REPÚBLICA DEL ECUADOR

Escuela Politécnica Nacional

"E SCIENTIA HOMINIS SALUS"

ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

SISTEMA DISTRIBUIDO PARA GESTIÓN DEL LABORATORIO DE INFORMÁTICA DE LA FIEE

PROYECTO PREVIO A LA OBTENCION DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y REDES DE INFORMACIÓN

MOSQUERA ASIMBAYA JOSE ALEJANDRO

VIÑAMAGUA QUEZADA DIEGO JAVIER

DIRECTOR: ING. RAÚL DAVID MEJIA NAVARRETE, MSc.

Quito, Septiembre 2015

DECLARACIÓN

Nosotros, José Alejandro Mosquera Asimbaya y Diego Javier Viñamagua Quezada, declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentado 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 normativa institucional vigente.

José Alejandro Mosquera Asimbaya

Diego Javier Viñamagua Quezada

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por José Alejandro Mosquera Asimbaya y Diego Javier Viñamagua Quezada, bajo mi supervisión.

Ing. Raúl David Mejía Navarrete, MSc.
DIRECTOR DEL PROYECTO

AGRADECIMIENTOS

Agradezco a mis padres: José Mosquera e Hilda Asimbaya por el apoyo que me dieron durante este tiempo.

A mis hermanas: Fanny, Sonia y Sofía, por darme consejo cuando lo necesité.

A mi novia: Jenny Casame por acompañarme, bridarme su apoyo y darme su amor durante todos estos años, gracias mi amor.

A mis amigos y compañeros, en especial al coautor de este proyecto: Diego Viñamagua; gracias a su apoyo se concluyó con éxito este trabajo.

Al director de este proyecto, David Mejía, por su tiempo, paciencia, ayuda y aportes a este proyecto.

A mis compañeras del laboratorio en donde trabajo: Jeaneth y Gaby.

José Mosquera

AGRADECIMIENTOS

Agradezco a mi padre Belarmino Viñamagua y a mi madre Olga Quezada, por su apoyo en todo momento para lograr mis metas; por su confianza y por enseñarme que la base es el trabajo y la constancia.

A mis hermanas Miriam, Glenda y Rosario, porque siempre han estado pendientes de mí.

Al director de este proyecto David Mejía por su ayuda al compartir sus conocimientos, tanto en las materias que dicta como en el desarrollo de este proyecto.

Al coautor de este proyecto, José Mosquera por su ayuda y apoyo durante el desarrollo del mismo.

A todos los profesores que han sido parte de mi formación académica de los cuales aprendí mucho.

A mis amigos por acompañarme durante toda esta travesía.

Diego Viñamagua

DEDICATORIA

Este proyecto está dedicado a mi novia Jenny quien ha sido mi apoyo incondicional en todo momento.

José Mosquera

DEDICATORIA

Este proyecto va dedicado a mis padres que me han apoyado en todo momento y a lo largo de mi carrera.

Diego Viñanagua

CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE CÓDIGO

RESUMEN

El presente Proyecto de Titulación consta del desarrollo y la implementación de un sistema de gestión para el Laboratorio de Informática de la Facultad de Ingeniería Eléctrica y Electrónica.

El sistema permite: gestionar los PC del laboratorio a través de sus diferentes aplicaciones, automatizar el préstamo de equipos mediante solicitudes, identificar diferentes perfiles de usuario y establecer una comunicación por voz entre usuarios con dispositivos Android.

El sistema de gestión está conformado por una aplicación principal, una base de datos, un servicio SIP, una aplicación profesor, una aplicación cliente y una aplicación Android.

La aplicación principal es la encargada de conectarse con la base datos y gestionar toda la información del sistema. La base de datos almacenará la información de: usuarios, ítems del catálogo de equipos, equipos a gestionar y solicitudes.

La aplicación profesor es la encargada de conectarse a los PC de su mismo grupo de trabajo para gestionarlos, además permite registrar usuarios y generar solicitudes para el préstamo de equipos.

La aplicación cliente es la encargada de recibir mensajes desde las otras aplicaciones para ejecutar acciones sobre los PC que gestiona.

La aplicación Android posee una interfaz desde la cual el usuario puede seleccionar distintas acciones de gestión, para esto se comunica con la aplicación principal que a su vez realiza la acción seleccionada en el PC a gestionar. Además muestra las solicitudes generadas desde la aplicación profesor y el usuario puede establecer una comunicación por voz con otros usuarios que dispongan de la aplicación.

Las aplicaciones de escritorio fueron implementadas usando el IDE Visual Studio 2010 y el lenguaje de programación C#. La aplicación Android fue implementada usando el IDE Android Studio y el lenguaje de programación Java. La base de datos fue implementada usando MySQL y para el servicio SIP se usó Asterisk.

En el primer capítulo se presenta conceptos sobre: el desarrollo de software, los sistemas distribuidos, el sistema operativo Android y la metodología e*Xtreme Programming*.

El segundo capítulo presenta los requerimientos del prototipo de sistema de gestión a partir del análisis de aplicaciones como iTalc y NetSupport Manager, posteriormente se detallan los módulos del sistema, se exponen las historias de usuario y finalmente se presentan los diagramas de clases y secuencia de las diferentes aplicaciones.

En el tercer capítulo se describe la implementación de las aplicaciones del sistema de gestión en base al diseño realizado.

El cuarto capítulo muestra los resultados de las pruebas realizadas al sistema de gestión; para esto se realizan pruebas unitarias, pruebas de funcionamiento en un ambiente controlado, en un ambiente real y pruebas de aceptación.

En el quinto capítulo se presentan las conclusiones y recomendaciones obtenidas a lo largo del proyecto.

Finalmente en los anexos se incluyen: manual de usuario del sistema de gestión, todo el código fuente de las aplicaciones e instaladores. Los anexos se encuentran en un CD adjunto a este documento.

PRESENTACIÓN

El presente Proyecto de Titulación se lo realizó con el objetivo de desarrollar un sistema que facilite la gestión de los equipos con los que cuenta el Laboratorio de Informática de la Facultad de Ingeniería Eléctrica y Electrónica. Este sistema permite que los administradores puedan efectuar acciones como: apagar, encender, bloquear, desbloquear y reiniciar de forma remota los equipos del laboratorio; y que los profesores podrán controlar las actividades que realizan sus estudiantes por medio del acceso remoto.

En el presente documento se detalla la información necesaria para entender el uso, funcionamiento e implementación del sistema y todos sus componentes; para esto se describe en forma breve las metodologías de software, el sistema Operativo Android, el servicio SIP, la base de datos, y el protocolo RDP.

El sistema se desarrolló usando la metodología de software e*Xtreme Programming*, la cual permite realizar un proceso de desarrollo documentado, esto permite que el equipo de trabajo pueda realizar el proyecto de una manera más organizada. Cabe mencionar que el sistema emplea un modelo de arquitectura cliente servidor.

Dentro de este proyecto se desarrollan tres aplicaciones de escritorio: una aplicación principal para manejar información de todo el sistema, una aplicación profesor para manejar información de su mismo grupo de trabajo, una aplicación cliente que permitirá a los PC ser gestionados y una aplicación Android que permitirá ejecutar acciones desde el dispositivo móvil. Además, el sistema también está compuesto por una base de datos y un servicio SIP.

Finalmente, se realizan pruebas de funcionamiento y aceptación a todo el sistema con el fin de comprobar que se cumplan los requerimientos planteados al inicio del proyecto.

CAPÍTULO 1 MARCO TEÓRICO

1.1 INTRODUCCIÓN

En este capítulo se presentarán los conceptos básicos, la descripción y el funcionamiento de los componentes necesarios para realizar el proyecto.

1.2 SISTEMA DISTRIBUIDO

1.2.1 DEFINICIÓN

Un sistema distribuido está conformado por un grupo de PC conectados a través de una red, los cuales pueden coordinar actividades y compartir recursos del sistema [1].

1.2.2 CARACTERÍSTICAS DE UN SISTEMA DISTRIBUIDO

A continuación se presentan las características de los sistemas distribuidos [1], [2].

1.2.2.1 Concurrencia

Es la capacidad de ejecutar varias tareas o procesos de manera simultánea, las cuales pueden ser ejecutadas en un solo equipo central o en un sistema distribuido.

1.2.2.2 Transparencia

La distribución de los recursos del sistema es independiente de la topología de la red; de esta manera si los recursos cambian de ubicación lógica el sistema debe seguir funcionando de la misma manera.

1.2.2.3 Escalabilidad

Un sistema distribuido puede crecer lo que significa que más recursos pueden ser añadidos sin disminuir el rendimiento del mismo.

1.2.2.4 Disponibilidad

Es el periodo de tiempo en que el sistema está en funcionamiento, este puede aumentar si se usa equipos de mejores prestaciones o equipos repetidores para que el sistema pueda seguir funcionado si alguno de ellos falla.

Los fallos en el sistema tanto en hardware como en software no pueden ser evitados al 100%.

1.2.3 SOCKETS

Socket es una abstracción que permite la comunicación entre distintos procesos que típicamente se ejecutan en diferentes equipos en la red, haciendo uso de los siguientes parámetros [3]:

- Protocolo de red y de transporte
- Direcciones de red
- Número de puerto

1.2.3.1 Tipos de socket

Se tienen dos tipos de socket, dependiendo del protocolo de transporte que se utilice.

Los sockets que utilizan el protocolo TCP[1] tienen las siguientes características:

- Son orientados a conexión
- Garantizan una correcta transmisión
- Los segmentos son entregados en orden

[1] TCP: *Transmission Control Protocol* es un protocolo de capa transporte orientado a conexión confiable.

- Se tienen acuses de recibo (ACK[2])

Los sockets que utilizan el protocolo UDP[3] tienen las siguientes características:

- No son orientados a conexión
- Los datagramas pueden viajar en cualquier orden
- No se garantiza la recepción de todos los paquetes

1.2.3.2 Funcionamiento de un socket

La **Figura 1.1** muestra la secuencia de llamadas para una comunicación orientada a conexión y la **Figura 1.2** muestra la secuencia de llamadas para una comunicación no orientada a conexión.

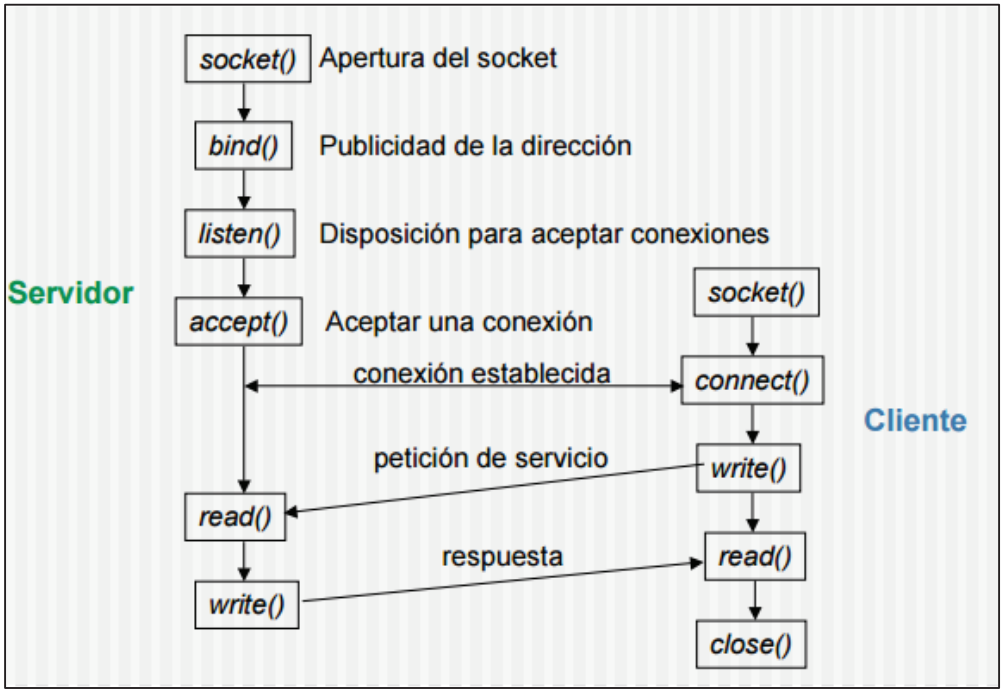


Figura 1.1. Secuencia de llamadas para una comunicación orientada a conexión [4]

[2] ACK: *acknowledgement* es un mensaje enviado desde el destino hacia el origen de la comunicación usado para confirmación.

[3] UDP: *User Datagram Protocol es un* protocolo de capa transporte no orientado a conexión *y* no confiable.

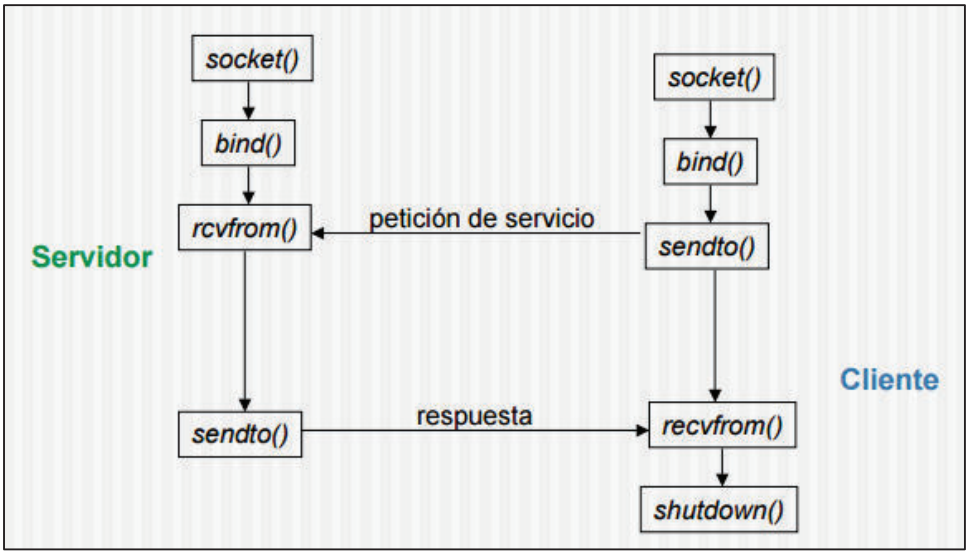


Figura 1.2. Secuencia de llamadas para una comunicación no orientada a conexión [4]

1.3 DESARROLLO DE SOFTWARE

El desarrollo de software consiste en construir aplicaciones mediante los requerimientos del usuario final. Estas servirán para un propósito específico y es el usuario final quien decidirá si el software construido cumplió con su propósito o no. Para esto es importante seguir un conjunto de pasos que van desde explorar el problema hasta determinar una solución.

Muchas personas pueden estar involucradas en el desarrollo de software; el cliente es quien propone el problema y los desarrolladores son quienes encuentran la solución a ese problema. [5]

1.3.1 PROCESO DE DESARROLLO DE SOFTWARE

Existen un conjunto de pasos a seguir para encontrar la solución a un problema. La **Figura 1.3** muestra los pasos del proceso de desarrollo de software. [6]

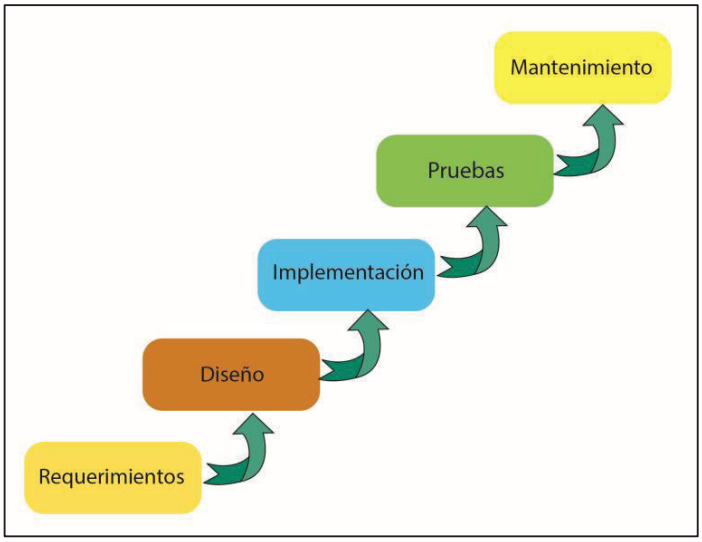


Figura 1.3. Procesos del Desarrollo de Software

- Requerimientos: En esta etapa el usuario propone lo que el software debe contener y la función que debe realizar, para esto se observan las necesidades y de manera general el entorno donde va a funcionar dicho software.
- Diseño: Con los requerimientos, los desarrolladores pueden empezar a proponer una solución; en este punto se incluirán todos los componentes que debe tener el software.
- Implementación: En este paso los desarrolladores eligen un lenguaje de programación para implementar la solución de acuerdo al diseño. La meta es cumplir con el objetivo para el cual se desarrolla el software.
- Pruebas: Una vez implementada la solución se procede a una etapa de pruebas donde se verifica el correcto funcionamiento del software desarrollado, en esta etapa, además, se pueden verificar posibles errores y corregirlos para lograr un mejor funcionamiento de este; el objetivo es lograr mayor calidad.
- Instalación: Finalizadas las pruebas, el software está listo para ser instalado y posteriormente usarlo.
- Mantenimiento: El software una vez instalado puede ser actualizado con versiones mejoradas del mismo en donde se pueden incluir correcciones de errores que no fueron encontrados durante la fase de pruebas.

Es recomendable también hacer pruebas continuas de forma paralela a la implementación.

1.3.2 METODOLOGÍA DE DESARROLLO DE SOFTWARE

La metodología es un conjunto de procedimientos, herramientas y técnicas para el desarrollo de software. La metodología que se elija determina: qué etapas tendrá el proyecto, qué tareas se realizan en cada etapa, qué restricciones se aplican y qué herramientas se utilizan [7].

Las metodologías de desarrollo de software pueden clasificarse en:

- Desarrollo convencional
- Desarrollo estructurado
- Desarrollo orientado a objetos
- Sistemas de tiempo real

También las metodologías de desarrollo de software pueden clasificarse en:

- Metodologías tradicionales
- Metodologías ágiles

1.3.2.1 Desarrollo convencional

Sus características son:

- No se usa ningún tipo de metodología
- No existe un análisis previo
- No hay forma de controlar lo que pasa durante la implementación
- Los resultados finales son impredecibles

1.3.2.2 Desarrollo estructurado

Sus características son:

- Se centra en las funciones que se usarán
- Se usa programación estructurada
- Se usan módulos en el desarrollo

1.3.2.3 Desarrollo orientado a objetos

Sus características son:

- Se usan objetos en vez de procedimientos
- Incluye fases como análisis, diseño e implementación
- Busca lograr una solución orientada a objetos en base a los requerimientos

1.3.2.4 Sistemas en tiempo real

Son sistemas en los cuales el tiempo de respuesta es fundamental para el correcto funcionamiento. Se dice que, si los datos llegaron fuera del tiempo establecido, estos serán inválidos y pueden ocasionar fallos en el sistema.

Ejemplos de este tipo de sistemas pueden ser: videoconferencias, sistemas de control en tiempo real, llamadas, etc.

1.3.2.5 Metodologías tradicionales

Estas se caracterizan por un tener una fuerte y estricta etapa de planificación, análisis y diseño antes de la implementación del sistema. Entre las metodologías tradicionales se tiene: Microsoft Solution Framework (MSF) y Microsoft Operation Framework. [8]

1.3.2.6 Metodologías ágiles

Se caracterizan por tener entregas pequeñas de software en cortos períodos de tiempo; además, el cliente y los desarrolladores trabajan en conjunto [8].

Entre las metodologías ágiles se tiene:

- *eXtreme Programming*
- Scrum
- *Dynamic Systems Development Method*
- *Adaptive Software Development*
- *Open Source Software Development*

1.3.3 EXTREME PROGRAMMING

"La simplicidad y la comunicación son extraordinariamente complementarias. Con más comunicación resulta más fácil identificar qué se debe y qué no se debe hacer. Cuanto más simple es el sistema, menos tendrá que comunicar sobre este, lo que lleva a una comunicación más completa, especialmente si se puede reducir el equipo de programadores". [9]

Es una metodología de desarrollo de software denominada como ágil y se diferencia de otras metodologías convencionales debido a su adaptabilidad. En este tipo de metodología es indispensable poder adaptar el software a cambios imprevistos en cualquier etapa de desarrollo.

La **Figura 1.4** muestra el proceso de desarrollo de software en *extreme programing* donde se puede ver que las diferentes etapas se solapan unas con otras, esto quiere decir que se puede trabajar de forma paralela entre ellas.

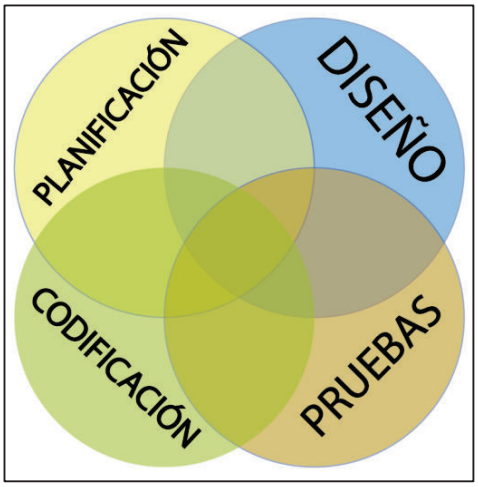


Figura 1.4. Proceso de desarrollo de software (e*Xtreme Programming*)

eXtreme Programming se basa en los siguientes valores:

- Simplicidad: Se simplifica el diseño para que el desarrollo sea más ágil; de esta manera se puede facilitar el mantenimiento. Es de mucha importancia refactorizar el código para mantenerlo simple a medida que crece. Los comentarios en el código deben ser moderados.

- Comunicación: El código debe ser entendible por todo el equipo de trabajo en donde también se incluye el usuario; los comentarios son de gran importancia debido a que en estos se explica la funcionalidad que prestan los diferentes métodos o variables dentro del programa.
- Realimentación: Al estar el usuario incluido dentro del proyecto, su opinión puede ser tomada en cuenta para realizar mejoras; para esto el software debe ser realizado en iteraciones las cuales pueden ser probadas, aquí se tiene un modelo de pruebas paralelo a la implementación.
- Coraje: El desarrollador debe saber cuándo desechar su código si este es ineficiente, no importa el esfuerzo o el tiempo que haya invertido en él.
- Respeto: Los miembros del equipo se respetan unos a otros; esto se debe dar debido a que todos los miembros tienen un mismo objetivo.

1.3.3.1 Características

A continuación se presentan las características de *eXtreme Programming* [10]*:*

- Desarrollo incremental: Se realizan pequeñas mejoras a medida que avanza la implementación del software.
- Pruebas continuas: Se puede crear un código de prueba antes de la codificación.
- Programación por pares: Se recomienda que las tareas de desarrollo se lleven a cabo por dos personas en un mismo puesto. El código se revisa y se corrige mientras se escribe.
- Integración con el usuario: Se recomienda que el usuario final o un representante de este trabaje junto con los desarrolladores.
- Corrección de errores: Se debe corregir todos los errores antes de avanzar con una nueva funcionalidad y hacer entregas frecuentes.
- Refactorizar código: Se debe reescribir ciertas partes del código para aumentar su legibilidad pero sin modificar la funcionalidad que ya se tiene.
- Propiedad del código compartido: No se debe dividir las responsabilidades en módulos; se debe intentar que todo el grupo de trabajo pueda corregir y extender cualquier parte del proyecto que se realiza.

- Simplicidad en el código: Si el código es simple, los errores pueden ser detectados más fácilmente y si todo funciona se pueden añadir nuevas funcionalidades si es necesario. El *extreme programming* se enfoca en hacer código simple.

1.3.3.2 Planificación de iteraciones

El equipo de *extreme programming* planifica una serie de iteraciones que van de acuerdo a las funciones que va a tener el software; se deben hacer entregas cada vez que se termina una iteración.

1.3.3.3 Historia de usuario

La funcionalidad de la aplicación se representa mediante historias de usuario [11].

Las historias de usuario son tareas específicas de programación que incluyen: un código para identificarla, el usuario a quien está orientado, una descripción, el programador a cargo, entre otras. Cada historia de usuario está acompañada de una prueba de aceptación. Deben ser escritas sin lenguaje técnico.

1.3.3.3.1 Elementos de una Historia de usuario

Los elementos que conforman una historia de usuario son:

- Número: Mediante este se distingue de forma única cada historia de usuario.
- Nombre historia de usuario: Nombre que identifica a cada historia de usuario.
- Usuario: Indica el perfil de usuario.
- Prioridad: Se califica en base a la importancia que la historia de usuario tendrá en la aplicación; así se tiene: *Alta* si es indispensable para el funcionamiento del sistema, *Media* si es necesaria pero no indispensable y *Baja* si no es indispensable.
- Iteración: Presenta el número de iteración asignada.
- Riesgo: Indica la complejidad de darle una solución a la historia de usuario.

- Estimación: Número de horas que se espera para el desarrollo de la historia de usuario.
- Programador responsable: Es la persona encargada de llevar a cabo con éxito la historia de usuario.
- Descripción: Se describe el requerimiento específico de la aplicación.
- Observaciones: Información adicional acerca de la historia de usuario. Puede ser opcional.

La **Figura 1.5** muestra la representación de una historia de usuario.

Historia de Usuario	
Número:	Nombre historia de usuario:
Usuario:	
Prioridad:	Iteración asignada:
Riesgo:	Estimación:
Programador responsable:	
Descripción:	
Observaciones:	

Figura 1.5. Modelo de historia de usuario [12]

1.3.3.4 Pruebas unitarias

Se deben realizar pruebas a los módulos del código de manera individual; una vez que los módulos individuales pasan estas pruebas se espera que el código funcione correctamente de manera colectiva.

1.3.3.5 Pruebas de aceptación

Son creadas basándose en las historias de usuario. Se las realiza al finalizar una iteración. Se deben plantear diferentes escenarios y con esto se espera verificar

que la historia de usuario fue correctamente desarrollada.

1.3.3.5.1 Elementos de una Prueba de aceptación

Los elementos que conforman una prueba de aceptación son:

- Número: Mediante este se distingue de forma única cada prueba de aceptación.
- Historia de usuario: Indica la historia de usuario a la cual pertenece la prueba.
- Nombre: Nombre que identifica a cada prueba.
- Descripción: De una forma resumida se describe lo que debe cumplir la prueba.
- Condiciones de Ejecución: Indica las condiciones que se deben cumplir para realizar la prueba.
- Pasos de Ejecuciones: Indica los pasos que se siguen para realizar la prueba.
- Resultado Esperado: Indica el resultado que se espera en la prueba
- Evaluación de la prueba: Indica si se cumplió o no el objetivo de la prueba.

La **Figura 1.6** muestra cómo se define una prueba de aceptación.

Prueba de Aceptación	
Número:	
Historia de Usuario:	
Nombre:	
Descripción:	
Condiciones de Ejecución:	
Pasos de Ejecución:	
Resultado Esperado:	
Evaluación de la prueba:	

Figura 1.6. Modelo de prueba de aceptación [13]

1.4 ANDROID

Es un sistema operativo basado en Linux, que fue pensado inicialmente para teléfonos móviles.

Las aplicaciones de Android son creadas con el lenguaje de programación Java, esto implica que es necesaria una máquina virtual para ejecutar estas aplicaciones. La máquina virtual que usa Android es llamada Dalvik [14].

1.4.1 ARQUITECTURA

La arquitectura de Android está formada por los siguientes componentes:

- Linux *Kernel*: Es el nivel más bajo de software que interactúa con el hardware del dispositivo.
- *Libraries*: Incluye un conjunto de librerías que proveen las funciones necesarias al sistema Android.
- Android *Runtime*: Es un entorno de ejecución de aplicaciones que utiliza el sistema operativo móvil Android.
- *Application Framework*: Proporciona una plataforma libre para el desarrollo de aplicaciones.
- *Applications*: Conjunto de aplicaciones básicas como: SMS[4], mapas e-mail y navegador.

La **Figura 1.7** muestra los componentes de la arquitectura de Android.

Para conocer más detalles acerca de la arquitectura de Android se recomienda revisar [15]**.**

1.4.2 DALVIK

Dalvik es una máquina virtual que se ejecuta como un proceso normal dentro del sistema operativo; su propósito es proporcionar un entorno de ejecución independiente entre sistema operativo y hardware.

[4] SMS: *Short Message Service* es un servicio disponible en los teléfonos móviles que permite el envío de mensajes cortos.

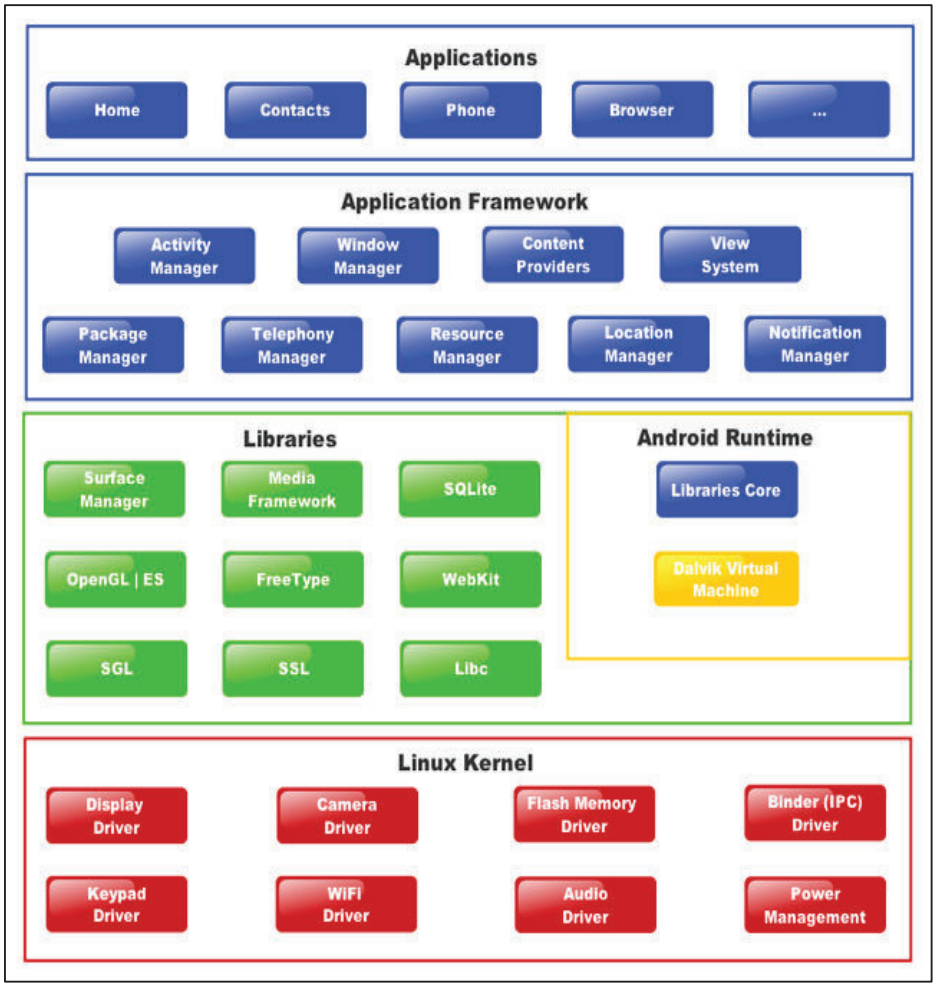


Figura 1.7. Estructura Android [16]

1.4.2.1 Características de Dalvik

Dalvik presenta las siguientes características:

- El *bytecode*[5] generado se transforma a *bytecodes* de Java; posteriormente se transforma en el usado por Android, así primero se generan los archivos .class[6], estos se transforman en archivos .dex[7] que se comprimen en paquetes APK[8].

[5] *Bytecode*: Es un archivo con datos binarios que contiene un programa ejecutable.
[6] .class: Es un archivo de java que puede ser ejecutado en el *Java Virtual Machine* .
[7] .dex: Es un formato de archivo que contiene compilado el código escrito para Android.
[8] APK: *Android Aplication Package* es un archivo de instalación para el sistema operativo Android.

- Está diseñada para ejecutar varias instancias de la propia máquina simultáneamente.
- El formato de los archivos ejecutables es compacto y óptimo para sistemas limitados con poca memoria y baja velocidad de procesador.

1.4.3 VERSIONES MÁS USADAS DE ANDROID

A continuación se presentan las versiones de Android que son usadas con frecuencia:

1.4.3.1 Gingerbread

Fue lanzada el 6 de diciembre del 2010. Está basada en el núcleo 2.6.35 de Linux e incluye las siguientes características [17]:

- Actualización de interfaz de usuario
- Soporte para tamaños y resoluciones de pantalla extra grandes WXGA[9]
- Soporte nativo para SIP[10] y telefonía VoIP[11]
- Soporte para NFC[12]

1.4.3.2 Honeycomb

Fue lanzada el 22 de febrero del 2011. Está basada en el núcleo 2.6.36 de Linux e incluye las siguientes características [18]:

- Soporte optimizado para tablets
- Agrega una barra de sistema con características de acceso rápido
- Modo incógnito en el navegador

---

[9] WXGA: *Wide Extended Graphics Array,* es una norma de visualización de gráficos.
[10] SIP: *Session Initiation Protoco*, es un protocolo desarrollado para iniciar, modificar y finalizar sesiones donde intervienen elementos multimedia.
[11] VoIP: Voz por Protocolo de Internet es un grupo de recursos que hacen posible que la señal de voz viaje a través de Internet empleando un protocolo IP.
[12] NFC: *Near Field Communicatio*n es una tecnología de comunicación inalámbrica, de corto alcance y alta frecuencia que permite el intercambio de datos entre dispositivos.

1.4.3.3 Ice Cream Sandwich

Fue lanzada el 19 de octubre del 2011. Está basada en el núcleo 3.0.1 de Linux e incluye las siguientes características [19]:

- Integra un mecanismo para captura de pantalla
- Corrector ortográfico del teclado
- Desbloqueo facial

1.4.3.4 Jelly Bean

Fue lanzada el 27 de junio del 2012. Está basada en el núcleo 3.0.31 de Linux e incluye las siguientes características [20]:

- Soporta *Bluetooth*[13] 4.0
- Mejoras en seguridad

1.4.3.5 KitKat

Fue lanzada el 31 de octubre del 2013, sus principales características son [21]:

- Posibilidad de impresión mediante Wi-Fi[14]
- Los monitores de actividad de red y señal han sido desplazados al menú de ajustes rápidos
- Se puede ocultar el teclado simplemente pulsando en una parte vacía de la pantalla
- Arreglos de seguridad en la depuración USB

1.4.3.6 Lollipop

Fue lanzada el 3 de noviembre del 2014; algunas de sus características son [22]:

- Soporte para CPU de 64 de bits
- Se incluye la opción de ajustes rápidos desde la cual se tiene acceso a control emparejado dispositivos *Bluetooth* y redes Wi-Fi

[13] *Bluetooth*: Es una especificación industrial para Redes Inalámbricas de Área Personal.
[14] Wi-Fi: Es un mecanismo de conexión de dispositivos electrónicos de forma inalámbrica.

- Soporte para múltiples tarjetas SIM[15]
- *Project Volta*[16], para mejoras en la vida de la batería

1.5 BASE DE DATOS

Es un conjunto de datos que pertenecen a un mismo contexto, los cuales se almacenan ordenadamente para su posterior uso. Estos datos almacenados pueden relacionarse entre sí y definir un determinado objeto.

1.5.1 Componentes de una base de datos

Existen tres elementos que son parte una base de datos los cuales son:

- Tablas: Es una estructura que sirve para almacenar los datos.
- Consultas: Permiten acceder a los datos de una o más tablas y filtrarlos según algún criterio de búsqueda.
- Formularios: Son pantallas para facilitar el ingreso de datos.

1.5.2 Sistema de Gestión de Base de Datos (SGBD)

Es un software muy específico que sirve de interfaz entre la parte donde se almacenan los datos, el usuario y las aplicaciones que utiliza. Está compuesto por DDL[17], DML[18] y SQL[19].

Se pueden encontrar sistemas gestores de bases de datos como por ejemplo: Microsoft Access, Microsoft SQL Server, Oracle, MySQL, etc. [23]

---

[15] SIM: *Subscriber Identity Module* es una tarjeta inteligente desmontable usada en teléfonos móviles, módems.
[16] *Project Volta*: conjunto de características que ayudan a que Android consuma menos batería.
[17] DDL: *Data Definition Languaje* es un lenguaje para definir y describir los objetos de la base de datos, su estructura, relaciones y restricciones.
[18] DML: *Data Manipulation Language* es un lenguaje proporcionado por los sistemas gestores de bases de datos que permite a los usuarios de la misma llevar a cabo las tareas de consulta o modificación de los datos.
[19] SQL: *Structured Query Language* es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en ellas.

La **Figura 1.8** muestra la interacción entre los usuarios y la base de datos mediante un sistema gestor de base de datos.

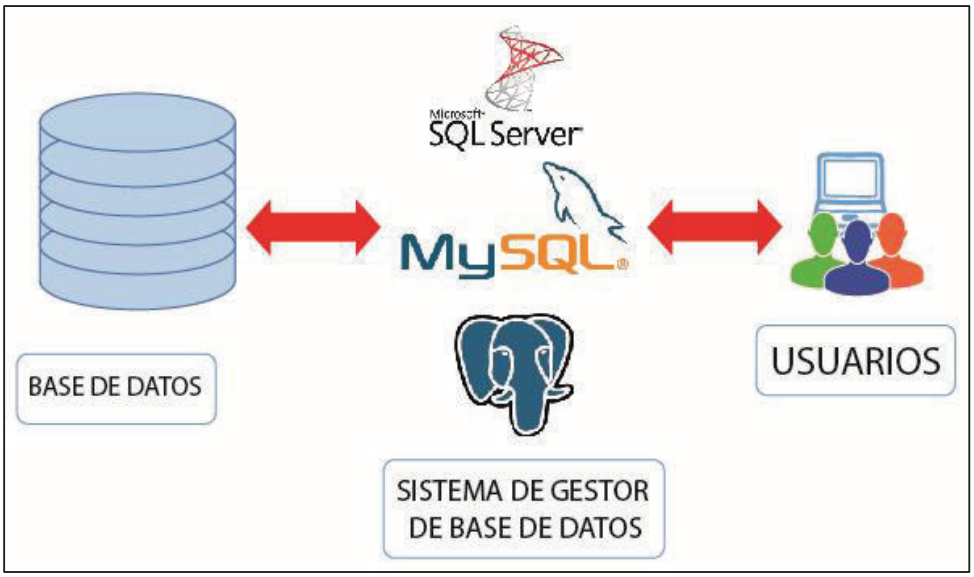


Figura 1.8. Estructura de una base de datos

1.6 PROTOCOLO SIP

Es un protocolo de control y señalización usado frecuentemente en los sistemas de Telefonía IP. Fue desarrollado por el IETF[20] y se encuentra definido en el RFC 3261 [24]. Este protocolo permite crear, modificar y finalizar sesiones multimedia entre dos o más usuarios [25].

1.6.1 FUNCIONES

SIP cuenta con las siguientes funciones:

- Soporte para movilidad
- Disponibilidad del usuario
- Establecimiento y mantenimiento de sesiones

SIP permite la interacción entre dispositivos haciendo uso de varios tipos de mensajes propios del protocolo. Estos mensajes proporcionan la capacidad de

[20] IETF: *Internet Engineering Task Force* es una organización internacional abierta de normalización, que tiene como objetivos el contribuir a la ingeniería de Internet.

registrar o invitar a usuarios a sesiones, establecer comunicaciones entre dos o más dispositivos y finalizar la sesión. La **Figura 1.9** muestra el esquema de conexión SIP para una llamada telefónica [26].

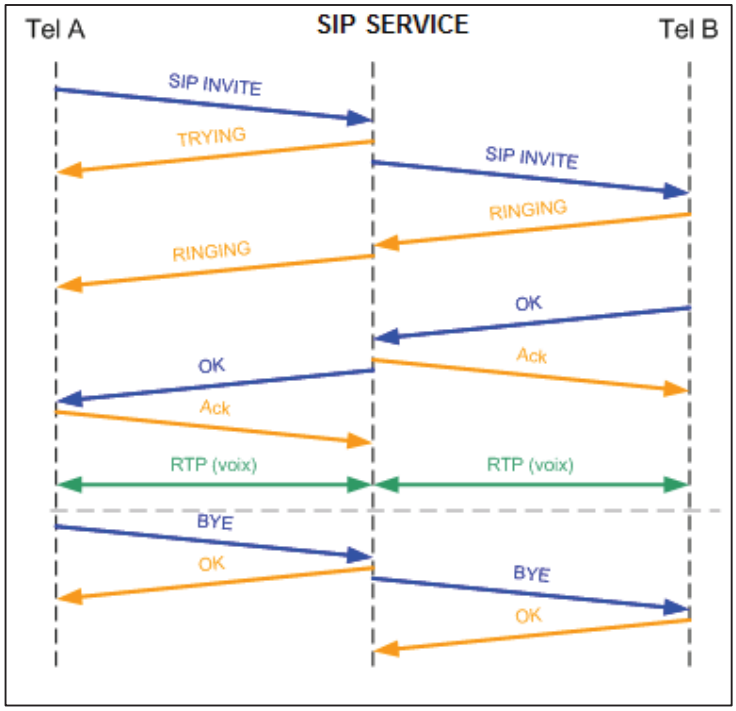


Figura 1.9. Esquema de conexión SIP

1.6.2 CARACTERÍSTICAS

SIP tiene las siguientes características:

- Control de llamadas sin estado
- Proporciona escalabilidad entre dispositivos telefónicos y servidores
- Necesita menos ciclos de CPU para generar mensajes de señalización en comparación con otros protocolos
- Es independiente de la capa de transporte
- Soporta autenticación mediante mecanismos de HTTP

SIP no está limitado a la Telefonía IP, también permite establecer presencia del usuario. Los mensajes SIP pueden transportar *payload* de tipo arbitrario o cualquier aplicación que haga uso de sesiones como:

- Sistemas de realidad virtual distribuidos

- Juegos en red
- Videoconferencia

1.7 PROTOCOLO RDP

Es un protocolo desarrollado por Microsoft para comunicación entre un terminal y un servidor por medio de la recepción de información proveniente de los periféricos del terminal.

1.7.1 CARACTERÍSTICAS

A continuación se presentan las características de RDP.

- La información gráfica generada por el servidor se envía a través del canal de comunicaciones al terminal
- Interpreta la información y se muestra en la pantalla. Las acciones de los periféricos se envían cifradas al servidor
- La información intercambiada entre equipos se comprime para mejorar el rendimiento
- RDP funciona en el puerto TCP 3389 y se usa para administrar equipos de manera remota [27]

CAPÍTULO 2 DISEÑO DEL SISTEMA

2.1 INTRODUCCIÓN

En este capítulo se presentarán los requerimientos para el prototipo de sistema de gestión, que desde ahora se lo llamará sistema, a partir del análisis de aplicaciones como iTalc[21] y NetSupport Mananger[22], que tienen una funcionalidad similar a las del sistema propuesto.

Para obtener los requerimientos básicos del sistema se describen ciertas características básicas de los programas antes mencionados, así como su funcionamiento. Posteriormente se realiza una comparación entre las aplicaciones analizadas y sobre la base de los resultados se establecen los requerimientos mínimos del sistema.

Para finalizar con los requerimientos mínimos obtenidos y con otros requerimientos propuestos en el plan de Proyecto de Titulación, se procede a realizar el diseño de cada uno de los componentes que están considerados en el sistema.

2.2 ANÁLISIS DE APLICACIONES

2.2.1 iTALC

2.2.1.1 Introducción

iTalc es una herramienta didáctica enfocada para profesores, que permite ver y controlar otros PC en la misma red, soporta sistemas operativos como Linux y Windows (XP, 7) y puede usarse en ambientes heterogéneos. La herramienta es gratuita por lo que no es necesario pagar por ningún tipo de licencia [28].

Se puede contactar a los desarrolladores para obtener soporte, pero este tiene un costo.

---

[21] iTalc: *Intelligent Teaching And Learning with Computers* es un software de control remoto con licencia libre.

[22] NetSupport Manager : Software de control remoto multiplataforma.

Esta herramienta ha sido diseñada para ser usada en escuelas; sin embargo, ofrece muchas posibilidades para los profesores como:

- Ver que está pasando en un PC.
- Controlar remotamente un PC.
- Mostrar la pantalla del profesor en las pantallas de los PC.
- Bloquear un PC.
- Enviar mensajes a un PC.

Los usuarios tienen una interfaz desde la cual se puede controlar los PC y ver cuáles están encendidos; además cuenta con tareas básicas de administración como encender, apagar y reiniciar.

En las siguientes secciones se analiza el funcionamiento de la aplicación.

2.2.1.2 Funcionalidad de la aplicación

2.2.1.2.1 Configuración de iTalc

Permite configurar aspectos básicos de la aplicación. La **Figura 2.1** muestra una captura de pantalla de la interfaz de configuración.

Se debe configurar al menos un usuario para poder usar la aplicación. Este puede ser uno que exista en el sistema operativo o uno nuevo. La **Figura 2.2** muestra una captura de pantalla de la interfaz que permite agregar usuarios.

Si se emplea un usuario existente se debe autenticar previamente para poder usar iTalc.

2.2.1.2.2 Pantalla principal

La pantalla principal está formada por dos barras: una horizontal y otra vertical, las cuales contienen los botones que proporcionan las diferentes funcionalidades que ofrece la aplicación.

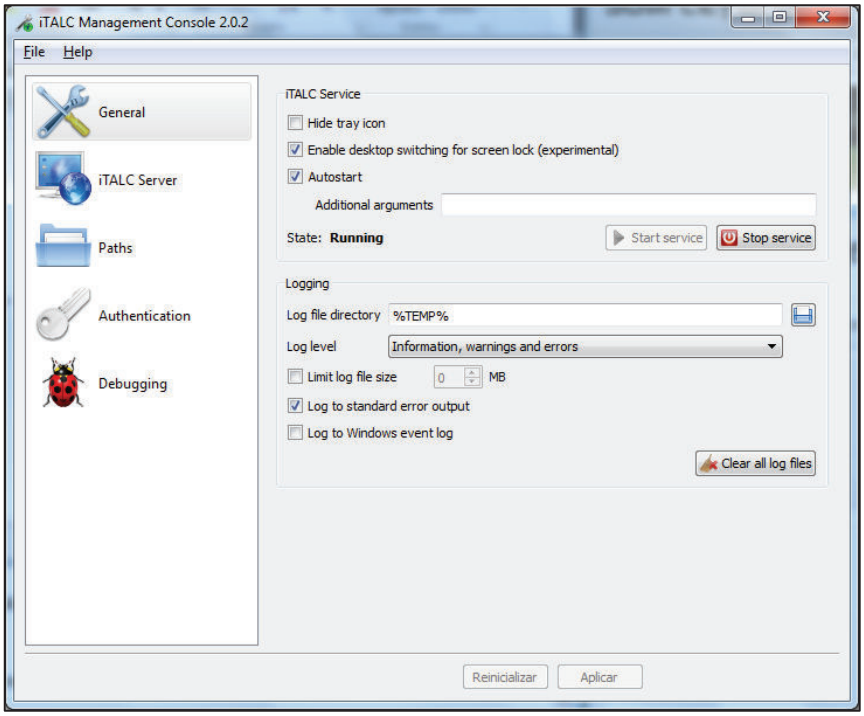


Figura 2.1. Ventana de configuración en iTalc

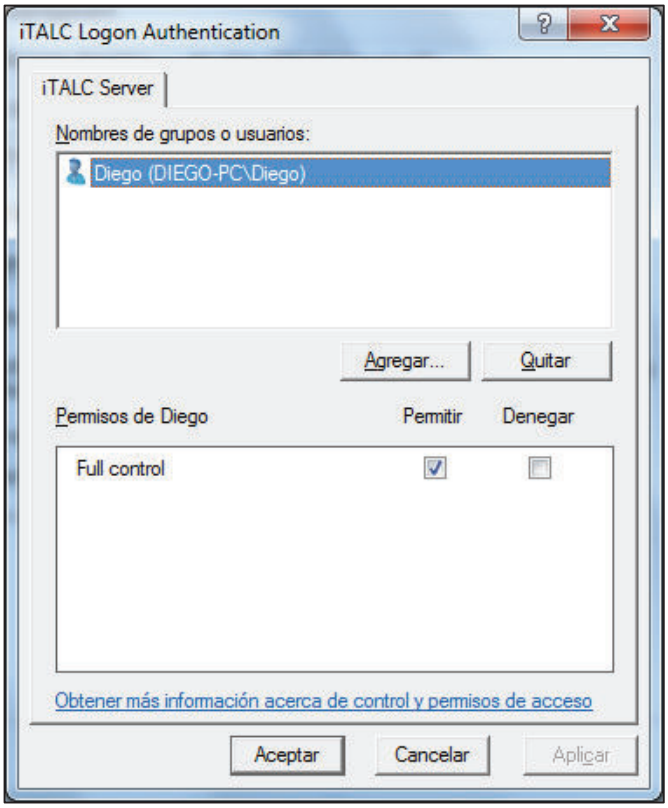


Figura 2.2. Ventana para agregar usuarios en iTalc

La barra horizontal tiene las siguientes opciones:

- `Clase`: Esta opción abre un menú donde se puede seleccionar la clase activa[23].
- `Overview`: Este es el modo por defecto que permite tener una visión de todos los PC disponibles en la clase activa.
- `Fullscreen Demo`: En este modo la pantalla del profesor se mostrará en una ventana nueva en los PC; además, el usuario no podrá realizar ninguna acción en el PC.
- `Window Demo`: En este modo la pantalla del profesor se mostrará en una ventana nueva en los PC; sin embargo, el usuario podrá seguir usando el PC sin restricción.
- `Bloquear Todas`: Con esta opción se bloquearán todos los PC disponibles en la clase activa.
- `Mensaje`: Envía mensajes a los PC.
- `Encender`: Esta función enciende los PC.
- `Apagar`: Esta función apaga los PC.
- `Soporte técnico`: Esta función permite conectarse a un PC sin necesidad de estar agregado a la clase activa.

La **Figura 2.3** muestra una captura de pantalla de la barra de trabajo de iTalc.

Figura 2.3. Barra de trabajo de iTalc

2.2.1.2.3 Configuración de un nuevo PC

Para agregar un nuevo PC es necesario agregar una nueva clase desde la sección `Administrador de Clases`. La **Figura 2.4** muestra una captura de la pantalla del `Administrador de Clases`.

Una vez agregado el PC se lo puede visualizar en la pantalla principal, desde donde se pueden realizar las acciones anteriormente mencionadas.

[23] Clase activa: Conjunto de los PC que fueron agregados a un grupo que se lo denomina clase.

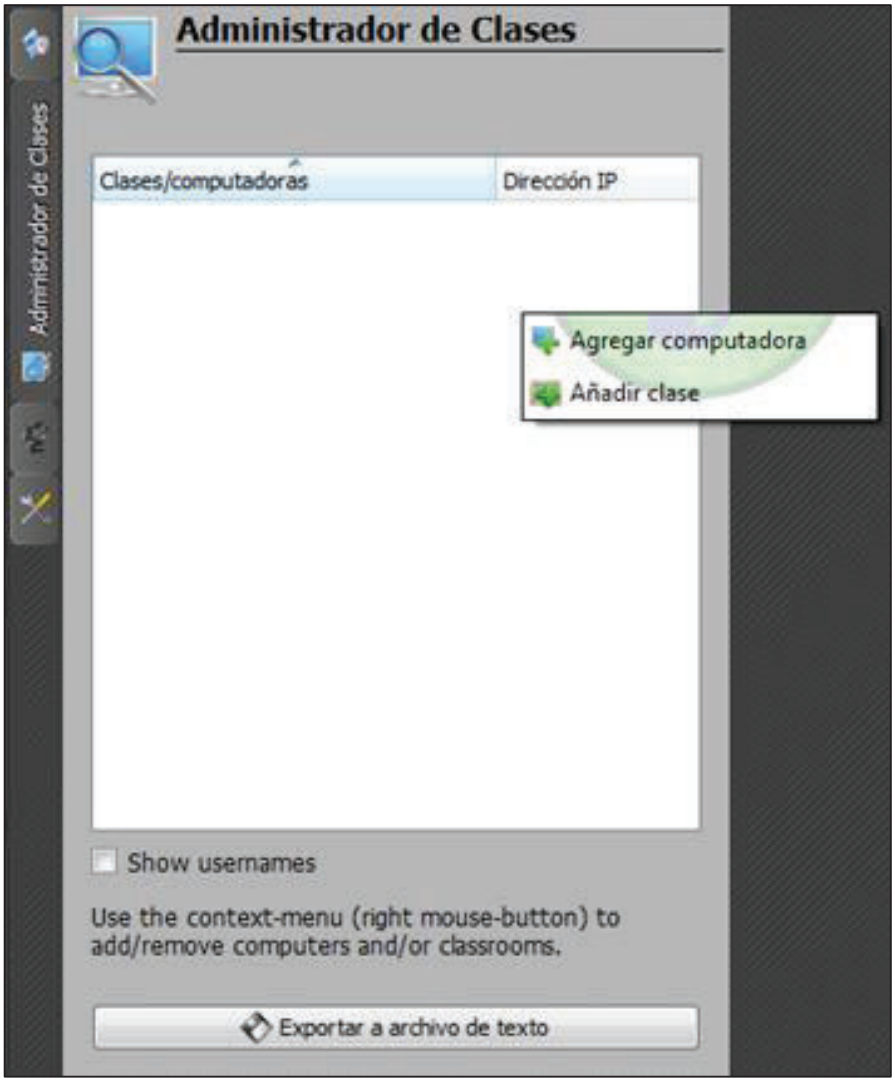


Figura 2.4. Administrador de clases en iTalc

2.2.1.2.4 Pantalla de control de un PC

Esta pantalla aparece al hacer doble clic en los PC disponibles y en esta se pueden realizar acciones como bloquear, desbloquear, controlar o agrandar la ventana a pantalla completa. La **Figura 2.5** muestra una captura de pantalla del control de un PC.

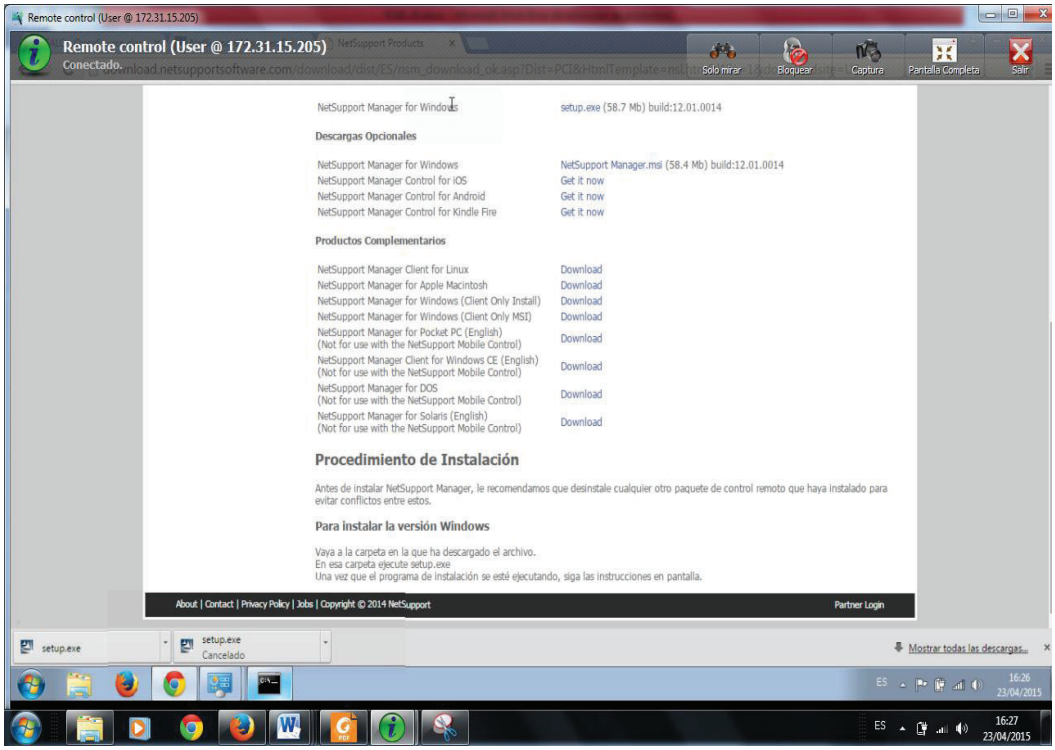


Figura 2.5. Pantalla de control de un PC desde iTalc

2.2.1.2.5 Problemas encontrados

Después de las pruebas realizadas a la aplicación se han determinado los siguientes problemas:

- La opción de bloquear PC no funciona correctamente; al presionar el botón bloquear no todos los PC realizan esta acción con éxito.
- Cuando un PC se bloquea exitosamente es muy sencillo quitar el bloqueo impuesto por el programa iniciando el administrador de tareas y finalizando el proceso.
- Se debe tener el mismo usuario tanto en los PC cliente como en PC del profesor.
- Se presentan fallos en la aplicación, los cuales ocasionan que se cierre constantemente. La **Figura 2.6** muestra una captura de pantalla de un mensaje del cierre inesperado de la aplicación.

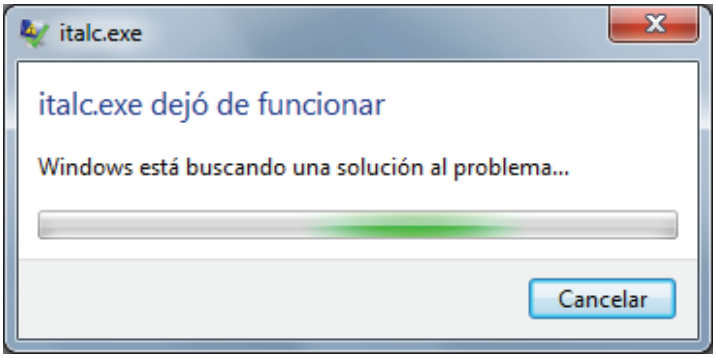


Figura 2.6. Mensaje de cierre inesperado de iTalc

2.2.2 NETSUPPORT MANAGER

2.2.2.1 Introducción

La aplicación NetSupport Manager fue desarrollado por la compañía NetSupport cuya sede se encuentra en Reino Unido. Se encuentra instalada en aproximadamente 12.000.000 de PC; además ofrece una solución para control remoto muy estable y con diferentes funcionalidades además ser segura y reconocida a nivel mundial [29].

NetSupport Manager cuenta con una aplicación multiplataforma desde la cual se puede tener un control remoto de los PC, administrar archivos, diagnosticar el PC, transmitir audio, entre otras.

Se debe comprar una licencia única de por vida; el precio varía dependiendo del número de PC a los cuales se les va a instalar la aplicación.

2.2.2.2 Funcionalidad de la aplicación

2.2.2.2.1 Pantalla principal

Una vez instalada e iniciada la aplicación, se muestra la pantalla principal; en esta se encuentran las diferentes opciones de administración.

No es necesaria ninguna configuración previa para usar la aplicación.

En la pantalla principal se encuentra la barra de herramientas, la cual tiene las siguientes opciones:

- `Nuevo`: Permite agregar nuevos PC o grupos.
- `Conectar`: Establece la conexión con los PC seleccionados.
- `Desconectar`: Termina la conexión con los PC seleccionados.
- `Examinar`: Busca los PC en la red.
- `Acciones`: Permite realizar acciones como enviar mensajes o comandos.
- `Escritorio`: Establece una conexión para controlar remotamente el PC.
- `Gestionar`: Permite encender, apagar o iniciar sesión en los PC.
- `Archivo`: Permite la transferencia de archivos.
- `Mostrar`: Muestra la pantalla del profesor en los PC.

La **Figura 2.7** muestra la barra de herramientas de Netsupport Manager

Figura 2.7. Barra de herramientas de NetSupport Manager

En la barra de herramientas se encuentra la opción `Examinar` desde la cual se puede agregar automáticamente los PC, en los cuales esté instalada la aplicación cliente de NetSupport Manager. La **Figura 2.8** muestra la sección `Examinar` de la barra de herramientas.

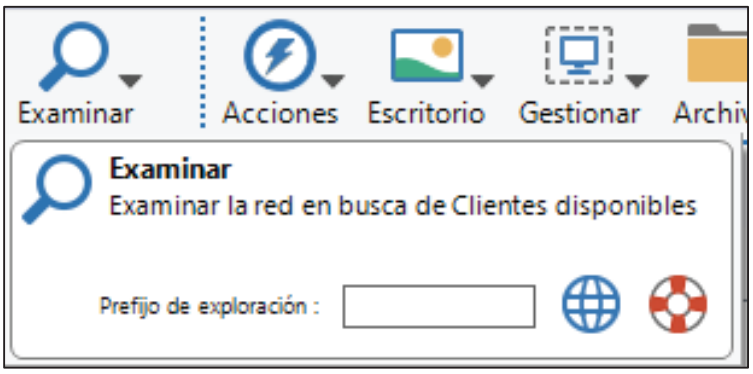


Figura 2.8. Opción examinar de NetSupport Manager

La opción `Gestionar` dispone de un menú que permite realizar acciones como: reiniciar, apagar, encender, cerrar e iniciar sesión. La **Figura 2.9** muestra las diferentes opciones del menú `Gestionar`.

Una vez agregados los PC, estos se muestran en la pantalla principal. Al acercar el mouse se puede ver información del PC como nombre de usuario, fecha y hora del PC, dirección IP y sistema operativo.

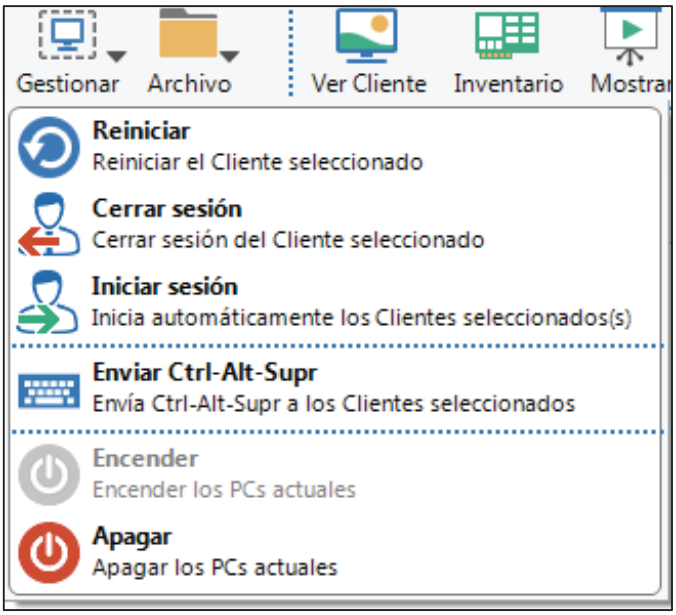


Figura 2.9. Menú gestionar de NetSupport Manager

2.2.2.2.2 Pantalla de control de un PC

Esta pantalla se muestra al hacer doble clic en los PC que fueron registrados en la aplicación y permite realizar acciones como: modo ver, ajustar, pantalla completa, capturar imagen, inventario, transferir, conversar, lanzar aplicación, entre otras. La **Figura 2.10** muestra una captura de la pantalla de la ventana de control de un PC.

NetSupport Manager cuenta con la posibilidad de enviar la pantalla del PC del profesor a los PC de los estudiantes. La **Figura 2.11** muestra una captura de pantalla de la opción para enviar pantallas.

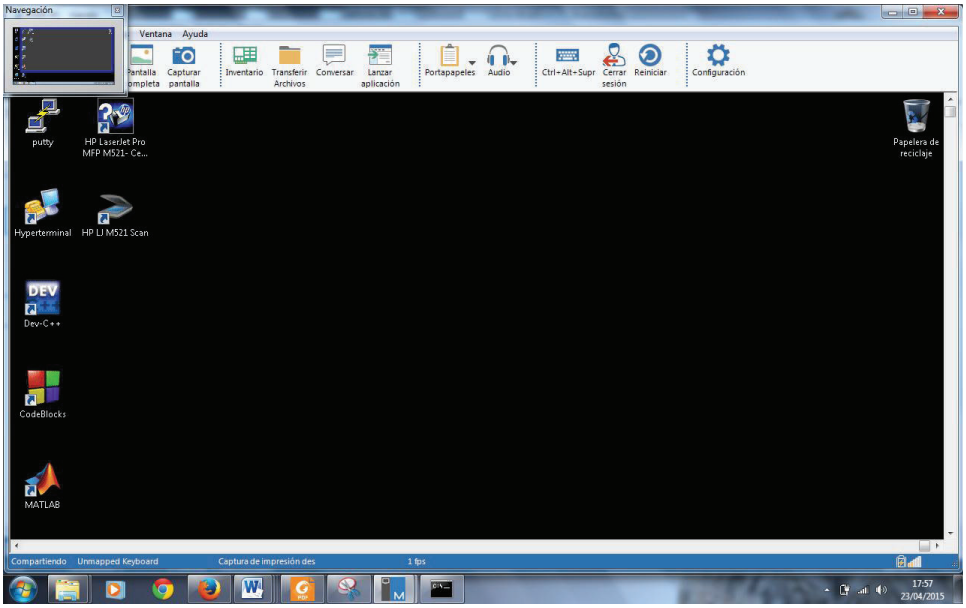


Figura 2.10. Ventana de control de un PC en NetSupport Manager

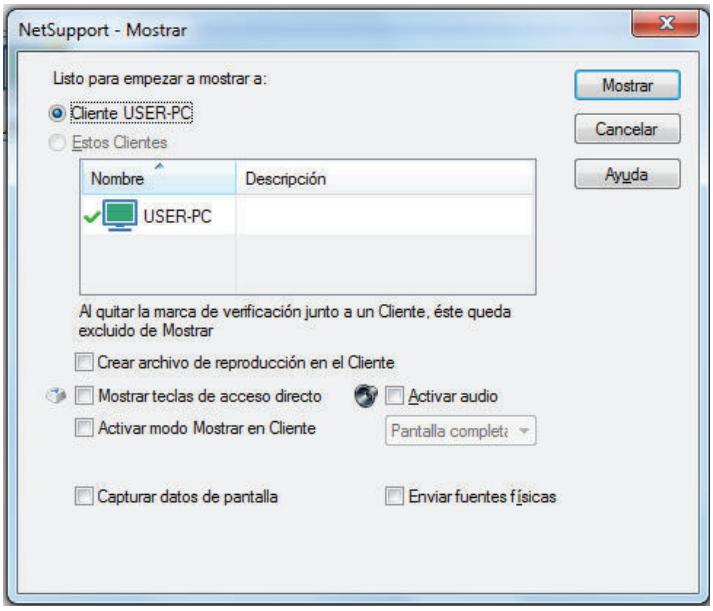


Figura 2.11. Ventana de opción para enviar pantalla

2.2.2.2.3 Problemas encontrados

Después de las pruebas realizadas a la aplicación se han determinado los siguientes problemas:

- NetSupport Manager no cuenta con la opción de bloquear y desbloquear.

- El PC se puede desvincular con mucha facilidad de la aplicación al desconectar el cable de conexión de red.

2.2.3 COMPARACIÓN DE APLICACIONES

En la **Tabla 2.1** se presenta una comparación de las aplicaciones anteriormente analizadas.

Características	iTalc	Netsupport Manager
Autenticación	Es necesario tener una cuenta de usuario de Windows con contraseña para la autenticación cuando se inicia la aplicación	No cuenta con ningún método de autenticación
Pantalla principal	Cuenta con una interfaz intuitiva para el usuario	Cuenta con una interfaz intuitiva para el usuario
Barra de herramientas	Muestra información detallada de cada uno de sus elementos	Muestra información detallada de cada uno de sus elementos
Agregar nuevos PC	Se debe agregar manualmente cada uno de los PC	Cuenta con la opción examinar que simplifica la detección de los PC
Pantalla de control de los PC	Presenta una interfaz con opciones básicas para el control del PC	Presenta una interfaz con opciones básicas para el control del PC
Gestión	Encender, reiniciar, apagar, bloquear, desbloquear	Encender, reiniciar, apagar
Licencia	Gratuita	Pagada, costo depende del número de PC
Soporte	Pagado	Pagado

Tabla 2.1. Comparación de las aplicaciones iTalc y NetSupport Manager

Al comprar las aplicaciones se puede apreciar que NetSupport Manager cuenta con mejores prestaciones al ser una aplicación ampliamente desarrollada sin embargo al ser una versión de pago el usuario el usuario podría usar versiones no autorizadas, esto podría implicar problemas legales a futuro.

2.3 REQUERIMIENTOS DEL SISTEMA

Después de realizar el análisis de las aplicaciones iTalc y NetSupport Manager, se obtuvo una serie de requerimientos básicos para el sistema, las cuales se resumen en los siguientes párrafos.

El sistema estará compuesto por seis componentes fundamentales: una aplicación principal que permita manejar la información del sistema e interactuar con otras aplicaciones del mismo; una aplicación cliente la cual permitirá realizar las funciones de gestión en cada PC; una aplicación profesor la cual tiene la función de controlar a las aplicaciones clientes y comunicarse con la aplicación principal; una aplicación Android que permitirá recibir notificaciones de la aplicación principal y enviar información por medio de un protocolo para realizar distintas acciones en el sistema; una base de datos que almacenará la información del sistema y un servicio SIP que permita comunicarse por voz entre las aplicaciones Android. En la **Figura 2.12** se muestra el esquema general del sistema.

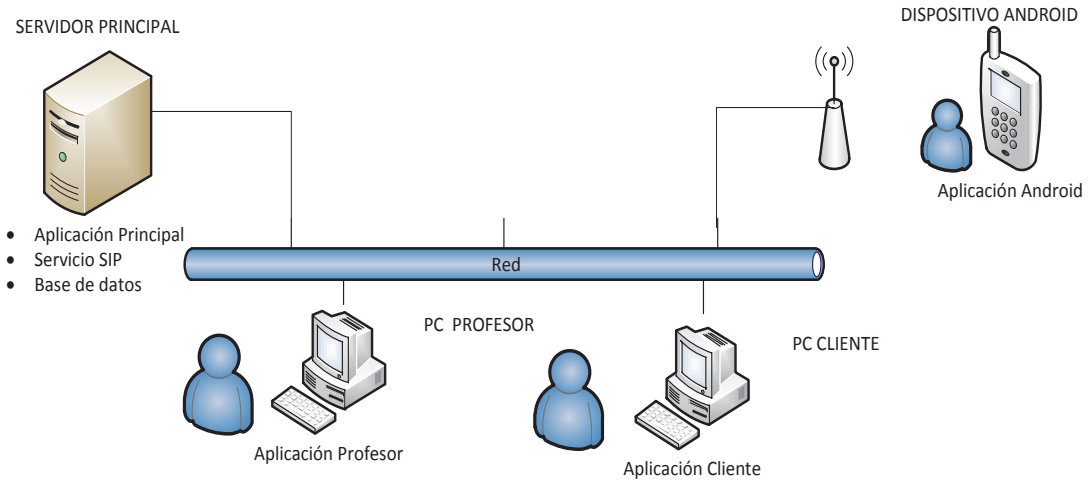


Figura 2.12. Esquema general del sistema

La aplicación principal tiene cuatro funciones:

- Inicio de sesión: Permitirá que los usuarios ingresen al sistema con perfiles de Profesor o Administrador.

- Gestión de PC: Permitirá gestionar a los PC del sistema y realizar acciones como: apagar, encender, bloquear, desbloquear y acceder mediante RDP para ver y controlar el PC.
- Administración de datos: Permitirá la conexión con la Base de datos, además se podrá insertar, actualizar y eliminar registros de la Base de datos.
- Préstamo de equipos: Permitirá obtener información de los ítems del catálogo de equipos y de las solicitudes activas.

La aplicación profesor tendrá dos funciones:

- Controlador: Permitirá conectarse con los PC a gestionar y realizar acciones como: apagar, encender, acceder mediante RDP para ver y controlar el PC, bloquear, y desbloquear.
- Cliente: Permitirá enviar información a la aplicación principal como: dirección IP, dirección MAC, nombre de equipo, grupo de trabajo al que pertenece el equipo. También permitirá acceder al préstamo de equipos de la aplicación principal.

La aplicación cliente dispondrá de la funcionalidad necesaria para conectarse a la aplicación profesor o a la aplicación principal y permitirá que el PC sea controlado por dichas aplicaciones.

La aplicación Android permitirá comunicarse con la aplicación principal para usar las siguientes funcionalidades:

- Préstamo de equipos: Se podrá visualizar las solicitudes activas y cambiar su estado.
- Inicio de sesión: Permite el acceso a la aplicación únicamente a los usuarios con perfil de Administrador.
- Gestión de PC: Permitirá prender, apagar, bloquear y desbloquear los PC gestionados.
- Comunicación por voz: Permitirá mediante una lista de usuarios registrados en el sistema establecer una comunicación de voz mediante SIP.

La base de datos permitirá guardar la información de: usuarios, solicitudes, catálogo de equipos, equipos a gestionar y grupos de trabajo.

El servicio SIP permitirá establecer una comunicación de voz dentro de la red entre las aplicaciones Android.

El sistema además dispondrá de las siguientes funcionalidades generales

- Autenticar: El sistema cuenta con dos tipos de perfiles de usuarios: Profesor y Administrador, por lo que debe tener algún mecanismo que permita identificar estos perfiles y diferenciar las funciones de cada uno.
- Registrar usuarios: El sistema debe permitir registrar usuarios con los perfiles: Profesor y Administrador desde las diferentes aplicaciones. Los usuarios posteriormente podrán ser eliminados o modificados.
- Recuperar contraseña: En caso de que los usuarios del sistema olviden su contraseña, este generará una nueva contraseña la cual será enviada al correo electrónico registrado. Posteriormente esta contraseña puede ser cambiada.
- Registrar ítems al catálogo de equipos: El sistema permitirá ingresar nuevos ítems pertenecientes al inventario del Laboratorio de Informática de la FIEE al catálogo de equipos, los cuales posteriormente pueden ser eliminados o modificados.
- Gestionar PC: El sistema propuesto plantea la gestión de los diferentes PC ubicados en el Laboratorio de Informática de la FIEE, para lo cual permitirá establecer una comunicación entre la aplicación profesor o la aplicación principal con la aplicación cliente, para realizar acciones como: apagar, encender, reiniciar, bloquear, desbloquear, reiniciar en modo *Frozen*[24] y *Thawed*[25], acceder remotamente con RDP y obtener información.
- Reservar equipos: Se podrá acceder al listado de ítems dentro del catálogo de equipos y de esta manera reservarlos especificando la fecha y el turno[26] en el cual se desea ocuparlos.
- Ver solicitudes: Se podrá visualizar las solicitudes activas y archivadas, estas serán almacenadas en la base de datos. Una solicitud activa es aquella que fue generada y aún no ha sido atendida; una solicitud

[24] Reiniciar en modo *Frozen*: Reinicia los PC de tal forma que el sistema operativo no permita realizar cambios.
[25] Reiniciar en modo *Thawed*: Reinicia los PC de tal forma que el sistema operativo permita cambios.
[26] Turno: permitirá identificar para que turno del día se requiere el equipo.

archivada es aquella que ya se atendió y fue cambiada de estado por el Administrador.

- Cambiar estado de solicitudes: Se puede cambiar el estado de una solicitud activa a una solicitud archivada.

La **Tabla 2.2** muestra las funcionalidades con las que cuenta cada aplicación

Funcionalidades	Profesor	Principal	Android
Autenticar	X		X
Registrar usuarios	X	X	
Recuperar contraseña	X		
Registrar ítems al catálogo de equipos		X	
Gestionar PC	X	X	
Reservar equipos	X		
Ver solicitudes		X	
Cambiar estado de solicitudes		X	X

Tabla 2.2 Funcionalidades por aplicación

2.4 DIAGRAMA DE CASOS DE USO DEL SISTEMA

En esta sección se presentarán los diagramas de casos de uso que fueron generados a partir de los requerimientos anteriormente descritos tomando en cuenta los perfiles de usuario. La **Figura 2.13** muestra el diagrama de casos de uso de la aplicación principal, la **Figura 2.14** muestra el diagrama de casos de uso de la aplicación profesor y la **Figura 2.15** muestra el diagrama de casos de uso de la aplicación Android.

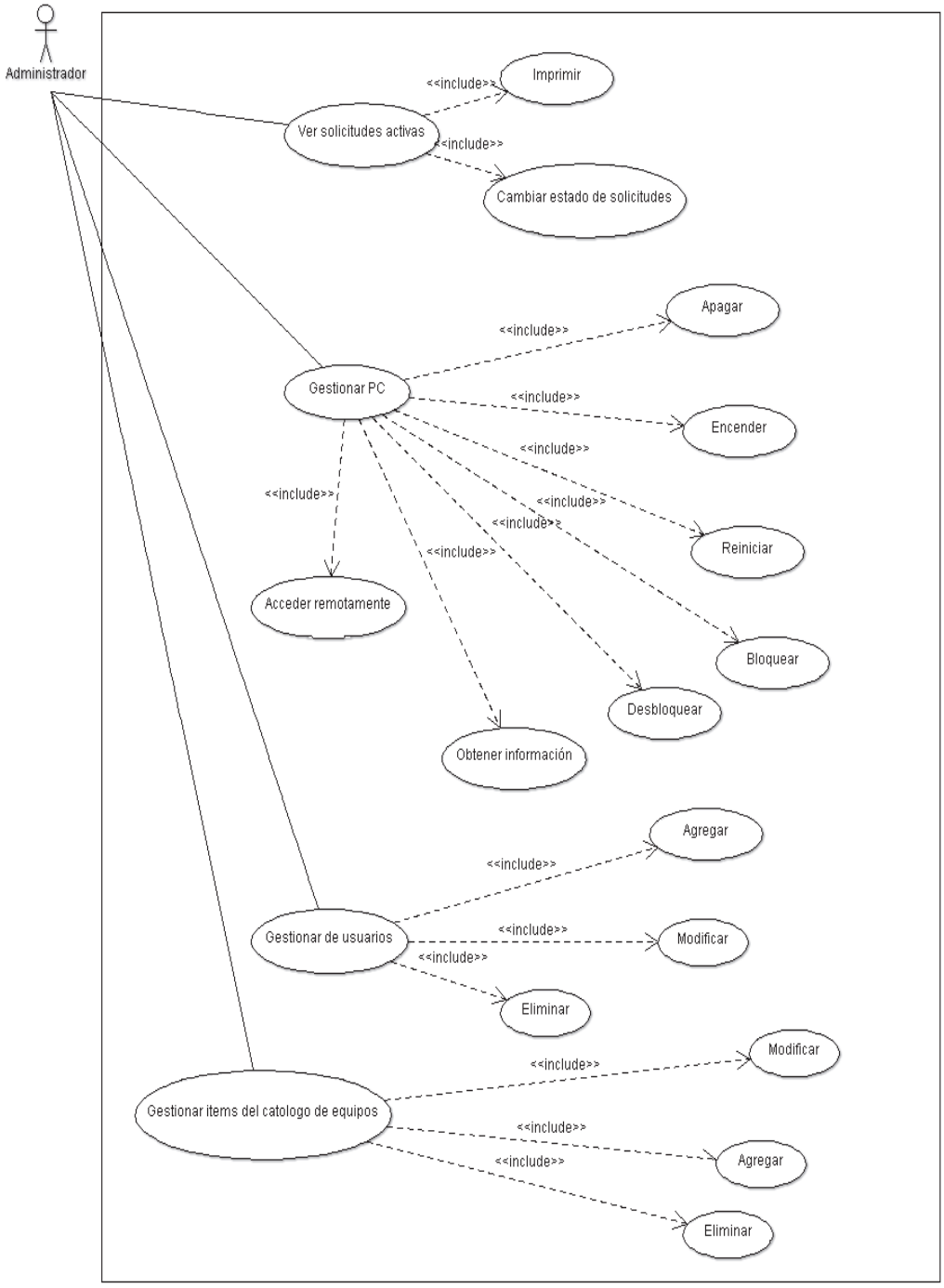


Figura 2.13. Diagrama de casos de uso de la aplicación principal

Figura 2.14. Diagrama de casos de uso de la aplicación profesor

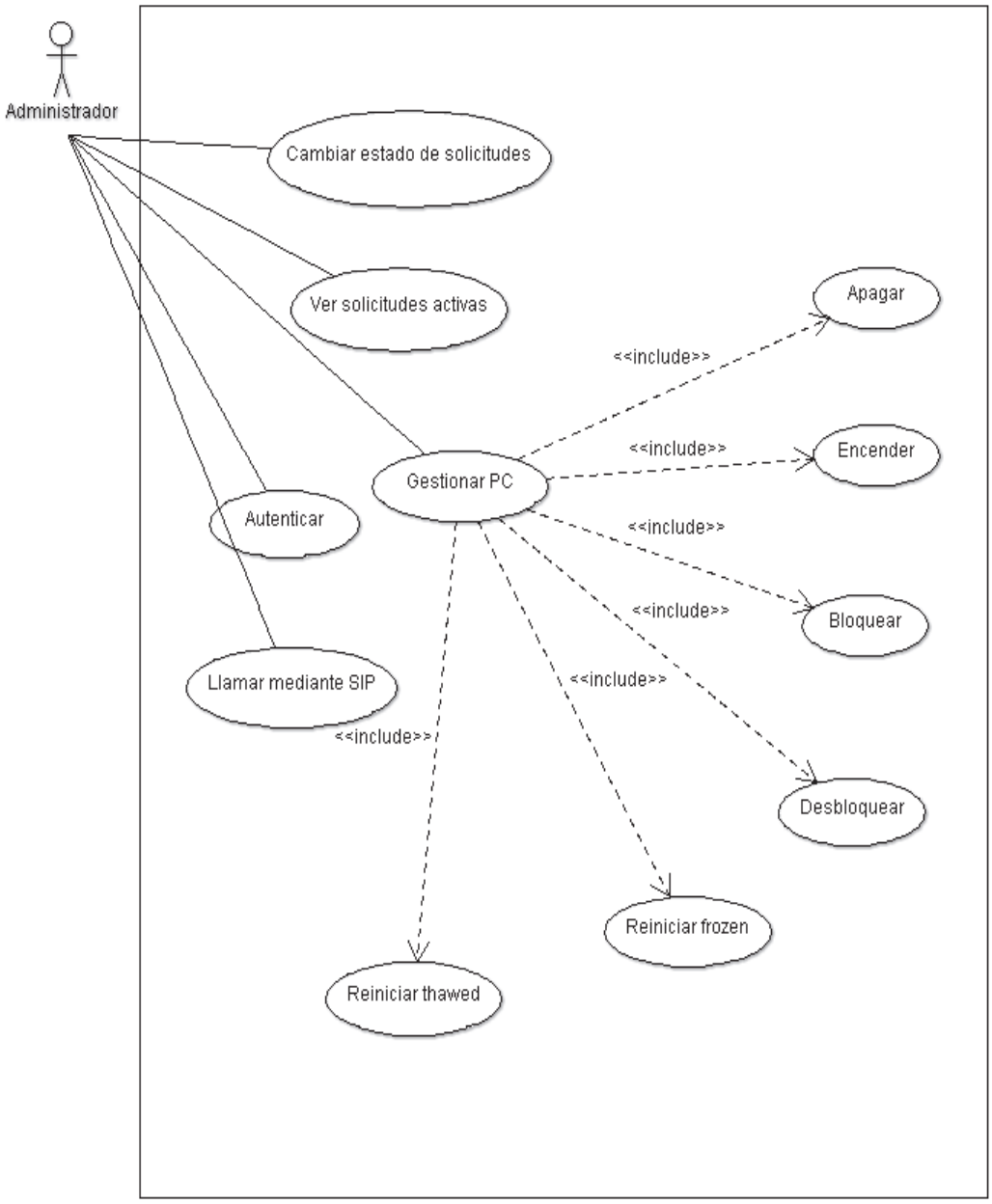


Figura 2.15. Diagrama de casos de uso de la aplicación Android

2.5 PLANIFICACIÓN DE ITERACIONES

Se han organizado los requerimientos usando historias de usuarios.

2.5.1 HISTORIAS DE USUARIO

La **Figura 2.16** muestra el formato que se usará para el número de historia de usuario.

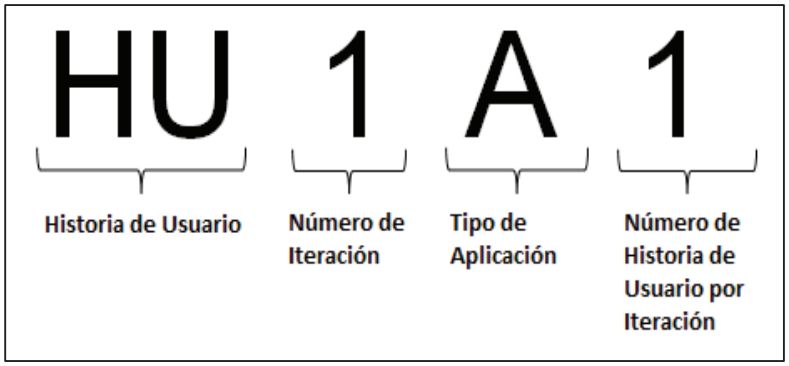


Figura 2.16. Formato del número de historia de usuario

Se tiene las siguientes aplicaciones: (A) aplicación Android, (B) base de datos, (C) aplicación cliente, (P) aplicación profesor, (S) aplicación principal.

2.5.1.1 Primera iteración

En esta sección se indican las historias de usuario que permitan un funcionamiento mínimo del sistema. La **Tabla 2.3** muestra la historia de usuario "Creación de base de datos", la **Tabla 2.4** muestra la historia de usuario "Autenticación de usuarios", la **Tabla 2.5** muestra la historia de usuario "Registro de nuevos usuarios", la **Tabla 2.6** muestra la historia de usuario "Recuperación de contraseña" y por último la **Tabla 2.7** y la **Tabla 2.8** muestran la historia de usuario "Registro de nuevos ítems en el catálogo de equipos".

2.5.1.2 Segunda iteración

En la segunda iteración se encuentran las historias de usuario que permiten la gestión de PC. La **Tabla 2.9** muestra la historia de usuario "Acción encender", la **Tabla 2.10** y la **Tabla 2.11** muestran la historia de usuario "Acción apagar", la **Tabla 2.12** muestra la historia de usuario "Acción reiniciar", la **Tabla 2.13** muestra

la historia de usuario “Acción bloquear”, la **Tabla 2.14** muestra la historia de usuario “Acción desbloquear”, la **Tabla 2.15** muestra la historia de usuario “Control acceso remoto” y finalmente la **Tabla 2.16** y la **Tabla 2.17** muestran la historia de usuario “Obtener información”.

Historia de Usuario	
Número: HU1B1	**Nombre historia de usuario:** Diseño de la base de datos
Usuario: Administrador, Profesor	
Prioridad: Alta	**Iteración asignada:** 1
Riesgo: Medio	**Estimación:** 20 horas
Programador responsable: Diego Viñamagua	
Descripción: Se requiere almacenar datos como: usuarios, ítems para el catálogo de equipos, solicitudes, información de los PC y grupos de trabajo	
Observaciones: Se debe diseñar la base de datos con las tablas necesarias para almacenar los datos requeridos. Las tablas serán: usuarios, equipos, solicitudes, computadores, grupos.	

Tabla 2.3. Historia de usuario - Diseño de la base de datos

Historia de Usuario	
Número: HU1S2	**Nombre historia de usuario:** Autenticación de usuarios
Usuario: Administrador, Profesor	
Prioridad: Alta	**Iteración asignada:** 1
Riesgo: Medio	**Estimación:** 20 horas
Programador responsable: Diego Viñamagua	
Descripción: Se requiere un mecanismo de autenticación para acceder a ciertas funcionalidades para los perfiles de administrador o profesor.	
Observaciones: Mediante este mecanismo los usuarios podrán acceder al sistema con diferentes perfiles.	

Tabla 2.4. Historia de usuario - Autenticación de usuarios

Historia de Usuario	
Número: HU1S3	**Nombre historia de usuario:** Registro de nuevos usuarios
Usuario: Administrador, Profesor	
Prioridad: Alta	**Iteración asignada:** 1
Riesgo: Medio	**Estimación:** 20 horas
Programador responsable: José Mosquera	
Descripción: Se requiere un mecanismo el cual permitirá crear nuevos usuarios con diferentes perfiles y almacenarlos en la base de datos.	
Observaciones: Este mecanismo será usado posteriormente por el profesor y el administrador para agregar nuevos usuarios.	

Tabla 2.5. Historia de usuario - Registro de nuevos usuarios

Historia de Usuario	
Número: HU1P4	**Nombre historia de usuario:** Recuperación de contraseña
Usuario: Administrador	
Prioridad: Alta	**Iteración asignada:** 1
Riesgo: Medio	**Estimación:** 20 horas
Programador responsable: Diego Viñamagua	
Descripción: Se requiere un mecanismo para crear una clave temporal y otro mecanismo para enviar esta clave por correo electrónico	
Observaciones: El usuario previamente en su registro debe ingresar un correo electrónico el cual se usará para la recuperación de contraseña.	

Tabla 2.6. Historia de usuario - Recuperación de contraseña

Historia de Usuario	
Número: HU1S5	**Nombre historia de usuario:** Registro de nuevos ítems en el catálogo de equipos
Usuario: Administrador	
Prioridad: Alta	**Iteración asignada:** 1

Tabla 2.7. Historia de usuario - Registro de nuevos ítems en el catálogo de equipos (Primera parte)

Riesgo: Medio	**Estimación:** 20 horas
Programador responsable: José Mosquera	
Descripción: Se requiere un mecanismo el cual permitirá agregar nuevos ítems al catálogo de equipos disponible en el sistema.	
Observaciones: Los ítems disponibles en el catálogo del sistema serán usados para el préstamo de equipos y tendrá los siguientes parámetros: nombre, apellido, curso, extensión SIP (solo en usuarios con perfil Administrador), usuario.	

Tabla 2.8. Historia de usuario - Registro de nuevos ítems en el catálogo de equipos (Segunda parte)

Historia de Usuario	
Número: HU2C1	**Nombre historia de usuario:** Acción encender
Usuario: Administrador, Profesor	
Prioridad: Media	**Iteración asignada:** 2
Riesgo: Bajo	**Estimación:** 4 horas
Programador responsable: José Mosquera	
Descripción: Se requiere un mecanismo para ejecutar la acción de encender en los PC clientes.	
Observaciones: Se debe enviar un mensaje Wake On Lan[27] a los PC cliente.	

Tabla 2.9. Historia de usuario - Acción encender

Historia de Usuario	
Número: HU2C2	**Nombre historia de usuario:** Acción apagar
Usuario: Administrador, Profesor	
Prioridad: Media	**Iteración asignada:** 2

Tabla 2.10. Historia de usuario - Acción apagar (Primera parte)

[27] *Wake On LAN:* es un estándar de redes Ethernet que permite encender remotamente computadoras apagadas.

Riesgo: Bajo	**Estimación:** 4 horas
Programador responsable: José Mosquera	
Descripción: Se requiere un mecanismo para ejecutar la acción de apagar en los PC en la aplicación cliente.	
Observaciones:	

Tabla 2.11. Historia de usuario - Acción apagar (Segunda parte)

Historia de Usuario	
Número: HU2C3	**Nombre historia de usuario:** Acción reiniciar
Usuario: Administrador, Profesor	
Prioridad: Media	**Iteración asignada:** 2
Riesgo: Bajo	**Estimación:** 4 horas
Programador responsable: Diego Viñamagua	
Descripción: Se requiere un mecanismo para ejecutar la acción de reiniciar en los PC en la aplicación cliente.	
Observaciones:	

Tabla 2.12. Historia de usuario - Acción reiniciar

Historia de Usuario	
Número: HU2C4	**Nombre historia de usuario:** Acción bloquear
Usuario: Administrador, Profesor	
Prioridad: Media	**Iteración asignada:** 2
Riesgo: Bajo	**Estimación:** 4 horas
Programador responsable: José Mosquera	
Descripción: Se requiere un mecanismo para ejecutar la acción de bloquear en los PC en la aplicación cliente.	
Observaciones: Se debe usar los métodos del sistema que permitan bloquear los periféricos de entrada.	

Tabla 2.13. Historia de usuario - Acción bloquear

Historia de Usuario	
Número: HU2C5	**Nombre historia de usuario:** Acción desbloquear
Usuario: Administrador, Profesor	
Prioridad: Media	**Iteración asignada:** 2
Riesgo: Bajo	**Estimación:** 4 horas
Programador responsable: Diego Viñamagua	
Descripción: Se requiere un mecanismo para ejecutar la acción de desbloquear en los PC en la aplicación cliente.	
Observaciones: Se debe usar los métodos del sistema que permitan desbloquear los periféricos de entrada.	

Tabla 2.14. Historia de usuario - Acción desbloquear

Historia de Usuario	
Número: HU2C6	**Nombre historia de usuario:** Acceso remoto
Usuario: Administrador, Profesor	
Prioridad: Media	**Iteración asignada:** 2
Riesgo: Medio	**Estimación:** 16 horas
Programador responsable: José Mosquera	
Descripción: Se requiere un mecanismo para acceder remotamente a los PC con la aplicación cliente.	
Observaciones: Mediante RDP se tomará el control del PC	

Tabla 2.15. Historia de usuario - Acceso remoto

Historia de Usuario	
Número: HU2C7	**Nombre historia de usuario:** Obtener información
Usuario: Administrador, Profesor	
Prioridad: Media	**Iteración asignada:** 2

Tabla 2.16. Historia de usuario – Obtener información (Primera parte)

Riesgo: Bajo	**Estimación:** 4 horas
Programador responsable: Diego Viñamagua	
Descripción: Se requiere un mecanismo para obtener información básica del PC gestionado.	
Observaciones: La información que se obtendrá será: IP, MAC, Grupo de trabajo	

Tabla 2.17. Historia de usuario – Obtener información (Segunda parte)

2.5.1.3 Tercera iteración

En la tercera iteración se encuentran las historias de usuario que permiten el acceso a la base de datos. La **Tabla 2.18** y la **Tabla 2.19** muestran la historia de usuario "Gestión de usuarios", la **Tabla 2.20** muestra la historia de usuario "Gestión ítem del catálogo de equipos", la **Tabla 2.21** muestra la historia de usuario "Gestión de solicitudes activas", la **Tabla 2.22** muestra la historia de usuario "Solicitudes archivadas" y la **Tabla 2.23** muestra la historia de usuario "Gestión de PC en BDD".

2.5.1.4 Cuarta iteración

En la cuarta iteración se encuentran las historias de usuario que permiten la comunicación entre las distintas aplicaciones. La **Tabla 2.24** muestra la historia de usuario "Comunicación aplicación Android con aplicación principal", la **Tabla 2.25** muestra la historia de usuario "Comunicación aplicación profesor con aplicación principal", la **Tabla 2.26** muestra la historia de usuario "Comunicación aplicación cliente con aplicación profesor o principal" y finalmente la **Tabla 2.27** y la **Tabla 2.28** muestra la historia de usuario "Comunicación por voz"

Historia de Usuario	
Número: HU3B1	**Nombre historia de usuario:** Gestión de usuarios
Usuario: Administrador	
Prioridad: Media	**Iteración asignada:** 3

Tabla 2.18. Historia de usuario - Gestión de usuarios (Primera parte)

Riesgo: Media	**Estimación:** 16 horas
Programador responsable: José Mosquera	
Descripción: El Administrador debe tener la opción de: modificar y eliminar usuarios del sistema.	
Observaciones:	

Tabla 2.19. Historia de usuario - Gestión de usuarios (Segunda parte)

Historia de Usuario	
Número: HU3B2	**Nombre historia de usuario:** Gestión ítems del catálogo de equipos
Usuario: Administrador	
Prioridad: Media	**Iteración asignada:** 3
Riesgo: Medio	**Estimación:** 16 horas
Programador responsable: Diego Viñamagua	
Descripción El Administrador debe tener la opción de: agregar, modificar y eliminar los ítems del catálogo de equipos del sistema.	
Observaciones: Se creará los métodos necesarios para conectarse a la base de datos, para agregar, modificar o eliminar los ítems del catálogo de equipos del sistema	

Tabla 2.20. Historia de usuario - Gestión ítem del catálogo de equipos

Historia de Usuario	
Número: HU3B3	**Nombre historia de usuario:** Gestión de solicitudes activas
Usuario: Administrador	
Prioridad: Media	**Iteración asignada:** 3
Riesgo: Medio	**Estimación:** 16 horas
Programador responsable: José Mosquera	
Descripción: El Administrador debe tener la opción de: visualizar y archivar solicitudes del sistema.	
Observaciones: Las solicitudes activas se mostrarán tanto en la aplicación Android como en la aplicación principal.	

Tabla 2.21. Historia de usuario - Gestión de solicitudes activas

Historia de Usuario	
Número: HU3B4	**Nombre historia de usuario:** Solicitudes archivadas
Usuario: Administrador	
Prioridad: Media	**Iteración asignada:** 3
Riesgo: Medio	**Estimación:** 16 horas
Programador responsable: Diego Viñamagua	
Descripción: El Administrador debe tener la opción de visualizar solicitudes que hayan sido archivadas.	
Observaciones:	

Tabla 2.22. Historia de usuario -Solicitudes archivadas

Historia de Usuario	
Número: HU3B5	**Nombre historia de usuario:** Gestión de PC en BDD
Usuario: Administrador	
Prioridad: Media	**Iteración asignada:** 3
Riesgo: Medio	**Estimación:** 16 horas
Programador responsable: Diego Viñamagua	
Descripción: El Administrador debe tener la opción de: agregar y eliminar los PC del sistema.	
Observaciones:	

Tabla 2.23. Historia de usuario - Gestión de PC en BDD

Historia de Usuario	
Número: HU4AS1	**Nombre historia de usuario:** Comunicación aplicación Android con aplicación principal
Usuario: Administrador	
Prioridad: Alta	**Iteración asignada:** 4
Riesgo: Alto	**Estimación:** 48 horas
Programador responsable: José Mosquera	
Descripción: Se requiere un mecanismo para establecer la comunicación entre la aplicación Android y la aplicación principal.	
Observaciones:	

Tabla 2.24. Historia de usuario - Comunicación aplicación Android con aplicación principal

Historia de Usuario	
Número: HU4PS2	**Nombre historia de usuario:** Comunicación aplicación profesor con aplicación principal
Usuario: Administrador	
Prioridad: Alta	**Iteración asignada:** 4
Riesgo: Alto	**Estimación:** 48 horas
Programador responsable: José Mosquera	
Descripción: Se requiere un mecanismo para establecer la comunicación entre la aplicación profesor y la aplicación principal.	
Observaciones:	

Tabla 2.25. Historia de usuario - Comunicación aplicación profesor con aplicación principal

Historia de Usuario	
Número: HU4CPS3	**Nombre historia de usuario:** Comunicación aplicación cliente con aplicación profesor o principal
Usuario: Administrador	
Prioridad: Alta	**Iteración asignada:** 4
Riesgo: Alto	**Estimación:** 48 horas
Programador responsable: José Mosquera	
Descripción: Se requiere un mecanismo para establecer la comunicación entre la aplicación cliente y la aplicación profesor o principal.	
Observaciones:	

Tabla 2.26. Historia de usuario - Comunicación aplicación cliente con aplicación profesor o principal

Historia de Usuario	
Número: HU4AS4	**Nombre historia de usuario:** Comunicación por voz
Usuario: Administrador	
Prioridad: Alta	**Iteración asignada:** 4

Tabla 2.27. Historia de usuario - Comunicación por voz (Primera parte)

Riesgo: Alto	**Estimación:** 48 horas
Programador responsable: José Mosquera	
Descripción: Se requiere un mecanismo para establecer una comunicación por voz entre los dispositivos Android	
Observaciones:	

Tabla 2.28. Historia de usuario - Comunicación por voz (Segunda parte)

2.5.2 ESTIMACIÓN DE TIEMPO DE DESARROLLO POR ITERACIONES

El tiempo que se estima para desarrollar los componentes de cada historia de usuario, fue considerado tomando en cuenta el riesgo y la prioridad de cada una y fue establecido en base a la experiencia del equipo de trabajo. La **Tabla 2.29** muestra los tiempos asignados a las diferentes combinaciones de riesgo y prioridad.

La **Tabla 2.30** muestra el tiempo de desarrollo de la primera iteración, la **Tabla 2.31** muestra el tiempo de desarrollo de la segunda iteración, la **Tabla 2.32** muestra el tiempo de desarrollo de la tercera iteración, la **Tabla 2.33** muestra el tiempo de desarrollo de la cuarta iteración.

Riesgo	Prioridad	Tiempo (Horas)
Bajo	Baja	2
Bajo	Media	4
Bajo	Alta	8
Medio	Baja	12
Medio	Media	16
Medio	Alta	20
Alto	Baja	24
Alto	Media	36
Alto	Alta	48

Tabla 2.29. Asignación de tiempo de desarrollo por riesgo en desarrollo

Código	Historia de Usuario	Prioridad	Riesgo	Tiempo (Horas)
HU1B1	Diseño de la base de datos	Alta	Medio	20
HU1S2	Autenticación de usuarios	Alta	Medio	20
HU1S3	Registro de nuevos usuarios	Alta	Medio	20
HU1P4	Recuperación de contraseña	Alta	Medio	20
HU1S5	Registro de nuevos ítems en el catálogo de equipos	Alta	Medio	20
			Total	100

Tabla 2.30. Tiempo de desarrollo de la primera iteración

Código	Historia de Usuario	Prioridad	Riesgo	Tiempo (Horas)
HU2C1	Acción encender	Media	Bajo	4
HU2C2	Acción apagar	Media	Bajo	4
HU2C3	Acción reiniciar	Media	Bajo	4
HU2C4	Acción bloquear	Media	Bajo	4
HU2C5	Acción desbloquear	Media	Bajo	4
HU2C6	Acceso Remoto	Media	Medio	16
HU2C7	Obtener información	Media	Bajo	4
			Total	40

Tabla 2.31. Tiempo de desarrollo de la segunda iteración

Código	Historia de Usuario	Prioridad	Riesgo	Tiempo (Horas)
HU3B1	Gestión de usuarios	Media	Medio	16
HU3B2	Gestión ítem del catálogo de equipos	Media	Medio	16
HU3B3	Gestión de solicitudes activas	Media	Medio	16
HU3B4	Solicitudes archivas	Media	Medio	16
HU3B5	Gestión de PC en BDD	Media	Medio	16
			Total	80

Tabla 2.32. Tiempo de desarrollo de la tercera iteración

Código	Historia de Usuario	Prioridad	Riesgo	Tiempo (Horas)
HU4AS1	Comunicación aplicación Android con aplicación principal	Alta	Alto	48
HU4PS2	Comunicación aplicación profesor con aplicación principal	Alta	Alto	48
HU4CPS 2	Comunicación aplicación cliente con aplicación profesor o principal	Alta	Alto	48
HU4AS2	Comunicación por voz	Alta	Alto	48
			Total	192

Tabla 2.33. Tiempo de desarrollo de la cuarta iteración

2.6 DISEÑO DEL SISTEMA

En esta sección se presentan los diagramas de clases y secuencia del sistema. En base a la información que fue obtenida del análisis de las aplicaciones, se procede a obtener los requerimientos mínimos para el sistema. Para ello se tomará como referencia la metodología de desarrollo eXtreme Programming.

La **Figura 2.17** muestra el diagrama de interacción entre las aplicaciones del sistema.

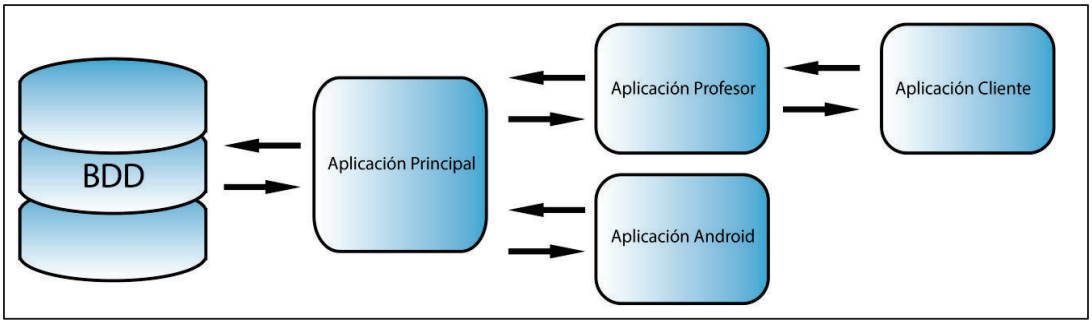


Figura 2.17. Interacción de aplicaciones en el sistema

El sistema emplea un modelo de tres capas, de esta manera se pretende disminuir la complejidad y tener el sistema de forma estructurada. La capa de datos se encarga de obtener y almacenar información. La capa de negocio está encargada de pasar la información de la capa de datos a la capa de presentación. Para finalizar la capa de presentación se encarga de mostrar los resultados al usuario además de interactuar con el mismo.

2.6.1 CAPA DE DATOS

Previo al desarrollo de las aplicaciones es fundamental contar con una estructura de datos. Este puede cambiar a lo largo del desarrollo del sistema debido a los requerimientos que van surgiendo. El sistema gestor de base de datos que se utiliza es MySQL[28]. Se requiere de cinco tablas para el manejo del sistema, las cuales son:

- Tabla equipos: Almacenará los ítems del catálogo de equipos disponibles en el sistema.
- Tabla grupos: Almacenará la información de los diferentes grupos de trabajo.
- Tabla solicitudes: Almacenará las solicitudes que se manejarán en el sistema.
- Tabla computadores: Almacenará la información de los computadores que serán gestionados por el sistema.
- Tabla usuarios: Almacenará los usuarios que se manejan en el sistema.

La **Figura 2.18** muestra el diagrama de la base de datos.

Para realizar la comunicación con la base de datos se usa la clase `ConexionBDD` que contiene los métodos necesarios para establecer y cerrar la conexión con la base de datos además de intercambiar información. La **Tabla 2.34** y la **Tabla 2.35** muestran la descripción de los métodos encargados de esta tarea.

2.6.2 CAPA DE NEGOCIO

Esta capa se encarga de la lógica de: comunicaciones, autenticación, registro de usuarios, recuperación de contraseña, registro de ítems al catálogo de equipos, gestión de PC, reserva de equipos. Estas funciones se implementan de acuerdo a cada aplicación.

Se puede decir que la capa de negocios es la intermediaria entre la capa de datos y la capa de presentación.

[28] MySQL es un sistema de gestión de bases de datos relacional

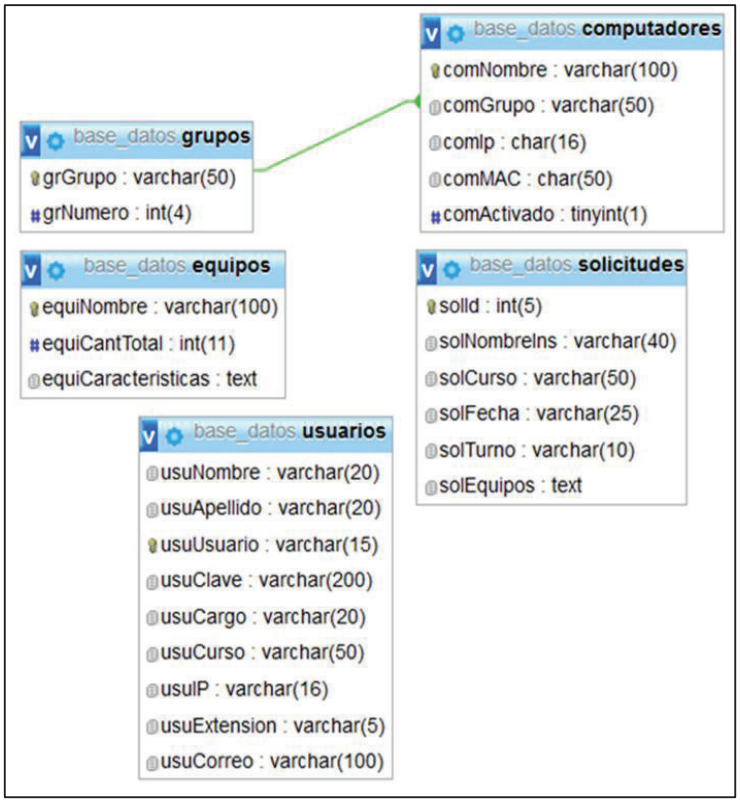


Figura 2.18. Diagrama de la base de datos

METODO	DESCRIPCIÓN
ConexionBDD	
`IniciarConexion`	Establece una conexión a la BDD
`EjecutarComandoAgregarUsuarios`	Agrega un usuario a la BDD
`EjecutarComandoAgregarComputadores`	Agrega un computador a la BDD
`EjecutarComandoAgregaGrupo`	Agrega un grupo a la BDD
`EjecutarComandoAgregarEquipo`	Agrega un equipo a la BDD
`EjecutarComandoAgregarSolicitudActiva`	Agrega una solicitud activa a la BDD

Tabla 2.34. Métodos de la clase ConexionBDD (Primera parte)

METODO	DESCRIPCIÓN
ConexionBDD	
`EjecutarComandoAgregarSolicitudArchivada`	Agrega una solicitud archivada a la BDD
`EjecutarComandoEliminarSolicitud`	Elimina una solicitud activa de la BDD
`EjecutarComandoEliminarEquipo`	Elimina un equipo de la BDD
`EjecutarComandoEliminarUsuario`	Elimina un usuario de la BDD
`EjecutarComandoLeerGrupos`	Recupera todos los grupos de la BDD y los agrega a un `ArrayList`
`EjecutarComandoLeerComputadores`	Recupera todos los computadores de la BDD y los agrega a un `ArrayList`
`EjecutarComandoLeerEquipos`	Recupera todos los equipos de la BDD y los agrega a un `ArrayList`
`EjecutarComandoLeerUsuarios`	Recupera todos los usuarios de la BDD y los agrega a un `ArrayList`
`EjecutarComandoLeerSolicitudesActivas`	Recupera todas las solicitudes activas de la BDD y los agrega a un `ArrayList`
`EjecutarComandoLeerSolicitudesArchivadas`	Recupera todas las solicitudes archivas de la BDD y los agrega a un `ArrayList`
`EjecutarRecuperarUltimaSolicitud`	Recupera de la BDD la última solicitud activa
`CerrarConexion`	Cierra la conexión activa de la BDD

Tabla 2.35. Métodos de la clase ConexionBDD (Segunda parte)

2.6.2.1 Aplicación Cliente

Cumple con la función de recibir mensajes desde la aplicación profesor y la aplicación principal; estos son recibidos mediante sockets y serán interpretados por la aplicación cliente para realizar una acción específica. La **Figura 2.19** muestra el diagrama de clases de la aplicación cliente. La **Tabla 2.36** describe los principales métodos implementados en la aplicación cliente.

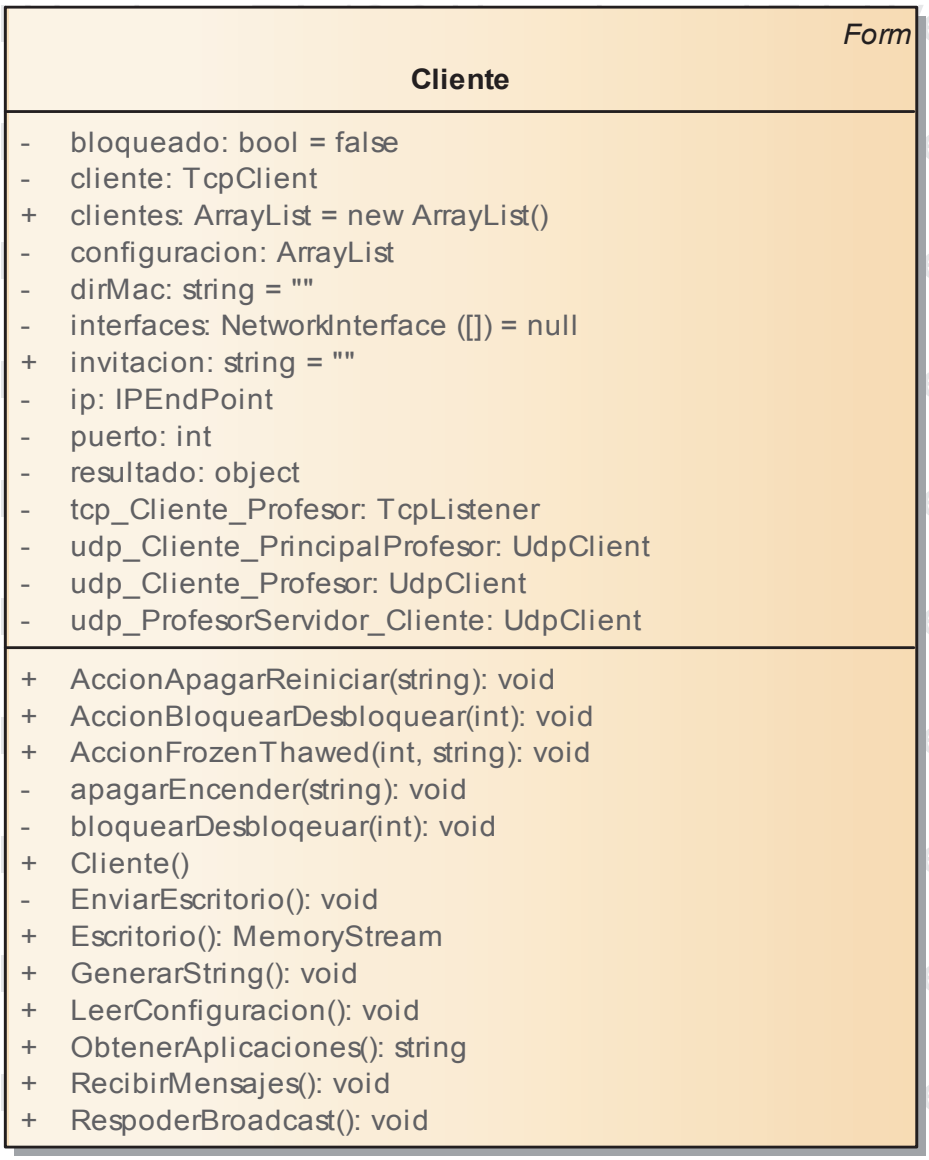


Figura 2.19. Diagrama de clases de la aplicación cliente

METODO	DESCRIPCIÓN
`RecibirMensajes`	Recibe broadcast y mensajes de la aplicación profesor y aplicación principal
`ResponderBroadcast`	Responde al recibir broadcast generado desde la aplicación profesor o la aplicación principal
`ObtenerAplicaciones`	Busca si la aplicación WORD o la aplicación FoxitReader se encuentran abiertas
`EnviarEscritorio`	Envia las capturas de pantalla obtenidas por el método Escritorio
`Escritorio`	Obtiene capturas de pantalla del escritorio del PC
`CambiarTamanoImagen`	Reduce el tamaño de las capturas de pantalla antes de ser enviadas
`AccionApagarReiniciar`	Apaga o reinicia el PC en función del parámetro que se recibe
`AccionBloquearDesbloquear`	Bloquea o desbloquea el PC en función del parámetro que se recibe
`AccionFrozenThawed`	Reinicia en modo *Frozen* o *Thawed* el PC en función del parámetro que se recibe
`GenerarString`	Genera el `string` de invitación que será usado para el acceso remoto

Tabla 2.36. Métodos de la aplicación cliente

2.6.2.2 Aplicación Profesor

Cumple con la función de enviar información a la aplicación principal como: dirección IP, dirección MAC, nombre de equipo, grupo de trabajo correspondiente a los PC que gestiona. Además se pueden generar solicitudes para préstamo de equipos.

Por otro lado también permitirá conectarse a los PC que cuenten con la aplicación cliente y realizar acciones como: apagar, encender, acceder mediante RDP, reiniciar en modo *Frozen*, reiniciar en modo *Thawed*, bloquear y desbloquear.

La **Figura 2.20** muestra el diagrama de clases de la aplicación profesor. La **Tabla 2.37** describen los métodos principales que fueron implementados en la aplicación profesor.

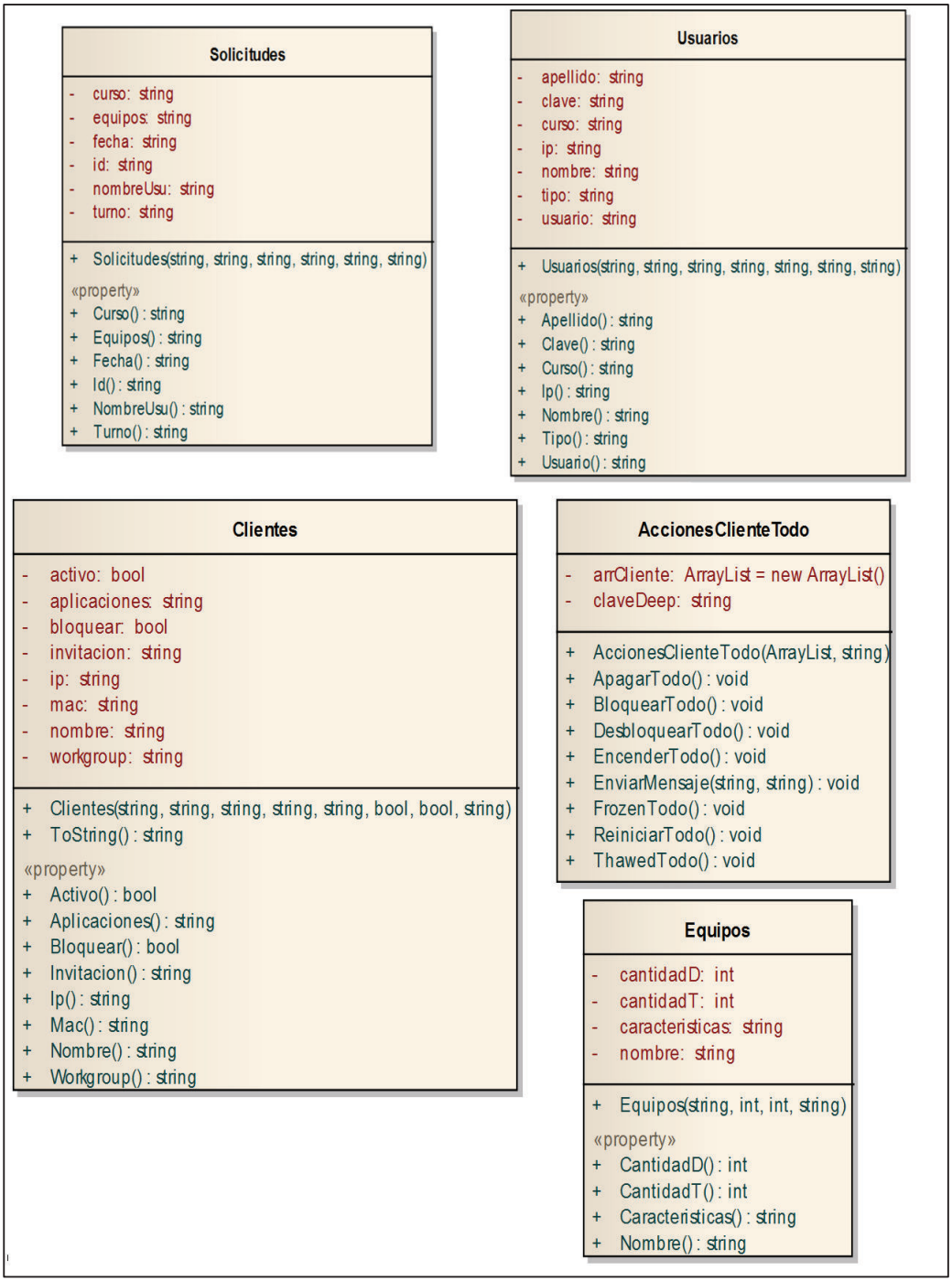


Figura 2.20. Diagrama de clases aplicación profesor

METODO	DESCRIPCIÓN
AccionesClienteTodo	
`ApagarTodo`	Envía el mensaje apagar a la aplicación cliente
`EncenderTodo`	Se genera un paquete `Wake on Lan` hacia el PC
`BloquearTodo`	Envía el mensaje bloquear a la aplicación cliente
AccionesClienteTodo	
`DesbloquearTodo`	Envía el mensaje desbloquear a la aplicación cliente
`ReiniciarTodo`	Envía el mensaje reiniciar a la aplicación cliente
`FrozenTodo`	Envía el mensaje *Frozen* a la aplicación cliente
`ThawedTodo`	Envía el mensaje *Thawed* a la aplicación cliente
Cifrar	
`CifrarMD5`	Cifra la palabra que ingresa como atributo y la devuelve cifrada con MD5[29]

Tabla 2.37. Métodos de la aplicación profesor

2.6.2.3 Aplicación Principal

La aplicación principal cumplirá con la función de enviar y recibir información desde y hacia la aplicación Android y la aplicación profesor. Por lo que permitirá:

- Reservar o solicitar ítems de un catálogo equipos disponibles
- Ingresar al sistema con diferentes perfiles de usuario
- Comunicarse con la aplicación cliente de los PC del sistema y realizar acciones como: apagar, encender, reiniciar en modo *Frozen*, reiniciar en modo *Thawed*, bloquear, desbloquear y acceder mediante RDP
- La conexión y gestión de información de la base de datos

La **Figura 2.21** muestra los diagramas de clases de la aplicación principal.

[29] MD5: es un algoritmo de cifrado de 128 bits ampliamente usado.

Figura 2.21. Diagrama de clases aplicación principal

2.6.2.4 Aplicación Android

La aplicación Android cumplirá con la función de comunicarse a través de un protocolo con la aplicación principal para:

- Visualizar solicitudes activas y cambiar el estado de la solicitud
- Iniciar sesión

- Encender, apagar, reiniciar en modo *Frozen*, reiniciar en modo *Thawed*, bloquear, desbloquear los PC con la aplicación cliente
- Realizar una comunicación por voz entre usuarios que dispongan de la aplicación mediante SIP
- Recibir notificaciones pendientes desde la aplicación principal

La **Figura 2.22** muestra los diagramas de clases de la aplicación Android. La **Tabla 2.38** describen los métodos principales de la aplicación Android.

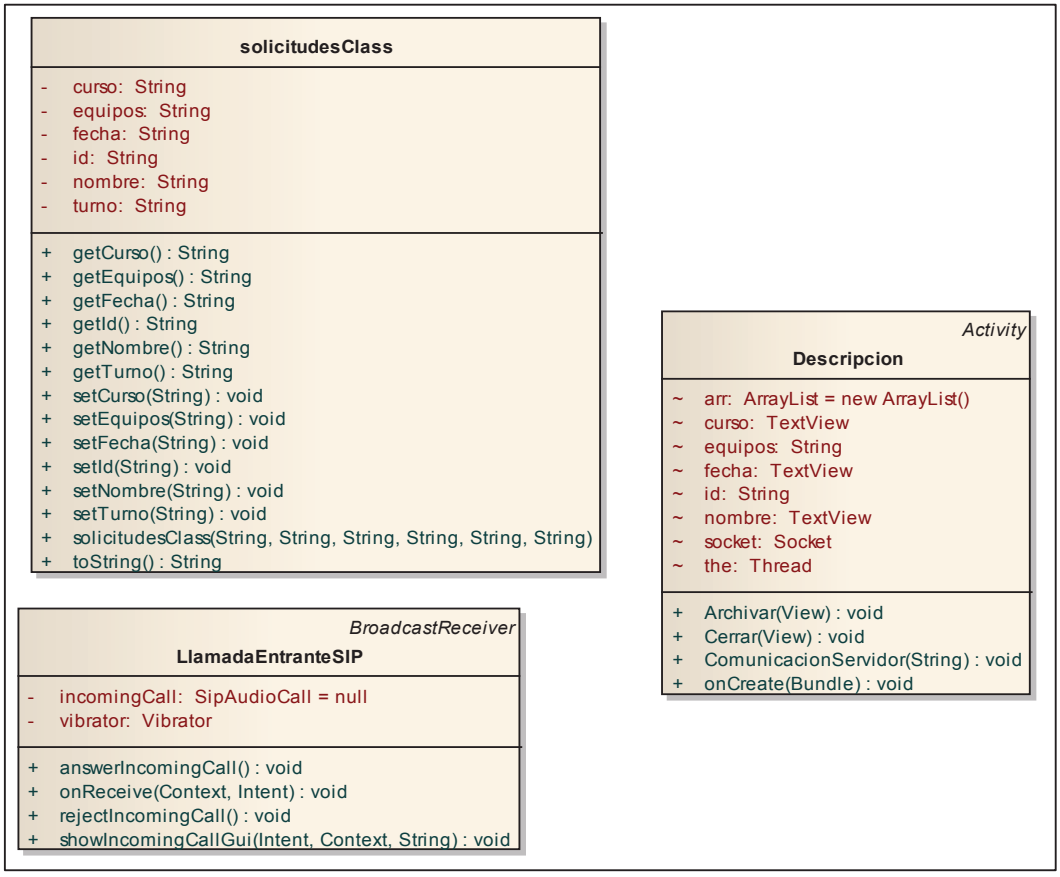


Figura 2.22. Diagrama de clases aplicación Android

2.6.3 CAPA DE PRESENTACIÓN

En esta capa se implementan las interfaces gráficas de la aplicación Android, la aplicación cliente, la aplicación profesor y la aplicación principal.

METODO	DESCRIPCIÓN
Autenticación	
`ActualizarPreferencias`	Guarda el nombre de usuario y contraseña para mantener la sesión iniciada
`ComuniacionServidor`	Envía los distintos mensajes a la aplicación principal
Cifrar	
`ObtenerMD5`	Obtiene el hash MD5
Descripción	
`Archivar`	Elimina la solicitud activa y la almacena en solicitud archivadas las cuales no son mostradas en la aplicación Android
`ComunicacionServidor`	Envía los distintos mensajes a la aplicación principal

Tabla 2.38. Métodos de la aplicación Android

2.6.3.1 Aplicación cliente

La aplicación estará conformada por un solo formulario que se visualizara el momento de bloquear el PC. La **Figura 2.23** muestra el bosquejo de la pantalla de bloqueo de la aplicación cliente.

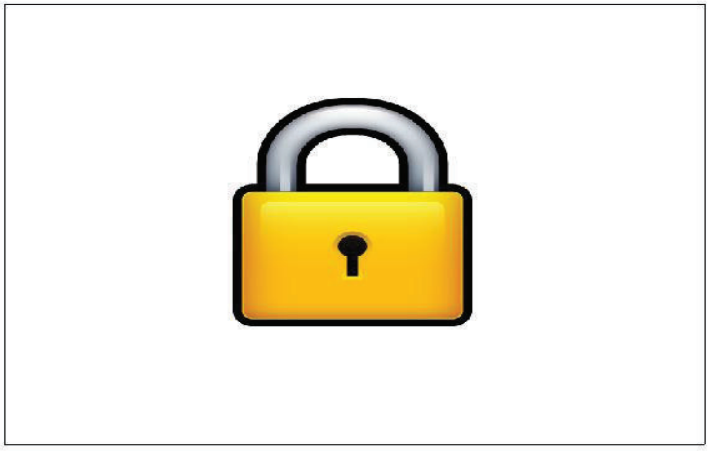


Figura 2.23. Bosquejo de la pantalla de bloqueo de la aplicación cliente

2.6.3.2 Aplicación profesor

La aplicación estará conformada por 8 formularios. A continuación se menciona la función de cada uno de los formularios:

- `frmProfesor`: Será el formulario principal; en este se encontrarán las opciones de gestión para los PC de su misma sala; además, se podrán abrir otros formularios.
- `frmAutenticacion`: Servirá para restringir el acceso a las opciones de gestión: reiniciar en modo *Frozen* y reiniciar en modo *Thawed*; también se restringirá el acceso a la sección de préstamo de equipos.
- `frmPedirEquipos`: Servirá para poder generar una solicitud de equipos.
- `frmEnviarCorreo`: Servirá para poder reestablecer la contraseña.
- `frmCambiarClave`: Servirá para cambiar la clave en caso de ser necesario.
- `frmConfiguracion`: Servirá para configurar la dirección IP de la aplicación principal, puertos y clave de *DeepFreeze*.
- `frmRegistro`: Servirá para registrar un nuevo usuario con perfil de profesor.
- `frmVentanaProf`: Servirá para poder controlar mediante RDP a los PC con la aplicación cliente.

La **Figura 2.24** muestra el bosquejo de la pantalla de principal de la aplicación profesor.

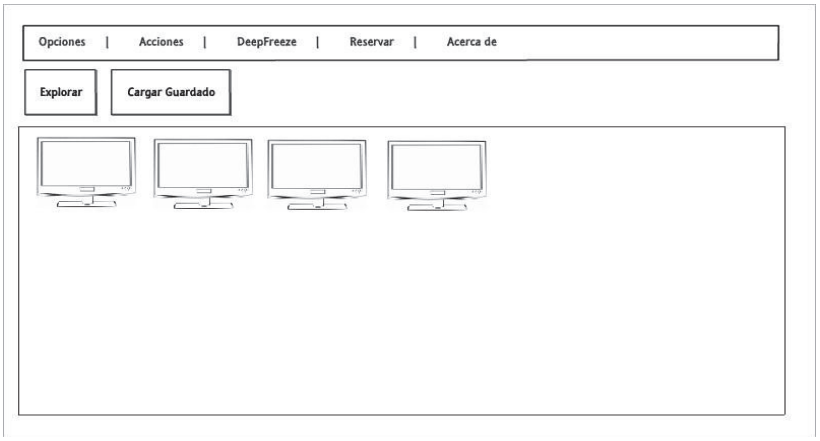


Figura 2.24. Bosquejo de la pantalla de principal de la aplicación profesor

2.6.3.3 Aplicación principal

La aplicación estará conformada por 11 formularios. A continuación se menciona la función de cada uno de los formularios:

- `frmPrincipal`: Será el formulario principal; en este se encontrarán las opciones de gestión para los PC de todas las salas; además, se podrán abrir otros formularios.
- `frmUsuarios`: Servirá para visualizar los usuarios que tiene el sistema.
- `frmAgregarUsuario`: Servirá para agregar usuarios con perfiles de Profesor o Administrador.
- `frmModificarUsuario`: Servirá para modificar algún parámetro de los usuarios.
- `frmEquipos`: Servirá para visualizar los ítems del catálogo de equipos que tiene el sistema.
- `frmAgregarEquipo`: Servirá para agregar items al catálogo de equipos.
- `frmModificarEquipo`: Servirá para modificar algún parámetro de los ítems del catálogo de equipos.
- `frmSolicitudesActivas`: Servirá para visualizar solicitudes activas.
- `frmSolicitudesGuardadas`: Servirá para visualizar solicitudes guardadas.
- `frmVerSolicitud`: Servirá para ver el detalle de cada una de las solicitudes ya sean activas o guardadas; desde este también se podrá imprimir la solicitud.
- `frmVentanaServ`: Servirá para poder controlar mediante RDP a los PC con la aplicación cliente

La **Figura 2.25** muestra el bosquejo de la pantalla de principal de la aplicación principal.

2.6.3.4 Aplicación Android

La aplicación estará conformada por 6 actividades. A continuación se menciona la función de cada una de las actividades:

- `Autenticacion`: En esta actividad se ingresarán los datos de un usuario con perfil de Administrador; si los datos ingresados son correctos se podrá acceder a todas las funciones de la aplicación Android.
- `Adminstrar`: En esta actividad se cargará el listado de PC de las diferentes salas con las que cuenta en Laboratorio de Informática.
- `Solicitudes`: Se mostrarán las solicitudes activas del sistema.
- `Descripcion`: En esta actividad se mostrará la descripción de cada solicitud activa que haya sido seleccionada.
- `Llamada`: En esta actividad se cargará un listado con los usuarios del sistema con perfil Administrar, de los cuales se podrá seleccionar uno y establecer una comunicación por voz.
- `Contestar`: Es la actividad que aparecerá cuando se recibe una llamada.

Además de las actividades descritas se cuenta con un menú de configuración.

La **Figura 2.26** muestra el Bosquejo de la pantalla de principal de la aplicación Android.

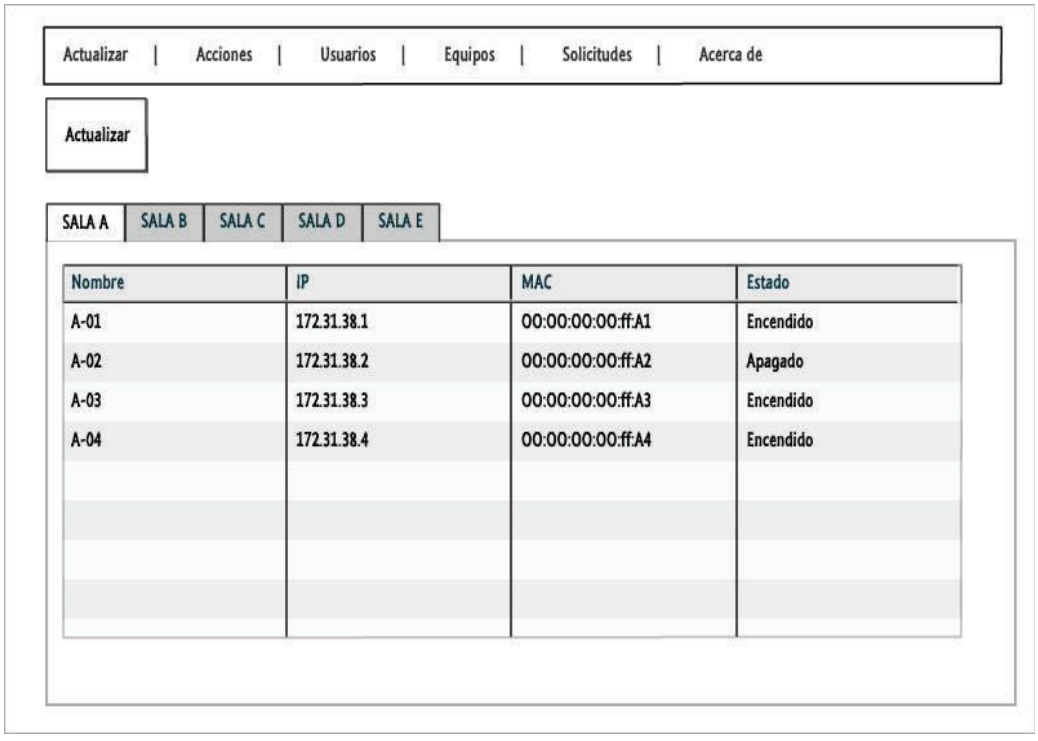


Figura 2.25. Bosquejo de la pantalla de principal de la aplicación principal

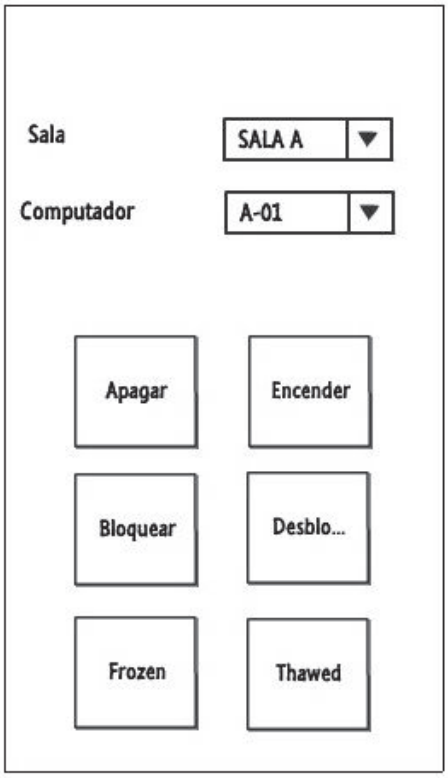


Figura 2.26. Bosquejo de la pantalla de principal de la aplicación Android

2.6.4 DIAGRAMAS DE SECUENCIA

Los diagramas de secuencia ayudan a entender de mejor manera las interacciones entre las aplicaciones del sistema. En esta sección se muestran los diagramas de secuencia de las acciones principales del sistema.

La **Figura 2.27** muestra el diagrama de secuencia para petición de solicitudes en Android, donde se observa que en la aplicación Android se requiere autenticación por parte del Administrador para visualizar las solicitudes.

La **Figura 2.28** muestra el diagrama de secuencia entre usuario Administrador, aplicación Android, aplicación principal y aplicación cliente**,** donde se observa que en la aplicación Android se requiere autenticación por parte del Administrador para recibir los grupos y los PC sobre los cuales se puede realizar acciones.

La **Figura 2.29** muestra el diagrama de secuencia para la gestión de usuarios, equipos y solicitudes desde la aplicación principal, donde se puede observar que el administrador puede administrar usuarios, equipos y archivar solicitudes.

La **Figura 2.30** muestra el diagrama de secuencia entre usuario profesor, aplicación profesor, aplicación principal y aplicación cliente, donde se puede observar que el profesor puede realizar acciones sobre los PC clientes, además puede registrar un usuario y autenticarse para generar una solicitud.

Finalmente la **Figura 2.31** muestra el diagrama de secuencia de *broadcast*, donde se puede observar que tanto la aplicación principal como la aplicación profesor envían mensajes de *broadcast* a la aplicación cliente para verificar de esta.

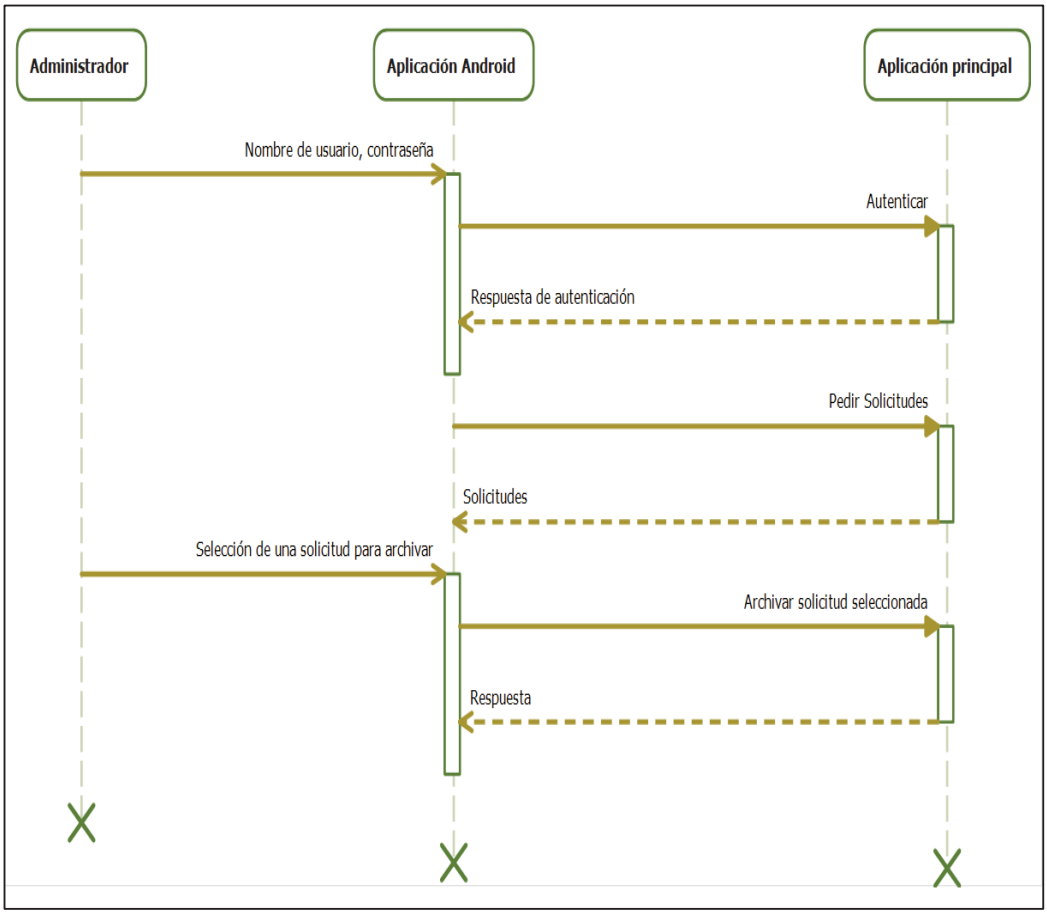


Figura 2.27. Diagrama de secuencia para petición de solicitudes en Android

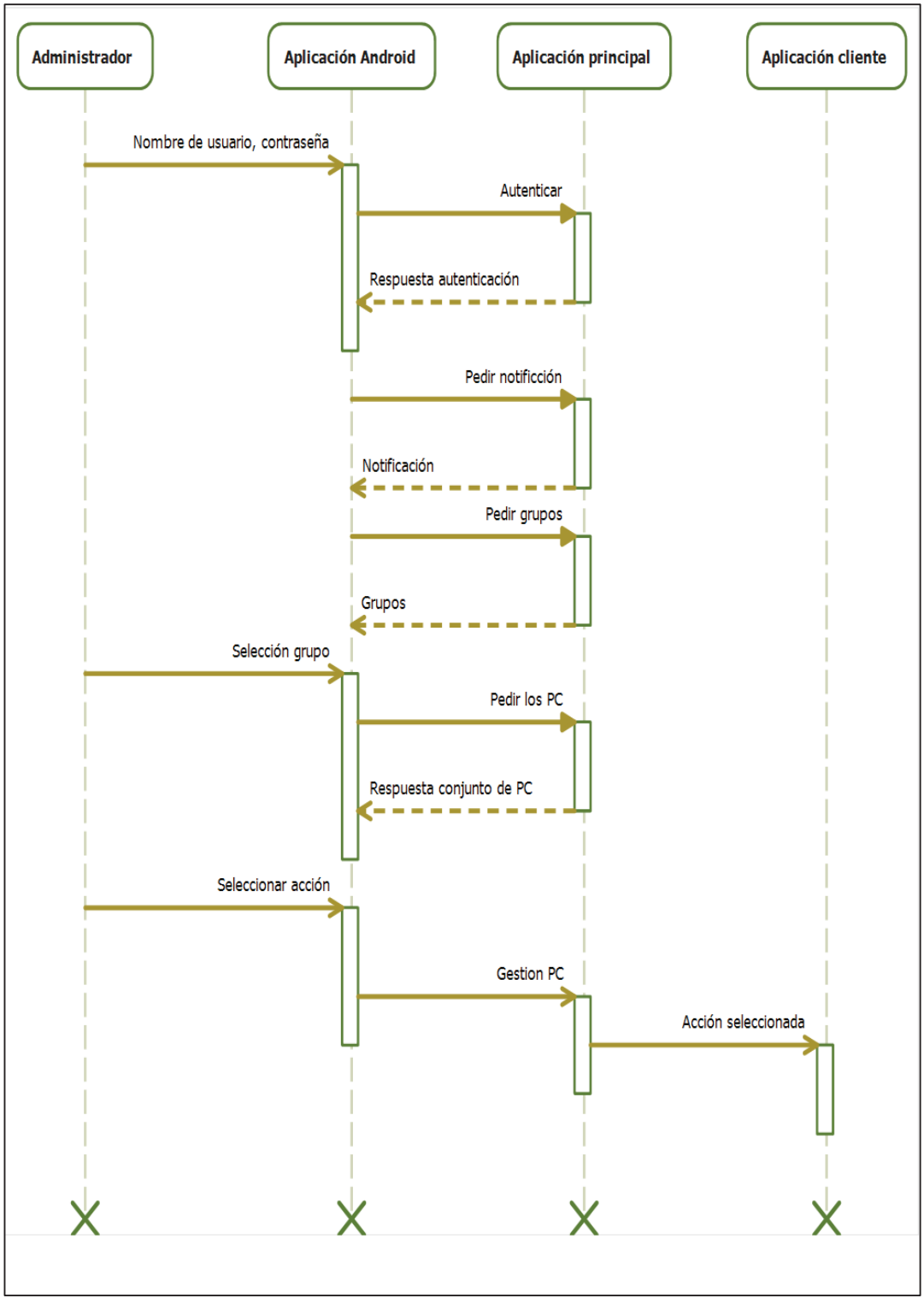


Figura 2.28. Diagrama de secuencia entre usuario Administrador, aplicación Android, aplicación principal y aplicación cliente

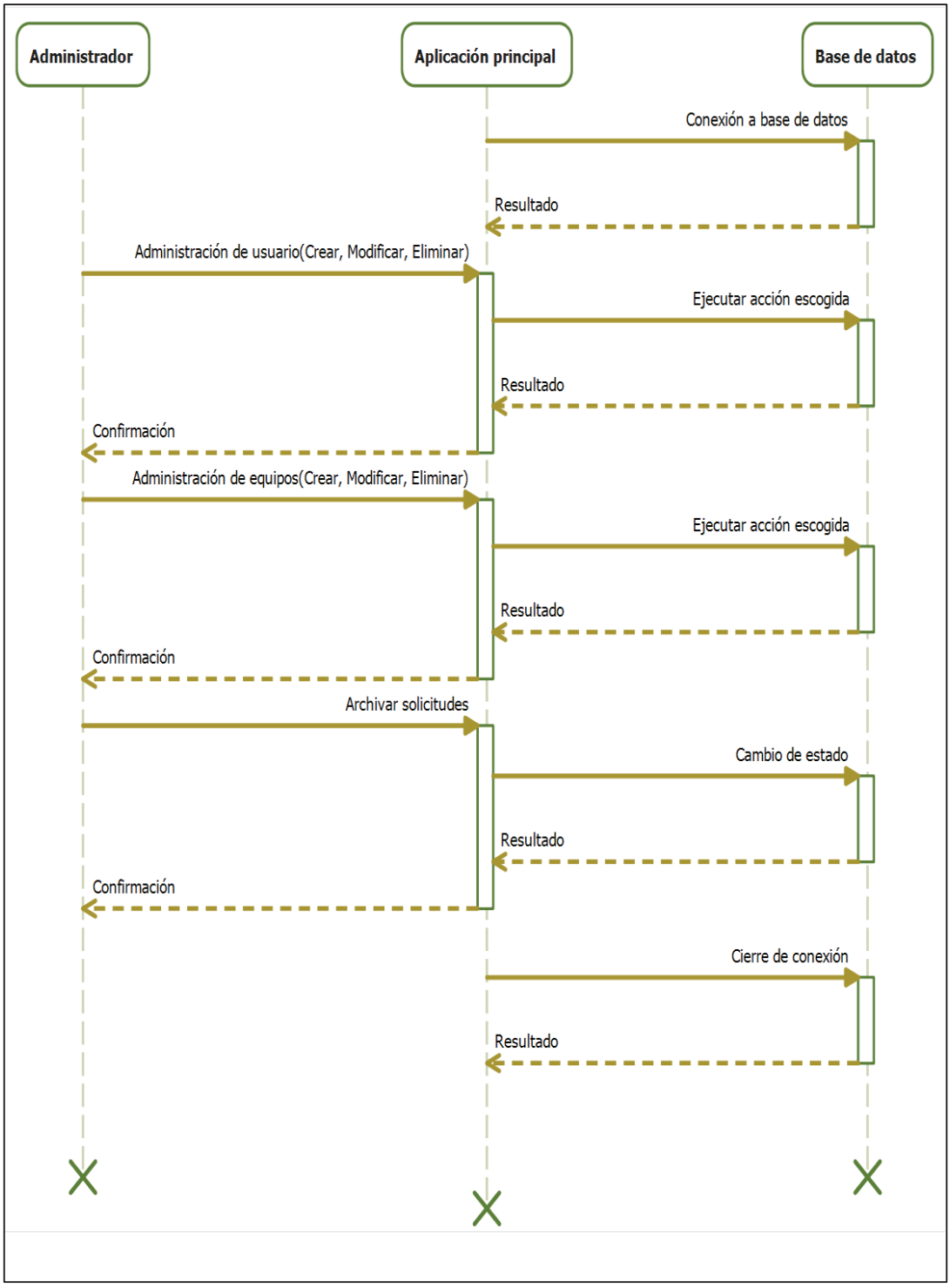


Figura 2.29. Diagrama de secuencia para la gestión de usuarios, equipos y solicitudes desde aplicación principal.

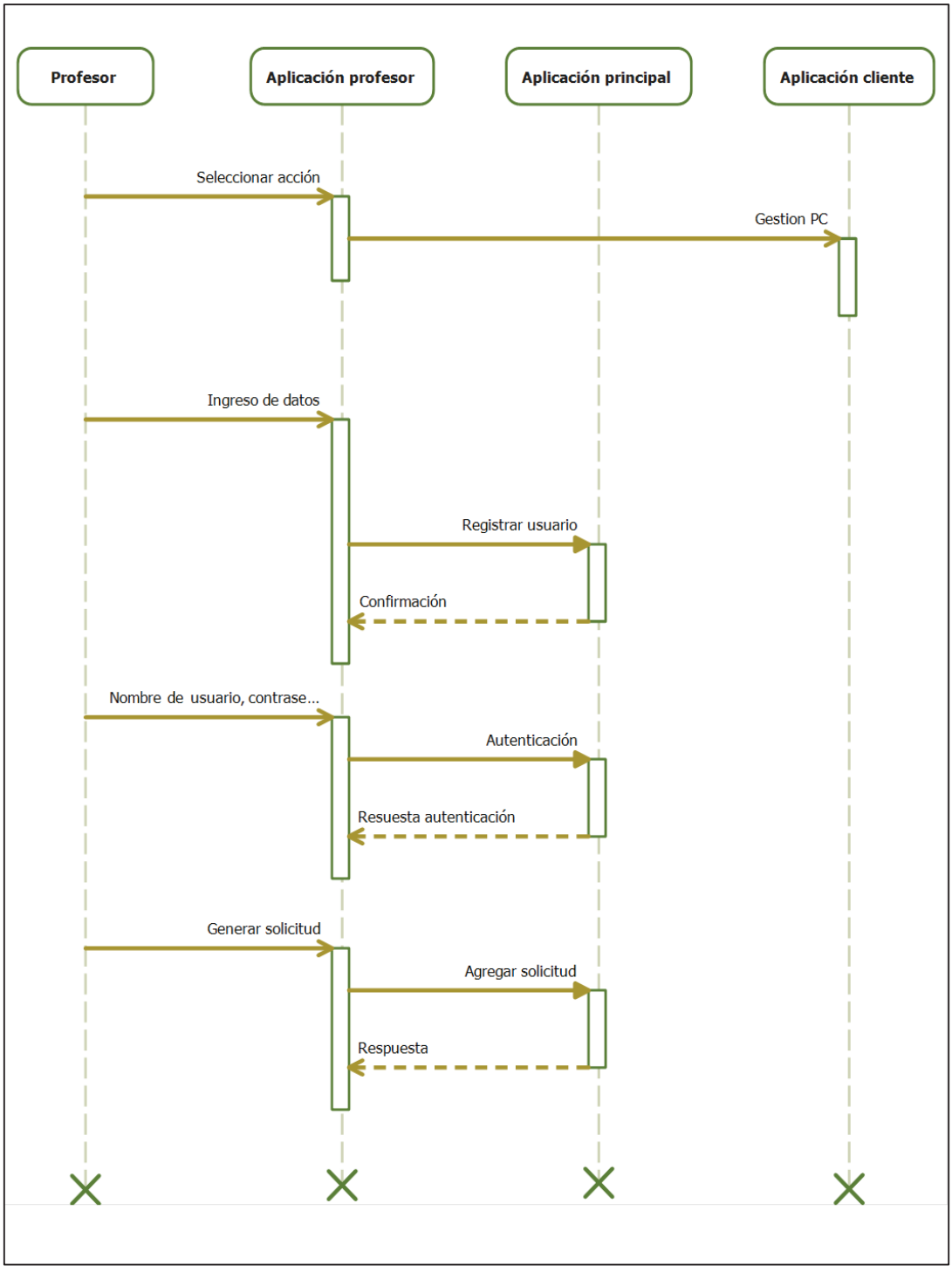


Figura 2.30. Diagrama de secuencia entre usuario profesor, aplicación profesor, aplicación principal y aplicación cliente

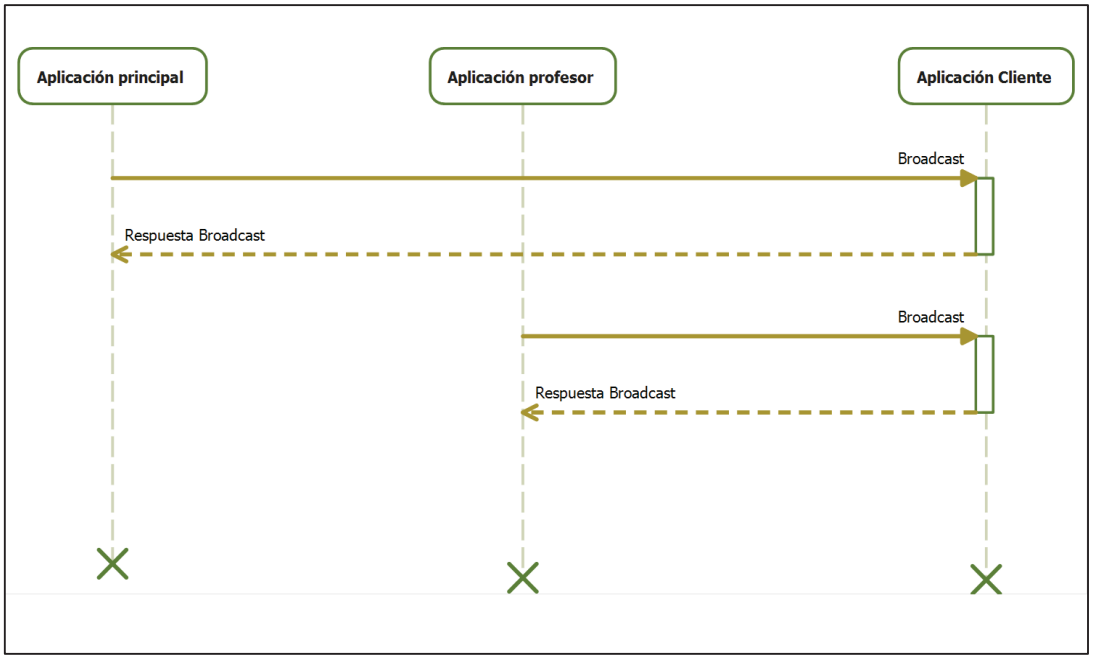


Figura 2.31. Diagrama de secuencia de Broadcast

CAPÍTULO 3 IMPLEMENTACIÓN DEL SISTEMA

3.1 INTRODUCCIÓN

En este capítulo se describe la implementación de todas las aplicaciones que forman parte del sistema, así como los complementos necesarios para su correcto funcionamiento.

El sistema fue implementado con un equipo de trabajo conformado por dos personas, por lo que fue necesario establecer ciertas normas para que el código sea fácil de manipular por el equipo de trabajo.

3.2 NORMAS PARA LA IMPLEMENTACIÓN

3.2.1 BASE DE DATOS

3.2.1.1 Tablas

Los nombres de las tablas serán escritos con minúsculas e indicarán con un nombre descriptivo la función que cumplirá cada tabla en el sistema. Por ejemplo, la tabla que contiene la información de los equipos se llamará: `equipos`.

3.2.1.2 Atributos de las tablas

El nombre de cada atributo empezará con las primeras letras de la tabla correspondiente y como es nombre compuesto se usará la notación *camelCase*[30]. Por ejemplo, en la tabla `equipos` el atributo que almacenará el nombre del equipo será: `equiNombre`.

[30] *camelCase*: Es un estilo de escritura que se aplica a frases o palabras compuestas. La primera letra de cada palabra debe ser mayúscula a excepción de la primera palabra en la cual va con minúscula.

3.2.2 APLICACIÓN WINDOWS Y ANDROID

3.2.2.1 Clases

Los nombres de las clases que se utilizan deben ser descriptivos de acuerdo a la función que van realizar. Para estas se usa la notación *PascalCase*[31]. Por ejemplo, la clase que tiene los métodos necesarios para el uso de la base de datos se llama `ConexionBDD`.

3.2.2.2 Métodos

Al igual que para las clases se usará la notación *PascalCas*e, teniendo en cuenta que el nombre debe ser descriptivo para que indique de manera clara el funcionamiento de cada método. Por ejemplo, en la clase `ConexionBDD` se tiene el método `IniciarConexion` el cual inicializa el `string` de conexión e inicia la conexión con la base de datos.

3.2.2.3 Variables

Para el caso de las variables se usa la notación *camelCase*; su nombre debe ser claro y descriptivo para que sean fáciles de manipular dentro del código. Por ejemplo, en la clase `ConexionBDD` se tiene la variable `cadenaConexion`, del tipo `string`, que será usada para almacenar la cadena de conexión para la BDD.

3.3 IMPLEMENTACIÓN DE LA BASE DATOS

La base de datos se implementó en MySQL y para su administración se hizo uso de *phpMyadmin*[32]. Las tablas necesarias para el funcionamiento del sistema se mencionan a continuación.

---

[31] *PascalCase*: Es un estilo de escritura que se aplica a frases o palabras compuestas. La primera letra de cada palabra debe ser mayúscula.

[32] *phpMyadmin*: Es una herramienta escrita en PHP con la intención de manejar la administración de MySQL a través de páginas web.

3.3.1 Tabla computadores

Almacenará la información de los computadores que serán gestionados por el sistema. El **Código 3.1** muestra las líneas de código necesarias para la creación de la tabla `usuario`. Como se puede observar esta tabla está formada por cuatro atributos:

- `comNombre`: Posee un tipo de dato `varchar` y almacenará el nombre del computador.
- `comGrupo`: Posee un tipo de dato `varchar` y almacenará el grupo del computador.
- `comIp`: Posee un tipo de dato `char` debido a que se conoce el tamaño del dato; almacenará la dirección IP del computador.
- `comMAC`: Posee un tipo de dato `char` debido a que se conoce el tamaño del dato; almacenará la MAC del computador.

```
CREATE TABLE IF NOT EXISTS `computadores` (
  `comNombre` varchar(100) COLLATE utf8_unicode_ci NOT NULL DEFAULT '',
  `comGrupo` varchar(50) COLLATE utf8_unicode_ci DEFAULT NULL,
  `comIp` char(16) COLLATE utf8_unicode_ci DEFAULT NULL,
  `comMAC` char(50) COLLATE utf8_unicode_ci DEFAULT NULL
)
```

Código 3.1. Creación de tabla computadores

3.3.2 Tabla equipos

Almacenará los ítems del catálogo de equipos disponibles en el sistema. El **Código 3.2** muestra las líneas de código necesarias para la creación de la tabla `equipos`.

Como se puede observar esta tabla está formada por tres atributos:

- `equiNombre`: Posee un tipo de dato `varchar` y almacenará el nombre del equipo.
- `equiCantTotal`: Posee un tipo de dato `int` y almacenará la cantidad disponible del equipo.

- equiCaracteristicas: Posee un tipo de dato text debido a que se almacenará gran cantidad de texto; almacenará las características del equipo.

```
CREATE TABLE IF NOT EXISTS `equipos` (
  `equiNombre` varchar(100) NOT NULL,
  `equiCantTotal` int(11) DEFAULT NULL,
  `equiCaracteristicas` text
)
```

Código 3.2. Creación de tabla equipos

3.3.3 Tabla grupos

Almacenará la información de los diferentes grupos de trabajo que están presentes en el sistema. El **Código 3.3** muestra las líneas de código necesarias para la creación de la tabla grupos. Como se puede observar esta tabla está formada por dos atributos

- grGrupo: Posee un tipo de dato varchar y almacenará el nombre del grupo.
- grNumero: Posee un tipo de dato int y almacenará la cantidad de PC que posee este grupo.

```
CREATE TABLE IF NOT EXISTS `grupos` (
  `grGrupo` varchar(50) COLLATE utf8_unicode_ci NOT NULL DEFAULT '',
  `grNumero` int(4) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

Código 3.3. Creación de tabla grupos

3.3.4 Tabla solicitudes

Almacena las solicitudes activas que se manejan en el sistema. El **Código 3.4** muestra las líneas de código necesarias para la creación de la tabla solicitudes. Como se puede observar esta tabla está formada por seis atributos

- `solId`: Posee un tipo de dato `int` y almacenará una identificación única para cada solicitud.
- `solNombreIns`: Posee un tipo de dato `varchar` y almacenará el nombre del profesor que generó la solicitud.
- `solCurso`: Posee un tipo de dato `varchar` y almacenará el curso para el cual fue generado la solicitud.
- `solFecha`: Posee un tipo de dato `varchar` y almacenará la fecha para la que se desea la solicitud.
- `solTurno`: Posee un tipo de dato `varchar` y almacenará el turno para el que se desea la solicitud.
- `solEquipos`: Posee un tipo de dato `text` debido a que se almacenará gran cantidad de texto; almacenará los equipos que se requieren en la solicitud.

```
CREATE TABLE IF NOT EXISTS `solicitudes` (
  `solId` int(5) NOT NULL,
  `solNombreIns` varchar(40) NOT NULL,
  `solCurso` varchar(50) NOT NULL,
  `solFecha` varchar(25) NOT NULL,
  `solTurno` varchar(10) NOT NULL,
  `solEquipos` text NOT NULL
) ENGINE=InnoDB AUTO_INCREMENT=47 DEFAULT CHARSET=latin1;
```

Código 3.4. Creación de tabla solicitudes

3.3.5 Tabla usuarios

Almacenará los usuarios que se manejan en el sistema. El **Código 3.5** muestra las líneas de código necesarias para la creación de la tabla `usuarios`. Como se puede observar esta tabla está formada por nueve atributos

- `usuNombre`: Posee un tipo de dato `varchar` y almacenará el nombre del usuario.
- `usuApellido`: Posee un tipo de dato `varchar` y almacenará el apellido del usuario.

- usuUsuario: Posee un tipo de dato varchar y almacenará el nombre de usuario.
- usuClave: Posee un tipo de dato varchar y almacenará la clave cifrada.
- usuCargo: Posee un tipo de dato varchar y almacenará el tipo de usuario.
- usuCurso: Posee un tipo de dato varchar y almacenará el curso del profesor; en el caso de administrador el valor por defecto será “Admin”
- usuIP: Posee un tipo de dato char debido a que se conoce el tamaño del dato, almacenará la IP del dispositivo Android desde el cual se conectó el usuario por última vez.
- usuExtension: Posee un tipo de dato varchar y almacenará la extensión con la que se registra un usuario con perfil administrador.
- usuCorreo: Posee un tipo de dato varchar y almacenará el correo que registra el usuario.

```
CREATE TABLE IF NOT EXISTS `usuarios` (
  `usuNombre` varchar(20) COLLATE utf8_unicode_ci NOT NULL,
  `usuApellido` varchar(20) COLLATE utf8_unicode_ci NOT NULL,
  `usuUsuario` varchar(15) COLLATE utf8_unicode_ci NOT NULL DEFAULT '',
  `usuClave` varchar(200) COLLATE utf8_unicode_ci DEFAULT NULL,
  `usuCargo` varchar(20) COLLATE utf8_unicode_ci NOT NULL,
  `usuCurso` varchar(50) COLLATE utf8_unicode_ci NOT NULL,
  `usuIP` varchar(16) COLLATE utf8_unicode_ci NOT NULL,
  `usuExtension` varchar(5) COLLATE utf8_unicode_ci NOT NULL,
  `usuCorreo` varchar(100) COLLATE utf8_unicode_ci NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

Código 3.5 Creación de tabla usuarios

3.4 IMPLEMENTACIÓN DE LAS APLICACIONES

3.4.1 IMPLEMENTACIÓN DE LA APLICACIÓN PRINCIPAL

La aplicación principal es la encargada de controlar todo el sistema. Fue implementada en Visual Studio 2010 usando el lenguaje de programación C#. Esta aplicación será usada por los administradores; mediante una interfaz gráfica que tiene varias opciones para la gestión del sistema. La **Figura 3.1** muestra una captura de pantalla de la aplicación principal.

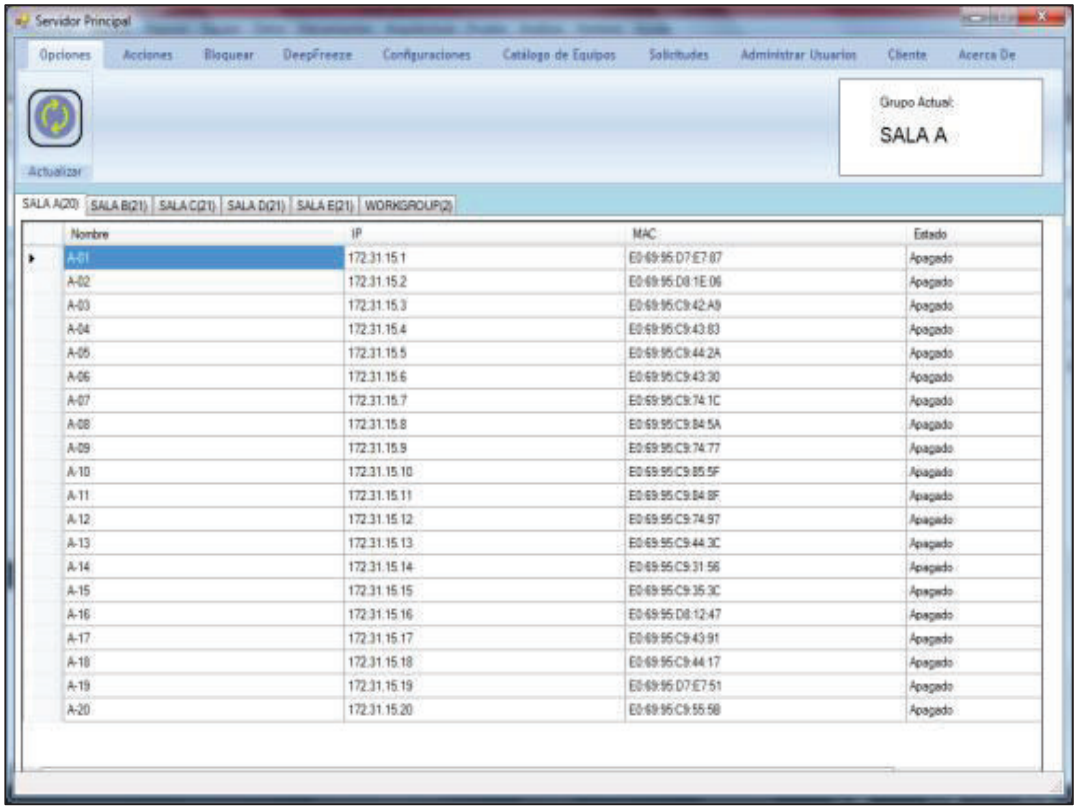


Figura 3.1. Aplicación principal

A continuación se detallan las secciones más importantes del código de la aplicación principal; el código completo se lo puede encontrar en el **ANEXO B.1**

3.4.1.1 Comunicación con la base de datos

Para la comunicación de la aplicación principal con la base datos es necesario agregar la referencia `MySql.Data`, esta no es propia de Visual Studio por lo que el complemento necesario debe ser descargado de [30].

Como se mencionó en el capítulo anterior, se usa la clase `ConexionBDD` para obtener la información de la base de datos.

El **Código 3.6** muestra las líneas de código que permite agregar nuevos usuarios o reemplazarlos si ya existen en el sistema.

```
//Metodo para insertar nuevos registros en la tabla Usuarios en la base de datos
public void EjecutarComandoAgregarUsuario(string nombre, string apellido,
    string usuario, string clave, string cargo, string curso, string extension, string correo)
{
    comando.Parameters.Clear();
    comando.CommandText = "REPLACE INTO usuarios (usuNombre,usuApellido,usuUsuario,usuClave,usuCargo,"+
    "usuCurso,usuExtension,usuCorreo) values (@nombre,@apellido,@usu,@clave,@cargo,@curso,@extension,@correo)";
    comando.Parameters.AddWithValue("@nombre", nombre);
    comando.Parameters.AddWithValue("@apellido", apellido);
    comando.Parameters.AddWithValue("@usu", usuario);
    comando.Parameters.AddWithValue("@clave", clave);
    comando.Parameters.AddWithValue("@cargo", cargo);
    comando.Parameters.AddWithValue("@curso", curso);
    comando.Parameters.AddWithValue("@extension", extension);
    comando.Parameters.AddWithValue("@correo", correo);
    comando.Connection = conexion;
    comando.ExecuteNonQuery();

}
```

Código 3.6. Método `EjecutarComandoAgregarUsuario` en la clase `conexionBDD`

El **Código 3.7** muestra el método para eliminar usuarios del sistema.

```
//Metodo para eliminar registros en la tabla Usuarios en la base de datos
public void EjecutarComandoEliminarUsuario(string nombredeUsuario)
{
    comando.Parameters.Clear();
    comando.CommandText = "DELETE FROM usuarios WHERE  usuUsuario=@usuNombre";
    comando.Parameters.AddWithValue("@usuNombre", nombredeUsuario);
    comando.Connection = conexion;
    comando.ExecuteNonQuery();
}
```

Código 3.7. Método `EjecutarComandoEliminarUsuario` en la clase `conexionBDD`

El **Código 3.8** muestra el método necesario para leer los usuarios que existen en el sistema.

Para las tablas equipos y solicitudes se tienen métodos similares a los mencionados anteriormente.

3.4.1.2 Función para inicio de sesión

Como parte de la funcionalidad para el inicio de sesión se tiene el método `VerificarUsuario` el cual comprueba que un usuario esté registrado en el sistema. Para esto se conecta con la base de datos mediante la clase

ConexiónBDD con el objeto conBDD; el método devuelve el valor “no” en caso de no haber encontrado el usuario. Si, el usuario tiene un perfil de administrador se devuelve “si@numero_extension” y si el usuario tiene un perfil profesor se devuelve “si@”. El segundo parámetro sirve para configurar automáticamente en la aplicación Android la extensión SIP, la cual se usará posteriormente para la comunicación por voz. En el **Código 3.9** se muestra el método VerificarUsuario.

```
//Metodo que realiza una consulta a la base de datos para recuperar información de la tabla Usuarios
public ArrayList EjecutarComandoLeerUsuarios()
{
    arrUsuarios = new ArrayList();
    MySqlDataReader LectorBDD;
    comando.CommandText = "Select * from usuarios";
    comando.Connection = conexion;
    LectorBDD = comando.ExecuteReader();
    while (LectorBDD.Read())
    {
        Usuarios usuario= new Usuarios(LectorBDD["usuUsuario"].ToString(),
            LectorBDD["usuClave"].ToString(), LectorBDD["usuNombre"].ToString(),
            LectorBDD["usuApellido"].ToString(), LectorBDD["usuCurso"].ToString(),
            LectorBDD["usuCargo"].ToString(), LectorBDD["usuIP"].ToString(),
            LectorBDD["usuExtension"].ToString(), LectorBDD["usuCorreo"].ToString());
        arrUsuarios.Add(usuari);

    }

    return arrUsuarios;
}
```

Código 3.8. Método `EjecutarComandoLeerUsuarios` en la clase `conexionBDD`

```
//Para verificar que el usuario se encuentre registrado o no en la base de datos
//este metodo recibe como parametro el usuario y la contraseña
public string VerificarUsuario(string usuario,string contraseña)
{
    conBDD.IniciarConexion();
    arrUsuarios = conBDD.EjecutarComandoLeerUsuarios();
    conBDD.CerrarConexion();
    string encontrado = "no";
    foreach (Usuarios usu in arrUsuarios)
    {
        if (usu.Usuario == usuario && usu.Clave == contraseña)
        {
            if (usu.Tipo == "Administrador")
            {
                encontrado = "si" + "@" + usu.Extension;
                break;
            }

        }
    }
    return encontrado;
}
```

Código 3.9. Método `VerificarUsuario`

3.4.1.3 Gestión de PC

Para gestionar los PC con la aplicación cliente se usan dos métodos; el primero determina si la orden vino directamente desde la aplicación principal o si la aplicación Android es la que requiere realizar la acción, si es la primera opción el usuario deberá confirmar la orden antes de ser realizada. El **Código 3.10** muestra el método `HacerAccionTodo`.

```
// Comprueba si la orden viene desde la aplicación principal o desde la aplicación Android
public void HacerAccionTodo(ArrayList arrayActual ,string accion,bool computadora)
{
    if (computadora == true)
    {
        if (MessageBox.Show("Estas seguro de " + accion + " " + arrayActual.Count.ToString() +
            " computadores?", "Accion", MessageBoxButtons.YesNo,MessageBoxIcon.Question) == DialogResult.Yes)
        {
            HacerAccionFinal(arrayActual, accion);
        }
    }
    else
    {
        HacerAccionFinal(arrayActual, accion);
    }
}
```

Código 3.10. Método `HacerAccionTodo`

Posteriormente, las dos opciones ejecutan el método `HacerAccionFinal` el cual depende directamente de la variable `accion`, esta variable puede tomar los valores: “Apagar”, “Encender”, “Reiniciar”, “Reiniciar Frozen”, “Reiniciar Thawed, Bloquear”, “Desbloquear”. Dependiendo de la acción El **Código 3.11** muestra un fragmento del código del método `HacerAccionFinal`.

3.4.1.4 Recepción de mensajes desde la aplicación profesor

Parte importante del sistema son las comunicaciones que deben existir entre las distintas aplicaciones de este. Para la comunicación entre la aplicación principal y aplicación profesor se usa un puerto UDP que puede ser configurado; para la implementación se usó el puerto 8083; al usar un protocolo no orientado a conexión y sin confirmación la interacción entre aplicaciones se realiza de una forma más rápida. También los PC que ejecutan estas aplicaciones están conectados usando un medio alámbrico por lo que las pérdidas que puedan

ocurrir son mínimas. Principalmente el método `TareaMensajesProfesor` espera a recibir mensajes en el puerto anteriormente mencionado; los mensajes posibles son:

- `nuevoGrupo`: Para guardar un nuevo grupo de trabajo.
- `nuevoComputador`: Para guardar un nuevo PC.
- `nuevaSolicitud`: Para informar a la aplicación principal que se ha generado una nueva solicitud.
- `preguntarUsuario`: Para preguntar si un usuario específico está registrado en el sistema.
- `agregarUsuario`: Para agregar usuarios que hayan sido registrados desde la aplicación profesor.
- `enviarCorreo`: Para enviar un correo electrónico en caso de requerir una contraseña nueva.
- `cambiarClave`: Para cambiar la contraseña de un usuario específico.

El **Código 3.12** muestra una parte del método `TareaMensajesProfesor`.

```
// Realiza la accion correspondiente del método HacerAccionTodo
public void HacerAccionFinal(ArrayList arrayActual ,string accion)
{
    LeerConfiguracion();
    switch (accion) {
        case "Apagar":

            foreach (Computadores comp in arrayActual)
            {
                AccionesCliente acc = new AccionesCliente(comp.Ip,comp.Mac,claveDeepFrez);
                acc.apagar();
            }
        break;
         case "Encender":

            foreach (Computadores comp in arrayActual)
            {
                AccionesCliente acc = new AccionesCliente(comp.Ip, comp.Mac, claveDeepFrez);
                acc.encender();
            }
        break;
         case "Reiniciar":

            foreach (Computadores comp in arrayActual)
            {
                AccionesCliente acc = new AccionesCliente(comp.Ip, comp.Mac, claveDeepFrez);
                acc.reiniciar();
            }
        break;
                ...
```

Código 3.11. Método `HacerAccionFinal`

```
public void TareaMensajesProfesor()
{
    ///// Recibe mensajes de computador Profesor
    UdpClient udp = new UdpClient(8083);
    IPEndPoint ip = new IPEndPoint(IPAddress.Any, 8083);
    while (true)
    {
        byte[] bytes = udp.Receive(ref ip);
        string mensaje = Encoding.UTF8.GetString(bytes, 0, bytes.Length);
        string[] mensajeDividido = mensaje.Split('&');
        string mensaje1 = mensajeDividido[0];
        string mensaje2 = mensajeDividido[1];
        bool encontrado = false;
        switch (mensaje1)
        {
            case "nuevoGrupo":
                    conBDD.IniciarConexion();
                    conBDD.EjecutarComandoAgregaGrupo(mensaje2, 0, "");
                    conBDD.CerrarConexion();
                    break;

            case "nuevoComputador":
                    string[] words = mensaje2.Split('@');
                    conBDD.IniciarConexion();
                    conBDD.EjecutarComandoAgregarComputadores(words[0], words[1], words[2], words[3]);
                    conBDD.CerrarConexion();
                    break;
                        .
                        .
                        .
```

Código 3.12. Método `TareaMensajesProfesor`

3.4.1.5 Recepción de mensajes desde la aplicación Android

La comunicación entre la aplicación Android y aplicación principal se realiza usando un puerto TCP que puede ser configurado; para la implementación se usó el puerto 20000; debido a que los dispositivos con Android se conectan a la red usando un medio inalámbrico, es necesaria una comunicación confiable para el intercambio de información entre las aplicaciones. Además, debido a que es una aplicación multicliente es necesario un método que atienda a las aplicaciones Android y las ubique posteriormente en su propio puerto y en su propio hilo. El **Código 3.13** muestra el método `TareaClientesAndroid`.

Posteriormente se ejecuta el método `TareaMensajeAndroid` el cual identifica un mensaje compuesto, que desde este momento se lo llamará trama, enviado por la aplicación Android. Dicha trama está compuesta por 3 o 4 mensajes separados con el signo “&”. La **Figura 3.2** muestra el formato de la trama Android.

```
//recibe mensajes desde dispositivo android
public void TareaClientesAndroid()
{
    //Asignamos al puerto 8080 para escuchar por clientes 1
    socketEscuchaAndroid = new TcpListener(IPAddress.Any, puertoAndroid);
    socketEscuchaAndroid.Start();
    while (true)//lazo infinito que aceptara peticiones
    {
        try
        {
            socketClienteAndroid = socketEscuchaAndroid.AcceptSocket();
            if (socketClienteAndroid.Connected)
            {
                clientesAndroid.Add(socketClienteAndroid);
                //si se encuentra una peticion se crear un nuevo hilo para atender esa peticion
                ThreadStart iniciadorHiloCliente = new ThreadStart(TareaMensajesAndroid);
                Thread hiloCliente = new Thread(iniciadorHiloCliente);
                hiloCliente.IsBackground = true;
                hiloCliente.Start();
            }
        }
        catch
        {
            Console.WriteLine("Error de conexión");
        }
    }
}
```

Código 3.13. Método `TareaClientesAndroid`

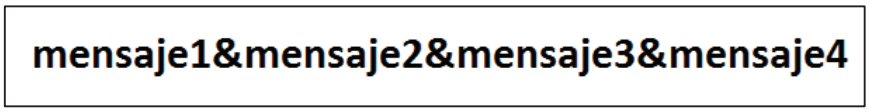


Figura 3.2. Trama Android

El `mensaje1` será la instrucción a realizar y puede ser:

- `pedir`: La aplicación Android pide los grupos de trabajo a la aplicación principal.
- `pedirGrupo`: La aplicación Android pide los PC que forman parte de un grupo de trabajo.
- `aG`: Apaga un grupo.
- `eG`: Enciende un grupo.
- `bG`: Bloquea un grupo.
- `dG`: Desbloquea un grupo.
- `fG`: Reinicia en modo *Frozen* un grupo.
- `tG`: Reinicia en modo *Thawed* un grupo.
- `aI`: Apaga un PC seleccionado de un grupo específico.

- `eI`: Enciende un PC seleccionado de un grupo específico.
- `bI`: Bloquea un PC seleccionado de un grupo específico.
- `dI`: Desbloquea un PC seleccionado de un grupo específico.
- `fI`: Reinicia en modo *Frozen* un PC seleccionado de un grupo específico.
- `tI`: Reinicia en modo *Thawed* un PC seleccionado de un grupo específico.

Para los anteriores el `mensaje2` indica el grupo, y el `mensaje3` indica el PC seleccionado. Además el `mensaje1` también puede ser:

- `solicitudes`: Envía las solicitudes activas almacenadas en el sistema.
- `archivarSolicitud`: Se archiva una solicitud activa. El `mensaje2` indica el id de dicha solicitud.
- `autenticacion`: Verifica si un usuario existe y es administrador. El `mensaje2` indica el nombre de usuario, y el `mensaje3` indica la contraseña.
- `pedirNotificacion`: Se envía la última solicitud activa a la aplicación Android si existe, además se actualiza la IP del usuario para que la aplicación principal pueda encontrar a la aplicación Android en la red; esto para futuras notificaciones. El `mensaje4` indica el usuario a actualizar.
- `pedirExtension`: Se envían las extensiones SIP de los usuarios con perfil administrador del sistema.

El **Código 3.14** muestra una parte del método `TareaMensajeAndroid`.

3.4.1.6 Leer archivo de configuración

La aplicación principal deberá leer y almacenar la clave que será utilizada para realizar las acciones del programa *DeepFreeze*; esta clave será almacenada en un `ArrayList` llamado `configuracion`. El **Código 3.15** muestra el método `LeerConfiguracion`.

```
public void TareaMensajesAndroid()// maneja a las clientes android en hilos diferentes
{
    //Se asigna un socket diferente para cada cliente
    Socket clienteAndroid = (Socket)clientesAndroid[clientesAndroid.Count - 1];
    NetworkStream datos = new NetworkStream(clienteAndroid);

    BinaryWriter datosEnviar = new BinaryWriter(datos); //se envia el tamaño del array servicios
    BinaryReader datosRecibir = new BinaryReader(datos);

        try
        {
           byte[] buffer = new byte[1024];
           int sent = clienteAndroid.Receive(buffer, 0, buffer.Length, SocketFlags.None);
           string datagrama = Encoding.UTF8.GetString(buffer, 0, sent);
           string[] mensajeDividido = datagrama.Split('&');
           string mensaje1 = mensajeDividido[0];
           string mensaje2 = mensajeDividido[1];
           string mensaje3 = mensajeDividido[2];
           string mensaje4 = "";
            try
            {
                 mensaje4 = mensajeDividido[3];
            }
            catch { }

            switch (mensaje1)
            {
                case "aG":
                    ApagarGrupo(mensaje2);
                    break;
                    .
                    .
                    .
```

Código 3.14. Método `TareaMensajeAndroid`

```
public void LeerConfiguracion()
{
    configuracion = new ArrayList();
    string line;
    //Se crea un objeto del tipo StreamReader para leer el archivo de
    //configuracion que se encuentra en la carpeta de la aplicación
    StreamReader file = new System.IO.StreamReader("conf.txt");
    while ((line = file.ReadLine()) != null)
    {
        if (line != "")
        {
            configuracion.Add(line);
        }
    }

    file.Close();
    claveDeepFrez = configuracion[0].ToString();
}
```

Código 3.15. Método `LeerConfiguracion`

3.4.1.7 Comunicación con la aplicación cliente

Los mensajes que se intercambian con la aplicación cliente se envían mediante el método `EnviarMensajesCliente`; la comunicación se realiza usando un puerto UDP que puede ser configurado; para la implementación se usó el puerto 10000. El **Código 3.16** muestra el mensaje `EnviarMensajeCliente`.

```
// se comunica con el cliente a gestionar
public void EnviarMensajeCliente(string ipDir, string mensaje)
{

    UdpClient udp_principal_cliente = new UdpClient();
    IPEndPoint ip = new IPEndPoint(IPAddress.Parse(ipDir), puertoCliente);
    byte[] bytes = Encoding.ASCII.GetBytes(mensaje+"&");
    udp_principal_cliente.Send(bytes, bytes.Length, ip);
}
```

Código 3.16. Método `EnviarMensajeCliente`

3.4.1.8 Envío de broadcast

Para encontrar los PC que ejecuten la aplicación cliente, la aplicación principal envía *broadcast* a la red. El **Código 3.17** muestra el método `EnviarBroadcast`.

```
//Para enviar mensajes de broadcast a los PC a ser gestionados
public void EnviarBroadcast()
{
    try
    {
        arrComputadoresActivados.Clear();
        cliente = new UdpClient();
        IPEndPoint ip = new IPEndPoint(IPAddress.Broadcast, puertoCliente);
        byte[] bytes = Encoding.ASCII.GetBytes("servidor&");
        cliente.Send(bytes, bytes.Length, ip);
    }
    catch
    {
        Console.WriteLine("Error en el envio de broadcast");
    }

}
```

Código 3.17. Método `EnviarBroadcast`

3.4.1.9 Recepción de broadcast

La aplicación principal cuenta con un método que espera recibir las respuestas del *broadcast*; en la trama se reciben nombre, grupo de trabajo, MAC, separados

por el símbolo "&". El **Código 3.18** muestra el método `RecibirRespuestasBroadcast`.

```
//Espera a recibir respuesta de broadcast
public void RecibirRespuestasBroadcast()
{
    udp_cliente_principal = new UdpClient(puertoCliente+1);
    IPEndPoint ip = new IPEndPoint(IPAddress.Any, puertoCliente+1);
    while (true)
    {
        try
        {
            byte[] bytes = udp_cliente_principal.Receive(ref ip);
            string mensaje = Encoding.ASCII.GetString(bytes, 0, bytes.Length);
            string[] mensaje_dividido = mensaje.Split('&');

            Computadores cli = new Computadores(mensaje_dividido[0], mensaje_dividido[1],
                ip.Address.ToString(), mensaje_dividido[2], true,
                Convert.ToBoolean(mensaje_dividido[3]));
                arrComputadoresActivados.Add(cli);
        }
        catch
        {
            Console.WriteLine("Error en la recepción de broadcast");
        }
    }
}
```

Código 3.18. Método `RecibirRespuestasBroadcast`

3.4.2 IMPLEMENTACIÓN DE LA APLICACIÓN PROFESOR

Es la encargada de controlar los PC que tengan el mismo grupo de trabajo. Fue implementada en Visual Studio 2010 usando el lenguaje de programación C#. Esta aplicación será usada por los profesores y administradores, mediante una interfaz gráfica. La **Figura 3.3** muestra la captura de pantalla de la aplicación profesor.

A continuación se detallan las secciones más importantes del código de la aplicación profesor; el código completo se lo puede encontrar en el **ANEXO C.1**

3.4.2.1 Gestión de PC

Para gestionar los PC se usan dos métodos. El primero comprueba las aplicaciones cliente; en esta sección se llamarán componentes, que fueron seleccionados por el usuario desde el elemento `ListView` y guarda esos

componentes en un `ArrayList`. El **Código 3.19** muestra el método `ComprobarSeleccion`.

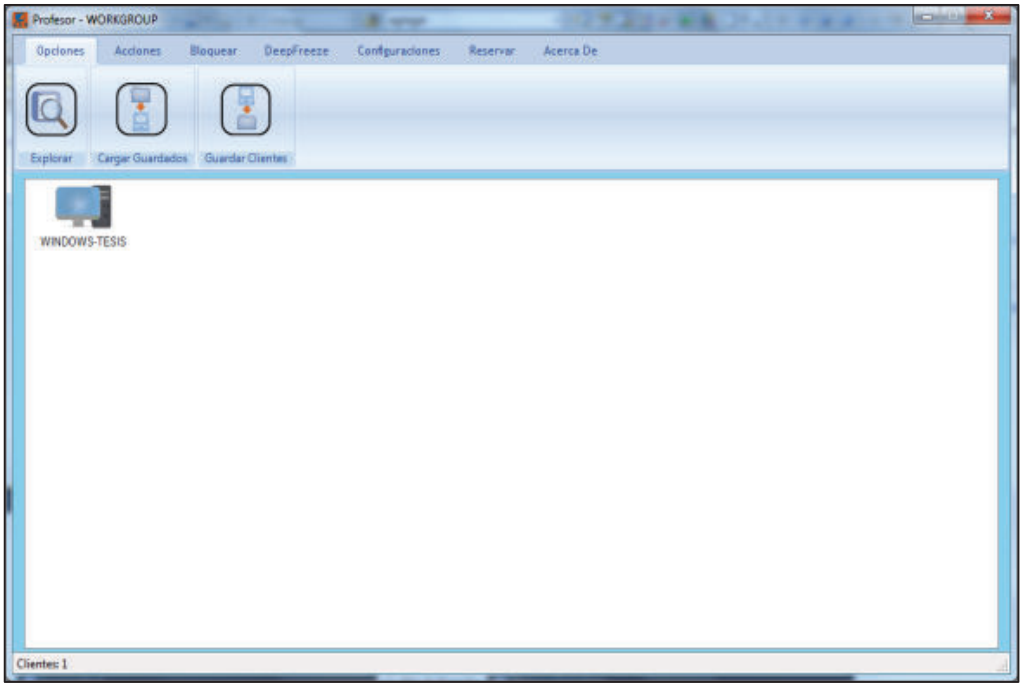


Figura 3.3. Aplicación profesor

```
public ArrayList ComprobarSeleccion()
{
    ArrayList arrTemporal = new ArrayList();
    foreach (Clientes cli in arrClientesDefinitivo)
    {
        foreach (ListViewItem item in lvPrincipal.Items)
            if (item.Selected == true)
            {
                if (item.Text == cli.Nombre)
                {
                    Clientes client = new Clientes(cli.Nombre, cli.Ip,
                        cli.Mac, cli.Workgroup, cli.Invitacion, cli.Activo,
                        cli.Bloquear,cli.Aplicaciones);
                    arrTemporal.Add(client);
                }
            }
    }
    return arrTemporal;
}
```

Código 3.19. Método `ComprobarSeleccion`

El segundo método es `HacerAccionTod`; este depende directamente de la variable `accion`, que puede tomar los valores: "`Apagar`", "`Encender`", "`Reiniciar`", "`Reiniciar Frozen`", "`Reiniciar Thawed`", "`Bloquear`", "`Desbloquear`". El **Código 3.20** muestra el método `HacerAccionTodo`.

```
// se encarga de identifica el tipo de acción solicitada
public void HacerAccionTodo(string accion)
{
    if (VerificarCantidad(accion)== "0") {
        MessageBox.Show("Seleccione al menos un PC que permita esta acción", "Error",
            MessageBoxButtons.OK, MessageBoxIcon.Error); }
    else{
    if (MessageBox.Show("Esta seguro de "+accion+" "+VerificarCantidad(accion)+
        " PC", "Accion",MessageBoxButtons.YesNo,MessageBoxIcon.Question) == DialogResult.Yes)
    {


            LeerConfiguracion();
            AccionesClienteTodo acc = new AccionesClienteTodo(ComprobarSeleccion(), claveDeep);
        switch (accion)
        {
            case "Apagar":

                acc.ApagarTodo();
            break;
            case "Encender":

                acc.EncenderTodo();
            break;
                  .
                  .
                  .
```

Código 3.20. Método `HacerAccionTodo`

Previo a realizar la acción se verifica la cantidad de componentes seleccionados y el estado de estos para presentar al usuario un mensaje de confirmación. El **Código 3.21** muestra parte del método `VerificarCantidad`.

```
public string VerificarCantidad(string accion) {
    int contador = 0;
    string cantidad="";
    switch (accion)
    {
        case "Apagar":
            foreach (Clientes cli in ComprobarSeleccion())
            {
                if (cli.Activo == true)
                {
                    contador = contador + 1;
                }
            }
            cantidad = contador.ToString();
            break;
               .
               .
               .
```

Código 3.21. Método `VerificarCantidad`

3.4.2.2 Dibujar componentes

Parte importante de la aplicación profesor es poder mostrar los componentes de una manera fácil de entender para el usuario; estos pueden tener los siguientes estados: apagado, encendido, bloqueado y aplicaciones abiertas (Word y Foxit Reader). El método `DibujarComponentes` distingue cada estado el cual será representado por una imagen. La **Figura 3.4** muestra los íconos de los distintos estados de los componentes. El **Código 3.22** muestra parte del método `DibujarComponentes`.

Figura 3.4. Estados de los componentes

```
public void DibujarComponentes(ArrayList arr)
{
    arrClientesDefinitivo = arr;
    toolStripStatusLabel1.Text = "Clientes: " + arr.Count.ToString();
    lvPrincipal.Items.Clear();
    imageList1.Images.Clear();
    this.lvPrincipal.View = View.LargeIcon;
    this.imageList1.ImageSize = new Size(60, 50);
    this.lvPrincipal.LargeImageList = this.imageList1;
    this.lvPrincipal.ContextMenuStrip = this.contextMenuStrip1;
    lvPrincipal.DoubleBuffering(true);
    lvPrincipal.Sorting = SortOrder.Ascending;
    int j = 0;
    foreach (Clientes cli in arr)
    {
        // comprueba el estado del cliente previo a mostrarse
        if (cli.Activo == true && cli.Bloquear == false)
        {
     // comprueba si las aplciaciones word o foxit reader se encuentran abiertas
            if (cli.Aplicaciones == "WINWORD@")
            {
                imageList1.Images.Add(cli.Nombre, Properties.Resources.word);
            }
```

Código 3.22. Método `DibujarComponentes`

Se debe tomar en cuenta que los componentes se vuelven a dibujar cada cierto tiempo para mantener la información visual actualizada; para esto se comprueba si hay componentes seleccionados y se los almacena en un `ArrayList`. El **Código 3.23** muestra el método `GuardarSeleccion` que permite guardar la selección de los componentes cuando se vuelve a dibujar.

```
// guarda los elentos seleccionados antes de redibujar
public void GuardarSeleccion()
{
    arrSeleccionados.Clear();

    foreach (ListViewItem st in lvPrincipal.Items)
    {

        if (st.Selected == true)
        {
            arrSeleccionados.Add(st.Text);
        }
    }
}
```

Código 3.23. Método `GuardarSelección`

3.4.2.3 Cargar componentes locales

Los componentes se almacenan localmente en un archivo y además se envían a la aplicación principal para mantener la información de sistema actualizada. El **Código 3.24** muestra el método `CargarGuardado`.

```
public void CargarGuardado()
{
  // carga los PC guardados localmente
    arrCliente.Clear();
    string line;
    StreamReader archivo = new System.IO.StreamReader("h.txt");
    while ((line = archivo.ReadLine()) != null)
    {
        if (line != "")
        {
            string[] words = line.Split('&');
            Clientes cli = new Clientes(words[0], words[1], words[2], words[3], "",false,false,"");
            arrCliente.Add(cli);
        }
    }

    archivo.Close();
}
```

Código 3.24. Método `CargarGuardado`

3.4.2.4 Comunicación con aplicación principal

Para la comunicación con la aplicación principal se usa el método `EnviarMensajeServidor`; como se mencionó anteriormente se usa un puerto UDP que puede ser configurado desde un formulario de configuración, para la implementación se usó el puerto 8083. El **Código 3.25** muestra el método `EnviarMensajeServidor`.

```
// metodo usadao para la comunicación con la aplicación principal
public void EnviarMensajeServidor(string mensaje)
{
    LeerConfiguracion();

        UdpClient udp_profesor_principal = new UdpClient();
        IPEndPoint ip = new IPEndPoint(IPAddress.Parse(ipServidor), puertoPrincipal);
        byte[] bytes = Encoding.ASCII.GetBytes(mensaje);
        udp_profesor_principal.Send(bytes, bytes.Length, ip);

}
```

Código 3.25. Método `EnviarMensajeServidor`

Parte de la comunicación con la aplicación principal involucra los métodos `VerificarUsuario`, `EnviarCorreo` y `EnviarComputadoresServidor`. El **Código 3.26** muestra el método `EnviarComputadoresServidor`.

```
//mantiene actualizado al servidor principal
public void EnviarComputadoresServidor()
{
    foreach (Clientes cli in arrClientesDefinitivo)
    {
        EnviarMensajeServidor("nuevoComputador&" + cli.Nombre +
            "@" + cli.Workgroup + "@" + cli.Ip + "@" + cli.Mac);
        EnviarMensajeServidor("nuevoGrupo&" + cli.Workgroup);

    }
    ObtenerDatos datos = new ObtenerDatos();
    EnviarMensajeServidor("nuevoComputador&" + datos.obtenerNombre() +
        "@" + grupoTrabajo + "@" + datos.obtenerIp() + "@" + datos.obtenerMAC());

}
```

Código 3.26. Método `EnviarComputadorServidor`

3.4.2.5 Comunicación con la aplicación cliente

Los mensajes que se intercambian con la aplicación cliente son enviados por el método `EnviarMensajeCliente`. La comunicación se realiza con un puerto UDP que puede ser configurado; para la implementación se usó el puerto 10000. El **Código 3.27** muestra el método `EnviarMensajeCliente`.

```
// metodo usadao para la comunicación con la aplicación cliente
public void EnviarMensajeCliente(string ipDir, string mensaje)
{
    UdpClient udp_profesor_cliente = new UdpClient();
    IPEndPoint ip = new IPEndPoint(IPAddress.Parse(ipDir), puertoCliente);
    byte[] bytes = Encoding.ASCII.GetBytes(mensaje);
    udp_profesor_cliente.Send(bytes, bytes.Length, ip);
}
```

Código 3.27. Método `EnviarMensajeCliente`

3.4.2.6 Iniciar sesión

Para poder acceder a la funcionalidad de préstamo de equipos es necesario autenticarse en el sistema. Los perfiles que pueden acceder a esta sección son: Profesor y Administrador; el método `VerificarUsuario` se comunica con la aplicación principal y espera a recibir una respuesta afirmativa o negativa. El **Código 3.28** muestra una parte del método `VerificarUsuario`.

Para poder acceder a la sección *DeepFreeze* también es necesario autenticarse, pero en este caso solo lo pueden hacer usuarios con el perfil de Administrador, el método `VerificarAdministrador` es el encargado de realizar esta tarea. El **Código 3.29** muestra una parte del método `VerificarAdministrador` que se diferencia de `VerificarUsuario` al comprobar si el tipo de usuario es Administrador. La **Figura 3.5** muestra la captura de pantalla del formulario de autenticación.

```
public void VerificarUsuario()
{
    if (txtClave.Text == "" || txtUsu.Text == "")
    {
        MessageBox.Show("Ingrese datos");// control de los datos ingresados
    }
    else
    {   // se comunica con la aplicación principal
        UdpClient udp_profesor_principal = new UdpClient();
        IPEndPoint ipServ = new IPEndPoint(IPAddress. arse(ipServidor),
        puertoPrincipal);
        //Cifrar clave con MD5
        string claveCifrada = cifrado.CifrarMd5(txtClave.Text);
        byte[] bytes = Encoding.UTF8.GetBytes("preguntar_usuario&" + txtUsu.Text
            + "@" + claveCifrada);
        udp_profesor_principal.Send(bytes, bytes.Length, ipServ);
        try
        {   // espera a recibir un mensaje desde la aplicación principal
            udp_profesor_principal.Client.ReceiveTimeout=2000;
            bytes = udp_profesor_principal.Receive(ref ipServ);
            string datagrama = Encoding.UTF8.GetString(bytes, 0, bytes.Length);
            string[] mensajeDividido = datagrama.Split('&');
            string mensaje = mensajeDividido[0];
            string mensaje1 = mensajeDividido[1];
            string mensaje2 = mensajeDividido[2];
            udp_profesor_principal.Close();
```

Código 3.28. Método `VerificarUsuario`

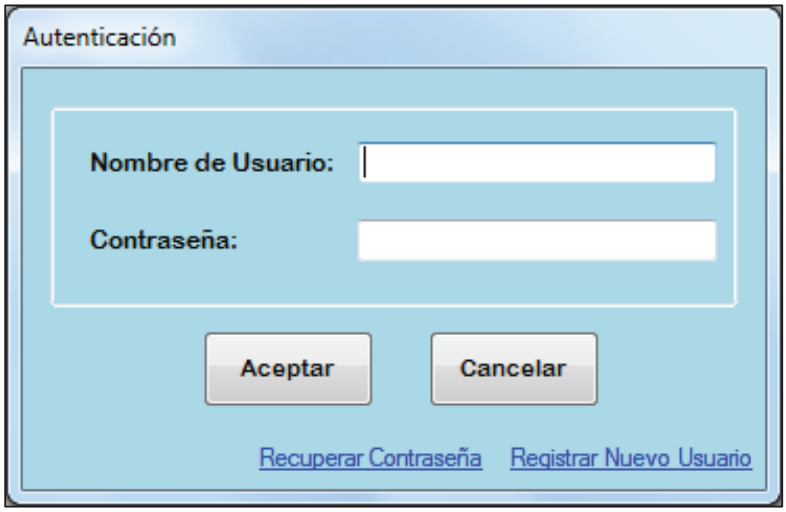


Figura 3.5. Formulario de autenticación

```
public String VerificarAdministrador()
{
    UdpClient udp_profesor_principal = new UdpClient();
    IPEndPoint ip = new IPEndPoint(IPAddress.Parse(ipServidor),
    puertoPrincipal);
    string claveCifrada = cifrado.CifrarMd5(txtClave.Text);
    // se comunica con la aplicación principal para preguntar por el usuario
    byte[] bytes = Encoding.UTF8.GetBytes("preguntarUsuario&" +
        txtUsu.Text + "@" + claveCifrada);
    udp_profesor_principal.Send(bytes, bytes.Length, ip);
    try
    {
        udp_profesor_principal.Client.ReceiveTimeout = 2000;
        // espera respuesta de la aplicación principal
        bytes = udp_profesor_principal.Receive(ref ip);
        string datagrama = Encoding.UTF8.GetString(bytes, 0, bytes.Length);

        string[] mensajeDividido = datagrama.Split('&');
        string mensaje = mensajeDividido[0];
        string mensaje1 = mensajeDividido[1];
        string mensaje2 = mensajeDividido[2];
        string mensaje3 = "";
        try
        {
            mensaje3 = mensajeDividido[3];
        }
        catch
        {
        }
        udp_profesor_principal.Close();
        if (mensaje == "" || mensaje3.Trim() != "Administrador")
            :
```

Código 3.29. Método `VerificarAdministrador`

3.4.2.7 Préstamo de equipos

El formulario cuenta con la posibilidad de generar una solitud de equipos; para esto se usa la interfaz gráfica que se presenta al usuario. Se puede escoger una fecha y un turno, con los métodos `VerificarDia` y `VerificarCantidad` se procede a determinar la cantidad disponible de equipos en ese momento. El **Código 3.30** muestra el método `VerificarDia`. El **Código 3.31** muestra parte el método `VerificarCantidad`. La **Figura 3.6** muestra la captura de pantalla del formulario de préstamo de equipos.

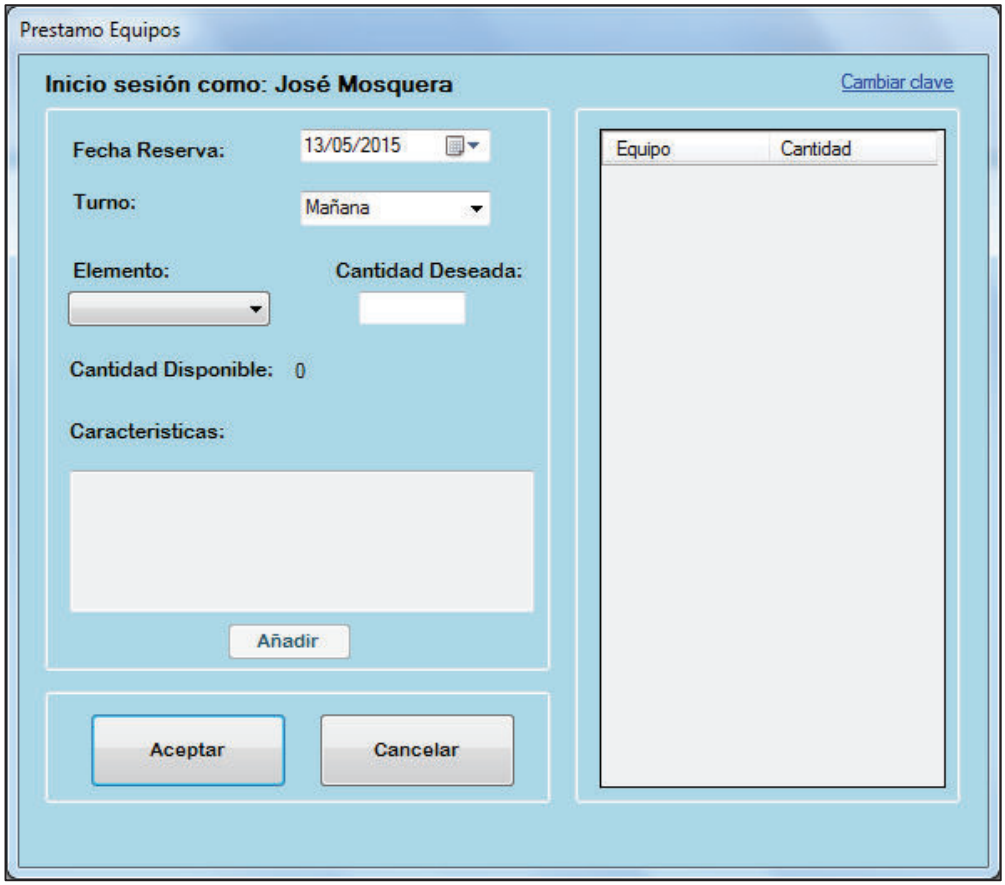


Figura 3.6. Formulario de préstamo de equipos

```
public void VerificarDia() {
    // verifica el día seleccionado y dependiendo de la
    //seleccion se generan los turno: primero,
    //segundo, mañana o noche
    dia = dtpFecha.Value.ToString("ddd");
    if (dia == "lun" || dia == "mar" || dia == "mié" ||
        dia == "jue" || dia == "vie")
    {
        cbTurno.Items.Clear();
        cbTurno.Items.Add("Mañana");
        cbTurno.Text = "Mañana";
        cbTurno.Items.Add("Noche");
    }
    if (dia == "sáb")
    {
        cbTurno.Items.Clear();

        cbTurno.Items.Add("Primero");
        cbTurno.Text = "Primero";
        cbTurno.Items.Add("Segundo");
    }
}
```

Código 3.30. Método `VerificarDia`

```
        }
    }

    foreach (DataGridViewRow r in dgTemporal.Rows)
    {
        if (r.Cells["equipo"].Value.ToString() == cbElemento.Text)
        {
            // se leeen los elementos del datagridview
            cantidadTemporal2 = Convert.ToInt32(r.Cells["cantidad"].Value);
        }
    }
    foreach (Equipos equi in Equipos())
    {
        if (equi.Nombre == cbElemento.Text)
        {
            // por ultimo se calcula la cantidad disponible
            //tomando en cuenta los equipso reservados
            //por otros usuarios y los equipos añadidos al datagridview
            lblCantidad.Text =
                (equi.CantidadT - cantidadTemporal-cantidadTemporal2).ToString();
        }
    }
}
```

Código 3.31. Método `VerificarCantidad`

3.4.3 IMPLEMENTACIÓN DE LA APLICACIÓN CLIENTE

La aplicación cliente recibe mensajes de la aplicación profesor y aplicación principal; en esta sección se las llamarán aplicaciones, estos mensajes son procesados para realizar una acción, también como un complemento en el PC donde se ejecuta la aplicación cliente se instala un servicio el cual verifica que esta esté activa y caso contrario la inicia. Cuenta con un solo formulario que se activa cuando el mensaje recibido es “bloquear”. Está implementada en Visual Studio 2010 usando el lenguaje de programación C#. La **Figura 3.7** muestra la captura de pantalla del formulario de la aplicación cliente.

Figura 3.7. Formulario de la aplicación cliente

A continuación se detallan las secciones más importantes del código de la aplicación cliente; el código completo se lo puede encontrar en el **ANEXO D.1**

3.4.3.1 Generar string de conexión

Para que las aplicaciones puedan conectarse por RDP a la aplicación cliente se debe generar un `string` de invitación. El método `GenerarString` es el encargado de generarlo. El **Código 3.32** muestra el método `GenerarString`.

```
public void GenerarString()
{
   try
    {
       //Se crea una nueva sesión RDP se establece una conexion
        sesionRdp = new RDPSession();
        sesionRdp.OnAttendeeConnected += new
            _IRDPSessionEvents_OnAttendeeConnectedEventHandler(OnAttendeeConnected);
        sesionRdp.OnAttendeeDisconnected += new
            _IRDPSessionEvents_OnAttendeeDisconnectedEventHandler(OnAttendeeDisconnected);
        sesionRdp.OnControlLevelChangeRequest += new
            _IRDPSessionEvents_OnControlLevelChangeRequestEventHandler(OnControlLevelChangeRequest);
        sesionRdp.Open();
        IRDPSRAPIInvitation pInvitation = sesionRdp.Invitations.CreateInvitation("Tesis", "Laboratorio", "", 5);
        invitacion = pInvitation.ConnectionString; // se crear el string de conexión
    }
    catch
    {
    }

}
```

Código 3.32. Método `GenerarString`

3.4.3.2 Recibir mensajes de aplicaciones

La aplicación cliente espera a recibir una trama de las aplicaciones; está compuesta por 2 mensajes separados por el símbolo “&”. La **Figura 3.8** muestra la trama que recibe la aplicación cliente.

Figura 3.8. Trama recibida por la aplicación cliente

El `mensaje1` puede tomar los siguientes valores:

- `profesor`: Indica que el broadcast llegó desde la aplicación profesor, en este caso comprueba con el `mensaje2` que la PC tenga el mismo grupo de trabajo antes de responder.
- `servidor`: Indica que el broadcast llegó desde la aplicación principal y responde al mensaje.
- `apagar`: Ejecuta la acción apagar.
- `reiniciar`: Ejecuta la acción encender.
- `fre`: Ejecuta la acción reiniciar en modo *Frozen*.
- `defre`: Ejecuta la acción reiniciar en modo *Thawed*.
- `invitacion`: Responde con el `string` de invitación
- `bloquear`: Ejecuta la acción bloquear.
- `desbloquear`: Ejecuta la acción desbloquear.
- `cerrar`: Cierra la aplicación cliente.
- `fin`: Finaliza él envió de imágenes a la aplicación profesor.

El **Código 3.33** muestra el método `RecibirMensajes`.

3.4.3.3 Responder broadcast

Una vez que la aplicación cliente detecta broadcast debe responder para informar a las aplicaciones que está activa, el broadcast se responde en un puerto UDP

que puede ser configurado; para la implementación se usó el puerto 10001. El **Código 3.34** muestra el método `ResponderBroadcast`.

```
while (true)
{

    byte[] bytes = udp_ProfesorServidor_Cliente.Receive(ref ip);//buffer para recibir mensajes del profesor
    string mensaje = Encoding.UTF8.GetString(bytes, 0, bytes.Length);
    string[] mensaje_dividido = mensaje.Split('&');
    switch (mensaje_dividido[0])
    {
           //Dependiendo del mesanje recibido se realiza la accion correspondiente
        case "profesor":
             if (mensaje_dividido[1] == resultado.ToString())//si el grupo de trabajo es igual
                   {
                       RespoderBroadcast();
                   }
            break;
        case "servidor":
            RespoderBroadcast();
            break;
        case "Apagar":

            lista.Invoke(new apagarEncender(AccionApagarReiniciar), new object[] { "5" });
        break;
        case "Reiniciar":

            lista.Invoke(new apagarEncender(AccionApagarReiniciar), new object[] { "6" });
        break;
        case "fre":

            AccionFrozenThawed(1, mensaje_dividido[1]);
        break;
        case "defre":
         AccionFrozenThawed(2, mensaje_dividido[1]);
        break;
```

Código 3.33. Método `RecibirMensajes`

```
public void RespoderBroadcast()
{
    try
    {
        //se cargan las aplicaciones que se estan ejecutando
        string aplicaciones = ObtenerAplicaciones();
        //responde broadcast
        IPEndPoint ipservidor = new IPEndPoint(ip.Address, puerto +1);
        byte[] bytes = Encoding.ASCII.GetBytes(Environment.MachineName + "&" +
            resultado.ToString() + "&" + dirMac + "&" + bloqueado.ToString() +
            "&" + aplicaciones);
        udp_Cliente_Profesor.Send(bytes, bytes.Length, ipservidor);
    }
    catch { Console.WriteLine("Error al responder broadcast");
    }
}
```

Código 3.34. Método `ResponderBroadcast`

3.4.3.4 Obtener aplicaciones

Para que la aplicación profesor pueda detectar las aplicaciones que están abiertas, la aplicación cliente debe ejecutar el método `ObtenerAplicaciones`; cabe mencionar que solo verifica si la aplicación WORD o la aplicación

FoxitReader están abiertas. El **Código 3.35** muestra el método `ObtenerAplicaciones`.

```
public string ObtenerAplicaciones() {
    string aplicaciones="";
    ArrayList arr = new ArrayList();
    Process procesoActual = Process.GetCurrentProcess();
    // Matriz de procesos
    Process[] procesos = Process.GetProcesses();
    // Recorremos los procesos en ejecución
    foreach (Process p in procesos)
    {
        if (!arr.Contains(p.ProcessName))
            arr.Add(p.ProcessName);
    }
    foreach (string s in arr)
    {
        if (s == "WINWORD" || s == "FoxitReader")
        {
            aplicaciones = aplicaciones + s + "@";
        }
    }
    return aplicaciones;
}
```

Código 3.35. Método `ObtenerAplicaciones`

3.4.3.5 Acciones apagar y reiniciar

Si las aplicaciones dan la orden a la aplicación cliente de apagar o reiniciar, se usa el método `AccionApagarReiniciar`; para esto se usa la librería propia del framework `System.Management` y se envía un parámetro el cual varía dependiendo de la opción requerida. El **Código 3.36** muestra el método `AccionApagarReiniciar`.

```
public void AccionApagarReiniciar(string accion)
 {
     ManagementBaseObject mboShutdown = null;
     ManagementClass mcWin32 = new ManagementClass("Win32_OperatingSystem");
     mcWin32.Get();
     mcWin32.Scope.Options.EnablePrivileges = true;
     ManagementBaseObject mboShutdownParams = mcWin32.GetMethodParameters("Win32Shutdown");
     mboShutdownParams["Flags"] = accion;
     mboShutdownParams["Reserved"] = "0";
     foreach (ManagementObject manObj in mcWin32.GetInstances())
     {
         mboShutdown = manObj.InvokeMethod("Win32Shutdown", mboShutdownParams, null);
     }
 }
```

Código 3.36. Método `AccionApagarReiniciar`

3.4.3.6 Acciones bloquear y desbloquear

Si las aplicaciones dan la orden a la aplicación cliente de bloquear o desbloquear, se usa el método `AccionBloquearDesbloquear`; para esto se usa la clase `BloquearMouseTeclado` que realiza un bloqueo o desbloqueo de los periféricos a bajo nivel. El **Código 3.37** muestra el método `AccionBloquearDesbloquear`.

```
public void AccionBloquearDesbloquear(int numero)
{
   switch (numero)
   {
       case 1:
        BloquearMouseTeclado.bloquearMouseTeclado();
        this.WindowState = System.Windows.Forms.FormWindowState.Maximized;
        /// Para que el Form este por encima de todo
        this.Visible = true;
        this.TopLevel = true;
        this.TopMost = true;
        Cursor.Position = new Point(Right, Top);//para que no aparezca el mouse
      break;
      case 2:
           //Bloquea teclado
        BloquearMouseTeclado.desbloquearMouseTeclado();
        this.WindowState = System.Windows.Forms.FormWindowState.Normal;
        this.Visible = false;
     break;
   }
}
```

Código 3.37. Método `AccionBloquearDesbloquear`

3.4.3.7 Acciones reiniciar en modo Frozen y reiniciar en modo Thawed

Si las aplicaciones dan la orden a la aplicación cliente de reiniciar en modo *Frozen* o reiniciar en modo *Thawed*, se emula el presionar de una serie de teclas que hacen que las acciones mencionadas se realicen con éxito. Se necesita la clave del programa *deepfreeze* la cual es enviada por las aplicaciones. El **Código 3.38** muestra el método `AccionFrozenThawed`.

3.4.4 IMPLEMENTACIÓN DE LA APLICACIÓN ANDROID

Es usada por los usuarios con perfil Administrador; está implementada en Android Studio usando el lenguaje de programación Java, mediante una interfaz gráfica que tiene varias opciones para: gestionar el sistema, ver solicitudes activas y

cambiar el estado de estas y comunicarse por voz usando SIP. La **Figura 3.9** muestra la captura de pantalla de la aplicación Android.

```
public void AccionFrozenThawed(int accion,string clave)
{
    //Teclas necesarias para ejecutar el programa DEEPFREEZE
    InputSimulator.SimulateKeyDown(VirtualKeyCode.CONTROL);
    InputSimulator.SimulateKeyDown(VirtualKeyCode.MENU);
    InputSimulator.SimulateKeyDown(VirtualKeyCode.SHIFT);
    InputSimulator.SimulateKeyDown(VirtualKeyCode.F6);
    InputSimulator.SimulateKeyUp(VirtualKeyCode.CONTROL);
    InputSimulator.SimulateKeyUp(VirtualKeyCode.MENU);
    InputSimulator.SimulateKeyUp(VirtualKeyCode.SHIFT);
    InputSimulator.SimulateKeyUp(VirtualKeyCode.F6);
    System.Threading.Thread.Sleep(1000);
    //se escribe la clave para entrar al programa
    InputSimulator.SimulateTextEntry(clave);
    System.Threading.Thread.Sleep(1000);
    SendKeys.SendWait("{ENTER}");
    //se ejecutan las acciones necesarias para
    //realizar el frozen o thawed
    switch (accion)
    {
        case 1:
            SendKeys.SendWait("%f");
            break;

        case 2:
            SendKeys.SendWait("%t");
            break;
    }
    SendKeys.SendWait("%r");
    SendKeys.SendWait("{ENTER}");
    SendKeys.SendWait("%s");
}
```

Código 3.38. Método `AccionFrozenThawed`

Figura 3.9. Aplicación Android

A continuación se detallan las secciones más importantes del código de la aplicación Android; el código completo se lo puede encontrar en el **ANEXO E.1**

3.4.4.1 Manifiesto de Android

Es uno de los archivos principales y debe ser declarado correctamente para que la aplicación Android no tenga problemas de funcionamiento; en este se define los permisos que va a tener la aplicación, las clases utilizadas para llamar a los Activity, el nivel de API utilizado, etc. El **Código 3.39** muestra el archivo `AndroidManifest`

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.android.example.tesisandroid" >
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.VIBRATE" />
    <uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity android:name=".Main" android:label="@string/app_name">
        </activity>
        - <activity android:name=".Autenticacion" android:label="Tesis">
        - <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
    </activity>
        <service android:enabled="true" android:name=".ServicioNotificaciones"
            android:icon="@drawable/ic_launcher" />
        <activity android:name=".Configuraciones" android:label="Configuraciones" />
        <activity android:name=".Descripcion" android:label="Descripcion"
            android:theme="@android:style/Theme.Dialog" />
        <activity android:name=".LLamadaActivity" android:label="Llamadas" />
        <activity android:name=".Contestar" android:label="Llamadas" />
    </application>
    <uses-permission android:name="android.permission.USE_SIP" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.VIBRATE" />
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
    <uses-permission android:name="android.permission.WAKE_LOCK" />
    <uses-permission android:name="android.permission.RECORD_AUDIO" />
    <uses-feature android:name="android.hardware.sip.voip" android:required="true" />
    <uses-feature android:name="android.hardware.wifi" android:required="true" />
    <uses-feature android:name="android.hardware.microphone" android:required="true" />
```

Código 3.39. Archivo AndroidManifest de la aplicación Android

3.4.4.2 Comunicación con aplicación principal

La comunicación se realiza usando un puerto TCP que puede ser configurado; para la implementación se usó el puerto 20000, los mensajes que puede enviar a la aplicación principal fueron descritos en la sección 3.4.1.5. El **Código 3.40** muestra el método `ComunicacionServidor`.

```
public void ComunicacionServidor(final String usuario, final String pass){
the = new Thread((Runnable) () -> {
        try {
            SharedPreferences prefs =
                    PreferenceManager.getDefaultSharedPreferences(getBaseContext());
            IpServidor =prefs.getString("domainPref", "");
             Cifrar cif = new Cifrar();
            String datagrama="Autenticacion"+"&"+usuario+"&"+pass;
            socket = new Socket(IpServidor,puertoServidor);
            socket.setSoTimeout(2000);
            DataOutputStream out =new DataOutputStream(socket.getOutputStream()),
            byte[] buf = datagrama.getBytes("UTF-8");
            out.write(buf, 0, buf.length);
            DataInputStream in = new
                    DataInputStream(new BufferedInputStream(socket.getInputStream()));
            byte[] buffer = new byte[1024];
            int i=in.read( buffer);
            String serverMessage = new String(buffer, 0,i,"utf-8");
            String[] array = serverMessage.split("@");
            if(array[0].equals("si")) {
                SharedPreferences settings =
                        getSharedPreferences("MisPreferencias", Context.MODE_PRIVATE);
                SharedPreferences.Editor editor = settings.edit();
                Cifrar ci = new Cifrar();
                editor.putString("usuario", nombreU.getText().toString());
                editor.putString("clave", ci.ObtenerMd5(claveU.getText().toString()));
                editor.commit();
                SharedPreferences.Editor editorName;
                editorName = prefs.edit();
                editorName .putString("namePref", array[1]);
                editorName .commit();
                Intent intent = new Intent(Autenticacion.this, Main.class);
                intent.putExtra("usuario",usuario);
                intent.putExtra("clave",pass);
                startActivity(intent);
                socket.close();
                finish();
            }
```

Código 3.40. Método `ComunicacionServidor`

3.4.4.3 Autenticación

Para usar la aplicación Android es necesario pasar por un proceso de autenticación. La **Figura 3.10** muestra la captura de pantalla de la actividad `Autenticacion`.

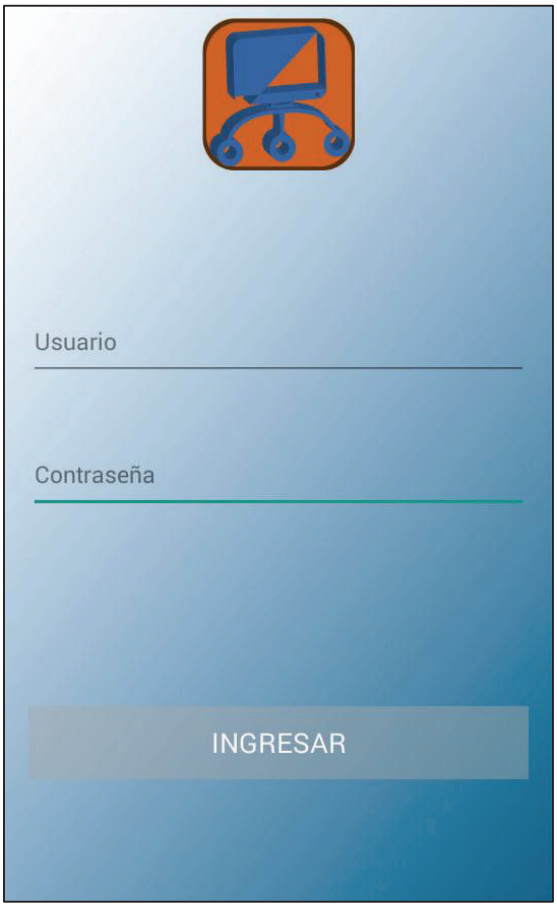


Figura 3.10. Actividad `Autenticacion`

Los datos ingresados por el usuario son enviados a la aplicación principal la cual responde afirmativa o negativamente. Solo los usuarios con perfil Administrador pueden hacer uso de la aplicación Android. El **Código 3.41** muestra una parte de la funcionalidad de autenticación.

```
if(array[0].equals("si")) {
    SharedPreferences settings =
            getSharedPreferences("MisPreferencias", Context.MODE_PRIVATE);
    SharedPreferences.Editor editor = settings.edit();
    Cifrar ci = new Cifrar();
    editor.putString("usuario", nombreU.getText().toString());
    editor.putString("clave", ci.ObtenerMd5(claveU.getText().toString()));
    editor.commit();
    SharedPreferences.Editor editorName;
    editorName = prefs.edit();
    editorName .putString("namePref", array[1]);
    editorName .commit();
    Intent intent = new Intent(Autenticacion.this, Main.class);
    intent.putExtra("usuario",usuario);
    intent.putExtra("clave",pass);
    startActivity(intent);
    socket.close();
    finish();
}
if(array[0].equals("no")) {
    SharedPreferences settings = getSharedPreferences("MisPreferencias", Context.MODE_PRIVATE);
    SharedPreferences.Editor editor = settings.edit();
```

Código 3.41. Método `Autenticacion`

3.4.4.4 Mostrar solicitudes activas

El detalle de las solicitudes activas es mostrado al iniciar la actividad `Descripcion`; a esta se le envían los parámetros por medio del método `putExtra` propio de Android. La **Figura 3.11** muestra la captura de pantalla de la actividad `Descripcion`. El **Código 3.42** indica el método `MostrarDescripcion`.

```
public void MostrarDescripcion(String id, String nombre, String curso,
                               String fecha, String turno, String equipos){
    Intent i =new Intent(getActivity(),Descripcion.class);
    i.putExtra("id",id);
    i.putExtra("nombre",nombre);
    i.putExtra("curso",curso);
    i.putExtra("fecha",fecha);
    i.putExtra("turno",turno);
    i.putExtra("equipos",equipos);
    startActivity(i);
}
```

Código 3.42. Método `MostrarDescripcion`

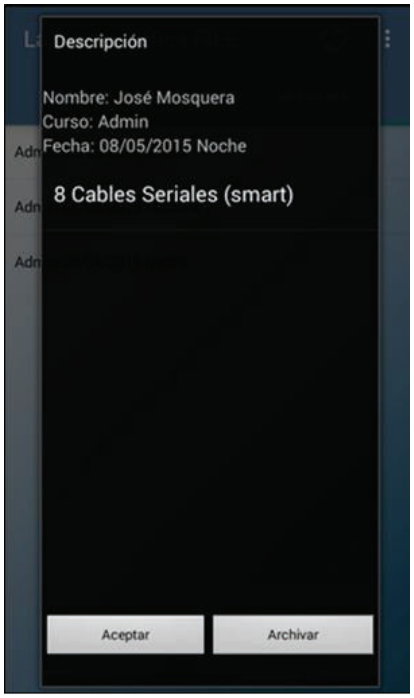


Figura 3.11. Actividad `Descripcion`

3.4.4.5 Archivar solicitudes

Es posible cambiar el estado de una solicitud activa por medio del método `Archivar`; de esta manera la solicitud será archivada. Para esto se comunica con la aplicación principal la cual realiza las acciones necesarias para ejecutar con éxito la acción. El **Código 3.43** muestra el método `Archivar`.

```
public void Archivar(View v) {
    AlertDialog.Builder alertDialog = new AlertDialog.Builder(this);
    alertDialog.setMessage("Esta seguro de archivar la solicitud?");
    alertDialog.setTitle("Orden");
    alertDialog.setIcon(android.R.drawable.ic_dialog_alert);
    alertDialog.setCancelable(false);
    alertDialog.setPositiveButton("Si", (dialog, which) -> {
            ComunicacionServidor("archivarSolicitud");
            Toast.makeText(getApplicationContext(), "Solicitud Archivada",
                    Toast.LENGTH_LONG).show();
            finish();
    });
    alertDialog.setNegativeButton("No", new DialogInterface.OnClickListener()
    {
        public void onClick(DialogInterface dialog, int which)
        {
        }
    });
    alertDialog.show();
}
```

Código 3.43. Método `Archivar`

3.4.4.6 Recibir notificaciones

La aplicación principal puede enviar notificaciones a la aplicación Android, para esto se utiliza un puerto UDP que puede ser configurado; para la implementación se usó el puerto 20003, al momento de recibir una notificación esta se muestra en la barra de notificaciones del dispositivo Android. El **Código 3.44** muestra el hilo que recibe las notificaciones.

```
Thread th = new Thread((Runnable) () -> {

        try {
            DatagramSocket sock1 = new DatagramSocket( puertoServidor +3);
            byte[] buffer1 = new byte[1024];
            DatagramPacket datagram1 = new DatagramPacket(buffer1, buffer1.length);
            while(true)
            {
                sock1.receive(datagram1);
                String message = new String(datagram1.getData(), 0, datagram1.getLength(), "utf-8");
                String[] array = message .split("@");
                    try {
                        LanzarNotificacion(array[0], array[1], array[2], array[3], array[4], array[5]);
                    }
                    catch(Exception e)
                    {}
            }
        }
        catch (Exception e) {
            Toast.makeText(ServicioNotificaciones.this,"Fallo de recepción",Toast.LENGTH_SHORT).show();
        }
```

Código 3.44. Hilo para recibir notificaciones

3.4.4.7 Llamadas

La función de llamadas usa el protocolo SIP; para esto la aplicación Android se comunica con el servicio SIP que se ejecuta en el mismo PC que la aplicación principal. El **Código 3.45** muestra el método para registrar con el servidor SIP

3.4.4.8 Iniciar y colgar llamada

En este se establecen los parámetros necesarios para realizar la llamada y finalizarla; se sobrescriben los métodos `onCallEstablished` y `onCallEnded` propios del API SIP. El **Código 3.46** muestra el método `IniciarLlamada`, el **Código 3.47** muestra el método `Colgar` y el **Código 3.48** muestra las líneas de código para recibir una llamada.

```
manager.setRegistrationListener(me.getUriString(),
        new SipRegistrationListener() {
            public void onRegistering(String localProfileUri) {
                Log.v("registrando","registrando");
                updateStatus("Registrando con servidor SIP...");
            }
            public void onRegistrationDone(String localProfileUri,
                                           long expiryTime) {
                Log.v("listo","listo");
                updateStatus("Listo para llamadas");
            }

            public void onRegistrationFailed(
                    String localProfileUri, int errorCode,
                    String errorMessage) {
                Log.v("fallo","fallo");
                updateStatus("Fallo de registro.  Revisar la configuración.");
            }
        });
```

Código 3.45. Método para registrar con el servidor SIP

3.4.4.9 Actividad contestar

Al recibir una llamada se tiene la opción de contestar o rechazar. La **Figura 3.12** muestra la captura de pantalla de la actividad `Contestar`.

```
public void IniciarLlamada(View view) {
    try {
        SipAudioCall.Listener listener = new SipAudioCall.Listener() {
            @Override
            public void onCallEstablished(SipAudioCall call) {
                call.startAudio();
                call.setSpeakerMode(false);
              if (call.isMuted()) {
                call.toggleMute();
                }
                updateStatus(call, 2);
            }
            @Override
            public void onCallEnded(SipAudioCall call) {
                updateStatus("Llamada finalizada");
            }
        };
```

Código 3.46. Método `IniciarLlamada`

```
public void Colgar(View view){

   if (call != null) {
        try {
            call.endCall();

            updateStatus("Lamada finalizada");

        } catch (SipException se) { }
         call.close();
        closeLocalProfile();
        initializeManager();
   }

}
```

Código 3.47. Método `Colgar`

```
final SipAudioCall.Listener listener = onRinging(call, caller) -> {
        try {
        } catch (Exception e)
        {
            e.printStackTrace();
        }
};
long pattern[]={0,200,100,300,400};
//Start the vibration
vibrator = (Vibrator)context.getSystemService(Context.VIBRATOR_SERVICE);
//start vibration with repeated count, use -1 if you don't want to repeat the vibration
vibrator.vibrate(pattern, 0);
LLamadaSIP wtActivity = (LLamadaSIP) context;
incomingCall = wtActivity.manager.takeAudioCall(intent, listener);
showIncomingCallGui(intent, context,incomingCall.getPeerProfile().getDisplayName());
wtActivity.call = incomingCall;
```

Código 3.48. Líneas de código para recibir una llamada

3.4.5 IMPLEMENTACIÓN DEL SERVICIO SIP

Para la implementación del servicio SIP es necesaria la instalación y configuración de Asterisk para Windows. Se usó Asterisk sobre Windows debido a que se modificarán los archivos de configuración desde la aplicación principal. La versión instalada es 0.66b [31].

Los archivos de configuración modificados son `sip.conf` y `extension.conf`; estos se modifican desde la aplicación principal con las extensiones de los nuevos usuarios registrados. Además se crearon dos archivos: `iniciarasterisk.bat`

`y programa.vbs` para reiniciar el servicio SIP al momento de agregar nuevas extensiones. El **Código 3.49** muestra las líneas de código para recargar el servicio SIP, el **Código 3.50** muestran las líneas de código para abrir la consola de configuración Asterisk.

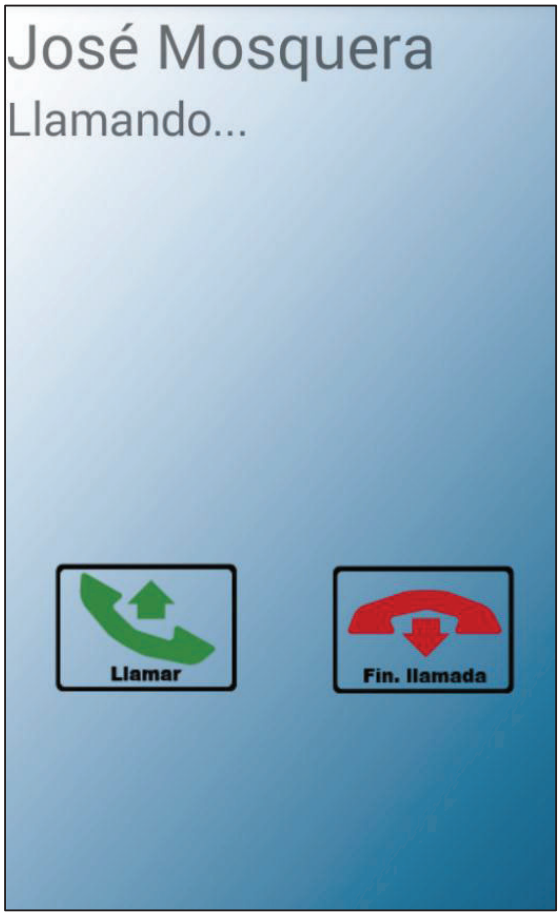


Figura 3.12. Actividad `Contestar`

```
Set WshShell = WScript.CreateObject("WScript.Shell")
WshShell.SendKeys "{ENTER}"
WshShell.SendKeys "sip reload"
WshShell.SendKeys "{ENTER}"
WshShell.SendKeys "load app_dial.so"
WshShell.SendKeys "{ENTER}"
WshShell.SendKeys "EXIT"
WshShell.SendKeys "{ENTER}"
```

Código 3.49. Líneas de código para recargar el servicio SIP

```
@ECHO OFF
CD C:\cygroot\bin
START asterisk.exe -r
wscript "C:\Aplicación Principal\programa.vbs"
```

Código 3.50. Líneas de código para abrir la consola de configuración Asterisk

3.4.6 IMPLEMENTACIÓN DE LOS INSTALADORES

La implementación de los instaladores se realizó usando Visual Studio 2010; para esto se creó un nuevo proyecto del tipo instalador. La **Figura 3.13** muestra la captura de pantalla de la ventana de Nuevo proyecto de Visual Studio 2010.

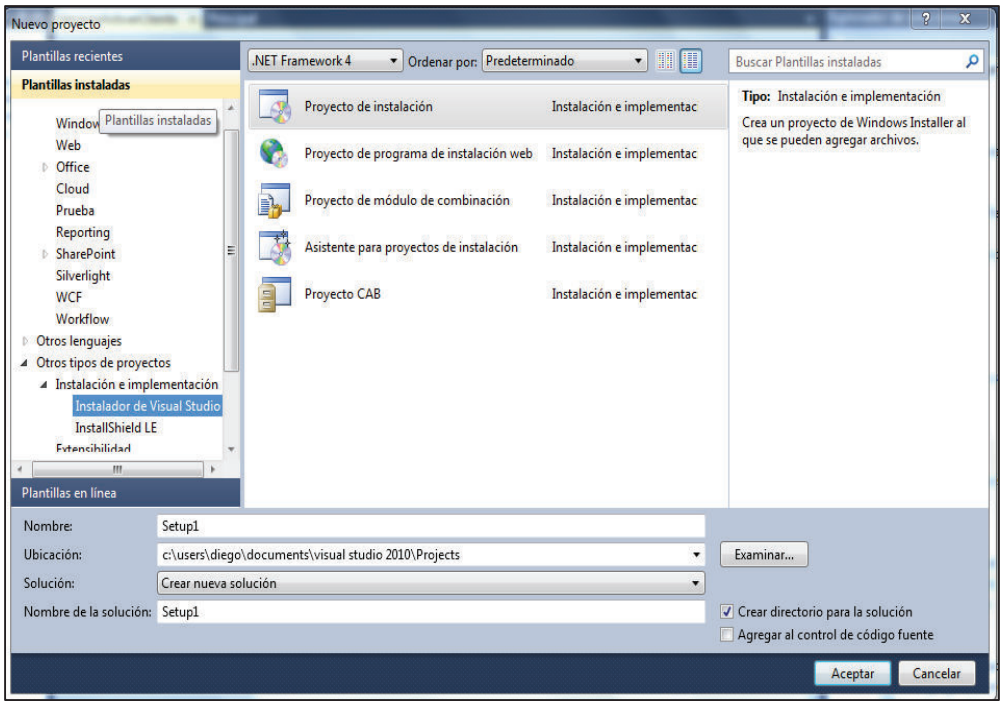


Figura 3.13. Ventana de Nuevo proyecto de Visual Studio 2010

Posteriormente se deben agregar los archivos que requieren ser instalados. La **Figura 3.14** muestra la captura de pantalla de sistema de archivos del instalador.

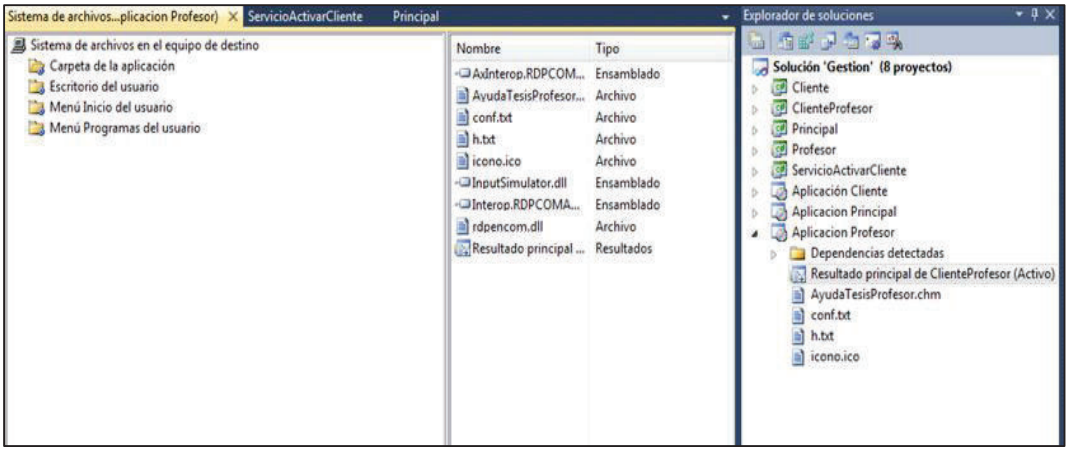


Figura 3.14. Pantalla de sistema de archivos del instalador

CAPÍTULO 4 PRUEBA DEL SISTEMA

4.1 INTRODUCCIÓN

En este capítulo se muestran los resultados obtenidos en las pruebas realizadas al sistema. Para esto se realizan pruebas unitarias a los métodos de las aplicaciones del sistema; también se realizan pruebas de funcionamiento en dos ambientes: controlado y real. El ambiente controlado consiste en utilizar tres máquinas virtuales y el real consiste en desplegar el sistema en el laboratorio de la FIEE.

El objetivo de realizar estas pruebas es para verificar que el sistema cumpla con los requisitos planteados previamente, y corregir errores del sistema en caso de ser necesario.

Posteriormente se realizan pruebas de aceptación y para concluir se compara el sistema y las aplicaciones descritas en el capítulo dos.

4.2 PRUEBAS UNITARIAS

Las pruebas unitarias comprobarán que los componentes del código funcionen individualmente dando valores y esperando respuestas ya sean estas correctas o incorrectas. Si los componentes funcionan individualmente de manera adecuada, al momento de integrarlos el sistema se obtendrá la respuesta deseada.

4.2.1 PRUEBAS UNITARIAS DEL SISTEMA

Visual Studio 2010 permite realizar este tipo de pruebas. La vista de pruebas presenta los componentes que se fueron seleccionados y se pueden ejecutar en cualquier momento. Cabe mencionar que solo los métodos que retornan valores pueden ser sometidos a pruebas unitarias.

En las pruebas que se realizaron para verificar el correcto funcionamiento del sistema se usa el método `AreEqual`; con este se comprueba que un valor dado sea igual al obtenido por el método del cual se está realizando la prueba. Se

deben inicializar los valores deseados; por ejemplo, para el método `VerificarUsuarioTest` se inicializan las variables `usuario` y `contraseña`; además, la variable `expected` sirve para inicializar el valor esperado. Posteriormente con el objeto `target` se llama al método `VerificarUsuario` y el resultado es almacenado en la variable `actual`. Finalmente como se mencionó, el método `AreEqual` comprueba las variables `expected` y `actual`.

El **Código 4.1** muestra el método `VerificarUsuarioTest`, el **Código 4.2** muestra el método `CifrarMD5Test`, el **Código 4.3** muestra el método `ComprobarSeleccionTest`, el **Código 4.4** muestra el método `GenerarStringTest` y el **Código 4.5** muestra el método `ObtenerAplicacionesTest`.

```
[TestMethod()]
public void VerificarUsuarioTest()
{
    Class1 target = new Class1();
    string usuario = "usu";
    string contraseña = "pass";
    string expected = "si";
    string actual;
    actual = target.VerificarUsuario(usuario, contraseña);
    Assert.AreEqual(expected, actual);

}
```

Código 4.1. Método de prueba `VerificarUsuarioTest`

```
[TestMethod()]
public void CifrarMD5Test()
{
    Class1 target = new Class1();
    string clave = "diego";
    string expected = "078c007bd92ddec308ae2f5115c1775d";
    string actual;
    actual = target.CifrarMD5(clave);
    Assert.AreEqual(expected, actual);

}
```

Código 4.2. Método de prueba `CifrarMD5Test`

```
[TestMethod()]
public void ComprobarSeleccionTest()
{
    Class1 target = new Class1();
    ArrayList array = new ArrayList();
    array.Add("1");
    array.Add("2");
    array.Add("3");
    int expected = 3;
    int actual;
    actual = target.ComprobarSeleccion(array);
    Assert.AreEqual(expected, actual);

}
```

Código 4.3. Método de prueba `ComprobarSeleccionTest`

```
[TestMethod()]
public void GenerarStringTest()
{
    Class1 target = new Class1();
    string expected = "<E><A KH=\"GedSDwqKw5hgQiWOmgDM0uI4sh4=\"" +
    "ID=\"Tesis\"/><C><T ID=\"1\" SID=\"1394226121\"><L" +
    "P=\"1198\" N=\"fe80::9054:7892:65f7:7029%11\"/><L" +
    "P=\"1199\" N=\"2801:0:270:44:9054:7892:65f7:7029\"/>" +
    " <L P=\"1200\" N=\"2801:0:270:44:e833:5151:8e55:8d3b\"/>" +
    " <L P=\"1201\" N=\"172.31.39.100\"/></T></C></E>" +
    "\r\\n\"";
    string actual;
    actual = target.GenerarString();
    Assert.AreEqual(expected, actual);

}
```

Código 4.4. Método de prueba `GenerarStringTest`

```
[TestMethod()]
public void ObtenerAplicacionesTest()
{
    Class1 target = new Class1();
    string expected = "WINWORD@Foxitreader";
    string actual;
    actual = target.ObtenerAplicaciones();
    Assert.AreEqual(expected, actual);

}
```

Código 4.5. Método de prueba `ObtenerAplicacionesTest`

Los resultados obtenidos de las pruebas unitarias se pueden encontrar en la sección “Resultados de pruebas”; allí se ven los nombres de los métodos que fueron sometidos a pruebas y aparece un símbolo de color verde que indica que la prueba pasó con éxito. La **Figura 4.1** muestra una captura de pantalla de la ventana resultados de pruebas.

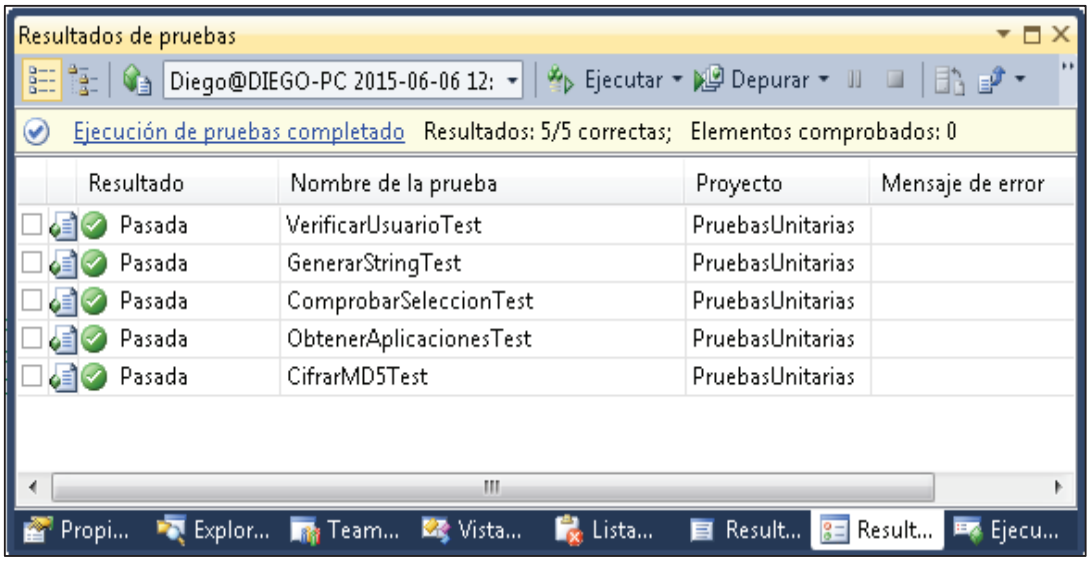


Figura 4.1. Pantalla de resultados de pruebas

4.3 PRUEBAS DE FUNCIONAMIENTO

4.3.1 AMBIENTE CONTROLADO

Las pruebas en ambiente controlado fueron realizadas en máquinas virtuales (una por cada aplicación) empleando el sistema operativo Windows Ultimate de 32 bits. La **Figura 4.2** muestra las características de las máquinas virtuales empleadas.

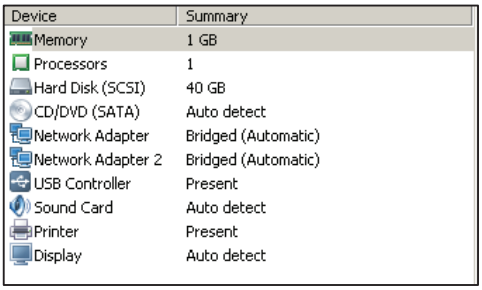


Figura 4.2. Características de la máquina virtual

Como se mencionó anteriormente, para el ambiente controlado se usan máquinas virtuales para cada aplicación; sin embargo, la aplicación Android funciona directamente en el dispositivo Android. La **Figura 4.3** muestra el esquema para el ambiente controlado.

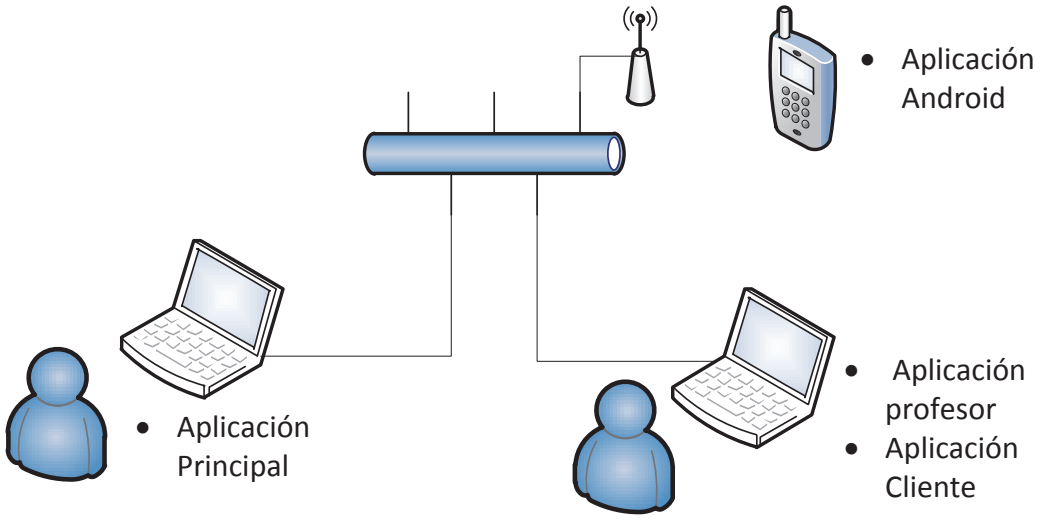


Figura 4.3. Esquema para el ambiente controlado

4.3.1.1 Aplicación principal

4.3.1.1.1 Prueba de instalación

Se deben instalar los requerimientos previos, estos son:

- WAMP 2.5
- AsteriskWin32 0.66b
- Microsoft .NET Framework 4.0

La **Figura 4.4** muestra una captura de pantalla de la instalación de WAMP 2.5, la **Figura 4.5** muestra una captura de pantalla de la instalación de AsteriskWin32 0.66b, la **Figura 4.6** muestra una captura de pantalla de la instalación de Microsoft .NET Framework 4.0.

Una vez instalados los requerimientos previos se puede ejecutar el paquete de instalación que fue generado para la aplicación principal. La **Figura 4.7** muestra la captura de pantalla de la instalación de la aplicación principal.

Finalizada la instalación, los archivos necesarios se copian a la ubicación `c:\Aplicación Principal` y se crea un acceso directo. La **Figura 4.8** muestra una captura de pantalla del acceso directo a la aplicación principal.

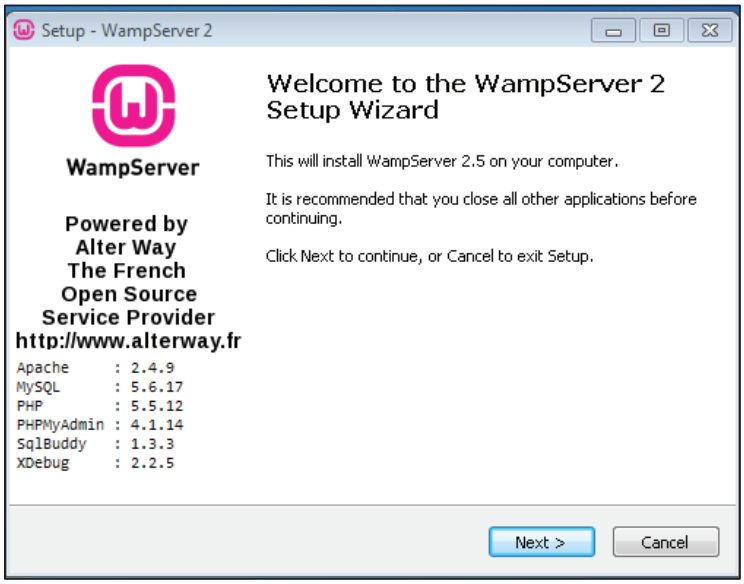


Figura 4.4. Pantalla de instalación de WAMP 2.5

Figura 4.5. Pantalla de instalación de AsteriskWin32 0.66b

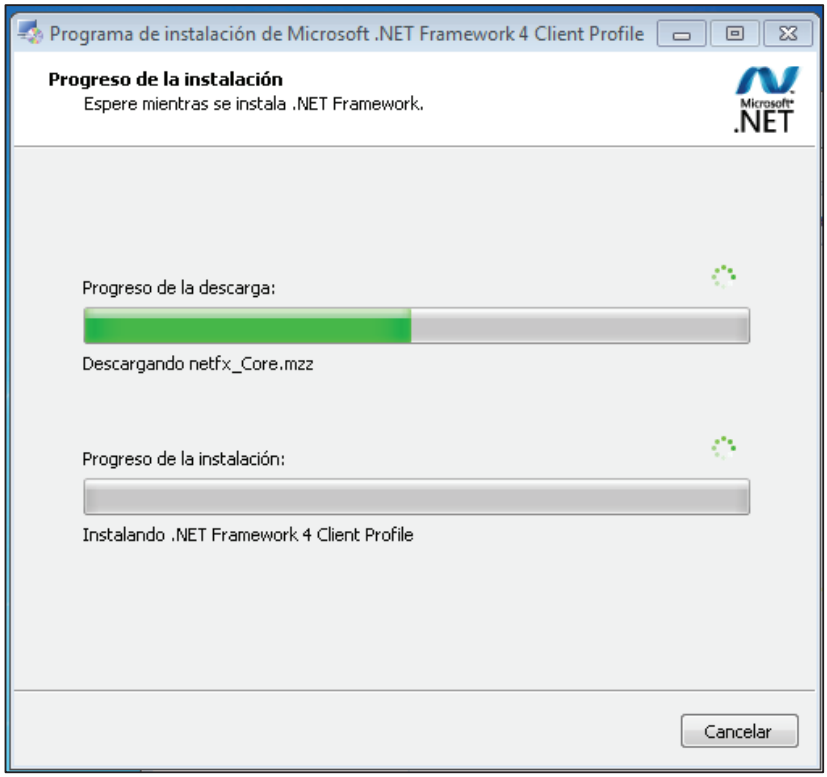


Figura 4.6. Pantalla de instalación de Microsoft .NET Framework 4.0

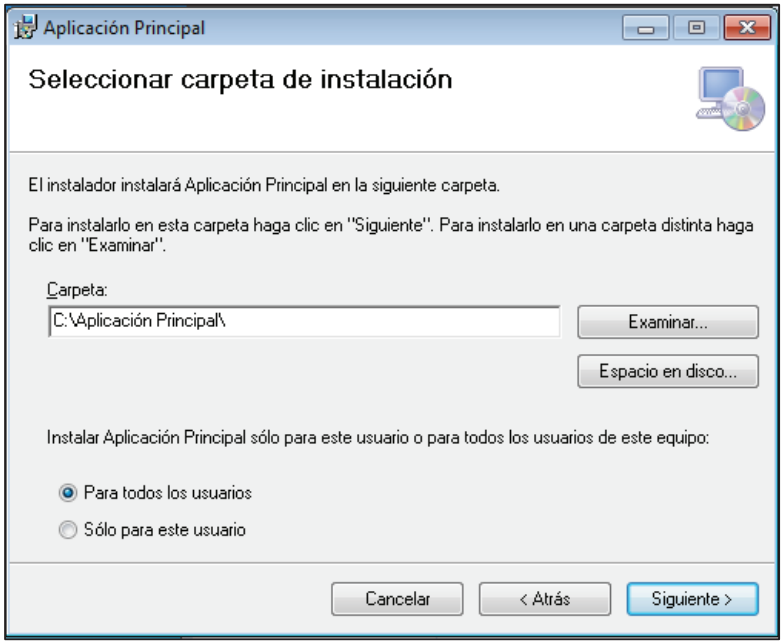


Figura 4.7. Pantalla de instalación de la aplicación principal

Figura 4.8. Acceso directo Aplicación principal

Se puede ejecutar la aplicación principal, sin embargo esta no mostrará ninguna información; para obtener datos se deben ejecutar la aplicación profesor y esta enviará los datos de sus clientes asociados; también se puede crear usuarios con perfil Profesor y generar solicitudes. La **Figura 4.9** muestra una captura de la pantalla primaria de la aplicación principal.

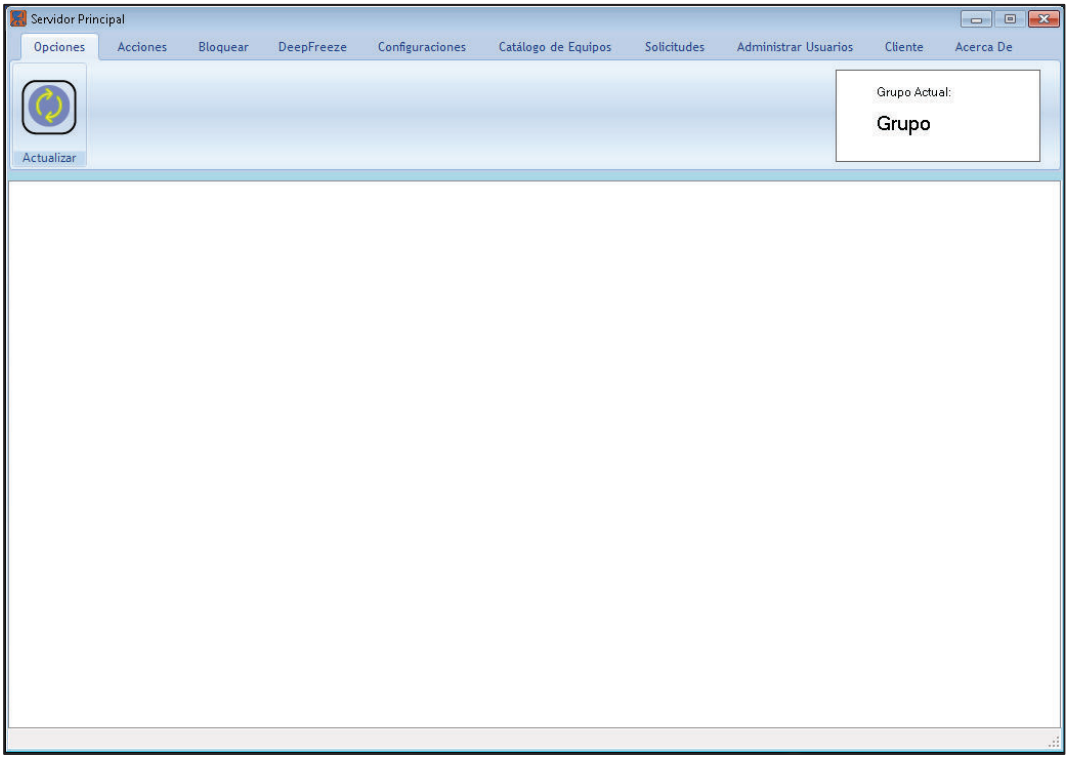


Figura 4.9. Pantalla primaria de la aplicación principal

4.3.1.1.2 Prueba de recepción de datos desde la aplicación profesor

Con la aplicación profesor instalada esta puede enviar la información de sus clientes asociados y la información de esta. La aplicación principal guarda la

información recibida en la base de datos y desde este momento aparecen en la pantalla primaria. La **Figura 4.10** muestra la captura de pantalla de los PC en la pantalla primaria.

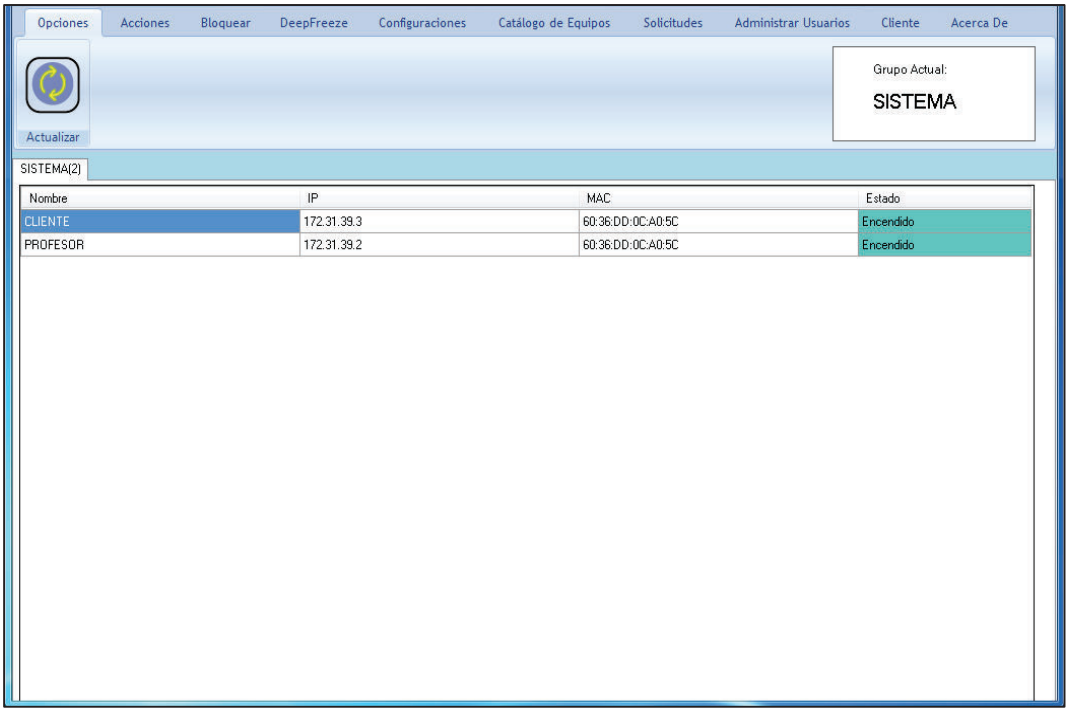


Figura 4.10. Pantalla de los PC agregados en la pantalla primaria

4.3.1.1.3 Prueba de registro de usuarios

Para esta prueba se agrega un usuario con perfil Administrador. La **Figura 4.11** muestra una captura de pantalla del formulario agregar usuarios.

4.3.1.1.4 Prueba de visualización de usuarios

Los usuarios pueden ser visualizados desde la sección `Administrar Usuarios/Ver Usuarios`. La **Figura 4.12** muestra la captura de pantalla de la visualización de usuarios

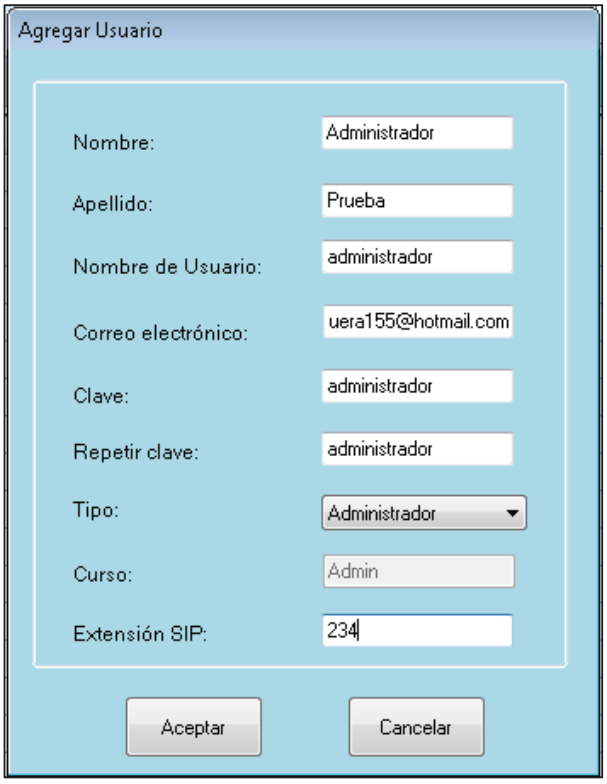


Figura 4.11. Formulario Agregar Usuario

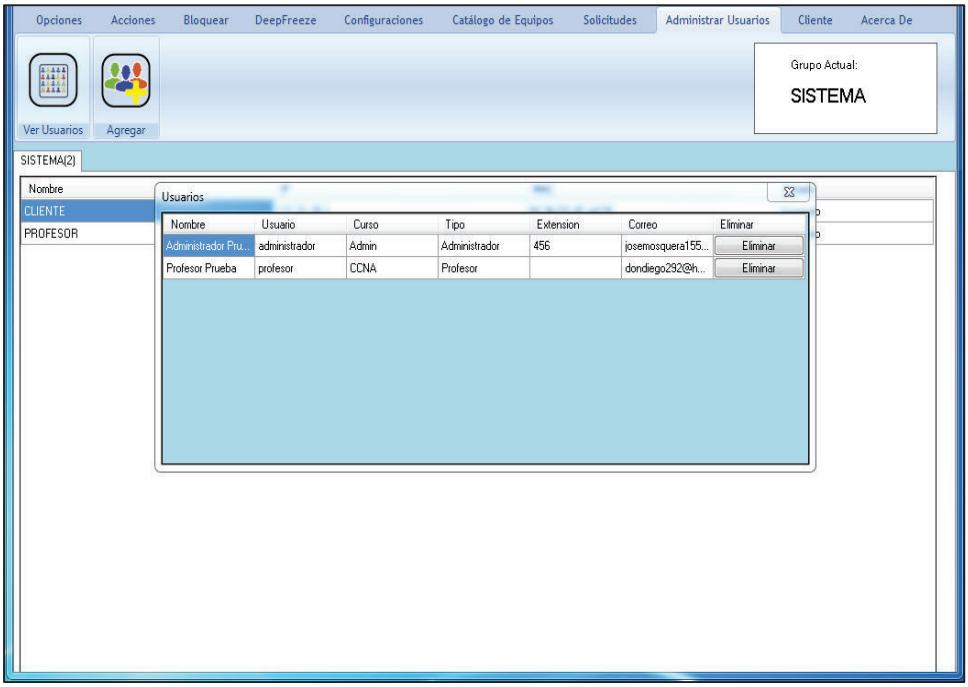


Figura 4.12. Pantalla de visualización de usuarios

4.3.1.1.5 Prueba de comunicación con aplicación profesor y cliente

Una de las acciones que se pueden realizar es el acceso remoto; al hacer doble clic sobre los elementos de `DataGridView` se intenta una conexión, en caso de ser positiva se despliega una ventana en donde se puede ver y controlar el escritorio remoto, en caso de ser negativa se informa al usuario. La **Figura 4.13** muestra la captura de pantalla del acceso remoto al PC cliente, la **Figura 4.14** muestra la captura de pantalla del acceso remoto al PC profesor la **Figura 4.15** muestra la captura de pantalla del mensaje de error al recibir una respuesta negativa después de intentar conectarse con un PC no disponible.

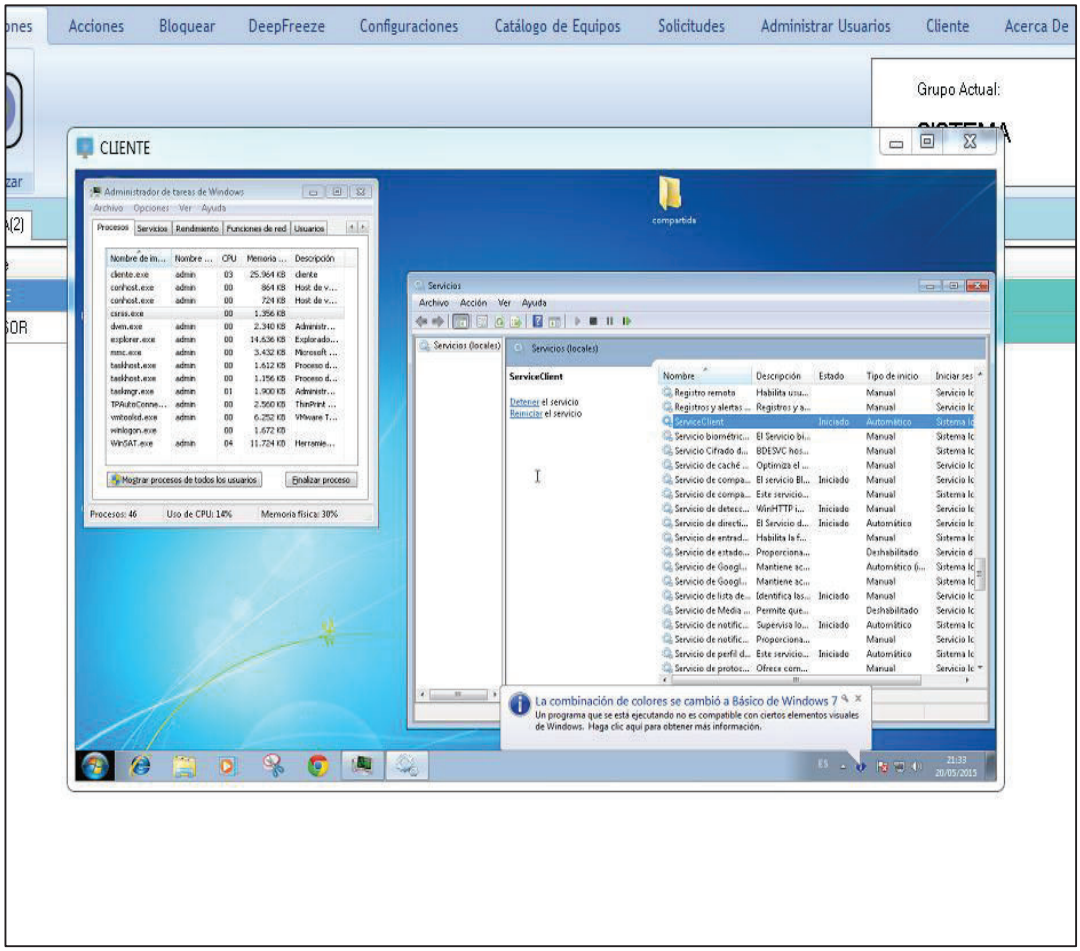


Figura 4.13. Pantalla de acceso remoto al PC cliente desde aplicación principal

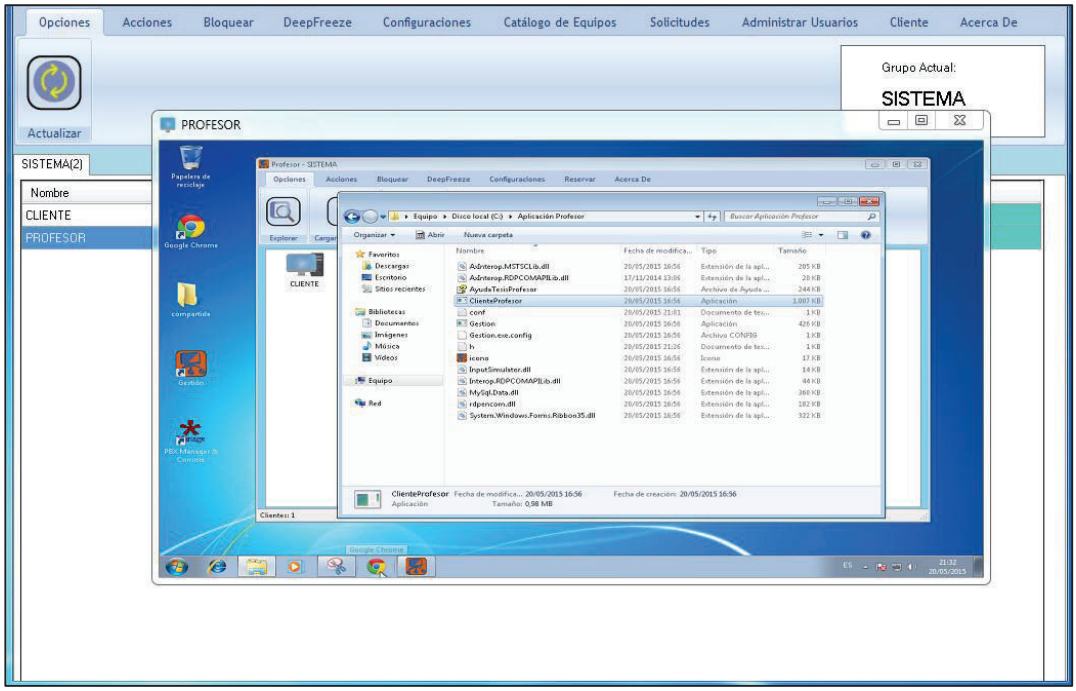


Figura 4.14. Pantalla de acceso remoto al PC profesor desde aplicación principal

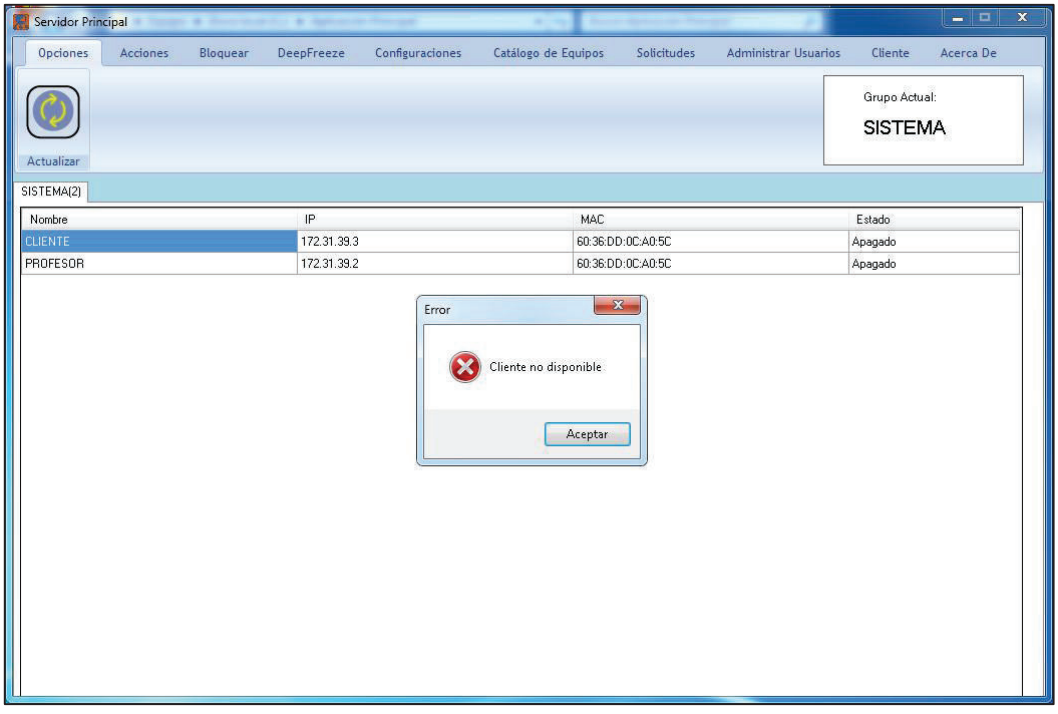


Figura 4.15. Mensaje de error “Cliente no disponible”

4.3.1.2 Aplicación profesor

4.3.1.2.1 Prueba de instalación

Se deben instalar Microsoft .NET Framework 4.0 para el correcto funcionamiento de la aplicación. Una vez instalado el requerimiento previo se puede ejecutar el paquete de instalación que fue generado para la aplicación profesor. La **Figura 4.16** muestra la captura de pantalla de instalación de la aplicación profesor.

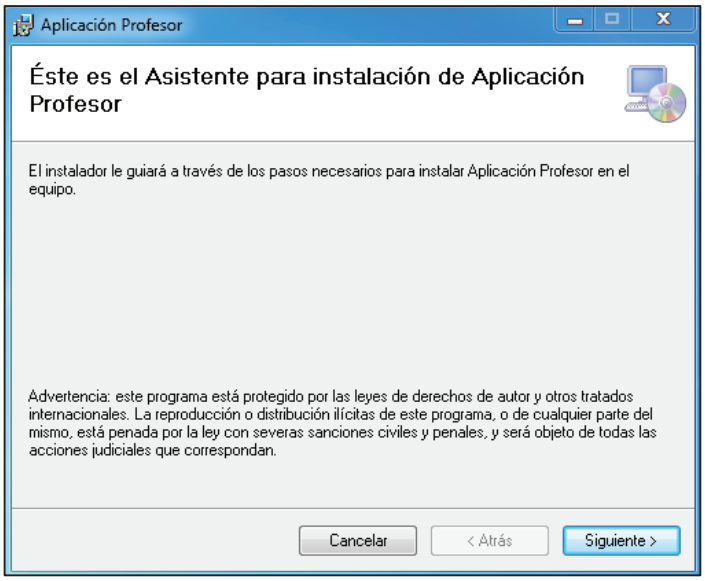


Figura 4.16. Pantalla de instalación de la aplicación profesor

Una vez finalizada la instalación los archivos necesarios se copian a la ubicación `c:\Aplicación Profesor` y se crea un acceso directo. La **Figura 4.17** muestra una captura de pantalla del acceso directo a la aplicación profesor.

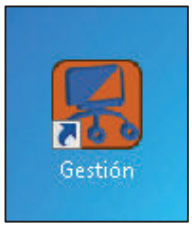


Figura 4.17. Acceso directo Aplicación profesor

La **Figura 4.18** muestra una captura de la pantalla primaria de la aplicación profesor

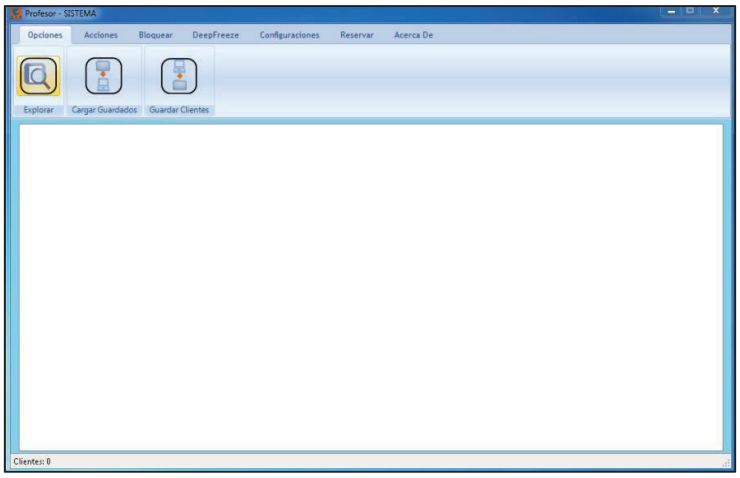


Figura 4.18. Pantalla primaria de la aplicación profesor

4.3.1.2.2 Prueba de comunicación con aplicación cliente

Para comunicarse con la aplicación cliente por primera vez es necesario presionar el botón `Explorar` con lo cual se genera un petición y así los PC que están dentro del mismo grupo responden a esta. Posteriormente se puede guardar la información de los componentes presionando el botón `Guardar clientes` y la próxima vez que se ejecute la aplicación profesor se cargarán automáticamente estos. La **Figura 4.19** muestra la captura de pantalla después de presionar al botón explorar.

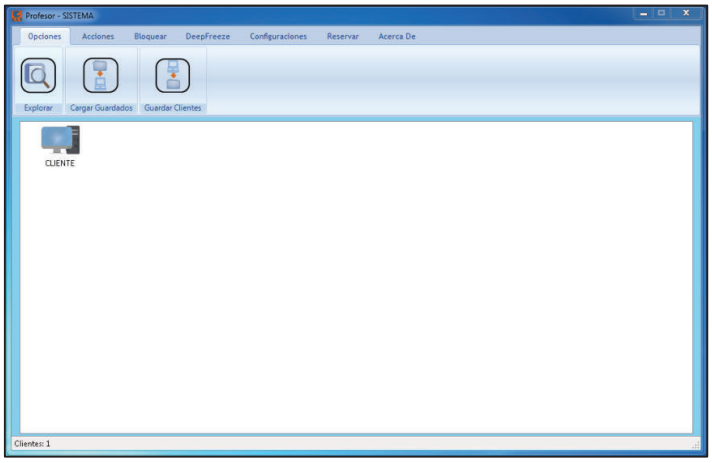


Figura 4.19. Captura de pantalla después de presionar al botón explorar

Una de las acciones que se pueden realizar es el acceso remoto; al hacer doble clic sobre los componentes del `ListView` se intenta una conexión; en caso de ser exitosa se despliega una ventana en donde se puede ver y controlar el escritorio remoto. La **Figura 4.20** muestra la captura de pantalla del acceso remoto al PC cliente.

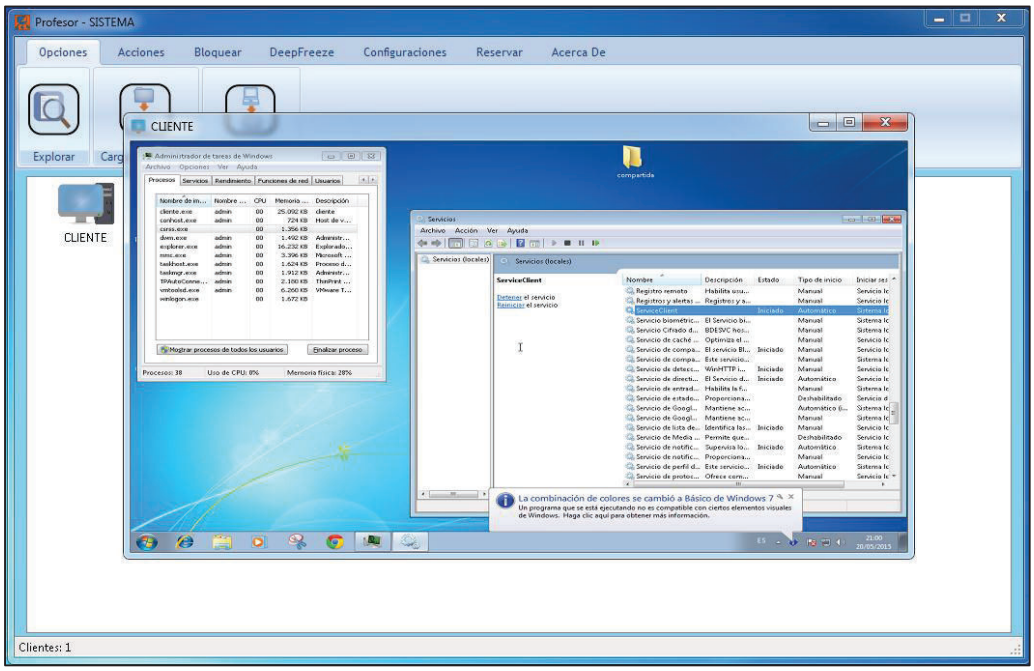


Figura 4.20. Pantalla de acceso remoto al PC cliente desde aplicación profesor

4.3.1.2.3 Prueba de registro de usuario

La aplicación profesor cuenta con la posibilidad de registrar usuarios con perfil Profesor al sistema. Se creó un usuario para la prueba; una vez con los datos completos se recibe un correo electrónico con la clave de ingreso, que posteriormente puede ser cambiada. La **Figura 4.21** muestra la captura de pantalla del formulario de registro de usuarios, la **Figura 4.22** muestra la captura de pantalla del correo electrónico recibido.

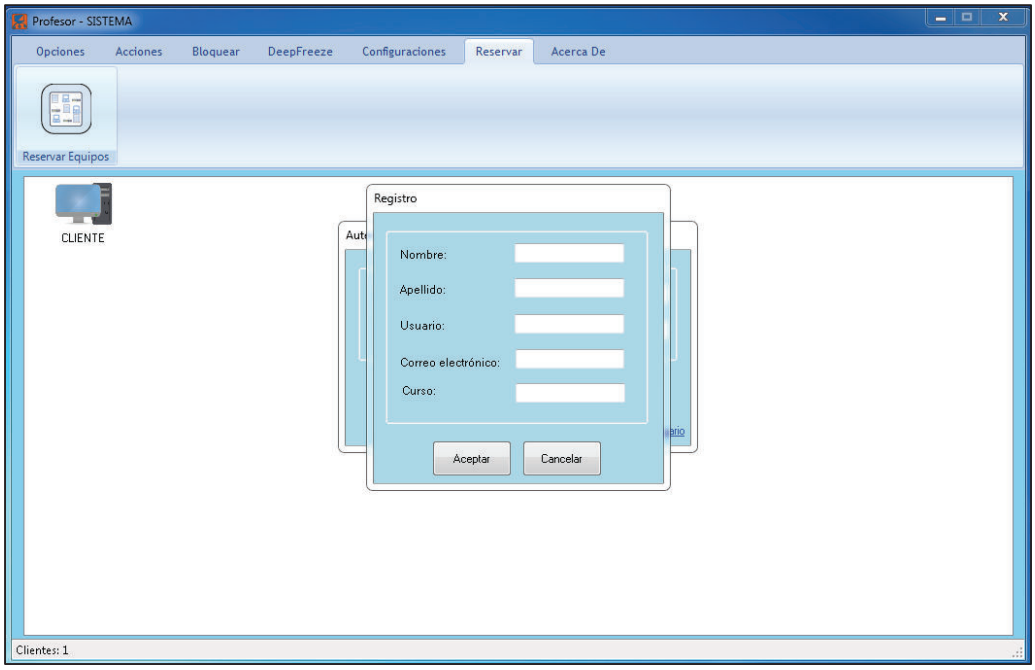


Figura 4.21. Formulario de registro de usuarios.

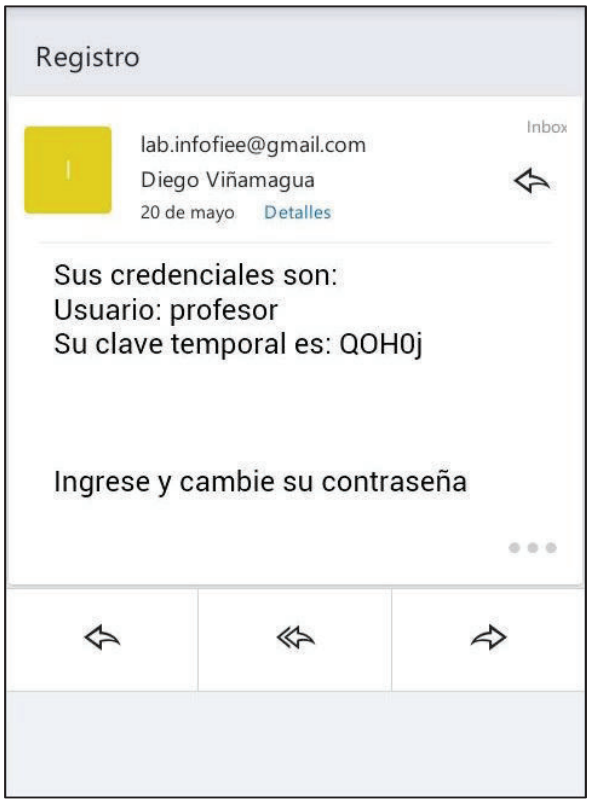


Figura 4.22. Correo electrónico recibido después de registrar un usuario

4.3.1.3 Aplicación cliente

Se deben instalar Microsoft .NET Framework 4.0 para el funcionamiento correcto de la aplicación. Una vez instalado el requerimiento previo se puede ejecutar el paquete de instalación que fue generado para la aplicación cliente. La **Figura 4.23** muestra la captura de pantalla de instalación de la aplicación cliente. La **Figura 4.24** muestra la captura de pantalla del administrador de tareas de Windows donde se puede ver que la aplicación cliente se está ejecutando.

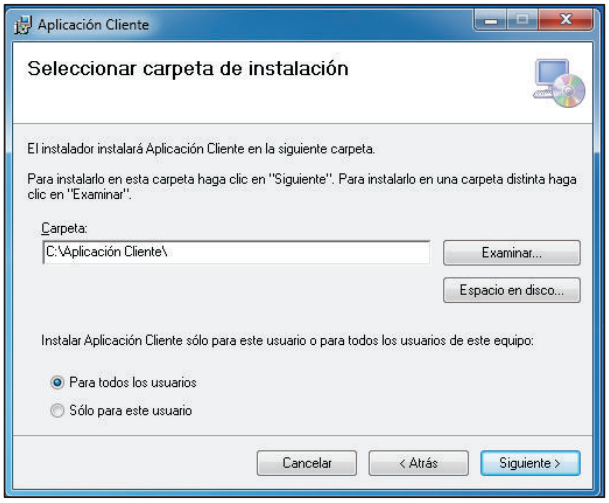


Figura 4.23. Pantalla de instalación de la aplicación cliente

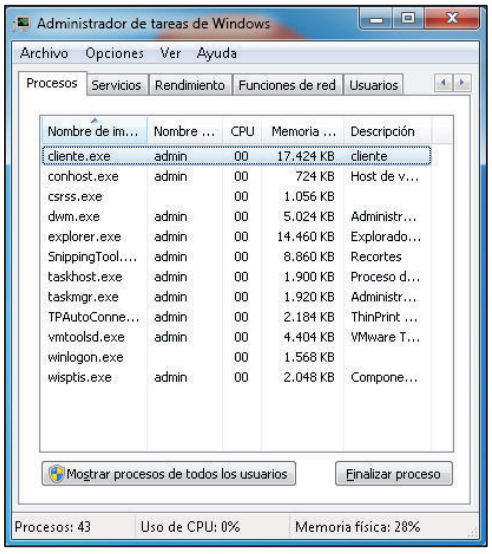


Figura 4.24. Administrador de tareas de Windows

4.3.2 AMBIENTE REAL

Las pruebas en ambiente real fueron realizadas usando los PC ubicados en el Laboratorio de Informática de la FIEE; estos cuentan con sistema operativo Windows Profesional de 64 bits.Los PC son de 2 tipos: rojos y plomos. La **Tabla 4.1** muestra las características de los PC rojos, la **Tabla 4.2** muestra las características de los PC plomos.

Componente	Característica
Procesador	Intel Core i7
RAM	4 GB
Disco duro	500 GB
Tarjeta de video dedicado	1 GB

Tabla 4.1. Características PC rojo

Componente	Característica
Procesador	Intel Core 2 DUO
RAM	2 GB
Disco duro	500 GB
Tarjeta de video dedicado	No tiene

Tabla 4.2. Características PC plomo

A continuación se muestran las pruebas realizadas en el ambiente real. Para esto se utilizaron las salas A y E, las cuales están ubicadas en el Laboratorio de Informática de la FIEE. La Figura 4.25 muestra el esquema para el ambiente real.

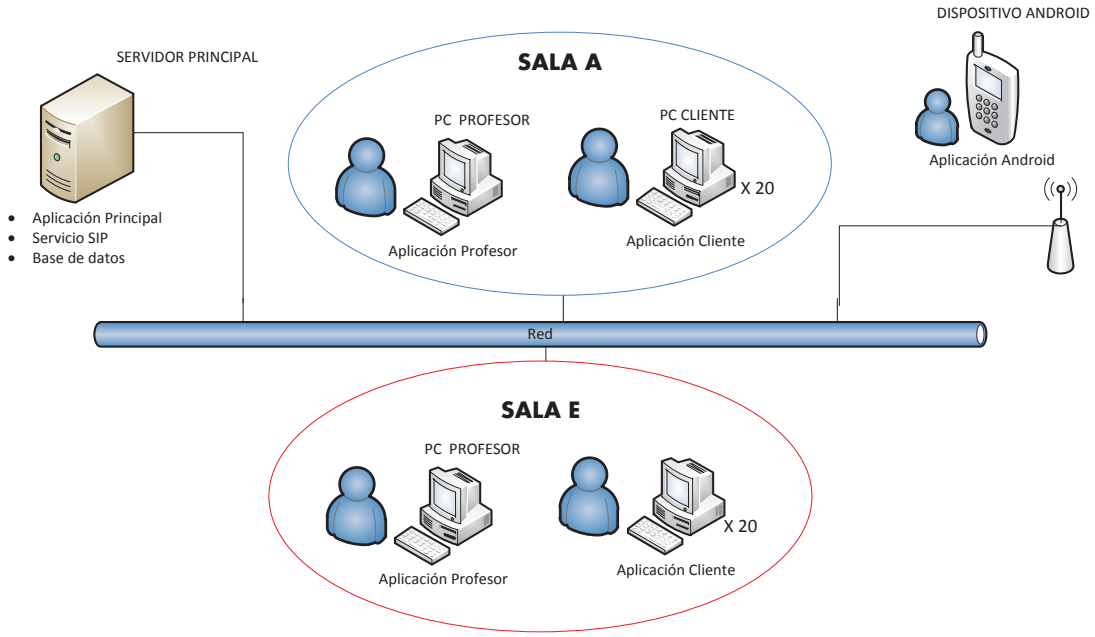


Figura 4.25. Esquema para el ambiente real

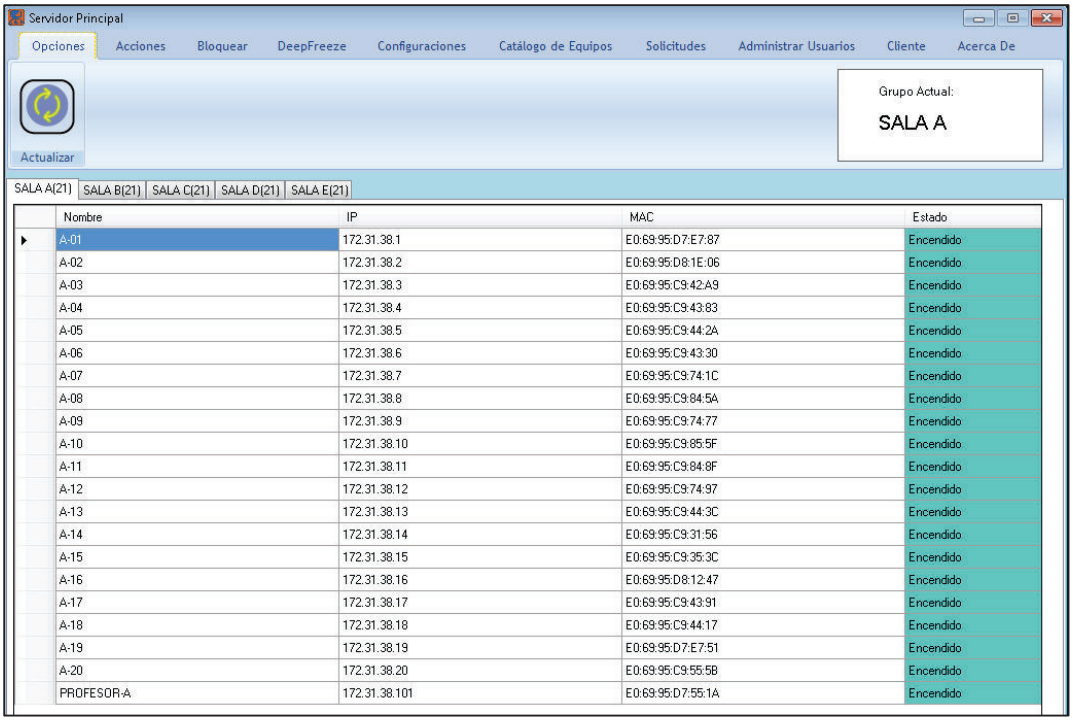
4.3.2.1 Aplicación principal

La prueba de instalación de la aplicación principal funcionó de la misma manera que en el ambiente controlado.

4.3.2.1.1 Prueba de recepción de datos desde la aplicación profesor

La prueba consiste en que se presenten los datos de los PC que fueron recibidos desde la aplicación profesor.

Con la aplicación profesor instalada en las salas, esta enviará la información de los PC de su misma sala y la información de esta. La aplicación principal guarda la información recibida en la base de datos y desde este momento aparece en la pantalla primaria. La **Figura 4.26** muestra la captura de pantalla de los PC de la SALA A en pantalla primaria, la **Figura 4.27** muestra la captura de pantalla de los PC de la SALA A almacenados en la base de datos, la **Figura 4.28** muestra la captura de pantalla de los PC de la SALA E en pantalla primaria, la **Figura 4.29** muestra la captura de pantalla de los PC de la SALA E almacenados en la base de datos.

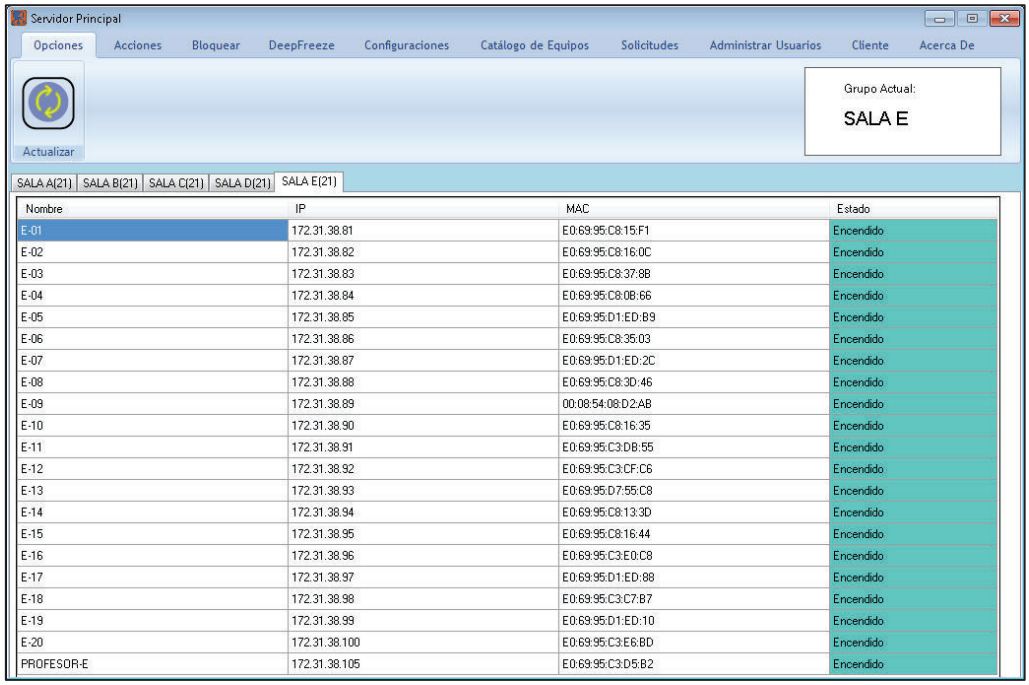


Nombre	IP	MAC	Estado
A-01	172.31.38.1	E0:69:95:D7:E7:87	Encendido
A-02	172.31.38.2	E0:69:95:D8:1E:06	Encendido
A-03	172.31.38.3	E0:69:95:C9:42:A9	Encendido
A-04	172.31.38.4	E0:69:95:C9:43:83	Encendido
A-05	172.31.38.5	E0:69:95:C9:44:2A	Encendido
A-06	172.31.38.6	E0:69:95:C9:43:30	Encendido
A-07	172.31.38.7	E0:69:95:C9:74:1C	Encendido
A-08	172.31.38.8	E0:69:95:C9:84:5A	Encendido
A-09	172.31.38.9	E0:69:95:C9:74:77	Encendido
A-10	172.31.38.10	E0:69:95:C9:85:5F	Encendido
A-11	172.31.38.11	E0:69:95:C9:84:8F	Encendido
A-12	172.31.38.12	E0:69:95:C9:74:97	Encendido
A-13	172.31.38.13	E0:69:95:C9:44:3C	Encendido
A-14	172.31.38.14	E0:69:95:C9:31:56	Encendido
A-15	172.31.38.15	E0:69:95:C9:35:3C	Encendido
A-16	172.31.38.16	E0:69:95:D8:12:47	Encendido
A-17	172.31.38.17	E0:69:95:C9:43:91	Encendido
A-18	172.31.38.18	E0:69:95:C9:44:17	Encendido
A-19	172.31.38.19	E0:69:95:D7:E7:51	Encendido
A-20	172.31.38.20	E0:69:95:C9:55:5B	Encendido
PROFESOR-A	172.31.38.101	E0:69:95:D7:55:1A	Encendido

Figura 4.26. PC de la SALA A en pantalla primaria

Editar Copiar Borrar A-01	SALA A	172.31.38.1	E06995D7E787
Editar Copiar Borrar A-02	SALA A	172.31.38.2	E06995D81E06
Editar Copiar Borrar A-03	SALA A	172.31.38.3	E06995C942A9
Editar Copiar Borrar A-04	SALA A	172.31.38.4	E06995C94383
Editar Copiar Borrar A-05	SALA A	172.31.38.5	E06995C9442A
Editar Copiar Borrar A-06	SALA A	172.31.38.6	E06995C94330
Editar Copiar Borrar A-07	SALA A	172.31.38.7	E06995C9741C
Editar Copiar Borrar A-08	SALA A	172.31.38.8	E06995C9845A
Editar Copiar Borrar A-09	SALA A	172.31.38.9	E06995C97477
Editar Copiar Borrar A-10	SALA A	172.31.38.10	E06995C9855F
Editar Copiar Borrar A-11	SALA A	172.31.38.11	E06995C9848F
Editar Copiar Borrar A-12	SALA A	172.31.38.12	E06995C97497
Editar Copiar Borrar A-13	SALA A	172.31.38.13	E06995C9443C
Editar Copiar Borrar A-14	SALA A	172.31.38.14	E06995C93156
Editar Copiar Borrar A-15	SALA A	172.31.38.15	E06995C9353C
Editar Copiar Borrar A-16	SALA A	172.31.38.16	E06995D81247
Editar Copiar Borrar A-17	SALA A	172.31.38.17	E06995C94391
Editar Copiar Borrar A-18	SALA A	172.31.38.18	E06995C94417
Editar Copiar Borrar A-19	SALA A	172.31.38.19	E06995D7E751
Editar Copiar Borrar A-20	SALA A	172.31.38.20	E06995C9555B

Figura 4.27. PC de la SALA A almacenados en la base de datos



Nombre	IP	MAC	Estado
E-01	172.31.38.81	E0:69:95:C8:15:F1	Encendido
E-02	172.31.38.82	E0:69:95:C8:16:0C	Encendido
E-03	172.31.38.83	E0:69:95:C8:37:8B	Encendido
E-04	172.31.38.84	E0:69:95:C8:0B:66	Encendido
E-05	172.31.38.85	E0:69:95:D1:ED:B9	Encendido
E-06	172.31.38.86	E0:69:95:C8:35:03	Encendido
E-07	172.31.38.87	E0:69:95:D1:ED:2C	Encendido
E-08	172.31.38.88	E0:69:95:C8:3D:46	Encendido
E-09	172.31.38.89	00:08:54:08:D2:AB	Encendido
E-10	172.31.38.90	E0:69:95:C8:16:35	Encendido
E-11	172.31.38.91	E0:69:95:C3:DB:55	Encendido
E-12	172.31.38.92	E0:69:95:C3:CF:C6	Encendido
E-13	172.31.38.93	E0:69:95:D7:55:C8	Encendido
E-14	172.31.38.94	E0:69:95:C8:13:3D	Encendido
E-15	172.31.38.95	E0:69:95:C8:16:44	Encendido
E-16	172.31.38.96	E0:69:95:C3:E0:C8	Encendido
E-17	172.31.38.97	E0:69:95:D1:ED:88	Encendido
E-18	172.31.38.98	E0:69:95:C3:C7:B7	Encendido
E-19	172.31.38.99	E0:69:95:D1:ED:10	Encendido
E-20	172.31.38.100	E0:69:95:C3:E6:BD	Encendido
PROFESOR-E	172.31.38.105	E0:69:95:C3:D5:B2	Encendido

Figura 4.28. PC de la SALA E en pantalla primaria

	comNombre	comGrupo	comIp	comMAC
Editar Copiar Borrar	E-01	SALA E	172.31.38.81	E06995C815F1
Editar Copiar Borrar	E-02	SALA E	172.31.38.82	E06995C8160C
Editar Copiar Borrar	E-03	SALA E	172.31.38.83	E06995C8378B
Editar Copiar Borrar	E-04	SALA E	172.31.38.84	E06995C80B66
Editar Copiar Borrar	E-05	SALA E	172.31.38.85	E06995D1EDB9
Editar Copiar Borrar	E-06	SALA E	172.31.38.86	E06995C83503
Editar Copiar Borrar	E-07	SALA E	172.31.38.87	E06995D1ED2C
Editar Copiar Borrar	E-08	SALA E	172.31.38.88	E06995C83D46
Editar Copiar Borrar	E-09	SALA E	172.31.38.89	001CC0DE6B22
Editar Copiar Borrar	E-10	SALA E	172.31.38.90	E06995C81635
Editar Copiar Borrar	E-11	SALA E	172.31.38.91	E06995C3DB55
Editar Copiar Borrar	E-12	SALA E	172.31.38.92	E06995C3CFC6
Editar Copiar Borrar	E-13	SALA E	172.31.38.93	E06995D755C8
Editar Copiar Borrar	E-14	SALA E	172.31.38.94	E06995C8133D
Editar Copiar Borrar	E-15	SALA E	172.31.38.95	E06995C81644
Editar Copiar Borrar	E-16	SALA E	172.31.38.96	E06995C3E0C8
Editar Copiar Borrar	E-17	SALA E	172.31.38.97	E06995D1ED88
Editar Copiar Borrar	E-18	SALA E	172.31.38.98	E06995C3C7B7
Editar Copiar Borrar	E-19	SALA E	172.31.38.99	E06995D1ED10
Editar Copiar Borrar	E-20	SALA E	172.31.38.100	E06995C3E6BD

Figura 4.29. PC de la SALA E almacenados en la base de datos

También la aplicación principal recibe datos desde la aplicación profesor como: usuarios registrados con perfil Profesor y solicitudes generadas. La **Figura 4.30** muestra la Captura de pantalla del usuario profesor en el formulario `Usuarios`, la **Figura 4.31** muestra la captura de pantalla del formulario `Solicitudes Activas` donde se puede ver la solicitud generada desde la aplicación profesor, la **Figura 4.32** muestra la captura de pantalla de del usuario profesor almacenado en la base de datos.

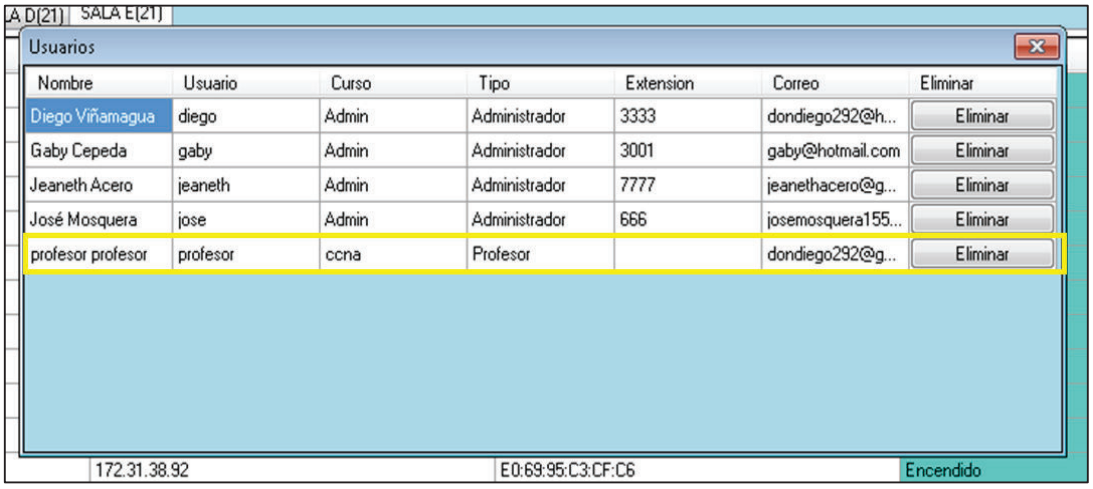


Figura 4.30. Usuario "profesor" en el formulario Usuarios

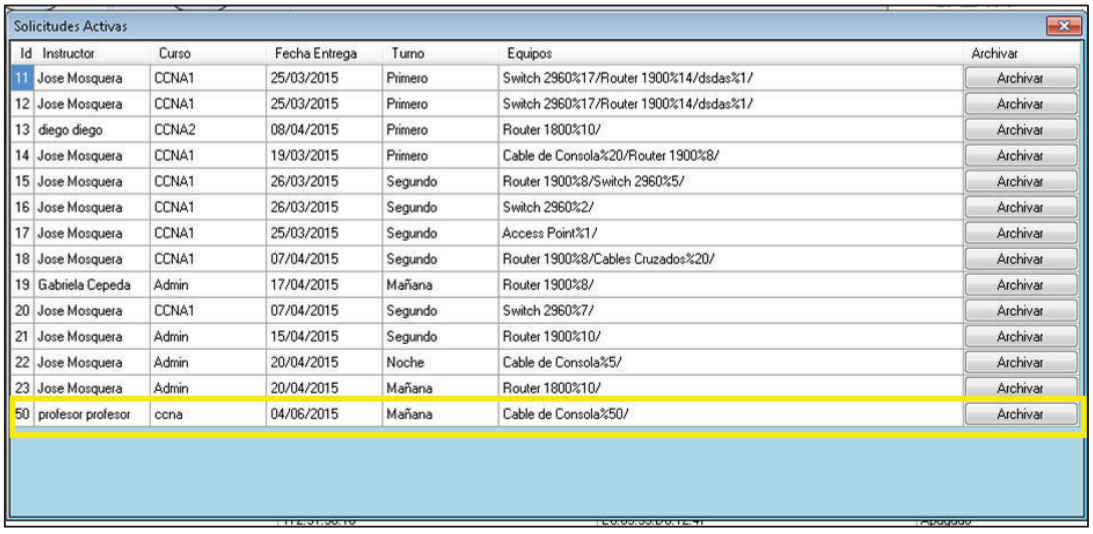
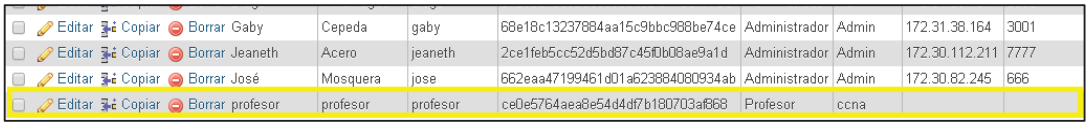


Figura 4.31. Formulario Solicitudes Activas



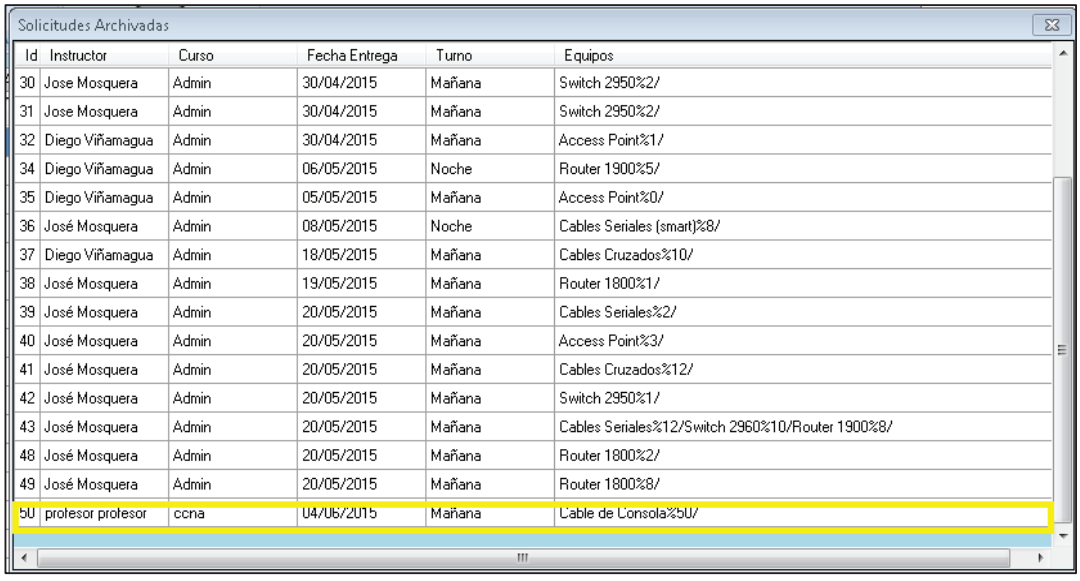
Editar Copiar Borrar Gaby	Cepeda	gaby	68e18c13237884aa15c9bbc988be74ce	Administrador	Admin	172.31.38.164	3001
Editar Copiar Borrar Jeaneth	Acero	jeaneth	2ce1feb5cc52d5bd87c45f0b08ae9a1d	Administrador	Admin	172.30.112.211	7777
Editar Copiar Borrar José	Mosquera	jose	662eaa47199461d01a623884080934ab	Administrador	Admin	172.30.82.245	666
Editar Copiar Borrar profesor	profesor	profesor	ce0e5764aea8e54d4df7b180703af868	Profesor	ccna		

Figura 4.32. Usuario "profesor" almacenado en la base de datos

4.3.2.1.2 Prueba de archivar solicitud e imprimir

La prueba consiste en archivar una solicitud activa e imprimirla.

Al presionar el botón archivar en el formulario `Solicitudes Activas` estas cambian de estado y se las puede visualizar en el formulario `Solicitudes Archivadas`. Se tiene la posibilidad de ver el detalle de cada una de las solicitudes; al dar doble clic sobre estas se abre el formulario `Ver Solicitud` en el cual se tiene la opción de Imprimir. Para esta prueba se utilizó la impresión en PDF. La **Figura 4.33** muestra la captura de pantalla del formulario `Solicitudes Archivadas`, la **Figura 4.34** muestra la captura de pantalla del formulario `Ver Solicitud`, la **Figura 4.35** muestra la captura de pantalla de la solicitud impresa en PDF.



Solicitudes Archivadas

Id	Instructor	Curso	Fecha Entrega	Turno	Equipos
30	Jose Mosquera	Admin	30/04/2015	Mañana	Switch 2950%2/
31	Jose Mosquera	Admin	30/04/2015	Mañana	Switch 2950%2/
32	Diego Viñamagua	Admin	30/04/2015	Mañana	Access Point%1/
34	Diego Viñamagua	Admin	06/05/2015	Noche	Router 1900%5/
35	Diego Viñamagua	Admin	05/05/2015	Mañana	Access Point%0/
36	José Mosquera	Admin	08/05/2015	Noche	Cables Seriales (smart)%8/
37	Diego Viñamagua	Admin	18/05/2015	Mañana	Cables Cruzados%10/
38	José Mosquera	Admin	19/05/2015	Mañana	Router 1800%1/
39	José Mosquera	Admin	20/05/2015	Mañana	Cables Seriales%2/
40	José Mosquera	Admin	20/05/2015	Mañana	Access Point%3/
41	José Mosquera	Admin	20/05/2015	Mañana	Cables Cruzados%12/
42	José Mosquera	Admin	20/05/2015	Mañana	Switch 2950%1/
43	José Mosquera	Admin	20/05/2015	Mañana	Cables Seriales%12/Switch 2960%10/Router 1900%8/
48	José Mosquera	Admin	20/05/2015	Mañana	Router 1800%2/
49	José Mosquera	Admin	20/05/2015	Mañana	Router 1800%8/
50	profesor profesor	ccna	04/06/2015	Mañana	Cable de Consola%50/

Figura 4.33. Formulario Solicitudes Archivadas

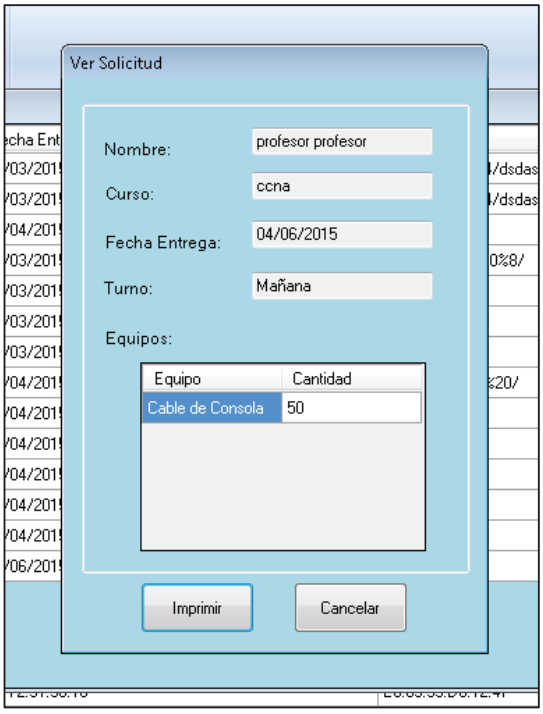


Figura 4.34. Formulario Ver Solicitud

SOLICITUD PARA PRESTAMO DE EQUIPOS

Nombre Instructor: profesor profesor
Curso a Cargo: ccna
Fecha de Reserva: 04/06/2015
Detalle de Equipos y Elementos:

Equipo	Cantidad
Cable de Consola	50

Me comprometo a hacer que los alumnos utilicen los equipos de la mejor manera y que entreguen de forma ordenada todos los elementos adicionales prestados.

profesor profesor

Figura 4.35. Solicitud impresa en PDF

4.3.2.1.3 Prueba de registro de usuario con perfil administrador

La prueba consiste en agregar un usuario con perfil Administrador y comprobar que se validen los campos del formulario.

En el formulario `Agregar Usuario` se debe completar todos los campos, caso contrario se informará de los campos vacíos, además el correo electrónico debe tener el formato correcto. La **Figura 4.36** muestra la captura de pantalla del formulario agregar usuario con campos incompletos, la **Figura 4.37** muestra la captura de pantalla del formulario agregar usuario con correo electrónico incorrecto, la **Figura 4.38** muestra la captura de pantalla del mensaje de error de extensión, la **Figura 4.39** muestra la captura de pantalla del mensaje de error de usuario o correo electrónico, la **Figura 4.40** muestra la captura de pantalla del usuario administrador en el formulario Usuarios y la **Figura 4.41** muestra la captura de pantalla de del usuario profesor almacenado en la base de datos.

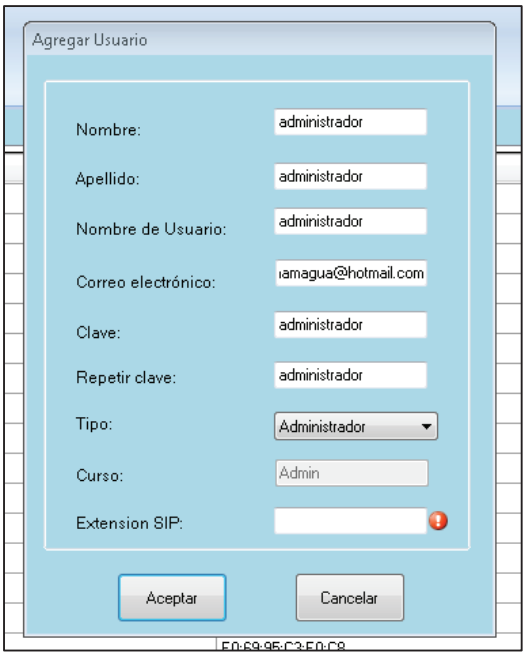


Figura 4.36. Formulario Agregar Usuario con campos incompletos

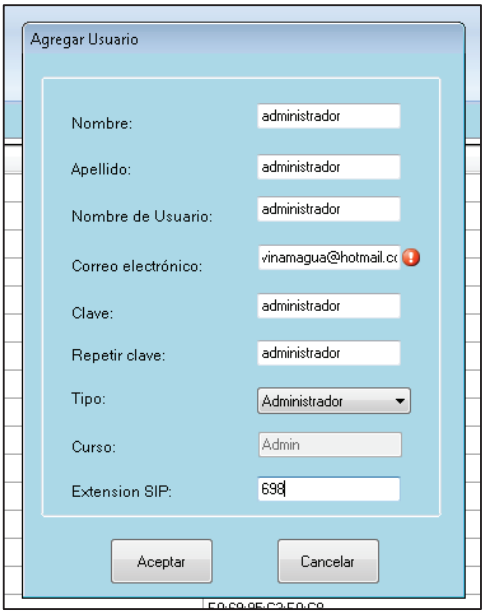


Figura 4.37. Formulario Agregar Usuario con correo electrónico incorrecto

Figura 4.38. Mensaje de error "Extensión SIP ya registrada"

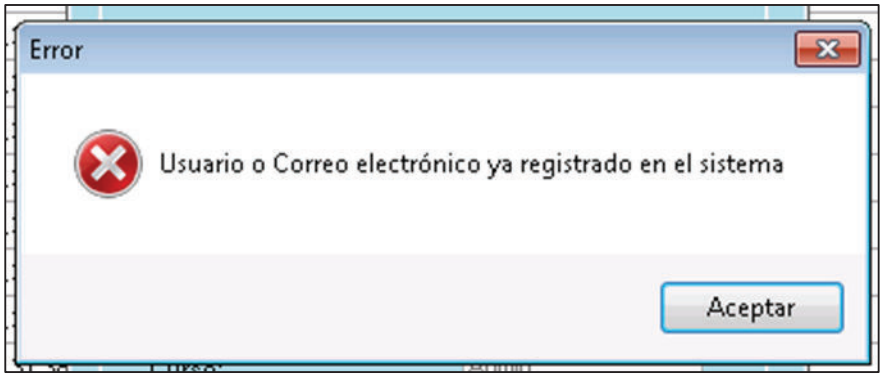


Figura 4.39. Mensaje de error "Usuario o Correo electrónico ya registrado en el sistema"

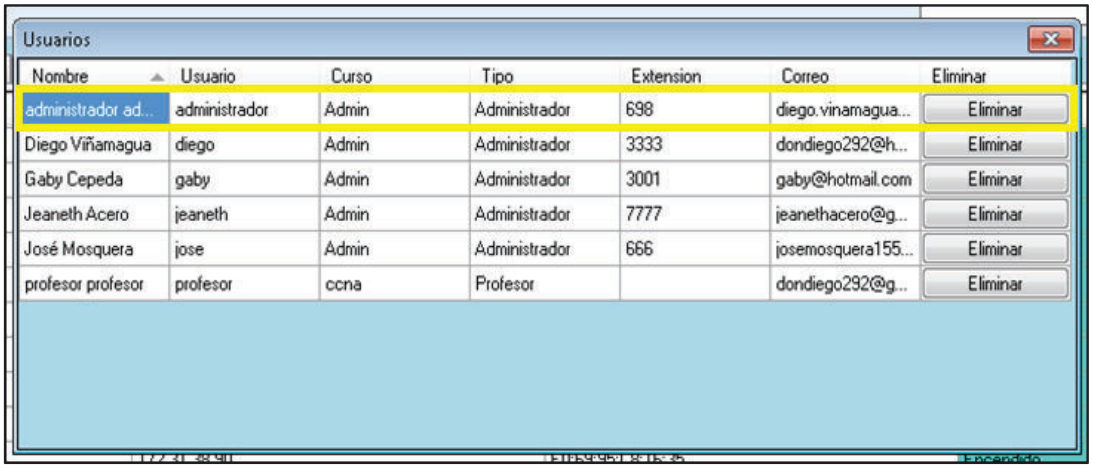


Figura 4.40. Usuario "administrador" en el formulario Usuarios

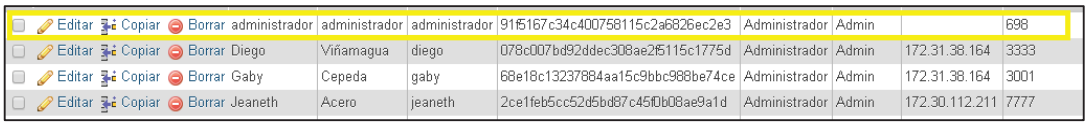


Figura 4.41. Usuario "administrador" almacenado en la base de datos

4.3.2.2 Aplicación profesor

La prueba de instalación de la aplicación profesor funcionó de la misma manera que en el ambiente controlado.

4.3.2.2.1 Prueba de comunicación con aplicación cliente

La prueba consiste en explorar los PC que tengan instalada la aplicación cliente y que pertenezcan al mismo grupo de trabajo; además se bloquearán, desbloquearán y finalmente se accederá remotamente a los PC.

Como se mencionó anteriormente, para comunicarse con la aplicación cliente por primera vez es necesario presionar el botón `Explorar`. La **Figura 4.42** muestra la captura de pantalla de la aplicación profesor en la SALA A después de presionar el botón explorar, la **Figura 4.43** muestra la captura de pantalla de la aplicación profesor en la SALA E después de presionar el botón explorar.

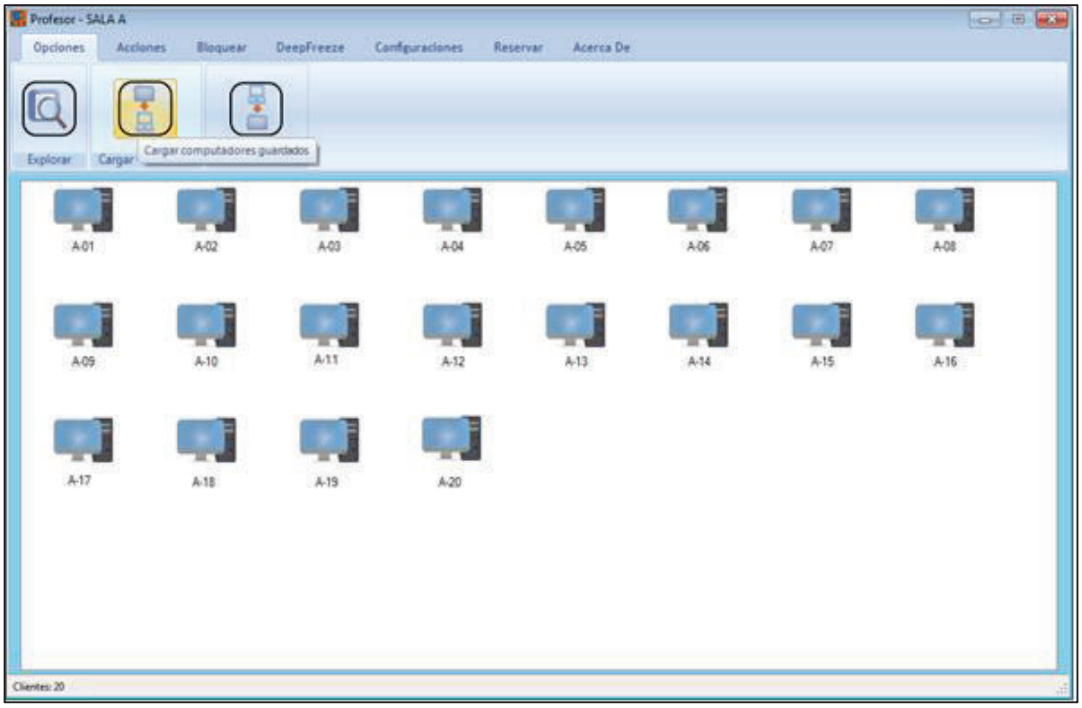


Figura 4.42. Aplicación profesor en la SALA A después de presionar el botón explorar

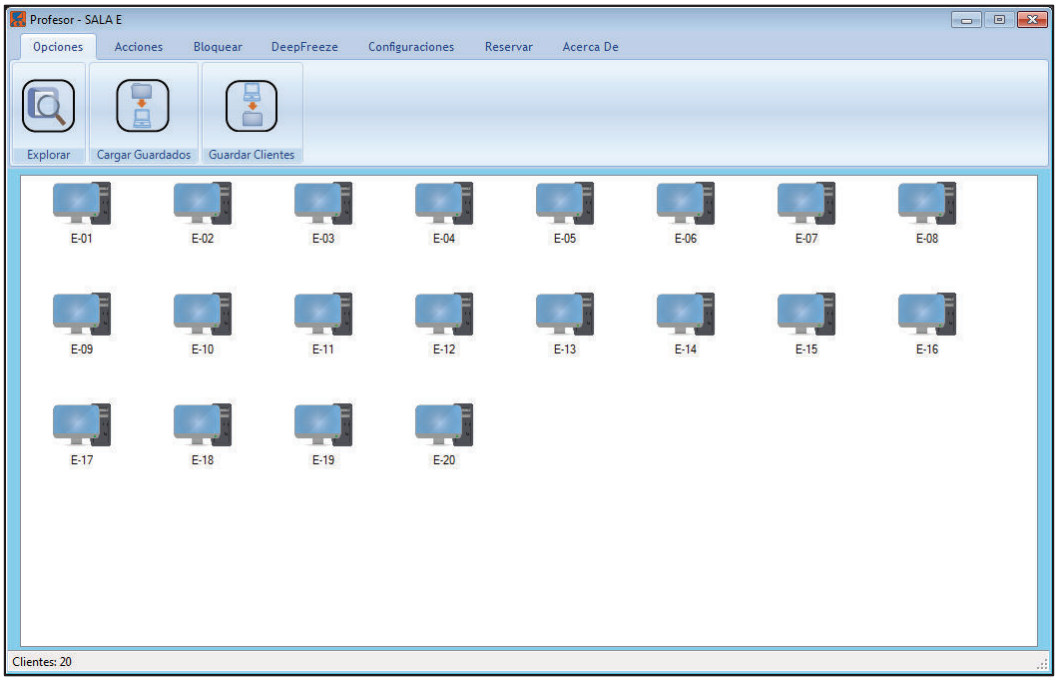


Figura 4.43. Aplicación profesor en la SALA E después de presionar el botón explorar

Una vez agregados los clientes se pueden realizar las siguientes acciones: Apagar, Encender, Bloquear, Desbloquear, Reiniciar, Reiniciar en modo *Frozen* y Reiniciar en modo *Thawed*.

Para realizar estas acciones; en primer, lugar es necesario seleccionar al menos un PC. La **Figura 4.44** muestra la captura de pantalla de los PC seleccionados, la **Figura 4.45** muestra la **c**aptura de pantalla del mensaje de error de selección, esto se produce cuando no existen PC para realizar la acción requerida.

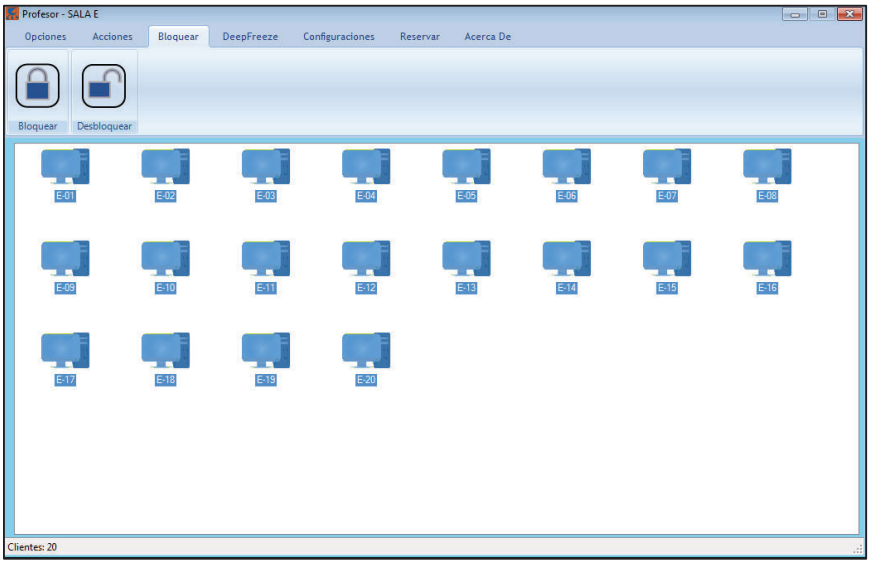


Figura 4.44. PC seleccionados en la sala E

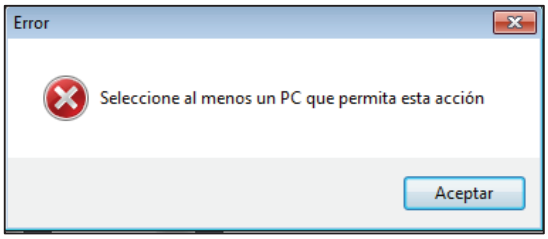


Figura 4.45. Mensaje de error "Seleccione al menos un PC que permita esta acción"

Posteriormente se puede presionar el botón `Bloquear` y aparecerá un mensaje de confirmación. La **Figura 4.46** muestra la captura de pantalla del mensaje de confirmación para Bloquear.

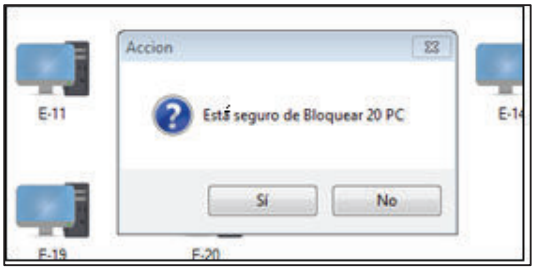


Figura 4.46. Mensaje de confirmación "Está seguro de Bloquear"

En caso de una respuesta afirmativa del usuario, los PC que fueron seleccionados proceden a bloquearse. La **Figura 4.47** muestra la captura de pantalla de los PC bloqueados.

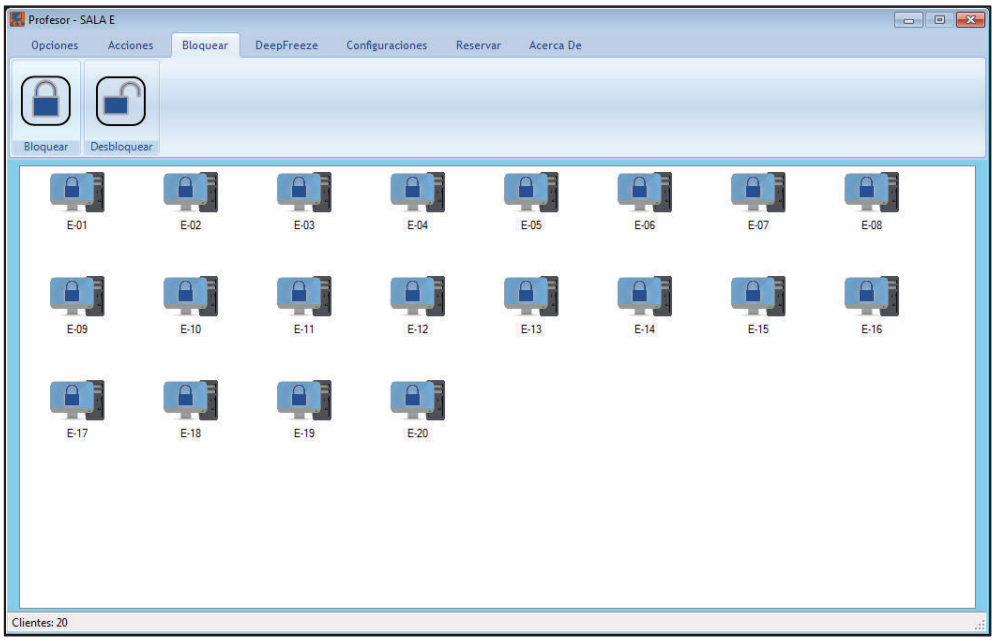


Figura 4.47. PC bloqueados en la sala E

Para desbloquear se deben seleccionar igualmente los PC y presionar el botón `Desbloquear` y nuevamente aparece el mensaje de confirmación. La **Figura 4.48** muestra la captura de pantalla del mensaje de confirmación para Desbloquear.

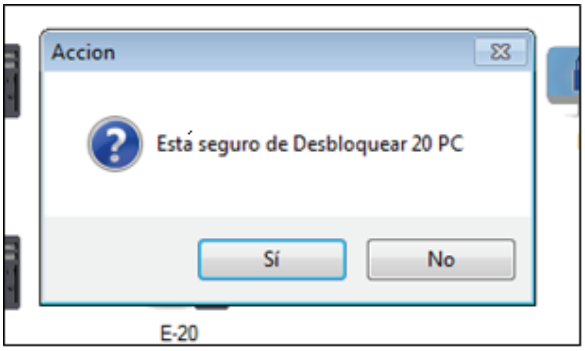


Figura 4.48. Mensaje de confirmación "Está seguro de Desbloquear"

Las acciones Apagar, Encender, Reiniciar, Reiniciar en modo *Frozen* y Reiniciar en modo *Thawed*, funcionaron correctamente.

Para el acceso remoto se hace doble clic sobre los componentes de `ListView`, en este momento se intenta una conexión; en caso de ser positiva se despliega una ventana en donde se puede ver y controlar el escritorio remoto. La **Figura 4.49** muestra la captura de pantalla del acceso remoto al PC E-01.

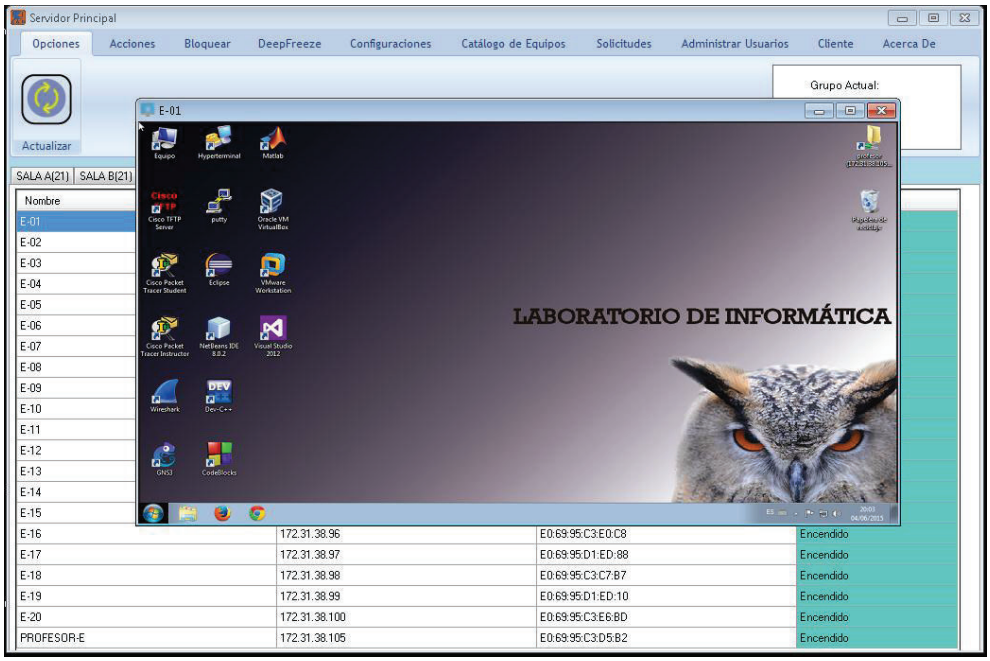


Figura 4.49. Acceso remoto al PC E-01

En caso de que el intento sea fallido se presenta un mensaje al usuario. La **Figura 4.50** muestra la captura de pantalla del mensaje de error.

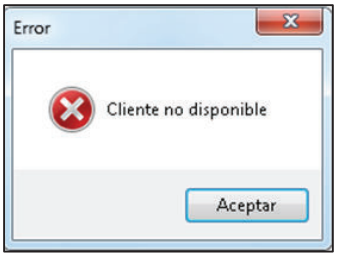


Figura 4.50. Mensaje de error "Cliente no disponible"

En los PC ubicados en la SALA A se realizaron las mismas pruebas mencionadas anteriormente sin problema alguno.

4.3.2.2.2 Prueba de registro de usuario

La prueba consiste en registrar un usuario con perfil Profesor.

Para esto se debe ir a la sección `Reservar`, se abre el formulario de `Autenticación`, donde se tiene la opción de registrar el usuario; si los campos están en blanco se informa de estos. Una vez que los campos están completos se presiona el botón aceptar y un correo electrónico es enviado con la clave generada automáticamente. La **Figura 4.51** muestra la captura de pantalla del formulario de registro con los campos incompletos, la **Figura 4.52** muestra la captura de pantalla del registro con los campos completos, la **Figura 4.53** muestra la captura de pantalla del mensaje de información de envío de correo electrónico, la **Figura 4.54** muestra la captura de pantalla del correo electrónico recibido.

Figura 4.51. Formulario de registro con campos incompletos

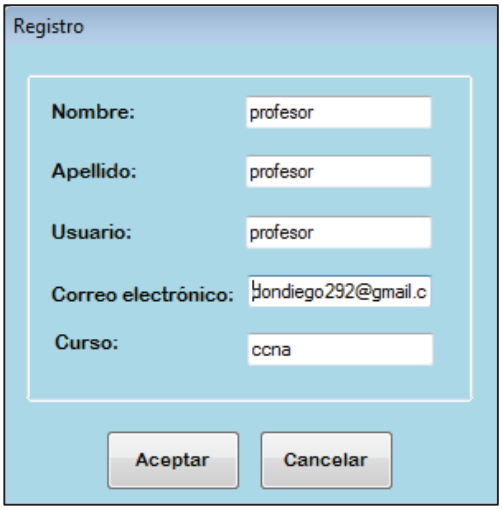


Figura 4.52. Formulario registro con campos completos

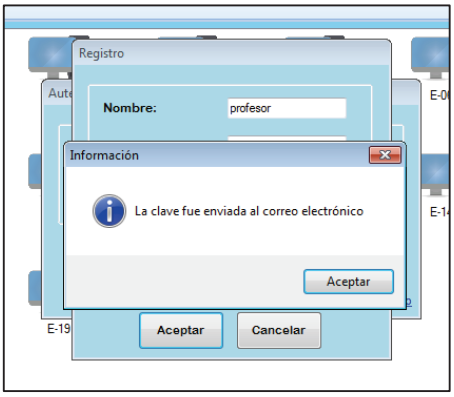


Figura 4.53. Mensaje de información "La clave fue enviada al correo electrónico"

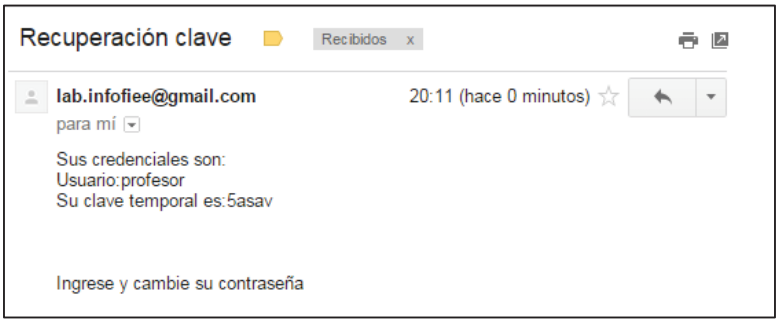


Figura 4.54. Correo electrónico recibido

4.3.2.2.3 Prueba de autenticación

La prueba consiste en tratar de autenticar un usuario cuando la aplicación principal no está activa, cuando los datos son incorrectos, cuando los campos están vacíos y finalmente con un usuario correcto.

El formulario `Autenticación` antes de intentar una comunicación con la aplicación principal comprueba que los campos estén llenos. La **Figura 4.55** muestra la captura de pantalla del mensaje Ingrese datos, la **Figura 4.56** muestra la captura de pantalla del mensaje Servidor no encontrado, la **Figura 4.57** muestra la captura de pantalla del mensaje Datos incorrectos, la **Figura 4.58** muestra la captura de pantalla del formulario Autenticación con datos correctos.

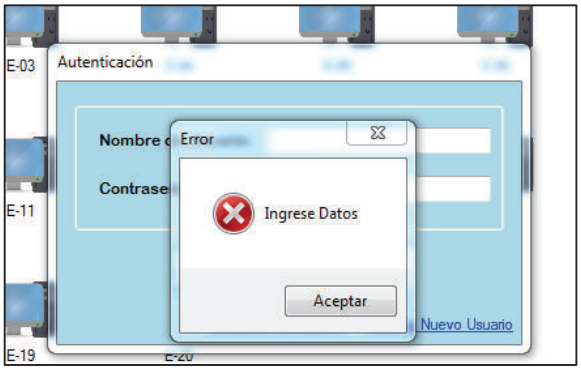


Figura 4.55. Mensaje de error "Ingrese datos"

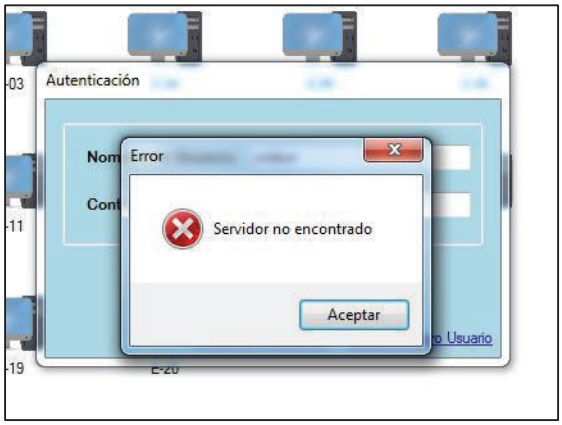


Figura 4.56. Mensaje de error "Servidor no encontrado"

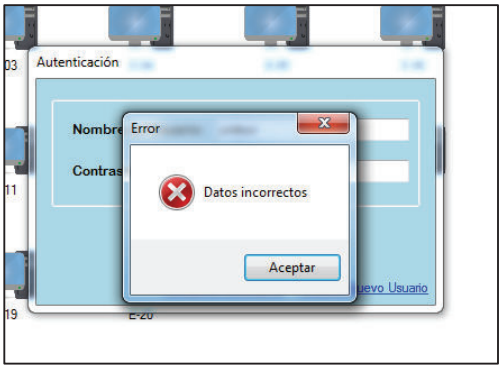


Figura 4.57. Mensaje de error "Datos incorrectos"

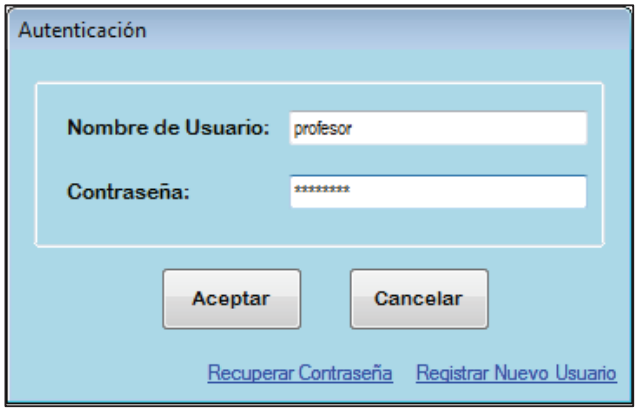


Figura 4.58. Formulario Autenticación con datos correctos

4.3.2.2.4 Prueba para generar una nueva solicitud

La prueba consiste en ingresar una cantidad no disponible de equipos y finalmente enviar la una solicitud.

Una vez superada la fase de autenticación se presenta el formulario `Préstamo de Equipos`; por defecto está deshabilitado el botón añadir, es necesario escribir un valor en el campo `Cantidad Deseada`, si se ingresa una cantidad superior a la disponible se informa al usuario. La **Figura 4.59** muestra la captura de pantalla del formulario de préstamo de equipos, la **Figura 4.60** muestra la captura de pantalla del mensaje Cantidad no disponible, la **Figura 4.61** muestra la captura de pantalla del resumen de la solicitud y la **Figura 4.62** muestra la captura de pantalla del mensaje Solicitud enviada con Éxito.

Figura 4.59. Formulario de préstamo de equipos

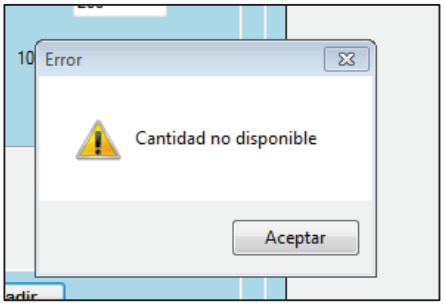


Figura 4.60. Mensaje de advertencia "Cantidad no disponible"

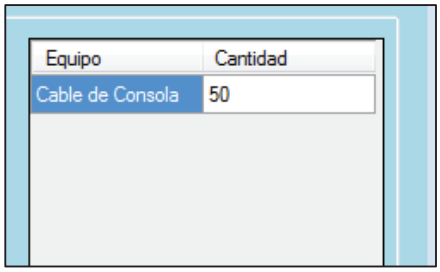


Figura 4.61. Resumen de la solicitud

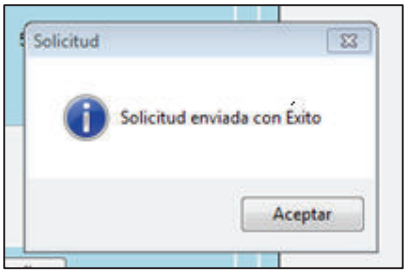


Figura 4.62. Mensaje de información "Solicitud enviada con Éxito"

4.3.2.3 Aplicación Cliente

La prueba de instalación funcionó de la misma manera que en el ambiente controlado.

4.3.2.3.1 Prueba de bloqueo

La prueba consiste en bloquear los PC de la sala E en los cuales está instalada la aplicación cliente. La **Figura 4.63** muestra la fotografía de la Sala E con los PC bloqueados.

Figura 4.63. Fotografía de la Sala E con los PC bloqueados.

4.3.2.4 Aplicación Android

La aplicación Android al usar SIP debe correr sobre sistema operativo Android versión 2.3.0 como mínimo; sin embargo, para las pruebas realizadas se usó la versión 4.4.2, la aplicación usa aproximadamente 4 MB de memoria en el dispositivo Android.

4.3.2.4.1 Prueba de autenticación

Esta prueba consistirá en ingresar al sistema con una IP incorrecta, intentar ingresar con un usuario que no existe y con un usuario válido. La **Figura 4.64** muestra la captura de pantalla del mensaje de `Servidor no encontrado`, la **Figura 4.65** muestra la captura de pantalla del mensaje de `Usuario no encontrado`, la **Figura 4.66** muestra la captura de pantalla de autenticación correcta.

Figura 4.64. Mensaje de "Servidor no encontrado"

Figura 4.65. Mensaje de "Usuario no encontrado"

Figura 4.66. Mensaje de autenticación correcta

4.3.2.4.2 Prueba de recepción de datos

La prueba consiste en visualizar: los grupos del sistema, los pc del sistema y las solicitudes activas que son enviadas desde la aplicación principal. La **Figura 4.67** muestra captura de pantalla de los grupos recibidos, la **Figura 4.68** muestra la captura de pantalla de los PC recibidos, la **Figura 4.69** muestra la captura de pantalla de las solicitudes recibidas.

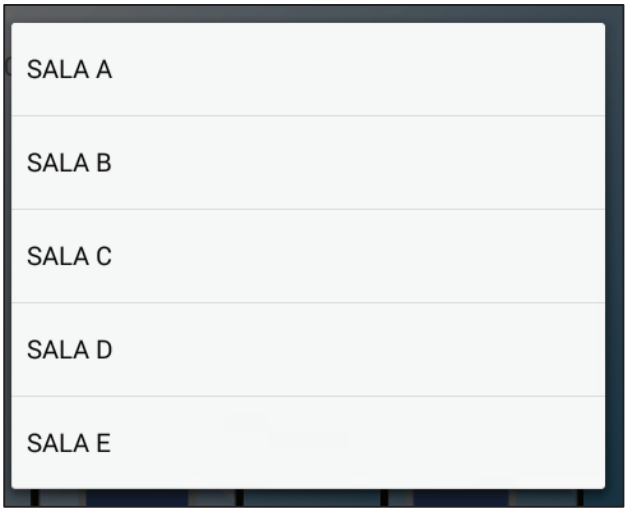


Figura 4.67. Grupos recibidos

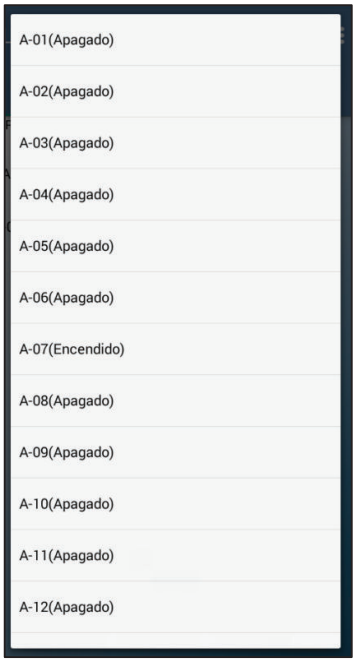


Figura 4.68. PC recibidos

4.3.2.4.3 Prueba de recepción de notificaciones y visualización de solicitudes

La prueba consiste en visualizar la notificación que envía la aplicación principal. La **Figura 4.70** muestra la captura de pantalla de la notificación recibida, la **Figura 4.71** muestra la captura de pantalla del detalle de la solicitud

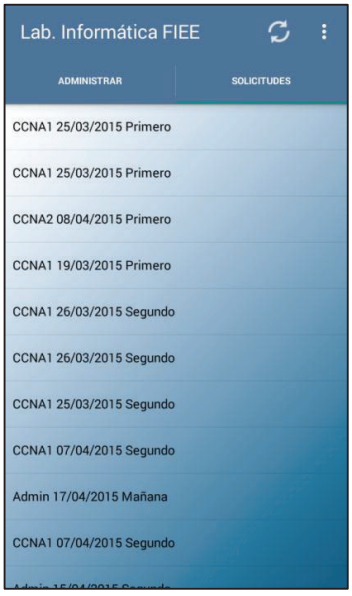


Figura 4.69. Solicitudes recibidas

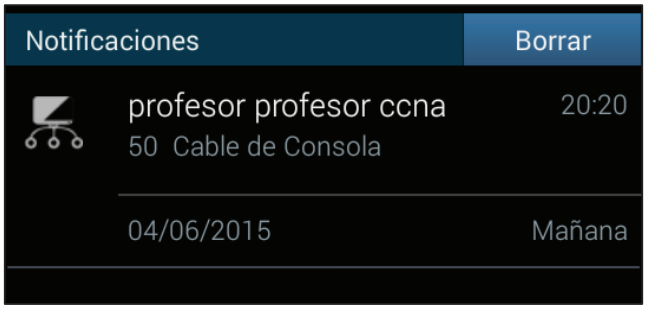


Figura 4.70. Notificación recibida

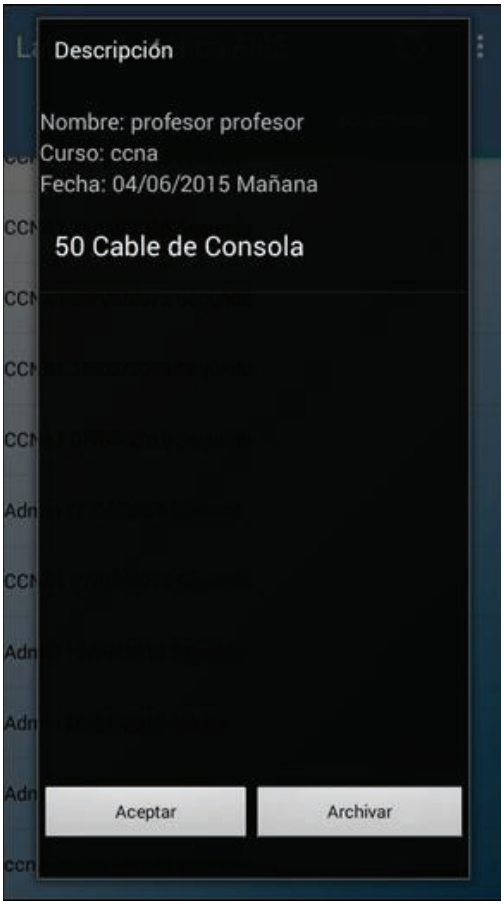


Figura 4.71. Detalle de la solicitud

4.3.2.4.4 Prueba de llamada

La prueba consiste en registrarse en el servidor SIP y visualizar los usuarios existentes. La **Figura 4.72** muestra la captura de pantalla del estado `Registrando con servidor SIP`, la **Figura 4.73** muestra la captura de

pantalla de los usuarios disponibles para llamadas SIP, la **Figura 4.74** muestra la captura de pantalla del estado `Listo para llamadas`.

Figura 4.72. Estado "Registrando con servidor SIP"

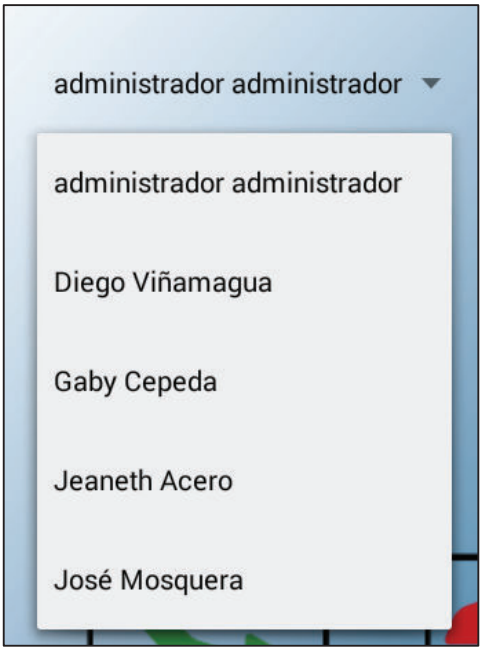


Figura 4.73. Usuarios disponibles para llamadas SIP

Figura 4.74. Captura de pantalla del estado listo para llamadas

4.4 ERRORES ENCONTRADOS Y CORREGIDOS DURANTE LA ETAPA DE PRUEBAS

En las pruebas realizadas se encontraron los siguientes errores:

4.4.1 ERROR DE MICROSOFT .NET FRAMEWORK 4.0

4.4.1.1 Problema

Al no tener el Microsoft .NET Framework 4.0 necesario instalado en el PC que ejecuta la aplicación se produce este error. La **Figura 4.75** muestra la captura de pantalla del error de Microsoft .NET Framework 4.0.

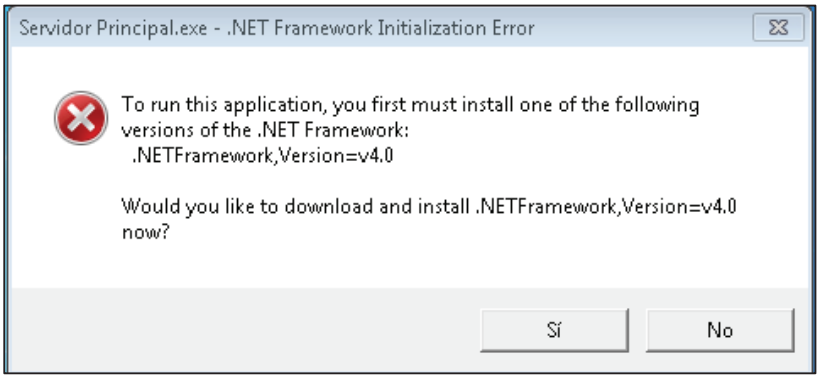


Figura 4.75. Error de Microsoft .NET Framework 4.0

4.4.1.2 Solución

Se debe instalar el Microsoft .NET Framework 4.0 previo a la instalación de las aplicaciones.

4.4.2 ERROR DE BASE DE DATOS

4.4.2.1 Problema

Se produce una excepción no controlada al momento de ejecutar la aplicación principal y no tener datos almacenados en la tabla computadores. La **Figura 4.76** muestra la captura de pantalla con el error de base de datos en blanco.

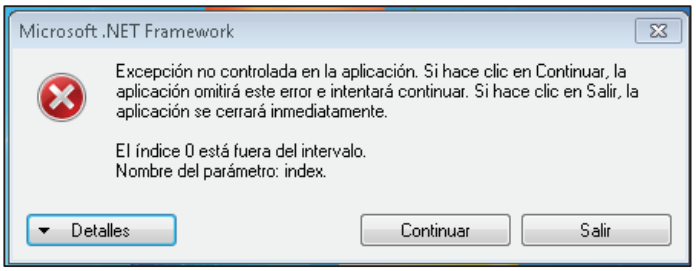


Figura 4.76. Error de base de datos en blanco

4.4.2.2 Solución

Se controló la excepción informando al usuario que es necesario ejecutar la aplicación profesor. La **Figura 4.77** muestra la captura de pantalla del mensaje de información.

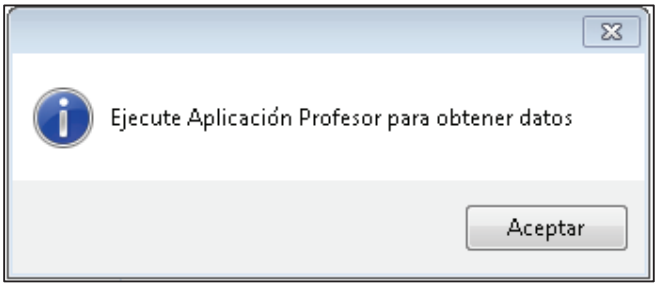


Figura 4.77. Mensaje de información "Ejecute Aplicación Profesor para obtener datos".

4.4.3 ERROR DE VISUALIZACIÓN EN ACCESO REMOTO

4.4.3.1 Problema

Al hacer pruebas en los PC portátiles del laboratorio de la FIEE se encontró que existe un problema de visualización. La **Figura 4.78** muestra la captura de pantalla error en la visualización del acceso remoto

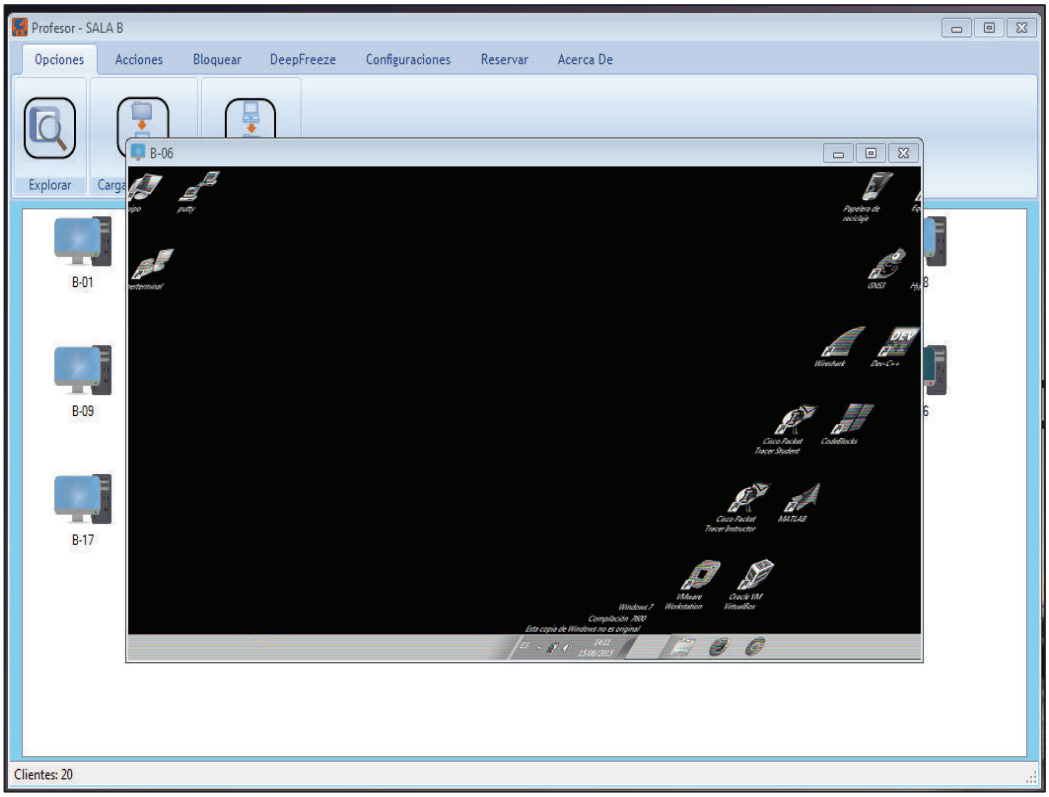


Figura 4.78. Error en la visualización del acceso remoto

4.4.3.2 Solución

Se cambió la resolución de pantalla en los PC portátiles a 1360x768 pixeles y con esto se corrigió el error. La **Figura 4.79** muestra la captura de pantalla de visualización correcta del acceso remoto.

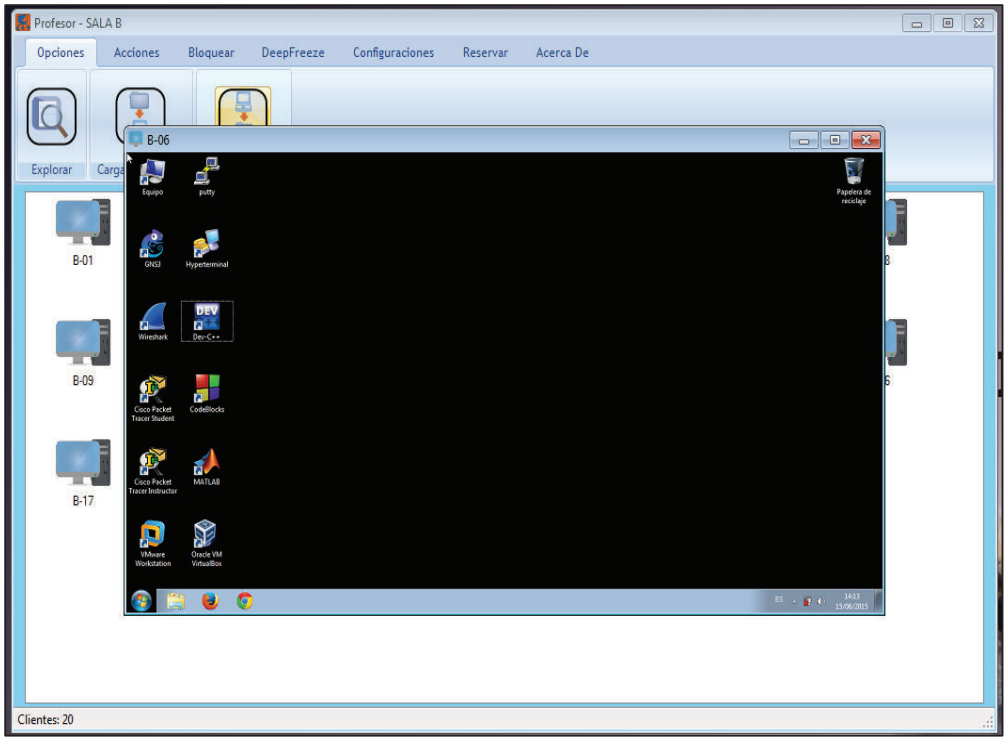


Figura 4.79. Visualización correcta del acceso remoto

4.5 PRUEBAS DE ACEPTACIÓN

Las pruebas de aceptación se realizan tomando en cuenta las historias de usuario con el fin de comprar que los requerimientos fueron realizados con éxito.

La **Figura 4.80** muestra el formato que se usará para el número en las pruebas de aceptación.

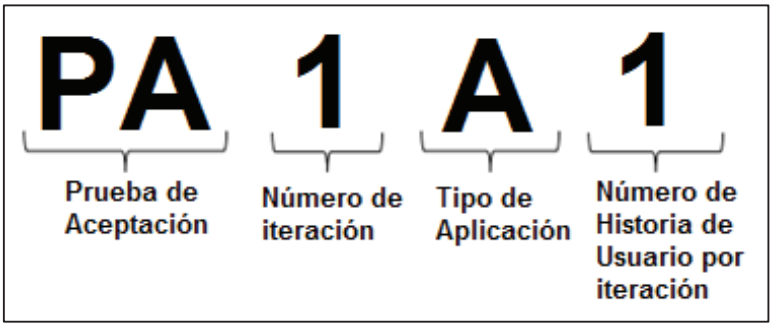


Figura 4.80. Formato del número para pruebas de aceptación

Se tiene las siguientes aplicaciones: (A) aplicación Android, (B) base de datos, (C) aplicación cliente, (P) aplicación profesor, (S) aplicación principal.

4.5.1.1 Primera iteración

En esta sección se indican las pruebas de aceptación que permitan un funcionamiento mínimo del sistema. La **Tabla 4.3** muestra la prueba de aceptación "Creación de base de datos", la **Tabla 4.4** muestra la prueba de aceptación "Autenticación de usuarios", la **Tabla 4.5** muestra la prueba de aceptación "Registro de nuevos usuarios", la **Tabla 4.6** muestra la prueba de aceptación "Recuperación de contraseña" y la **Tabla 4.7** muestra la prueba de aceptación "Registro de nuevos ítems en el catálogo de equipos".

Prueba de Aceptación	
Número:	PA1B1
Historia de Usuario:	HU1B1 - Diseño de la base de datos
Nombre:	Diseño de la base de datos
Descripción:	Se desarrolló una base de datos para almacenar la información que requiere el sistema.
Condiciones de Ejecución:	Acceso a la base de datos desde la interfaz web de *phpMyAdmin* y además establecer una conexión con la aplicación principal.
Pasos de Ejecución:	Acceder a la base de datos mediante *phpMyAdmin* y la aplicación principal
Resultado Esperado:	La aplicación principal podrá hacer uso de los servicios de la base de datos
Evaluación de la prueba:	Se alcanzó el resultado esperado

Tabla 4.3. Prueba de aceptación - Creación de base de datos

Prueba de Aceptación	
Número:	PA1S2
Historia de Usuario:	HU1S2 - Autenticación de usuarios
Nombre:	Autenticación de usuarios
Descripción:	Los usuarios acceden al sistema con un nombre de usuario y contraseña, validando estos datos el usuario establece una sesión. El inicio de sesión es obligatorio para acceder a la aplicación Android y a ciertas funcionalidades de la aplicación Profesor
Condiciones de Ejecución:	Conexión con el servidor principal Debe existir un usuario registrado en la base de datos En la aplicación profesor la autenticación es para ciertas funcionalidades como: prestamos de equipos y *deep freeze*.
Pasos de Ejecución:	Se ejecuta a la aplicación Android En la pantalla de autenticación se ingresa usuario y contraseña.
Resultado Esperado:	Se accede a la aplicación y se crea una sesión con el usuario ingresado.
Evaluación de la prueba:	Se alcanzó el resultado esperado

Tabla 4.4. Prueba de aceptación - Autenticación de usuarios

Prueba de Aceptación	
Número:	PA1S3
Historia de Usuario:	HU1S3 - Registro de nuevos usuarios
Nombre:	Registro de nuevos usuarios
Descripción:	Se registra nuevos usuarios tanto con perfil administrador como profesor
Condiciones de Ejecución:	Desde la aplicación principal se puede registrar nuevos usuarios tanto con perfil profesor como administrador y desde la aplicación profesor los usuarios pueden auto registrarse y el usuario se crea con perfil profesor.
Pasos de Ejecuciones:	En la aplicación principal se ingresa en la opción Administrar Usuarios y luego en agregar usuarios Para la aplicación profesor en la opción reservar equipos en la ventana de autenticación existe la opción Registrar usuario.
Resultado Esperado:	Se crea un nuevo usuario en la tabla usuarios con la información ingresada
Evaluación de la prueba:	Se alcanzó el resultado esperado

Tabla 4.5. Prueba de aceptación - Registro de nuevos usuarios

Prueba de Aceptación	
Número:	PA1P4
Historia de Usuario:	HU1P4 - Recuperación de contraseña
Nombre:	Recuperación de contraseña
Descripción:	En caso de olvido de contraseña se recibe un correo con una nueva contraseña
Condiciones de Ejecución:	El usuario que desee recuperar la contraseña previamente debe estar registrado en el sistema.
Pasos de Ejecuciones:	Ingresar en la aplicación profesor en la sección Reserva Reservar equipos Opción recuperar contraseña Ingresar el correo con el que se registró en el sistema Se recibe un correo electrónico con la contraseña
Resultado Esperado:	El usuario recibirá un correo electrónico con una nueva contraseña para ingresar al sistema
Evaluación de la prueba:	Se alcanzó el resultado esperado

Tabla 4.6. Prueba de aceptación - Recuperación de contraseña

Prueba de Aceptación	
Número:	PA1S5
Historia de Usuario:	HU1S5 - Registro de nuevos ítems en el catálogo de equipos
Nombre:	Registro de nuevos ítems en el catálogo de equipos
Descripción:	Se implementó en la aplicación principal la opción de registrar nuevos ítems en el catálogo de equipos.
Condiciones de Ejecución:	Esta opción está disponible desde la aplicación principal Se debe tener conexión con la base de datos
Pasos de Ejecución:	Opción Catalogo de equipos Agregar Llenar los campos para completar el registro del nuevo ítem.
Resultado Esperado:	Poder visualizar el nuevo ítem registrado en la base de datos
Evaluación de la prueba:	Se alcanzó el resultado esperado

Tabla 4.7. Prueba de aceptación - Registro de nuevos ítems en el catálogo de equipos

4.5.1.2 Segunda iteración

En la segunda iteración se encuentran las pruebas de aceptación que permiten la gestión de PC. La **Tabla 4.8** muestra la prueba de aceptación "Acción encender", la **Tabla 4.9** muestra la prueba de aceptación "Acción apagar", la **Tabla 4.10** muestra la prueba de aceptación "Acción reiniciar", la **Tabla 4.11** muestra la prueba de aceptación "Acción bloquear", la **Tabla 4.12** muestra la prueba de aceptación "Acción desbloquear", la **Tabla 4.13** muestra la prueba de aceptación "Control acceso remoto" y la **Tabla 4.14** muestra la prueba de aceptación "Obtener información".

Prueba de Aceptación	
Número:	PA2C1
Historia de Usuario:	HU2C1 - Acción encender
Nombre:	Acción encender
Descripción:	Mediante esta acción se puede encender uno o varios equipos dentro de la red del laboratorio
Condiciones de Ejecución:	Debe estar ejecutándose la aplicación cliente en los PC que se desee encender La opción para realizar esta opción está disponible desde la aplicación Android, aplicación profesor y aplicación principal.
Pasos de Ejecuciones:	En la aplicación Android se ingresa a la aplicación con un usuario registrado en el sistema, se selecciona la sala y los PC a los cuales se desee hacer la acción y finalmente se presiona el ícono con la opción encender En la aplicación Principal se selecciona el PC al cual se desee hacer la acción y dando clic derecho se muestra un menú en el cual se da clic en la opción encender si se desea hacerlo de manera individual; en el caso de querer realizar la acción de toda la sala se da clic en la opción Acciones y se da clic en el ícono de encender y la acción se realizará en la sala actual. En la aplicación profesor se debe seleccionar los íconos de los PC que se desee realizar la acción y en la opción Acciones se da clic en el icono encender.
Resultado Esperado:	Encender uno o los PC seleccionados
Evaluación de la prueba:	Se alcanzó el resultado esperado

Tabla 4.8. Prueba de aceptación - Acción encender

Prueba de Aceptación	
Número:	PA2C2 - Acción apagar
Historia de Usuario:	HU2C2 - Acción apagar
Nombre:	Acción apagar
Descripción:	Mediante esta acción se puede apagar uno o varios equipos dentro de la red del laboratorio
Condiciones de Ejecución:	Debe estar ejecutándose la aplicación cliente en los PC que se desee apagar La opción para realizar esta opción está disponible desde la aplicación Android, aplicación profesor y aplicación principal.
Pasos de Ejecuciones:	En la aplicación Android se ingresa a la aplicación con un usuario registrado en el sistema; se selecciona la sala y los PC a los cuales se desee hacer la acción y finalmente se presiona el ícono con la opción apagar En la aplicación Principal se selecciona el PC al cual se desee hacer la acción y dando clic derecho se muestra un menú en el cual se da clic en la opción apagar si se desea hacerlo de manera individual; en el caso de querer realizar la acción de toda la sala se da clic en la opción Acciones y se da clic en el ícono de apagar y la acción se realizará en la sala actual. En la aplicación profesor se debe seleccionar los íconos de los PC que se desee realizar la acción y en la opción Acciones se da clic en el icono encender.
Resultado Esperado:	Apagar uno o los PC seleccionados
Evaluación de la prueba:	Se alcanzó el resultado esperado

Tabla 4.9. Prueba de aceptación - Acción apagar

4.5.1.3 Tercera iteración

En la tercera iteración se encuentran las pruebas de aceptación que permiten el acceso a la base de datos. La **Tabla 4.15** muestra la prueba de aceptación "Gestión de usuarios", la **Tabla 4.16** muestra la prueba de aceptación "Gestión ítem del catálogo de equipos", la **Tabla 4.17** muestra la prueba de aceptación

“Gestión de solicitudes activas”, la **Tabla 4.18** muestra la prueba de aceptación “Solicitudes archivadas” y la **Tabla 4.19** muestra la prueba de aceptación “Gestión de PC en BDD”.

Prueba de Aceptación	
Número:	PA2C3
Historia de Usuario:	HU2C3 - Acción reiniciar
Nombre:	Acción reiniciar
Descripción:	Mediante esta acción se puede reiniciar uno o varios equipos dentro de la red del laboratorio
Condiciones de Ejecución:	Debe estar ejecutándose la aplicación cliente en los PC que se desee reiniciar La opción para realizar esta opción está disponible desde la aplicación profesor y aplicación principal.
Pasos de Ejecuciones:	En la aplicación Principal, se selecciona el PC al cual se desee hacer la acción y dando clic derecho se muestra un menú en el cual se da clic en la opción encender si se desea hacerlo de manera individual, en el caso de querer realizar la acción de todas la sala se da clic en la opción Acciones y se da clic en el icono de reiniciar y la acción se realizará en la sala actual. En la aplicación profesor se debe seleccionar los iconos de los PC que se desee realizar la acción y en la opción Acciones se da clic en el icono reiniciar.
Resultado Esperado:	Reiniciar uno o los PC seleccionados
Evaluación de la prueba:	Se alcanzó el resultado esperado

Tabla 4.10. Prueba de aceptación - Acción reiniciar

Prueba de Aceptación	
Número:	PA2C4
Historia de Usuario:	HU2C4 - Acción bloquear
Nombre:	Acción bloquear
Descripción:	Mediante esta acción se puede bloquear uno o varios equipos dentro de la red del laboratorio
Condiciones de Ejecución:	Debe estar ejecutándose la aplicación cliente en los PC que se desee bloquear La opción para realizar esta opción está disponible desde la aplicación Android, aplicación profesor y aplicación principal.
Pasos de Ejecuciones:	En la aplicación Android se ingresa a la aplicación con un usuario registrado en el sistema, se selecciona la sala y los PC a los cuales se desee hacer la acción y finalmente se presiona el ícono con la opción bloquear En la aplicación Principal se selecciona el PC al cual se desee hacer la acción y dando clic derecho se muestra un menú en el cual se da clic en la opción encender si se desea hacerlo de manera individual, en el caso de querer realizar la acción de toda la sala se da clic en la opción Acciones y se da clic en el ícono de bloquear y la acción se realizará en la sala actual. En la aplicación profesor se debe seleccionar los íconos de los PC que se desee realizar la acción y en la opción Acciones se da clic en el icono bloquear.
Resultado Esperado:	Desbloquear uno o los PC seleccionados
Evaluación de la prueba:	Se alcanzó el resultado esperado

Tabla 4.11. Prueba de aceptación - Acción bloquear

Prueba de Aceptación	
Número:	PA2C5
Historia de Usuario:	HU2C5 - Acción desbloquear
Nombre:	Acción desbloquear
Descripción:	Mediante esta acción se puede desbloquear uno o varios equipos dentro de la red del laboratorio
Condiciones de Ejecución:	Debe estar ejecutándose la aplicación cliente en los PC que se desee desbloquear La opción para realizar esta opción está disponible desde la aplicación Android, aplicación profesor y aplicación principal.
Pasos de Ejecución:	En la aplicación Android se ingresa a la aplicación con un usuario registrado en el sistema, se selecciona la sala y los PC a los cuales se desee hacer la acción y finalmente se presiona el ícono con la opción desbloquear En la aplicación Principa, se selecciona el PC al cual se desee hacer la acción y dando clic derecho se muestra un menú en el cual se da clic en la opción desbloquear si se desea hacerlo de manera individual, en el caso de querer realizar la acción de todas la sala se da clic en la opción Acciones y se da clic en el ícono de desbloquear y la acción se realizará en la sala actual. En la aplicación profesor se debe seleccionar los íconos de los PC que se desee realizar la acción y en la opción Acciones se da clic en el icono desbloquear.
Resultado Esperado:	Desbloquear uno o los PC seleccionados
Evaluación de la prueba:	Se alcanzó el resultado esperado

Tabla 4.12. Prueba de aceptación - Acción desbloquear

Prueba de Aceptación	
Número:	PA2C6
Historia de Usuario:	HU2C6 - Acceso remoto
Nombre:	Acceso remoto
Descripción:	Se accede de forma remota a los PC clientes
Condiciones de Ejecución:	Debe estar ejecutándose la aplicación cliente en los PC que se desee realizar el acceso remoto La opción para realizar esta opción está disponible desde la aplicación profesor y aplicación principal.
Pasos de Ejecución:	En la aplicación principal se debe dar doble clic en el nombre del PC y se abre una ventana en la cual se muestra la pantalla del PC cliente. En la aplicación profesor se debe dar doble clic en el ícono del PC y se abre una ventana en la cual se muestra la pantalla del PC cliente.
Resultado Esperado:	Acceder de forma remota a los PC clientes
Evaluación de la prueba:	Se alcanzó el resultado esperado

Tabla 4.13. Prueba de aceptación - Control acceso remoto

Prueba de Aceptación	
Número:	PA2C7
Historia de Usuario:	HU2C7 - Obtener información
Nombre:	Obtener información
Descripción:	Se obtiene información básica de un PC seleccionado como: IP, MAC, nombre de equipo y grupo
Condiciones de Ejecución:	Debe estar ejecutándose la aplicación cliente en los PC que se desee conocer la información La opción para realizar esta opción está disponible desde la aplicación profesor y aplicación principal.
Pasos de Ejecuciones:	En la aplicación profesor se da clic en el icono del PC que se desee ver la información y se mostrará una ventana en la cual se puede visualizar la información del PC En la aplicación principal se debe dar clic en el nombre del PC que se desee ver la información y se mostrará una ventana con la información del PC
Resultado Esperado:	Visualizar la información básica de un PC seleccionado: IP, MAC, nombre de equipo y grupo
Evaluación de la prueba:	Se alcanzó el resultado esperado

Tabla 4.14. Prueba de aceptación - Obtener Información

Prueba de Aceptación	
Número:	PA3B1
Historia de Usuario:	HU3B1 - Gestión de usuarios
Nombre:	Gestión de usuarios
Descripción:	Permite modificar, eliminar usuarios registrados en el sistema
Condiciones de Ejecución:	Funcionalidad desde la aplicación principal Conexión con la base de datos
Pasos de Ejecución:	Entrar en la opción Administrar usuarios Clic en ver usuarios En la ventana ver usuarios se visualiza los usuarios registrados en el sistema Por cada usuarios se tiene la opción de eliminar Para modificar se debe dar doble clic en el nombre del usuario Se mostrará una ventana en la cual se modifica los campos necesarios y se da clic en el botón aceptar para guardar los cambios
Resultado Esperado:	Modificar o eliminar usuarios registrados en el sistema
Evaluación de la prueba:	Se alcanzó el resultado esperado

Tabla 4.15. Prueba de aceptación - Gestión de usuarios

Prueba de Aceptación	
Número:	PA3B2
Historia de Usuario:	HU3B2 - Gestión ítems del catálogo de equipos
Nombre:	Gestión ítems del catálogo de equipos
Descripción:	Permite modificar, eliminar ítems del catálogo registrados en el sistema
Condiciones de Ejecución:	Funcionalidad desde la aplicación principal Conexión con la base de datos
Pasos de Ejecuciones:	Entrar en la opción Catálogo de equipos Clic en ver equipos En la ventana ver equipos se visualiza los equipos del catálogo de ítems registrados en el sistema Por cada item se tiene la opción de eliminar Para modificar se debe dar doble clic en el nombre del item Se mostrara una ventana en la cual se modifica los campos necesarios y se da clic en el botón aceptar para guardar los cambios
Resultado Esperado:	Modificar, eliminar ítems del catálogo registrados en el sistema
Evaluación de la prueba:	Se alcanzó el resultado esperado

Tabla 4.16. Prueba de aceptación - Gestión ítem del catálogo de equipos

Prueba de Aceptación	
Número:	PA3B3
Historia de Usuario:	HU3B3 - Gestión de solicitudes activas
Nombre:	Gestión de solicitudes activas
Descripción:	Permite cambiar de estado las solicitudes y también se tiene la opción de imprimir la solicitud
Condiciones de Ejecución:	Funcionalidad desde la aplicación principal Conexión con la base de datos
Pasos de Ejecución:	Entrar en la opción Solicitudes Clic en Ver Solicitudes Activas En la Ver Solicitudes Activas se visualiza las solicitudes activas registrados en el sistema Por cada solicitud activa se tiene la opción de archivar Para imprimir se debe dar doble clic en la solicitud Se mostrara una ventana en la cual se tiene la opción de imprimir.
Resultado Esperado:	Cambiar de estado las solicitudes y también poderlas imprimir
Evaluación de la prueba:	Se alcanzó el resultado esperado

Tabla 4.17. Prueba de aceptación - Gestión de solicitudes activas

Prueba de Aceptación	
Número:	PA3B4
Historia de Usuario:	HU3B4 - Solicitudes archivadas
Nombre:	Solicitudes archivadas
Descripción:	Visualizar las solicitudes archivadas y se tiene la opción de imprimirlas
Condiciones de Ejecución:	Funcionalidad desde la aplicación principal Conexión con la base de datos
Pasos de Ejecuciones:	Entrar en la opción Solicitudes Clic en Ver Solicitudes Guardadas En la ventana Ver Solicitudes Guardadas se visualiza las solicitudes archivadas registrados en el sistema Para imprimir se debe dar doble clic en la solicitud Se mostrará una ventana en la cual se tiene la opción de imprimir.
Resultado Esperado:	Visualizar las solicitudes archivadas e imprimirlas
Evaluación de la prueba:	Se alcanzó el resultado esperado

Tabla 4.18. Prueba de aceptación - Solicitudes archivadas

Prueba de Aceptación	
Número:	PA3B5
Historia de Usuario:	HU3B5 - Gestión de PC en BDD
Nombre:	Gestión de PC en BDD
Descripción:	Permite modificar, eliminar los PC registrados en el sistema
Condiciones de Ejecución:	Funcionalidad desde la aplicación principal Conexión con la base de datos
Pasos de Ejecución:	Clic en el icono editar PC en la pantalla principal Se abre una ventana en la cual se pueden modificar los campos o eliminar el PC.
Resultado Esperado:	Modificar o eliminar los PC registrados en el sistema.
Evaluación de la prueba:	Se alcanzó el resultado esperado

Tabla 4.19. Prueba de aceptación - Gestión de PC en BDD

4.5.1.4 Cuarta iteración

En la cuarta iteración se encuentran las pruebas de aceptación que permiten la comunicación entre las distintas aplicaciones. La **Tabla 4.20** y la **Tabla 4.21** muestran la prueba de aceptación "Comunicación aplicación Android con aplicación principal", la **Tabla 4.22** muestra la prueba de aceptación "Comunicación aplicación profesor con aplicación principal", la **Tabla 4.23** y la **Tabla 4.24** muestran la prueba de aceptación "Comunicación aplicación cliente con aplicación profesor o principal", la **Tabla 4.25** muestra la prueba de aceptación "Comunicación por voz".

Prueba de Aceptación	
Número:	HU4AS1
Historia de Usuario:	HU4AS1 - Comunicación aplicación Android con aplicación principal
Nombre:	Comunicación aplicación Android con aplicación principal
Descripción:	Se establecerá una comunicación TCP por medio de la red
Condiciones de Ejecución:	Se debe ejecutar la aplicación principal y aplicación Android Se debe configurar la dirección IP de la aplicación principal en la aplicación Android

Tabla 4.20. Prueba de aceptación - Comunicación aplicación Android con aplicación principal (Primera parte)

Pasos de Ejecución:	Se conecta el dispositivo Android a la red Se conecta la aplicación principal a la red Se realiza una prueba de autenticación la genera mensajes que se intercambian entre las aplicaciones
Resultado Esperado:	Inicio de sesión de usuario registrado
Evaluación de la prueba:	Se alcanzó el resultado esperado

Tabla 4.21. Prueba de aceptación - Comunicación aplicación Android con aplicación principal (Segunda parte)

Prueba de Aceptación	
Número:	HU4PS2
Historia de Usuario:	HU4PS2 - Comunicación aplicación profesor con aplicación principal
Nombre:	Comunicación aplicación profesor con aplicación principal
Descripción:	Se establecerá una comunicación UDP por medio de la red
Condiciones de Ejecución:	Se debe ejecutar la aplicación principal y aplicación profesor Se debe configurar la dirección IP de la aplicación principal en la aplicación profesor
Pasos de Ejecuciones:	Se conecta el aplicación profesor a la red Se conecta la aplicación principal a la red Se presiona el botón guardar computadores en la aplicación profesor así se establece una comunicación con la aplicación principal y se envía la información de los PC
Resultado Esperado:	Obtener la información de los PC asociados a la aplicación profesor en la aplicación principal
Evaluación de la prueba:	Se alcanzó el resultado esperado

Tabla 4.22. Prueba de aceptación - Comunicación aplicación profesor con aplicación principal

Prueba de Aceptación	
Número:	HU4CPS3
Historia de Usuario:	HU4CPS3 - Comunicación aplicación cliente con aplicación profesor o principal
Nombre:	Comunicación aplicación cliente con aplicación profesor o principal

Tabla 4.23. Prueba de aceptación - Comunicación aplicación cliente con aplicación profesor o principal (Primera parte)

Descripción:	Se establecerá una comunicación UDP por medio de la red
Condiciones de Ejecución:	Se debe ejecutar la aplicación principal, aplicación profesor y aplicación cliente
Pasos de Ejecuciones:	Las aplicaciones principal y profesor envían broadcast continuamente a la aplicación cliente y esta responde con su estado
Resultado Esperado:	Obtener el estado de los PC clientes
Evaluación de la prueba:	Se alcanzó el resultado esperado

Tabla 4.24. Prueba de aceptación - Comunicación aplicación cliente con aplicación profesor o principal (Segunda parte)

Prueba de Aceptación	
Número:	HU4AS4
Historia de Usuario:	HU4AS4 - Comunicación por voz
Nombre:	Comunicación por voz
Descripción:	Se realiza una comunicación por voz entre dispositivos Android
Condiciones de Ejecución:	Se debe tener un usuario de tipo administrador registrado en el sistema Debe estar corriendo el servicio SIP Deber estar corriendo la aplicación principal Se debe ejecutar la aplicación Android
Pasos de Ejecuciones:	Ingresar a la aplicación Android con usuario y contraseña Ingresar al menú y entrar a la opción llamadas Seleccionar al usuario que se desee llamar y presionar el botón llamar.
Resultado Esperado:	Realizar una llamada de voz entre dispositivos Android
Evaluación de la prueba:	Se alcanzó el resultado esperado

Tabla 4.25. Prueba de aceptación - Comunicación por voz

4.6 COMPARACIÓN ENTRE APLICACIONES

Para finalizar con el capítulo se presenta una tabla comparativa entre las aplicaciones analizadas en el Capítulo 2 y el sistema. La **Tabla 4.26** muestra la comparación de las aplicaciones iTalc, Netsupport y el sistema.

Características	iTalc	Netsupport Manager	Sistema
Autenticación	Es necesario tener una cuenta de usuario de Windows con contraseña para la autenticación cuando se inicia la aplicación	No cuenta con ningún método de autenticación	Autenticación para ciertas partes del sistema, las cuales son - Préstamo de quipos - Reiniciar en modo *Frozen* y *Thawed* - Ingresar a la aplicación Android
Barra de herramientas	Muestra información detallada de cada uno de sus elementos	Muestra información detallada de cada uno de sus elementos	Muestra información detallada de cada uno de sus elementos
Agregar nuevos PC	Se debe agregar manualmente cada uno de los PC	Cuenta con la opción examinar que simplifica la detección de los PC	Cuenta con la opción explorar que detecta los PC de su mismo grupo de trabajo
Pantalla de control de los PC	Presenta una interfaz con opciones básicas para el control del PC	Presenta una interfaz con opciones básicas para el control del PC	Presenta la pantalla de los PC sin ningún tipo de opción
Gestión	Encender, Reiniciar, Apagar, Bloquear, Desbloquear	Encender, Reiniciar, Rpagar	Encender, Apagar, Reiniciar, Bloquear, Desbloquear, Reiniciar en modo *Frozen*, Reiniciar en modo *Thawed*
Soporte	Pagado	Pagado	No se ofrece soporte
Préstamo de equipos	No	No	Se pueden generar solicitudes
Aplicación móvil	No	No	Posee opciones básicas de gestión
Llamadas entre Dispositivos Móviles	No	No	Sí, en dispositivos Android mediante SIP

Tabla 4.26. Comparación de las aplicaciones iTalc, Netsupport y el sistema.

Al final se puede observar que el proyecto cumple con los elementos básicos establecidos anteriormente; se debe tomar en cuenta que las aplicaciones con las que se compara llevan años en el mercado por lo que evidentemente tienen un desarrollo más depurado y fueron útiles para obtener una idea general de la funcionalidad de un sistema de gestión.

CAPÍTULO 5 CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

- El funcionamiento de SIP en medios inalámbricos no es confiable, por lo que en las pruebas realizadas se obtuvo retardo en la voz y problema de pérdida de paquetes; en la etapa de pruebas se utilizó varios códecs pero ninguno mostró buenos resultados. Además se debe tomar en cuenta que la potencia de la señal del Access Point a la que se conectan los dispositivos Android no es muy buena.

- En las aplicaciones se usan hilos, los cuales ayudan a que estas no se detenga mientras procesa las distintas peticiones que se generan desde otras aplicaciones del sistema; así, si se demora en dar respuesta a una petición la aplicación sigue funcionando correctamente.

- Cuando se desarrolla una aplicación en Android los componentes visuales se trabajan de manera independiente de la lógica, por lo que se puede trabajar paralelamente en el diseño de la interfaz y en la funcionalidad de la aplicación.
- Si el Laboratorio de Informática de la FIEE donde se desplegó el sistema crece, solo es necesario la exploración de los nuevos PC, de esta forma se puede seguir gestionando a todo el laboratorio.

- La parte del préstamo de equipos con la que cuenta el sistema permite tener un mejor manejo del inventario de equipos; anteriormente, el registro de préstamos era manual y no se tenía ningún control sobre este. Ahora se puede ver quién pidió los equipos y cuando lo hizo, además el módulo de impresión permite dejar constancia física de lo requerido.

- El uso de la metodología de software *extreme programming* permitió aprovechar de mejor manera el tiempo debido a que el desarrollo fue

dividido en iteraciones, las cuales especificaban las funciones del sistema individualmente.

- Las pruebas de aceptación permitieron verificar que se cumplieron los requerimientos planteados en las historias de usuarios

- Las aplicaciones analizadas permitieron tener una base para poder determinar el funcionamiento de un sistema de gestión.

- El sistema posee una secuencia de escape al presionar las teclas control + alt + suprimir la cual siempre lleva a mostrar la pantalla del sistema de Windows, a pesar de que el computador cliente que encuentre en estado bloqueado se podrá acceder a esta pantalla.

- La aplicación Android fue desarrollada para funcionar en dispositivos móviles con las versiones del sistema operativo Android superiores al API nivel 9 o su equivalente la versión Android 2.3 GINGERBREAD, ya que a partir de esta versión se incluyen las funcionalidades necesarias para el uso del servicio SIP.

- El uso de sistemas distribuidos ayuda a optimizar el tiempo de realización de los procesos; en el caso del sistema de gestión, se ha optimizado varias tareas como encender apagar, reiniciar los PC en modo *Thawed* y *Frozen*. Para estos dos últimas se evidencia una gran reducción de tiempos ya que sin el sistema se empleaba 1 minutos aproximadamente por PC es decir unos 10 minutos por sala tomando en cuenta que se realizaba esto entre dos personas; ahora con el sistema instalado en los PC esto se hace en cuestión de segundos.

- Los mensajes de *broadcast* dentro del sistema ayudan a detectar los PC con aplicación cliente de manera automática; de esta manera se puede ver el estado los PC y no es necesario agregarlos de uno en uno de forma manual.

- En el archivo AndroidManifest se deben especificar todos los permisos que se desean utilizar como: SIP, Internet, acceso al estado de WIFI, etc. Además se debe dar el permiso para usar algunos componentes de hardware del dispositivo.

5.2 RECOMENDACIONES

- Es necesario configurar los PC previamente para que puedan recibir los paquetes de Wake on LAN; esto se hace desde el BIOS y desde el sistema operativo; de esta manera la tarjeta de red se queda activa consumiendo poca energía y así el PC podrá recibir el mensaje correspondiente a la MAC de su tarjeta de red. En el laboratorio de la FIEE se usa un programa llamado BootIT, el cual es un administrador de particiones. Cuando por error se envían dos mensajes Wake on LAN el PC se congela impidiendo que se inicie correctamente, este error no se pudo solucionar debido a que el programa mencionado es privado y no se puede tener acceso al código fuente. Además, es importante evitar que el PC no suspenda su alimentación eléctrica; si esto sucede, la tarjeta de red queda inactiva y el mensaje Wake on LAN no funcionará adecuadamente.

- Para el uso de las llamadas mediante SIP se recomienda realizar un estudio para implementar una red inalámbrica en la Laboratorio de Informática de la FIEE que tenga QoS; de esta manera se tendrá una mejor calidad en las llamadas.

- Se debe sacar respaldos de la base de datos para evitar pérdidas: como usuarios o solicitudes archivadas; de esta forma se podrán recuperar los datos.

- Se recomienda seguir con el desarrollo del sistema para agregar nuevas funcionalidades al mismo; de esta forma se pueden considerar aspectos

que no se tomaron en cuenta en la propuesta inicial del proyecto como bloquear puertos USB, bloquear el Internet en los PC que ejecutan la aplicación cliente, transferencia de archivos o pasar la pantalla del PC profesor al PC cliente.

- La aplicación principal debe estar instalada en un PC que cumpla con funciones de Servidor, para que esta esté disponible cuando las otras aplicaciones requieran conectarse.

- En la aplicación Android para visualizar de una manera óptima las opciones de gestión es recomendable un dispositivo Android con una pantalla de 4.3 pulgadas como mínimo; sin embargo, si no se dispone de un dispositivo con ese tamaño de pantalla de igual forma se visualizará con ayuda de un scrollbar.

- En los PC portátiles del Laboratorio de Informática de la FIEE que tienen instalada la aplicación cliente se debe tener una resolución de pantalla de 1360x768 pixeles para que su visualización con la aplicación principal y la del profesor por medio de acceso remoto sea adecuada y no presente ningún tipo de distorsión.
- Al instalar el servidor Asterisk en una máquina virtual no se cargan todos los módulos necesarios para realizar llamadas como es el caso del app_dial.so, así que se debe cargar este por línea de comandos para el correcto funcionamiento del servidor.

- El sistema desarrollado puede ser usado en otros laboratorios para gestionar los PC; la Escuela Politécnica Nacional cuenta con diferentes laboratorios de computación donde se podría implementar para hacer más eficiente la labor de las personas encargadas.

- Para hacer uso de la aplicación Android del sistema de gestión es recomendable estar conectado a la red inalámbrica del Laboratorio de Informática, debido a que si se usan otras redes inalámbricas que son

difundidas en la Escuela Politécnica Nacional es posible que tengan bloqueados los puertos que usa la aplicación.

- Para hacer cambios en los atributos de las tablas de la base de datos como nombre de usuario, nombre de los PC, nombre de los equipos u otros atributos que no puedan ser cambiados desde las aplicaciones del sistema se recomienda hacerlo directamente en la base de datos.

- Se recomienda cerrar la sesión de la aplicación Android una vez se termine su uso para que así personas no autorizadas no usen las funciones que esta tiene, en caso de que se tenga acceso al dispositivo móvil del administrador.

- En caso de tener conflicto con otras aplicaciones debido al uso de los puertos, se recomienda cambiar estos en las aplicaciones del sistema mediante los diferentes formularios de configuración.

REFERENCIAS BIBLIOGRÁFICAS

[1] Marek Rouchal, Martin Wilck Jens Lippmann. (2002, Feb.) SISTEMAS DISTRIBUIDOS. [Online]. http://www.ibiblio.org/pub/linux/docs/LuCaS/Manuales-LuCAS/doc-manual-openMosix-1.0/doc-manual-openMosix_html-1.0/node11.html

[2] Andrew S. Tananbaum, *Sistemas Operativos Distribuidos*, Primera ed., Luis Gerardo Cedeño Placencia, Ed. México : Prentice Hall Hispanoamerica S A, 1996.

[3] Departamento de Tecnoloxías da Información e as Comunicacións. MANUAL DE SOCKETS EN C. [Online]. http://www.fic.udc.es/files/asignaturas/56XR/files/ManualSocketsC-2009.pdf

[4] Oscar Déniz Suárez. (2006, Julio) Comunicación mediante sockets. [Online]. http://sopa.dis.ulpgc.es/progsis/material-didactico-teorico/tema7_1transporpagina.pdf

[5] Steve McConnell, *Code Complete*, Segunda ed. Redmond, Washington: Microsoft Press, 2006.

[6] Anónimo. (2015, Aug.) Fases de desarrollo del Software - Software del K2. [Online]. http://www.softwaredelk2.com/empresa-software-k2/proceso-de-desarrollo-del-software/

[7] Armando Medina. (2015, Aug.) Metodologia de desarrollo de software. [Online]. http://www.academia.edu/4984909/Metodologia_de_desarrollo_de_software

[8] Anónimo. Metodologias para el desarrollo de software. [Online]. http://procesosdesoftware.wikispaces.com/METODOLOGIAS+PARA+DESARROLLO+DE+SOFTWARE

[9] Roger S. Pressman, *Ingeniería del software: un enfoque práctico*., 1997.

[10] José Joskowicz. (2015, Apr.) Reglas y Prácticas en. [Online]. http://iie.fing.edu.uy/~josej/docs/XP%20-%20Jose%20Joskowicz.pdf

[11] Ing. José Joskowicz. (2008, Febrero) Reglas y Prácticas en eXtreme Programming. [Online]. http://iie.fing.edu.uy/~josej/docs/XP%20-%20Jose%20Joskowicz.pdf

[12] Anónimo. Historia de usuario - Scrum Manager BoK. [Online]. http://www.scrummanager.net/bok/index.php?title=Historia_de_usuario

[13] Ing. Software (Equipo 02). Metologia XP. [Online]. http://ingsoftware072301.obolog.es/metodologia-xp-2012877

[14] Android. (2015, July) Android, the world's most popular mobile platform. [Online]. http://developer.android.com/about/android.html

[15] Victor Reyes Andres Santos, *Diseño e implementación de un sistema prototipo de carro de compras virtual empleando comunicaciones NFC y el sistema Android*. Quito, Ecuador, 2014.

[16] xatakandroid. [Online]. http://www.xatakandroid.com/sistema-operativo/que-es-android

[17] Android. (2015, July) Gingerbread. [Online]. http://developer.android.com/about/versions/android-2.3-highlights.html

[18] Android. (2015, July) Honeycomb. [Online]. http://developer.android.com/about/versions/android-3.0-highlights.html

[19] Android. (2015, July) Ice Cream Sandwich. [Online]. http://developer.android.com/about/versions/android-4.0-highlights.html

[20] Android. (2015, July) Jelly Bean. [Online]. http://developer.android.com/about/versions/jelly-bean.html

[21] Android. (2015, July) Android KitKat. [Online]. http://developer.android.com/about/versions/kitkat.html

[22] Android. (2015, July) Android Lollipop. [Online]. http://developer.android.com/about/versions/lollipop.html

[23] Abraham Silberschatz, *Fundamentos de bases de datos*. Aravaca, Madrid: Cuarta edición, 2002. [Online]. https://unefazuliasistemas.files.wordpress.com/2011/04/fundamentos-de-bases-de-datos-silberschatz-korth-sudarshan.pdf

[24] Network Working Group. (2002, Junio) SIP: Session Initiation Protocol. [Online]. https://www.ietf.org/rfc/rfc3261.txt

[25] Anónimo. Session Initiation Protocol. [Online]. http://www.quarea.com/es/sip_session_initiation_protocol

[26] Anónimo. (2015, Aug.) SIP - Asterisk Wiki. [Online]. http://www.wikiasterisk.com/index.php?title=SIP

[27] Anónimo. Qué es el protocolo RDP. [Online]. http://ordenador.wingwit.com/Redes/other-computer-networking/78636.html#.VZRhYvl_Okp

[28] Tobias Doerffel. iTalc. [Online]. http://italc.sourceforge.net/

[29] NetSupport Software Ltd. (2015, Apr.) NetSupport Manager. [Online]. http://www.netsupportmanager.com/ES/index.asp

[30] MySQL. (2015, Jan.) MySQL : Download Connector/Net. [Online]. https://dev.mysql.com/downloads/connector/net/

[31] AsteriskWin32. (2015, Enero) AsteriskWin32 - The Open Source PBX for Windows. [Online]. http://www.asteriskwin32.com/

ANEXOS

Los Anexos se encuentran en un CD adjunto a este documento.

ANEXO A: Manual de usuario

ANEXO B: Aplicación principal

- **ANEXO B.1:** Código Fuente
- **ANEXO B.2:** Instalador

ANEXO C: Aplicación profesor

- **ANEXO C.1:** Código Fuente
- **ANEXO C.2:** Instalador

ANEXO D: Aplicación cliente

- **ANEXO D.1:** Código Fuente
- **ANEXO D.2:** Instalador

ANEXO E: Aplicación Android

- **ANEXO E.1:** Código Fuente
- **ANEXO E.2:** Instalador

ANEXO F: Script base de datos

ANEXO G: Proyecto completo

- **ANEXO G.1:** Aplicación Android
- **ANEXO G.2:** Aplicaciones de escritorio