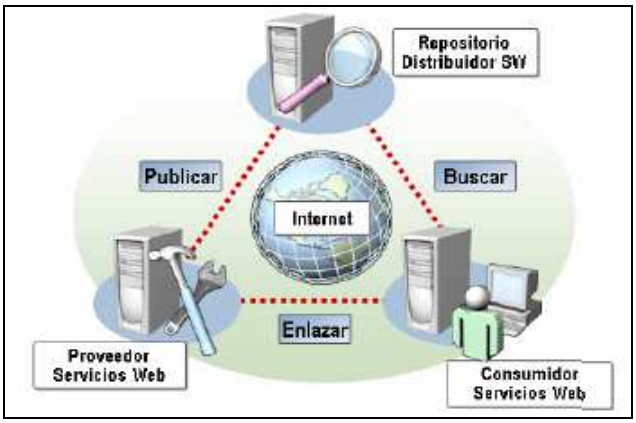
- Orientación a la Red

Los servicios en la mayoría de los casos son utilizados a través de la red, aunque este no es un requisito absoluto.

- Neutral a la plataforma

Los mensajes se envían en un formato estándar y neutral a la plataforma, distribuido a través de las interfaces (XML).

Figura 1-1 Arquitectura Orientada a Servicios



Elaboración y Fuente:
http://pangea.upv.es/N+ISIS05/documents/VicentePelechano/ServiciosWeb.pdf

1.1.2. COMPONENTES DE LA ARQUITECTURA ORIENTADA A SERVICIOS

1.1.2.1. Servicio

Un servicio funciona como una aplicación autónoma e independiente, es decir no depende del estado de otras funciones o procesos. Expone una interfaz de llamado basada en mensajes, capaz de ser accedidas a través de la red, los servicios por lo general incluyen lógica de negocios, como manejo de estado (datos), relevantes en la solución del problema para el cual fueron diseñados.

Los mensajes que son utilizados para la comunicación hacia y desde el servicio deben contener o referenciar toda la información necesaria para entenderlo. Se pretende que exista el mínimo posible de llamadas entre el cliente y el servicio.

Los servicios se diferencian de un componente distribuido en:

- Mucho menos acoplados con sus aplicaciones cliente que los componentes
- De menor granularidad que los componentes
- Son administrados y controlados de manera independiente
- Expone su funcionalidad a través de protocolos abiertos e independientes de plataforma
- Su localización en la red es transparente, de esta manera se garantiza la escalabilidad y tolerancia a fallos.

La implementación ideal de un servicio, exige resolver ciertos inconvenientes técnicos, inherentes a su modelo.

- Utilización de mensajería confiable, debido a que los tiempos de llamado, tamaño de los mensajes y la comunicación de la red no son despreciables.
- Desarrollo de planes de contingencia que eviten la parálisis de las aplicaciones y servicios, cuando se presenten problemas en la configuración de la red, ya que la respuesta del servicio se ve afectada directamente por estos aspectos.
- Debe manejar comunicaciones no confiables, mensajes impredecibles, reintentos, mensajes fuera de secuencia, etc.

1.1.2.2. Descripción del servicio

Contiene el detalle de la interfaz e implementación del servicio, esto incluye tipos de datos, operaciones, información de enlace y ubicación en la red, puede incluir categorización y otro tipo de información que facilitarían el descubrimiento. La descripción completa es un conjunto de documentos de descripción XML. La descripción del servicio puede ser publicada directamente a un solicitante o a un directorio.

1.1.3. ROLES DE LA ARQUITECTURA ORIENTADA A SERVICIOS

1.1.3.1. Proveedor de servicios

El proveedor es un servicio o un sistema que hace que los Web services estén disponibles sobre una red como Internet. Un proveedor de servicio publica el servicio o un agente de servicios.

1.1.3.2. Agente de servicios

El agente de servicios es un servidor o sistema de red que mantiene un directorio de los web services ofrecidos por diferentes compañías. El consumidor de servicios es un servidor o sistema de red que accede y utiliza un web service.

1.1.3.3. Consumidor de servicios

El consumidor interactúa con el agente de servicios para encontrar un web service que satisfaga sus necesidades, después el consumidor se enlaza, inicia el contacto con el proveedor de servicio y se establece la comunicación entre las dos aplicaciones o sistemas de computación.

Compañías que agregan contenido o servicios pueden beneficiarse más directamente volviéndose consumidores de servicios, estás compañías se comunican con gran variedad de fuentes de datos para agrupar información y presentarla a sus consumidores.

Cualquier compañía que crea o mantiene Software y quiere hacer que este Software esté disponible sobre una red puede convertirse en proveedor de web

services, muchas compañías hacen disponibles los Web services solo dentro de su organización o entre socios de negocios.

Las compañías que crean y desarrollan Software no necesariamente son proveedores de servicios, estas empresas pueden contratar hosting en otras compañías. Los proveedores de servicios de Internet (ISP, Internet Server Providers) o compañías de Web-hosting son las empresas más probables para proveer de servicio de Web services porque estás organizaciones ya poseen una infraestructura web, y además pueden desarrollar y comercializar sus propios web services.

Los agentes de servicio se comunican a los consumidores y los direccionan a los proveedores de servicio apropiados, un ejemplo de los agentes de servicio son los operadores de registros UDDI. También existen otras empresas que actúan como agentes, en las cuales se publican web services.

1.1.4. PLATAFORMAS DE LA ARQUITECTURA ORIENTADA A SERVICIOS

Aquí se detalla las plataformas para desarrollo y funcionamiento de los web services de los principales vendedores mundiales. Es siempre bueno revisar la página web de cada vendedor para ver las versiones actualizadas de cada producto.

1.1.4.1. BEA WebLogic

La plataforma BEA WebLogic 9.0 es una aplicación simple e integrada para construir, extender, integrar, desplegar y manejar aplicaciones como un proceso de negocios de principio a fin. La plataforma WebLogic es una plataforma para aplicaciones orientada a servicios y web services a gran escala que simplifica la integración de recursos empresariales en aplicaciones y procesos de negocio.

Consiste en varias aplicaciones.

- BEA WebLogic Server: Un servidor de aplicaciones

- BEA WebLogic Integration: Incluye aplicaciones de integración, manejo de procesos de negocio e integración de negocio a negocio.
- BEA WebLogic Workshop: Un ambiente de desarrollo para construir aplicaciones como web services, utiliza controles Java y componentes J2EE.
- BEA WebLogic Portal: Una infraestructura de portal empresarial.

1.1.4.2. JBOSS

JBOSS es el primer servidor de aplicaciones de código abierto implementado en Java puro, al estar basado en Java, JBOSS puede ser utilizado en cualquier sistema operativo que lo soporte, preparado para la producción y certificado J2EE 1.4, cuenta con soporte total para web J2EE services y arquitectura orientada a servicios (SOA), disponible en el mercado, ofreciendo una plataforma de alto rendimiento, dentro de las características destacadas de JBOSS es un producto de licencia de código abierto por lo tanto puede ser distribuido sin restricciones.

JBOSS al no disponer de un motor especializado propio para web services, se puede integrar con Axis que funciona como un motor SOAP, al residir un motor SOAP dentro en un servidor de aplicaciones Java, se permite que los web services puedan ser publicados y de esta manera ser accesibles desde otras plataformas y ó lenguajes que también soporten SOAP.

1.1.4.3. Tomcat

Tomcat también llamado Jakarta Tocat o Apache Tomcat funciona como servidor web con soporte de servlets y JSPs, incluye el comilador Jasper, que compila JSPs convirtiéndolas en servlets, Tomcat puede funcionar como servidor web por sí mismo, en sus inicios, existió la percepción que funcionando de forma autónoma era sólo recomendable para entornos de desarrollo y entornos con requisitos mínimos de velocidad y gestión de transacciones, la percepción de hoy en día es diferente y Tomcat es usado como servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad.

Tomcat al no disponer de un motor especializado propio para web services, se puede integrar con Axis que funciona como un motor SOAP, al residir un motor SOAP dentro en un servidor de aplicaciones o motor de servlet, se permite que los web services puedan ser publicados y de esta manera ser accesibles desde otras plataformas y ó lenguajes que también soporten SOAP.

1.1.4.4. Microsoft .NET

Microsoft introdujo el concepto de web services en el 2000 con su iniciativa .NET, y es por supuesto la empresa líder en el mercado de web services, según estimaciones se cree que Microsoft está un año adelantado en el desarrollo de tecnologías para web services.

La muy conocida solución integrada Visual Studio .NET y su variedad de lenguajes de programación permite la creación de web services. En el centro de .NET está el .NET framework que maneja y ejecuta las aplicaciones y provee de las librerías de clases. Microsoft incluye una tecnología para web services DISCO (Discovery of Services) para localizar web services en un directorio particular en el servidor. La creación de archivo DISCO y WSDL es automática.

“Hoy en día los programas de Microsoft pueden interactuar con web services incluyendo toda la suite de Microsoft Office y la gran variedad de servidores. web services son parte integral de Windows Server 2003” [3].

Para manejo de transacciones de negocios, definición de procesos de negocio y comunicaciones entre sistemas, Microsoft tiene la tecnología business talk basada en XML en su servidor Microsoft BizTalk framework y BizTalk Schema Library. Además Microsoft incluye algunos web services como .NET My Services, Map Point .NET.

1.1.4.5. Oracle Application Server y Developer Suite

El servidor de aplicaciones Oracle es un producto integrado de Software para crear y desplegar portales e-business, aplicaciones y web services. Oracle 10g

Application Server provee una arquitectura basada en estándares que permite integrar productos de otros vendedores para desarrollar web services eficientemente.

1.2. INTRODUCCION A LOS WEB SERVICES

1.2.1. HISTORIA DE LOS WEB SERVICES

Desde la década de los 90 y aún más con la aparición del Internet, las empresas desarrolladoras de software han tenido la necesidad e inquietud de buscar o contar con algún método para lograr integrar sistemas de software y hardware. Para esto muchas compañías empezaron a crear de forma individual la mejor manera de lograr esta integración, el tiempo pasaba y la competencia entre ellas cada vez era más fuerte buscando la tecnología integradora.

Debido al crecimiento del Internet a niveles tan altos y el impacto causado por las tecnologías de la información en las últimas dos décadas, la manera de hacer negocios y la comunicación entre las personas y las empresas cambió de rotundamente. Bajo este contexto se hacía cada vez mayor la necesidad de integrar y compartir información entre distintas plataformas tanto de software y hardware.

Las empresas se dieron cuenta que era casi imposible crear una plataforma integrando de forma individual, es así que deciden atacar el problema de raíz. Para esto decidieron que en lugar de crear la mejor plataforma integradora, era mucho mejor buscar un lenguaje común de intercambio de información aprovechando los estándares existentes en el mercado. Es de esta forma que nacen los Web Services.

1.2.2. CUANDO USAR WEB SERVICES

Para resolver un problema en cualquier organización, primero se necesita tener un claro entendimiento del negocio y sus necesidades, y de las herramientas que se pueden utilizar para resolver tal problema. Los Web Services son una solución muy poderosa, pero no siempre son la solución para todos los

problemas del negocio. Los Web Services tienen su gran ventaja en la interoperabilidad; existen también situaciones en las que los clientes y proveedores poseen una plataforma en común que puede proveer una solución optimizada para esa plataforma. Para elegir usar Web Services se debe conocer la infraestructura actual y además considerar cualquier cambio a futuro.

Los Web Service son comúnmente conocidos como una tecnología de Internet, por el funcionamiento y no podríamos asumir una específica, es por tal razón que los Web Services son una solución para Internet.

Entre otros asuntos tenemos que considerar algunos de los problemas típicos que nos presenta el Internet, es así que si el problema presenta una transferencia para una gran cantidad de datos, los Web Services no serían la mejor opción. Entonces una de las principales consideraciones debería ser el ancho de banda que puede manejar la aplicación.

Una oportunidad ideal para usar Web Services es proveer información a disposición de compañías y personas externas, y no solamente esto, sino también un Web Service debe proveer funcionalidad como pueden ser objetos y métodos que provean algún servicio.

1.2.3. VENTAJAS DE USAR WEB SERVICES

- Los web services operan utilizando estándares abiertos, lo que permite la comunicación entre aplicaciones escritas en diferentes lenguajes y entre diferentes plataformas.
- Un web service puede ser utilizado para la comunicación con múltiples compañías.
- Comparativamente con otras soluciones, los web services son fáciles y no muy costosos de implementar, esto porque utiliza la infraestructura existente.
- Los web services pueden reducir significativamente los costos de comunicaciones negocio a negocio (Business-to-Bussines o B2B) e integración de aplicaciones ofreciendo así un tangible retorno de inversión.

La mayor ventaja es la de usar estándares abiertos, Word Wide Web Consortium (W3C) es una organización que define tecnologías para el Web, está encargada de asegurar que tales especificaciones se mantengan abiertas a cualquier vendedor. Además Microsoft e IBM están promoviendo la interoperabilidad entre implementaciones de web services y son dos de los miembros fundadores de Web Services Interoperability Organization (WS-I).

1.2.4. BENEFICIOS DE LOS WEB SERVICES SOBRE PROVEEDORES DE SERVICIO DE APLICACIONES

La idea de ofrecer software como un servicio por Internet no es única para los web services.

Desde 1998 los Proveedores de Servicio de Aplicaciones (ASPs – Aplication Service Providers) han provisto de aplicaciones personalizadas de negocio por Internet. Los ASPs usualmente desarrollan un conjunto base de aplicaciones y luego los acomodan para satisfacer a clientes específicos. Al usar ASPs para acceder a las aplicaciones de negocio, las compañías eliminan el costo de desarrollar y mantener sus propias aplicaciones; en lugar de eso, las compañías realizan un pago por uso y sus empleados acceden a las aplicaciones necesarias por Internet. Los ASPs asumen la responsabilidad de mantener y si fuese necesario actualizar la aplicación.

A pesar de ofrecer tales servicios, los ASPs no ganaron gran aceptación y tampoco resultaron en un buen negocio por el costo de mantener y actualizar el software. Un ejemplo muy claro fueron los ASPs que mantienen aplicaciones de e-commerce que tenían que realizar frecuentes actualizaciones de los inventarios para la información provista por la compañía que contrataba el servicio. La mayoría de los ASPs hacían los contratos donde cobraban por el número de clientes que usaban la aplicación en un tiempo específico, y esto no cubría los costos de proveer el servicio.

Los web services en cambio ofrecen una nueva forma de organizar el software como un servicio de Internet, y las ventajas de los web services seguramente

mejorarán la iniciativa de proveer software como un servicio, los web services pueden ser pequeños componentes que realizan funciones específicas; esto en sí se debe a que las aplicaciones provistas por los ASPs (Hosted Applications) típicamente ofrecían una interfase de usuario mientras que los web services ofrecen una interfaz de programación. Esto no solo mejora la flexibilidad para los consumidores de construir aplicaciones únicas sino que más importante aún hace que las actualizaciones sean más fáciles de desplegar y menos costosas al aprovechar la ventaja de los web services que pueden acceder directamente a las bases de datos de la misma compañía consumidora para hacer consultas.

Es por esto que los ASPs están modificando sus infraestructuras técnicas y modelos de negocio para convertirse más en Proveedores de web services.

1.2.5. MODELO DE ENTREGA DE WEB SERVICES

Las tecnologías para los web services pueden crear nuevas oportunidades de negocio. Para los vendedores de software representa un nuevo método de distribución de sus productos como suscripciones de servicio sobre Internet. Otras compañías desarrollan sus propias aplicaciones de software para satisfacer sus necesidades corporativas como manejo de cadena de proveedores, administración de recursos humanos y control de inventarios. Estas compañías pueden mejorar sus modelos de negocio e incrementar sus beneficios agrupando sus procesos del negocio como Web Services, luego auspiciando sus servicios a otras compañías que requieran una funcionalidad similar.

Además existe la necesidad de que nuevos negocios sirvan a la industria de los Web Services y actúen como intermediarios entre los web services y los consumidores.

1.2.6. ACUERDOS A NIVEL DE SERVICIO (SLAS)

La mayoría de compradores de productos software los hacen adquiriendo licencias que describen los términos de uso de la aplicación de software. En cambio, muchos web services con pago en cuotas o web services accesibles

solo con compra, son pagados en base a suscripciones. Los parámetros de las suscripciones a los web services son descritas en Acuerdos a Nivel de Servicio llamados SLAs (service-level agreements). SLAs son contratos legales en los cuales un proveedor de servicio describe el nivel de servicio que garantiza por un Web Service específico, y que cubren solo por un tiempo estipulado luego del cual el acuerdo es renegociado.

Los SLAs pueden definir relaciones entre proveedores y consumidores o también entre las empresas desarrolladores de un producto que contratan hosting de un proveedor de servicio. Una importante función de los SLAs es la de determinar la calidad del servicio (QoS – quality of service). La calidad del servicio es definida por factores como la probabilidad de que un servicio pueda responder una petición en un tiempo dado, porque también un servicio ejecuta sus tareas, que tan rápido funciona el servicio y que tan fiable y seguro es. Los requerimientos de calidad de servicio incluyen niveles de disponibilidad, accesibilidad, integridad, desempeño, confiabilidad, seguridad y composición en base a estándares.

Los SLAs son parte importante para el éxito de los Web Services ya que los consumidores necesitan confiar que los servicios que contratan van a cumplir con ciertos requerimientos de calidad. Otra forma de que los consumidores confíen en que van a tener una esperada calidad en el servicio comprando servicios a través de los agentes de servicio. El mayor interés de los agentes de Web Services es que los compradores confíen y que no se cree una mala reputación, por esta razón el comprador puede esperar una alta calidad en el servicio.

1.2.7. COMPRANDO WEB SERVICES

Existen algunos mecanismos disponibles para la compra de los web services. Primero consideremos los web services que están disponibles en el Internet gratis. Los web services que son gratis, generalmente son versiones limitadas o como condición se tiene que poner el logo de la empresa creadora. Otros web services gratis son públicos para que los usuarios se familiaricen con la

tecnología y luego regresen para consumir los servicios comprándolos, además no existe garantía en el nivel de servicio de los web services gratis en el Internet.

Otro mecanismo es el pago por uso que incluye un contador que registra el número de veces que se ha realizado una petición de uso del servicio; como muchos servicios, mientras más peticiones se compra, el costo por unidad es menor; este mecanismo requiere el prepago por uso y cuando el número de invocaciones a llegado al límite el proveedor y consumidor negocian una renovación.

El método que probablemente se convierta en el más popular es el de suscripción por cuotas dónde los consumidores pagan por el uso limitado de web services durante un período de tiempo, por ejemplo cada mes o cada año. Este método es beneficioso para el consumidor y para el proveedor que puede saber de antemano cuales serán sus ingresos, por supuesto que sería una desventaja si no se estima bien el uso del servicio de los consumidores.

El último mecanismo es el pago una sola vez por el tiempo de vida del servicio, el que puede ser útil en el caso de eventos que tienen un tiempo limitado de existencia como las compañías presidenciales o los juegos olímpicos.

Muchos otros métodos de pago pueden existir, especialmente entre socios de negocios y cadenas de proveedores donde el pago puede no solo ser de dinero.

1.2.8. PUBLICANDO WEB SERVICES

Las compañías que publican web services pueden tener diferentes modelos de su negocio y pueden utilizar los Web de acuerdo a sus necesidades entre los modelos de negocio podemos destacar tres categorías:

- Servicio a Consumidor (S2C – service-to-customer) Web Services
- Servicio a Negocio (S2B – service-to-business) Web Services
- Servicio a Empleado (S2E – service–to-employee) Web Services.

1.2.8.1. S2C Web Services

Son web services ofrecidos directamente a individuos para uso personal. Los agentes de web services listan en sus sitios Web muchos S2C web services tales como XMethods y SalCentral. Algunos ejemplos de Web Services que se pueden encontrar en estos sitios incluyen información de títulos de noticias, marcadores deportivos, datos de la bolsa de valores, información de tráfico, mapas, temperatura, otros realizan cambio de moneda o traductores, o permiten comunicarse con juegos interactivos en línea.

Muchos de estos Web Services son gratis y no son de gran utilidad. Uno de los más útiles quizá sea Passport de Microsoft.

Microsoft.NET Passport

En 1999, Microsoft empezó a correr .NET Passport, un simple servicio de autenticación que guarda la información del usuario y permite la autenticación automática para los sitios participantes, es decir no es necesario tener que ingresar el nombre de un usuario y contraseña en cada uno de los sitios que participan del .NET Passport. Cuando un usuario se registra en .NET Passport ya sea creando una cuenta de correo electrónico en Hotmail o a través del sitio Web de Microsoft .NET Passport (www.passport.com), el sistema asigna un identificador único para Passport (PUID – passport unique identifier), un perfil de usuario, información de la credencial y opcionalmente información para realizar compras (Express Parchase) que contiene datos como tarjeta de crédito y dirección. Cada usuario de Passport puede elegir una clave adicional de seguridad de cuatro dígitos, la que puede ser utilizada por los sitios Web como un método adicional de autenticación. Microsoft ha establecido que más de 200 millones de usuarios están registrados en Passport, este número no establece el número de personas que usan Passport ya que para cada usuario que registra una cuenta de correo en MSN o Hotmail, automáticamente se crea una cuenta de Passport. Este servicio fue uno de los primeros en utilizar la tecnología de los web services.

1.2.8.2. S2B Web Services

Los web services tienen la habilidad de comunicar a los negocios y compartir información sin importar la plataforma, al reconocer esto muchas compañías están desarrollando web services ha ser accedidos por otras compañías, lo que pueden simplificar significativamente los procesos como transacciones. La mayoría de negocios permite el acceso a sus Web Services solo a sus socios de negocio; tales servicios pueden resolver una amplia gama de necesidades de comunicación entre negocios. Por ejemplo al usar web Services para conectar una aplicación de comercio de un socio de negocios, un e-business puede vender los productos o servicios del negocio a través de su propio sitio web, uno de los pioneros en realizar este tipo de implementación fue Dollar Rent A Car Systems que desarrollo un web service que se enlazaba su sistema de reservaciones al sitio web de southwest Airlines Company, es decir al comprar los boletos de avión en SouthWest, los usuarios pueden hacer las reservaciones de autos desde la misma página web; asimismo podría otra empresa utilizar el mismo Web Service para proveer ese servicio y se evita la necesidad de crear otro servicio para cada socio de negocios.

Solución orientada a distribuidores de automóviles[1]

Ford utiliza una solución orientada a distribuidores de automóviles que se basa en la utilización de un web service que cumple las funciones de un integrador entre las agencias Ford, actualmente tiene cobertura para Guadalajara y México, fue creado para resolver la problemática en el duplicado de las tareas, inconsistencia e integridad de la información y registros administrativos deficientes, obteniendo beneficios en la eliminación de las tareas duplicadas, seguridad, confiabilidad y facilidad de auditoria interna, aumentó la productividad de los procesos de la distribuidora, reducción drástica de los esfuerzos necesarios para generar información confiable y oportuna

[1] CUAUHTÉMOC Freyre, Solución ERP orientada a distribuidores de automóviles http://www.ibm.com/mx/businesscenter/solutions/solution_erp2.phtml

1.2.8.3. S2E Web Services

Son web services específicamente diseñados para el uso de empleados, algunos sirven para entregar información a empleados, otros simplifican las interacciones entre empleados. Por ejemplo un web service podría permitir que una compañía envíe notificaciones a sus empleados sobre citas y reuniones, el web service accedería a la aplicación del empleado y modificaría el itinerario en su PDA o teléfono celular, una de las empresas que utiliza este tipo de web service es Merck Sharp & Dohme.

Merck Sharp & Dohme: mejora de procesos internos extendiendo Microsoft Excel con web service[1]

Merck Sharp & Dohme (Argentina) Inc., filial de Merck & Co., Inc. (también conocida como MSD), representante en Argentina de Merck & Co., Inc. es una compañía farmacéutica líder mundial en investigación, con más de 50 años en el país. En la Argentina comercializa una amplia variedad de medicamentos y vacunas, relacionadas con el asma, la aterosclerosis, glaucoma, hipertensión, artritis y dolor agudo, osteoporosis y SIDA.

En su constante búsqueda por incrementar su agilidad operativa, en el marco de un mercado altamente competitivo, la compañía decidió implementar web services a nivel regional, como paradigma tecnológico para la integración de aplicaciones, procesos, información y personas.

El primer caso en el que se desarrolló un piloto utilizando web services sería el referido a la carga de planillas creadas con Microsoft Excel en los sistemas centrales que corren en un servidor AS400, una problemática que se manifestaba en distintos procesos. En particular, se decidió comenzar con un caso puntual: la planilla de gastos.

El proceso de rendición de gastos era una tarea que requería mucho trabajo dedicado a la carga manual de los datos. Los usuarios completaban una

[1] Microsoft, http://www.microsoft.com/conosur/casosexito/caso/merck.asp

planilla Excel, la enviaban por email y luego en el área de finanzas se realizaba la carga manual de los datos en los sistemas centrales para su futura verificación, aprobación y generación del pago. Esta operatoria era muy tediosa y aumentaba la posibilidad de errores de tipeo al momento de la segunda carga. El proceso era lento y por ende, costoso.

Al utilizar web services basados en estándares, MSD Argentina logró exportar la información ingresada por lo usuarios en Microsoft Excel directamente en los sistemas centrales, lo que permite eliminar el tiempo y los riesgos de error que implican la carga manual de los datos.

En la siguiente tabla se muestra en resumen los diferentes tipos de web services que existen, cómo se muestra en la Tabla 1-1

Tabla 1-1 Tipos de Web Services

Servicio A Consumidor (S2C)	Web Service ofrecidos directamente a individuos para uso personal.
Servicio A Negocio (S2B)	Web Service que permite la comunicación e intercambio de información entre negocios.
Servicio A Empleado (S2E)	Web Service creados para dar algún servicio a los empleados de una organización.

Elaboración y Fuente: Los Autores

1.2.9. TECNOLOGÍAS UTILIZADAS EN LOS WEB SERVICES

1.2.9.1. SOAP

Simple Object Access Protocol (SOAP) es un protocolo ligero basado en XML para intercambiar información estructurada y con tipo. SOAP es el que permite usar la funcionalidad de máquinas remotas sin necesidad de saber algo específico sobre la máquina. XML es usado en los Web Services para representar datos, y los datos necesitan tener un esquema en común, para esto SOAP es el que define el esquema.

La especificación SOAP divide a un mensaje SOAP en cuatro partes, de ésta solo uno es obligatorio y es la envoltura SOAP que encapsula el mensaje. La segunda parte describe los tipos de datos definidos en la aplicación, la tercera parte describe un patrón de intercambio de mensaje de petición o respuesta llamado de procedimiento remoto (RPC Remote Procedure Call) y la cuarta parte de un mensaje SOAP define una conexión entre SOAP y HTTP.

Cada mensaje SOAP contiene un elemento envoltura compuesto por un elemento opcional la cabecera y un elemento requerido el cuerpo. La cabecera puede contener información sobre el mensaje, instrucciones de conversión para nodos que reciben el mensaje e información de seguridad. El cuerpo describe el propósito del mensaje SOAP y contiene la carga del mensaje tal como datos o instrucciones para la aplicación receptora.
En un intercambio de mensajes SOAP se procede de la siguiente forma:

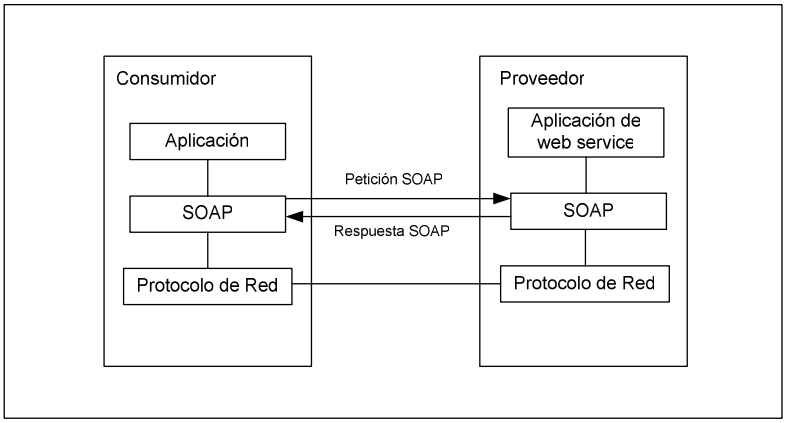
La aplicación del consumidor crea un mensaje SOAP que es una petición para invocar al web service provisto por el proveedor de servicio. El documento XML en el cuerpo del mensaje puede ser una petición SOAP RPC como es indicado en la descripción del servicio. El cliente SOAP interactúa con el protocolo de red apropiado como por ejemplo HTTP para enviar el mensaje de red.

Uno de los propósitos del mensaje SOAP es ejecutar un procedimiento a través de la red, es por eso que un mensaje SOAP enviado por la red hacia un web service es un RPC.

Cuando se entrega el mensaje al proveedor de servicio, el servidor SOAP enruta el mensaje hacia el web service y es el responsable de convertir el mensaje XML en objetos comprensibles por el lenguaje de programación utilizado donde se utilizan las reglas de codificación incluidas en el mensaje SOAP.
El web service es el responsable de procesar el mensaje y conceder una respuesta que es también un mensaje SOAP e igualmente se envía el mensaje por la red con el protocolo establecido.

Cuando la respuesta llega, el mensaje XML es convertido en objetos específicos del lenguaje de programación del consumidor y es presentado a la aplicación correspondiente.

Figura 1-2: Arquitectura SOAP



Elaboración y Fuente: Los Autores

1.2.9.1.1. Alternativas de SOAP

Algunas compañías han usado otra tecnología XML RPC en lugar de SOAP para algunas implementaciones. Según IBM el protocolo XML reemplazará a SOAP como el estándar en la industria como protocolo de mensajería, para esto el W3C tiene que lanzar el protocolo XML como estándar. Además de utilizar SOAP o XML, algunas empresas pueden desarrollar sus propias alternativas basadas en XML para la comunicación de mensajes, pero la desventaja es que se requiere gran experiencia en esos desarrollos y los mayores vendedores de software han adoptado productos SOAP que ofrecen mejores características y funcionalidad.

Algunas aplicaciones para implementar SOAP en sistemas empresariales son: SOAP Toolkit de Microsoft, Web Services Toolkit de IBM y Axis de Apache.

1.2.9.2. WSDL

Es por la descripción del servicio que un proveedor de servicio comunica todas las especificaciones para que el consumidor del servicio invoque al web service. Esta descripción del servicio es parte clave en la estructura de los web services poco acoplada. Web Services Description Language (WSDL) es un

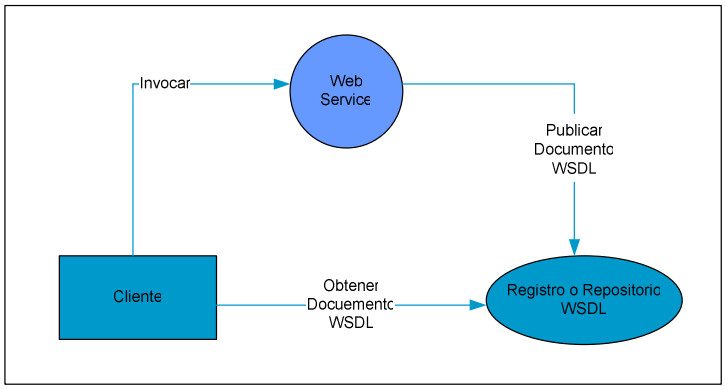
lenguaje basado en XML que es utilizador por la aplicación cliente para obtener información técnica del web service con la cual se puedan comunicar.

Muchos web services publican el documento WSDL por el Internet que contiene definiciones que describen al web wervice en formato XML. El documento WSDL especifica las capacidades que tiene el servicio, su ubicación en la web e instrucciones sobre como acceder al web service. Un documento WSDL define cual va ser la estructura de los mensajes que el web service puede enviar o recibir.

Cuando un web service es publicado, el administrador publica un enlace hacia el documento WSDL del web service en alguno de los registros XML u otro repositorio, entonces, el archivo WSDL está disponible cuando una aplicación busca el registro y logra ubicar el Web Service. El cliente accede a documento WSDL para obtener la información sobre el web service y para crear los mensajes SOAP con la correcta estructura. Utilizando esta información, la aplicación cliente invoca al Web service.

El documento WSDL puede estar publicado ya sea en un repositorio o en cualquier otro lugar ya que es un URL que direcciona hacia este documento.

Figura 1-3: Web Services Description Language



Elaboración y Fuente: Los Autores

1.2.9.3. UDDI

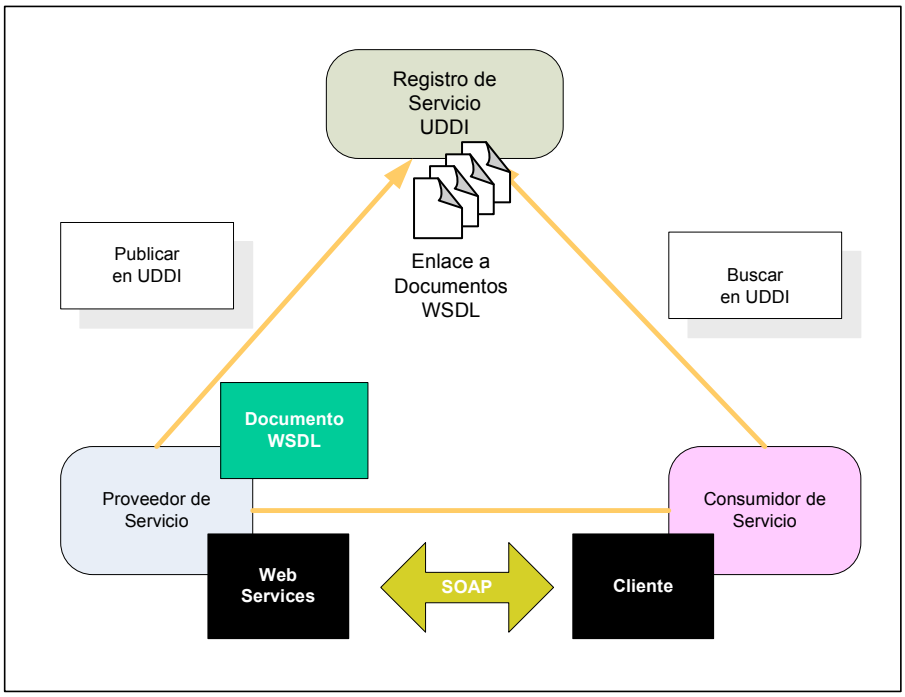
Universal Description, Discovery and Integration (UDDI) permite la creación de registros web service para ser buscados por consumidores. Las

especificaciones UDDI definen la manera para publicar y descubrir información sobre los web services. Descubrimiento (Discovery) es el proceso de ubicar un web service por medio de registros. Usando un registro, una empresa puede localizar todos los servicios disponibles requeridos y puede comparar entre los servicios ofrecidos para ubicar el que llene sus expectativas.

La información de UDDI se aloja en nodos de operador, empresas que se han comprometido a ejecutar un nodo público conforme a la especificación que rige el consorcio UDDI.org. En la actualidad existen dos nodos públicos que se ajustan a la versión 1 de la especificación UDDI: Microsoft aloja uno e IBM el otro. Hewlett Packard se ha comprometido a alojar un nodo bajo la versión 2 de la especificación. Los operadores del host deben replicar datos entre ellos a través de un canal seguro, para conseguir la redundancia de la información en el registro UDDI. Se pueden publicar los datos en un nodo y descubrirlos en otro tras la réplica. Actualmente, la réplica se produce cada 24 horas. En el futuro, este intervalo entre réplicas se reducirá, ya que habrá más aplicaciones que dependan de los datos de UDDI[1].

[1] http://www.desarrolloweb.com/articulos/1589.php

Figura 1-4: Interacción de Protocolos



Elaboración y Fuente: Los Autores

1.2.9.4. Otras Tecnologías

ebXML, define su propia estructura de registro por medio del cual los consumidores de servicios pueden acceder a documentos XML que contienen información sobre los proveedores de servicio. El registro permite que las compañías inicien acuerdos de negocios y realicen transacciones.

WS-Inspection, es una tecnología desarrollada por Microsoft e IBM que define como una aplicación cliente puede localizar descripciones de WS que residen en un servidor de WS. Un documento WS-Inspection es mantenido por el proveedor de servicio y contiene referencias hacia los documentos de descripción de WS como WSDL en el servicio. WS. Inspection entonces sirve cuando desarrolladores conocen el servicio que contiene los WS que pueden utilizar.

CAPITULO 2. FUNDAMENTO TEÓRICO

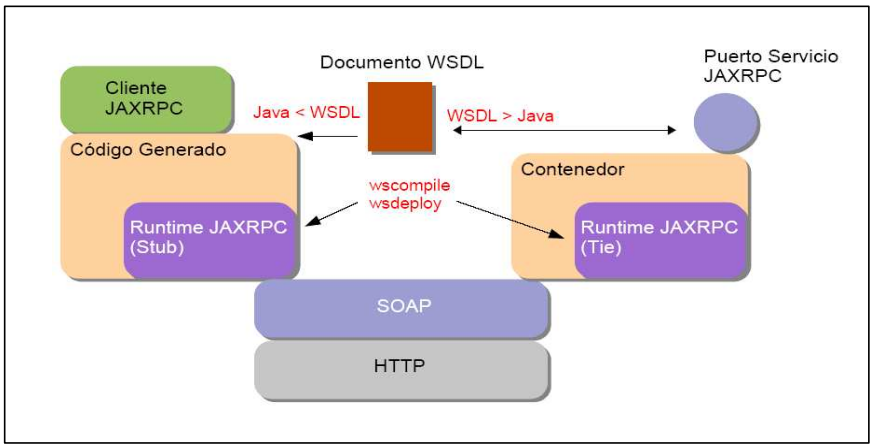
2.1. WEB SERVICES IMPLEMENTADOS EN PLATAFORMA JAVA

2.1.1. ARQUITECTURA DE WEB SERVICES EN J2EE[1]: JAXRPC

En el entorno J2EE los servicios web se construyen sobre JAX-RPC (Java API for XML-based RPC). Este es un API para construir web service y clientes que utilizan llamadas a procedimientos remotos (RPC) y XML.[2]

En JAX-RPC, se entiende por llamada a un procedimiento remoto representado por un protocolo basado en XML, como SOAP. La especificación SOAP define la envoltura estructural, las reglas de codificación y la convención para representar llamadas y respuestas a procedimientos remotos. Estas llamadas y respuestas se transmiten como mensajes SOAP (ficheros XML) sobre HTTP.

Figura 2-1: Arquitectura de Web Services en J2EE:JAXRPC



Elaboración y Fuente: Héctor Jiménez Desarrollo de web services, http://www.amereiaf.org.mx/cursos/VisionWebServicesJava.pdf

[1] Java 2 Enterprise Edition

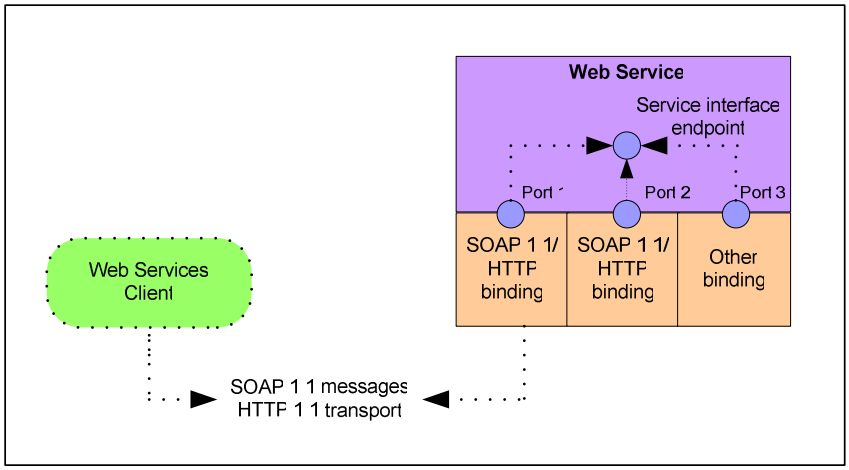
[2] http://www.programacion.com/java/tutorial/jap_aplic_jboss/10

2.1.1.1. Servicio, Puertos y Bindings

El Service endpoint interface define las operaciones que ofrece el servicio, mientras que el acceso al servicio se logra mediante un binding a través de un port.

Para crear un web service desde cero se debe escribir una interfaz Java con un método para cada operación, teniéndose que escribir una clase que implemente dicha interfaz.

Figura 2-2: Servicios, puertos y bindings



Elaboración y Fuente: Los Autores

2.1.2. ARQUITECTURA DE WEB SERVICES EN: AXIS

Axis es un motor SOAP, un framework para construir tanto la parte servidor como cliente de un Servicios Web, Axis incluye:

- Incluye un servidor
- Un servlet que se integra con motores de servlets como Tomcat
- Soporte extensivo del estándar Web service description language
- Una herramienta para generar clases java a partir de WSDL

Axis está disponible en el fichero axis.jar que implementa la API JAX-RPC API declarada en los ficheros JAR jaxrpc.jar y saaj.jar. Requiere varias bibliotecas de ayudar, para log, procesamiento WSDL e introspección.

2.1.2.1. Handlers y el Message Path en Axis

El funcionamiento lógico de AXIS es invocar a una serie de Handlers cada uno en orden, el orden particular es determinado por dos factores:

- Configuración del deploy
- Si el motor es un cliente o un servidor

El objeto que se pasa en cada invocación del Handler es un MessageContext, un MessageContext es una estructura que contiene varias piezas importantes:

- Un mensaje de petición
- Un mensaje de respuesta
- Un paquete de características

Existen dos maneras básicas de invocar a AXIS:

- Como servidor, un TransportListener creará un MessageContext e invocará al Axis.
- Como cliente, el código de uso (ayudado generalmente por el modelo de programación del cliente del Axis) generará un MessageContext e invocará al Axis

En cualquier caso, el trabajo del framework de Axis es simplemente pasar el MessageContext que resulta a través del sistema configurado de Handlers, que tiene una oportunidad de hacer lo que se diseña para hacer con el MessageContext.

2.1.2.1.1. Message Path en el Servidor

Los cilindros pequeños representan a un Handler, los cilindros más grandes representan los Chains (colecciones de Handler). Cómo se muestra en la figura 2-3

Un mensaje llega (de una cierta manera protocolo-específico) un TransportListener. En este caso, asume que el Listener es un servlet del HTTP. Es el trabajo del Listener empaquetar los datos del protocolo específico en un objeto del Message, y puesto el mensaje en un MessageContext. El MessageContext también es cargado con varias características por el Listener.

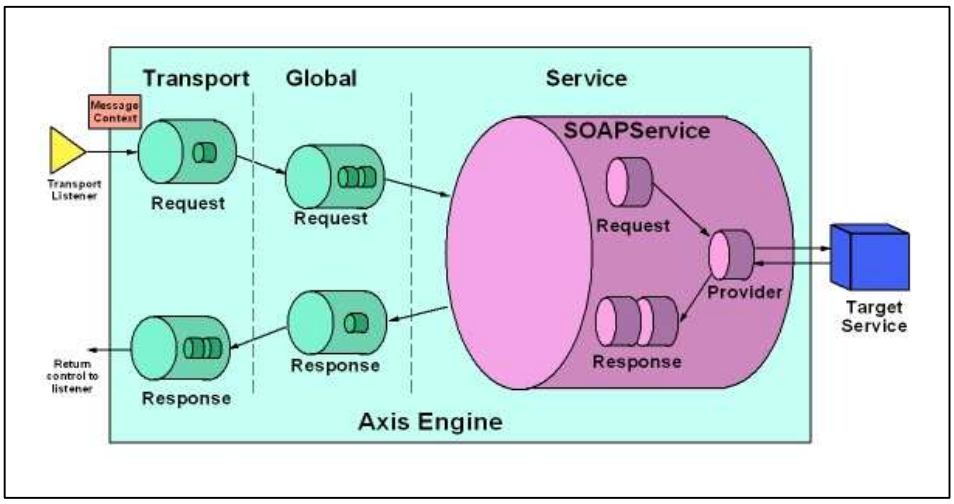
El primer trabajo del motor de Axis es mirar el nombre del transporte. El transport es un objeto que contiene una cadena de petición, una cadena de respuesta o ambas. Un Chain es un Handler que consiste en una secuencia de handlers que se invocan alternadamente. Si existe un chain de petición del transport, será invocada, pasando el MessageContext en el método de invocación. Esto dará lugar a llamar a todos los tratantes especificados en la configuración de la chain de petición.

Después de que el handler de la petición del transport, el motor localiza un chain de petición global, si está configurado, después invoca a cualquier handler especificado.

Los servicios en Axis son típicamente de la clase *SOAPService* que pueden contener los chain de petición, de respuesta y un provider, que es simplemente un transport responsable de poner la lógica final de la ejecución del servicio.

Para las peticiones RPC, el provider es la clase de RPC.Provider. Este es otro transport, que al momento de ser invocado llama a un objeto backend de java determinada por el parámetro del *className* especificado en el tiempo del despliegue. Utiliza la convención del RPC de SOAP para determinar el método para llamar, y se cerciora de los tipos XML.

Figura 2-3: Message Path en el Servidor



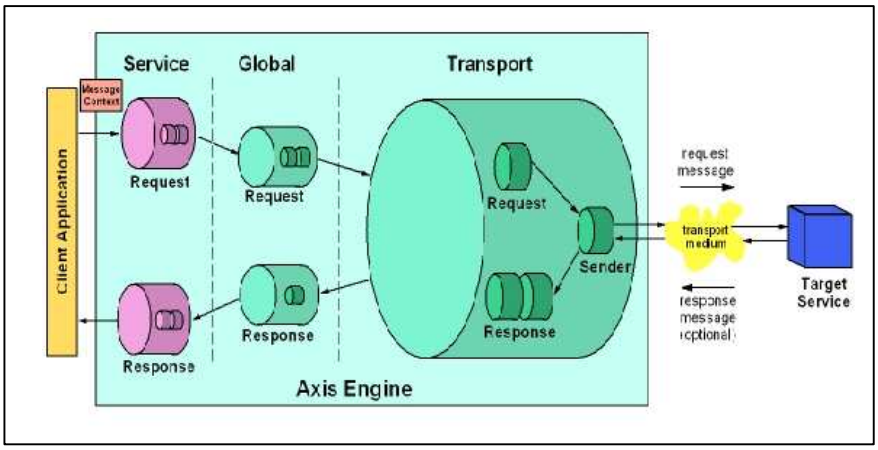
Elaboración y Fuente: Axis Architecture Guide, http://ws.apache.org/axis/java/architecture-guide.html

2.1.2.1.2. Message Path en el Cliente

El lado del cliente no es proveer, puesto que el servicio esta siendo proveído desde un nodo remoto. Las cadenas de petición y respuesta del servicio realizan el proceso servicio-específico del mensaje de petición en su salida del sistema, y también del mensaje de respuesta.

Después de la cadena de petición del servicio, se invoca la cadena de petición global, si esta existe, seguido por el transporte. El remitente del transporte, es un manejador especial cuyo trabajo es realizar cualquier operación protocolo-específico necesario para obtener el mensaje desde el servidor. La respuesta se pone en el campo del responseMessage del MessageContext, y el MessageContext después propaga a través de las cadenas de respuesta primero al transporte, luego el global, y finalmente el servicio.

Figura 2-4: Message Path en el Cliente



Elaboración y Fuente: Axis Architecture Guide, http://ws.apache.org/axis/java/architecture-guide.html

2.1.2.2. Desarrollo del Web Service Usando WSDL2Java de Axis

Como hemos visto en la introducción teórica de la práctica, el lenguaje WSDL permite describir la interfaz que un Servicio Web ofrece a las aplicaciones remotas. Una situación frecuente cuando se trabaja con servicios Web, es que se desarrolle únicamente una parte del sistema: sólo el cliente o sólo el servidor, siendo terceras partes las que desarrollan la otra parte. Para que esto sea posible, es interesante poder disponer de la especificación en WSDL del servicio, puesto que si estamos desarrollando el cliente nos interesará saber

qué operaciones permite el servicio y si desarrollamos el servidor, deberemos publicar nuestro WSDL para facilitar a terceras partes el desarrollo de sus clientes

Axis proporciona herramientas de apoyo para generar código java a partir de la descripción WSDL de un servicio, esta herramienta consiste en un compilador denominado WSDL2java que permite generar automáticamente stubs y otros ficheros muy útiles para la codificación de clientes.

Básicamente el compilador genera por cada tipo una clase Java, además de una clase adicional similar al manejador si el tipo utiliza un parámetro de entrada/salida. Por cada entrada wsdl:portType del fichero WSDL se genera una interfaz java. Por cada entrada wsdl:binding del WSDL se genera un stub, por cada entrada wsdl:service se genera una interfaz java y un objeto locutor que nos permitirá obtener instancias de stubs que se utilizarán para invocar los servicios remotos.

2.1.2.2.1. Binding en la parte servidora: skeleton

- Stub es el Proxy en el cliente de un service Web
- El Skeleton es el Proxy del Servicio Web en el servidor

Tabla 2-1: Binding en la parte servidora Skeleton

cláusulas WSDL	clases generadas Java
Para cada binding	Una clase skeleton
	Una clase de implementación
Para todos los servicios	Un archivo deploy.wsdd
	Un archivo undeploy.wsdd

Elaboración y Fuente: Los Autores

2.1.2.2.2. Los Skeleton

- Intermediario entre el motor de Axis y la implementación del servicio
- Su nombre es el binding con el sufijo Skeleton

2.2. WEB SERVICES IMPLEMENTADOS EN PLATAFORMA .NET

2.2.1. ARQUITECTURA DE WEB SERVICES: .NET 2003

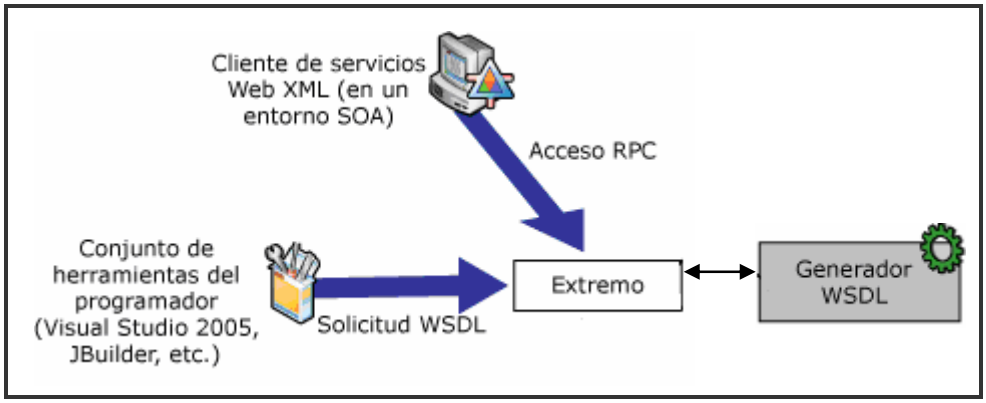
Visual Studio .NET 2003 es un conjunto de tecnologías de software para conectar su mundo de información, gente, sistemas y dispositivos. Permite un nivel sin precedente de integración de software a través del uso de servicios web XML: pequeños, directos, bloques de aplicaciones construidas que se conectan con cada uno, así como a otras aplicaciones grandes vía Internet.

Proporciona la habilidad de construir, hospedar e implementar de una manera rápida y confiable, utiliza soluciones seguras usando servicios web XML, la plataforma provee una suite de herramientas de desarrollo, aplicaciones clientes, servicios web XML y servidores necesarios para intervenir en este mundo conectado.

Para utilizar los servicios web XML, debe establecerse un extremo HTTP en el servidor. Este extremo es básicamente la puerta de enlace a través de la cual los clientes basados en HTTP pueden enviar consultar al servidor. Tras establecer un extremo HTTP, las funciones definidas por los usuarios pueden agregarse o hacer que estén disponibles para los usuarios del extremo. Cuando se habitan, los procedimientos y las funciones se especifican como métodos web.

Estos servicios web pueden describirse utilizando el formato WSDL y se devuelve a los clientes SOAP para cualquier extremo HTTP en el que WSDL esté habilitado.

Figura 2-5: Escenario web services .NET



Elaboración: Los Autores
Fuente: http://msdn2.microsoft.com/es-es/library/ms188266.aspx

2.2.2. ARQUITECTURA DE WEB SERVICES: VISUAL STUDIO .NET 2005

Visual Studio 2005 es un conjunto completo de herramientas de desarrollo para la generación de aplicaciones Web ASP.NET, servicios Web XML, aplicaciones de escritorio y aplicaciones móviles. Visual Basic, Visual C++, Visual C# y Visual J# utilizan el mismo entorno de desarrollo integrado (IDE), que les permite compartir herramientas y facilita la creación de soluciones en varios lenguajes, Asimismo, dichos lenguajes aprovechan las funciones de .NET framework, que ofrece acceso a tecnologías clave para simplificar el desarrollo de aplicaciones web ASP y servicios web XML.

Los servicios web XML son aplicaciones que pueden recibir solicitudes y datos mediante XML a través de HTTP, no están ligados a una tecnología de componentes particular o a una convención de llamada de objetos y, por tanto, se puede obtener acceso a ellos mediante cualquier lenguaje, modelo de componente o sistema operativo.

El lenguaje de marcado extensible (XML) proporciona un método para describir datos estructurados, XML es un subconjunto de SGML optimizado para la entrega a través del web. El Consorcio World Wide Web (W3C) define los estándares de XML para que los datos estructurados sean uniformes e independientes de las aplicaciones. Visual Studio es totalmente compatible con el código XML e incluye el diseñador XML para facilitar la edición de XL y la creación de esquemas XML.

.NET framework es un entorno multilenguaje que permite generar, implantar y ejecutar aplicaciones y servicios web XML, consta de tres partes principales:

2.2.2.1. Infraestructura de Web Services

- XML (eXtensible Markup Language)

Formato universal para documentos estructurados y datos en la Web administrado por W3C

- UDDI (Universal Description, Discovery and Integration)

Servicio de directorio que permite publicar y/o describir servicios Web

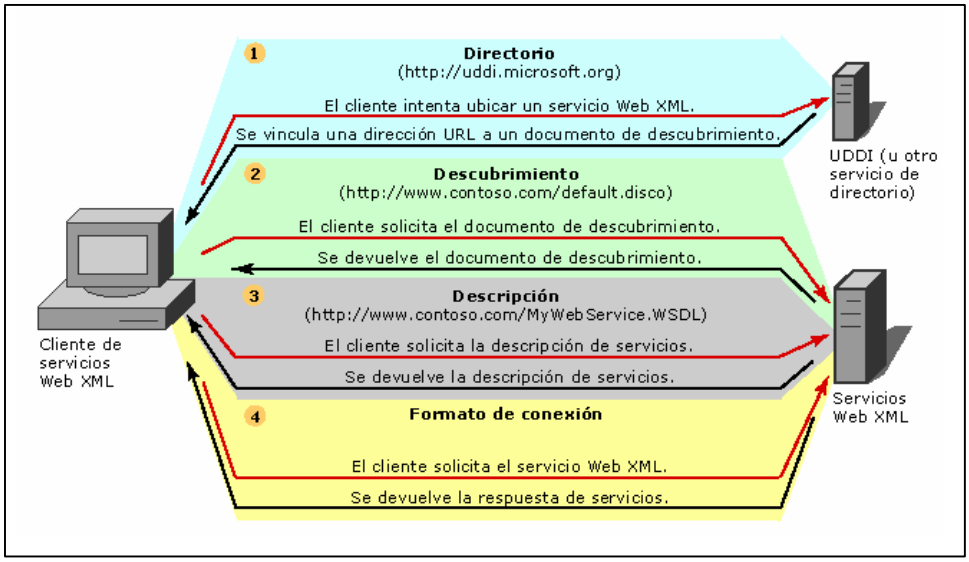
- WSDL (Web Services Description Language)

Una gramática basada en XML que permite describir las capacidades de un servicio Web

- SOAP (Simple Object Access Protocol)

Protocolo ligero para el intercambio de información en entornos distribuidos y descentralizados administrado por W3C

Figura 2-6 Infraestructura de Servicios Web



Elaboración y Fuente: Guía del desarrollador de .NET Framework, http://msdn2.microsoft.com/es-es/library/sd5s0c6d.aspx

2.2.2.2. Desarrollo del Web Services usando visual Studio .NET 2005

- Generar un proxy del servicio Web
- Publicar el contrato WSDL

2.2.2.3. Invocar un Web Service

- Identificar el objeto o URI del servicio a usar.

```
http://localhost/MathService/Service1.asmx
```

- Ejecutar Wsdl.exe en la URI del servicio

```
wsdl http://localhost/MathService/Service1.asmx?wsdl
```

- Escribir el código cliente que llama al objeto remoto

```
Service1 salesTaxService = new Service1 ();
float totalAmount = salesTaxService.SalesTax(amount,taxRate);
```

- Construir el cliente y el Proxy

```
csc salestaxclient.cs service1.cs
```

2.2.3. DEFINICIÓN

El Benchmark es una técnica utilizada para medir el rendimiento de un sistema o parte de el, frecuentemente en comparación con algún parámetro de referencia.

Para medir el rendimiento se emplean programas informáticos, estos no sólo pueden ayudar en la comparación de diferentes sistemas sino que además son capaces de evaluar las prestaciones de un equipo con diferentes configuraciones de Software y Hardware.

2.3. ESTADÍGRAFOS

2.3.1. TEORÍA DE ESTADÍGRAFOS

Estadística es aquella disciplina científica que se preocupa de obtener, ordenar y sistematizar información de tal manera que esta se transforme en un insumo útil para la toma de decisiones.

Se dice Población o Universo a la colección de toda posible información que caracteriza a un fenómeno determinado, se denomina Muestra a un subconjunto representativo de las observaciones que componen una población.

Atributo es una característica no medible o cualitativa de un objeto en tanto que una variable es cualquier característica de un objeto de estudio susceptible de ser cuantificada.

Debido a que prácticamente cualquier característica puede ser medida la definición de variable es muy extensa por lo que se ha clasificado en diferentes tipos, de la siguiente manera:

2.3.1.1. Variables discretas y continuas

2.3.1.1.1. Variable discreta

Son aquellas que no admite valores intermedios entre los distintos valores de la variable.

2.3.1.1.2. Variables continúas

Estas variables, admiten valores intermedios entre los distintos valores de la variable.

2.3.1.2. Las variables y el tiempo.

Si una variable determinada no se ve influenciada en lo absoluto por el transcurso del tiempo, se dice que dicha variable tiene un carácter atemporal y no ordinal.

Cuando una variable modifica su comportamiento en momentos distintos, dicha variable tiene un carácter temporal u ordinal

2.3.1.3. Estadígrafos

Se denominan estadígrafos a los valores que cuantifican características de un grupo de datos, tales como la dispersión de los mismos, la tendencia a concentrarse alrededor de un cierto punto, etc.

Son valores que ilustran (grafican) una característica saliente de un grupo de datos. Por lo general, la información que se extrae del sistema aislado para su estudio es muy voluminosa y el manejo de grandes cantidades de números se torna difícil, engorroso y con grandes posibilidades de cometer equivocaciones.

Considerando la *nube* de datos, en unas pocas *gotas*, ricas en información, se puede facilitar enormemente el manejo de las cantidades ingentes de datos. Y además, la transmisión de los resultados obtenidos, a otros investigadores se facilita y resume.

2.3.2. ESTADÍGRAFOS DE POSICIÓN O DE TENDENCIA CENTRAL

Los estadígrafos de tendencia central alrededor de cuál valor se agrupan los datos obtenidos.

- La media
- La mediana
- La moda
- La media geométrica

Si se graficaran todos los valores, esa especie de nube de puntos se distribuiría alrededor de una zona central, donde se ubican estos índices.

2.3.2.1. Media Aritmética.

La media aritmética, promedio o simplemente media es el estadígrafo de más común utilización. De todos los estadígrafos de posición la media es el más estable, si se calcula para diferentes muestras de la misma población.

La fórmula de común uso para el cálculo de la media es la suma de todos los valores de la variable dividida para el total de observaciones.

2.3.2.2. Mediana

La mediana o valor mediano de un conjunto de datos corresponde a un valor de la variable que supera a lo sumo a la mitad de las observaciones y que a la vez es superada a lo sumo por la otra mitad, una vez ordenados los datos en forma creciente o decreciente.

Por definición la mediana divide a la población o muestra que se está investigando en dos partes iguales, con igual número de observaciones cada una.

2.3.2.2.1. Fráctiles.

Hay diferentes tipos de fractiles, pero por lo general son valores que dejan por debajo de el una cierta fracción de datos ordenados en forma creciente y el resto por encima, cuando la fracción es la mitad se trata de la mediana. Las más comunes son:

- Quartiles

Las fracciones son cuartas partes del total, el primero quartil Q1 deja el 25% de los valores por debajo, el segundo quartil Q2 es igual a la mediana y el tercer quartil Q3 deja el 75% de los valores por debajo.

- Deciles

Las fracciones son décimas del total, el primer decil D1 deja el 10% de los valores por debajo y el resto por encima, y el quinto decil D5 es la mediana.

- Percentiles

Las fracciones son centésimas partes del total; así el percentil catorce P14 deja el 14% de los valores por debajo, y el percentil cincuenta P50 es la mediana.

2.3.2.3. Moda

La moda de un conjunto de observaciones es aquel valor de la variable que se repite más veces; es decir, es el valor de la variable que tiene mayor frecuencia absoluta.

2.3.2.4. Media Geométrica

La media geométrica de un conjunto de n observaciones se define como la raíz de índice n del producto de observaciones.

La media geométrica es un estadígrafo de posición que se utiliza para describir las tendencias centrales en variables que tienen tasas de crecimiento relativamente constantes.

2.3.3. ESTADIGRAFOS DE DISPERSIÓN

Los estadígrafos de dispersion indican que tan esparcidos están los datos obtenidos.

- Rango
- Desvío estándar
- Varianza

Son los índices más conocidos de este grupo. Se debe definir la dispersión de los datos respecto de algo. Conviene hacerlo respecto de la media aritmética (como lo hace el desvío estándar) porque minimiza los cuadrados de los desvíos.

2.3.3.1. Rango

Se define como la diferencia entre un valor máximo y mínimo de un grupo de datos. De fácil cálculo y compresión tiene la desventaja de ser la media más grosera de la dispersión. Dos grupos de datos, con muy distinta dispersión pueden llegar a tener rangos similares, uno de ellos puede tener 99% de los valores junto al mínimo y el otro 99% junto al máximo, pero al tener extremos iguales sus rangos resultarían iguales, a pesar de ser tan disímiles intrínsicamente.

2.3.3.2. Desviación

Las desviaciones son valores que indican cuanto se aleja un determinado valor de los valores de la variable; es decir, es la diferencia entre cada valor

observado y uno determinado, que puede ser la media aritmética, la mediana o un origen de trabajo elegido arbitrariamente.

2.3.3.2.1. Desviación media

Es la forma como se dispersan los datos con respecto a la media aritmética.

2.3.3.2.2. Desviación mediana

Se obtiene sumando las diferencias absolutas entre los valores de la variable y la mediana y dividiendo luego para el número de datos.

2.3.3.3. Desviación cuadrática media o varianza

Es un parámetro de distribución normal que mide la dispersión de los desvíos de la media.

Expresa la dispersión en unidades distintas a las de la variable, mientras mayor es la dispersión de las observaciones mayor es la magnitud de sus desviaciones respecto a la media y por consiguiente más alto el valor numérico de la varianza.

2.3.3.3.1. Desviación Típica o estándar

Este estadígrafo expresa en forma más real los resultados de la varianza, se la obtiene extrayéndole la raíz cuadrada a la varianza.

CAPITULO 3. DESARROLLO DEL BENCHMARK

Este capítulo se lo realizará utilizando el Proceso Unificado, acoplado a la realidad de la herramienta de pruebas a construir.

3.1. CAPTURA DE REQUERIMIENTOS DEL BENCHMARK

3.1.1. INTRODUCCION

La herramienta que se construirá para realizar el estudio comparativo planteado, parte de la existencia de dos web services que cumplen funcionalidades similares, éstos se encuentran desarrollados en las plataformas .NET y Java respectivamente.

Actualmente, las plataformas Java y .NET ofrecen facilidades similares en la creación y uso de web services, motivo por el cual las hemos tomado en cuenta para nuestro estudio.

Hemos considerado un web service básico publicado en un servidor de aplicaciones, el web service es invocado con diferentes tipos y tamaños de argumentos en ambas plataformas, lo cual nos permitirá comparar su rendimiento.

Entiéndase como Benchmark la herramienta a construirse, la que nos permitirá medir el rendimiento de varios tipos de llamadas a los web services, las que detallaremos a continuación:

3.1.1.1. Respuesta Vacío

Envía y recibe un mensaje vacío. Con esto se probará el rendimiento de la infraestructura del web service.

3.1.1.2. Respuesta Estructura

Envía y recibe un arreglo de tamaño n, cada elemento es una estructura compuesta de tres tipos de datos: un entero, un flotante y un string.

3.1.1.3. Respuesta Lista Enlazada

Envía y recibe una lista enlazada de n elementos, cada elemento es una estructura como la que se define en Respuesta Estructura.

3.1.1.4. Respuesta Sintético

Envía y recibe múltiples parámetros de diferentes tipos: String y arreglo de bytes de 1 KB de tamaño.

3.1.1.5. Respuesta Orden

Este método emula el funcionamiento de una aplicación transaccional en la que se tiene una orden por cliente, ésta a su vez tiene ítems, y su respectiva dirección. Este proceso se lleva a cabo tomando como entradas el ID de la orden y del cliente, devolviendo una orden generada randómicamente.

La técnica inmediata para identificar los requisitos del Benchmark se basa en los casos de uso. Estos casos de uso capturan tanto los requisitos funcionales como los no funcionales que son específicos de cada caso de uso.

3.1.2. REQUERIMIENTOS

El modelo de casos de uso captura todos los requisitos funcionales de la herramienta, definiremos un caso de uso de la siguiente manera: “Un caso de uso especifica una secuencia de acciones, incluyendo variantes, que el sistema puede llevar a cabo y que generan un resultado que es observado por un actor”[4].

A continuación se listan los requerimientos funcionales que la herramienta de pruebas cumplirá:

- Medir el Número de Peticiones Atendidas.
- Calcular el Tiempo de Respuesta de las peticiones atendidas en milisegundos.
- Calcular la Disponibilidad de los web services.
- Medir el Porcentaje de ocupación de CPU.

- Medir el Porcentaje de ocupación de Memoria.
- Obtener Resultados del Número de Peticiones Atendidas.
- Obtener Resultados del Cálculo Tiempo de Respuesta de las peticiones atendidas.
- Obtener Resultados del Cálculo de la Disponibilidad de los web services.
- Obtener Resultados del Porcentaje de ocupación de CPU.
- Obtener Resultados del Porcentaje de ocupación de Memoria.

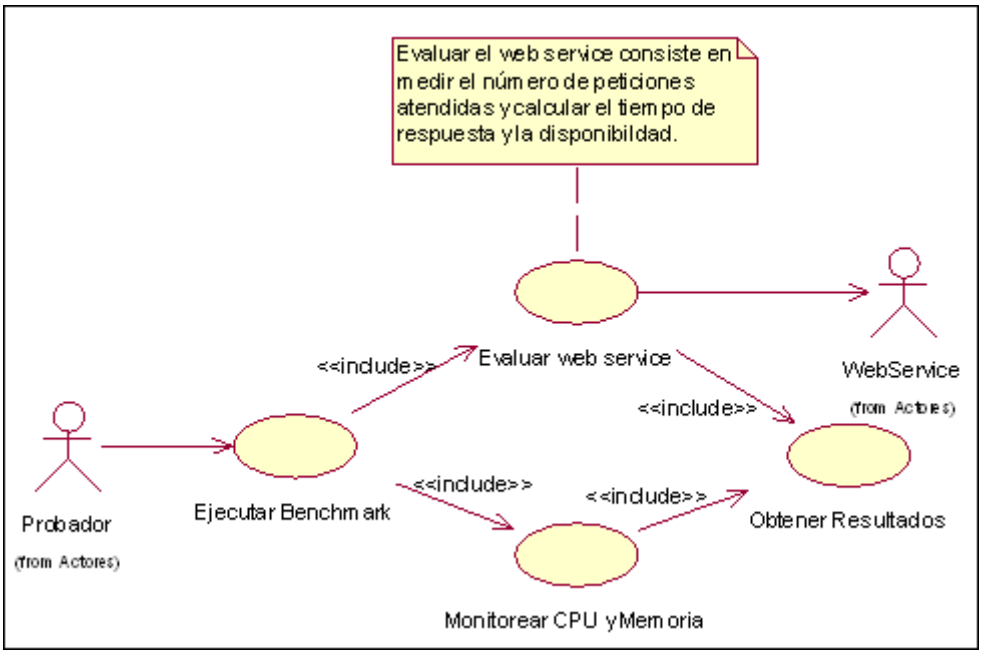
3.1.2.1.1. Diagrama de Casos de Uso y Descripción

Se han identificado cuatro servicios que proveerá la construcción del Benchmark para realizar un estudio comparativo de web services desarrollados en Java y .Net, los mismos que se representan como casos de uso.

- Ejecutar Benchmark.
- Evaluar web service.
- Monitorear CPU y Memoria
- Obtener Resultados.

3.1.2.1.2. Diagrama de Casos de Uso

Figura 3-1 Diagrama de Casos de Uso



Elaboración y Fuente: Los Autores

3.1.2.1.2.1. Actores

Probador:

El Probador representa a la persona encargada de ejecutar el Benchmark con el fin de obtener los resultados de las mediciones y los cálculos propuestos anteriormente.

WebService:

El WebService representa al web service que será evaluado por el Benchmark.

3.1.2.1.2.2. Ejecutar Benchmark

Tabla 3-1 Caso de Uso Ejecutar Benchmark

Descripción	El objetivo de este caso de uso es permitir al Probador empezar la ejecución del Benchmark para evaluar el web service y monitorear CPU y Memoria.	
Precondiciones	El Probador ha establecido los parámetros con los que se harán las pruebas de evaluación de los web services hechos en .NET y Java.	
Secuencia Normal	**Paso**	**Acción**
	1.	Los parámetros de prueba son cargados al Benchmark.
Poscondiciones	No existen poscondiciones.	
Excepciones	**Paso**	**Acción**
	1.	Si existe algún error en los parámetros se presenta el respectivo mensaje de error.
	2.	La ejecución del Benchmark finaliza inmediatamente.

Elaboración y Fuente: Los Autores

3.1.2.1.2.3. Evaluar web service

Tabla 3-2 Caso de Uso Evaluar web service

Descripción	El objetivo de este caso de uso es medir y calcular los índices del web service.	
Precondiciones	El Benchmark debe estar en ejecución.	
Secuencia Normal	**Paso**	**Acción**
	1.	El Benchmark invoca a los métodos del web service correspondiente al indicado en los parámetros (puede ser el que está construido en la plataforma .NET o Java).
	2.	Mide el número de peticiones atendidas por el web service (ver Glosario en la sección 3.1.2.2).
	3.	Calcula el tiempo de respuesta del web service (ver Glosario en la sección 3.1.2.2).
	4.	Calcula la disponibilidad del web service (ver Glosario en la sección 3.1.2.2).
Poscondiciones	No existen poscondiciones.	
Excepciones	**Paso**	**Acción**
	1.	Si el web service no pudo ser invocado se presenta el error correspondiente.
	2.	La ejecución del Benchmark finaliza inmediatamente.
Observaciones	Este caso de uso se ejecuta cuando una vez iniciado el caso de uso Monitorear CPU y Memoria.	

Elaboración y Fuente: Los Autores

3.1.2.1.2.4. Monitorear CPU y Memoria

Tabla 3-3 Caso de Uso Medir Porcentaje de Uso de Memoria

Descripción	El objetivo de este caso de uso es monitorear CPU y Memoria en el transcurso de la prueba.	
Precondiciones	El Benchmark debe estar en ejecución.	
Secuencia Normal	**Paso**	**Acción**
	1.	Se miden los contadores para CPU: USER_CPU, IDLE_CPU, TOTAL_CPU y para memoria: USED_MEMORY en forma simultánea (ver Glosario en la sección 3.1.2.2).
Poscondiciones	No existen poscondiciones.	
Excepciones	**Paso**	**Acción**
	1.	Si existe algún error durante el monitoreo, este se detiene.
	2.	La ejecución del Benchmark finaliza inmediatamente.
Observaciones	Este caso de uso termina una vez finalizado la ejecución del caso de uso Evaluar web service.	

Elaboración y Fuente: Los Autores

3.1.2.1.2.5. Obtener Resultados

Tabla 3-4 Caso de Uso Obtener Resultados

Descripción	El objetivo de este caso de uso es proveer al usuario los datos medidos ó calculados durante la ejecución del Benchmark.	
Precondiciones	Los casos de uso Evaluar web service y Monitorear CPU y Memoria debieron ser ejecutados.	
Secuencia Normal	**Paso**	**Acción**
	1.	Se recolectan los datos obtenidos por los casos de uso Evaluar web service y Monitorear CPU y Memoria.
Poscondiciones	Finaliza la ejecución del Benchmark.	
Excepciones	**Paso**	**Acción**
	1.	En caso de existir un error, se presenta el mensaje correspondiente.
	2.	La ejecución del Benchmark finaliza inmediatamente.

Elaboración y Fuente: Los Autores

Los diagramas de estados de cada uno de los casos de uso se encuentran el Anexo 1

3.1.2.2. Glosario

Tabla 3-5 Glosario de Términos

Término	Descripción
Agente	Es el simulador de la carga que recibe el web service.
Disponibilidad	Cantidad que representa el número de peticiones atendidas por unidad de tiempo, se calcula mediante la división de número total de peticiones atendidas y el tiempo de estabilización.

Término	Descripción
Endpoint	Dirección en donde se encuentra alojado el web service a evaluar.
IDLE_CPU	“Porcentaje de tiempo en el que el procesador se encuentra en espera en un intervalo de tiempo”[5].
Índice	Variable que permite evaluar el rendimiento del web service, los índices son: número de peticiones atendidas, tiempo de respuesta y disponibilidad.
Número de bytes	Cantidad que representa el tamaño del arreglo de bytes, utilizado dentro del método Respuesta Sintético del web service.
Número de Peticiones Atendidas	Número total de llamadas atendidas por el web service durante la prueba.
Parámetros	Son las variables con las que se realizará la prueba, las mismas que son: agentes, tiempo ascenso, estabilización descenso, endpoint, Respuesta Vacío, Respuesta Estructura, Respuesta Lista Enlazada, Respuesta Sintético, Respuesta Orden, número de bytes, tamaño de lista.
Probador	El Probador representa a una persona que es la encargada de ejecutar el Benchmark para evaluar los web services.
Respuesta Estructura	Representa el número de veces que se llamará al método del mismo nombre del web service.
Respuesta Lista Enlazada	Representa el número de veces que se llamará al método del mismo nombre del web service.
Respuesta Orden	Representa el número de veces que se llamará al método del mismo nombre del web service.
Respuesta Sintético	Representa el número de veces que se llamará al método del mismo nombre del web service.
Respuesta Vacío	Representa el número de veces que se llamará al método del mismo nombre del web service.

Término	Descripción
Tiempo de ascenso	Tiempo que representa el inicio del lanzamiento de la carga suministrada al web service.
Tiempo de descenso	Tiempo que transcurre desde la finalización de la toma de datos hasta que termina la prueba.
Tiempo de estabilización	Tiempo en el cual la carga suministrada se estabiliza, durante este tiempo de la prueba se realiza la toma de datos propuestos para este estudio.
Tamaño de lista	Cantidad que representa el tamaño de la lista enlazada dentro del método Respuesta Lista Enlazada, así como también el tamaño del arreglo del método Respuesta Estructura del web service.
Tiempo de Respuesta	Tiempo que demora el web service en responder una petición, se calcula mediante la división entre el total de la duración de las llamadas y el número total de estas.
TOTAL_CPU	"Porcentaje de tiempo durante el cual, el procesador estuvo ocupado mientras ejecutaba un subproceso distinto al subproceso del proceso idle"[1].
USED_MEMORY	"Cantidad de memoria física que está siendo utilizada"[1]
USER_CPU	"Porcentaje de tiempo que el procesador dedica a la ejecución en modo usuario"[1].
WebService	El actor WebService representa al web service que será evaluado por el Benchmark.

Elaboración y Fuente: Los Autores

[1] Ver Bibliografía [5]

3.2. ANÁLISIS DEL BENCHMARK

3.2.1. MODELO DE ANÁLISIS

3.2.1.1. Clases de Análisis

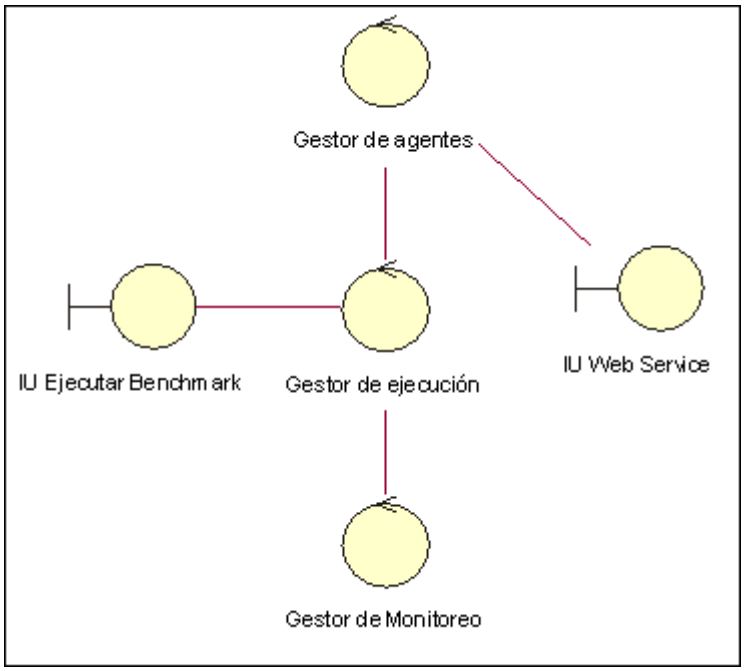
Mediante el análisis de cada uno de los casos de uso definidos en el Modelo de Casos de Uso, se han identificado las siguientes clases, que mediante su interacción satisfacen y cumplen con todos los servicios que ofrece el Benchmark.

1. Gestor de agentes
2. Gestor de ejecución
3. Gestor de Monitoreo
4. IU Ejecutar Benchmark
5. IU Web Service

En el diagrama de clases siguiente se resume las relaciones existentes entre cada una de las clases identificadas:

3.2.1.1.1. Diagrama de Clases

Figura 3-2 Diagrama de Clases de Análisis



Elaboración y Fuente: Los Autores

A continuación se presenta el análisis de cada una de las clases identificadas.

3.2.1.1.2. Gestor de agentes

Clase de control encargada de lanzar los threads para simular la carga concurrente e invocar a los métodos del web service a través de la interfaz IU Web Service.

3.2.1.1.2.1. Roles de Clase

Se crea un objeto de tipo Gestor de agentes cuando el Gestor de ejecución realiza la petición de evaluar el web service. Una vez evaluado el web service entrega los resultados a la clase llamante y es destruido.

3.2.1.1.2.2. Responsabilidades de Clase

La clase Gestor de agentes tiene las siguientes responsabilidades:

- Invocar a los métodos del web service a través de la interfaz IU Web Service.
- Medir número de peticiones atendidas, calcular tiempo de respuesta y disponibilidad
- Entregar los resultados a la clase Gestor de ejecución.

3.2.1.1.2.3. Atributos

Tabla 3-6 Lista de atributos de la clase Gestor de agentes

Nombre	Tipo
Ascenso	Cantidad
Bytes	Cantidad
Descenso	Cantidad
Endpoint	Cadena
Estabilización	Cantidad
Estructura	Cantidad
Lista Enlazada	Cantidad
Orden	Cantidad

Nombre	Tipo
Sintético	Cantidad
Tamaño Lista	Cantidad
Vacío	Cantidad

Elaboración y Fuente: Los Autores

3.2.1.1.3. Gestor de ejecución

Clase de control encargada de iniciar y finalizar la ejecución del Benchmark.

3.2.1.1.3.1. Roles de Clase

Se crea un objeto de tipo Gestor de Ejecución cuando el Probador realiza la petición de ejecución de Benchmark mediante la interfaz IU Ejecutar Benchmark para medir y calcular los índices propuestos. Una vez recopilados los datos de estos, éste objeto es destruido.

3.2.1.1.3.2. Responsabilidades de Clase

La clase Gestor de ejecución tiene las siguientes responsabilidades:

- Cargar los parámetros con los que se realizará la prueba.
- Llamar a la clase Gestor de Monitoreo para a través de ésta tomar los contadores de CPU y Memoria.
- Llamar a la clase Gestor de agentes para evaluar el web service.
- Recopilar y entregar los resultados a la clase IU Ejecutar Benchmark

3.2.1.1.3.3. Atributos

Tabla 3-7 Lista de atributos de la clase Gestor de ejecución

Nombre	Tipo
Agentes	Cantidad

Elaboración y Fuente: Los Autores

3.2.1.1.4. Gestor de Monitoreo

Clase de control encargada del monitoreo de CPU y Memoria durante la prueba.

3.2.1.1.4.1. Roles de Clase

Un objeto de tipo Gestor de Monitoreo es creado luego de que el Probador ha realizado la petición de ejecución del Benchmark, una vez finalizada la misma, éste es destruido.

3.2.1.1.4.2. Responsabilidades de Clase

La clase Gestor de Monitoreo tiene las siguientes responsabilidades:

- Recibir de la clase Gestor de ejecución la petición de monitoreo de CPU y Memoria.
- Monitorear los índices de CPU y Memoria durante la prueba.
- Entregar la información de los índices de CPU y Memoria a la clase Gestor de ejecución.

3.2.1.1.5. IU Ejecutar Benchmark

Interfaz que permite al Probador ejecutar el Benchmark.

3.2.1.1.5.1. Roles de Clase

Un objeto tipo IU Ejecutar Benchmark es creado para permitir al Probador ejecutar el Benchmark, éste es destruido una vez que el Probador ha recibido los resultados de la prueba.

3.2.1.1.5.2. Responsabilidades de Clase

La interfaz IU Ejecutar Benchmark tiene las siguientes responsabilidades:

- Recibir del Probador la petición de ejecutar Benchmark
- Invocar o crear una instancia de un objeto de la clase Gestor de ejecución.
- Entregar al Probador los resultados obtenidos de la prueba.

3.2.1.1.6. IU Web Service

Interfaz que permite al Benchmark comunicarse con el web service.

3.2.1.1.6.1. Roles de Clase

Un objeto tipo IU Web Service es creado para permitir a la clase Gestor de agentes para evaluar el web service, una vez evaluado el web service es destruido.

3.2.1.1.6.2. Responsabilidades de Clase

La clase IU Web Service tiene las siguientes responsabilidades:

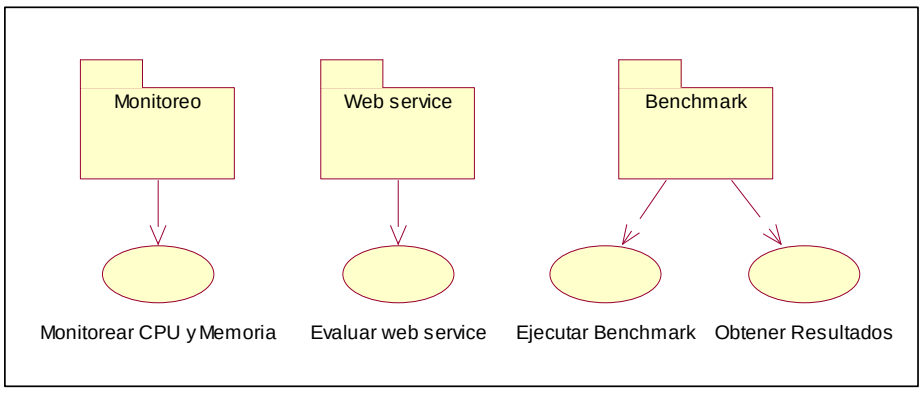
- Recibir de la clase Gestor de agentes la petición de invocar a los métodos del web service.
- Invocar a los métodos del web service descritos en la sección 3.1.1.

3.2.1.2. Paquetes de Análisis

Se han definido tres paquetes de análisis, los mismos que se describen a continuación:

- Benchmark: Agrupa a las funciones que permiten ejecutar el Benchmark y obtener los resultados.
- Monitoreo: Agrupa la funcionalidad de monitorear CPU y Memoria durante la prueba.
- Web Service: Agrupa la funcionalidad de evaluar el web service.

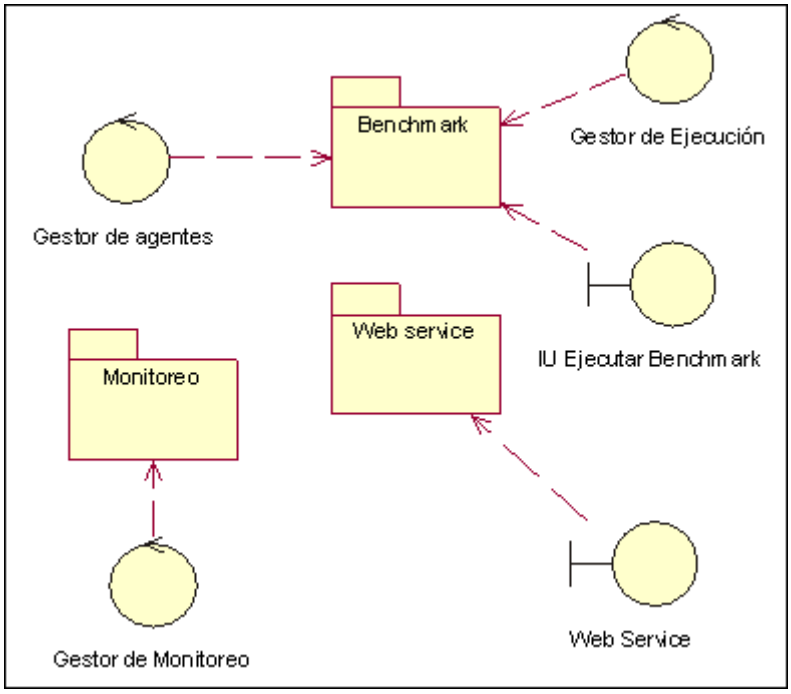
Figura 3-3 Paquetes de Análisis



Elaboración y Fuente: Los Autores

3.2.1.2.1. Distribución de Clases y Paquetes de Análisis

Figura 3-4 Distribución de Clases y Paquetes de Análisis



Elaboración y Fuente: Los Autores

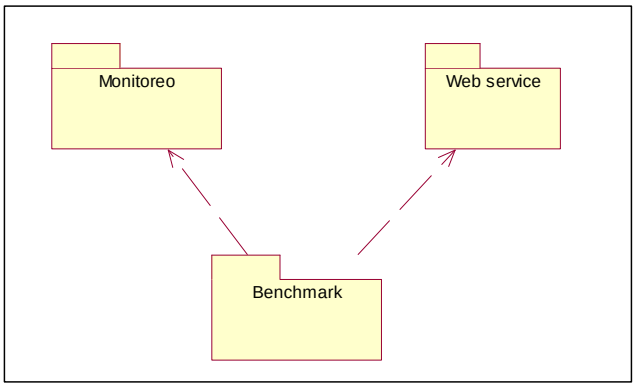
3.2.1.2.2. Relaciones entre paquetes

En el proceso de análisis se han identificado tres paquetes que son: Benchmark, Monitoreo y Web Service.

Las relaciones entre los paquetes son las siguientes:

- Benchmark – Monitoreo: Se requiere realizar un monitoreo de CPU y Memoria durante la ejecución del Benchmark
- Benchmark – Web Service: Se requiere realizar la evaluación durante la ejecución del Benchmark.

Figura 3-5 Relaciones entre paquetes



Elaboración y Fuente: Los Autores

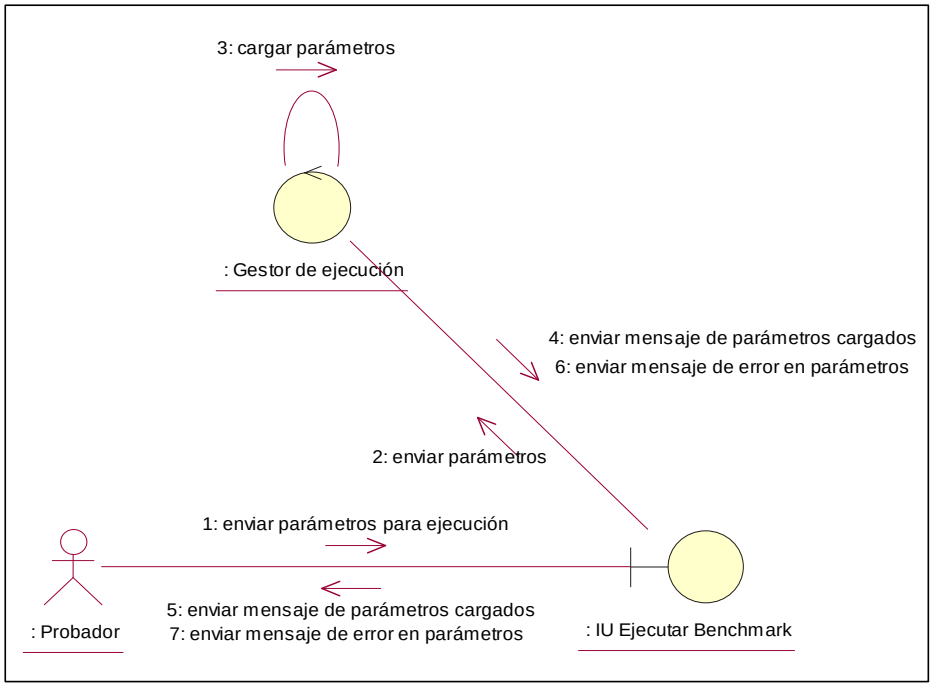
3.2.1.3. Realización de Casos de Uso – Análisis

Los Diagramas de Clases de la realización de cada caso de uso del Benchmark se encuentran en el Anexo 2.

3.2.1.3.1. Caso de Uso Ejecutar Benchmark

3.2.1.3.1.1. Diagrama de Colaboración

Figura 3-6 Diagrama de Colaboración del Caso de Uso Ejecutar Benchmark



Elaboración y Fuente: Los Autores

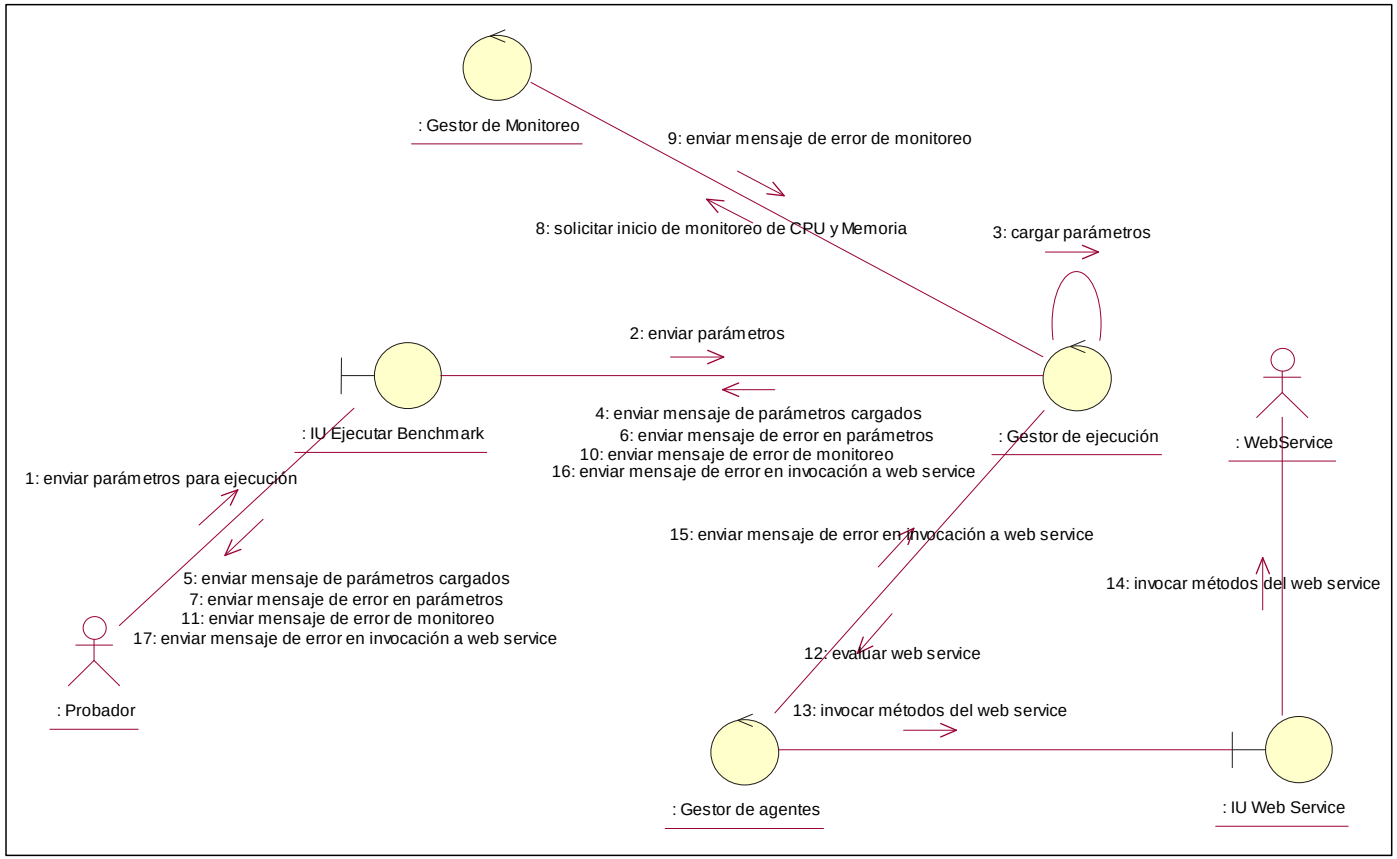
3.2.1.3.1.2. Flujo se Sucesos – Análisis

El Probador envía los parámetros con los que se ejecutará el Benchmark a través de la interfaz *IU Ejecutar Benchmark*, ésta a su vez envía los parámetros a la clase Gestor de ejecución para posteriormente cargarlos (1, 2, 3). En caso de que no existan errores en los parámetros, la clase *Gestor de ejecución* envía el mensaje de que los parámetros son correctos a la interfaz *IU Ejecutar Benchmark* y ésta al Probador, caso contrario envía el mensaje de error en los parámetros, impidiéndole la ejecución del Benchmark (4, 5, 6, 7).

3.2.1.3.2. Caso de Uso Evaluar web service

3.2.1.3.2.1. Diagrama de Colaboración

Figura 3-7 Diagrama de Colaboración del Caso de Uso Evaluar web service



Elaboración y Fuente: Los Autores

3.2.1.3.2.2. Flujo de Sucesos – Análisis

El Probador envía los parámetros con los que se ejecutará el Benchmark a través de la interfaz *IU Ejecutar Benchmark*, ésta a su vez envía los parámetros a la clase Gestor de ejecución para posteriormente cargarlos (1, 2, 3). En caso de que no existan errores en los parámetros, la clase *Gestor de ejecución* envía el mensaje de que los parámetros son correctos a la interfaz *IU Ejecutar Benchmark* y ésta al Probador, caso contrario envía el mensaje de error en los parámetros, impidiéndole la ejecución del Benchmark (4, 5, 6, 7).

La clase *Gestor de ejecución* solicita a la clase *Gestor de Monitoreo* el inicio del monitoreo de CPU y Memoria, en caso de error, transmite el mensaje respectivo a la clase *Gestor de ejecución*, luego lo pasa al Probador a través de la interfaz *IU Ejecutar Benchmark* (8. 9. 10. 11). Si esto sucede la ejecución del Benchmark finaliza.

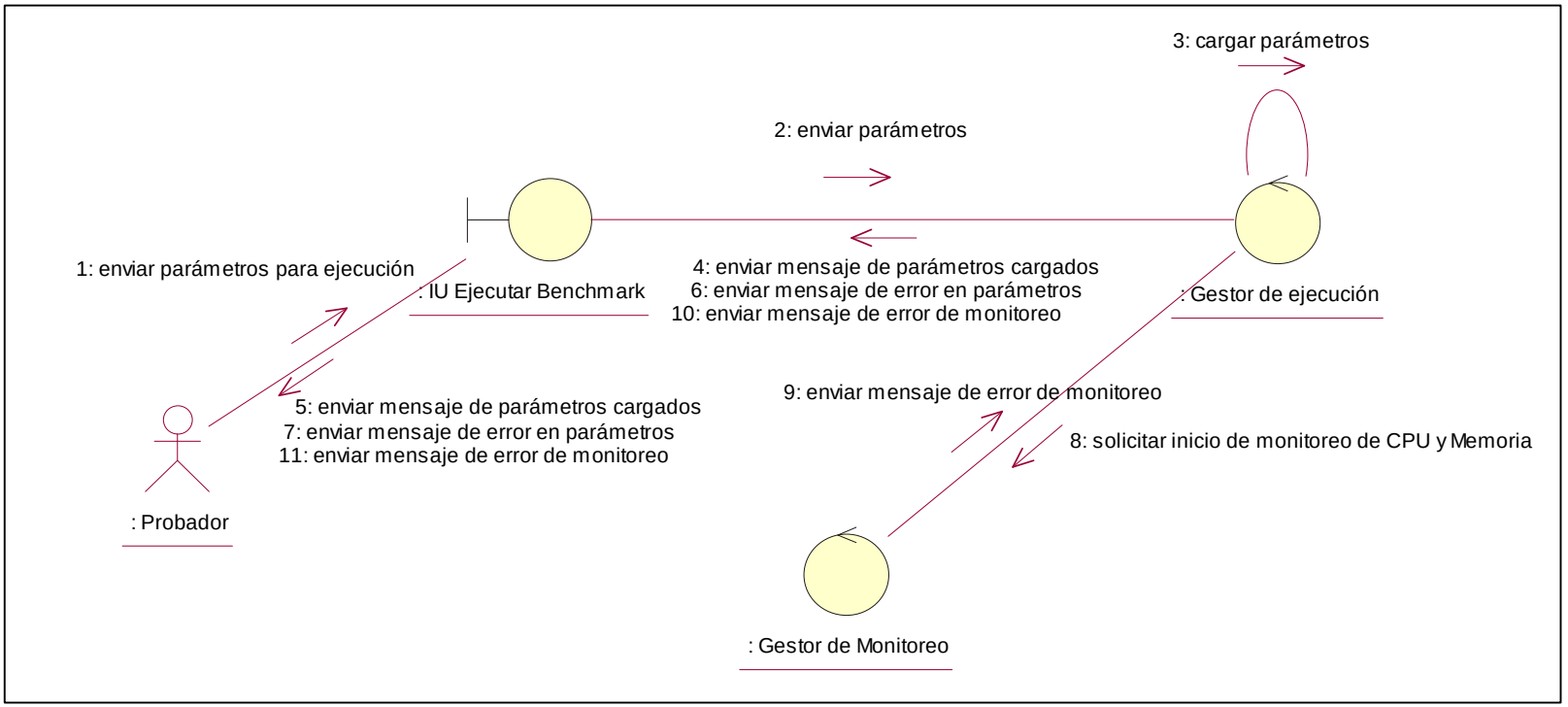
La clase *Gestor de ejecución* envía la petición evaluar web service (medir número de peticiones atendidas, calcular tiempo de respuesta y disponibilidad) a la clase *Gestor de agentes*, la misma que llama a la interfaz *IU Web Service* para que invoque a los métodos del web service (12, 13, 14).

En caso de que la interfaz *IU Web Service* no se pueda comunicar con el web service, la clase *Gestor de agentes* envía el respectivo mensaje de error a la clase *Gestor de ejecución,* y ésta a su vez lo envía a la interfaz *IU Ejecutar Benchmark* para que comunique al Probador, la ejecución del Benchmark finaliza (15, 16, 17).

3.2.1.3.3. Caso de Uso Monitorear CPU y Memoria

3.2.1.3.3.1. Diagrama de Colaboración

Figura 3-8 Diagrama de Colaboración del Caso de Uso Monitorear CPU y Memoria



Elaboración y Fuente: Los Autores

3.2.1.3.3.2. Flujo de Sucesos – Análisis

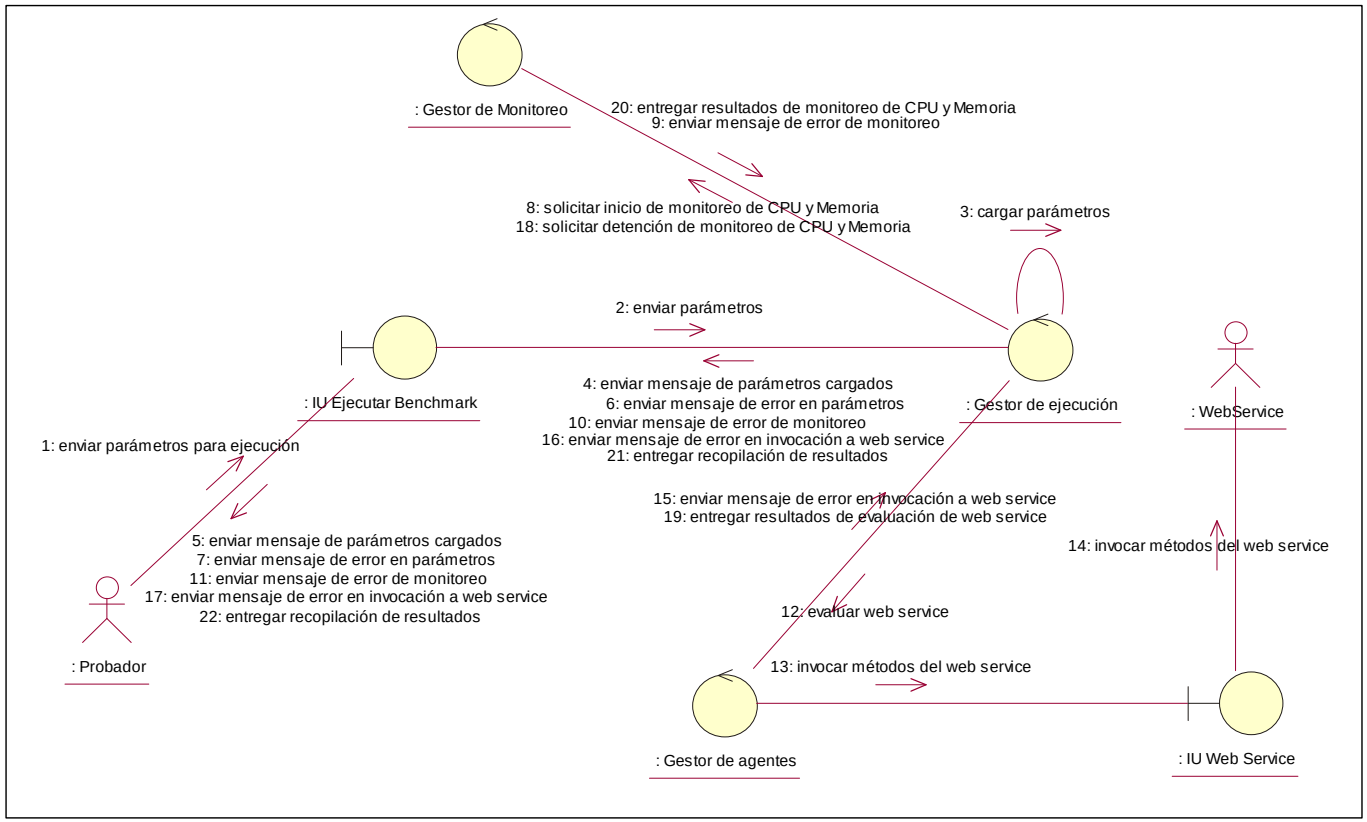
El Probador envía los parámetros con los que se ejecutará el Benchmark a través de la interfaz *IU Ejecutar Benchmark*, ésta a su vez envía los parámetros a la clase Gestor de ejecución para posteriormente cargarlos (1, 2, 3). En caso de que no existan errores en los parámetros, la clase *Gestor de ejecución* envía el mensaje de que los parámetros son correctos a la interfaz *IU Ejecutar Benchmark* y ésta al Probador, caso contrario envía el mensaje de error en los parámetros, impidiéndole la ejecución del Benchmark (4, 5, 6, 7).

La clase *Gestor de ejecución* solicita a la clase *Gestor de Monitoreo* el inicio del monitoreo de CPU y Memoria, en caso de error, transmite el mensaje respectivo a la clase *Gestor de ejecución*, luego lo pasa al Probador a través de la interfaz *IU Ejecutar Benchmark* (8. 9. 10. 11). Si esto sucede la ejecución del Benchmark finaliza.

3.2.1.3.4. Caso de Uso Obtener Resultados

3.2.1.3.4.1. Diagrama de Colaboración

Figura 3-9 Diagrama de Colaboración del Caso de Uso Obtener Resultados



Elaboración y Fuente: Los Autores

3.2.1.3.4.2. Flujo de Sucesos – Análisis

El Probador envía los parámetros con los que se ejecutará el Benchmark a través de la interfaz *IU Ejecutar Benchmark*, ésta a su vez envía los parámetros a la clase Gestor de ejecución para posteriormente cargarlos (1, 2, 3). En caso de que no existan errores en los parámetros, la clase *Gestor de ejecución* envía el mensaje de que los parámetros son correctos a la interfaz *IU Ejecutar Benchmark* y ésta al Probador, caso contrario envía el mensaje de error en los parámetros, impidiéndole la ejecución del Benchmark (4, 5, 6, 7).

La clase *Gestor de ejecución* solicita a la clase *Gestor de Monitoreo* el inicio del monitoreo de CPU y Memoria, en caso de error, transmite el mensaje respectivo a la clase *Gestor de ejecución*, luego lo pasa al Probador a través de la interfaz *IU Ejecutar Benchmark* (8. 9. 10. 11). Si esto sucede la ejecución del Benchmark finaliza.

La clase *Gestor de ejecución* envía la petición evaluar web service (medir número de peticiones atendidas, calcular tiempo de respuesta y disponibilidad) a la clase *Gestor de agentes*, la misma que llama a la interfaz *IU Web Service* para que invoque a los métodos del web service (12, 13, 14).

En caso de que la interfaz *IU Web Service* no se pueda comunicar con el web service, la clase *Gestor de agentes* envía el respectivo mensaje de error a la clase *Gestor de ejecución,* y ésta a su vez lo envía a la interfaz *IU Ejecutar Benchmark* para que comunique al Probador, la ejecución del Benchmark finaliza (15, 16, 17).

La clase *Gestor de ejecución* solicita a la clase *Gestor de Monitoreo* que detenga el monitoreo (18).

La clase *Gestor de agentes* entrega los resultados de los índices número de peticiones atendidas, tiempo de respuesta y disponibilidad del web service a la clase *Gestor de ejecución*. La clase *Gestor de Monitoreo* entrega los resultados de los contadores de CPU y Memoria a la clase *Gestor de ejecución.* Los

resultados recopilados llegan al Probador desde la clase *Gestor de ejecución* por medio de la interfaz *IU Ejecutar Benchmark*, la ejecución del Benchmark finaliza (19, 20, 21, 22).

3.3. DISEÑO DEL BENCHMARK

3.3.1. MODELO DE DISEÑO

3.3.1.1. Clases de Diseño

En base al análisis de la construcción del Benchmark para realizar un estudio comparativo entre web services desarrollados en Java y .Net se han identificado las siguientes clases de diseño:

Tabla 3-8 Transición de las Clases de Análisis a las de Diseño

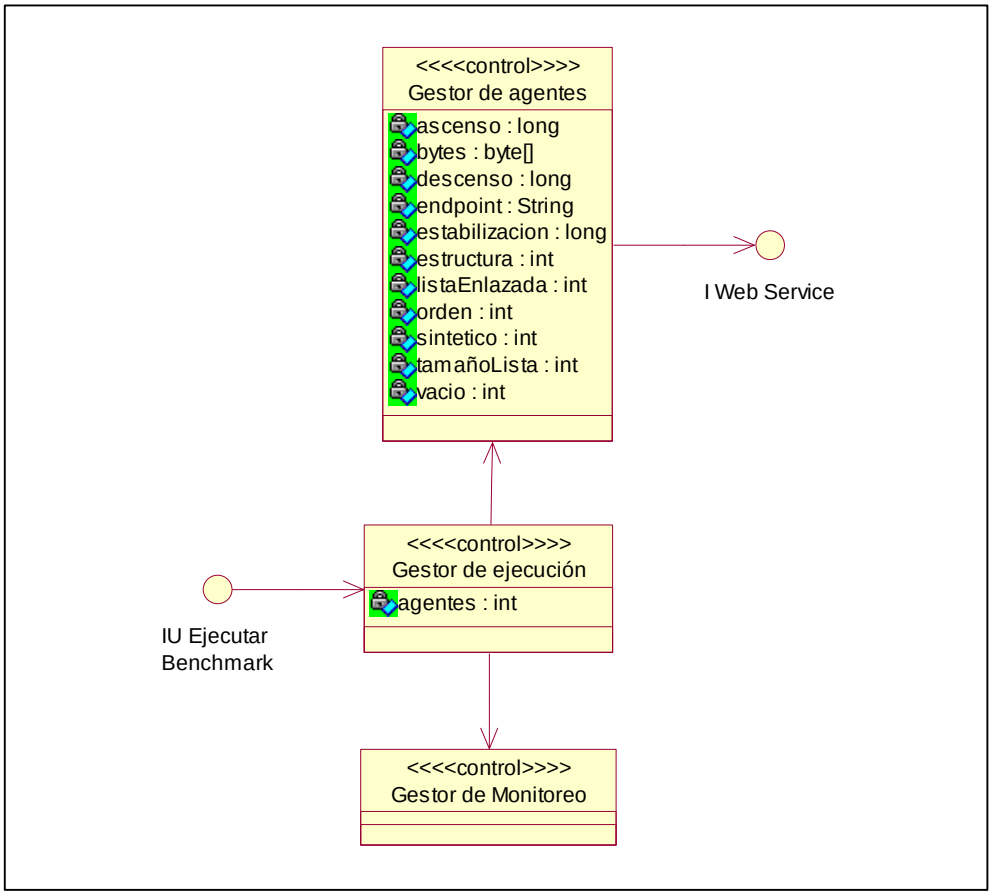
Modelo de Análisis	Modelo de Diseño
Gestor de agentes	Gestor de agentes
Gestor de Ejecución	Gestor de Ejecución
Gestor de Monitoreo	Gestor de Monitoreo
IU Ejecutar Benchmark	IU Ejecutar Benchmark
IU Web Service	I Web Service

Elaboración y Fuente: Los Autores

En el siguiente diagrama se presentan las interacciones existentes entre cada una de las clases de diseño identificadas.

3.3.1.1.1. Diagrama de Clases

Figura 3-10 Diagrama de Clases de Diseño



Elaboración y Fuente: Los Autores

A continuación se presenta el diseño de cada una de las clases identificadas.

3.3.1.1.2. Gestor de agentes

Esta clase sirve para evaluar al web service.

3.3.1.1.2.1. Atributos

Tabla 3-9 Lista de atributos de la clase Gestor de agentes

Nombre del Atributo	Descripción
Ascenso	Representa el tiempo de ascenso
Bytes	Tamaño del arreglo de bytes.
Descenso	Representa el tiempo de descenso

Nombre del Atributo	Descripción
Endpoint	Representa la dirección en dónde se encuentra alojado el web service a evaluar
Estabilización	Representa el tiempo de estabilización
Estructura	Representa el método Respuesta Estructura del web service
Lista Enlazada	Representa el método Respuesta Lista Enlazada del web service
Orden	Representa el método Respuesta Orden del web service
Sintético	Representa el método Respuesta Sintético del web service
Tamaño Lista	Representa el tamaño de la lista enlazada y del arreglo Estructura
Vacío	Representa el método Respuesta Vacío del web service

Elaboración y Fuente: Los Autores

3.3.1.1.2.2. Métodos

Tabla 3-10 Métodos de la clase Gestor de agentes

Nombre del Método	Descripción
calcularDisponibilidad	Método que permite calcular la disponibilidad del web service.
calcularTiempoRespuesta	Método que permite calcular el tiempo de respuesta del web service.
invocarWebService	Método que permite invocar al web service.
medirNumeroPeticiones	Método que permite medir el número de peticiones atendidas por el web service.
obtenerDisponibilidad	Método que devuelve la disponibilidad del web service.
obtenerNumeroPeticiones	Método que devuelve el número de peticiones atendidas por el web service.

Nombre del Método	Descripción
obtenerTiempoRespuesta	Método que devuelve el tiempo de respuesta del web service.

Elaboración y Fuente: Los Autores

3.3.1.1.3. Gestor de ejecución

Esta clase sirve para iniciar y finalizar la ejecución del Benchmark.

3.3.1.1.3.1. Atributos

Tabla 3-11 Lista de atributos de la clase Gestor de ejecución

Nombre del Atributo	Descripción
Agentes	Representa el número de agentes que se utilizarán en la prueba

Elaboración y Fuente: Los Autores

3.3.1.1.3.2. Métodos

Tabla 3-12 Métodos de la clase Gestor de ejecución

Nombre del Método	Descripción
cargarParametros	Método que carga los parámetros con los que se realizará la prueba.
recolectarResultados	Método que recopila los datos de los índices del web service y los contadores de CPU y Memoria.

Elaboración y Fuente: Los Autores

3.3.1.1.4. Gestor de Monitoreo

Esta clase sirve para monitorear CPU y Memoria durante la prueba.

3.3.1.1.4.1. Métodos

Tabla 3-13 Métodos de la clase Gestor de Monitoreo

Nombre del Método	Descripción
detenerMonitoreo	Método que detiene el monitoreo de CPU y Memoria.

Nombre del Método	Descripción
iniciarMonitoreo	Método que inicia el monitoreo de CPU y Memoria.
obtenerContadores	Método que devuelve los contadores de CPU y Memoria.

Elaboración y Fuente: Los Autores

3.3.1.1.5. I Web Service

Esta interfaz expone los métodos necesarios para invocar al web service

3.3.1.1.5.1. Métodos

Tabla 3-14 Métodos de la interfaz I Web Service

Nombre del Método	Descripción
respuestaEstructura	Método que invoca al método del web service del mismo nombre.
respuestaListaEnlazada	Método que invoca al método del web service del mismo nombre.
respuestaOrden	Método que invoca al método del web service del mismo nombre.
respuestaSintetico	Método que invoca al método del web service del mismo nombre.
respuestaVacio	Método que invoca al método del web service del mismo nombre.

Elaboración y Fuente: Los Autores

3.3.1.1.6. IU Ejecutar Benchmark

Esta interfaz expone los métodos necesarios para ejecutar el Benchmark.

3.3.1.1.6.1. Métodos

Tabla 3-15 Métodos de la interfaz IU Ejecutar Benchmark

Nombre del Método	Descripción
cargarParametros	Método que carga los parámetros con los que se realizará la prueba.
recolectarResultados	Método que recopila los datos de los índices del web service y los contadores de CPU y Memoria.

Elaboración y Fuente: Los Autores

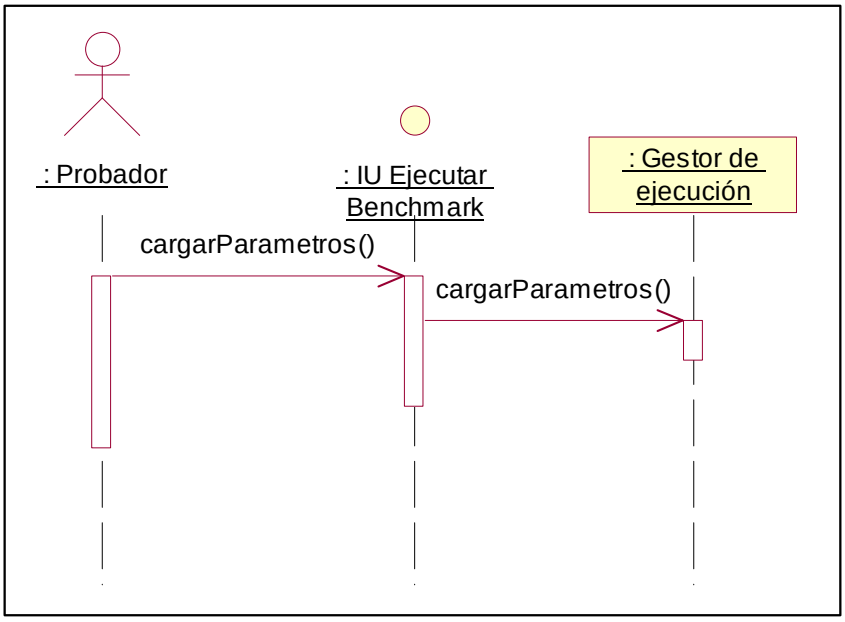
3.3.2. REALIZACIÓN DE CASOS DE USO - DISEÑO

Los Diagramas de Clases de la realización de cada caso de uso del Benchmark se adjuntan en el Anexo 3.

3.3.2.1. Caso de Uso Ejecutar Benchmark

3.3.2.1.1. Diagrama de Interacción

Figura 3-11 Diagrama de Interacción del Caso de Uso Ejecutar Benchmark



Elaboración y Fuente: Los Autores

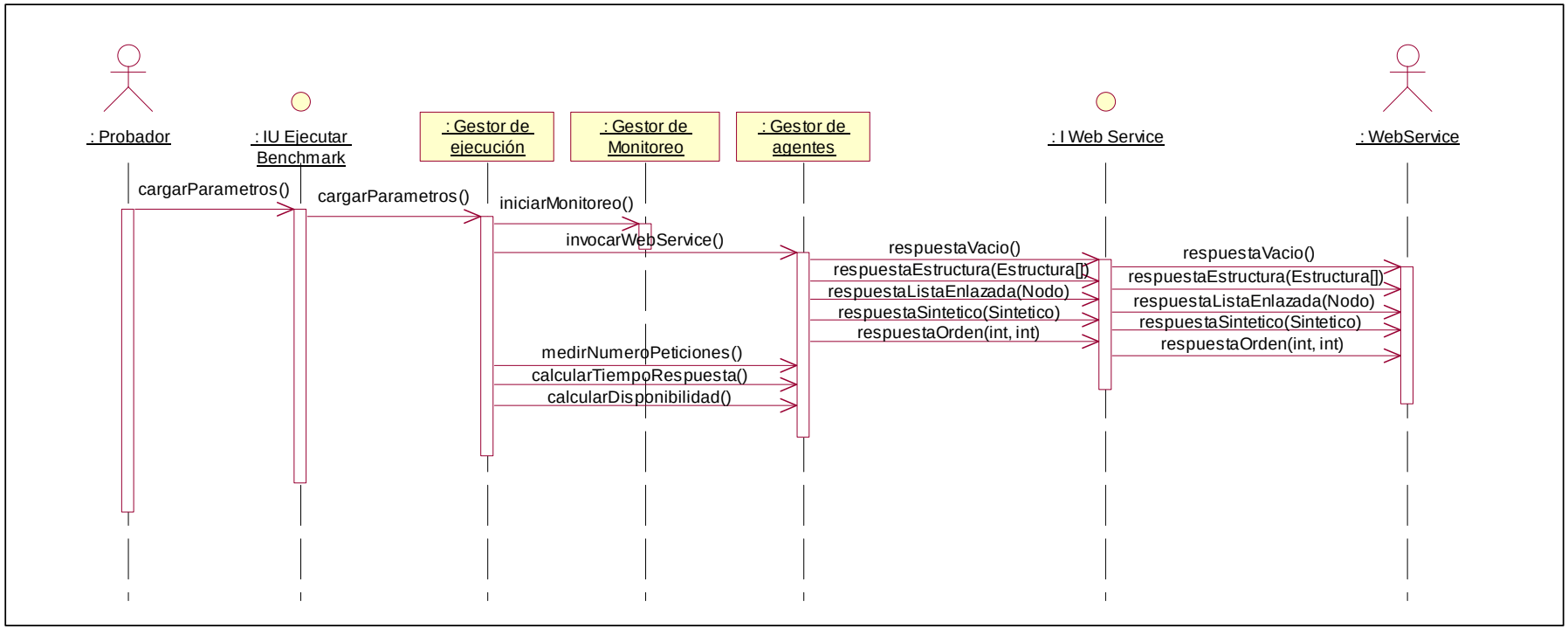
3.3.2.1.2. Flujo de Sucesos - Diseño

El Probador realiza una petición de ejecución de Benchmark enviando los parámetros para realizar la prueba mediante la interfaz *IU Ejecutar Benchmark*, ésta envía los parámetros hacia la clase *Gestor de agentes*, con esto se inicia la ejecución del Benchmark.

3.3.2.2. Caso de Uso Evaluar web service

3.3.2.2.1. Diagrama de Interacción

Figura 3-12 Diagrama de Interacción del Caso de Uso Evaluar web service



Elaboración y Fuente: Los Autores

3.3.2.2.2. Flujo de Sucesos - Diseño

El Probador realiza una petición de ejecución de Benchmark enviando los parámetros para realizar la prueba mediante la interfaz *IU Ejecutar Benchmark*, ésta envía los parámetros hacia la clase *Gestor de ejecución*, con esto se inicia la ejecución del Benchmark.

La clase *Gestor de ejecución* solicita a la clase *Gestor de Monitoreo* que inicie el monitoreo de CPU y Memoria.

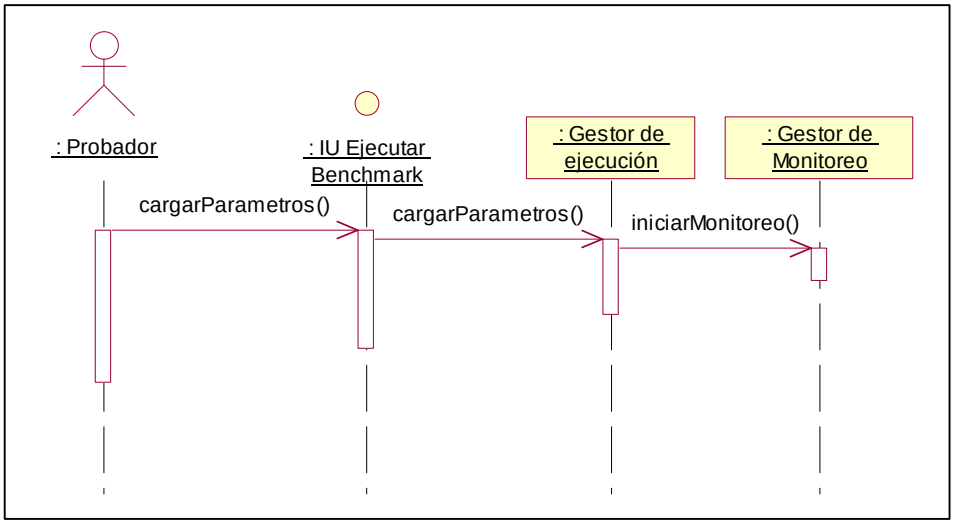
La clase *Gestor de ejecución* solicita a la clase *Gestor de agentes* que invoque al web service. La clase *Gestor de agentes* llama a los métodos RespuestaVacio, RespuestaListaEnlazada, RespuestaEstructura, RespuestaSintetico y RespuestaOrden de la interfaz *I Web Service*, la misma que llamará a los métodos respectivos del web service.

La clase *Gestor de ejecución* llama a los métodos medirNumeroPeticiones, calcularTiempoRespuesta y calcularDisponibilidad de la clase *Gestor de agentes* para medir o calcular los índices del web service.

3.3.2.3. Caso de Uso Monitorear CPU y Memoria

3.3.2.3.1. Diagrama de Interacción

Figura 3-13 Diagrama de Interacción del Caso de Uso Monitorear CPU y Memoria



Elaboración y Fuente: Los Autores

3.3.2.3.2. Flujo de Sucesos - Diseño

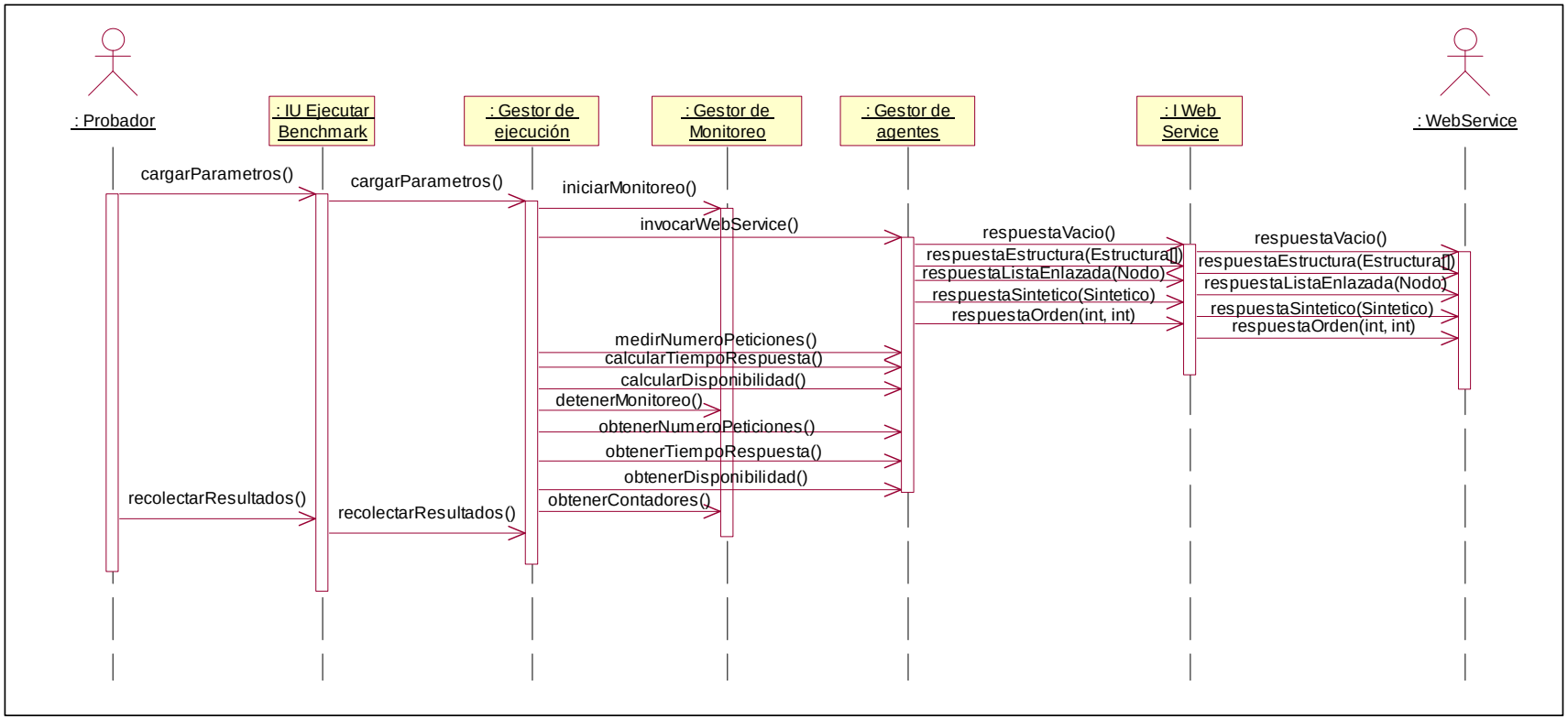
El Probador realiza una petición de ejecución de Benchmark enviando los parámetros para realizar la prueba mediante la interfaz *IU Ejecutar Benchmark*, ésta envía los parámetros hacia la clase *Gestor de ejecución*, con esto se inicia la ejecución del Benchmark.

La clase *Gestor de ejecución* solicita a la clase *Gestor de Monitoreo* que inicie el monitoreo de CPU y Memoria.

3.3.2.4. Caso de Uso Obtener Resultados

3.3.2.4.1. Diagrama de Interacción

Figura 3-14 Diagrama de Interacción del Caso de Uso Obtener Resultados



Elaboración y Fuente: Los Autores

3.3.2.4.2. Flujo de Sucesos – Diseño

El Probador realiza una petición de ejecución de Benchmark enviando los parámetros para realizar la prueba mediante la interfaz *IU Ejecutar Benchmark*, ésta envía los parámetros hacia la clase *Gestor de ejecución*, con esto se inicia la ejecución del Benchmark.

La clase *Gestor de ejecución* solicita a la clase *Gestor de Monitoreo* que inicie el monitoreo de CPU y Memoria.

La clase *Gestor de ejecución* solicita a la clase *Gestor de agentes* que invoque al web service. La clase *Gestor de agentes* llama a los métodos RespuestaVacio, RespuestaListaEnlazada, RespuestaEstructura, RespuestaSintetico y RespuestaOrden de la interfaz *I Web Service*, la misma que llamará a los métodos respectivos del web service.

La clase *Gestor de ejecución* llama a los métodos medirNumeroPeticiones, calcularTiempoRespuesta y calcularDisponibilidad de la clase *Gestor de agentes* para medir o calcular los índices del web service.

La clase *Gestor de ejecución* solicita a la clase *Gestor de Monitoreo* que detenga el monitoreo.

La clase *Gestor de ejecución* solicita a la clase *Gestor de agentes* los resultados de los índices: número de peticiones, tiempo de respuesta y disponibilidad, así como también solicita a la clase *Gestor de Monitoreo* los contadores de CPU y Memoria.

La clase *Gestor de ejecución* entrega los resultados recopilados a la interfaz *IU Ejecutar Benchmark* y ésta al Probador.

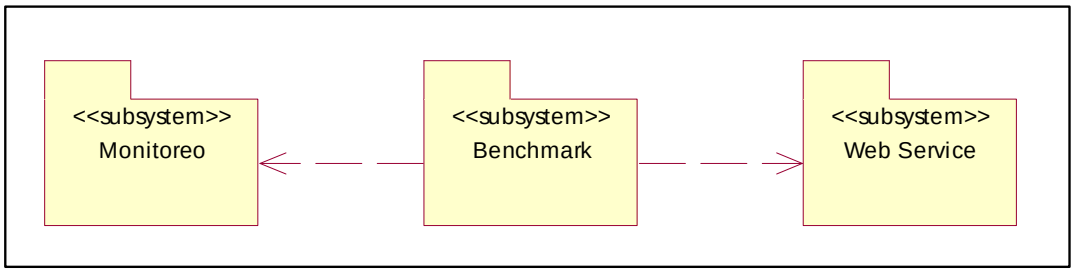
3.3.3. SUSBSISTEMAS DE DISEÑO

De acuerdo al análisis y al diseño del Benchmark, se han encontrado los siguientes subsistemas:

- Benchmark: Subsistema principal de la herramienta, encargado de permitir la ejecución del Benchmark, y obtener los resultados de los contadores de CPU y Memoria, así como también los índices del web service.
- Monitoreo: Subsistema encargado de permitir el monitoreo de CPU y Memoria.
- Web Service: Subsistema encargado de evaluar el web service.

3.3.3.1. Dependencias entre Subsistemas de Diseño

Figura 3-15 Dependencias entre Subsistemas de Diseño



Elaboración y Fuente: Los Autores

3.3.4. INTERFACES

De acuerdo al apartado anterior las interfaces encontradas son las siguientes:

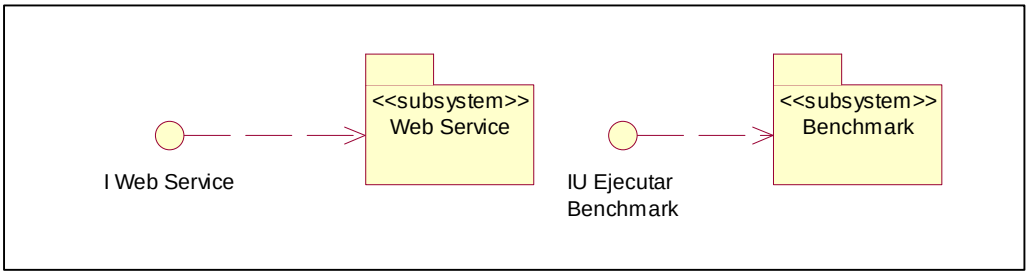
Tabla 3-16 Interfaces definidas para los Subsistemas de Diseño

Subsistema	Interfaz
Benchmark	IU Ejecutar Benchmark
Web Service	I Web Service

Elaboración y Fuente: Los Autores

3.3.4.1. Diagrama de Interfaces para los Subsistemas de Diseño

Figura 3-16 Diagrama de Interfaces para los Subsistemas de Diseño



Elaboración y Fuente: Los Autores

3.3.5. DESCRIPCIÓN DE LA ARQUITECTURA (VISTA DEL MODELO DE DISEÑO)

De acuerdo a la fase de diseño del Benchmark se han definido tres subsistemas, con sus responsabilidades claramente especificadas que son:

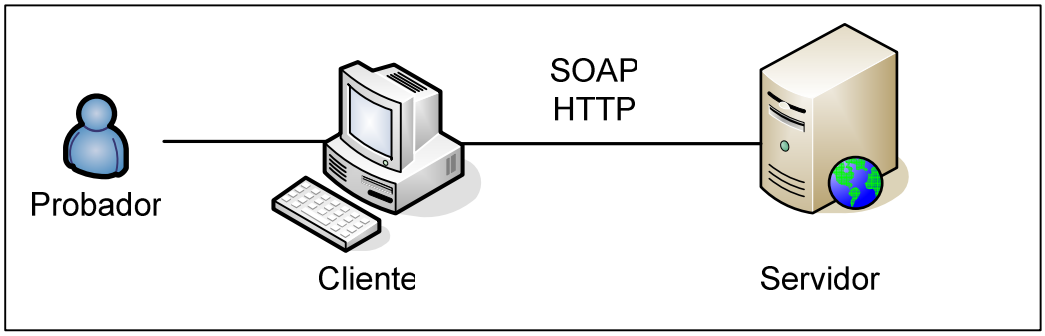
- Benchmark: Tiene la responsabilidad de ejecutar el Benchmark y de obtener los resultados.
- Monitoreo: Tiene la responsabilidad de monitorear CPU y Memoria.
- Web Service: Tiene la responsabilidad de comunicarse con el web service para evaluarlo.

El Subsistema Benchmark constituye el núcleo de la herramienta, puesto que los demás Subsistemas dependen de éste para poder realizar sus acciones.

3.3.6. MODELO DE DESPLIEGUE

Los componentes del Benchmark que permitirá realizar un estudio comparativo de Web Services desarrollados en plataformas Java y .NET Subsistemas del Benchmark se encuentran distribuidos en un nodo o más nodos clientes y el web service a evaluar se encuentra en el nodo servidor. La comunicación entre los nodos servidor y cliente se la realizará bajo los protocolos SOAP(Service Object Access Protocol) y HTTP (Hyper Text Transfer Protocol).

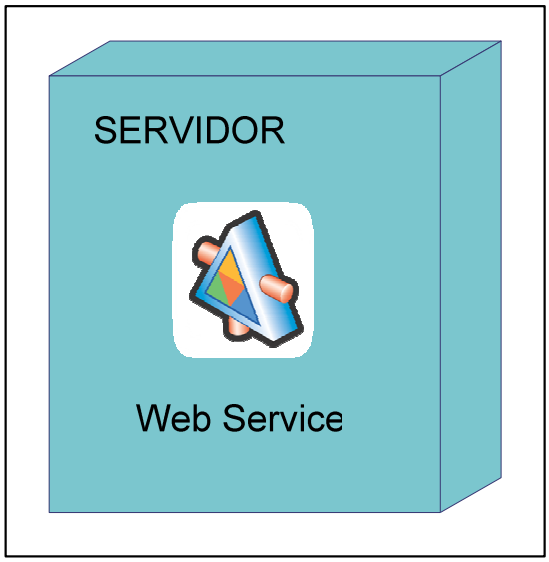
Figura 3-17 Diagrama de despliegue



Elaboración y Fuente: Los Autores

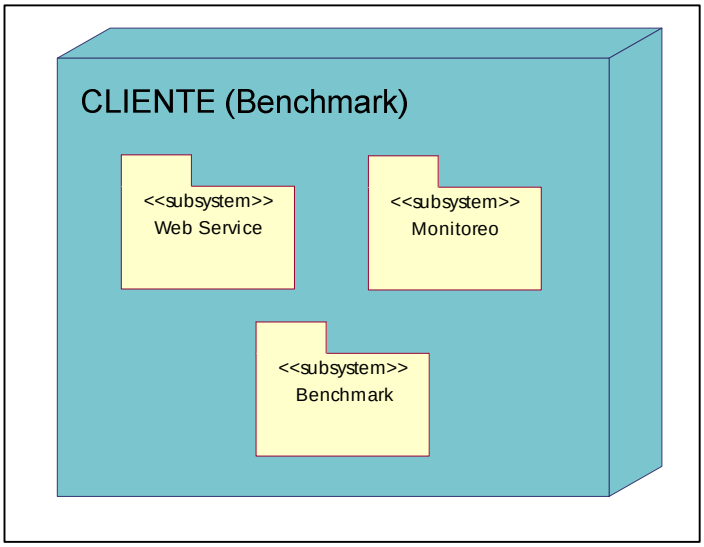
La distribución de los subsistemas se los hará dentro del nodo cliente (Benchmark), el siguiente diagrama presenta la distribución de los componentes entre los distintos nodos.

Figura 3-18 Nodo Servidor



Elaboración y Fuente: Los Autores

Figura 3-19 Nodo Cliente



Elaboración y Fuente: Los Autores

3.3.7. DESCRIPCIÓN DE LA ARQUITECTURA (VISTA DEL MODELO DE DESPLIEGUE)

Los componentes del Benchmark se encuentran distribuidos en un nodo o más nodos clientes, en el nodo Cliente se colocan todos los Subsistemas necesarios para la interacción del Benchmark con el Probador y en el nodo servidor se coloca el web service para evaluarlo con el Benchmark.

3.4. IMPLEMENTACIÓN DEL BENCHMARK

El objetivo de la implementación es plasmar los requerimientos especificados en el diseño para la construcción del Benchmark, que permitirá realizar el estudio comparativo entre web services desarrollados en plataformas Java y .NET.

Para la implementación del Benchmark, se han considerado algunas herramientas de desarrollo, las mismas que han sido evaluadas para de entre ellas seleccionar aquella que más se ajuste a los requisitos de implementación.

3.4.1. DESCRIPCIÓN DE HERRAMIENTAS A UTILIZARSE PARA EL DESARROLLO DEL BENCHMARK

3.4.1.1. Selección de la herramienta de desarrollo

La selección de la herramienta de desarrollo para el Benchmark constituye un elemento importante, puesto que de ésta selección dependerá el rendimiento, facilidad de uso del producto final y el tiempo que se empleará para la construcción del mismo.

En base a los requisitos de implementación que se han definido en el capitulo 3, se ha identificado que la herramienta en la que se desarrollará el Benchmark deberá cumplir con los siguientes criterios:

- Soporte de Orientación a Objetos.
- Soporte para ejecución multithread[1].
- Nivel de conocimiento de la herramienta.
- Facilidad de obtención de la herramienta.
- Bajo costo de desarrollo.
- La herramienta debe facilitar la construcción de aplicaciones orientada a servicios

Para la evaluación de las herramientas de desarrollo se han seleccionado las siguientes plataformas de desarrollo:

- .NET
- Java

3.4.1.1.1. .NET

.NET es una plataforma de desarrollo que reúne un conjunto de lenguajes de programación, además ofrece una serie de herramientas que permiten crear aplicaciones robustas y escalables.

[1] Término, de frecuente uso en programación, que define múltiples procesos ejecutándose en un computador al mismo tiempo y de manera independiente

La arquitectura de esta plataforma es similar a que posee Java, debido a que utiliza una máquina virtual llamada Common Language Runtime (CLR), que es la encargada de compilar el código fuente de cualquiera de los lenguajes La herramienta de desarrollo compila el código fuente de cualquiera de los lenguajes soportados por .Net en un mismo código, denominado código intermedio (MSIL, Microsoft Intermediate Lenguaje). Para generar dicho código el compilador se basa en el Common Language Specification (CLS) que determina las reglas necesarias para crear código MSIL compatible con el CLR.

De esta forma, indistintamente de la herramienta de desarrollo utilizada y del lenguaje elegido, el código generado es siempre el mismo, ya que el MSIL es el único lenguaje que entiende directamente el CLR. Este código es transparente al desarrollo de la aplicación ya que lo genera automáticamente el compilador.
Sin embargo, el código generado en MSIL no es código máquina y por tanto no puede ejecutarse directamente. Se necesita un segundo paso en el que una herramienta denominada compilador JIT (Just-In-Time) genera el código máquina real que se ejecuta en la plataforma que tenga la computadora.

De esta forma se consigue con .Net cierta independencia de la plataforma, ya que cada plataforma puede tener su compilador JIT y crear su propio código máquina a partir del código MSIL.

La compilación JIT la realiza el CLR a medida que se invocan los métodos en el programa y, el código ejecutable obtenido, se almacena en la memoria caché de la computadora, siendo recompilado sólo cuando se produce algún cambio en el código fuente.

.NET soporta multithread, lo que permite crear aplicaciones que manejen distintos procesos a la vez, al igual que en Java, se lo maneja a nivel de máquina virtual, facilitando su programación.

.Net permite diseñar aplicaciones orientadas a servicios, los mismos que sirven para comunicarse entre sí con el mínimo grado de acoplamiento. El uso de la comunicación basada en mensajes ayuda a desacoplar la disponibilidad y

escalabilidad de los servicios, y basarse en los estándares de la industria, como los servicios Web XML, permite la integración con las demás plataformas y tecnologías.

.NET no es una herramienta de libre distribución, razón por la cual se debe tomar en cuenta el costo para obtener la licencia cuando se desarrollen aplicaciones.

3.4.1.1.2. Java 1.5

Java a más de ser un lenguaje de programación, constituye una plataforma completa de desarrollo de aplicaciones, razón por la cual podemos encontrar un sinnúmero de herramientas que facilitan la construcción de diversos tipos de aplicaciones.
Java ofrece independencia de plataforma, esto significa que programas escrito en este lenguaje, pueden ejecutarse igualmente en cualquier tipo de hardware, esto se explica debido a que el código escrito en Java es interpretado por una máquina virtual llamada Java Virtual Machina que funciona como capa intermedia entre el sistema operativo y la lógica funcional de la aplicación. La disponibilidad de la Máquina Virtual para los diversos sistemas operativos existentes es la única limitante en cuanto a la portabilidad de las aplicaciones hechas en Java.

Java brinda un soporte completo para la programación orientada a objetos, y a diferencia de otros lenguajes, ésta característica fue concebida para su construcción desde un inicio. Esta plataforma de desarrollo soporta los tres paradigmas de la programación orientada a objetos: encapsulamiento, herencia y polimorfismo.

Una característica propia de Java es permitir el uso de las capacidades de él o los procesadores para la ejecución de un hilo. Además brinda un esquema de multithread completo y fácil de implementar, basado en clases e interfaces, de esta manera simplifica la comunicación entre procesos mediante el uso de objetos comunes cuyo acceso puede ser sincronizado a través de métodos o

secciones de código declaradas como tal. Esta sincronización es definida en java de manera declarativa sobre el código, por lo que se evita tener que pensar en procesos complejos de bloqueos y desbloqueos, delegando esta tarea a la máquina virtual.

Java es una plataforma que brinda la posibilidad de construir aplicaciones que funcionen dentro de una arquitectura orientada a servicios, específicamente aplicaciones basadas en servicios web que sean flexibles, escalables y que posean la característica de interoperabilidad con otras aplicaciones diseñadas con otras tecnologías.

Se dispone de la versión 1.5 del Java Development Kit de la Sun Microsystems[1], esta herramienta es de libre distribución, por lo que no tiene ningún costo su licencia.

A continuación se presenta un cuadro comparativo entre las herramientas seleccionadas, tomando en cuenta los requisitos que deberían cumplir para la posterior construcción del Benchmark.

Tabla 3-17 Cuadro comparativo entre las herramientas de desarrollo

Características / **Herramientas**	**Soporte de Orientación a Objetos**	**Facilidad de obtención**	**Soporte para ejecución multithread**	**Nivel de conocimiento**	**Costo de desarrollo**	**TOTAL**
Java 1.5	5	5	5	4	5	24
.NET	5	5	5	2	1	18

Elaboración y Fuente: Los Autores

La valoración cuantitativa que se ha considerado es la siguiente:

- 1 malo

[1] Empresa que desarrolló Java, encargada de su mantenimiento y actualización.

- 2 regular
- 3 bueno
- 4 muy bueno
- 5 sobresaliente

3.4.2. DESCRIPCIÓN DE LA HERRAMIENTA DE DESARROLLO

Después de haber hecho la comparación entre las posibles herramientas para la construcción del Benchmark, que permitirá realizar el estudio comparativo entre web services desarrollados en plataformas Java y .NET, frente a los criterios de evaluación, se ha llegado a la conclusión de que Java es la mejor opción, ya que cumple con los requerimientos de implementación especificados en la fase de diseño como lo muestra la valoración total de la tabla 3-17.

A finales de los años ochenta Sun Microsystems decide introducirse en el mercado de la electrónica de consumo y más concretamente en los equipos domésticos, incluyendo la televisión interactiva. Java, nace como un lenguaje ideado en sus comienzos para programar electrodomésticos, en sus primeras versiones se llamó OAK.

Las principales características de Java que la hacen una plataforma robusta y flexible son las siguientes:

- Simple: Java ofrece toda la funcionalidad de un lenguaje potente, pero sin las características menos usadas y más confusas de éstos. Java elimina muchas de las características de otros lenguajes, para mantener reducidas las especificaciones del lenguaje y añadir características muy útiles como el garbage collector (reciclador de memoria dinámica). No es necesario preocuparse de liberar memoria, el reciclador se encarga de ello y como es un thread de baja prioridad, cuando entra en acción, permite liberar bloques de memoria muy grandes, lo que reduce la fragmentación de la memoria.

- Orientado a objetos: Java es un lenguaje que soporta programación orientada a objetos.

- Distribuido: Java se ha diseñado para trabajar en ambiente de redes y contienen una gran biblioteca de clases para la utilización del protocolo TCP/IP, incluyendo HTTP y FTP. El código Java se puede manipular a través de recursos URL.

- Interpretado: El compilador Java traduce cada archivo fuente (clases) a código de bytes (Bytecode), que puede ser interpretado por todas las máquinas que den soporte a un visualizador de que funcione con Java. Este Bytecode no es específico de una máquina determinada, por lo que no se compila y enlaza como en el ciclo clásico, sino que se interpreta.
- Robusto: Java realiza verificaciones en busca de problemas tanto en tiempo de compilación como en tiempo de ejecución. La comprobación de punteros y límites de arreglos se lo realiza de manera automática, esto permite reducir los errores de programación.

- Portable: Por ser un lenguaje interpretado Java es completamente independiente de rutinas específicas de la plataforma, su código es interpretado por una máquina virtual, encargada de realizar las llamadas a rutinas de la plataforma en la que se encuentra la aplicación.

- Seguro: El código generado en Java es seguro, debido a que elimina la posibilidad de acceder ilegalmente a la memoria dentro del sistema operativo. La Máquina Virtual de Java es la encargada de administrar los accesos a los recursos del sistema operativo.

- Dinámico: El lenguaje Java y su sistema de ejecución en tiempo real son dinámicos en la fase de enlazado. Las clases sólo se enlazan a medida que son necesitadas. Se pueden enlazar nuevos módulos de código bajo demanda, procedente de fuentes muy variadas, incluso desde una red.

- Multithread: Al ser multithreaded, Java permite muchas actividades simultáneas en un programa. Al estar los threads construidos en el lenguaje, son más fáciles de usar y más robustos. El beneficio de ser

multithreaded consiste en un mejor rendimiento interactivo y mejor comportamiento en tiempo real.

- Flexible: Mediante interfaces, se facilita completamente la separación de el ¿Qué hace? una clase y el ¿Cómo? lo hace, debido a que Java permite instanciar objetos sin conocer su implementación sino hasta en tiempo de ejecución.

- Escalable: Java permite construir aplicaciones dentro de una arquitectura orientada a servicios con un mínimo grado de acoplamiento, logrando de esta manera que haya muy poca dependencia entre la aplicación solicitante y el servicio que ésta usa.

3.4.3. CONSTRUCCIÓN DEL BENCHMARK

3.4.3.1. Estándares de programación

Para la construcción del Benchmark que permitirá hacer un estudio comparativo entre web services en plataformas Java y .NET, se seguirán los estándares de programación para Java (Code Conventions for the Java Programming Language[1]), propuestos por Sun Microsystems, que es la empresa desarrolladora del lenguaje Java y encargada de establecer estas normas.

3.4.3.1.1. Idioma

El idioma que se utilizará para codificar el Benchmark será el inglés, por ser un lenguaje universal que permite emplear los estándares propuestos con una mayor facilidad que otro idioma.

Adicionalmente la mayoría de información que se pueden encontrar en libros o en foros se encuentra en este idioma.

[1] Ver Bibliografía [15].

El uso de este idioma facilitará a otros desarrolladores la comprensión del código escrito.

3.4.3.1.2. Archivos

Los nombres de los archivos fuente escritos en Java, estarán formados por el nombre de la clase más la extensión ".java".

Los archivos resultantes de la ejecución deben tener el mismo nombre que sus respectivos archivos fuentes, con la diferencia que llevarán la extensión ".class". Ejemplo:

- WSTestClient.java para la clase WSTestClient
- WSTestClient.class para el fuente WSTestClient.java

3.4.3.1.3. Comentarios

Los programas desarrollados en Java, tienen dos tipos de comentarios: de implantación y de documentación. Los comentarios de implantación se encuentran delimitados entre "/*…*/" y "//". Estos sirven para explicar una sección de código. Ejemplo:

```
/* calculo de porcentaje de disponibilidad */
// se inicia el monitoreo de cpu y memoria
```

Los segundos tipos de comentarios se encuentran delimitados entre "/**…*/". Estos sirven para generar la documentación del código fuente escrito, se puede exportar a archivos HTML utilizando la herramienta javadoc. Ejemplo:

```
/** Clase Agente, esta clase sirve para…
 * @ version 1.0
 * @ autor Daysi Peñaherrera & Miguel Mármol
*/
```

En este tipo de comentarios se puede usar etiquetas que especifican de mejor manera el comentario, dependiendo del contexto. Las mismas que se listan a continuación:

Tabla 3-18 Etiquetas para generación de documentación en Java

Frase Reservada	Descripción	Contexto
@param	Permite describir el parámetro ó los parámetros que recibe un método.	Comentario de método
@version	Permite describir la versión del código escrito, se colocan en los comentarios de cada clase.	Comentario de clase
@return	Permite describir el parámetro de retorno de un método.	Comentario de método
@throws	Permite detallar la razón por la que un método arroja una excepción.	Comentario de método
@autor	Permite especificar el o los nombres de los autores.	Comentario de clase
@see	Permite indicar una referencia a otra clase, método o atributo.	Cualquier comentario.

Elaboración y Fuente: Los Autores

3.4.3.1.4. Convenciones de Nombramiento

Para generar un código que sea de fácil entendimiento se han propuesto reglas para nombrar los diferentes elementos de programación de una aplicación, A continuación se describen los principales.

3.4.3.1.4.1. Clases e Interfaces

Los nombres de las clases deben ser sustantivos, si se requiere utilizar uno o más, se debe poner la primera letra de cada uno con mayúsculas y el resto en minúsculas. En lo posible los nombres de las clases deben ser simples y descriptivos, evitando utilizar abreviaciones, salvo en los casos en que estas sean comunes, como por ejemplo URL. Ejemplo:

- class Agent
- interface ServiceSoap

3.4.3.1.4.2. Métodos

Los métodos deben ser verbos, la primera palabra en minúsculas, de existir más de una, a partir de la segunda, se debe poner la primera letra con mayúsculas y el resto en minúsculas, de igual manera para el resto de palabras. Ejemplo:

- runTest();
- cpuMonitoringUpdate();

3.4.3.1.4.3. Variables e Instancias

Las variables e instancias deben estar escritas con la primera palabra en minúsculas, a partir de la segunda palabra se usará la primera letra con mayúsculas y el resto en minúsculas.

Los nombres deben ser cortos y representativos, se debe evitar el uso de nombres de variables de una sola letra, excepto cuando la variable sea temporal. Para estos casos se puede utilizar i, j, k,m y n para enteros, c, d, y e para caracteres. Ejemplo:

- Agent agent.
- double callDuration.
- int m,n.

3.4.3.1.4.4. Constantes

Los nombres de las constantes deben estar escritas en mayúsculas separadas por subguiones “_”. Ejemplo:

- static final long RAMP_DOWN = 0L;

3.4.3.2. Modelo de Implementación

3.4.3.3. Clases de Implementación

Se han identificado las siguientes clases en base al diseño del Benchmark:

Tabla 3-19 Transición de las clases de diseño a las de implementación

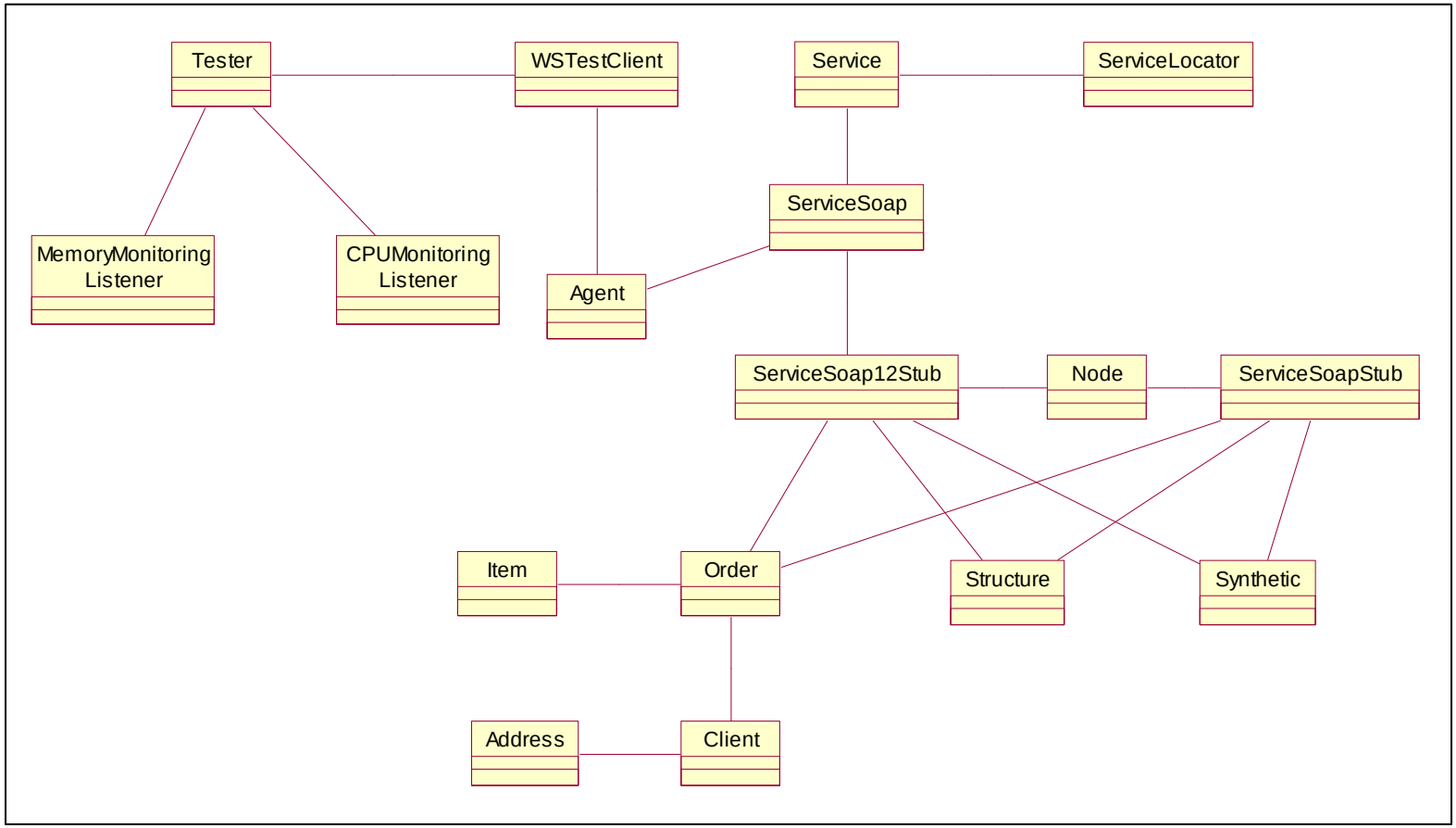
Modelo del Diseño	Modelo de Implementación
Gestor de agentes	Agent
Gestor de ejecución	WSTestClient
Gestor de Monitoreo	CPUMonitoringListener MemoryMonitoringListener
IU Ejecutar Benchmark	Tester
I Web Service	Address Client Item Node ServiceLocator ServiceSoap ServiceSoap12 ServiceSoapStub Structure Synthetic

Elaboración y Fuente: Los Autores

Cabe mencionar que las clases que permitirán hacer el monitoreo de CPU y Memoria han sido construidas por terceros, es decir se utilizará una librería Java llamada JSysmon[1].

[1] http://sourceforge.net/projects/jsysmon/

Figura 3-20 Diagrama de clases de implementación



Elaboración y Fuente: Los Autores

3.4.3.4. Componentes

Cada una de las clases de implementación serán encapsuladas en componentes, tipo archivo, con el mismo nombre más la terminación ".java" de acuerdo a los estándares de programación definidos en la sección 3.4.3.1. A continuación se presentan cada uno de estos componentes y se describa el papel que cumplen dentro de la aplicación que implementa el Benchmark para realizar el estudio comparativo entre web services desarrollados en las plataformas Java y .Net.

Adicionalmente existen cuatro archivos planos que serán considerados como componentes, en el primero se guardan los parámetros para las pruebas. En el segundo se guardan los resultados de las mediciones y cálculos de los índices del web service, en el tercer y cuarto se guardan los datos de las mediciones tomadas de CPU y Memoria respectivamente.

3.4.3.4.1. Componente Address.java

Componente que contiene la información de una dirección de un cliente que será generado en una Orden[1].

3.4.3.4.2. Componente Agent.java

Componente que permite generar los threads que simularán la carga suministrada al web service.

3.4.3.4.3. Componente Client.java

Componente que contiene la información de un Cliente que será generado en una Orden.

3.4.3.4.4. Componente CPU.resultados

Componente que almacenará los datos de los índices de la utilización de CPU.

---

[1] Ver sección 3.1.1.5

3.4.3.4.5. Componente CPUMonitoringListener.java

Componente que permite realizar el monitoreo de CPU.

3.4.3.4.6. Componente Item.java

Componente que contiene la información de un Registro que será generado en una Orden.

3.4.3.4.7. Componente memoria.resultados

Componente que almacenará los datos de los índices de la utilización de memoria.

3.4.3.4.8. Componente MemoryMonitoringListener.java

Componente que permite realizar el monitoreo de Memoria.

3.4.3.4.9. Componente Node.java

Componente que contiene la información de un Nodo para implementar una lista enlazada.[1]

3.4.3.4.10. Componente Order.java

Componente que contiene la información de una Orden. [2]

3.4.3.4.11. Componente parametros.props

Componente que permitirá establecer las propiedades con la que se ejecutarán las pruebas.

---

[1] Ver sección 3.1.1.3

[2] Ver sección 3.1.1.5

3.4.3.4.12. Componente probador.resultados

Componente que almacenará los datos de los índices Tiempo de Respuesta, Número de peticiones atendidas y Disponibilidad.

3.4.3.4.13. Componente Service.java

Componente generado automáticamente por la herramienta Apache AXIS[1] a partir del lenguaje de definición del Web Service WSDL que permitirá invocar a los Web Services.

3.4.3.4.14. Componente ServiceLocator.java

Componente generado automáticamente por la herramienta Apache AXIS[1] a partir del lenguaje de definición del Web Service (WSDL), que permitirá invocar al mismo.

3.4.3.4.15. Componente ServiceSoap.java

Componente generado automáticamente por la herramienta Apache AXIS[1] a partir del lenguaje de definición del Web Service WSDL que permitirá invocar al mismo.

3.4.3.4.16. Componente ServiceSoap12Stub.java

Componente generado automáticamente por la herramienta Apache AXIS[1] a partir del lenguaje de definición del Web Service WSDL que permitirá invocar al mismo.

3.4.3.4.17. Componente ServiceSoapStub.java

Componente generado automáticamente por la herramienta Apache AXIS[1] a partir del lenguaje de definición del Web Service WSDL que permitirá invocar al mismo.

---

[1] WSDL2Java Axis

3.4.3.4.18. Componente Structure.java

Componente que contiene la información de una Estructura[1].

3.4.3.4.19. Componente Synthetic.java

Componente que contiene la información de Sintético[2].

3.4.3.4.20. Componente Tester.java

Componente que permite al Probador iniciar la ejecución del Benchmark.

3.4.3.4.21. Componente WSTestClient.java

Componente en el que se encuentra el método *main* del Benchmark, es la responsable de inicializar los parámetros del mismo y recolectar los resultados de las mediciones y los cálculos realizados durante el tiempo que dura la prueba.

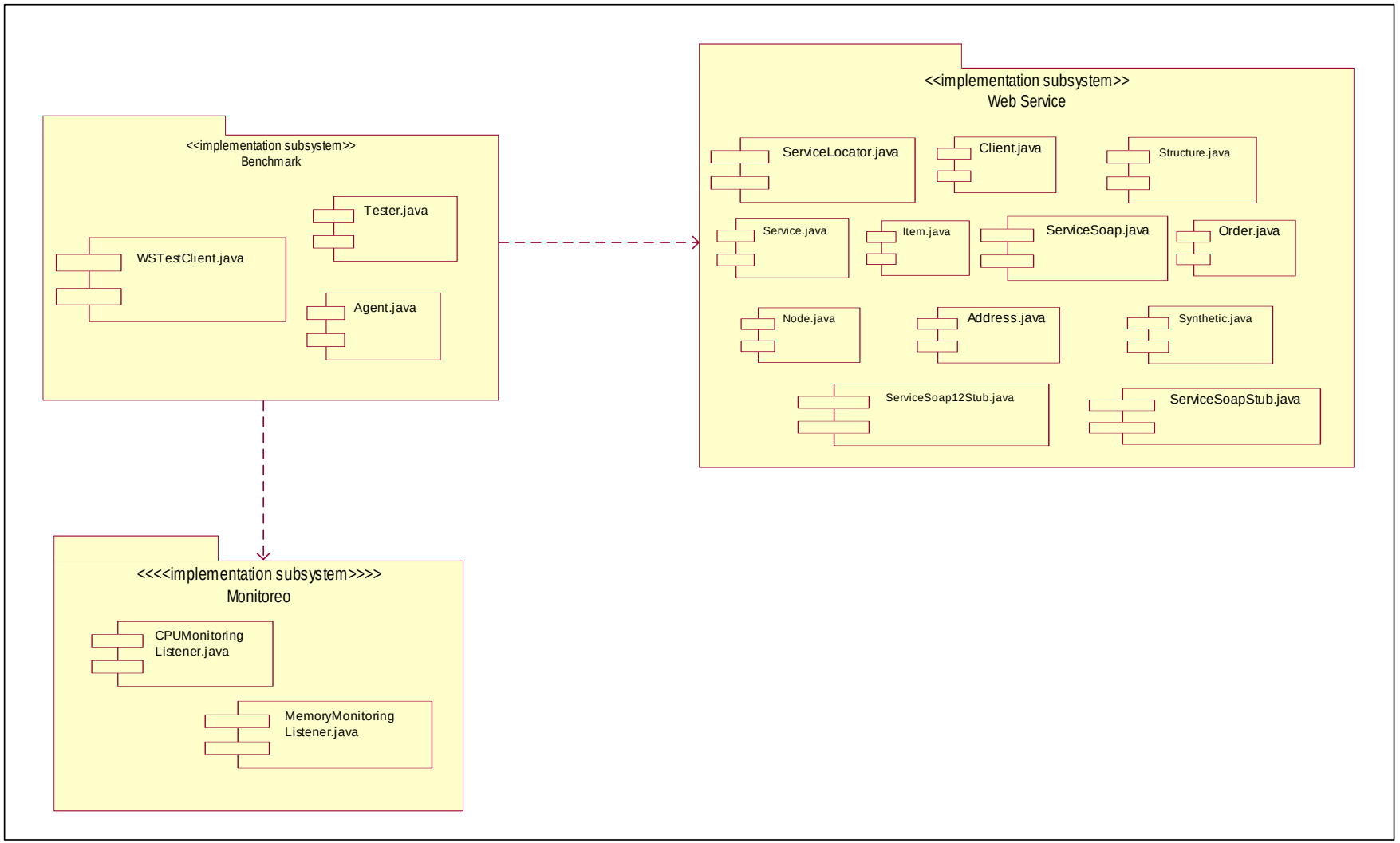
3.4.3.5. Subsistemas de Implementación

Los subsistemas de implementación han sido identificados a partir de los subsistemas de diseño y de la relación existente entre los componentes de implementación, en el siguiente diagrama se muestra la forma en que se encuentran distribuidos para organizarlos adecuadamente:

[1] Ver sección 3.1.1.2

[2] Ver sección 3.1.1.4

Figura 3-21 Diagrama de subsistemas de implementación



Elaboración y Fuente: Los Autores

A continuación se describe la funcionalidad de cada uno de los subsistemas de implementación identificados

3.4.3.5.1. Subsistema Benchmark

Este subsistema agrupa a todos los componentes que permiten la ejecución del Benchmark y la recolección de resultados de los índices del web service y del monitoreo de CPU y Memoria.

El Subsistema Benchmark tiene los siguientes componentes:

- Agent.java
- Tester.java
- WSTestClient.java

3.4.3.5.2. Subsistema Monitoreo

Este subsistema agrupa a todos los componentes que permiten realizar el monitoreo de CPU y Memoria mientras se realiza la prueba.

El Subsistema Monitoreo tiene los siguientes componentes:

- CPUMonitoringListener.java
- MemoryMonitoringListener.java

3.4.3.5.3. Subsistema Web Service

Este subsistema agrupa a todos los componentes que permiten invocar a los métodos del web service.

El Subsistema Web Service tiene los siguientes componentes:

- Address.java
- Client.java
- Item.java
- Node.java
- ServiceLocator-java
- ServiceSoap.java

- ServiceSoap12.java
- ServiceSoapStub.java
- Structure.java
- Synthetic.java

3.4.3.6. Descripción de la Arquitectura (Vista del modelo de Implementación)

La aplicación que implementa el Benchmark para realizar el estudio comparativo entre web services desarrollados en las plataformas Java y .Net, se encuentra desarrollado adoptando el paradigma de la programación orientada a objetos, mediante el uso de la herramienta Java, seleccionada como la más adecuada en la sección 3.4.1.1, por sus fortalezas en orientación a objetos, portabilidad, multithreading y soporte para aplicaciones que funcionen dentro de una arquitectura orientada a servicios.

Los componentes más importantes dentro del funcionamiento del Benchmark son los siguientes:

- Agent: Sirve para que el componente WSTestClient envíe el número de threads (simulación de carga) con los que se realizará la prueba.
- WSTestClient: Este componente es el que se encarga de la ejecución del Benchmark, así como de solicitar el monitoreo de CPU y Memoria y la evaluación del web service, además de esto recopila todos los resultados generados.

3.4.3.7. Plan de integración de construcciones

La construcción del Benchmark se la realizará en dos iteraciones.

3.4.3.7.1. Primera Iteración

3.4.3.7.1.1. Funcionalidades a Implementar

En la primera iteración de la construcción se implementarán los componentes que permitirán al Benchmark cumplir con los servicios:

- Ejecutar Benchmark
- Evaluar web service

- Obtener Resultados

3.4.3.7.1.1.1. Componentes a desarrollar

Se desarrollarán los siguientes componentes en esta iteración:

- Agent.java
- Client.java
- Item.java
- Node.java
- Order.java
- Service.java
- ServiceLocator.java
- ServiceSoap.java
- ServiceSoap12Stub.java
- ServiceSoapStub.java
- Structure.java
- Synthetic.java
- Tester.java
- WSTestClient.java

3.4.3.7.2. Segunda Iteración

3.4.3.7.2.1. Funcionalidades a Implementar

En la segunda iteración de la construcción se implementarán los componentes que permitirán al Benchmark cumplir con el servicio de monitoreo de CPU y Memoria

3.4.3.7.2.1.1. Componentes a desarrollar

En esta iteración se desarrollarán los siguientes componentes:

- CPUMonitoringListener.java
- MemoryMonitoringListener.java

Una vez realizadas las iteraciones el Benchmark debe ser capaz de evaluar el web service y monitorear CPU y Memoria mientras esto se realiza. Además debe almacenar los resultados obtenidos en los archivos planos.

3.4.4. PRUEBAS DEL BENCHMARK

Después de terminar la fase de implementación del Benchmark, se procederá a la verificación de la implementación de los requisitos funcionales definidos en la

sección 3.1. También se probará que los servicios que provee el Benchmark arrojen los resultados esperados.

3.4.4.1. Modelo de Pruebas

Para el desarrollo de las pruebas del Benchmark se analizará cada caso de uso especificado en la sección 3.1.2.1 del capítulo 3, estos casos de uso se transforman en casos de prueba dentro del modelo de pruebas. En base a esto se tiene lo siguientes casos de prueba:

3.4.4.1.1. Caso de Prueba Ejecutar Benchmark

El objetivo de este caso de prueba será verificar que el Benchmark cargue el archivo de propiedades correctamente.

3.4.4.1.1.1. Procedimiento de Prueba

Tabla 3-20 Procedimiento de prueba del caso de prueba Ejecutar Benchmark

Objetivo de la Prueba	Prerrequisitos	Resultados Esperados	Procedimiento de Prueba
Verificar que el Benchmark cargue el archivo de propiedades correctamente	Debe existir el archivo de propiedades con los parámetros para ejecutar las pruebas.	El Benchmark cargue correctamente el archivo	Correr el archivo ejecutable del Benchmark y esperar el mensaje que informa que los parámetros han sido cargados.

Elaboración y Fuente: Los Autores

3.4.4.1.1.2. Componente de Prueba

No existe ningún componente de prueba que se necesite utilizar, ya que la ejecución del Benchmark lo hace el probador de forma manual.

3.4.4.1.1.3. Plan de Prueba

Para el desarrollo de esta prueba se requieren dos computadoras interconectadas entre si mediante una red, en un computador deben estar corriendo los web services y en el otro computador debe estar corriendo el Benchmark, debe existir un usuario que realice la prueba en base a los procedimientos definidos anteriormente.

3.4.4.1.1.4. Defecto

Tabla 3-21: Resultados Obtenidos caso de prueba Ejecutar Benchmark

Objetivo de la Prueba	Resultados Obtenidos
Verificar que el Benchmark cargue el archivo de propiedades correctamente	El archivo de propiedades se cargo correctamente.

Elaboración y Fuente: Los Autores

De acuerdo a los resultados obtenidos en base a los procedimientos definidos en la sección 3.4.4.1.1, se concluye que no existen defectos ni errores en este caso de prueba, ya que los resultados que se obtuvieron fueron los que se esperaban.

3.4.4.1.1.5. Evaluación de Prueba

Los resultados obtenidos por el caso de prueba son los resultados esperados, por lo que cumple con los objetivos planteados.

3.4.4.1.2. Caso de Prueba Evaluar Web Service

3.4.4.1.2.1. Plan de Prueba

Los objetivos de este caso de prueba serán:

- Medir el número de peticiones
- Calcular el tiempo de respuesta
- Calcular el porcentaje de disponibilidad

3.4.4.1.2.2. Procedimiento de Prueba

Tabla 3-22 Procedimiento de prueba caso de prueba Evaluar Web Service

Objetivo de la Prueba	Prerrequisitos	Resultados Esperados	Procedimiento de Prueba
Medir el número de peticiones, tiempo de respuesta, calcular el porcentaje de disponibilidad.	El Benchmark debe estar en ejecución	El número de peticione ha sido medido y se ha calculado el tiempo de respuesta y la disponibilidad del web service	Correr el archivo ejecutable del Benchmark y esperar el mensaje que informa que los parámetros han sido cargados, una vez realizados estos, el Benchmark se encarga de crear el número de threads que se especificó en el archivo de propiedades, los crea y en base a cada thread creado empieza a realizar la toma de mediciones y los cálculos respectivos.

Elaboración y Fuente: Los Autores

3.4.4.1.2.3. Componente de Prueba

No existe ningún componente de prueba que se necesite utilizar ya que la ejecución del Benchmark lo hace el probador de forma manual.

3.4.4.1.2.4. Plan de Prueba

Para el desarrollo de esta prueba se requieren dos computadoras interconectadas entre si mediante una red, en un computador deben estar corriendo los web services y en el otro computador debe estar corriendo el Benchmark, debe existir un usuario que realice la prueba en base a los procedimientos definidos anteriormente.

3.4.4.1.2.5. Defecto

Tabla 3-23: Resultados obtenidos caso de prueba Evaluar Web Service

Objetivo de la Prueba	Resultados Obtenidos
Medir el número de peticiones atendidas, tiempo de respuesta, calcular la disponibilidad.	El número de peticiones atendidas han sido medidas y se ha calculado el tiempo de respuesta y la disponibilidad del web service, las mediciones y cálculos se realizaron satisfactoriamente

Elaboración y Fuente: Los Autores

De acuerdo a los resultados obtenidos en base a los procedimientos definidos en la sección 3.4.4.1.2, se concluye que no existen defectos ni errores en este caso de prueba, ya que los resultados que se obtuvieron fueron los que se esperaban.

3.4.4.1.2.6. Evaluación de Prueba

Los resultados obtenidos por el caso de prueba son los resultados esperados, por lo que cumple con los objetivos planteados.

3.4.4.1.3. Caso de Prueba Monitorear CPU y Memoria

3.4.4.1.3.1. Plan de Prueba

Los objetivos de este caso de prueba serán:

- Monitorear CPU
- Monitorear Memoria

3.4.4.1.3.2. Procedimientos de Prueba

Tabla 3-24 Procedimiento de prueba caso de Monitorear CPU y Memoria

Objetivo de la Prueba	Prerrequisitos	Resultados Esperados	Procedimiento de Prueba
Monitorear CPU y Memoria	El Benchmark debe estar en ejecución	Los contadores de CPU: USER_CPU, IDLE_CPU y TOTAL_CPU han sido medidos al igual que los contadores de Memoria USED_MEMORY	Correr el archivo ejecutable del Benchmark y esperar el mensaje que informa que los parámetros han sido cargados, una vez realizado esto, el Benchmark se encarga de crear el número de threads que se especificó en el archivo de propiedades, al mismo tiempo empieza el monitoreo de CPU y Memoria

Elaboración y Fuente: Los Autores

3.4.4.1.3.3. Componente de Prueba

No existe ningún componente de prueba que se necesite utilizar ya que la ejecución del Benchmark lo hace el probador de forma manual.

3.4.4.1.3.4. Plan de Prueba

Para el desarrollo de esta prueba se requieren dos computadoras interconectadas entre si mediante una red, en un computador deben estar corriendo los web services y en el otro computador debe estar corriendo el Benchmark, debe existir un usuario que realice la prueba en base a los procedimientos definidos anteriormente

3.4.4.1.3.5. Defecto

Tabla 3-25 Resultados obtenidos caso de prueba Monitorear CPU y Memoria

Objetivo de la Prueba	Resultados Obtenidos
Monitorear CPU y Memoria	Los contadores de CPU: USER_CPU, IDLE_CPU y TOTAL_CPU han sido medidos al igual que los contadores de Memoria USED_MEMORY, las mediciones han sido tomadas satisfactoriamente

Elaboración y Fuente: Los Autores

De acuerdo a los resultados obtenidos en base a los procedimientos definidos en la sección 3.4.4.1.3, se concluye que no existen defectos ni errores en este caso de prueba, ya que los resultados que se obtuvieron fueron los que se esperaban.

3.4.4.1.3.6. Evaluación de Prueba

Los resultados obtenidos por el caso de prueba son los resultados esperados, por lo que cumple con los objetivos planteados.

3.4.4.1.4. Caso de Prueba Obtener Resultados

3.4.4.1.4.1. Plan de Prueba

Los objetivos de este caso de prueba serán:

- Obtener los resultados de la evacuación del web service.
- Obtener los resultados de los contadores de CPU y Memoria.

3.4.4.1.4.2. Procedimientos de Prueba

Tabla 3-26 Procedimiento de prueba caso de prueba Obtener Resultados

Objetivo de la Prueba	Prerrequisitos	Resultados Esperados	Procedimiento de Prueba
Obtener los resultados	El Benchmark debe estar en ejecución	El archivo probador.resultados debe contener la información de la medición del número de peticiones atendidas, cálculo del tiempo de respuesta y disponibilidad, de la misma manera los archivos CPU.resultados debe contener la información de los índices de CPU y el Memoria.resultados, debe contener la información de los índices de Memoria medidas obtenidas y los cálculos	Correr el archivo ejecutable del Benchmark y esperar el mensaje que informa que los parámetros han sido cargados, una vez realizado esto, el Benchmark se encarga de crear el número de threads que se especificó en el archivo de propiedades, al mismo tiempo que evalúa el web service se empieza el monitoreo de CPU y Memoria, lo va almacenando durante la realización de la prueba en intervalos de un segundo, los resultados se guardan en el archivo CPU.resultados y Memoria.resultados respectivamente

Elaboración y Fuente: Los Autores

3.4.4.1.4.3. Componente de Prueba

No existe ningún componente de prueba que se necesite utilizar ya que la ejecución del Benchmark lo hace el probador de forma manual.

3.4.4.1.4.4. Plan de Prueba

Para el desarrollo de esta prueba se requieren dos computadoras interconectadas entre si mediante una red, en un computador deben estar corriendo los web services y en el otro computador debe estar corriendo el Benchmark, debe existir un usuario que realice la prueba en base a los procedimientos definidos anteriormente

3.4.4.1.4.5. Defecto

Tabla 3-27 Resultados obtenidos caso de prueba Obtener Resultados

Objetivo de la Prueba	Resultados Obtenidos
Obtener resultados	Los resultados de las mediciones y cálculos han sido obtenido satisfactoriamente

Elaboración y Fuente: Los Autores

De acuerdo a los resultados obtenidos en base a los procedimientos definidos en el sección 3.4.4.1.4, se concluye que no existen defectos ni errores en este caso de prueba, ya que los resultados que se obtuvieron fueron los que se esperaban.

3.4.4.1.4.6. Evaluación de Prueba

Los resultados obtenidos por el caso de prueba son los resultados esperados, por lo que cumple con los objetivos planteados.

3.5. PLANIFICACIÓN DE PRUEBAS PARA COMPARACIÓN

La planificación de pruebas para comparación tienen como objetivo principal obtener los índices que nos permitirán evaluar el web service utilizando sesiones simultáneas y sostenidas para los servicios específicos: (*Número de Peticiones Atendidas, Tiempo de Respuesta, Disponibilidad* así como también los contadores de CPU: USER_CPU, IDLE_CPU, TOTAL_CPU y Memoria USED_MEMORY).

Se generarán peticiones a l servicio web en paralelo desde una máquina cliente (Benchmark) simulando ser un usuario del web service que realizará una o varias invocaciones a los diferentes métodos del web service (*RespuestaVacio, RespuestaEstructura, RespuestaListaEnlazada, RespuestaSintetico y RespuestaOrden*), hasta lograr estresar al servicio sometido a prueba, para nuestro caso el web service.

3.6. APLICACIÓN DEL BENCHMARK

3.6.1. INSTALACIÓN

En la siguiente tabla se muestran las configuraciones de hardware, tanto para el servidor, como para el cliente:

Tabla 3-28 Configuraciones de Hardware

Equipo	Elemento	Valor
Servidor	Procesador	Pentium 4 2.40 GHz
	Memoria	512 MB
	Disco Duro	80 GB
	Tarjeta de Red	10/100 Mbps
Cliente	Procesador	Centrino 1.73 GHz
	Memoria	512 MB
	Disco Duro	60 GB
	Tarjeta de Red	10/100 Mbps

Elaboración y Fuente: Los Autores

Las configuraciones de software para cliente y servidor se las presenta en la tabla siguiente:

Tabla 3-29 Configuraciones de software

Equipo	Tipo de Aplicación	Nombre
Servidor	Sistema Operativo	Microsoft Windows Server 2003 Standard Edition Service Pack 1
	Servidor de Aplicaciones	Internet Information Server Framework 2.0
		JBOSS 4.0.4
		Tomcat 5.0 for JWSDP
		OIAS 9.0.3
		WebLogic 8.1
Cliente	Sistema Operativo	Microsoft Windows XP Professional Service Pack 2
	Plataforma de desarrollo	JDK 1.5

Elaboración y Fuente: Los Autores

3.6.2. DETALLE DE LA INSTALACIÓN Y CONFIGURACIÓN

El procedimiento de Instalación y Configuración del Benchmark, y de los servidores de aplicaciones, se detalla en el Manual de Instalación, Anexo 4.

3.6.3. AMBIENTE DE PRUEBAS

Para realizar las pruebas sobre la plataforma .NET se utilizará la única implementación existente:

- Internet Information Server Framework 2.0

Cómo consecuencia del éxito del lenguaje de programación Java, el término servidor de aplicaciones usualmente hace referencia a un servidor de aplicaciones J2EE, cómo por ejemplo los de las casas comerciales: Oracle Corporation, Bea Systems, así como también servidores de aplicaciones de libre licenciamiento muy conocidos en la actualidad cómo los de Apache Software Foundation.

Todas las implementaciones desarrolladas por las casas anteriormente mencionadas se ajustan a los estándares que provee J2EE para que un servidor de aplicaciones pueda ser considerado como tal dentro de la plataforma Java.

Para realizar las pruebas sobre la plataforma Java, hemos considerado los siguientes servidores de aplicaciones:

- OIAS 9.0.3(Oracle Corporation)
- WebLogic 8.1 (Bea Systems)
- JBOSS 4.0.4 (Apache Software Foundation)
- Tomcat 5.0 for JWSDP (Apache Software Foundation)

3.6.3.1. Definición de escenarios para la realización de pruebas con los servidores de aplicaciones en las plataformas Java y .NET

Para realizar las pruebas de comparación entre servidores de aplicaciones se definen los siguientes escenarios:

- La primera prueba distribuye la carga equitativamente entre los métodos: RespuestaVacio, RespuestaListaEnlazada, RespuestaEstructura, RespuestaSintetico y RespuestaOrden es decir cada una constituye el 20% del 100% de la prueba.
- La segunda prueba asigna el 100% de la carga al método RespuestaVacio, quedando el resto de métodos con 0% de carga.
- La tercera prueba asigna el 100% de la carga al método RespuestaListaEnlazada, quedando el resto de métodos con 0% de carga.
- La cuarta prueba asigna el 100% de la carga al método RespuestaEstructura, quedando el resto de métodos con 0% de carga.
- La quinta prueba asigna el 100% de la carga al método RespuestaSintetico, quedando el resto de métodos con 0% de carga.
- La sexta prueba asigna el 100% de la carga al método RespuestaOrden, quedando el resto de métodos con 0% de carga.

Para todos los escenarios anteriormente descritos se utilizarán los siguientes parámetros:

- Agentes: 10, 20, 30, 40, 50 y 60.
- Tiempo Ascenso 10 segundos.
- Tiempo Estabilización 80 segundos.
- Tiempo Descenso 10 segundos.
- Tamaño de la Lista 100.
- Número de Bytes 1024.

Cabe destacar que se han definido estos tiempos tomando en cuenta los estados nuevo thread, correspondiente al tiempo de ascenso, significa que el sistema, aún no asignado ningún recurso para él.

El estado ejecutable, correspondiente al tiempo de estabilización, dónde se asignan los recursos del sistema para la ejecución del mismo.

El estado muerto, correspondiente al tiempo de descenso, significa que el thread ha terminado de ejecutar sus instrucciones cumpliendo con su ciclo de vida.

Cada prueba se la realizará tres veces con la finalidad de obtener los suficientes datos para poder aplicar estadígrafos.

3.6.3.2. Selección de Servidor de Aplicaciones para la plataforma Java

Para seleccionar el servidor de aplicaciones para la plataforma Java, se ha tomado en cuenta los datos y los gráficos resultantes de la primera prueba descrita en el apartado anterior a manera de ejemplo.

Se realizó un gráfico comparativo entre los servidores de aplicaciones de la plataforma Java, por cada uno de los índices: número de llamadas atendidas, tiempo de respuesta y disponibilidad con la finalidad de determinar cuál es el más robusto.

3.6.3.2.1. Número de Peticiones Atendidas

Los resultados para el índice *Número de llamadas atendidas* obtenidos en esta prueba se los han tabulado y se los presenta a continuación:

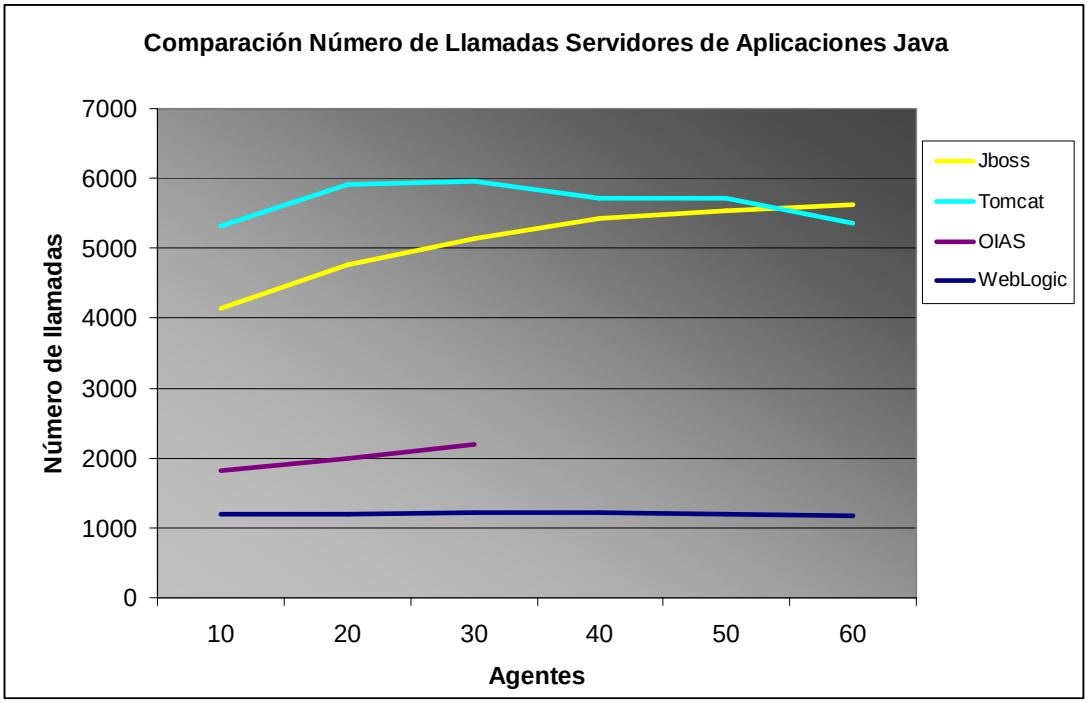
Tabla 3-30 Número de peticiones atendidas por los servidores de aplicaciones de la plataforma Java

Agentes	Jboss	Tomcat	OIAS	WebLogic
10	4148	5309	1822	1192
20	4752	5904	1996	1203
30	5146	5952	2204	1221
40	5422	5709	x	1208
50	5531	5707	x	1203
60	5620	5357	x	1178

Elaboración y Fuente: Los Autores

En la figura 3-21 se muestra el gráfico del comportamiento de los servidores de aplicaciones respecto a este índice.

Figura 3-22 Comparación Número de llamadas entre servidores de aplicaciones Java



Elaboración y Fuente: Los Autores

En el eje de las abscisas encontramos el número de agentes, mientras en el eje de las ordenadas se encuentra representado el número de llamadas atendidas durante la prueba.

El número de llamadas atendidas por el servidor de aplicaciones WebLogic tiende a crecer hasta llegar a su máximo valor, a partir de este empieza a decrecer el número de llamadas mientras la carga aumenta.

El número de llamadas atendidas por el servidor de aplicaciones OIAS tiende a crecer hasta llegar a su máximo valor, a partir de este, el servidor de aplicaciones no respondió a un número mayor de carga.

El número de llamadas atendidas por el servidor de aplicaciones JBOSS tiende a crecer conforme se va aumentando la carga.

El número de llamadas atendidas por el servidor de aplicaciones TOMCAT tiende a crecer hasta llegar a su máximo valor, a partir de este, empieza a decrecer el número de llamadas mientras la carga aumenta.

3.6.3.2.2. Tiempo de Respuesta

Los resultados para el índice *Tiempo de respuesta* obtenidos en esta prueba se los han tabulado y se los presenta a continuación:

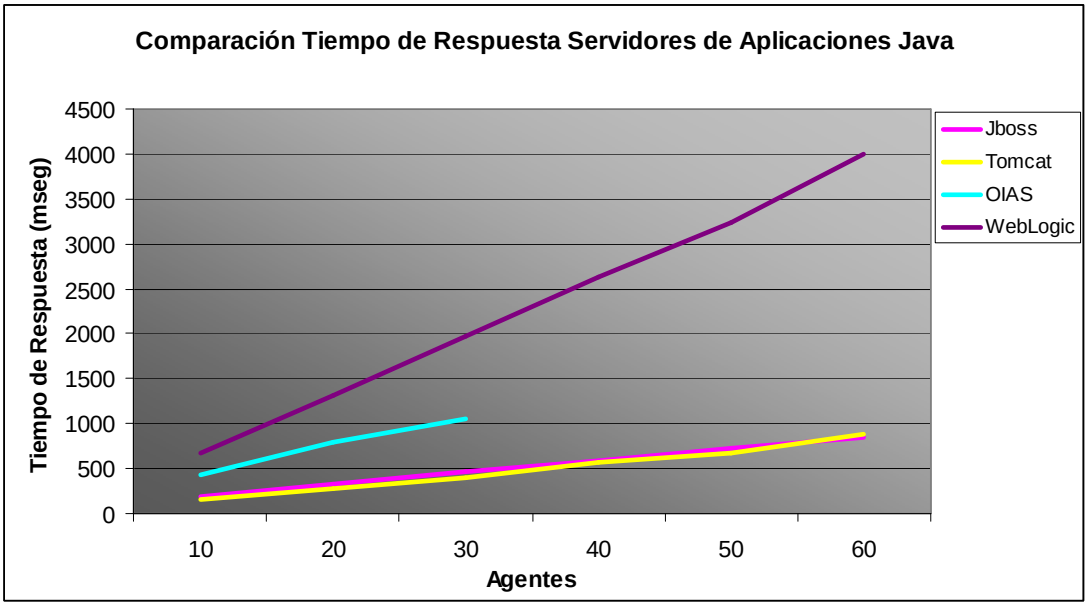
Tabla 3-31 Tiempo de respuesta para los servidores de aplicaciones Java

Agentes	Jboss	Tomcat	OIAS	WebLogic
10	192	149	439	671
20	337	271	801	1324
30	463	402	1054	1974
40	586	564	x	2639
50	719	679	x	3237
60	854	886	x	4003

Elaboración y Fuente: Los Autores

En la figura 3-22 se muestra el gráfico del comportamiento de los servidores de aplicaciones respecto a este índice.

Figura 3-23 Comparación del tiempo de respuesta para los servidores de aplicaciones Java



Elaboración y Fuente: Los Autores

En el eje de las abscisas encontramos el número de agentes, mientras en el eje de las ordenadas se encuentra representado el tiempo de respuesta en milisegundos durante la prueba.

El tiempo de respuesta en los servidores de aplicaciones TOMCAT, JBOSS y WebLogic tienden a crecer mientras la carga aumenta, siendo JBOSS el que aumenta en intervalos más cortos de tiempo respecto al resto.

El tiempo de respuesta del servidor de aplicaciones OIAS tiende a crecer hasta llegar a su máximo valor, a partir de este, el servidor de aplicaciones no respondió a un número mayor de carga.

3.6.3.2.3. Disponibilidad

Los resultados para el índice *Disponibilidad* obtenidos en esta prueba se los han tabulado y se los presenta a continuación:

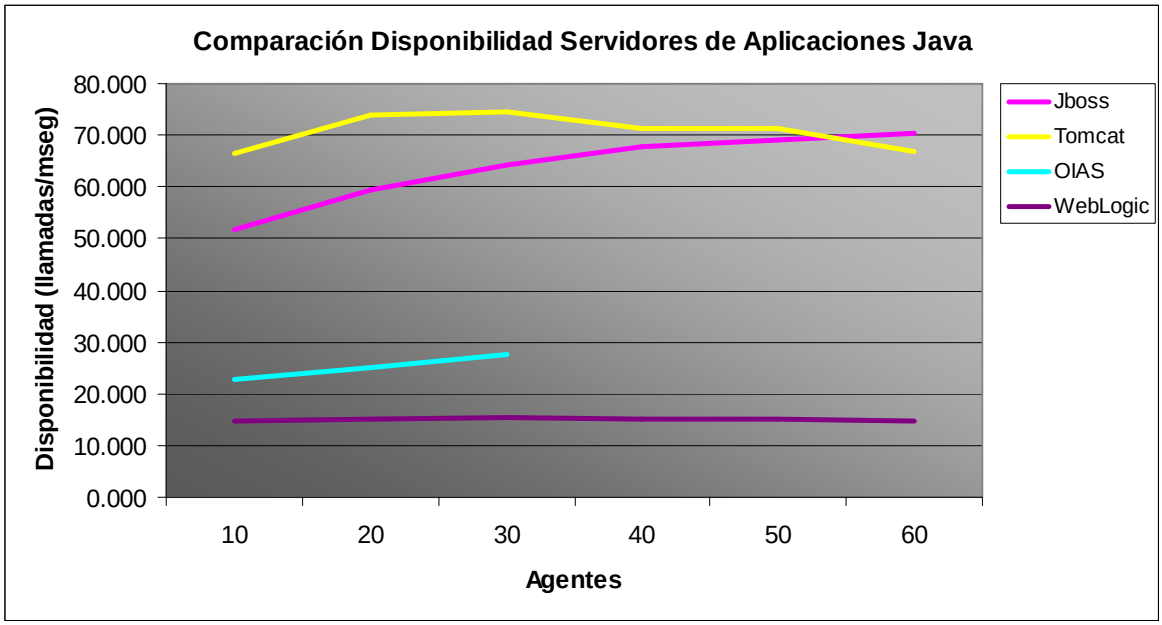
Tabla 3-32 Disponibilidad para los servidores de aplicaciones Java

Agentes	Jboss	Tomcat	OIAS	WebLogic
10	51.850	66.363	22.775	14.900
20	59.400	73.800	24.950	15.038
30	64.325	74.400	27.550	15.262
40	67.775	71.363	x	15.100
50	69.138	71.337	x	15.037
60	70.25	66.963	x	14.725

Elaboración y Fuente: Los Autores

En la figura 3-23 se muestra el gráfico del comportamiento de los servidores de aplicaciones respecto a este índice.

Figura 3-24 Comparación de Disponibilidad de los servidores de aplicaciones Java



Elaboración y Fuente: Los Autores

En el eje de las abscisas encontramos el número de agentes, mientras en el eje de las ordenadas se encuentra representada la disponibilidad durante la prueba.

La disponibilidad del servidor de aplicaciones WebLogic tiende a crecer hasta llegar a su máximo valor, a partir de este empieza a decrecer el número de llamadas mientras la carga aumenta.

La disponibilidad del servidor de aplicaciones OIAS tiende a crecer hasta llegar a su máximo valor, a partir de este, el servidor de aplicaciones no respondió a un número mayor de carga.

La disponibilidad del servidor de aplicaciones JBOSS tiende a crecer conforme se va aumentando la carga.

La disponibilidad del servidor de aplicaciones TOMCAT tiende a crecer hasta llegar a su máximo valor, a partir de este, empieza a decrecer el número de llamadas mientras la carga aumenta.

Con el análisis realizado podemos inferir que el servidor de aplicaciones JBOSS es el más robusto puesto que al aumentar la carga este permanece disponible permitiendo de esta manera la atención de llamadas sin afectar los intervalos en los que aumenta el tiempo de respuesta.

Las tablas comparativas de las demás pruebas de cada uno de los índices se adjuntan en Anexo 5.

Los gráficos de las demás pruebas de cada uno de los índices se adjuntan en el Anexo 6.

3.7. MEDICIÓN DE ÍNDICES Y GENERACIÓN DE RESULTADOS

Los índices medidos serán almacenados dentro de tres archivos planos, en el primero se almacenarán los datos correspondientes a, Número de Peticiones atendidas, Tiempo de Respuesta y Porcentaje de disponibilidad. Este archivo tiene por nombre “probador.resultados” y la estructura se presenta a continuación:

- Número de Llamadas
- Respuesta en msec
- Carga Estabilizacion [llamadas/s]

En el segundo archivo llamado “CPU.resultados” se almacenarán los índices correspondientes a CPU, la estructura se presenta a continuación:

- USER
- NICE
- SYSTEM
- IDLE
- IRQ
- SOFT_IRQ
- STEAL
- IO
- TOTAL

En el tercer archivo llamado “Memoria.resultados” se almacenarán los índices correspondientes a CPU, la estructura se presenta a continuación:

- TOTAL_MEM
- USED
- FREE
- BUFFERS
- CACHED
- TOTALSWP
- USEDSWP
- FREESWP
- CACHEDSWP

Los archivos han sido diseñados de tal manera que puedan ser abiertos con hojas de cálculo, lo cual permite elaborar gráficos representativos, brindando así una mayor facilidad para realizar el estudio planteado.

En el Anexo 7 se encuentran los resultados de las pruebas realizadas.

CAPITULO 4. ANÁLISIS DE RESULTADOS

Para evaluar correctamente los datos tabulados, es necesario expresarlos mediante gráficos estadísticos, en los cuales se represente de manera clara el estudio comparativo realizado.

Para realizar el análisis de los resultados obtenidos, se llevará a cabo el siguiente proceso:

- Generación de datos

Una vez obtenidos los resultados se los procesará utilizando el estadígrafo que refleje de mejor manera el comportamiento de los índices, de esta forma se obtiene un promedio entre las pruebas realizadas.

- Tabulación de resultados

Para la tabulación de datos se tendrá una tabla por cada uno de los índices evaluados, que permitirá diferenciar los comportamientos de los web services publicados en los servidores de aplicaciones JBOSS para la plataforma Java y IIS para la plataforma .NET respectivamente.

Con estos resultados se elaborará un gráfico individual por servidor de aplicaciones para los índices: Número de Peticiones Atendidas, Tiempo de Respuesta y Disponibilidad en función de la carga

Para los índices: Porcentaje de uso de CPU se tendrá un gráfico por servidor de aplicaciones de los índices: Porcentaje USER, Porcentaje IDLE y TOTAL que se consideran representativos para explicar de mejor manera el uso de CPU en función del tiempo empleado para la prueba.

Para el índice: Porcentaje de uso de Memoria se tendrá un gráfico por servidor de aplicaciones del índice: USED que se consideran representativos para explicar de mejor manera el uso de Memoria en función del tiempo empleado para la prueba.

Para la comparación se han realizado gráficos de barras, uno por cada índice evaluado y que especificarán el comportamiento de cada web service en función de la carga.

Para analizar el web service tanto en el servidor de aplicaciones JBOSS como en el servidor de aplicaciones IIS, se ha tomado en cuenta los datos y los gráficos resultantes de la primera prueba descrita en la sección 3.6.3.1 de este documento a manera de ejemplo.

La prueba consiste en distribuir la carga equitativamente entre los métodos: RespuestaVacio, RespuestaListaEnlazada, RespuestaEstructura, RespuestaSintetico y RespuestaOrden, utilizando 10, 20, 30, 40, 50 y 60 Agentes respectivamente, así también los tiempos de ascenso, estabilización y descenso serán: 10. 80 y 10 segundos respectivamente.

4.1. ANÁLISIS DE RESULTADOS DEL WEB SERVICE IMPLEMENTADO EN LA PLATAFORMA JAVA

4.1.1. ÍNDICE NÚMERO DE LLAMADAS ATENDIDAS POR EL WEB SERVICE EN EL SERVIDOR DE APLICACIONES JBOSS.

La tabla 4-1 muestra en la primera columna el número de agentes, en las tres siguientes el Número de Peticiones Atendidas (Np) y en la última la media obtenida de las tres corridas.

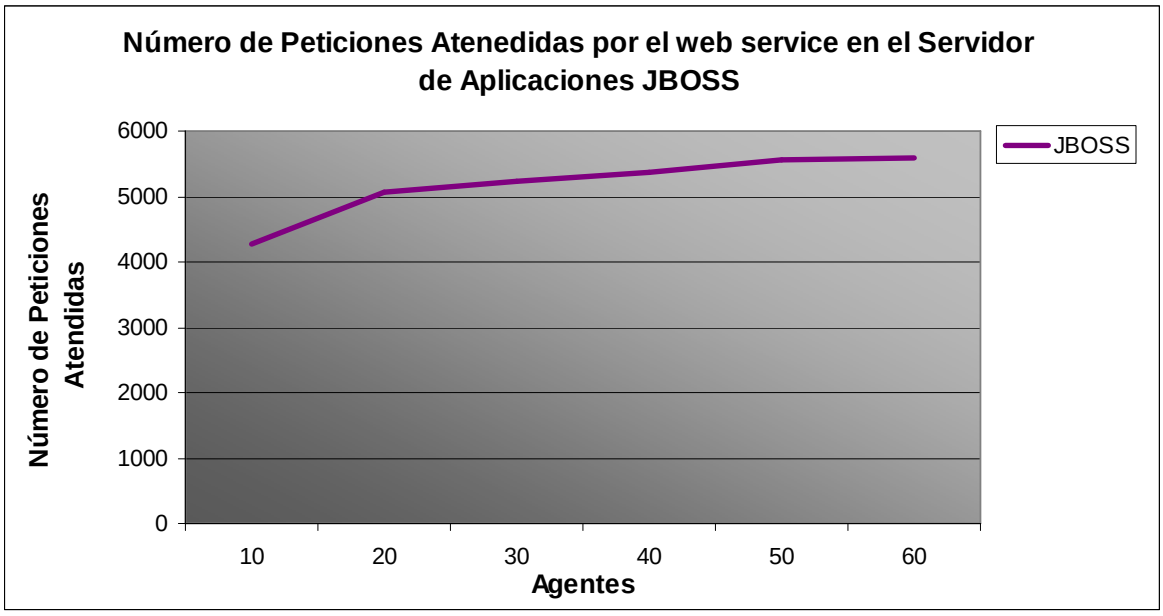
Tabla 4-1: Número de llamadas atendidas por el web service en el servidor de aplicaciones JBOSS

Agentes	1	2	3	Media ΣNp/3
10	4298	4280	4204	4261
20	5053	5089	5062	5068
30	5286	5226	5218	5243
40	5365	5386	5338	5363
50	5577	5559	5512	5549
60	5615	5592	5557	5588

Elaboración y Fuente: Los Autores

El gráfico de la figura 4-1 se generó utilizando los agentes y la media que se encuentran en la tabla 4-1.

Figura 4-1: Número de Peticiones Atendidas por el web service en el Servidor de Aplicaciones JBOSS



Elaboración y Fuente: Los Autores

En el eje de las abscisas se muestra el número de agentes que intervienen en la prueba, mientras que en el eje de las ordenadas se encuentran el Número de Peticiones Atendidas por el web service.

El grafico nos indica que el Número de Peticiones Atendidas por el web service en el servidor de aplicaciones JBOSS depende directamente de la carga que se suministre hacia el mismo.

4.1.2. ÍNDICE TIEMPO DE RESPUESTA DEL WEB SERVICE EN EL SERVIDOR DE APLICACIONES JBOSS

La tabla 4-2 muestra en la primera columna el número de agentes, en las tres siguientes el Tiempo de Repuesta (Tr) y en la última la media obtenida de las tres corridas.

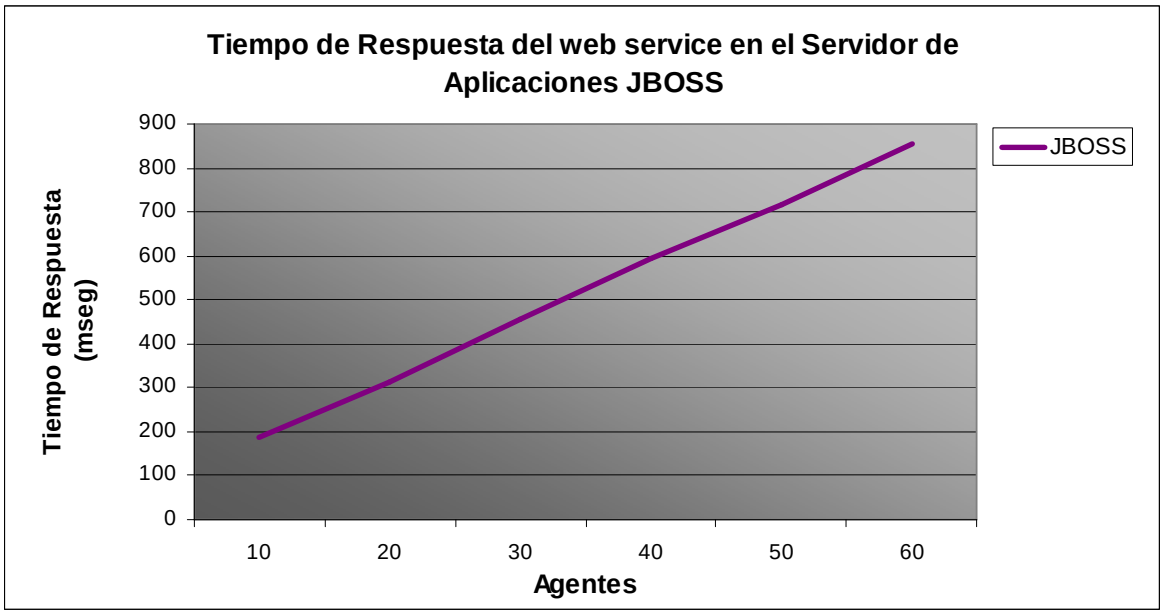
Tabla 4-2: Tiempo de Respuesta del web service en el servidor de aplicaciones JBOSS

Agentes	1	2	3	Media ΣNr/3
10	185	186	190	187
20	316	313	315	315
30	453	456	457	455
40	592	592	597	594
50	713	712	724	716
60	859	857	844	853

Elaboración y Fuente: Los Autores

El gráfico de la figura 4-2 se generó utilizando los agentes y la media que se encuentran en la tabla 4-2.

Figura 4-2: Tiempo de Respuesta del web service en el Servidor de Aplicaciones JBOSS



Elaboración y Fuente: Los Autores

En el eje de las abscisas se muestra el número de agentes que intervienen en la prueba, mientras que en el eje de las ordenadas se el Tiempo de Respuesta del web service expresado en milisegundos.

El gráfico nos indica que el Tiempo de Respuesta del web service en el servidor de aplicaciones JBOSS es directamente proporcional a la carga que se suministre hacia el mismo.

4.1.3. ÍNDICE DISPONIBILIDAD DEL WEB SERVICE EN EL SERVIDOR DE APLICACIONES JBOSS

La tabla 4-3 muestra en la primera columna el número de agentes, en las tres siguientes la Disponibilidad del web service (D) y en la última la media obtenida de las tres corridas.

Tabla 4-3: Disponibilidad del web service en el servidor de aplicaciones JBOSS

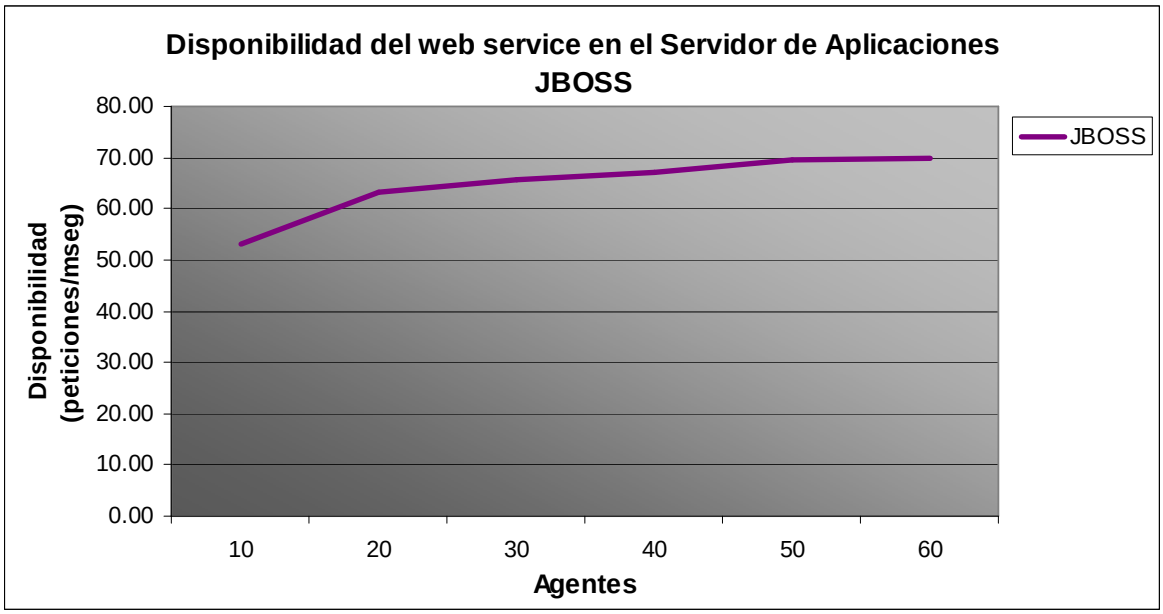
Agentes	1	2	3	Media ΣD/3
10	53.72	53.50	52.55	53.26
20	63.16	63.61	63.27	63.35
30	66.07	65.33	65.23	65.54

Agentes	1	2	3	Media ΣD/3
40	67.06	67.33	66.73	67.04
50	69.71	69.49	68.90	69.37
60	70.19	69.90	69.46	69.85

Elaboración y Fuente: Los Autores

El gráfico de la figura 4-3 se generó utilizando los agentes y la media que se encuentran en la tabla 4-3.

Figura 4-3: Disponibilidad del web service en el Servidor de Aplicaciones JBOSS



Elaboración y Fuente: Los Autores

En el eje de las abscisas se muestra el número de agentes que intervienen en la prueba, mientras que en el eje de las ordenadas la disponibilidad del web service expresada en llamadas por milisegundo.

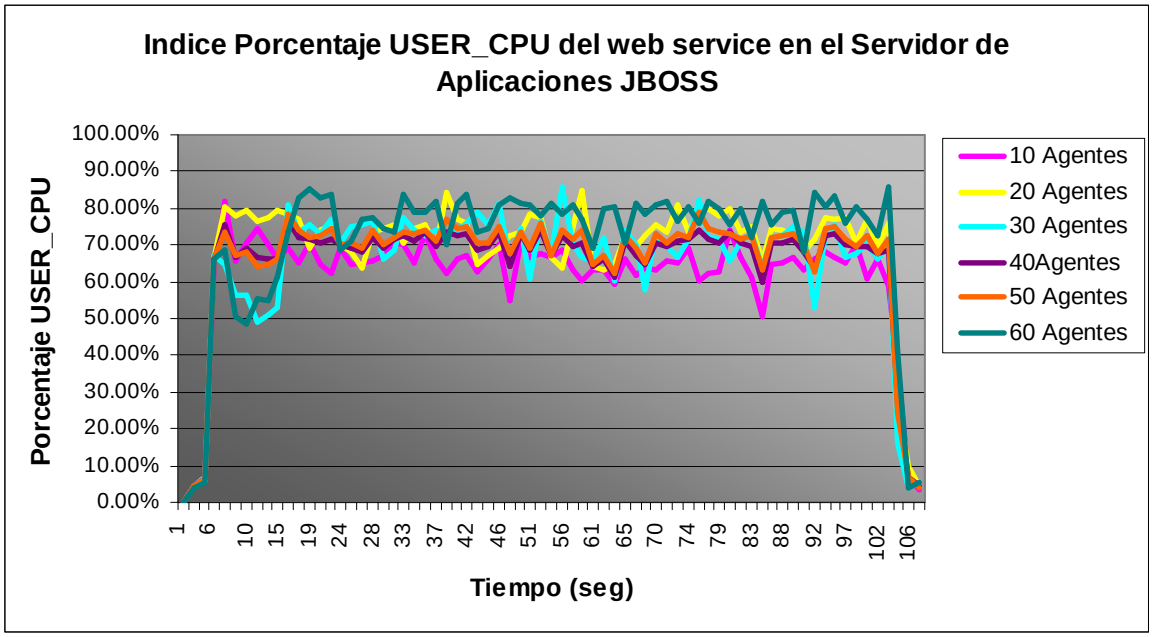
El gráfico nos indica que la disponibilidad del web service en el servidor de aplicaciones JBOSS depende directamente de la carga que se suministre hacia el mismo.

4.1.4. ÍNDICES DE DISPONIBILIDAD DE CPU DEL SERVIDOR DE APLICACIONES JBOSS

4.1.4.1. Índice USER_CPU del web service en el Servidor de Aplicaciones JBOSS

El gráfico de la figura 4-4 se generó utilizando los valores de la media del índice USER de la tabla que se encuentra en el Anexo 8.

Figura 4-4: Porcentaje USER_CPU del web service en el Servidor de Aplicaciones JBOSS



Elaboración y Fuente: Los Autores

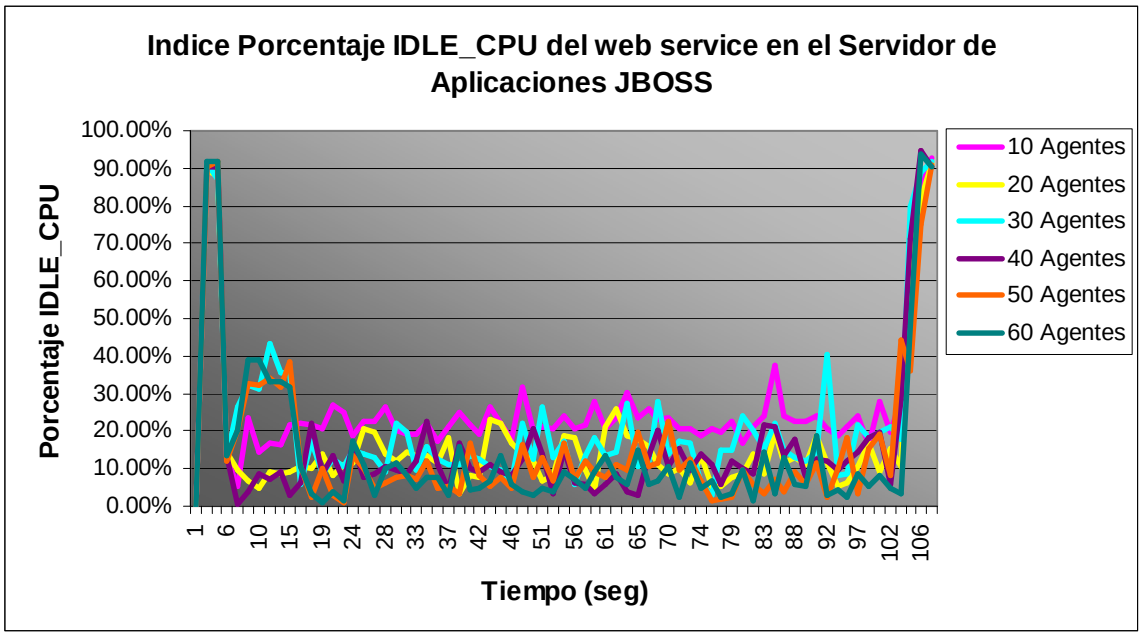
En el eje de las abscisas se muestra el tiempo expresado en segundos que intervienen en la prueba, mientras que en el eje de las ordenadas el porcentaje de tiempo que usa el web service del procesador.

El gráfico nos indica el porcentaje de tiempo que usa el procesador durante la ejecución del web service en el servidor de aplicaciones JBOSS, depende de la carga que se suministre, es decir, a medida que se aumente esta, el porcentaje de uso del procesador crecerá, caso contrario disminuirá.

4.1.4.2. Índice IDLE_CPU del web service en el Servidor de Aplicaciones JBOSS

El gráfico de la figura 4-5 se generó utilizando los valores de la media del índice IDLE de la tabla que se encuentra en el Anexo 9.

Figura 4-5: Porcentaje IDLE_CPU del web service en el Servidor de Aplicaciones JBOSS



Elaboración y Fuente: Los Autores

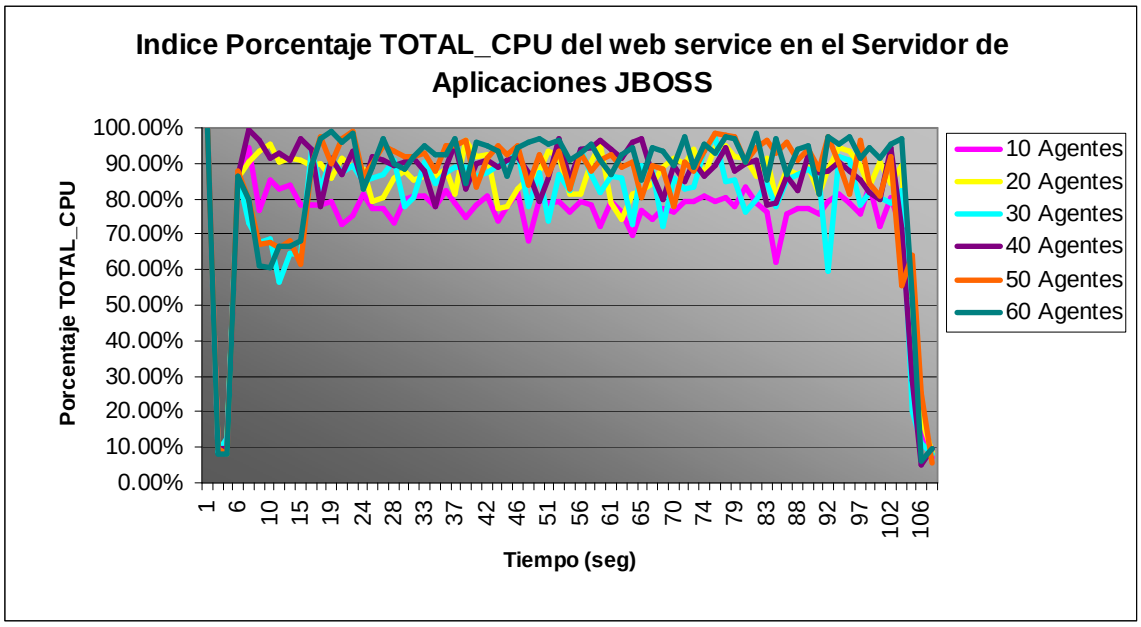
En el eje de las abscisas se muestra el tiempo expresado en segundos que intervienen en la prueba, mientras que en el eje de las ordenadas el porcentaje de idle del tiempo.

El gráfico nos indica el porcentaje de tiempo que el procesador se encuentra en espera durante la ejecución del web service en el servidor de aplicaciones JBOSS, este porcentaje depende de la carga que se suministre en forma inversa, es decir, a medida que ésta aumenta, el porcentaje de tiempo en el que el procesador se encuentra en espera es menor, en caso contrario aumentará.

4.1.4.3. Índice TOTAL_CPU del web service en el Servidor de Aplicaciones JBOSS

El gráfico de la figura 4-6 se generó utilizando los valores de la media del índice TOTAL de la tabla que se encuentra en el Anexo 10.

Figura 4-6: Porcentaje TOTAL_CPU del web service en el Servidor de Aplicaciones JBOSS



Elaboración y Fuente: Los Autores

En el eje de las abscisas se muestra el tiempo expresado en segundos que intervienen en la prueba, mientras que en el eje de las ordenadas el porcentaje de tiempo del procesador.

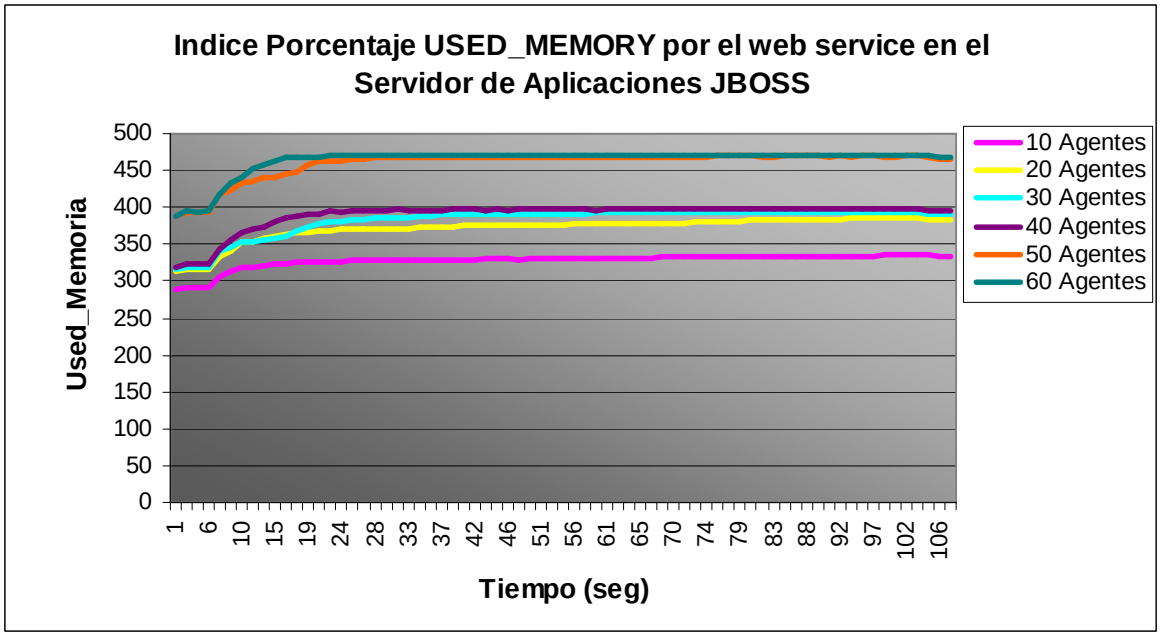
El gráfico nos indica el porcentaje de tiempo en el que el procesador permanece ocupado durante la ejecución del web service en el servidor de aplicaciones JBOSS, este porcentaje depende directamente de la carga que se suministre, es decir, a medida que ésta aumenta, el porcentaje de tiempo que el procesador permanece ocupado es mayor, en caso contrario disminuirá.

4.1.5. INDICES DE DISPONIBILIDAD DE MEMORIA EN EL SERVIDOR DE APLICACIONES JBOSS

4.1.5.1. Índice USED_MEM por el web service en el Servidor de Aplicaciones JBOSS

El gráfico de la figura 4-7 se generó utilizando los valores del índice USED de la tabla que se encuentra en el Anexo 11.

Figura 4-7: Porcentaje USED_MEMORY por el web service en el Servidor de Aplicaciones JBOSS



Elaboración y Fuente: Los Autores

En el eje de las abscisas se muestra el tiempo expresado en segundos que intervienen en la prueba, mientras que en el eje de las ordenadas la cantidad de memoria física usada.

El gráfico nos indica que la cantidad de memoria física que está siendo utilizada durante la ejecución del web service en el servidor de aplicaciones JBOSS depende de la carga que se suministre, es decir, a medida que ésta aumenta, la cantidad de memoria física ocupada es mayor, en caso contrario disminuirá.

En el Anexo 12 se adjuntan las tablas de las demás pruebas para cada índice.

En el Anexo 13 se adjuntan los gráficos de las demás pruebas para cada índice.

4.2. ANÁLISIS DE RESULTADOS DEL WEB SERVICE IMPLEMENTADO EN LA PLATAFORMA .NET

4.2.1. ÍNDICE NÚMERO DE LLAMADAS ATENDIDAS POR EL WEB SERVICE EN EL SERVIDOR DE APLICACIONES IIS.

La tabla 4-4 muestra en la primera columna el número de agentes, en las tres siguientes el Número de Peticiones Atenidas (Np) y en la última la media obtenida de las tres corridas.

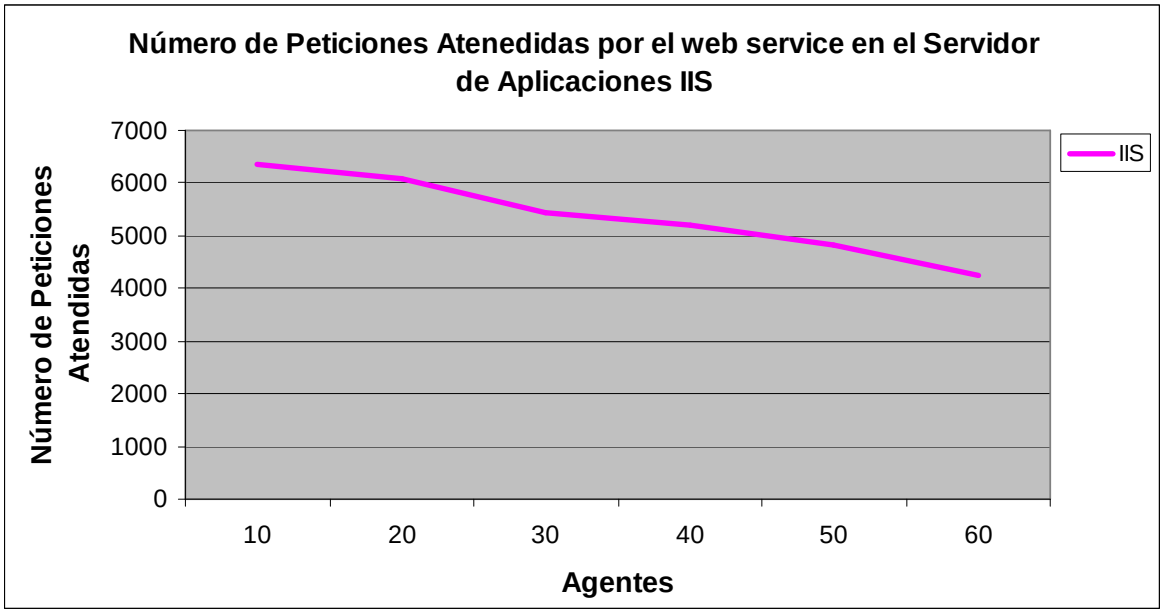
Tabla 4-4: Número de llamadas atendidas por el web service en el servidor de aplicaciones IIS.

Agentes	1	2	3	Media ΣNp/3
10	6596	6220	6276	6364
20	5989	6073	6139	6067
30	5330	5361	5614	5435
40	5185	5182	5222	5196
50	4903	4715	4881	4833
60	4008	4268	4468	4248

Elaboración y Fuente: Los Autores

El gráfico de la figura 4-8 se generó utilizando los agentes y la media que se encuentran en la tabla 4-4.

Figura 4-8: Número de Peticiones Atendidas por el web service en el Servidor de Aplicaciones IIS.



Elaboración y Fuente: Los Autores

En el eje de las abscisas se muestra el número de agentes que intervienen en la prueba, mientras que en el eje de las ordenadas se encuentran el Número de Peticiones Atendidas por el web service.

El grafico nos indica que el Número de Peticiones Atendidas por el web service en el servidor de aplicaciones IIS depende inversamente de la carga que se suministre, es decir a medida que ésta aumenta, el Número de Peticiones Atendidas disminuirá, en caso contrario aumentará.

4.2.2. ÍNDICE TIEMPO DE RESPUESTA DEL WEB SERVICE EN EL SERVIDOR DE APLICACIONES IIS

La tabla 4-5 muestra en la primera columna el número de agentes, en las tres siguientes el Tiempo de repuesta (Tr) y en la última la media obtenida de las tres corridas.

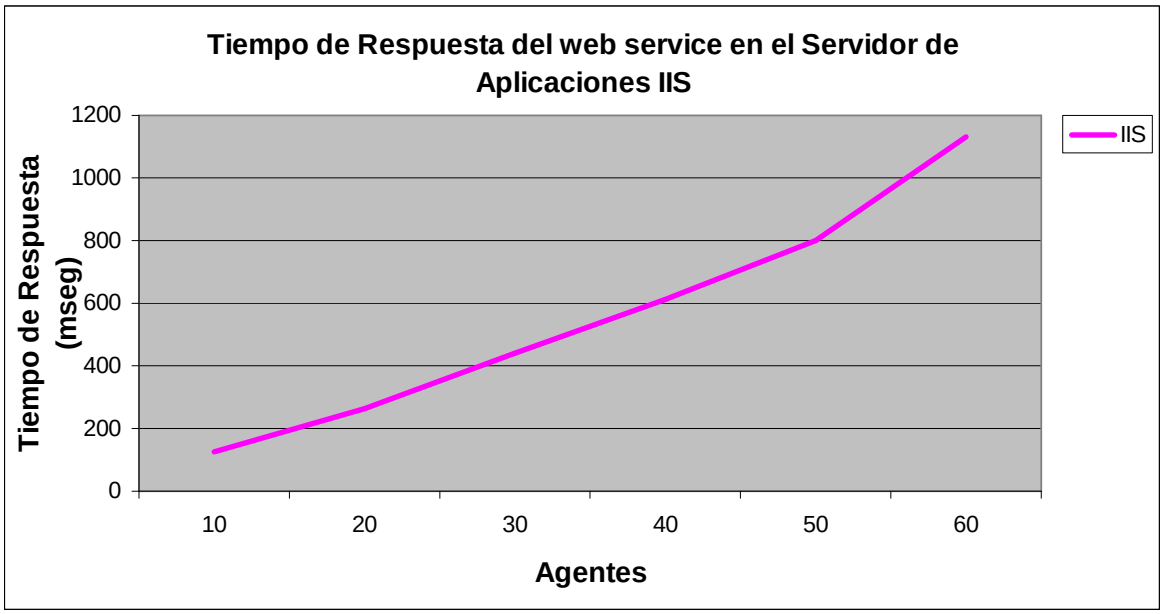
Tabla 4-5: Tiempo de Respuesta del web service en el Servidor de Aplicaciones IIS

Agentes	1	2	3	Media ΣNr/3
10	120	128	127	125
20	266	263	258	262
30	450	447	422	440
40	620	606	608	611
50	803	790	800	798
60	1217	1101	1083	1134

Elaboración y Fuente: Los Autores

El gráfico de la figura 4.9 se generó utilizando los agentes y la media que se encuentran en la tabla 4-5.

Figura 4-9: Tiempo de respuesta del web service en el Servidor de Aplicaciones IIS.



Elaboración y Fuente: Los Autores

En el eje de las abscisas se muestra el número de agentes que intervienen en la prueba, mientras que en el eje de las ordenadas se el Tiempo de Respuesta del web service expresado en milisegundos.

El gráfico nos indica que el Tiempo de Respuesta del web service en el servidor de aplicaciones IIS es depende directamente de la carga que se suministre.

En la tabla 4-5 se puede ver que los intervalos de tiempo aumentan en mayor cantidad a medida que la carga también aumenta.

4.2.3. ÍNDICE DISPONIBILIDAD DEL WEB SERVICE EN EL SERVIDOR DE APLICACIONES IIS

La tabla 4-6 muestra en la primera columna el número de agentes, en las tres siguientes la disponibilidad del web service (D) y en la última la media obtenida de las tres corridas.

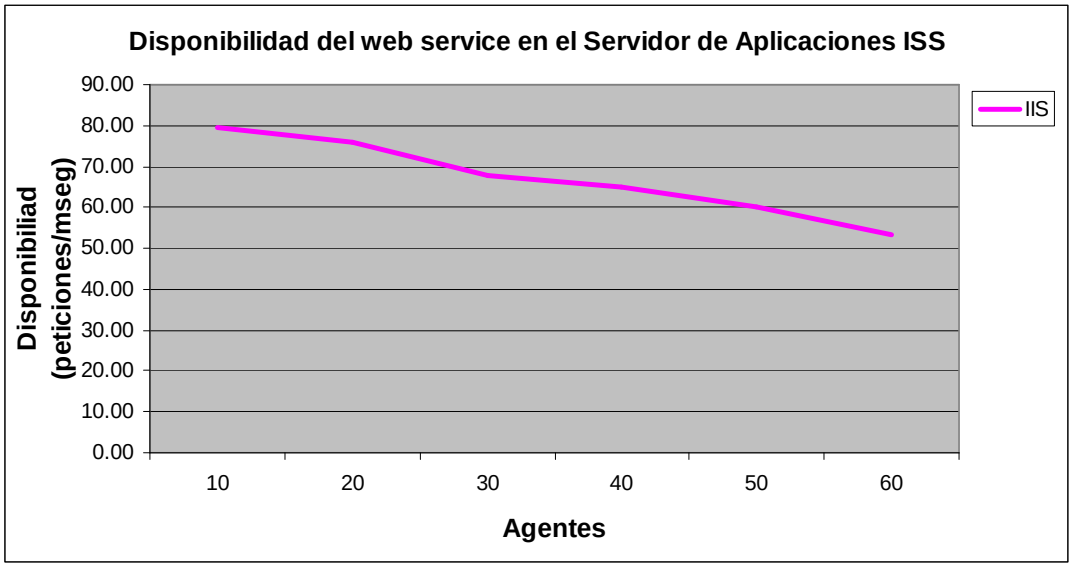
Tabla 4-6: Disponibilidad del web service en el Servidor de Aplicaciones IIS

Agentes	1	2	3	Media ΣD/3
10	82.45	77.75	78.45	79.55
20	74.86	75.91	76.74	75.84
30	66.63	67.01	70.18	67.94
40	64.81	64.77	65.28	64.95
50	61.29	59.54	60.12	60.32
60	50.10	53.35	55.85	53.10

Elaboración y Fuente: Los Autores

El gráfico de la figura 4-10 se generó utilizando los agentes y la media que se encuentran en la tabla 4-6.

Figura 4-10: Disponibilidad del web service en el Servidor de Aplicaciones IIS



Elaboración y Fuente: Los Autores

En el eje de las abscisas se muestra el número de agentes que intervienen en la prueba, mientras que en el eje de las ordenadas la disponibilidad del web service expresada en llamadas por milisegundo.

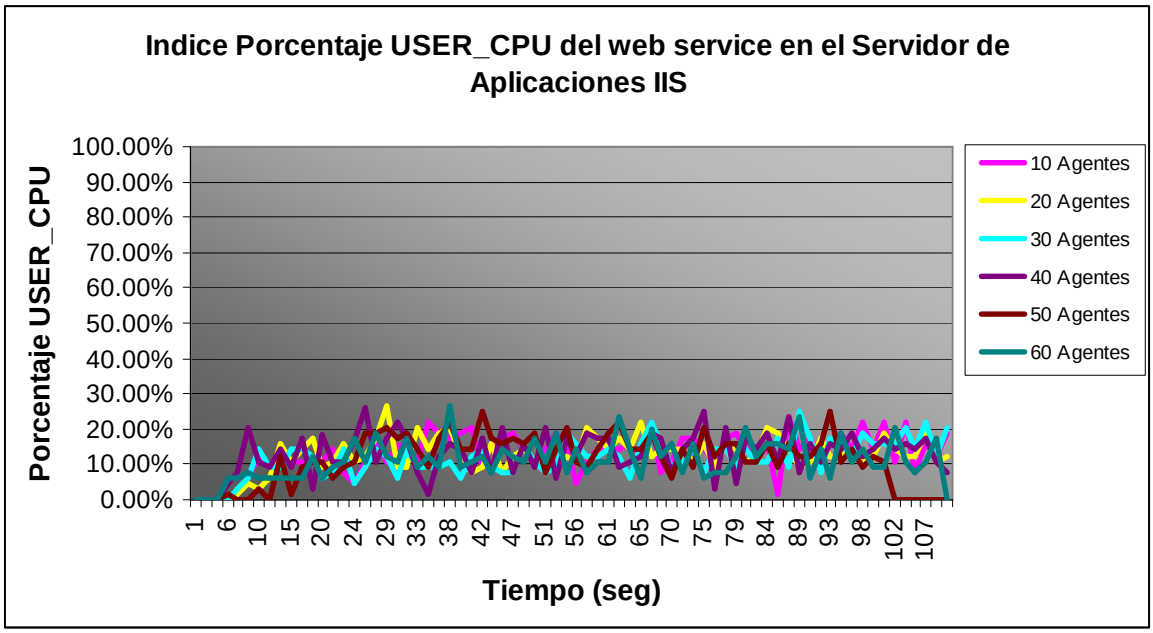
El gráfico nos indica que la disponibilidad del web service en el servidor de aplicaciones JBOSS depende inversamente de la carga que se suministre, es decir a medida que ésta aumenta, la Disponibilidad disminuirá, en caso contrario aumentará.

4.2.4. ÍNDICES DE DISPONIBILIDAD DE CPU DEL SERVIDOR DE APLICACIONES JBOSS

4.2.4.1. Índice USER_CPU del web service en el Servidor de Aplicaciones IIS

El gráfico de la figura 4-11 se generó utilizando los valores de la media del índice USER de la tabla que se encuentra en el Anexo 14.

Figura 4-11: Porcentaje USER_CPU del web service en el Servidor de Aplicaciones IIS.



Elaboración y Fuente: Los Autores

En el eje de las abscisas se muestra el tiempo expresado en segundos que intervienen en la prueba, mientras que en el eje de las ordenadas el porcentaje de tiempo que usa el web service del procesador.

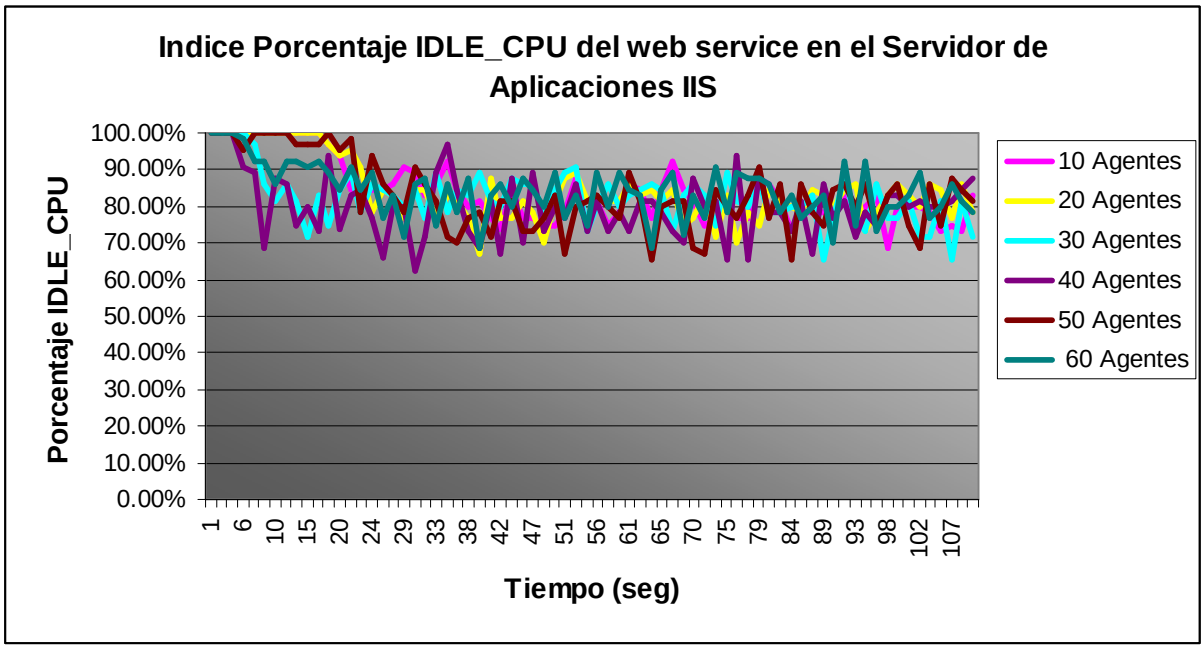
El gráfico nos indica el porcentaje de tiempo que usa el procesador durante la ejecución del web service en el servidor de aplicaciones JBOSS, es independiente de la carga que se suministre.

Además se puede ver que el web service en el servidor de aplicaciones IIS no ocupa el procesador en su totalidad mientras esta atendiendo las peticiones realizadas.

4.2.4.2. Índice IDLE_CPU del web service en el Servidor de Aplicaciones IIS

El gráfico de la figura 4-12 se generó utilizando los valores de la media del índice IDLE de la tabla que se encuentra en el Anexo 15.

Figura 4-12: Porcentaje IDLE_CPU del web service en el Servidor de Aplicaciones IIS.



Elaboración y Fuente: Los Autores

En el eje de las abscisas se muestra el tiempo expresado en segundos que intervienen en la prueba, mientras que en el eje de las ordenadas el porcentaje de idle del tiempo.

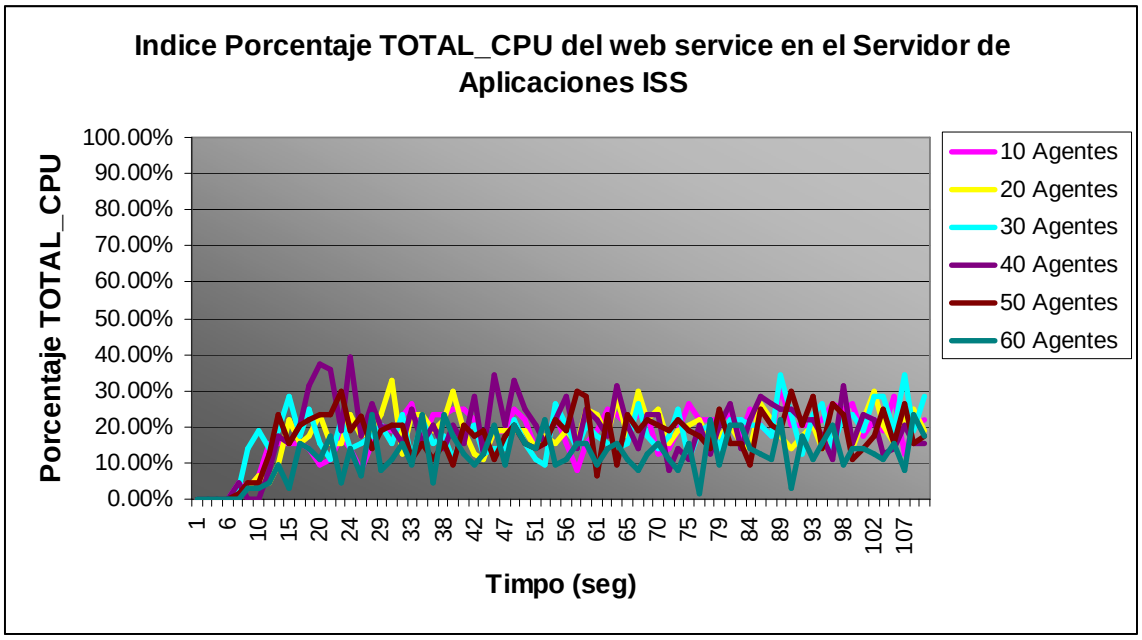
El gráfico nos indica el porcentaje de tiempo que el procesador se encuentra en espera durante la ejecución del web service en el servidor de aplicaciones JBOSS, este porcentaje es independiente de la carga que se suministre.

Además se puede ver que el web service en el servidor de aplicaciones IIS no ocupa el procesador en su totalidad, por lo que el porcentaje de tiempo en el que el procesador está en espera es alto.

4.2.4.3. Índice TOTAL_CPU para el web service en el Servidor de Aplicaciones IIS

El gráfico de la figura 4-13 se generó utilizando los valores de la media que se encuentran en la tabla del Anexo 16.

Figura 4-13: Porcentaje TOTAL_CPU del web service en el Servidor de Aplicaciones IIS



Elaboración y Fuente: Los Autores

En el eje de las abscisas se muestra el tiempo expresado en segundos que intervienen en la prueba, mientras que en el eje de las ordenadas el porcentaje de tiempo del procesador.

El gráfico nos indica que el porcentaje de tiempo en el que el procesador permanece ocupado durante la ejecución del web service en el servidor de aplicaciones IIS es independiente de la carga que se suministre.

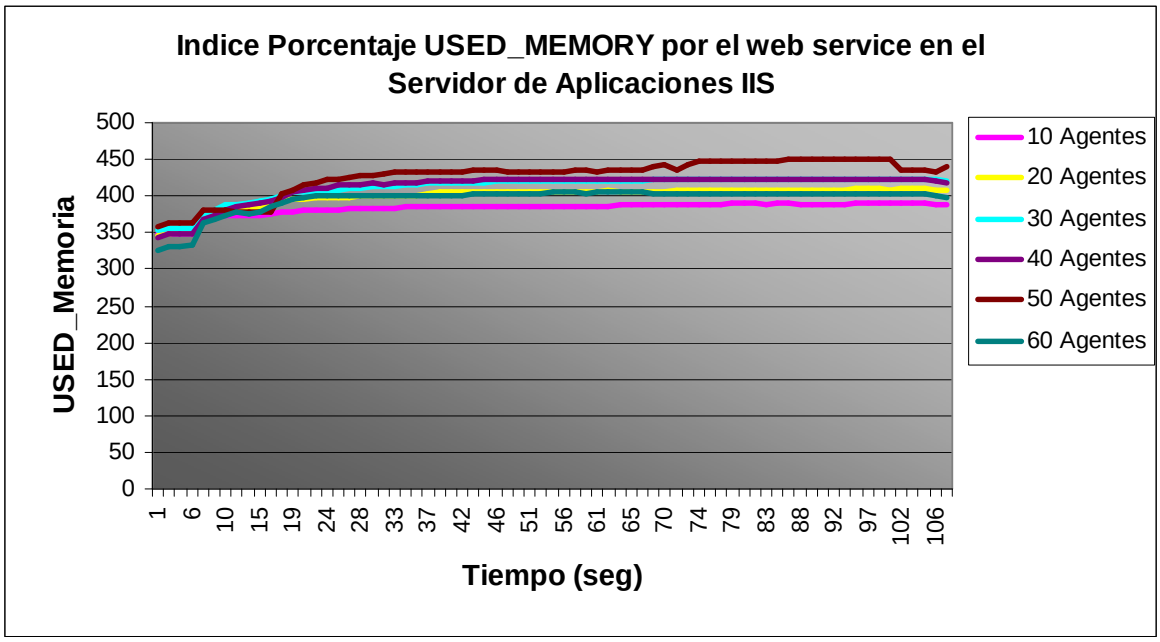
Además se puede ver que el web service en el servidor de aplicaciones IIS no ocupa el procesador en su totalidad mientras esta atendiendo las peticiones realizadas.

4.2.5. INDICES DE DISPONIBILIDAD DE MEMORIA EN EL SERVIDOR DE APLICACIONES IIS

4.2.5.1. Índice USED_MEM por el web service en el Servidor de Aplicaciones IIS

El gráfico de la figura 4-14 se generó utilizando los valores del índice USED de la tabla que se encuentra en el Anexo 17.

Figura 4-14: Porcentaje USED_MEMORY por el web service en el Servidor de Aplicaciones IIS



Elaboración y Fuente: Los Autores

En el eje de las abscisas se muestra el tiempo expresado en segundos que intervienen en la prueba, mientras que en el eje de las ordenadas la cantidad de memoria física usada.

El gráfico nos indica que la cantidad de memoria física que está siendo utilizada durante la ejecución del web service en el servidor de aplicaciones IIS depende directamente de la carga que se suministre, es decir, a medida que ésta aumenta, la cantidad de memoria física ocupada es mayor, en caso contrario disminuirá.
Esto solo sucede hasta cierto punto, a partir de éste la forma en la que el servidor de aplicaciones usa la memoria física es irregular.

En el Anexo 18 se adjuntan las tablas de las demás pruebas para cada índice.

En el Anexo 19 se adjuntan los gráficos de las demás pruebas para cada índice.

4.3. COMPARACIÓN DE LOS RESULTADOS OBTENIDOS

Para realizar este análisis se tiene una tabla de datos por cada índice evaluado en donde se exponen los valores medidos de los dos servidores de aplicaciones en los que se encuentran los web services. Estos datos han sido utilizados para generar los gráficos comparativos.

4.3.1. ÍNDICE NÚMERO DE PETICIONES ATENDIDAS DEL WEB SERVICE

La tabla 4-7 muestra en la primera columna el número de agentes, en las siguientes la media del número de peticiones atendidas por cada web service.

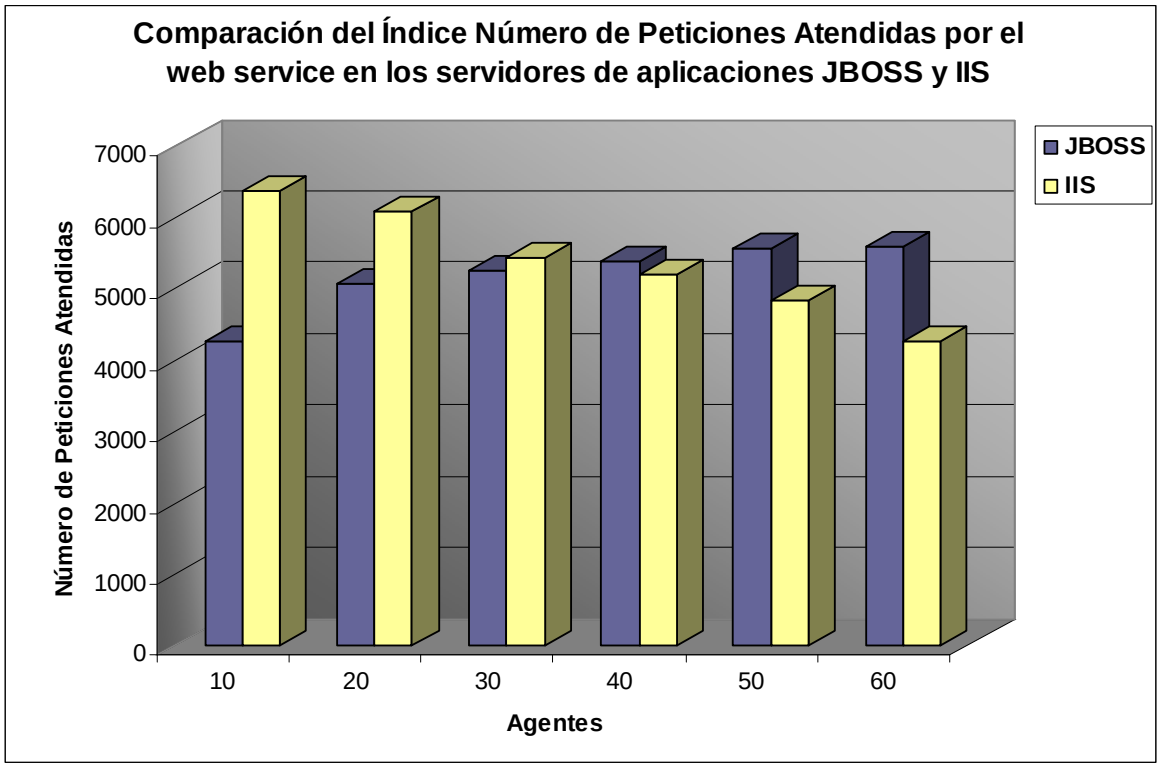
Tabla 4-7: Número de Peticiones Atendidas JBOSS - ISS

Agentes	JBOSS	IIS
10	4261	6364
20	5068	6067
30	5243	5435
40	5363	5196
50	5549	4833
60	5588	4248

Elaboración y Fuente: Los Autores

El gráfico de la figura 4-15 se generó utilizando los agentes y las medias que se encuentran en la tabla 4-7.

Figura 4-15: Comparación Número de Peticiones Atendidas por los web services en los servidores de aplicaciones JBOSS y IIS.



Elaboración y Fuente: Los Autores

En el eje de las abscisas se muestra el número de agentes que intervienen en la prueba, mientras que en el eje de las ordenadas se encuentran el Número de Peticiones Atendidas por los web services.

El grafico nos indica que el Número de Peticiones Atendidas por el web service en el servidor de aplicaciones IIS, inicia con un número alto de peticiones atendidas comparado con el web service en el servidor de aplicaciones JBOSS, a medida que la carga aumenta, este número va disminuyendo, lo que no sucede con su similar de la plataforma Java.

4.3.2. ÍNDICE TIEMPO DE RESPUESTA DEL WEB SERVICE

La tabla 4-8 muestra en la primera columna el número de agentes, en las siguientes la media del tiempo de respuesta de los web services.

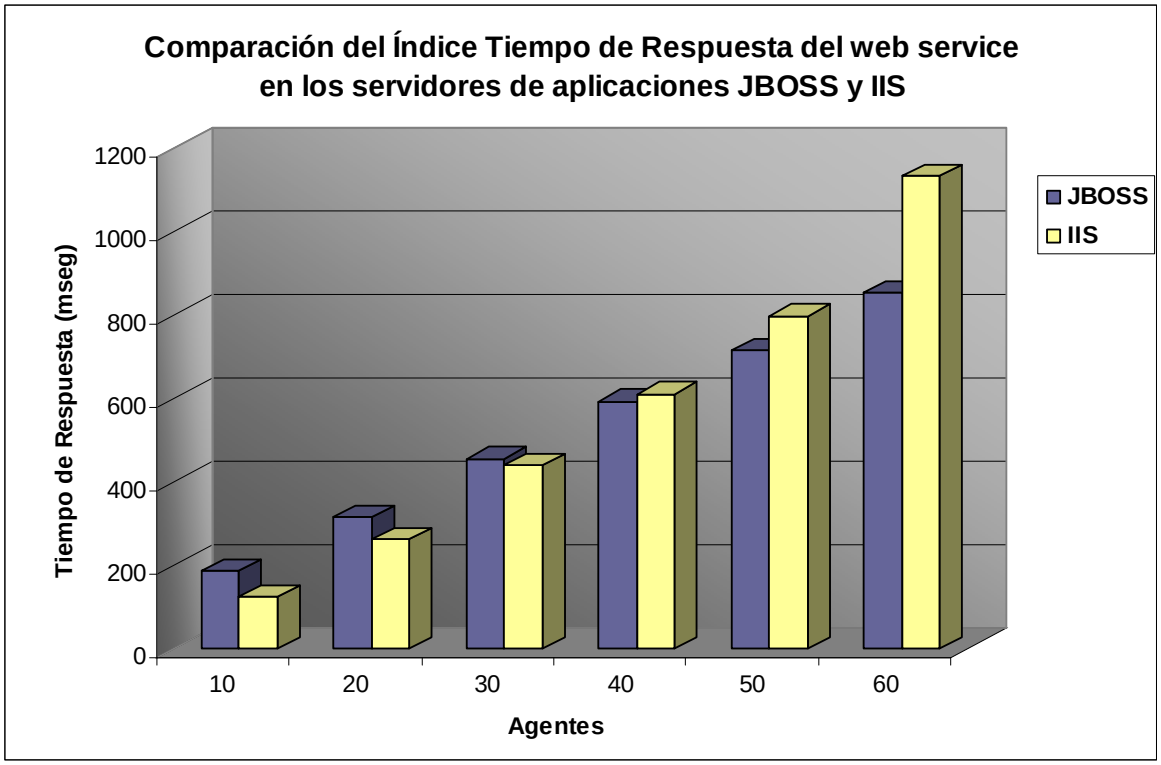
Tabla 4-8: Tiempo de Respuesta JBOSS – ISS

Agentes	JBOSS	IIS
10	187	125
20	315	262
30	455	440
40	594	611
50	716	798
60	853	1134

Elaboración y Fuente: Los Autores

El gráfico de la figura 4-16 se generó utilizando los agentes y las medias que se encuentran en la tabla 4-8.

Figura 4-16: Comparación Tiempo de Respuesta de los web services en los servidores de aplicaciones JBOSS y IIS.



Elaboración y Fuente: Los Autores

En el eje de las abscisas se muestra el número de agentes que intervienen en la prueba, mientras que en el eje de las ordenadas se encuentran el Tiempo de respuesta de los web services.

El grafico nos indica que el Tiempo de Respuesta de los web services en el servidor de aplicaciones JBOSS, inicia con un tiempo de respuesta superior en comparación al que emplea el otro web service, a medida que la carga aumenta, en los puntos más altos de la carga, el tiempo de respuesta en el servidor de aplicaciones IIS llega a ser superior al tiempo utilizado por el servidor de aplicaciones JBOSS

4.3.3. ÍNDICE DISPONIBILIDAD DEL WEB SERVICE

La tabla 4-9 muestra en la primera columna el número de agentes, en las siguientes la media del tiempo de respuesta de los web services.

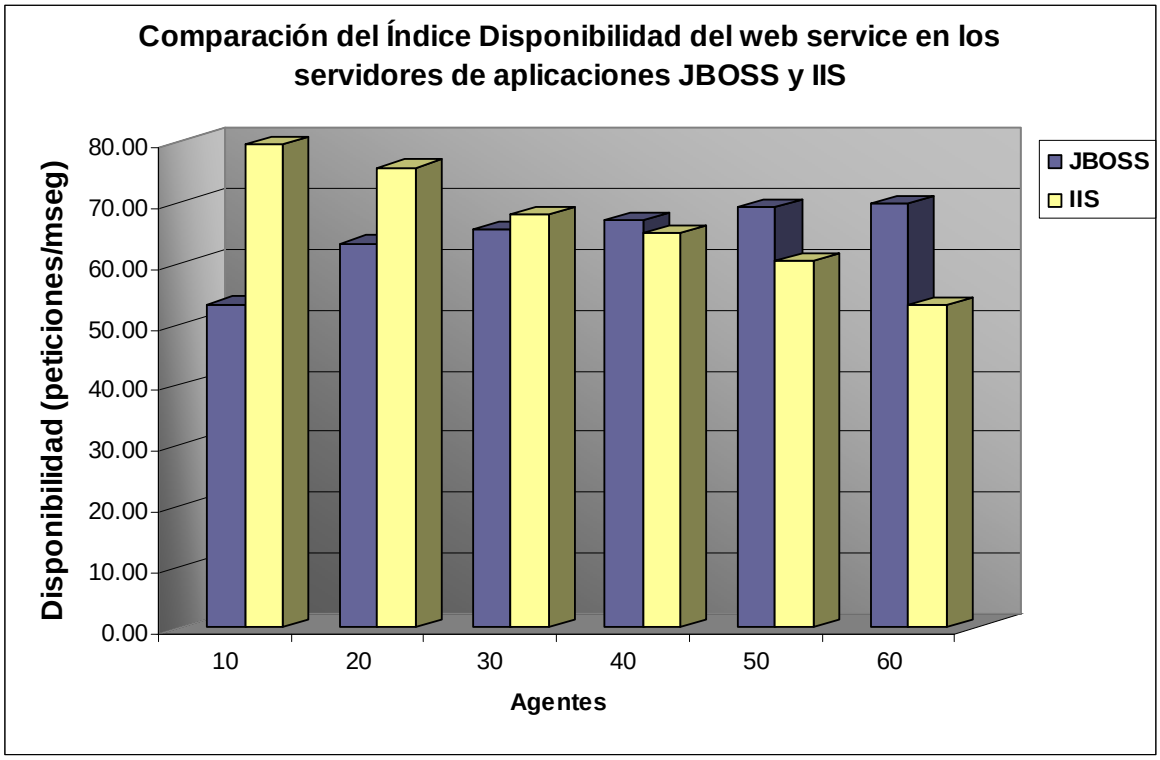
Tabla 4-9: Tiempo de Respuesta JBOSS – ISS

Agentes	JBOSS	IIS
10	187	125
20	315	262
30	455	440
40	594	611
50	716	798
60	853	1134

Elaboración y Fuente: Los Autores

El gráfico de la figura 4-17 se generó utilizando los agentes y las medias que se encuentran en la tabla 4-9.

Figura 4-17: Comparación de Disponibilidad de los web services en los servidores de aplicaciones JBOSS y IIS.



Elaboración y Fuente: Los Autores

En el eje de las abscisas se muestra el número de agentes que intervienen en la prueba, mientras que en el eje de las ordenadas se encuentran el Tiempo de respuesta de los web services.

El grafico nos indica que la disponibilidad de los web services en el servidor de aplicaciones IIS con número de carga pequeña es superior a la disponibilidad en el servidor de aplicaciones JBOSS, a medida que la carga aumenta, la disponibilidad de los web services en el servidor de aplicaciones JBOSS aumenta, lo cual no sucede con el servidor de aplicaciones de la plataforma .NET.

En el Anexo 20 se adjuntan las tablas de las demás pruebas para cada índice.

En el Anexo 21 se adjuntan los gráficos de las demás pruebas para cada índice.

4.3.4. ÍNDICE DISPONIBILIDAD DE CPU UTILIZADO POR EL WEB SERVICE

El porcentaje de tiempo del procesador que usa el web service alojado en el servidor de aplicaciones JBOSS es superior al porcentaje de tiempo del procesador usado por el web service alojado en el servidor de aplicaciones de la plataforma .NET.

El porcentaje de tiempo en el que el procesador se encuentra en espera durante la ejecución web service alojado en el servidor de aplicaciones JBOSS es inferior al porcentaje de tiempo del procesador usado por el web service alojado en el servidor de aplicaciones de la plataforma .NET.

El porcentaje de tiempo total que el procesador se encuentra ocupado mientras ejecuta un subproceso diferente al subproceso del proceso Idle durante la ejecución web service alojado en el servidor de aplicaciones JBOSS es superior al porcentaje de tiempo total que el procesador se encuentra ocupado mientras se ejecuta el web service alojado en el servidor de aplicaciones de la plataforma .NET.

4.3.5. ÍNDICE DISPONIBILIDAD DE MEMORIA UTILIZADO POR EL WEB SERVICE

El tamaño de la memoria física que utiliza el web service alojado en el servidor de aplicaciones IIS es superior al tamaño de memoria física que usa el web service alojado en el servidor de aplicaciones de la plataforma Java, el tamaño de la memoria física usada por web service en IIS disminuye mientras la carga aumenta, puesto que el número de peticiones disminuye, el tiempo de respuesta aumenta y la disponibilidad también disminuye, razón por la que el uso de memoria física tiende a disminuir.

CAPITULO 5. CONCLUSIONES Y RECOMENDACIONES

En este trabajo se ha abordado el tema construcción de un Benchmark para realizar un estudio comparativo entre web services desarrollados en plataformas Java y .NET, con la finalidad de evaluar el rendimiento de los mismos. Se obtuvieron las siguientes conclusiones y recomendaciones.

Las conclusiones que se presentan, recopilan las principales ideas del contenido de cada uno de los capítulos desarrollados en este trabajo, se exponen ideas que abarcan la experiencia que se consiguió durante la elaboración de los mismos.

Las recomendaciones pretenden orientar a la elaboración de futuros proyectos para la titulación que se relacionen con la evaluación de web services..

5.1. CONCLUSIONES

- La construcción del Benchmark para realizar un estudio comparativo entre web services desarrollados en plataformas Java y .NET, ha permitido cumplir con el objetivo principal así como también con los objetivos específicos que fueron planteados para la realización del presente trabajo.

- La documentación que se expone en este trabajo es de utilidad para tener una mayor conocimiento a cerca de los fundamentos, arquitectura, roles y plataformas de la arquitectura orientada a servicios, al mismo tiempo se presenta una introducción a los web services en la que se describe una breve historia de los mismos, cuando usarlos y las tecnologías que utilizan.

- Se presenta el fundamento teórico de la forma en la que se encuentran implementados los web services tanto en la plataforma Java así como en la plataforma .NET, conceptos básicos de benchmark y estadígrafos, de manera que permitan definir mejores alternativas para desarrollar un benchmark para evaluar el rendimiento de web services, y así representar correctamente la lógica funcional de éste.

- El uso de una metodología formal de ingeniería de software dentro de la especificación de servicios permiten definir de manera precisa y completa la funcionalidad del Benchmark, mediante la documentación de requerimientos, análisis y diseño.

- El Proceso Rational Unificado o RUP (Rational Unified Process), es un proceso muy útil para el desarrollo de software, junto con el Lenguaje Unificado de Modelado (UML) se constituye en la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

- Al emplear una metodología orientada a objetos, es de gran importancia que en la selección de la herramienta se considere que esta soporte los paradigmas de la orientación a objetos, los mismos que son: encapsulamiento, herencia y polimorfismo, de esta manera se garantizaría tener un modelo que sea una representación plena de lo modelado en el análisis y diseño, además facilita la implementación.

- El lenguaje de programación Java constituye una poderosa plataforma, si se trata de construir sistemas totalmente orientados a objetos, dado que se acopla a herramientas case como Rational Rose Enterprise Edition , facilitando el paso de la fase de diseño a la fase de implementación.

- Los web services han llegado a constituir la tecnología más importante para la comunicación. El funcionamiento de las llamadas de servicios basadas en SOAP es crucial y debe ser tomada en consideración cuando se evalúan.

- La programación de aplicaciones multiproceso permite ejecutar simultáneamente varios procedimientos. Las aplicaciones multiproceso, al dividir los programas en tareas independientes, pueden mejorar considerablemente el rendimiento.

- Adoptar la plataforma Java para construir aplicaciones que funcionen dentro de una arquitectura orientada a servicios es una buena alternativa por los beneficios que ofrece al ser de libre de distribución frente a otras plataformas comerciales que tienen un costo de por medio

- La plataforma .NET a través de su herramienta de desarrollo Visual Studio 2005 ofrece un entorno de trabajo amigable, así como también un conjunto de controles gráficos que facilitan la construcción de aplicaciones a los desarrolladores.

- El benchmark que evalúa el rendimiento de web services implementados en plataformas Java y .NET tiene una importante característica la misma que es contar con un archivo de configuración desde el que se envía la dirección url del servidor de aplicaciones en el que se encuentra alojado el web service, sea este en plataforma Java o .NET, este archivo de configuración permite al mismo tiempo seleccionar los métodos que expone el web service que se desea probar.

- Las pruebas indican que el web service alojado sobre el servidor de aplicaciones JBOSS de la plataforma Java, utiliza al máximo el procesador, esto podría generar cuellos de botella si el servidor es usado en otras actividades que involucren mayor uso de procesador.

- Finalmente este trabajo representa para nosotros un fuerte crecimiento profesional y personal al abordar uno de los temas que esta evolucionando a gran velocidad y ha tenido gran acogida en muchas empresas con la finalidad de conseguir la completa integración de los procesos de negocio de una organización con otras.

5.2. RECOMENDACIONES

- El presente trabajo puede ser tomado como modelo de referencia para la implementación de una evaluación de seguridades entre web services desarrollados en plataformas Java y .NET.

- Previo a la definición de las características y requerimientos que debe tener un benchmark para evaluar el rendimiento de web services, se recomienda realizar una investigación de soluciones existentes que ofrezcan servicios similares, con la finalidad de poder definir de forma clara el alcance los servicios que prestará el Benchmark

- Se recomienda utilizar la plataforma Java por ser una herramienta que facilita el desarrollo de aplicaciones orientadas a objetos, portable , y sobre todo por ser una herramienta de libre distribución que no involucra tener que comprar una licencia para su funcionamiento como sucede con otras herramientas de desarrollo orientadas a objetos

- Para la construcción de modelos y diagramas que requieren ser elaborados por utilizar el proceso unificado de desarrollo de software es recomendable el uso de herramientas CASE. En caso específico Rational Rose Enterprise Edition.

- Cuando se desarrolla sistemas es necesario que el grupo de desarrollo tenga experiencia con la herramienta que emplee, para cumplir con los plazos que se establezcan en la planificación.

- Es importante establecer estándares en cuanto a programación se refiere, puesto que mejora la legibilidad del código, facilitando a otros desarrolladores comprenderlo rápidamente.

- Durante las pruebas realizadas en este proyecto se ha utilizado sólo un cliente físico que simuló peticiones concurrentes hacia el web service, por lo tanto se considera que este proyecto se puede hacer extensible probando con más de un cliente físico, esto se lo podría conseguir instalando en cada cliente físico un programa de terceros que simule carga.

BIBLIOGRAFÍA

1. GONZALEZ, Haaron. “Fundamentos de la Arquitectura Orientada a Servicios”, http://msmvps.com/blogs/haarongonzalez/archive/2006/04/19/91679.aspx.

2. PELECHANO, Vicente. “Servicios Web. Estándares, Extensiones y Perspectivas de Futuro”, http://pangea.upv.es/N+ISIS05/documents/VicentePelechano/ServiciosWeb.pdf.

3. .ZAMBRANO, Paúl. “Introducción a los web services en las organizaciones”. Octubre 2004.

4. RUMBAUGH, James; JACOBSON, Ivar; BOOCH, Graddy. “El proceso unificado de desarrollo”. Editorial Addison-Wesley. 2000.

5. PERFORMANCE MONITOR Windows Server 2003.

6. VILLA, Cristina, VIZCAINO, Paúl. “Construcción de un Benchmark que permita comparar el rendimiento de los drivers JDBC en Oracle”. Agosto 2003.

7. CEVALLOS, Carlos, SUÁREZ, Christian. “Diseño y Construcción de un Protocolo de Transferencia de Archivos Tolerante a Fallas”. Enero 2003.

8. GALINDO, Edwin. “Estadística para la Administración y la Ingeniería”. Editorial Gráficas Mediavilla Hnos. 1999.

9. SINGH, Inderjeet, BRYDON, Sean, MURRAY, Grey, RAMACHANDRAN, Vijay, VIOLLEAU, Thierry, STEARNS, Beth. “Designing Web Services with the J2EETM 1.4 Platform JAX-RPC, SOAP and XML Technologies. Enero 2004.

10. FERRARA Alex, MACDONALD, Matthew. “Programming .NET Web Services”. 2004.

11. NEWCOMER, Eric, LOMOW, Greg. “Understanding SOA with Web Services”. David Chappell Series Editor. 2005.

12. WIKIPEDIA. “Wikipedia La enciclopedia libre”. http://es.wikipedia.org/wiki/Servidor_de_aplicaciones.

13. FROUFE, Agustín. “Tutorial de Java”. 1996, http://www.cica.es/formacion/JavaTut/Cap7/estado.html.

14. APACHE SOFTWARE FOUNDATION. “Axis Architecture Guide”, http://ws.apache.org/axis/java/architecture-guide.html.

15. SUN MICROSYSTEMS. “Code Conventions for the Java Language Programming”, http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html.

GLOSARIO DE TÉRMINOS

El glosario de términos se encuentra en la sección 3.1.2.2