Universidad de las Ciencias Informáticas
Facultad 2

Título: Desarrollo del módulo comunicador del Sistema para la Gestión Integral de los Costos de Llamadas en Pizarra Telefónicas-PABX.

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS.

Autor(es): Laura Acebedo Arzuaga.

Arianna Silveira García.

Tutor: Ing. Aliennis M. González Hurtado.

Co-Tutor: Ing. Félix Maikel García.

Ciudad de La Habana, 21 de Junio de 2010

"Año 52 de la Revolución"

DECLARACIÓN DE AUTORÍA

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

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

Laura Acebedo Arzuaga

_____

Firma del Autor

Arianna Silveira García

_____

Firma del Autor

Aliennis M. González Hurtado

_____

Firma del Tutor

AGRADECIMIENTOS

A mis padres por confiar en mí y por apoyarme en los momentos más difíciles durante la carrera.

A mis tutores Félix Maikel y Aliennis por su ayuda incondicional.

A mis compañeros de clase, especialmente a los que estuvieron cerca durante estos cinco años.

A todos los integrantes del proyecto PBX que de una forma u otra nos apoyaron.

Arianna

A nuestro jefe de proyecto y amigo Félix Maikel por apoyarnos siempre que lo necesitamos y por ser un ejemplo para todos.

A nuestra tutora Aliennis por sus consejos.

A los integrantes del proyecto PBX.

Laura

DEDICATORIA

A las personas más importantes de mi vida, mis padres que gracias a su apoyo me he convertido en lo que soy hoy. A mi abuela Elsa que aunque no este conmigo, contribuyó a mi decisión de seguir superándome. A toda mi familia que la quiero con el alma y los extraño mucho.

Arianna

A mi familia por ser mi mayor soporte durante estos cinco años.

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

A todas las personas que me han brindado su apoyo en esta dura tarea que es tratar de ser mejor cada día.

Laura

RESUMEN

Las instituciones que adquieren pizarras telefónicas necesitan un software que les permita monitorear y controlar toda la información relacionada con las llamadas que se realizan. Estas son detalladas en una trama SMDR (Station Message Detail Recording) generada por la pizarra telefónica.

La diversidad de modelos de pizarras telefónicas implican diversos problemas, por ejemplo, cada tipo de pizarra genera uno o varios formatos SMDR que necesitan de herramientas que los puedan interpretar y adquirir distintos tipos de estas para cada formato SMDR resulta complejo y costoso, debido a que son privadas y no siempre la entidad tiene el presupuesto suficiente para pagar por ellas.

Por esta razón el objetivo del trabajo de diploma fue desarrollar un módulo para la obtención y almacenamiento de los datos a partir de una pizarra telefónica que permita conexión por el puerto RS-232C, para el control y monitoreo de los registros SMDR.

Se seleccionó la metodología y las herramientas que contribuyeron a complementar y organizar el proceso de desarrollo del sistema.

Al finalizar el desarrollo del módulo comunicador se espera alcanzar una herramienta que permita manejar toda la información que se obtiene de las pizarras telefónicas de forma eficiente, independientemente del modelo de las mismas o del tipo de formato de las tramas SMDR.

Palabras claves

PABX, SMDR

TABLA DE CONTENIDOS

AGRADECIMIENTOS II

DEDICATORIA III

RESUMEN 1

INTRODUCCIÓN 4

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA 8

1.1 Introducción 8

1.2 Pizarras telefónicas o PABX 8

1.2.1 Marcas y Modelos de pizarras telefónicas. 9

1.2.2 SMDR (Station Message Detail Recording) 11

1.3 Metodologías de desarrollo 13

1.4 Lenguaje Unificado de Modelado (UML) 15

1.5 Lenguajes de programación 16

1.6 Entorno de Desarrollo Integrado (IDE) 17

1.7 Herramienta Case de Modelado con UML 18

1.8 Sistema Gestor de Base de Datos (SGBD) 19

1.9 Frameworks y Componentes 20

1.10 Conclusiones 21

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA 22

2.1 Introducción 22

2.1.1 Problema y situación problémica 22

2.1.2 Objeto de automatización 22

2.1.3 Información que se maneja 23
2.2 Propuesta de sistema 23
2.3 Modelo de Dominio 23
2.4 Especificación de los requisitos del Software 25
2.4.1 Requerimientos funcionales. 25
2.4.2 Requerimientos no funcionales 29
2.5 Modelo de Casos de Uso del Sistema. 32
2.5.1 Definición de los actores del sistema a automatizar 32
2.5.2 Diagrama de paquetes 32
2.5.3 Diagrama de casos de uso del sistema 33
2.5.4 Descripción de los Casos de Uso del Sistema 35
2.6 Conclusiones 38
CAPÍTULO 3: DISEÑO E IMPLEMENTACIÓN DEL SISTEMA 39
3.1 Introducción 39
3.2 Modelo de Diseño 39
3.2.1 Diagrama de Paquete del diseño 39
3.2.2 Diagrama de clases del diseño por paquetes 40
3.2.2.1 Diagrama de clases del diseño. Paquete Configuración 40
3.2.2.2 Diagrama de clases del diseño. Paquete Administración 41
3.2.2.3 Diagrama de clase del diseño. Paquete Controlar registro SMDR 43
3.2.3 Arquitectura 44
3.2.4 Patrones 44

3.2.4.1 Patrones de arquitectura 45
3.2.4.2 Patrones de Diseño 46
3.3 Diseño de Base de Datos 49
3.3.1 Modelo lógico de Datos 49
3.3.2 Modelo físico de Datos del sistema 51
3.4 Diagrama de Despliegue 51
3.5 Diagrama de Componentes 52
3.6 Conclusiones 59
CAPÍTULO 4: ESTUDIO DE FACTIBILIDAD 60
4.1 Introducción 60
4.2 Estimación 60
4.3 Beneficios Tangibles e Intangibles. 66
4.4 Análisis de costos y beneficios 67
4.5 Conclusiones 67
CONCLUSIONES 68
RECOMENDACIONES 69
BIBLIOGRAFÍA 70
REFERENCIAS BIBLIOGRÁFICAS 73
GLOSARIO 74

INTRODUCCIÓN

Las pizarras telefónicas surgen con la necesidad de administrar el tráfico de las llamadas entre líneas internas de instituciones y la red de telefonía local pública (CTR). Estas permiten realizar llamadas más rápidas y con mayor ahorro económico por concepto de tráfico telefónico.

A nivel mundial se comercializan estas pizarras telefónicas, existiendo diversos proveedores entre los que se encuentran: MITEL, ALCATEL, PANASONIC, ERICSSON, INFINITY, TOSHIBA y LG; de los cuales existen varios modelos.

La Empresa de Telecomunicaciones de Cuba S.A. (ETECSA) desde el año 1994, ofrece la venta de pizarras telefónicas privadas. El servicio está orientado a las pequeñas, medianas y grandes entidades que tienen gran movilidad en extensas áreas de trabajo, ya sea en universidades, naves de almacén, fábricas, e incluso grandes edificaciones.

Las instituciones cubanas que adquieren las pizarras telefónicas privadas, perciben como principal objetivo la comunicación telefónica entre sus diferentes estructuras organizativas; restándole importancia al control que este proceso amerita. Es variada y detallada la información que se obtiene de los registros que generan las mismas.

Cada tipo de pizarra telefónica genera uno o varios formatos SMDR; por lo que se precisa de un mecanismo que interprete el registro SMDR y lo traduzca a un formato humanamente entendible. Adquirir y operar con una herramienta distinta para cada tipo de pizarra resultaría costoso y complejo.

Al analizar la situación existente se delinean los siguientes problemas:

- Diversidad de pizarras telefónicas instaladas en todo el país.
- Diversidad en los registros SMDR generados.
- Inexistencia de soluciones informáticas libres del pago de licencias para su uso y explotación.
- Inexistencia de una herramienta genérica que proporcione el control y monitoreo de registros SMDR, independientemente de la pizarra instalada.

- Imposibilidad para realizar un análisis periódico del consumo real y el presupuesto planificado para cada área o departamento.

Este proceso es una actividad compleja, porque requiere de la interacción entre una PBX y la computadora dedicada a recopilar los datos, por el manejo de información en tiempo real, y la prestación del servicio de forma continua.

De ahí surge la necesidad de implementar un sistema que enmiende los problemas existentes en el país, referentes a la interacción con dichas pizarras telefónicas, por lo que es imprescindible la implementación de un módulo capaz de obtener los datos de cualquier pizarra telefónica que permita conexión por el puerto RS-232C y almacenarlo en la base de datos del sistema.

Por lo anteriormente descrito se plantea el siguiente **problema científico**: ¿Cómo obtener los datos de un SMDR generado por una pizarra telefónica que permita conexión por el puerto RS-232C?

Se define como **objeto de estudio** el funcionamiento de las pizarras telefónicas.

Para esto el **campo de acción** se enmarca en los procesos para la obtención y almacenamiento de datos del SGIC de Llamadas en Pizarras Telefónicas – PABX existentes en el país.

Como **objetivo general** se define desarrollar un módulo para la obtención y almacenamiento de los datos a partir de una pizarra telefónica que permita conexión por el puerto RS-232C, para el control y monitoreo de los registros SMDR.

A partir del objetivo general se definen como **objetivos específicos** los siguientes:

- Diseñar los procesos que intervienen en la obtención de los datos brindados por la pizarra telefónica.
- Implementar un módulo que permita la obtención y almacenamiento de estos datos.

Para dar cumplimiento a lo anteriormente planteado se definen las siguientes **tareas de investigación**:

- Investigación del funcionamiento de las pizarras telefónicas existentes en Cuba para delinear sus características más significativas.

- Identificación de los mecanismos de comunicación de las pizarras telefónicas para estructurar una forma genérica de realizarla.
- Comparación de las diferentes herramientas y metodologías para definir las posibles a utilizar en el desarrollo del sistema.

Para el desarrollo de este trabajo se plantea la siguiente **idea a defender**:

La implementación del módulo Comunicador, permitirá la comunicación entre la PBX y el software para obtener y almacenar los datos enviados por la pizarra.

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

- ✓ **Métodos Teóricos:**

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

Modelación: Este método resultará importante para el sistema en cuanto a la selección de la metodología que se utilizará, ya que en la mayoría de estas se hace muy necesaria la creación de varios modelos, pues estos permitirán una reproducción ampliada de la realidad, además de que posibilitará descubrir y estudiar nuevas relaciones y cualidades del objeto de estudio.

- ✓ **Métodos Empíricos:**

Entrevista: Este método se utilizará para la realización del sistema debido a que para obtener el servicio con la calidad que requiere se realizarán una serie de entrevistas con el cliente, y sobre la base de estas se trabajará para satisfacer sus necesidades.

El trabajo de diploma está estructurado en 4 capítulos:

Capítulo 1: Contiene la **Fundamentación Teórica**, donde se hace un estudio de los diferentes modelos de pizarras telefónicas, sus características principales, conceptos fundamentales; se muestran además las diferentes herramientas y metodología a utilizar.

Capítulo 2: Se definen las **Características del Sistema**, se propone una solución a través de una descripción detallada de un modelo de dominio, y los requisitos funcionales y no funcionales del sistema.

Capítulo 3: **Diseño e implementación del sistema**, donde se modelan los diagramas de clases del diseño, el modelo de componente, el modelo de despliegue y el modelo de datos.

Capítulo 4: Estudio de factibilidad se muestra el costo estimado del proyecto y un análisis de los beneficios que reporta.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En este capítulo se presenta la fundamentación teórica, donde se exponen los distintos tipos de pizarras telefónicas, sus funciones principales, además se especifican las distintas herramientas y metodologías a utilizar para el desarrollo del sistema.

1.2 Pizarras telefónicas o PABX

PBX / PABX (***P**rivate **B**ranch E**x**change / **P**rivate **A**utomatic **B**ranch E**X**change*): Es una red telefónica privada utilizada dentro de una compañía, en la que los usuarios de la misma comparten determinada cantidad de líneas externas para recibir y realizar llamadas hacia el exterior, lo cual resulta mucho más económico que conectar una línea de teléfono externa a cada uno de los teléfonos que se utilicen en la compañía.

Una PBX actúa como una ramificación de la red primaria pública de teléfono, por lo que los usuarios no se comunican al exterior mediante líneas telefónicas convencionales, sino que al estar la PBX directamente conectado a la red telefónica pública, será esta misma la que enrute la llamada hasta su destino final mediante enlaces unificados de transporte de voz llamados líneas troncales.

Las pizarras telefónicas mantienen tres funciones esenciales: establecer llamadas (internas o externas) entre dos o más usuarios, mantener la comunicación durante el tiempo que lo requiera el usuario y proveer información para contabilidad y/o facturación de llamadas.

Existen varios tipos de llamadas telefónicas que se realizan por medio de una PBX: Llamadas entrantes, llamadas salientes y llamadas internas.

Llamadas entrantes: son las llamadas que se realizan desde líneas externas de una red telefónica pública a una central telefónica de la empresa.

Llamadas salientes: son las llamadas realizadas desde una extensión de la empresa hacia una red telefónica pública.

Llamadas internas: son llamadas realizadas entre las extensiones de la empresa que además se consideran gratuitas.

1.2.1 Marcas y Modelos de pizarras telefónicas.

En la actualidad las pizarras telefónicas más comercializadas en Cuba son: Alcatel, Mitel, Panasonic, Ericsson, Infinity, Toshiba y LG. A continuación se presentan una serie de características de algunas de ellas.

PANASONIC

Diseñados para hacer más fluidas las comunicaciones del hogar u oficina pequeña.

Modelo **KX-TES824** y **KX-TEM824:**

- Acepta una capacidad máxima de 8 líneas CO (tronco) y 24 extensiones.
- Presentan de 2 a 4 puertos para correo de voz respectivamente.

Modelo **KX-TEA308LA:**

- Ideal para negocios pequeños u oficinas en casa que requieren un sistema flexible con un alto grado de sofisticación.
- Acepta 3 líneas CO y 8 extensiones híbridas.
- Presenta 2 puertos para correo de voz.

INFINITY

Modelo **TX-2400L/XL:**

- Puede proveer hasta 3480 puertos Digitales/Analógicos sin bloqueo.
- Tiene una unidad de control de servicio (SCU) que provee funciones importantes y necesarias para la operación del sistema, mantenimiento y control.

- Cuenta con las tarjetas de Interfaz Universal de Línea (ULI) e Interfaz Universal Digital (UDI) que son empleados para acomodar las distintas aplicaciones analógicas o digitales.
- Facilita el mantenimiento remoto.

ALCATEL

Modelo **OmniPCX Office**

- Cuenta con una capacidad de 6 a 236 extensiones.
- Es un sistema que integra voz, Internet y datos.
- Es una solución modular y escalable que puede adaptarse a las necesidades específicas de negocios, organización y tecnología de la información de cada uno de los clientes.
- Permite una gran flexibilidad de configuraciones y servicios.
- Se ofrece en 3 modelos diferentes, que pueden montarse fácilmente en un armario de 19 pulgadas.

MITEL

Modelo **Mitel 3300 Cxi**

- Es la solución ideal de Telefonía IP para empresas y sucursales.
- Capaz de soportar extensiones IP o Analógicas y es de diseño compacto.

Modelo **Mitel IP-1000**

- Paquete Básico de 2 troncales y hasta 8 extensiones (4 alámbricos y 4 inalámbricos).
- Presenta puertos DSL, WAN y LAN.

ERICSSON

Modelo **Ericsson BusinessPhone 50** (BP50)

- Presenta 64 extensiones digitales, 32 extensiones analógicas y 64 extensiones inalámbricas.
- Optimizada para sistemas de telefonía digital.
- Es la solución para pequeñas y medianas empresas.

Modelo **Ericsson BusinessPhone 250** (BP250)

- Es un avanzado sistema de telefonía para pequeñas y medianas empresas.
- Construido para una mejor administración de la información, para reducir los costes, y permitir un mejor servicio a sus clientes.
- Cubre las necesidades de empresas u organizaciones que necesiten hasta 200 extensiones.

Cada tipo de estas pizarras telefónicas generan uno o varios formatos SMDR, diferenciándose entre ellos de acuerdo a la pizarra instalada.

1.2.2 SMDR (Station Message Detail Recording)

Los SMDR son registros que almacenan diferentes parámetros relacionados con las llamadas salientes y entrantes para cada extensión. La longitud de estos varía en dependencia de la información que recogen los mismos y por el modelo de PABX por la cual son generados. La información contenida en estos registros también difiere en cuanto al nivel de detalle que se quiera obtener de la llamada. En un registro de llamada puede incluirse información como:

- Fecha inicio de la llamada.
- Hora inicio de la llamada.
- Extensión que realiza la llamada.
- Número marcado.
- Duración de la llamada.

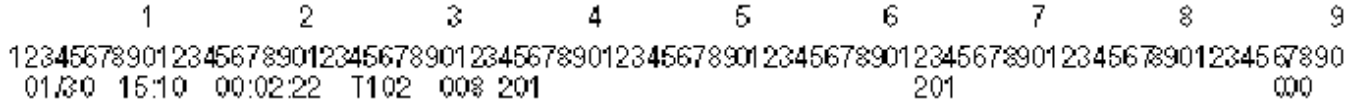
- Número de tronco utilizado en la llamada.
- Código de cuenta insertado para efectuar la llamada.
- Dirección de la llamada.

Ejemplos de registros SMDR

Llamada externa saliente

La llamada se realizó el 13 de Junio a las 11:42 AM. Se obtuvo el número de tronco 54 y se marcó el número 1-613-555-2122. La supervisión de respuesta fue dada. La conversación duró 8 minutos y 29 segundos. SMDR perteneciente a una pizarra Mitel.

Llamada externa entrante



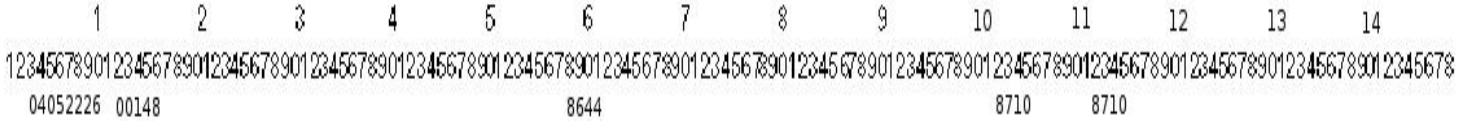
La llamada fue realizada el 30 de enero a las 3:10 PM, a través del tronco 102 hacia la extensión 201. La extensión contesto después de 8 segundos de timbrado, la cual tuvo una duración de 2 minutos y 22 segundos. SMDR perteneciente a una pizarra Mitel.

Tabla 1 Detalles del SMDR. Pizarra Mitel con una longitud de 90 caracteres.

Columnas (bits)	Descripción
2-3	Mes
5-6	día
8-9	hora
11-12	minutos
15-22	Duración de la llamada
24-27	Parte que llama

34-59	Dígitos marcados o número marcado
62-65	Parte que recibe la llamada
73-84	Código de cuenta (opcional)

Llamada interna:



La llamada fue realizada desde la extensión 8644 a la extensión 8710, el 5 de abril a las 10:26 PM, con una duración de 148 segundos. SMDR perteneciente a una pizarra Ericsson, modelo MXONE.

Tabla 2 Detalles del SMDR. Pizarra Ericsson, modelo MXONE con una longitud de 148 caracteres.

Columnas (bits)	Descripción
3-4	Mes
5-6	día
7-8	hora
9-10	minutos
12-16	Duración de la llamada
58-67	Parte que llama
85-90	Código de cuenta (opcional)
102-110	Parte que recibe la llamada
112-131	Dígitos marcados o número marcado

1.3 Metodologías de desarrollo

En la actualidad el empleo de las metodologías de desarrollo se ha convertido en un elemento fundamental para elevar la calidad del software a través de una mayor transparencia y control sobre el proceso.

Estas metodologías establecen un conjunto de actividades que definen cómo se debe hacer el software, quién debe hacer cada actividad, cuándo hacerla y qué se debe hacer.

Rational Unified Process (RUP)

RUP (Proceso de desarrollo de software): Es un conjunto de actividades necesarias para transformar los requisitos de los usuarios en un sistema de software. Puede ser utilizado para diversos sistemas de software, en diferentes áreas de aplicación, diferentes tipos de organizaciones y para diferentes tamaños de proyecto. (1)

Las características principales que definen a RUP son:

Guiado por casos de uso donde los casos de uso definen lo que el usuario desea a partir de la captura de requisitos y la modelación del negocio.

Centrado en la arquitectura característica que brinda una visión completa del sistema, se describen los procesos del negocio que son más importantes, para comprenderlo, desarrollarlo y producirlo de una forma eficaz.

Iterativo e incremental donde cada fase se desarrolla en iteraciones, de forma tal que se pueda dividir en pequeños proyectos mejorando su comprensión y desarrollo.

La vida de un sistema a través de RUP está definida por una serie de ciclos. Este consta de cuatro fases (Inicio, Elaboración, Construcción y Transición) y dentro de estas pueden ser ejecutadas una o varias iteraciones. Define 9 disciplinas de trabajo posibilitando una mayor organización y entendimiento del software.

Extreme Programing (XP)

XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo.

Se basa en la retroalimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios.

XP intenta reducir la complejidad del software por medio de un trabajo orientado directamente al objetivo, basado en las relaciones interpersonales y la velocidad de reacción.

Presenta un diseño evolutivo lo que hace que no se le dé apenas importancia al análisis como fase independiente, puesto que se trabaja exclusivamente en función de las necesidades del momento.

Selección de la metodología a utilizar.

Teniendo en cuenta las características y peculiaridades de las metodologías de desarrollo, se determina aplicar RUP, pues de este se tiene una mayor experiencia y dominio. Es la forma disciplinada de asignar tareas y responsabilidades en un equipo de desarrollo de software. Busca detectar defectos en las fases iniciales. Intenta reducir el número de cambios tanto como sea posible. Los clientes interactúan con el equipo de desarrollo mediante reuniones. Se adapta a cualquier proyecto. Se logra una documentación amplia permitiendo un mejor manejo de la información referente al proyecto a desarrollar: RUP es un proceso basado en la documentación, permitiendo así reconocer los errores a tiempo.

1.4 Lenguaje Unificado de Modelado (UML)

El Lenguaje Unificado de Modelado (UML) “es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Captura decisiones y conocimientos sobre los sistemas que se deben construir. Se usa para entender, diseñar, hojear, configurar, mantener, y controlar la información sobre tales sistemas. Está pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios. Tiene partes estáticas, dinámicas, de entorno y organizativas. Está pensado para ser utilizado en herramientas interactivas de modelado visual que tengan generadores de código así como generadores de informes. La especificación de UML no define un proceso estándar pero está pensado para ser útil en un proceso de desarrollo iterativo. Pretende dar apoyo a la mayoría de los procesos de desarrollo orientados a objetos” (2).

1.5 Lenguajes de programación

Un lenguaje de programación permite crear programas mediante un conjunto de instrucciones, operadores y reglas de sintaxis; que pone a disposición del programador, para que este pueda comunicarse con los dispositivos de hardware y software existentes.

Actualmente existe un gran número de lenguajes de programación, que por sus diferentes características y potencialidades son utilizados en el desarrollo de sistemas, eligiéndose según las peculiaridades del software a implementar.

Java: es un lenguaje orientado a objeto, diseñado para crear software altamente fiable y para soportar aplicaciones que serán ejecutadas en los más variados entornos de red.

Es un lenguaje extendido. Su característica más distintiva es que es multiplataforma, permitiendo que los programas desarrollados en él, puedan funcionar en cualquier ordenador independientemente del sistema operativo que utilice y que puedan ejecutarse sin cambios en diferentes tipos de arquitecturas y dispositivos computacionales.

Tiene comprobaciones de seguridad estrictas y se puede controlar a que recursos acceden los componentes.

Al tener gestión de memoria automática es más difícil cometer errores de pérdida de memoria por olvido de liberación de los bloques no usados. Es además un lenguaje elegante y fácil de aprender.

“Gracias al API de java se puede ampliar el lenguaje para que sea capaz de, por ejemplo, comunicarse con equipos mediante red, acceder a bases de datos, crear páginas HTML dinámicas y crear aplicaciones visuales al estilo Windows” (3).

Python: Es un lenguaje de scripting independiente de plataforma, preparado para realizar cualquier tipo de programa, desde aplicaciones Windows a servidores de red o incluso, páginas web. Es un lenguaje interpretado, lo que significa que no se necesita compilar el código fuente para poder ejecutarlo, lo que ofrece ventajas como la rapidez de desarrollo e inconvenientes como una menor velocidad. Permite la herencia múltiple así como incorporar rutinas compiladas en C.

Es multiplataforma pues existen versiones disponibles en muchos sistemas informáticos distintos, aunque necesita de un intérprete programado para él. Ofrece en muchos casos una manera sencilla de crear programas con componentes reutilizables; además es un lenguaje de programación multiparadigma, lo cual significa que más que forzar a los programadores a adoptar un estilo particular de programación, permite varios estilos: programación orientada a objetos, programación estructurada y programación funcional.

C++: Es un lenguaje orientado a objeto al que se le añadieron características y cualidades de las que carecía el lenguaje C. Depende mucho del hardware, es uno de los lenguajes más potentes porque permite programar a alto y a bajo nivel pero es complicado porque los programadores deben hacerlo casi todo por ellos mismos. Una particularidad del C++ es la posibilidad de redefinir los operadores (sobrecarga de operadores), y de poder crear nuevos tipos que se comporten como tipos fundamentales. Por otra parte, este lenguaje es particularmente difícil debido a que los programadores necesitan una gran disciplina para no cometer errores y además son estos los encargados de planificar la gestión de memoria y los errores, por este concepto suelen ser graves.

Selección del Lenguaje de Programación

Después de haber realizado un estudio previo se escogió Java como el lenguaje de programación a utilizar pues es un lenguaje altamente fiable para el desarrollo de aplicaciones de escritorio, presenta un tiempo de ejecución relativamente bajo, es multiplataforma y contiene librerías que facilitan al programador desarrollar una aplicación segura. Además, es un lenguaje que ha sido ampliamente probado obteniéndose buenos resultados y cuenta con una amplia documentación.

1.6 Entorno de Desarrollo Integrado (IDE)

Un IDE 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. Proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación.

Netbeans: Es un proyecto de código abierto que permite desarrollar aplicaciones de escritorio. Es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está

escrito en Java, pero puede servir para cualquier otro lenguaje de programación. Es un producto libre y gratuito sin restricciones de uso. Contiene todos los módulos necesarios para el desarrollo de aplicaciones Java. Permite la ingeniería inversa, así como la integración con diferentes frameworks. Soporta el desarrollo de todos los tipos de aplicación Java (J2SE, web, EJB y aplicaciones móviles). En los últimos años netbeans se ha convertido en un **IDE** apto para la mayoría de los lenguajes de programación de código abierto modernos.

Eclipse: Eclipse es un entorno de desarrollo integrado de código abierto, multiplataforma. Emplea módulos para proporcionar toda su funcionalidad. Estos le permiten trabajar con lenguajes para procesado de texto, aplicaciones en red y Sistema de gestión de base de datos. La arquitectura plugin permite escribir cualquier extensión deseada en el ambiente, como sería gestión de la configuración. Se provee soporte para Java. En cuanto a las aplicaciones clientes, eclipse provee al programador con frameworks para el desarrollo de aplicaciones gráficas, definición y manipulación de modelos de software y aplicaciones web. Eclipse dispone de un Editor de texto con resaltado de sintaxis. La compilación es en tiempo real. Asimismo, a través de "plugin" libremente disponibles es posible añadir control de versiones con Subversion e integración con Hibernate.

Selección del IDE de desarrollo

Por lo anteriormente expuesto se escoge Netbeans como el IDE de desarrollo pues contiene todos los módulos necesarios para el desarrollo de aplicaciones Java, permitiéndole al usuario empezar a trabajar inmediatamente. Es usada como una estructura de integración para desarrollar aplicaciones de escritorio grandes. Funciona en diversos sistemas operativos como Linux, Windows, Mac OS. Es 100 % Java. Es un IDE ligero e intuitivo para la implementación de interfaces.

1.7 Herramienta Case de Modelado con UML

Las herramientas CASE (***C****omputer* ***A****ided* ***S****oftware* ***E****ngineering*) de modelado con UML permiten aplicar la metodología de análisis y diseño orientados a objetos y abstraerse del código fuente, en un nivel donde la arquitectura y el diseño se tornan más obvios y más fáciles de entender y modificar.

¿Por que Visual Paradigm?

Se seleccionó **Visual Paradigm** debido a que es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Es una herramienta multiplataforma, muy fácil de usar y con un ambiente gráfico agradable para el usuario. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación, así como generar código java. Permite modelar base de datos. Aporta una excelente interoperabilidad con otras herramientas CASE y muchos de los entornos de desarrollo (IDE).

1.8 Sistema Gestor de Base de Datos (SGBD)

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

Definir una base de datos especificando tipos, estructuras y restricciones de datos, así como realizar consultas, actualizar la base de datos y generar informes, además de guardar los datos en algún medio controlado por el mismo SGBD.

¿Por qué PostgreSQL?

Se seleccionó **PostgreSQL** debido a que es un sistema de gestión de bases de datos objeto-relacional (ORDBMS), que está ampliamente considerado como el sistema de bases de datos de código abierto más avanzado del mundo. Es capaz de manejar complejas rutinas y reglas. Ejemplos de su avanzada funcionalidad son consultas SQL declarativas, control de concurrencia multi-versión, soporte multi-usuario, transacciones, optimización de consultas, herencia, y arreglos. Posee una gran escalabilidad, haciéndolo idóneo para ser usado en aplicaciones que realicen varias peticiones al día. Permite la gestión de usuarios, como también los permisos asignados a cada unos de ellos. Es altamente extensible pues soporta operadores, funciones, métodos de acceso y tipos de datos definidos por el usuario. Incluye características avanzadas tales como los joins. Soporta integridad referencial, la cual es utilizada para

garantizar la validez de los datos de la base de datos. Posee alta concurrencia permitiendo que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos.

1.9 Frameworks y Componentes

Un framework es un conjunto de funciones o código genérico que realiza tareas comunes y frecuentes en todo tipo de aplicaciones. Esto brinda una base sólida sobre la cual desarrollar aplicaciones concretas y permite obviar los componentes más triviales y genéricos del desarrollo. Son construidos en base a lenguajes orientados a objetos, permitiendo una mejor modularización de los componentes y óptima reutilización de código. (4)

Hibernate: parte de una filosofía de mapear objetos Java, también conocidos como “POJOs” (*Plain Old Java Objects*). “Es una capa de persistencia objeto/relacional y un generador de sentencias sql. Permite diseñar objetos persistentes que podrán incluir polimorfismo, relaciones, colecciones, y un gran número de tipos de datos” (5). Es de código abierto y se integra en cualquier tipo de aplicación justo por encima del contenedor de datos.

Con Hibernate no es necesario escribir código específico en los objetos ni hacer que hereden de clases determinadas, sino que trabaja con ficheros XML y objetos que proporciona la librería. Una de las principales características de Hibernate es su flexibilidad, envolviéndolo todo bajo un marco de trabajo común. Posee gran escalabilidad y un mapeado flexible. En su funcionamiento genera las sentencias SQL y libera al desarrollador del manejo manual de los datos que resultan de la ejecución de dichas sentencias, manteniendo la portabilidad entre todas las bases de datos con un ligero incremento en el tiempo de ejecución.

Ofrece un lenguaje de consulta de datos llamado HQL (*Hibernate Query Language*), al mismo tiempo que una API para construir las consultas programáticamente (conocida como "criteria"). Es capaz de generar todos los ficheros de configuración y los objetos asociados.

Spring: Framework de código abierto, que proporciona la posibilidad de integrarse con otras herramientas o frameworks, para esto brinda diferentes módulos, según la herramienta o el frameworks a integrar. Ofrece a los desarrolladores soluciones bien documentadas y fáciles de usar. Promueve la reutilización

del código pues está diseñado con interfaces que pueden ser utilizadas por los desarrolladores. En caso de que ocurran excepciones estas son capturadas y traducidas a excepciones del mismo spring. Se enfoca en el manejo de objetos de negocio dentro de una arquitectura en capas.

Librería rxtx-2.1-7r2

Es una librería nativa para java que permite controlar el puerto serie de la PC, funcionando en diferentes Sistemas Operativos.

1.10 Conclusiones

Teniendo en cuenta el estudio realizado previamente sobre las principales tendencias que existen actualmente acerca de las tecnologías y herramientas más usadas en el campo de la informática, se toma como decisión desarrollar una aplicación de escritorio, sobre el lenguaje Java, con la integración de los frameworks Spring e Hibernate, ganando en velocidad de desarrollo y aplicando el patrón n capas, lo que facilita el desarrollo del producto disminuyendo el acoplamiento. Para lograr tal resultado se propone el uso de las herramientas Netbeans y Visual Paradigm por las facilidades que estas brindan. Todo este proceso será orientado por la metodología RUP, la cual constituye una guía de cómo se debe desarrollar el software.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

2.1 Introducción

En este capítulo se describen las principales características y funcionalidades del sistema. Se identifican los actores y casos de uso que permiten conformar el Modelo de Casos de Uso del Sistema. Se detallan los requisitos funcionales y no funcionales, el modelo de dominio, las descripciones textuales de los casos de uso y el diagrama de paquetes.

2.1.1 Problema y situación problémica

Las empresas que utilizan las pizarras telefónicas privadas se enfrentan a diversos problemas al interactuar con ellas. Debido a que cada PABX tiene sus particularidades, no es factible para las entidades que las adquieren comprar un software diferente cada vez que estas necesiten cambiar la pizarra debido a que resultaría costoso. Es por esto que se hace necesario el desarrollo de un sistema que permita ser configurado para ser adaptado a las condiciones específicas de la PABX que tenga la entidad. Para esto es imprescindible implementar un módulo que permita, además de las configuraciones necesarias para la comunicación de la pizarra con la PC, obtener los datos y almacenarlos en una base de datos. El software debe ser además libre y sin restricciones de uso.

2.1.2 Objeto de automatización

El objeto de automatización es el proceso de comunicación entre la PABX que se vaya a utilizar en una empresa determinada y la PC donde se estará ejecutando el sistema, a través del puerto RS-232C teniendo en cuenta los parámetros de configuración y las características específicas de la misma.

El sistema permitirá a los usuarios realizar las configuraciones necesarias de acuerdo con las características específicas de su PABX, gestionar los registros SMDR, tener información en tiempo real de los datos que están recibiendo y recibir avisos en caso de que ocurra algún tipo de error.

2.1.3 Información que se maneja

El módulo comunicador maneja toda la información referente a las características principales de las PABX y específicamente de los SMDR que estas generan posibilitando tener todos los datos de cada llamada realizada o recibida de acuerdo con la configuración que el usuario haya especificado al levantarse el sistema.

2.2 Propuesta de sistema

Con el fin de darle solución a los problemas planteados y dadas las necesidades del cliente, se propone diseñar e implementar un sistema que permita la interacción con la PABX, permitiendo opciones de configuración de registros y parámetros del puerto serie. Después de ser configurada una conexión para la comunicación, el sistema debe recibir constantemente los SMDR generados por la pizarra. Estos serán traducidos y almacenados después de comprobar si son válidos. En caso de ocurrir algún tipo de error, será debidamente informado. El sistema mostrará un ícono de estado que será verde mientras no existan errores, en caso contrario se mostrará en color rojo; además este permitirá el acceso rápido a diversas informaciones que brinda el software y que deseen ser consultadas por el usuario.

2.3 Modelo de Dominio

En el sistema que se desea realizar no se identifican de forma clara los actores, trabajadores y los procesos del negocio, por lo que se decide realizar un modelo de dominio, el cual es un subconjunto del modelo de objeto del negocio donde se representan los conceptos y eventos fundamentales que se expresan como clases con la cardinalidad que existe entre ellas.

En el siguiente modelo de dominio se especifican las relaciones que existen entre los principales conceptos que interactúan en el sistema:

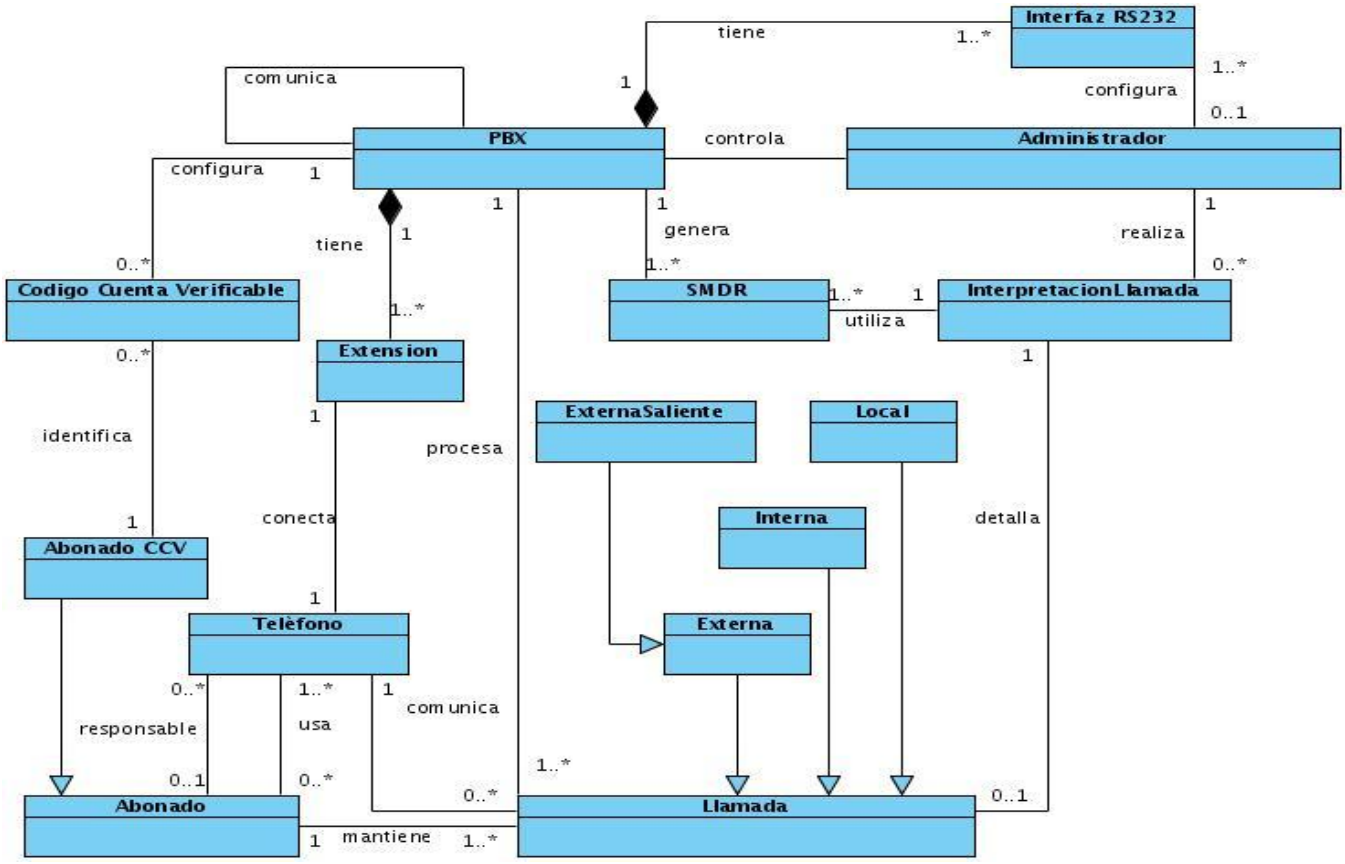


Figura 1 Modelo de Dominio.

Para una mejor comprensión del modelo de dominio, a continuación se explican los conceptos involucrados en el mismo:

La persona autorizada a interactuar con el sistema es el **administrador** quien además es el encargado de instalarlo y configurarlo. Este configura la **interfaz del puerto RS-232C**. Esta interfaz es la que soporta el **SMDR** enviado por la pizarra, tiene asociado una **PBX** que además se comunica con otras pizarras (PBX) y es controlada por el administrador.

La PBX tiene **extensiones** que no son más que los canales de comunicación entre la PBX y los teléfonos que se conectan a estas extensiones. Un **teléfono** es usado por un abonado para mantener una llamada que se comunica mediante este. El **abonado** puede ser o no responsable de un teléfono. Existe un

abonadoCCV que es identificado por un **código de cuenta verificable (CCV)** que es configurado por la PBX. Un CCV es una serie de números única para cada usuario.

Las **llamadas** realizadas por el abonado son procesadas por una PBX y pueden ser **internas** (llamadas realizadas dentro de la empresa), **externas** (llamadas que se realizan desde fuera de la empresa hacia el interior de la misma), **externas salientes** (llamadas que se realizan desde el interior de la empresa hacia el exterior de la misma) y **locales** (llamadas con características similares a las externas pero no se facturan). Las llamadas son detalladas mediante una **interpretación de llamada**. Para realizar la interpretación de las llamadas se utiliza los SMDR que genera la PBX para obtener todos los detalles de la misma. Esta interpretación es realizada por el administrador.

2.4 Especificación de los requisitos del Software

Los requerimientos son la condición o capacidad que tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente.

2.4.1 Requerimientos funcionales.

Son capacidades o condiciones que el sistema debe cumplir.

RF1 Autenticar usuario

Verificar que el usuario este correctamente validado y tenga los permisos para acceder a los recursos que solicita.

RF1.1 Cargar configuración

Cargar la configuración correspondiente al usuario que se ha autenticado, según las funcionalidades a las que este tiene permiso.

RF2 Gestionar parámetros de configuración de la conexión con la PBX

RF2.1 Configurar parámetros de la conexión con la PBX

Seleccionar la marca de la PBX y su modelo y configurar los parámetros: formato SMDR y parámetros de puerto serie.

RF2.2 Eliminar parámetros de la conexión con la PBX

Seleccionar la conexión que se encuentra configurada y eliminarla.

RF2.3 Mostrar detalles de la conexión con la PBX

Mostrar listado con: modelo de PBX, registro SMDR, bits por segundo, bits de datos, paridad y bit de parada.

RF3 Gestionar la configuración base

RF3.1 Registrar parámetros de configuración base

Registrar la información que se desea recibir de la PBX que puede ser: llamada entrante y llamada saliente.

RF4 Configurar modo de inicio

Establecer si al iniciarse el sistema se iniciará la aplicación o no.

RF5 Configurar persistencia de los registros SMDR localmente

RF5.1 Establecer ruta local de persistencia de registros SMDR

Especificar si se salvarán o no los parámetros asociados al registro SMDR, además de establecer la ruta local donde serán almacenados.

RF5.2 Configurar fichero local

Crear nuevo fichero local para almacenar los parámetros asociados al registro SMDR, y nombrarlo empleando el formato AAAA_MM.txt.

RF5.3 Almacenar registros SMDR en fichero local

Guardar los parámetros asociados a los registros SMDR en un fichero.

RF6 Gestionar trama SMDR

RF6.1 Registrar datos de trama SMDR

Registrar en la BD la configuración de los campos que conforman la nueva trama SMDR, así como los campos marca de PBX y modelo de PBX.

RF6.2 Modificar datos de la configuración de trama SMDR

Buscar la trama a la cual se le quiera modificar la información y se modifica.

RF6.3 Eliminar trama SMDR

Buscar la trama que se desea eliminar y si no está siendo utilizada por el sistema, puede proceder a eliminarla.

RF6.4 Buscar trama SMDR

Filtrar por los criterios establecidos: marca de PBX, modelo de PBX, especificación de trama SMDR y mostrar listado de trama SMDR que coincidan con los criterios de búsqueda.

RF6.5 Mostrar detalles de trama SMDR

Buscar la trama de la cual se desean ver los detalles y recuperar la información de las mismas.

RF7 Registrar CCV

Registrar el parámetro CCV.

RF8 Cargar plantilla de registros SMDR

Cargar un determinado registro SMDR para posteriormente ser utilizado.

RF9 Mostrar información

RF9.1 Mostrar resumen de funcionamiento.

Mostrar una ventana con el resumen de informaciones generales del sistema.

RF9.2 Mostrar estado.

Ejecutar continuamente la aplicación y su estado se mostrará a través de un icono en el tray-bar que será verde mientras no exista error. Al surgir un error el icono cambia a rojo y muestra el mensaje del error producido.

RF9.3 Mostrar información de llamada telefónica.

Mostrar información en tiempo real de las llamadas telefónicas realizadas.

RF10 Controlar registro SMDR

Capturar los SMDR y posteriormente almacenarlos en la BD.

RF11 Chequear error

Chequear la información enviada por la PBX en busca de errores e identificar los mismos.

RF12 Gestionar parámetros de configuración de la conexión Telnet

RF12.1 Configurar parámetros de la conexión Telnet

Configurar los parámetros: Host remoto, Usuario, Contraseña, Contraseña de intercambio y Puerto.

RF12.2 Eliminar parámetros de la conexión Telnet

Seleccionar la opción eliminar conexión telnet y elimina la que se encuentre configurada.

RF12.3 Mostrar detalles de la conexión Telnet

Mostrar en una ventana la información de la conexión telnet que se encuentre configurada, especificando los parámetros: Host remoto, Usuario y Puerto.

RF13 Gestionar comando de bloqueo.

RF13.1 Registrar comando de bloqueo

Se selecciona a quién se le aplicará el comando de bloqueo y luego se especifican los parámetros requeridos para configurar las secciones que conformarán el comando de bloqueo.

RF13.2 Mostrar comando de bloqueo

Mostrar en una ventana todos los comandos de bloqueo que han sido configurados.

RF13.3 Mostrar detalles del comando de bloqueo

Mostrar en una ventana la información del comando de bloqueo que se desee detallar.

RF13.4 Eliminar comando de bloqueo

Buscar los comandos de bloqueo existentes y seleccionar el que se desea eliminar.

2.4.2 Requerimientos no funcionales

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener.

1. Usabilidad

El administrador del sistema, necesitará una preparación previa para operar la misma. Se requiere que posea un nivel medio o alto en conocimientos de computación, aunque el manejo de la aplicación es sencillo, pues la mayoría de las operaciones se realizan de forma automática. Las operaciones no automatizadas son de configuración, y visualización de reportes, las que son fácilmente comprensibles por cualquier usuario.

2. Fiabilidad

2.1 Disponibilidad

Es necesaria la ejecución de la aplicación las veinticuatro horas del día, para la actualización de la base de datos en tiempo real. Esta característica es fundamental debido a que continuamente se obtendrán datos importantes en los registros SMDR generados por la pizarra. El sistema deberá detectar y mostrar los diferentes tipos de errores críticos que puedan surgir durante su ejecución.

3. Eficiencia

La eficiencia del producto estará determinada en gran medida por la velocidad que se logre en la transferencia de datos entre la PBX y la computadora en la que sea instalado el sistema.

Es necesario que la velocidad del puerto esté acorde con el volumen de información y la rapidez con que es generado por la pizarra, para lograr un mejor rendimiento del sistema. Esta velocidad varía en dependencia de las características de la pizarra telefónica.

4. Soporte

4.1 Entrega manual de ayuda

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

4.2 Impartir capacitación

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

5. Restricciones de diseño

5.1 Lenguaje de programación

Lenguaje de programación Java.

5.2 Biblioteca de clases

Comunicación con la planta telefónica a través del puerto RS-232C se emplea la librería rxtx-2.1-7r2.

6. Requisitos para la documentación de usuarios en línea y ayuda del sistema.

El sistema debe entregarse con un manual o ayuda para el usuario.

7. Componentes comprados

El software no usará ningún componente comprado.

8. Interfaz

8.1 Interfaz de usuario

La aplicación propuesta poseerá una interfaz sencilla dirigida directamente al usuario del sistema. El diseño se realizará siguiendo las formalidades de las ventanas de Windows para mayor comodidad. Este software no intercambiará ningún tipo de información con un sistema mayor. Es un producto independiente, autónomo, que realizará sus propias funciones.

8.2 Interfaces hardware

Permitir la interacción con plantas telefónicas privadas de los fabricantes Alcatel, Ericsson, PANASONIC, LG, Toshiba, Infinity y MITEL; realizando la comunicación por el puerto serie RS-232C.

8.3 Interfaces software

Soportar la compatibilidad con el sistema operativo GNU/Linux, logrando interacción con los directorios y ficheros de su sistema de archivos.

8.4 Interfaces de comunicación

Comunicación entre la PBX y una PC: Puerto serie (RS-232C) disponible en la PC; dicho puerto se encontrará todo el tiempo en modo de recepción de datos mientras se ejecute la aplicación.

Comunicación de red local: Se empleará el protocolo TCP/IP.

9. Requisitos de licencia

El software no tiene ningún requisito de licencia o restricción de uso.

10. Requisitos legales, de derecho de autor y otros

El soporte se encuentra en fase de discusión entre las partes involucradas.

2.5 Modelo de Casos de Uso del Sistema.

2.5.1 Definición de los actores del sistema a automatizar

Actor	Descripción
Administrador de PBX	Encargado de gestionar las tramas SMDR, definiendo su estructura. Responsable de establecer la conexión con la PBX, así como las configuraciones asociadas al almacenamiento de las llamadas telefónicas. (RF2, RF2.1, RF2.2, RF2.3, RF3, RF3.1, RF4, RF5, RF5.1, RF6, RF6.1, RF6.2, RF6.3, RF6.4, RF6.5, RF9, RF9.1, RF12, RF12.1, RF12.2, RF12.3, RF13, RF13.1, RF13.2, RF13.3, RF13.4)
PBX	Actor que representa a la pizarra telefónica. (RF5, RF5.2, F5.3, RF7, RF8, RF9, RF9.2, RF9.3, RF10, RF11)

2.5.2 Diagrama de paquetes

Los diagramas de paquetes se usan para organizar el sistema en elementos más pequeños y fáciles de interpretar. Estos muestran cómo un sistema está dividido en agrupaciones lógicas mostrando las dependencias entre esas agrupaciones.

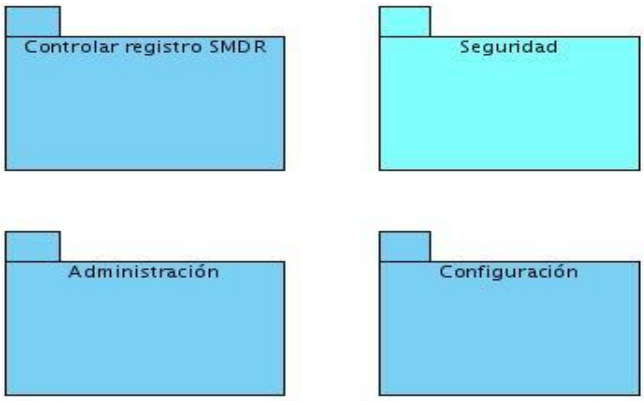


Figura 2 Diagrama de Paquetes.

2.5.3 Diagrama de casos de uso del sistema

Un diagrama de casos de uso del sistema representa gráficamente a los procesos y su interacción con los actores.

Diagrama de casos de uso del sistema, paquete “Seguridad”

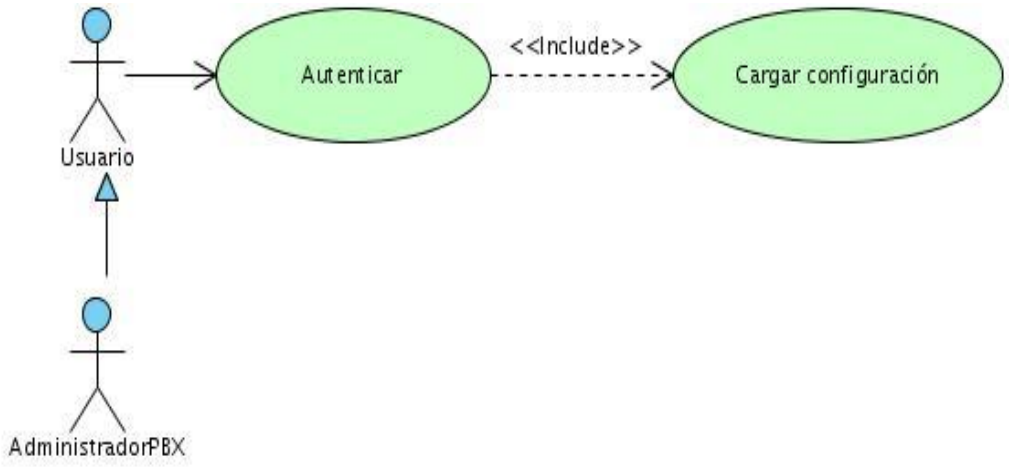


Diagrama de casos de uso del sistema, paquete “Controlar registro SMDR”

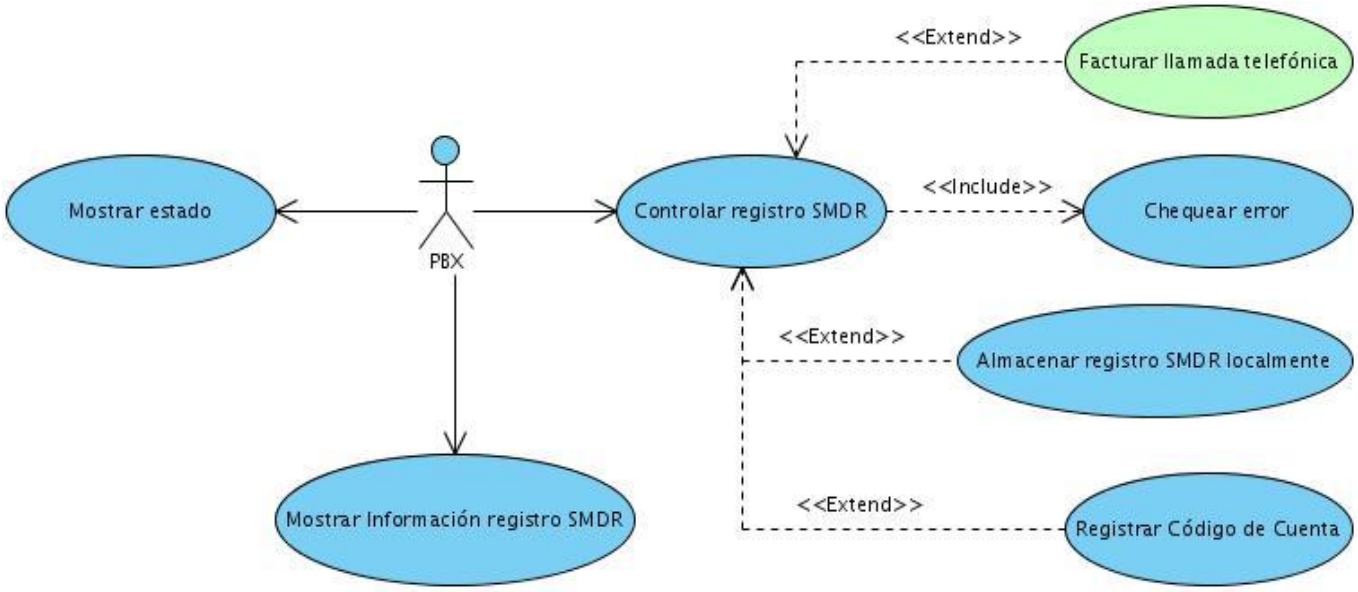


Diagrama de casos de uso del sistema, paquete “Configuración”

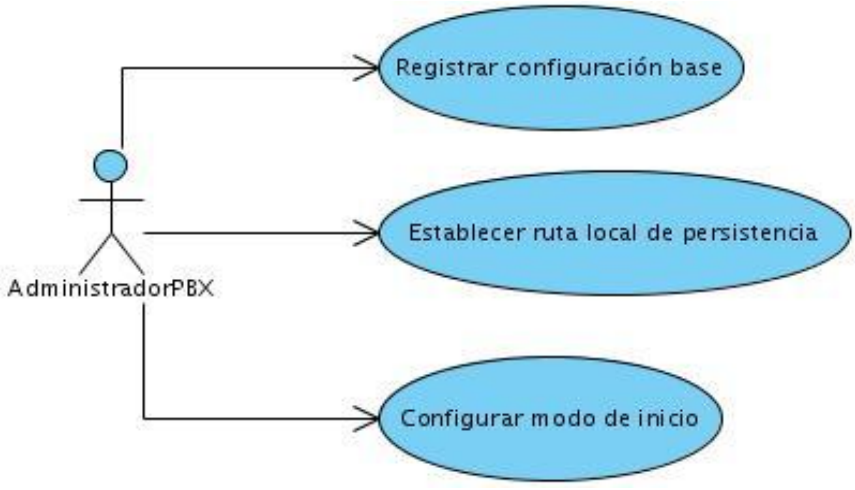
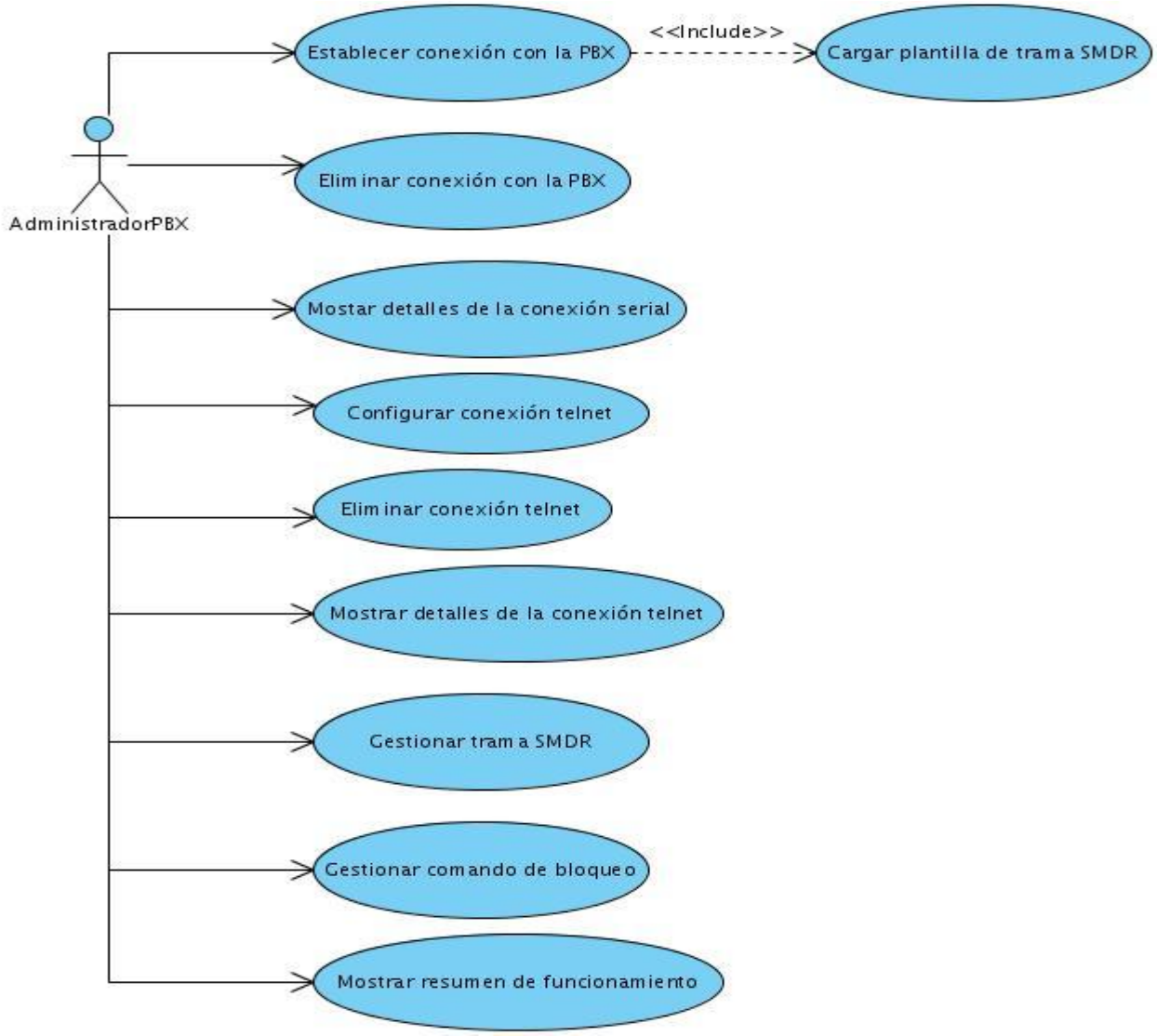


Diagrama de casos de uso del sistema, paquete “Administración”



2.5.4 Descripción de los Casos de Uso del Sistema

Para alcanzar un mejor entendimiento de la funcionalidad de los casos de usos se especifican las respectivas descripciones.

Tabla 3 Descripción del Caso de Uso del Sistema. Establecer conexión con la PBX.

Caso de Uso:	**Establecer conexión con la PBX**
Actores:	Administrador PBX
Resumen:	El caso de uso inicia cuando el Administrador PBX selecciona la opción "Conexión", se muestra la opción para establecer la conexión con la PBX. Inserta los parámetros necesarios para establecer la conexión con la PBX, se toma de la BD el formato de la trama SMDR asociado.
Precondiciones:	El sistema conoce las tramas SMDR asociadas a cada una de las PBX.
Referencias	RF2, RF2.1
Prioridad	Crítico

Tabla 4 Descripción del Caso de Uso del Sistema. Registrar configuración base.

Caso de Uso:	**Registrar configuración base**.
Actores:	Administrador PBX
Resumen:	El caso de uso inicia cuando el Administrador PBX selecciona la opción "Configuración", se muestra la opción para modificar los parámetros de la configuración base. Se muestran los parámetros actuales de dicha configuración permitiendo establecer la información que se desea recibir de la PBX.
Precondiciones:	El administrador del sistema debe estar autenticado.
Referencias	RF3, RF3.1
Prioridad	Crítico

Tabla 5 Descripción del Caso de Uso del Sistema. Controlar registro SMDR.

Caso de Uso:	**Controlar registro SMDR**
Actores:	PBX
Resumen:	El caso de uso inicia cuando la PBX envía a la aplicación los registros SMDR asociado a las llamadas telefónicas a través de la conexión configurada. La aplicación captura los registros SMDR, los procesa, interpreta y posteriormente almacena en la Base de Datos.
Precondiciones:	Establecer la conexión entre la PBX y la aplicación.
Referencias	RF10, RF5.2, RF5.3, RF7, RF11
Prioridad	Crítico

Tabla 6 Descripción del Caso de Uso del Sistema. Gestionar trama SMDR.

Caso de Uso:	**Gestionar trama SMDR**
Actores:	Administrador PBX
Resumen:	El caso de uso inicia cuando el Administrador PBX selecciona la opción “SMDR” para registrar, modificar, eliminar, buscar o mostrar los detalles de la trama SMDR asociada a una PBX.
Precondiciones:	No aplica
Referencias	RF6, RF6.1, RF6.2, RF6.3, RF6.4, RF6.5
Prioridad	Crítico

Tabla 7 Descripción del Caso de Uso del Sistema. Mostrar estado.

Caso de Uso:	**Mostrar estado**
Actores:	PBX
Resumen:	El caso de uso se inicia cuando la aplicación recibe los registros SMDR asociados a las llamadas telefónicas a través de la conexión configurada. La aplicación se ejecutará continuamente y su estado se mostrará a través de un ícono en el tray-bar, que será verde mientras no exista error y rojo al surgir un error. Dependiendo del estado se muestra el total de llamadas procesadas desde el inicio de la aplicación o el mensaje del último error producido.
Precondiciones:	La PBX se encuentra conectada a la aplicación a través del puerto serie.
Referencias	RF9, RF9.2
Prioridad	Crítico

Tabla 8 Descripción del Caso de Uso del Sistema. Establecer ruta local de persistencia.

Caso de Uso:	**Establecer ruta local de persistencia**
Actores:	Administrador PBX
Resumen:	El caso de uso inicia cuando el Administrador de la PBX selecciona la opción “Configuración”, se muestra la opción para establecer la ruta local de persistencia.
Precondiciones:	No aplica
Referencias	RF5, RF5.1
Prioridad	Crítico

2.6 Conclusiones

En este capítulo se representaron las funcionalidades que el sistema debe cumplir a través de un Modelo de Dominio. Se analizaron y describieron los requisitos funcionales y no funcionales. Además, se identificaron y describieron los actores y casos de uso, estableciéndose las relaciones entre ellos en un Diagrama de Casos de Uso del Sistema.

CAPÍTULO 3: DISEÑO E IMPLEMENTACIÓN DEL SISTEMA

3.1 Introducción

En este capítulo se tratan los temas relacionados con el diseño e implementación de la propuesta del sistema, para ello se comenzará con una referencia al modelo de diseño, dando paso a los diferentes diagramas a tratar en el capítulo.

3.2 Modelo de Diseño

El modelo de diseño “describe la realización física de los casos de uso centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema a considerar”. (6)

3.2.1 Diagrama de Paquete del diseño

Un paquete de diseño es una colección de clases, relaciones, realizaciones de casos de usos, diagramas y otros paquetes. Es usado para estructurar el modelo de diseño mediante su división en partes más pequeñas y agrupar elementos relacionados de dicho modelo con propósitos organizacionales.

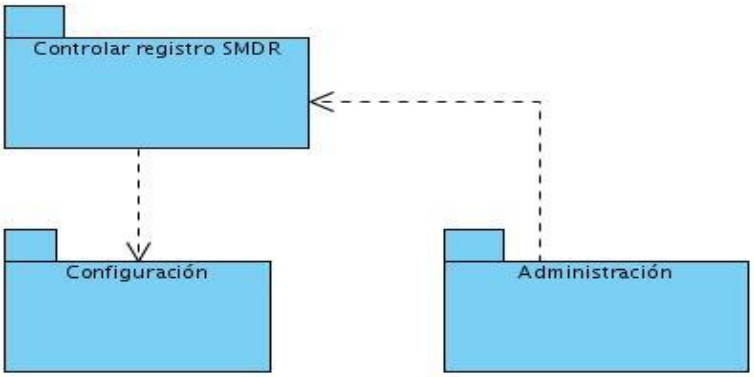


Figura 3 Diagrama de Paquete del diseño.

3.2.2 Diagrama de clases del diseño por paquetes

Un diagrama de clases es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos.

Para alcanzar un mejor entendimiento de las clases del diseño se propone la siguiente leyenda:

Clases de la capa Presentación. Clases Acceso a Datos.

Clases del Negocio. Clases del Dominio.

3.2.2.1 Diagrama de clases del diseño. Paquete Configuración

El paquete Configuración agrupa las clases encargadas de la configuración del sistema.

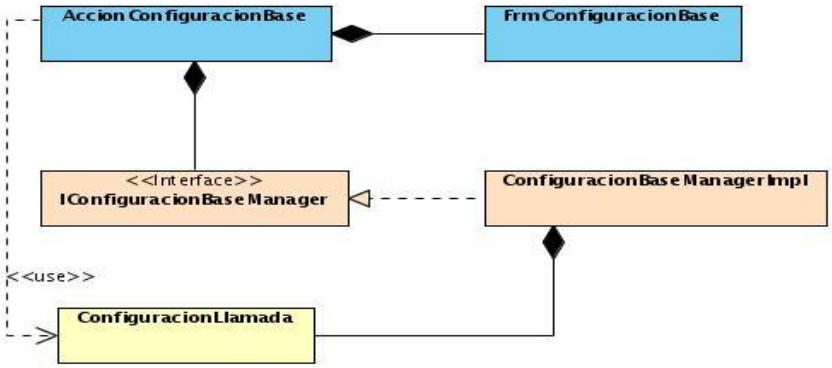


Figura 4 Diagrama de clases del diseño. Paquete Configuración. Caso de uso Registrar Configuración Base.

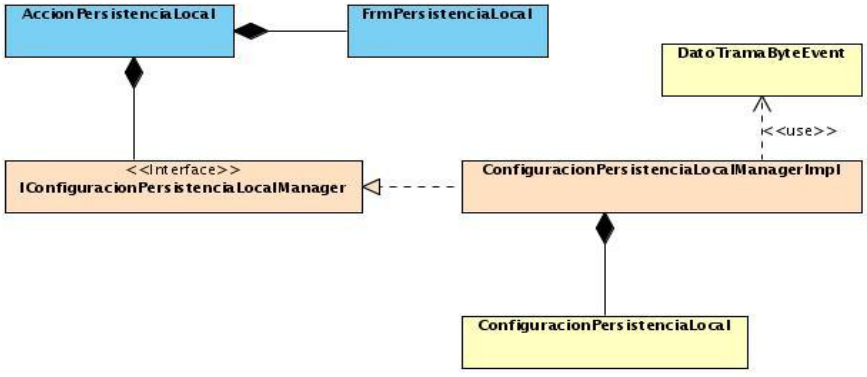


Figura 5 Diagrama de clases del diseño. Paquete Configuración. Caso de uso Establecer ruta local de persistencia.

3.2.2.2 Diagrama de clases del diseño. Paquete Administración

El paquete Administración agrupa las clases encargadas de establecer y mantener la conexión con la PBX, así como la gestión (crear, buscar, modificar y ver detalles) de las tramas SMDR.

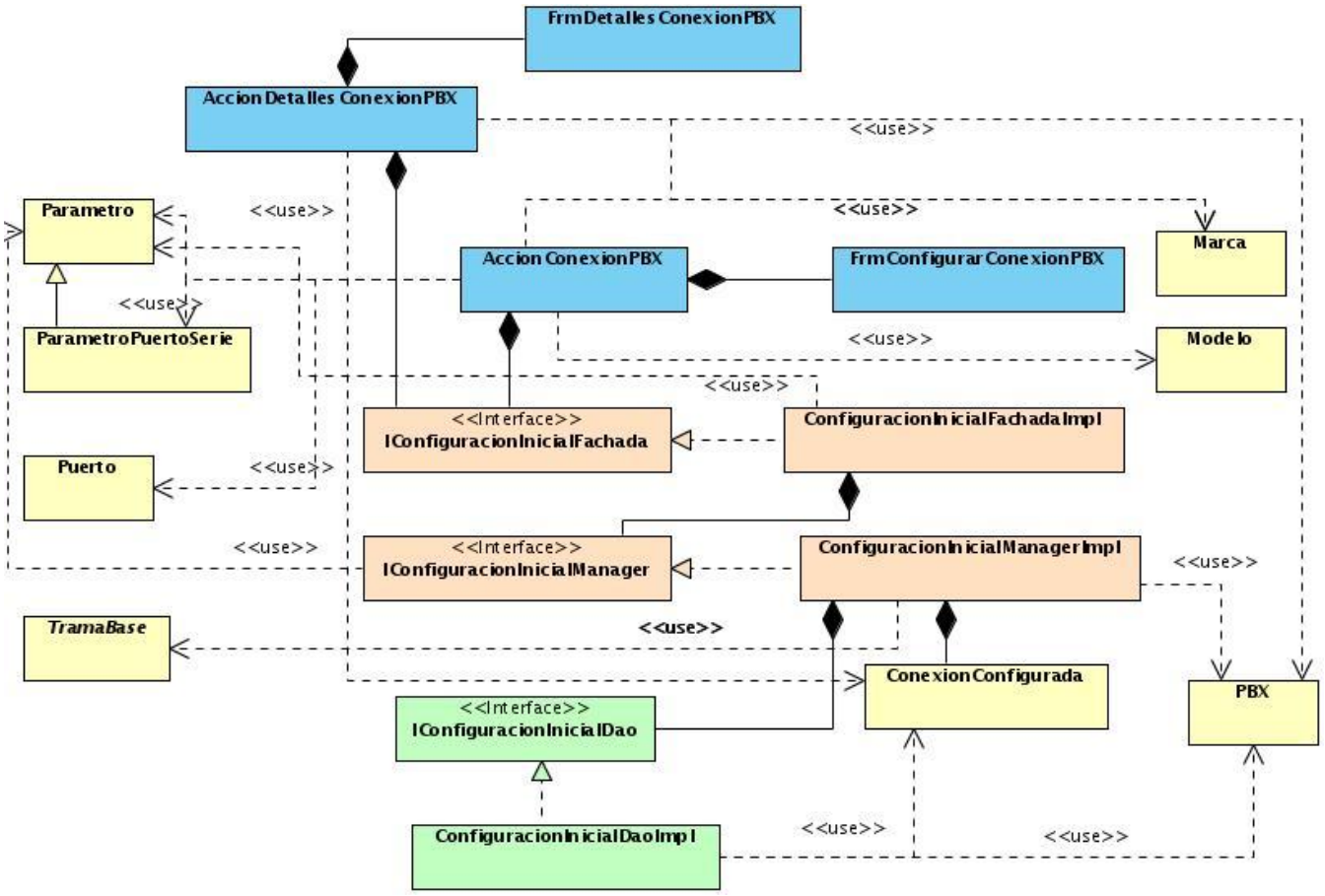


Figura 6 Diagrama de clases del diseño. Paquete Administración. Caso de uso Establecer conexión con la PBX y Mostrar detalles de la conexión.

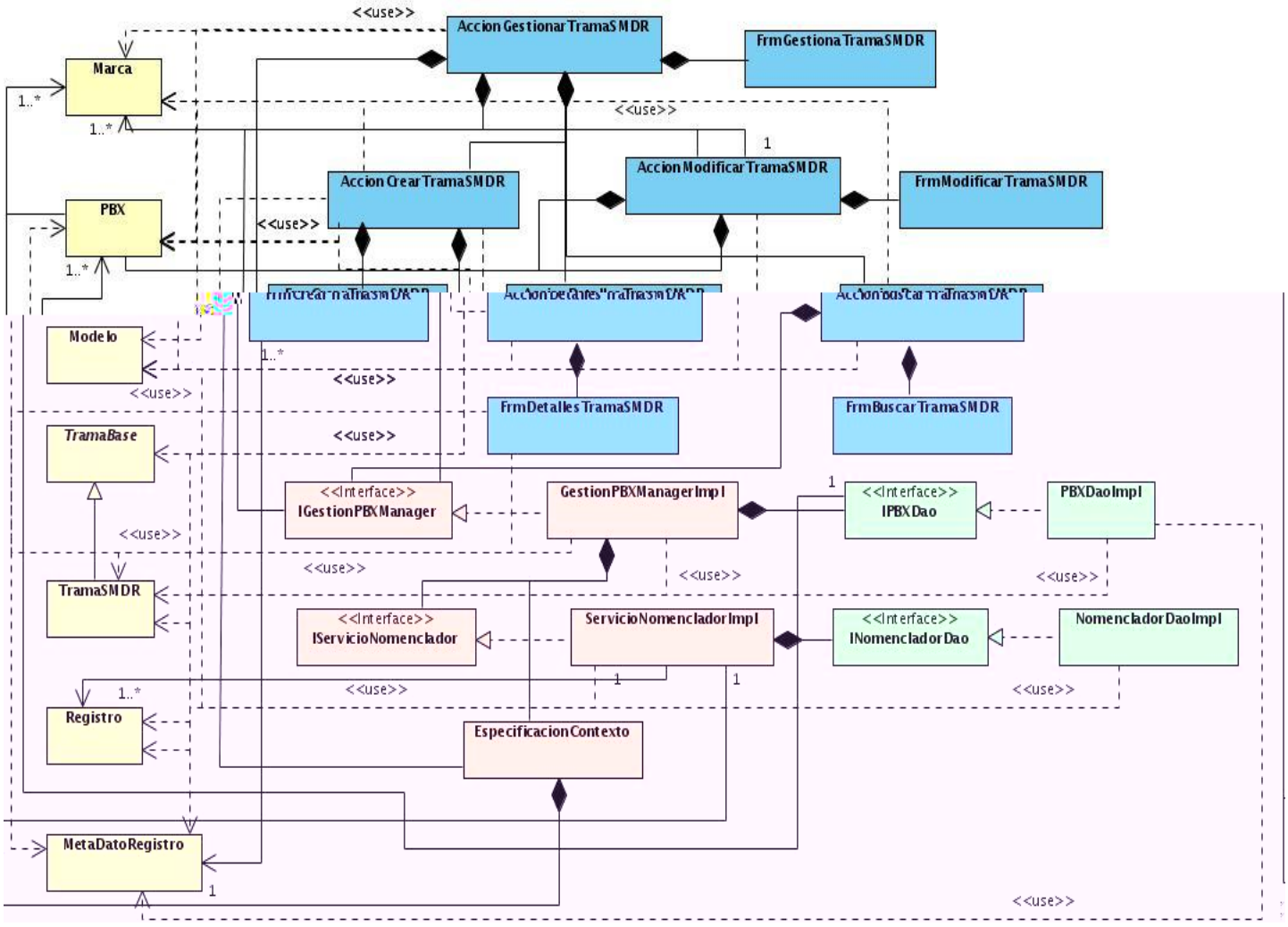


Figura 7 Diagrama de clases del diseño. Paquete Administración. Caso de uso Gestionar trama SMDR.

3.2.2.3 Diagrama de clase del diseño. Paquete Controlar registro SMDR

El paquete Controlar registro SMDR reúne todas las clases que tienen la responsabilidad de controlar la información enviada por la pizarra telefónica, además de permitir que se conozca el estado del sistema en tiempo real.

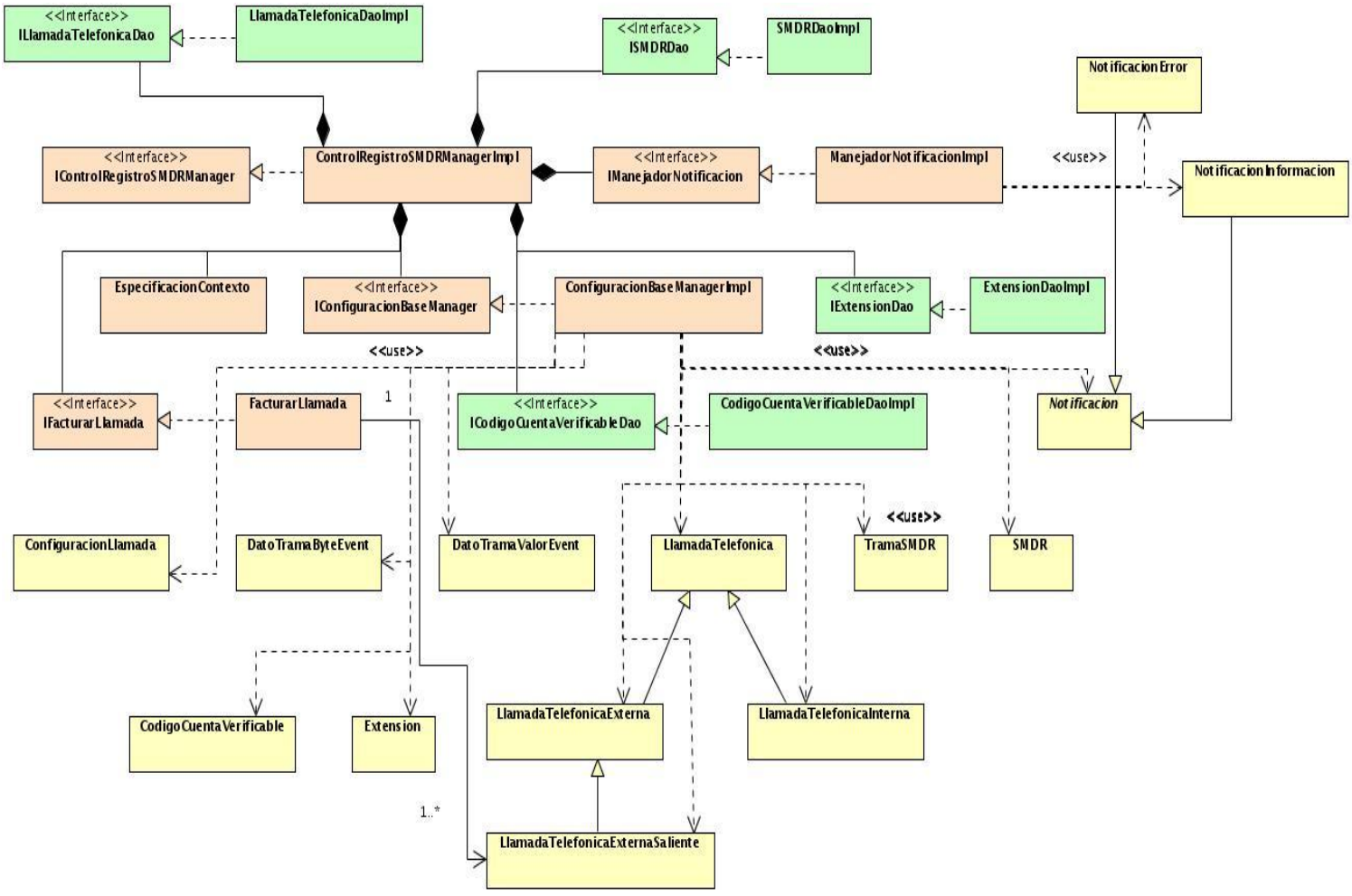


Figura 8 Diagrama de clases del diseño. Paquete Controlar registro SMDR. Caso de uso Controlar registro SMDR.

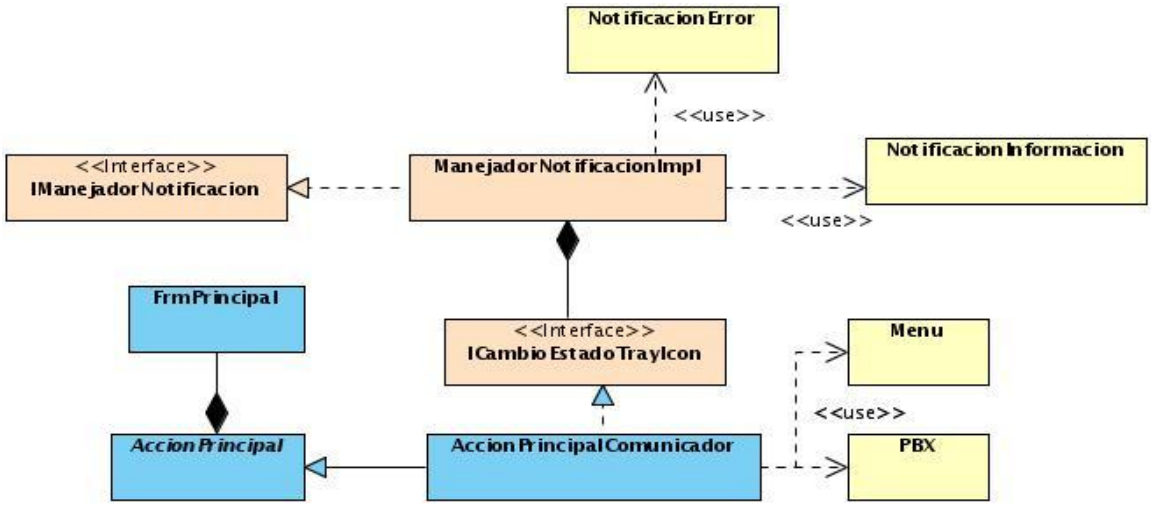


Figura 9 Diagrama de clases del diseño. Paquete Controlar registro SMDR. Caso de uso Mostrar estado.

3.2.3 Arquitectura

Una arquitectura no es más que un esqueleto o base de una aplicación. Es un diseño que muestra los bloques de construcción físicos y lógicos de una aplicación distribuida (o algún otro sistema de software) y las relaciones entre ellos. Indica la estructura, funcionamiento e interacción entre las partes del software.

La Arquitectura de Software establece los fundamentos para que analistas, diseñadores, programadores, y otros miembros del equipo trabajen en una línea común que permita alcanzar los objetivos del sistema de información, cubriendo todas las necesidades.

3.2.4 Patrones

Los patrones ayudan a construir la experiencia colectiva de Ingeniería de Software. Son una abstracción de "problema – solución". Se ocupan de problemas recurrentes. Identifican y especifican abstracciones de niveles más altos que componentes o clases individuales y proporcionan vocabulario y entendimiento común.

3.2.4.1 Patrones de arquitectura

“Un patrón de arquitectura de software describe un problema particular y recurrente del diseño, que surge en un contexto específico, y presenta un esquema genérico y probado de su solución”. (7)

Patrón de arquitectura propuesto

El modelo n capas ha emergido como la arquitectura predominante para la construcción de aplicaciones multiplataforma. Proporciona aplicaciones robustas y con mayor flexibilidad, brindando soluciones fiables para resolver problemas inmersos en cambios constantes. Ofrece modularidad, escalabilidad y simplifica el desarrollo del sistema. Los sistemas basados en esta arquitectura tienen el potencial de reducir el coste total de mantenimiento.

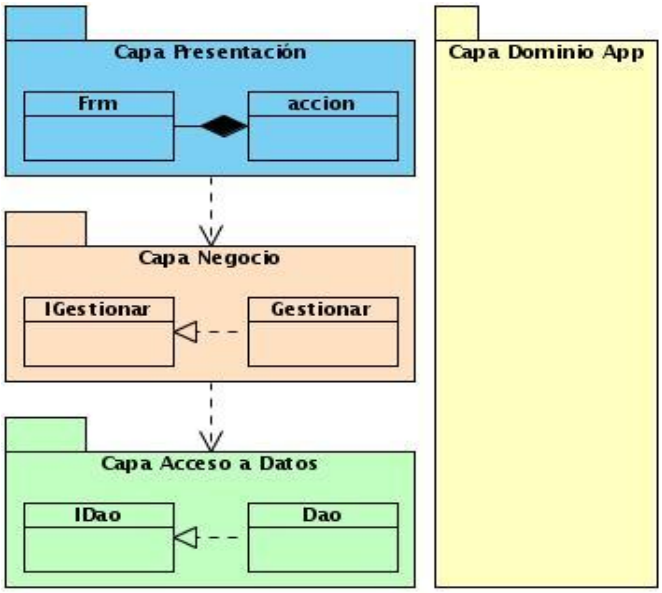


Figura 10 Arquitectura del sistema.

Para el desarrollo de la aplicación se implementará dicha arquitectura distribuida en:

Capa de presentación: presenta el sistema al usuario, le comunica la información y captura la información del usuario en un mínimo de procesos. Esta capa contiene los formularios y las clases que implementan a estos. Se comunica únicamente con la capa de negocio.

Capa de negocio: es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio e incluso de lógica del negocio porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos almacenar o recuperar datos de él.

Capa de datos: es donde residen las clases que representan los datos y es la encargada de acceder a los mismos. Está formada por un gestor de bases de datos que realiza todo el almacenamiento de los datos, recibe solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

Capa dominio de aplicación: está formado por un conjunto de clases que representan de manera abstracta objetos del mundo real. Las mismas contienen información que el sistema debe manipular a través de las capas, dándole en cada una de ellas el uso necesario. Dado que las demás capas se van a encargar de hacer cumplir las restricciones del negocio, el dominio constituye el soporte para la transferencia de datos desde el acceso a datos hasta la presentación y viceversa.

3.2.4.2 Patrones de Diseño

"Los patrones de diseño son soluciones simples a problemas específicos y comunes del diseño orientado a objetos, que una vez entendido su funcionamiento, los diseños serán más flexibles, modulares y reutilizables". (8)

Patrones GoF utilizados en el desarrollo del sistema

Patrón: Singleton (Instancia única): Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia.

Clasificación: Creacional.

Problema a resolver: Mantener los datos del usuario autenticado en la aplicación.

Solución: Garantiza que en todo momento solo exista un objeto de una clase particular.

Implementación:

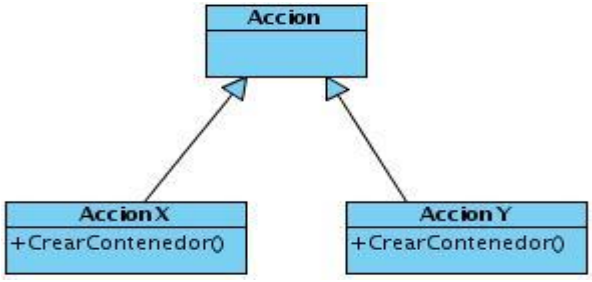
```
private static UsuarioUnico objeto;

public static UsuarioUnico getInstancia () {

if (objeto == null)

objeto= new UsuarioUnico ();

return objeto;

}
```

Patrón: Factory Method (Método de fabricación): Centraliza en una clase constructora la creación de objetos de un subtipo de un tipo determinado, ocultando al usuario la casuística para elegir el subtipo que crear.

Clasificación: Creacional.

Problema a resolver: Crear formularios con características diferentes entre ellos.

Solución: Garantiza la creación de objetos concretos.



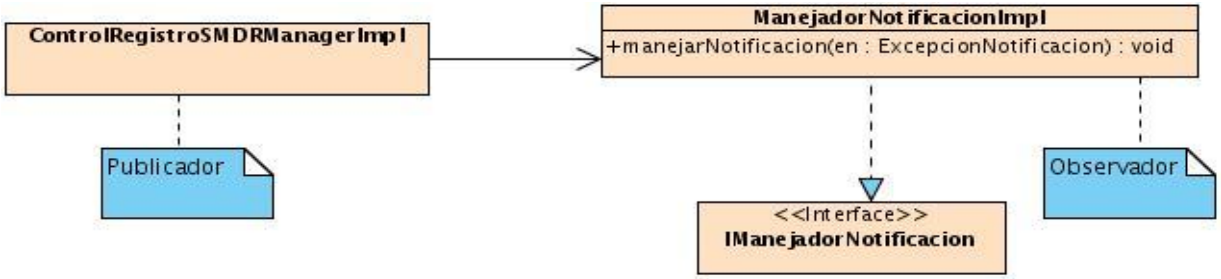
Patrón: Observer (Observador): Define una dependencia de uno-a-muchos entre objetos, de forma que cuando un objeto cambie de estado se notifique y actualicen automáticamente todos los objetos que dependen de él.

Clasificación: Comportamiento.

Problema a resolver: Mantener las dependencias entre objetos, sin necesidad de conocer al otro objeto.

Solución: Garantiza mantener las dependencias entre objetos, sin que se conozcan el uno al otro.

Implementación:



Patrón: Inversión de Control

Se utiliza para delegar en otro componente, un framework por ejemplo, la responsabilidad de crear instancias necesarias.

Implementación:

```
<bean id="frmModificarTramaSMDR" factory-bean="accionModificarTramaSMDR" factory-method="crearContenedor"></bean>
```

Patrón: Inyección de Dependencias. Suministra objetos a una clase, en lugar de ser la propia clase quien cree el objeto.

Problema: Crear componentes reutilizables. Creación de objetos.

Solución: Inyecta a cada objeto los objetos necesarios según las relaciones plasmadas en un fichero de configuración.

Implementación:

```
<bean id="accionCrearTramaSMDR" class="comunicador.presentacion.accion.AccionCrearTramaSMDR">

<property name="container" ref="frmCrearTramaSMDR"></property>
```

```
<property name=”servicioNomenclador” ref=”servicioNomencladorImpl”></property>
<property name=”gestionPBXManagerImpl” ref=” gestionPBXManagerImpl”></property>
 </bean>
public class AccionCrearTramaSMDR extends AccionModal {
        private IGentionPBXManager gestionPBXManagerImpl;
        private IServicioNomenclador servicioNomenclador;
        }
```

3.3 Diseño de Base de Datos

3.3.1 Modelo lógico de Datos

El modelo lógico de datos provee al usuario de una vista de las entidades lógicas de datos y sus relaciones, con independencia de la plataforma de base de datos a utilizar.

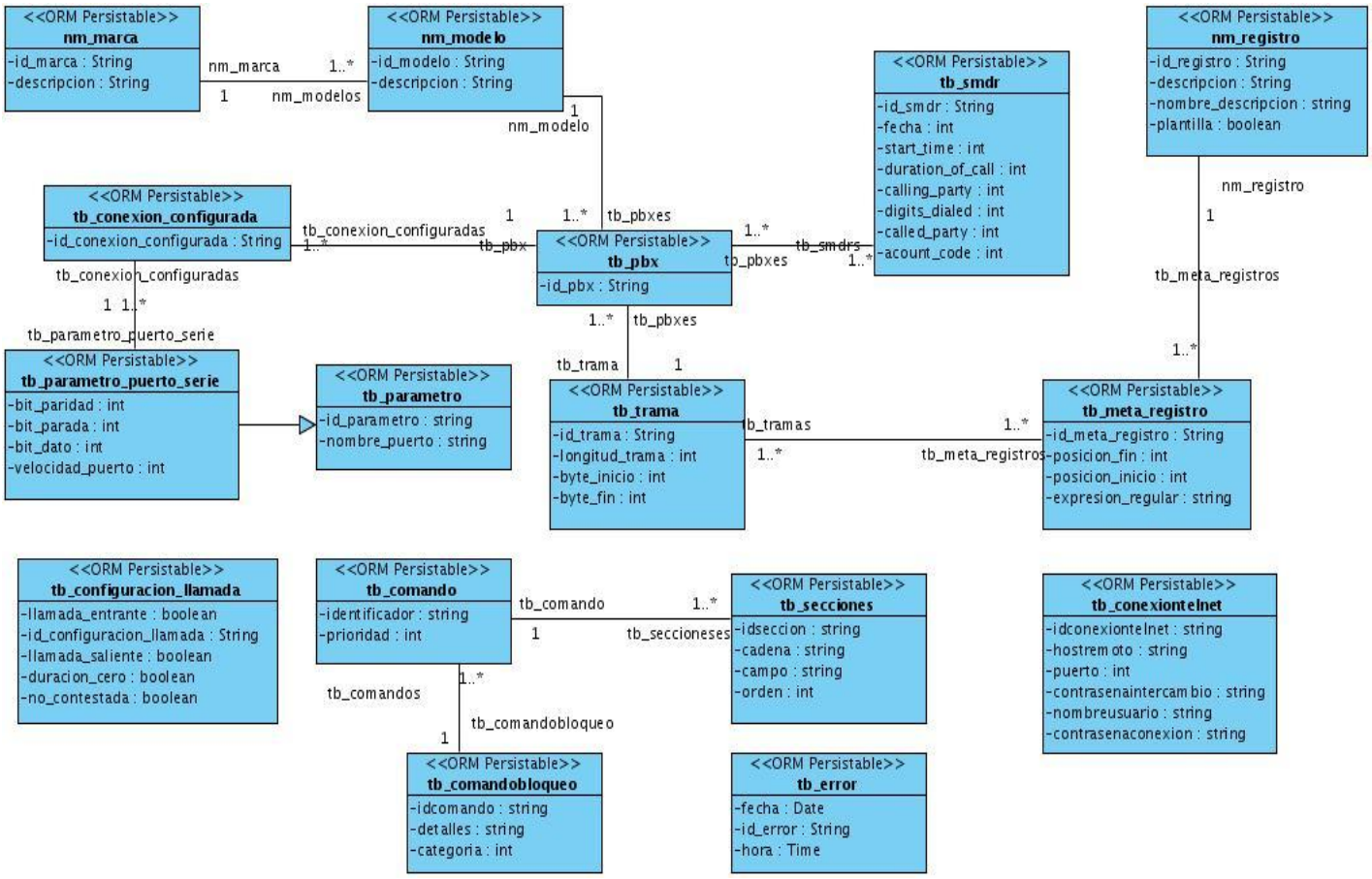


Figura 11 Diagrama de clases persistentes.

En la figura se muestra el diagrama de clases persistentes del sistema, el cual contiene los datos que se van almacenar de forma permanente en la base de datos para su posterior utilización.

3.3.2 Modelo físico de Datos del sistema

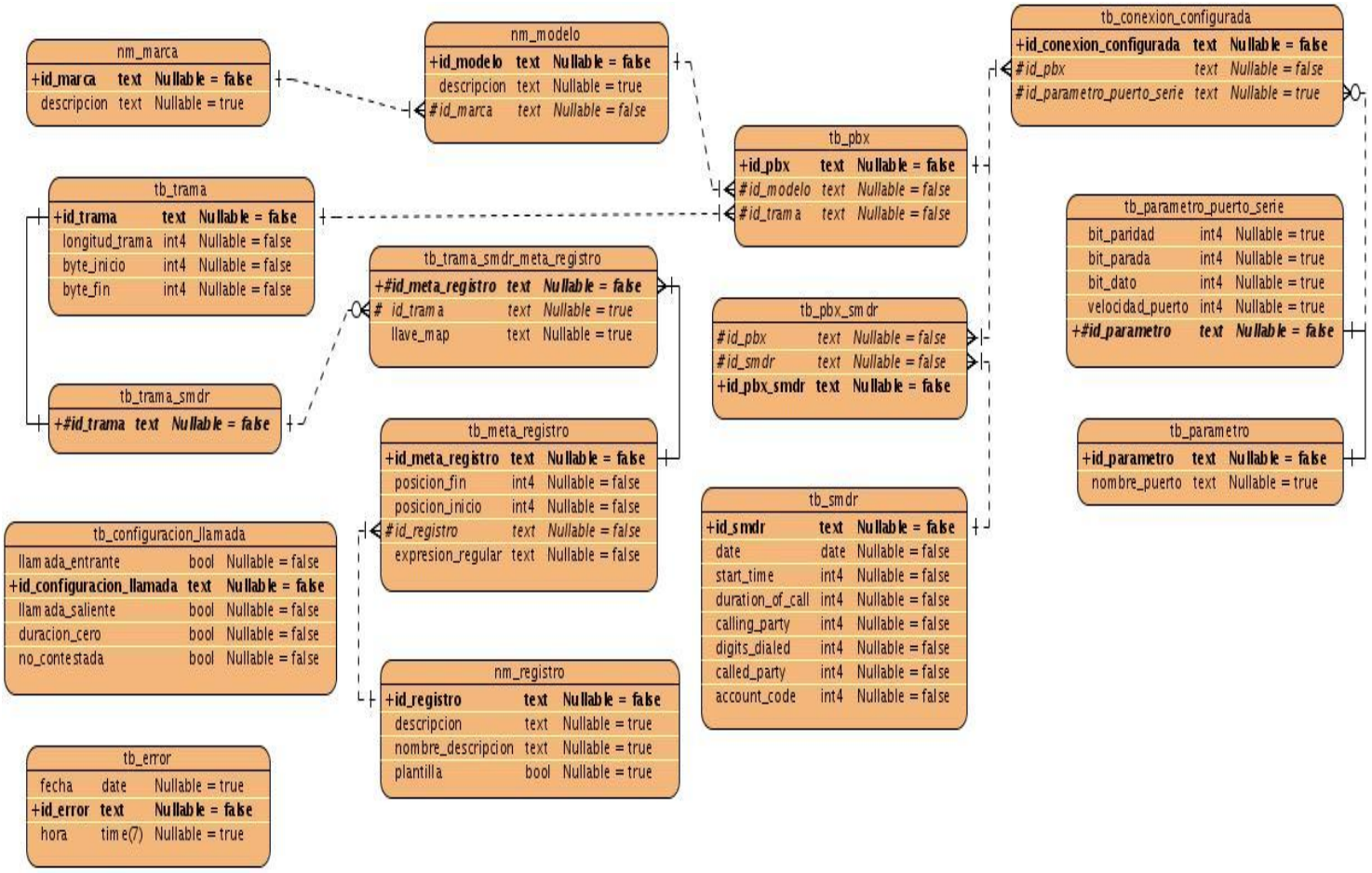


Figura 12 Diagrama entidad-relación del sistema.

La figura muestra el modelo entidad relación del sistema, el cual usa diagramas para representar la estructura natural de los datos, basándose en un conjunto de entidades, atributos de las entidades y sus interrelaciones.

3.4 Diagrama de Despliegue

El modelo de despliegue muestra las relaciones físicas de los distintos nodos que componen un sistema.

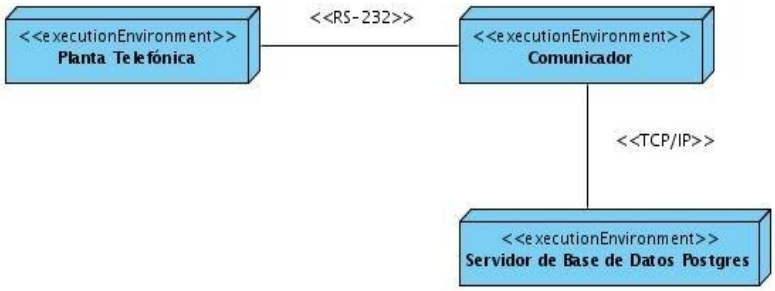


Figura 13 Diagrama de despliegue del sistema.

El diagrama de despliegue del sistema está conformado por una planta telefónica la cual se conecta al comunicador mediante el puerto RS-232 y a su vez el comunicador a través de TCP/IP almacena o selecciona la información requerida del servidor de base de datos (PostgreSQL).

3.5 Diagrama de Componentes

El modelo de implementación describe cómo los elementos del modelo de diseño (las clases), se implementan en términos de componentes. Muestra además la organización y las dependencias entre un conjunto de componentes.

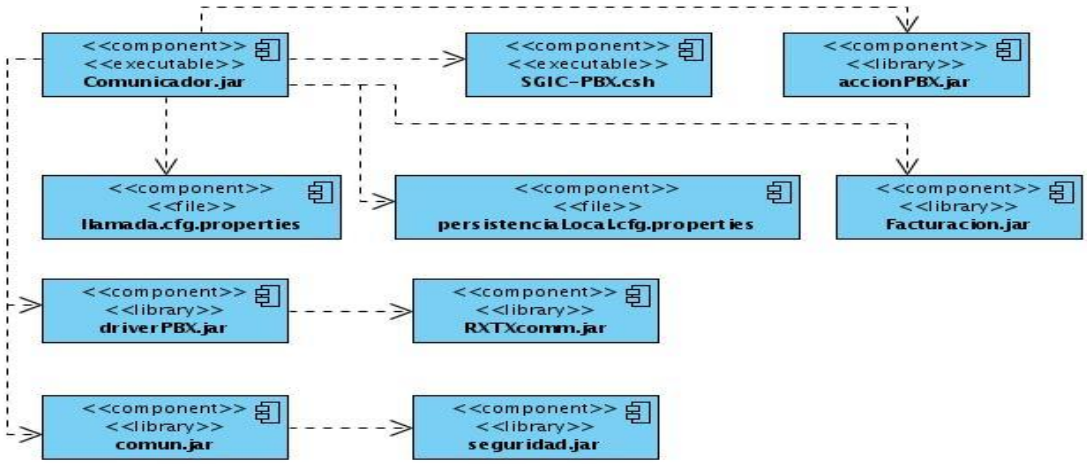


Figura 14 Diagrama de componentes del sistema.

Ejecutable del sistema

Componente Comunicador.jar agrupa cada una de las clases encargadas de obtener, almacenar e interpretar las tramas SMDR enviadas por la PBX, además de establecer y mantener la conexión con dicha pizarra.

Contenido

Comunicador.presentacion. accion

AccionGestionarTramaSMDR.java

AccionBuscarTramaSMDR.java

AccionCrearTramaSMDR.java

AccionDetallesTramaSMDR.java

AccionModificarTramaSMDR.java

AccionCancelarConexionPBX.java

AccionConexionPBX.java

AccionDetallesConexionPBX.java

AccionConfiguracionBase.java

AccionConfiguracionModoInicio.java

AccionPersistenaciaLocal.java

AccionResumenEstadistico.java

AccionComunicador.java

AccionPrincipalComunicador.java

Comunicador.presentacion.configuracion

Presentación.cfg.xml

Comunicador.presentacion.formulario

FrmGestionarTramaSMDR.java

FrmBuscarTramaSMDR.java

FrmCrearTramaSMDR.java

FrmModificarTramaSMDR.java

FrmPersistenciaLocal.java

FrmCancelarConexionPBX.java

FrmConfigurarConexionPBX.java

FrmDetallesConexionPBX.java

FrmConfiguracionBase.java

FrmResumenEstadistico.java

Comunicador.presentacion.servicio

AbstractFactoryTrayIcon.java

ActualizarComponente.java

FactoryTrayIconCerrarSesion.java

FactoryTrayIconCerrarVentana.java

ICambioEstadoTrayIcon.java

TrayIconPBX.java

Comunicador.negocio.configuracion

negocio.fachada.cfg.xml

negocio.manager.cfg.xml

Comunicador.negocio.fachada

IConfiguracionInicialFachada.java

Comunicador.negocio.fachada.impl

ConfiguracionInicialFachadaImpl.java

Comunicador.negocio.fichero

llamada.cfg.properties

persistenciaLocal.cfg.properties

Comunicador.negocio.manager

IConfiguracionInicialManager.java

IConfiguracionBaseManager.java

IConfiguracionInicialModoInicioManager.java

IConfiguracionPersistenciaLocalManager.java

IControlRegistroSMDRManager.java

IGestionPBXManager.java

IResumenEstadisticoManager.java

Comunicador.negocio.manager.impl

ConfiguracionInicialManagerImpl.java

ConfiguracionBaseManagerImpl.java

ConfiguracionInicialModoInicioManagerImpl.java

ConfiguracionPersistenciaLocalManagerImpl.java

ControlRegistroSMDRManagerImpl.java

GestionPBXManagerImpl.java

ResumenEstadisticoManagerImpl.java

ManejadorNotificacionImpl.java

Comunicador.negocio.servicio

EspecificacionByte.java

EspecificacionRS232.java

IServicioNomenclador.java

Nomenclador.java

NotificacionPBX.java

ServicioNomencladorImpl.java

Comunicador.accesoDatos

IConfiguracionInicialDao.java

IPBXDao.java

ISMDRDao.java

Comunicador.accesoDatos.impl

ConfiguracionInicialDaoImpl.java

PBXDaoImpl.java

SMDRDaoImpl.java

Comunicador.accesoDatos.configuracion

accesoDatos.cfg.xml

conexión_configurada.hbm.xml

hibernate.properties

hibernate.hbm.xml

metaDatoRegistro.hbm.xml

modelo.hbm.xml

parametro_puerto_serie.hbm.xml

pbx.hbm.xml

pbx_smdr.hbm.xml

registro.hbm.xml

tramaSMDR.hbm.xml

Comunicador.accesoDatos.servicio

INomencladorDao.java

NomencladorDaoImpl.java

Comunicador.dominioApp

ConexionConfigurada.java

ConfiguracionLlamada.java

ConfiguracionPersistenciaLocal.java

Marca.java

Modelo.java

PBX.java

PbxSmdr.java

Resumen.java

Comunicador.contexto

EspecificacionContexto.java

Comunicador.contexto.configuracion

contextoAplicacion.cgf.xml

driver.cfg.xml

menu.cfg.xml

seguridad.cfg.xml

Comunicador.recurso

detallar.gif

editar.gif

telefonoOff.gif

telefonoOn.gif

Comunicador.util

ReferenciaArchivo.java

Librerías

SGIC-PBX.csh: ejecutable de la ayuda del sistema.

accionPBX.jar librería que se encarga de controlar los errores del sistema, además de contener las acciones fundamentales y el contexto de la aplicación.

Facturacion.jar librería encargada de facturar las llamadas realizadas a través de la pizarra instalada.

driverPBX.jar librería que permite configurar el puerto RS-232, así como controlar e interpretar las tramas SMDR.

RXTXcomm.jar librería utilizada por el driverPBX para interactuar con el puerto RS-232.

comun.jar librería que contiene todas las clases comunes que son utilizadas por los diferentes módulos del proyecto en general.

seguridad.jar librería que se encarga de la seguridad del sistema, dígase la gestión de usuarios, encriptación de información, entre otros.

llamada.cfg.properties fichero donde se guarda el tipo de llamada que controlará el sistema.

persistenciaLocal.cfg.properties fichero donde se especifica la dirección donde se almacenarán los SMDR en forma de ficheros de texto.

3.6 Conclusiones

En este capítulo se realizó el diseño e implementación de todo el Módulo Comunicador del SGIC-PABX, obteniéndose los diferentes diagramas que requieren estos flujos de trabajo de Ingeniería de Software, se tuvo en cuenta para los diagramas de clase del diseño algunas de las facilidades que aportan los frameworks y los patrones de diseño que sin forzar al programador ayudan a la solución propuesta.

CAPÍTULO 4: ESTUDIO DE FACTIBILIDAD

4.1 Introducción

En este capítulo se hace un análisis de los costos, el esfuerzo y beneficios que proporciona el proyecto, llevado a cabo debido a la importancia que reporta el mismo y la ventaja que proporciona la utilización óptima de los recursos de la entidad.

El estudio de factibilidad se realiza mediante el análisis de Puntos de Casos de Uso, método de estimación de tiempo de desarrollo del proyecto, a partir de las características de sus requisitos, expresados en los casos de uso.

4.2 Estimación

Una vez determinados los casos de uso que guiarán el desarrollo del software, se puede predecir una estimación del tiempo de duración del proyecto mediante el análisis de Puntos de Casos de Uso.

“La estimación mediante el análisis de Puntos de Casos de Uso se trata de un método de estimación del tiempo de desarrollo de un proyecto mediante la asignación de "pesos" a un cierto número de factores que lo afectan, para finalmente, contabilizar el tiempo total estimado para el proyecto a partir de esos factores. A continuación se detallan los pasos a seguir para la realización de este método” (9).

Paso1. Cálculo de Puntos de Casos de Uso sin ajustar.

El cálculo de Puntos de casos de Uso sin ajustar se calcula mediante la siguiente ecuación:

UUCP = UAW + UUCW

Donde:

UUCP: Puntos de Casos de Uso sin ajustar

UAW: Factor de Peso de los Actores sin ajustar

UUCW: Factor de Peso de los Casos de Uso sin ajustar.

1.1 Factor de Peso de los Actores sin ajustar (UAW):

Este valor se calcula mediante un análisis de la cantidad de actores presentes en el sistema y la complejidad de cada uno de ellos. Los criterios a tener en cuenta se detallan a continuación.

Tipo de actor	Descripción	Factor de peso	Cantidad*Peso
Simple	Otro sistema que interactúa con el sistema a desarrollar mediante una interfaz de programación (API, Application Programming Interface).	1	0*1
Medio	Otro sistema que interactúa con el sistema a desarrollar mediante un protocolo o una interfaz basada en texto.	2	1*2
Complejo	Una persona que interactúa con el sistema mediante una interfaz gráfica.	3	1*3
Total UAW ((∑(actores*Peso))			5

1.2 Factor de Peso de los Casos de Uso sin ajustar (UUCW).

El valor del Factor de Peso de los Casos de Uso sin ajustar se calcula mediante un análisis de la cantidad de Casos de Uso presentes en el sistema y la complejidad de cada uno de ellos. La complejidad de los Casos de Uso se establece teniendo en cuenta la cantidad de transacciones efectuadas en el mismo, donde una transacción es una secuencia de actividades atómicas, es decir, se efectúa la secuencia de actividades completa, o no se efectúa ninguna de las actividades de la secuencia. Los criterios se muestran en la siguiente tabla:

Tipo de actor	Descripción	Factor de peso	Cantidad*Peso
Simple	El Caso de Uso contiene de 1 a 3 transacciones	5	16*5
Medio	El Caso de Uso contiene de 4 a 7 transacciones	10	1*10
Complejo	El Caso de Uso contiene más de 8 transacciones	15	2*15
Total UUCW((∑CU*Peso))			120

Luego:

UUCP = 5 + 120

UUCP = 125 (Factor de Peso de los Casos de Uso sin ajustar)

Paso2. Cálculo de Puntos de Casos de Uso ajustados

Ya obtenidos los Puntos de casos de uso sin ajustar, se debe ajustar este valor mediante la siguiente ecuación:

UCP = UUCP x TCF x EF

Donde:

UCP: Puntos de Casos de Uso ajustados

UUCP: Puntos de Casos de Uso sin ajustar

TCF: Factor de complejidad técnica

EF: Factor de ambiente

El factor de complejidad técnica (TCF) se calcula mediante la cuantificación de un conjunto de factores que determinan la complejidad técnica del sistema. Cada factor se cuantifica en un valor desde 0 a 5, donde 0 sería un aporte irrelevante y 5 sería un aporte muy relevante.

Factor	Descripción	Peso	Valor	Σ (Pesoi * Valori)
T1	Sistema distribuido	2	0	0
T2	Objetivos de performance o tiempo de respuesta	1	5	5
T3	Eficiencia del usuario final	1	3	3
T4	Procesamiento interno complejo	1	1	1
T5	El código debe ser	1	5	5

	reutilizable			
T6	Facilidad de instalación	0.5	4	2
T7	Facilidad de uso	0.5	5	2.5
T8	Portabilidad	2	5	10
T9	Facilidad de cambio	1	3	3
T10	Concurrencia	1	5	5
T11	Incluye objetivos especiales de seguridad	1	1	1
T12	Provee acceso directo a terceras partes	1	0	0
T13	Se requieren facilidades especiales de entrenamiento a los usuarios	1	2	2
Total TCF				39.5

- **Para Calcular TCF**

TCF = 0.6 + 0.01 * Σ (Peso i * Valor i)

TCF = 0.6 + 0.01 *39.5

TCF =0.995 (Factor de complejidad técnica)

El factor de ambiente (EF) está relacionado con las habilidades y entrenamiento del grupo de desarrollo que realiza el sistema. Cada factor se cuantifica con un valor desde 0 a 5.

Factor	Descripción	Peso	Valor	Σ (Pesoi * Valori)
E1	Familiaridad con el modelo de proyecto utilizado.	1.5	1	1.5
E2	Experiencia en la aplicación.	0.5	0	0
E3	Experiencia en orientación a objetos	1	4	4
E4	Capacidad del analista líder	0.5	4	2
E5	Motivación	1	5	5
E6	Estabilidad de los	2	3	6

	requerimientos			
E7	Personal part-time	-1	2	-2
E8	Dificultad del lenguaje de programación	-1	3	-3
Total EF				13,5

- **Para Calcular EF**

EF = 1.4 - 0.03 * Σ (Peso i * Valor i)

EF = 1.4 - 0.03 *13.5

EF = 0,995 (Factor de ambiente)

Calculando los puntos de Caso de Uso Ajustados quedaría:

UCP = UUCP * TCF * EF

UCP = 125 * 0.995 *0,995

UCP = 123,7531 (Puntos de Casos de Uso ajustados)

Paso3. Estimación de esfuerzo a través de los puntos de casos de uso.

Se tiene la siguiente ecuación:

E = UCP * CF

Donde:

E: Esfuerzo estimado en horas-hombre.

UCP: Puntos de Casos de Uso ajustados.

CF: Factor de conversión.

- **Para calcular CF:**

Para calcular el Factor de conversión se tiene en cuenta lo siguiente:

- Se contabilizan cuántos factores de los que afectan al Factor de ambiente están por debajo del valor medio (3), para los factores E1 a E6.
- Se contabilizan cuántos factores de los que afectan al Factor de ambiente están por encima del valor medio (3), para los factores E7 y E8.
- Como el total contabilizado es 3 se utiliza el factor de conversión 28 horas-hombre/Punto de Casos de Uso.

Total EF =3+0

Total EF =3.

E = UCP * CF

E = 123,7531 * 28 horas-hombre

E = 3465,09 horas-hombre (Esfuerzo estimado en horas-hombre)

Paso4. Cálculo del Esfuerzo de todo el proyecto:

Actividad	% esfuerzo	Valor esfuerzo
Negocio	10,00%	577,515 horas-hombre
Diseño	30,00%	1732,54 horas-hombre
Implementación	60,00%	3465,09 horas-hombre
Total	100%	5775,15 horas-hombre

EL esfuerzo total (ET) del proyecto sería:

ET = 5775,15 Horas-Hombres.

ET = 40,1052 Mes-Hombres.

El tiempo de desarrollo (TD) del proyecto sería:

TD = ET / CH

TD = 40,1052 Mes-Hombres **/ 1** Hombre

TD = 40,1052 Mes

Donde:

CH: Cantidad de Hombres.

Teniendo un hombre, se estima que el tiempo de desarrollo del proyecto sería aproximadamente 40 meses. Para un equipo de **8** personas el proyecto tiene una duración de **5** meses.

Paso5. Cálculo del costo total del proyecto

El costo total del proyecto (CT) sería:

CT = ET * CHM / CH

CT = 40,1052 Mes-Hombres *** 81** Hombre-Mes **/ 1** Hombre

CT = 3248,52

Donde:

CHM: Costo Hombre-Mes.

Asumiendo que el costo por Hombre-Mes es $81 se estima que el costo total del proyecto sería $ **3248,52** aproximadamente.

4.3 Beneficios Tangibles e Intangibles.

Se definen a los beneficios tangibles como aquellos que reportan ventajas económicas cuantificables.

El Módulo Comunicador para la Gestión Integral de los Costos de llamadas en Pizarras Telefónicas es un producto utilizado con fines comerciales u otros intereses similares. Su desarrollo permitirá procesar y almacenar la información brindada por los SMDR. El beneficio principal que reporta el sistema es contar con una solución informática que permita recopilar de manera eficiente la información de las llamadas realizadas hacia o desde la empresa.

Se definen los beneficios intangibles como aquellos que reportan beneficios organizativos, de funcionamiento o eficiencia.

En cuanto a los **beneficios intangibles** que reporta el sistema se pueden mencionar:

Facilidad de operación con una interfaz sencilla y amigable, con resultados inmediatos para administradores y operadores de las PBX.

Acceso a la información en tiempo real de las llamadas recibidas, así como los errores por envío de SMDR y errores de conexión.

Disminución de los gastos, debido a que no hay que comprar el software.

4.4 Análisis de costos y beneficios.

El Módulo Comunicador para la Gestión Integral de los Costos de llamadas en Pizarras Telefónicas no requiere de inversión de software porque las herramientas y la tecnología propuestas para su desarrollo son libres, por lo que una vez analizado el costo del proyecto y los beneficios que este reporta se puede concluir que el sistema es factible desarrollarlo y que su uso contribuirá a que se lleven a cabo eficientemente los procesos de obtención de información de las llamadas telefónicas para su uso posterior.

4.5 Conclusiones

En este capítulo se desarrolló la estimación por Puntos de Caso de Uso que resultó muy efectiva para estimar el esfuerzo del proyecto teniendo en cuenta los factores que influyen en el desarrollo del software. Se realizó un análisis de los costos y beneficios tangibles e intangibles que proporciona el proyecto que permitió valorar qué tan factible sería la realización del mismo.

CONCLUSIONES

En el presente trabajo de diploma se investigaron las principales características y funcionamiento de las pizarras telefónicas. En este se recoge todo el proceso de desarrollo del módulo Comunicador del Sistema para la Gestión Integral de los costos de llamadas en pizarras telefónicas (SGIC-PABX), logrando obtener una aplicación funcional que responde a cada uno de los requerimientos y objetivos planteados.

Se obtiene como resultado un software con las siguientes funcionalidades: permite opciones de configuración de parámetros del puerto serie y de ficheros. Selección del modelo y la marca de la PABX a la cual se va a conectar el sistema y creación de la trama SMDR que es utilizada por esa pizarra telefónica en particular. Interpretación de las llamadas recibidas en el sistema en un formato entendible. Información en tiempo real del total de llamadas recibidas, errores y otros datos de la aplicación.

Para lograr este resultado se utilizaron diferentes herramientas que ayudaron a complementar el desarrollo de la aplicación.

La metodología RUP que contribuyó a una mejor organización y entendimiento del sistema gracias a su ciclo de desarrollo. El lenguaje Java que además de ser orientado a objeto está diseñado para desarrollar sistemas altamente fiables, es un lenguaje compatible con el entorno de desarrollo seleccionado, el cual permite crear aplicaciones de escritorio, la ingeniería inversa, así como la integración con los frameworks spring e hibernate.

RECOMENDACIONES

Realizar una versión del Sistema para las pizarras que no tengan conexión por vía serial.

Realizar las pruebas necesarias para evaluar y garantizar la calidad del sistema.

BIBLIOGRAFÍA

1. **Jacobson, Ivar.** El proceso unificado de desarrollo de software. *El proceso unificado de desarrollo de software.* Madrid : Pearson Education , 2000.

2. **Rumbaugh, James.** El lenguaje de Modelado. Manual de Referencia. [book auth.] Ivar Jacobson, Grady Booch James Rumbaugh. *El lenguaje de Modelado. Manual de Referencia.* 1998.

3. Curso de java. *Características del Lenguaje.* [Online] [Cited: 12 14, 2009.] http://tikal.cifn.unam.mx/~jsegura/LCGII/java3.htm.

4. **Celis, Ismael.** El ataque de los Frameworks. *El ataque de los Frameworks.* [Online] Factoría de Internet. [Cited: 04 12, 2010.] http://www.webtaller.com/maletin/articulos/el-ataque-de-los-frameworks.php.

5. **González, Héctor Suárez.** JavaHispano. *Manual Hibernate.* [Online] [Cited: 01 6, 2010.] http://www.javahispano.org/contenidos/es/manual_hibernate/.

6. **Jacobson, Ivar.** El proceso unificado de desarrollo de software. [book auth.] James Rumbaugh, Grady Booch Ivar Jacobson. *El proceso unificado de desarrollo de software.* Madrid : Pearson Education, 2000.

7. **Oktaba, Hanna.** Introducción a Patrones. *Introducción a Patrones.* [Online] [Cited: 03 20, 2010.] http://www.mcc.unam.mx/~cursos/Algoritmos/javaDC99-2/patrones.html.

8. **Garcia, Joaquin.** Patrones de Diseño. *Patrones de Diseño.* [Online] 05 27, 2005. [Cited: 03 18, 2010.] http://www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php.

9. eva.uci.cu. *Clase Teórico Practica 2 Estimación.* [Online] [Cited: 04 15, 2010.] http://eva.uci.cu/mod/resource/view.php?id=22433.

10. PostGreSQL vs. MySQL. *PostGreSQL.* [Online] [Cited: 12 16, 2009.] http://danielpecos.com/docs/mysql_postgres/x15.html.

11. Garbage Collector. *Sistema Gestor de Base de Datos (SGBD).* [Online] 11 1, 2004. [Cited: 12 16, 2009.] http://www.error500.net/garbagecollector/archives/categorias/bases_de_datos/sistema_gestor_de_base_de_datos_sgbd.php.

12. TiendasLinux. *Ventajas de PostgreSQL.* [Online] [Cited: 12 17, 2009.] http://soporte.tiendalinux.com/portal/Portfolio/postgresql_ventajas_html.

13. MiTecnologico. *Definición Lenguaje de Programación.* [Online] [Cited: 12 13, 2009.] http://www.mitecnologico.com/Main/DefinicionDeLenguajeDeProgramacion.

14. **Alvarez, Miguel Angel.** desarrolloweb. *Que es Java.* [Online] [Cited: 12 14, 2009.] http://www.desarrolloweb.com/articulos/497.php.

15. **García, Felipe U. Pérez.** El lenguaje C. *El lenguaje C.* [Online] [Cited: 12 14, 2009.] http://www.articulandia.com/premium/article.php/12-03-2007El-Lenguaje-C.htm.

16. **Sanchez, María A. Mendoza.** informatizate. *Metodologias de desarrollo de software.* [Online] [Cited: 12 12, 2009.] http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html.

17. sourceforge. *introducción a eXtreme Programming.* [Online] [Cited: 12 11, 2009.] http://oness.sourceforge.net/proyecto/html/ch05.html.

18. visual paradigm para UML. *visual paradigm para UML.* [Online] [Cited: 01 8, 2010.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(M%C3%8D)_14720_p/.

19. Conclusiones. Spring. [Online] [Cited: 01 9, 2010.] http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/.../capitulo6.pdf.

20. **Bagüés, Ramiro Lago.** Framework Spring. [Online] [Cited: 01 9, 2010.] http://www.proactiva-calidad.com/java/spring/introduccionSpring.html.

21. **Cerda, Felipe.** Netbeans. [Online] [Cited: 12 14, 2001.] http://www.techbloog.com/talks/netbeans65es_cl.pdf.

22. **Alvarez, Miguel Angel.** desarrolloweb. *Que es Python.* [Online] [Cited: 12 14, 2009.] http://www.desarrolloweb.com/articulos/1325.php.

23. Arquitectura de Software. [Online] [Cited: 03 15, 2010.] http://docs.sun.com/app/docs/doc/819-3589/6n5q3ttr9?l=es&a=view.

24. eva.uci.cu. *Conferencia 6.* [Online] [Cited: 04 5, 2010.] http://eva.uci.cu/mod/resource/view.php?id=22095.

25. **Booch, Grady, Jacobson, Ivar and Rumbaugh, James.** *El Proceso Unificado de Desarrollo de Software.* 2000.

26. **Flores, Mirian Milagros Diaz.** RUP vs XP. [Online] [Cited: 02 15, 2010.] http://www.usmp.edu.pe/publicaciones/boletin/fia/info49/articulos/RUP%20vs.%20XP.pdf.

REFERENCIAS BIBLIOGRÁFICAS

1. **Jacobson, Ivar.** El proceso unificado de desarrollo de software. *El proceso unificado de desarrollo de software.* Madrid : Pearson Education , 2000.

2. **Rumbaugh, James.** El lenguaje de Modelado. Manual de Referencia. [aut. libro] Ivar Jacobson, Grady Booch James Rumbaugh. *El lenguaje de Modelado. Manual de Referencia.* 1998.

3. Curso de java. *Características del Lenguaje.* [En línea] [Citado el: 14 de 12 de 2009.] http://tikal.cifn.unam.mx/~jsegura/LCGII/java3.htm.

4. **Celis, Ismael.** El ataque de los Frameworks. *El ataque de los Frameworks.* [En línea] Factoría de Internet. [Citado el: 12 de 04 de 2010.] http://www.webtaller.com/maletin/articulos/el-ataque-de-los-frameworks.php.

5. **González, Héctor Suárez.** JavaHispano. *Manual Hibernate.* [En línea] [Citado el: 6 de 01 de 2010.] http://www.javahispano.org/contenidos/es/manual_hibernate/.

6. **Jacobson, Ivar.** El proceso unificado de desarrollo de software. [aut. libro] James Rumbaugh, Grady Booch Ivar Jacobson. *El proceso unificado de desarrollo de software.* Madrid : Pearson Education, 2000.

7. **Oktaba, Hanna.** Introducción a Patrones. *Introducción a Patrones.* [En línea] [Citado el: 20 de 03 de 2010.] http://www.mcc.unam.mx/~cursos/Algoritmos/javaDC99-2/patrones.html.

8. **Garcia, Joaquin.** Patrones de Diseño. *Patrones de Diseño.* [En línea] 27 de 05 de 2005. [Citado el: 18 de 03 de 2010.] http://www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php.

9. eva.uci.cu. *Clase Teórico Practica 2 Estimación.* [En línea] [Citado el: 15 de 04 de 2010.] http://eva.uci.cu/mod/resource/view.php?id=22433.

GLOSARIO

Control de versiones: es la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo.

Complemento o plugin es una aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy especifica.

EJB (*Enterprise JavaBeans*): Proporcionan un modelo de componentes distribuido estándar del lado del servidor. El objetivo de los EJB es dotar al programador de un modelo que le permita abstraerse de los problemas generales de una aplicación empresarial (concurrencia, transacciones, persistencia, seguridad, etc.) para centrarse en el desarrollo de la lógica de negocio en sí. El hecho de estar basado en componentes permite que éstos sean flexibles y sobre todo reutilizables.

Hibernate: Framework de código abierto. Es una herramienta de Mapeo objeto-relacional para la plataforma Java que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) que permiten establecer estas relaciones.

IDE: 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. Proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación.

J2SE (*Java 2, Standard Edition*): Plataforma java para el desarrollo de software.

Lenguaje de scripting o interpretado: es un lenguaje de programación que está diseñado para ser ejecutado por medio de un intérprete, en contraste con los lenguajes compilados.

Multi-Version Concurrency Control (MVCC): es una técnica avanzada para mejorar las prestaciones de una base de datos en un entorno multiusuario.

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

RUP: Es un conjunto de actividades necesarias para transformar los requisitos de los usuarios en un sistema de software.

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

SMDR: registros generados por las pizarras telefónicas que contienen detalles de las llamadas realizadas a través de ellas.

Spring: Framework de código abierto de desarrollo de aplicaciones para la plataforma Java.

Sistemas de control de versiones facilitan la administración de las distintas versiones de cada producto desarrollado, así como las posibles especializaciones realizadas.

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

XP: es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo.