Universidad de las Ciencias Informáticas

Facultad 1

Trabajo de Diploma para optar por el Título de

Ingeniera en Ciencias Informáticas

Título: Arquitectura del módulo webphone para NovaDesk

Diplomante: Raydelmis Fernández Torres

Tutora: Msc. Yenisleydi Cariaga Cristo

Co-Tutora: Ing.Lisandra Cala Hernández

Ciudad de La Habana, junio, 2011

Frase

"... Si los jóvenes fallan, todo fallará. Es mi más profunda convicción que la juventud cubana luchará por impedirlo. Creo en ustedes."

Declaración de Autoría

Declaro ser autora del Trabajo de Diploma *"***Arquitectura del módulo webphone para NovaDesk"** y confiero a la Universidad de las Ciencias Informáticas (UCI) los derechos patrimoniales del mismo, con carácter exclusivo.

Para que así conste firmamos el presente a los 4 días del mes de junio y del año 2011.

_____

Raydelmis Fernández Torres
Autora

_____

Msc. Yenisleydi Cariaga Cristo
Tutora

_____

Ingeniera Lisandra Cala Hernández
Co- Tutora

Datos de Contacto de los tutores del Trabajo de Diploma

Nombre y Apellidos: MsC. Yenisleydi Cariaga Cristo

Email: yeni@uci.cu

Síntesis del Tutor:

Licenciada en Sociología por la Facultad de Filosofía, Historia y Sociología de la Universidad de la Habana en el año 2004. Máster en Ciencias en Estudios Sociales de la Ciencia y la Tecnología por la Cátedra CTS de la Universidad de la Habana en el año 2008. Profesora asistente de la Universidad de las Ciencias Informáticas. Ha cursado diversos Diplomados y Cursos de Postgrados, entre ellos Diplomado en Innovación y Docencia Universitaria, Plataforma de Tele formación AprenDist, Curso de Ética Informática. Imparte las asignaturas del perfil de las Ciencias Sociales, aunque se especializa en Problemas Sociales de la Ciencia y la Tecnología. Imparte el Curso de Registros, Patentes y Licencias, del Segundo Perfil del Centro CESOL de la Facultad 1. Imparte el Curso de Postgrado de Metodología de la Investigación Científica. Es Especialista Funcional del Grupo SIMAYS. Ha participado en diversos eventos nacionales e internacionales.

Dedicatoria

A mi abuela Miriam que se que si estuviera viva estaría muy orgullosa de mi. Mami te quiero mucho y siempre te querré.

A mis padres que los adoro con mi vida. Mamá y papá los quiero mucho.

A mi hermano que es mi gran orgullo. Te quiero mucho Rodnan.

A mi novio que lo quiero mucho. Te adoro Fer.

Agradecimientos

Agradezco a mi familia que es lo más grande y hermoso que tengo en la vida y lo más cercano a ser prefecto.

A mi mamá Lourdes por ser la mejor madre del mundo, la persona más dulce y cariñosa que conozco en la vida. Por llenarme de regocijo cada vez que aprecio como le brinda su amor a mis amigos y como es querida por ellos. Por ser mi mayor concejera y amiga. Por ser siempre tan paciente conmigo y comprensiva.

A mi papá Luis por demostrarme que padre no es cualquiera porque cualquier persona no hace la labor tan bella que él ha hecho conmigo. Le agradezco mucho por cuidar siempre de mi, de mi mamá y de mi hermano. Por quererme tanto, por darme el orgullo de decir que mi padre es mi amigo, mi hermano y mi mamá.

A mi hermano Rodnan por ser mi ejemplo a seguir, por ser esa persona que algún día quisiera llegar a ser. Por cuidarme, celarme, regañarme, quererme, apoyarme, por hacer que mis primeros años en esta universidad fueran más fáciles. Gracias por ser ese hermano que todo el mundo quisiera tener.

A mi novio Fernando por hacer mi vida más feliz con su presencia, por permitirme ser parte de la suya, de su mundo, por ser una parte importante del mío. Por ayudarme a que mis días fueran más agradables. Ojalá nunca tuviera que irme.

A mi primo Ale que es como un hermano para mí. Aunque la distancia nos ha alejado un poco, mi cariño hacia ti sigue igual de fuerte.

A mis hermanitas Marcia y Yanara por ser mis fieles compañeras de infancia y mis más grandes amigas.

A Yanet por ser una persona muy especial en mi vida, por haber querido tanto a mi hermano, por querer a mis padres y por estar siempre pendiente de mí.

A Yadira por formar parte en tan poco tiempo de las personas más importantes en mi vida. Por ayudarme mucho cuando era solo una persona conocida para ella.

A mis grandes amigos y amigas Edito, Adriel, Raydel, Haniel, Dayron, Fidel, Arleny, Yudelky y Bibiana. Son lo mejor que me ha pasado en la universidad y el recuerdo más bello que me llevo. Gracias por ser tan especiales conmigo, por ayudarme siempre, siempre los voy a querer.

A mis tutoras por ayudarme siempre que lo necesité y por animarme a seguir adelante.

A mis compañeros de estudio por toda la ayuda que me ha prestado.

Agradezco a todas las personas que de una forma u otra hicieron posible la realización de este trabajo

Resumen

En la actualidad existe un gran auge en la industria de software. Cuba a pesar de su difícil situación económica se esfuerza por informatizar los procesos fundamentales de la sociedad. La Universidad de las Ciencias Informáticas ha sido una institución decisiva en esta política, ya que sus estudiantes y profesores están vinculados a proyectos productivos donde desarrollan software tanto para instituciones nacionales, como para exportar. Muchos de estos proyectos utilizan herramientas libres.

En las facultades de la universidad se encuentra instalado el sistema operativo GNU/Linux, que muchas veces provoca que los usuarios presenten dudas a la hora de trabajar en este sistema. Como paliativo a esta situación, la universidad cuenta con un portal que funciona como centro de soporte, donde siempre hay un técnico de guardia para atender las necesidades de los clientes. Los reportes se realizan a través del chat o del reporte de incidencias, que no funciona de manera óptima. Este portal no cuenta con un módulo que permita la comunicación Volp.

Para darle solución al problema existente se crea la arquitectura de un módulo que permitirá la comunicación Volp. En esta investigación se definieron las herramientas, lenguaje de programación, los requisitos funcionales y no funcionales, las clases, patrones de diseño y todo lo necesario para la futura implementación del módulo.

Palabras claves: Software Libre, Webphone, Volp

Índice de contenido

Tabla de contenido

Índice de Ilustraciones

Índice de Tablas

Introducción al Trabajo de Diploma

En la actualidad la industria del software se desarrolla a un ritmo acelerado, tomando mayor espacio cada vez en empresas y centros públicos en todo el planeta. Muchos países se esfuerzan por alcanzar un gran avance en este campo y una reconocida posición en el mercado. Cuba a pesar de su difícil situación económica se ha introducido en la esfera de la producción del software, brindándole gran ayuda en el sustento de la economía.

Con el objetivo de lograr la informatización de la sociedad cubana y elevar la producción del software, se han creado centros productivos en la isla como es el caso de la Universidad de las Ciencias Informáticas (UCI). En esta universidad no solo se desarrolla software con fines de exportación, sino también con el objetivo de lograr un crecimiento científico y tecnológico en todas las esferas del país.

Entre los diferentes proyectos que se llevan a cabo en esta universidad, se encuentran algunos dedicados específicamente a la producción de software libre (SWL). Ejemplo de ello es el Centro de Desarrollo de Software Libre (CESOL), que se encarga de realizar los productos que diferencian a Nova de sus homólogos en el SWL y dar soporte y asesoría.

Con el objetivo de fortalecer los servicios de migración, asesoría y sistema, se crea dentro de CESOL el proyecto Sistemas Integrados de Migración, Asesoría y Sistemas (SIMAYS) . Dentro de los servicios que ofrece SIMAYS se encuentra la Herramienta Soporte Técnico NovaDesk, cuyo grupo de migración se encarga de todos los aspectos que garanticen la continuidad, disponibilidad y calidad del servicio prestado al usuario.

En los laboratorios docentes de la universidad se cuenta con el sistema operativo GNU/Linux con la distribución cubana Nova, que fue creada en la antigua facultad 10. En muchas ocasiones, los usuarios que interactúan con este sistema operativo presentan dudas acerca del trabajo con algunas herramientas que este posee.

La universidad cuenta con un portal de servicios de atención de incidencias para la distribución cubana GNU/Linux Nova, donde existe un chat on-line con técnicos de NovaDesk, que aclaran las dudas existentes. El usuario muchas veces no queda satisfecho del todo con la aclaración, debido a que es difícil identificar el problema por parte de los técnicos mediante esta vía de comunicación. El portal no posee una modalidad de atención en línea a través de VoIP. No existe un módulo que garantice la comunicación a través de teléfonos (celulares y fijos), computadoras y la línea suscriptora con los agentes receptores

que atienden estas llamadas, para brindar el servicio de ayuda ante cualquier eventualidad con el funcionamiento del sistema operativo GNU/Linux Nova.

Para darle solución a esta problemática se plantea el siguiente problema científico: ¿Cuál arquitectura de software brindará mejores prestaciones al módulo webphone para NovaDesk?

Como objetivo general de la investigación se plantea: definir la arquitectura del módulo de reportes de incidencias por medio de voz, para NovaDesk. Definiéndose los siguientes objetivos específicos:

- Realizar un estudio del arte referente a los diferentes tipos de diseños arquitectónicos usando metodologías ágiles.
- Definir los patrones arquitectónicos del módulo webphone.
- Proponer la arquitectura del módulo webphone.

El objeto de estudio de esta investigación está centrado en el proceso de desarrollo de software del módulo webphone para NovaDesk

El campo de acción está enmarcado en la arquitectura de software del módulo webphone para NovaDesk empleando metodología ágil.

Se plantea como **idea a defender**: el módulo webphone mejora la comunicación del sistema para gestión de soporte NovaDesk con el usuario.

Trazándose para el cumplimiento de los objetivos anteriores las siguientes **tareas de la investigación:**

- Revisión de la bibliografía especializada acerca de las distintas arquitecturas para diseñar un módulo webphone.
- Análisis de las herramientas que se emplearán en el desarrollo del sistema propuesto.
- Descripción de la arquitectura del sistema y patrones de diseño a emplear.

Durante todo el proceso investigativo que se realizó se utilizaron un conjunto de métodos científicos de investigación. Estos métodos se clasifican en:

Teóricos: Constituyen el enfoque general para abordar los problemas científicos, de ahí que posibiliten

profundizar en las regularidades y cualidades esenciales de los fenómenos.

Dentro de los métodos teóricos se emplearon los siguientes:

Histórico – lógico: A partir de elementos identificados en la creación de modelos de arquitectura, se estudiaron los fenómenos de su trayectoria, lo que permitió establecer la sucesión cronológica para el estudio e investigación de sus antecedentes, evolución y tendencias. Este método se utiliza en la investigación para estudiar la teoría conocida hasta el momento, todo lo que se ha hecho acerca del tema que se trata en el trabajo.

Sistémico: Se fueron creando sistemas de trabajo según las necesidades de las tareas que había que resolver. Estos sistemas fueron creados con un orden lógico.

El contenido de este documento está estructurado en tres capítulos:

El Capítulo 1. Fundamentación teórica: se realiza un estudio detallado de las tendencias, tecnologías y metodologías actuales con el fin de proponer las más adecuadas para la solución del problema.

El Capítulo 2. Concepción del módulo webphone: en este capítulo se realiza la propuesta de la arquitectura del módulo webphone para NovaDesk, teniendo en cuenta plataforma, lenguaje de programación, se definen las funcionalidades del módulo webphone expresándose de forma priorizada en una lista de reserva del producto (LRP); luego son especificadas a través de las historias de usuario (HU) con sus respectivas tareas de ingeniería

El **Capítulo 3.** Descripción de la arquitectura: en este capítulo se definen los patrones arquitectónicos que se van a utilizar y clases.

Cuenta además con conclusiones, recomendaciones, referencias bibliográficas, bibliografía consultada. y anexos.

Posee también un glosario de términos y siglas.

Capítulo 1 – Fundamentación Teórica.

A partir de la década de los noventa, el campo de las comunicaciones de voz ha venido desarrollando nuevas tecnologías, una de ellas es la tecnología de voz sobre el protocolo IP (VoIP), donde la principal innovación es la implementación de un teléfono en software, conocido como softphone. Actualmente, el crecimiento de la tecnología VoIP ha sido significativo, tanto que empresas y usuarios particulares hacen uso de ella.

1. *¿Qué es un Webphone?*

El webphone es un teléfono Volp que se ejecuta desde un entorno web, es un servicio de emisión y recepción de llamadas telefónicas, que favorece y mejora la atención al cliente y la comunicación entre usuarios. Permite las llamadas telefónicas entre clientes y empresas. Es aplicable a cualquier sector empresarial. **[1]**

1.1 *Ejemplo de webphone existentes y sus características.*

1.1.1- *Skype.*

Skype es un programa que permite realizar llamadas de voz entre ordenadores y entre ordenadores y teléfonos fijos o móviles. Posee una interfaz gráfica amigable debido a su semejanza con los teléfonos tradicionales y su alta calidad de imagen. Tiene un grupo de prestaciones que hacen de él una opción altamente explotadas por los usuarios. Su última versión está integrada a la red social Facebook. **[2]**

El programa ha sido desarrollado en lenguaje Pascal usando el entorno Delphi, más tarde fue portado a GNU/Linux y basándolo en las librerías Qt. El éxito de Skype reside en la gran compresión de datos que realiza, sin afectar prácticamente a la calidad de la transmisión de voz. **[3]**

1.1.2- Doddle WebPhone.

Doddle es un Teléfono Web SIP gratuito a través del cual se pueden realizar llamadas telefónicas desde una página web. Funciona con dos proveedores de Voz sobre IP. **[4]**

Posee voz de alta definición (Higth Definition) y una integración con su proveedor de VoIP. Es compatible con los adaptadores de teléfono análogo de VoIP (ATA / equipo físico). La integración que tiene con el servidor es J2EE (Java) /.NET / PHP / base de datos. Es soportado en todos los navegadores. **[5]**

1.1.3- Mizu-WebPhone.

Es un software de teléfono de VoIP que puede ser operado desde páginas web. Se puede llamar a cualquier otro teléfono web / teléfono IP o cualquier línea terrestre y número móvil, mediante un proveedor de servicio de VoIP de su elección. **[6]**

El teléfono es implementado como una applet / aplicación de java y es totalmente independiente de la plataforma, ejecutando java sobre cualquier navegador. Puede ser usado como un softphone funcionando en un sitio web. Todas las características de direccionamiento de llamada pueden estar activadas. **[Ídem]**

Mizu-WebPhone es compatible con cualquier servidor de VoIP usual o dispositivo como Cisco, Asterisk, softphones, ATA y otros. Utiliza el Applet de java usual, y requiere de la instalación de la máquina virtual de java. Funciona directamente de todos los navegadores. Los protocolos de transporte que utiliza son el UDP, el protocolo TCP, el TLS, túnel de HTTP. El cortafuegos que posee es estable.

Las llamadas pueden ser iniciadas escribiendo en la máquina un número de teléfono o por la API de JavaScript. Se puede conectar directamente a su servidor como cualquier otro teléfono IP de equipo físico. Está basado en SIP. **[Ídem]**

1.1.4- NetCom Satelital.

Como los anteriores NetComSatelital es un teléfono VoIP capaz de ejecutarse desde páginas web. Está basado en el protocolo estándar SIP, es compatible con todos los dispositivos SIP del mercado, lo cual permite la comunicación con cualquier usuario del servicio de telefonía IP de NetCom Satelital. **[7]**

El webphone está implementado como un applet de Java, lo cual garantiza una gran independencia de la plataforma del usuario, ya sea Windows, Linux, Mac o dispositivos móviles. Es necesario

instalar la máquina vitual de java para realizar o recibir llamadas a través del sistema. Si el navegador web del usuario soporta Java, el webphone se ejecutará directamente desde la web.

La mayoría de los clientes SIP almacenan la información de la cuenta (usuario, contraseña, servidor, etc.) en la computadora local. Esto hace que si no se toman las medidas pertinentes, cualquier otra persona que utilice esa misma computadora puede hacer llamadas que involucren costos para el usuario. **[Ídem]**

Requisitos no funcionales de NetComSatelital

El sistema puede ser utilizado en cualquier computadora superior a Pentium2. Básicamente se necesita:

- Un navegador web que soporte Java.
- Parlantes y micrófono para realizar la comunicación.
- Además del acceso a internet y una cuenta VoIP registrada en el sistema de NetCom Satelital.

Una vez dentro de la web los usuarios pueden llamar a otros usuarios del sistema VoIP de manera totalmente gratuita y llamar a cualquier lugar del mundo a un costo reducido.

No obstante, la arquitectura del webphone permite su implementación en otro tipo de escenarios, tales como:

- Click2Call: se coloca solamente un botón en la página web (por ejemplo, "Llámenos ahora"), y al pulsarlo se llama automáticamente a un número prefijado. Ideal para servicios de venta o soporte técnico.
- Integración con bases de datos: el webphone se convierte efectivamente en un discador que llama a una lista de números predefinidos. Ideal para centrales telefónicas. **[ídem]**

Comparado con un softphone tradicional

- El proceso de configuración es más directo y sencillo.
- Es más estándar ya que el usuario para establecer la comunicación solo tiene que acceder a la web en la que se encuentra el webphone.e instalar la máquina vistual de java.

Comparado con otros servicios de voz vía software.

A diferencia de servicios como MSN Messenger, Google Talk y demás, el webphone de NetCom Satelital está basado en el protocolo SIP, lo cual permite establecer comunicaciones no solamente con usuarios de una misma red, sino con cualquier teléfono del mundo, ya sea fijo o móvil.

Comparado con Flash.

Aunque se pueden hacer llamadas VoIP desde Flash, ésta es una alternativa muy ineficiente debido a que Flash no implementa protocolos VoIP, por lo cual es necesario instalar un servidor Flash Media Server y realizar las conversiones de señalización de todas las llamadas allí. Este es un proceso que consume mucha capacidad de CPU y acarrea una alta tasa de errores. **[Ídem]**

1.1.5- Soft4® Webphone.

Es un teléfono VoIp el cual crea una experiencia en línea interactiva y atractiva para los clientes mediante la adición de características avanzadas y la integración con otros servicios, tales como: vídeo llamada, correo de vídeo, centros de llamadas y videoconferencia, así como la posibilidad de participar con los medios de comunicación social y colaboración. **[8]**

Proporciona todas las funciones PBX estándar, además de servicios totalmente integrados de audio full - dúplex, vídeo full- dúplex y SMS.

Soft4® Webphone es un cliente web basado en SIP que facilita el hacer llamadas de video entre computadoras y teléfonos móviles. Los usuarios pueden acceder a estos servicios avanzados, simplemente marcando un número. Esto significa que el usuario no necesita descargar ni configurar software en las computadoras, teléfonos IP o teléfonos celulares. **[Ídem]**

Ventajas

- No requiere configuración de hardware. Sólo requiere un navegador web y conexión a Internet para funcionar.
- Proporciona control sobre las llamadas entrantes, salientes y sin contestar.
- Se puede utilizar en cualquier lugar.
- Funciones de fácil uso - cualquier persona puede utilizarlo, sin importar su conocimiento de computación.

- Reduce costos
- Sitio web de gran alcance
- Utiliza seguridad SSL. Proporciona autenticación y privacidad de la información utilizando la criptografía.
- Gestión de contactos - Permite al usuario crear grupos de usuarios y designar como lista blanca (usuarios permitidos) y lista negra (usuarios bloqueados). **[ídem]**

1.2- Arquitectura de Software.

Una arquitectura de software, también denominada arquitectura lógica, consiste en un conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del software para un sistema de información.

Establece los fundamentos para que analistas, diseñadores, probadores, programadores, y el resto de los integrantes del equipo o grupo de desarrollo trabajen en una línea común que permita alcanzar los objetivos del sistema de información, cubriendo todas las necesidades. **[9]**

La arquitectura de software dirige el desarrollo de los sistemas de software y contribuye a que estos se lleven a cabo en los límites establecidos de costos y tiempo, y fundamentalmente debe garantizar que se cumpla con los requisitos funcionales y no funcionales de los usuarios. La arquitectura de software es el resultado del trabajo durante todo el ciclo de vida del proyecto, y principalmente en las primeras iteraciones de un grupo de trabajo encabezado por el arquitecto (o grupo de arquitectura, en dependencia de las dimensiones del proyecto). **[Ídem]**

1.3- Patrones de arquitectura

Los patrones arquitectónicos son los que definen la estructura de un sistema software, los cuales a su vez se componen de subsistemas con sus responsabilidades, también tienen una serie de directivas para organizar los componentes del mismo sistema, con el objetivo de facilitar la tarea del diseño de tal sistema.

Un patrón de arquitectura de software describe un problema particular y recurrente del diseño, que surge en un contexto específico, y presenta un esquema genérico y probado de su solución. Este

esquema se especifica describiendo las componentes, con sus responsabilidades, relaciones, y las formas en que colaboran.**[ídem]**

1.3.1- Patrón Capas

La funcionalidad principal de este patrón es crear una modularidad del sistema asignando a cada capa funcionalidades específicas y bien definidas.

- **La capa de la Presentación**

Esta capa reúne todos los aspectos del software que tiene que ver con las interfaces y la interacción con los diferentes tipos de usuarios. Estos típicamente incluyen el manejo y aspecto de las ventanas, el formato de los reportes, menús, gráficos y elementos multimedia en general. .**[Ídem]**

- **La capa del Dominio de la Aplicación**

Esta capa reúne todos los aspectos del software que automatizan o apoyan los procesos de negocio que llevan a cabo los usuarios. Estos aspectos típicamente incluyen las tareas que forman parte de los procesos, las reglas y restricciones que aplican. Esta capa también recibe el nombre de la capa de la Lógica de la Aplicación. .**[Ídem]**

- **La capa del Acceso a Datos**

Esta capa reúne todos los aspectos del software que tienen que ver con el manejo de los datos

persistentes, por lo que también se le denomina la capa de las Bases de Datos.

Existen especificaciones de este patrón donde se manipula como será la comunicación entre cada una de las capas definidas. .**[ídem]**

1.4- Análisis de posibles arquitecturas a utilizar.

1.4.1- Arquitectura Cliente Servidor.

Agrupa conjuntos de elementos que efectúan procesos distribuidos y computo cooperativo.

La tecnología Cliente/Servidor es el procesamiento cooperativo de la información por medio de un conjunto de procesadores, en el cual múltiples clientes, solicitan requerimientos a uno o más

servidores centrales. Estos clientes no tienen que estar necesariamente distribuidos geográficamente. **[10]**

Desde el punto de vista funcional, se puede definir la tecnología Cliente/Servidor como una arquitectura distribuida que permite a los usuarios finales obtener acceso a la información de forma transparente aún en entornos multiplataforma. Se trata pues, de la arquitectura más extendida en la realización de Sistemas Distribuidos.

En esta arquitectura la capacidad de proceso está repartida entre los clientes y los servidores, aunque son más importantes las ventajas de tipo organizativo debidas a la centralización de la gestión de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema.

La separación entre cliente y servidor es una separación de tipo lógico, donde el servidor no se ejecuta necesariamente sobre una sola máquina ni es necesariamente un sólo programa. Una disposición muy común son los sistemas multicapa en los que el servidor se descompone en diferentes programas que pueden ser ejecutados por diferentes computadoras aumentando así el grado de distribución del sistema. **[ídem]**

Una disposición muy común son los sistemas multicapa en los que el servidor se descompone en diferentes programas que pueden ser ejecutados por diferentes computadoras aumentando así el grado de distribución del sistema. Un sistema Cliente/Servidor es un sistema de información distribuido basado en las siguientes características:

- **Servicio:** unidad básica de diseño. El servidor los proporciona y el cliente los utiliza.
- **Recursos compartidos:** Muchos clientes utilizan los mismos servidores y, a través de ellos, comparten tanto recursos lógicos como físicos.
- **Protocolos asimétricos:** Los clientes inician “conversaciones”. Los servidores esperan su establecimiento pasivamente. **[ídem]**.
- **Transparencia de localización física de los servidores y clientes:** El cliente no tiene por qué saber dónde se encuentra situado el recurso que desea utilizar.
- **Independencia de la plataforma hardware y software que se emplee.**
- **Sistemas débilmente acoplados:** Interacción basada en envío de mensajes.

- **Encapsulamiento de servicios:** Los detalles de la implementación de un servicio son transparentes al cliente.
- **Escalabilidad horizontal (añadir clientes) y vertical (ampliar potencia de los servidores):** Se puede aumentar la capacidad de clientes y servidores por separado.
- **Integridad:** Datos y programas centralizados en servidores facilitan su integridad y mantenimiento.
- **Centralización del control:** Los accesos, recursos y la integridad de los datos son controlados por el servidor de forma que un programa cliente defectuoso o no autorizado no pueda dañar el sistema.

En el modelo usual Cliente/Servidor, un servidor, se activa y espera las solicitudes de los clientes. Habitualmente, programas cliente múltiples comparten los servicios de un programa servidor común. Tanto los programas cliente como los servidores son con frecuencia parte de un programa o aplicación mayores. **[Ídem]**

El esquema de funcionamiento de un Sistema Cliente/Servidor sería:

1- El cliente solicita una información al servidor.

2- El servidor recibe la petición del cliente.

3- El servidor procesa dicha solicitud.

4- El servidor envía el resultado obtenido al cliente.

5- El cliente recibe el resultado y lo procesa.

Componentes de la arquitectura Cliente/Servidor.

El modelo Cliente/Servidor es un modelo basado en la idea del servicio, en el que el cliente es un proceso consumidor de servicios y el servidor es un proceso proveedor de servicios. Además esta relación está establecida en función del intercambio de mensajes que es el único elemento de acoplamiento entre ambos.

Del párrafo anterior se deducen los tres elementos fundamentales sobre los cuales se desarrollan e implantan los sistemas Cliente/Servidor: el proceso cliente que es quien inicia el diálogo, el proceso

servidor que pasivamente espera a que lleguen peticiones de servicio y el middleware que corresponde a la interfaz que provee la conectividad entre el cliente y el servidor para poder intercambiar mensajes.

Para entender en forma más ordenada y clara los conceptos y elementos involucrados en esta tecnología se puede aplicar una descomposición o arquitectura de niveles. Esta descomposición principalmente consiste en separar los elementos estructurales de esta tecnología en función de aspectos más funcionales de la misma:

- **Nivel de Presentación:** Agrupa a todos los elementos asociados al componente Cliente.
- **Nivel de Aplicación:** Agrupa a todos los elementos asociados al componente Servidor.
- **Nivel de comunicación:** Agrupa a todos los elementos que hacen posible la comunicación entre los componentes Cliente y servidor.
- **Nivel de base de datos:** Agrupa a todas las actividades asociadas al acceso de los datos.

Este modelo de descomposición en niveles, como se verá más adelante, permite introducir más claramente la discusión del desarrollo de aplicaciones en arquitecturas de hardware y software en planos.

Elementos principales.

Cliente

Un cliente es todo proceso que reclama servicios de otro, el cliente es el proceso que permite al usuario formular los requerimientos y pasarlos al servidor. Se lo conoce con el término front-end. Este normalmente maneja todas las funciones relacionadas con la manipulación y despliegue de datos, por lo que están desarrollados sobre plataformas que permiten construir interfaces gráficas de usuario (GUI), además de acceder a los servicios distribuidos en cualquier parte de la red.**[11]**

Las funciones que lleva a cabo el proceso cliente se resumen en los siguientes puntos:

- Administrar la interfaz de usuario.
- Interactuar con el usuario.
- Procesar la lógica de la aplicación y hacer validaciones locales.
- Generar requerimientos de bases de datos.
- Recibir resultados del servidor.

- Formatear resultados.

La funcionalidad del proceso cliente marca la operativa de la aplicación. De este modo el cliente se puede clasificar en:

- Cliente basado en aplicación de usuario. Si los datos son de baja interacción y están fuertemente relacionados con la actividad de los usuarios de esos clientes.
- Cliente basado en lógica de negocio. Toma datos suministrados por el usuario y/o la base de datos y efectúa los cálculos necesarios según los requerimientos del usuario.

Servidor

Un servidor es todo proceso que proporciona un servicio a otros. Es el proceso encargado de atender a múltiples clientes que hacen peticiones de algún recurso administrado por él. Al proceso servidor se lo conoce con el término back-end. El servidor normalmente maneja todas las funciones relacionadas con la mayoría de las reglas del negocio y los recursos de datos. Las principales funciones que lleva a cabo el proceso servidor se enumeran a continuación:

1- Aceptar los requerimientos de bases de datos que hacen los clientes.

2- Procesar requerimientos de bases de datos.

3- Formatear datos para trasmitirlos a los clientes.

4- Procesar la lógica de la aplicación y realizar validaciones a nivel de bases de datos.**[ídem]**

Puede darse el caso que un servidor actúe a su vez como cliente de otro servidor. Existen numerosos tipos de servidores, cada uno de los cuales da lugar a un tipo de arquitectura Cliente/Servidor diferente. Desde el punto de vista de la arquitectura cliente/servidor y del procesamiento cooperativo un servidor es un servicio software que atiende las peticiones de procesos software clientes.

Middleware

El middleware es un módulo intermedio que actúa como conductor entre sistemas permitiendo a cualquier usuario de sistemas de información comunicarse con varias fuentes de información que se encuentran conectadas por una red. En el caso que nos concierne, es el intermediario entre el cliente y el servidor y se ejecuta en ambas partes.

La utilización del middleware permite desarrollar aplicaciones en arquitectura Cliente/Servidor independizando los servidores y clientes, facilitando la interrelación entre ellos.

El middleware se estructura en tres niveles:

1- Protocolo de transporte.

2- Network Operating System (NOS).

3- Protocolo específico del servicio.

Las principales características de un middleware son:

1- Simplifica el proceso de desarrollo de aplicaciones al independizar los entornos propietarios.

2- Permite la interconectividad de los Sistemas de Información del Organismo.

3- Proporciona mayor control del negocio al poder contar con información procedente de distintas plataformas sobre el mismo soporte.

4- Facilita el desarrollo de sistemas complejos con diferentes tecnologías y arquitecturas.

5- Dentro de los inconvenientes más importantes destacan la mayor carga de máquina necesaria para que puedan funcionar. **[Ídem]**

Tipos de arquitectura Cliente/Servidor

Uno de los aspectos claves para entender la tecnología Cliente/Servidor, y por tanto contar con la capacidad de proponer y llevar a cabo soluciones de este tipo, es llegar a conocer la arquitectura de este modelo y los conceptos o ideas asociados al mismo. Más allá de entender los componentes cliente/middleware/servidor, es preciso analizar ciertas relaciones entre éstos, que pueden definir el tipo de solución que se ajusta de mejor forma a las estadísticas y restricciones acerca de los eventos y requerimientos de información que se obtuvieron en la etapa de análisis de un determinado proyecto. .**[Ídem]**

De hecho el analista deberá conocer estos eventos o restricciones del proyecto para que a partir de ahí, se puedan hacer las consideraciones y estimaciones de la futura configuración, teniendo en cuenta aspectos como por ejemplo, la oportunidad de la información, tiempo de respuesta, tamaños de registros, tamaño de bases de datos, estimaciones del tráfico de red, distribución geográfica tanto de los procesos como de los datos, etc.

Modelos Cliente/Servidor

Una de las clasificaciones mejor conocidas de las arquitecturas Cliente/Servidor se basa en la idea de planos, la cual es una variación sobre la división o clasificación por tamaño de componentes. Esto se debe a que se trata de definir el modo en que las prestaciones funcionales de la aplicación serán asignadas, y en qué proporción, tanto al cliente como al servidor. Dichas prestaciones se deben agrupar entre los tres componentes clásicos para Cliente/Servidor: interfaz de usuario, lógica de negocios y los datos compartidos, cada uno de los cuales corresponde a un plano. La mala elección de uno u otro modelo puede llegar a tener consecuencias fatales.

A nivel de software

Este enfoque o clasificación es el más generalizado y el que más se ajusta a los enfoques modernos, dado que se fundamenta en los componentes lógicos de la estructura Cliente/Servidor y en la madurez y popularidad de la computación distribuida. Por ejemplo, esto permite hablar de servidores de aplicación Modelos Cliente/Servidor

Una de las clasificaciones mejor conocidas de las arquitecturas Cliente/Servidor se basa en la idea de planos, la cual es una variación sobre la división o clasificación por tamaño de componentes. Esto se debe a que se trata de definir el modo en que las prestaciones funcionales de la aplicación serán asignadas, y en qué proporción, tanto al cliente como al servidor. Dichas prestaciones se deben agrupar entre los tres componentes clásicos para Cliente/Servidor: interfaz de usuario, lógica de negocios y los datos compartidos, cada uno de los cuales corresponde a un plano. Ni que decir tiene que la mala elección de uno u otro modelo puede llegar a tener consecuencias fatales. .**[Ídem]**

Modelo Cliente/Servidor 2 capas

Las aplicaciones cliente-servidor clásicas o de 2 capas como su nombre lo indica agrupan la lógica de presentación (interfaz) y la lógica de aplicación en la máquina cliente y acceden a fuentes de datos compartidos a través de una conexión de red que se encuentran en el servidor de datos. Estas aplicaciones de 2 capas trabajan bien en aplicaciones a escala de departamentos con un modesto número de usuarios, una base de datos sencilla y una red segura y rápida.

La ventaja que presenta este tipo de aplicaciones es que los datos están centralizados. Esta centralización beneficia a la empresa pues es más fácil compartir los datos.

Modelo Cliente/Servidor 3 capas

Esta estructura se caracteriza por elaborar la aplicación en base a dos capas principales de software, más la capa correspondiente al servidor de base de datos. Al igual que en la arquitectura dos capas, y según las decisiones de diseño que se tomen, se puede balancear la carga de trabajo entre el proceso cliente y el nuevo proceso correspondiente al servidor de aplicación.

En este esquema el cliente envía mensajes directamente al servidor de aplicación el cual debe administrar y responder todas las solicitudes. Es el servidor, dependiendo del tipo de solicitud, quien accede y se conecta con la base de datos.

Ventajas:

- Reduce el tráfico de información en la red por lo que mejora el rendimiento de los sistemas (especialmente respecto a la estructura en dos planos).
- Brinda una mayor flexibilidad de desarrollo y de elección de plataformas sobre la cual montar las aplicaciones. Provee escalabilidad horizontal y vertical.
- Se mantiene la independencia entre el código de la aplicación (reglas y conocimiento del negocio) y los datos, mejorando la portabilidad de las aplicaciones.
- Los lenguajes sobre los cuales se desarrollan las aplicaciones son estándares lo que hace más exportables las aplicaciones entre plataformas.
- Dado que mejora el rendimiento al optimizar el flujo de información entre componentes, permite construir sistemas críticos de alta fiabilidad.
- El mismo hecho de localizar las reglas del negocio en su propio ambiente, en vez de distribuirlos en la capa de interfaz de usuario, permite reducir el impacto de hacer mantenimiento, cambios urgentes de última hora o mejoras al sistema.
- Disminuye el número de usuarios (licencias) conectados a la base de datos. .**[ídem]**

Inconvenientes:

- Dependiendo de la elección de los lenguajes de desarrollo, puede presentar mayor complejidad en comparación con Cliente/Servidor dos planos.
- Existen pocos proveedores de herramientas integradas de desarrollo con relación al modelo Cliente/Servidor dos planos y normalmente son de alto costo.

Las distintas arquitecturas cliente/servidor presentan variaciones acerca de cómo son distribuidas las diferentes funciones de las aplicaciones de sistemas entre el cliente y el servidor, sobre la base de los conceptos de los tres componentes generales de cualquier sistema de información:

- **La lógica de acceso a datos.** Funciones que gestionan todas las interacciones entre el SW y los almacenes de datos (archivos, bases de datos, etc) incluyendo recuperación/consulta, actualización, seguridad y control de concurrencia.
- **La lógica de presentación.** Funciones que gestionan la interfaz entre los usuarios del sistema y el SW, incluyendo la visualización e impresión de formas y reportes, y la posibilidad de validar entradas del sistema.
- **La lógica de negocio o lógica de la aplicación.** Funciones que transforman entradas en salidas, incluyendo desde simples sumas hasta complejos modelos matemáticos, financieros, científicos, de ingeniería, etc.**[ídem]**

1.4.2- Arquitectura en Capas.

La arquitectura en capa define el patrón en capas como una organización jerárquica. Lo que posibilita un diseño basado en niveles de abstracción creciente, posibilitando a los implementadores particionar un problema en una secuencia de pasos incrementales. Este estilo de desarrollo en varios niveles, facilita que en caso que ocurra algún cambio, sólo se tendrían que realizar las correcciones necesarias en el nivel requerido sin tener que revisar código de otros niveles.

La programación por capas es un estilo de programación en la que el objetivo primordial es dividir, fraccionar y llegar a separar la Presentación (donde muestras y obtienes datos), de la Lógica de Negocio (donde realizas operaciones) y con todo esto se obtendría independencia, por lo que si hay cambios de arquitectura se puede acceder a los datos o cambiar el negocio o modificar la presentación y solo sería en esa fracción de la capa. **[12]**

En dichas arquitecturas a cada nivel se le confía una misión simple, lo que permite el diseño de arquitecturas escalables (que pueden ampliarse con facilidad en caso de que las necesidades aumenten).

La arquitectura en capas es algo referente a la arquitectura de software, pero realmente si una aplicación no tiene arquitectura de capas adecuada para el tipo de necesidad se podría decir que no cumple con la programación orientada a objetos. Una arquitectura como ésta puede tener tantas capas como se necesite, dependiendo siempre de la complejidad de la aplicación.

2- Capa de Presentación.

En esta capa se diseña todo lo que constituye la interfaz gráfica y la interacción del usuario con el software.

Se comunica únicamente con la capa de negocio. Representa el conjunto de componentes que genera la información que se representará en la interfaz de usuario del cliente. La norma es que la mayor parte de las interacciones con el usuario se realicen en las JSP debido a su flexibilidad, ya que integran etiquetas XHTML, HTML, XML y JSF.

La capa de presentación se implementa mediante tecnologías JSP, Servlets, con las que se construyen y conectan las interfaces gráficas con la capa de lógica del negocio. La función de la capa de presentación es la de proveer una interfaz para realizar la transferencia de datos. Es la encargada de interactuar con el usuario, obtener, validar y enviar los datos al servidor.

Los componentes de la interfaz de usuario deben mostrar datos al usuario, obtener y validar los datos procedentes del mismo e interpretar las acciones de éste que indican que desea realizar una operación con los datos. Asimismo, la interfaz debe filtrar las acciones disponibles con el fin de permitir al usuario realizar sólo aquellas operaciones que le sean necesarias en un momento determinado. Los componentes de interfaz de usuario: **[ídem]**

- No inicializan, participan ni votan en transacciones.
- Presentan una referencia al componente de proceso de usuario actual si necesitan mostrar sus datos o actuar en su estado.
- Pueden encapsular tanto la funcionalidad de visualización como un controlador.

Las interfaces web deben ser consistentes con la información que se requiere, no se deben utilizar más campos de los necesarios, así como la información requerida tiene que ser especificada de manera clara y concisa, no debe haber más que lo necesario en cada formulario. El objetivo principal de este componente es facilitar al usuario la interacción con la misma.

Dentro de la parte técnica, la capa de presentación contiene los objetos encargados de comunicar al usuario con el sistema mediante el intercambio de información, capturando y desplegando los datos necesarios para realizar alguna tarea.

3- Capa de lógica de negocio:

En esta capa residen los objetos de negocio que interactúan con los objetos de dominio. Se denomina capa de negocio (e incluso de lógica del negocio) pues es aquí donde se establecen todas las reglas que deben cumplirse para un correcto funcionamiento lógico de la aplicación.

Los objetos de negocio, separan los datos y la lógica de negocio haciendo uso del modelo de objetos definido, dichos objetos de negocio son los encargados de procesar la información proveniente tanto de la Presentación, como de la Capa de Acceso a Datos.

Los objetos de negocio son los objetos del modelo que implementan la lógica de negocio. Sus funciones fundamentales son:

- Realizar la validación de los datos introducidos por el usuario.
- Ejecutar la petición realizada por el usuario. Para ello podrá valerse de transacciones contra Sistemas Host, consultas a bases de datos, consultas a proveedores de contenidos, etc.
- Generar los objetos que utilizarán las Vistas para mostrar los resultados obtenidos.

Su función es garantizar que toda la lógica de negocio esté bien localizada y no mezclada con objetos de otras capas. Esto conlleva grandes ventajas, pues define los objetos de negocio, y crea las interfaces de servicio con sus implementaciones, las cuales hacen uso de los objetos de dominio. Además de encontrarse toda la lógica de la aplicación y el modelo de objetos encargados de la manipulación de los datos existentes, así como el procesamiento de la información ingresada o solicitada por el usuario en la capa de presentación.

En la misma se reciben solicitudes de la capa de presentación, la cual procesa y obtiene los resultados invocando a la capa de Acceso a Datos. La comunicación con otros sistemas que actúan en conjunto, se hace mediante esta capa, es en ella están expuestas el conjunto de funcionalidades o métodos a exportar. **[Ídem]**

La responsabilidad de esta capa es implementar lógica de negocio que será consumida por la interfaz del sistema y por otros sistemas a través de los servicios compartidos. Recibe los datos que ingresó el usuario del sistema mediante la capa de presentación, luego los procesa y crea objetos según lo que se necesite hacer con los datos.

Al encapsular los datos, la aplicación asegura mantener la consistencia de los mismos, así como obtener información precisa de las bases de datos e ingresar en las mismas solamente la información necesaria, a

través de la capa de acceso a datos, asegurando así no tener datos duplicados ni en las bases de datos, ni en los reportes solicitados por el usuario.

Existen tres tipos de componentes de negocio fundamentales:

- **Lógica de Negocio:** Implementan la funcionalidad de negocio del sistema.
- **Entidades de negocio:** Representan las entidades del sistema.
- **Workflow:** Implementan procesos de negocio en los cuales participan entidades y lógica de negocio. **[ídem]**

3- **Capa de Acceso a Datos:**

La capa de Acceso a Datos es el puente entre la capa de Lógica de Negocio y el Sistema de Base de Datos. Encapsula la lógica de acceso a datos. Aquí se encuentran componentes que hacen transparente el acceso a la base de datos. Este es el lugar idóneo para implementar los objetos de acceso a datos, permitiendo ingresar, obtener, actualizar y eliminar información del Sistema de Bases de Datos.

Los Objetos de Acceso a Datos (DAOS) encapsulan la persistencia de los objetos de dominios, proveen la persistencia de los objetos transitorios y las actualizaciones de los objetos existentes en la base de datos.

La misma es la responsable de unir todos los índices de funciones y estructuras que devuelven los servicios y combinarlos en una macro estructura. Su función es mantener sincronizada en todo momento la información existente en el sistema web y la que se encuentra en la base de datos. También permite el intercambio de información con clientes y proveedores.

Esta capa de acceso a datos se relaciona directamente con el patrón DataMapper pues permite manejar toda la carga y almacenamiento entre el modelo de dominio y la base de datos, logrando que ambos varíen independientemente. DataMapper permite disponer de dos objetos. El primero representa la lógica de negocio. El segundo el acceso a datos. Se trata de realizar una correlación entre objetos de lógica de negocio y tablas de la base de datos. **[Ídem]**

Luego de ser tratados los conceptos más importantes y necesarios para lograr una mejor comprensión del tema, se requiere tener bien definidos los requisitos y las herramientas que van a ser utilizadas para el desarrollo de la arquitectura.

Capítulo 2 – Concepción del módulo Webphone

El desarrollo de la arquitectura de software es una de las etapas fundamentales y, en muchos casos, la más importante en el desarrollo de software, pues es aquí donde los profesionales aportan todos sus conocimientos, creatividad y experiencia para crear la mejor propuesta de solución que se dará al cliente que cumpla con los requerimientos funcionales y no funcionales establecidos para el sistema en desarrollo, así como sus preocupaciones principales de lo que esperan del sistema.

Luego de ser caracterizados los puntos fundamentales para definir un webphone, se debe definir los requisitos técnicos y operacionales a partir del diseño de una arquitectura.

2.1 Concepción y definición del módulo Webphone para NovaDesk

2.1.1- Solución propuesta

Cuando se inicia el proceso de comunicación entre un usuario y uno de los técnicos de NovaDesk, primeramente el usuario se autentica en el portal web e inicia su sesión. Esto implica que el mismo pueda acceder a las distintas opciones que brinda el portal, en este caso acceder al módulo que permite la comunicación Volp. El módulo se encargará de establecer la correcta comunicación en el portal, ayudando a una mejor identificación del tipo de problema que presenta el solicitante de ayuda. Mejorará el intercambio de información entre el cliente y el técnico, pues las dudas se expresaran de forma clara al igual que las respuestas a las mismas.

2.1.2- ¿Cómo sería un webphone para NovaDesk?

Webphone es un módulo que brindará la emisión y recepción de llamadas telefónicas mediante un entorno web. Su interfaz es sencilla debido a que no `posee muchas funcionalidades.

2.13- Misión

Establecer comunicaciones telefónicas mediante la web, entre uno de los técnicos de Nova y los usuarios que interactúan con este sistema operativo. Garantizar una correcta comunicación entre los mismos. Brindar una interfaz de fácil usabilidad para el usuario.

2.1.4- Visión

Lograr una correcta y rápida comunicación entre los técnicos y el usuario. Que este módulo pueda ser utilizado desde cualquier lugar de la Universidad de las Ciencias Informáticas y que en un futuro se use desde lugares exteriores a la Universidad.

2.2- *Dominio y Requisitos*

En la fase de definición de la metodología de desarrollo SXP se identifican los requisitos claves del sistema y del software. Esta cuenta con tres tareas principales: ingenierías de sistemas o de información, planificación del proyecto y análisis de los requisitos. (Peñalver Gladys.2008)

Los requisitos que se capturan son priorizados y expresados en una Lista de Reserva del Producto. Cuando un proyecto comienza es muy difícil tener claro todos los requerimientos sobre el producto. Sin embargo, suelen surgir los más importantes que casi siempre son más que suficientes para una Iteración. Aunque puede crecer y modificarse a medida que se obtiene más conocimiento acerca del producto y del cliente. Con la restricción de que solo puede cambiarse entre las iteraciones.

Siempre el objetivo es asegurar que el producto definido al terminar la lista es el más correcto, útil y competitivo posible y para esto la lista debe acompañar los cambios en el entorno y el producto. En la confección de esta lista participan el Jefe de proyecto, el gerente, el cliente y el equipo de proyecto.

2.2.1- Modelo de Dominio.

Un modelo del dominio captura los tipos de objetos más importantes o los eventos que suceden en el entorno donde estará el sistema.

Este modelo es el utilizado en esta investigación debido a la poca estructuración de los procesos del negocio. A continuación se definen los conceptos fundamentales a utilizar en el modelo de dominio.

Definición de Conceptos fundamentales

Cliente: persona que accede al módulo para iniciar la comunicación.

Webphone: módulo que permite establecer la comunicación VoIP.

Llamada: comunicación entre el cliente y el técnico.

Técnicos: personas que reciben la llamada.

- **Diagrama de clases de Dominio**

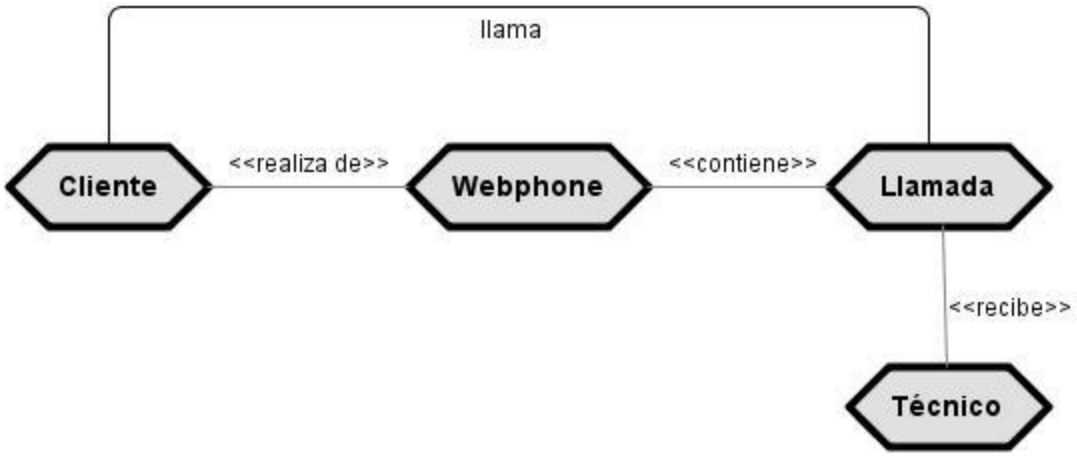


Ilustración 1Modelo de Dominio

Descripción del modelo de Dominio

El cliente es un usuario que realiza una llamada a través del módulo Webphone con el objetivo de solicitar ayuda a un técnico. Un cliente puede realizar solo una llamada a la vez, la cual es atendida por uno de los técnicos que estén disponibles en ese momento para hablar.

2.3 – Herramientas Metodología y Tecnologías asociadas al desarrollo del sistema.

En el desarrollo de todo sistema informático es de vital importancia la selección de las herramientas, lenguajes y tecnologías a utilizar, paso que garantizará, de realizarse correctamente, un óptimo desempeño del sistema. Para el desarrollo del módulo Webphone la selección se realizó valorando que el sistema a desarrollar, estaría orientado a funcionar sobre el sistema operativo GNU/Linux.

2.3.1- Central VoIP: Asterisk [AST]

Se plantea utilizar el servidor Asterisk PBX que es el más poderoso, flexible y extenso software de telecomunicaciones de código abierto.

Este servidor está diseñado para conectar cualquier hardware telefónico o cualquier tipo de software de telefonía de manera transparente y consistente. Tradicionalmente, los productos telefónicos son diseñados para ejecutar una tarea específica en una red. Sin embargo, gran cantidad de aplicaciones de telefonía

comparten gran cantidad de tecnología. Asterisk toma ventaja de esta sinergia para crear un solo entorno de desarrollo que puede ser moldeado a cualquier necesidad que el usuario requiera. **[13]**

Los protocolos de comunicación soportados para realizar VoIP son ADSI, IAX, SIP y H.323.

También es importante destacar los servicios que brinda este servidor que algunos serán útiles en el desarrollo de este módulo.

- Transferencia de llamadas, internas y externas.
- Desvío de llamadas si está ocupado o no contesta.
- Opción No molestar (Do Not Disturb).
- Parking de llamadas (Call Parking).
- Llamada en espera (Hold).
- Grupos de llamada (Ring groups).
- Identificador de llamante (CallerID).
- Sistema DISA12. (método por el cual una persona externa a la oficina puede realizar llamadas a través de la central).
- Operadora Digital (menús interactivos y guiados).
- Música en espera y en transferencia (ficheros MP3 actualizables por el usuario).
- Captura de llamadas de forma remota (remote pickup).
- Buzones de voz (general, individuales, por grupos) protegidos por contraseña.
- Gestión de listas negras (números telefónicos con acceso prohibido).
- Salas de conferencia (2 o más terminales simultáneamente).
- Gestión de colas de llamadas entrantes.
- Grabación de llamadas entrantes y salientes.
- Monitorización de llamadas en curso.
- Soporta videoconferencia con protocolos SIP e IAX2. **[ídem]**

2.3.3- Protocolos de Control de Llamadas y Señalización: IAX2 (Inter Asterisk Exchange)

Para interactuar con esta Central telefónica (Asterisk) es necesario utilizar un protocolo para trasmitir media, se decidió escoger el protocolo IAX2, ya que este congestiona en menor medida los canales de comunicación empleados pues sus mensajes son binarios y no de texto como los del protocolo SIP.

Proporciona control y transmisión de flujos de datos multimedia sobre redes IP, cuyas principales aplicaciones son videoconferencias y presentaciones remotas Establece sesiones internas que pueden utilizar cualquier códec para transmisión de voz o video. **[14]**

2.3.4- Ingeniería de software asistida por computadoras (CASE): Visual Paradigm for UML

Se empleará Visual Paradigm for UML (VP-UML) como herramienta de diseño UML que permite modelar los artefactos generados durante todo el desarrollo del proyecto; así como para diseñar los prototipos de interfaz de usuario.

2.3.5- Sistema de control de versiones: (Subversion)

Este sistema de control de versiones permite tener un control de los artefactos y códigos de los entregables durante todo el desarrollo del proyecto hasta obtener el producto final. Y como herramienta para conectarse a Subversion y realizar operaciones sobre el repositorio se elige el Rapidsvn, por su rapidez y facilidad de uso.

2.3.6- Lenguaje de programación: Applet de Java

Un lenguaje de programación es un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Es utilizado para controlar el comportamiento físico y lógico de una máquina.

La alternativa seria utilizar el Applet de java, ya que java cuenta con librerías que permiten interactuar con IAX2, la desventaja del mismo es que para ejecutarlo del lado del cliente debe tener instalado una maquina virtual de java, además teniendo en cuenta que java es un lenguaje intermedio, seria una dificultad en cuanto a velocidad y eficiencia de la aplicación.

2.3.7- Entornos integrados de desarrollo (IDE): NetBeans

Un entorno de desarrollo integrado o, en inglés, Integrated Development Environment ('IDE'), es un programa compuesto por un conjunto de herramientas para un programador. Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación. Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes.

El webphone elaborado como un Applet de Java requiere un entorno como Netbeans con diversas funcionalidades para trabajar con el lenguaje Java. Además el diseño de la interfaz gráfica también se puede realizar utilizando de Netbeans sin necesidad de requerir a un mayor número de herramientas.

2.3.8- Metodología Ágil de Desarrollo: SXP

2.4 – Estructura de la distribución física del sistema

La distribución física del sistema define la estructuración física que tendrá el mismo, estas estructuras son representadas a través de nodos que responden a los elementos hardware que intervienen en el negocio y sobre los cuales se ejecutarán los elementos software.

Dentro del contenido web del portal de Herramienta Soporte Técnico NovaDesk el usuario encontrará el módulo Webphone que le permitirá realizar llamada gracias al protocolo IAX2, la implementación de este protocolo (IaxClient) se comunicará con el servidor VoIP (Asterisk) el cual recibirá los mensajes para hacer una llamada, colgar una llamada, entre otras acciones.

A continuación se muestra la estructura de la distribución física correspondiente al módulo Webphone:

Ilustración 2 Estructura física del módulo Webphone

2.5 - Lista de Reserva del producto de Webphone

La lista de reserva del producto es una lista priorizada que define el trabajo que se va a realizar en el proyecto. Cuando un proyecto comienza es muy difícil tener claro todos los requerimientos sobre el producto. Sin embargo, suelen surgir los más importantes que casi siempre son más que suficientes para una Iteración. Aunque puede crecer y modificarse a medida que se obtiene más conocimiento acerca del producto y del cliente. Con la restricción de que solo puede cambiarse entre las iteraciones. (Peñalver Gladys.2008)

Siempre el objetivo es asegurar que el producto definido al terminar la lista es el más correcto, útil y competitivo posible y para esto la lista debe acompañar los cambios en el entorno y el producto. Considerar que los elementos que tienen más o menos influencian en el éxito del proyecto en un momento dado; por lo que los elementos con mayor prioridad son los que se realizan primero.

Tabla 1 Lista de Reserva del Producto de Webphone

RF (Requisitos Funcionales)				
Asignado a	**Item***	**Descripción**	**Estimación**	**Estimado por**
Módulo Webphone				
	Prioridad	***Alta***		
José Gustavo	12	Iniciar Llamada	2	Raydelmis
José Gustavo	13	Contestar Llamada	1	Raydelmis
José Gustavo	14	Terminar Llamada	2	Raydelmis
	Prioridad	***Media***		
José Gustavo	15	Habilitar Volumen.	5días	Raydelmis

José Gustavo	16	Deshabilitar Volumen	1día	Raydelmis
José Gustavo	17	Modificar Volumen	3 días	Raydelmis
Prioridad		***Baja***		
José Gustavo	18	Solicitar Transferencia de Llamada	1	Raydelmis
RNF (Requisitos No Funcionales)				
	1	Crear una interfaz sencilla y amigable desde la cual se pueda acceder a las distintas opciones que brinda el módulo.		
	2	Cualquier persona puede utilizarlo, sin importar su conocimiento de cómputo,		
	3	El módulo podrá funcionar en una red de buen rendimiento.		
	4	El módulo deberá brindar servicios siempre y cuando esté conectada la persona encargada de dar soporte a las incidencias.		

	5	El módulo admite hablar simultáneamente con varias personas.		
	6	La comunicación se establecerá únicamente entre los desarrolladores y usuarios y no entre usuarios.		
	7	Deben ser utilizadas herramientas libres para su desarrollo.		
	8	Las llamadas que las personas pueden hacer son a través del dispositivo VoIP o contactar directamente con un teléfono móvil.		
	9	Basado en IAX,		
	10	El navegador web debe soportar java.		
	11	La interfaz del módulo debe estar relacionada con la identidad del proyecto SIMAYS.		

2.6 – Historias de usuarios y tareas de ingeniería

Una historia de usuario es una representación de un requerimiento de software escrito en una o dos frases utilizando el lenguaje común del usuario. Las historias de usuario son utilizadas en las metodologías de desarrollo ágiles para la especificación de requerimientos. Tienen el mismo propósito que los casos de uso pero existen diferencias entre estas y la tradicional especificación de requisitos. La principal diferencia es

el nivel de detalle. Las historias de usuario solamente proporcionaran los detalles sobre la estimación del riesgo y cuánto tiempo conllevará la implementación de dicha historia de usuario. (Peñalver Gladys.2008)

Las tareas de ingeniería definen cada una de las actividades asociadas a las HU y permiten organizar el proceso de implementación, así como conocer el grado de complejidad de cada HU, teniendo en cuenta la cantidad de tareas asociadas.

Conocida la lista de reserva del producto se pueden definir las historias de usuarios arquitectónicamente significativos para el desarrollo del módulo de Webphone.

2.6.1- Iniciar Llamada

Tabla 2Historia de Usuario Iniciar Llamada

		Historia de Usuario
Número: 1	**Nombre de Historia de Usuario:** Iniciar Llamada	
Modificación de Historia de Usuario Número: 1		
Usuario: José Gustavo Suarez Mantilla		**Iteración Asignada:** 1
Prioridad en Negocio: Muy alta		**Puntos Estimados:** 2 semanas
Riesgo en Desarrollo: Medio		**Puntos Reales:**
Descripción: Es una comunicación directa por voz, sencilla de establecer por el cliente		
Observaciones:		
Prototipo de Interfaz:		

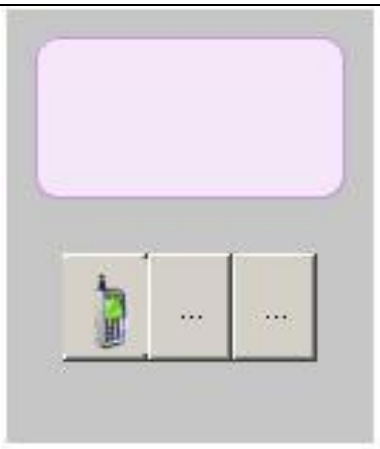


Ilustración 3 Prototipo de Iniciar Llamada

2.6.2- Contestar Llamada

Tabla 3Historia de Usuario Contestar Llamada

		Historia de Usuario
Número: 2	**Nombre de Historia de Usuario:** Contestar Llamada	
Modificación de Historia de Usuario Número: 1		
Usuario: José Gustavo Suarez Mantilla		**Iteración Asignada: 1**
Prioridad en Negocio: Muy alta		**Puntos Estimados:** 2 semanas
Riesgo en Desarrollo: Medio		**Puntos Reales**
Descripción: Se encarga de responder a llamadas que se reciben por parte de los usuarios.		
Observaciones:		
Prototipo de Interfaz:		

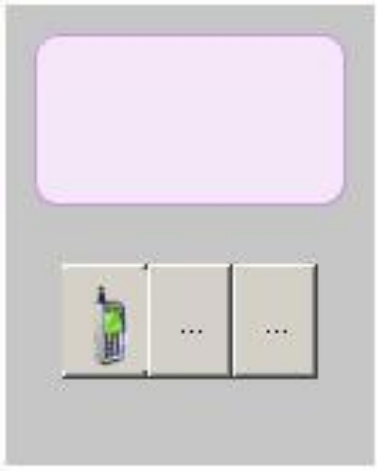


Ilustración 4 Prototipo de Contestar Llamada

2.6.3- Gestionar volumen

Tabla 4Historia de Usuario Gestionar volumen

		Historia de Usuario
Número: 3	**Nombre de Historia de Usuario:** Gestionar volumen	
Modificación de Historia de Usuario Número: 1		
Usuario: José Gustavo Suarez Mantilla		**Iteración Asignada: 2**
Prioridad en Negocio: Muy Alta		**Puntos Estimados:** 1
Riesgo en Desarrollo: Medio		**Puntos Reales:** [Tiempo real]
Descripción: Las funcionalidades que se desarrollan en esta historia de usuario son: 1- Habilitar audio 2- Deshabilitar audio 3- Modificar audio		
Observaciones:		
Prototipo de Interfaz:		

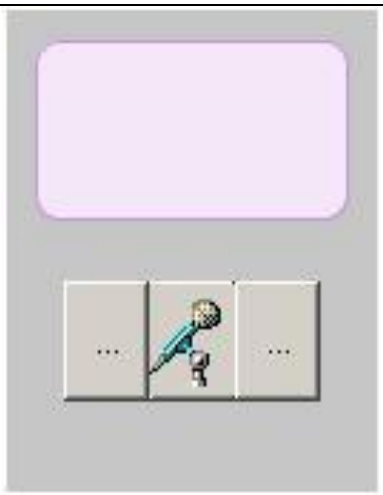


Ilustración 5 Prototipo de Gestionar Audio

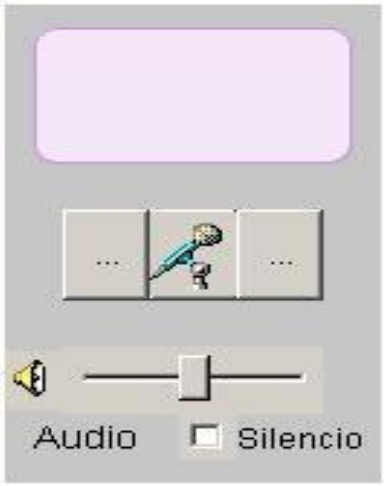


Ilustración 6Prototipo de Habilitar Volumen

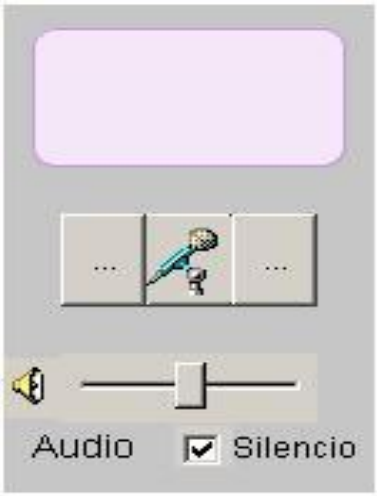


Ilustración 7 Prototipo de Modificar Volumen

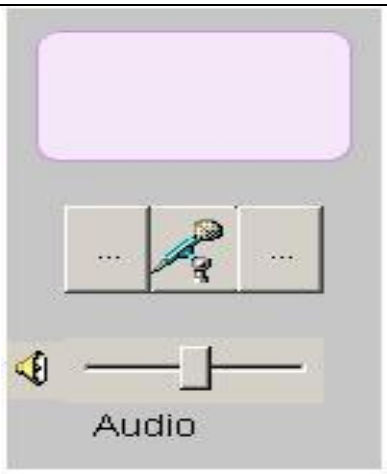


Ilustración 8 Prototipo de Deshabilitar Volumen

Tareas de ingenierías asociadas a la Historia de Usuario

Tabla 5Tarea de ingeniería Habilitar Volumen

Tarea de Ingeniería	
Número Tarea: 1	**Número Historia de Usuario:** Gestionar volumen
Nombre Tarea: Habilitar volumen	
Tipo de Tarea: Desarrollo.	**Puntos Estimados:** 4 días.
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: José Gustavo Suarez Matilla	
Descripción: El usuario solicita la opción de habilitar volumen luego de ser establecida la comunicación.	

Tabla 6Tarea de ingeniería Deshabilitar Volumen

Tarea de Ingeniería	
Número Tarea: 2	**Número Historia de Usuario:** Gestionar volumen

Nombre Tarea: Deshabilitar volumen	
Tipo de Tarea: Desarrollo.	**Puntos Estimados:** 3 días
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: José Gustavo Suarez Matilla	
Descripción: El usuario solicita la opción de deshabilitar volumen luego de ser establecida la comunicación.	

Tabla 7Tarea de ingeniería Modificar Volumen

Tarea de Ingeniería	
Número Tarea: 3	**Número Historia de Usuario:** Gestionar volumen
Nombre Tarea: Modificar volumen	
Tipo de Tarea: Desarrollo.	**Puntos Estimados:** 1 semana.
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: José Gustavo Suarez Matilla	
Descripción: El usuario solicita la opción de subir o bajar el volumen según este lo estime conveniente.	

2.6.4- Solicitar Transferencia de Llamada

Tabla 8Historia de Usuario Solicitar Transferencia de Llamada

Historia de Usuario	
Número: 4	**Nombre de Historia de Usuario:** Solicitar Transferencia de Llamada
Modificación de Historia de Usuario Número: 1	
Usuario: : José Gustavo Suarez Matilla	**Iteración Asignada:** 3

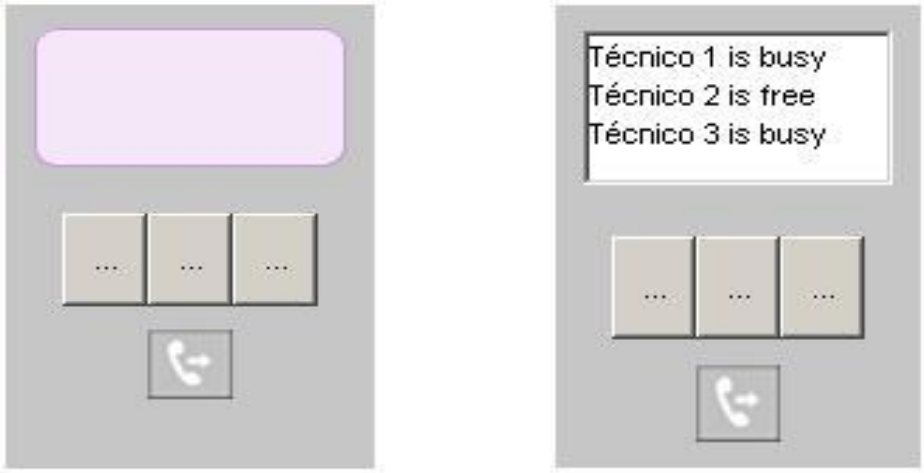
Prioridad en Negocio: Muy alta	**Puntos Estimados:** 1 semanas
Riesgo en Desarrollo: Medio	**Puntos Reales:**
Descripción: El usuario solicita transferir una llamada luego de ser establecida la comunicación	
Observaciones: Se muestra una lista con el estado de los técnicos para saber a cual se le transfiere la llamada.	
Prototipo de Interfaz: 	

Ilustración 9 Prototipo de Transferir Llamada

2.6.5- Terminar Llamada

Tabla 9 Historia de Usuario Terminar Llamada

		Historia de Usuario
Número: 5	**Nombre de Historia de Usuario:** Terminar Llamada	
Modificación de Historia de Usuario Número:		
Usuario: : José Gustavo Suarez Mantilla		**Iteración Asignada:** 1
Prioridad en Negocio: Muy alta		**Puntos Estimados:** 1 semanas
Riesgo en Desarrollo: Medio		**Puntos Reales:**

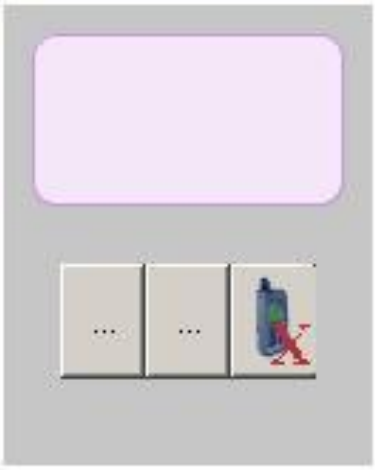
Descripción: El usuario decide finalizar la comunicación.
Observaciones:
Prototipo de Interfaz: 

Ilustración 10 Prototipo de Terminar Llamada

2.7 – Plan de Release del módulo Webphone.

El plan de raleases no es más que una plantilla donde se recogen las iteraciones a realizar con sus características, además del orden de las historias de usuario con su planificación estimada para ser implementadas. (Peñalver Gladys.2008)

Tabla 10Plan de Release

Release	Descripción de la iteración	Orden de la HU a implementar	Duración total
1	Desarrollo de las Historias de Usuario de alta prioridad.	1,2,5	4 semanas

2	Desarrollo de las Historias de Usuario que tienen prioridad media.	3	3 semanas
3	Desarrollo de las Historias de Usuario de prioridad baja	4	1 semana

En el presente capítulo se ha realizado un modelo de dominio para ofrecer una visión general de cómo se desarrolla este proceso, dando a conocer cuáles son los conceptos fundamentales que faltan para la realización y compresión del módulo en general. Se determinaron los requerimientos que va a tener la aplicación y con los que debe además cumplir, se realizaron las descripciones textuales de cada uno de los casos de uso identificados, ofreciendo mayor claridad en cuanto a la compresión de cómo debe funcionar el sistema.

Capítulo 3 – Descripción de la Arquitectura

3.1- Estructuración de los componentes

Los patrones arquitectónicos expresan una organización estructural para un sistema de software. Proveen un conjunto de subsistemas predefinidos e incluyen reglas y lineamientos para conectarlos. **[16]**

3.1.1- Fundamentación de los patrones utilizados

La descripción de la arquitectura es importante y a la vez necesaria para el desarrollo de un sistema de software, pues constituye una expresión del sistema y de su evolución, permite una mejor comunicación entre los clientes y los desarrolladores, así como la evaluación y comparación de arquitecturas de forma consistente. Facilita la planificación, gestión y ejecución de las actividades del desarrollo del sistema ya que es una forma de verificar la correcta implementación de acuerdo a la descripción arquitectural. **[Ídem]**

3.1.1.1- Estilos arquitectónicos

Los patrones arquitectónicos expresan una organización estructural para un sistema de software. Proveen un conjunto de subsistemas predefinidos e incluyen reglas y lineamientos para conectarlos. **[17]**

Arquitectura Cliente/Servidor

En los nodos clientes generalmente encontramos presentación de usuario y en los nodos servidores la lógica del negocio. La arquitectura cliente/servidor es una forma de dividir y especializar programas y equipos de cómputo de forma que la tarea que cada uno de ellos realiza se efectúa con la mayor eficiencia posible y permita simplificar las actualizaciones y mantenimiento del sistema. **[18]**

Esta arquitectura se propone para el módulo Webphone en general, este módulo se basa en un clásico modelo cliente/servidor, donde los clientes son los usuarios que solicitan la comunicación, y el servidor es la computadora que les proveerá a los clientes el portal web que posee este módulo, este servidor se activa y espera la solicitud de los clientes, la procesa y envía el resultado obtenido.

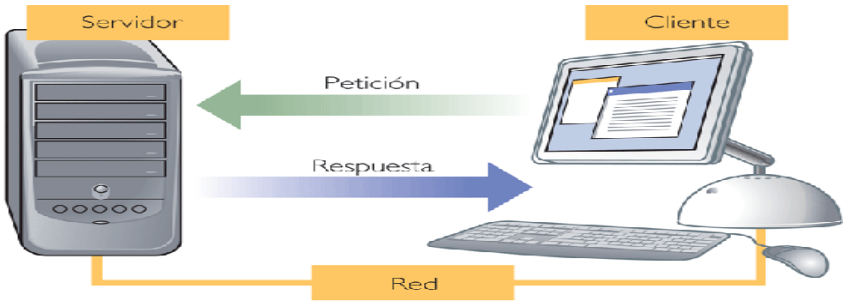


Ilustración 11 Arquitectura Cliente/Servidor

Arquitectura en Capas

La arquitectura de software es, a grandes rasgos, una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se la percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema. La arquitectura de un sistema es la vista conceptual de la estructura de este. Toda aplicación contiene código de presentación, código de procesamiento de datos y código de almacenamiento de datos. Un elemento importante dentro de la arquitectura es la forma en que se va a integrar todo el negocio del sistema, para lo cual se debe definir la estructura del software y establecer una estrategia de integración de las diferentes partes o componentes. **[18]**

Se propone el diseño de una arquitectura de software distribuida, dicha propuesta es la utilización de una **arquitectura cliente/servidor en 2 capas**, lo cual simplifica la comprensión y la organización del sistema, reduciendo las dependencias de forma que las capas más bajas no sean conscientes de ningún detalle o interfaz de las superiores.

La calidad tan especial de este estilo arquitectónico consiste en separar la lógica de la aplicación y convertirla en una capa intermedia bien definida. Esta separación entre la lógica de aplicación de la interfaz de usuario añade flexibilidad al diseño de la aplicación. Pueden construirse y desplegarse múltiples interfaces de usuario sin cambiar en absoluto la lógica de aplicación siempre que esté presente una interfaz claramente definida a la capa de presentación.

Para el módulo Webphone las 2 capas que se definieron fueron: Presentación (Interfaz de usuario), Lógica de Negocio o Dominio (tareas y reglas que rigen el proceso).

- **Capa de Presentación**

Es la que interactúa directamente con el usuario, captura la información entrada por éste y hace las peticiones a la capa inferior mostrando al usuario la respuesta proveniente de ésta. Esta capa esta conformada por el paquete jiaxTest. Únicamente se comunica con la capa de lógica de negocio

- **Capa de Lógica de Negocio**

Está conformada por los paquetes que integran el sistema, estos paquetes son net.sourforge.iaxclient y net.sourceforge.iaxclient.jni los cuales se ajustan a las historias de usuarios arquitectónicamente significativas y a los requisitos. Únicamente se comunica con la capa de Acceso a Datos. **[Ídem]**

3.1.1.1- Patrones de Diseño

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces.

Un patrón de diseño es una solución a un problema de diseño. Para que una solución sea considerada un patrón debe poseer ciertas características. Una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores. Otra es que debe ser reusable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias. **[19]**

No es obligatorio utilizar los patrones, solo es aconsejable en el caso de tener el mismo problema o similar que soluciona el patrón, siempre teniendo en cuenta que en un caso particular puede no ser aplicable. Abusar o forzar el uso de los patrones puede ser un error.

En el modelo de diseño del módulo de Webphone se tuvieron en cuenta los patrones de asignación de responsabilidades (GRASP) y los patrones GOF que se verán a continuación.

- *Bajo Acoplamiento*

Este patrón se utilizó con la idea de tener las clases lo menos ligadas entre sí posibles. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases.

- *Patrón Fachada (Facade)*

Este patrón se evidencia porque todas las solicitudes pasan a través de la clase PhonePanel ya que esta va a ser la intermediaria entre las peticiones de IAXTestApplet y la capa de dominio del sistema.

3.2- Modelo del diseño

Se define el diseño del sistema, sin entrar en especificaciones, ni detalles, solo lo que el diseñador necesita para hacer un primer entregable del sistema.

El paquete sobre el que se trabajó principalmente es jiaxTest. Otros paquetes necesarios y empleados son:

1- net.sourforge.iaxclient

2- ·net.sourceforge.iaxclient.jni

- jiaxTest: Este paquete es el desarrollado para la creación del Applet de Java. La clase PhonePanel es la encargada de la presentación visual.
- net.sourforge.iaxclient y net.sourceforge.iaxclient.jni: El paquete IaxClient y el paquete iaxclient.jni son los que contienen las clases que implementan el protocolo IAX y su manejo a un nivel más alto en Java. **[15]**

A continuación se muestra como fue diseñado este sistema en términos de clases y paquetes, en los anexos podrá encontrarse las clases por separado con sus atributos y métodos.

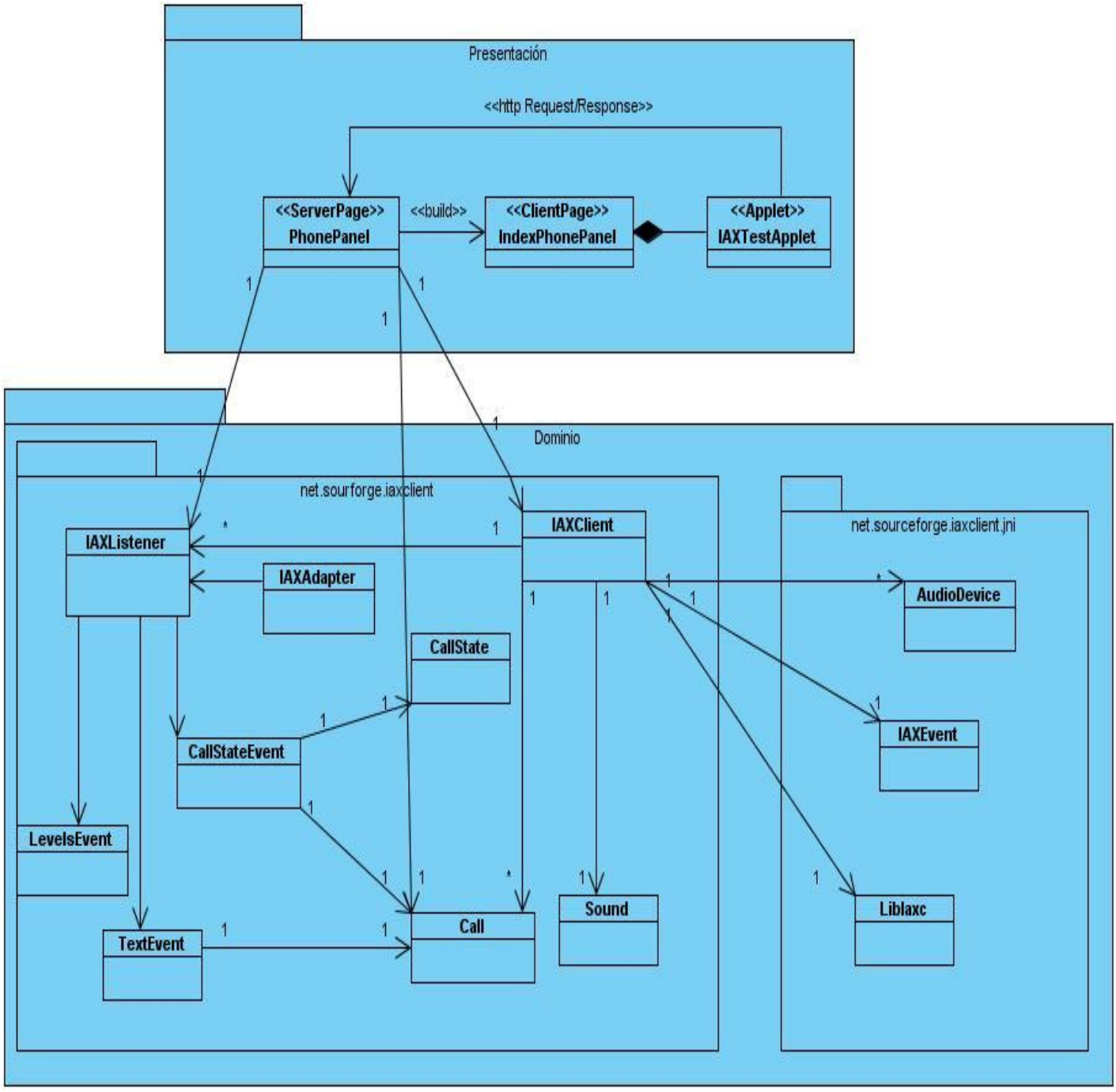


Ilustración 12 Diagrama de Clases del diseño del módulo Webphone

Las principales clases identificadas para el desarrollo del módulo Webphone son:

- IaxTestApplet y PhonePanel: Son las dos clases que implementan el Applet de Java. Contemplan la interfaz gráfica que permite la interacción del usuario.
- IaxClient: Es la clase principal que gestiona todo el módulo Webphone.
- Liblaxc: Es la clase que interactúa de forma cercana con el protocolo IAX.
- IAXEvent, IAXListener e IAXAdapter: Son las clases de las que se vale IaxClient para emplear las funcionalidades que posibilita el protocolo IAX.

IaxClient es una biblioteca multiplataforma que maneja el protocolo IAX2. Entre sus funcionalidades está la del manejo de varias llamadas, el control completo de los dispositivos de audio (selección de los dispositivos de entrada y salida, así como el tono del timbre, notificaciones de los niveles de audio, funcionalidades de mezcla de sonido con salidas).

3. 2.1- Clase IAXListener

Tabla 11 Clase IAXListener

Método	Descripción	Parámetros	Valor de Retorno
TextReceived	Devuelve el id del texto del evento	$e:TextEvent	int
Call	Devuelve el id del estado de una llamada	$e:Call State Event	int
LevelsChanged	Cambia el nivel	$e:LevelsEvent	int

3. 2.2- Clase IAXAdapter

Tabla 12Clase IAXAdapter

Método	Descripción	Parámetros	Valor de Retorno
TextReceived	Devuelve el id del texto del evento	$e:TextEvent	int
Call	Devuelve el id del estado de una llamada	$e:Call State Event	int
LevelsChanged	Cambia el nivel	$e:LevelsEvent	int

3. 2.3- Clase LevelsEvent

Tabla 13Clase LevelsEvent

Método	Descripción	Parámetros	Valor de Retorno
getInput	Obtiene el nivel de entrada del evento		float
getOutput	Obtiene el nivel de salida del evento		float

3. 2.4- Clase TextEvent

Tabla 14 Clase TextEvent

Método	Descripción	Parámetros	Valor de Retorno
getType	Obtiene el número del tipo del texto de evento		int
getCall	Obtiene una llamada		float
getMessage	Obtiene un mensaje		string

3. 2.5- Clase CallStateEvent

Tabla 15 Clase CallStateEvent

Método	Descripción	Parámetros	Valor de Retorno
getCall	Obtiene una llamada		Call
getState	Obtiene el estado de una llamada		CallState

3. 2.6- Clase CallState

Tabla 16 Clase CallState

Método	Descripción	Parámetros	Valor de Retorno
isFree	Comprueba si la línea está o no está libre		boolean
isActive	Comprueba si la línea está o no está activa		boolean
isRinging	Comprueba si la línea está o no está dando timbre		boolean
isComplete	Comprueba si la línea está o no está completa		boolean
isBusy	Comprueba si la línea está o no está ocupada		boolean
isTransfer	Comprueba si se transfiere o no la llamada		boolean

3. 2.7- Clase Call

Tabla 17 Clase Call

Método	Descripción	Parámetros	Valor de Retorno
getCallNo	Obtiene el número de la llamada		int
getState	Obtiene el estado de la llamada		CallState
setCallNo	Modifica el número de llamada	$ CallNo: int	boolean
setState	Modifica el estado de la llamada	$ state:CallState	boolean

3. 2.8- IaxClient

Tabla 18 IAXClient

Método	Descripción	Parámetros	Valor de Retorno
getInstance	Obtiene la clase instanceada		JIAXClient
Initialize	Se inicializa IaxClient	$audType:int; $nCalls.int	int
Call	Crea la llamada	$num: string	
answerCall	Respuesta a la llamada	$c: Call	
blindTransferCall	Transfiere la llamada	$c: Call; $number: string	
rejectCall	Rechaza llamada	$call:Call	
selectCall	Selecciona la llamada	$c: Call	
getFirstFreeCall	Obtiene la primera llamada libre		Call
getSelectCall	Obtiene la llamada seleccionada		Call
getAudioInput	Obtiene el audio de entrada		AudioDevice
getAudioOuput	Obtiene el audio de salida		AudioDevice
getAudioRing	Obtiene el audio del timbre		AudioDevice
setAudioDevice	Modifica los dispositivos del audio	$input: Audio Device; $ouput: Audio Device; $ring: Audio Device	
playSound	Hace que el sonido comience	$sound:Sound; $ring: boolean	
stopSound	Hace que el sonido se detenga	$sound:Sound	

addIAXListener	Crea la clase IAXListener	$l: IAXListener	
removeIAXListener	Elimina la clase IAXListener	$l: IAXListener	
Run	Compila		
isLibraryLoaded	Comprueba si la librería está o no está cargada		boolean
checkLibrary	Chequea la librería		boolean
loadLibrary	Carga la librería		
getCallNo	Obtiene el número de una llamada	$call:Call	int
getCall	Obtiene una llamada	$callNo: int	Call

3. 2.9- AudioDevice

Tabla 19 AudioDevice

Método	Descripción	Parámetros	Valor de Retorno
getName	Obtiene el nombre del dispositivo de audio		string
getCapabilities	Obtiene las capacidades del audio		long
getDevId	Obtiene el id del dispositivo		int

3. 2.10- IAXEvent

Tabla 20 IAXEvent

Método	Descripción	Parámetros	Valor de Retorno
getDate	Obtiene el día		Date

3. 2.11- Sound

Tabla 21 Sound

Método	Descripción	Parámetros	Valor de Retorno
Init	Inicializa el sonido mediante el tono	$tonelist: string	
Set	Modifica el sonido	$post: int; $freq:double; $freq2:double; $duration:double	int
Init	Inicializa el sonido mediante la frecuencia, la duración,etc	$freq1: double; $freq2:double; $duration:double; $proc:double	
Load	Carga el sonido mediante el nombre de un archive	$filename: String	Sound
Load	Carga el archive	$file: File	Sound
setRepeat	Modificar la forma de repetir el sonido.	$repeat: int	
getRepeat	Obtiene la forma de repetir el sonido.		int

3. 2.12- Liblaxc

Tabla 22 Liblaxc

Método	Descripción	Parámetros	Valor de Retorno
Shutdown	Termina la librería.		
setFormats	Modifica el formato de la librería.	$preferred:int; $allowed: int	
setMinOutgoingFramesize	Modifica el tamaño del minuto de salida de la aplicación.	$samples: int	
setCallerId	Modifica el id de la llamada	$name: String, $number: String	
processCall	Procesa la llamada		
Call	Asigna el número de la llamada.	$num. String	
dumpAllCalls	Muestra el historial de todas las llamadas		
dumpCall	Muestra el historial de una llamada		
rejectCall	Rechaza la llamada		
sendText	Envía un texto	$test: String	
setAudioOutput	Modifica el audio de salida	$mode: int	
getAudioDevices	Obtiene los dispositivos de audio.		AudioDevice
getInputLevel	Obtiene el nivel de salida		double

getOuputLevel	Obtiene el nivel de entrada		double
setInputLevel	Modifica el nivel de entrada	$level: double	int
setOuputLevel	Modifica el nivel de salida	$level: double	int

Como resultado del estudio hecho, se identificaron las clases, se desarrollaron los diagramas de clases de análisis, además se explicó como se aplican los patrones de diseños que fueron seleccionados. Finalmente quedó establecida la arquitectura del módulo webphone

Conclusiones

Se realizó un estudio sobre los diferentes webphone existentes a nivel mundial y se analizaron las diferentes características que fueran similares en cuanto a las funcionalidades del módulo a desarrollar, llegando a la conclusión de que la realización del mismo traerá grandes beneficios a los estudiantes de la Universidad.

Se realizó el análisis y diseño de una posible solución para la futura implementación del módulo de Webphone, obteniéndose como resultado todos los artefactos necesarios. También se realizó un estudio profundo acerca de varios elementos arquitectónicos que existen en la actualidad, para hacer uso de las mejores técnicas de diseño arquitectónico.

Con la implementación del módulo propuesto será posible una mejor calidad y eficiencia en la comunicación entre usuarios y técnicos.

.

Recomendaciones

Se recomienda realizar la implementación del módulo propuesto, así como agregar nuevas funcionalidades al sistema de acuerdo con las expectativas del cliente.

.

Bibliografía Referenciada

[1] http://www.webphone.es/faqs.php#g1

[2] http://www.skype.com/intl/es/home

[3] http://es.wikipedia.org/wiki/Skype#Requisitos_del_sistema

[4] http://www.freedownloadmanager.org/es/downloads/webphone_gratis/

[5] http://www.doddlephone.com/

[6] http://www.mizu-voip.com/Products/WebPhone.aspx

[7] http://www.netcomsatelital.com.ar/es/voip/porque-webphone.html

[8] http://www.soft4evolve.com/es/webphone/

[9] Douglas Schmidt, Michael Stal, Hans Rohnert, Frank Buschman, PATTERN-ORIENTED SOFTWARE ARCHITECTURE. Volumen 1.

[10] Ernst-Erich Doberkat. "Pipes and filters: Modelling a software architecture through relations". Internes Memorandum des Fachbereich Informatik Lehrstuhl für Software Technologie, Universidad de Dortmund, Memo N° 123, Junio de 2002.

[11] Mary Shaw y David Garlan. Software Architecture: Perspectives on an emerging discipline. Upper Saddle River, Prentice Hall, 1996.

[12] Steve Burbeck. "Application programming in Smalltalk-80: How to use Model-View-Controller (MVC)". University of Illinois in Urbana-Champaign, Smalltalk Archive, http://st-www.cs.uiuc.edu/users/smarch/st-docs/mvc.html.

[13]http://upcommons.upc.edu/pfc/bitstream/2099.1/6798/2/Implantaci%C3%B3n%20de%20un%20sistema%20VoIP%20basado%20en%20Asterisk.pdf

[14] http://bieec.epn.edu.ec:8180/dspace/bitstream/123456789/1412/4/T11370%20CAP%20I.pdf

[15]http://translate.googleusercontent.com/translate_c?hl=es&sl=en&u=http://www.scribd.com/doc/13301699/Tesis-FOSS&prev=/search%3Fq%3DIAXListener%26hl%3Des%26lr%3D%26biw%3D1280%26bih%3D561%26prmd%3Divns&rurl=translate.google.com&usg=ALkJrhgqVM98JBUGklBem_RWXjxWws7jrg

[16] David Garlan y Mary Shaw. “An introduction to software architecture”. CMU Software Engineering Institute Technical Report, CMU/SEI-94-TR-21, ESC-TR-94-21, 1994.

[17] Carlos Reynoso, N. K. (2004) Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.

[Disponible en: http://www.willydev.net/descargas/prev/Estiloypatron.pdf]

[18] Ríos GIL, J.J. Sistemas de Información. [en línea]. http://www.it.uc3m.es/: Departamento de Ingeniería Telemática Escuela Politécnica Superior Universidad Carlos III de Madrid, 13 de octubre de 2005 [citado 23 de febrero de 2007]. [Disponible en: http://www.it.uc3m.es/mcfp/docencia/si/material/1_cli-ser_mcfp.pdf]

[19] Craig Larman. UML y Patrones. 2a edición, Madrid, Prentice Hall, 2003.

Bibliografía Consultada

1- Carlos E. Cuesta. Arquitecturas de Software. Ingeniería del Software I, Universidad Rey Juan Carlos.

2- Carlos Reynoso, N. K. (2004) Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.

3- Clements, P. (1996). A Survey of Architecture Description Languages. Proceedings of the International Workshop on Software Specification and Design.

4- Douglas Schmidt, Michael Stal, Hans Rohnert, Frank Buschman, PATTERN-ORIENTED SOFTWARE ARCHITECTURE. Volumen 1.

5- Douglas Schmidt, Michael Stal, Hans Rohnert, Frank Buschman, PATTERN-ORIENTED SOFTWARE ARCHITECTURE. Volumen 2.

6- Kruchten, P. (1995). The 4+1 View Model of Architecture. IEEE Software.

7- Mary Shaw, D. G. (1996). Software Architecture. Perspectives on an Emerging Discipline, Prentice Hall.

8- Meneses Abad, Penalver Romero, Rodríguez Villar, Fernández Céspedes, and Pino García. Metodología ágil para el desarrollo de proyecto de software libre. March 2009. Available from World Wide Web: <http://gforge.f10.uci.cu/plugins/scmsvn/viewcvs.php/*checkout*/Metodologia_SXP.odt?root=magmpr>.

9- Robert Monroe, A. K., Ralph Melton y David Garlan (1997) Architectural Styles, design patterns, and objects. IEEE Software.

10- Roger S Presuman. 2005. Ingeniería del Software, Un enfoque práctico. 5 end.

11- Society, S. E. S. C. o. t. I. C. (2000). "IEEE Recommended Practice for Architectural Description of Software-Intensive Systems."Bosch, J. (2000). Design & Use of Software Architectures. Addison Wesley.

12- http://www.voip-info.org/wiki/view/Asterisk+manager+API

13- http://www.voip-info.org/wiki/view/Aynchronous+Javascript+Asterisk+Manager+(AJAM)

Anexos

Anexo 1: Clases de Diseño

Anexo1.1: Paquete jiaxTest

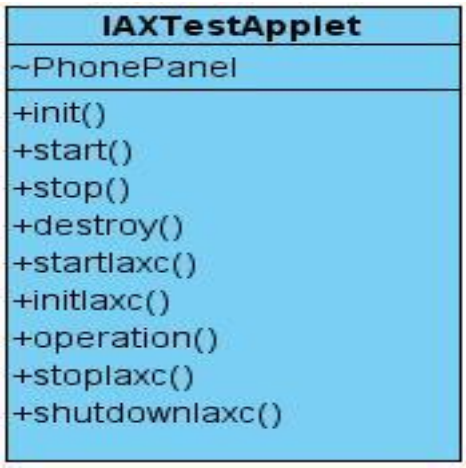


Ilustración 13 Clase IAXTestApplet

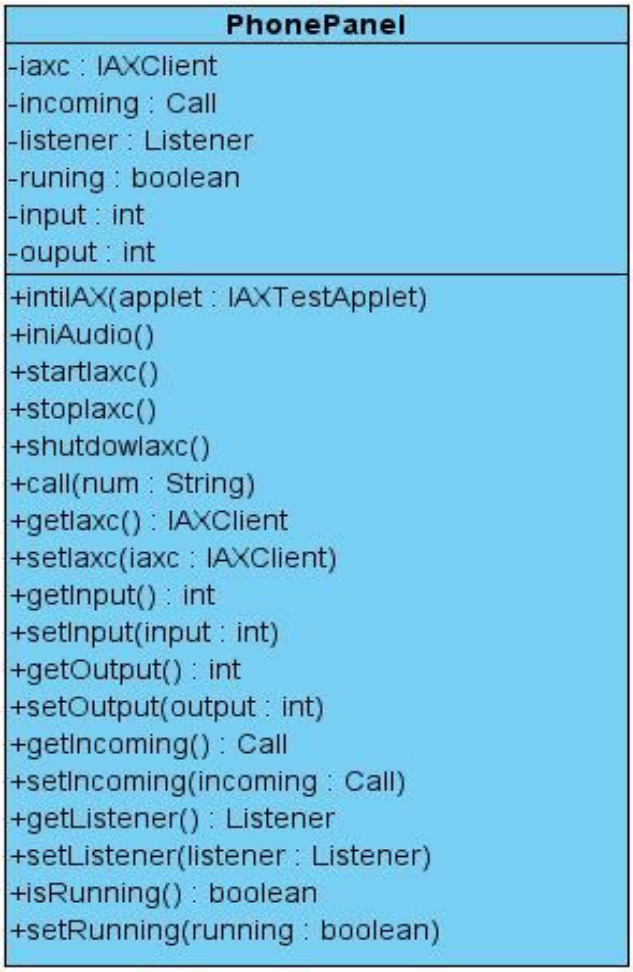


Ilustración 14 Clase PhonePanel

Anexo1.2: Paquete net.sourforge.iaxclient

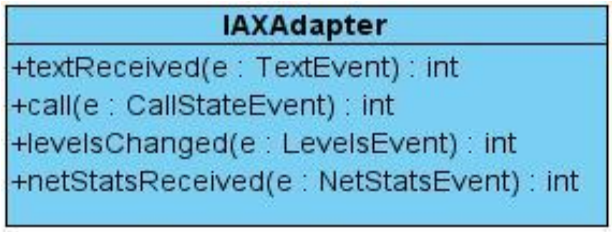


Ilustración 15 Clase IAXAdapter

Ilustración 16 Clase IAXListener

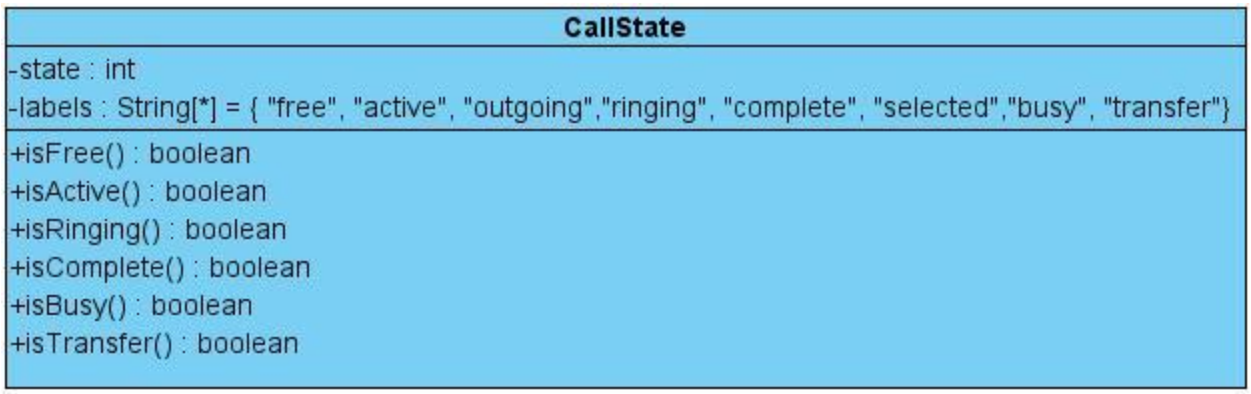


Ilustración 17 Clase CallState

Ilustración 18 Clase CallStateEvent

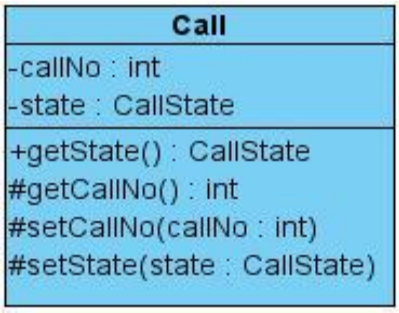


Ilustración 19 Clase Call

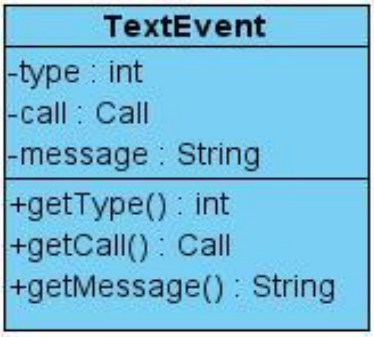


Ilustración 20 TextEvent

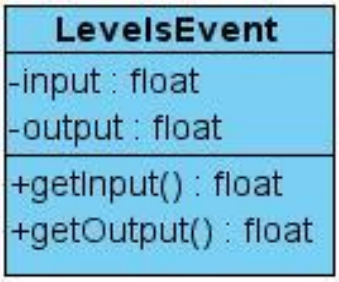


Ilustración 21 LevelsEvent

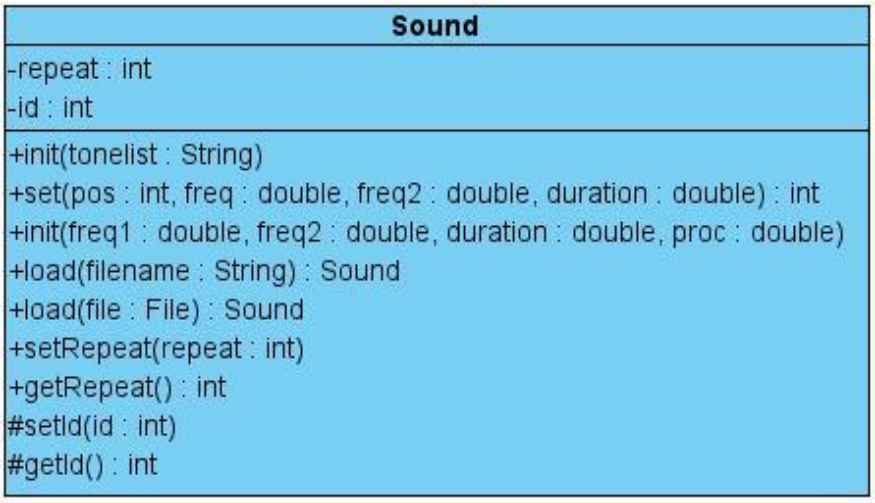


Ilustración 22 Sound

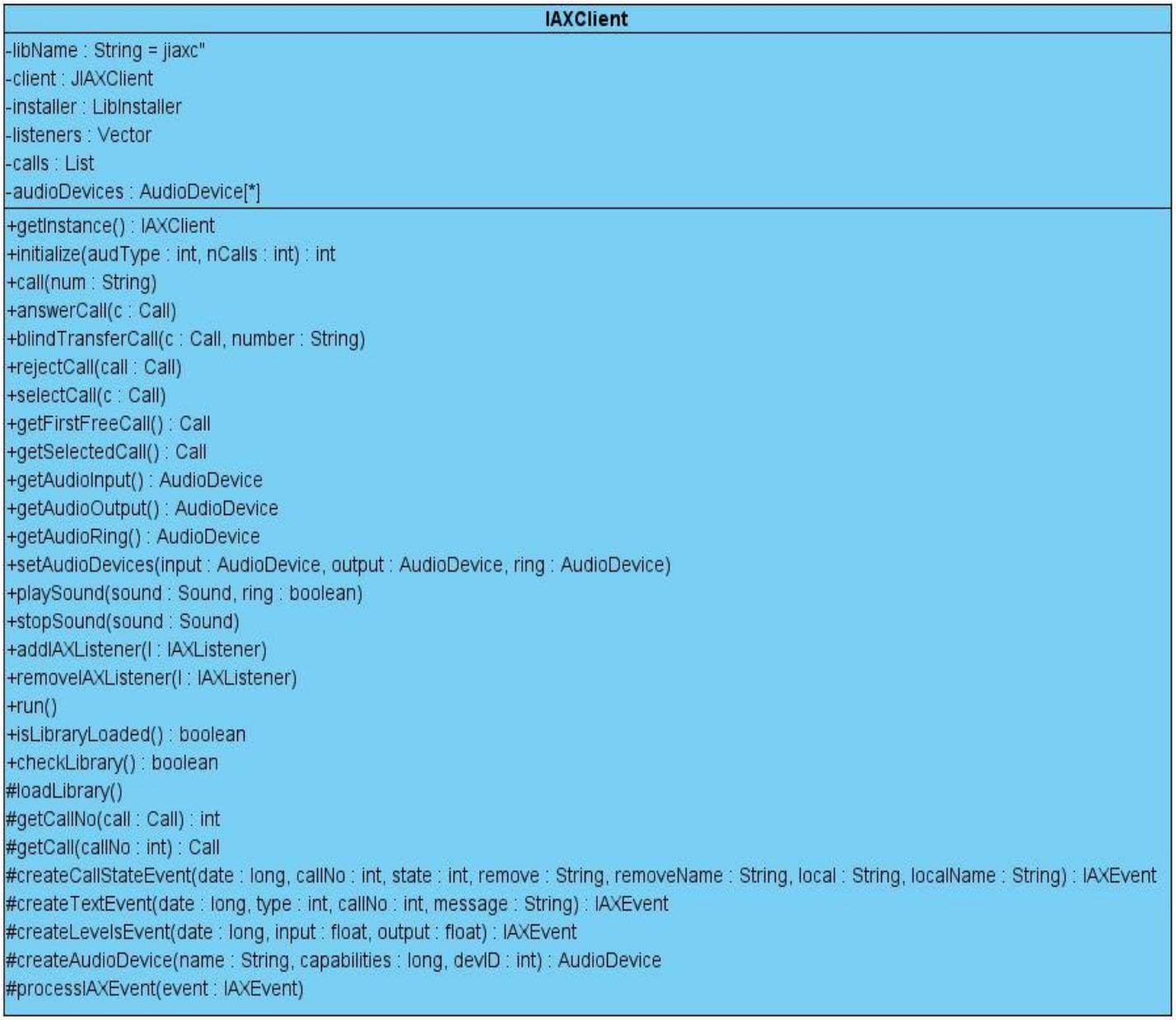


Ilustración 23 IAXClient

Anexo1.3: Paquete net.sourceforge.iaxclient.jni

Ilustración 24 AudioDevice

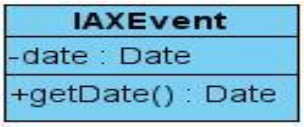


Ilustración 25 IAXEvent

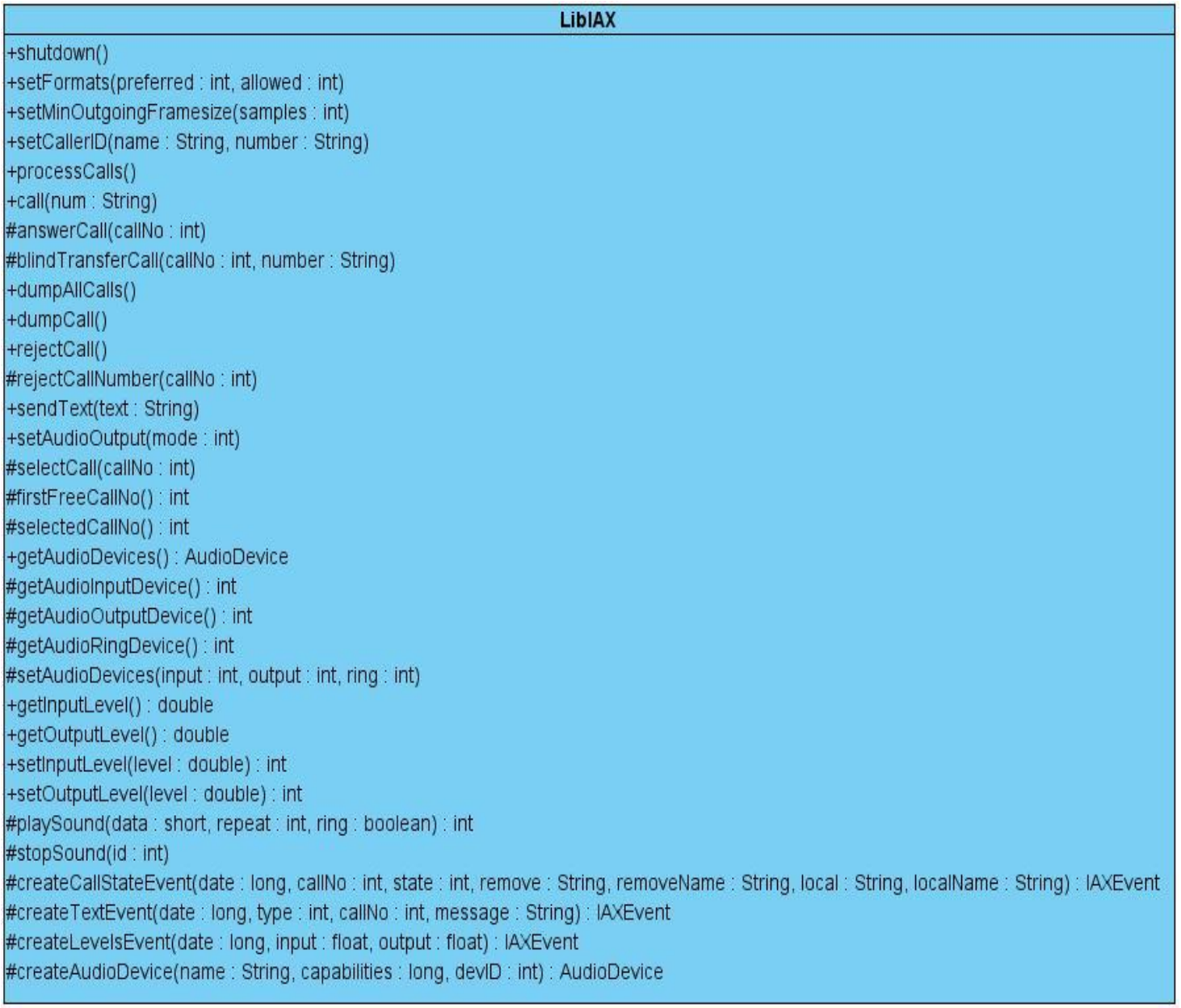


Ilustración 26 LibIAX

Glosario de Términos.

HD: Sistemas de alta definición

Cisco: compañía que trae incorporado en su catálogo de productos la telefonía Volp.

Asterisk: servidor que está diseñado para conectar cualquier hardware telefónico o cualquier tipo de software de telefonía de manera transparente y consistente. Es un programa de software libre (bajo licencia GPL) que proporciona funcionalidades de una central telefónica (PBX)

PBX: central telefónica conectada directamente a la red pública de telefonía por medio de líneas troncales para gestionar además de las llamadas internas, las entrantes y salientes con autonomía sobre cualquier otra central telefónica.

VoIp: telefonía a través de Internet, también conocida como telefonía IP o voz sobre IP. Básicamente es la transmisión de datos de voz sobre redes basadas en IP.

ATA: Adaptador Telefónico Analógico

Applet:: pequeños programas hechos en Java, que se transfieren con las páginas web y que el navegador ejecuta en el espacio de la página.

UDP: Protocolo Datagrama de Usuario. Es un protocolo no orientado a conexión de la capa de transporte del modelo TCP/IP. Este protocolo es muy simple ya que no proporciona detección de errores.

TCP: Protocolo de Control de Transmisión. Es uno de los principales protocolos de la capa de transporte del modelo TCP/IP. En el nivel de aplicación, posibilita la administración de datos que vienen del nivel más bajo del modelo, o van hacia él.

TLS: protocolo para establecer sesiones seguras.

Túnel de HTTP: posibilita la transferencia de datos ilimitada, permite utilizar los usos de internet a pesar de cortafuegos restrictivos.

API: Una interfaz de programación de aplicaciones o API (del inglés Application Programming Interface) es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a

objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Son usados generalmente en las bibliotecas.

SIP: es un protocolo de señalización, según el modelo OSI de capa aplicación, que controla la iniciación, modificación y terminación de llamadas o sesiones multimedia interactivas.

Full dúplex: transmisión de datos en dos direcciones simultáneamente.

SMS: en inglés es acrónimo de servicio de mensajes cortos ("Short Message Service"), sistema de mensajes de texto para teléfonos móviles

SSL: protocolo de capa de conexión segura. Es un protocolo criptográfico que proporciona comunicaciones seguras por una red..

JSP: es un acrónimo de Java Server Pages, Páginas de Servidor Java. Es, pues, una tecnología orientada a crear páginas web con programación en Java.

XHTML: lenguaje extensible de marcado de hipertexto. Es el lenguaje de marcado pensado para sustituir a HTML como estándar para las páginas web.

HTML Lenguaje de Marcado de Hipertexto. Es el lenguaje de marcado predominante para la elaboración de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes

XML: lenguaje de marcas extensible. Es un metalenguaje extensible de etiquetas. No es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. Es una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen mucho más grande y con unas posibilidades mucho mayores

JSF: es una tecnología y framework para aplicaciones Java basadas en web que simplifica el desarrollo de interfaces de usuario en aplicaciones Java EE

Java EE: es una plataforma de programación(parte de la Plataforma Java) para desarrollar y ejecutar software de aplicaciones en Lenguaje de programación Java

ServLets: La palabra servlet deriva de otra anterior, applet. Por contraposición, un servlet es un programa que se ejecuta en un servidor.

Datta Mapper: mapeo de datos. Es el proceso de creación de elementos de datos de asignaciones entre dos distintos modelos de datos

ADSI: Active Directory Service Interfaces. Es un conjunto de interfaces COM para acceder a las funciones de los servicios de directorio de los proveedores de red diferentes

IAX:_protocolo de comunicación entre Asterisk

H.323: (protocolo para transmisión de tráfico de voz en tiempo real): direcciona el control y señalización de una llamada y describe cómo una sesión entre 2 terminales H.323 es creada y mantenida.

Mp3: es una abreviación para "MPEG-1 Audio Layer 3", que es un formato de compresión digital de audio. Es una manera de almacenar sonido de forma que ocupe muy poco espacio en memoria.

IAX2:_La segunda versión del protocolo de comunicación entre Asterisk (Inter Asterisk Exchange) se conoce como IAX2.16 IAX2 es una alternativa al protocolo de señalización SIP