Universidad de las Ciencias Informáticas

FACULTAD 2

TRABAJO PARA OPTAR POR EL TÍTULO DE INGENIERÍA EN CIENCIAS INFORMÁTICAS.

Distribuidor Automático de Llamadas (ACD) para PLATEL

Autores:

Yusnel Cruz Castillo

Liorge Corria Sanchez

Tutor:

Ing. Eduardo Javier Roig Sastre

Ciudad de la Habana 2009

"Año del 50 Aniversario del Triunfo de la Revolución"

La inteligencia consiste no sólo en el conocimiento, sino también en la destreza de aplicar los conocimientos en la práctica.

Aristóteles

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año_____.

Autores:

_____ _____

Yusnel Cruz Castillo **Liorge Corria Sanchez**

Tutor:

_____

Ing. Eduardo Javier Roig Sastre

De Yusnel:

A mis padres Miriam y Rolando por estar siempre a mi lado cuando los necesite y por darme su apoyo en todo momento y ser un ejemplo digno de seguir.

A mi hermana Liset por ser tan comprensiva y brindarme todo su apoyo.

A toda mi familia por confiar en mí y brindarme todo el cariño del mundo.

A mi novia Ketty por acompañarme durante tres años de mi vida y darme su confianza, cariño y ser tan paciente en los momentos difíciles.

A mis amigos por estar conmigo en los buenos y malos momentos.

De Liorge:

A mi Madre, por ser la fuente de mi vida, cuyos recuerdos están cada día más presentes en mi mente y mi corazón. Te extraño Mamita

A mi Abuelita, Por ser como una madre, por creer siempre en mí, por sentirse orgullosa y por darme fuerzas para estudiar. Te adoro abuelita.

A mi Papá, Tía Barbarita y Tata por depositar tanta y tanta confianza en mí y saber guiarme por el camino correcto.

A mis Hermanitos: Cesar y Lisi, por lo mucho que me quieren.

A mis Tíos: Mari, Jorgito, Juan, Jorge Luis, Ruso, Belkis, Alex, Edunia, Made, Juan Carlos, Pili, Mary, Alexi, Chochi y Casique, por ser tan especiales conmigo.

A mis Primos: Yaqui, Leo, Tito, Leyanis, Leyaris, Sailet, Yuli, Osmani, Osneidi, Osmel, Maikel, Maidel, Y Mainel, por ser tan amables y cariñosos.

A: Tía Amelia, Abuelo Cesar, Abuelita Halla, Rosi, Yadanis, Mamita, Norma, Sengue, Yesi, Lety, Benao, Yendri y Coca. Por preocuparse por mí en todo momento.

En Ramírez A: Luibita, Compa, Richi, Carmen, Mambí, Pucha, Michel, Erso, Mercedes, Amiesly, Yamilka, Elena, Abel, Roidy, Titi y la mujer. Gracias por su apoyo incondicional en todo momento.

A Virama Toda y en especial a: Rosa, Chino, Yoan, Yarana, Olber, Palma, Dalia, Paquito, Tío Franki, Lalo, Yunier, Yamilka, Angelita, Nieve, Gollin, Eslita, Mari, Beto, Leydier, Robertico, Martica, Silvino, Iris, Fernando, Mimi, Vallejo, Alfredo, Beny, Yamilkita y el Primo Yulier. Por el afecto que sienten por mí.

A mis amigos de toda la vida y los que conocí en este maravilloso lugar.

A mi Tutor.

Gracias a todos por apoyarme y quererme.

De Yusnel:

Dedico este trabajo a mis padres que son mi mayor fuente de inspiración y razón de ser, a mi familia y amigos.

De Liorge:

Dedico este trabajo con mucho amor y cariño a mi linda Mamita.

A quien me consagró su vida y en tan solo unos minutos se fue cuando más falta me hacia, a quien de niño sufrió por mis fiebres y desde algún lugar me cuida todavía, a quien me puso un nombre sin saber si lo quería y me educó con honores. Gracias por haberme dado la vida, por ser hermosa conmigo, por curarme mis heridas.

Cuando despierto en las mañanas añoro tu tierno beso y una lágrima en mis ojos da repuesta a lo que siento, quiero que sepas que duermo pensando en ti, estudio pensando en ti, río pensando en ti, que cuando juego fútbol y anoto un gol te lo dedico a ti, que a cada lugar que voy, siento que estás junto a mi y el título de Ingeniero me lo gané para ti.

Te quiero y te extraño un mundo mamita.

Resumen

El presente trabajo tiene como tema central el desarrollo de una herramienta que permite la configuración y supervisión de un distribuidor automático de llamadas (ACD), el cual está basado en la PBX de software libre Asterisk, el mismo es una aplicación web y consta de tres partes fundamentales, las cuales son, configuración del sistema, monitorización en tiempo real y reportes históricos. Se realiza un estudio sobre las tendencias actuales que existen para desarrollar aplicaciones de este tipo, así como el análisis y selección de las tecnologías y herramientas necesarias. Se exponen los resultados obtenidos de la investigación realizada sobre las metodologías de desarrollo de software, y se justifica la selección de una de ellas. Finalmente se desarrollan las fases correspondientes a la metodología de desarrollo antes seleccionada.

Introducción

Con las nuevas tecnologías de la información y la comunicación las empresas se abren a una nueva forma de relacionarse con sus clientes, en donde la incorporación de servicios de Call Center ó centro de llamadas han pasado a ser una de las piezas claves en el conjunto de las gestiones de la calidad de los servicios y la satisfacción de los clientes, estos no sólo son vistos como un canal para reforzar la labor del vendedor, sino también, como una efectiva solución de negocios capaz de adaptarse a las cambiantes expectativas de sus clientes, asegurando así su lealtad.

En este escenario, los Call Center se proyectan como una alternativa para que las empresas puedan resolver efectivamente su proceso de toma de pedidos, al integrar a su gestión comercial, tecnología y recursos humanos capaces de llevar a cabo esta función bajo la administración y supervisión constante de un experto.

Actualmente la Universidad de las Ciencias Informáticas (UCI), en conjunto con la Empresa de Telecomunicaciones de Cuba.SA (ETECSA) y las Fuerzas Armadas Revolucionarias (FAR) han decidido emprender un proyecto que percibe el desarrollo de una Plataforma de Atención Telefónica (PLATEL).

El núcleo de la plataforma PLATEL es Asterisk, una aplicación de software libre bajo licencia *GPL* (acrónimo de General Public License), que proporciona funcionalidades de una central telefónica *PBX* (Private Branch Exchange por sus siglas en inglés). Como cualquier PBX, se encarga de establecer conexiones entre terminales de una misma empresa, o de hacer que se cursen llamadas al exterior. Asterisk incluye muchas características, anteriormente sólo disponibles en costosos sistemas propietarios PBX: como buzón de voz, conferencias, Respuesta de Voz Interactiva (IVR), distribución automática de llamadas (ACD), entre otras.

Actualmente la configuración del distribuidor automático de llamadas de la PBX Asterisk se hace en ficheros y de forma manual, haciendo muy engorroso y difícil este trabajo. Si a esto se le agrega que a la hora de obtener reportes históricos que ayuden a mejorar los niveles de servicio de PLATEL, así como el de monitorear las colas y agentes en tiempo real, se hace a base de comandos, se puede percatar de lo tedioso que resulta realizar todas estas tareas y es precisamente esta la situación problémica a la que se le pretende dar solución con la realización del presente trabajo de diploma.

Se puede definir entonces como **problema científico:** ¿Cómo facilitar el trabajo de configuración y supervisión del distribuidor automático de llamadas para PLATEL?

Este problema se enmarca en el **objeto de estudio:** procesos de configuración y supervisión del distribuidor automático de llamadas en una planta telefónica y en el **campo de acción:** procesos de configuración y supervisión del distribuidor automático de llamadas para PLATEL.

El **objetivo general** que se persigue con la realización de este trabajo es Desarrollar una herramienta que permita la configuración y supervisión del Distribuidor Automático de Llamadas para PLATEL.

Las **tareas de la investigación** que se proponen para dar cumplimiento al objetivo trazado son:

1. Realizar un estudio del estado del arte de software automatizados existentes.
2. Estudiar y definir las herramientas y lenguajes a utilizar para el desarrollo del producto.
3. Estudiar la Plataforma Telefónica Asterisk.
4. Analizar, diseñar e implementar el producto.

Capítulo 1: Fundamentación Teórica

1.1 Introducción

Este capítulo tiene como principal objetivo explicar de una manera más profunda qué es un Distribuidor Automático de Llamadas, así como realizar un estudio detallado del estado del arte para tener una mayor comprensión de este trabajo. Además se analizarán las posibles herramientas de desarrollo y metodologías a utilizar para la elaboración del software.

1.2 Estado del Arte

Un Distribuidor Automático de Llamadas o *ACD* (Automatic Call Distribution por sus siglas en inglés) es la tecnología de Call Center que permite:

- Distribuir las llamadas de entrada haciendo llegar cada llamada al mejor agente, en el mejor tiempo.
- Poner en cola las llamadas cuando todos los agentes estén ocupados, y avisar a quien llama del estatus de su llamada.
- Entregar reportes de llamadas recibidas, atendidas, abandonadas, etc., así como tiempos promedio en llamada, tiempos promedio en espera, llamadas en cola, niveles de servicio y mucho más.

Un ACD ayuda a los agentes a concentrarse en atender cada llamada y a mejorar sus niveles de servicio disminuyendo el tiempo promedio de llamada, aumentando su productividad a las horas pico y canalizando las llamadas al mejor agente disponible para atender ese tipo de llamada.

Además permite obtener reportes que ayudan a mejorar el desempeño del Centro de Llamadas.

Un ACD permite distribuir de la mejor manera la carga de trabajo entre los agentes, y ayuda a manejar la espera de una manera justa para los que llaman.

Además proporciona estadísticas que permiten ir mejorando los niveles de servicio y satisfacción por parte de los clientes, así como a diferenciar al personal que mejor trabaja del que menos lo hace.

1.2.1 Sistemas Automatizados existentes

Calixta ACD

"Calixta ACD es un poderoso sistema de distribución de llamadas telefónicas. Es uno de los mejores sistemas de distribución de llamadas del mercado. Con distribución por habilidad de los agentes y un manejo muy poderoso de políticas de transferencia, además de poseer una interfaz amigable. Es utilizado por los Centro de Llamadas más importantes de Latinoamérica para atender millones de llamadas al mes. Se comunica con todas las bases de datos comerciales del mercado, que cuenten con interface abierta de comunicación (ejemplos: Oracle, Informix, Access, mySQL, SQL Server, Postgre, etc.). Permite crecer desde 5 hasta 1000 agentes, realizar llamadas fuera de la base de datos y mucho más. Además cuenta con módulos opcionales de grabación de llamadas y pantallas, envío de mensajes pregrabados y personalizados, envío y recepción de SMS y manejo combinado de llamadas de salida, marcación predictiva, reconocimiento de voz y texto a voz, entre otras opciones". [1]

Aspect Spectrum ACD

"Aspect Spectrum ACD es un distribuidor automático de llamadas con capacidades altamente eficaces de enrutamiento, permitiéndoles a los llamantes obtener los recursos iniciales adecuados. Posee opciones avanzadas que ofrecen un servicio personalizado, esto asegura que los agentes y clientes reciban lo mejor de cada contacto. El manejo de contactos multicanales le permite brindar un servicio constante a través de todos los canales de comunicación, y el enrutamiento de múltiples sitios, ofrece al cliente el acceso a los recursos que mejor satisfagan sus necesidades particulares.

Permite mantener una visión histórica y completa en tiempo real del rendimiento del Centro de Llamadas. Con las alarmas en tiempo real se pueden hacer cambios a las estrategias basados en datos fundamentales tales como: el nivel de servicio y el volumen de llamadas abandonadas". [2]

Aspect CallCenter ACD

"Aspect CallCenter ACD es una solución de procesamiento de llamadas sofisticada que aumenta la satisfacción del cliente y la eficiencia del Centro de Llamadas con un enrutamiento mejorado. Su interfaz de desarrollador, sencilla y fácil de usar, simplifica la satisfacción de las cambiantes necesidades del negocio. Los desarrolladores

pueden crear sofisticados supuestos de enrutamiento simplemente arrastrando iconos a un área de trabajo y uniéndolos para indicar el modo en que se deben enrutar las llamadas.

Aumenta las satisfacciones del cliente con opciones de enrutamiento eficaces y flexibles que permiten el acceso de forma rápida y precisa a los agentes más preparados para resolver los problemas concretos de los clientes.

Es una solución de enrutamiento de grandes volúmenes de llamadas, compatible con las operaciones de negocios decisivas de las principales industrias de todo el mundo. Ofrece una amplia gama de capacidades de enrutamiento de llamadas que incrementan los ingresos, reducen los gastos y aumentan la fidelidad del cliente. Este sistema de gestión de llamadas, procesa hasta 300.000 llamadas por hora, enruta llamadas según las capacidades de los agentes, pone en cola simultáneamente llamadas en múltiples ubicaciones, proporciona herramientas para la creación de reportes en tiempo real e historiales, y mucho más.

Los sistemas de distribución automática de llamadas antes presentados poseen sin lugar a dudas características que los ubican entre los primeros a nivel mundial, pero presentan las limitantes de ser sistemas propietarios y no cumplir con las especificidades de la plataforma PLATEL. Debido a esto surge la necesidad de realizar un estudio detallado de las tendencias y tecnologías actuales para desarrollar una herramienta que se ajuste a las necesidades de PLATEL, basándose en software libre". [3]

1.3 Servidores Web

1.3.1 Apache

Apache es un servidor de páginas web de código abierto, multiplataforma y modular, el cual se desarrolla dentro del proyecto HTTP Server de la Apache Software Foundation. Presenta entre otras características mensajes de error altamente configurables, bases de datos de autenticación y negociado de contenido, pero fue criticado por la falta de una interfaz gráfica que ayude en su configuración. "Se ha adaptado siempre a una gran variedad de entornos a través de su diseño modular. Este diseño permite a los administradores de sitios web elegir qué características van a ser incluidas en el servidor, seleccionando que módulos se van a cargar, ya sea al compilar o al ejecutar el servidor". [4]

“La mayoría de las vulnerabilidades de la seguridad descubiertas y resueltas tan sólo pueden ser aprovechadas por usuarios locales y no remotamente. Sin embargo, algunas se pueden accionar remotamente en ciertas situaciones, o explotar por los usuarios locales malévolos en las disposiciones de recibimiento compartidas que utilizan PHP como módulo de Apache”. [5]

Características principales

- Trabaja sobre múltiples plataformas (Unix, Linux, MacOSX, Win32, OS2, etc.).
- Incluye módulos que se cargan de forma dinámica.
- Soporta CGI, Perl, PHP.
- Soporte para Bases de datos.
- Soporte SSL para transacciones seguras.
- Incluye soporte para host virtuales.
- Soporta HTTP 1.1.
- Código Abierto.
- Rápido.
- Eficiente.

1.3.2 Lighttpd

Lighttpd es un servidor web diseñado para ser rápido, seguro, flexible, y fiel a los estándares. Está optimizado para entornos donde la velocidad es muy importante, y por eso consume menos CPU y memoria RAM que otros servidores. Por todo lo que ofrece, Lighttpd es apropiado para cualquier servidor que tenga problemas de carga. Lighttpd es software libre y se distribuye bajo la licencia BSD. Funciona en GNU/Linux y UNIX de forma oficial. Para Microsoft Windows actualmente hay una distribución conocida como Lighttpd For Windows mantenida por Kevin Worthington.

Características

- Virtual hosting (alojar varios dominios en la misma IP).
- Soporte para PHP, Ruby, y otros.
- Compresión (gzip, bzip2).
- Consumo de memoria constante.
- Redirecciones HTTP, y reescrituras de URL.
- Puede enviar partes de un fichero (rangos).
- Permite módulos externos.

1.3.3 Thttpd

Thttpd es un servidor web de código libre disponible para la mayoría de las variantes de Unix. Se caracteriza por ser simple, pequeño, portátil, rápido, y seguro, ya que utiliza los requerimientos mínimos de un servidor HTTP. Esto lo hace ideal para servir grandes volúmenes de información estática. Se caracteriza por ser:

- Simple, ya que sólo maneja el mínimo necesario para poner en práctica el protocolo HTTP, algunas veces un poco más que el mínimo.
- Pequeño, porque tiene un pequeño tamaño de período de explotación, ya que no se divide en dos partes y es muy cuidadoso sobre la asignación de memoria.
- Portátil, pues se compila limpiamente sobre la mayoría de sistemas operativos, expresamente incluyendo FreeBSD, SunOS 4, Solaris 2, BSD/OS, Linux, OSF.
- Rápido, porque en el empleo típico es sobre todo más rápido que los mejores servidores “destacados” (APACHE), y bajo la carga extrema es mucho más rápido.
- Seguro, porque se extiende a grandes longitudes para proteger el servidor WEB contra ataques de otros sitios.

Ventajas:

- El administrador puede restringir la transferencia de archivos de imagen JPEG.
- Los promedios de carga se caen debido a la reducción de la transferencia gráfica.

Desventajas:

- "No posee las mismas aplicaciones que se pueden obtener de un software estándar como el Servidor Web Apache". [6]

1.3.5 Servidor Web seleccionado

Luego de realizarse un estudio profundo de los servidores web, se decidió utilizar Apache por las siguientes razones:

Ventajas

- Este servidor junto con el módulo mod_rewrite puede convertirse en una herramienta muy útil para crear páginas con enlaces amigables para los buscadores.
- Es un software libre.
- Multiplataforma.
- Modular.
- Extensible.
- Presenta mensajes de error altamente configurables.

1.4 Gestores de Base de Datos

"Una base de datos es un conjunto de datos interrelacionados entre sí, almacenados con carácter más o menos permanente en la computadora. O sea, que una base de datos puede considerarse una colección de datos variables en el tiempo.

El software que permite la utilización y/o la actualización de los datos almacenados en una (o varias) base(s) de datos por uno o varios usuarios desde diferentes puntos de vista y a la vez, se denomina sistema de gestión de bases de datos (SGBD)". [7]

1.4.1 MySQL

"MySQL es un sistema de administración de bases de datos relacional. Se trata de un programa capaz de almacenar una enorme cantidad de datos de gran variedad y de distribuirlos para cubrir las necesidades de cualquier tipo de organización, desde pequeños establecimientos comerciales a grandes empresas y organismos administrativos". [8]

Ventajas:

- Consume muy pocos recursos de CPU y memoria.
- Tiene buen rendimiento.
- Posee un tamaño del registro sin límite.
- Control de acceso usuarios-tablas-permisos.

Desventajas:

- Carece de soporte para transacciones, rollback y subconsultas.
- El hecho de que no maneje la integridad referencial, hace de este gestor una solución pobre para muchos campos de aplicación, sobre todo para aquellos programadores que provienen de otros gestores que sí que poseen esta característica.
- No es viable su uso en grandes bases de datos, a las que se acceda continuamente, ya que no implementa una buena escalabilidad.
- Actualmente SUN Microsystem tiene en su poder a MySQL, prohibiendo la descarga de futuras versiones desde cuba.

1.4.1 Oracle

Oracle es un sistema de administración de base de datos disponible para un ancho rango de plataformas. Es fácil de desarrollar y administrar, puede manipular todo tipo de administración de datos y ofrece una excepcional disponibilidad, escalabilidad, fiabilidad y seguridad. La seguridad de las aplicaciones y bases de datos de Oracle permiten la administración completa de los usuarios, el monitoreo de potenciales

ataques y la asignación de privilegios de acceso a través de toda la empresa. Las últimas versiones han sido certificadas para poder trabajar sobre plataforma Linux. [9]

Ventajas:

- Permite implementar diseños "activos", con triggers y procedimientos almacenados, con una integridad referencial declarativa bastante potente.
- Permite el uso de particiones para la mejora de la eficiencia, de replicación e incluso ciertas versiones admiten la administración de bases de datos distribuidas.
- El software del servidor puede ejecutarse en multitud de sistemas operativos.

Desventajas:

- El precio de las patentes y licencias del equipamiento que requiere Oracle, con respecto a otros sistemas gestores de bases de datos constituye uno de sus principales problemas.
- Otro problema es la necesidad de ajustes. Un error frecuente consiste en pensar que basta instalar el Oracle en un servidor y enchufar directamente las aplicaciones clientes. Un Oracle mal configurado puede ser desesperantemente lento.

1.4.3 PostgreSQL

PostgreSQL es un servidor de base de datos objeto-relacional libre. Como muchos otros proyectos open source, el desarrollo de PostgreSQL no es manejado por una sola compañía sino que es dirigido por una comunidad de desarrolladores y organizaciones comerciales las cuales trabajan en su desarrollo, dicha comunidad es denominada el PGDG (PostgreSQL Global Development Group). Fue el pionero en muchos de los conceptos existentes en el sistema objeto-relacional actual, incluido más tarde en otros sistemas de gestión comerciales.

“PostgreSQL es un sistema objeto-relacional, ya que incluye características de la orientación a objetos, como puede ser la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional. A pesar de esto,

PostgreSQL no es un sistema de gestión de bases de datos puramente orientado a objetos". [10]

Ventajas:

- Soporte de protocolo de comunicación encriptado por SSL.
- Extensiones para alta disponibilidad, nuevos tipos de índices, datos espaciales, minería de datos, etc.
- Incorpora una estructura de datos array.
- Incorpora funciones de diversa índole: manejo de fechas, geométricas, orientadas a operaciones con redes, etc.
- Permite la declaración de funciones propias, así como la definición de disparadores.
- Soporta el uso de índices, reglas y vistas.
- Incluye herencia entre tablas (aunque no entre objetos, ya que no existen), por lo que a este gestor de bases de datos se le incluye entre los gestores objeto-relacionales.
- Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos.

Desventajas:

- Tiene un límite de 8K por fila, aunque se puede aumentar a 32K, con una disminución considerable del rendimiento.
- Es lento.

1.4.4 Servidor de Base de datos seleccionado

Para el desarrollo del sistema se realizó un detallado estudio de los Gestores de Base de Datos, el cual permitió que se pudiera realizar una elección certera. En dicho estudio se llegó a la conclusión de que el Gestor que se utilizará es PostgreSQL, por las siguientes razones:

- Puede ser utilizado, modificado y distribuido por cualquiera gratuitamente, para cualquier propósito ya sea con fines privados, comerciales o académicos.
- Corre en casi todos los principales sistemas operativos: Linux, Unix, Windows, etc.
- Soporte nativo para los lenguajes más populares: PHP, C, C++, Perl, Python, etc.

Además de las facilidades antes mencionadas se decidió escoger este gestor porque es altamente escalable tanto en la cantidad de datos que puede manipular como en la cantidad de usuarios concurrentes que puede atender, lo cual se ajusta a las características del producto a desarrollar.

1.5 Metodologías de Desarrollo de Software

"Para desarrollar un proyecto de software es necesario establecer un enfoque disciplinado y sistemático. Las metodologías de desarrollo influyen directamente en el proceso de construcción y se elaboran a partir del marco definido por uno o más ciclos de vida". [11]

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos software.

Después de ver y tener en cuenta de que si es necesario utilizar una metodología en especifico para el desarrollo de cualquier producto, se impone la tarea de decidir cuál de esas metodologías existentes será la más adecuada o la mejor, cuál se corresponde más con el tipo de producto que se quiere obtener, cuál está más acorde con las características del equipo de trabajo, del entorno, del tiempo máximo para la entrega, en fin, deben considerarse aspectos muy importantes antes de adoptar una determinada metodología de desarrollo.

Hoy en día existen muchas tendencias de metodologías que brindan diferentes marcos que los desarrolladores pueden emplear a la hora de realizar su trabajo, algunas de ellas son:

- RUP (Proceso Unificado del Rational)
- XP (Programación Extrema)

1.5.1 Proceso Unificado de Desarrollo de Software (RUP)

RUP es el resultado de varios años de desarrollo y uso práctico en el que se han unificado técnicas de desarrollo, a través del UML, y trabajo de muchas metodologías utilizadas por los clientes. La versión que se ha estandarizado nació en 1998 y se conoció en sus inicios como Proceso Unificado de Rational 5.0; de ahí las siglas con las que se identifica a este proceso de desarrollo.

"En RUP se han agrupado las actividades en grupos lógicos definiéndose 9 flujos de trabajo principales. Los 6 primeros son conocidos como flujos de ingeniería y los tres últimos como de apoyo". [11]

RUP se caracteriza por ser:

- Centrado en la arquitectura: La arquitectura muestra una visión completa del sistema y se describen los elementos más importantes para la construcción del software.
- Dirigido por casos de uso: Los casos de uso muestran los que los usuarios necesitan y desean, esto se obtiene a partir de la modelación del negocio quedando plasmado en la especificación de requisitos.
- Iterativo e Incremental: RUP propone que cada fase se desarrolle en iteraciones. Es muy útil dividir el proyecto en mini proyectos ya que el desarrollo de cada uno de ellos es una iteración que contribuye al incremento del proyecto general.
- RUP propone 9 flujos de trabajo 6 de ingeniería y 3 de apoyo, los flujos de trabajo ingenieriles son: Modelamiento del Negocio, Requerimientos, Análisis y Diseño, Implementación, Prueba, Despliegue.

Ventajas:

- Evaluación en cada fase que permite cambios de objetivos.
- Funciona bien en proyectos de innovación.
- Sigue los pasos intuitivos necesarios a la hora de desarrollar el software.
- Seguimiento detallado en cada una de las fases.

Desventajas:

- La evaluación de riesgos es compleja.
- Excesiva flexibilidad para algunos proyectos.
- El cliente deberá ser capaz de describir y entender qué es lo que desea a un gran nivel de detalle, para poder acordar un alcance del proyecto con él.

1.5.2 Programación Extrema (XP)

Actualmente La Programación Extrema se utiliza en proyectos riesgosos cuando se tienen requerimientos dinámicos. Las reglas y prácticas de XP son muy simples, cada una soporta a la otra creando una integración única que forma la metodología de desarrollo de software. Esta metodología pertenece al grupo de metodologías ágiles. Es una metodología basada en la simplicidad, comunicación, retroalimentación y el coraje. En la Programación Extrema cada integrante es una parte del “TODO”. El equipo está compuesto por un representante del negocio llamado cliente que se sienta y trabaja junto a ellos todos los días, esto permite que cualquier duda a aclarar puede ser resuelta al instante por estar el cliente siempre disponible para el equipo de desarrollo.

Algunas reglas y prácticas de la Programación Extrema

Se escriben los casos de historia (similares a los casos de uso de RUP). El proyecto se divide por iteraciones, se realiza una reunión diaria con el objetivo de analizar y trazarse tareas. El diseño en XP se caracteriza por ser simple y porque ninguna funcionalidad se añade tempranamente. El código del proyecto debe ser escrito según estándares aprobados. La primera unidad que se implementa es la unidad de prueba dándosele a este elemento una importancia significativa dentro de la Programación Extrema. En XP la realización de test y la experiencia de los clientes adquieren una mayor importancia, haciendo que se trabaje en ciclos de menor tiempo.

Ventajas:

- Cuatro ojos ven más que dos. Al trabajar de dos en dos, el código será de mayor calidad desde el mismo momento de crearlo y tendrá menos fallos.

- Los programadores novatos aprenderán de los expertos al emparejarse con ellos.
- Si una pareja realiza un pedazo de código susceptible de ser reutilizado en el proyecto, hay dos programadores que lo saben y que lo reutilizarán cuando puedan (ya que saben cómo funciona), enseñándolo a sus nuevos compañeros. De esta manera el conocimiento del código ya hecho se propaga de forma natural entre todos los programadores del equipo.
- El estilo de programación tiende a unificarse.
- Apropiado para entornos volátiles.
- Estar preparados para el cambio, significa reducir su coste.
- Planificación más transparente para los clientes, conocen las fechas de entrega de funcionalidades vitales para su negocio.
- Permite definir en cada iteración cuales son los objetivos de la siguiente.
- Permite obtener retroalimentación de los usuarios.
- La presión está a lo largo de todo el proyecto y no en una entrega final.

Desventajas:

- Delimitar el alcance del proyecto con el cliente.

1.5.3 Scrum

Tanto Scrum como la Programación Extrema (XP) requieren que los equipos completen algún tipo de producto potencialmente liberable al final de cada iteración. Estas iteraciones están diseñadas para ser cortas y de duración fija.

“Estos equipos aceptan que puede que se equivoquen por el camino, pero también son conscientes de que la mejor manera de encontrar dichos errores es dejar de pensar en el software a un nivel teórico de análisis y diseño y sumergirse en él, ensuciarse las manos y comenzar a construir el producto”. [12]

Valores de Scrum:

- Equipos auto-dirigidos y auto-organizados. No hay manager que decida, la excepción, es el Scrum Máster que debe ser 50% programador y que resuelve problemas, pero no manda. Los observadores externos; pueden observar, pero no interferir ni opinar.

- Una vez elegida una tarea, no se agrega trabajo extra. En caso que se agregue algo, se recomienda quitar alguna otra cosa.

- Encuentros diarios que impiden caer en el dilema: ¿Cómo es que un proyecto puede atrasarse un año?: Un día a la vez.

- Iteraciones de treinta días; se admite que sean más frecuentes.

- Demostración a participantes externos al fin de cada iteración.

- "Al principio de cada iteración, existe un planeamiento adaptable guiado por el cliente". [12]

1.5.4 Metodología seleccionada

Debido a que se dispone de un corto período tiempo para el desarrollo del software, poca disponibilidad del personal y existe una buena comunicación entre los clientes y los programadores del sistema, donde la prioridad es satisfacer al mismo mediante tempranas y continuas entregas de software en iteraciones que no excedan los 30 días, se decidió utilizar la metodología ágil Programación Extrema (*XP*) y como envoltorio para la gestión Scrum.

Beneficios de XP y Scrum:

- La agilidad de gestión y mecanismos de control de Scrum son aplicables para cualquier tipo de proyecto, incluyendo las iniciativas empresariales que consisten en múltiples, simultáneos de desarrollo de software, desarrollo empresarial, re-ingeniería, comercialización, apoyo y ejecución de proyectos. La realización de los proyectos con los beneficios de auto-organización; equipos de iteración (o Sprint) dirigida a objetivo, en lugar de dirigirse historia.

- En la Programación Extrema, cuando los proyectos son envueltos por

Scrum, pasan a ser escalables y se pueden ejecutar simultáneamente.

- Con XP y Scrum los proyectos pueden beneficiarse de los negocios de valor ADM (métrica para medir el retorno de la inversión y la gestión de la iniciativa). [12]

1.6 Entorno de desarrollo (IDE)

El entorno de desarrollo integrado (IDE por sus siglas en inglés) de Eclipse emplea módulos (en inglés plug-in) para proporcionar toda su funcionalidad, a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no. Esta característica le permite a Eclipse extenderse usando lenguajes de programación como Ruby, PHP y Python entre otros.

“Easy Eclipse empaqueta el entorno de desarrollo Eclipse junto con una cuidadosa selección de módulos Open Source para obtener un IDE final excepcionalmente bueno para el desarrollo de aplicaciones en PHP, Python, Ruby o Java, con todos los plugins ya instalados y configurados se puede comenzar el trabajo de forma sencilla y rápida. Easy Eclipse dispone de varias distribuciones para desarrollo entre ellas una optimizada para el desarrollo en Python”. [13]

Las principales fortalezas de esta herramienta son:

- Es opensource y multiplataforma.
- Posee integración con Sistemas Controladores de Versiones (SVN por sus siglas en inglés).
- Posee un completamiento de código muy bueno el cual contribuye al desarrollo ágil de la aplicación.
- Posee una extensa documentación y soporte en internet.
- El consumo es aceptable lo cual no interfiere con el desarrollo.

1.7 Tecnologías del lado del cliente

Son las que están insertadas en la página HTML del cliente, siendo interpretadas y ejecutadas por el navegador. Por una parte, se encargan de gestionar la comunicación con el servidor, de solicitar un servicio concreto y de recibir los datos enviados por el

mismo. Ofrecen los comandos necesarios para utilizar las presentaciones que brinda el servidor. Su correcta funcionalidad depende del soporte de la versión del navegador a ser utilizado por el usuario.

AJAX

"AJAX, acrónimo de JavaScript asíncrono y XML, no es una tecnología en sí misma; es una técnica de desarrollo web para crear aplicaciones interactivas. En realidad, se trata de la unión de varias tecnologías independientes". [14]

Las tecnologías que forman AJAX son:

- XHTML o **HTML**, para crear una presentación basada en estándares.
- DOM, para la interacción y manipulación dinámica de la presentación.
- XML, XSLT y JSON, para el intercambio y la manipulación de información.
- XMLHttpRequest, para el intercambio asíncrono de información.
- **JavaScript**, para unir todas las demás tecnologías.

"Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De está forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad, usabilidad en las aplicaciones y mejor respuesta a las acciones del usuario". [14]

HTML

HTML, Hyper Text Markup Language (Lenguaje de Marca de Hipertextos), es un lenguaje de programación muy sencillo que se utiliza para crear los textos y las páginas web. Es justamente un lenguaje que se basa en las marcas para crear los hipertextos. Está compuesto por etiquetas que definen la estructura y el formato del documento que verá el usuario a través de la web. Esas etiquetas son leídas por el navegador o visualizador, es decir el programa que se utiliza para navegar, y que es el que ejecuta las funciones creadas en HTML permitiendo que puedan ser visibles en nuestra máquina. Una de las características es que, además del texto, permite que se creen enlaces entre distintas partes del mismo documento o entre distintas fuentes de información a través de hiperenlaces o hipervínculos, e incluso insertar otros

elementos como imágenes y sonidos. La principal ventaja que tiene HTML es la gran variedad de navegadores y exploradores que lo soportan. Debido a esto, se ha convertido en el formato más usado para la transmisión de documentos electrónicos a través de Internet. Actualmente existe un gran número de navegadores por lo que no todos pueden interpretar el código HTML de la misma manera. Por esa razón es que quienes crean las páginas, chequean que pueda ser leída al menos por los navegadores más populares.

CSS

CSS es un lenguaje de hojas de estilos creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML. CSS es la mejor forma de separar los contenidos y su presentación; y es imprescindible para crear páginas web complejas.

“Dicho lenguaje se utiliza para definir el aspecto de todos los contenidos, es decir, el color, tamaño y tipo de letra de los párrafos de texto, la separación entre titulares y párrafos, además de la tabulación con la que se muestran los elementos de una lista”. [14]

Ventajas:

- Obliga a crear documentos HTML/XHTML bien definidos y con significado completo.
- Mejora la accesibilidad del documento.
- Reduce la complejidad de su mantenimiento.
- Permite visualizar el mismo documento en infinidad de dispositivos diferentes.

Desventajas:

Presenta incompatibilidad entre diferentes navegadores, por lo que si algo se ve excelente en un navegador es posible que en otro no se vea igual, o que varíe la visualización incluso en las diferentes versiones de un mismo navegador.

JavaScript

"Es un lenguaje de programación interpretado, multiplataforma y parcialmente orientado a objetos, desarrollado para incrementar las funcionalidades del lenguaje HTML. Permite crear efectos especiales, interactuar con el visitante y brinda funciones básicas que son soportadas por la mayoría de los navegadores que se utilizan diariamente". [15]

Ventajas:

- Tiempo de transferencia : El mantener la ejecución de lógica en el servidor de páginas implica que deben llevarse a cabo varias peticiones, esto es, si se necesita realizar un cálculo o se le están solicitando datos a una persona, esta información debe ser transferida hasta el servidor de páginas para ser ejecutada o validada, en cambio, si se utiliza un "Scripting Language" es posible realizar esta lógica instantáneamente en el cliente, sin incurrir en una transferencia de información adicional.
- Permite elaborar aplicaciones Web que simulen características de aplicaciones de escritorio.
- Es un código "interpretado" por el cliente.
- Es un código integrado a HTML.
- Reutilización de código de programación.
- El lenguaje de scripting es seguro y fiable.

Desventajas:

Uno de los inconvenientes de este lenguaje es que tiene que estar activado en los navegadores para que su uso sea posible. Por otro lado no le proporciona al programador control total de la página web. En ocasiones los desarrolladores deben realizar diferentes implementaciones de código JavaScript para sus aplicaciones, debido a que no todos los navegadores interpretan de la misma manera el código.

1.8 Lenguajes de Programación

La AGI (del inglés: Asterisk Gateway Interface), proporciona una interfaz estándar externa por la que los programas pueden controlar el plan de discado de Asterisk. Por lo general, los scripts de la AGI son utilizados para desarrollar lógica avanzada, comunicarse con bases de datos relacionales (PostgreSQL, MySQL entre otras), y acceder a otros recursos externos. Pasando el control del plan de marcación a un script externo de la AGI de Asterisk se pueden realizar fácilmente tareas que de otra manera serían difíciles o imposibles.

La PBX Asterisk brinda una interfaz estándar para los scripts de la AGI, los cuales pueden ser escritos en casi cualquier lenguaje de programación moderno. Pero los lenguajes más comúnmente utilizados y que se destacan en la programación AGI son: Perl, Java, PHP, Python.

1.8.1 Java

Es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.

Con la programación en Java, se pueden realizar distintos aplicativos, como son applets, que son aplicaciones especiales, que se ejecutan dentro de un navegador al ser cargada una pagina HTML en un servidor WEB. Por lo general los applets son programas pequeños y de propósitos específicos.

La independencia de plataforma es una de las razones por las que Java es utilizado para desarrollar aplicaciones que se publiquen en internet, ya que muchas personas deben tener acceso con ordenadores distintos. Pero no se queda ahí, Java está desarrollándose incluso para distintos tipos de dispositivos además del ordenador, como móviles y agendas.

1.8.2 Python

“Es un lenguaje de programación creado en el año 1990 por Guido van Rossum, es el sucesor del lenguaje de programación ABC. Los usuarios lo consideran como un lenguaje más limpio para programar; tiene una sintaxis muy visual, gracias a una

notación identada (con márgenes) de obligado cumplimiento". [16]

Es un lenguaje fácil de aprender y potente, multiplataforma, de propósito general. La elegante sintaxis de Python, su gestión de tipos dinámica y su naturaleza interpretada hacen de él, el lenguaje ideal para guiones (scripts) y desarrollo rápido de aplicaciones en muchas áreas y en la mayoría de las plataformas; desde aplicaciones Windows a servidores de red o incluso páginas web.

"La implementación de Python es bajo la licencia de código abierto por lo que puede utilizarse libremente para desarrollar aplicaciones de uso comercial. La licencia de Python es administrada por la Fundación de Software de Python". [16]

En los últimos años el lenguaje se ha hecho muy popular, gracias a varias razones como:

- La cantidad de librerías que contiene, tipos de datos y funciones incorporadas en el propio lenguaje, que ayudan a realizar muchas tareas habituales sin necesidad de tener que programarlas desde cero.
- La sencillez y velocidad con la que se crean los programas. Un programa en Python puede tener de 3 a 5 líneas de código menos que su equivalente en Java o C.
- La cantidad de plataformas en las que se puede desarrollar, como Unix, Windows, OS/2, Mac, Amiga y otros.

Ventajas:

- Desarrollo más rápido: Se puede escribir un programa, salvarlo y ejecutarlo. En un lenguaje compilado se tiene que pasar por los pasos de compilar y enlazar el software, lo cual puede ser un proceso lento.
- Multiplataforma: El mismo código funciona en cualquier sistema operativo, la única condición es que disponga del intérprete del lenguaje.

Desventajas:

- Lentitud: Los programas interpretados son más lentos que los compilados. Sin embargo los programas interpretados suelen ser cortos (como los scripts), en los que la diferencia es inapreciable.

1.8.3 Perl

Es un lenguaje muy flexible con una gran cantidad de módulos ya implementados, toma características de C (Lenguaje de programación) y del lenguaje interpretado Shell. Una diferencia fundamental de Perl con respecto a los otros lenguajes es que no limita el tamaño de los datos con los que trabaja, el límite lo pone la memoria que en ese momento se encuentre disponible.

La principal causa de la poca apariencia de Perl es por la afición de sus desarrolladores a la codificación en "una línea", empaquetando numerosas funcionalidades en una sola línea de código. Bien escritos los scripts en Perl se asemejan bastante a PHP.

1.8.4 PHP

Es un lenguaje de programación utilizado para la creación de sitios web. PHP es un acrónimo recursivo que significa “PHP Hypertext Pre-Processor”, (inicialmente se llamó Personal Home Page). Surgió en 1995, desarrollado por PHP Group.

PHP es un lenguaje de script interpretado en el lado del servidor utilizado para la generación de páginas web dinámicas, embebidas en páginas HTML y ejecutadas en el servidor. PHP no necesita ser compilado para ejecutarse. Para su funcionamiento necesita tener instalado Apache o *IIS* (Internet Information Server) con las librerías de PHP. La mayor parte de su sintaxis ha sido tomada de C, Java y Perl con algunas características específicas. Los archivos cuentan con la extensión (php).

Ventajas:

- Soporta la orientación a objeto, lo que brinda potencia al lenguaje.
- Es un lenguaje multiplataforma.
- Capacidad de conexión con la mayoría de los manejadores de base de datos: MySQL, PostgreSQL, Oracle, MS SQL Server, entre otras.

- Capacidad de expandir su potencial utilizando módulos.
- Posee documentación en su página oficial, la cual incluye descripción y ejemplos de cada una de sus funciones.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Incluye gran cantidad de funciones.
- No requiere definición de tipos de variables ni manejo detallado a bajo nivel.

Desventajas:

- Todo el trabajo lo realiza el servidor y no delega al cliente. Por tanto puede ser más ineficiente a medida que las solicitudes aumenten en número.
- La legibilidad del código puede verse afectada al mezclar sentencias HTML y PHP.
- La programación orientada a objetos es aún muy deficiente para aplicaciones grandes.
- Dificulta la modularización.
- Dificulta la organización por capas de la aplicación.
- Se necesita instalar un servidor web.

1.8.5 Lenguaje seleccionado

De los lenguajes tratados anteriormente Python es el más indicado para el desarrollo de la aplicación Web, cuenta con gran cantidad de funciones y librerías, es un lenguaje sencillo de propósito general y rápido de programar, versátil, ágil y multiplataforma. Su código bien organizado proporciona un mayor entendimiento para un posterior desarrollo y mantenimiento de la aplicación por cualquier programador.

1.9 Framework

“El concepto framework viene aparejado con el desarrollo de software. Un framework es una estructura de soporte robusta y bien definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, un framework puede incluir

soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto". [17]

Los Framework se han convertido en la piedra angular de la moderna ingeniería del software ya que están compuestos por componentes personalizables e intercambiables para el desarrollo de una aplicación, en otras palabras, es una aplicación genérica incompleta y/o configurable a la que se le puede añadir las últimas piezas para construir una aplicación concreta.

Los objetivos principales que persigue un framework son: acelerar el proceso de desarrollo, reutilizar código ya existente y promover buenas prácticas de desarrollo como el uso de patrones.

Un framework Web, por tanto, pude definirse como un conjunto de clases y componentes que forman un diseño reutilizable al cual se le pueden incluir módulos para facilitar y agilizar el desarrollo de aplicaciones Web como Django (Python), Ruby on Rails (Ruby) y Codelgniter (PHP). Por lo general pueden ser orientados a la interfaz de usuario, aplicaciones de publicación de documentos y/o a la combinación de estos elementos.

La mayoría de los Frameworks Web incorporan en su arquitectura el patrón MVC, separando los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres capas distintas, donde la vista es la página HTML y el código que provee de datos dinámicos a la página; el modelo es el SGBD y la lógica del negocio; y el controlador es el responsable de recibir los eventos de entrada desde la vista.

1.9.1 TurboGears

TurboGears constituye un framework diseñado para desarrollar aplicaciones Web con Python, está compuesto por una serie de componentes que le permiten al desarrollador hacer aplicaciones web de forma rápida. Todos estos componentes están empaquetados por separado, y pueden ser descargados y utilizados sin TurboGears. Posee amplia documentación, es multiplataforma, cuenta con una numerosa comunidad de desarrolladores y usuarios.

Los principales componentes son:

MochiKit: Una biblioteca JavaScript que incluye componentes tales como logging (del inglés), arrastrar (drag: del inglés), soltar (drop: del inglés), y otros efectos visuales que son comunes en muchas bibliotecas. Permite añadir comportamiento asíncrono (AJAX) a una aplicación web, carga y manipulación de conjuntos de datos JSON y contiene un conjunto de funciones para facilitar la creación dinámica de componentes, llamada Mochikid.DOM. Sus principales características son:

- Gestión de tareas asíncronas.
- Funciones de manipulación de objetos, arrays y comparaciones.
- Capa de acceso al DOM.
- Iteraciones y enumeradores en JavaScript.
- Capacidades de log.
- Efectos visuales y funciones de color.
- Funciones Python.

Kid: Una plantilla de lenguaje que le permite incluir código Python incrustado en una página XHTML. Se puede incrustar código Python, explícitamente con una etiqueta, o utilizar un conjunto de atributos. Incluye estructuras condicionales y bucles para definir el contenido dinámico. Las plantillas son constituidas por XML, lenguaje similar a Zope compilado a Python byte-code.

CherryPy: Un framework orientado a objetos que permite desarrollar aplicaciones web en Python. CherryPy incluye mecanismos para hacer simples operaciones centradas en la web tales como mapeo de URL solicitadas a métodos Python usando decoradores, control del servidor y solicitudes de filtrado.

SQLAlchemy: Es un moderno Mapeador Objeto-Relacional, que proporciona un potente y flexible sistema de gestión de la conexión entre la memoria y los objetos de Python, además posee almacenamiento de datos relacional que proporciona la persistencia de los objetos.

TurboGears también incluye una amplia variedad de otras funciones y servicios,

incluidos los basados en la web y líneas de comandos, servicios de ayuda, administrador de tareas del proyecto, un servidor web incorporado, un marco de prueba y una herramienta basada en web para la administración del modelo de objetos, entre otros. Incorpora la arquitectura MVC de la misma manera que la mayoría de los demás Frameworks actuales de aplicaciones web (como Ruby on Rails).

1.9.2 Dojo Toolkit

Dojo es un framework que contiene *APIs* (Application Programming Interface) y *Widgets* (controles) para facilitar el desarrollo de aplicaciones Web que utilicen tecnología AJAX. Contiene un sistema de empaquetado inteligente, abstracción de eventos, almacenamiento de APIs en el cliente, e interacción de APIs con AJAX. Resuelve asuntos de usabilidad comunes como pueden ser la navegación y detección del navegador, soportar cambios de URL en la barra de URLs para luego regresar a ellas y la habilidad de degradar cuando AJAX/JavaScript no es completamente soportado en el cliente. Es conocido como "la navaja suiza del ejército de las bibliotecas JavaScript". Proporciona una gama más amplia de opciones en una sola biblioteca JavaScript y es compatible con navegadores antiguos

1.10 Plataforma

1.10.1 Aplicación Web

Una aplicación Web es un sistema informático que los usuarios utilizan accediendo a un servidor Web a través de Internet o de una intranet. La facilidad para actualizar y mantener aplicaciones Web sin distribuir e instalar software en miles de potenciales clientes es otra razón de su popularidad. Aplicaciones como los web mails, wikis o tiendas en línea son ejemplos bien conocidos de aplicaciones Web. Una ventaja principal es que deberían funcionar igual independientemente de la versión del sistema operativo instalado en el cliente. En vez de crear clientes para Windows, Mac OS X, GNU/Linux, y otros sistemas operativos, la aplicación es escrita una vez y es mostrada casi en todas las plataformas.

Ventajas:

- Desarrollo barato, sencillo y rápido.
- Datos centralizados y fácil integración de datos de múltiples fuentes.

- Permiten el desarrollo de comunidades que dan valor a las aplicaciones (software social).
- Una empresa puede migrar de sistema operativo o cambiar el hardware libremente sin afectar el funcionamiento de las aplicaciones del servidor.
- No se requieren complicadas combinaciones de hardware/software para utilizar estas aplicaciones. Solo un computador con un buen navegador Web.
- Actualizar o hacer cambios en el software es sencillo y sin riesgos de incompatibilidades. Existe solo una versión en el servidor lo que implica que no hay que distribuirla entre los demás computadores. El proceso es rápido y limpio.
- Se facilita el trabajo a distancia. Se puede trabajar desde cualquier PC (computadora personal) o computador portátil con conexión a Internet o a una red interna o privada.
- Al funcionar en un navegador, se requiere un conocimiento básico de informática para utilizar una aplicación Web.

Todas estas ventajas dejan claro el potencial de las aplicaciones Web. “La utilización de ésta tecnología conlleva a reducir costos y complicaciones, y proporciona mayor libertad a la hora de realizar cualquier tipo de cambios”. [18]

1.10.2 Modelo Cliente – Servidor

Se puede definir la computación cliente/servidor como una arquitectura distribuida que permite a los usuarios finales obtener acceso a la información en forma transparente aún en entornos multiplataforma. Las comunicaciones en el cliente y el servidor son generalmente bidireccionales, el cliente envía un mensaje solicitando un determinado servicio (petición) a un servidor y este envía uno o varios mensajes con la respuesta. En un sistema distribuido cada máquina puede cumplir uno u otro rol.

En otras palabras la arquitectura cliente/servidor es una extensión de programación modular en la que la base fundamental es separar una gran pieza de software en módulos con el fin de proveer usabilidad, flexibilidad, interoperabilidad y escalabilidad en las comunicaciones y hacer más fácil el desarrollo y mejorar su mantenimiento, cuenta con tres elementos fundamentales; cliente, servidor y red de comunicación.

La utilización de las diferentes aplicaciones o servicios de Internet se lleva a cabo respondiendo a este modelo, que es una forma de especializar terminales y programas para que las actividades y tareas se ejecuten con la mayor eficiencia posible. Mediante esta arquitectura el usuario puede acceder a la información sin tener en cuenta su ubicación física y donde pueda estar alojada la misma.

La idea es tratar a una computadora como un instrumento, que por sí sola pueda realizar muchas tareas, pero con la consideración de que realice aquellas que son más adecuadas a sus características. Esto se aplica tanto a clientes como servidores, la forma más estándar de aplicación y uso de estos sistemas es mediante la explotación de las PC's a través de interfaces gráficas de usuario; mientras que la administración de datos y su seguridad e integridad se deja a cargo de computadoras centrales. Usualmente la mayoría del trabajo pesado se hace en el proceso llamado servidor y el o los procesos clientes sólo se ocupan de la interacción con el usuario. De esta forma un servidor brinda servicios a múltiples clientes de forma concurrente y los cambios realizados en las plataformas de los clientes o de los servidores, ya sean por actualización o por reemplazo tecnológico, se realizan de una manera transparente para el usuario final.

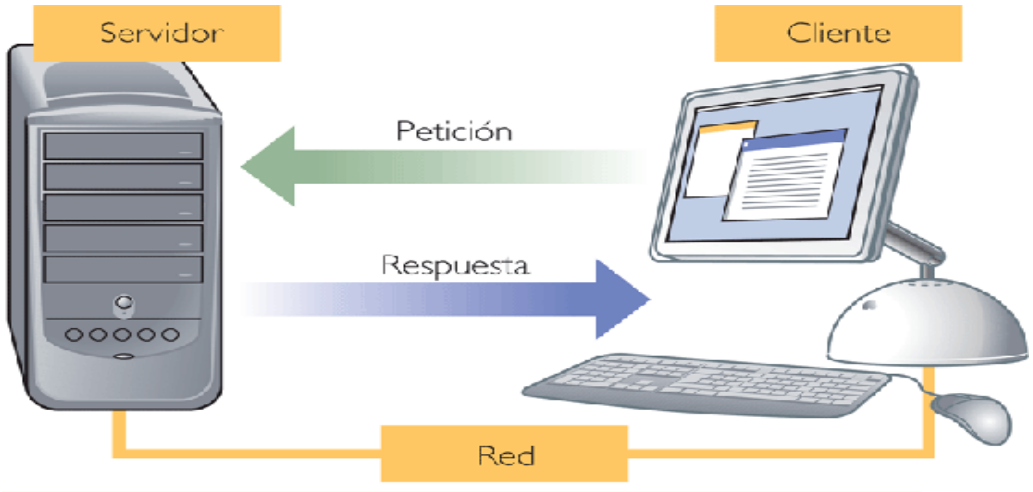


Figura1.0 Modelo Cliente Servidor

- El servidor presenta a todos sus clientes una interfaz única y bien definida.
- El cliente no necesita conocer la lógica del servidor, sólo su interfaz externa.
- Las tareas del cliente y del servidor tienen diferentes requerimientos en cuanto a recursos de cómputo como velocidad del procesador, memoria, velocidad y

capacidades del disco y dispositivos de entrada y salida.

- Las funciones del cliente y el servidor pueden estar en plataformas separadas, o en la misma plataforma.

- La relación establecida puede ser de muchos a uno, en la que un servidor puede dar servicio a muchos clientes, regulando su acceso a recursos compartidos.

- La estructura inherentemente modular facilita además la integración de nuevas tecnologías y el crecimiento de la infraestructura computacional, favoreciendo así la escalabilidad de las soluciones.

1.11 Conclusiones

En este capítulo se ha hecho un análisis de algunos lenguajes, metodologías y herramientas, siendo elegidos los mejores candidatos para darle solución a la propuesta. Después de un profundo análisis se ha llegado a la conclusión de que es importante realizar una aplicación Web teniendo en cuenta la facilidad de acceso a la misma desde distintas ubicaciones. Es de vital importancia el dominio de estas herramientas seleccionadas y analizar la mejor forma de aplicarlas, para desarrollar un sistema de máxima calidad que cumpla con los requisitos propuestos y brinde al cliente una versión del producto que satisfaga sus intereses

Capítulo II. Características del Sistema

2.1 Introducción

En este capítulo se realiza un análisis de las características del sistema a desarrollar, haciendo hincapié en la situación problémica que da origen al mismo. Además se abordan aspectos de gran importancia como son la propuesta de la arquitectura, el objeto de automatización y las interfaces de usuarios.

2.2 Objeto de Automatización

Con el desarrollo de esta aplicación se automatizarán los procesos de configuración de las colas de llamadas y agentes que atienden las mismas en la PBX asterisk, así como los procesos de reportes en tiempo real e históricos que se brindan a los administradores del sistema para llevar estadísticas que ayuden a la empresa a mejorar los niveles de servicio en cada momento.

2.3 Información que se maneja

La información que se maneja es la brindada por el servidor asterisk ya sea a través de sus ficheros de configuración o la generada por el mismo al enviarle comandos.

2.4 Propuesta del Sistema

Se ha decidido darle solución al problema mediante una aplicación Web que facilite las tareas antes mencionadas en el menor tiempo posible. Esta aplicación se realizará utilizando el framework Turbogears, cumpliendo los requisitos pedidos por el usuario

2.5 Arquitectura del sistema

Para un mejor funcionamiento y rendimiento del sistema se propone una arquitectura distribuida, donde se cuenta con una PC cliente conectada al servidor de aplicaciones PLATEL con el módulo ACD, el cual hace peticiones al servidor Asterisk y al servidor de Base de Datos (BD), los que se encuentran relacionados entre sí. El sistema además puede funcionar sobre una arquitectura centralizada, ubicándose en una misma PC el servidor de aplicaciones, servidor Asterisk y el servidor de BD.(VER ANEXO #2)

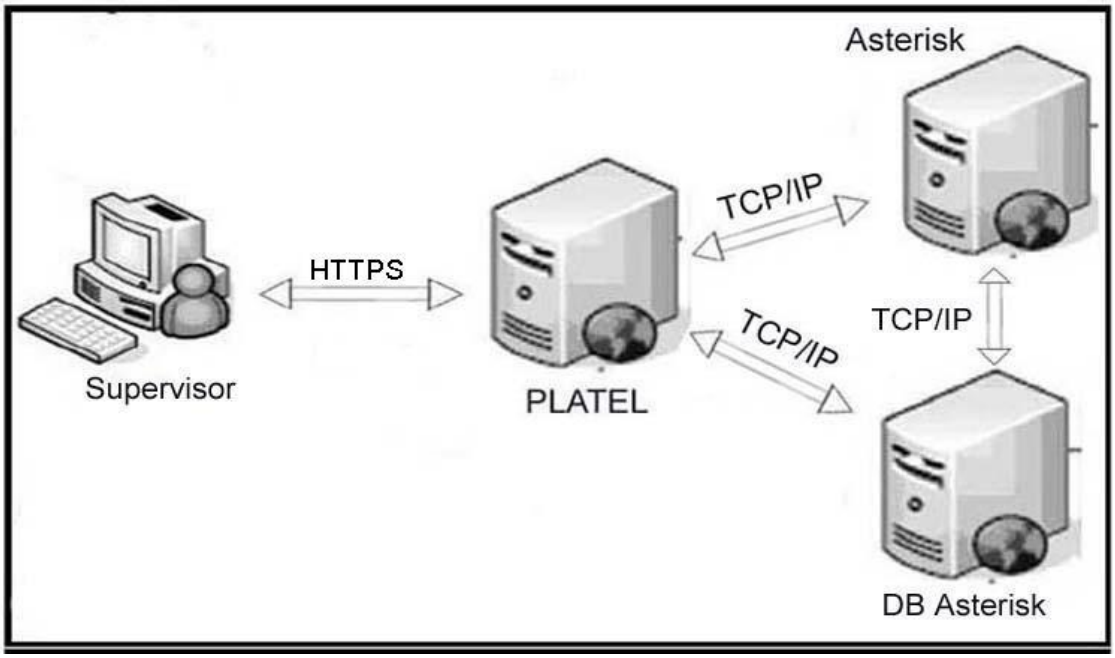


Figura 2.0 Arquitectura del sistema propuesto

Distribuidor Automático de Llamadas para PLATEL: Esta aplicación es la encargada de automatizar el trabajo con los ficheros de asterisk para la configuración del Distribuidor Automático de Llamadas, brindando la posibilidad de guardar dicha configuración en BD. Brinda además reportes en tiempo real y reportes históricos de las colas de llamadas y los puestos de trabajo.

2.6 Diseño de la Base de Datos

Una de las tareas más importantes a la hora de construir una aplicación Web es la base de datos, ya que uno de sus objetivos fundamentales es brindar la persistencia al modelo que se describe en el capítulo.

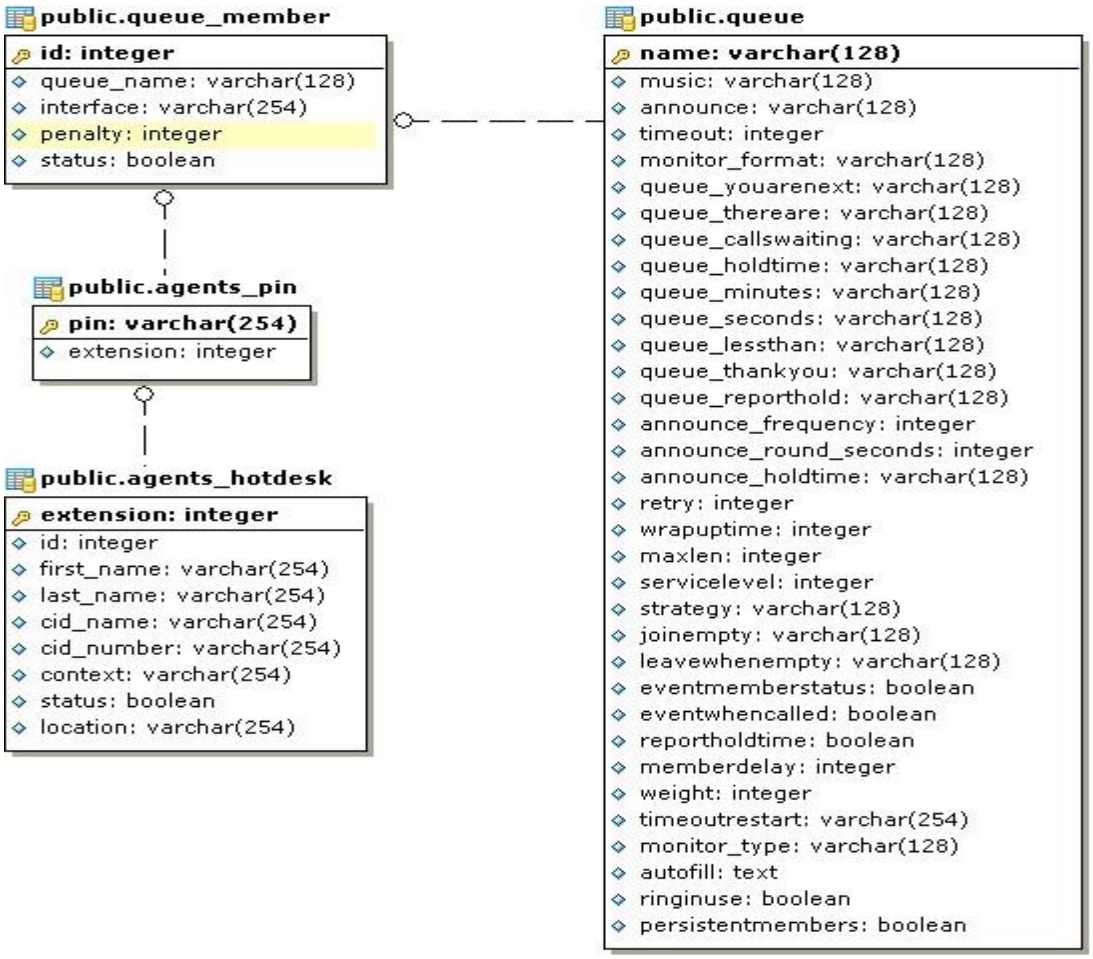


Figura 2.1 Diagrama Entidad Relación de la BD

2.7 Interfaz de usuario

Configurar Colas

Interfaz mediante la cual el supervisor del sistema realiza las distintas operaciones sobre las colas de llamadas. Las mismas pueden ser, agregar una cola, eliminar una cola, agregar un miembro a una cola o eliminarle un miembro a una cola, dependiendo de que actividad sea la que desee realizar en cada momento. (VER ANEXO #1)

Configurar Agentes

Interfaz mediante la cual el supervisor del sistema realiza las distintas operaciones sobre los agentes. Las mismas pueden ser, agregar un agente, eliminar un agente, o asignarle pines. Dependiendo la opción que se seleccione será la operación que se realizará en cada momento. (VER ANEXO #1)

Colas en Tiempo Real

Esta interfaz será manejada por el supervisor del sistema, desde la misma se podrá monitorear en tiempo real la cola de llamadas seleccionada, de la cual se mostraran una serie de datos así como las llamadas que se encuentran en la misma. (VER ANEXO #1)

Agentes en Tiempo Real

Esta interfaz será manejada por el supervisor del sistema, desde la misma se podrá monitorear en tiempo real los agentes seleccionados, los cuales pueden ser los que se encuentren online, offline o todos al mismo tiempo. (VER ANEXO #1)

Reporte Histórico de Colas

En esta interfaz se muestran datos históricos de las colas en el rango de tiempo seleccionado por el supervisor del sistema. (VER ANEXO #1)

Reporte Histórico de Puestos

En esta interfaz se muestran datos históricos de los agentes en el rango de tiempo seleccionado por el supervisor del sistema. (VER ANEXO #1)2.8 Requisitos no funcionales del sistema

Requerimientos de apariencia o interfaz externa

La aplicación propuesta será usada por personas que tengan o no conocimientos básicos de informática y de Asterisk, la interfaz debe ser amigable y fácil de usar, de manera que no sea una dificultad para los usuarios el trabajo con la misma.

Requerimientos de usabilidad

A los supervisores del sistema se les dará un adiestramiento básico en el uso de la aplicación. Estas personas tendrán un nivel de acceso amplio en la aplicación para poder darle respuesta a cada incidente ocurrido.

Requerimientos de rendimiento

La eficiencia del producto estará determinada en gran medida por el aprovechamiento de los recursos que se disponen en el modelo cliente/servidor. La aplicación propuesta debe ser rápida y el tiempo de respuesta debe ser el mínimo posible, adecuado a la rapidez con que el cliente requiere la respuesta a su petición.

Requerimiento de portabilidad

La herramienta podrá ser usada bajo cualquier distribución de Linux, utilizando una arquitectura distribuida, donde se cuenta con un servidor de aplicaciones, el servidor Asterisk y el servidor de Base de Datos en máquinas separadas. Además estos servidores pueden encontrarse en la misma PC sin ocasionar problema alguno, conformando una arquitectura centralizada.

Requerimientos de seguridad

- **Confiabilidad:** La información manejada por el sistema debe estar protegida de acceso no autorizado.
- **Integridad:** La información manejada por el sistema debe ser objeto de cuidadosa protección contra la corrupción y estados de inconsistencia.
- **Disponibilidad:** La aplicación deberá estar disponible en todo momento para aquellas personas con acceso a la información y los mecanismos utilizados para lograr la seguridad no deben ser un obstáculo a los usuarios para obtener los datos deseados en un momento dado.

Requerimientos de software

En las computadoras de los usuarios solo se requiere el navegador Web Mozilla Firefox 3.0 o superior con la opción de JavaScript activada y cualquier distribución de Linux.

Requerimientos de hardware

En el cliente se requiere una máquina con 256 MB de RAM como mínimo, el servidor Web debe tener 512 MB de RAM como mínimo, todas las máquinas implicadas en la

funcionalidad de la aplicación deben estar conectadas a la red de al menos 100 Mbps de velocidad.

2.9 Conclusiones

En el presente capítulo se realizó un estudio detallado de las características del sistema a implementar obteniéndose como resultado una propuesta del sistema así como una descripción de las distintas interfaces que componen el mismo y la propuesta de la arquitectura a implementar.

Capítulo 3 Exploración y Planificación

3.1 Introducción

En el presente capítulo se hace alusión a las fases de exploración y planificación propias de la metodología de desarrollo utilizada para la implementación del sistema que se propone. Se exponen además los artefactos generados durante el transcurso de las mismas.

3.2 Fase de exploración

“La metodología de desarrollo Extreme Programming comienza con la fase de exploración. Durante esta se realiza el proceso de identificación de las historias de usuario (UH, del inglés: User Histories), así como la familiarización de los equipos de trabajo con las tecnologías y herramientas seleccionadas para la construcción del proyecto”. [19]

3.2.1 Historias de Usuario

“Las historias de usuario son la forma en que se especifican en XP los requisitos del sistema. Estas se escriben desde la perspectiva del cliente aunque los desarrolladores pueden brindar también su ayuda en la identificación de las mismas”. [19] “El contenido de estas debe ser concreto y sencillo”. [20]

Durante la fase de exploración se identificaron diez historias de usuario, las cuales se detallan a continuación.

Tabla 3.2.1 Historia de usuario Configurar Cola

Historia de Usuario	
Número: 1	Usuario: Supervisor
Nombre historia: Configurar Cola	
Prioridad en negocio: Alta	Riesgo en desarrollo: Medio
Puntos estimados: 1.5	Iteración asignada: 1
Programador responsable: Yusnel Cruz Castillo-Liorge Corria Sanchez	
Descripción: El Supervisor introduce el nombre, estrategia, música, longitud, entre otros parámetros con los que desea que la cola quede configurada en la base de datos.	

Tabla 3.2.2 Historia de usuario Configurar Agente

Historia de Usuario	
Número: 2	Usuario: Supervisor
Nombre historia: Configurar Agente	
Prioridad en negocio: Alta	Riesgo en desarrollo: Medio
Puntos estimados: 1	Iteración asignada: 1
Programador responsable: Yusnel Cruz Castillo-Liorge Corria Sanchez	
Descripción: El Supervisor introduce el nombre, apellido, alias y la extensión con los cuales desea configurar el agente en la base de datos.	

Tabla 3.2.3 Historia de usuario Asignar Agente a Cola

Historia de Usuario	
Número: 3	Usuario: Supervisor
Nombre historia: Asignar Agente a Cola	
Prioridad en negocio: Alta	Riesgo en desarrollo: Medio
Puntos estimados: 1	Iteración asignada: 1
Programador responsable: Yusnel Cruz Castillo-Liorge Corria Sanchez	
Descripción: El Supervisor selecciona la cola y el agente que quiere asignar a dicha cola, además inserta un número de penalidad para el agente.	

Tabla 3.2.4 Historia de usuario Eliminar Agente de Cola

Historia de Usuario	
Número: 4	Usuario: Supervisor
Nombre historia: Eliminar Agente de Cola	
Prioridad en negocio: Media	Riesgo en desarrollo: Bajo
Puntos estimados: 1	Iteración asignada: 2
Programador responsable: Yusnel Cruz Castillo-Liorge Corria Sanchez	
Descripción: El Supervisor selecciona la cola y el agente que quiere eliminar de dicha cola.	

Tabla 3.2.5 Historia de usuario Eliminar Cola

Historia de Usuario	
Número: 5	Usuario: Supervisor
Nombre historia: Eliminar Cola	
Prioridad en negocio: Media	Riesgo en desarrollo: Bajo
Puntos estimados: 1	Iteración asignada: 2
Programador responsable: Yusnel Cruz Castillo-Liorge Corria Sanchez	
Descripción: El Supervisor escoge la cola que desea eliminar de la base de datos.	

Tabla 3.2.6 Historia de usuario Eliminar Agente

Historia de Usuario	
Número: 6	Usuario: Supervisor
Nombre historia: Eliminar Agente	
Prioridad en negocio: Media	Riesgo en desarrollo: Bajo
Puntos estimados: 1	Iteración asignada: 2
Programador responsable: Yusnel Cruz Castillo-Liorge Corria Sanchez	
Descripción: El Supervisor selecciona el agente que desea eliminar de la base de datos	

Tabla 3.2.7 Historia de usuario Monitorear Agentes

Historia de Usuario	
Número: 7	Usuario: Supervisor
Nombre historia: Monitorear Agentes	
Prioridad en negocio: Media	Riesgo en desarrollo: Alto
Puntos estimados: 2	Iteración asignada: 3
Programador responsable: Yusnel Cruz Castillo-Liorge Corria Sanchez	
Descripción: El Supervisor selecciona el criterio (todos, offline, online) por el que desea monitorear los agentes.	

Tabla 3.2.8 Historia de usuario Monitorear Cola

Historia de Usuario	
Número: 8	Usuario: Supervisor
Nombre historia: Monitorear Cola	
Prioridad en negocio: Media	Riesgo en desarrollo: Alto
Puntos estimados: 2	Iteración asignada: 3
Programador responsable: Yusnel Cruz Castillo-Liorge Corria Sanchez	
Descripción: El Supervisor selecciona la cola que desea monitorear.	

Tabla 3.2.9 Historia de usuario Reporte Histórico de Puestos

Historia de Usuario	
Número: 9	Usuario: Supervisor
Nombre historia: Reporte Histórico de Puestos	
Prioridad en negocio: Baja	Riesgo en desarrollo: Medio
Puntos estimados: 2	Iteración asignada: 4
Programador responsable: Yusnel Cruz Castillo-Liorge Corria Sanchez	
Descripción: El Supervisor selecciona el rango de tiempo en el que desea ver los reportes de los puestos (número de llamadas atendidas, tiempo promedio en llamadas, porcentaje de llamadas atendidas, etc.).	

Tabla 3.2.10 Historia de usuario Reporte Histórico de Colas

Historia de Usuario	
Número: 10	Usuario: Supervisor
Nombre historia: Reporte Histórico de Colas	
Prioridad en negocio: Baja	Riesgo en desarrollo: Medio
Puntos estimados: 2	Iteración asignada: 4
Programador responsable: Yusnel Cruz Castillo-Liorge Corria Sanchez	
Descripción: El Supervisor selecciona el rango de tiempo en el que desea ver los reportes de las colas. (número de llamadas entrante, atendidas, perdidas, etc.).	

3.3 Planificación

"Durante la fase de planificación se realiza una estimación del esfuerzo que costará implementar cada historia de usuario. Este se expresa utilizando como medida el punto. Un punto se considera como una semana ideal de trabajo donde los miembros de los equipos de desarrollo trabajan el tiempo planeado sin ningún tipo de interrupción". [21]. "Esta estimación incluye todo el esfuerzo asociado a la implementación de la historia de usuario, por ejemplo: las pruebas unitarias, la integración y refactorización del código y la preparación y ejecución de las pruebas de aceptación". [19]

3.3.1. Estimación de esfuerzo por Historias de Usuario

Para el desarrollo de la aplicación propuesta en este trabajo se realizó una estimación del esfuerzo para cada una de las historias de usuario identificadas, llegándose a los siguientes resultados.

Tabla 3.3.1 Estimación de esfuerzo por historia de usuario

Historias de Usuario	Puntos estimados
Configurar Cola	1.5
Configurar Agente	1
Asignar Agente a Cola	1
Eliminar Agente de Cola	1
Eliminar Cola	1
Eliminar Agente	1
Monitorear Agentes	2
Monitorear Cola	2

Reporte Histórico de Puestos	2
Reporte Histórico de Colas	2

3.3.2. Plan de Iteraciones

Una vez identificadas las historias de usuario del sistema y estimado el esfuerzo dedicado a la realización de cada una de ellas se procede a la planificación de la etapa de implementación de la aplicación. Se decidió realizar la implementación de la aplicación en cuatro iteraciones, detalladas a continuación.

Iteración 1

Con la implementación de estas historias de usuario en esta iteración quedarán configurados en el servidor Asterisk las colas de llamadas y los agentes. Así como la opción de asignarle miembros a las colas.

Iteración 2

Esta iteración tiene como objetivo la implementación de las historias de usuario que permitirán eliminar las colas y los agentes asi como la opción de eliminar agentes de determinadas colas

Iteración 3

Esta iteración tiene como objetivo la implementación de las historias de usuario que permitirán monitorear en tiempo real las colas y los agentes.

Iteración 4

La implementación de estas historias de usuarios en esta iteración permitirá obtener reportes históricos de las colas y los puestos. Y al finalizar la misma quedará terminada la versión 1.0 del producto final del módulo ACD.

3.3.3 Plan de duración de las iteraciones

Como parte del ciclo de vida de un proyecto utilizando la metodología XP se crea el plan de duración de cada una de las iteraciones, según los equipos de desarrollo

con que se cuente, en este caso se realiza para el único equipo de desarrollo que participa en la implementación del software. Este plan se encarga de mostrar las historias de usuario que serán cumplimentadas en cada una de las iteraciones, así como la duración estimada de de cada una y el orden en que se implementarán.

Tabla 3.3.2 Plan de duración de las iteraciones

Iteración	Orden de las historias de usuario a implementar	Duración total de la iteración
Iteración 1	1.1- Configurar Cola 1.2- Configurar Agente 1.3- Asignar Agente a Cola	3.5 semanas.
Iteración 2	2.1- Eliminar Agente de Cola 2.2- Eliminar Cola 2.3- Eliminar Agente	3 semanas
Iteración 3	3.1- Monitorear Agentes 3.2- Monitorear Cola	4 semanas.
Iteración 4	4.1- Reporte Histórico de Puestos 4.2- Reporte Histórico de Colas	4 semanas.

3.3.4 Plan de entregas

Plan de entregas aproximadas ideado para la fase de implementación. Al finalizar la cuarta iteración se obtendrá la primera versión del producto final del módulo ACD.

Tabla 3.3.4 Plan de entregas

Software	Final 1ra Iteración 4ta semana de Enero.	Final 2da Iteración 3ra semana de Febrero.	Final 3ra Iteración 3ra semana de Marzo.	Final 4ta Iteración 3ra semana de Abril.
ACD	0.1	0.2	0.3	1.0

3.4 Conclusiones

En este capítulo se abordó todo lo referente a las fases de exploración y planificación del proyecto, haciendo una descripción de cada uno de los artefactos generados en las mismas durante su desarrollo dando una idea de la magnitud del proyecto a realizar.

Capítulo 4 Implementación y Pruebas

4.1 Introducción

”XP plantea que la implementación de un software se hace de forma iterativa, obteniendo al final de cada iteración un producto funcional que debe ser probado y mostrado al cliente para que los desarrolladores trabajen conociendo las opiniones dadas sobre el mismo”. [21] En el presente capítulo se detallan las iteraciones llevadas a cabo durante la etapa de construcción del sistema, exponiendo las tareas generadas por cada historia de usuario, así como las pruebas de aceptación efectuadas sobre el proyecto.

Durante el transcurso de las iteraciones se realiza la implementación de las historias de usuario seleccionadas para cada una de estas. Al inicio de las mismas, se lleva a cabo una revisión del plan de iteraciones y se modificaría de ser necesario.

4.2 Iteración 1

Durante esta iteración se abordaron las historias de usuario elegidas y se construyó la base de la arquitectura del sistema con el fin de obtener un producto con las funcionalidades primarias para ser mostrado al cliente y obtener una rápida y amplia retroalimentación del mismo.

Tabla 4.2.1 Historias de usuario implementadas en la primera iteración

Historias de Usuario	Estimación	Real
Configurar Cola	1.5	1.5
Configurar Agente	1	1
Asignar Agente a Cola	1	1

4.2.1 Tareas de las historias de usuario implementadas en la primera iteración

Configurar Cola

Tabla 4.2.2. Tarea #1 de la historia de usuario Configurar Cola

Tarea	
Número tarea: 1	Número historia: 1
Nombre tarea: Modelar la tabla en la que se guarda la configuración de las colas.	
Tipo de tarea: Desarrollo	Puntos estimados: 0.27
Fecha inicio: 5 Enero 2009	Fecha fin: 6 Enero 2009
Programador responsable: Yusnel Cruz Castillo-Liorge Corria Sanchez	
Descripción: Se modela la tabla en la base datos en la que se guardará la configuración de la cola.	

Tabla 4.2.3. Tarea #2 de la historia de usuario Configurar Cola

Tarea	
Número tarea: 2	Número historia: 1
Nombre tarea: Implementar funcionalidad para configurar las colas.	
Tipo de tarea: Desarrollo	Puntos estimados: 0.41
Fecha inicio: 7 Enero 2009	Fecha fin: 9 Enero 2009
Programador responsable: Yusnel Cruz Castillo-Liorge Corria Sanchez	
Descripción: Se implementa la funcionalidad para guardar los datos en la tabla	

antes modelada.

Tabla 4.2.4 Tarea #3 de la historia de usuario Configurar Cola

Tarea	
Número tarea: 3	Número historia: 1
Nombre tarea: Implementar interfaz visual	
Tipo de tarea: Desarrollo	Puntos estimados: 0.84
Fecha inicio: 10 Enero 2009	Fecha fin: 14 Enero 2009
Programador responsable: Yusnel Cruz Castillo-Liorge Corria Sanchez	
Descripción: Se implementa la interfaz visual para enviar los datos de la cola.	

Configurar Agente

Tabla 4.2.5. Tarea #1 de la historia de usuario Configurar Agente

Tarea	
Número tarea: 1	Número historia: 2
Nombre tarea: Modelar la tabla en la que se guarda la configuración de los agentes.	
Tipo de tarea: Desarrollo	Puntos estimados: 0.33
Fecha inicio: 22 Enero 2009	Fecha fin: 24 Enero 2009
Programador responsable: Yusnel Cruz Castillo-Liorge Corria Sanchez	
Descripción: Se modela la tabla en la base datos en la cual se guardará la configuración del agente.	

Tabla 4.2.6. Tarea #2 de la historia de usuario Configurar Agente

Tarea	
Número tarea: 2	Número historia: 2
Nombre tarea: Implementar funcionalidad para configurar los agentes.	
Tipo de tarea: Desarrollo	Puntos estimados: 0.33
Fecha inicio: 24 Enero 2009	Fecha fin: 26 Enero 2009
Programador responsable: Yusnel Cruz Castillo-Liorge Corria Sanchez	
Descripción: Se implementa la funcionalidad para guardar los datos en la tabla antes modelada.	

Tabla 4.2.7 Tarea #3 de la historia de usuario Configurar Agente

Tarea	
Número tarea: 3	Número historia: 2
Nombre tarea: Implementar interfaz visual	
Tipo de tarea: Desarrollo	Puntos estimados: 0.33
Fecha inicio: 26 Enero 2009	Fecha fin: 28 Enero 2009
Programador responsable: Yusnel Cruz Castillo-Liorge Corria Sanchez	
Descripción: Se implementa la interfaz visual para enviar los datos del agente.	

Asignar Agente a Cola

Tabla 4.2.8 Tarea #2 de la historia de usuario Asignar Agente a Cola

Tarea	
Número tarea: 1	Número historia: 3
Nombre tarea: Implementar la funcionalidad para asignar el agente a una cola.	
Tipo de tarea: Desarrollo	Puntos estimados: 0.5
Fecha inicio: 5 Febrero 2009	Fecha fin: 8 Febrero 2009
Programador responsable: Yusnel Cruz Castillo-Liorge Corria Sanchez	
Descripción: Se implementa en la clase controladora la funcionalidad que permitirá asignar un agente a la cola.	

Tabla 4.2.9 Tarea #2 de la historia de usuario Asignar Agente a Cola

Tarea	
Número tarea: 2	Número historia: 3
Nombre tarea: Implementar la interfaz visual.	
Tipo de tarea: Desarrollo	Puntos estimados: 0.5
Fecha inicio: 8 Febrero 2009	Fecha fin: 11 Febrero 2009
Programador responsable: Yusnel Cruz Castillo-Liorge Corria Sanchez	
Descripción: Se implementa la interfaz visual para asignar un agente a la cola.	

4.3 Iteración 2

En el transcurso de esta iteración se implementaron las historias de usuario que permiten eliminar las colas de llamadas y los agentes, así como la opción de asignar agentes a las colas.

Tabla 4.3.1 Historias de usuario implementadas en la segunda iteración

Historias de Usuario	Estimación	Real
Eliminar Agente de Cola	1	1
Eliminar Cola	1	1
Eliminar Agente	1	1

4.3.1 Tareas de las historias de usuario implementadas en la segunda iteración

Eliminar Agente de Cola

Tabla 4.3.2 Tarea #1 de la historia de usuario Eliminar Agente de Cola

Tarea	
Número tarea: 1	Número historia: 4
Nombre tarea: Implementar la funcionalidad para eliminar agente de cola	
Tipo de tarea: Desarrollo	Puntos estimados: 0.5
Fecha inicio: 12 Febrero 2009	Fecha fin: 15 Febrero 2009
Programador responsable: Yusnel Cruz Castillo-Liorge Corria Sanchez	
Descripción: Se implementa en la clase controladora la funcionalidad que permitirá eliminar el agente de una cola.	

Tabla 4.3.3 Tarea #2 de la historia de usuario Eliminar Agente de cola

Tarea	
Número tarea: 2	Número historia: 4
Nombre tarea: Implementar la interfaz visual	
Tipo de tarea: Desarrollo	Puntos estimados: 0.5
Fecha inicio: 15 Febrero 2009	Fecha fin: 18 Febrero 2009
Programador responsable: Yusnel Cruz Castillo-Liorge Corria Sanchez	
Descripción: Se implementa la interfaz visual para eliminar el agente de la cola.	

Eliminar Cola

Tabla 4.3.4 Tarea #1 de la historia de usuario Eliminar Cola

Tarea	
Número tarea: 1	Número historia: 5
Nombre tarea: Implementar la funcionalidad para eliminar la cola	
Tipo de tarea: Desarrollo	Puntos estimados: 0.5
Fecha inicio: 15 Enero 2009	Fecha fin: 18 Enero 2009
Programador responsable: Yusnel Cruz Castillo-Liorge Corria Sanchez	
Descripción: Se implementa en el controlador la funcionalidad para eliminar cola.	

Tabla 4.3.5 Tarea #2 de la historia de usuario Eliminar Cola

Tarea	
Número tarea: 2	Número historia: 5
Nombre tarea: Implementar la interfaz visual	
Tipo de tarea: Desarrollo	Puntos estimados: 0.5
Fecha inicio: 18 Enero 2009	Fecha fin: 21 Enero 2009
Programador responsable: Yusnel Cruz Castillo-Liorge Corria Sanchez	
Descripción: Se implementa la interfaz visual para eliminar la cola.	

Eliminar Agente

Tabla 4.3.6 Tarea #1 de la historia de usuario Eliminar Agente

Tarea	
Número tarea: 1	Número historia: 6
Nombre tarea: Implementar la funcionalidad para eliminar agente	
Tipo de tarea: Desarrollo	Puntos estimados: 0.5
Fecha inicio: 29 Enero 2009	Fecha fin: 1 Febrero 2009
Programador responsable: Yusnel Cruz Castillo-Liorge Corria Sanchez	
Descripción: Se implementa en la clase controladora la funcionalidad que permite eliminar agente.	

Tabla 4.3.7 Tarea #2 de la historia de usuario Eliminar Agente

Tarea	
Número tarea: 2	Número historia: 6
Nombre tarea: Implementar la interfaz visual	
Tipo de tarea: Desarrollo	Puntos estimados: 0.5
Fecha inicio: 1 Febrero 2009	Fecha fin: 4 Febrero 2009
Programador responsable: Yusnel Cruz Castillo-Liorge Corria Sanchez	
Descripción: Se implementa la interfaz visual para eliminar el agente.	

4.4 Iteración 3

En el transcurso de esta iteración se implementaron las historias de usuario que permiten obtener reportes en tiempo real de los agentes y las colas.

Tabla 4.4.1 Historias de usuario implementadas en la tercera iteración

Historias de Usuario	Estimación	Real
Monitorear Agentes	2	2
Monitorear Cola	2	2

4.4.1 Tareas de las historias de usuario implementadas en la tercera iteración

Monitorear Agentes

Tabla 4.4.2 Tarea #1 de la historia de usuario Monitorear Agentes

Tarea	
Número tarea: 1	Número historia: 7
Nombre tarea: Implementar la funcionalidad para monitorear agente.	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha inicio: 19 Febrero 2009	Fecha fin: 25 Febrero 2009
Programador responsable: Yusnel Cruz Castillo-Liorge Corria Sanchez.	
Descripción: Se implementa en el controlador la funcionalidad que va a permitir monitorear los agentes que estén configurados en la base de datos del servidor Asterisk.	

Tabla 4.4.3 Tarea #2 de la historia de usuario Monitorear Agentes

Tarea	
Número tarea: 2	Número historia: 7
Nombre tarea: Implementar la interfaz visual.	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha inicio: 25 Febrero 2009	Fecha fin: 3 Marzo 2009
Programador responsable: Yusnel Cruz Castillo-Liorge Corria Sanchez.	

Descripción: Se implementa la interfaz visual para mostrar los datos obtenidos.

Monitorear Cola

Tabla 4.4.4 Tarea #1 de la historia de usuario Monitorear Cola

Tarea	
Número tarea: 1	Número historia: 8
Nombre tarea: Implementar la funcionalidad para monitorear cola.	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha inicio: 4 Marzo 2009	Fecha fin: 10 Marzo 2009
Programador responsable: Yusnel Cruz Castillo-Liorge Corria Sanchez.	
Descripción: Se implementa en el controlador la funcionalidad que va a permitir monitorear las colas que estén configurados en la base de datos del servidor Asterisk.	

Tabla 4.4.5 Tarea #2 de la historia de usuario Monitorear Cola

Tarea	
Número tarea: 2	Número historia: 8
Nombre tarea: Implementar la interfaz visual.	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha inicio: 10 Marzo 2009	Fecha fin: 16 Marzo 2009

Programador responsable: Yusnel Cruz Castillo-Liorge Corria Sanchez.
Descripción: Si implementa la interfaz visual mediante la cual se muestran los datos obtenidos.

4.5 Iteración 4

En el transcurso de esta iteración se implementaron las historias de usuario que permiten obtener reportes históricos de los puestos de trabajo y las colas. Al final de esta iteración se obtendrá la primera versión del software.

Tabla 4.5.1 Historias de usuario implementadas en la cuarta iteración

Historias de Usuario	Estimación	Real
Reporte Histórico de Puestos	2	2
Reporte Histórico de Colas	2	2

4.5.1 Tareas de las historias de usuario implementadas en la cuarta iteración

Reporte Histórico de Puestos

Tabla 4.5.2 Tarea #1 de la historia de usuario Reporte Histórico de Puestos

Tarea	
Número tarea: 1	Número historia: 9
Nombre tarea: Implementar la funcionalidad para el Reporte Histórico de Puestos.	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha inicio: 17 Marzo 2009	Fecha fin: 23 Marzo 2009
Programador responsable: Yusnel Cruz Castillo-Liorge Corria Sanchez.	

Descripción: Se implementa en el controlador la funcionalidad que va a permitir obtener los reportes históricos de los puestos de trabajo.

Tabla 4.5.3 Tarea #2 de la historia de usuario Reporte Histórico de Puestos

Tarea	
Número tarea: 2	Número historia: 9
Nombre tarea: Implementar la Interfaz visual	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha inicio: 23 Marzo 2009	Fecha fin: 29 Marzo 2009
Programador responsable: Yusnel Cruz Castillo-Liorge Corria Sanchez.	
Descripción: Se implementa la interfaz en la cual se muestran los reportes de los puestos de trabajo.	

Reporte Histórico de Colas

Tabla 4.5.4 Tarea #1 de la historia de usuario Reporte Histórico de Colas

Tarea	
Número tarea: 1	Número historia: 10
Nombre tarea: Implementar la funcionalidad para el Reporte Histórico de Colas.	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha inicio: 30 Marzo 2009	Fecha fin: 5 Abril 2009

Programador responsable: Yusnel Cruz Castillo-Liorge Corria Sanchez.
Descripción: Se implementa en el controlador la funcionalidad para obtener los reportes de cola de llamadas en un rango de tiempo.

Tabla 4.5.5 Tarea #2 de la historia de usuario Reporte Histórico de Colas

Tarea	
Número tarea: 2	Número historia: 10
Nombre tarea: Implementar la interfaz visual	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha inicio: 5 Abril 2009	Fecha fin: 11 Abril 2009
Programador responsable: Yusnel Cruz Castillo-Liorge Corria Sanchez.	
Descripción: Se implementa la interfaz en la cual se muestran los reportes históricos de las colas.	

4.6 Diagramas de Clase

“Para el diseño de las aplicaciones, la metodología XP no requiere la representación del sistema mediante diagramas de clases utilizando notación UML, en su lugar se usan otras técnicas como las tarjetas CRC. No obstante el uso de estos puede aplicarse siempre y cuando mejore la comunicación, no sea un peso su mantenimiento, no sean extensos y se enfoquen en la información importante”. [21]

4.7 Pruebas

Las pruebas dan la oportunidad de saber si lo implementado es lo que en realidad se tenía en mente. Ellas indican que el trabajo funciona, cuando no exista ninguna prueba que pudiese originar un fallo en el sistema, entonces se habrá terminado por

completo.

”Uno de los pilares fundamentales de XP es el proceso de pruebas. Mediante este método se reduce el número de errores no detectados así como el tiempo entre la introducción de este en el sistema y su detección”. [22]. Todo esto tiende a elevar la calidad de los productos desarrollados y la seguridad de los programadores a la hora de introducir cambios o modificaciones.

“XP divide las pruebas en dos grupos: pruebas unitarias, desarrolladas por los programadores y encargadas de verificar el código de forma automática y las pruebas de aceptación, destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida además de comprobar que dicha funcionalidad sea la esperada por el cliente”. [22]

4.8 Pruebas de Aceptación

Las pruebas de aceptación son pruebas de caja negra que se crean a partir de las historias de usuario [22]. Durante las iteraciones las historias de usuario seleccionadas serán traducidas a pruebas de aceptación. En ellas se especifican, desde la perspectiva del cliente, los escenarios para probar que una historia de usuario ha sido implementada correctamente. “Una historia de usuario puede tener todas las pruebas de aceptación que necesite para asegurar su correcto funcionamiento. El objetivo final de estas es garantizar que los requerimientos han sido cumplidos y que el sistema es aceptable”. [20]

Tabla 4.8.1 Prueba de aceptación para la historia de usuario Configurar Cola

Caso de Prueba de Aceptación	
Código: HU1_P1	Historia de usuario: Configurar Cola.
Nombre: Configurar la cola en la base de datos.	
Descripción: Probar que se escriban correctamente los parámetros en la tabla queue de la base de datos.	
Condiciones de ejecución: La aplicación debe ser ejecutada normalmente,	

insertando los datos introducidos en la tabla correspondiente de la BD.
Entrada / Pasos de ejecución: Se introducen parámetros válidos.
Resultado esperado: Los parámetros son insertados correctamente en la tabla queue.
Evaluación de la prueba: Satisfactoria.

Tabla 4.8.2 Prueba de aceptación para la historia de usuario Configurar Cola

Caso de Prueba de Aceptación	
Código: HU1_P2	Historia de usuario: Configurar Cola.
Nombre: Configurar cola con parámetros no válidos.	
Descripción: Probar que no se escriban los parámetros en la tabla.	
Condiciones de ejecución: La aplicación debe ser ejecutada normalmente, sin escribir los datos introducidos en la tabla de la base de datos.	
Entrada / Pasos de ejecución: Se introducen datos no válidos.	
Resultado esperado: El dato no es reconocido y no se escriben los campos correspondientes de la tabla, y se muestra un mensaje de error.	
Evaluación de la prueba: Satisfactoria.	

Tabla 4.8.3 Prueba de aceptación para la historia de usuario Configurar Agente

Caso de Prueba de Aceptación	
Código: HU2_P1	Historia de usuario: Configurar Agente.
Nombre: Configurar el agente en la base de datos.	
Descripción: Probar que se escriban correctamente los parámetros en la tabla agente de la base de datos.	
Condiciones de ejecución: La aplicación debe ser ejecutada normalmente, insertando los datos introducidos en la tabla correspondiente de la base de datos.	
Entrada / Pasos de ejecución: Se introducen parámetros válidos.	
Resultado esperado: Los parámetros son insertados correctamente en la tabla agente.	
Evaluación de la prueba: Satisfactoria.	

Tabla 4.8.4 Prueba de aceptación para la historia de usuario Configurar Agente

Caso de Prueba de Aceptación	
Código: HU2_P2	Historia de usuario: Configurar Agente.
Nombre: Configurar agente con parámetros no válidos.	
Descripción: Probar que no se escriban los parámetros en la tabla.	
Condiciones de ejecución: La aplicación debe ser ejecutada normalmente, sin escribir los datos introducidos en la tabla de la base de datos.	
Entrada / Pasos de ejecución: Se introducen datos no válidos.	

Resultado esperado: El dato no es reconocido y no se escriben los campos correspondientes de la tabla, y se muestra un mensaje de error.
Evaluación de la prueba: Satisfactoria.

Tabla 4.8.5 Prueba de aceptación para la historia de usuario Asignar Agente a Cola

Caso de Prueba de Aceptación	
Código: HU3_P1	Historia de usuario: Asignar Agente a Cola.
Nombre: Asignar un agente a la cola correctamente.	
Descripción: Probar que el agente se le asigna a la cola correctamente.	
Condiciones de ejecución: La aplicación debe ser ejecutada normalmente, asignando el agente a la cola.	
Entrada / Pasos de ejecución: se escoge agente y cola, ambos válidos.	
Resultado esperado: El agente es asignado a la cola satisfactoriamente.	
Evaluación de la prueba: Satisfactoria.	

Tabla 4.8.6 Prueba de aceptación para la historia de usuario Asignar Agente a Cola

Caso de Prueba de Aceptación	
Código: HU3_P2	Historia de usuario: Asignar Agente a Cola.

Nombre: Asignar agente no existente a la cola.
Descripción: Probar que el agente no se le asigna a la cola.
Condiciones de ejecución: La aplicación debe ser ejecutada normalmente, mostrando un mensaje de error.
Entrada / Pasos de ejecución: Se introduce un agente no válido y una cola válida.
Resultado esperado: El agente no se le asigna a la cola y se muestra un mensaje de error.
Evaluación de la prueba: Satisfactoria.

Tabla 4.8.7 Prueba de aceptación para la historia de usuario Eliminar Agente de Cola

Caso de Prueba de Aceptación	
Código: HU4_P1	Historia de usuario: Eliminar Agente de Cola.
Nombre: Eliminar un agente de una cola.	
Descripción: Probar que se elimina el agente de la cola correctamente.	
Condiciones de ejecución: La aplicación debe ser ejecutada normalmente, eliminando el agente de la cola.	
Entrada / Pasos de ejecución: Se intenta eliminar un agente de una cola.	
Resultado esperado: El agente se elimina de la cola satisfactoriamente.	
Evaluación de la prueba: Satisfactoria.	

Tabla 4.8.8 Prueba de aceptación para la historia de usuario Eliminar Agente de Cola

Caso de Prueba de Aceptación	
Código: HU4_P2	Historia de usuario: Eliminar Agente de Cola.
Nombre: Eliminar un agente no existente de una cola.	
Descripción: Probar que no se elimina el agente de la cola.	
Condiciones de ejecución: La aplicación debe ser ejecutada normalmente, mostrando un mensaje de error.	
Entrada / Pasos de ejecución: Se intenta eliminar un agente no existente de una cola.	
Resultado esperado: El agente no se elimina de la cola y se muestra un mensaje de error.	
Evaluación de la prueba: Satisfactoria.	

Tabla 4.8.9 Prueba de aceptación para la historia de usuario Eliminar Cola

Caso de Prueba de Aceptación	
Código: HU5_P1	Historia de usuario: Eliminar Cola.
Nombre: Eliminar una cola de la base de datos.	
Descripción: Probar que se elimina la cola correctamente.	
Condiciones de ejecución: La aplicación debe ser ejecutada normalmente, eliminando la cola escogida de la base de datos.	

Entrada / Pasos de ejecución: Se intenta eliminar una cola válida.
Resultado esperado: La cola se elimina satisfactoriamente.
Evaluación de la prueba: Satisfactoria.

Tabla 4.8.10 Prueba de aceptación para la historia de usuario Eliminar Cola

Caso de Prueba de Aceptación	
Código: HU5_P2	Historia de usuario: Eliminar Cola.
Nombre: Eliminar una Cola no válida.	
Descripción: Probar que no se elimine la cola.	
Condiciones de ejecución: La aplicación debe ser ejecutada normalmente, mostrando el mensaje de error ´La cola no existe´.	
Entrada / Pasos de ejecución: Se intenta eliminar una cola que no existe.	
Resultado esperado: La cola no es eliminada y se muestra un mensaje de error informando que la cola no existe en la base de datos.	
Evaluación de la prueba: Satisfactoria.	

Tabla 4.8.11 Prueba de aceptación para la historia de usuario Eliminar Agente

Caso de Prueba de Aceptación	
Código: HU6_P1	Historia de usuario: Eliminar Agente.
Nombre: Eliminar un agente de la base de datos.	

Descripción: Probar que se elimina el agente correctamente.
Condiciones de ejecución: La aplicación debe ser ejecutada normalmente, eliminando el agente escogido de la base de datos.
Entrada / Pasos de ejecución: Se intenta eliminar un agente válido.
Resultado esperado: El agente se elimina satisfactoriamente.
Evaluación de la prueba: Satisfactoria.

Tabla 4.8.12 Prueba de aceptación para la historia de usuario Eliminar Agente

Caso de Prueba de Aceptación	
Código: HU6_P2	Historia de usuario: Eliminar Agente.
Nombre: Eliminar un agente no válido.	
Descripción: Probar que no se elimine el agente.	
Condiciones de ejecución: La aplicación debe ser ejecutada normalmente, mostrando el mensaje de error ´El agente no existe´.	
Entrada / Pasos de ejecución: Se intenta eliminar un agente que no existe.	
Resultado esperado: El agente no es eliminado y se muestra un mensaje de error informando que el agente no existe en la base de datos.	
Evaluación de la prueba: Satisfactoria.	

Tabla 4.8.13 Prueba de aceptación para la historia de usuario Monitorear Agentes

Caso de Prueba de Aceptación	
Código: HU7_P1	Historia de usuario: Monitorear Agentes.
Nombre: Monitorear agentes en tiempo real.	
Descripción: Probar que se obtienen los datos de los agentes.	
Condiciones de ejecución: La aplicación debe ser ejecutada normalmente, mostrando los datos correspondientes de los agentes en tiempo real.	
Entrada / Pasos de ejecución: Se escoge la opción por la que desea monitorear los agentes en tiempo real.	
Resultado esperado: Se muestran los datos (nombre, estado, etc.) correspondientes a los agentes escogidos.	
Evaluación de la prueba: Satisfactoria.	

Tabla 4.8.14 Prueba de aceptación para la historia de usuario Monitorear Cola

Caso de Prueba de Aceptación	
Código: HU8_P1	Historia de usuario: Monitorear Cola.
Nombre: Monitorear cola en tiempo real.	
Descripción: Probar que se obtienen los datos de la cola escogida.	
Condiciones de ejecución: La aplicación debe ser ejecutada normalmente, mostrando los datos correspondientes de la cola escogida.	

Entrada / Pasos de ejecución: Se escoge la cola que desea monitorear.
Resultado esperado: Se muestran los datos en tiempo real(llamadas en cola, llamadas perdidas, llamadas perdidas, etc.) correspondientes a la cola escogida.
Evaluación de la prueba: Satisfactoria.

Tabla 4.8.15 Prueba de aceptación para la historia de usuario Monitorear Cola

Caso de Prueba de Aceptación	
Código: HU8_P2	Historia de usuario: Monitorear Cola.
Nombre: Monitorear cola no válida en tiempo real.	
Descripción: Probar que no se obtienen datos de la cola introducida.	
Condiciones de ejecución: La aplicación debe ser ejecutada normalmente, mostrando un mensaje de error.	
Entrada / Pasos de ejecución: Se intenta monitorear una cola que no existe.	
Resultado esperado: Se muestra un mensaje de error.	
Evaluación de la prueba: Satisfactoria.	

Tabla 4.8.16 Prueba de aceptación para la historia de usuario Reporte Histórico de Puestos

Caso de Prueba de Aceptación	
Código: HU9_P1	Historia de usuario: Reporte Histórico de Puestos.

Nombre: Obtener reportes históricos de los puestos.
Descripción: Probar que se obtienen los datos de los puestos.
Condiciones de ejecución: La aplicación debe ser ejecutada normalmente, mostrando los datos correspondientes de los puestos.
Entrada / Pasos de ejecución: Se escoge el rango de tiempo en el cual desea obtener los reportes.
Resultado esperado: Se muestran los datos (llamadas atendidas, promedio en llamadas, etc.)de los puestos en el rango de tiempo escogido.
Evaluación de la prueba: Satisfactoria.

Tabla 4.8.17 Prueba de aceptación para la historia de usuario Reporte Histórico de Puestos

Caso de Prueba de Aceptación	
Código: HU9_P2	Historia de usuario: Reporte Histórico de Puestos.
Nombre: Obtener reportes históricos de los puestos en un rango de tiempo no válido.	
Descripción: Probar que no se obtienen los datos de los puestos de trabajo.	
Condiciones de ejecución: La aplicación debe ser ejecutada normalmente, mostrando un mensaje de error.	
Entrada / Pasos de ejecución: Se escoge un rango de tiempo no válido para obtener los reportes de los puestos.	
Resultado esperado: Se muestra un mensaje de error y no se obtienen datos de los	

puestos.
Evaluación de la prueba: Satisfactoria.

Tabla 4.8.18 Prueba de aceptación para la historia de usuario Reporte Histórico de Colas

Caso de Prueba de Aceptación	
Código: HU10_P1	Historia de usuario: Reporte Histórico de Colas.
Nombre: Obtener reporte histórico de las colas.	
Descripción: Probar que se obtienen los datos deseados de las colas.	
Condiciones de ejecución: La aplicación debe ser ejecutada normalmente, mostrando los datos correspondientes de las colas.	
Entrada / Pasos de ejecución: Se escoge el rango de tiempo en el cual se deseas obtener los reportes de las colas.	
Resultado esperado: Se muestran los datos (llamadas perdidas, llamadas tomadas, porciento de atención, etc.) de las colas en el rango de tiempo escogido.	
Evaluación de la prueba: Satisfactoria.	

Tabla 4.8.19 Prueba de aceptación para la historia de usuario Reporte Histórico de Colas

Caso de Prueba de Aceptación	
Código: HU10_P2	Historia de usuario: Reporte Histórico de Colas.

Nombre: Obtener reportes históricos de las colas en un rango de tiempo no válido.
Descripción: Probar que no se obtienen los datos de las colas.
Condiciones de ejecución: La aplicación debe ser ejecutada normalmente, mostrando un mensaje de error.
Entrada / Pasos de ejecución: Se escoge un rango de tiempo no válido para obtener los reportes de las colas.
Resultado esperado: Se muestra un mensaje de error y no se obtienen datos de las colas.
Evaluación de la prueba: Satisfactoria.

4.9 Conclusiones

En el presente capítulo se tratan las etapas de implementación y pruebas del software en desarrollo. Para ello se exponen todos los artefactos generados, realizando una descripción de cada uno de ellos.

Conclusiones generales

Al finalizar la investigación llevada a cabo para el desarrollo del presente trabajo se consideran cumplidos los objetivos trazados, a este planteamiento se puede arribar a partir de las siguientes conclusiones:

- Se realizó un estudio previo que posibilitó la selección de la metodología, el lenguaje de programación y el entorno de desarrollo a utilizar en la elaboración del software.

- Se cumplió el objetivo por el cuál se llevó a cabo este trabajo. Desarrollar una herramienta que permite la configuración y supervisión del Distribuidor Automático de Llamadas para PLATEL.

- Los resultados obtenidos de las pruebas realizadas correspondieron con los esperados, demostrando de esta manera la concordancia entre lo exigido por el cliente y la solución automatizada.

Es necesario destacar que el proceso de desarrollo del sistema no supuso gastos de recursos ya que la infraestructura de producción estaba creada. Además, el soporte utilizado para la realización de la aplicación es totalmente libre por lo que no se incurrieron en gastos referentes al pago de licencias y se siguieron las políticas de software libre.

Recomendaciones

Luego de la culminación del presente trabajo y haber estado inmersos en el mundo de las telecomunicaciones y los centros de llamadas, se hace imprescindible recomendar algunos aspectos para seguir mejorando el mismo, los cuales son:

- Difundir el contenido del producto a todos lo estudiantes y profesores del proyecto Call Center en la UCI para que continúen con la investigación de nuevas tecnologías informáticas para así adicionar nuevas funcionalidades y garantizar mejoras en futuras versiones del Sistema.

- Permitir imprimir los reportes históricos de las colas y los puestos en diferentes formatos.

- Adicionar nuevos reportes a la herramienta que ayuden a mejorar los niveles de servicio de la plataforma PLATEL.

- Mejorar aún más el diseño de la herramienta.

Referencias Bibliográficas

[1] Auronix. "CalixtaACD", 2006. [Disponible en]:
http://www.estudiod3.com/Auronix2005a/CALIXTA/calixtaACD/QUEES.HTM

[2] Aspect Software. "Aspect® Spectrum® ACD", 2007. [Disponible en]:
http://apacconnect2006.com/content/1200Products/2300ACD/3200AspectSpectrumACD/Aspect-Spectrum-ACD-BrochureLA-Spanish_A4.pdf

[3] Aspect Software. "Aspect® CallCenter® ACD", 2008. [Disponible en]
http://www.aspect.com/brochures/Aspect_CallCenterACD_Brochure_A4_Spanish.pdf

[4] Ciberaula. “Ciberaula”, 2006. [Disponible en]:
http://linux.ciberaula.com/articulo/linux_apache_intro

[5] The Apache Software Foundation.” Documentación del Servidor HTTP Apache 2.0”, 2007. [Disponible en]: http://httpd.apache.org/docs/2.0/es

[6] Programación en la WEB. “Seminario web”. 2007. [Disponible en]:
http://seminarioproweb.blogspot.com/2007/11/servidor-thttpd.html

[7] Rosa, M. “Diseño de Bases de Datos”. 2002

[8] Ian, G. “La Biblia de MySQL”, Anaya Multimedia. 2005

[9] Elmasri y Navathe. “Fundamentos de Sistemas de Bases de Datos”, Addison Wesley. 2007

[10] Daniel, P. “PostGreSQL vs. MySQL”, 2006. [Disponible en]:
http://www.netpecos.org/docs/mysql_postgres/x15.html

[11] Jacobson, I. Booch, G. Rumbaugh, J. “El proceso unificado de Software”, Pearson Educación, S.A. 2000.

[12] Schwaber, K. Beedle, M. Martin, R. “Agile Software Development with SCRUM. Prentice Hall”. 2001

[13] vivab0rg. "Software", 2006. [Disponible en]:
http://www.vivalinux.com.ar/soft/easy-eclipse.html

[14] Javier, E.”Introducción a AJAX”. 2007

[15] Jerry, B. "Aplicaciones JavaScript", O'Reilly Media. 2006

[16] Guido, V. Fred L. "Guía de aprendizaje de Python". 2003

[17] Javier J. G. "¿Qué es un framework web?". 2008

[18] Ventajas e Inconvenientes de las aplicaciones Web. 2007. [Disponible en]: http://www.avidos.net/blogold/aplicaciones-web

[19] Beck, K. "Extreme Programming Explained". Addison Wesley. 2000.

[20] Poppendieck, M. Poppendieck, T. "Lean Software Development: An Agile Toolkit for Software Development Managers", Addison Wesley. 2003.

[21] Beck, K. Fowler, M. "Planeando en Programación Extrema". 2000.

[22] Crispin, L. House, T. "Probando la Programación Extrema", Addison Wesley. 2002.

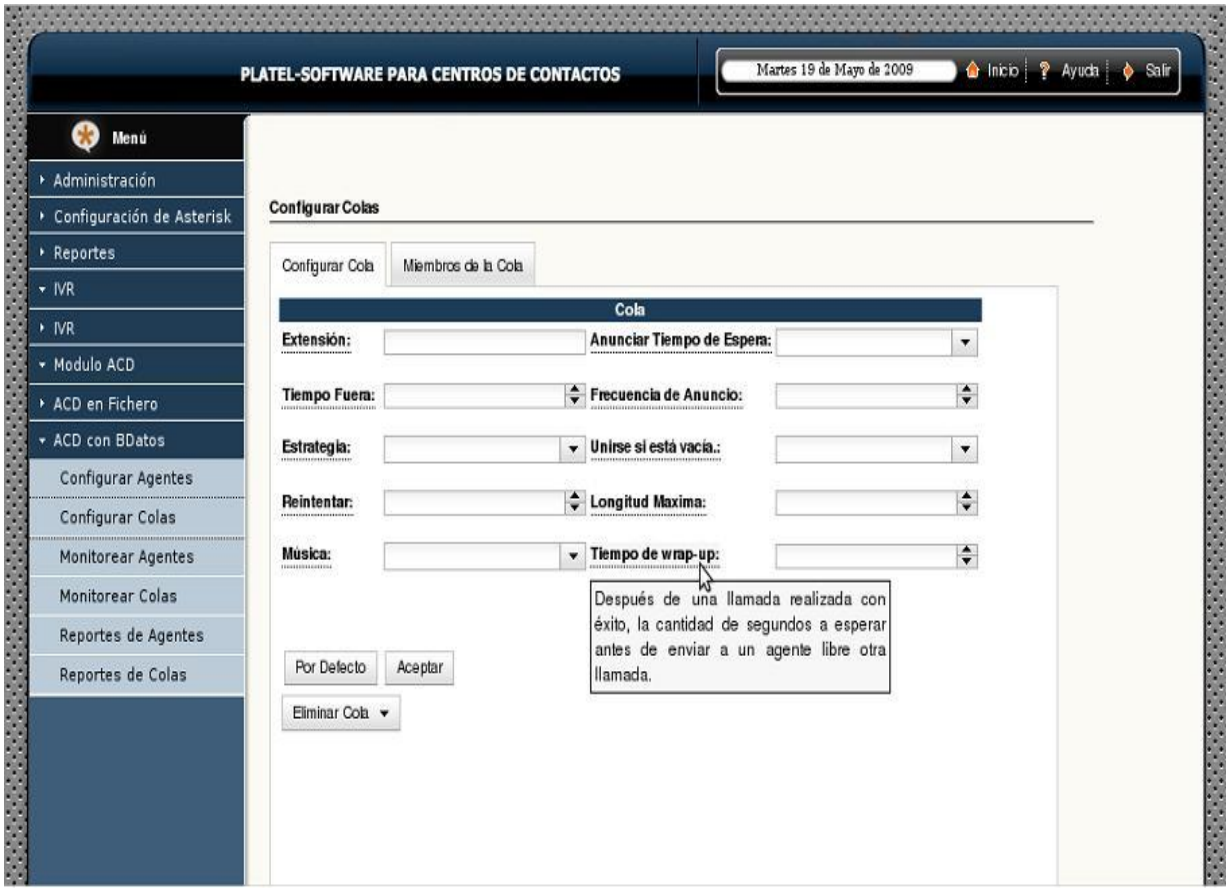
Bibliografía

- Van, M. Jim, M. Leif, S. “Asterisk The Future of Telephony”. Versión 2, Estados Unidos de América, O’Reilly Media. 2007
- Beck, K. “Extreme Programming Explained”. Estados Unidos de América, Addison Wesley. 2000
- Poppendieck, M. “Software Development: An Agile Toolkit for Software Development Managers”. Estados Unidos de América, Addison Wesley. 2003
- Theodore, W. “Switching to VoIP”. O'Reilly. 2002
- Schwaber, K. Beedle, M. Martin, R. “Agile Software Development with SCRUM”. Prentice Hall. 2001
- Sitio oficial de Python. “About Python”. 2009 [Disponible en]: http://www.python.org/about
- Guido, V. Fred L. “Guía de aprendizaje de Python”. 2003
- García, R. y Félix O. “Metodologías de desarrollo de software”. 2006
- Abrahamson, P. Salo, O. Ronkainen, J. Warsta, J. “Agile software development methods Review and analysis”. VTT Publications. 2002.
- Montejano G. “Metodologías de desarrollo de software ágiles”. 2006.
- Mousqués, G. “Desarrollo Ágil de Software”. 2004
- Ciberaula. “Ciberaula”. 2006. [Disponible en]: http://linux.ciberaula.com/articulo/linux_apache_intro
- The Apache Software Foundation. “Documentación del Servidor HTTP Apache 2.0”. 2007. [Disponible en]: http://httpd.apache.org/docs/2.0/es
- Página oficial de Asterisk. 2007. [Disponible en]: http://www.asterisk.org
- PostgreSQL. Tutorial de PostgreSQL. 2007. [Disponible en]: http://es.tldp.org/Postgresql-es/web/navegable/tutorial/tutorial.html

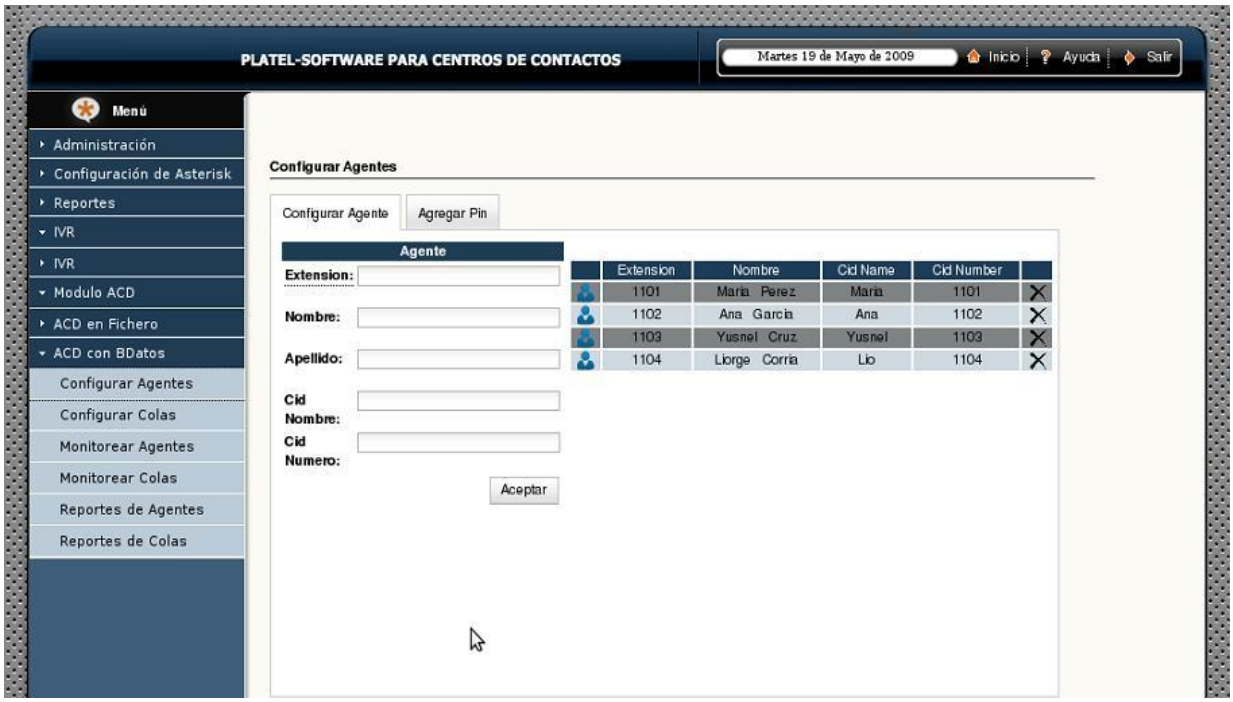
- Daniel, P. “PostgreSQL vs. MySQL”, 2006. [Disponible en]: http://www.netpecos.org/docs/mysql_postgres/x15.html
- Nicholas, C. “Professional JavaScript for Web Developers”. 2002
- Javier, E.”Introducción a AJAX”. 2007
- Jerry, B. "Aplicaciones JavaScript". Estados Unidos de América, O’Reilly Media. 2006
- Beck, K. Fowler, M. ” Planeando en Programación Extrema”. 2000
- Allen, D. Jeffrey, E. Chris, M. “Aprenda a Pensar Como un Programador con Python “.Primera Edición, Estados Unidos de América. 2002
- Gomillon, L. David, D. “Build Telephony System with Asterisk“, Primera Edición, Estados Unidos de América. 2005
- Flavio E. “Asterisk PBX. Guía de configuración”. Primera Edición, Brasil. 2007
- Jacobson, I. Booch, G. Rumbaugh, J. “El proceso unificado de Software”, Pearson Educación. 2000
- Jerry, B. "Aplicaciones JavaScript". Estados Unidos de América, O’Reilly Media. 2006
- Ian, G. “La Biblia de MySQL”, Anaya Multimedia. 2005
- Elmasri y Navathe. “Fundamentos de Sistemas de Bases de Datos”. Estados Unidos de América, Addison Wesley. 2007
- Crispin, L. House, T. “Probando la Programación Extrema”, Estados Unidos de América, Addison Wesley. 2002
- Marino, P. “Introducción al lenguaje XML”. Primera Edición, Grupo Eidos. 2005
- Ventajas e Inconvenientes de las aplicaciones Web. 2007. [Disponible en]: http://www.avidos.net/blogold/aplicaciones-web
- Sitio oficial de Calixta ACD. [Disponible en]: http://www.Calixta.com
- Sitio oficial de TurboGears. [Disponible en]: http://turbogears.org

Anexos

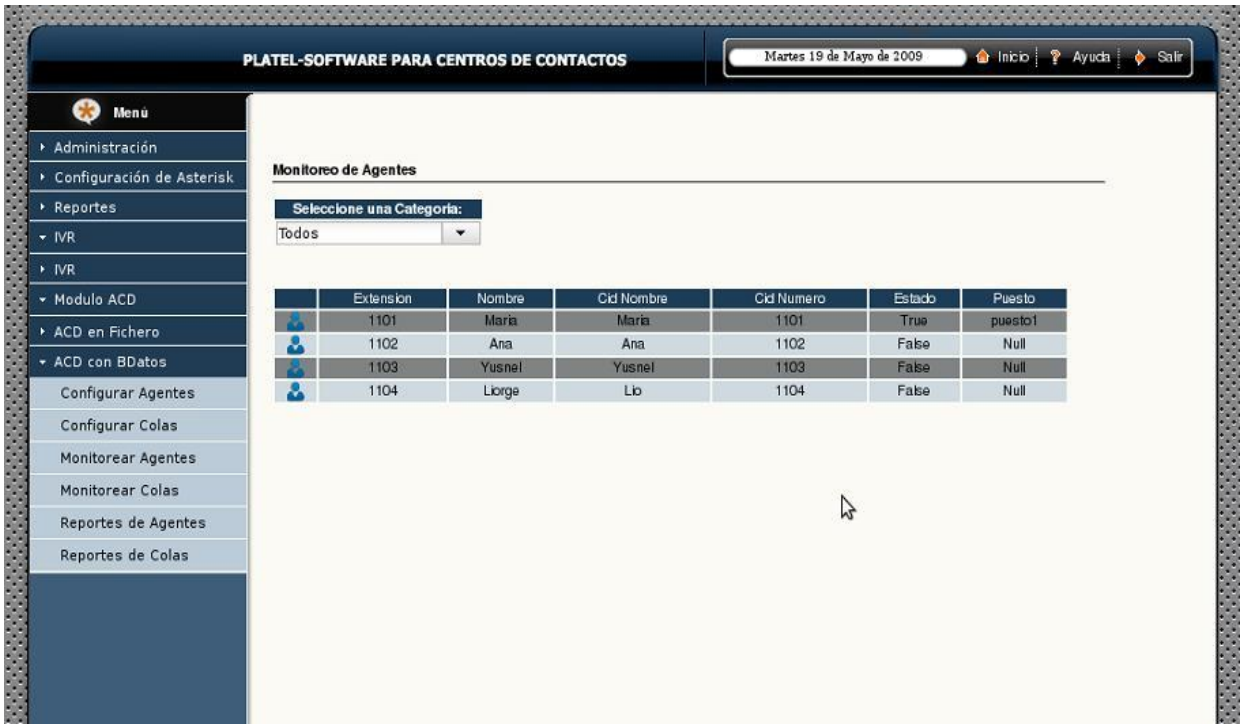
Anexo #1: Interfaces de Usuarios



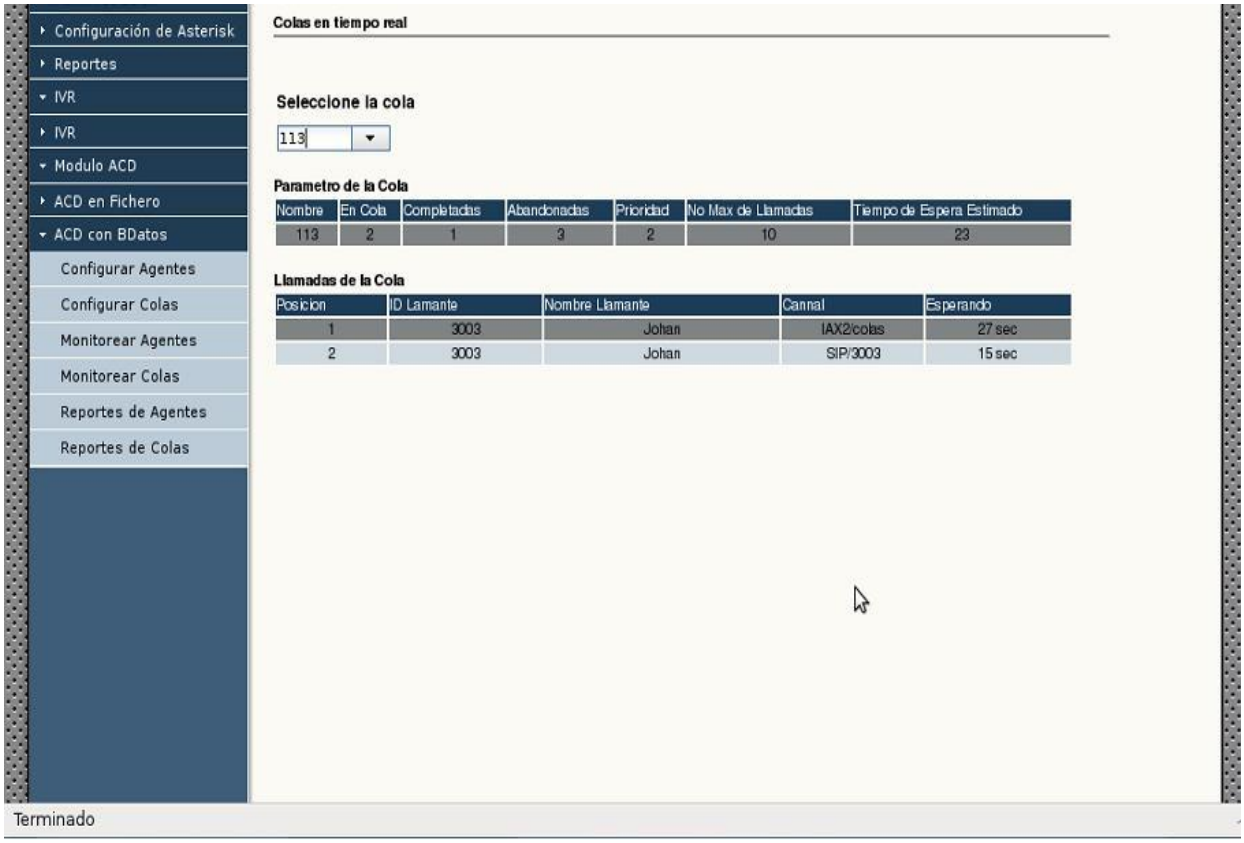
Anexo I Figura 2.2 Interfaz de usuario Configurar Colas



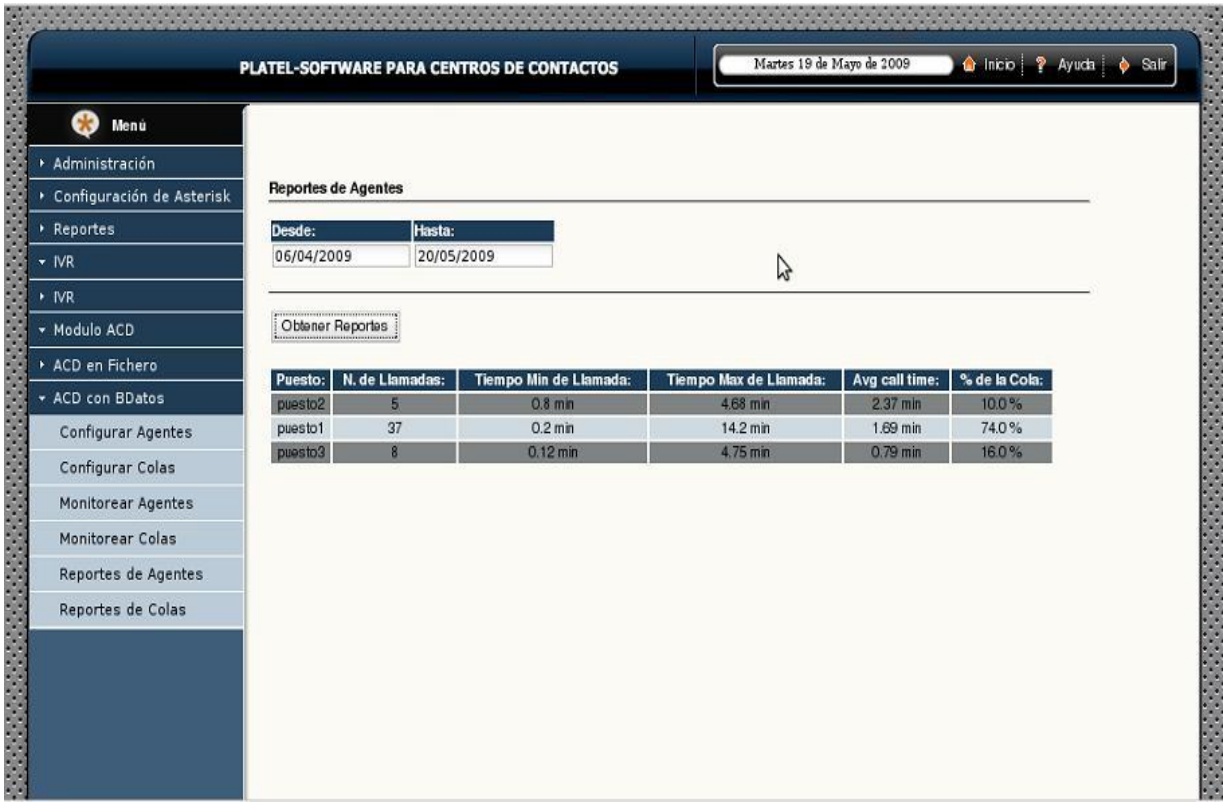
Anexo I Figura 2.3 Interfaz de usuario Configurar Agentes



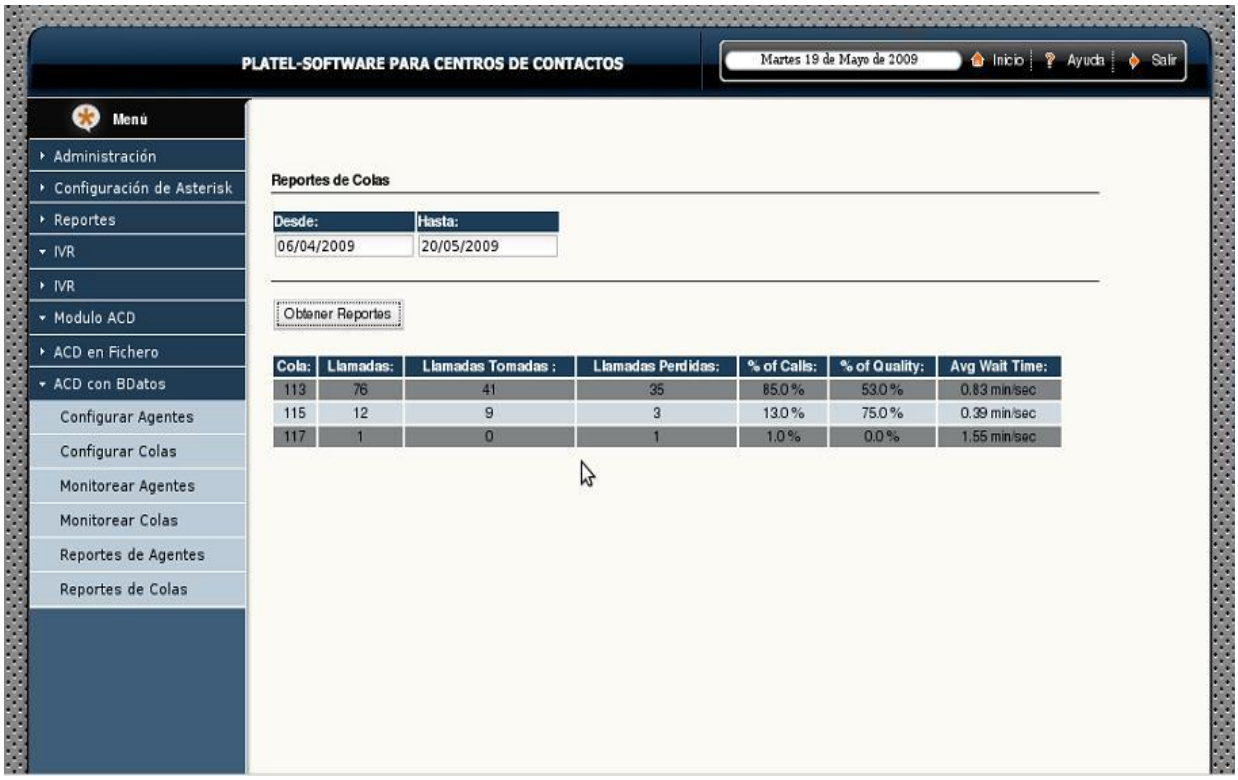
Anexo I Figura 2.4 Interfaz de usuario Monitorear Agentes



Anexo I Figura 2.5 Interfaz de usuario Monitorear Colas



Anexo I Figura 2.6 Interfaz de usuario Reporte de Agentes



Anexo I Figura 2.7 Interfaz de usuario Reporte de Colas

Anexo #2 Arquitectura Centralizada

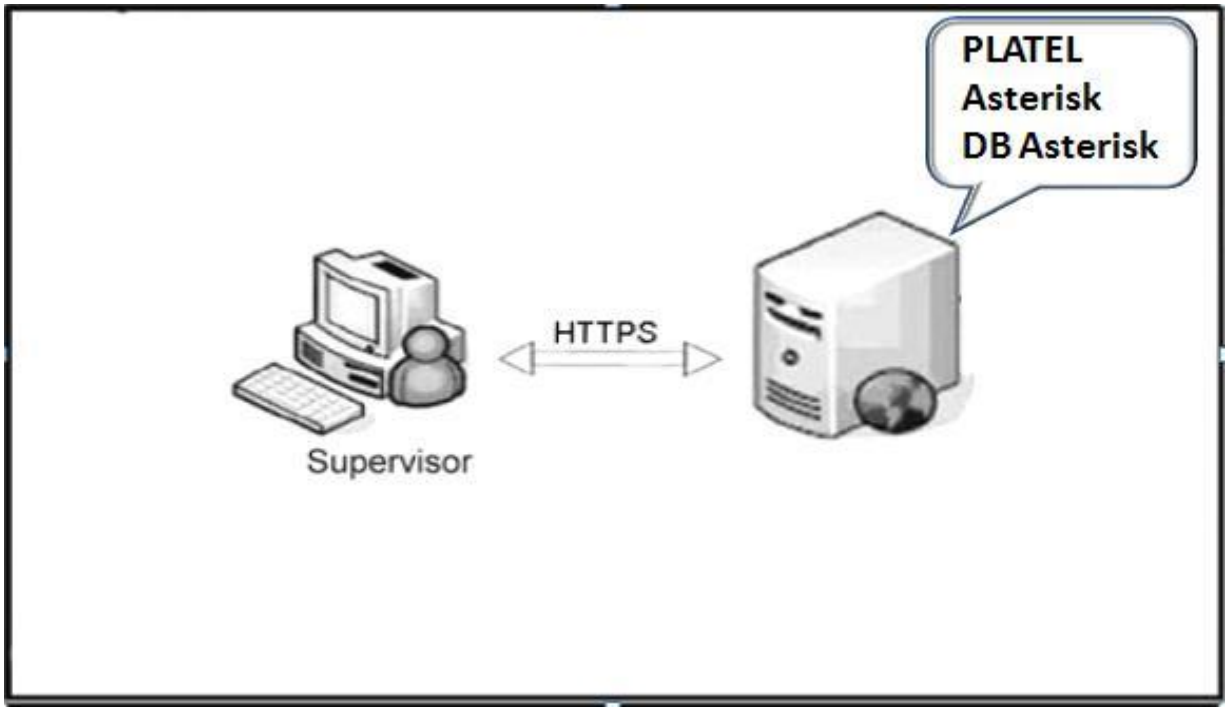


Figura 2.8 Arquitectura Centralizada

Glosario de términos

ACD: Distribuidor Automático de Llamadas o ACD (Automatic Call Distribution por sus siglas en inglés) es la tecnología de Centro de Llamadas que permite distribuir de forma eficiente las llamadas que entran al mismo.

Asterisk: Es una aplicación de software libre (bajo licencia GPL) que proporciona funcionalidades de una central telefónica (PBX). Como cualquier PBX, se puede conectar un número determinado de teléfonos para hacer llamadas entre sí e incluso conectar a un proveedor de VoIP o bien a una tarjeta RDSI tanto básicos como primarios.

Agentes (Operadores): Aquellas personas que contestan las llamadas en un Centro de Llamadas. Se encargan no sólo de contestar las llamadas, también tienen la capacidad de asesorar y atender cualquier inquietud de los usuarios.

APIs: Es el conjunto de funciones y procedimientos (o métodos, si se refiere a programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

AJAX: Acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas.

Centro de Llamadas: Es un sistema integrado de telefonía y computación orientado a potenciar las 3 labores más importantes de una empresa, por medio de una comunicación telefónica las cuales son, la adquisición de clientes, el mantenimiento de clientes y el cobro a través del sistema telefónico.

Framework: Es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, un framework puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

GPL: La GNU General Public License (inglés: Licencia Pública General) es una licencia creada por la Free Software Foundation y orientada principalmente a los términos de distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es Software Libre.

IDE: Entorno de desarrollo integrado o en inglés Integrated Development Environment. Es un programa compuesto por un conjunto de herramientas para un programador desde el que se pueden editar programas, compilarlos y depurarlos.

Metodologías de Desarrollo: Se define como un conjunto de filosofías, etapas, procedimientos, reglas, técnicas, herramientas, documentación y aspectos de formación para los desarrolladores de sistemas de información.

MVC: De sus siglas en inglés Model View Controller, significa Modelo Vista Controlador, es un patrón de arquitectura de software que permite realizar la programación multicapa, separando en una aplicación los datos, la interfaz del usuario y la lógica.

Open Source: Código abierto (Open Source por sus siglas en inglés) es el término con el que se conoce al software distribuido y desarrollado libremente. Free en inglés puede tener diferentes significados: gratuidad y libertad. Por ello, por un lado, permite pensar en "software por el que no hay que pagar" (software gratuito) y por otro, se adapta al significado que se pretendió originalmente (software que posee ciertas libertades).

PBX (Private Branch Exchange): Centralita Telefónica Privadas Centralita de tamaño particular (para casas u oficinas) que interconecta las extensiones internas de los teléfonos y proporciona conexión con la red telefónica exterior.

Protocolo: Conjunto de normas que rigen un determinado proceso de comunicación.

SoftPBX: PBX basado en Software; esta aplicación realiza las mismas funcionalidades de una PBX convencional.

VoIP: Voice Over Internet Protocol, Voz sobre Protocolo de Internet. Es un grupo de recursos que hacen posible que la señal de voz viaje a través de Internet empleando un protocolo IP (Internet Protocol).

Widgets: Pequeña aplicación o programa, usualmente presentado en archivos o ficheros pequeños que son ejecutados por un motor de widgets o Widget Engine.

XP: Extreme Programming, Metodología de desarrollo de software basada en valores como simplicidad, comunicación, retroalimentación y coraje. Es una disciplina de desarrollo de software que persigue simplificar los procesos de desarrollo. Fue diseñada para ser usada con equipos de desarrollo pequeños que necesiten desarrollos ágiles y con requerimientos cambiantes.