Universidad de las Ciencias Informáticas

Facultad 2

Servicio de Facturación para Elastix

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autores: Danae Pérez Arias.

Rainer Segura Peña.

Tutor: Ing. Félix Maikel García Pérez.

Ciudad de la Habana, Junio 2011

…todos los días hay que luchar porque ese amor a la humanidad viviente se transforme en hechos concretos, en actos que sirvan de ejemplo, de movilización.

Che

Declaración de Autoría

Declaramos ser los autores de la presente tesis, reconociendo a la Universidad de las Ciencias Informáticas los derechos patrimoniales de manera exclusiva.

Para que así conste, firmamos la presente a los ___ días del mes de _____ de _____

_____ _____

Danae Pérez Arias

Firma Autor

Rainer Segura Peña

Firma Autor

_____

Firma del Tutor

Ing. Félix Maikel García Pérez

Dedicatoria

Danae

Dedico esta tesis a mi mamá, a mis abuelos Félix y Georgina y a mi bisabuela materna que aunque ya no está físicamente supo siempre darme su cariño incondicional, su dedicación a mi persona desde pequeñita y todo su amor de abuela. A ellos va esta dedicatoria, porque sin su ayuda no hubiese podido alcanzar mis objetivos y lograr ser la persona que soy hoy.

Rainer

Dedico esta tesis a toda mi familia.

A mis padres, por su comprensión y ayuda en momentos malos y buenos. Me han enseñado a encarar las adversidades sin perder nunca la dignidad. Me han dado todo lo que soy como persona, mis valores, mis principios, mi perseverancia y mi empeño, y todo ello con una gran dosis de amor y sin pedir nunca nada a cambio.

A mi hermano Reinier, a mis abuelos, mis tíos y tías, mis primos y a la familia de mi novia que siempre estuvieron pendientes de mis resultados y me han dado su apoyo incondicional.

A Mamaíta y el Chino por ser mis amigos y padres en Venezuela y aquí en Cuba.

A Alexis Chirino por su ayuda incondicional en Venezuela y ser como un padre en todo el tiempo que estuve allí.

A mi novia Arletti, a ella especialmente le dedico esta Tesis. Por su paciencia, por su comprensión, por su empeño, por su fuerza, por su amor, por ser tal y como es,... porque la amo. Es la persona que más directamente ha sufrido las consecuencias del trabajo realizado. Nunca le podré estar suficientemente agradecido.

Agradecimientos

Danae

A mi mamá, por ser mi amiga, por apoyarme, por ayudarme a levantarme en cada momento, por su confianza y fe en mí.

A mis abuelos maternos Georgina y Félix, por guiarme, cuidarme y ser mi ejemplo a seguir.

A mi abuelita paterna Milagro por preocuparse siempre por mí.

A mi prima Maylen por ser más que mi prima, mi hermana de toda la vida.

A mis tíos y primos por su apoyo y cariño.

A Indira por cuidarme como a una hermana.

A mis amistades del pre Maybel, Diamelys, Cecilia, Yoa, Martica, Damilé y Denny, por demostrarme cada día que puedo contar con ellos.

A Yadier por ser mi mejor amigo, por estar a mi lado incondicionalmente estos 5 años, por su ayuda, sus consejos y sus regaños; gracias por existir.

A Diane por ser mi fiel confidente, por apoyarme.

A Puchito por hacerme reír incluso los días en que estaba triste.

A Cary, Malena, Maipú y el Flaquito por convertirse en mi familia de aquí de la Universidad.

A mi compañero de tesis Rainer por ser un excelente amigo, por su ayuda.

A la profesora Aymee por ser mi otra madre aquí en la escuela, por sus consejos y críticas que me han servido de mucho.

A mi tutor Félix por su paciencia, ayuda y los conocimientos que me transmitió.

A Arian porque aunque estemos lejos sus consejos me han servido de mucho y se que puedo contar con él.

A todas aquellas personas que he conocido y con las que he compartido estos 5 años de universidad, aquellas que siguieron y otras que no están pero que nunca olvidaré, especialmente aquellas de mi grupo inolvidable de 1er año (2101) y de 3er año(2302).

Agradecimientos

A los profesores que de una forma u otra se relacionaron conmigo y me ayudaron con sus conocimientos a lograr mis sueños. A toda la Universidad y Facultad 2 por forjarme y educarme como una verdadera profesional.

Rainer

Quiero agradecer a toda mi familia, en especial a mis padres, mi hermano, mis abuelos, mis tías, tíos, primos, a mi novia y su familia.

A mi compañera de Tesis de Danae por ser más que compañera de tesis por ser amiga.

A mi amigo Chino y su esposa Mamaíta por ser como mis padres. A Alexis Chirino.

A mi tutor Felix.

A la profesora de física Aymee, a Osmany y Yanetsy.

A mis compañeros y hermanos de estudio durante estos 5 años Husseyn, Yamil, Julio, Puchi, Jorge, Alexei, René, Aballe.

A todos mis compañeros de proyecto, y a todos los que ayudaron en este trabajo.

Resumen

La automatización de reportes y facturas es un punto esencial para todo servicio que se brinda a través de una planta telefónica. Si se hace todo este proceso de forma visual a través de una plataforma web se garantiza un mejor manejo y funcionamiento de la planta. Elastix, como distribución de software libre orientada a las comunicaciones unificadas integra una interfaz web sencilla y fácil de usar que gestiona gran parte de los servicios de Asterisk, cuenta con un sistema de facturación que calcula el costo de las llamadas externas-salientes que se realizan hacia una red pública, pero todo este proceso no es llevado a cabo en un tiempo real. El presente trabajo permite la creación de un Servicio de Facturación Telefónica en tiempo real, obteniendo como resultado un grupo de funcionalidades que harán más completo el funcionamiento del Elastix en cuanto a configuración de las tarifas telefónicas y reporte de los costos por concepto de las llamadas externas-salientes que se realizan hacia una red pública a través de Asterisk.

Palabras Claves

Facturación, Red Pública, Llamadas, Reportes, Planta Telefónica.

INTRODUCCIÓN

La Era de las Telecomunicaciones ha traído grandes cambios en la sociedad. Actualmente existen diversas formas de establecer una comunicación desde cualquier parte del mundo y no es limitante cuán lejos se encuentra algo o alguien para obtener información.

En estos momentos, cuando el avance de las tecnologías se hace mayor, el desarrollo de las comunicaciones crece proporcionalmente junto al alcance que estas pueden tener, siendo parte casi indispensable de las actividades cotidianas del ser humano.

El auge de las tecnologías ha ido más allá de establecer una comunicación entre dos personas, posibilitando diversas funcionalidades que han venido surgiendo a partir de la creación del primer dispositivo capaz de permitir el establecimiento de una conversación. Desde la creación de este dispositivo conocido como teléfono, la rama de la telefonía ha representado un papel importante en cualquier sociedad. Surge entonces en el mundo una gran demanda de servicios telefónicos, creándose la necesidad de gestionar las llamadas que se realizaban de un lugar a otro, y es en este entorno que nacen las conocidas plantas telefónicas.

Estas plantas telefónicas juegan un importante papel dentro de empresas o instituciones, independientemente del alto precio de adquisición y soporte que tienen. Una alternativa para mejorar en este aspecto es optar por aquellas plantas telefónicas sobre software libre, por la gran demanda que tienen a nivel mundial y el grupo de funcionalidades que brindan.

Dentro de estos sistemas basados en software libre podemos encontrar a Asterisk, como el más usado y reconocido internacionalmente; basado en soluciones de telefonía, ofrece variadas y flexibles características que se esperan de una PBX (Private Branch Exchanche o Centralita Telefónica Privada) y más. Muchas de sus funcionalidades pueden ser gestionadas a través de diversas aplicaciones, una de las distribuciones de GNU/Linux[1] que integra una interfaz web para gestionar Asterisk es Elastix que

[1] GNU/Linux es uno de los términos empleados para referirse a la combinación del núcleo o *kernel* libre similar a Unix denominado Linux.

actualmente cuenta con una gran comunidad a nivel mundial, por ser la solución que integra todos los medios y alternativas de comunicación que existen en el ámbito empresarial.

Cuba lleva a cabo una revolución educacional y de salud, también decide volcarse hacia la informatización de su sociedad en cuanto a las telecomunicaciones y tecnologías basadas en software libre.

La Universidad de las Ciencias Informáticas (UCI), como centro vanguardia en la rama de la informática se ha convertido en una potente empresa productora de software. Entre sus objetivos figura el de fortalecer la economía del país y para lograrlo, se ha dividido en estructuras productivas especializadas en diversos temas.

El centro de telemática (TLM), forma parte de una de las estructuras productivas en la que se ha dividido la UCI, no solo se dedica al desarrollo de aplicaciones relacionadas con las telecomunicaciones, además, ofrece el servicio de montaje de CallCenter (Centro de Llamadas) en la hermana República Bolivariana de Venezuela y en diversos proyectos dentro de la misma universidad.

Entre las aplicaciones de alternativa libre que figuran en un CallCenter se encuentra Asterisk, diseñado principalmente para entornos corporativos. Actualmente los productos más vendidos en todo el mundo son basados en servicios, específicamente servicios integrados, donde el software cumple todas las funcionalidades que desean los clientes. Elastix, como distribución de software libre orientada a las comunicaciones unificadas es una de las soluciones que más desarrollo tiene, que lleva la tecnología de punta en cuanto a integración de servicios tales como correo de voz, correo electrónico, fax, soporte para VoIP[2], mensajería instantánea y video-llamadas en su versión 2.0. Dentro de los servicios que gestiona Elastix se encuentra un sistema de facturación, el mismo permite a una empresa que posea varios teléfonos en sus distintos departamentos u oficinas, calcular el valor monetario que los usuarios deben abonar por concepto de las llamadas telefónicas que se realizan. Para el sistema de facturación de Elastix las llamadas realizadas internamente no presentan coste alguno, sin embargo, este sistema permite calcular el costo de las llamadas realizadas a los teléfonos de redes externas a la entidad. Actualmente la facturación que se realiza no cubre las necesidades de una entidad que quiera realizar el proceso de facturación en tiempo real, además de que no permite crear múltiples tarifas telefónicas sobre una misma área, y los costos de las llamadas no son registrados en una fuente de datos. Sumado a esto, el sistema

[2] Término usado en telefonía IP para definir los servicios que se usan para transmitir voz usando el protocolo IP.

de facturación solamente permite gestionar las tarifas utilizando SQLite, de esta manera, se imposibilita el uso de distintas fuentes de datos.

A partir de la situación problémica anteriormente planteada, se deriva el siguiente **Problema a resolver:** ¿Cómo gestionar en tiempo real la facturación de llamadas telefónicas realizadas a entidades externas, a través de Elastix?

Con el fin de solucionar el problema a resolver asumido, el **Objeto de Estudio** del presente trabajo son los procesos de facturación de llamadas telefónicas.

Teniendo en cuenta el problema a resolver anterior se define como el **Objetivo General**: Desarrollar un Servicio de Facturación integrado a Elastix que permita calcular en un tiempo real los costos de las llamadas telefónicas realizadas a entidades externas.

Se enmarca como **Campo de Acción** el proceso de facturación integrado a Elastix.

Y para cumplimentar el objetivo general se definen los siguientes **Objetivos Específicos:**

- ✓ Identificar los procesos que intervienen en la facturación de llamadas telefónicas.
- ✓ Diseñar un Servicio de Facturación Telefónica integrado a Elastix.
- ✓ Implementar un Servicio de Facturación Telefónica integrado a Elastix.
- ✓ Realizar pruebas al Servicio de Facturación.

La **Idea a Defender** plantea que si se desarrolla un Servicio de Facturación integrado a Elastix se permite calcular en un tiempo real los costos de las llamadas telefónicas realizadas a entidades externas.

Para dar cumplimiento al objetivo propuesto se definen las siguientes **Tareas de Investigación:**

1. Análisis del sistema Asterisk para identificar los mecanismos de obtención de los datos de las llamadas telefónicas realizadas a través del mismo.
2. Estudio del sistema de gestión Elastix para el correcto desarrollo de las nuevas funcionalidades vinculadas a él.
3. Identificación de los diferentes procesos que intervienen en la facturación de llamadas telefónicas para un correcto diseño e implementación de los mismos.

4. Comparación y selección de la tecnología, metodología y herramientas que intervienen en el desarrollo de aplicaciones para plantas telefónicas a emplear en el desarrollo del servicio integrado a Elastix.
5. Implementación de cada una de las funcionalidades correspondientes al Servicio de Facturación telefónica integrado a Elastix.
6. Realización de pruebas a cada una de las funcionalidades correspondientes al Servicio de Facturación telefónica integrado a Elastix.

Para apoyar el desarrollo de la investigación se emplean los siguientes métodos científicos:

- **Métodos Teóricos:**

Analítico sintético: Este método permitirá analizar las teorías y los documentos referentes al objetivo de la investigación, facilitando de esta forma la extracción de los elementos más importantes relacionados con el objeto de estudio. Además de que posibilitará construir el camino a seguir, a partir del análisis detallado de cada uno de los documentos previamente mencionados.

Histórico-Lógico: Este método permite estudiar todo lo relacionado con los sistemas de facturación telefónicos y plantas telefónicas, para así obtener un conocimiento histórico de su desarrollo y comportamiento tanto a nivel internacional como nacional.

Modelación: En el desarrollo se crean reproducciones simplificadas de la realidad, por ejemplo, todos los modelos y diagramas presentados. Estos permitirán una reproducción ampliada de la realidad, además de que posibilitará descubrir y estudiar nuevas relaciones y cualidades del objeto de estudio.

- **Métodos Empíricos:**

Entrevista: Este método se utilizará para la realización del sistema pues que para obtener el servicio con la calidad que requiere se realizarán una serie de entrevistas con el cliente, y sobre la base de estas se trabajará para satisfacer sus necesidades. De esta manera, se realizaron varias entrevistas con el cliente para definir los principales problemas existentes a los que se les daría solución posteriormente, identificando los procesos que intervienen en la facturación de llamadas telefónicas. Además, se entrevistó a varias personas especializadas en el trabajo con Asterisk y Elastix definiendo como se relacionaban estos dos softwares.

El presente trabajo de diploma consta de 4 capítulos donde se describe todo el proceso de investigación.

Capítulo 1. Fundamentación Teórica: describe los principales conceptos a tratar: Elastix, Facturación de llamadas telefónicas y Asterisk como software que proporciona las funcionalidades de una planta telefónica. Se definirá la metodología de desarrollo, herramientas y tecnologías a utilizar para la aplicación. También los procesos que definen los sistemas de facturación.

Capítulo 2. Exploración y Propuesta del Sistema: en este capítulo es donde se caracteriza el sistema: se plantean a grandes rasgos las Historias de Usuarios (HU) que son de interés para la primera entrega del producto así como una propuesta del prototipo no funcional de la aplicación.

Capítulo 3. Diseño, Implementación y Prueba: según la metodología de desarrollo utilizada, este capítulo se refiere a las propias fases de la misma, definiendo la arquitectura del sistema y con ello patrones arquitectónicos empleados. Se procederá a abordar las HU definidas y tareas de la ingeniería en cada iteración para luego hacer una revisión de las mismas.

Capítulo 4. Estudio de la factibilidad: este capítulo se encarga de estudiar la factibilidad del sistema.

Capítulo 1. Fundamentación Teórica

1 Introducción

En este capítulo se profundizan los conceptos fundamentales para el desarrollo del Servicio de Facturación para Elastix. Se realiza un estudio sobre los sistemas de facturación utilizados por empresas líderes en el mundo y se exponen un conjunto de características de las tecnologías, herramientas y metodología utilizada.

1.2 Conceptos a tratar

1.2.1 Llamada

Una llamada es una señal o aviso que se realiza. Permite hacer referencia a invocar, convocar, citar o nombrar a alguien o algo.

1.2.2 Llamadas Telefónicas

En términos de telefonía, una llamada telefónica es la acción de emitir una señal o aviso por medio de un teléfono.

Existen diferentes tipos de llamadas telefónicas:

1.2.2.1 Llamadas Internas

Son aquellas llamadas que se realizan dentro de una determinada área que no requieren de costo alguno.

1.2.2.2 Llamadas externas

Son aquellas llamadas que se realizan hacia o desde un número que no se encuentra en la red privada de la empresa. Se clasifican en entrantes y salientes.

1.2.2.2.1 Llamadas externas-entrantes

Son aquellas llamadas realizadas desde un número ubicado en la red pública hacia un número que se encuentra en la red privada.

1.2.2.2.2 Llamadas externas-salientes

Son aquellas llamadas realizadas desde un número que se encuentra dentro de la red privada hacia un número que se encuentra en la red pública, por estas características son a las que se les realiza el proceso de facturación.

1.2.3 Facturación

El término facturación proviene de la acción y efecto de facturar. Facturar es la realización de una operación económica, por lo general se trata de una compraventa, es aquello donde el que vende puede rendir sus cuentas.

1.2.4 Facturación de Llamadas Telefónicas

Facturar en forma del costo monetario las llamadas telefónicas, es calcular en función del impuesto, duración y tarifa de las mismas, cuánto el usuario debe abonar por concepto de las llamadas que realiza a través de una planta telefónica.

1.2.5 Planta Telefónica

Dispositivo utilizado por una empresa de telefonía para operar las llamadas telefónicas en el sentido de hacer conexiones y retransmisiones de información de voz. Permite conectar de manera correcta a los abonados al servicio telefónico entre sí, poniendo en contacto al abonado que llama con el destinatario de la llamada (abonado de destino), mantener la comunicación durante el tiempo que lo requiera el abonado y proveer información para la facturación de llamadas.

Las plantas telefónicas también cuentan con un sin número de funcionalidades que hacen más cómodo el servicio de telefonía ofrecido a los abonados, dentro de sus funcionalidades se encuentran: permitir marcar el último número marcado, dejar mensajes, llamadas en espera, alarma, entre otros.

Actualmente se han destacado varios proveedores mundiales de plantas telefónicas, entre ellos tenemos: ALCATEL, MITEL, PANASONIC, ERICSSON, INFINITY, TOSHIBA y LG.

1.2.6 Asterisk

Asterisk es un software que provee todas las características que se esperan de una PBX (Private Branch Exchange o Centralita Telefónica Privadas), usa el concepto de software libre y es promovido por la empresa Digium que se encarga entre otros aspectos del desenvolvimiento de código fuente y en

hardware de telefonía de bajo costo que funciona con Asterisk. Entre los servicios que proporciona Asterisk se encuentra: voicemail con directorios, conferencias, respuesta de voz interactiva, llamada en espera y grabado de llamadas. La versión del software que se utilizará en el presente trabajo será la 1.6.

1.3 Sistemas de Facturación Telefónica en el mundo

Actualmente en el mundo existen sistemas vinculados a Asterisk, como por ejemplo: FreePBX, Billing IP Calling Card, A2Billing, AstBill y Elastix. Todos estos sistemas cuentan con interfaces amigables y fáciles de usar para el usuario. También brindan la posibilidad de consultar los reportes de las llamadas, así como configurar todo el proceso de facturación aunque se considera que en conjunto no cuentan con un eficiente sistema de reporte y facturación que responda a todas las demandas de la telefonía actual.

A continuación se muestran varias características de estos sistemas.

1.3.1 FreePBX

Software de telefonía bajo licencia GPL dedicado a la administración de PBX Asterisk. Cuenta con una interfaz de usuario fácil de usar que permite tener el funcionamiento de Asterisk inmediatamente, sin tener que modificar sus ficheros de configuración manualmente. Entre sus principales funcionalidades cuenta con un Registro Detallado de Llamadas y la opción de generar reportes dependiendo de criterios. Este sistema no implementa la factura de las llamadas y no cuenta con gráficas que ilustren el comportamiento de las llamadas. (1)

1.3.2 Billing IP Calling Card

Sistema realizado sobre plataforma libre. Presenta una solución de software apta para todas aquellas empresas que quieran dar servicio de telefonía. Este software basado en un servidor, cuenta con todos los elementos necesarios para la administración, el control y los cobros de servicios telefónicos que se provean. En combinación con la plataforma PBX-IP de CallFon representa un verdadero sistema de funcionalidades de telefonía direccionada a la prestación de servicios, permitiendo la facturación de las comunicaciones, pre o postpago, incluido los reportes y estadísticas del funcionamiento de las líneas IP[3] o las TDM[4], configurables para proveer un amplio rango de servicios, tarifas y facturación. (2)

[3] Internet Protocolo. Es un número que identifica de manera lógica y jerárquica a una interfaz de un dispositivo.

[4] Multiplexación por División de Tiempo. Técnica de multiplexación en la que los distintos canales se transmiten en distintos instantes de tiempo utilizando todo el ancho de banda asignado.

1.3.3 A2Billing

Sistema de facturación completo para Asterisk realizado sobre plataforma libre. Se puede utilizar para proporcionar servicios de facturación independientes (pre o postpago), presentación de informes y estadísticas sobre redes IP y TDM basadas en la voz y se puede configurar para proveer una amplia gama de servicios, tasa de llamadas, preparar y enviar facturas, así como aceptar pagos a través de una serie de proveedores de servicios de pago. Posee gran auge gracias a su sencillez, su documentación es abundante aunque con respecto a los reportes no posee gran fortaleza, ya que se centra principalmente en la factura, además muchas de las configuraciones que se deben hacer se realizan sobre ficheros y no sobre bases de datos. (3)

1.3.4 AstBill

AstBill es un software gratuito de código abierto bajo la licencia GPL. Basado en el tiempo de expedición y de facturación para Asterisk y VoIP. Permite el control del crédito en las llamadas salientes, muestra el balance, los gastos, pagos y el número de llamadas en cada cuenta. Permite la administración prepago y postpago a los clientes y control total de crédito de las cuentas de usuario. (4)

1.3.5 Elastix

Elastix es una distribución libre de Servidor de Comunicaciones Unificadas. Integra varias herramientas disponibles para centrales telefónicas e implementa gran parte de sus funcionalidades sobre software Asterisk, mediante una interfaz gráfica y fácil de usar.

Este software fue iniciado y actualmente es mantenido por la compañía ecuatoriana PaloSanto Solutions y permite integrar en un solo paquete la mayoría de las funcionalidades de VoIP PBX, Fax, Mensajería Instantánea y Correo electrónico así como añadir su propio conjunto de utilidades y creación de módulos.

Las características proveídas por Elastix son muchas y variadas. Elastix integra varios paquetes de software, cada uno incluye su propio conjunto de características. Además, añade nuevas interfaces para el control y reportes de sí mismo, lo que lo hace un paquete completo.

Algunas de las características proveídas por Elastix son (5):

- ✓ Soporte para video. Se puede usar video-llamadas con Elastix en su versión 2.0.

- ✓ Soporte para virtualización. Es posible correr múltiples máquinas virtuales de Elastix sobre la misma caja.
- ✓ Interfaz web para el usuario, realmente amigable.
- ✓ "Fax y email" para faxes entrantes. También se puede enviar algún documento digital a un número de fax a través de una impresora virtual.
- ✓ Interfaz para tarifas.
- ✓ Configuración gráfica de parámetros de red.
- ✓ Opciones para reiniciar/apagar remotamente.
- ✓ Reportes de llamadas entrantes/salientes y uso de canales.

Desde sus inicios la interfaz web del Elastix ha estado en constante cambio tanto en arquitectura como desarrollo web, con el objetivo de mejorar la interacción con los desarrolladores y el usuario. Fue por esto que se desarrolló sobre el framework NEO, el cual está basado principalmente en la necesidad de los desarrolladores.

Elastix también puede ser utilizado para facturar llamadas, con el inconveniente de que todas las tarifas asignadas a los troncales son por minutos y no permite el cálculo de los costos en un tiempo real, convirtiendo así su sistema en un sistema de facturación estático.

Independientemente de los inconvenientes del sistema Elastix, el presente trabajo desarrolla un Servicio de Facturación en tiempo real, que complementa la potencialidad de sus funcionalidades en cuanto a reportes y facturación de llamadas. La versión del software que se utilizará en el presente trabajo será la 2.0.0-21.

1.4 Sistemas de Facturación Telefónica en Cuba

Cuba no está exenta de los avances con respecto a la telefonía, con la aparición en 1994 de la Empresa de Telecomunicaciones de Cuba (ETECSA) se dio un importante paso en cuanto a servicios de telefonía, pero, consultar todo el proceso de facturación de una planta telefónica basada en Asterisk se hace un poco complejo. Actualmente en los centros telefónicos del país se cuenta con el hardware necesario para

llevar la comunicación a la inmensa mayoría de los lugares, pero, la adquisición de este hardware también lleva a la compra del software que controla su funcionamiento y permite su configuración. Lo ideal sería desarrollar un software que permitiera adaptarse a cualquiera de las plantas telefónicas que se poseen independientemente de su marca.

En la UCI se han realizado varios estudios sobre cómo poder implantar una PBX basada en Asterisk y hasta estos momentos no se dispone de ninguno. Sin embargo, dentro del centro de TLM se desarrolló un software configurable para diferentes plantas telefónicas, dicho software tiene por nombre Sistema de Gestión Integral de los Costos de Llamadas en Pizarras Telefónicas (SGIC-PABX).

1.4.1 SGIC-PABX

Aplicación desktop desarrollada sobre lenguaje Java, creado para adaptarlo a diferentes plantas telefónicas, actualmente es configurable para plantas telefónicas de distintos proveedores mundiales de pizarras telefónicas como ERICSSON, MITEL, ALCATEL e INFINITY, de los cuales existen varios modelos en todo el territorio nacional. Uno de sus módulos, el Explorador de Llamada, cuenta con un sistema de facturación de llamadas en tiempo real, que permite la gestión de áreas, zonas, formas de cobros y tarifas telefónicas, posibilitando calcular el costo de las llamadas que se realicen hacia dentro o fuera de la institución donde se despliegue. También hace un reporte completo de todas las llamadas y muestra este reporte gráficamente.

El presente trabajo, basado en el sistema de facturación de SGIC-PABX decide llevar a cabo este servicio sobre la interfaz web del Elastix.

1.5 Metodologías de Desarrollo:

El software ha sido elemento fundamental en la evolución de los sistemas de computadoras y ha venido desarrollándose por años hasta convertirse en una industria por sí mismo. Junto a esto surgió la ingeniería, que permite obtener un software confiable y eficiente ante las exigencias de un cliente. Su desarrollo no es fácil, es por eso que desde su surgimiento se hizo necesario documentar aquellos procedimientos, técnicas y herramientas que permitan a los desarrolladores crear nuevas soluciones, de ahí el término metodología.

Existen aquellas metodologías más tradicionales que se centran especialmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir, y las

herramientas y notaciones que se usarán, estas han resultado ser efectivas en un gran número de proyectos.

Por otra parte, se encuentran las metodologías ágiles, las cuales dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas, actualmente se dice que están revolucionando la manera de producir software.

Entonces se puede afirmar que las metodologías tradicionales o pesadas, como comúnmente se les conoce, tienen una fuerte planificación durante el proceso de desarrollo de software mientras que las metodologías ágiles se basan en desarrollar un software incremental, sencillo y adaptable.

1.5.1 Proceso Unificado de Modelado

Dentro de las metodologías de desarrollo tradicionales más conocidas y usadas en el mundo entero se encuentra Rational Unified Process o Proceso Unificado de Modelado (RUP) a continuación se muestran varias características de la misma:

- ✓ 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 lo 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, y Despliegue.

1.5.2 Programación Extrema

Dentro de las metodologías ágiles o ligeras más conocidas y usadas está Extreme Programming o Programación Extrema (XP) que básicamente plantea que se trabaje directamente con el cliente haciendo

pequeñas iteraciones y como resultado mini entregas cada dos semanas, donde no existe más documentación que el propio código.

Sustituyendo los requisitos y casos de usos que plantea RUP, se utilizan las HU con el objetivo que cada una no puede demorarse más de una semana, permitiendo el trabajo en parejas, o sea, dos programadores por máquina, donde los programadores menos experimentados aprenden de los más experimentados.

XP también define un estándar de codificación, posibilitando que en vez de que los programadores desarrollen cada uno en su propio estilo lo hagan sobre uno solo, el que está definido por la metodología, logrando uniformidad y organización. En un equipo de desarrollo si una pareja de programadores logra alguna parte de un código que sea reutilizable se lo comunica de inmediato a los demás programadores, por eso se dice que en XP el código no es de nadie, todo el equipo tiene el derecho de manipularlo con la esperanza de que en las manos de cada uno se pueda perfeccionar.

Características del equipo de desarrollo:

Tabla 1: Características del equipo de desarrollo.

Características	**Valoración**
Composición del equipo de desarrollo	3 personas ✓ 1 cliente ✓ 2 desarrolladores
Experiencia productiva	(baja, media, alta) ✓ media
Conocimientos de Metodologías Ágiles.	(bajo, medio, avanzado) ✓ cliente(bajo)

	✓ desarrolladores(bajo)
Conocimientos de Metodologías Tradicionales.	(bajo, medio, avanzado) ✓ cliente(medio) ✓ desarrolladores(medio)
Conocimientos de Lenguaje de Programación PHP.	(bajo, medio, avanzado) ✓ cliente(medio) ✓ desarrolladores(medio)

Características del cliente:

Tabla 2: Características del cliente.

Características	**Valoración**
Cliente	Se encuentra dentro de la universidad, específicamente dentro del proyecto Comunicaciones Unificadas.
Relación con el cliente	Forma parte del equipo de desarrollo.
Disponibilidad de tiempo	Ilimitada
Retroalimentación cliente-equipo desarrollo	(continua, discontinua) ✓ continua

1.5.3 Metodología de desarrollo seleccionada

Se decide emplear una metodología ágil para el desarrollo del presente trabajo y específicamente se utilizará XP, además de cumplir con todas y cada una de las características planteadas anteriormente, se adapta adecuadamente al desarrollo del presente proyecto y a las condiciones en que se realiza el mismo.

A continuación se exponen las razones de la elección:

- ✓ A pesar de que la experiencia del equipo de desarrollo con respecto a las metodologías ágiles sea bajo, el desconocimiento de una herramienta no impide la utilización de la misma, existen guías y documentación suficiente para investigar en este aspecto.
- ✓ Es un proyecto pequeño donde todo el trabajo se realiza por una pareja de programadores.
- ✓ Los requisitos tienden a cambiar frecuentemente, así, conforme avanza el trabajo, el cliente puede agregar nuevas HU, dividirlas o simplemente eliminarlas. XP permite al equipo la modificación de sus planes conforme a todo lo anterior.
- ✓ El cliente forma parte del equipo de desarrollo, logrando una mejor retroalimentación, corrección de errores y así un producto que satisfaga todas las necesidades del mismo.
- ✓ No se necesita la generación de tantos artefactos y roles pues como se había planteado es un proyecto pequeño que esta centrado en ser desarrollado en el menor tiempo posible.
- ✓ Se tiene la posibilidad de desarrollar algo y probarlo permitiendo terminarlo e integrarlo todo.

Por todo lo planteado anteriormente se decide rechazar como metodología de desarrollo RUP, basándose en las características del equipo de desarrollo de la Tabla 1 y las Características del Cliente de la Tabla 2 que se corresponden perfectamente con la metodología XP.

1.6 Sistemas Gestores de Base de Datos

Existen varias formas de guardar información, una de las más conocidas y seguras son las bases de datos las cuales desempeñan un papel crucial en casi todas las áreas de aplicaciones de las computadoras.

Una base de datos es una colección de información organizada por filas, campos y columnas, “es un conjunto de datos relacionados entre sí, entendiéndose por dato los hechos que pueden registrarse y que tienen un significado implícito”. Un Sistema Gestor de Base de Datos (SGBD) es un conjunto de programas que permiten a los usuarios crear, mantener, construir y manipular base de datos para diversas aplicaciones. (6)

Es por ello que los SGBD se han convertido en el instrumento o soporte básico más ampliamente usado en la gestión de los sistemas informáticos.

A continuación se muestran un conjunto de características de Sistemas Gestores de Bases de Datos:

1.6.1 MySQL

Sistema Gestor de Base de Datos relacional, multihilo y multiusuario, que funciona en diferentes plataformas, sencillo de usar e increíblemente rápido. Es gratis para aplicaciones no comerciales. Posee un sistema de privilegios y contraseñas que son muy flexibles y seguros, las contraseñas son seguras porque todo el tráfico de contraseñas está encriptado cuando se conecta con un servidor. También soporta a grandes bases de datos. Las principales características de MySQL son:

- ✓ Aprovecha la potencia de sistemas multiprocesadores consumiendo muy pocos recursos y memoria, gracias a su implementación multihilo.
- ✓ Soporta gran cantidad de tipos de datos para las columnas.
- ✓ Dispone de API's en gran cantidad de lenguajes (C, C++, Java, PHP).
- ✓ Gran portabilidad entre sistemas.
- ✓ Soporta hasta 32 índices por tabla.
- ✓ Gestión de usuarios y contraseñas, manteniendo un muy buen nivel de seguridad en los datos.
- ✓ Tiene buen rendimiento.
- ✓ Posee un tamaño del registro sin límite.
- ✓ Control de acceso usuarios-tablas-permisos.

1.6.2 PostgreSQL

PostgreSQL es un Sistema de Gestión de Bases de Datos Objeto-Relacionales (ORDBMS) libre, que está ampliamente considerado como el sistema de bases de datos de código abierto más avanzado del mundo. Posee una serie de características positivas respecto a otros, por ejemplo, a diferencia de MySQL, tiene gran escalabilidad haciéndole capaz de soportar una mayor cantidad de peticiones simultáneas de manera correcta. Las principales características de PostgreSQL son:

- ✓ Capaz de ajustarse al número de procesadores y a la cantidad de memoria que posee el sistema de forma óptima.
- ✓ Incluye características de la orientación a objetos, como puede ser la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional aunque no se considera un sistema de gestión de bases de datos puramente orientado a objetos.
- ✓ Extensiones para alta disponibilidad, nuevos tipos de índices y minería de datos.
- ✓ Incorpora una estructura de datos array.
- ✓ Incorpora funciones de diversa índole: manejo de fechas, geométricas, orientadas a operaciones con redes.
- ✓ Soporta el uso de índices, reglas y vistas.
- ✓ Incluye herencia entre tablas aunque no entre objetos, ya que no existen.
- ✓ Incluye características avanzadas tales como los joins.
- ✓ Soporta integridad referencial, la cual es utilizada para garantizar la validez de los datos de la base de datos.

1.6.3 SQLite

SQLite es un sistema gestor de base de datos relacional, que a diferencia de los sistemas de gestión de bases de datos cliente-servidor, no es un proceso independiente con el que el programa principal se comunica si no que su biblioteca se enlaza con el programa pasando a ser parte integral del mismo. Sus principales características son:

- ✓ Varios procesos o hilos pueden acceder a la misma base de datos sin problemas.
- ✓ Varios accesos de lectura pueden ser servidos en paralelo.
- ✓ Un acceso de escritura sólo puede ser servido si no se está sirviendo ningún otro acceso concurrentemente.
- ✓ No posee configuración, esto significa que de la forma en que fue creado y diseñado SQLite, no necesita ser instalado. No prender, reiniciar o apagar un servidor, e incluso configurarlo. Esta

cualidad permite que no haya un administrador de base de datos para crear las tablas, vistas y asignar permisos.

- ✓ Es portable, lo que significa que puede ser ejecutado en diferentes sistemas operativos, como Windows, Linux, BSD, Mac OS X, Solaris y sus bases de datos pueden ser fácilmente portadas sin ninguna configuración o administración.
- ✓ Se ejecuta en muchas plataformas, ofrece buen rendimiento pues realiza operaciones de manera eficiente y es más rápido que MySQL y PostgreSQL.
- ✓ Es de dominio público, y por tanto, es libre de utilizar para cualquier propósito sin costo y se puede redistribuir libremente.

El framework NEO sobre el que se desarrolla la interfaz web de Elastix permite gestionar múltiples bases de datos, posibilitando a diversas instituciones que desarrollen sobre esta plataforma, adaptar su software, en dependencia de las características de su entorno. La aplicación a desarrollar es independiente del gestor de base de datos, lo que significa que puede gestionar diversas fuentes de datos, esta es la razón por la que se decide realizar pruebas sobre sistemas gestores de bases de datos PostgreSQL, MySQL y SQLite; en caso de que no estén disponibles ninguna de las anteriores se posibilita el uso de un fichero.

1.7 Herramientas de Desarrollo

1.7.1 Lenguaje programación del lado del servidor

1.7.1.1 Lenguaje de Programación PHP

PHP es un lenguaje de código abierto muy popular especialmente adecuado para desarrollo web y que puede ser incrustado en HTML. Puede ser utilizado en cualquiera de los principales sistemas operativos del mercado, incluyendo Linux, muchas variantes Unix (incluyendo HP-UX, Solaris y OpenBSD), Microsoft Windows, Mac OS X, RISC OS entre otros (7). Es utilizado para la creación de páginas web dinámicas utilizado fundamentalmente para la programación del lado del servidor aunque actualmente también se utiliza para la creación de aplicaciones con interfaz gráfica. Sus siglas en inglés significan “Hypertext Pre-Processor”.

El lenguaje PHP no necesita ser compilado para ejecutarse. Soporta la orientación a objeto y herencia así como capacidad de conexión con la mayoría de los manejadores de base de datos: MySQL, PostgreSQL, Oracle, MS SQL Server, SQLite, ODBC, entre otras. También incluye gran cantidad de funciones y no

requiere definición de tipos de variables ni manejo detallado de bajo nivel. La mayoría de sus sintaxis tienen un gran parecido con los lenguajes más comunes de programación estructurada, como C y Perl con algunas características específicas y la extensión de sus archivos es .php.

1.7.1.2 Estándar de codificación para PHP en Elastix.

El framework de Elastix, NEO, utiliza en su programación el lenguaje PHP aprovechando así todas y cada una de las propiedades que este brinda. Elastix establece una estándar de codificación para PHP permitiendo mejor legibilidad en el código, de esta forma los desarrolladores no tendrán problema en entender el código de otros desarrolladores que quieran colaborar.

1.7.2 Lenguaje de programación del lado del cliente

1.7.2.1 HTML

HTML es el lenguaje de marcado predominante para la elaboración de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes. Este lenguaje es utilizado para el diseño de las interfaces que se corresponderán con las que establece la interfaz web del Elastix.

1.7.2.2 Java Script

Java Script es un lenguaje de scripting basado en objetos, utilizado para acceder a objetos en aplicaciones. Principalmente, se utiliza integrado en un navegador web permitiendo el desarrollo de interfaces de usuario mejoradas y páginas web dinámicas.

En el presente trabajo se utiliza este lenguaje principalmente para el desarrollo de las ventanas de confirmación de alguna operación realizada.

1.7.2.3 CSS

Las hojas de estilo en cascada (en inglés Cascading Style Sheets) se usan para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML). Estas hojas de estilo permiten en este trabajo seguir las pautas del diseño de la interfaz web de Elastix.

1.7.3 Framework NEO

El uso de los framework actualmente se ha convertido en un punto importante para la ingeniería de software ya que están compuestos por componentes que se pueden personalizar o intercambiar cuando desarrollamos una aplicación. El uso de un framework persigue tres objetivos importantes, acelerar el proceso del desarrollo de una aplicación, reutilización de código ya existente y correcta aplicación de los patrones de la ingeniería.

La naturaleza de la arquitectura de este framework es Modelo Vista Controlador (MVC2), el número se refiere a la versión del mismo, y su desarrollo en la Programación Orientada a Objetos (POO). Posee algunos tipos de datos especiales, validaciones automáticas, recursos de autorización y autenticación, informes y otras herramientas utilitarias que facilitan y optimizan las operaciones.

1.7.4 Visual Paradigm

Visual Paradigm para UML es una herramienta profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una rápida construcción de aplicaciones de calidad, mejores y a un menor costo. Permite dibujar todos los tipos de diagramas de clases, ingeniería inversa, generar código desde diagramas y generar documentación (8).

Se utiliza en este trabajo de diploma la herramienta Visual Paradigm en su versión 3.4, además de todas las características planteadas anteriormente, por ser multiplataforma, muy fácil de usar y con un ambiente gráfico agradable para el usuario, además permite modelar base de datos y transformación de diagramas de Entidad-Relación en tablas de base de datos, posibilitando la creación y diseño del modelo de datos así como el script de la base de datos utilizada para el desarrollo del Servicio de Facturación.

1.7.5 Zen Studio

En el caso del Servicio de Facturación que propone el presente trabajo, el lenguaje de programación que se utilizará será PHP y aunque este lenguaje puede ser utilizado desde editores de texto se hace necesaria la utilización de un editor especializado en el mismo por eso se utiliza fundamentalmente el Zen Studio en su versión 7.1.

Zend Studio o Zend Development Environment es un completo entorno de desarrollo integrado para el lenguaje de programación PHP. Está escrito en Java, y está disponible para las plataformas Microsoft Windows, Mac OS X y GNU/Linux

Dentro de sus características posee: (9)

- ✓ No requiere la instalación previa de PHP ni del entorno de ejecución de Java.
- ✓ Soporte para PHP 4 y PHP 5.
- ✓ Resaltado de sintaxis, autocompletado de código, ayuda de código y lista de parámetros de funciones y métodos de clase.

1.7.6 Dreamweaver

Dreamweaver es una herramienta profesional para la creación de sitios web, proporciona una potente combinación de herramientas visuales de diseño, funciones de desarrollo de aplicaciones y soporte para la edición del código, características que permiten a los desarrolladores y diseñadores con más o menos experiencia crear rápidamente sitios web.

Las funcionalidades del Dreamweaver se han ampliado en ámbitos esenciales como XML y CSS, también es compatible con todas las principales tecnologías de servidor como, por ejemplo, ColdFusion, PHP, ASP, ASP.NET y JSP, para que los desarrolladores puedan dar vida a sus diseños (10).

En el presente trabajo se utiliza esta herramienta en su versión 8.0, para el diseño de las interfaces que se integrarán a la interfaz web del Elastix y con las cuales el usuario va a interactuar de una forma cómoda y sencilla.

1.8 Conclusiones

En este capítulo se analizaron los principales conceptos, sistemas de facturación existentes en el mundo que utilizan la plataforma telefónica Asterisk, así como la situación con respecto a las plantas telefónicas y sistemas de facturación en el país. También, se seleccionó la metodología, sistema gestor de base de datos, lenguaje de programación y herramientas de desarrollo que cumplieron con todas las características necesarias que permita dar solución a la propuesta.

Capítulo 2. Características del Sistema, Exploración y Planificación

2 Introducción

En este capítulo se describen las características del sistema a desarrollar. Se identifican los procesos existentes con el objetivo de comprender aquellos que intervienen en la facturación de llamadas telefónicas, para un correcto diseño e implementación de los mismos. También, se hace alusión a las fases de Exploración y Planificación propias de la metodología de desarrollo utilizada, donde se confeccionan las HU importantes para cada iteración definida por el equipo de desarrollo.

2.1 Propuesta del Sistema

La solución propuesta está diseñada para aquellas instituciones cuyos servicios telefónicos se gestionen mediante la interfaz web de Elastix utilizando Asterisk como software que provee todas las funcionalidades de una central telefónica. El presente Servicio de Facturación permite la gestión de varios sistemas gestores de base de datos y un fichero. Los sistemas gestores de base de datos y el fichero, se escogerán según las características de la institución o empresa donde se utilice el servicio.

El sistema está compuesto por dos subsistemas, uno encargado de facturar las llamadas en un tiempo real y otro encargado de gestionar las tarifas telefónicas utilizadas en el proceso de facturación.

Ilustración 1 Sistema de Facturación

Esta solución se describe a través de los diagramas de procesos que se muestran:

2.1.1 Servicio de Facturación:

El Servicio de Facturación de Elastix comienza cuando Asterisk obtiene los datos de las llamadas telefónicas que se realizan. Estos datos son guardados en una base de datos. A partir de aquí, se verificará si las llamadas han sido respondidas y si son del tipo externas-salientes. Cumpliendo estas condiciones se procede a facturar las llamadas guardando los datos de las mismas en una base de datos.

A continuación se muestra el modelo del siguiente proceso:

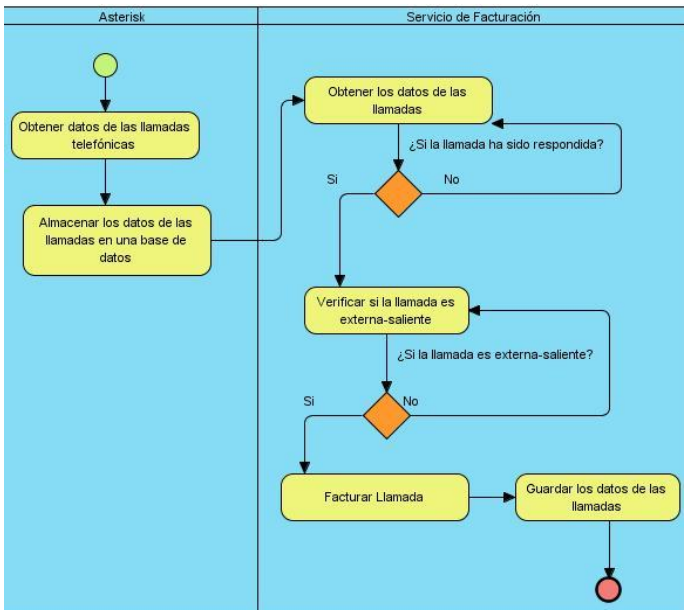


Ilustración 2 Servicio de Facturación

2.1.1.1 Actividades asociadas a Facturar Llamada.

Identificación de Zonas y Áreas

Número Marcado: 045 568258

Código de teleselección	Zona	Posición de Inicio	Longitud Código	Impuesto	
23	Z1	1	2	0.5	← 45 NO
31	Z6	1	2	0.2	← 45 NO
7	Z10	1	2	0.3	← 45 NO
45	Z5	1	2	0.2	← 45 SI

De la llamada telefónica se extrae el número marcado. De la base de datos se cargan las áreas y las zonas, creándose una relación entre ellas. De las áreas se toma la posición de inicio y la longitud del código de teleselección y se busca en el número marcado. Una vez obtenido el código de teleselección se

hace coincidir con cada unos de los números de las áreas cargadas hasta determinar el área al que pertenece dicho código. Conociendo el área se obtiene la zona asociada a la misma.

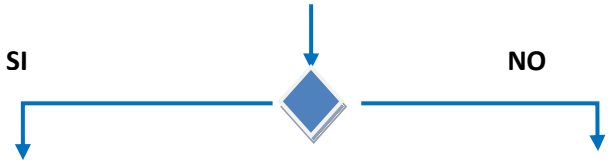
Un **área** es la agrupación de un conjunto de números telefónicos agrupados por un mismo código de teleselección que permiten determinar la localización de la llamada.

Una **zona** es la agrupación de las áreas teniendo en cuenta la distancia existente entre estas, para establecer una tarifa por concepto de zona. Cada zona puede contener una o varias áreas.

Identificación de las tarifas telefónicas

Fecha: 20/05/2011 Hora: 13:00

Fechas Especiales
2/05/2011
10/05/2011
12/05/2011



Tarifa Especial

Tipo	Hora Inicio	Hora Fin	Costo
Especial	06:00	15:00	0.50
Especial	15:00	06:00	0.25

13:00 = 0.50

Tarifa Normal

Tipo	Hora Inicio	Hora Fin	Costo
Normal	06:00	15:00	1.00
Normal	15:00	06:00	0.50

14:00 = 1.00

Para este ejemplo de devuelve **$1.00**

De la llamada se extrae la fech . ɔ la zona se extraen las tarifas normales y especiales. Se carga el listado de fechas especiales y se hace coincidir la fecha de la llamada con las fechas registradas como especiales. En caso de que la fecha de la llamada se encuentre entre las fechas especiales, se cargan las tarifas especiales, de lo contrario se cargan las tarifas normales. Una vez

obtenido el listado de tarifas a las que pertenece la llamada, se ubica en el rango de hora definido en cada tarifa la hora de realización de la llamada. Conociendo en que rango de hora se realizó la llamada se determina el impuesto definido en esa tarifa.

Las **tarifas especiales** son aquellas que tienen asociado fechas específicas, entiéndase por fechas especiales: día de las madres, días feriados, entre otros.

Las **tarifas normales** son las tarifas utilizadas diariamente.

2.1.1.2 Facturación de Llamadas Telefónicas

El costo de las llamadas telefónicas se obtiene multiplicando la tarifa de la llamada con la duración de la llamada en prepago o postpago, sumando el impuesto establecido por la conexión telefónica.

Valor Monetario = Tarifa * Duración de la Llamada $\begin{cases} \text{Prepago} \\ \text{Postpago} \end{cases}$ **+ Impuesto**

Postpago: Se cobra una vez realizada la llamada.
Ejemplo: cobro 1 peso cada 4 segundos, una llamada que tenga una duración de 3 segundos abona 0 pesos y una llamada con duración 9 segundos abona 2 pesos.

Prepago: Se cobra antes de realizar la llamada.
Ejemplo: cobro 1 pesos cada 4 segundos, una llamada que tenga duración 3 segundos abona 1 peso y una llamada que tenga duración 9 segundos abona 3 pesos.

2.1.2 Gestión de las tarifas telefónicas utilizadas en el proceso de facturación:

Para la gestión de las tarifas utilizadas en el proceso de facturación es el usuario quien lo inicia, introduciendo los datos en la interfaz web de Elastix. Los datos son validados por el sistema, notificando un mensaje de error si son incorrectos y guardándolos en una fuente de datos, si son correctos. Luego, si el servicio está iniciado, el sistema lo reinicia, cargando posteriormente los datos de la configuración, pero si el servicio está detenido, el sistema solamente procede a cargar los datos de la configuración.
A continuación se muestra el modelo del siguiente proceso:

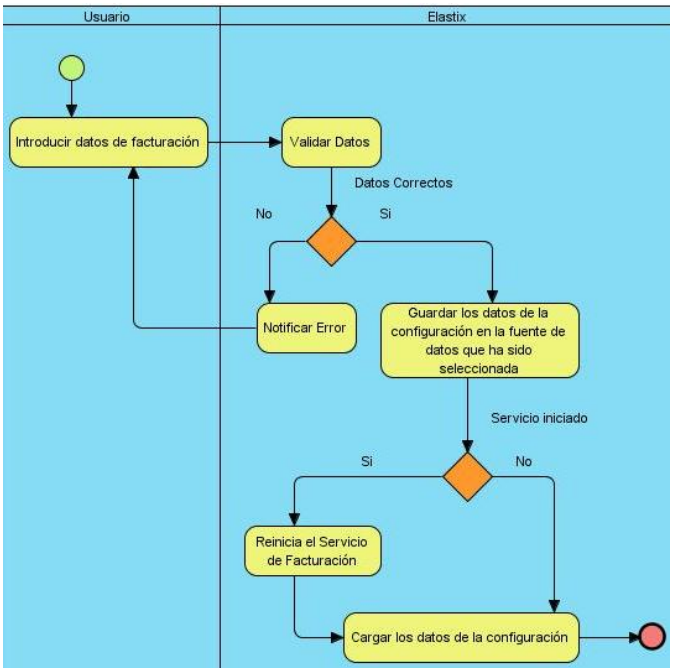


Ilustración 3 Gestión de las Tarifas Telefónicas

2.2 Funcionalidades del Sistema de Facturación

El Sistema de Facturación debe ser capaz de permitir:

- ✓ **Configurar la fuente de datos**
- ✓ **Gestionar tarifas telefónicas**

1. Insertar nueva tarifa telefónica
2. Modificar tarifa telefónica
3. Eliminar tarifa telefónica
4. Buscar una tarifa telefónica.
5. Mostrar detalles de tarifas telefónicas

- ✓ **Gestionar Áreas**

1. Insertar nueva área
2. Modificar área
3. Eliminar área
4. Buscar área
5. Mostrar detalles de las áreas

- ✓ **Gestionar Zonas**

1. Insertar nueva zona
2. Modificar zona
3. Eliminar zona
4. Buscar zona
5. Mostrar detalles de las zonas.

✓ **Gestionar Forma de Cobro**

1. Insertar nueva forma de cobro
2. Modificar forma de cobro
3. Eliminar forma de cobro
4. Mostrar detalles de formas de cobro

✓ **Gestionar Fechas Especiales**

1. Insertar nueva fecha especial
2. Eliminar fecha especial
3. Mostrar detalles de una fecha especial

✓ **Facturar Llamada Telefónica**

✓ **Realizar reportes**

✓ **Controlar Servicio de Facturación**

1. Iniciar Servicio de Facturación
2. Detener Servicio de Facturación
3. Reiniciar Servicio de Facturación

2.3 Requisitos del sistema de facturación

Para un correcto funcionamiento del sistema se deben tener en cuenta los siguientes requisitos no funcionales:

2.3.1 Usabilidad

Se necesitará una preparación previa para operar con el sistema. Se requiere un nivel medio o alto de conocimientos de computación, aunque el manejo de la aplicación es sencillo, permitiendo la fácil comprensión por el usuario.

2.3.2 Disponibilidad

Se hace necesaria la ejecución de la aplicación las veinticuatro horas del día, para la actualización de la base de datos en tiempo real. Esta característica es fundamental debido a que continuamente se obtendrán datos importantes en los registros generados por la planta telefónica.

2.3.3 Eficiencia

La eficiencia del servicio estará determinada en su mayoría por la velocidad de obtención de los datos de las llamadas que se realicen a través de Asterisk.

2.3.4 Hardware

Para la instalación de la aplicación se debe disponer de una computadora de 512 GB de RAM o superior, 160 GB de disco duro o superior.

2.3.5 Software

Se requiere la instalación del PHP 5.1. Elastix como distribución de software libre de comunicaciones unificadas que integra una interfaz web, soporta compatibilidad con sistema operativo Centos 5.5, navegadores Internet Explorer y Mozilla Firefox. Se requiere la comunicación entre el Elastix en su versión 2.0 y Asterisk en su versión 1.6.

2.3.6 Interfaz de usuario

La aplicación propuesta poseerá una interfaz sencilla dirigida a las personas que se relacionen con el sistema. El diseño se realiza siguiendo las pautas de la interfaz web del Elastix. El servicio que se desarrolla es un servicio independiente que realiza sus propias funciones.

2.3.7 Soporte

2.3.7.1 Manual de Ayuda

Al finalizar el proyecto se entregará al cliente unido a todos los entregables un Manual de Ayuda para los usuarios, que les servirá para aprender a interactuar con el sistema.

2.3.7.2 Capacitación

Se impartirá una capacitación por parte del equipo de desarrollo a los trabajadores que utilizarán el sistema.

2.3.8 Restricciones de diseño e implementación

Se hace uso del estándar de codificación que propone el framework utilizado.

2.4 Personas relacionadas con el sistema

Se define como persona relacionada con el sistema aquel que administra todo el Servicio de Facturación y obtiene un resultado de todos los procesos que se ejecutan en el sistema.

En este caso, el **Administrador,** es la persona encargada de gestionar todo el Servicio de Facturación de las llamadas telefónicas a través de la interfaz web de Elastix.

2.5 Fase de Exploración

La fase de Exploración es la primera fase definida por la metodología XP. Es aquí donde se define el alcance real del sistema permitiendo una familiarización del equipo de desarrollo con las herramientas, tecnologías y procesos. Esta fase comienza por la creación de una serie de historias, llamadas HU las cuales definen mediante su redacción qué es lo que verdaderamente necesita el cliente.

2.5.1 Historias de Usuario

Las HU son similares a los escenarios utilizados en la descripción de los casos de usos de RUP, con la excepción de que no se limitan a la descripción de la interfaz de usuario y son definidas por el propio cliente según las necesidades del sistema.

Las HU solamente proporcionarán una breve descripción de las mismas, detalles sobre la estimación del riesgo y cuánto tiempo será empleado en su implementación. Es el cliente el encargado de asignarle una prioridad a cada HU y es el equipo de desarrollo el encargado de asignarle un costo, este se traduce en las semanas que llevará el desarrollo de las mismas.

2.5.1.1 Clasificación de las Historias de Usuario

La prioridad en el negocio:

Alta: Se le otorga a las HU que resultan funcionalidades fundamentales en el desarrollo del sistema, a las que el cliente define como principales para el control integral del sistema.

Media: Se le otorga a las HU que resultan para el cliente como funcionalidades a tener en cuenta, sin que estas tengan una afectación sobre el sistema que se esté desarrollando.

Baja: Se le otorga a las HU que constituyen funcionalidades que sirven de ayuda al control de elementos asociados al equipo de desarrollo, a la estructura y no tienen nada que ver con el sistema en desarrollo.

El riesgo en su desarrollo:

Alta: Cuando en la implementación de las HU se consideran la posible existencia de errores que lleven la inoperatividad del código.

Media: Cuando pueden aparecer errores en la implementación de la HU que puedan retrasar la entrega de la versión.

Baja: Cuando pueden aparecer errores que serán tratados con relativa facilidad sin que traigan perjuicios para el desarrollo del proyecto.

Las HU serán representadas mediante tablas divididas por secciones. El cliente y el equipo de desarrollo trabajan en conjunto para definir como agrupar las HU para su lanzamiento. A continuación se muestran las HU de prioridad alta:

Tabla 3: HU #1 Configurar la fuente de datos

Historia de Usuario	
Número: 1	**Nombre de Historia de Usuario:** Configurar la fuente de datos
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Danae Pérez Arias Rainer Segura Peña	**Iteración Asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados**: 2/3
Riesgo en Desarrollo: Alto	**Puntos Reales:** 1
Descripción: El administrador puede seleccionar la fuente sobre la que se gestionarán los datos, ya sea de una fuente da datos como PostgreSQL, MySQL, SQLite o un fichero.	
Observaciones: En caso de que se desee cambiar de gestor de base de datos ejecutándose el servicio, automáticamente este se reinicia, pues de lo contrario sigue guardando datos en la base de datos anterior.	
Prototipo de	

interfaz:

Tabla 4: HU #25 Iniciar Servicio de Facturación

Historia de Usuario	
Número: 25	**Nombre de Historia de Usuario:** Iniciar Servicio de Facturación
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Danae Pérez Arias Rainer Segura Peña	**Iteración Asignada:** 1
Prioridad en negocio: Alto	**Puntos estimados**: 2/3
Riesgo en Desarrollo: Alto	**Puntos Reales:** 1
Descripción: El administrador puede iniciar el Servicio de Facturación, cargando previamente todos los datos de las llamadas que no han sido facturadas.	
Observaciones: Al iniciar el servicio se guardan en un log los datos del proceso de facturación.	
Prototipo de Interfaz: 	

Tabla 5: HU #28 Facturar Llamada Telefónica

Historia de Usuario

Número: 28	**Nombre de Historia de Usuario:** Facturar Llamada Telefónica
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Danae Pérez Arias Rainer Segura Peña	**Iteración Asignada:** 2
Prioridad en negocio: Alta	**Puntos estimados**: 3
Riesgo en Desarrollo: Alto	**Puntos Reales:** 3
Descripción: El sistema calcula el valor monetario de las llamadas externas-salientes que se realizan en un tiempo real.	
Observaciones:	
Prototipo de interfaz: No aplica	

Las HU de prioridad media y baja podrán consultarse en el Anexo I.

2.6 Planificación

La actividad de planeación en la metodología XP comienza con la creación de una serie de HU que describen las características y funcionalidades requeridas para el software a construir. Durante la fase de planificación se realiza una estimación del esfuerzo que costará implementar cada HU. 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. Esta estimación incluye todo el esfuerzo asociado a la implementación de la HU.

2.6.1 Estimación de esfuerzo por Historias de Usuario

A continuación se muestra la estimación del esfuerzo por cada HU propuesta para el desarrollo de la aplicación:

Tabla 6: Estimación de esfuerzo por Historias de Usuario

Historia de Usuario	Puntos de estimación
1 Configurar la fuente de datos	2
2 Insertar nueva tarifa telefónica	1
3 Modificar tarifa telefónica	1
4 Eliminar tarifa telefónica	1
5 Buscar tarifa telefónica	1
6 Mostrar detalles de tarifas telefónicas	1
7 Insertar nueva área	1
8 Modificar área	1
9 Eliminar área	1
10 Buscar área	1
11 Mostrar detalles de las áreas	1
12 Insertar nueva zona	1
13 Modificar zona	1
14 Eliminar zona	1
15 Buscar zona	1
16 Mostrar detalles de zonas	1
17 Insertar nueva forma de cobro	1
18 Modificar forma de cobro	1
19 Eliminar forma de cobro	1
20 Mostrar detalles de formas de cobro	1

21 Insertar nueva fecha especial	1
22 Eliminar fecha especial	1
23 Mostrar detalles fechas especiales	1
24 Realizar reportes	1
25 Iniciar Servicio de Facturación	2
26 Reiniciar Servicio de Facturación	2
27 Detener Servicio de Facturación	2
28 Facturar Llamada Telefónica	3

2.6.2 Plan de Iteraciones

Después de ser identificadas y descritas las HU y estimar el esfuerzo dedicado a la realización de cada una de ellas, se procede a la planificación de la fase de implementación estableciendo una división de dos iteraciones.

2.6.3 Iteración 1

En la iteración 1 se llevará a cabo el desarrollo de las HU del número 1 hasta el número 27, pertenecientes al subsistema que gestiona las tarifas telefónicas.

2.6.4 Iteración 2

En la iteración 2 se llevará a cabo el desarrollo de la HU número 28, perteneciente al subsistema que se encarga de facturar las llamadas. Al terminar esta iteración se obtendrá una versión 1.0 del producto final y a partir de aquí el sistema se pondrá en función para ser evaluado.

2.6.5 Plan de duración de las iteraciones

El plan de duración de las iteraciones se encarga de mostrar las HU en el orden en que se implementarán en cada iteración así como la duración estimada de las mismas.

Tabla 7: Plan de duración de las iteraciones

Iteración	Orden de la Historias de usuario a implementar.	Duración total
1	Configurar la fuente de datos Insertar nueva tarifa telefónica Modificar tarifa telefónica Eliminar tarifa telefónica Buscar tarifa telefónica Mostrar detalles de tarifas telefónicas Insertar nueva área Modificar área Eliminar área Buscar área Mostrar detalles de las áreas Insertar nueva zona Modificar zona Eliminar zona Buscar zona Mostrar detalles de zonas Insertar nueva forma de cobro Modificar forma de cobro Eliminar forma de cobro Mostrar detalles de formas de cobro Insertar nueva fecha especial	31 semanas

	Eliminar fecha especial Mostrar detalles fechas especiales Realizar reportes Iniciar Servicio de Facturación Reiniciar Servicio de Facturación Detener Servicio de Facturación	
2	Facturar Llamada Telefónica	3 semanas

2.6.6 Plan de entregas

El plan de entrega detalla la fecha fin de cada iteración, los productos obtenidos divididos por subsistemas, así como el módulo sobre el cual se está implementando.

SGTT: subsistema gestionar tarifas telefónicas.

SFLLT: subsistema facturar llamadas telefónicas.

Tabla 8: Plan de entregas

Módulo	Final de la Iteración 1 (13 de abril del 2011)	Final de la Iteración 2 (9 de mayo del 2011)
Facturación	SGTT v0.1	SFLLT v0.1

2.7 Conclusiones

En este capítulo se describió la propuesta del sistema a desarrollar. Se identificaron las funcionalidades y requisitos que el sistema debe cumplir, así como la descripción de las HU divididas por iteraciones y la planificación del esfuerzo dedicado a la realización de cada una de ellas en el orden en que se les dará cumplimiento según las necesidades del cliente.

Capítulo 3. Diseño, Implementación y Prueba

3 Introducción

En este capítulo se describen las fases de diseño, implementación y prueba, propias de la metodología de desarrollo XP. Se identifican y organizan las clases relevantes para las funcionalidades del sistema así como el patrón arquitectónico utilizado para la aplicación web. Se procederá al diseño de la base de datos y abordar las tareas de la ingeniería definidas. Por último, y no menos importante se realizarán las pruebas al software las cuales se derivan de las Historias de Usuario y Tareas de la Ingeniería que se han implementado como parte del lanzamiento del software.

3.1 Patrón de Arquitectura

Los sistemas de software crecen proporcionalmente, hoy se han perfeccionado tanto los algoritmos, que los códigos han dejado de convertirse en un problema. Actualmente el diseño de los sistemas constituye un nuevo reto, los patrones arquitecturales expresan la estructura fundamental para el sistema de software a desarrollar. Para el desarrollo del presente sistema el patrón de arquitectura para aplicaciones web que se seguirá es el Modelo -Vista -Controlador (MVC).

3.1.1 Arquitectura Modelo Vista Controlador

Es una arquitectura usada para Web, donde sus siglas significan Model-View-Controler o Modelo-Vista-Controlador. Esta arquitectura se encuentra estructurada por 3 capas: Vista, Control y Modelo. La naturaleza del framework sobre el que se desarrolla la interfaz web de Elastix es MVC y se encuentra estructurada de la siguiente manera:

Capa Vista

Esta capa está compuesta por el código HTML, por los estilos CSS y los scripts como Java Script.

Capa Control

Esta capa es la encargada de recoger y entregar los datos a la capa de vista. Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista. Para esto se usan sesiones y una librería. Para recoger los datos se usan los métodos GET y POST establecidos en los formularios HTML. Esta capa se encuentra mayormente constituida por archivos de clases.

Capa Modelo

La participación de esta capa es para realizar las transacciones obtener y enviar datos a la capa de datos. Típicamente el modelo de clases contendrá funciones para consultar, insertar y actualizar información de la base de datos. Idealmente debería ser independiente de la base de datos utilizada y notifica los cambios que se hacen en el sistema.

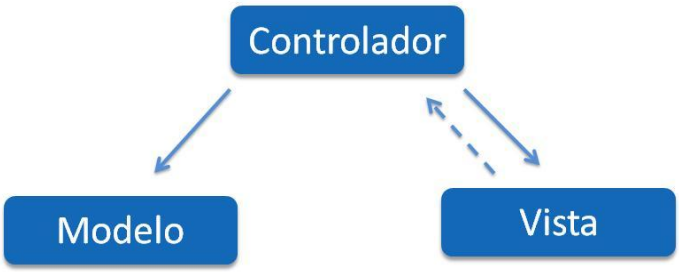


Ilustración 4 Patrón arquitectónico MVC

Es importante destacar que el framework NEO establece un estándar en su contenido o dentro de una carpeta. La carpeta ***themes*** define la capa vista, la carpeta ***libs*** define la capa modelo y el archivo ***index.php*** define la capa control.

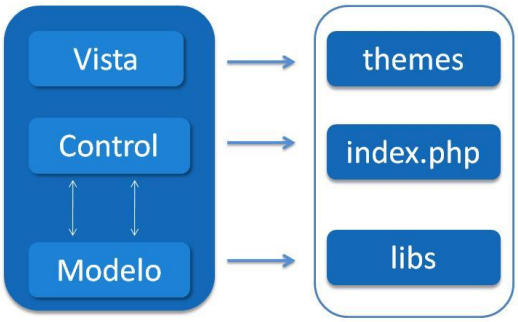


Ilustración 5 Framework NEO basado en arquitectura MVC

3.2 Diagrama de Clases

A continuación se presentan los diagramas de clases divididos por las capas de la arquitectura MVC, para un mejor entendimiento del funcionamiento del framework NEO basado en esta arquitectura para el Servicio de Facturación a desarrollar.

3.2.1 Capa Vista

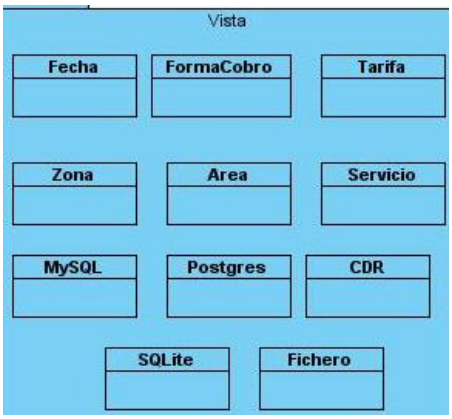


Ilustración 6 Archivos de la Capa Vista

Area, Zona, FormaCobro, Tarifa, Fecha, Fichero, PostgreSQL, MySQL, SQLite, CDR, Servicio: estos archivos se encuentran dentro de la carpeta ***themes*** la cual define la capa vista de la arquitectura MVC. Los mismos definen las plantillas *HTML*, *CSS*, *JavaScript* y su extensión es ***.tpl***.

3.2.2 Capa Controladora

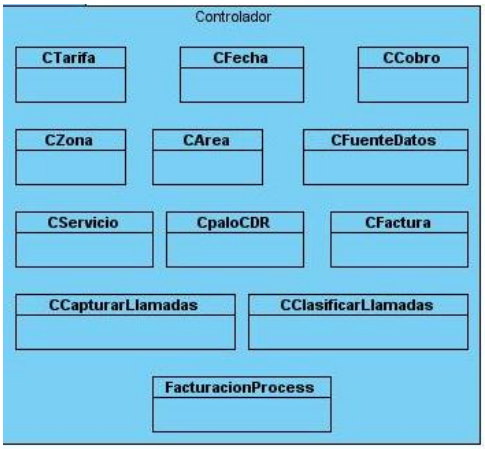


Ilustración 7 Clases de la Capa Controladora

CArea, CZona, CCobro, CTarifa, CFecha: estos archivos pertenecientes a cada módulo, son el punto de entrada al mismo. Es lo primero que el framework NEO tratará de ejecutar para invocar al módulo. Es quien decide qué pantalla mostrar dependiendo de los parámetros recibidos. Estas clases cuentan con un

arreglo de componentes que son invocados por los archivos ***.tpl*** para la confección de los formularios, también poseen una instancia de la clase modelo correspondiente.

CFactura: esta clase cuenta con una instancia de la clase modelo paloDBFacturacion. Se encarga de calcular el costo por cada llamada telefónica que se realiza en un tiempo real. Es la clase encargada de cargar en memoria todos los datos de las entidades de la base de datos.

CCapturarLlamadas: es la clase encargada de capturar las llamadas que se están efectuando a través de Asterisk.

CServicio: es la clase encargada de ejecutar las funciones de iniciar, detener y reiniciar el Servicio de Facturación.

CpaloCDR: esta clase es la encargada de filtrar las llamadas que se están efectuando a través de Asterisk, para luego mostrar los datos de las mismas.

CClasificarLlamadas: esta clase es la encargada de clasificar las llamadas que han sido capturadas por la clase CCapturarLlamadas creando una instancia de la misma. Tiene la responsabilidad de clasificar las llamadas que han sido respondidas y que son del tipo externas-salientes.

FacturacionProcess: esta clase es la encargada de controlar todo el proceso de facturación de llamadas. Cuenta con una instancia de la clase modelo paloDBFacturacion, CClasificarLlamadas, CCapturadoraLlamada y CFactura.

3.2.3 Capa Modelo

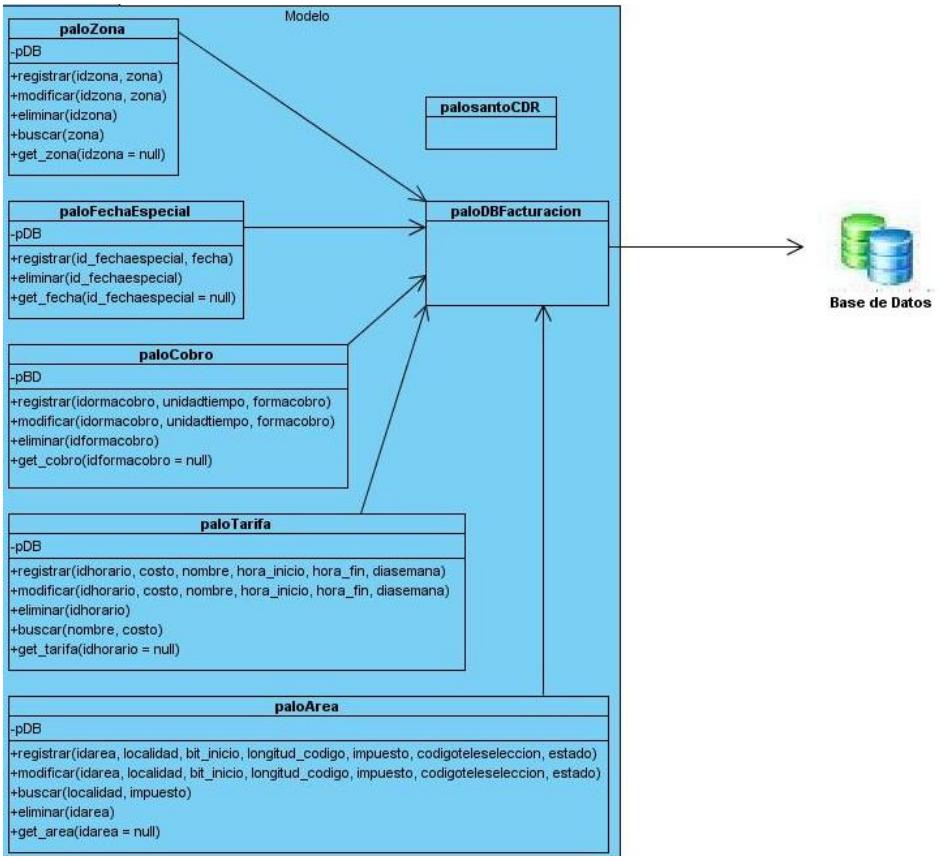


Ilustración 8 Clases de la Capa Modelo y Entidades de la Base de Datos

paloDBFacturacion: clase que gestiona la conexión a la base de datos y las relaciones con las entidades que componen la misma.

paloReportCDR: clase que gestiona la conexión a la base de datos donde se guardan los datos de las llamadas que se están efectuando a través de Asterisk

paloArea: clase responsable de registrar, modificar, eliminar, buscar y obtener detalles de las áreas. Esta clase posee una instancia de la clase paloDBFacturacion.

paloZona: clase responsable de registrar, modificar, eliminar, buscar y obtener detalles de las zonas. Esta clase posee una instancia de la clase.

paloCobro: clase responsable de registrar, modificar, eliminar y obtener detalles de la forma de cobro. Esta clase posee una instancia de la clase paloDBFacturacion.

paloTarifa: clase responsable de registrar, modificar, eliminar y obtener detalles de las tarifas telefónicas. Esta clase posee una instancia de la clase paloDBFacturacion.

paloFechaEspecial: clase responsable de registrar, eliminar y obtener detalles de las fechas especiales. Esta clase posee una instancia de la clase paloDBFacturacion.

3.3 Patrones de Diseño

Un patrón es una solución a un problema en un contexto, codifica conocimiento específico acumulado por la experiencia en un dominio.

Un patrón de diseño es:

- ✓ Una solución estándar para un problema común de programación.
- ✓ Una técnica para flexibilizar el código haciéndolo satisfacer ciertos criterios.
- ✓ Un proyecto o estructura de implementación que logra una finalidad determinada.
- ✓ Un lenguaje de programación de alto nivel.
- ✓ Una manera más práctica de describir ciertos aspectos de la organización de un programa.
- ✓ Conexiones entre componentes de programas.
- ✓ La forma de un diagrama de objeto o de un modelo de objeto.

3.3.1 Patrones para Asignar Responsabilidades (GRASP)

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones.

Dentro de los patrones GRASP utilizados en el desarrollo del sistema se encuentran los siguientes:

Experto

Problema: ¿Cómo lograr que cada clase cumpla con la responsabilidad que le corresponde?

Solución: Asignar una responsabilidad a la clase que tiene la información necesaria para cumplirla.

Las clases paloDBFacturacion, paloArea, paloZona, paloCobro, paloTarifa, paloFechaEspecial, CFactura, CCapturadoraLlamada y CClasificarLlamadas contarán con la información necesaria para cumplir cada una las responsabilidades que le corresponden.

Creador

Problema: ¿Cómo lograr relacionar una clase con la clase responsable de realizar la conexión a la base de datos?

Solución: Asignarle a una clase la responsabilidad de crear una instancia de otra clase.

Las clases paloArea, paloZona, paloCobro, paloTarifa, paloFechaEspecial, CFactura, CCapturadoraLlamada, CClasificarLlamadas y FacturacionProcess serán las responsables de crear una nueva instancia de la clase que realiza la conexión a la base de datos (paloDBFacturación)**.**

Controlador

Problema: ¿Cómo lograr atender un evento del sistema?

Solución: Un Controlador es un objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema. Define además el método de su operación.

La clase CCapturadoraLlamada será la responsable de atender un evento del sistema, es la clase que estará constantemente capturando las llamadas que se están efectuando en un tiempo real.

Alta Cohesión

Problema: ¿Cómo lograr que las clases trabajen en su misma área de aplicación?

Solución: Este patrón es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme.

A las clases paloDBFacturacion, paloArea, paloZona, paloCobro, paloTarifa, paloFechaEspecial, CFactura, CCapturadoraLlamada, CClasificarLlamadas y FacturacionProcess se le asignarán

responsabilidades con el objetivo que trabajen en la misma área de aplicación y que no tengan mucha complejidad.

Bajo Acoplamiento

Problema: ¿Cómo lograr que una clase no dependa de otras clases?

Solución: Este patrón es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Acoplamiento bajo significa que una clase no depende de muchas clases.

A las clases CFactura, CCapturadoraLlamada y CClasificarLlamadas se les asignarán responsabilidades de forma tal que solo se comuniquen con la clase que se encarga de controlar el proceso de facturación (FacturacionProcess).

3.4 Tarjetas Clase – Responsabilidad – Colaborador

La metodología de desarrollo XP como parte de la fase de diseño propone el modelado de Clase-Responsabilidad-Colaborador (CRC), lo que constituye un modelo simple de organizar las clases más relevantes para las funcionalidades del sistema. Este modelado CRC utiliza tarjetas, con el objetivo de desarrollar una representación organizada de las clases.

Un modelo CRC es en realidad una colección de tarjetas índices estándar que representan clases. Las tarjetas se dividen en tres secciones. A lo largo del borde superior de la tarjeta se escribe el nombre de la clase. En el cuerpo de la tarjeta se listan las responsabilidades de la clase a la izquierda y los colaboradores a la derecha (11).

Una **responsabilidad** es cualquier cosa que la clase sabe o hace. Los **colaboradores** son aquellas clases que se requieren para que una clase reciba la información necesaria para completar una responsabilidad. A continuación las tarjetas CRC correspondientes a las clases más relevantes para las funcionalidades del Servicio de Facturación:

Tabla 9: Clase CClasificarLlamada

Clase CClasificarLlamada

Descripción: clase encargada de clasificar las llamadas que han sido capturadas en un tiempo real.	
Responsabilidad	**Colaborador**
Clasificar las llamadas que han sido respondidas	FacturacionProcess
Clasificar las llamadas de tipo externas-salientes	

Tabla 10: Clase CCapturadoraLlamadas

Clase CCapturadoraLlamadas	
Descripción: clase encargada de capturar las llamadas que se están efectuando en un tiempo real.	
Responsabilidad	**Colaborador**
Capturar los datos de las llamadas que se están efectuando	FacturacionProcess
Cargar en memoria listado de las entidades de la base de datos	

Tabla 11: Clase CFactura

Clase CFactura	
Descripción: clase encargada de calcular el valor monetario que el usuario debe abonar por concepto de las llamadas externas-salientes que realiza.	
Responsabilidad	**Colaborador**
Carga en memoria todos los datos de las entidades de la base de datos.	paloReportCDR
Calcular el costo de las llamadas telefónicas	FacturacionProcess
	paloDBFacturacion

Tabla 12: Clase FacturacionProcess

Clase FacturacionProcess	
Descripción: clase encargada de controlar todo el proceso de facturación de llamadas.	
Responsabilidad	**Colaborador**
Inicia el proceso de facturación	CClasificarLlamada
Detiene el proceso facturación	CCapturadoraLlamadas
	Clase CFactura

	paloDBFacturacion

Las restantes tablas correspondientes a las tarjetas CRC que no se presentaron anteriormente pueden encontrarse en el Anexo II.

3.5 Diagrama de clases persistentes

Las clases persistentes son aquellas que tienen durabilidad en el tiempo. Son el punto de partida para la creación del modelo físico de datos.

A continuación el diagrama de clases persistentes correspondiente al sistema de facturación:

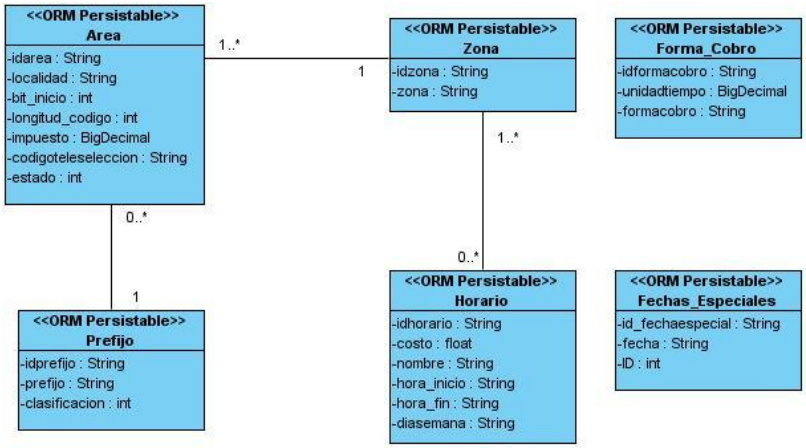


Ilustración 9 Clases Persistentes

Area: entidad que indica la localidad donde se efectúa una llamada (localidad, provincia o municipio).

Zona: entidad que relaciona las áreas con tarifas, según la distancia.

Forma_Cobro: entidad que especifica la forma en que se cobrarán las llamadas (prepago o postpago).

Prefijo: entidad que especifica hacia qué lugar externo se está realizando la llamada.

Horario: entidad que especifica el costo de las llamadas en función de la hora.

Fechas_Especiales: entidad que especifica en qué fecha las tarifas de las llamadas estarán a un precio diferente de lo normal.

3.6 Modelo Físico de la Base de Datos

Partiendo del diagrama de clases persistente visto anteriormente, se define el siguiente modelo físico de la base de datos:

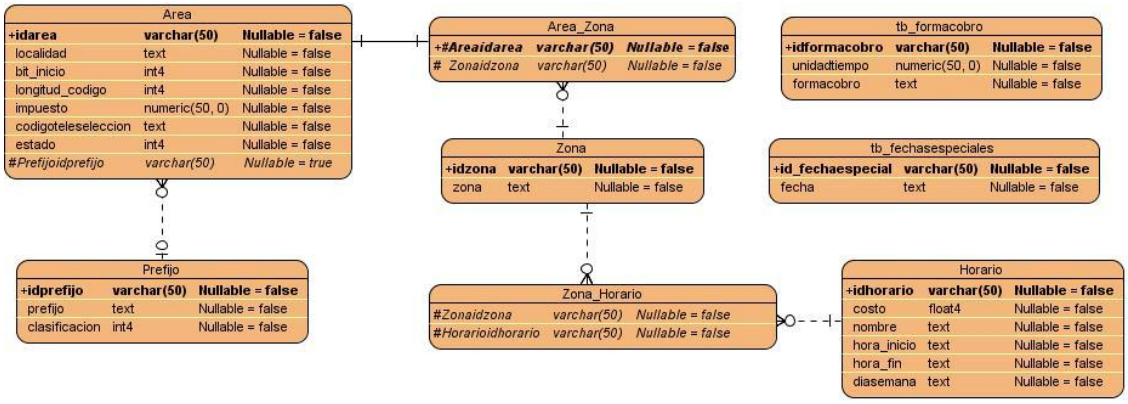


Ilustración 10 Modelo Físico de la Base de Datos

La entidad **Area_Zona** tiene la responsabilidad de guardar las relaciones que existen entre las zonas y las áreas vigentes, de esta forma logramos que existan áreas que estén relacionadas con llamadas que en algún momento se realizaron hacia dichas áreas pero que ya no tienen relación con ninguna zona, así se garantiza el historial de las llamadas.

3.7 Tareas de la Ingeniería

XP plantea que la implementación de un software se hace iterativamente, obteniendo al culminar cada iteración un producto funcional, que debe ser probado y mostrado al cliente. Durante el transcurso de las iteraciones, se realiza la implementación de las HU definidas por el cliente y descritas por el equipo de desarrollo en la etapa de Exploración. Como parte de este plan, se descomponen estas HU en tareas de la ingeniería las cuales son asignadas a los programadores para ser implementadas durante la iteración correspondiente.

Las tareas de la ingeniería serán representadas mediante tablas divididas por secciones. A continuación las tareas correspondientes a las HU de prioridad más alta:

Tabla 13: Tarea #1 Configurar la fuente de datos

Tarea de Ingeniería	
Número Tarea: 1	**Número Historia de Usuario: 1**
Nombre Tarea: Configurar la fuente de datos	
Tipo de Tarea: Desarrollo	**Puntos Estimados:**6 /6
Fecha Inicio: 20-01-2011	**Fecha Fin:** 27-01-2011
Programador Responsable: Rainer Segura Peña	
Descripción: El administrador puede seleccionar la fuente de la que se leerán los datos de las llamadas, ya sea de una base de datos o un fichero. Los datos que deberá introducir si seleccionó un gestor de base de datos como PostgreSQL o MySQL son: dirección del host, nombre de usuario de la fuente de datos y contraseña de la fuente de datos. Si la selección es SQLite o fichero solamente debe escoger cualquiera de estas opciones y conectarse.	

Tabla 14: Tarea #25 Iniciar Servicio de Facturación

Tarea de Ingeniería	
Número Tarea: 25	**Número Historia de Usuario: 25**
Nombre Tarea: Iniciar Servicio de Facturación	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 6/18
Fecha Inicio: 21-03-2011	**Fecha Fin:** 28-03-2011
Programador Responsable: Rainer Segura Peña	
Descripción: El administrador puede iniciar el Servicio de Facturación, cargando previamente todos los datos de las llamadas que no han sido facturadas, seguidos de esta operación se registra en un log los datos del proceso.	

Tabla 15: Tarea #28 Facturar Llamada

Tarea de Ingeniería	
Número Tarea: 28	**Número Historia de Usuario: 28**

Nombre Tarea: Facturar Llamada Telefónica	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 18/18
Fecha Inicio: 14-04-2011	**Fecha Fin:** 9-05-2011
Programador Responsable: Rainer Segura Peña	
Descripción: El sistema captura las llamadas que se están realizando a través de la planta telefónica Asterisk, luego las clasifica verificando que hayan sido respondidas y que sean del tipo externas salientes, para posteriormente calcular el valor monetario que el usuario debe abonar por concepto de las llamadas telefónicas que ha realizado.	

Las tareas de la ingeniería correspondientes a las HU de prioridad media y baja podrán encontrarse en el Anexo III.

3.8 Pruebas

Dentro de los instrumentos capaces de medir el estado de calidad de un producto se encuentran las pruebas. El proceso de pruebas se dirige fundamentalmente a componentes del software o al sistema de software en general, con el objetivo de medir hasta cuando el software cumple las funcionalidades establecidas por el cliente. Propio de la metodología XP, se lleva a cabo la Fase de Prueba. Durante el desarrollo de software, XP establece probar constantemente tanto como sea posible, esto permite un aumento de la calidad del sistema desarrollado reduciendo el número de errores no detectados.

XP divide las pruebas del sistema en dos grupos:

Pruebas Unitarias: las pruebas unitarias son las encargadas de verificar el código y son diseñadas por los programadores. Cada uno de los desarrolladores tiene que ir probando constantemente lo que va obteniendo en el transcurso de la implementación de un sistema, para garantizar que las funcionalidades exigidas por el cliente estén siendo implementadas correctamente.

Las pruebas unitarias fueron desarrolladas constantemente cada vez que se terminaba de implementar alguna funcionalidad probándola directamente en el entorno real.

Pruebas de Aceptación: las pruebas de aceptación también llamadas pruebas del cliente las especifica el cliente y se enfocan en las características generales y las funcionalidades del sistema. En estas serán probadas las funcionalidades exigidas por el cliente, descritas en las HU que se han implementado.

Las pruebas de aceptación correspondiente a cada una de las funcionalidades del Servicio de Facturación serán representadas mediante tablas divididas por las siguientes secciones:

- ✓ **Clases Válidas**: se hará la descripción de cada uno de los pasos seguidos durante el desarrollo de la prueba, se tendrá en cuenta cada una de las entradas válidas que hace el usuario con el objetivo de ver si se obtiene el resultado esperado.
- ✓ **Clases Inválidas**: se hará la descripción de cada uno de los pasos seguidos durante el desarrollo de la prueba, se tendrá en cuenta cada una de las posibles entradas inválidas que hace el usuario con el objetivo de ver si se obtiene el resultado esperado y cómo responde el sistema.
- ✓ **Resultado Esperado**: se hará una breve descripción del resultado que se espera ya sea para entradas válidas o entradas inválidas.
- ✓ **Resultado de la Prueba**: se hará una breve descripción del resultado que se obtiene.
- ✓ **Observaciones**: algún señalamiento o advertencia que sea necesario hacerle a la sección que se está probando.

Las pruebas de aceptación se llevarán a cabo redactando los casos de prueba, teniendo en cuenta el orden de las HU y la prioridad que ha sido asignada a las funcionalidades. Luego se hará la planificación con el cliente de cuándo y cuáles pruebas serán llevadas a cabo, para así reunir los miembros del proyecto seleccionados para realizarlas. Finalmente, se completarán cada uno de los campos de las tablas de las pruebas de aceptación con el resultado de la prueba. Las pruebas de aceptación correspondiente a las funcionalidades del Servicio de Facturación pueden encontrarse en el Anexo IV.

Luego de realizarse las pruebas de aceptación se obtienen un total de 38 no conformidades donde 19 fueron resueltas, 19 no procedían y no quedó ninguna pendiente por resolver, ilustrando estos resultados en la siguiente gráfica:

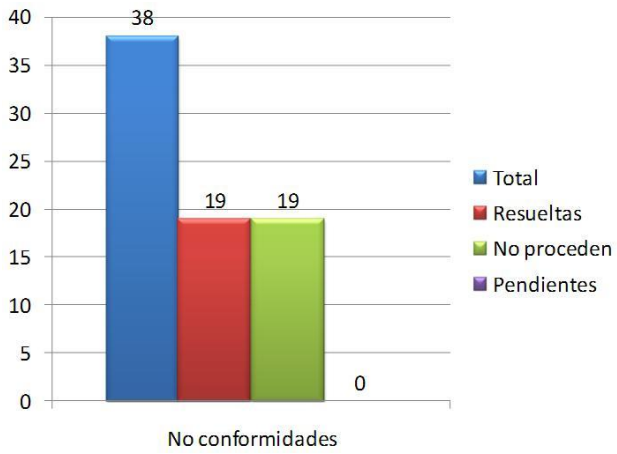


Ilustración 11 Resultado de las Pruebas de Aceptación

También, como constancia de la realización de estas pruebas a la aplicación se cuenta con un Acta de Liberación de Productos de Software que se puede encontrar en el Anexo V.

3.9 Conclusiones

En este capítulo se realizó un profundo análisis sobre el patrón arquitectónico utilizado por el framework sobre el que se desarrolla la aplicación. Se plantearon los problemas de implementación y diseño fundamentales para darle solución con los patrones de diseño. Se identificaron y describieron las tareas de la ingeniería listas para su implementación, correspondiente a las HU descritas en el capítulo anterior. Se identificaron las clases que persisten en el tiempo para obtener a partir de las mismas el modelo físico de la base de datos. Por último, y no menos importante, se llevaron a cabo las pruebas de aceptación logrando la satisfacción del cliente con el software desarrollado.

Capítulo 4. Estudio de la Factibilidad

4 Introducción

En el proceso de desarrollo de software es de vital importancia llevar a cabo la planificación y estimación del esfuerzo con que se realiza el mismo. La estimación consiste en desarrollar varias técnicas que permitan solucionar grandes problemas dividiéndolos en problemas más pequeños. Es aquí donde se identifican los recursos necesarios para el desarrollo del producto, dígase recursos de hardware, software, tiempo, esfuerzo y costo. El presente capítulo realiza el estudio de la factibilidad del sistema propuesto.

4.1 Modelo matemático COCOMO II

COCOMO II consiste en aplicar funciones matemáticas sobre los Puntos de Función sin Ajustar (UFP) o la cantidad de líneas de código (SLOC) estimados para un proyecto. Estas ecuaciones se encuentran ponderadas por factores de costo que influyen en el esfuerzo requerido para el desarrollo del software. Está compuesto por tres modelos adaptables a los diferentes sectores generadores y desarrolladores de aplicaciones y el tipo de información disponible para sostener la estimación del coste del software. Estos modelos son: Composición de aplicación, Diseño Anticipado y Post-Arquitectura.

El estudio de factibilidad de este sistema se realizó mediante el modelo de Diseño Anticipado que se utiliza en las primeras etapas del desarrollo, en las cuales se evalúan las alternativas de hardware y software de un proyecto.

4.2 Características del proyecto

A continuación se llevará a cabo la estimación del proyecto que consiste en la obtención de los Puntos de Función desajustados, los cuales están dados por la suma de cada una de las entradas, las salidas y las consultas externas del sistema, así como los archivos lógicos internos y de interfaz externos.

4.2.1 Entradas Externas

Se definen como un proceso elemental mediante el cual ciertos datos cruzan la frontera del sistema desde afuera hacia adentro. (12)

Tabla 16: Entradas Externas

Nombre de la entrada externa.	Cantidad de	Cantidad de	Clasificación

	ficheros.	elementos de datos.	(Simple, Media. Compleja)
Configurar la fuente de datos	1	4	Simple
Insertar nueva tarifa telefónica	1	5	Simple
Modificar tarifa telefónica	1	5	Simple
Eliminar tarifa telefónica	1	1	Simple
Buscar tarifa telefónica	1	2	Simple
Insertar nueva área	1	5	Simple
Modificar área	1	4	Simple
Eliminar área	1	1	Simple
Buscar área	1	2	Simple
Insertar nueva zona	1	3	Simple
Modificar zona	1	3	Simple
Eliminar zona	1	1	Simple
Buscar zona	1	1	Simple
Insertar nueva forma de cobro	1	3	Simple
Modificar forma de cobro	1	3	Simple
Eliminar forma de cobro	1	1	Simple
Insertar nueva fecha especial	1	1	Simple
Eliminar fecha especial	1	1	Simple
Iniciar Servicio de Facturación	1	1	Simple
Detener Servicio de Facturación	1	1	Simple
Reiniciar Servicio de Facturación	1	1	Simple

Total		**49**	

4.2.2 Salidas Externas

Se definen como un proceso elemental con componentes de entrada y de salida mediante el cual datos simples y datos derivados cruzan las fronteras del sistema desde adentro hacia afuera. (12)

Tabla 17: Salidas Externas

Nombre de la salida externa.	Cantidad de Ficheros.	Cantidad de elementos de datos.	Clasificación (Simple, Media. Compleja)
Mostrar detalles de tarifas telefónicas	1	4	Simple
Mostrar detalles de las áreas	1	5	Simple
Mostrar detalles de zonas	1	1	Simple
Mostrar detalles de formas de cobro	1	2	Simple
Mostrar detalles fechas especiales	1	2	Simple
Realizar reportes	1	9	Simple
Total		**23**	

4.2.3 Consultas Externas

Se definen como un proceso elemental con componentes de entrada y de salida donde un usuario del sistema rescata datos de uno o más Archivos Lógicos Internos o Archivos de Interfaz Externos. Los datos de entrada no actualizan ni mantienen ningún archivo (lógico interno o de interfaz externo) y los datos de salida no contienen datos derivados (es decir, los datos de salida son básicamente los mismos que se obtienen de los archivos). (12)

Tabla 18: Consultas Externas

Nombre de la Petición.	Cantidad de ficheros.	Cantidad de elementos de datos.	Clasificación (Simple, Media. Compleja)
Consultar la Base de Datos	1	1 tabla	Simple

Asterisk: AsteriskCDRDB			
Total		**1**	

4.2.4 Archivos Lógicos Internos

Constituyen un grupo de datos relacionados lógicamente e identificables por el usuario, que residen enteramente dentro de los límites del sistema y se mantienen a través de entradas externas. (12)

Tabla 19: Archivos Lógicos Internos

Nombre de la Petición.	Cantidad de ficheros.	Cantidad de elementos de datos.	Clasificación (Simple, Media. Compleja)
Base de datos del Servicio de Facturación: Facturación	1	8 tablas	Simple
Ficheros que gestionan los datos de las tarifas.	1	8 elementos	Simple
Total		**16**	

4.2.5 Archivos de Interfaz Externos

Son un grupo de datos relacionados lógicamente e identificables por el usuario, que se utilizan solamente para fines de referencia. Los datos residen enteramente fuera de los límites del sistema y se mantienen por las Entradas Externas de otras aplicaciones, es decir, cada Archivo de Interfaz Externo es un Archivo Lógico Interno de otra aplicación. (12)

Tabla 20: Archivos de Interfaz Externos

Nombre de la interfaz externa.	Cantidad de ficheros.	Cantidad de elementos de datos.	Clasificación (Simple, Media. Compleja)
Llamadas Telefónicas realizadas a través de Asterisk que son capturadas por el sistema Elastix.	1	9	Simple
Total		**9**	

4.2.6 Puntos de función desajustados

Los puntos de función desajustados se obtienen a partir de las características del sistema que fueron expuestas con anterioridad, multiplicando la cantidad existente de cada una de las entradas externas, salidas externas, consultas externas, archivos lógicos internos y archivos de interfaz externo con el peso correspondiente a cada uno de ellos.

Tabla 21: Puntos de función desajustados

Elementos	Simple		Medio		Complejo		Subtotal
	No.	Peso	No.	Peso	No.	Peso	
Entradas Externas	49	3	0	4	0	6	147
Salidas Externas	23	4	0	5	0	7	92
Consultas Externas	1	3	0	4	0	6	3
Archivos Lógicos Internos	16	7	0	10	0	15	112
Archivos de Interfaz Externos	9	5	0	7	0	10	45
Total UFP							**399**

Apoyarse en la tabla de peso del factor de complejidad. [Anexo VI]

4.3 Cálculos de instrucciones fuentes, esfuerzo, tiempo de desarrollo, cantidad de hombres y costo.

4.3.1 Cálculos de instrucciones fuentes

Luego de obtenida la cantidad total de puntos desajustados se procede a calcular las instrucciones fuentes del proyecto, para esto se utiliza la siguiente ecuación:

SLOC = UFP x ratio

Donde:

SLOC: cantidad de instrucciones fuentes por puntos de función.

UFP: cantidad de puntos desajustados que como resultado se obtuvo un valor de 399.

ratio: conversión de puntos de función desajustados a líneas de código para el lenguaje PHP con un valor de 59.

A partir de la ecuación anterior:

SLOC = 399 x 59 = 23541

KSLOC: Miles de líneas de código fuente. Es el tamaño del software a desarrollar.

Por tanto:

KSLOC = SLOC / 1000 = 23.5

Los datos anteriores correspondientes a las características del proyecto han sido registrados en la siguiente tabla:

Tabla 22: Características del proyecto

Características	Valor
Puntos de función desajustados.	399
Lenguaje PHP	59
Instrucciones fuentes por puntos de función(SLOC)	23541
Instrucciones fuentes(KSLOC)	23.5

4.3.2 Cálculo del esfuerzo nominal

A continuación se procede al cálculo del esfuerzo nominal a partir de la fórmula completa del modelo de diseño anticipado mencionado anteriormente:

Ecuación #1

$$PM_{Nominal} = A \times (Size)^{B}$$

Donde:

$PM_{Nominal}$: Esfuerzo nominal requerido en meses – hombres.

Ax: Constante que se utiliza para capturar los efectos multiplicativos en el esfuerzo requerido de acuerdo al crecimiento de tamaño del software, tiene un valor de 2.94.

Size: Tamaño estimado del software en Puntos de Función sin Ajustar, en este caso 23.5 KSLOC.

B: Constante denominada Factor escalar y su valor está dado por la resultante de los aspectos positivos sobre los negativos que presenta el proyecto:

Ecuación #2

$$B = 0.91 + 0.01 \times \sum_{i=1}^{5} (W_i)$$

Donde:

W_i: Variables escalares que indican las características que el proyecto presenta en lo que a su complejidad y entorno de desarrollo se refiere.

4.3.2.1 Precedentes (PREC).

El factor de precedencia toma en cuenta el grado de experiencia previa en relación al producto a desarrollar, tanto en aspectos organizacionales como en el conocimiento del software y hardware a utilizar. (12)

4.3.2.2 Flexibilidad de desarrollo (FLEX).

El factor de flexibilidad considera el nivel de exigencia en el cumplimiento de los requerimientos preestablecidos, plazos de tiempos y especificaciones de interfaz. (12)

4.3.2.3 Cohesión del equipo (TEAM).

El factor de escala denominado cohesión del equipo tiene en cuenta las dificultades entre los participantes del proyecto: usuarios, clientes y desarrolladores. Estas dificultades pueden surgir por diferencias culturales, dificultad en la conciliación de objetivos, falta de experiencia y familiaridad con el trabajo en equipo. (12)

4.3.2.4 Solución de riesgos (RESL).

Este factor involucra aspectos relacionados al conocimiento de los ítems de riesgo crítico y al modo de abordarlos dentro del proyecto. (12)

4.3.2.5 Madurez del proceso (PMAT).

El procedimiento para determinar el factor PMAT se basa en el Modelo de Capacidad de Madurez[5] (CMM) propuesto por el Software Engineering Institute.

Una de las formas de calcularlo y la que se utiliza en el presente trabajo de diploma es la captura del nivel de madurez de la organización, resultado de la evaluación según CMM y asignándole el valor correspondiente según la tabla de Factor PMAT de acuerdo al nivel de CMM [Anexo V].

A continuación la siguiente tabla muestra los valores que tomaron estas variables:

Tabla 23: Valores de los Factores de Escala

Nombre	Valor	Justificación
PREC	1.24	Internacionalmente existen varios servicios de facturación pero ninguno con las características del que se está desarrollando. A nivel nacional existe un servicio de facturación en tiempo real sobre el que se basa el presente servicio.
FLEX	1.01	La flexibilidad con la que cuenta permite el cumplimiento de los requisitos establecidos inicialmente.
TEAM	1.10	El equipo de desarrollo presenta una alta cohesión.
RESL	2.03	Se han identificado los riesgos que puedan llegar a la inoperatividad del sistema y aquellos que no.
PMAT	1.56	El software cuenta con la madurez necesaria para cumplir todas sus funcionalidades.
Total $(\sum_{i=1}^{5}(W_i))$	**6.94**	

Apoyarse en las tablas de Factor de Escala [Anexo VI].

Calcular

Luego de haber obtenido el valor total de W_i se procede al cálculo de los valores de la **Ecuación #2**:

[5] Modelo de evaluación de los procesos de una organización.

$B = 0.91 + 0.01 \times 6.94 \approx \mathbf{1}$

Sustituyendo el valor de B en la **Ecuación #1**:

$PM_{Nominal} \approx 2.94 \times (23.5)^{0.97} \approx \mathbf{62.9\ meses / hombres}$

La productividad del proyecto se calcula:

$$\mathbf{Productividad_{Nominal} = KSLOC / PM_{Nominal}}$$

$$Productividad_{Nominal} = 23541 / 62.9 = 374.3$$

4.3.3 Ajuste del esfuerzo nominal

El esfuerzo que ha sido calculado es una valor nominal el cual debe ser ajustado a las características del sistema a desarrollar, para esto se tendrán en cuenta un conjunto de Multiplicadores de Esfuerzo (MEi) los cuales representan las características del sistema y expresan su impacto en el desarrollo total de producto.

4.3.3.1 Multiplicadores de esfuerzo

Los siete multiplicadores de esfuerzo divididos por categorías son:

Tabla 24: Características de los multiplicadores de esfuerzo

Categorías	Multiplicadores	Características
Producto	RCPX	Confiabilidad y Complejidad del producto
	RUSE	Reusabilidad Requerida
Plataforma	PDIF	Dificultad de la Plataforma
Personal	PERS	Aptitud del Personal
	PREX	Experiencia del Personal
Proyecto	FCIL	Facilidades
	SCED	Cronograma de Desarrollo Requerido

Tabla 25: Valores de los Multiplicadores de Esfuerzo

Multiplicadores	Valores	Justificación

RCPX	1.30	La complejidad del producto es alta.
RUSE	1.00	En un futuro puede reutilizarse gran parte del código.
PDIF	1.00	Se desarrolla sobre una plataforma estable, uso de memoria y almacenamiento normal.
PERS	0.83	La capacidad del personal es alta.
PREX	1.00	La experiencia del personal en cuanto a utilización de herramientas y lenguaje es media.
FCIL	0.87	Los entornos de desarrollo integrados al presente sistema facilitan su trabajo.
SCED	1.00	El sistema ha sido desarrollado de acuerdo al cronograma propuesto y en el tiempo establecido.
Total (EM)	**0.93**	

Apoyarse en la tabla de Multiplicadores de Esfuerzo [Anexo VI].

$$PM_{ajustado} = PM_{Nominal} \times \prod (ME_t)$$

$$PM_{ajustado} = 62.9 \text{ meses / hombres} \times 0.93 \approx 59.04 \text{ meses / hombres}$$

4.3.4 Tiempo de desarrollo del software

El tiempo de desarrollo del software esta dado por la siguiente ecuación:

Ecuación #3

$$TDEV = C \times (PM_{Ajustado})^{F}$$

Donde:

TDEV: tiempo de desarrollo del software

C: constante de valor 3.64

$PM_{Ajustado}$: valor calculado anteriormente, 59.04 meses/hombres.

F: valor que se calcula por la siguiente ecuación:

Ecuación #4

$$F = D + 0.2 \times 0.01 \times \sum SF$$

Donde

D: constante cuyo valor es 0.24.

SF: valor de los factores de escala que han sido calculado anteriormente.

Calcular

Sustituyendo los valores en la **Ecuación #4**

$$F = 0.24 + 0.2 \times 0.01 \times 6.94 \approx 0.25$$

Sustituyendo los valores en la **Ecuación #3**

$$TDEV = 3.64 \times (59.04)^{0.25} \approx 10 \text{ meses}$$

4.3.5 Costo total del proyecto

El cálculo del costo total del proyecto esta dado por la siguiente ecuación:

Ecuación #5

C = CHM x PM

Donde:

C: costo total

CHM: costo teniendo en cuenta salario de todos los obreros, el cual se calcula por la siguiente ecuación:

Ecuación #6

CHM = CH x sal

Sal: salario medio por cada trabajador que en este caso se escoge un valor de 100 pesos por cada trabajador.

CH: cantidad de personas destinadas al proyecto. Para el desarrollo del presente trabajo se cuenta con 2 desarrolladores.

Sustituyendo los valores de CH en la **Ecuación #6**

CHM = 2 x 100 **=** 200

Sustituyendo los valores de CHM en la **Ecuación #5**

C = 200 x 59.04 **=** 11 808

4.3.6 Resultado

A continuación la tabla que muestra los resultados obtenidos luego de haber efectuados todos los cálculos de esfuerzo y costo requeridos para el desarrollo del sistema.

Tabla 26: Resultados obtenidos

Cálculo de:	Valor
Esfuerzo	59.04 meses/hombres
Tiempo de desarrollo	10 meses
Cantidad de hombres	2 hombres
Salario medio	100 pesos
Costo	11 808

4.3.7 Análisis del costo

En todo desarrollo de software se lleva a cabo el análisis de los costos que conlleva su producción, estos costos deben ser justificados a partir de los beneficios que aporta el mismo. El sistema propuesto, por las características que presenta, conlleva a un gasto relativamente grande puesto que aquí influye en su gran mayoría el salario de los trabajadores, pero en cuanto a las herramientas y tecnologías utilizadas para su desarrollo principalmente son basadas en software libre, es decir, que no requieren el pago de licencia. Además este software desarrollado sobre plataforma libre es de utilidad en aquellas empresas donde se despliegue, ya que se ahorra en la compra de software y hardware de planta telefónica. También, este permite consultar los reportes de las facturas, posibilitando a los operadores de las empresas hacer un análisis de los costos de forma diaria, semanal, mensual y anual sin necesidad de realizar este proceso de forma manual.

Luego de realizar los cálculos pertinentes sobre el tiempo de desarrollo del sistema se arriba a la conclusión de que es factible su desarrollo comparando los costos de producción con los beneficios que aporta.

4.4 Conclusiones

En el presente capítulo se realizó un análisis de la factibilidad del sistema, obteniendo los valores de esfuerzo, tiempo y costo necesario para el desarrollo del producto. Estos valores permitieron hacer pequeñas comparaciones con respecto al costo de producción y los beneficios que el sistema aporta.

CONCLUSIONES

En el presente trabajo se describieron todos los procesos que fueron identificados y que intervienen en la facturación de llamadas telefónicas, obteniendo una mejor comprensión de cómo funciona el servicio desarrollado. Se seleccionaron las herramientas y la metodología de desarrollo de software a utilizar, así como la tecnología que interviene en el desarrollo de aplicaciones para plantas telefónicas. Se hizo un profundo análisis y comprensión del estilo arquitectónico y los patrones de diseño utilizados para un correcto diseño e implementación de la aplicación.

Con el desarrollo del Servicio de Facturación integrado a la interfaz web de Elastix se dio solución al siguiente problema:

- ✓ Inexistencia de un servicio que se integre a la interfaz web de Elastix que calcule el valor monetario que el usuario debe abonar por concepto de las llamadas externas-salientes que realiza a través de la planta telefónica Asterisk, en un tiempo real.
- ✓ Deficiencia en el sistema de facturación de Elastix, ya que no permite crear múltiples tarifas telefónicas sobre una misma área y los costos de las llamadas no son registrados en una fuente de datos.
- ✓ Imposibilidad en el uso de múltiples fuentes de datos, pues el sistema de facturación de Elastix solamente permite gestionar las tarifas utilizando SQLite.

Por todo lo anteriormente expuesto, se concluye que los objetivos propuestos para el presente trabajo se han cumplido satisfactoriamente, poniendo en práctica todas y cada una de las tareas propuestas para el desarrollo del Servicio de Facturación.

RECOMENDACIONES

A partir de las conclusiones abordadas se listan las recomendaciones en vistas de posibles mejoras:

- ✓ Publicar el presente trabajo para que sea utilizado en aquellas organizaciones del país donde existe al menos una planta telefónica, a fin de conseguir mejorar el trabajo con la facturación de llamadas telefónicas y el uso de software libre.

- ✓ Realizar un estudio sobre el impacto económico del presente software, comparando los gastos que se realizan actualmente en la compra del software y hardware de planta telefónica en todo el país con los gastos que traería si se llegara a desplegar esta aplicación basada en software libre.

REFERENCIAS BIBLIOGRÁFICAS

1. **Montero, Nicolás y Acosta, Luis E.** *Sistema de Reporte y facturacion de PBX Asterisk.* [Documento] Habana : Universidad de las Ciencias Informáticas, 2009.

2. CallFon. [En línea] [Citado el: 22 de Noviembre de 2010.] http://www.callfon.com.ar/descripcion_s.php?info=billing_calling_card.php.

3. A2Billing. [En línea] [Citado el: Noviembre de 23 de 2010.] http://www.asterisk2billing.org/cgi-bin/trac.cgi.

4. astBill. [En línea] [Citado el: 24 de Noviembre de 2010.] http://astbill.com/whatis.

5. *Elastix 0.9-alpha,Manual del Usuario en Español.* [Documento] Ecuador : Emp. Palosanto Solutions, 2009.

6. Conferencia introductoria a los sistemas de bases de datos. [En línea] Departamento central de Base de dato de la Universidad de las Ciencias Informáticas, 2009. [Citado el: 5 de Diciembre de 2010.] http://eva.uci.cu/file.php/624/2._Clases/Semana_1/1ra_frecuencia/MApoyo/C1_Introductoria.pdf.

7. **The PHP Group.** PHP. [En línea] [Citado el: 6 de Diciembre de 2010.] http://www.php.net/manual/es/intro-whatis.php.

8. Freedown Load Manager. [En línea] [Citado el: 7 de Diciembre de 2010.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%5Bcuenta_de_Plataforma_de_Java_14715_p/.

9. **The PHP Company.** Zend Studio. [En línea] [Citado el: 8 de Diciembre de 2010.] http://www.zend.com/en/products/studio/features.

10. **Adobe Developer Connection.** Nuevas Funciones y ventajas de Dreamweaver. [En línea] [Citado el: 10 de Diciembre de 2010.] http://www.adobe.com/es/devnet/dreamweaver/articles/dw8_newfeatures.html.

11. **Pressman, Roger S.** *Ingeniería de software. Un enfoque práctico. Capítulo 8 Modelado de Análisis.* [Documento]

12. **García Martínez, María del Pilar y Crespo Vázquez, Daniel.** *Implementación de un módulo para el envío de mensajes a celulares con el componente SAM.* [Documento] Ciudad de La Habana : s.n., 2010.

BIBLIOGRAFÍA

1. **Landívar, Edgar.** *Comunicaciones Unificadas en Elastix.* [Documento] 2009.
2. PostgreSQL. *PostgreSQL Global Development Group.* [En línea] [Consultado el: 10 de Noviembre de 2010.] http://www.postgresql.org/.
3. Asterisk-ES. [En línea] [Consultado el: 25 de Octubre de 2010.] http://comunidad.asterisk-es.org.
4. NEO framework. [En línea] [Consultado el: 15 de Octubre de 2010.] http://www.neoframework.org.
5. Sun Microsystems, Inc. [En línea] [Consultado el: 2 de Noviembre de 2010.] http://www.mysql.com.
6. PHP. [En línea] [Consultado el: 30 de Noviembre de 2010.] http://www.php.net/manual/es/index.php.
7. La Revista Informatica. com. [En línea] [Consultado el: 2 de Octubre de 2010.] http://www.larevistainformatica.com/CENTRALES-TELEFONICA.html.
8. **Villar, Malay Rodríguez.** Introducción de procedimientos Ágiles en la producción de software en la Facultad 7 de la Universidad de las Ciencias Informáticas. [En línea] Junio de 2007. [Consultado el: 3 de Noviembre de 2010.] http://bibliodoc.uci.cu/TD/TD_0693_07.pdf.
9. **Ulacia Díaz, Yoandy.** Plataforma de Gestión de Contenidos para Dispositivos Móviles. Módulo de Facturación . [En línea] Junio de 2009. [Consultado el: 30 de Octubre de 2010.] http://bibliodoc.uci.cu/TD/TD_2706_09.pdf.
10. **Grau Abalo, Ricardo, Correa Valdés, Cecilia y Rojas Betancur, Mauricio.** *METODOLOGÍA DE LA INVESTIGACIÓN.* [Documento] Ibagué : s.n.
11. **Canós, José H., Letelier, Patricio y Penadés, Mª Carmen.** *Métodologías Ágiles en el Desarrollo de Software.* [Documento] Valencia : Universidad Politécnica de Valencia.
12. **Fernández Escribano, Gerardo.** *Introducción a Extreme Programming.* [Documento]
13. **Robaina Camejo, Ayme.** *Módulo Facturación para el Sistema de Gestión Integral de Costos de Llamadas - PABX.* [Documento] s.l. : Facultad 2. Universidad de las Ciencias Informáticas, Junio 2010.
14. **Arañó Garcés, Eduardo y Marrero Echemendia, Jessica.** *Interfaz para acceso al servidor de correos Zimbra mediante dispositivos móviles.* [Documento] Ciudad de la Habana : s.n., 2010.
15. **Sommerville, Ian.** *Ingeniería de Software. Séptima Edición.* [Documento] Madrid : Pearson educación, 2005.
16. **Gómez, Adriana, y otros.** *UN MODELO DE ESTIMACION DE PROYECTOS DE SOFTWARE.* [Documento] 2009.

17. **Emp. Palosanto Solutions.** Elastix. [En línea] 2006-2008. [Consultado el: 15 de Noviembre de 2010.] http://www.elastix.org, http://www.elastix.org/es/informacion-del-producto/caracterisiticas.html.

18. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El proceso unificado de desarrollo de software.* Madrid : Pearson Education, 2000.

19. **Larman, Craig.** *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* [Documento] México : Pearson Education, 1999.

20. **Van Meggelen, Jim, Madsen, Leif y Smith, Jared.** *Asterisk. The Future of de Telephony.* [Documento] 2007.

21. **Kniberg, Henrik.** *SCRUM Y XP DESDE LAS TRINCHERAS.* [Documento] s.l. : C4Media Inc., 2007.

22. **Prieto, Félix.** *Patrones de Diseño.* [Documento] s.l. : Universidad de Valladolid, 2008/09.

GLOSARIO DE TÉRMINOS

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 voz sobre IP.

Call Center: Es una unidad funcional dentro de la empresa (o bien una empresa en si misma) diseñada para manejar grandes volúmenes de llamadas telefónicas entrantes y salientes desde y hacia sus clientes, con el propósito de dar soporte a las operaciones cotidianas de la entidad.

Métodos GET y POST: son dos métodos diferentes definidos en HTTP, pero ambos son capaces de enviar remisiones de formas al servidor. **GET** es usado para obtener un archivo u otro recurso, posiblemente con parámetros especificando más exactamente lo que se necesita. **POST** es usado para enviar un pedazo de datos al servidor para ser procesado, cualquier cosa que esto signifique.

MVC: del inglés Model-View-Controller, en español Modelo-Vista-Controlador, patrón utilizado en el diseño y desarrollo web.

PBX-PABX: central telefónica privada utilizada dentro de una empresa, en la que los usuarios de la misma pueden realizar llamadas tanto internas como externas.

POO: Programación Orientada a Objetos. Es un paradigma de programación que usa objetos y sus interacciones, para diseñar aplicaciones y programas de ordenador. Está basado en varias técnicas, incluyendo herencia, modularidad, polimorfismo y encapsulamiento.

SGBD: sistema gestor de base de datos es un conjunto de programas que permite crear y mantener una base de datos, asegurando su integridad, confidencialidad y seguridad.

UML: es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Captura decisiones y conocimientos sobre los sistemas que se deben construir.

XML: del inglés: Extensible Markup Language; lenguaje de descripción de páginas de Internet, diseñado con la intención de reemplazar al estándar actual HTML.