Universidad de las Ciencias Informáticas.

Facultad 2.

Título: Cierre de Ticket de Asistencia Técnica.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas.

Autores: Denia Margarita Delgado Valenzuela.

Javier Romero Rodríguez.

Tutores: Erick Pérez Castillo.

Aymara Marín Días.

Ciudad de La Habana, junio, 2011.

"Año 53 de la Revolución".

DECLARACIÓN DE AUTORÍA.

Declaramos ser autores de la presente tesis y autorizamos a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio. Para que así conste firmamos la presente a los ____ días del mes de _____ del año_____.

_____
Denia Margarita Delgado Valenzuela
(Autor)

_____
Javier Romero Rodríguez
(Autor)

_____
Erick Pérez Castillo
(Tutor)

_____
Aymara Marín Días
(Tutor)

RESUMEN.

El presente trabajo trata acerca del desarrollo de una IVR[1]mediante la cual los técnicos de ETECSA[2] puedan realizar los reportes de las interrupciones telefónicas, que atienden diariamente, sin necesidad de interactuar con una operadora. La aplicación permitirá a ETECSA realizar una mejor distribución del trabajo, que realizan sus operadoras, así como la realización de un control más riguroso del trabajo que realizan sus técnicos y a estos últimos les facilitará la creación del reporte que construyen vía telefónica al concluir la reparación.

Para el desarrollo del sistema se realizaron varios estudios acerca de los procesos que se llevan a cabo actualmente en ETECSA para la creación de un reporte de interrupción y de las herramientas que se utilizan actualmente en el mundo para gestionar las interrupciones telefónicas. Se recogieron así los aspectos más importantes de estas herramientas que hicieron posible el desarrollo del sistema.

Algunos datos que se manejan en los estudios realizados son las características de hardware con que contará el servidor donde correrá la aplicación así como el sistema operativo sobre el cual estará corriendo el mismo, todo esto con el objetivo de optimizar los recursos y obtener un buen rendimiento en la aplicación.

Para guiar el proceso de desarrollo del software se seleccionó la metodología XP[3] y para la implementación del sistema se utilizó el lenguaje de programación PHP[4] como lenguaje del lado del servidor, siguiendo el patrón arquitectónico Modelo-Vista-Controlador, además se utilizará la SoftPBX[5] de código abierto Asterisk para el proceso de interacción con la IVR.

Palabras claves: IVR, Asterisk, Rendimiento, SoftPBX.

[1]Acrónimo de InteractiveVoice Response, Respuesta de Voz Interactiva.
[2]Empresa de Telecomunicaciones de Cuba S.A.
[3]Acrónimo de Extreme Programming, Programación Extrema.
[4]Acrónimo de Hypertext Pre-processor, Pre-procesador de Hipertexto.
[5]Acrónimo de Private Branch Exchange, Central Telefónica Privada.

ÍNDICE DE CONTENIDO.

INTRODUCCIÓN.

Actualmente en el mundo, la telefonía es uno de los medios de comunicación más poderosos, la misma está muy desarrollada en algunos países, realizándose procesos complejos de forma totalmente automática a través de IVRs, sin el intercambio con operadoras telefónicas, lo que proporciona a las empresas, grandes ganancias y la disminución de pérdidas económicas por contratación de personal, ante la crisis económica en la que se encuentra inmerso el mundo.

Nuestro país no está ajeno a esta crisis, es por esto que se está llevando a cabo una reestructuración en los centros laborales para hacer una distribución óptima de la fuerza de trabajo. Debido a esto se está tratando de automatizar la mayor cantidad de procesos posibles en vista a un uso racional de los recursos. Toda esta automatización viene de la mano con uso del software libre, lo que permite grandes ahorros al país por concepto de licencias de software. Es precisamente la telefonía una de las ramas donde se pueden aplicar de forma más eficiente estas medidas.

ETECSA es la encargada de brindar servicios telefónicos a las empresas y a la población en general, contando con una gran cantidad de abonados telefónicos. Esta empresa es también la encargada de ofrecer los servicios de reparaciones de teléfonos, líneas telefónicas y todo cuanto tenga que ver con interrupciones telefónicas en el país. Este proceso es realizado por los técnicos de la empresa, los cuales luego de hacer la reparación llaman a un número de servicio, donde contactan con operadoras, las cuales registran los datos de las reparaciones realizadas. Esta forma de cerrar la interrupción no es muy eficiente ya que la interacción con la operadora puede demorar alrededor de 4 minutos, lo cual representa un tiempo muy prolongado, y en ese momento otro técnico podría estar esperando para cerrar otra interrupción. Además el técnico carece de la posibilidad de comprobar automáticamente la calidad del sonido en el teléfono reparado, por lo que no se puede comprobar la calidad del servicio brindado. Por otra parte esta operadora podría estar brindando sus servicios a otro proceso de la empresa si los técnicos contaran con una forma automática de realizar este proceso de dar cierre a las interrupciones.

Las deficiencias señaladas anteriormente hacen necesaria la creación de una nueva aplicación informática. La tarea es designada por ETECSA y asumida por la UCI[6].

A partir del análisis de los problemas planteados, los esfuerzos estarán encaminados a resolver el siguiente **problema**: ¿Cómo automatizar el proceso de cierre de interrupciones telefónicas?

Teniendo en cuenta el problema a resolver planteado anteriormente se define como **objeto de estudio** el proceso de informatización del cierre de interrupciones telefónicas. Siendo el **campo de acción** del presente trabajo precisamente el proceso de informatización del cierre de las interrupciones telefónicas en ETECSA con la ayuda de un sistema IVR.

Para el desarrollo de la investigación se traza como **objetivo general**: Construir una aplicación capaz de informatizar el proceso del cierre de interrupciones telefónicas en ETECSA.

Se plantean además como **objetivos específicos**:

- Construir una aplicación IVR capaz de realizar el proceso de cierre de las interrupciones telefónicas.
- Construir una aplicación informática capaz de gestionar los reportes de las interrupciones, así como los datos de los técnicos.
- Validar la solución propuesta.

Para lograr dichos objetivos se plantearon las siguientes **tareas**:

- Realización de un estudio de las principales técnicas y tecnologías actuales para el desarrollo de IVRs.
- Instalación y configuración de la SoftPBX Asterisk para la construcción de una IVR capaz de automatizar el proceso de cierre de interrupciones telefónicas.
- Realización de un estudio sobre las posibles metodologías de desarrollo a utilizar.
- Realización delas pruebas de software, correspondientes a la metodología de desarrollo utilizada.

[6]Universidad de las Ciencias Informáticas.

La **idea a defender** del siguiente trabajo es: La automatización del proceso de cierre de interrupciones telefónicas en ETECSA, permitirá a los técnicos de dicha empresa realizar su trabajo con mayor rapidez y calidad, además, permitirá a la empresa realizar una mejor distribución de las operadoras telefónicas con que cuenta actualmente.

Para dar soporte a la idea anteriormente planteada se utilizaron los siguientes métodos de investigación:

- Analítico–sintético: Se utilizó durante el proceso de revisión bibliográfica para conocer el funcionamiento de las IVRs en la SoftPBX Asterisk y al realizar el análisis de las principales técnicas, tecnologías y metodologías de desarrollo de software para lograr la confección de la aplicación.
- Inductivo-deductivo: Permitió pasar de lo particular a lo general y viceversa favoreciendo objetivamente el enlace que se establece en la realidad entre lo singular y lo general ya que ambas se complementan mutuamente en el proceso de desarrollo científico.
- Modelación: Pues se crean abstracciones que explican la realidad, por ejemplo, todos los modelos y diagramas presentados.

El aporte práctico esperado del trabajo es: La obtención de un sistema que permita a los técnicos de ETECSA realizar el proceso de cierre de interrupciones de forma automática.

Para organizar el presente documento se ha dividido en 5 capítulos estructurados como sigue:

Capítulo 1: Fundamentación Teórica: Se expone el estado del arte, mostrando los sistemas vinculados al campo de acción. Al mismo tiempo se describe el objeto de estudio. También se detallan tendencias y tecnologías actuales utilizadas para el desarrollo del sistema y el motivo de su utilización. Se realiza una caracterización del modelo de desarrollo aplicado, así como delas herramientas utilizadas para desarrollar los artefactos.

Capítulo 2: Características del Sistema: Se enfatiza en las características del sistema a desarrollar, haciendo un análisis de los flujos de trabajo involucrados en el campo de acción y el objeto de automatización de la propuesta.

Capítulo 3: Exploración y planificación: Se detallan los artefactos generados durante las fases de exploración y planificación del proyecto.

Capítulo 4: Implementación y pruebas: Se expone todo lo relacionado a los procesos de implementación y pruebas del sistema.

Capítulo 5: Estudio de factibilidad: Se lleva a cabo un estudio de factibilidad del sistema.

CAPÍTULO I. FUNDAMENTACIÓN TEÓRICA.

1.1 Introducción.

En el presente capítulo se profundizan en algunos conceptos fundamentales que intervienen en el proceso de desarrollo de un sistema IVR, así como para qué son usados los mismos a nivel mundial y nacional y cuáles son las empresas líderes en el desarrollo de aplicaciones de este tipo. Además se exponen las principales características de las herramientas y tecnologías utilizadas para el desarrollo de la solución propuesta.

1.2 Sistemas IVR.

Un sistema IVR no es más que, como su nombre lo indica un sistema de respuesta por voz interactiva. Se puede definir genéricamente como la tecnología que permite la interacción entre una persona que llama por teléfono y una computadora. Las personas interactúan con los sistemas IVR presionando las teclas del teléfono o grabando su voz en respuesta a las preguntas digitalizadas que realiza la computadora. Está basado en muchos ambientes multiaplicaciones de alto volumen, que permiten a los clientes acceder e interactuar con computadoras y servidores de archivos, basados en redes de área local, a través de un amplio rango de aplicaciones y servicios.

Un sistema IVR permite a un SoftPBX responder a un llamado telefónico, guiando al usuario a través de un diálogo automatizado entre diferentes opciones de servicios. Los mismos se pueden valer para la interacción con el cliente del uso de base de datos relacionales.

1.2.1 Ventajas de los sistemas IVR.

Los sistemas IVR nos proporcionan numerosas ventajas las mismas están dadas por la necesidad de las empresas de brindar un mejor servicio a sus clientes y posicionarse de manera positiva en el mercado mundial además de ahorrar en inversiones y personal de trabajo. Entre las principales ventajas de los sistemas IVR se encuentran:

• Mejor servicio de atención al cliente: El IVR brinda soporte a muchas empresas, instituciones y organizaciones de cualquier índole, al automatizar la atención y respuesta telefónica mejora su servicio de atención al cliente, al ofrecer al cliente consultas y transacciones rápidas sin la necesidad de acercarse a una agencia, o esperar en la cola del Centro de Llamadas para que un ejecutivo de atención al cliente, lo atienda.

• Reduce el porcentaje de abandono de llamadas: Es uno de los parámetros para medir el nivel de servicio en un Centro de Llamadas, este parámetro se define como la cantidad de llamadas pérdidas luego de que esta ha ingresado a la cola de espera. Al tener un menor tráfico de llamadas entrantes, tendremos un menor porcentaje de abandono de llamadas.

• Permite servicio 24 horas: Ofrece a los usuarios las funciones de acceso automatizado a la información, servicios automatizados, 24horas de operación, calidad de las respuestas de voz y niveles de seguridad. El IVR permite que la información que se encuentra en sus servidores se encuentre disponible para el público que lo requiera.

• Acceso en distintos idiomas: Se pueden dar los mensajes en distintos idiomas. Gracias a los IVRs cualquier organización puede incrementarlos niveles de atención al cliente y optimizar procesos de marketing, además de extender las horas de atención al cliente mejorando la diseminación de la información.

1.2.2 Desventajas de los sistemas IVR.

El tema de desventajas del sistema IVR está relacionado con factores como:

• Complejidad en los menús de acceso: En ocasiones es muy complejo ir avanzando por los menús hasta llegar a la opción requerida debido a que se tiene que usar la memoria auditiva, lo que ocasiona que el usuario se desespere y acabe preguntando por un operador telefónico, esto se resuelve con un buen diseño del IVR para que el usuario no tenga que memorizar un menú de opciones muy extenso.

• El sistema no tiene una atención personalizada: El sistema no tiene una atención absolutamente personalizada, aunque constituye un buen acercamiento a la misma, lo cual puede resultar molesto para

ciertos clientes, esto se resuelve brindándole al cliente la opción de luego de realizada una consulta contactar a un asesor de servicio al cliente.

• Falta de capacidad del sistema para atender llamadas simultáneas: Si en horas pico el número de usuarios que soporta la PBX se sobrepasa y no se puede encaminar adecuadamente las llamadas, también sería éste un motivo de angustia para los clientes desatendidos en esos horarios, aunque en realidad con este sistema IVR se puede encaminar de mejor manera el tráfico que en el caso de un Centro de Llamadas por sí solo, pero más que una desventaja del IVR este problema sería inherente al dimensionamiento adecuado de la red del sistema telefónico de la organización o empresa.

1.2.3 Usos más frecuentes de los sistemas IVR.

Los sistemas de IVR se utilizan para varias aplicaciones del cliente, incluyendo la respuesta de voz automática y la realimentación, directorio automático, enrutamiento de llamada, así como los sistemas de mensajería de voz, los cuales funcionan como un sofisticado contestador automático, o los sistemas interactivos de respuesta de voz y fax, los cuales permiten a las personas interactuar con bases de datos computarizadas a través del teclado telefónico o comandos hablados. Este sistema se suele utilizar en el ambiente bancario llamándose “Banca Telefónica”, pero se incluyen a muchos tipos de industrias como educación, hoteles, servicio social, entre otras. El IVR puede llamar al cliente, puede hacer gestión de cartera vencida, recibir fax, el sistema de “pago-por-evento” de las compañías de cable es otro ejemplo de aplicaciones de IVR.

Los sistemas IVR son usados por numerosas instituciones a nivel mundial por las ventajas y ahorros económicos que proveen. Entre estas instituciones se encuentran:

- Bancos.
- Hoteles.
- Centros de reparaciones.
- Fábricas.
- Comercios.
- Seguros.

- Oficinas de correos.
- Centros de distribución.
- Hospitales.
- Universidades.
- Empresas de transportes.
- Telemarketing.
- Encuestas.

En Cuba son usados por ETECSA para el sistema “Propia” de tarjetas telefónicas, para la realización de encuestas vía telefónica y además son usados por otras empresas que cuentan con pizarras telefónicas como por ejemplo la UCI, algunos hospitales, la empresa eléctrica, las empresas de ómnibus, entre otras.

1.3 Líderes internacionales en la producción de sistemas IVR.

Pronexus: Desde 1994, Pronexus ha ofrecido aplicaciones de voz probadas y herramientas de desarrollo de IVR en las que confían los desarrolladores para ayudarles a integrar la voz con los sistemas de negocios, incluyendo comunicaciones unificadas, gestión de relaciones con el cliente y sistemas de automatización del servicio de campo. (1)

El software VBVoice® de Pronexus es un paquete de herramientas muy eficaz para el desarrollo rápido de aplicaciones con soluciones IVR. VBVoice permite la creación de soluciones IVR entrantes y salientes con infinidad de ventajas, entre otras: operadores automáticos, pagos electrónicos, aplicaciones de fax, notificaciones, sondeos y encuestas.

Las aplicaciones de IVR desarrolladas con VBVoice se encuentran implementadas en una gran variedad de rubros, desde actividades bancarias a entidades del gobierno, servicios de atención médica, compañías de seguros y empresas de servicios públicos.

El paquete de herramientas IVR de VBVoice se integra con Microsoft® Visual Studio® y combina un entorno visual del flujo de llamadas de fácil implementación con controles totalmente programables y opciones de vanguardia, como las tecnologías de reconocimiento de voz y la conversión de texto en voz.

Avaya: Es una empresa privada de telecomunicaciones que se especializa en el sector de la telefonía y centros de llamadas. Avaya tiene unos 18000 empleados desde 2007, el 40% de los cuales se encuentran fuera de los Estados Unidos, la sede mundial se encuentra en Basking Ridge, Nueva Jersey. (2)

El 29 de julio de 2009 – Avaya Inc. anunció que la compañía recibió la más alta calificación -- "Altamente Positivo" – en el reporte "MarketScope para Sistemas IVRs y Portales de Voz Empresarial 2009" emitido por Gartner. El reporte evaluó a los proveedores líderes de sistemas de respuesta de voz y aplicaciones.

MarketScope es una marca registrada el 28 de mayo de 2009 por Gartner Inc. reutilizado bajo permiso. MarketScope es una evaluación de posicionamiento en el mercado durante un período específico de tiempo.

Alcatel–Lucent: Es una empresa multinacional resultado de la fusión de la empresa francesa Alcatel y la estadounidense Lucent Technologies. La empresa provee hardware, software y servicios para proveedores de servicios de telecomunicaciones y empresas. Alcatel-Lucent vende equipamiento para redes de telefonía fija y móvil, redes de datos y de distribución de vídeo y televisión. (3)

En el año 2010 Alcatel-Lucent anunció que Genesys, su marca de software para centros de contacto, ha recibido la puntuación "Altamente Positivo", la más alta posible.

Genesys es el proveedor líder de software de administración de interacciones de clientes mediante teléfono, Web y dispositivos móviles. La suite de software Genesys administra conversaciones de clientes en múltiples canales y recursos, auto servicio y servicio asistido, para cumplir las necesidades de los clientes, optimizar los objetivos de la atención al cliente y utilizar recursos de forma eficaz. El software de Genesys dirige más de 100 millones de interacciones de clientes cada día para 4,000 empresas e instituciones gubernamentales en 180 países.

1.4 Metodología de desarrollo de software.

En un proyecto de desarrollo de software la metodología define Quién debe hacer, Qué, Cuándo y Cómo debe hacerlo. La metodología constituye la columna vertebral del proceso de desarrollo de software. No

existe una metodología universal. Las características de cada proyecto (equipo de desarrollo, recursos, entre otros.) exigen que el proceso sea configurable.

Para la realización del sistema propuesto en este trabajo se decidió utilizar la metodología de desarrollo XP, la cual está basada en valores de simplicidad, comunicación y retroalimentación. Funciona mediante la unión de todo el equipo y la aplicación de prácticas simples, con suficiente retroalimentación para permitirle al equipo ver donde están y adaptar sus prácticas a cada situación única.

1.4.1 Características de XP.

- Los diseñadores y programadores se comunican efectivamente con el cliente y entre ellos mismos.
- Los diseños del software se mantienen sencillos y libres de complejidad o pretensiones excesivas.
- Se obtiene retroalimentación de usuarios y clientes desde el primer día gracias a las baterías de pruebas.
- No introduce funcionalidades antes que sean necesarias.
- El cliente o el usuario se convierten en miembro del equipo.

1.4.2 Prácticas de XP.

Programación en parejas: La XP incluye, como una de sus prácticas estándar, la programación en parejas. Nadie programa en solitario, siempre hay dos personas delante del ordenador. Ésta es una de las características que más se cuestiona al comienzo de la adopción de la XP dentro de un equipo, pero en la práctica se acepta rápidamente y de forma entusiasta. El hecho de que todas las decisiones las tomen al menos dos personas proporciona un mecanismo de seguridad enormemente valioso.

Con dos personas responsabilizándose del código en cada momento, es menos probable que se caiga en la tentación de dejar de escribir pruebas. Es muy difícil que dos personas se salten tareas por descuido o negligencia ya que el código siempre está siendo revisado por otra persona.

Refactorización: El código de la mayor parte de las aplicaciones empieza en un razonable buen estado, para luego deteriorarse de forma progresiva. El coste desorbitado del mantenimiento, modificación y ampliación de aplicaciones ya existente se debe en gran parte a este hecho.

Uno de los objetivos de la XP es mantener la curva de costes tan plana como sea posible, por lo que existen una serie de mecanismos destinados a mantener el código en buen estado, modificándolo activamente para que conserve claridad y sencillez. A este proceso básico para mantener el código en buena forma se le llama refactorización.

La refactorización no sólo sirve para mantener el código legible y sencillo: también se utiliza cuando resulta conveniente modificar código existente para hacer más fácil implementar nuevas funcionalidades.

Cliente en el equipo: El cliente siempre está disponible para resolver dudas y para decidir qué y qué no se hace en cada momento, en función de los intereses del negocio. Debido a su inmersión dentro del equipo, y a que es él quien decide qué y qué no se hace, junto con los tests que verifican si la funcionalidad es la correcta y deseada, el cliente obtiene información absolutamente realista del estado del proyecto.

Releases pequeñas: Siguiendo la política de la XP de dar el máximo valor posible en cada momento, se intenta liberar nuevas versiones de las aplicaciones con frecuencia. Éstas deben ser tan pequeñas como sea posible, aunque deben añadir suficiente valor como para que resulten valiosas para el cliente.

Semanas de 40 horas: La Programación Extrema lleva a un modo de trabajo en que el equipo está siempre al 100%. Una semana de 40 horas en las que se dedica la mayor parte del tiempo a tareas que suponen un avance puede dar mucho de sí, y hace innecesario recurrir a sobreesfuerzos excepto en casos extremos.

Además, el sobreesfuerzo continuado pronto lleva a un rendimiento menor y a un deterioro de la moral de todo el equipo.

Estándares de codificación: Para conseguir que el código se encuentre en buen estado y que cualquier persona del equipo pueda modificar cualquier parte del código es imprescindible que el estilo de codificación sea consistente. Un estándar de codificación es necesario para soportar otras prácticas de la XP.

Uso de Metáforas: La comunicación fluida es uno de los valores más importantes de la Programación Extrema, para conseguir que esto ocurra es imprescindible, entre otras cosas, utilizar el vocabulario del negocio. También es fundamental huir de definiciones abstractas. Dicho de otro modo, la XP no pretende seguir la letra de la ley, sino su espíritu. Dentro de este enfoque es fundamental buscar continuamente metáforas que comuniquen intenciones y resulten descriptivas, enfatizando el qué por delante del cómo.

1.5 Herramientas y tecnologías utilizadas.

1.5.1 Servidor web.

El servidor **Apache** es un servidor web HTTP[7]de código abierto para plataformas GNU/Linux, Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP y la noción de sitio virtual.(5)

El servidor Apache se desarrolla dentro del proyecto HTTP Server de la Apache Software Fundation.

Apache es un servidor web flexible, rápido y eficiente, continuamente actualizado y adaptado a los nuevos protocolos. Entre sus características destacan:

- Multiplataforma
- Es un servidor de web conforme al protocolo HTTP/1.1

1.5.2 Lenguajes de programación web.

Lenguajes del lado del cliente.

El lenguaje **HTML[8]** fue diseñado para estructurar documentos y presentarlos en forma de hipertexto, estableciendo relaciones unidireccionales entre ellos (hipervínculos). Inicialmente fue concebido para visualizar e interconectar el contenido de documentos electrónicos, considerando por ello un pequeño conjunto de etiquetas.

[7]Acrónimo de Hypertext Transfer Protocol, Protocolo de Transferencia de Hipertexto.
[8]Acrónimo de HyperTextMarkupLanguage, Lenguaje Marcado de Hipertexto.

JavaScript: Es un lenguaje de programación que permite interactuar con el navegador de manera dinámica y eficaz, que no requiere compilación y es utilizado principalmente en páginas Web, con una sintaxis semejante a la del lenguaje Java y el lenguaje C. Gracias a su compatibilidad con la mayoría de los navegadores modernos, es el lenguaje de programación del lado del cliente más utilizado, lo que le permite validar de forma eficiente formularios, detectar navegadores y mejorar el diseño, además de ser un lenguaje fácil de aprender.

Lenguajes del lado del servidor.

PHP: Es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas. Es usado principalmente en la interpretación del lado del servidor pero actualmente puede ser utilizado desde una interfaz de línea de comandos o en la creación de otros tipos de programas. Este lenguaje está diseñado especialmente para desarrollo web y puede ser embebido dentro de código HTML. Generalmente se ejecuta en un servidor web, tomando el código en PHP como su entrada y creando páginas web como salida. Puede ser desplegado en la mayoría de los servidores web y en casi todos los sistemas operativos y plataformas sin costo alguno. PHP se encuentra instalado en más de 20 millones de sitios web y en un millón de servidores.

1.5.3 Entorno de desarrollo integrado.

Un IDE[9] es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes. En algunos lenguajes, un IDE puede funcionar como un sistema en tiempo de ejecución, en donde se permite utilizar el lenguaje de programación en forma interactiva, sin necesidad de trabajo orientado a archivos de texto.

Geany: Es un editor de texto ligero con características básicas de entorno de desarrollo integrado. Está disponible para distintos sistemas operativos, como GNU/Linux, Mac, Solaris y Microsoft Windows. Es distribuido como software libre bajo la Licencia Pública General de GNU. Tiene soporte para muchos lenguajes de programación distintos, como C, C++, Java, JavaScript, PHP, HTML, Python, Perl, Ruby,

[9] Entorno de desarrollo integrado.

Fortran, Pascal y Haskell. Algunas de las características más destacadas de Geany son: autocompletado, soporte multidocumento, soporte de proyectos, coloreado de sintaxis y emulador de terminal incrustado. (6)

1.5.4 Frameworks.

Un framework (marco de trabajo), en el desarrollo de software, es una estructura de soporte definida, mediante la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo que extiende o utiliza las aplicaciones del dominio.

CEDRUX: Es un ambiente de trabajo que está integrado por diferentes framework como son ExtJs, Doctrine, Zend Framework y UCID[10].

Zend Framework: Es un framework para desarrollo de aplicaciones Web y servicios Web con PHP. Brinda soluciones para construir sitios web modernos, robustos y seguros. Además es código abierto y trabaja con PHP 5. (7)

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[11] pero ampliándola con diferentes características.
- Proporciona mecanismos de filtrado y validación de entradas de datos.

[10]Acrónimo de Unidad de Compatibilidad Integrada de Desarrollo.
[11]Acrónimo de PHP Data Objects, Capa de Abstracción de Acceso a Datos para PHP.

ExtJs Framework: Es un framework JavaScript para el desarrollo de aplicaciones Web. Tiene un sistema dual de licencia: Comercial y Código Abierto. Este framework puede correr en cualquier plataforma que pueda procesar y devolver datos estructurados (PHP, Java, .NET y algunas otras) en tiempo de ejecución carga y crea todos los objetos HTML a través del uso intenso de DOM[12]. Los datos son obtenidos mediante mensajes AJAX[13] a través de XML[14] y/o JSON[15]. (8)

Dispone de un conjunto de componentes para incluir dentro de una aplicación web, como:

- Cuadros y áreas de texto.
- Campos para fechas.
- Campos numéricos.
- Combobox.
- Radiobuttons y checkboxs.
- Editor HTML.
- Elementos de datos (con modos de sólo lectura, datos ordenables, columnas que se pueden bloquear y arrastrar, etc.).
- Árbol de datos.
- Pestañas.
- Barra de herramientas.
- Menús al estilo de Windows.
- Paneles divisibles en secciones.
- Sliders.

También contiene numerosas funcionalidades que permiten añadir interactividad a las páginas HTML.

Doctrine Framework: El Framework Doctrine es un potente y completo sistema ORM[16].

[12] AcrónimodeDocumentObjectModel, Modelo de Objetos del Documento.
[13] Acrónimo de Asynchronous JavaScript And XML, JavaScript asíncrono y XML.
[14] Acrónimo eXtensible Markup Language, Lenguaje de Marcas Extensible.
[15] Acrónimo de JavaScript ObjectNotation, Notación de Objetos JavaScript.
[16] Acrónimo de ObjectRelationalMapper, Mapas de Relaciones de Objetos.

Un ORM es una técnica de programación que nos permite convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional, es decir, las tablas de nuestra base de datos pasan a ser clases y los registros objetos que podemos manejar con facilidad.

Utilizar un ORM tiene una serie de ventajas que nos facilitan enormemente tareas comunes y de mantenimiento:

- Reutilización: La principal ventaja que aporta un ORM es la reutilización permitiendo llamar a los métodos de un objeto de datos desde distintas partes de la aplicación e incluso desde diferentes aplicaciones.
- Encapsulación: La capa ORM encapsula la lógica de los datos pudiendo hacer cambios que afectan a toda la aplicación únicamente modificando una función.
- Portabilidad: Utilizar una capa de abstracción nos permite cambiar en mitad de un proyecto de una base de datos MySQL a una Oracle sin ningún tipo de complicación. Esto es debido a que se usa una sintaxis MySQL, Oracle o SQLite para acceder a nuestro modelo, sino una sintaxis propia del ORM utilizado que es capaz de traducir a diferentes tipos de bases de datos.
- Seguridad: Los ORM suelen implementar mecanismos de seguridad que protegen nuestra aplicación de los ataques más comunes como inyecciones SQL.
- Mantenimiento del código: Gracias al correcto orden de la capa de datos, modificar y mantener nuestro código es una tarea sencilla.

Doctrine es un ORM para PHP 5.2.3 y posterior. Además de todas las ventajas que conlleva un ORM, uno de sus puntos fuertes es su lenguaje DQL[17]. (9)

Cuando trabajamos con Doctrine, necesitamos informar a su motor interno de cual es el modelo de nuestra aplicación, para ello podemos hacer ingeniería inversa de nuestra base de datos existente, o si empezamos la aplicación desde 0, se debe crear el modelo en la sintaxis específica que nos propone Doctrine y luego generar toda la base de datos.

[17]Acrónimo de Doctrine QueryLanguage, Lenguaje de Consulta de Doctrine.

UCID Framework: Es el Framework encargado del trabajo con la vistas. Abarca la integración de ExtJs Framework con el sistema incluyendo el integrador de interfaz, el generador de interfaz dinámica y la impresión de documentos. Integra la iconografía, los diferentes temas de escritorio de la aplicación, el multilenguaje.

1.5.5 Gestor de base de datos.

Los SGBD[18] son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan.

PostgreSQL: Es un servidor de base de datos relacional libre. Tiene soporte total para transacciones, disparadores, vistas, procedimientos almacenados, almacenamiento de objetos de gran tamaño. Se destaca en ejecutar consultas complejas, consultas sobre vistas, subconsultas y joins de gran tamaño. (10)

Permite la definición de tipos de datos personalizados e incluye un modelo de seguridad completo. Como toda herramienta de software libre PostgreSQL tiene, entre otras ventajas, las de contar con una gran comunidad de desarrollo en Internet, su código fuente está disponible sin costo alguno y algo muy importante es que dicha herramienta es multiplataforma. Fue diseñado para ambientes de alto volumen de datos. Soporta transacciones y desde la versión 7.0, claves ajenas con comprobaciones de integridad referencial. Tiene buen soporte para vistas y procedimientos almacenados en el servidor, además tiene ciertas características orientadas a objetos.

1.5.6 Herramienta de administración de base de datos.

pgAdminIII: Es una aplicación gráfica para trabajar el gestor de bases de datos PostgreSQL, siendo una de las más completas y populares con licencia de código abierto. Está escrita en C++ y permite que se pueda usar en Linux, Windows y otros sistemas operativos. Es capaz de gestionar versiones a partir de la PostgreSQL 7.3 ejecutándose en cualquier plataforma.

[18]Sistemas de Gestión de Base de Datos.

pgAdmin III está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SQL[19] simples hasta desarrollar bases de datos complejas. La interfaz gráfica soporta todas las características de PostgreSQL y facilita enormemente la administración. La aplicación también incluye un editor SQL con resaltado de sintaxis así como un editor de código de la parte del servidor.

1.5.7 Control de versiones.

Una versión, revisión o edición de un producto, es el estado en el que se encuentra en un momento dado en su desarrollo o modificación. Se llama control de versiones a la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo.

Subversion: es un software de sistema de control de versiones diseñado específicamente para reemplazar al popular CVS[20], el cual posee varias deficiencias. Es software libre y se le conoce también como SVN por ser ese el nombre de la herramienta de línea de comandos. Una característica importante de Subversion es que, a diferencia de CVS, los archivos versionados no tienen cada uno un número de revisión independiente. En cambio, todo el repositorio tiene un único número de versión que identifica un estado común de todos los archivos del repositorio en cierto punto del tiempo.

1.6 Conclusiones parciales.

En el presente capítulo se abordaron los términos más importantes usados en el mundo de las IVRs, así como sus principales usos, ventajas y desventajas que provee el uso de las mismas. También se expusieron las principales herramientas y tecnologías a utilizar destacando el uso del software libre, ya que el mismo proporciona valores morales como: Solidaridad, Colectivismo, Honestidad, Responsabilidad Social, Sensibilidad Humana, Altruismo, Soberanía, Libertad, Independencia, Amor al trabajo, Ingenio, Originalidad, Conocimiento y ofrece grandes ahorros al país por concepto de pago de licencias.

[19] Acrónimo de SStructured Query Language, Lenguaje de Consultas Estructurado.
[20] Acrónimo de Concurrent Versions System, Sistema de Versiones Concurrentes.

CAPÍTULO II. EXPLORACIÓN Y PLANIFICACIÓN

2.1 Introducción.

En este capítulo se abordarán los temas relacionados con las características principales con que va a contar el sistema a implementar, además se hace alusión a las fases de exploración y planificación propias de la metodología de desarrollo utilizada para la implementación del sistema que se propone. Se exponen además los artefactos generados durante el transcurso de las mismas.

2.2 Características del sistema.

2.2.1 Objeto de automatización.

Actualmente el proceso de cierre de interrupciones telefónicas en ETECSA se efectúa por parte de los de los técnicos a través de la interacción con operadoras telefónicas, las cuales recogen los datos de la labor realizada por los técnicos. Este proceso se podría realizar de forma semiautomática es decir con la interacción del técnico con un sistema informático. De esta forma se estaría sustituyendo el trabajo de la operadora lo cual permitiría a ETECSA realizar una mejor distribución de su fuerza de trabajo.

2.2.2 Propuesta del sistema.

El sistema Cierre de Ticket de Asistencia Técnica (CITAT) tiene como propósito desarrollar una solución informática que se encargue de todo lo referente al cierre del ticket que se genera cuando un técnico de ETECSA realiza una reparación. Lo anterior se realizará por medio de un IVR con un menú de árbol de selección múltiple, se prevén a los menos tres niveles de submenús controlados por medio del discado con tonos DTMF[21]. La IVR correrá en un servidor SoftPBX Asterisk. Además se creará una aplicación que permita visualizar y modificar las reparaciones hechas por parte de los técnicos así como gestionar los datos de los mismos. La aplicación brindará además un sistema de reportería dinámica mediante la cual

[21]Acrónimo de Dual-Tone Multi-Frequency, Multifrecuencia de doble tono.

se podrán exportar los datos de las interrupciones deseadas por el usuario, con los campos de interés para él.

Características no funcionales del sistema.

Un requerimiento no funcional es una característica o cualidad que todo sistema debe poseer, para la realización de la aplicación propuesta, se han identificado los siguientes requisitos no funcionales.

- **Características de apariencia o interfaz externa.**

La aplicación propuesta será usada por personas que pueden o no tener conocimientos básicos de informática, por lo que la interfaz debe ser amigable y fácil de usar.

- **Características de usabilidad.**

A los administradores del sistema se les capacitará en el uso de la aplicación. Estas personas tendrán un nivel de acceso amplio en la aplicación.

- **Características de rendimiento.**

Para el funcionamiento óptimo se seguirán técnicas de elaboración de sitios Web, que faciliten el acceso rápido a las páginas. La eficiencia del sistema estará dada en gran medida por el aprovechamiento de los recursos y la propuesta debe ser rápida con un tiempo de respuesta mínimo.

- **Características de portabilidad.**

El sistema podrá ser accedido desde cualquier sistema operativo. La aplicación, el servidor de Base de Datos y el servidor Asterisk pueden estar en la misma máquina conformando una arquitectura centralizada, aunque es recomendable tener una arquitectura distribuida con los servidores en máquinas diferentes.

- **Características de seguridad.**

Confiabilidad: La información manejada por el sistema debe estar protegida de acceso no autorizado.

Integridad: La información manejada por el sistema debe ser objeto de una cuidadosa protección contra la corrupción y estados de inconsistencia.

Disponibilidad: La aplicación deberá estar disponible en todo momento para aquellas personas con acceso y los mecanismos utilizados para lograr la seguridad no deben ser un obstáculo a los usuarios para obtener los servicios deseados en un momento dado.

- **Características de software.**

En las computadoras de los usuarios solo se requiere un navegador Web Mozilla Firefox u otro que brinde soporte para DOM 2.0 y Java Script.

- **Características de hardware.**

En el cliente se requiere una máquina con 256 MB de RAM como mínimo, un procesador igual o mayor a 1.40 GHZ y una tarjeta de red, todos los servidores deben tener 1GB de RAM, 160 GB de disco duro mínimo, además deben de tener UPS, un lector de DVD o CD y una tarjeta de red.

2.3 Propuesta de arquitectura del sistema.

El técnico después de realizar la reparación telefónica llama a un número de servicio y es atendido por un IVR con el que interactúa para introducir los datos de la reparación realizada. Más tarde la persona encargada puede modificar los estados de la reparación accediendo a la aplicación web. Tanto la aplicación web como el servidor Asterisk interactúan con un sistema gestor de base de datos PostgreSQL lo que les permite leer y escribir los datos en el mismo luego de la interacción con los usuarios.

Imagen 1: Propuesta de arquitectura del sistema.

2.3.1 Arquitectura cliente servidor.

Para el desarrollo del sistema se decidió utilizar el modelo cliente-servidor ya que permite que los usuarios puedan acceder desde cualquier parte al sistema. El mismo consiste básicamente en un cliente que realiza peticiones a otro programa (el servidor) el cual le da respuesta.

En el modelo cliente-servidor, el cliente envía un mensaje solicitando un determinado servicio (hace una petición) a un servidor, y este envía uno o varios mensajes con la respuesta (provee el servicio). En un sistema distribuido cada máquina puede cumplir el rol de servidor para algunas tareas y el rol de cliente para otras.

Características del modelo cliente servidor.

- Combinación de un cliente que interactúa con el usuario, y un servidor que interactúa con los recursos compartidos. El proceso del cliente proporciona la interfaz entre el usuario y el resto del sistema. El proceso del servidor actúa como un motor de software que maneja recursos compartidos tales como bases de datos, impresoras, módems, etc.
- Las tareas del cliente y del servidor tienen diferentes requerimientos en cuanto a recursos de cómputo como velocidad del procesador, memoria, velocidad y capacidades del disco.
- Se establece una relación entre procesos distintos, los cuales pueden ser ejecutados en la misma máquina o en máquinas diferentes distribuidas a lo largo de la red.
- La relación establecida puede ser de muchos a uno, en la que un servidor puede dar servicio a muchos clientes, regulando su acceso a recursos compartidos.
- Los clientes corresponden a procesos activos en cuanto a que son éstos los que hacen peticiones de servicios a los servidores. Estos últimos tienen un carácter pasivo ya que esperan las peticiones de los clientes.
- No existe otra relación entre clientes y servidores que no sea la que se establece a través del intercambio de mensajes entre ambos. El mensaje es el mecanismo para la petición y entrega de solicitudes de servicio.

- El ambiente es heterogéneo. La plataforma de hardware, el sistema operativo del cliente y del servidor no son siempre la misma. Precisamente una de las principales ventajas de esta arquitectura es la posibilidad de conectar clientes y servidores independientemente de sus plataformas.

Ventajas del modelo cliente servidor.

- El esquema cliente servidor facilita la integración entre sistemas diferentes y comparte información, permitiendo, por ejemplo que las máquinas ya existentes puedan ser utilizadas pero a través de interfaces más amigables al usuario. De esta manera, podemos integrar PCs con sistemas medianos y grandes, sin necesidad de que todos tengan que utilizar el mismo sistema operacional.
- Al favorecer el uso de interfaces gráficas interactivas, los sistemas construidos bajo este esquema tienen mayor interacción y más intuitiva con el usuario. En el uso de interfaces gráficas para el usuario, el esquema cliente servidor presenta la ventaja, con respecto a uno centralizado, de que no es siempre necesario transmitir información gráfica por la red pues esta puede residir en el cliente, lo cual permite aprovechar mejor el ancho de banda de la red.
- Una ventaja adicional del uso del esquema cliente servidor es que es más rápido el mantenimiento y el desarrollo de aplicaciones, pues se pueden emplear las herramientas existentes (por ejemplo los servidores de SQL).
- La estructura inherentemente modular facilita además la integración de nuevas tecnologías y el crecimiento de la infraestructura computacional, favoreciendo así la escalabilidad de las soluciones.
- El esquema cliente servidor contribuye además, a proporcionar, a los diferentes departamentos de una organización, soluciones locales, pero permitiendo la integración de la información relevante a nivel global.

2.3.2 Patrones de arquitectura.

Modelo Vista Controlador.

Para el desarrollo del módulo web se seguirá el estilo arquitectónico Modelo Vista Controlador que es el utilizado por el framework elegido siendo este estilo aplicable al desarrollo de cualquier aplicación

independientemente del lenguaje de programación elegido y separa los datos, la interfaz de usuario y la lógica de control en tres componentes distintos:

- Modelo
- Vista
- Controlador

La capa Modelo es la representación específica de la información con la cual el sistema opera; es la encargada de todo el acceso a los datos y las funciones; lo que es llamado lógica de negocio. Esta capa es la encargada de llevar un registro de las vistas y controladores del sistema y cada acceso a datos se pone en su función individual garantizando que si se cambia de gestor de bases de datos solo afecte a este tipo de funciones y no al resto de la aplicación maximizando de esta forma la adaptabilidad a los cambios de la aplicación.

La capa Vista es la encargada de la interacción con el usuario en aplicaciones Web mostrando la información del modelo al usuario. Tener la capa vista separada de la capa controlador permite poder realizar cambios en esta sin tener que tocar más que una parte completamente delimitada del código.

La capa Controlador es la encargada de la unión de las capas vista y modelo, esta capa es la que escucha los cambios en la vista y se los envía al modelo, el cual le regresa los datos a la vista.

Ventajas del uso del MVC.

La ventaja fundamental de esta división en capas se encuentra centrada en la medida de que cada una de estas capas puede ser sustituida sin afectar a las otras ya que provee una separación total entre la lógica del negocio y presentación, además pueden existir diferentes vistas para un mismo modelo trayendo como resultado que la división de código de este estilo arquitectónico haga más fácil la portabilidad y la adaptación a los requerimientos del usuario. Al existir la separación de vistas, controladores y modelos es más sencillo realizar labores de mejora como:

- Agregar nuevas vistas.
- Agregar nuevas formas de recolectar las ordenes del usuario

- Modificar los objetos de negocios bien sea para mejorar el rendimiento o para migrar la tecnología.
- Las labores de mantenimiento se simplifican.
- Las correcciones solo se deben hacer en un solo lugar.

2.4 Fase de exploración.

La metodología de desarrollo Extreme Programming comienza con la fase de exploración. Durante esta se realiza el proceso de identificación de las historias de usuario, así como la familiarización de los equipos de trabajo con las tecnologías y herramientas seleccionadas para la construcción del proyecto. (11)

2.4.1 Flujo de procesos del sistema propuesto.

El técnico de ETECSA luego de reparar una interrupción telefónica llama desde el abonado reparado a un número de servicio donde será atendido por un software IVR el cual le comunicará el número desde donde está llamando y le pedirá que inserte su clave identificativa, si la clave es correcta se le indicará que grabe un mensaje de voz; en caso de que la misma no sea correcta se le permitirá rectificarla hasta tres veces (en caso de no introducirlas correctamente en ninguno de los tres intentos posibles, se bloqueará el número del cual esté llamando por tres minutos); una vez concluida la grabación se le realizará una llamada reproduciéndole el mensaje grabado, de esta forma se puede comprobar además de la calidad de la fonía, la del timbrado. Si el mensaje se reprodujo con la calidad deseada se procederá a insertar con la ayuda de un menú de voz los datos de la interrupción reparada sino se le indicará al técnico que continúe con la reparación.

El jefe de mesa de prueba verifica las reparaciones realizadas por los técnicos e interactúa con el sistema pudiendo en el mismo cambiar los estados de las interrupciones y eliminarlas en caso de que no se desee tener registro de las mismas, así como exportar los datos de las interrupciones visualizadas y realizar búsquedas según criterios predefinidos.

El jefe del departamento de recursos humanos interactúa con el sistema que le permite insertar los datos de los técnicos de ETECSA, modificarlos, realizar una búsqueda y eliminarlos en caso de que ya no trabajen en la entidad.

Los diagramas de procesos asociados al sistema se exponen en el anexo: Diagramas de procesos del sistema.

2.4.2 Personas relacionadas con el sistema.

Se define como persona relacionada al sistema toda aquella que obtiene un resultado o interactúa con el sistema de una forma u otra.

Tabla 1: Personas relacionadas con el sistema.

Personas Relacionadas con el Sistema	Justificación.
Técnico.	Es el encargado de insertar el reporte de cierre de interrupciones en el sistema.
Jefe de mesa de pruebas.	Es el encargado de comprobar las reparaciones y gestionar las mismas
Jefe del departamento de recursos humanos.	Es el encargado de gestionar la información de los técnicos.

2.4.3 Historias de usuario.

Las historias de usuario son la forma en que se especifican en XP los requisitos del sistema. Estas se escriben desde la perspectiva del cliente aunque los desarrolladores pueden brindar también su ayuda en la identificación de las mismas. El contenido de estas debe ser concreto y sencillo. Durante la fase de exploración se identificaron once historias de usuario, las cuales se detallan a continuación. (11)

Tabla 2: Autenticar a un técnico a través del teléfono.

Historia de usuario.	
Número: 1.	**Nombre historia de usuario:** Autenticar a un técnico a través del teléfono.
Usuario: Técnico.	**Iteración asignada:** 1.

Prioridad en negocio: Muy Alta.	**Riesgo en desarrollo:** Alto.
Puntos estimados: 2/5.	**Puntos reales:** 2/5.
Descripción: El técnico marca a través del teléfono su clave personal (Clave IVR) y única (la cual es numérica y asignada al momento de insertar el técnico) con la cual el sistema lo reconoce y autentica.	
Observaciones: De ocurrir un error en el proceso de autenticación, como es el caso de la introducción de una clave incorrecta, se reproducirá un mensaje de error y se le permitirá al técnico reinsertar su clave hasta dos veces más. Si inserta la clave tres veces mal se bloqueará el número de servicio cinco minutos para el teléfono de donde está llamando el técnico.	

Tabla 3: Comprobar la calidad de la fonía.

Historia de usuario	
Número: 2.	**Nombre historia de usuario:** Comprobar la calidad de la fonía.
Usuario: Técnico.	**Iteración asignada:** 1.
Prioridad en negocio: Muy Alta.	**Riesgo en desarrollo:** Alto.
Puntos estimados: 2/5.	**Puntos reales:** 2/5.
Descripción: El técnico, una vez autenticado graba un mensaje de voz, y al concluir el mismo cuelga. A continuación se le debe realizar una llamada reproduciéndole el mensaje grabado.	
Observaciones: En el caso de que este mensaje no tenga la calidad deseada el técnico cuelga y continúa con la reparación. En caso de ocurrir lo anterior el sistema guardará un ticket de cierre de interrupción en estado “Pendiente” así quedará la constancia de que el técnico estuvo trabajando.	

Tabla 4: Insertar reporte de cierre de interrupción.

Historia de usuario

Número: 3.	**Nombre historia de usuario:** Insertar reporte de cierre de interrupción.
Usuario: Técnico.	**Iteración asignada:** 1.
Prioridad en negocio: Muy Alta.	**Riesgo en desarrollo:** Alto.
Puntos estimados: 2/5.	**Puntos reales:** 2/5.
Descripción: El técnico una vez autenticado y después de comprobar la calidad de la fonía satisfactoriamente, introduce los datos correspondientes al tipo de rotura reparada por él, auxiliado de un IVR (el menú IVR se encuentra definido en el Anexo 1), una vez recogidos los datos, se le reproduce al técnico un mensaje con los datos insertados por él, y de ser correctos, el reporte es almacenado.	
Observaciones: • En caso de presentarse un error en la entrada de los datos, el sistema brindará la posibilidad de regresar al menú para corregirlo. • Si durante la reproducción del mensaje que contiene los datos insertados el técnico se da cuenta que cometió un error puede regresar al menú pulsando el carácter *. • En caso de no presionar ningún caracter o de introducir uno que no se encuentre en el menú del IVR, se repetirá el mismo nivel del menú.	

Tabla 5: Modificar el estado de un reporte de interrupción.

Historia de usuario	
Número: 4.	**Nombre historia de usuario:** Modificar el estado de un reporte de interrupción.
Usuario: Jefe de mesa de pruebas.	**Iteración asignada:** 2.
Prioridad en negocio: Alta.	**Riesgo en desarrollo:** Medio.
Puntos estimados: 3/5.	**Puntos reales:** 3/5.
Descripción: La interrupción seleccionada, originalmente en el estado de “Abierto”, se cambia de estado a: “Cerrado” en caso de que haya sido reparada correctamente la extensión telefónica mencionada en dicho reporte de interrupción, “Cerrado insatisfactorio” en caso de continuar con problemas, o	

"Pendiente" en el caso de que el técnico no lograse arreglar la interrupción.
Observaciones: • Esta verificación de si fue o no bien reparada la extensión, se efectúa por parte del personal encargado de comprobar el reporte, realizando una llamada a la extensión mencionada en dicho reporte. • La opción de "Modificar estado" no será habilitada, hasta tanto no se seleccione un reporte de interrupción.

Tabla 6: Eliminar un reporte de interrupción.

Historia de usuario	
Número: 5.	**Nombre historia de usuario:** Eliminar un reporte de interrupción.
Usuario: Jefe de mesa de pruebas.	**Iteración asignada:** 2.
Prioridad en negocio: Alta.	**Riesgo en desarrollo:** Medio.
Puntos estimados: 3/5.	**Puntos reales:** 3/5.
Descripción: Una vez atendida una interrupción, de no necesitarse más el registro de la misma (ya que el mismo se almacena en el SIPREC) se selecciona a partir de la lista que muestra de todas las interrupciones que existen o a partir de una búsqueda realizada anteriormente dado unos criterios establecidos por el usuario y se elige la opción de "Eliminar". A continuación se mostrará un mensaje de confirmación de eliminación, y una vez seleccionada la opción de confirmación, se muestra un mensaje de información como que la acción se ejecutó correctamente.	
Observaciones: El botón de "Eliminar" no será habilitado hasta tanto no se seleccione una interrupción.	

Tabla 7: Exportar la información de los reportes de interrupción.

Historia de usuario	
Número: 6.	**Nombre historia de usuario:** Exportar la información de los reportes de

	interrupción.
Usuario: Jefe de mesa de pruebas.	**Iteración asignada:** 2.
Prioridad en negocio: Alta.	**Riesgo en desarrollo:** Alto.
Puntos estimados: 4/5.	**Puntos reales:** 4/5.
Descripción: Se exporta toda la información en los reportes de interrupciones (Teléfono reparado, Clave de cierre (este valor indica el tipo de ruptura que tenía el teléfono), Estado (el cual puede ser: Abierto, Cerrado, Cerrado Insatisfactorio o Pendiente), Fecha y hora en que ocurrió la reparación), al formato seleccionado (CSV o XSL) y a la dirección especificada.	
Observaciones:	

Tabla 8: Mostrar interrupciones según un criterio de búsqueda.

Historia de usuario	
Número: 7.	**Nombre historia de usuario:** Mostrar interrupciones según un criterio de búsqueda.
Usuario: Jefe de mesa de pruebas.	**Iteración asignada:** 2.
Prioridad en negocio: Alta.	**Riesgo en desarrollo:** Medio.
Puntos estimados: 3/5.	**Puntos reales:** 3/5.
Descripción: Se muestra una lista con las interrupciones coincidentes con los campos de búsqueda, los cuales incluyen un periodo de tiempo (Desde, Hasta), el estado de la interrupción o una coincidencia textual en cualquiera de los otros campos (Teléfono reparado, Clave de cierre o el Nombre del técnico que efectuó la reparación).	
Observaciones: En caso de no haber ninguna coincidencia, se mostrará la lista vacía. La búsqueda se debe realizar, utilizando como mínimo uno de los criterios existentes.	

Tabla 9: Insertar datos de un técnico.

Historia de usuario	
Número: 8.	**Nombre historia de usuario:** Insertar datos de un técnico.
Usuario: Jefe de del departamento de recursos humanos.	**Iteración asignada:** 3.
Prioridad en negocio: Media.	**Riesgo en desarrollo:** Medio.
Puntos estimados: 3/5.	**Puntos reales:** 3/5.
Descripción: Al seleccionar la opción: “Adicionar” se muestra una interfaz que permite introducir los datos del nuevo técnico, los cuales son: el nombre (Nombre), apellidos (Apellidos), el carnet de identidad (Carnet de identidad), el número del móvil (Móvil), se selecciona el nivel de escolaridad (Nivel de escolaridad), el territorio al que pertenece (Territorio), la clave para autenticarse en el SIPREC (Clave SIPREC), la clave que utilizará para autenticarse en el sistema IVR (Clave IVR) y la matrícula del vehículo que maneja (Matrícula), en caso de tenerlo.	
Observaciones: • Todos los campos son obligatorios, a excepción de la matrícula del vehículo. • La clave SIPREC guarda una relación con el territorio al que pertenece el técnico, por ejemplo, un técnico de Pinar del Río tendría una clave como: **PH**123, **PH** hace referencia a la provincia Pinar del Río, esta clave es introducida por el encargado de gestionar los valores del técnico. En correspondencia, la clave IVR utiliza una codificación solo numérica, ejemplo, para el mismo técnico una clave válida seria: **168**123, correspondiendo el **168** al código de su provincia, esta clave se generará automáticamente después de introducir la clave SIPREC, manteniendo constante la parte de la clave que no se refiere a la provincia y sustituyendo la otra por su correspondiente numérico. • Los datos insertados deben tener en caso de la matrícula 6 caracteres y en caso del carnet de identidad 11. La clave IVR debe tener entre 4 y 7 caracteres y la clave SIPREC debe tener entre 4 y 5 caracteres. • En caso de haber algún problema con los datos del técnico (como es la falta de un campo	

obligatorio o la introducción de un dato no válido) se marcará el campo en el que se ha cometido el error con una línea de color rojo, y al pasar el mouse sobre el mismo se mostrará una notificación con la descripción del error.

Tabla 10: Modificar datos de un técnico.

Historia de usuario	
Número: 9.	**Nombre historia de usuario:** Modificar datos de un técnico.
Usuario: Jefe de del departamento de recursos humanos.	**Iteración asignada:** 3.
Prioridad en negocio: Media.	**Riesgo en desarrollo:** Medio.
Puntos estimados: 3/5.	**Puntos reales:** 3/5.
Descripción: Se muestran y se da la posibilidad de modificar los datos del técnico seleccionado (Nombre, Apellidos, Carnet de identidad, Móvil, Clave IVR, Clave SIPREC, Territorio, Nivel de escolaridad y Matrícula) una vez que se elija la opción “Modificar”, si no se selecciona ningún técnico la opción de “Modificar” debe estar inhabilitada.	
Observaciones: En caso de haber algún problema con los datos del mismo (como es la falta de un campo obligatorio o la introducción de un dato no válido) se marcará el campo en el que se ha cometido el error con una línea de color rojo, y al pasar el mouse sobre el mismo se mostrará una notificación con la descripción del error.	

Tabla 11: Eliminar los datos de un técnico.

Historia de usuario	
Número: 10.	**Nombre historia de usuario:** Eliminar los datos de un técnico.
Usuario: Jefe de del departamento de recursos humanos.	**Iteración asignada:** 3.

Prioridad en negocio: Media.	**Riesgo en desarrollo:** Medio.
Puntos estimados: 3/5.	**Puntos reales:** 3/5.
Descripción: Se selecciona un técnico a partir de la lista que se muestra de todos los técnicos que existen o a partir de una búsqueda realizada anteriormente dado unos criterios establecidos por el usuario y se elige la opción de “Eliminar”. A continuación se mostrará un mensaje de confirmación de eliminación, y una vez seleccionada la opción de confirmación pues se muestra un mensaje de información como que la acción se ejecutó correctamente.	
Observaciones: Solo se habilitará la opción de “Eliminar” si existe un técnico seleccionado.	

Tabla 12: Mostrar técnicos según un criterio de búsqueda.

Historia de usuario	
Número: 11.	**Nombre historia de usuario:** Mostrar técnicos según un criterio de búsqueda.
Usuario: Jefe de del departamento de recursos humanos.	**Iteración asignada:** 3.
Prioridad en negocio: Media.	**Riesgo en desarrollo:** Medio.
Puntos estimados: 3/5.	**Puntos reales:** 3/5.
Descripción: Se muestra una lista con los técnicos que posean alguna coincidencia en sus campos (Nombre, Apellidos, Carnet de identidad, Clave SIPREC, Móvil, Territorio, Nivel de escolaridad y Medio de transporte) con el valor del campo “Buscar”.	
Observaciones: En caso de no haber ninguna coincidencia, debe mostrarse un mensaje donde se comunique que no se encontró ningún resultado a partir de los criterios de búsquedas proporcionados.	

2.5 Planificación.

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

2.5.1 Estimación de esfuerzo por historias de usuario.

Para el desarrollo del sistema propuesto en este trabajo se realizó una estimación del esfuerzo para cada una de las historias de usuario identificadas, llegándose a los resultados que se muestran a continuación.

Tabla 13: Estimación de esfuerzo por historia de usuario.

Historia de usuario	Puntos estimados
Autenticar a un técnico a través del teléfono.	2/5
Comprobar la calidad de la fonía.	2/5
Insertar reporte de cierre de interrupción.	2/5
Modificar el estado de un reporte de interrupción.	3/5
Eliminar un reporte de interrupción.	3/5
Exportar la información de los reportes de interrupción.	4/5
Mostrar interrupciones según un criterio de búsqueda.	3/5
Insertar datos de un técnico.	3/5
Modificar datos de un técnico.	3/5
Eliminar los datos de un técnico.	3/5
Mostrar técnicos según un criterio de búsqueda.	3/5

2.5.2 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 la planificación de la etapa de implementación del proyecto. En base a lo antes mencionado se decide realizar esta en tres iteraciones, las cuales se detallan a continuación.

Iteración 1.

Esta iteración tiene como objetivo la implementación de las historias de usuario con prioridad muy alta. Durante el transcurso de la misma se creará la base de la arquitectura del sistema con una funcionalidad mínima haciendo mayor énfasis en la implementación del IVR. Al final de esta se contará con una primera versión de prueba, la cual será mostrada al cliente con el objetivo de obtener una retroalimentación para el grupo de trabajo.

Iteración 2.

El objetivo de esta es la implementación de las historias de usuario de prioridad alta. Al finalizar se contará con una versión de prueba con las funcionalidades concernientes al módulo de interrupciones el cual permitirá visualizar, modificar, eliminar y exportar las interrupciones reparadas por los técnicos. Esta será mostrada al cliente con el objetivo de realizar cambios necesarios en base a la opinión del mismo.

Iteración 3.

Durante el transcurso de esta se implementaron las historias de usuario de prioridad media. Al finalizar la misma se constará de la versión 1.0 del producto final, adicionando todo lo concerniente a la gestión de los técnicos. Como resultado de esta, el sistema será puesto en funcionamiento durante un período de tiempo para evaluar su desempeño.

Plan de duración de las iteraciones.

Como parte del ciclo de vida de un proyecto utilizando XP se crea el plan de duración de cada una de las iteraciones, en este caso se hace para el único equipo de desarrollo con que se cuenta. Este plan se

encarga de mostrar las historias de usuario que serán abordadas en cada una de las iteraciones, así como la duración estimada de estas últimas y el orden en que se implementaran las UH. (11)

Tabla 14: Equipo de desarrollo #1.

Iteración	Orden de la historias de usuario a implementar	Duración total de la iteración
Iteración 1	• Autenticar a un técnico a través del teléfono. • Comprobar la calidad de la fonía. • Insertar reporte de cierre de interrupción.	3 semanas.
Iteración 2	• Modificar el estado de un reporte de interrupción. • Eliminar un reporte de interrupción. • Exportar la información de los reportes de interrupción. • Mostrar interrupciones según un criterio de búsqueda.	4 semanas
Iteración 3	• Insertar datos de un técnico. • Modificar datos de un técnico. • Eliminar los datos de un técnico. • Mostrar técnicos según un criterio de búsqueda.	3 semanas

2.5.3 Plan de entregas

A continuación se presenta el plan de entregas ideado para la fase de implementación. Como producto del mismo se harán entregas del sistema al finalizar cada iteración en la fecha aproximada que se indica en la siguiente tabla.

Tabla 15: Plan de entregas.

Módulo.	Final 1ra Iteración. 3ra semana de febrero	Final 2da Iteración. 3ra semana de marzo.	Final 3ra Iteración. 3ra semana de abril.
IVR.	0.1		1.0
Interrupciones.		0.2	1.0

Técnicos.			1.0

2.6 Conclusiones parciales.

En el presente capítulo se realizó un análisis sobre los procesos llevados a cabo, vinculados al campo de acción del trabajo, enfatizando en cuales de ellos fueron objeto de automatización, además de la exposición de una descripción detallada sobre las características de la solución propuesta las fases elaboración y planificación correspondientes a la metodología de desarrollo de software .

CAPÍTULO III. DISEÑO DEL SISTEMA

3.1 Introducción.

En el presente capítulo se hace alusión a la fase de diseño propia de la metodología de desarrollo utilizada para la implementación del sistema que se propone. Se exponen además los artefactos generados durante el transcurso de la misma.

3.2 Patrones de Diseño

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular, el cual es el encargado de identificar Clases, Instancias, Roles, Colaboraciones y la distribución de Responsabilidades. El conocimiento de los patrones de diseño implementados por CEDRUX es vital para entender el funcionamiento del mismo y por ende lograr los objetivos trazados.

3.2.1 Patrones GOF que implementa el CEDRUX.

Decorador: Como complemento del CEDRUX el Zend Framework implementa este patrón, el cual es el encargado de asignarle responsabilidades a objetos de manera dinámica y añadir funcionalidades de manera dinámica.

Singleton: Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. El sistema posee una instancia única del controlador frontal disponible mediante este patrón para lograr una vía de entrada única a las solicitudes.

Facade: Permite utilizar una interfaz común para un conjunto de interfaces del módulo, haciendo que el mismo sea más fácil de usar.

Factory: Proporciona una interfaz para la creación de objetos interdependientes o interrelacionados, sin especificar sus clases concretas. El sistema utiliza este patrón en la implementación de un conjunto de clases para el acceso a datos.

3.2.2 Patrones para Asignar Responsabilidades (GRASP) que implementa el CEDRUX.

Creador: Se aplica para la asignación de responsabilidades a las clases relacionadas con la creación de objetos, de forma tal que una instancia de un objeto sólo pueda ser creada por el objeto que contiene la información necesaria para ello. El uso de este patrón permite crear las dependencias mínimas necesarias entre las clases, lo que favorece al mantenimiento del sistema y ofrece mejores oportunidades de reutilización.

Experto: Se aplica para la asignación de responsabilidades a las clases de forma tal que las mismas contengan la información necesaria para poder ejecutar una acción específica. El uso de este patrón permitirá a los objetos valerse de su propia información para hacer lo que se les pide, favorece la existencia de mínimas relaciones entre las clases, lo que permite contar con un sistema robusto y fácil de mantener.

Bajo acoplamiento: El acoplamiento mide la fuerza con que una clase está conectada a otra, de esta forma una clase con bajo acoplamiento debe tener un número mínimo de dependencia con otras clases. Las diferentes clases controladoras sólo dependen de un único controlador frontal para realizar sus funcionalidades. Este patrón se tuvo presente debido a la importancia que se le atribuye a realizar un diseño de clases independientes que puedan soportar los cambios de una manera fácil y que a su vez permitan la reutilización.

Alta cohesión: Se aplica para realizar un diseño que evite contener clases con un alto grado de abstracción, que asuman responsabilidades que podían haber delegado a otros objetos o que tengan responsabilidades muy complejas. Se tienen las clases controladoras que se encargan de ejecutar acciones de acuerdo a las peticiones que le llegan y las clases de acceso a datos que interactúan con el modelo, de forma tal que se elimina la sobrecarga de funcionalidades en las clases controladoras.

Controlador: Se aplica para realizar las asignaciones en cuanto al manejo de los eventos del sistema y definir sus operaciones.

3.3 Tarjetas de Clase, Responsabilidad y Colaboración.

Para el diseño de las aplicaciones, la metodología XP no requiere la representación del sistema mediante diagramas de clases utilizando notación UML, en su lugar se usan otras técnicas como las tarjetas CRC (12).

Esta técnica se usa para guiar el sistema a través de análisis guiados por la responsabilidad donde las clases se examinan, se filtran y se refinan en base a sus responsabilidades con respecto al sistema. El nombre de la clase se coloca a modo de título en la tarjeta, las responsabilidades se colocan a la izquierda, y las clases que se implican en cada responsabilidad a la derecha, en la misma línea que su requerimiento correspondiente. Las tarjetas CRC identificadas durante el desarrollo del sistema propuesto pueden verse en el Anexo 2.

3.4 Diseño de la base de datos.

La construcción de la base de datos es una de las tareas más importantes del diseño de las aplicaciones, pues en este se reflejan todas las relaciones y datos necesarios para el correcto funcionamiento de la aplicación. Para el desarrollo de la aplicación se identificaron seis tablas, con sus respectivas relaciones. A continuación se muestra el diagrama Entidad-Relación diseñado para la aplicación:

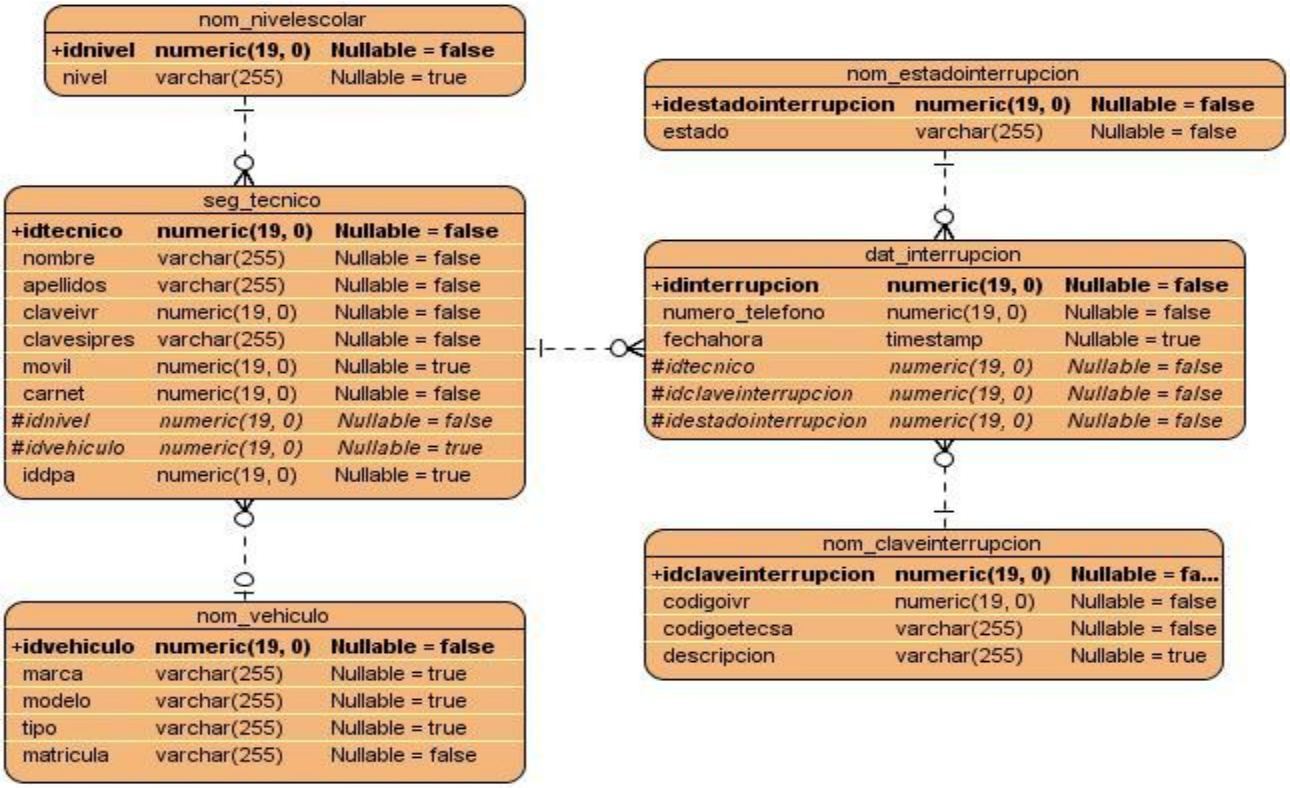

Imagen 2: Modelo Entidad Relación.

3.5 Conclusiones parciales.

Como parte del presente capítulo se abordó todo lo referente a la fase diseño del proyecto, haciendo una descripción de cada uno de los artefactos generados durante el transcurso de la misma.

CAPÍTULO IV. IMPLEMENTACIÓN Y PRUEBAS.

4.1 Introducción.

XP plantea que la implementación de un software debe realizarse de forma iterativa, obteniendo al culminar cada iteración un producto funcional que debe ser probado y mostrado al cliente para retroalimentar a los desarrolladores con la opinión de este. En el presente capítulo se detallan las tres iteraciones llevadas a cabo durante la etapa de construcción del sistema, exponiendo las tareas generadas por cada historia de usuario, así como las pruebas de aceptación efectuadas sobre el proyecto.

4.2 Iteración I.

Durante esta iteración se abordaron las tareas referentes a las historias de usuarios de mayor prioridad, con el fin de obtener una rápida retroalimentación con el cliente.

Tabla 16: Tiempo de las tareas abordadas en la iteración.

Tareas criticas	**Tiempo estimado**	**Tiempo real**
Implementar las funcionalidades que permitan autenticar a los técnicos a través del teléfono.	1/5	1/5
Implementar las funcionalidades que permitan comprobar la calidad de la fonía.	1/5	1/5
Implementar las funcionalidades que permitan insertar los datos de un reporte de cierre de interrupción.	1/5	1/5
Total:	**3/5**	**3/5**

4.2.1 Tareas abordadas en la iteración I.

Tabla 17: Tarea de la historia de usuario: Autenticar técnico a través del teléfono.

Tarea de ingeniería.	
Número de la tarea: 1.	**Número de historia de usuario:** 1.
Nombre de la tarea: Implementar las funcionalidades que permitan autenticar a los técnicos a través del teléfono.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 1/5.
Fecha de inicio: 19/02/2011.	**Fecha de fin:** 19/02/2011.
Programador responsable: Denia Margarita Delgado Valenzuela y Javier Romero Rodríguez.	
Descripción: Se crea un AGI capaz de capturar a través de DTMF la clave IVR de los técnicos de ETECSA y comprobar su autenticidad, permitiéndoles el acceso al sistema, en caso de ser correcta, en otro caso se permite la reinserción de la clave un máximo de 3 veces, bloqueando el teléfono si no introduce la clave correctamente.	

Tabla 18: Tarea de la historia de usuario: Comprobar la calidad de la fonía.

Tarea de ingeniería.	
Número de la tarea: 2.	**Número de historia de usuario:** 2.
Nombre de la tarea: Implementar las funcionalidades que permitan comprobar la calidad de la fonía.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 1/5.
Fecha de inicio: 20/02/2011.	**Fecha de fin:** 20/02/2011.
Programador responsable: Denia Margarita Delgado Valenzuela y Javier Romero Rodríguez.	
Descripción: Se crea una funcionalidad que permite: almacenar un mensaje de voz, luego, llama al	

abonado en cuestión y le reproduce dicho mensaje; el técnico indica a través de discado DTMF la calidad del mensaje escuchado.

Tabla 19: Tarea de la historia de usuario: Insertar los datos de un reporte de cierre de interrupción.

Tarea de ingeniería.	
Número de la tarea: 3.	**Número de historia de usuario:** 3.
Nombre de la tarea: Implementar las funcionalidades que permitan insertar los datos de un reporte de cierre de interrupción.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 1/5.
Fecha de inicio: 21/02/2011.	**Fecha de fin:** 21/02/2011.
Programador responsable: Denia Margarita Delgado Valenzuela y Javier Romero Rodríguez.	
Descripción: Se crea un AGI capaz de reproducir, en el abonado, un menú de voz con los datos correspondientes a las reparaciones, y permite seleccionar a través de discado DTMF la reparación realizada. Luego reproduce un mensaje con los datos: tipo de reparación, abonado reparado, fecha y hora de la reparación, e inserta dichos valores en la bese de datos.	

4.3 Iteración II.

Durante el transcurso de la presente iteración se procedió a elaborar las tares de implementación para las funcionalidades concernientes a la gestión de las interrupciones vía Web.

Tabla 20: Tiempo de las tareas abordadas en la iteración.

Tareas medias	**Tiempo estimado**	**Tiempo real**
Implementar las funcionalidades que permitan modificar el estado de un reporte de interrupción.	2/5	2/5

Implementar las funcionalidades que permitan eliminar un reporte de interrupción.	2/5	2/5
Implementar las funcionalidades que permitan exportar la información de los reportes de interrupciones.	2/5	2/5
Implementar las funcionalidades que permitan mostrar interrupciones según un criterio de búsqueda.	2/5	2/5
Total:	**8/5**	**8/5**

4.3.1 Tareas abordadas en la iteración II.

Tabla 21: Tarea de la historia de usuario: Modificar el estado de un reporte de interrupción.

Tarea de ingeniería.	
Número de la tarea: 4.	**Número de historia de usuario:** 4.
Nombre de la tarea: Implementar las funcionalidades que permitan modificar el estado de un reporte de interrupción.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 2/5.
Fecha de inicio: 21/02/2011.	**Fecha de fin:** 22/02/2011.
Programador responsable: Denia Margarita Delgado Valenzuela y Javier Romero Rodríguez.	
Descripción: Se crea un formulario en la clase interfaz, en el cual se cargan los posibles estados de una interrupción (Abierto, Cerrado, Cerrado insatisfactorio, Pendiente), y se valida que no se envíen al servidos datos incorrectos, ejemplo campos vacíos, además se implementa una función encargada de enviar los datos (Identificador de la interrupción, e identificador del estado seleccionado) a la clase controladora. Se implementa un método en la clase controladora encargado de recoger los datos que se envían desde la clase interfaz, y de ahí se envían a la clase modelo.	

En la clase modelo se crea un método que ejecuta la consulta encargada de modificar el estado de la interrupción en la base de datos.

Tabla 22: Tarea de la historia de usuario: Eliminar un reporte de interrupción.

Tarea de ingeniería.	
Número de la tarea: 5.	**Número de historia de usuario:** 5.
Nombre de la tarea: Implementar las funcionalidades que permitan eliminar un reporte de interrupción.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 2/5.
Fecha de inicio: 22/02/2011.	**Fecha de fin:** 24/02/2011.
Programador responsable: Denia Margarita Delgado Valenzuela y Javier Romero Rodríguez.	
Descripción: Se crea en la clase interfaz una función que permite enviar los datos (Identificador de la interrupción) a la clase controladora. En la clase controladora se recogen los datos que se envían desde la clase interfaz, y de ahí se envían a la clase modelo. En la clase modelo se crea un método que ejecuta la consulta encargada de eliminar el reporte de interrupción en la base de datos.	

Tabla 23: Tarea de la historia de usuario: Exportar la información de los reportes de interrupciones.

Tarea de ingeniería.	
Número de la tarea: 6.	**Número de historia de usuario:** 6.
Nombre de la tarea: Implementar las funcionalidades que permitan exportar la información de los reportes de interrupciones.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 2/5.
Fecha de inicio: 22/02/2011.	**Fecha de fin:** 24/02/2011.

Programador responsable: Denia Margarita Delgado Valenzuela y Javier Romero Rodríguez.
Descripción: Se crea en la clase interfaz una función que permite enviar los datos (pueden variar ya que la persona encargada de exportar puede decidir las columnas que va a exportar y el orden de las mismas) que se desean exportar a la clase controladora. En la clase controladora se recogen los datos que se envían desde la clase interfaz y se crea el fichero en dependencia del formato que se haya elegido en la interfaz.

Tabla 24: Tarea de la historia de usuario: Mostrar interrupciones según un criterio de búsqueda.

Tarea de ingeniería.	
Número de la tarea: 7.	**Número de historia de usuario:** 7.
Nombre de la tarea: Implementar las funcionalidades que permitan mostrar interrupciones según un criterio de búsqueda.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 2/5.
Fecha de inicio: 28/02/2011.	**Fecha de fin:** 01/03/2011.
Programador responsable: Denia Margarita Delgado Valenzuela y Javier Romero Rodríguez.	
Descripción: Se crea en la clase interfaz una función que permite enviar los criterios de búsqueda (coincidencia textual, periodo de tiempo y estado de la interrupción) a la clase controladora. En la clase controladora se recogen los datos que se envían desde la clase interfaz, y de ahí se envían a la clase modelo. En la clase modelo se crea un método que ejecuta la consulta encargada obtener todos los datos que coincidan con los criterios de búsqueda.	

4.4 Iteración III.

En el transcurso de esta iteración se implementaron las historias de usuario restantes que involucraban las funcionalidades concernientes a la gestión de los técnicos. Al culminar esta, se consta de un producto listo para su puesta en funcionamiento.

Tabla 25: Tiempo de las tareas abordadas en la iteración.

Tareas bajas	**Tiempo estimado**	**Tiempo real**
Implementar las funcionalidades que permitan insertar un técnico	2/5	2/5
Implementar las funcionalidades que permitan modificar los datos de un técnico.	2/5	2/5
Implementar las funcionalidades que permitan eliminar un técnico.	2/5	2/5
Implementar las funcionalidades que permitan buscar un técnico.	2/5	2/5
Total:	**8/5**	**8/5**

4.4.1 Tareas abordadas en la iteración III.

Tabla 26: Tarea de la historia de usuario: Insertar un técnico.

Tarea de ingeniería.	
Número de la tarea: 8.	**Número de historia de usuario:** 8.
Nombre de la tarea: Implementar las funcionalidades que permitan insertar un técnico.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 2/5.

Fecha de inicio: 30/03/2011.	**Fecha de fin:** 01/04/2011.
Programador responsable: Denia Margarita Delgado Valenzuela y Javier Romero Rodríguez.	
Descripción: Se crea en la clase interfaz una función que permite introducir los datos del técnico (Nombre, Apellidos, Carnet de identidad, Número del móvil, Nivel de escolaridad, Territorio al que pertenece, Clave para autenticarse en el SIPREC, Clave que utilizará para autenticarse en el sistema IVR y la Matrícula del vehículo que maneja, en caso de tenerlo) y enviarlos a la clase controladora. En la clase controladora se recogen los datos que se envían desde la clase interfaz, y de ahí se envían a la clase modelo. En la clase modelo se crea un método que ejecuta la consulta encargada almacenar los datos del técnico en la base de datos.	

Tabla 27: Tarea de la historia de usuario: Modificar los datos de un técnico.

Tarea de ingeniería.	
Número de la tarea: 9.	**Número de historia de usuario:** 9.
Nombre de la tarea: Implementar las funcionalidades que permitan modificar los datos de un técnico.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 2/5.
Fecha de inicio: 01/04/2011.	**Fecha de fin:** 04/04/2011.
Programador responsable: Denia Margarita Delgado Valenzuela y Javier Romero Rodríguez.	
Descripción: Se crea en la clase interfaz una función que permite cargar los datos de un técnico seleccionado para introducir los nuevo datos del técnico (Nombre, Apellidos, Carnet de identidad, Número del móvil, Nivel de escolaridad, Territorio al que pertenece, Clave para autenticarse en el SIPREC, Clave que utilizará para autenticarse en el sistema IVR y la Matrícula del vehículo que maneja, en caso de tenerlo) y enviarlos a la clase controladora. En la clase controladora se recogen los datos que se envían desde la clase interfaz, y de ahí se envían a la clase modelo.	

En la clase modelo se crea un método que ejecuta la consulta encargada almacenar los datos del técnico en la base de datos.

Tabla 28: Tarea de la historia de usuario: Eliminar un técnico.

Tarea de ingeniería.	
Número de la tarea: 10.	**Número de historia de usuario:** 10.
Nombre de la tarea: Implementar las funcionalidades que permitan eliminar un técnico.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 2/5.
Fecha de inicio: 04/04/2011.	**Fecha de fin:** 05/04/2011.
Programador responsable: Denia Margarita Delgado Valenzuela y Javier Romero Rodríguez.	
Descripción: Se crea en la clase interfaz una función que permite enviar el identificador de un técnico a la clase controladora. En la clase controladora se crea un método que envía este dato a la clase modelo, y en esta se válida la existencia del técnico, y se elimina.	

Tabla 29: Tarea de la historia de usuario: Buscar un técnico.

Tarea de ingeniería.	
Número de la tarea: 11.	**Número de historia de usuario:** 11.
Nombre de la tarea: Implementar las funcionalidades que permitan buscar un técnico.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 2/5.
Fecha de inicio: 05/04/2011.	**Fecha de fin:** 07/04/2011.
Programador responsable: Denia Margarita Delgado Valenzuela y Javier Romero Rodríguez.	
Descripción: Se crea en la clase interfaz una función que permite enviar el criterio de búsqueda a la	

clase controladora. En la clase *controladora se crea un método que envía este dato a la clase modelo, y en esta se crea una función que encuentra los técnicos cuyos datos (Nombre, Apellidos, Carnet de identidad, Clave SIPREC, Móvil, Territorio,* Nivel de escolaridad y Medio de transporte) se hallen coincidencias con el valor enviado.

4.5 Pruebas

Uno de los pilares fundamentales de XP es el proceso de pruebas (13), el cual anima a los desarrolladores a probar constantemente y tanto como sea posible. Mediante esta filosofía se reduce el número de errores no detectados así como el tiempo entre la introducción de este en el sistema y su detección (14). Todo esto contribuye a elevar la calidad de los productos desarrollados y la seguridad de los programadores a la hora de introducir cambios y modificaciones.

La elaboración de los casos de pruebas por historia de usuario, en la UCI, se llevan a cabo a través de el análisis de escenarios (posibles eventos que ocurrirán sobre el sistema), variables (todas los datos que introduce el usuario en el sistema) y realizando una descripción de lo que ocurre en el sistema ante estos eventos. Las pruebas realizadas a la aplicación pueden verse en el Anexo 3 y la descripción de las variables involucradas en el Anexo 4.

4.6 Conclusiones parciales.

En el presente capítulo se hizo alusión a las etapas de implementación y pruebas del software. Para ello se expusieron todos los artefactos generados, realizando una descripción de cada uno de ellos.

CAPITULO V ESTUDIO DE FACTIBILIDAD.

5.1 Introducción.

En la planificación del proceso de desarrollo de software es de vital importancia la estimación, la cual consiste en determinar con cierto grado de certeza los recursos necesarios para el desarrollo del mismo, ya sean recursos de hardware, software, esfuerzo, tiempo y costo. En el presente capítulo se realizará un estudio de factibilidad para la realización del sistema propuesto mediante una estimación de tamaño, esfuerzo y planificación necesaria para llevar a cabo el mismo.

5.2 Modelo matemático COCOMO II.

COCOMO, propuesto y desarrollado por Barry Boehm, es uno de los modelos de estimación de costos mejor documentados y utilizados. El modelo permite determinar el esfuerzo y tiempo que se requiere en un proyecto de software a partir de una medida del tamaño del mismo expresada en el número de líneas de código que se estimen generar para la creación del producto software.

5.3 Características del Proyecto.

El primer paso a llevar a cabo para la estimación del proyecto consiste en la obtención de los Puntos de Función desajustados, los cuales están dados por la suma de cada una de las entradas, las salidas y las consultas externas del sistema, así como los archivos lógicos internos y de interfaz externos. A continuación se muestran cada una de estas características aplicadas al software en cuestión.

5.3.1 Entradas externas.

Se definen como un proceso elemental mediante el cual ciertos datos cruzan la frontera del sistema desde afuera hacia adentro. En el caso particular de la aplicación propuesta se cuenta con siete entradas externas, especificadas en la siguiente tabla:

Tabla 30: Entradas externas.

Nombre de la entrada externa.	Cantidad de ficheros.	Cantidad de elementos de datos.	Clasificación (Simple, Media. Compleja)
Autenticar a un técnico a través del teléfono.	1	1	Simple
Comprobar la calidad de la fonía.	2	1	Simple
Modificar el estado de un reporte de interrupción.	1	1	Simple
Eliminar un reporte de interrupción.	1	1	Simple
Insertar datos de un técnico.	1	8	Simple
Modificar datos de un técnico.	1	8	Simple
Eliminar los datos de un técnico.	1	1	Simple
Total		21	

5.3.2 Salidas externas.

Se definen como un proceso elemental con componentes de entrada y de salida mediante el cual datos simples y datos derivados (esto es, datos que se calculan a partir de otros datos) cruzan las fronteras del sistema desde adentro hacia afuera. Las salidas externas vinculadas al proyecto se describen a continuación.

Tabla 31: Salidas externas.

Nombre de la salida externa.	Cantidad de ficheros.	Cantidad de elementos de datos.	Clasificación (Simple, Media. Compleja)
Exportar la información de los	1	4	Simple

reportes de interrupción.			
Mostrar interrupciones según un criterio de búsqueda.	1	4	Simple
Mostrar técnicos según un criterio de búsqueda.	1	8	Simple
Total		**16**	

5.3.3 Consultas externas.

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

Tabla 32: Consultas externas.

Nombre de la Petición.	Cantidad de ficheros.	Cantidad de elementos de datos.	Clasificación (Simple, Media. Compleja)
Total		**0**	

5.3.4 Archivos lógicos internos.

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

Tabla 33: Fichero lógico interno.

Nombre de la petición.	Cantidad de ficheros.	Cantidad de elementos de datos.	Clasificación (Simple, Media. Compleja)
Base Batos	1	6(tablas)	Simple.

AGI Autenticar_Tecnico	1	1(tablas)	Simple.
AGI Insertar_Interrupcion	1	6(tablas)	Simple.
Hora	1	1	Simple.
Total		**14**	

5.3.5 Archivos de interfaz externos.

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

Tabla 34: Interfaces externas.

Nombre de la interfaz externa.	Cantidad de ficheros.	Cantidad de elementos de datos.	Clasificación (Simple, Media. Compleja)
Total		**0**	

5.3.6 Puntos de función desajustados.

La siguiente tabla está basada en las características del sistema anteriormente expuestas, el producto de la cantidad existente de cada una de ellas y el peso correspondiente a las mismas dan como resultado final, los puntos de función desajustados pertenecientes al proyecto.

Tabla 35: Puntos de función desajustados.

Elementos.	Simple.		Medio.		Complejo.		Subtotal.
	No.	Peso.	No.	Peso.	No.	Peso.	
Entradas externas.	21	3	0	4	0	6	63
Salidas externas.	16	4	0	5	0	7	64
Consultas externas.	0	3	0	4	0	6	0

Fichero lógico interno.	14	7	0	10	0	15	98
Fichero interfaz interno.	0	5	0	7	0	10	0
Total (UFP):							**225**

Tabla 36: Peso del factor de complejidad. (15)

Tipo de función.	**Peso del factor de complejidad.**		
	Bajo.	**Promedio.**	**Alto.**
Entradas externas (Inputs).	3	4	6
Salidas externas (Outputs).	4	5	7
Archivo lógicos internos (Archivos).	7	10	15
Archivos externos de interfase (Interfases).	5	7	10
Consultas externas (Queries).	3	4	6

5.4 Cálculos de instrucciones.

5.4.1 Cálculo de instrucciones fuentes.

Una vez obtenida la cantidad total de puntos de función desajustados pertenecientes al proyecto, se procede a calcular la cantidad de instrucciones fuentes del mismo, para lo cual se utiliza la ecuación que se muestra seguidamente:

$$\mathbf{SLOC = UFPxratio}$$

Donde:

SLOC:Cantidad de instrucciones fuente: 13275.

UFP: Puntos de función desajustados: 225.

ratio: Conversión de puntos de función desajustados a líneas de código para el lenguaje PHP: 59.

Partiendo de la ecuación anterior se obtiene el siguiente resultado:

$\mathbf{SLOC} = 225x59$

$\mathbf{SLOC} = 13275$

$\mathbf{KSLOC} = 13.3$: Tamaño del software a desarrollar.

Las características del sistema y valores obtenidos anteriormente han sido plasmadas en la siguiente tabla:

Tabla 37: Características del sistema.

Características	Valor
Puntos de función desajustados.	225
Lenguaje (PHP).	59
Instrucciones fuentes por puntos de función.	13275 SLOC
Instrucciones fuentes.	13.3 KSLOC

5.4.2 Cálculo de esfuerzo nominal.

Posteriormente se procede al cálculo del esfuerzo nominal, ecuación que se toma como base tanto en el método de diseño preliminar, al cual se hace alusión anteriormente, como en el modelo Post arquitectura, ambos definidos por COCOMO II.

$$\mathbf{PM_{Nominal}} = \mathbf{Ax(Size)^B}$$

$$\mathbf{PM_{Nominal}} = \mathbf{2.94\,x(13.3)^1}$$

$$\mathbf{PM_{Nominal}} = \mathbf{39.1\ meses/hombre}$$

Donde:

$\mathbf{PM_{Nominal}}$:Esfuerzo nominal requerido en meses / hombres.

Size: Tamaño estimado del software en Puntos de Función sin Ajustar. (4.1 KSLOC)

A: Constante que se utiliza para capturar los efectos multiplicativos en el esfuerzo requerido de acuerdo al crecimiento de tamaño del software. (2.94).

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

$$\mathbf{B = 0.91 + 0.01\,x \sum (W_i)}$$

$\mathbf{B}$ =0.91 + 0.01 x 8.8

$\mathbf{B}$ =1

Donde:

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

Precedentes (PREC).

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

Flexibilidad de desarrollo (FLEX).

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

Cohesión del equipo (TEAM).

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

Solución de riesgos (RESL).

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

Madurez del proceso (PMAT).

El procedimiento para determinar el PMAT es establecer el porcentaje de cumplimiento de cada una de las Áreas evaluando el grado de cumplimiento de las metas correspondientes.

La siguiente tabla muestra los valores asignados a cada una de estas variables:

Tabla 38: Factores de escala.

Nombre.	**Valor.**	**Justificación.**
PREC	3.72	En Cuba, se utilizan las IVRs, pero nunca para un proceso tan complejo.
FLEX	1.01	Cuenta con alta flexibilidad en cuanto a los requerimientos establecidos inicialmente.
TEAM	1.10	El equipo de desarrollo presenta una alta cohesión.
RESL	1.41	No se identificaron riesgos.
PMAT	1.56	Se cuenta con la experiencia necesaria como para que el software cumpla con las funcionalidades requeridas.
Total(SF)	**8.8**	

Tabla 39: Factores de escala. (15)

	VLO	LO	NOM	HI	VHI	XHI
PREC	6.20	4.96	3.72	2.40	1.24	0.00
FLEX	5.07	4.05	3.04	2.03	1.01	0.00
RESL	7.07	5.65	4.24	2.83	1.41	0.00
TEAM	5.48	4.38	3.29	2.19	1.10	0.00
PMAT	7.80	6.24	4.68	3.12	1.56	0.00

5.5 Ajuste del esfuerzo nominal.

El esfuerzo calculado anteriormente es un valor nominal y debe ser ajustado en base a las características del proyecto para lo cual se tiene un conjunto de Multiplicadores de Esfuerzo (MEi) que representan las características del proyecto y expresan su impacto en el desarrollo total del producto de software.

5.5.1 Ajuste del esfuerzo nominal clasificados en categorías.

Los 7 multiplicadores de esfuerzo son:

- **Del producto.**

RCPX: Confiabilidad y Complejidad del producto

RUSE: Reusabilidad Requerida

- **De la plataforma.**

PDIF: Dificultad de la Plataforma

- **Del personal.**

PERS: Aptitud del Personal

PREX: Experiencia del Personal

- **Del Proyecto.**

FCIL: Facilidades

SCED: Cronograma de Desarrollo Requerido

Tabla 40: Multiplicadores de esfuerzo.

Nombre.	Valor.	Justificación.
RCPX	1.30	La confidencialidad del sistema es alta.
RUSE	1.00	Pretende reutilizarse una parte del código.
PDIF	1.00	Uso de memoria y almacenamiento normal, plataforma estable.
PREX	0.71	Experiencia en el personal en cuanto a la utilización del lenguaje y herramientas.
PERS	0.63	La capacidad del personal es alta.
FCIL	0.73	La utilización de entornos de desarrollo integrados facilita en gran medida el trabajo.
SCED	1.00	El sistema se desarrollo en el tiempo establecido.
Total(EM)	**0.42**	

Tabla 41: Multiplicadores de esfuerzo del modelo del diseño temprano. (15)

	XLO	VLO	LO	NOM	HI	VHI	XHI
RCPX	0.73	0.81	0.98	1.00	1.30	1.74	2.38
RUSE	XXXX	XXXX	0.95	1.00	1.07	1.15	1.24
PDIF	XXXX	XXXX	0.87	1.00	1.29	1.81	2.61
PERS	2.12	1.62	1.26	1.00	0.83	0.63	0.50
PREX	1.59	1.33	1.12	1.00	0.87	0.71	0.62
FCIL	1.43	1.30	1.10	1.00	0.87	0.73	0.62
SCED	XXXX	1.43	1.14	1.00	1.00	1.00	XXXX
USR1	XXXX	1.00	1.00	1.00	1.00	1.00	XXXX
USR2	XXXX	1.00	1.00	1.00	1.00	1.00	XXXX

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

$\mathbf{PM_{ajustado}}$ = 39.1 meses/hombre x 0.42.

$\mathbf{PM_{ajustado}}$= 16.4 meses/hombre.

5.5.2 Calculo del tiempo de desarrollo del software.

El tiempo requerido para el desarrollo del proyecto está dado por la siguiente ecuación:

$$\mathbf{TDEV = C\ x\ (PM_{ajustado})^F}$$

$$\mathbf{TDEV = 3.64\ x\ (16.4)^{0.26}}$$

$$\mathbf{TDEV = 7.53}$$

Donde:

C:Constante con valor 3.64.

$\mathbf{PM_{ajustado}}$ = 22.11meses/hombre.

F: 0.26.

CALCULAR

$$\mathbf{F = D + 0.2\ x\ 0.01 x \sum SF}$$

F =0.24 + 0.2 x 0.01 x 8.8

F =0.26

Donde:

D: Constante cuyo valor es 0.24.

SF: Valor de los factores de escala.

4.3.5 Cálculo del costo total del proyecto.

Para el cálculo del costo total correspondiente al proyecto en cuestión, COCOMO II propone la siguiente ecuación:

$$\mathbf{C = CHM \times PM}$$

$$\mathbf{C} = 218 \times 16.4.$$

$$\mathbf{C} = 3575.2$$

CALCULAR CHM.

Donde:

C:Costo total.

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

$$\mathbf{CHM = CH \times sal}$$

$$\mathbf{CHM} = 2.18 \times 100.$$

$$\mathbf{CHM} = 218.$$

CALCULAR

Donde:

sal: Salario medio por cada trabajador.

CH: Cantidad de personas destinadas al proyecto, lo que es calculado a través de la ecuación:

$$\mathbf{CH} = {}^{\mathbf{PM}}/_{\mathbf{TDEV}}$$

$\mathbf{CH} =$ 16.4 / 7.53

$\mathbf{CH} =$2.18

5.5.3 Resultados.

En la siguiente tabla se muestran los resultados obtenidos luego de haber efectuado todos los cálculos para determinar el costo y esfuerzo requeridos por el proyecto.

Tabla 42: Resultados.

Cálculo de:	**Valor**
Esfuerzo.	16.4 meses/hombre.
Tiempo de desarrollo.	8 meses.
Cantidad de hombres.	2 hombres.
Salario medio.	100 pesos.
Costo.	3575.2 pesos.

5.6 Análisis de costo.

El desarrollo de un producto siempre tiene un costo de producción, el cual debe ser justificado en base a los beneficios reportados por el mismo. El sistema que se propone en este trabajo no conlleva a grandes gastos, puesto que solo es influyente el salario de los desarrolladores, por lo cual se concluye que su implementación es factible. Esto se debe en gran medida a la utilización de herramientas libres que no requieren el pago de licencia.

5.7 Conclusiones parciales.

En el presente capítulo se realizó un análisis de factibilidad de la solución propuesta, arribando a la conclusión de que es viable su desarrollo comparando los costos de producción con los beneficios reportados por su puesta en funcionamiento.

CONCLUSIONES.

Durante el transcurso de la investigación se dio cumplimiento al objetivo trazado para ello se construyó una aplicación capaz de informatizar el proceso del cierre de interrupciones telefónicas en ETECSA, dicha aplicación consta de una IVR capaz de realizar el proceso de cierre de las interrupciones telefónicas y una aplicación web capaz de gestionar los reportes de las interrupciones, así como los datos de los técnicos. Además de esto, se validó la solución propuesta, de forma satisfactoria.

La solución reportará a ETECSA beneficios tales como: la disminución del tiempo de interacción para cerrar una interrupción, posibilidad de reubicar las operadoras en lugares más importantes donde la empresa lo requiera e introducirá a esta entidad en el mundo del software libre como parte del proceso de migración que lleva a cabo nuestro país hacia tecnologías libres.

RECOMENDACIONES.

Con el objetivo de realizar un seguimiento de este trabajo se recomienda crear un medio de comunicación entre la aplicación y el sistema existente en ETECSA. Además podría usarse la aplicación como base para la construcción de un sistema de gestión de fuerza de trabajo.

BIBLIOGRAFÍA.

1. Sitio oficial de Pronexus. [En línea] [Citado el: 17 de 12 de 2010.] http://www.pronexus.com/.

2. Sitio oficial de Avaya Inc. [En línea] [Citado el: 01 de 12 de 2010.] http://www.avaya.com/.

3. Sitio oficial de Alcatel-Lucent. [En línea] [Citado el: 08 de 01 de 2011.] http://www.alcatel-lucent.com/.

4. **Maite Rodríguez Corbea, Meylin Ordóñez Pérez.***LA METODOLOGÍA XP APLICABLE AL DESARROLLO DEL SOFTWARE EDUCATIVO EN CUBA.* Ciudad de la Habana : Universidad de las Ciencias Informáticas (UCI), 2007.

5. Sitio Oficial del Apache. [En línea] [Citado el: 24 de Noviembre de 2010.] Disponible en:[http://apache.org/].

6. Sitio oficial del proyecto Geany. [En línea] [Citado el: 09 de 01 de 2011.] http://www.geany.org/.

7. Sitio oficial de Zend Framework. [En línea] [Citado el: 09 de 01 de 2011.] http://framework.zend.com/.

8. Sitio oficial de Sencha. [En línea] [Citado el: 09 de 01 de 2011.] http://www.sencha.com/.

9. Sito oficial del proyecto Doctrine. [En línea] [Citado el: 09 de 01 de 2011.] http://www.doctrine-project.org/.

10. Sitio Oficial del Postgresql. [En línea] [Citado el: 25 de Octubre de 2010.] Disponible en: [http://www.postgresql.org].

11. **Jeffries, Let Ronald.** XPrograming. [En línea] 2011. [Citado el: 10 de 1 de 2011.] http://xprogramming.com/.

12. **Wells, Don.** Sitio oficial de eXtreme Programing. [En línea] 28 de Septiembre de 2009. [Citado el: 2 de Diciembre de 2010.] http://www.extremeprogramming.org/.

13. **Beck, Kent.***Extreme Programming Explained: Embrace Change.* s.l. : Addison-Wesley. 0201616416.

14. **Crispin, Lisa y House, Tip.***Testing Extreme Programming.*

15. **Boehm, B.W, y otros.***The COCOMO 2.0 Software Cost Estimation Model.* 1995.

16. **Wells, J. Donovan.** Extreme Programming: A gentle introduction. [En línea] 15 de Abril de 2001. http://www.extremeprogramming.org/.

17. **Fernández Escribano, Gerardo.***Introducción a Extreming Programming.* 2002.

18. Award Winning UML Tool. *Visual Paradigm for UML User's Guide.* [En línea] 1 de Agosto de 2007. http://content.europe.visual-paradigm.com/media/documents_download/vpuml61ug1/vpuml_user_guide.pdf.

19. Object Management Group/Business Process Management Initiative. *Business Process Modeling Notation (BPMN) Information.* [En línea] OMG, 2 de Febrero de 2009. http://www.bpmn.org/Documents/FAQ.htm.

20. **Ambler, Scott.** The Agil Unified Process (AUP). [En línea] 16 de Agosto de 2008. http://www.ambysoft.com/unifiedprocess/agileUP.html.

21. —. Dr. Dobb's Portal. *The Agile Edge: Unified and Agile.* [En línea] 1 de Enero de 2006. http://www.ddj.com/architect/184415460 The Agile Edge: Unified and Agile.

22. —. Agile Model Driven Development (AMDD): The Key to Scaling Agile Software Development. [En línea] 2006. http://www.agilemodeling.com/essays/amdd.htm.

23. —. *The Agile Unified Process v1.1.* 2006.

24. **Ilizastegui Arriba, Damián y Plá Rodríguez, José Antonio .***Sistema para la integración continua de proyectos y el control de builds en la empresa Procyon Soluciones.* Ciudad de la Habana : s.n., 2007.

25. **García Hernández, Alién y Viltres Ramírez, Damián .***Implementación de los Componentes Configuración, Nomencladores y Recuperaciones del subsistema Inventario del Sistema de Gestión Integral CEDRUX.* Ciudad de la Habana : s.n., 2009.

26. **Tenrero Cabrera, Marianela y Morejón Borbon, Yoandry.***Manual de instalación de las herramientas para Sauxe 1.5.* Ciudad de la Habana : s.n., 2010.

27. **Piñera, Yadira.***Manual de usuario Subsistema Estructura y Composición.* Ciudad de la Habana : s.n., 2008.

28. Sitio oficial del proyecto Apache. [En línea] [Citado el: 09 de 01 de 2011.] http://httpd.apache.org/.

29. TecnoRetales. [En línea] [Citado el: 10 de 01 de 2011.] http://www.tecnoretales.com/programacion/que-es-doctrine-orm/.

30. Sitio Oficial del MIC. [En línea] [Citado el: 5 de Octubre de 2010.] Disponible en: [http://www.mic.gov.cu/HThemEmp.aspx?4.]..

31. Universidad Peru. [En línea] [Citado el: 9 de Noviembre de 2010.] Disponible en: [www.universidadperu.com/telecomunicaciones-peru.php()]..

32. **Alexa.** bds. [En línea] junio de 2009. [Citado el: 8 de Diciembre de 2010.] http://www.proyectosbds.com/software-y-programacion-a-medida/programacion-php-lamp-zf/mas-sobre-zend-framework/121/.

33. **Doctrine.** doctrine-project. [En línea] 2007. [Citado el: 8 de Diciembre de 2010.] http://www.doctrine-project.org/. doctrine-projec.

34. Agenda Telefónica Click to Dial. [En línea] [Citado el: 10 de Noviembre de 2010.] Presentación Disponible en:[http://www.google.com.cu/#q=agendas+telefonicas+basadas+en+asterisk&hl=es&biw=1024&bih=576&ei=2V3hTKnvNaHtnQeYr_WJDw&start=10&sa=N&fp=123fa313d55e5dee].

35. Metodologías de desarrollo de software. Capítulo 2. IAGP. [En línea] 2005. [Citado el: 11 de Octubre de 2010.] Disponible en: [http://www.um.es/docencia/barzana/IAGP/Iagp2.html].

36. **Espinosa, Humberto.** PresentacionES_PSQL. [En línea] [Citado el: 11 de Noviembre de 2010.] Disponible en: [http://www.lgs.com.ve/pres/PresentacionES_PSQL.pd f].

37. saludinova. [En línea] [Citado el: 9 de Noviembre de 2010.] Disponible en: [http://www.saludinova.com/practices/view/407].

38. Sitio oficial de Asterisk PBX. [En línea] [Citado el: 14 de Noviembre de 2010.] Disponible en: [www.asterisk.org].

39. **Acuña, Karenny Brito.** Selección de Metodologías de Desarrollo para Aplicaciones WEB en la Universidad de Cienfuegos. [En línea] 2009. [Citado el: 5 de Noviembre de 2010.] Disponible en: [http://www.eumed.net/libros/2009c/584/Metodologias%20tradicionales%20y%20metodologias%20agiles.htm].

40. **Jaime Alejandro Díaz Rojas, Tamara Jazmín Ramírez Andrade.***Desarrollo de Aplicaciones y Soluciones en la Central Asterisk de Telefonía IP en el Departamento de Electrónica de la UTFSM.* Chile : Universidad Federico Santa María, 2006.

41. **Magazine, Estrategia.** ERP. [En línea] 2008. [Citado el: 16 de Noviembre de 2010.] Disponible en:[http://www.gestiopolis.com/administracion-estrategia/estrategia/que-es-erp.htm].

42. **Wells, Don.** Extreme Programming: A gentle introduction. [En línea] 2006. [Citado el: 9 de Noviembre de 2010.] Disponible en: [http://www.extremeprogramming.org].

43. **Yaidel Alfredo Carvajal Rondón, Héctor Alexander Pérez Coello.***Herramienta para el diseño y soporte online de un Sistema de Respuesta de Voz Interactiva (PLATEL-IVR).* Ciudad de La Habana : Universidad de las Ciencias Informáticas (UCI) , 2009.

44. Sistema gestor de base de datos. [En línea] 2004. [Citado el: 10 de Noviembre de 2010.] Disponible en: [http://www.error500.net/garbagecollector/archives/categorias/bases_de_datos/sistema_gestor_de_base_de_datos_sgbd.php].

45. Introducción a Apache. [En línea] [Citado el: 24 de Noviembre de 2010.] Disponible en: [http://linux.ciberaula.com/articulo/linux_apache_intro/].

46. Sitio Ocicial de PHP. [En línea] [Citado el: 25 de Noviembre de 2010.] Disponible en: [http://www.php.net].

47. **Valdés, Damián Peréz.** Los diferentes lenguajes de programación para la web. [En línea] [Citado el: 25 de Noviembre de 2010.] Disponible en: [http://www.maestrosdelweb.com/principiantes/los-diferentes-lenguajes-de-programacion-para-la-web/].

48. Lenguajes de programacion web. [En línea] [Citado el: 25 de Noviembre de 2010.] Disponible en:[http://www.larevistainformatica.com/lenguajes-programacion-web.htm].

49. **Nicolás Montero Puñales, Luis Eduardo Acosta Aparicio.***Sistema de Reportes y Facturación de PBX Asterisk.* Ciudad de La Habana : Universidad de las Ciencias Informáticas(UCI), 2009.

50. **José Antonio Plá Rodríguez, Damián Ilizastegui Arriba.***Sistema para la integración continua de proyectos y el control de builds en la empresa Procyon Soluciones.* Ciudad de la Habana : Universidad de las Ciencias Informáticas (UCI), 2007.

51. VoIP-Info. [En línea] [Citado el: 10 de 01 de 2011.] http://www.voip-info.org.

52. Sitio oficial del proyecto Asterisk. [En línea] [Citado el: 10 de 01 de 2011.] http://www.asterisk.org/.

53. **Galli, Ricardo.** Elementos éticos del Software Libre. 2006.

54. **autores, Colectivo de.** Libro Blanco del Software Libre. España : s.n., 2004.

55. **Corp., Database System.** databasesystemscorp. [En línea] [Citado el: 11 de Noviembre de 2010.] Disponible en:[http://www.databasesystemscorp.com/pssurvey.htm].

56. **Victoria.** definicionabc. [En línea] 11 de Enero de 2009. [Citado el: 12 de Noviembre de 2010.] Disponible en:[http://www.definicionabc.com/tecnologia/erp.php].

57. **Cobranza, Software de.** softwaredecobranza. [En línea] [Citado el: 12 de Noviembre de 2010.] Disponible en:[http://www.softwaredecobranza.com/interactive-voice-response.aspx].

58. **Santos, Herminio Heredia.** maestrosdelweb. [En línea] [Citado el: 16 de Noviembre de 2010.] Disponible en:[http://www.maestrosdelweb.com/editorial/phpintro/].