Universidad de las Ciencias Informáticas.

Facultad 2.

Título: Sistema de Tele-Pago.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas.

Autor: Julio César Torres Macías.

Tutor: Ing. Alejandro Rodríguez Reyes.

Co-tutor: Ing. Ariel Díaz Rodríguez.

La Habana 2011.

DECLARACIÓN DE AUTORÍA.

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

Para que así conste firmo la presente a los _____ días del mes de _____del año _____.

_____

Julio César Torres Macías.

RESUMEN.

Los sistemas informáticos están siendo aceptados y adoptados por muchas empresas debido a las ventajas competitivas que proporcionan y a la optimización de los procesos en los cuales intervienen. Por todo esto, el país ha desarrollado la industria del software e intensificado el uso de las tecnologías y las comunicaciones, con el propósito de beneficiarse con las ventajas que brindan estos sistemas.

Por el amplio proceso de informatización que se lleva a cabo en la sociedad cubana y la alta demanda de la población respecto al pago de los servicios masivos como pueden ser el servicio eléctrico y el servicio hidráulico, se hace necesario la implementación de un sistema que permita pagar los mismos desde un teléfono, para así agilizar este proceso, ya que permitirá a los clientes pagar sus facturas vía telefónica a cualquier hora del día, con este fin surge Tele-Pago[1].

El país necesita adentrarse en el mundo de las nuevas tecnologías que han surgido con el paso de los años, dejar atrás las tradicionales y brindar servicios automatizados, como Tele-Pago. Se desarrolló un sistema para la gestión de los servicios vía telefónica, para ello se utilizó como software: Asterisk, el cual provee funcionalidades de una central telefónica, además es de código abierto, lo cual facilita su obtención sin tener que pagar licencias de forma sistemática. Provee funcionalidades como la creación de IVR[2], que es utilizada por empresas y entidades que reciben o efectúan muchas llamadas, a fin de reducir la necesidad de personal y costos.

En el presente trabajo se muestran las características del sistema, dando paso a la realización de las fases que propone la metodología de desarrollo seleccionada, mostrando los artefactos generados en cada una de estas.

[1] Tele-Pago: Se define como, un sistema para el pago de servicios vía telefónica utilizando IVR en Cuba.
[2] IVR, Interactive Voice Response/ Respuesta de Voz Interactiva.

TABLA DE CONTENIDO

INTRODUCCIÓN.

La informática como ciencia aplicada que abarca el estudio y aplicación del tratamiento automático de la información, es también definida como el procesamiento automático de la información. (1) Surge ante la necesidad de controlar y manejar el enorme flujo de información existente en el mundo actual, es una corriente innovadora, dinámica y popular, que ha cautivado la atención de muchas instituciones, organizaciones y empresas.

Disponer de sistemas digitales constituye una característica fundamental de modernización, ya que los grandes avances de la ciencia y la técnica logrados por la humanidad no se conciben sin la participación continua de estos sistemas, que se multiplican, evolucionan y desarrollan para el mejoramiento de los sistemas empresariales. Por todos estos avances, Cuba se ha interesado en intensificar el desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC) para lograr la informatización de todas las esferas del país e insertarse paulatinamente en el mercado internacional de software.

El país ha incrementado los servicios masivos a la población como son los brindados por la Empresa de Telecomunicaciones de Cuba (ETECSA), la empresa hidráulica y la empresa eléctrica, proporcionando una mayor aceptación por parte de la población y aumentando así el número de clientes. Debido a dicho aumento de clientes, el pago de estos servicios se ha dificultado, tanto para la empresa que requiere de mayor personal capacitado para atender todas las demandas, además del riesgo que corren los cobradores que circulan por la ciudad con gran cantidad de efectivo, como para los clientes que reciben dicho servicio, que en su gran mayoría se tienen que trasladar a la entidad a pagar por los servicios brindados, perdiendo tiempo debido a las enormes colas y adicionando a esto, que estos pagos se realizan en horario laboral. Todo esto trae consigo que estos procesos se hagan con poca eficiencia y en múltiples ocasiones ocurra atraso en los pagos, proporcionando gastos monetarios para la empresa, así como el retiro del servicio y sanciones a algunos clientes.

Con el propósito de eliminar estos inconvenientes, aumentar la eficiencia de los servicios y elevar la facilidad de pago de los clientes, surgió en Cuba un sistema para el pago de los servicios vía telefónica el cual se llamó Telebanca. Mediante este sistema los clientes pueden pagar sus servicios vía telefónica

interactuando con una operadora (persona encargada de responder la llamada), la cual recoge los datos necesarios para realizar la transacción.

El uso de una operadora trae consigo varios inconvenientes ya que es una persona encargada de estar a toda hora disponible para atender las llamadas, además de que puede cometer errores y el proceso de pago suele ser lento.

Por lo anteriormente planteado se puede inferir que Cuba no dispone de un sistema automatizado que realice la gestión del pago de los servicios vía telefónica.

Analizando lo antes expuesto se define como **problema científico:** ¿Cómo automatizar el proceso de pago de servicios vía telefónica en Cuba?

A partir de este problema se define como **objeto de estudio** la gestión del pago de servicios vía telefónica. Basado en esta idea y para lograr la solución pertinente se define como **objetivo general** del presente trabajo: Desarrollar un sistema de gestión de pago de servicios vía telefónica en Cuba. El **campo de acción** queda enmarcado en la gestión del pago de servicios a través de un sistema telefónico en Cuba utilizando una IVR.

Los **objetivos específicos** son los siguientes:

- Analizar cómo se realiza el pago de los servicios vía telefónica en Cuba.
- Realizar un estudio de diferentes sistemas de pago por teléfono que existen en el resto del mundo.
- Analizar y diseñar una solución para la gestión del pago de servicios por teléfono.
- Implementar el módulo de Tele-Pago.

Para dar cumplimiento a los objetivos propuestos así como controlar y evaluar el proceso investigativo se proponen como **tareas de investigación**:

- Estudio, configuración e instalación del sistema Asterisk.
- Realizar un estudio sobre el Marco de Trabajo SAUXE.
- Estudio de las interfaces para interactuar con el Asterisk.

Para cumplir los objetivos trazados, la investigación se basa en la siguiente **idea a defender**, con el sistema se logrará automatizar el proceso de gestión del pago de los servicios vía telefónica en Cuba.

En la presente investigación serán utilizados métodos teóricos y empíricos con el objetivo de obtener datos que permitan realizarla correctamente. Se analizarán documentos relacionados con los sistemas de pago de servicios, con el objetivo de extraer elementos importantes que se relacionen con el objeto de estudio y así poder alcanzar conocimientos generalizados. Además, se realizarán entrevistas a personas con conocimientos del tema para obtener una mayor información.

Métodos teóricos.

- Analítico–Sintético: Se utilizó durante el proceso de revisión bibliográfica para conocer el funcionamiento de las herramientas, metodologías y tecnologías de desarrollo de software necesarias para la confección de la aplicación.
- Histórico – Lógico: Se utilizó durante la investigación de las tendencias actuales existentes en el mundo respecto a los sistemas de pago de servicios vía telefónica.

Métodos empíricos.

- Observación: Se utilizó este método durante el proceso de investigación para comprobar las características y particularidades de algunos sistemas de pago de servicios vía telefónica existentes actualmente.

- Entrevista: Realizada de forma individual a personas con amplios conocimientos sobres estos sistemas de pago de servicios vía telefónica, con el objetivo de recopilar la mayor cantidad de información posible.

Estructura de los Capítulos.

El Capítulo 1: “Fundamentación Teórica”.

Incluye el estado del arte del tema a tratar, el análisis de las tendencias, tecnologías, metodologías y software usados en la actualidad para resolver el problema. Se incluyen además las herramientas y los lenguajes de programación propuestos para la modelación e implementación del sistema.

Capítulo 2: "Exploración y Planificación".

Se realiza la descripción de la solución propuesta donde se detallan las Historias de Usuario que representan las funciones y propiedades que el sistema debe cumplir, así como el tiempo estimado por iteraciones para dar cumplimiento a cada una de ellas.

Capítulo 3: "Diseño del Sistema".

Se exponen los aspectos relacionados con el diseño del sistema. Se describen las Historias de Usuario, que serán implementadas posteriormente atendiendo a su prioridad.

Capítulo 4: "Implementación y Prueba".

Se engloba los artefactos generados por la metodología de desarrollo del software propuesta en su fase: implementación y prueba.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA.

1.1 Introducción.

En el presente capítulo se hace un análisis de las tendencias y tecnologías actuales de los sistemas de Tele-Pago en el mundo. Se realiza una descripción detallada de algunos de los principales sistemas que realizan pago por teléfono existentes en el mundo. Se fundamenta el uso de las herramientas, las tecnologías, la metodología y el lenguaje de programación que se van a emplear, lo que constituye el basamento teórico de la investigación.

1.2 Definición de IVR.

IVR también conocido como VRU[3]. Es un sistema capaz de "interactuar" con la persona que realizó la llamada mediante una grabación de voz, reconociendo respuestas simples y eventos marcados en el teléfono. En general, la grabación le ofrece al oyente un menú de alternativas posibles, y la persona elige la opción más adecuada mediante el teclado numérico del teléfono. La finalidad del sistema es la interacción con el cliente de manera tal que se pueda discriminar en forma automática el motivo de la llamada y categorizarla, dado que el menú que el IVR le ofrece se puede ir ramificando. (2)

1.3 Ventajas de IVR.

Las ventajas que proporciona el sistema IVR no residen solamente en la reducción de costes y en el incremento de la eficiencia del centro de llamadas (call-center)[4], sino que también:

- ☎ Ayuda a reducir los turnos de los operadores y sus costes asociados.
- ☎ Incrementa las horas de servicio.
- ☎ Disminuye la tasa de llamadas perdidas.

[3] VRU: Unidad de Respuesta de Voz.
[4] call-center: Es un centro de servicio telefónico que tiene la capacidad de atender altos volúmenes de llamadas.

- Incrementa la disponibilidad de los operadores.
- Proporciona una tasa de redireccionamiento mejor, solucionando un mayor número de primeras llamadas.
- Mejora la flexibilidad para responder a las necesidades del cliente y picos de llamadas. (3)

1.4 Estado del Arte.

"Una empresa debe adquirir la infraestructura que le ayude a tomar decisiones rápidas y a resolver los retos de la actualidad; de lo contrario se quedará en el pasado". (4)

Los sistemas informáticos son en la actualidad una herramienta que bien implementada se convierte en un arma competitiva en los negocios, así como las empresas buscan diferenciarse de su competencia, los sistemas informáticos (software) son una manera de hacerlo. Actualmente, el desarrollo de las empresas se ve altamente reflejado en la calidad de los servicios, esta suele ser determinante para ganarle clientes a la competencia. Uno de los aspectos para lograr un mejor servicio es: vencer la barrera de la distancia, no necesariamente hay que ir a una oficina comercial a tramitar. Otro factor importante es el del tiempo, el máximo de horas que se puede perder en espera de ser atendido. Es por esto que la tendencia al desarrollo de las instituciones está altamente ligada al bienestar de sus usuarios y a la capacidad de retener a los mismos, ya sean casuales u ocasionales. Por esto las instituciones empresariales, con el aumento de los avances tecnológicos han creado una nueva tendencia de atender al cliente de forma no presencial, con el establecimiento de un nuevo servicio de pago, asentando al teléfono como principal vía de comunicación sin tener que contactar con una operadora que sirva de intermediario, entre la empresa y el cliente.

Actualmente Cuba cuenta con una Telebanca que en estos momentos se encuentra prestando servicios a la población. Las empresas asociadas con ese sistema son Aguas de la Habana y el MININT[5] para el pago de multas, pero todos estos procesos se realizan mediante el uso de una operadora. Por lo tanto, Cuba no cuenta con un sistema de Telebanca sin la acción de una persona encargada de atender las

[5] MININT: Ministerio del interior.

llamadas, lo que no sucede así con el resto del mundo. A nivel internacional, son muchas las empresas que disponen de este servicio; como por ejemplo el Banco Bilbao Vizcaya Argentaria (BBVA), que es uno de los grupos financieros más importantes de Europa que cuenta con 40 millones de clientes. Este banco comenzó su actividad en junio de 1994, además de ofrecer un servicio gratuito que permite al cliente, durante las 24 horas del día, todos los días del año, entrar en el BBVA sin tener que desplazarse, ni soportar incómodas esperas. Permite acceder a este servicio de banca telefónica marcando el número de teléfono determinado por el banco para prestar el servicio de banca telefónica, al que responde un sistema automático que atiende la llamada. (5)

Banco Agrícola es una importante institución salvadoreña que permite a sus clientes realizar de forma personalizada operaciones financieras, así como obtener información general de los diferentes productos mediante el uso del teléfono. Este servicio está disponible las 24 horas del día y los 7 días de la semana. (6)

La Banca Telefónica de BICE[6], es otras de las muchas instituciones que utilizan este tipo de pago, funciona las 24 horas del día, los 365 días del año. El cliente es atendido por un sistema automático, que le mencionará las opciones de servicios a realizar, ofreciéndole múltiples opciones, como la información de la cuenta corriente en todo momento y desde cualquier lugar. (7)

Scotiabank (Banco de Escocia) es uno de los cinco grandes bancos de Canadá. Es el banco más "internacional" de los bancos canadienses pues es el que más sucursales tiene fuera de su país, Cuenta con 3 centros de llamadas (call-centers) y ofrecen sus servicios por banca telefónica y banca online. El desarrollo y expansión del banco no sólo se localiza en el continente americano sino que también se encuentra en: Europa y África. (8)

1.5 Metodología y Herramientas utilizadas.

1.5.1 Metodología:

[6] BICE: Banco de Chile.

Una Metodología de Desarrollo de Software es un conjunto de técnicas, herramientas, procedimientos y soporte documental que ayuda a los desarrolladores a construir un software. Es la que durante el proceso de desarrollo del software define “quién está haciendo qué, cuándo y cómo para alcanzar un determinado objetivo”. (9)

Metodologías Tradicionales.

Teniendo en cuenta la filosofía de desarrollo de las metodologías, aquellas con mayor énfasis en la planificación y control del proyecto, en la especificación precisa de requisitos y modelado, reciben el apelativo de Metodologías Tradicionales o Pesadas. (10)

Estas metodologías tradicionales imponen una disciplina de trabajo sobre el proceso de desarrollo del software, con el fin de hacerlo más eficiente. Para ello se hace énfasis en la planificación total de todo el trabajo a realizar, y una vez que está todo detallado, comienza el ciclo de desarrollo del producto. Se centran especialmente en el control del proceso, mediante una rigurosa definición de roles, actividades, artefactos, herramientas, notaciones para el modelado y documentación detallada. Además, las metodologías tradicionales no se adaptan adecuadamente a los cambios, por lo que no son métodos adecuados cuando se trabaja en un entorno donde los requisitos no pueden predecirse o bien pueden variar.

Metodologías Ágiles.

Adaptarse al dinamismo de la sociedad actual implica ser “ágil”, es decir, tener la capacidad de proporcionar respuestas rápidas y ser adaptables al cambio. Ambas cualidades siempre han sido deseables, pero en el entorno de negocio actual resultan indispensables. Este requerimiento de agilidad en las empresas, gobiernos y cualquier otra organización provoca que el software también deba ser desarrollado de manera ágil.

Las necesidades de un cliente pueden sufrir cambios importantes desde el momento de contratación de un software hasta el momento de su entrega; y es mucho más importante satisfacer estos últimos

cambios, que lo requerido en el momento del contrato. Esto requiere procesos de software diferentes, que en lugar de rechazar los cambios sean capaces de incorporarlos.

Los procesos ágiles son una buena elección cuando se trabaja con requisitos desconocidos o variables. Si no existen requisitos estables, no existe la posibilidad de tener un diseño estable ni de seguir un proceso totalmente planificado, que no vaya a variar ni en tiempo ni en dinero. En estas situaciones, un proceso adaptativo será mucho más efectivo que un proceso predictivo. Por otra parte, los procesos de desarrollo adaptativos también facilitan la generación rápida de prototipos y de versiones previas a la entrega final, lo cual agradará al cliente. (11)

Las metodologías ágiles proporcionan una serie de pautas y principios que hacen la entrega del proyecto menos complicada y más satisfactoria tanto para los clientes como para los equipos de entrega del software. En la figura 1 se muestran algunos principios que rigen el desarrollo ágil.

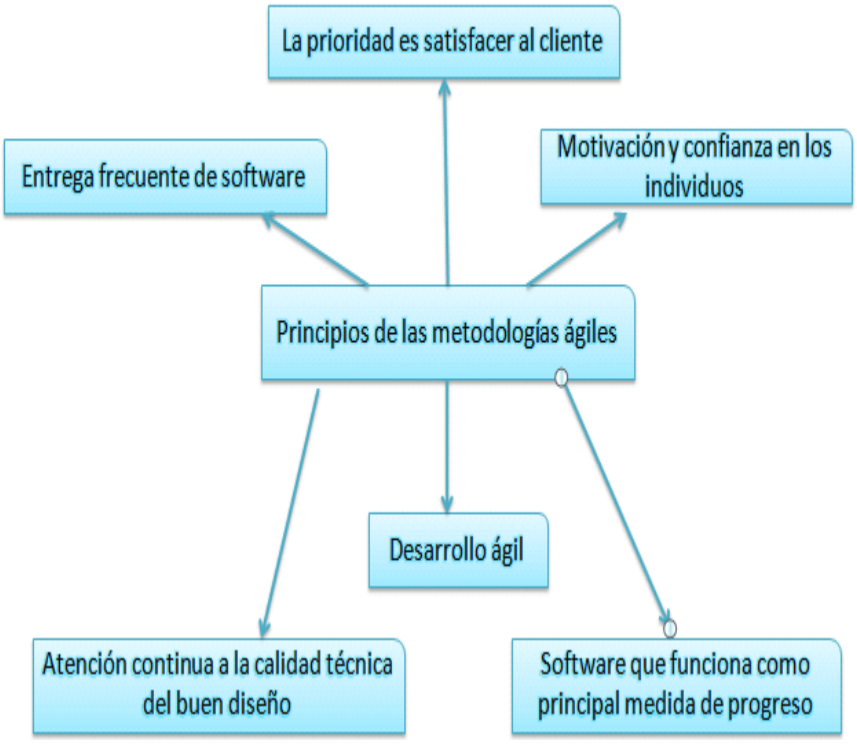

Figura 1 Principios del Manifiesto Ágil (11).

Metodología a utilizar.

Por todo lo anteriormente planteado y debido a que los requisitos del sistema a desarrollar no están bien definidos o bien pueden cambiar, se decidió utilizar una metodología ágil, ya que se adapta mejor a las necesidades del proyecto en desarrollo. Debido a que el proyecto es de tamaño pequeño, y solo se cuenta con 1 programador y el cliente como parte del equipo de desarrollo, además no existe un contrato firmado, ni se dispone de mucho tiempo, es que se escogió XP (Extreme Programming/Programación Extrema) como metodología. Además, el equipo de desarrollo cuenta con una vasta experiencia en esta metodología, lo cual permite que no se tenga que perder tiempo en el aprendizaje de otra.

XP.

XP es una metodología basada esencialmente en la simplicidad y agilidad. Los métodos ágiles, tales como eXtreme Programming, son estrategias de desarrollo de software que promueven prácticas que son adaptativas en vez de predictivas, centradas en las personas o en los equipos de desarrollo, iterativas, orientadas hacia prestaciones y hacia la entrega de comunicación intensiva, y que requieren que el negocio se involucre en forma directa. (12)

Entre sus ventajas resaltan la simplificación del proceso de desarrollo de software además de la reducción del costo del proyecto. Reduce el costo del cambio en las etapas de vida del sistema y evita la burocracia caracterizada en las metodologías robustas, la cual generalmente aumenta el costo de tiempo.

La metodología XP define cuatro variables para cualquier proyecto de software: costo, tiempo, calidad y alcance. Además, se especifica que de estas cuatro variables, sólo tres de ellas podrán ser fijadas arbitrariamente por actores externos al grupo de desarrolladores (clientes y jefes de proyecto). Este mecanismo indica que, por ejemplo, si el cliente establece el alcance y la calidad, y el jefe de proyecto el precio, el grupo de desarrollo tendrá libertad para determinar el tiempo que durará el proyecto.

Además, XP se basa en la comunicación fluida entre todos los participantes, la simplicidad en las soluciones implementadas, el coraje para enfrentar los cambios y realimentación continua entre el cliente

y el equipo de desarrollo. Este es un conjunto mínimo y consistente de valores que permitirán hacer la vida más fácil al grupo, la gerencia y los clientes. (13)

El ciclo de vida de un proyecto XP incluye, al igual que las otras metodologías, entender lo que el cliente necesita, estimar el esfuerzo, crear la solución y entregar el producto final al cliente. Sin embargo, XP propone un ciclo de vida dinámico, donde se admite expresamente que en muchos casos, los clientes no son capaces de especificar sus requerimientos al comienzo de un proyecto. Por esto, se trata de realizar ciclos de desarrollo cortos (llamados iteraciones), con entregables funcionales al finalizar cada ciclo. En cada iteración se realiza un ciclo completo de diseño, desarrollo y pruebas, pero utilizando un conjunto de reglas y prácticas que caracterizan a XP.

1.5.2 Herramientas CASE:

CASE (***C****omputer* ***A****ided* ***S****oftware* ***E****ngineering*, Ingeniería de Software Asistida por Ordenador), comprende un amplio abanico de diferentes tipos de programas que se utilizan para ayudar a las actividades de un proceso informático, como el análisis de requerimientos, el modelado de sistemas, la depuración y las pruebas. En la actualidad, todos los métodos vienen con tecnología CASE asociada, como los editores para las notaciones utilizadas en el método, los módulos de análisis que verifican el modelo del sistema según las reglas del método y generadores de informes que ayudan a crear la documentación del sistema. Las herramientas CASE también incluyen un generador de código que automáticamente genera código fuente a partir del modelo del sistema y de algunas guías de procesos para los ingenieros de software. (14)

Las herramientas CASE están destinadas a aumentar la productividad en el desarrollo del software reduciendo el coste de los mismos en términos de tiempo y dinero. Estas herramientas pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software, principalmente en tareas entre las cuales se pueden mencionar, el proceso de realizar un diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación y detección de errores.

Objetivos de las herramientas CASE:

- Mejorar la productividad en el desarrollo y mantenimiento del software.
- Aumentar la calidad del software.
- Reducir el tiempo, coste de desarrollo y mantenimiento de los sistemas informáticos.
- Mejorar la planificación de un proyecto.
- Aumentar la biblioteca de conocimientos informáticos de una empresa ayudando a la búsqueda de soluciones para los requisitos.
- Automatizar el desarrollo del software, la documentación, la generación de código, las pruebas de errores y la gestión del proyecto.
- Ayuda a la reutilización del software, la portabilidad y estandarización de la documentación.
- Gestión global en todas las fases del desarrollo de software con una misma herramienta.
- Facilitar el uso de las distintas metodologías propias de la ingeniería del software. (15)

Visual Paradigm para UML.

Visual Paradigm para UML[7] es una herramienta profesional que soporta el ciclo de vida completo del desarrollo del software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. (16)
Con esta herramienta pueden modelarse todos los diagramas UML y de entidad-relación necesarios para el desarrollo del software. Además, ayuda a una rápida construcción de aplicaciones con calidad a un menor coste. Brinda la posibilidad de generar código de ingeniería inversa, código desde diagramas y permite la creación de documentación.

De igual forma está diseñado para una amplia gama de usuarios, incluyendo ingenieros de software, analistas de sistema, analistas de negocio, arquitectos de sistema, así como todos los que estén implicados en la construcción de sistemas de software confiables a gran escala, usando un enfoque orientado a objetos. Es una herramienta multiplataforma. (17)

Rational Rose Enterprise Edition.

Rational Rose es una herramienta de software para el modelado visual de sistemas de software mediante UML. Permite especificar, analizar y diseñar el sistema antes de codificarlo. Esta herramienta mantiene la

[7] UML: Lenguaje Unificado de Modelado.

consistencia de los modelos del sistema, posee chequeo de la sintaxis UML y generación automática de documentación.

Facilita la modelación de los procesos del negocio, la captura de requisitos, análisis y diseño orientado a objetos, implementación del sistema mediante componentes y despliegue en las diferentes vistas, vista de casos de uso, vista lógica, vista de componentes y vista de despliegue.

Esta herramienta es compatible solamente con sistemas operativos de Microsoft, permite generar documentación y código fuente (de programas y bases de datos) a partir de los modelos para lenguajes como son: Java, C++, Ada, Visual Basic. También permite el uso de ingeniería inversa (obtención de los modelos a partir del código fuente) para diferentes lenguajes.

Racional Rose se integra con varios entornos de desarrollo, sobre todo con diversas versiones del Visual Studio. Usa un lenguaje estándar común para todo el equipo de desarrollo lo cual facilita la comunicación. Además, acelera la implementación de sistemas con la calidad requerida.

Presenta como desventaja la necesidad de mucha memoria RAM[8] para poder ser manejado de forma rápida y eficiente. (18)

1.5.3 Lenguaje de Modelado de Procesos.

Bizagi Process Modeler.

Bizagi Process Modeler es una herramienta de modelado de proceso que ofrece una completa plataforma de automatización de procesos diseñada para apoyar la transformación empresarial. Permite modelar, automatizar, ejecutar y mejorar los procesos de negocio a través de un entorno gráfico y con la mínima cantidad de código, alcanzando productividad, eficiencia, un crecimiento rentable y sostenido a largo plazo. Además, maneja el ciclo de vida completo de los procesos de negocio: Modelado, Automatización, Ejecución y Mejoramiento continuo. Cada una de estas etapas es administrada a través de distintos

[8] RAM: Random Access Memory.

componentes, los cuales permiten mediante un entorno gráfico y dinámico construir una solución basada en procesos. (19)

Notación para el Modelado de Procesos de Negocio (BPMN).

Business Process Modeling Notation (BPMN) es una notación gráfica que describe la lógica de los pasos de un proceso de negocio. Proporciona un lenguaje común para que las partes involucradas puedan comunicar los procesos de forma clara, completa y eficiente. De esta forma BPMN define la notación y semántica de un Diagrama de Procesos de Negocio. Esta notación ha sido especialmente diseñada para coordinar la secuencia de los procesos y los mensajes que fluyen entre los participantes de las diferentes actividades. Con el objetivo de permitir tanto a los modeladores como a las herramientas de modelado, algunas flexibilidades para extender la notación básica y proveer la habilidad de poder modelar diferentes contextos apropiadamente. (20)

¿Por qué usar BPMN?

- Es un estándar internacional de modelado de procesos aceptado por la comunidad de desarrollo del software.
- Es independiente de cualquier metodología de modelado de procesos.
- Crea un puente estandarizado para disminuir la brecha entre los procesos de negocio y la implementación de estos.
- Permite modelar los procesos de una manera unificada y estandarizada permitiendo un entendimiento a todas las personas de una organización. (20)

IDEF cero (IDEF0).

La traducción literal de las siglas IDEF es Integration Definition for Function Modeling (Definición de la integración para la modelización de las funciones). Es muy utilizado para describir procesos de negocio (atendiendo a los objetivos centrales) y existen numerosas aplicaciones de software que apoyan su desarrollo. Guía en la descripción de cada proceso (o actividad), está considerado como la combinación de cinco magnitudes básicas que se representan gráficamente como:

- Procesos o actividades.
- Entradas.
- Controles.
- Mecanismos o recursos para la realización de tareas.
- Salidas o resultados conseguidos en el proceso (que podrán ser a su vez entradas o controles de otros procesos. (21)

Lenguaje Unificado de Modelado (UML).

UML, del inglés Unified Modeling Language, es un lenguaje que permite especificar, visualizar y construir los artefactos de los sistemas de software. Permite modelar, construir y documentar los elementos que forman un sistema de software orientado a objetos, haciéndolo el estándar de la industria. Sus creadores son los autores de las tres metodologías orientadas a objetos más populares de la primera generación del análisis y diseño orientado a objetos. La finalidad de UML es describir modelos de sistemas, ya sea del mundo real o del mundo del software, basándose en la metodología orientada a objetos. UML está consolidado como el lenguaje estándar en el análisis y diseño de sistemas de cómputo. Permite establecer una serie de requerimientos y estructuras necesarias para diseñar un sistema de software previo al proceso de escribir código. (22)

1.5.4 Sistemas Gestores de Base de Datos.

Un Sistema Gestor de Base de Datos (SGBD) es un conjunto de programas que permiten crear y mantener una Base de Datos, asegurando su integridad, confidencialidad y seguridad. Los SGBD tienen como objetivo servir de interfaz entre la Base de Datos, el usuario y las aplicaciones. Este tipo de sistema debe permitir:

- Definir una Base de Datos: especificar tipos, estructuras y restricciones de datos.
- Construir la Base de Datos: guardar los datos en algún medio controlado por el mismo SGBD.
- Manipular la Base de Datos: realizar consultas, actualizarla, generar informes. (23)

PostgreSQL.

Es un SGBD, distribuido bajo la licencia BSD[9] y con su código de fuente disponible libremente. Es el SGBD libre más potente del mercado. PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando.

Otras características relevantes de PostgreSQL son:

- Es altamente extensible siendo capaz de soportar operadores, funciones, métodos de acceso y tipos de datos definidos por los usuarios.
- Funciona en casi todos los principales sistemas operativos: Linux, Unix y Windows.
- Documentación muy bien organizada, pública y libre, con comentarios de los propios usuarios.
- Soporte nativo para los lenguajes más populares del medio: PHP, C, C++, Perl, Python.

Actualmente PostgreSQL está ampliamente considerado como el SGBD de código abierto más avanzado, esto está enmarcado por poseer muchas características que tradicionalmente sólo se podían ver en productos comerciales de alto calibre. (24)

1.5.5 Frameworks.

Un framework es un conjunto de funciones, procedimientos y herramientas destinadas a la construcción de un determinado tipo de aplicaciones de manera generalista, es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Los frameworks hacen el trabajo más fácil, presentando funcionalidades y componentes reutilizables, que brindan mayor eficiencia a la hora de desarrollar una aplicación. (25)

Sauxe.

Para el desarrollo del sistema de configuración se utilizará el marco de trabajo Sauxe. Basado en la arquitectura en capas (presentación, negocio, servicio, dominio y acceso a datos) y emplea el MVC (Modelo Vista Controlador) para llevar a cabo la comunicación entre la capa de presentación y el negocio.

[9]BSD: Distribución de software Berkeley.

Sauxe a su vez viene integrado por diferentes framework, tales como: Zend Framework, Doctrine y ExtJS de JavaScript. (26)

ExtendJS (ExtJS).

ExtJS es un framework libre que basa toda su funcionalidad en JavaScript a través de diferentes librerías. Presenta ventanas, paneles, tablas, componentes de almacenamiento y agrupamiento, seleccionadores de fechas, mensajes emergentes y un sinnúmero de utilidades que son creadas en tiempo de ejecución, creándose además todos los objetos HTML[10], a través del uso intenso del DOM[11]. Dentro de las principales características de ExtJS, se encuentran:

- Controles de entrada de campos y áreas de texto.
- Campos de fecha, con seleccionador de fechas incluido.
- Campos numéricos.
- Listbox y combobox.
- Control de edición HTML.
- Control de Grids.
- Control de árbol.
- Paneles.
- Barra de herramientas.
- Menús con estilo de aplicaciones de escritorio.

Ventajas:

- La orientación a objetos intensa hacen modular todos sus scripts.
- El diseño está completamente separado de la funcionalidad.
- Funciones comunes como validación, combobox editables, ventanas arrastrables (con minimizar y maximizar), paneles editables, son muy fáciles de implementar.
- Amplia documentación, así como también una gran comunidad de desarrollo.

[10]HTML: Lenguaje de Marcado de Hipertexto.
[11] DOM: Document Object Model/ Modelo de Objetos del Documento.

- Código reutilizable. (27)

Zend Framework.

Se trata de un framework para desarrollo de aplicaciones web y servicios web con PHP, brinda soluciones para construir sitios web modernos, robustos y seguros. Tiene una ventaja muy importante y es que es desarrollado por la Compañía Zend, que esta a su vez respalda comercialmente a PHP.

Presenta entre otras las siguientes características:

- Simplifica la gestión de archivos de configuración.
- Proporciona los componentes que forman la infraestructura del patrón Modelo-Vista-Controlador.
- Proporciona una capa de acceso a Base de Datos, construida sobre PDO[12] pero ampliándola con diferentes características.
- Proporciona mecanismos de filtrado y validación de entradas de datos. (28)

Doctrine ORM.

El Framework Doctrine es un potente y completo sistema de mapas de relaciones de objetos (Object Relational Mapper, ORM por sus siglas en inglés) para PHP 5.2+ con una Base de Datos con capas de abstracción incorporada y además es multiplataforma. Una de sus principales características es la posibilidad de escribir consultas de Base de Datos en un dialecto orientado a objetos de propiedad SQL llamada Doctrine (Lenguaje de consulta DQL[13]), lo cual proporciona a los desarrolladores una poderosa alternativa a SQL que mantiene la flexibilidad sin necesidad de duplicar código innecesario. (29)

1.5.6 IDE de Desarrollo.

IDE es un entorno de desarrollo integrado (en inglés Integrated Development Environment) que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador,

[12]PDO:PHP Data Objects (capa de abstracción de acceso a datos para PHP)
[13] DQL: Doctrine Query Language

un depurador y un constructor de interfaz gráfica. Los IDE pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes. (30)

Zend Studio Neon.

Se trata de un programa de la empresa Zend, una de las que impulsa PHP, orientada a desarrollar aplicaciones web. Es un editor de texto para páginas PHP que proporciona un buen número de ayudas desde la creación y gestión de proyectos.
Zend Studio Neon está basado en Eclipse y es una plataforma de software de código abierto. (31)

1.5.7 Lenguaje de Programación.

El desarrollo de aplicaciones web ha tenido un auge en todo el mundo gracias a las ventajas que las mismas ofrecen a empresas e instituciones. Para el desarrollo de estas existen numerosos lenguajes de programación que se dividen en dos grupos, el primer grupo abarca los lenguajes que corren en el lado del cliente y el segundo los lenguajes que corren en el lado del servidor.

Lenguaje del lado del servidor.

Un lenguaje del lado del servidor es un lenguaje de programación que es reconocido, ejecutado e interpretado en el servidor, al cual el usuario no tendrá acceso. Procesa los datos mediante la interpretación de un script en el servidor para generar páginas web dinámicamente como respuesta, y que es independiente del navegador. (32)

Java.

Java es un lenguaje de programación sencillo, de propósito general, independiente de la plataforma de desarrollo, orientado a objetos, con una sintaxis muy parecida a la de C o C++. Fue creado por la compañía Sun Microsystems a principios de los 90. Su principal ventaja es la independencia de la plataforma. Como principal desventaja de Java se encuentra que el rendimiento de una aplicación hecha en este lenguaje depende de la eficiencia del compilador. (33)

Hypertext Preprocessor (PHP).

PHP es un lenguaje de programación interpretado, que es ejecutado del lado del servidor, es utilizado para generar páginas web dinámicas, embebido en páginas HTML, y ejecutado en el servidor, aunque puede ser utilizado para realizar aplicaciones con una interfaz gráfica, utilizando las debidas extensiones del lenguaje; la mayoría de su sintaxis ha sido obtenida de los lenguajes de programación C, Java y Perl, aunque tiene algunas sintaxis características y específicas de sí mismo. Es un lenguaje que tiene como objetivo principal permitir a los desarrolladores la generación dinámica de páginas web. Al ser ejecutado en el lado del servidor, brinda la posibilidad de acceder a los recursos que se encuentren en el servidor, como por ejemplo, una Base de Datos. PHP es un lenguaje independiente del navegador, o sea, el navegador no tiene que obligatoriamente soportar el lenguaje, pero para que las páginas PHP funcionen, el servidor donde se encuentran las páginas debe soportarlo. Debido a su condición de ser software libre posee una gran cantidad de características, que lo hacen una potente herramienta para la creación de páginas web dinámicas, y que lo convierten en uno de los lenguajes para generar páginas web más usado a nivel mundial. Dentro de las múltiples características de PHP se pueden definir como fundamentales las que siguen:

- Soporta las bases de datos: MySQL, PostgreSQL, Oracle, entre otras.
- Integrado con disímiles bibliotecas externas, brinda la posibilidad de generar documentos en formato PDF[14], hasta el análisis de código XML[15].
- Provee al usuario una solución simple y universal para la paginación dinámica fácilmente programable.
- El código se actualiza constantemente mediante extensiones y mejoras continuas del lenguaje.
- A través del uso de PHP, se pueden realizar diferentes funcionalidades, como por ejemplo el procesamiento de información en formularios, foros de discusión, manipulación de cookies y páginas dinámicas.
- Existen diferentes opciones de configuración para controlar su funcionamiento.
- Muy fácil de aprender.
- Se distribuye de forma gratuita bajo una licencia libre.

[14] PDF: Formato de Documento Portátil.
[15] XML: Lenguaje de marcas extensible.

- Posee la capacidad de expandir su potencial, a través de la gran cantidad de módulos, llamados extends.
- Posee una gran documentación en su sitio oficial, y en otros sitios de la comunidad de desarrollo. (34)

Por políticas de la Universidad se escogió Sauxe como marco de trabajo, el cual está integrado por 4 frameworks, entre ellos Zend, que es utilizado para la programación en PHP. Es por ello que se escogió este lenguaje para el desarrollo del sistema.

Lenguajes del lado del cliente.

Un lenguaje del lado del cliente es un lenguaje que es reconocido e interpretado por el navegador, al cual el cliente tiene acceso pudiendo observar el código en cualquier momento (32).

Hypertext Markup Language (HTML).

HTML es un lenguaje de marcado, diseñado para estructurar textos y presentarlos en forma de hipertexto, es el formato estándar de las páginas web, gracias a Internet y a los navegadores como Internet Explorer, Opera, Firefox o Nestcape se ha convertido en uno de los formatos más populares que existen para la construcción de documentos. Para crear una página web se pueden utilizar varios programas especializados en la creación de páginas, como por ejemplo, el Microsoft FrontPage. (34)

JavaScript.

JavaScript es un lenguaje de programación del lado del cliente, que es interpretado, lo cual significa que no requiere compilación. Es utilizado generalmente en páginas web, siendo embebido dentro del código HTML. La mayoría de los navegadores más usados son capaces de interpretar el código JavaScript incluidos en las páginas web. Este lenguaje es basado en prototipos, pues las clases nuevas se generan clonando las clases base (prototipo), y extendiendo sus funcionalidades. El JavaScript es sumamente utilizado en el mundo del desarrollo web por ser muy versátil y potente, tanto para la realización de pequeñas tareas como para la gestión de complejas aplicaciones. Con su utilización se puede evitar la

sobrecarga del lado del servidor ya que funciones como la validación de formularios, que antes se hacían del lado del servidor, se pueden realizar con este lenguaje y de una manera rápida. (35)

1.5.8 Servidor de aplicaciones web.

Un servidor de aplicaciones web es un programa que se ejecuta continuamente en un ordenador, manteniéndose a la espera de peticiones por parte de un cliente y que responde a estas peticiones adecuadamente, mediante una página web que se exhibirá en el navegador o mostrando el respectivo mensaje si se detectó algún error. (36)

Apache.

El servidor web Apache es un servidor HTTP[16] de código abierto para plataformas Unix (BSD, GNU/Linux), Windows, Macintosh, que implementa el protocolo HTTP/1.1. Tiene una alta facilidad de configuración en la creación y gestión de logs, ya que permite la creación de ficheros de log a disposición del administrador, de este modo se puede tener un mayor control sobre lo que sucede en el servidor. Apache es un servidor web flexible, rápido y eficiente, continuamente actualizado y adaptado a los nuevos protocolos.

Entre sus características más importantes se destacan:

- Corre en una multitud de sistemas operativos, lo que lo hace prácticamente universal.
- Apache es una tecnología gratuita, de código fuente abierto. Esto le da una transparencia a este software de manera que si quiere ver que es lo que está instalando como servidor, se puede saber, sin ningún secreto, sin ninguna puerta trasera.
- Apache es un servidor altamente configurable de diseño modular. Es muy sencillo ampliar las capacidades del servidor web Apache. Actualmente existen muchos módulos para Apache que son adaptables a este, y están ahí para ser instalados cuando se necesiten. Otra cosa importante es que cualquiera que posea una buena experiencia en la programación de C o Perl puede escribir un módulo para realizar una función determinada.

[16] HTTP: Protocolo de Transferencia de Hipertexto.

- Apache trabaja con gran cantidad de lenguajes de script. Teniendo todo el soporte que se necesita para tener páginas dinámicas.
- Apache permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Es posible configurar Apache para que ejecute un determinado script cuando ocurra un error en concreto. (37)

1.5.9 Software para PBX.

Asterisk 1.6.

Asterisk es un programa de software libre (bajo licencia GPL) que proporciona funcionalidades de una central telefónica. Como cualquier PBX[17], puede conectar un número determinado de teléfonos para hacer llamadas entre sí. Implementa las mismas características de una PBX tradicional pero es más flexible. Asterisk incluye muchas características anteriormente disponibles sólo en costosos sistemas PBX propietarios, como buzón de voz, conferencias, IVR y distribución automática de llamadas.

Algunas características de Asterisk:

- Enrutamiento y gestión de llamadas para las llamadas entrantes.
- La generación de llamadas salientes y de enrutamiento.
- Funciones de gestión de los medios de comunicación (grabar, reproducir, generar el tono.)
- Grabación de llamadas de detalle de la contabilidad y facturación. (38)

PHP AGI

Es una clase PHP para Asterisk Gateway Interface (AGI), esta librería está disponible para su uso y distribución bajo los términos de la licencia publica GNU. PHP AGI es una librería que maneja todas las opciones de Asterisk para que puedan ser usadas más fácilmente en el código PHP. (39)

Elastix.

[17] PBX: Ramal privado de conmutación automática (Una central telefónica privada).

Elastix es una distribución de GNU/Linux orientada a las comunicaciones unificadas que integra las mejores herramientas disponibles para PBXs basados en Asterisk es una interfaz simple y fácil de usar. Además añade su propio conjunto de utilidades y permite la creación de módulos de terceros para hacer de este el mejor paquete de software disponible para la telefonía de código abierto. (40)

1.6 Conclusiones.

En este capítulo se han expuesto los antecedentes de sistemas de Tele-Pago tanto a nivel mundial como nacional así como algunas de las metodologías y herramientas actuales que se utilizan para el desarrollo de este tipo de sistema. Con la elaboración de este capítulo se definió la metodología XP como guía para el desarrollo del sistema y BPMN como notación para el modelado de procesos del negocio, ya que permite modelar los procesos de una manera unificada y estandarizada, permitiendo un entendimiento a todas las personas de una organización. Además, se escogió Bizagi Process Modeler para la modelación de procesos del negocio a razón de que la aplicación web resultante de la automatización con Bizagi Process Modeler posee una característica muy importante, y es que cuando se modifica el proceso (cualquier elemento del modelo) la aplicación web refleja este cambio automáticamente, además de Visual Paradigm como herramienta CASE por ser una herramienta multiplataforma que permite el modelado orientado a objetos. Como servidor web se usará Apache por su capacidad de ejecutarse en casi todos los sistemas operativos y por su alta capacidad de configuración, para la gestión de la Base de Datos se eligió PostgreSQL por su condición de ser la herramienta libre más potente actualmente. El lenguaje de programación del lado del servidor a utilizar es PHP debido a que posee una gran comunidad de desarrollo, la cual implementa constantemente mejoras en su código, permite la combinación con un gran número de servidores de bases de datos entre los cuales se encuentran el que se utilizará, se utilizará en la aplicación el SAUXE, que a su vez viene integrado por diferentes frameworks, tales como: Zend Framework, ExtJS, Doctrine y el UCID, como entorno de desarrollo se utilizará Zend Studio Neon, ya que es un editor de texto para páginas PHP que proporciona un buen número de ayudas desde la creación y gestión de proyectos.

CAPITULO 2: EXPLORACIÓN Y PLANIFICACIÓN.

2.1 Introducción.

En este capítulo se realiza un análisis de las características del sistema a desarrollar, teniendo como aspecto principal la arquitectura propuesta para la solución. Además, se describen las diferentes etapas de la metodología XP, haciendo referencia a la fase de exploración y planificación. También se detallan las Historias de Usuario, que serán implementadas posteriormente atendiendo a su prioridad, se exponen los artefactos generados en el transcurso de las diferentes iteraciones.

2.2 Objeto de automatización.

Serán objetos de automatización los procesos vinculados a la interacción con el cliente, es decir, todo lo que realiza la operadora. Además del pago de los servicios brindados por determinada empresa, mediante la construcción de una IVR.

2.3 Propuesta del sistema.

El presente trabajo propone implementar un sistema que permita efectuar el pago de los servicios básicos a través del teléfono sin la intervención de una operadora. Para dar solución al mismo se pretende dividir el sistema en tres subsistemas como se muestra en la figura 2.

Los subsistemas propuestos son:

- Subsistema de almacenamiento de datos.
- Subsistema web de administración.
- Subsistema de cobro de servicios.

Figura 2 Diagrama del sistema propuesto para Tele-Pago.

El subsistema de almacenamiento de datos va a ser el encargado de almacenar, filtrar y proteger la integridad de los datos. Este subsistema va a contar con un gestor de bases de datos PostgreSQL y va a contener todas las bases de datos del sistema. El mismo es de vital importancia en el sistema de Tele-Pago pues va a ser el medio de comunicación entre el subsistema web de administración y el subsistema de cobro de servicios.

El subsistema web de administración va ser el sistema a través del cual el administrador va a gestionar el sistema de Tele-Pago. Este subsistema permitirá al administrador, adicionar nuevos mensajes de servicios a cobrar, mostrar una traza de las transacciones realizadas por los clientes y configurar los parámetros de conexión con el Asterisk.

El subsistema de cobro de servicios es el sistema mediante el cual los clientes van a pagar sus servicios a través del teléfono. Este subsistema va a utilizar Asterisk como central telefónica. El mismo va a contar con una IVR sencilla para que los clientes puedan navegar por sus opciones sin temor a perderse en un menú muy complicado.

2.4 Características del Sistema.

2.4.1 Características Funcionales del Sistema.

Significan las necesidades de los clientes para resolver un determinado problema o simplemente alcanzar el objetivo trazado.

Las características funcionales identificadas por el cliente fueron:

Subsistema web de administración.

- Adicionar Mensaje.
- Eliminar Mensaje.
- Mostrar Mensaje.
- Configurar Conexión.
- Mostrar Transacción.

Subsistema de cobro de servicios.

- Reproducir Mensaje.
- Validar y Enviar datos al banco.

2.4.2 Características no Funcionales del Sistema.

Características de apariencia o interfaz externa.

Subsistema web de administración: Los colores deben ser agradables y contrastantes, el contenido fácil de manipular y la navegación organizada y amena.

Subsistema de cobro de servicios: El menú de la IVR debe ser sencillo y claro para que los clientes puedan efectuar el pago navegando de forma organizada y amena.

Características de usabilidad.

Subsistema web de administración: El administrador del sistema tendrá un amplio acceso a toda la información que contiene la aplicación. Además, podrá interactuar y hacer uso del sistema con gran facilidad, teniendo la posibilidad de ver toda la información brindada por el sistema en el desarrollo del proceso.

Subsistema de cobro de servicios: Este subsistema debe ser accesible para todos los clientes que tengan a su mano un teléfono. Los clientes deben ser informados en todo momento de las acciones que van realizando en la IVR.

Características de seguridad.

La seguridad en este sistema tiene una extraordinaria importancia, por lo que se tuvieron en cuenta los aspectos de la confidencialidad y la disponibilidad. Esto quiere de decir que solo los usuarios autenticados tienen acceso a la información, la cual debe estar disponible siempre que estos estén autenticados. Los mecanismos para lograr la seguridad necesaria del sistema no deben ser un impedimento para acceder a la información.

Los tres subsistemas requieren de autenticación para su uso, lo que quiere decir que un usuario no autenticado no tendrá acceso a ningún tipo de información.

También como característica de seguridad muy importante está la comunicación entre los subsistemas, la cual se va a retomar más adelante. La comunicación con el banco es un punto clave en la seguridad del sistema por lo que se propondrán algunas técnicas para garantizar dicha seguridad.

Características de software.

Subsistema de almacenamiento de datos.

- PostgreSQL 8.3 ó superior e inferior a 8.4 (Recomendado PostgreSQL 8.3.9).
- SQL Estándar.
- Lenguaje de procedimientos para bases de datos en Postgres PL/PGSQL.

Subsistema web de administración.

- Firefox 3.0 ó superior (Recomendado Firefox 3.6).
- PHP 5.2.3 ó superior.
- Extensión de PHP para el trabajo con imágenes (php5_gd).
- Extensión de PHP para el trabajo con JSON (php5_json).
- Extensión de PHP para la transformación de formatos (php5_xsl).
- Extensión de PHP para conexión entre servidores distintos (php5_curl).
- Extensión de PHP para el uso de servicios web (php5_soap).
- Extensión de PHP para el acceso a bases de datos (php5_pdo).
- Extensiones de PHP para el acceso a bases de datos en PostgreSQL (php5_pgsql, php5_pdo_pgsql).
- Servidor Apache 2.0

Subsistema de cobro de servicios.

- Asterisk 1.6 o superior (se recomienda 1.6.2.10)
- PHP 5.1.6 o superior (se recomienda 5.1.6)
- Extensiones de PHP para el acceso a bases de datos en PostgreSQL (php5_pgsql)
- PHP AGI

- Sistema Operativo Elastix

Características de hardware.

Subsistema de almacenamiento de datos.

- El servidor de bases de datos puede estar alojado en el mismo host que el servidor web de administración para ahorrar recursos, puesto que el sitio de administración va a tener poca actividad. En caso de no ser así debería cumplir las mismas características del servidor web.

Subsistema web de administración.

- En el cliente se requiere una máquina con 256 MB[18] de RAM[19] como mínimo, un procesador igual o mayor a 1.40 GHZ[20] y una tarjeta de red de 100 MB/S[21] o mayor.
- El servidor debe tener un procesador igual o mayor a 1.40 GHZ, 1 GB[22] de RAM, 160 GB de disco duro mínimo y una tarjeta de red de 100 MB/S o mayor. También deben de tener respaldo energético y se debe almacenar en un lugar con restricciones de acceso por seguridad.

Subsistema de cobro de servicios.

- El servidor Asterisk hace un uso intensivo del CPU para el tratamiento de la voz y la reproducción de los mensajes, por lo que se recomienda un “Quad-Core Intel® Xeon® Processor E5440 (2.83GHz, 8M Cache)”.
- Debe tener 4 GB de RAM, 160 GB de disco duro mínimo y una tarjeta de red de 100 MB/S o mayor. También deben de tener respaldo energético y se debe almacenar en un lugar con restricciones de acceso por seguridad.

Para definir las características no funcionales del sistema, tanto de software como se hardware, se tuvo en cuenta las características planteadas en los documentos:

[18] MB: MegaBytes.
[19] RAM: Random Access Memory/ Memoria de Acceso Aleatorio.
[20] GHZ: Gigahertz.
[21] MB/S: MegaBytes por Segundo.
[22] GB: Gigabytes.

- PROPUESTA TÉCNICO FINANCIERA Servicio Atención Telefónica a Emergencias (171) Sub-proyecto Call Center. (41)
- Manual de instalación de las herramientas para Sauxe 1.5. (26)

2.5 Modelos de procesos del negocio.

En este epígrafe se muestran los modelos de procesos del negocio para cada subsistema**.**

Subsistema web de administración.

Los procesos del negocio de este subsistema se dividen en tres procesos fundamentales, los cuales son: gestionar los mensajes, configurar la conexión y mostrar las transacciones realizadas.

El proceso del negocio de gestionar los mensajes comienza cuando el administrador selecciona entre las opciones adicionar un mensaje y eliminar un mensaje. En caso de que se seleccione adicionar, el sistema le solicita los datos necesarios para guardar el nuevo mensaje. El administrador introduce los datos y el sistema valida que sean correctos, de no ser correctos, el sistema muestra un mensaje de error. Si los datos introducidos son válidos para la aplicación, esta los envía y guarda en el servidor Asterisk y en el servidor de Base de Datos, además muestra el mensaje en la tabla de mensajes con sus datos. En el caso de que se elija la opción de eliminar un mensaje, el administrador selecciona el mensaje a eliminar, el sistema muestra un mensaje alertando que se va a eliminar el mensaje y espera la reafirmación del administrador sobre la eliminación de ese mensaje, en caso de ser positiva, el mensaje es eliminado. Para un mejor entendimiento se recomienda ver la figura 3.

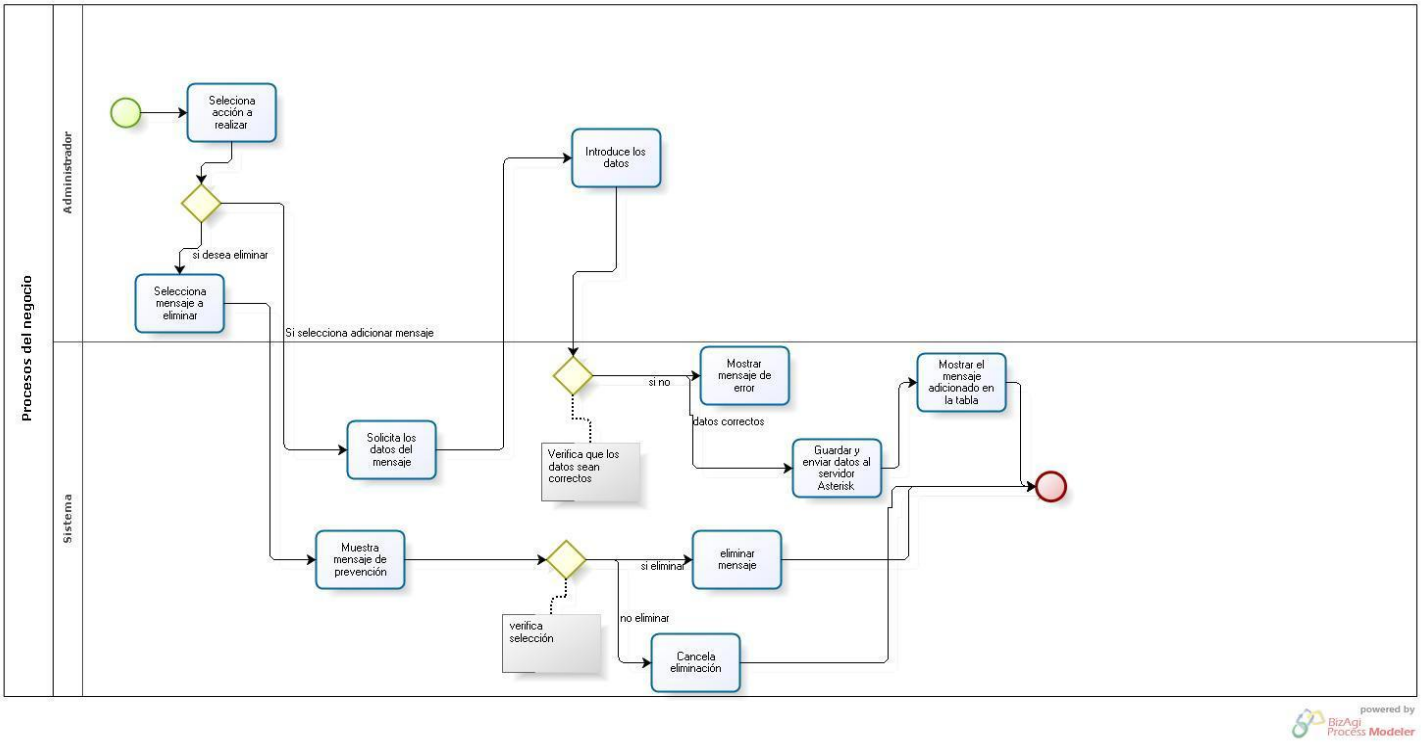

Figura 3 Diagrama de procesos del negocio (Gestión de mensajes).

El proceso de Configurar Conexión se inicia cuando el administrador decide modificar la configuración de la conexión con el servidor Asterisk. El sistema solicita los datos necesarios para modificar la configuración, el administrador introduce los datos y el sistema valida los datos entrados. De ser correctos la aplicación muestra un mensaje satisfactorio y guarda la nueva configuración, en caso de que los datos introducidos no sean válidos para el sistema, este muestra un mensaje de error y sigue solicitando los datos. Ver Figura 4.

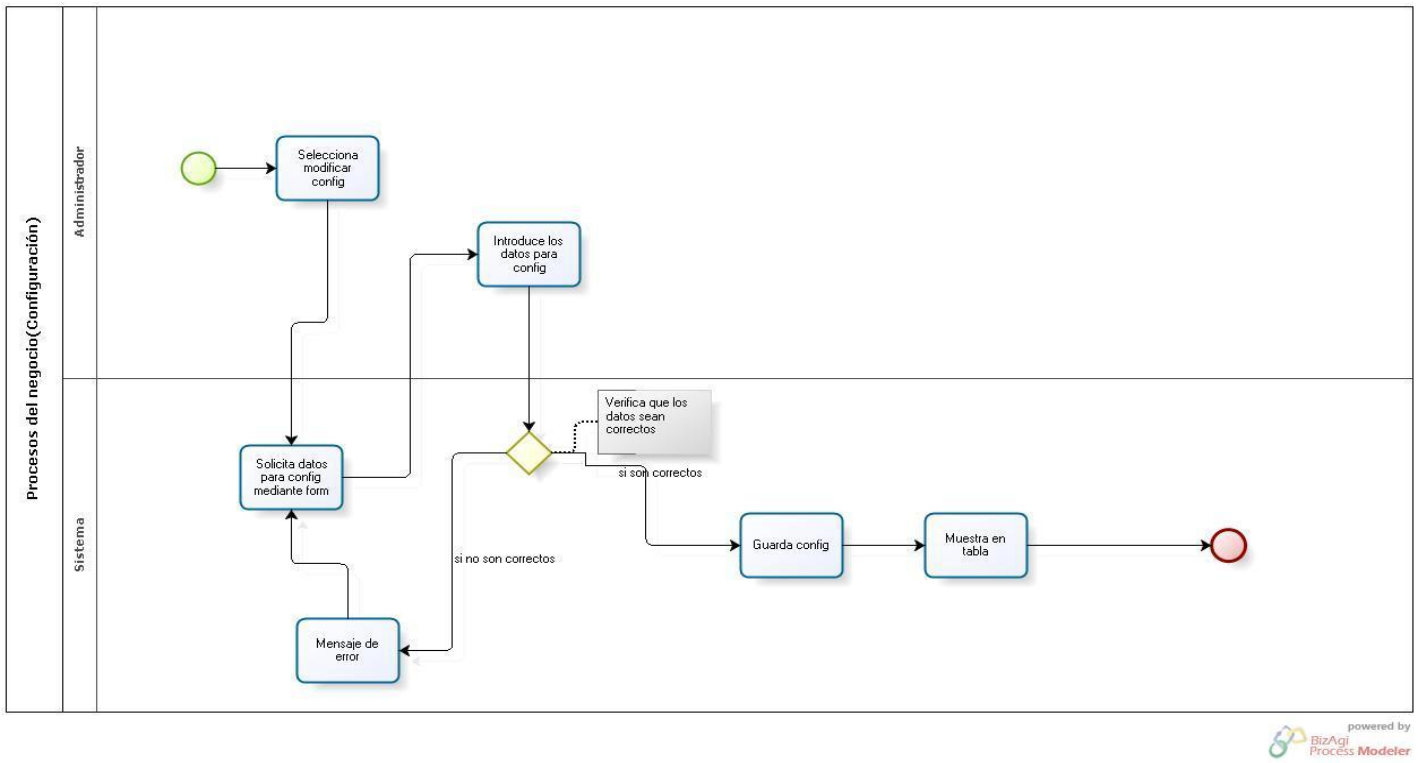

Figura 4 Diagrama de procesos del negocio (Configurar Conexión).

El proceso de mostrar las transacciones realizadas se inicia cuando el administrador escoge la opción de ver las transacciones. El sistema muestra todas las transacciones realizadas y permite también al administrador la opción de eliminar las trazas de transacciones que desee. Si el administrador decide eliminar una de estas el sistema le muestra un mensaje de advertencia y solicita una confirmación antes de eliminar la transacción. Si la respuesta del administrador es positiva la transacción es eliminada completamente del sistema, en caso negativo no se realizaría nada.

Subsistema de cobro de servicios.

Los procesos del negocio se inician con la llamada del cliente a la IVR, esta responde con un mensaje de bienvenida y luego solicita el número de la tarjeta. El cliente introduce el número de su tarjeta, seguidamente el sistema valida la entrada de datos y los envía hacia el banco para verificar la existencia de esa tarjeta. En caso de que los datos enviados no sean válidos el sistema solicita la

entrada de un número de tarjeta hasta una cantidad de tres intentos. Por otra parte, si la respuesta del banco es satisfactoria, el sistema solicita el número de ping, valida su entrada y lo envía al banco para comprobar que sea el ping de la tarjeta anteriormente introducida. En caso de que el número de ping sea incorrecto la IVR reproduce un mensaje de error y vuelve a solicitar el número de ping. Una vez que el usuario esté correctamente autenticado, la IVR reproduce los servicios que el banco brinda para que sean pagados por vía telefónica. Seguidamente el cliente selecciona el servicio a pagar, los datos son enviados al banco donde se comprobará: si el cliente tiene ese servicio contratado, si tiene saldo para pagarlo, además de que si la respuesta de las acciones mencionadas anteriormente es satisfactoria, finalmente, se realizará la transacción, en caso de que no sea así el sistema reproducirá un mensaje de error y volverá a reproducir los mensajes asociados a los servicios. Luego de que la transacción sea realizada la IVR preguntará si el cliente desea pagar otro servicio. Para un mejor entendimiento se recomienda ver el diagrama de procesos del negocio que se muestra a continuación en la figura 5.

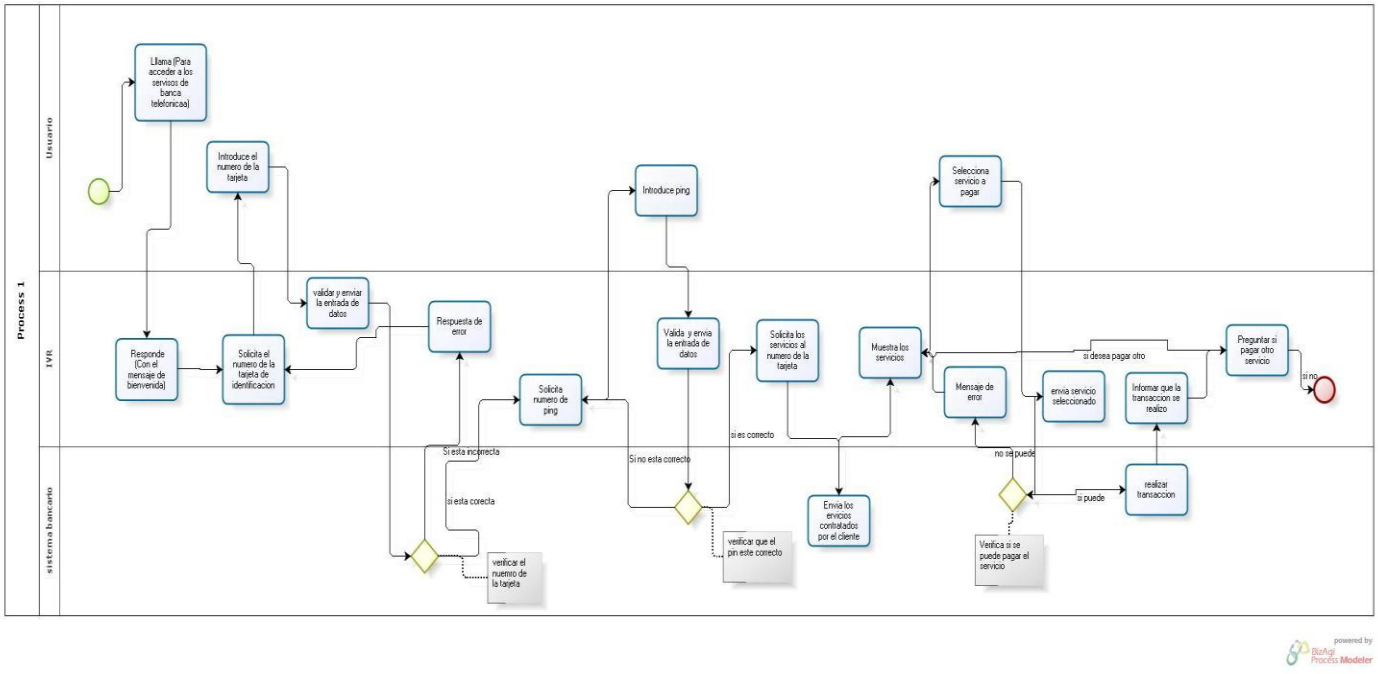

Figura 5 Diagrama de procesos del negocio (Subsistema de cobro de servicios).

2.6 Arquitectura del Sistema.

En los inicios de la informática, la programación se consideraba un arte y se desarrollaba como tal. Con el tiempo se han ido desarrollando formas y guías generales. A estas, se les ha denominado Arquitectura de Software, ya que indican la estructura, funcionamiento e interacción entre las partes del software. (42)

2.6.1 Arquitectura Cliente–Servidor.

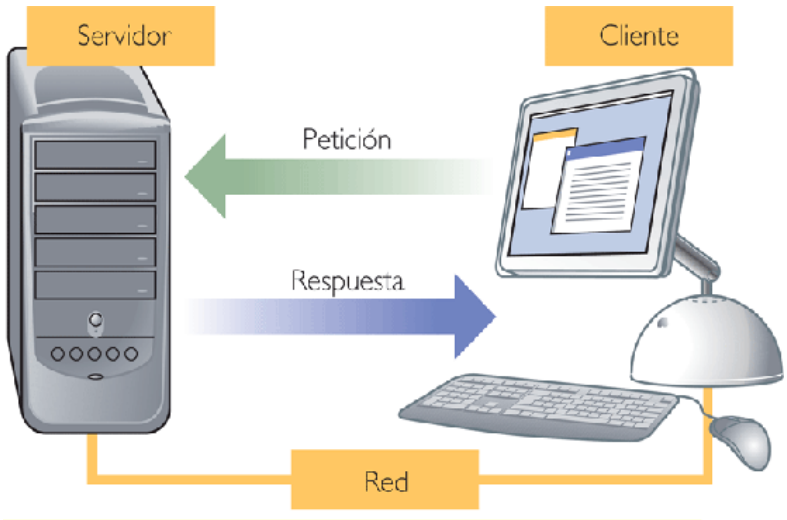

Figura 6 Arquitectura Cliente-Servidor.

La arquitectura cliente-servidor consiste básicamente en un cliente que realiza peticiones a otro programa (el servidor) que le da respuesta. Aunque esta idea se puede aplicar a programas que se ejecutan sobre una sola computadora es más ventajosa en un sistema operativo multiusuario distribuido a través de una red de computadoras.

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

La separación entre cliente y servidor es una separación de tipo lógico, donde el servidor no se ejecuta necesariamente sobre una sola máquina ni es necesariamente un sólo programa. Los tipos específicos de servidores incluyen los servidores web, los servidores de archivo, los servidores de correo, etc. Mientras que sus propósitos varían de unos servicios a otros, la arquitectura básica seguirá siendo la misma.

Una disposición muy común son los sistemas multicapa, en los que el servidor se descompone en diferentes programas que pueden ser ejecutados por diferentes computadoras aumentando así el grado de distribución del sistema. (43)

Ventajas.

La arquitectura Cliente-Servidor sustituye a la arquitectura monolítica en la que no hay distribución, tanto a nivel físico como a nivel lógico.

- Centralización del control: los accesos, recursos y la integridad de los datos son controlados por el servidor de forma que un programa cliente defectuoso o no autorizado no pueda dañar el sistema.
- Escalabilidad: se puede aumentar la capacidad de clientes y servidores por separado. Cualquier elemento puede ser aumentado (o mejorado) en cualquier momento, o se pueden añadir nuevos nodos a la red (clientes y/o servidores).
- Fácil mantenimiento: al estar distribuidas las funciones y responsabilidades entre varios ordenadores independientes, es posible reemplazar, reparar, actualizar, o incluso trasladar un servidor, mientras que sus clientes no se verán afectados por ese cambio (o se afectarán mínimamente). Esta independencia de los cambios también se conoce como encapsulación.

- ☎ Existen tecnologías, suficientemente desarrolladas, diseñadas para el paradigma de C/S que aseguran la seguridad en las transacciones, la amigabilidad de la interfaz, y la facilidad de empleo. (43)

Por todo lo antes planteado se decidió utilizar la arquitectura Cliente-Servidor para el desarrollo del sistema. Esta arquitectura se ve reflejada en todos los subsistemas que integran la solución propuesta. Por ejemplo en el subsistema web de administración el cliente será la página web con la que interactúa el administrador del sistema mediante el navegador Firefox, esta está implementada con código HTML y JavaScript utilizando el framework EXTJS. Este cliente hace las peticiones al servidor Apache donde está implementado en PHP la parte servidora de la aplicación. En el subsistema de almacenamiento de datos los clientes serán los subsistemas web de administración y el subsistema de cobro de servicios, los cuales realizarán peticiones al servidor de Base de Datos PostgreSQL. En el subsistema de cobro de servicios los clientes serán los teléfonos que en este caso son los que realizarán peticiones al servidor Asterisk.

2.6.2 Propuesta de la arquitectura del sistema.

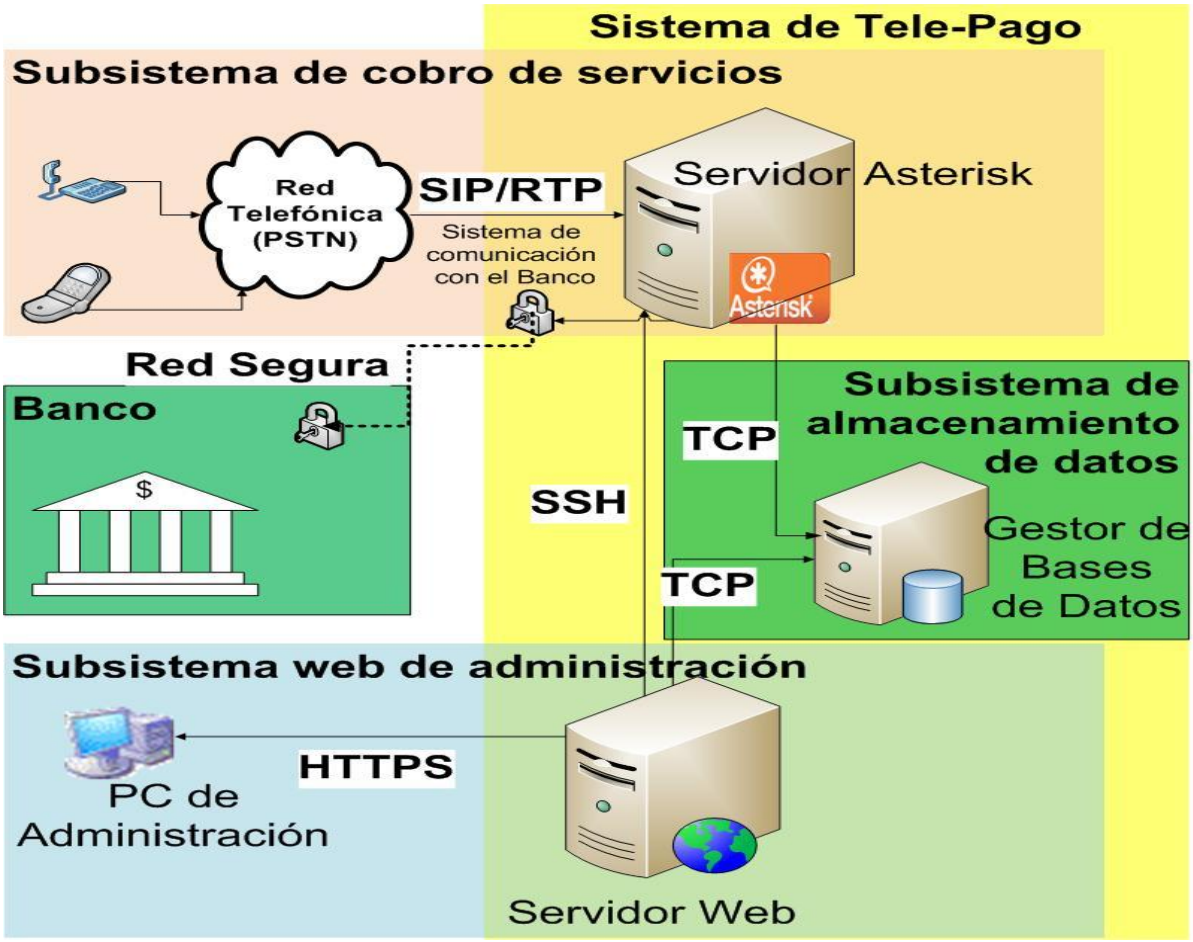

Figura 7 Propuesta de la arquitectura del sistema.

2.6.3 Funcionamiento de la herramienta sobre la arquitectura propuesta.

Como se muestra en la figura 7 el subsistema web de administración es un sistema cliente- servidor que utiliza el protocolo HTTPS[23] , que es un protocolo destinado a la transferencia segura de datos de Híper Texto, es decir, es la versión segura de HTTP. Es necesario el uso de HTTPS ya que la seguridad es un

[23] HTTPS: Protocolo seguro de transferencia de hipertexto.

punto clave en la solución propuesta. Este subsistema web es el encargado de gestionar toda la configuración y mostrar las todas las transacciones que los usuarios han efectuado vía telefónica. Cuando el administrador adiciona un nuevo mensaje los datos referentes al mensaje se almacenan en el subsistema de almacenamiento de datos, este proceso se hace a través del protocolo de transporte TCP[24], el cual garantiza que los datos sean entregados a su destino sin tener errores. Los ficheros de audio por ser de mayor tamaño son copiados directamente al subsistema de cobro de servicios. Para esto se utiliza el protocolo SSH[25], que es un protocolo donde la información viaja encriptada, permitiendo copiar los datos de forma segura.

En el subsistema de cobro de servicios los clientes hacen una llamada telefónica desde de la PSTN[26] que debe conectarse con el Asterisk a través de un Media Gateway que cumple con la función de ofrecer conectividad y traducción entre dos redes diferentes usando el protocolos SIP[27], el cual permite el establecimiento de sesiones entre dos o más usuarios, y el protocolo RTP[28], que es utilizado para la transmisión de información en tiempo real, como por ejemplo audio (44). Esta llamada va a ser contestada por el subsistema de cobro de servicios, el cual busca en el gestor de bases de datos del subsistema almacenamiento de datos mediante el protocolo TCP, los datos almacenados por el subsistema web de administración; con estos datos se reproducen los mensajes que tiene guardado el servidor Asterisk. Una vez culminada la transacción, es guardada en el gestor de bases de datos, para su posterior revisión por el administrador desde el sistema web de administración.

Para la conexión entre el banco y el subsistema de cobro de servicios la seguridad es un punto de vital importancia, por lo que se proponen la utilización de los siguientes mecanismos de seguridad:

- Como primer mecanismo de seguridad establecer una conexión de red segura, implantando una VPN[29] entre el Banco y el subsistema de cobro de servicios. Esta conexión permitirá verificar la identidad de los usuarios y restringir su acceso a aquellos que no se encuentren autorizados.

[24] TCP: Protocolo de Control de Transmisión.
[25] SSH: Secure Shell/ en español: intérprete de órdenes segura.
[26] PSTN: Red Telefónica Pública, Conmutada.
[27] SIP: Protocolo de Inicio de Sesiones.
[28] RTP: Real-time Transport Protocol /en español Protocolo de Transporte de Tiempo real.
[29] VPN: Red Privada Virtual.

Los datos enviados por esta conexión de red estarán cifrados para que no puedan ser leídos y firmados para verificar que los mismos no han sido alterados. (45)

- Como segundo mecanismo de comunicación y seguridad en el banco se plantea la creación de un servicio web utilizando SSL[30]. Este servicio publicará todas las funcionalidades que necesita el subsistema de cobro de servicios.

2.6.4 Estilo Arquitectónico.

Un estilo arquitectónico define las reglas generales de organización en términos de un patrón y las restricciones en la forma y la estructura de un grupo numeroso y variado de sistemas de software. En una forma más específica, un estilo determina el vocabulario de componentes y conectores que pueden ser utilizados en instancias de este estilo, con un conjunto de restricciones en las descripciones arquitectónicas.

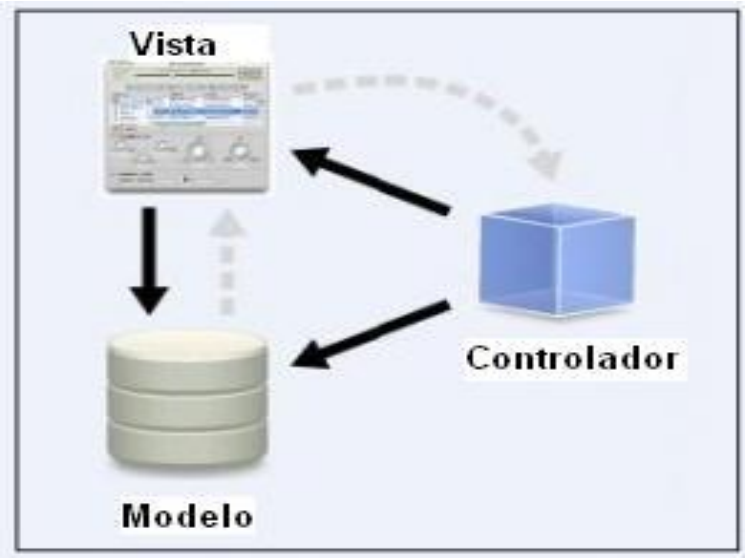

Figura 8 Estilo Arquitectónico Modelo-Vista-Controlador.

[30] SSL: Secure Socket Layer.

Modelo Vista Controlador (MVC) es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón de llamada y retorno MVC, se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página. El modelo es el responsable de acceder a la capa de almacenamiento de datos, y el controlador es el responsable de recibir los eventos de entrada desde la vista.

- **Modelo**: es la representación específica de la información con la cual el sistema opera. En resumen, el modelo se limita a lo relativo de la vista y su controlador facilitando las presentaciones visuales complejas. El sistema también puede operar con más datos no relativos a la presentación, haciendo uso integrado de otras lógicas de negocio y de datos afines con el sistema modelado.
- **Vista**: presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.
- **Controlador**: responde a eventos, usualmente acciones del usuario, e invoca peticiones al modelo y, probablemente, a la vista. (46)

Ventajas del MVC.

Este estilo arquitectónico presenta varias ventajas:

- Hay una clara separación entre los componentes de un programa; lo cual nos permite implementarlos por separado.
- Hay un API[31] muy bien definido; cualquiera que use el API, podrá reemplazar el Modelo, la Vista o el Controlador, sin aparente dificultad.
- La conexión entre el Modelo y sus Vistas es dinámica; se produce en tiempo de ejecución, no en tiempo de compilación. (46)

[31]Interfaz de Programación de Aplicaciones.

Zend framework propone una clara abstracción entre la vista y el modelo por lo tanto no existirá una comunicación directa entre estas, siendo la controladora la encargada de establecer la comunicación entre ellas. Para un mejor entendimiento se muestra en la figura 9 un ejemplo de cómo Zend framework utiliza MVC.

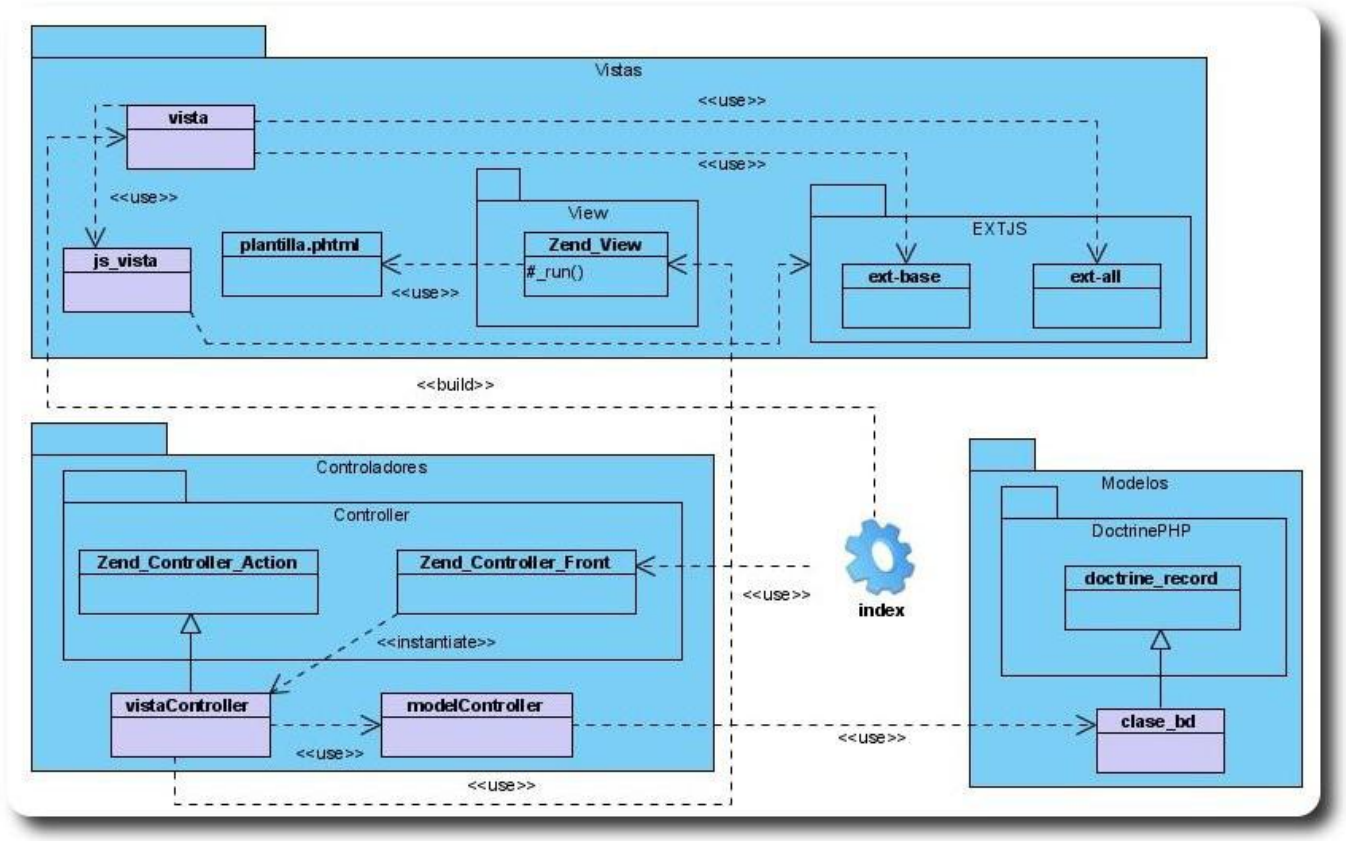

Figura 9 Vista de Gestión de Modelo Genérico para MVC (47).

Los paquetes View y Controller son los paquetes incluidos por defecto en la carpeta library del Zend Framework.

2.7 Personal Relacionado con el Sistema.

Se define como personal relacionado con el sistema a todas aquellas personas que realizan una función o interactúan con el sistema de una forma u otra.

Tabla 1 Personal Relacionado con el Sistema.

Personas relacionadas con el sistema	**Descripción**
Administrador	Es la persona facultada para la gestión del sistema en general. Es el encargado de agregar el mensaje asociado a un nuevo servicio a cobrar.
Cliente	Es la persona que interactúa con la IVR para realizar el pago de sus servicios.

2.8 Fase de Exploración.

"La metodología de desarrollo XP comienza con la fase de exploración. Durante esta se realiza el proceso de identificación de las Historias de Usuario, así como la familiarización de los equipos de trabajo con las tecnologías y herramientas seleccionadas para la construcción del proyecto." (48)

2.8.1 Historias de Usuario (HU).

"Las Historias de Usuario tienen el mismo propósito que los casos de uso, pero no son lo mismo. Las escriben los propios clientes, tal y como ven ellos las necesidades del sistema". (49) Por tanto, serán descripciones cortas y escritas en el lenguaje del usuario, sin terminología técnica.
A continuación la plantilla que se usará para definir las HU.

Se decidió tomar como 1 semana, 5 días laborables, trabajando 8 horas diarias, lo cual suma las 40 horas de trabajo semanal que propone XP.

Tabla 2 Historias de Usuario.

Historia de Usuario	
Número: (*Número de la historia de usuario incremental en el tiempo).*	**Usuario:** *(sistema que protagoniza la historia)*
Nombre Historia de Usuario: (*El nombre de la historia de usuario sería para identificarlas mejor entre los desarrolladores y el cliente).*	
Prioridad en Negocio: (*La importancia que tiene para el cliente*). (Alta / Media / Baja).	**Riesgo en Desarrollo:** (*Qué tan difícil es para el desarrollador*). (Alto / Medio / Bajo).
Puntos asignados: (*El tiempo que se demorará al desarrollo de la HU).*	**Iteración Asignada:** (*La iteración (liberación en nuestro proceso) a la que corresponde).*
Programador responsable: (*Persona responsable de la programación de la funcionalidad que contiene dicha historia de usuario*).	
Descripción: (Se e*specificará las operaciones del usuario y las respuestas que dará el sistema).*	
0bservaciones: (*Algunas observaciones de interés, como glosario, información sobre usuarios, etc.).*	

Tabla 3 Historia de Usuario: Adicionar Mensajes.

Historia Usuario	
Número: 1.	**Usuario:** Administrador.
Nombre Historia de Usuario: Adicionar Mensajes.	
Prioridad del Negocio: Medio.	**Riesgo de Desarrollo:** Medio.
Puntos asignados: 1/5.	Iteración Asignada: 2.
Programador responsable: Julio César Torres Macías	
Descripción: Se inicia cuando el administrador introduce los mensajes que van a ser guardados y posteriormente reproducidos por la IVR.	
Observaciones:	

Tabla 4 Historia de Usuario: Mostrar Mensajes.

Historia Usuario	
Número: 2.	**Usuario:** Administrador.
Nombre Historia de Usuario: Mostrar Mensajes.	
Prioridad del Negocio: Medio.	**Riesgo de Desarrollo:** Medio.
Puntos asignados: 1/5.	**Iteración Asignada:** 2.
Programador responsable: Julio César Torres Macías.	
Descripción: El sistema debe mostrar los mensajes introducidos por el administrador.	
Observaciones:	

Tabla 5 Historia de Usuario: Eliminar Mensajes.

Historia Usuario	
Número: 3.	**Usuario:** Administrador.
Nombre Historia de Usuario: Eliminar Mensajes.	
Prioridad del Negocio: Medio	**Riesgo de Desarrollo:** Medio.
Puntos asignados: 1/5.	**Iteración Asignada:** 2.
Programador responsable: Julio César Torres Macías.	
Descripción: El sistema debe eliminar los mensajes que sean seleccionados por el administrador.	
Observaciones:	

Tabla 6 Historia de Usuario: Reproducir Mensajes.

Historia Usuario.	
Número: 4.	**Usuario:** Todas las personas.
Nombre Historia de Usuario: Reproducir Mensaje.	
Prioridad del Negocio: Medio.	**Riesgo de Desarrollo:** Medio.
Puntos asignados: 1.	**Iteración Asignada:** 1.

Programador responsable: Julio César Torres Macías.
Descripción: La IVR debe ser capaz de cargar y reproducir los mensajes de los servicios que ofrece el banco almacenados con anterioridad.
Observaciones:

Tabla 7 Historia de Usuario: Validar y Enviar los datos al banco.

Historia Usuario.	
Número: 5.	**Usuario:** Todas las personas.
Nombre Historia de Usuario: Validar y Enviar los datos al banco.	
Prioridad del Negocio: Alta.	**Riesgo de Desarrollo:** Alto.
Puntos asignados: 2.	**Iteración Asignada:** 1.
Programador responsable: Julio César Torres Macías.	
Descripción: Se inicia cuando el usuario introduce los datos, el sistema valida la entrada de los mismos y se los envía al banco para que este los verifique.	
Observaciones:	

Tabla 8 Historia de Usuario: Configurar Conexión.

Historia Usuario.	
Número: 6.	**Usuario:** Administrador.
Nombre Historia de Usuario: Configurar Conexión.	
Prioridad del Negocio: Media.	**Riesgo de Desarrollo:** Medio.
Puntos asignados: 4/5.	**Iteración Asignada:** 2.
Programador responsable: Julio César Torres Macías.	
Descripción: Se inicia cuando el administrador del banco introduce los datos que se utilizarán para realizar la conexión con Asterisk.	
Observaciones:	

Tabla 9 Historia de Usuario: Mostrar Transacción.

Historia Usuario	
Número: 7.	**Usuario:** Administrador.
Nombre Historia de Usuario: Mostrar Transacción.	
Prioridad del Negocio: Media.	**Riesgo de Desarrollo:** Medio.
Puntos asignados: 3/5.	**Iteración Asignada:** 2.
Programador responsable: Julio César Torres Macías.	
Descripción: Se inicia cuando el administrador solicita las transacciones realizadas por el cliente y el sistema muestra todo lo realizado.	
Observaciones:	

2.9 Fase de Planificación.

"Durante esta fase se realiza una estimación del esfuerzo que costará implementar cada historia de usuario. Esta estimación 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 interrupción alguna." (42) Esta fase permite enmarcar el alcance del proyecto, que tiene como fin el desarrollo y la entrega del sistema requerido.

2.9.1 Estimación del esfuerzo.

Para determinar la velocidad del proyecto y poseer una guía a la cual ajustarse se realiza una estimación del esfuerzo para cada una de las Historias de Usuarios previamente identificadas.

Tabla 10 Estimación del esfuerzo.

No	**Historias de Usuarios.**	**Puntos estimados (en días).**
1	Adicionar Mensajes.	1/5
2	Mostrar Mensajes.	1/5
3	Eliminar Mensajes.	1/5

4	Reproducir Mensajes.	1
5	Validar y enviar datos al banco.	2
6	Configurar conexión.	4/5
7	Mostrar Transacción.	3/5

2.9.2 Plan de Iteraciones.

Una vez identificadas las Historias de Usuario del sistema y estimado el esfuerzo dedicado a la realización de cada una de estas, se procede a definir qué Historia de usuario será implementada para cada iteración, que no es más que la planificación de la etapa de implementación del sistema.

Es importante vigilar la velocidad del proyecto y el movimiento de Historias de Usuario. Puede ser necesario volver a calcular las Historias de Usuario y negociar el plan de entrega de tres a cinco iteraciones. Debido a que la implementación de las Historias de Usuario más importantes para el cliente se realizará siempre, se hará lo máximo posible por él y la dirección.

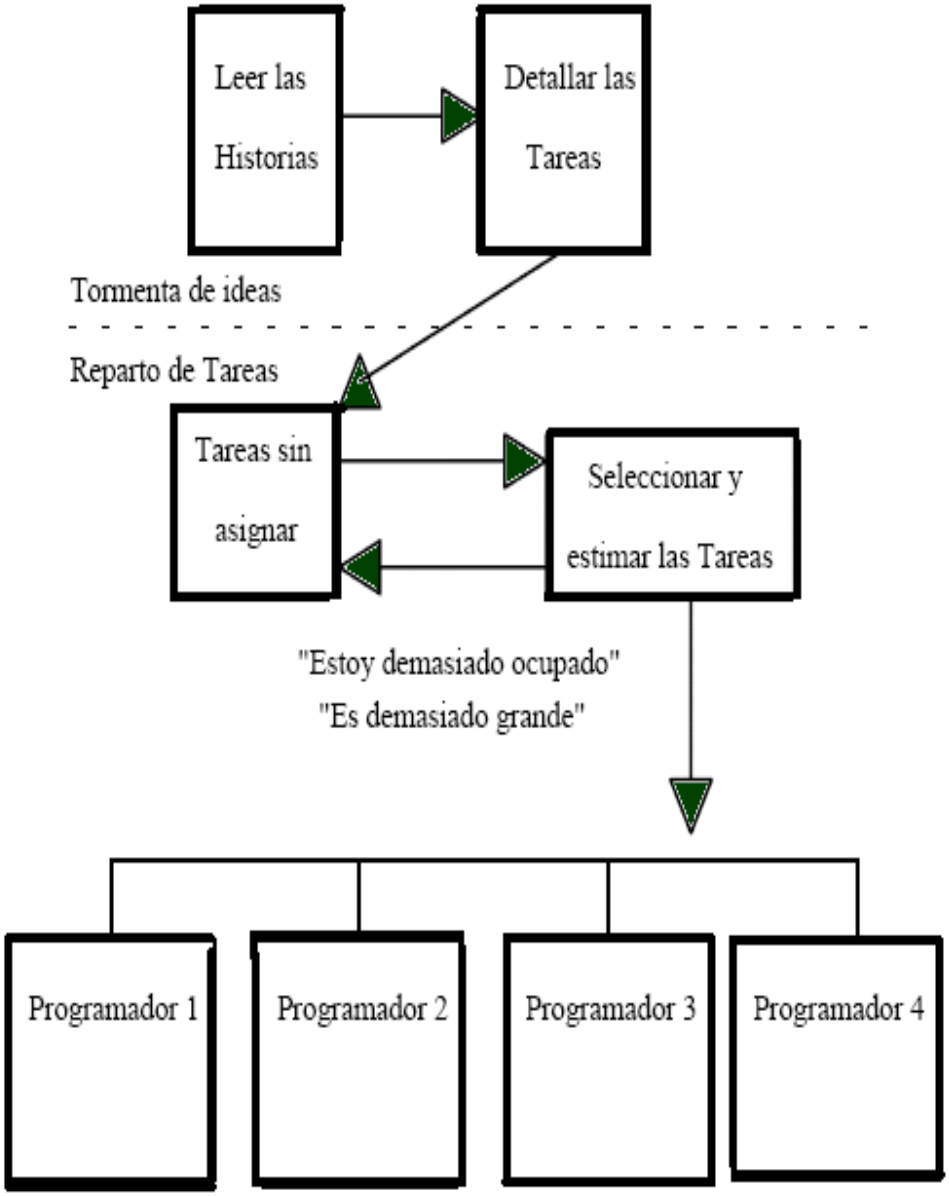

Figura 10 Plan de iteración. (50)

Iteración 1: En esta iteración se entregarán las Historias de Usuario 4 y 5 debido a que tiene mayor prioridad para el cliente las cuales son:

- ☎ Validar y enviar datos al banco.

- Reproducir mensajes.

Iteración 2: En esta iteración se realizarán las siguientes Historias de Usuario que son importantes para el cliente siendo estas las 7, 6, 2, 3 y 1:

- Adicionar Mensajes.
- Mostrar Mensajes.
- Eliminar Mensaje.
- Configurar Conexión.
- Mostrar Transacción.

2.9.3 Plan de duración de Iteraciones.

Como parte del ciclo de vida de un proyecto utilizando la metodología XP se crea el plan de duración de cada una de las iteraciones, según los equipos de desarrollo con que se cuente, en este caso se hace para el único equipo de desarrollo que se tiene. Este plan se encarga de mostrar las Historias de Usuario que serán complementadas en cada una de las iteraciones, así como la duración estimada de cada una y el orden en que se implementarán.

Tabla 11 Duración de las iteraciones.

Iteración.	Orden de Historia de Usuario a implementar.	Duración total de las Iteraciones.
1	Validar y enviar datos al banco. Reproducir Mensajes.	3 semanas.
2	Adicionar Mensajes. Mostrar Mensajes. Eliminar Mensajes. Configurar Conexión. Mostrar Transacción.	2 semanas.

2.9.4 Plan de entrega.

En el plan de entrega se definen las Historias de Usuario que se entregarán al final de cada iteración. Al finalizar la segunda iteración se obtendrá la primera versión del producto final.

Tabla 12 Plan de entrega.

Historias de Usuario.	**Final Iteración 1.**	**Final Iteración 2.**
Sistema para el pago de servicios vía telefónica.	V0.1	V 1.0

2.10 Conclusiones.

En este capítulo se han expuesto las principales características que debe poseer el sistema, para satisfacer las expectativas del cliente. Se exponen los procesos que serán objetos de automatización y la propuesta del sistema. Además, se generaron las HU que caracterizan al sistema, así como la durabilidad de cada una de ellas y los planes de iteraciones y entregas, donde se documentaron los artefactos generados.

CAPÍTULO 3: DISEÑO DEL SISTEMA.

3.1 Introducción.

En el presente capítulo se puntualiza todo lo referente al diseño del sistema, así como se generarán todos los artefactos pertenecientes a esta fase.

3.2 Tarjetas Cargo o clase, Responsabilidad y Colaboración (CRC).

El uso de las tarjetas C.R.C[32] permite al programador centrarse y apreciar el desarrollo orientado a objetos olvidándose de los malos hábitos de la programación clásica. Estas tarjetas representan objetos; la clase a la que pertenece el objeto se puede escribir en la parte de arriba de la tarjeta, en una columna a la izquierda se pueden escribir las responsabilidades u objetivos que debe cumplir el objeto y a la derecha, las clases que colaboran con cada responsabilidad. (49) XP propone tarjetas de cargo y no un diagrama de clases, pero la metodología tampoco lo excluye, por lo que se exponen ambos para un mayor entendimiento de las relaciones entre clases.

Diagrama de clases.

[32]CRC: Clase, Responsabilidades y Colaboración.

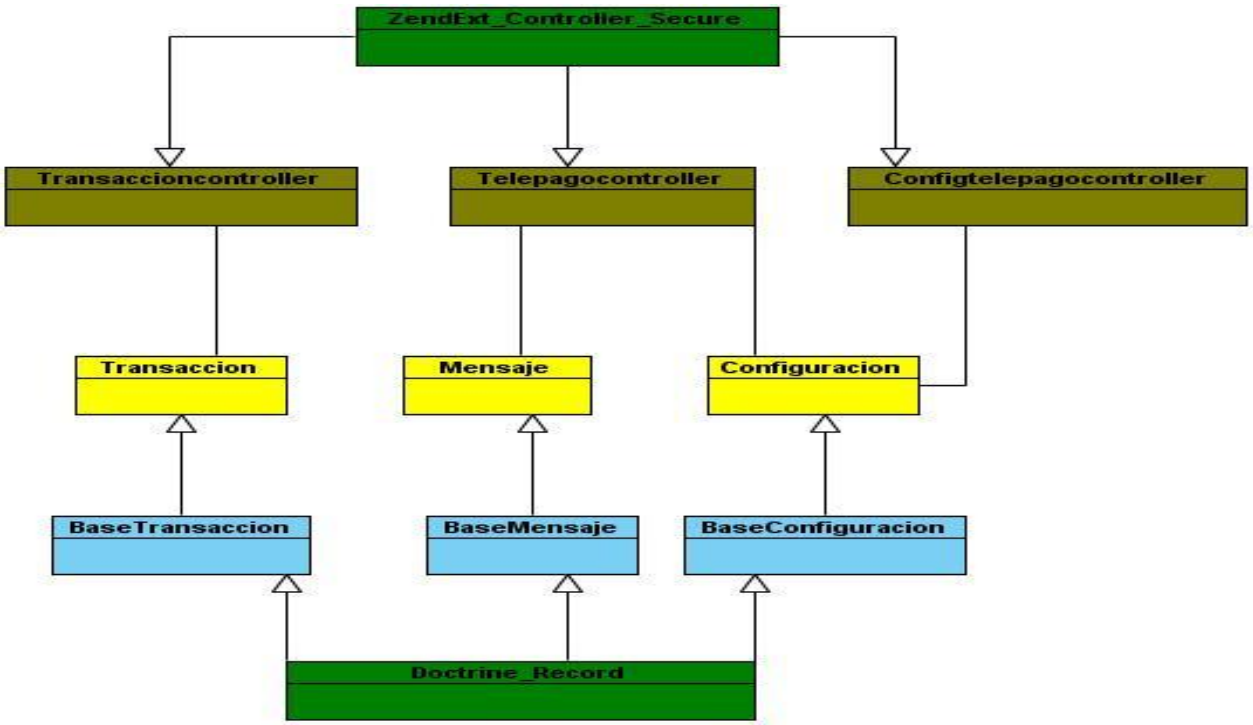

Figura 11 Diagrama de clases.

Tabla 13 Clase: telepagocontroller.

Responsabilidad.	**Clase Relacionada.**
Adicionar un mensaje. Eliminar un mensaje. Mostrar datos de los mensajes. Guardar mensajes en el servidor Asterisk.	Mensaje. Configuracion. ZendExt_Controller_Secure.

Tabla 14 Clase: Mensaje.

Responsabilidad.	**Clase Relacionada.**

Adicionar un mensaje. Eliminar un mensaje. Devolver todos los mensajes.	BaseMensaje.

Tabla 15 Clase: Transaccioncontroller.

Responsabilidad.	**Clase Relacionada.**
Mostrar la Transacción. Eliminar la Transacción.	Transaccion. ZendExt_Controller_Secure.

Tabla 16 Clase: Transaccion.

Responsabilidad.	**Clase Relacionada.**
Devolver la Transacción. Eliminar la Transacción.	BaseTransaccion.

Tabla 5 Clase: Configtelepagocontroller.

Responsabilidad.	**Clase Relacionada.**
Mostrar la configuración. Modificar la configuración.	Configuracion. ZendExt_Controller_Secure.

Tabla 6 Clase: Configuración.

Responsabilidad.	**Clase Relacionada.**
Devolver la configuración. Modificar la configuración.	BaseConfiguracion.

Tabla 7 Clase: BaseTransaccion.

Responsabilidad.	**Clase Relacionada.**

Clase en cargada de mapear la tabla transaccion.	Doctrine_Record.

Tabla 20 Clase: BaseMensaje.

Responsabilidad.	**Clase Relacionada.**
Clase en cargada de mapear la tabla mensaje.	Doctrine_Record.

Tabla 21 Clase: BaseConfiguracion.

Responsabilidad.	**Clase Relacionada.**
Clase en cargada de mapear la tabla configuracion.	Doctrine_Record.

3.3 Diseño de la Base de Datos.

"Una Base de Datos correctamente diseñada permite obtener acceso a información exacta y actualizada." (51)

El siguiente diseño que se presenta a continuación en la figura 12, muestra tres tablas, las cuales son:

Mensaje: es la tabla donde se almacenan todos los datos relacionados con los mensajes que serán reproducidos por la IVR. Estos datos se componen por un idmensaje el cual es un identificador para cada mensaje adicionado, un tipo que es el nombre del servicio al cual está asociado ese mensaje y una dirección que será donde estará la grabación para dicho mensaje.

Transacción: es la tabla donde se archivan todas las transacciones realizadas por el cliente. Posee un idtransaccion, el cual se utilizará para identificar la transacción, además del numerotarjeta que es la

tarjeta del cliente que realiza el pago, el atributo teléfono que es el número desde donde se hizo la llamada y la hora en que se realizó el pago.

Configuración: es la tabla en la cual se guarda la configuración para la conexión con el servidor Asterisk. Los atributos de esta son un id para la identificación, un usuario y una contraseña para la autenticación en el servidor, un IP para establecer la conexión y una dirección donde serán guardadas las grabaciones a reproducir.

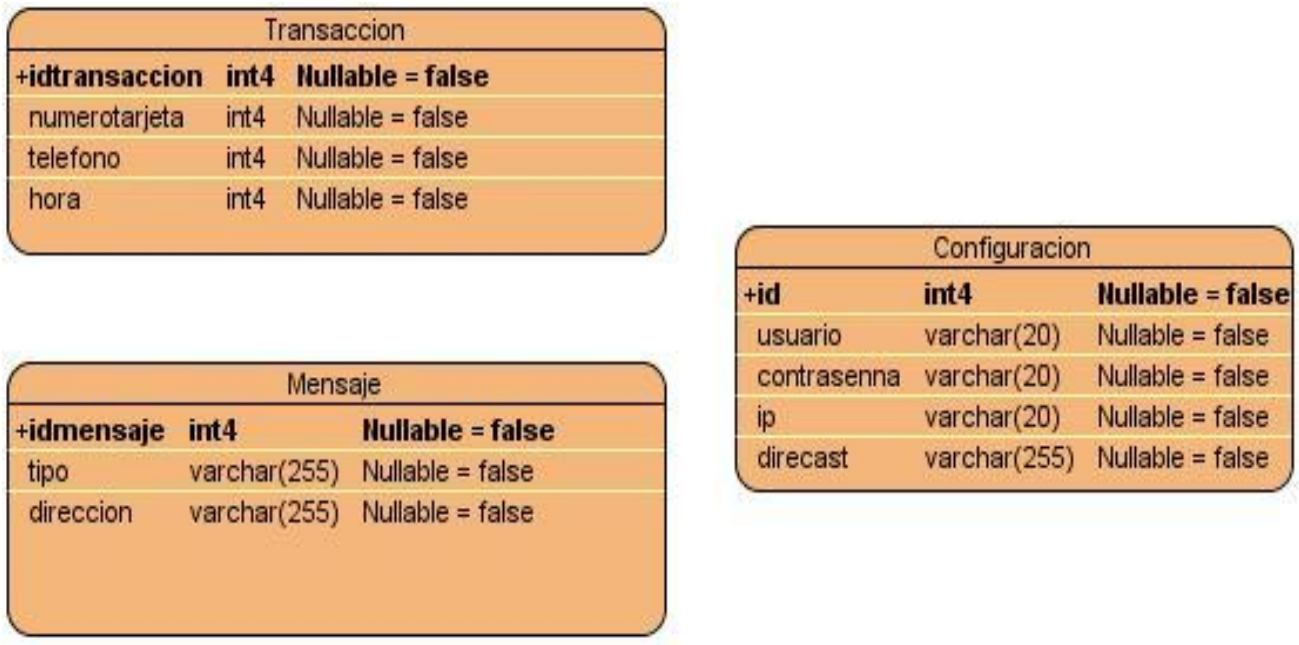

Figura 12 Modelo de Datos.

3.4 Patrones de Diseño.

Los patrones de diseño son vías para solucionar muchos problemas parecidos, de hecho, se centran en la parte usual del componente a diseñar y de esa forma son más específicos. Para lograr una mayor calidad en el diseño, se tuvieron en cuenta un conjunto de patrones los cuales proporcionan respuesta a un conjunto de problemas similares.

3.4.1 Patrones GRASP (Patrones de Software para la asignación General de Responsabilidad) que implementa Sauxe.

Creador: Asigna a una clase la responsabilidad de crear una instancia de otra., la creación de instancias es una de las actividades más comunes en un sistema orientado a objetos. En consecuencia, es útil contar con un principio general para la asignación de las responsabilidades de creación.

Controlador: Es un evento generado por actores externos. Se asocian con operaciones del sistema como respuestas a los eventos del sistema, tal como se relacionan los mensajes y los métodos.

Normalmente, un controlador delega en otros objetos el trabajo que se necesita hacer; coordina o controla la actividad. No realiza mucho trabajo por sí mismo.

Experto: La responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados (atributos).Una clase, contiene toda la información necesaria para realizar la labor que tiene encomendada.

Bajo Acoplamiento: El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases. Cada clase está acoplada a las clases estrictamente necesarias. Una clase con bajo (o débil) acoplamiento no depende de muchas otras. Una clase con alto (o fuerte) acoplamiento recurre a muchas otras.

Alta cohesión: La cohesión es una medida de la fuerza con la que se relacionan las clases y el grado de focalización de las responsabilidades de un elemento. Cada elemento del diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos. (52)

3.4.2 Patrones GOF (Gang-of-Four) que implementa Sauxe.

Fachada/Farcade: Proporciona una interfaz unificada de alto nivel que, representando a todo un subsistema, facilite su uso. La “fachada” satisface a la mayoría de los clientes, sin ocultar las funciones de menor nivel a aquellos que necesiten acceder a ellas.

Solitario /Singleton: Garantizar que una clase sólo tiene una única instancia, proporcionando un punto de acceso global a la misma.

Decorador/Decorator: Añade responsabilidades adicionales a un objeto dinámicamente, proporcionando una alternativa flexible a la especialización mediante herencia, cuando se trata de añadir funcionalidades. (53)

3.5 Conclusiones.

En este capítulo quedaron representados los resultados obtenidos en la fase de diseño del sistema, quedando definidas tanto las tarjetas CRC como el modelo de datos, y la descripción de los patrones de diseño utilizados.

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBAS.

4.1 Introducción.

"XP plantea que la implementación de un software se hace de forma iterativa, obteniendo al final de cada iteración un producto funcional que debe ser probado y mostrado al cliente para que los desarrolladores trabajen conociendo las opiniones dadas sobre el mismo". (48)En esta fase se efectúa la implementación de las HU que fueron seleccionadas para cada iteración. Como parte de este plan se crean tareas de programación para ayudar a organizar la implementación exitosa de las HU. A continuación se describen las tareas.

4.2 Tareas para cada Historia de Usuario.

Cada una de estas Historias de Usuario se transformará en tareas que serán desarrolladas por los programadores. Estas no son más que pasos lógicos a seguir por el programador para realizar la implementación de una HU. A continuación se detallan para cada una de las iteraciones las tareas a desarrollar por cada HU.

4.2.1 Iteración 1.

Tabla 22 Historias de Usuario de la iteración 1.

Historias de Usuario	Estimación	Real
Validar y enviar datos al banco.	2	2
Reproducir Mensajes.	1	1

Tabla 23 Tarea# 1 de la Historia de usuario: Reproducir mensajes.

Tarea.	
Número de la tarea: 1.	**Número de la Historia de Usuario:** 4.

Nombre de la tarea: Implementación de la funcionalidad para suministrar información.		
Tipo de tarea: Desarrollo.	**Fecha Inicio:** 10/02/11.	**Fecha fin:** 13/02/11.
Programador responsable: Julio César Torres Macías.		
Descripción: Se realiza la funcionalidad para reproducir los mensajes almacenados en la base de dato.		

Tabla 24 Tarea# 2 de la Historia de usuario: Validar y enviar datos al banco.

Tarea.		
Número de la tarea: 2.	**Número de la Historia de Usuario:** 5.	
Nombre de la tarea: Implementación de la funcionalidad para validar y enviar la información suministrada por el cliente.		
Tipo de tarea: Desarrollo.	**Fecha Inicio:** 13/02/11.	**Fecha fin:** 14/02/11.
Programador responsable: Julio César Torres Macías.		
Descripción: Se realiza la funcionalidad para obtener los datos que suministra el cliente y enviarlos al banco.		

4.2.2 Iteración 2.

Tabla 8 Historias de Usuario de la iteración 2.

Historias de Usuario.	**Estimación (en días).**	**Real (en días).**
Adicionar Mensajes.	1/5	1/5
Mostrar Mensajes.	1/5	1/5
Eliminar Mensajes.	1/5	1/5
Configurar Conexión.	4/5	4/5
Mostrar Transacción.	3/5	3/5

Tabla 26 Tarea# 3 de la Historia de usuario: Eliminar Mensaje.

Tarea.		
Número de la tarea: 3.	**Número de la Historia de Usuario:** 3.	
Nombre de la tarea: Implementación de la funcionalidad para Eliminar los mensajes.		
Tipo de tarea: Desarrollo.	**Fecha Inicio:** 20/02/11.	**Fecha fin:** 20/02/11.
Programador responsable: Julio César Torres Macías.		
Descripción: Se realiza la funcionalidad para eliminar los mensajes almacenados en la Base de Datos.		

Tabla 27 Tarea # 4 de la Historia de usuario: Configurar Conexión.

Tarea.		
Número de la tarea: 4.	**Número de la Historia de Usuario:** 3.	
Nombre de la tarea: Implementación de la interfaz de configuración.		
Tipo de tarea: Desarrollo.	**Fecha Inicio:** 25/02/11.	**Fecha fin:** 29/02/11.
Programador responsable: Julio César Torres Macías.		
Descripción: Se realiza la implementación de la interfaz que recogerá los datos para configurar la conexión con el servidor Asterisk.		

Tabla 28. Tarea# 5 de la Historia de usuario: Mostrar mensaje.

Tarea	
Número de la tarea: 5.	**Número de la Historia de Usuario:** 2.
Nombre de la tarea: Implementación de la funcionalidad para Mostrar los mensajes.	

Tipo de tarea: Desarrollo.	**Fecha Inicio:** 1/03/11.	**Fecha fin:** 2/03/11.
Programador responsable: Julio César Torres Macías.		
Descripción: Se realiza la funcionalidad para mostrar los mensajes almacenados en la base de dato.		

Tabla 29. Tarea# 6 de la Historia de usuario: Adicionar mensaje.

Tarea.		
Número de la tarea: 6.	**Número de la Historia de Usuario:** 1.	
Nombre de la tarea: Implementación de las funcionalidades de adicionar mensajes.		
Tipo de tarea: Desarrollo.	**Fecha Inicio:** 3/03/11.	**Fecha fin:** 4/03/11.
Programador responsable: Julio César Torres Macías.		
Descripción: Se realiza la implementación de las funcionalidades para adicionar los mensajes asociados a un servicio determinado.		

Tabla 30 Tarea# 7 de la Historia de usuario: Adicionar mensajes.

Tarea.		
Número de la tarea:7	**Número de la Historia de Usuario:**1	
Nombre de la tarea: Implementación de la interfaz para los mensajes.		
Tipo de tarea: Desarrollo	**Fecha Inicio:** 5/03/11.	**Fecha fin:** 7/03/11.
Programador responsable: Julio César Torres Macías		
Descripción: Se configura la interfaz para la funcionalidad que permite introducir los mensajes asociados a los servicios que posee el banco.		

Tabla 31 Tarea# 8 de la Historia de usuario Configurar Conexión.

Tarea.		
Número de la tarea: 8.	**Número de la Historia de Usuario:** 6.	
Nombre de la tarea: Implementación de la funcionalidad de configuración.		
Tipo de tarea: Desarrollo.	**Fecha Inicio:** 8/03/11.	**Fecha fin:** 11/03/11.
Programador responsable: Julio César Torres Macías.		
Descripción: Se realiza la implementación de la interfaz que recogerá los datos para realizar la conexión con el servidor Asterisk**.**		

Tabla 32 Tarea# 9 de la Historia de usuario: Mostrar Transacción.

Tarea.		
Número de la tarea: 9.	**Número de la Historia de Usuario:** 7.	
Nombre de la tarea: Implementación de las funcionalidades para mostrar la Transacción.		
Tipo de tarea: Desarrollo.	**Fecha Inicio:** 13/02/11.	**Fecha fin:** 15/02/11.
Programador responsable: Julio César Torres Macías.		
Descripción: Se realiza la implementación de las funcionalidades para obtener las transacciones realizadas por el usuario.		

4.3 Prueba.

Uno de los pilares fundamentales de XP es el proceso de pruebas. Mediante este método se reduce el número de errores no detectados, así como el tiempo entre la introducción de este en el sistema y su detección. Todo esto tiende a elevar la calidad del producto desarrollado y la seguridad de los programadores a la hora de introducir cambios o modificaciones. (54)

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

4.3.1 Pruebas unitarias.

Las pruebas unitarias se realizan para verificar que el código de la aplicación se comporta de la manera esperada. Estas pruebas proporcionan varias ventajas ya que brindan al programador una inmediata retroalimentación de como está realizando su trabajo.

Mediante el desarrollo de la aplicación se realizaron una serie de pruebas unitarias que ayudaron a corregir los errores encontrados en el código. A continuación se muestra un ejemplo.

```
{fieldLabel: 'ID', name: 'id', id:'id', width:190, allowBlank: false}
```

Este código permitía que en el campo de del identificador de la HU Configurar Conexión, se le pudieran introducir letras cuando el identificador es un número entero. Este error se pudo corregir insertándole una nueva línea de código, la cual valida que solo se puedan teclear números en ese campo.

```
{xtype:'numberfield',fieldLabel: 'ID', name: 'id', id:'id', width:190, allowBlank: false}
```

Tabla 33 Cantidad de Pruebas Unitarias realizadas.

Cantidad de Pruebas Realizadas.	Cantidad de Pruebas Satisfactorias.	Cantidad de Errores Encontrados.	Cantidad de Errores Corregidos.

18	13	5	5

Los errores encontrados fueron solucionados a tiempo, por tanto, se puede decir que las pruebas arrojaron un resultado satisfactorio.

4.3.2 Pruebas de aceptación.

Las pruebas de aceptación son creadas en base a las Historias de Usuarios, en cada ciclo de la iteración del desarrollo. El cliente debe especificar uno o diversos escenarios para comprobar que una historia de usuario ha sido correctamente implementada.

Las pruebas de aceptación son de gran importancia, dado que miden el grado de satisfacción del cliente con el producto desarrollado. Por lo tanto, son los clientes los responsables de verificar que los datos de estas pruebas sean correctos. Asimismo, en caso de que fallen varias pruebas, son ellos los encargados de indicar el orden de resolución de los fallos.

Una historia de usuario no se puede considerar terminada hasta tanto pase correctamente todas las pruebas de aceptación. Dado que la responsabilidad es grupal, es recomendable publicar los resultados de las pruebas de aceptación, de manera que todo el equipo esté al tanto de esta información. (55)

A las Historias de Usuario de este sistema se le realizaron todas las pruebas de aceptación pertinentes, obteniendo resultados satisfactorios. Para un mayor entendimiento, los casos de prueba de aceptación realizados están recogidos en el **Anexo 1**.

4.4 Conclusiones.

En este capítulo se trataron las etapas de implementación y pruebas del software en desarrollo. Para ello se mostraron todos los artefactos generados, realizando una descripción de cada uno de ellos, dando paso a las pruebas de aceptación, con las cuales se probaron los requisitos trazados.

CONCLUSIONES GENERALES.

A partir del desarrollo de un sistema que permita pagar los servicios contratados por un determinado cliente mediante vía telefónica, se puede arribar a las siguientes conclusiones:

- Se logró realizar un completo análisis y selección de las metodologías, herramientas y tecnologías necesarias para la creación del producto. Además, se realizó un estudio de los sistemas de banca telefónica existentes tanto a nivel nacional como internacional, para una mejor comprensión del funcionamiento de estos sistemas.

- Se obtuvo una aplicación que permite el pago de servicios contratados por el cliente, mediante el uso del teléfono y sin la interacción de una operadora. La aplicación obtenida también es configurable y resulta muy sencillo introducir un mensaje asociado a un nuevo servicio a cobrar.

- Se cumplieron los objetivos generales trazados, dado que el sistema logra sustituir a la operadora alcanzando una mayor eficiencia en la gestión del pago de servicios vía telefónica.

Con los resultados de este trabajo se ha logrado demostrar que es posible automatizar el pago de los servicios vía telefónica en Cuba. Abriendo la puerta a una nueva variante de a atender a los clientes mediante un sistema automático que se encargará de gestionar el pago de los servicios masivos vía telefónica.

RECOMENDACIONES.

Al concluir el desarrollo de este documento se recomienda:

- Continuar desarrollando la investigación con el fin de perfeccionar e incrementar las funcionalidades de la herramienta.
- Valorar la posibilidad de que el sistema sea capaz de trabajar con más de diez servicios, puesto que la idea inicial estaba pensada para atender solo los servicios básicos.
- Poner a prueba el sistema en la red pública de ETECSA.

REFERENCIA BIBLIOGRÁFICA.

1. **ROVIROSA, LIC. JOSÉ EDUARDO CALZADA.** Tribunalqro. [En línea] 22 de Abrir de 2011. [Citado el: 23 de Abrir de 2011.] Disponible en: [https://tribunalqro.gob.mx/biblio/leeDoc.php?cual=30513].

2. **Mejoresarticulos.** Mejoresarticulos. [En línea] [Citado el: 2 de Junio de 2011.] Disponible en:[http://www.mejoresarticulos.com.mx/].

3. **Voxdata.** Voxdata. [En línea] [Citado el: 2 de Junio de 2011.] Disponible en:[[http://www.voxdata.com.ar].

4. ERP, Columna vertebral del negocio . *El Economista .* Suplemento Tecnología, 2004.

5. **BBVA.** Banco Provincial. [En línea] [Citado el: 15 de Febrero de 2011.] Disponible en[https://www.provincial.com/tlvz/blueprovincial/lineaprovi/index.jsp].

6. **Agricola, Banco.** Banco Agricola. [En línea] [Citado el: 15 de Febrero de 2011.] Disponible en:[http://www.bancoagricola.com/].

7. **bicecorp.** bicecorp. [En línea] [Citado el: 16 de Febrero de 2011.] Disponible en:[http://www.bicecorp.com/inmobiliario/gen_superior.php?op=bt].

8. **Scotiabank.** Scotiabank. [En línea] [Citado el: 15 de Febrero de 2011.] Disponible en:[http://www.scotiabank.com.pe/atencion/b_telefonica.shtml].

9. **Robles, lic.David Armando Espinosa.** Slideshare. [En línea] [Citado el: 3 de Marzo de 2011.] Disponible:[http://www.slideshare.net].

10. **Molpeceres, Alberto.** Javahispano. [En línea] [Citado el: 20 de febrero de 2011.] Disponible: [http://www.javahispano.org/contenidos/archivo/71/metodos_desarrollo.pdf.].

11. **Acuña, Karenny Brito.** *METODOLOGÍAS DE DESARROLLO PARA APLICACIONES WEB.* Cienfuegos : Edición electrónica gratuita, 2009. ISBN-13: 978-84-692-6641-0.

12. **Anaya, Lic. Sandra Lorena.** Iered. [En línea] [Citado el: 15 de Marzo de 2011.] Disponible:[http://www.iered.org/archivos/Grupo_Vultur/Seminario/3-mecs/sesion3/RUP-Vs-XP.pdf].

13. **Reglas, Ing. José Joskowicz.** Fing. [En línea] [Citado el: 8 de abril de 2011.] Disponible:[http://iie.fing.edu.uy/~josej/docs/XP%20-%20Jose%20Joskowicz.pdf].

14. **IanSommerville.** *Ingeniería de Software.* Madrid : PEARSOEDUCACIÓN.SA, 2005. 712.

15. **Kendall, Kendall &.** *Analisis Y Diseño De Sistemas.* s.l. : Pearson Educación. 978-970-26-0577-5.

16. **Isis Imelda López Gonzales, Beatriz Amalia Riveras Carreras.** Scribd. [En línea] [Citado el: 2 de abril de 2011.] Disponible en:[http://es.scribd.com/doc/52247996/80/Visual-Paradigm-for-UML].

17. **Freedownloadmanager.** Freedownloadmanager. [En línea] [Citado el: 22 de Febrero de 2011.] Disponible en:[http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28Iglesia_Anglicana%29_%5BMac_OS_X_cuenta_14717_p/].

18. **Delgado, Ing. Ramiro.** Espe. [En línea] 4 de diciembre de 2006. [Citado el: 16 de Febrero de 2011.] Disponible en:[http://www3.espe.edu.ec:8700/bitstream/21000/547/1/T-ESPE-014997.pdf].

19. **Bizagi.** Bizagi. [En línea] 2009. [Citado el: 20 de Febrero de 2011.] Disponible en:[http://www.bizagi.com/docs/Standard%20Descripci%C3%B3n%20Funcional.pdf].

20. Milestone. [En línea] [Citado el: 6 de Febrero de 2011.] Disponible en:[http://www.milestone.com.mx/CursoModeladoNegociosBPMN.htm].

21. **Ariagnis Yero Guevara, Alejandro Rodríguez Reyes.** *Sistema de Gestión de Emergencias de Seguridad Ciudadana (171).* La Habana : s.n., 2007.

22. **IBM.** IBM. *IBM.* [En línea] [Citado el: 21 de Febrero de 2011.] Disponible en:[http://www-01.ibm.com/software/rational/uml/].

23. **Francisco Ruiz González, Óscar González Martín.** Oasis. [En línea] [Citado el: 17 de Febrero de 2011.] Disponible en:[http://oasis.cisc-ug.org/letzhune/cisc/tutoriales/sexto/Arquitecturas%20de%20sistemas%20de%20bases%20de%20datos.pdf].

24. **Martínez, Rafael.** Postgresql-es. [En línea] [Citado el: 25 de Febrero de 2011.] Disponible en: <http://www.postgresql-es.org/principal>..

25. **Gutiérrez., Javier J.** Lsi. [En línea] [Citado el: 6 de Marzo de 2011.] Disponible en:[http://www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf].

26. **(CESGE), Departamento de Tecnología del Centro de Soluciones de Gestión de Entidades.** *Manual de instalación de las herramientas para Sauxe 1.5.* La Habana : s.n.

27. **Sencha.** Sencha. [En línea] [Citado el: 5 de Marzo de 2011.] Disponible en:[http://www.sencha.com/products/extjs/].

28. **Wsisusolucionip.** Wsisusolucionip. [En línea] [Citado el: 25 de Febrero de 2011.] Disponible en:[http://www.wsisusolucionip.ec/WebsiteSurvey.asp?PId=53807].

29. **Doctrine.** Doctrine. [En línea] [Citado el: 25 de Febrero de 2011.] Disponible en:[http://www.doctrine-project.org/documentation/manual/1_2/en].

30. **Petra.euitio.uniovi.** Petra.euitio.uniovi. [En línea] [Citado el: 5 de Junio de 2011.] Disponible en:[http://petra.euitio.uniovi.es/~i1667065/HD/documentos/Entornos%20de%20Desarrollo%20Integrado.pdf].

31. **Tufuncion.** Tufuncion. [En línea] [Citado el: 23 de Noviembre de 2010.] Disponible en:[http://www.tufuncion.com/zend-studio]..

32. **Mitecnologico.** Mitecnologico. [En línea] [Citado el: 1 de Junio de 2011.] Disponible en:[http://www.mitecnologico.com/Main/LenguajesProgramacionDelLadoDelCliente].

33. **docstoc.** docstoc. [En línea] [Citado el: 21 de Febrero de 2011.] Disponible en:[http://www.docstoc.com/docs/278164/MANUAL-DE-JAVA].

34. **HTML.** HTML. [En línea] [Citado el: 18 de Febrero de 2011.] Disponible en:[http://www.desarrolloweb.com/articulos/que-es-html.html].

35. **Developer.** Developer. [En línea] [Citado el: 19 de Febrero de 2011.] Disponible en:[https://developer.mozilla.org/es/Gu%C3%ADa_JavaScript_1.5/Concepto_de_JavaScript].

36. **Designers, Brendig.** Brendig Designers. [En línea] [Citado el: 8 de Marzo de 2011.] Disponible en:[http://www.brendingdesigners.com.ar/articulos/servidorweb.html].

37. **Linux.ciberaula.** Linux.ciberaula. [En línea] [Citado el: 30 de Enero de 2011.] Disponible en:[http://linux.ciberaula.com/articulo/linux_apache_intro].

38. **Asterisk-es.** Asterisk-es. [En línea] [Citado el: 5 de Marzo de 2011.] Disponible en:[http://comunidad.asterisk-es.org/index.php?title=Introduccion_a_Asterisk].

39. **PHPAGI.** PHPAGI. [En línea] 30 de Septiembre de 2010. [Citado el: 16 de Marzo de 2011.] Disponible en:[http://phpagi.sourceforge.net/].

40. **solutions, Palosanto.** *Manual del Usuario en Español (Beta) Elastix 0.9-alpha.*

41. **Soluciones, TI.** *PROPUESTA TÉCNICO FINANCIERA .* Caracas : s.n., 2010.

42. **D., Carlos Reveco.** U-cursos. [En línea] [Citado el: 20 de Marzo de 2011.] Disponible en:[https://www.u-cursos.cl/ingenieria/2010/2/IN73J/1/material_docente/previsualizar?id_material=309111].

43. **Catarina.** Catarina. [En línea] [Citado el: 17 de Febrero de 2011.] Disponible en:[http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/marquez_a_bm/capitulo5.pdf].

44. **Jim Van Meggelen, Lief Madsen,and Jared Smith.** *Asterisk The Future of Telephony.* s.l. : ORELLY, 2007. ISBN-10: 0-596-51048-9.

45. **Fernández Hernández, Jesús y Alonso Berrocal, José Luis.** Gredos. [En línea] [Citado el: 1 de Junio de 2011.] Disponible

en:[http://gredos.usal.es/jspui/bitstream/10366/21739/1/DIA_Redes%20privadas%20virtuales.pdf].

46. **Deacon, John.** *Model-View-Controler(MVC) Architercture.* s.l. : Addison-Wesley, 2009. ISBN:0-321-26317-0.

47. *Diseño Avanzado de Aplicaciones Web.EXT – Zend Framework y Doctrine.* **Arias, Yuniel Eliades Proenza.** 8, 2009.

48. **Beck, K,.** *Extreme Programming Explained.* s.l. : AddisonWesley, 2000.

49. **STEPHENS, M., ROSENBERG, D.** *"Extreme programming refactored: the case against X.P".* California : s.n., 2003.

50. **Lisandra Díaz Figueredo, Dalaity Castiñeira Pilotos.** *Multimedia para el estudio de la asignatura: Artrópodos terrestres (no crustáceos).* La Habana : s.n., 2010.

51. Office. [En línea] [Citado el: 1 de Junio de 2011.] Disponible en:[http://office.microsoft.com/es-hn/access-help/conceptos-basicos-del-diseno-de-una-base-de-datos-HA001224247.aspx].

52. **Parra, D. Javier Garzás.** Javier Garzas. [En línea] [Citado el: 23 de Febrero de 2011.] Disponible en:[http://www.javiergarzas.com/downloads/JGarzas_Tesis_Doctoral_Capitulo_2_Estado_del_Arte_paginas_28-78.pdf].

53. **aprendizaje, Entorno Virtual de.** Entorno Virtual de aprendizaje. [En línea] 18 de Febrero de 2011. [Citado el: 2 de Marzo de 2011.] Disponible en:[http://eva.uci.cu/file.php/259/Curso_2010-2011/Semana_4/Seminario_1/Materiales_Basicos/patrones_gof.pdf].

54. **Crispin, L. y House, T.** *Probando la Programación Extrema. .* s.l. : AddisonWesley, 2002.

55. **J. J. Gutiérrez, M. J. Escalona, M. Mejías, J. Torres.** Lsi. [En línea] [Citado el: 19 de Abril de 2011.] Disponible:[http://www.lsi.us.es/~javierj/investigacion_ficheros/PSISEXTREMA.pdf].

BIBLIOGRAFÍA CONSULTADA.

1. **ROVIROSA, LIC. JOSÉ EDUARDO CALZADA.** Tribunalqro. [En línea] 22 de Abrir de 2011. [Citado el: 23 de Abrir de 2011.] Disponible en: [https://tribunalqro.gob.mx/biblio/leeDoc.php?cual=30513].

2. **Mejoresarticulos.** Mejoresarticulos. [En línea] [Citado el: 2 de Junio de 2011.] Disponible en:[http://www.mejoresarticulos.com.mx/].

3. **Voxdata.** Voxdata. [En línea] [Citado el: 2 de Junio de 2011.] Disponible en:[[http://www.voxdata.com.ar].

4. ERP, Columna vertebral del negocio . *El Economista .* Suplemento Tecnología, 2004.

5. **BBVA.** Banco Provincial. [En línea] [Citado el: 15 de Febrero de 2011.] Disponible en[https://www.provincial.com/tlvz/blueprovincial/lineaprovi/index.jsp].

6. **Agricola, Banco.** Banco Agricola. [En línea] [Citado el: 15 de Febrero de 2011.] Disponible en:[http://www.bancoagricola.com/].

7. **bicecorp.** bicecorp. [En línea] [Citado el: 16 de Febrero de 2011.] Disponible en:[http://www.bicecorp.com/inmobiliario/gen_superior.php?op=bt].

8. **Scotiabank.** Scotiabank. [En línea] [Citado el: 15 de Febrero de 2011.] Disponible en:[http://www.scotiabank.com.pe/atencion/b_telefonica.shtml].

9. **Robles, lic.David Armando Espinosa.** Slideshare. [En línea] [Citado el: 3 de Marzo de 2011.] Disponible:[http://www.slideshare.net].

10. **Molpeceres, Alberto.** Javahispano. [En línea] [Citado el: 20 de febrero de 2011.] Disponible: [http://www.javahispano.org/contenidos/archivo/71/metodos_desarrollo.pdf.].

11. **Acuña, Karenny Brito.** *METODOLOGÍAS DE DESARROLLO PARA APLICACIONES WEB.* Cienfuegos : Edición electrónica gratuita, 2009. ISBN-13: 978-84-692-6641-0.

12. **Anaya, Lic. Sandra Lorena.** Iered. [En línea] [Citado el: 15 de Marzo de 2011.] Disponible:[http://www.iered.org/archivos/Grupo_Vultur/Seminario/3-mecs/sesion3/RUP-Vs-XP.pdf].

13. **Reglas, Ing. José Joskowicz.** Fing. [En línea] [Citado el: 8 de abril de 2011.] Disponible:[http://iie.fing.edu.uy/~josej/docs/XP%20-%20Jose%20Joskowicz.pdf].

14. **IanSommerville.** *Ingeniería de Software.* Madrid : PEARSOEDUCACIÓN.SA, 2005. 712.

15. **Kendall, Kendall &.** *Analisis Y Diseño De Sistemas.* s.l. : Pearson Educación. 978-970-26-0577-5.

16. **Isis Imelda López Gonzales, Beatriz Amalia Riveras Carreras.** Scribd. [En línea] [Citado el: 2 de abril de 2011.] Disponible en:[http://es.scribd.com/doc/52247996/80/Visual-Paradigm-for-UML].

17. **Freedownloadmanager.** Freedownloadmanager. [En línea] [Citado el: 22 de Febrero de 2011.] Disponible

en:[http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28Iglesia_Anglicana%29_%5BMac_OS_X_cuenta_14717_p/].

18. **Delgado, Ing. Ramiro.** Espe. [En línea] 4 de diciembre de 2006. [Citado el: 16 de Febrero de 2011.] Disponible en:[http://www3.espe.edu.ec:8700/bitstream/21000/547/1/T-ESPE-014997.pdf].

19. **Bizagi.** Bizagi. [En línea] 2009. [Citado el: 20 de Febrero de 2011.] Disponible en:[http://www.bizagi.com/docs/Standard%20Descripci%C3%B3n%20Funcional.pdf].

20. Milestone. [En línea] [Citado el: 6 de Febrero de 2011.] Disponible en:[http://www.milestone.com.mx/CursoModeladoNegociosBPMN.htm].

21. **Ariagnis Yero Guevara, Alejandro Rodríguez Reyes.** *Sistema de Gestión de Emergencias de Seguridad Ciudadana (171).* La Habana : s.n., 2007.

22. **IBM.** IBM. *IBM.* [En línea] [Citado el: 21 de Febrero de 2011.] Disponible en:[http://www-01.ibm.com/software/rational/uml/].

23. **Francisco Ruiz González, Óscar González Martín.** Oasis. [En línea] [Citado el: 17 de Febrero de 2011.] Disponible en:[http://oasis.cisc-ug.org/letzhune/cisc/tutoriales/sexto/Arquitecturas%20de%20sistemas%20de%20bases%20de%20datos.pdf].

24. **Martínez, Rafael.** Postgresql-es. [En línea] [Citado el: 25 de Febrero de 2011.] Disponible en: <http://www.postgresql-es.org/principal>..

25. **Gutiérrez., Javier J.** Lsi. [En línea] [Citado el: 6 de Marzo de 2011.] Disponible en:[http://www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf].

26. **(CESGE), Departamento de Tecnología del Centro de Soluciones de Gestión de Entidades.** *Manual de instalación de las herramientas para Sauxe 1.5.* La Habana : s.n.

27. **Sencha.** Sencha. [En línea] [Citado el: 5 de Marzo de 2011.] Disponible en:[http://www.sencha.com/products/extjs/].

28. **Wsisusolucionip.** Wsisusolucionip. [En línea] [Citado el: 25 de Febrero de 2011.] Disponible en:[http://www.wsisusolucionip.ec/WebsiteSurvey.asp?PId=53807].

29. **Doctrine.** Doctrine. [En línea] [Citado el: 25 de Febrero de 2011.] Disponible en:[http://www.doctrine-project.org/documentation/manual/1_2/en].

30. **Petra.euitio.uniovi.** Petra.euitio.uniovi. [En línea] [Citado el: 5 de Junio de 2011.] Disponible en:[http://petra.euitio.uniovi.es/~i1667065/HD/documentos/Entornos%20de%20Desarrollo%20Integrado.pdf].

31. **Tufuncion.** Tufuncion. [En línea] [Citado el: 23 de Noviembre de 2010.] Disponible en:[http://www.tufuncion.com/zend-studio]..

32. **docstoc.** docstoc. [En línea] [Citado el: 21 de Febrero de 2011.] Disponible en:[http://www.docstoc.com/docs/278164/MANUAL-DE-JAVA].

33. **HTML.** HTML. [En línea] [Citado el: 18 de Febrero de 2011.] Disponible en:[http://www.desarrolloweb.com/articulos/que-es-html.html].

34. **Developer.** Developer. [En línea] [Citado el: 19 de Febrero de 2011.] Disponible en:[https://developer.mozilla.org/es/Gu%C3%ADa_JavaScript_1.5/Concepto_de_JavaScript].

35. **Designers, Brendig.** Brendig Designers. [En línea] [Citado el: 8 de Marzo de 2011.] Disponible en:[http://www.brendingdesigners.com.ar/articulos/servidorweb.html].

36. **Linux.ciberaula.** Linux.ciberaula. [En línea] [Citado el: 30 de Enero de 2011.] Disponible en:[http://linux.ciberaula.com/articulo/linux_apache_intro].

37. **Asterisk-es.** Asterisk-es. [En línea] [Citado el: 5 de Marzo de 2011.] Disponible en:[http://comunidad.asterisk-es.org/index.php?title=Introduccion_a_Asterisk].

38. **PHPAGI.** PHPAGI. [En línea] 30 de Septiembre de 2010. [Citado el: 16 de Marzo de 2011.] Disponible en:[http://phpagi.sourceforge.net/].

39. **solutions, Palosanto.** *Manual del Usuario en Español (Beta) Elastix 0.9-alpha.*

40. **Soluciones, TI.** *PROPUESTA TÉCNICO FINANCIERA .* Caracas : s.n., 2010.

41. **D., Carlos Reveco.** U-cursos. [En línea] [Citado el: 20 de Marzo de 2011.] Disponible en:[https://www.u-cursos.cl/ingenieria/2010/2/IN73J/1/material_docente/previsualizar?id_material=309111].

42. **Catarina.** Catarina. [En línea] [Citado el: 17 de Febrero de 2011.] Disponible en:[http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/marquez_a_bm/capitulo5.pdf].

43. **Jim Van Meggelen, Lief Madsen,and Jared Smith.** *Asterisk The Future of Telephony.* s.l. : ORELLY, 2007. ISBN-10: 0-596-51048-9.

44. **Deacon, John.** *Model-View-Controler(MVC) Architercture.* s.l. : Addison-Wesley, 2009. ISBN:0-321-26317-0.

45. *Diseño Avanzado de Aplicaciones Web.EXT – Zend Framework y Doctrine.* **Arias, Yuniel Eliades Proenza.** 8, 2009.

46. **Beck, K,.** *Extreme Programming Explained.* s.l. : AddisonWesley, 2000.

47. **STEPHENS, M., ROSENBERG, D.** *"Extreme programming refactored: the case against X.P".* California : s.n., 2003.

48. **Lisandra Díaz Figueredo, Dalaity Castiñeira Pilotos.** *Multimedia para el estudio de la asignatura: Artrópodos terrestres (no crustáceos).* La Habana : s.n., 2010.

49. Office. [En línea] [Citado el: 1 de Junio de 2011.] Disponible en:[http://office.microsoft.com/es-hn/access-help/conceptos-basicos-del-diseno-de-una-base-de-datos-HA001224247.aspx].

50. **Parra, D. Javier Garzás.** Javier Garzas. [En línea] [Citado el: 23 de Febrero de 2011.] Disponible en:[http://www.javiergarzas.com/downloads/JGarzas_Tesis_Doctoral_Capitulo_2_Estado_del_Arte_paginas_28-78.pdf].

51. **aprendizaje, Entorno Virtual de.** Entorno Virtual de aprendizaje. [En línea] 18 de Febrero de 2011. [Citado el: 2 de Marzo de 2011.] Disponible en:[http://eva.uci.cu/file.php/259/Curso_2010-2011/Semana_4/Seminario_1/Materiales_Basicos/patrones_gof.pdf].

52. **Crispin, L. y House, T.** *Probando la Programación Extrema. .* s.l. : AddisonWesley, 2002.

53. **J. J. Gutiérrez, M. J. Escalona, M. Mejías, J. Torres.** Lsi. [En línea] [Citado el: 19 de Abril de 2011.] Disponible:[http://www.lsi.us.es/~javierj/investigacion_ficheros/PSISEXTREMA.pdf].

54. **Lorenzo, Lic.David Batard.** *Sistema de Banca Telefónica.* La Habana : s.n., 2007.