Universidad de las Ciencias Informáticas
Facultad 2

Título: Módulo Facturación para el Sistema de Gestión Integral de Costos de Llamadas - PABX

Trabajo de Diploma para optar por el título de

Ingeniero Informático

Autor: Ayme Robaina Camejo

Tutor: Ing. Carlos Molina Villalobos

Co-tutor: Ing. Félix Maikel García Pérez

Ciudad Habana, 21 de Junio del 2010

¨Año 52 de la Revolución¨

Declaración de Autoría

Declaro ser la única autora de la presente tesis y reconozco 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 _____.

Ayme Robaina Camejo

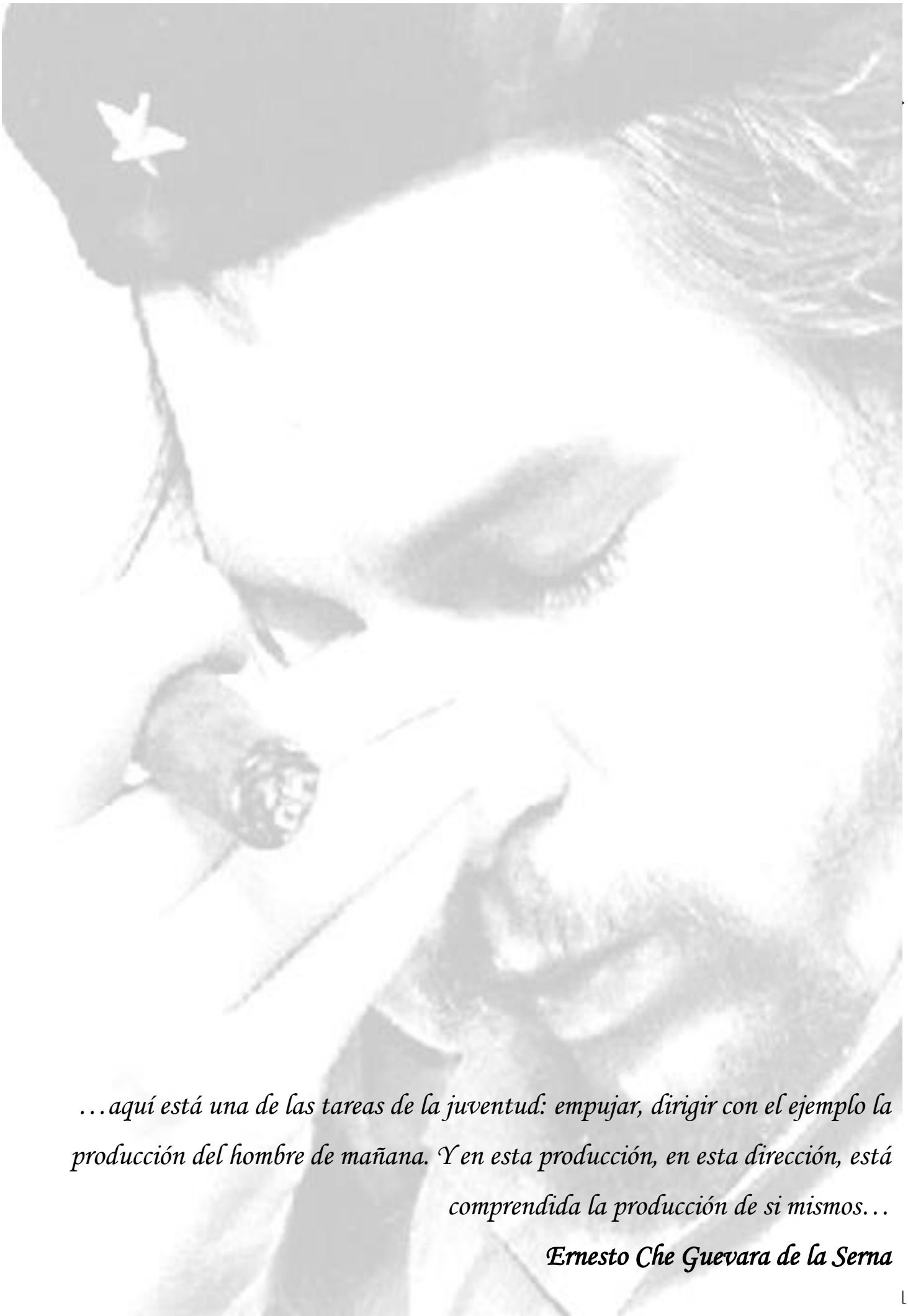
_____
Firma del Autor

Ing. Carlos Molina Villalobos

_____
Firma del Tutor

Ing. Félix Maikel García Pérez

_____
Firma del Co - Tutor



…aquí está una de las tareas de la juventud: empujar, dirigir con el ejemplo la producción del hombre de mañana. Y en esta producción, en esta dirección, está comprendida la producción de si mismos…

Ernesto Che Guevara de la Serna

Agradecimientos.

Agradezco:

A mi familia que es la razón de mi existir, sin su apoyo y confianza no lo hubiera logrado.

A mi mami por estar conmigo en todo momento, por ser mi amiga incondicional, por sus consejos, su amor y fe en mi. Por su educación, su ejemplo y confianza.

A mi papi por ser mi fuerza, por luchar por mi, por guiarme. Por su educación y confianza.

A mi tía Luisa por brindarme siempre su ayuda, por cuidarme y haber sido más que una tía, ser mi madre.

A mi novio por su amor, apoyo y comprensión. Por estar a mi lado de forma incondicional estos 5 años de sacrificio para los dos.

A mis tutores Carlos y Félix por su ayuda, paciencia, sus conocimientos aportados, que siempre me ayudaron en todo lo que estuvo a su alcance.

A mis vecinas Magnolia y Marisllanis por sus buenas críticas, sus consejos, apoyo y cariño.

A todas mis amistades de la carrera y algunas que están más lejos, por su amistad incondicional, sus consejos y su apoyo estando lejos de mi familia. Por hacer mejores mis días en la UCI.

Ayme Robaina Camejo

Dedicatoria.

Dedico este trabajo a toda mi familia, a mi mamá, mi papá, mi abuela, mis tías, mi novio, mis vecinas, mis amigos, todos los que de una forma u otra me han dado su apoyo incondicional, y me han alentado siempre a lograr mis objetivos. A ellos que son la razón de mi vida agradecerles por ayudarme ha hacer mis sueños realidad y ser lo que soy.

Mechi.

Resumen

Las instituciones que adquieren pizarras telefónicas privadas, perciben como principal objetivo la comunicación telefónica entre sus diferentes estructuras organizativas; restándole importancia al control que este proceso amerita. Las empresas solicitan un software para monitorear y controlar toda la información relacionada con las llamadas realizadas. Las peculiaridades de una llamada son detalladas por un *Station Message Detail Recording,* por sus siglas del ingles (SMDR) que genera la pizarra telefónica en tiempo real.

Una parte importante en este monitoreo, lo constituye la facturación de los servicios ofrecidos por la planta telefónica. Con los valores que conforman el SMDR, es posible establecer una relación directa entre servicio, consumidor y costo; brindando información valiosa que facilita la toma de decisiones. La ejecución de las acciones que se derivan del análisis de un posible informe de facturación depende de la disponibilidad y rapidez con que se obtenga la información.

El objetivo del trabajo es desarrollar el módulo Facturación del Sistema para la Gestión Integral de Costos de Llamadas en Pizarras Telefónicas (SGIC - PABX) para solucionar los problemas existentes en nuestro país respecto a la facturación de las llamadas telefónicas en las pizarras telefónicas privadas adquiridas por las diferentes instituciones.

Al finalizar el Módulo Facturación para SGIC – PABX se espera lograr una herramienta que brinde a los usuarios la posibilidad de mantener un control del consumo del presupuesto gastado por concepto de llamadas telefónicas a través del proceso de facturación, una vez sobrepasado este, el software automáticamente bloqueará las diferentes extensiones o códigos de cuenta verificable que tienen asociado dicho presupuesto.

PALABRAS CLAVE

SMDR, PABX, SGIC-PABX

TABLA DE CONTENIDOS

Índice de Tablas y Figuras

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Las telecomunicaciones en la historia del desarrollo humano, han merecido elogios, acortando distancias en el avance de los logros científicos técnicos. De manera sencilla, las telecomunicaciones son una forma de comunicación electrónica a distancia, que satisface las necesidades de enlace rápido que requiere la humanidad para la solución de sus problemas. Las redes inalámbricas, el fax, el internet y el teléfono, son en la actualidad algunos de los medios de telecomunicación de mayor eficacia en el desarrollo del siglo XXI.

Cuba no quedó atrás en el desarrollo de la telefonía en el transcurso del tiempo. Con la creación en 1994 de la Empresa de Telecomunicaciones de Cuba (ETECSA) S.A. en conjunto con la italiana STET, se dio un importante paso en la organización y puesta en funcionamiento de una nueva infraestructura para los servicios de telefonía.

Actualmente ETECSA entre múltiples servicios, ofrece la venta de pizarras telefónicas privadas (***P****rivate* ***B****ranch* *E****x****change* y ***P****rivate* ***A****utomatic* ***B****ranch* *E****x****change* por sus siglas en inglés respectivamente, *PBX* y PABX). El mismo 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.

Dentro de los distintos proveedores mundiales de pizarras telefónicas se encuentran: MITEL, ALCATEL, PANASONIC, ERICSSON, INFINITY, TOSHIBA y LG; de los cuales existen varios modelos en todo el territorio nacional.

Las instituciones 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. Las empresas solicitan un software para monitorear y controlar toda la información relacionada con las llamadas realizadas. Las peculiaridades de una llamada son detalladas por un SMDR que genera la pizarra telefónica en tiempo real.

Una parte importante en este monitoreo, sin lugar a dudas, lo constituye la facturación de los servicios ofrecidos por la planta telefónica. Con los valores que conforman el SMDR, es posible establecer una relación directa entre servicio, consumidor y costo; brindando información valiosa que facilita la toma de decisiones.

La ejecución de las acciones que se derivan del análisis de un posible informe de facturación depende de la disponibilidad y rapidez con que se obtenga la información.

Al analizar la situación existente se delinean los siguientes problemas:

- Inexistencia de un mecanismo que proporcione notificaciones automáticas al sobrepasar un presupuesto asignado por concepto de facturación de servicios.
- Imposibilidad para realizar un análisis periódico del consumo real y el presupuesto planificado para cada área o departamento.
- Sobre consumo del presupuesto asignado para las diferentes áreas o departamentos.

Este proceso es una actividad compleja, ya que requiere de la interacción entre una PABX y la computadora dedicada a recopilar los datos, por el manejo de información en tiempo real, y la prestación del servicio de forma ininterrumpida. Las funcionalidades desarrolladas para soportar dicho proceso estarán sustentadas sobre mecanismos que garanticen la flexibilidad y escalabilidad de los sistemas que participen.

Dada esta situación problémica se impone el siguiente problema científico: ¿Cómo determinar los costos de las llamadas en pizarras telefónicas? Se define como objeto de estudio los servicios de las pizarras telefónicas PABX y el campo de acción facturación de llamadas en pizarras telefónicas PABX.

Como objetivo general de esta investigación se establece: Diseñar e implementar los mecanismos para la facturación de llamadas telefónicas del Sistema para la Gestión Integral de los Costos de Llamadas en Pizarras Telefónicas – PABX. A partir de este se derivan los objetivos específicos siguientes:

- Estructurar los mecanismos que permitan el control del consumo por la realización de llamadas telefónicas.
- Especificar el mecanismo de notificación automático a través de un servidor de correo electrónico.
- Estructurar los mecanismos que permitan el bloqueo automático de las extensiones o códigos de cuenta verificable una vez sobrepasado el saldo asignado.

Para satisfacer los anteriores objetivos se plantean las siguientes tareas de investigación**:**

- Analizar los puntos de interacción con otros componentes del sistema identificando la estrategia de integración.
- Investigar las tecnologías existentes para delinear un mecanismo que posibilite el envío de mensajes a través del servidor de correo.
- Analizar las tecnologías de conexión TELNET y Serial para diseñar un mecanismo que posibilite el bloqueo de las extensiones o códigos de cuenta verificable una vez sobrepasado el saldo asignado.

- Estudiar y analizar los patrones de diseño y estilos arquitectónicos, fundamentando un diseño robusto y flexible.
- Diseñar la estructura de los comandos de bloqueo para obtener una solución genérica.
- Realizar el diseño e implementación de los mecanismos para la facturación de llamadas telefónicas del SGIC – PABX.

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 documento está estructurado de la siguiente manera:

- **Capítulo 1. Fundamentación Teórica:** Se exponen los lenguajes, herramientas, metodologías, sistemas y conceptos a utilizar para el desarrollo de la aplicación.
- **Capítulo 2. Características del Sistema:** Incluye la descripción del sistema y de los procesos que serán automatizados. Se define el modelo de dominio, se identifican los requerimientos y los casos de uso del sistema.
- **Capítulo 3. Diseño e Implementación:** Se modela el diagrama de clases del diseño, los diagramas de interacción, el modelo de despliegue y el diagrama de componentes.
- **Capítulo 4. Estudio de la Factibilidad:** Se realiza un análisis de los costos y beneficios proporcionados por el proyecto.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En este capítulo se presenta la fundamentación teórica de la tesis. Se exponen conceptos importantes relacionados con el tema de las pizarras telefónicas, se explican también los conceptos que intervienen y que son necesarios para lograr una mejor comprensión del funcionamiento del proceso de facturación. Se caracterizan además, algunos de los sistemas de facturación de llamadas telefónicas implementados.

Además, se describen los aspectos fundamentales del trabajo, así como las herramientas y la metodología a utilizar. Se exponen los puntos que se tuvieron en cuenta para la selección de la metodología de desarrollo a utilizar, el entorno de desarrollo y el lenguaje de programación.

1.2 Pizarras Telefónicas PABX, SMDR

Una PABX cuya traducción al español sería intercambiador automático de redes privadas, es una central telefónica que pertenece a una empresa. El término se refiere a equipos de comunicaciones telefónicas destinados a establecer y mantener llamadas tanto internas (llamadas entre extensiones) como externas (llamadas realizadas a las líneas de la red pública de teléfono) y recibir llamadas provenientes de la red publica hacia extensiones dentro de la red privada.

Una PABX mantiene tres funciones esenciales: establecer llamadas entre dos o más usuarios, mantener la comunicación durante el tiempo que lo requiera el usuario y proveer información para la facturación de llamadas.

Los SMDR son registros generados por las pizarras telefónicas en tiempo real, que contienen detalles de las llamadas que se realizan a través de ellas. Existen distintos proveedores de PABX y cada pizarra genera SMDR con diferentes formatos.

1.2.1 Llamadas Telefónicas

Las llamadas telefónicas son detalladas por los SMDR que son generados en tiempo real por la planta telefónica. La longitud de los SMDR 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.

1.2.1.1 Clasificación de los Registros SMDR (Internos o Externos)

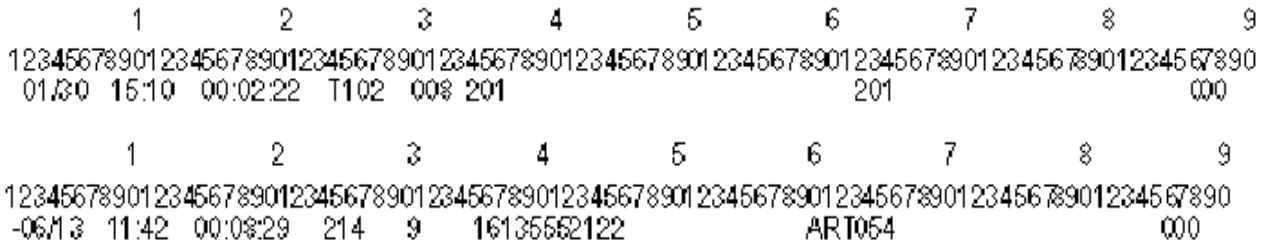
Los registros SMDR permiten clasificar las llamadas en internas y externas para cada extensión.

Ejemplo de SMDR que detalla una llamada interna:

04052226 00148 J 8710 **8644** 00 **8710** 8710

En este caso se realizó una llamada interna de la extensión 8644 a la extensión 8710.

Ejemplo de SMDR que detalla una llamada externa:



1.2.2 Llamadas Internas

Son llamadas gratuitas, ya que es la propia compañía la dueña de los dispositivos. El usuario marca directamente la extensión deseada sin pasar ésta por ninguna línea externa.

1.2.3 Llamadas Externas

Las llamadas externas son aquellas que se realizan hacia o desde un número que no se encuentra dentro de la red privada. Se clasifican en llamadas externas salientes y llamadas externas entrantes.

1.2.3.1 Llamadas Externas Entrantes

Las llamadas externas entrantes son las realizadas desde un número ubicado en la red pública hacia un número que se encuentra en la red privada.

1.2.3.2 Llamadas Externas Salientes

Las llamadas externas salientes son las realizadas desde un número que se encuentra dentro de la red privada hacia un número que se encuentra en la red pública. Son aquellas llamadas a las que se les realiza el proceso de facturación.

1.3 Facturación de Llamadas Telefónicas

La facturación de las llamadas telefónicas es el proceso que se realiza para obtener el valor monetario que se debe abonar por la realización de las mismas.

Para realizar la facturación de una llamada telefónica se tienen en cuenta varios factores:

- Número telefónico del cuál se realiza la llamada
- Fecha en que se realiza la llamada
- Hora de inicio de la llamada
- Duración
- Número marcado
- Área a la que pertenece el número marcado
- Zona en la que se ubica el área
- Tarifas correspondientes a la zona

1.3.1 Clasificación de las Llamadas Telefónicas Externas Salientes

Para clasificar las llamadas telefónicas externas salientes hay que tener en cuenta el lugar de origen y destino, para de esta manera, obtener la distancia entre los dos puntos que se comunican. Las llamadas son clasificadas en locales, de larga distancia, nacionales e internacionales.

1.3.1.1 Llamada Internacional

Las llamadas internacionales son aquellas donde el punto de origen y destino se encuentran en diferentes zonas territoriales, donde una zona territorial es un país.

1.3.1.2 Llamada Nacional

Las llamadas nacionales incluyen las llamadas locales y de larga distancia que son realizadas dentro del territorio nacional.

1.3.1.2.1 Llamada Externas Salientes Local

Las llamadas externas salientes locales son aquellas donde el punto de origen y de destino se encuentran ubicados en una misma área, pero es realizada desde la red privada hacia la red publica.

1.3.1.2.2 Llamada Larga Distancia

Las llamadas de larga distancia son aquellas donde el punto de origen y de destino se encuentran ubicados en diferentes áreas. Incluye las llamadas internacionales.

1.3.2 Elementos del Proceso de Facturación de Llamadas Telefónicas

1.3.2.1 Zonas y Áreas

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

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

En la siguiente tabla se muestra la organización de las áreas distribuidas en zonas, teniendo en cuenta la distancia que hay entre áreas.

Zonas LDN	*C. Habana*	*Habana*	*P. Río*	*I. Juventud*	*Matanzas*	*V. Clara*	*Cienfuegos*	*S. Spíritus*	*C. Ávila*	*Camagüey*	*Tunas*	*Holguín*	*Granma*	*S. Cuba*	*Guantanamo*
C. Habana.		***Z2***	***Z4***	***Z4***	***Z4***	***Z5***	***Z5***	***Z6***	***Z6***	***Z6***	***Z7***	***Z7***	***Z7***	***Z7***	***Z7***
La Habana	***Z2***		***Z4***	***Z4***	***Z4***	***Z5***	***Z5***	***Z6***	***Z6***	***Z6***	***Z7***	***Z7***	***Z7***	***Z7***	***Z7***
P. del Río	***Z4***	***Z4***		***Z4***	***Z5***	***Z6***	***Z6***	***Z6***	***Z6***	***Z7***	***Z7***	***Z7***	***Z7***	***Z7***	***Z7***
I. de la Juv.	***Z4***	***Z4***	***Z4***		***Z5***	***Z6***	***Z5***	***Z6***	***Z6***	***Z6***	***Z7***	***Z7***	***Z7***	***Z7***	***Z7***
Matanzas	***Z4***	***Z4***	***Z5***	***Z5***		***Z5***	***Z4***	***Z5***	***Z6***	***Z6***	***Z7***	***Z7***	***Z7***	***Z7***	***Z7***
Villa Clara	***Z5***	***Z5***	***Z6***	***Z6***	***Z5***		***Z3***	***Z3***	***Z4***	***Z5***	***Z6***	***Z6***	***Z6***	***Z6***	***Z7***
Cienfuegos	***Z5***	***Z5***	***Z6***	***Z5***	***Z4***	***Z3***		***Z4***	***Z5***	***Z5***	***Z6***	***Z6***	***Z6***	***Z7***	***Z7***
S. Spíritus	***Z6***	***Z6***	***Z6***	***Z6***	***Z5***	***Z3***	***Z4***		***Z3***	***Z5***	***Z5***	***Z6***	***Z6***	***Z6***	***Z6***
C. de Ávila	***Z6***	***Z6***	***Z6***	***Z6***	***Z6***	***Z4***	***Z5***	***Z3***		***Z4***	***Z5***	***Z5***	***Z5***	***Z6***	***Z6***
Camagüey	***Z6***	***Z6***	***Z7***	***Z6***	***Z6***	***Z5***	***Z5***	***Z5***	***Z4***		***Z4***	***Z5***	***Z5***	***Z5***	***Z6***
Las Tunas	***Z7***	***Z7***	***Z7***	***Z7***	***Z7***	***Z6***	***Z6***	***Z5***	***Z5***	***Z4***		***Z3***	***Z3***	***Z4***	***Z5***
Holguín	***Z7***	***Z7***	***Z7***	***Z7***	***Z7***	***Z6***	***Z6***	***Z6***	***Z5***	***Z5***	***Z3***		***Z3***	***Z4***	***Z4***
Granma	***Z7***	***Z7***	***Z7***	***Z7***	***Z7***	***Z6***	***Z6***	***Z6***	***Z5***	***Z5***	***Z3***	***Z3***		***Z4***	***Z4***
S. de Cuba	***Z7***	***Z7***	***Z7***	***Z7***	***Z7***	***Z6***	***Z7***	***Z6***	***Z6***	***Z5***	***Z4***	***Z4***	***Z4***		***Z3***
Guantánamo	***Z7***	***Z7***	***Z7***	***Z7***	***Z7***	***Z7***	***Z7***	***Z6***	***Z6***	***Z6***	***Z5***	***Z4***	***Z4***	***Z3***	

Tabla 1 Distribución de Zonas y Áreas

Distancias aéreas por zonas:

Z1: 6 – 24 km

Z2: 25 – 48 km

Z3: 49 – 80 km

Z4: 81 – 160 km

Z5: 161 – 280 km

Z6: 281 – 516 km

Z7: más de 516 km

1.3.2.2 Tarifas Telefónicas

En el proceso de facturación intervienen las tarifas especiales, que son aquellas que tienen asociado fechas específicas, entiéndase por fechas especiales: día de las madres, días feriados, entre otros.

Las tarifas normales son las tarifas utilizadas diariamente.

Las tarifas están compuestas por una hora de inicio, hora de fin y un costo, que representa el valor monetario que se debe abonar por la realización de la llamada telefónica en un determinado tiempo.

1.3.2.3 Formas de Pago

Post pago: Se cobra una vez realizada la llamada.

Ejemplo: cobro 1 pesos cada 4 minutos, una llamada que tenga una duración de 3 minutos abona 0 pesos y una llamada con duración 9 minutos abona 2 pesos.

Prepago: Se cobra antes de realizar la llamada.

Ejemplo: cobro 1 pesos cada 4 minutos, una llamada que tenga duración 3 minutos abona 1 peso y una llamada que tenga duración 9 minutos abona 3 pesos.

1.3.2.3 Descripción del Proceso de Facturación

Para realizar el proceso de facturación se extrae de la llamada, el número marcado, la fecha en que se realizó la misma, la hora de inicio de la llamada, y la duración. A partir del número marcado se determina el código de teleselección y una vez obtenido este se determina el área donde se ubica la llamada. Con el área se determina la zona donde se agrupa la misma y el impuesto que se le aplica al área por establecer la conexión. Una vez determinada la zona, se extraen todas las tarifas asociadas a esta zona. Con el listado de tarifas obtenidas y la fecha en que se realizó la llamada se determina si esta pertenece a una tarifa especial o a una tarifa normal. Utilizando la hora en que se realizó la llamada y el rango de horarios asociados a las tarifas se determina el costo que se debe tener en cuenta para facturar la llamada. Con la forma de cobro, la duración de la llamada, el costo y el impuesto aplicado al área se determina el valor monetario que se debe abonar por dicha llamada.

$$\text{Valor Monetario} = \text{Impuesto} + \begin{cases} \text{Post pago} \\ \\ \text{Prepago} \end{cases} * \text{Costo}$$

1.4 Análisis de otros Sistemas

- **Contaduría de Llamadas (Call Accounting Mate) 2.6.1.98**

Call Accounting Mate es un software de contaduría con capacidad industrial, rápido y confiable que puede ser usado en instituciones como oficinas, hospitales, universidades y organizaciones que necesitan definir los costos de telecomunicaciones específicos de individuos, departamentos o centros de costo. También puede ser usado para monitorear los costos de teléfono y productividad de todos y cada uno de los empleados de la compañía. (1)

- **Diario telefónico PbxTools (PbxTools PhoneJournal) 1.6.1006**

PbxTools PhoneJournal (Diario telefónico) es un software para contabilidad de llamadas diseñado para oficinas pequeñas y medianas que le puede ayudar a reducir la factura telefónica y darle información en tiempo-real acerca de las llamadas entrantes y salientes en la empresa. La aplicación captura los datos de las llamadas desde la unidad PBX y los almacena en una base de datos. Esta información se usa para generar informes que apuntan a las llamadas más caras y su distribución por hora, usuario, extensión, partida, etc. (2)

1.5 Selección de la Metodología de Desarrollo

El impacto de elegir la mejor metodología de desarrollo de software para el desarrollo de un proyecto, es trascendental para la obtención de un producto con calidad.

El objetivo es desarrollar software rápidamente, obtener resultados periódicamente en intervalos de tiempo relativamente cortos, respondiendo a los cambios que pueden surgir durante el proyecto, y considerando además las prácticas esenciales para asegurar la calidad del producto.

La adopción de una metodología de desarrollo de software debe estar condicionada por:

- La experiencia del grupo de desarrollo en el uso de las mismas.
- La magnitud del proyecto.
- La estabilidad del personal que integra el equipo de desarrollo.

- La comunicación con el cliente.
- La complejidad del proyecto.
- La urgencia del producto.

1.5.1 Análisis Cualitativo y Cuantitativo del Entorno del Proyecto SGIC-PABX

Características del equipo de desarrollo

Característica	Valoración
Composición	Heterogénea (17 personas): • 35,3% hombres, 64,7% mujeres • 88.2% estudiantes, 11,8% profesores • 47% 3ro, 11.8% 4to, 29.4% 5to y 11.8% profesores
Método de trabajo	(planificación rígida, planificación adaptativa) • planificación rígida
Carga de trabajo	Ambas categorías tienen doble vinculación docente-productiva. (baja, adecuada, exigente, sobrecargada) • Estudiante: Exigente • Profesor: Exigente
Madurez	(baja, media, alta) • Baja
Experiencia productiva	(baja, media, alta) • Media
Conocimiento de Metodologías Tradicionales	(bajo, medio, avanzado) • 3er Bajo • 4to Medio • 5to Medio • Profesor Medio
Conocimiento de Metodologías Agiles	(bajo, medio, avanzado) • 3er Bajo • 4to Bajo • 5to Bajo • Profesor Bajo
Conocimiento de Lenguaje de Programación Java.	(bajo, medio, avanzado) • 3er Bajo • 4to Medio • 5to Medio • Profesor Avanzado
Conocimiento de Lenguaje de Programación Python.	(bajo, medio, avanzado) • 3er Bajo • 4to Bajo • 5to Bajo • Profesor Medio

Tabla 2. Características del equipo de desarrollo del proyecto.

Características del Cliente

Característica	Valoración
Cliente	Empresa ETECSA con un representante local en la UCI.
Relación con el cliente	Encuentros planificados.
Disponibilidad de tiempo	Limitada; presencia, atención y chequeo es compartido.
Estado (organización)	(estable, inestable) • Estable
Retroalimentación cliente-equipo desarrollo	(continua, discontinua) • discontinua

Tabla 3. Características del Cliente.

Características del proyecto

El proyecto está concebido como un proyecto grande con solución a largo plazo. Se considera complejo por la intervención de términos un poco técnicos y fuera del alcance de los integrantes del equipo de desarrollo, como son el funcionamiento de las pizarras telefónicas y registro SMDR. En cuanto a los requisitos y otros cambios ocurridos desde la concepción del proyecto hasta hoy se considera que ha sido volátil.

1.5.1.1 Características de las Metodologías Ágiles

Las metodologías ágiles surgen como una extensión a las metodologías tradicionales para mejorar el desarrollo de sistemas, según el tipo de proyecto y empresa. A continuación se mencionan algunas de ellas:

- Xtreme Programing, por sus siglas en ingles (XP).
- Scrum
- MSF
- Cristal

De manera general, las metodologías ágiles pueden explicarse a través de los siguientes 4 principios generales:

- Al individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas. Dado que el proceso de desarrollo es creativo, no es posible pensar que las personas trabajen respondiendo a órdenes o procesos rígidos.
- Desarrollar software que funcione es más importante que conseguir una buena documentación. Puesto que si el software no funciona la documentación no vale de nada. A nivel interno puede haber documentación, pero solo la necesaria y a nivel externo lo que el cliente requiera.

- La colaboración con el cliente más que la negociación de un contrato. Supone que la satisfacción del cliente con el producto será mayor, mientras exista una conversación y realimentación continua entre este y la empresa.
- Responder a los cambios más que seguir estrictamente un plan. Puesto que si un proyecto de software no es capaz de adaptarse a los cambios fracasará, especialmente en productos de gran envergadura. La estrategia de planificación se basa en: planes detallados para las próximas semanas, planes aproximados para los próximos meses y muy generales para plazos mayores.

1.5.1.2 RUP

Luego de un amplio proceso de selección de la metodología a usar, para elaborar este sistema se decidió utilizar Rational Unified Process (RUP). RUP es una propuesta para el desarrollo de software basada en el desarrollo iterativo y el modelamiento visual, usando UML como lenguaje de modelado para describir el sistema. Esto permite incorporar al proceso de desarrollo de software un mejor control de los requerimientos. Posibilita además la distribución del trabajo en diversos frentes de forma simultánea y proporciona el modo en que el equipo de proyecto puede trabajar de una forma más conjunta con los clientes; lo que favorece a una mayor organización y entendimiento de lo que realmente el cliente necesita.

Características de RUP

- **Dirigido por Casos de Uso.**

Los casos de uso reflejan lo que los usuarios futuros necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requerimientos. Los casos de uso guían el proceso de desarrollo ya que los modelos que se obtienen, como resultado de los diferentes flujos de trabajo, representan la realización de estos. (3)

- **Centrado en la arquitectura.**

La arquitectura en un sistema software se describe mediante diferentes vistas del sistema en construcción. Este concepto incluye los aspectos estáticos y dinámicos más significativos del sistema. Esta se refleja en los casos de uso pues cada producto tiene tanto una función como una forma, ninguna es suficiente por sí sola. (3)

- **Iterativo e incremental.**

Resulta práctico dividir el trabajo en partes más pequeñas o mini-proyectos, los cuales no son más que iteraciones que resultan en un incremento. Una iteración es una secuencia de actividades con un plan establecido y criterios de evaluación, cuyo resultado es una versión del software. Entre los beneficios de la iteración se encuentran: reducir el coste del riesgo al coste de un solo incremento, y lograr menos riesgo de sacar el producto al mercado en una fecha que acelere el ritmo de desarrollo. Las necesidades del usuario y correspondientes requisitos no puedan definirse completamente al principio, se requieren iteraciones sucesivas. (3)

1.5.1.3 Selección de la metodología

La decisión de emplear una Metodología Ágil para el desarrollo del proyecto se rechaza porque:

- La experiencia de trabajo con las Metodologías Ágiles tiene un nivel bajo para los integrantes del equipo.
- El equipo de desarrollo por su heterogeneidad y el poco tiempo de creado que tiene, se considera inmaduro y con poca experiencia productiva.
- Por las responsabilidades del cliente, el mismo no podrá formar parte del equipo de desarrollo.

Por tanto, se decide escoger RUP porque las características del equipo de desarrollo recogidas en la *Tabla 2*, las características del cliente en la *Tabla 3* y las características del proyecto expuestas anteriormente así lo ratifican.

1.6 Lenguaje de Programación

1.6.1 C++

C++ es un lenguaje imperativo orientado a objetos derivado del C. En realidad es un superconjunto de C, que nació para añadirle cualidades y características de las que carecía. El resultado es que como su ancestro, sigue muy ligado al hardware subyacente, manteniendo una considerable potencia para programación a bajo nivel, pero se la han añadido elementos que le permiten también un estilo de programación con alto nivel de abstracción.

No es un lenguaje orientado a objetos puro (en el sentido en que puede serlo Java por ejemplo), además no nació como un ejercicio académico de diseño. Se trata simplemente del sucesor de un lenguaje de programación hecho por programadores (de alto nivel) para programadores, lo que se traduce en un diseño pragmático al que se le han ido añadiendo todos los elementos que la práctica aconsejaba como necesarios, con independencia de su belleza o purismo conceptual. Estos condicionantes tienen su cara y su cruz; en ocasiones son motivos de ciertos "reproches" por parte de

sus detractores, en otras, estas características son precisamente una cualidad. De hecho, en el diseño de la Librería Estándar C++ se ha usado ampliamente esta dualidad (ser mezcla de un lenguaje tradicional con elementos de POO), lo que ha permitido un modelo muy avanzado de programación extraordinariamente flexible (programación genérica).

Desde luego, C++ es un lenguaje de programación extremadamente largo y complejo. A pesar de todo, ha experimentado un extraordinario éxito desde su creación. De hecho, muchos sistemas operativos, compiladores e intérpretes han sido escritos en C++ (el propio Windows y Java). Una de las razones de su éxito es ser un lenguaje de propósito general que se adapta a múltiples situaciones.

Tanto sus fervientes defensores como sus acérrimos detractores han hecho correr ríos de tinta ensalzando sus cualidades o subrayando sus miserias, aunque todo el mundo parece estar de acuerdo en que es largo y complejo. Ha servido de justificación para el diseño de otros lenguajes que intentan eliminar sus inconvenientes al tiempo que mantener sus virtudes (C# y Java por ejemplo). (4)

1.6.2 Java

El lenguaje de programación Java, fue diseñado por la compañía Sun Microsystems Inc., con el propósito de crear un lenguaje que pudiera funcionar en redes computacionales heterogéneas, y que fuera independiente de la plataforma en la que se vaya a ejecutar. Esto significa que un programa de Java puede ejecutarse en cualquier máquina o plataforma. El lenguaje fue diseñado teniendo en cuenta las siguientes características: (5)

- **Simple**: elimina la complejidad de los lenguajes como "C" y da paso al contexto de los lenguajes modernos orientados a objetos. La filosofía de programación orientada a objetos es diferente de la programación convencional.
- **Robusto**: maneja la memoria de la computadora. No hay necesidad de preocuparse por apuntadores, memoria que no se esté utilizando, etc. Java realiza todo esto sin necesidad de que el programador se lo indique.
- **Seguro**: tiene ciertas políticas que evitan se puedan codificar virus con este lenguaje. Existen muchas restricciones, especialmente para los applets, que limitan lo que se puede y no puede hacer con los recursos críticos de una computadora.
- **Portable**: como el código compilado de Java (conocido como byte code) es interpretado, un programa compilado de Java puede ser utilizado por cualquier computadora que tenga implementado el intérprete de Java.

- **Independiente a la arquitectura**: al compilar un programa en Java, el código resultante es un tipo de código binario. Este código es interpretado por diferentes computadoras de igual manera, solamente hay que implementar un intérprete para cada plataforma. De esa manera Java logra ser un lenguaje que no depende de una arquitectura computacional definida.
- **Dinámico**: no requiere que se compilen todas las clases de un programa para que este funcione. Al realizar una modificación a una clase, Java se encarga de realizar un Dynamic Bynding o un Dynamic Loading para encontrar las clases.

1.6.3 Python

Python es un lenguaje simple. Ofrece mucha más comprobación de errores que C y, al ser un lenguaje de muy alto nivel, tiene incluidos tipos de datos de alto nivel, como matrices flexibles y diccionarios, que llevarían días de programación en C. Dados sus tipos de datos más generales, se puede aplicar a un rango de problemas más amplio que Awk o incluso Perl, pero muchas cosas son, al menos, igual de fáciles en Python que en esos lenguajes. Python te permite dividir su programa en módulos reutilizables desde otros programas Python. Viene con una gran colección de módulos estándar que se pueden utilizar como base para los programas o como ejemplos para empezar a aprender Python. (6)

Características

- **Multiplataforma**: hay versiones disponibles de Python en muchos sistemas informáticos distintos. Originalmente se desarrolló para Unix, aunque cualquier sistema es compatible con el lenguaje siempre y cuando exista un intérprete programado para él.
- **Interpretado**: quiere decir que no se debe compilar el código antes de su ejecución. En realidad sí que se realiza una compilación, pero esta se realiza de manera transparente para el programador. En ciertos casos, cuando se ejecuta por primera vez un código, se producen unos bytecodes que se guardan en el sistema y que sirven para acelerar la compilación implícita que realiza el intérprete cada vez que se ejecuta el mismo código.
- **Interactivo:** python dispone de un intérprete por línea de comandos en el que se pueden introducir sentencias. Cada sentencia se ejecuta y produce un resultado visible, que puede ayudarnos a entender mejor el lenguaje y probar los resultados de la ejecución de porciones de código rápidamente.
- **Orientado a Objetos**: la programación orientada a objetos está soportada en Python y ofrece en muchos casos una manera sencilla de crear programas con componentes reutilizables.

- **Funciones y librerías:** dispone de muchas funciones incorporadas en el propio lenguaje, para el tratamiento de strings, números, archivos, etc. Además, existen muchas librerías que podemos importar en los programas para tratar temas específicos como la programación de ventanas o sistemas en red o cosas tan interesantes como crear archivos comprimidos en .zip.
- **Sintaxis clara:** por último, destacar que Python tiene una sintaxis muy visual, gracias a una notación identada (con márgenes) de obligado cumplimiento. En muchos lenguajes, para separar porciones de código, se utilizan elementos como las llaves o las palabras clave begin y end. Para separar las porciones de código en Python se debe tabular hacia dentro, colocando un margen al código que iría dentro de una función o un bucle. Esto ayuda a que todos los programadores adopten unas mismas notaciones y que los programas de cualquier persona tengan un aspecto muy similar.

1.6.4 Selección del Lenguaje de Programación

Los lenguajes de programación mencionados anteriormente al igual que otros, permiten la implementación del Módulo Facturación del Sistema para la Gestión Integral de Costos de llamadas telefónicas en pizarras telefónicas. Decidir cuál es el lenguaje ideal para desarrollar una aplicación no es una tarea fácil.

Para el desarrollo de este sistema de decidió utilizar el lenguaje de programación Java porque es una tecnología robusta, flexible y probada. Consta con una amplia documentación y una comunidad especializada de millones de desarrolladores. Es un lenguaje que ha sido ampliamente probado obteniéndose buenos resultados.. Además, el equipo de desarrollo del proyecto posee un mayor conocimiento de este lenguaje para programar.

1.7 Herramienta Case

1.7.1 Visual Paradigm

Visual Paradigm para UML 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. El software de modelado UML ayuda a una rápida construcción de aplicaciones de calidad, mejores y a un menor costo. Permite dibujar todos los tipos de diagramas de clases, ingeniería inversa, generar código desde diagramas y generar documentación. (7)

Dentro de sus características fundamentales están:

- **Multiplataforma:** Soportada en plataformas Java para Sistemas Operativos Windows, Linux y Mac OS X.
- **Interoperabilidad:** Intercambia diagramas UML y modelos con otras herramientas. Soporta exportar e importar a XMI19, XML y archivos Excel.
- **Generación de Documentación:** Comparte y genera los diagramas y diseños en formatos como PDF20, HTML y Microsoft Word.
- **Editor de Detalles de Casos de Uso:** Entorno todo en uno para la especificación de los detalles de los casos de uso, incluyendo la especificación del modelo general y de las descripciones de los casos de uso.
- **Ingeniería de Código:** Permite generación de código e ingeniería inversa en lenguajes como **Java**, C++, CORBA, IDL, PHP, XML Schema, Ada, Python, C#, VB .NET, ODL, Flash ActionScript, Delphi, Perl y Rugby.
- **Integración con Entornos de Desarrollo:** Apoyo al ciclo de vida completo de desarrollo del software: análisis, diseño e implementación, en IDE como Eclipse, Microsoft Visual Studio, **NetBeans**, Sun ONE, Oracle JDeveloper, JBuilder y otros.

1.7.2 Selección de la Herramienta Case a utilizar

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). Además, la UCI posee licencia para esta herramienta.

1.8 Gestor de Base de Datos

1.8.1 PostgreSQL

PostgreSQL 8.3 es un gestor de bases de datos relacional, libre y gratuito. Está liberado bajo la licencia BSD, lo que significa que se puede disponer de su código fuente, modificarlo a voluntad y redistribuirlo libremente. Características generales de PostgreSQL: (8)

- **Objeto-Relacional**: PostgreSQL aproxima los datos a un modelo objeto-relacional, y 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 arrays (arreglos).
- **Altamente Extensible**: Soporta operadores, funciones, métodos de acceso y brinda la posibilidad de que los usuarios puedan crear sus propios tipos de datos.
- **Integridad Referencial**: Soporta integridad referencial, la cuál es utilizada para garantizar la validez de los datos de la base de datos.
- **API Flexible**: La flexibilidad del API de PostgreSQL ha permitido proporcionar soporte al desarrollo fácilmente para el RDBMS PostgreSQL. Estas interfaces incluyen Object Pascal, Python, Perl, PHP, Java/JDBC, C/C++, y Pike.
- **Lenguajes Procedurales**: PostgreSQL tiene soporte para lenguajes procedurales internos, incluyendo un lenguaje nativo denominado PL/pgSQL. Este lenguaje es comparable al lenguaje procedural de Oracle, PL/SQL.
- **Multi-Version Concurrency Control MVCC** (Control de Concurrencia Multi-versión): MVCC es la tecnología que PostgreSQL utiliza para evitar bloqueos innecesarios. MVCC posibilita que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos, ya que mantiene una ruta para todas las transacciones realizadas por los usuarios de la base de datos. Es capaz entonces de manejar los registros sin necesidad de que estén disponibles. Esta estrategia es superior al uso de bloqueos por tabla o por filas comunes en otras bases de datos, eliminando la necesidad del uso de bloqueos explícitos.
- **Write Ahead Logging (WAL)**: La característica de PostgreSQL conocida como Write Ahead Logging incrementa la dependencia de la base de datos al registro de cambios antes de que estos sean escritos en la base de datos. Esto garantiza que en el hipotético caso de que la base de datos tenga fallos, existirá un registro de las transacciones a partir del cual podremos restaurar la base de datos. Esto puede ser enormemente beneficioso, ya que cualquier cambio que no fue escrito en la base de datos pueda ser recuperado usando el dato que fue previamente registrado. Una vez que el sistema ha quedado restaurado, un usuario puede continuar trabajando desde el punto en que lo dejó cuando falló la base de datos.

1.8.2 MySQL

MySQL es un gestor de base de datos sencillo de usar e increíblemente rápido. Es gratis para aplicaciones no comerciales. Las principales características de MySQL: (9)

- Aprovecha la potencia de sistemas multiprocesadores, gracias a su implementación multihilo.
- Soporta gran cantidad de tipos de datos para las columnas.
- Dispone de API's en gran cantidad de lenguajes (C, C++, Java, PHP).
- Gran portabilidad entre sistemas.
- Soporta hasta 32 índices por tabla.
- Gestión de usuarios y passwords, manteniendo un muy buen nivel de seguridad en los datos.

1.8.3 Selección de Gestor de Base de Datos

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 Entorno de Desarrollo Integrado

1.9.1 NetBeans

Es una plataforma para el desarrollo de aplicaciones de escritorio usando el lenguaje Java y un entorno de desarrollo integrado (*IDE*) para desarrollar sobre esta plataforma. La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. (10)

NetBeans IDE 6.1 provee varias nuevas características y mejoras, como funciones de edición enriquecida de JavaScript, soporte para el uso del framework web Spring, y mejor integración con

MySQL. Esta versión también provee mejor rendimiento, especialmente un arranque más rápido (más de 40%), menor consumo de memoria y mejor respuesta cuando se trabajan con proyectos grandes.

Ventajas:

- **Código abierto, patrocinado por Sun:** NetBeans ofrece las ventajas de que Sun Microsystems lo ha creado, respaldado y ha abierto su código.
- **La integración de múltiples herramientas y protocolos proporciona razones para la migración:** las amplias posibilidades de desarrollo multiplataforma de NetBeans atraen a muchos desarrolladores que han trabajado con otras herramientas.
- **Facilidad de uso durante todo el ciclo de Desarrollo:** La facilidad de uso de NetBeans también resulta llamativa, ya que no es necesario ir más allá del portal netbeans.org, para buscar las características adicionales y los "plugins' de los paquetes de funciones que abarcan una diversidad de requerimientos — desde C/C++ hasta movilidad y la Web. Una vez que se descarga, NetBeans es fácil de utilizar, gracias a su interfaz de usuario y a funciones.
- **NetBeans maneja la complejidad de la arquitectura orientada al servicio (SOA):** La automatización de los requerimientos de diseño demuestra ser particularmente importante en el diseño de aplicaciones para una SOA, donde los desarrolladores trabajan generalmente con múltiples tecnologías y protocolos.
- **Promoviendo las mejores prácticas para la productividad del grupo:** NetBeans elimina la necesidad de los equipos de desarrollo que tienen que invertir demasiado tiempo manteniendo los modelos actualizados con revisiones exhaustivas que garanticen la actualización de indicadores y códigos de anotación.
- **Creado para acelerar el Desarrollo:** Permitir que los desarrolladores se concentren en la lógica comercial durante todo el ciclo de desarrollo de la aplicación es también una reflexión de cómo el software debe satisfacer los requerimientos del mercado, en la actualidad.

1.9.2 Eclipse

La plataforma Eclipse consiste en un Entorno de Desarrollo Integrado (IDE, Integrated Development Environment) abierto y extensible. Un IDE es un programa compuesto por un conjunto de herramientas útiles para un desarrollador de software. Como elementos básicos, un IDE cuenta con en un editor de código, un compilador/intérprete y un depurador. Eclipse sirve como IDE Java y cuenta con numerosas herramientas de desarrollo de software. También da soporte a otros lenguajes de programación, como

son C/C++, Cobol, Fortran, PHP o Python. A la plataforma base de Eclipse se le pueden añadir extensiones (plugins) para extender la funcionalidad.

Eclipse es también una comunidad de usuarios, extendiendo constantemente las áreas de aplicación cubiertas. Un ejemplo es el recientemente creado Eclipse Modeling Project, cubriendo casi todas las áreas de Model Driven Engineering.

Eclipse fue desarrollado originalmente por IBM como el sucesor de su familia de herramientas para VisualAge. Eclipse es ahora desarrollado por la Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios. (11)

1.9.3 Selección del IDE de Desarrollo

Después de conocer las características de los IDEs de desarrollo mencionados anteriormente, se decidió utilizar el Netbeans IDE para el desarrollo de la aplicación, pues es un producto libre y gratuito sin restricciones de uso. Soporta el lenguaje java que escogimos con anterioridad. Además, la plataforma Netbeans es una base modular y extensible usada como una estructura de integración para crear aplicaciones de escritorio grandes, permitiéndole al desarrollador enfocarse en la lógica específica de su aplicación. Consta con gran éxito, una gran base de usuarios y una comunidad en constante crecimiento.

1.10 Selección de Frameworks

Un framework, constituye una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio. Para el desarrollo del Módulo Facturación de SGIC-PBX se llevó a cabo una selección de Frameworks que brindan soporte a cada una de las capas componentes de la arquitectura, de las cuales se expondrá una breve descripción.

1.10.1 Spring

Spring es un framework que ha venido a revolucionar la manera de programar aplicaciones Java debido a la facilidad de crear componentes reutilizables, además de integrarse fácilmente con otros Frameworks como lo son Hibernate, iBatis, Struts, entre otros; formando una poderosa herramienta para el desarrollo de aplicaciones empresariales. Spring constituye además un framework de código abierto que interviene en todas las capas arquitectónicas de una aplicación. A pesar de que Spring no obliga a usar un modelo de programación en particular, se ha popularizado en la comunidad de

programadores en Java al considerársele una alternativa y un sustituto del modelo de Enterprise JavaBean. Por su diseño el framework ofrece mucha libertad a los desarrolladores en Java y soluciones muy bien documentadas y fáciles de usar para las prácticas comunes en la industria. (12)

Dentro de las ventajas que ofrece Spring, se encuentra:

- Facilita la manipulación de los objetos.
- Reduce la proliferación de *Singletons.*
- Elimina la necesidad de usar distintos y variados tipos de ficheros de configuración.
- Permite el uso de la programación orientada a aspectos.

1.10.2 Hibernate

Solución ORM (Object-Relational Mapping) para Java, constituye el framework utilizado en el desarrollo de la capa de acceso a datos del Modulo de Facturación de SGIC - PBX, se selecciona a partir de sus características y ventajas.

Hibernate 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. De una manera muy rápida y optimizada puede generar bases de datos en cualquiera de los entornos soportados: Oracle, DB2, MySQL; permitiendo la abstracción de la base de datos utilizada, lo que posibilita la migración del gestor libremente y sin mayores problemas para el equipo de desarrollo.

Hibernate tiene como objetivo fundamental solucionar el problema de la diferencia entre los dos modelos de datos coexistentes en una aplicación: el orientado a objetos y el relacional. Para lograr esto le permite al desarrollador detallar cómo es su modelo de datos, qué relaciones existen y qué forma tienen. Con esta información Hibernate hace posible a la aplicación manipular los datos de la base operando sobre objetos, con todas las características de la programación orientada a objetos. Hibernate 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 todos los motores de bases de datos. Está diseñado para ser flexible en cuanto al esquema de tablas utilizado, para poder adaptarse a su uso sobre una base de datos ya existente. También tiene la funcionalidad de crear la base de datos a partir de la información disponible. Asimismo 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"). Hibernate además es Open Source y la licencia del producto está eximida de costo. (13)

1.11 Conclusiones

Se analizaron las principales tendencias acerca de las tecnologías y herramientas utilizadas para el desarrollo de aplicaciones, teniendo en cuenta que la aplicación a desarrollar es de escritorio se seleccionó como lenguaje de programación Java con la integración de los frameworks Spring e Hibernate ganando en velocidad de desarrollo. Se propone la utilización del Entorno de Desarrollo Netbeans, facilitando el trabajo a los desarrolladores y como herramienta CASE Visual Paradigm para estandarizar los procesos de desarrollo del sistema. Todo este proceso será orientado por la metodología RUP, la cuál constituye una guía de cómo se debe desarrollar el software.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

2.1 Introducción

En este capítulo se representa la descripción de cada una de las actividades que se llevan a cabo para obtener los elementos que intervienen en la facturación de las llamadas telefónicas, para de esta manera lograr un mejor entendimiento de cómo funciona el proceso de facturación.

Además, se presenta la propuesta del Sistema de Facturación de Llamadas Telefónicas. Se describen los requisitos funcionales y no funcionales del sistema. Se identifican actores y casos de uso (CU), obteniendo como resultado el Modelo de Casos de Uso del Sistema donde se describen detalladamente todos los CU.

2.2 Solución Propuesta del Sistema de Facturación de SGIC – PBX

Una vez que se ejecuta todo el proceso de interpretación del SMDR y se registran los datos que detallan la llamada realizada en la base de datos, se propone diseñar un sistema que lleve a cabo todo el proceso de facturación de las llamadas externas salientes, realizar un análisis del presupuesto planificado y una vez sobrepasado el saldo asignado, notificar y bloquear la extensión o código de cuenta verificable al que estará asociada dicha llamada. Le permitirá a los usuarios autenticados como administrador de la PABX, mantener el control del consumo por la realización de las llamadas telefónicas, gestionar las tarifas telefónicas dando la opción de registrar una nueva tarifa, modificarla, eliminarla o mostrar detalles de alguna de ellas, gestionar fechas especiales, gestionar áreas y gestionar las formas de cobro.

2.2.1 Actividades Asociadas al Proceso de Facturación

2.2.1.1 Identificación de zonas y áreas

Número Marcado: 023572055

Código de Teleselección	Zona	Bit de Inicio	Longitud Código	Impuesto	
022	Zona 7	0	3	0.5	← **023 No**
043	Zona 4	0	3	0.2	← **023 No**
031	Zona 6	0	3	0.4	← **023 No**
023	Zona 7	0	3	0.5	← **023 Zona 7**

De la llamada telefónica se extrae el número marcado. De la base de datos se cargan las áreas y las zonas, creándose una relación entre ellas. De las áreas se toma el bit de inicio y la longitud del código de teleselección y se busca en el número marcado. Una vez obtenido el código de teleselección se hace coincidir con cada unos de los números de las áreas cargadas hasta determinar el área al que pertenece dicho código. Conociendo el área se obtiene la zona asociada a la misma.

2.2.1.2 Identificación de las Tarifas Telefónicas

Fecha: 21/05/2010 Hora: 14:00

Fechas Especiales
01/05/2010
08/05/2010
09/05/2010

Si **No**

Tarifa Especial

Tipo	Hora Inicio	Hora Fin	Costo
Especial	06:00	18:00	0.50
Especial	18:00	06:00	0.25

Tarifa Normal

Tipo	Hora Inicio	Hora Fin	Costo
Normal	06:00	18:00	1.00
Normal	18:00	06:00	0.50

14:00 = 0.50 **14:00 = 1.00**

Para este ejemplo se devuelve **$ 1.00.**

De la llamada se extrae la fecha en que fue realizada y la hora. Conociendo la zona se extraen las tarifas normales y especiales. Se carga el listado de fechas especiales y se hace coincidir la fecha de la llamada con las fechas registradas como especiales. En caso de que la fecha de llamada se encuentre entre las fechas especiales, se cargan las tarifas especiales, de lo contrario se cargan las

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

2.2.1.2 Facturación de Llamadas Telefónicas

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

Anteriormente fueron explicados los procesos a seguir para obtener los elementos necesarios para calcular finalmente el valor monetario de la llamada. A partir de ahí, el costo de la llamada se obtiene sumando el impuesto establecido por la conexión telefónica con la forma de pago, que puede se postpago o prepago y se multiplica con la duración de la llamada telefónica.

2.3 Modelo de Dominio

Un modelo de dominio 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.

Se decide realizar un modelo de dominio debido a que en el sistema que se desea implementar no se identifican de forma clara los actores, ni los trabajadores, ni los procesos del negocio. Este modelo permitirá mostrar de forma visual los elementos fundamentales que se manejan en el Módulo Facturación del Sistema para la Gestión Integral de Costos de Llamadas SGIC-PABX.

2.3.1 Diagrama de Clases del Dominio

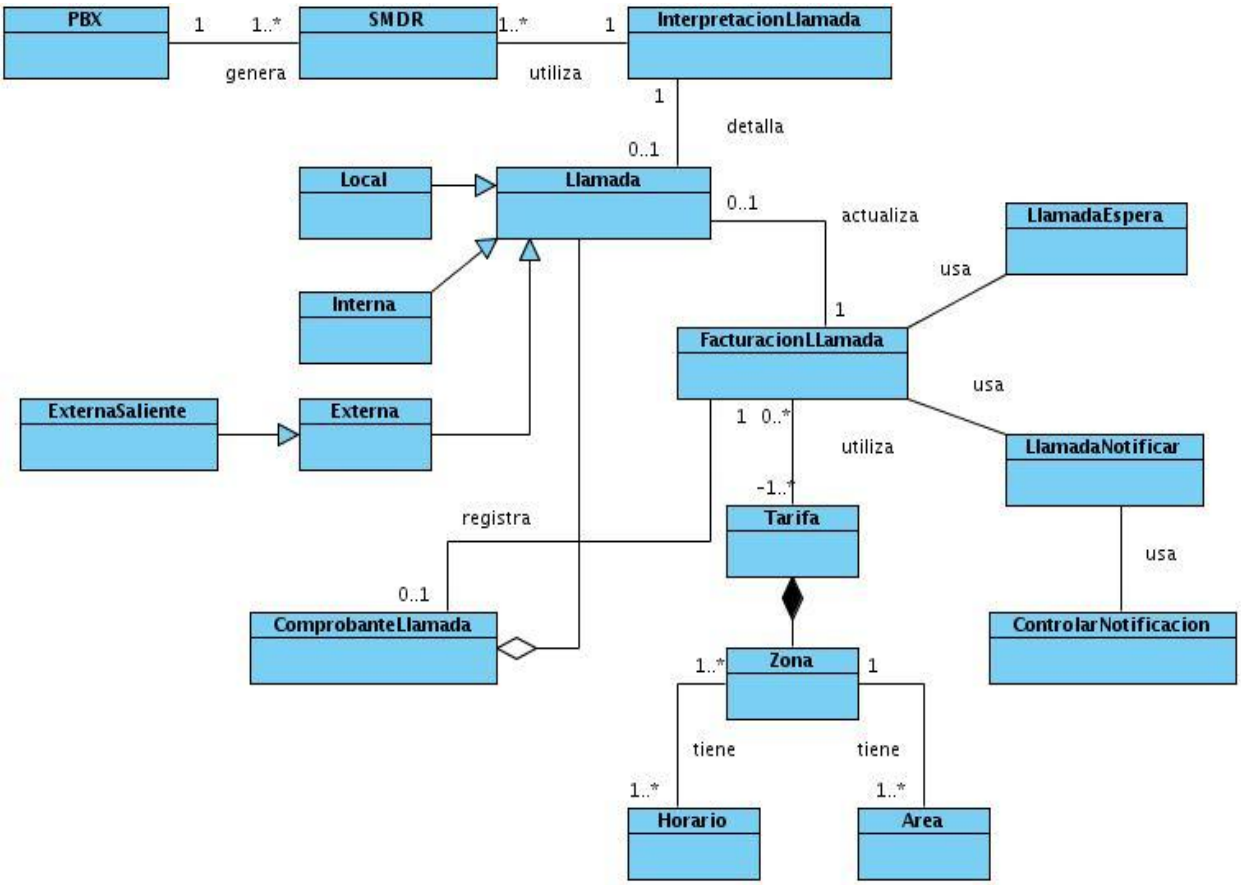


Figura 1. Modelo del Dominio

2.3.2 Descripción de los Conceptos del Dominio

Descripción de la clase: PBX

Pizarra Telefónica.

Descripción de la clase: SMDR

Registro generado por la PBX, que detalla las llamadas telefónicas.

Descripción de la clase: Interpretación Llamada

Proceso que se lleva a cabo para obtener los detalles de las llamadas.

Descripción de la clase: Llamada

Es la comunicación que se establece entre el abonado y el teléfono al cual quiere comunicarse.

Descripción de la clase: Llamada Interna

Llamadas efectuadas dentro de la empresa.

Descripción de la clase: Llamada Externa

Generalización de las llamadas efectuadas desde o hacia la empresa.

Descripción de la clase: Externa Saliente

Llamadas efectuadas desde la empresa hacia fuera de esta y que se factura.

Descripción de la clase: Llamada Local

Llamada efectuada dentro de la misma localidad o ciudad.

Descripción de la clase: Facturación de Llamada

Determina los costos de las llamadas.

Descripción de la clase: Tarifa

Lista que especifica los costos por tiempo de las llamadas.

Descripción de la clase: Comprobante de Llamada

Reporte generado al facturar la llamada.

Descripción de la clase: Horario

Entidad que especifica el costo de las llamadas en función de la hora.

Descripción de la clase: Área

Localidad donde se efectúa una llamada (localidad, provincia o municipio).

Descripción de la clase: Zona

Entidad que relaciona las aéreas con tarifas, según la distancia.

Descripción de la clase: Llamada Espera

Entidad que representa las llamadas externas salientes a facturar.

Descripción de la clase: Llamada Notificar

Entidad que representa las llamadas externas salientes que están asociadas a un código de cuenta verificable o extensión que ha excedido el saldo asignado y debe ser notificada.

Descripción de la clase: Controlar Notificación

Notifica cuando haya sido excedido el saldo asignado.

2.3.3 Descripción del Modelo de Dominio

La **pizarra telefónica** (PBAX) genera registros **SMDR**, estos registros detallan las llamadas telefónicas, las cuales luego del proceso de **interpretación de llamadas** son clasificadas en **Local**, **Interna** y **Externas Salientes**. A las **llamadas en espera** que son clasificadas como externas salientes, se les realiza el proceso de **facturación**. Para determinar el costo de la llamada por concepto de facturación se tienen en cuenta varios factores, como son el **área**, la **zona**, y el **horario** en que es realizada la misma. Determinado el costo de la llamada, si el saldo gastado excede el saldo asignado, esta pasa a una cola de **llamadas a notificar,** mediante el proceso de **controlar notificación** se envía un correo al responsable de la extensión o código de cuenta verificable al que está asociada dicha llamada y se procede a bloquear.

2.4 Requisitos Funcionales

La clave del éxito en el desarrollo de un software esta dada fundamentalmente en el éxito del logro de una buena comunicación entre los usuarios y el equipo del proyecto, llegando a un entendimiento de lo que hay que hacer.

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir. Ellos no alteran la funcionalidad del producto, esto quiere decir que los requerimientos funcionales se mantienen invariables sin importar con qué propiedades o cualidades se relacionen. Todas las ideas que los usuarios, clientes y miembros del equipo aporten sobre lo que debe hacer el sistema deben ser analizadas como candidatas a requisito. (14)

2.4.1 Requisitos Funcionales

RF1 Gestionar tarifa telefónica

RF1.1 Registrar tarifa telefónica.

RF1.2 Modificar tarifa telefónica.

RF1.3 Eliminar tarifa telefónica.

RF1.4 Buscar tarifa telefónica.

RF1.5 Mostrar detalles de tarifa telefónica.

RF2 Facturar llamada telefónica

RF2.1 Determinar costo de la llamada.

RF3 Gestionar Área

RF3.1 Registrar área.

RF3.2 Eliminar área.

RF3.3 Modificar área.

RF3.4 Buscar área.

RF4 Gestionar Forma de Cobro

RF 4.1 Registrar forma de cobro.

RF 4.2 Modificar forma de cobro.

RF 4.3 Eliminar forma de cobro.

RF5 Gestionar fechas especiales.

RF5.1 Registrar fecha especial.

RF5.2 Eliminar fecha especial.

RF5.3 Mostrar detalles de fecha especial.

RF6 Notificar consumo de presupuesto.

RF7 Bloquear código de cuenta verificable o extensión.

2.4.2 Requisitos no Funcionales

Los requerimientos no funcionales son las propiedades o cualidades que el producto debe tener para que sea atractivo, usable, rápido y confiable. (14)

Para lograr una buena calidad en el sistema y de esta manera satisfacer al cliente se listaron las siguientes propiedades o cualidades:

- **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.

- **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.

- **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 subsistema Comunicador.

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.

- **Requisito de Soporte**

Entregar 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.

Impartir Capacitación

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

- **Restricciones de diseño**

  Lenguaje de programación

  Lenguaje de programación Java.

- **Requisito para la documentación del usuario**

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

- **Requisito de interfaz**

Interfaces 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.

Interfaces Hardware

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

Interfaces Software

Soportar la compatibilidad con los sistemas operativos GNU/Linux y Windows, logrando interacción con los directorios y ficheros de sus sistemas de archivos.

Interfaces de comunicación

Comunicación entre la PABX y una PC: Puerto serie (RS232C) 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.

2.5 Modelo de Casos de Uso del Sistema

En este epígrafe se representan los actores que intervienen así como los casos de uso del sistema, determinando el “Diagrama de Casos de Uso del Sistema”.

Un caso de uso es una secuencia de transacciones que son desarrolladas por un sistema en respuesta a un evento que inicia un actor sobre el propio sistema. Los diagramas de casos de uso sirven para especificar la funcionalidad y el comportamiento de un sistema mediante su interacción con los usuarios y/o otros sistemas. O lo que es igual, un diagrama que muestra la relación entre los actores y los casos de uso del sistema. Los diagramas de casos de uso se utilizan para ilustrar los requerimientos del sistema al mostrar como reacciona una respuesta a eventos que se producen en el mismo.

2.5.1 Definición de los Actores del Sistema

Un actor es una idealización de una persona externa, de un proceso, o de un agente que interactúa con un sistema, un subsistema o una clase. Un actor caracteriza las interacciones que los usuarios exteriores pueden tener con el sistema. Un actor puede ser un humano, otro sistema informático o cierto proceso ejecutable.

Actor	Descripción
Administrador de PBX	Encargado de gestionar las fechas especiales, la forma de cobro,

	las tarifas especiales, y las áreas.
PBX	Actor que representa la pizarra telefónica.
Servidor de Correo	Actor que representa un servidor de correo.

Tabla 4. Definición de los Actores del Sistema

2.5.2. Diagrama de Casos de Uso del Sistema

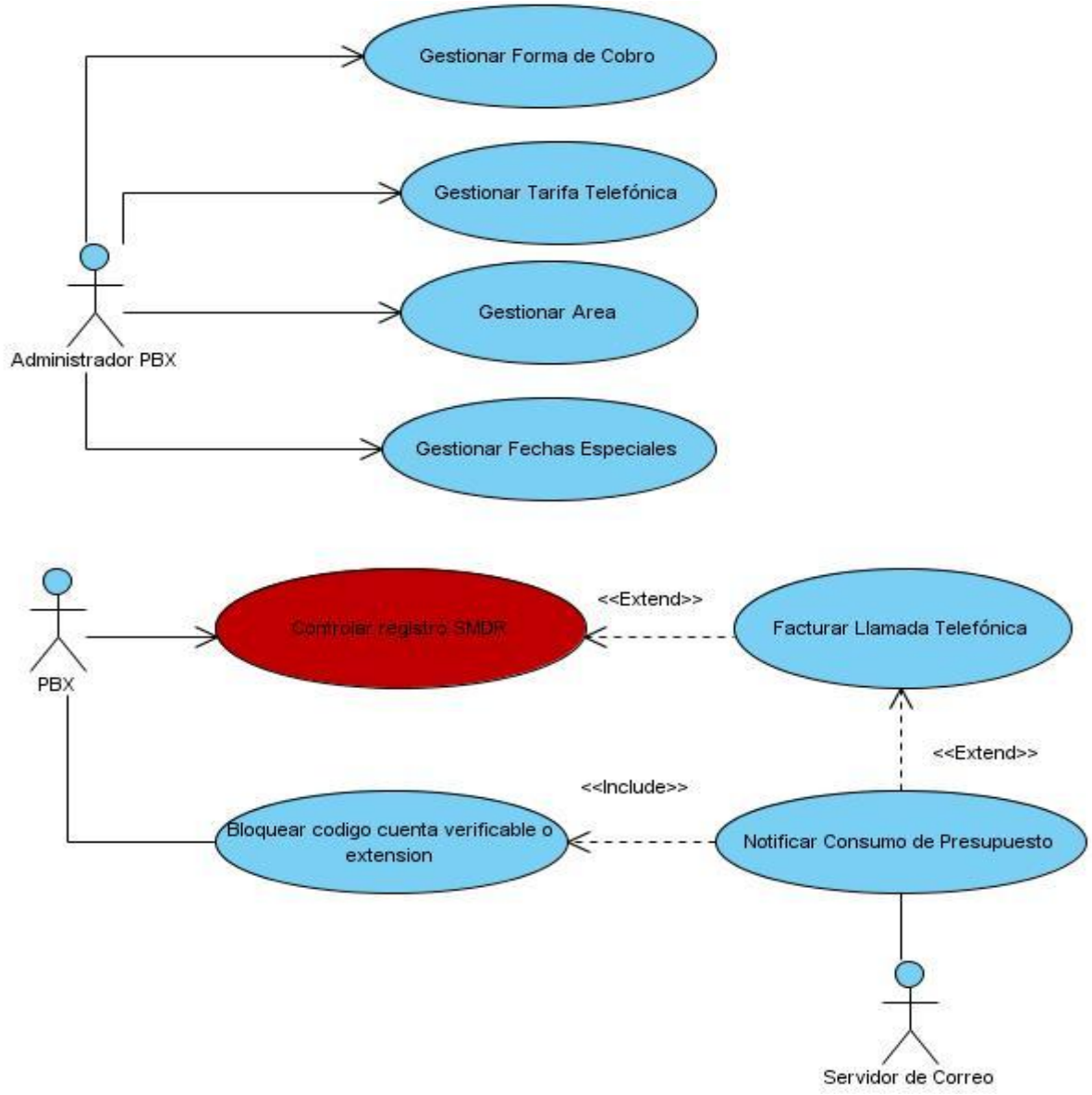


Figura 2. Diagrama de Casos de Uso de Sistema

2.5.3 Breve Descripción de los Casos de Uso del Sistema

Los casos de uso son artefactos que describen, en forma de acciones y reacciones, el comportamiento del sistema desde el punto de vista del usuario, a continuación se hace una breve descripción de cada

caso de uso, que fue determinado para satisfacer los requisitos funcionales del sistema y el actor responsable de inicializarlo.

2.5.3.1 Caso de Uso ¨Gestionar Tarifa Telefónica¨

El propósito del caso de uso Gestionar Tarifa es de registrar una nueva tarifa telefónica, permitirá modificar las tarifas registradas, así como eliminarlas y mostrar detalles de alguna de ellas. El actor que inicializa el caso de uso es el administrador de la PBX. Los requisitos asociados son:

- Registrar tarifa telefónica.
- Modificar tarifa telefónica.
- Eliminar tarifa telefónica.
- Buscar tarifa telefónica.

2.5.3.2 Caso de Uso ¨ Facturar Llamada Telefónica¨

El propósito del caso de uso Facturar Llamada Telefónica es determinar el valor a abonar por concepto de realización de llamada telefónica. El caso de uso es inicializado por el sistema, una vez que se registre una llamada externa saliente el sistema automáticamente le realizará el proceso de facturación. Requisito asociado al caso de uso:

- Determinar costo de la llamada.

2.5.3.3 Caso de Uso “Gestionar Área”

El propósito del caso de uso Gestionar Área es de registrar una nueva área, permitirá modificar las áreas registradas, así como eliminarlas y mostrar detalles de ellas. El actor que inicializa el caso de uso es el administrador de la PBX. Los requisitos asociados son:

- Registrar área.
- Eliminar área.
- Modificar área.
- Buscar área.

2.5.3.4 Caso de Uso “Gestionar Forma de Cobro”

El propósito del caso de uso Gestionar Forma de Cobro es de registrar una nueva forma de cobro, permitirá modificar la forma de cobro registrada, así como eliminarla. El actor que inicializa el caso de uso es el administrador de la PBX. Los requisitos asociados son:

- Registrar forma de cobro.
- Modificar forma de cobro.
- Eliminar forma de cobro.

2.5.3.5 Caso de Uso "Gestionar Fechas Especiales"

El propósito del caso de uso Gestionar Fechas Especiales es de registrar una nueva fecha especial, permitirá eliminarlas y mostrar detalles de alguna de ellas. El actor que inicializa el caso de uso es el administrador de la PBX. Los requisitos asociados son:

- Registrar fecha especial.
- Eliminar fecha especial.
- Mostrar detalles de fecha especial.

2.5.3.6 Caso de Uso "Notificar Consumo de Presupuesto"

El caso de uso Notificar Consumo de Presupuesto tiene como objetivo verificar si el consumo total del presupuesto por llamadas realizadas es menor que el presupuesto asignado, si este consumo es excedido, se notifica al responsable de la extensión telefónica de la cual se realizó la llamada o al abonado que tiene asignado el código de cuenta verificable mediante el envió de un correo, informándole que ha sobrepasado el presupuesto que se le fue asignado y posteriormente se bloquea dicha extensión o el código de cuenta verificable.

2.5.3.7 Caso de Uso"Bloquear código de cuenta verificable o extensión"

El caso de uso Bloquear código de cuenta verificable o extensión se ejecuta una vez que se haya ejecutado el caso de uso "Notificar Consumo de Presupuesto" y tiene como objetivo bloquear el código de cuenta verificable o extensión asociado a la llamada que ya fue facturada y excedió el saldo asignado.

2.6 Conclusiones

En este capítulo fue descrita la propuesta de solución y las funcionalidades que el sistema debe cumplir a través de un modelo de dominio. Se analizaron y describieron los requisitos funcionales y no funcionales, los cuales son de gran importancia para el desarrollo del sistema ya que constituyen la funcionalidad del mismo. Además, se identificaron y describieron los Actores y Casos de Uso, estableciéndose las relaciones correspondientes entre cada uno de ellos en el Diagrama de Casos de Uso del sistema.

CAPÍTULO 3: DISEÑO E IMPLEMENTACIÓN DEL SISTEMA

3.1 Introducción

En el presente capítulo se tratan los temas relacionados con el diseño y la implementación del sistema. Se definen los modelos de clases del diseño y el modelo de implementación para el módulo de Facturación para el Sistema para la Gestión Integral de Costos de Llamadas. Además, se definen el modelo de despliegue y modelo de componentes.

3.2 Diseño

El diseño es un refinamiento del análisis que tiene en cuenta los requisitos no funcionales, en definitiva, CÓMO cumple el sistema sus objetivos. El diseño debe ser suficiente para que el sistema pueda ser implementado sin ambigüedades. De hecho, cuando la precisión del diseño es muy grande, la implementación puede ser hecha por un generador automático de código. En el diseño además se tienen en cuenta diferentes patrones de diseño que no son más que diferentes soluciones que ya existen a los principales problemas que han surgido en el desarrollo de software.

3.2.1 Modelo de Diseño

El Modelo de Diseño es una abstracción del Modelo de Implementación y su código fuente, el cual fundamentalmente se emplea para representar y documentar su diseño. El Modelo de Diseño puede contener: los diagramas, las clases, paquetes, subsistemas, cápsulas, protocolos, interfaces, relaciones, colaboraciones, atributos, las realizaciones de los casos de uso, entre otros que se puedan considerar para el sistema en desarrollo.

3.2.1.1 Diagrama de Clases del Diseño

A continuación se representan las clases del modelo de diseño del módulo Facturación, para un mejor entendimiento:

Leyenda:


Clases de la Capa Presentación.


Clases del Negocio.


Clases Acceso a Datos.


Clases del Dominio.

3.2.1.1.1 Caso de Uso" Facturar Llamada"

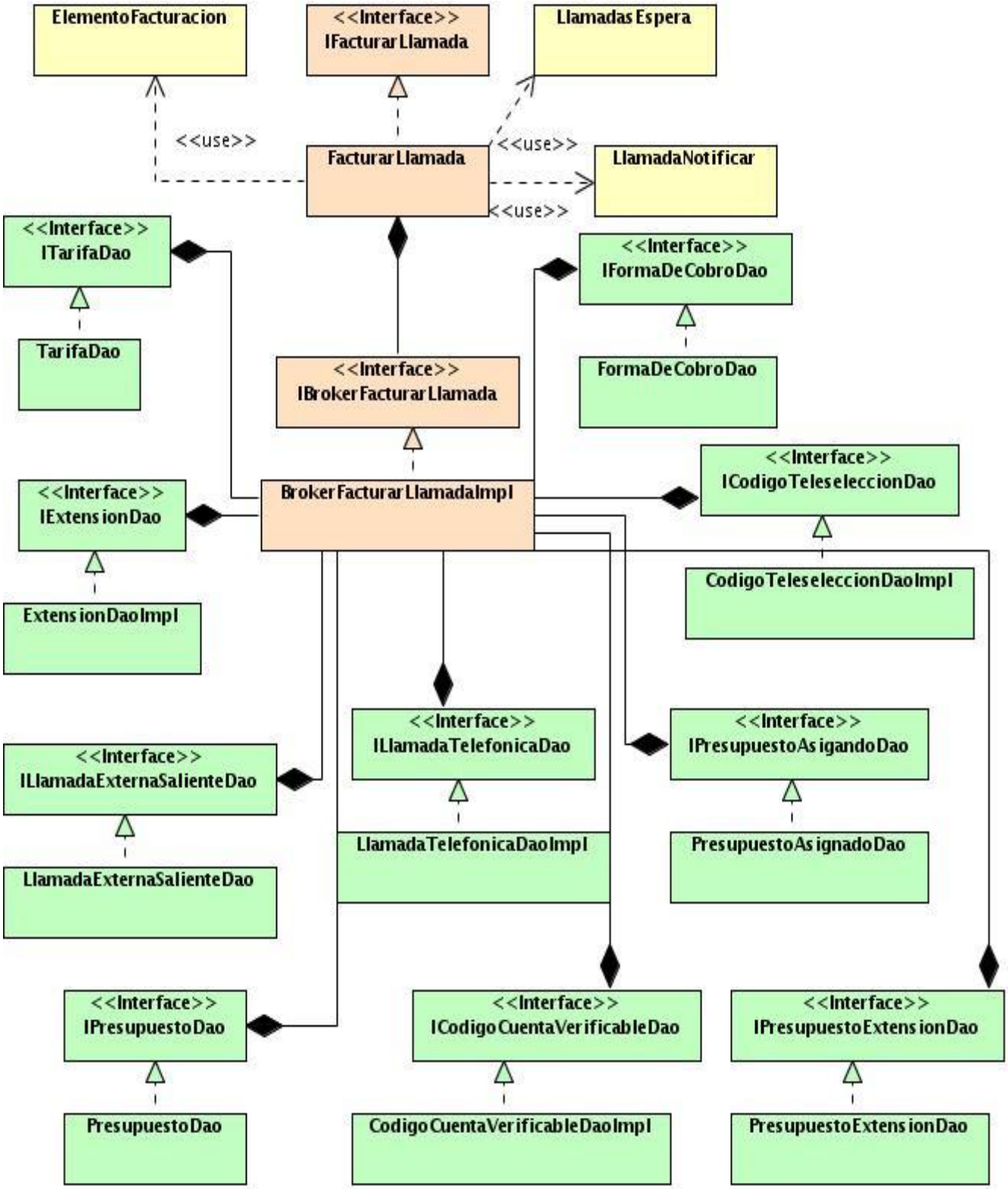


Figura 3. Diagrama de Clases del Diseño. Caso de Uso "Facturar Llamada"

3.2.1.1.2 Caso de Uso "Gestionar Área"

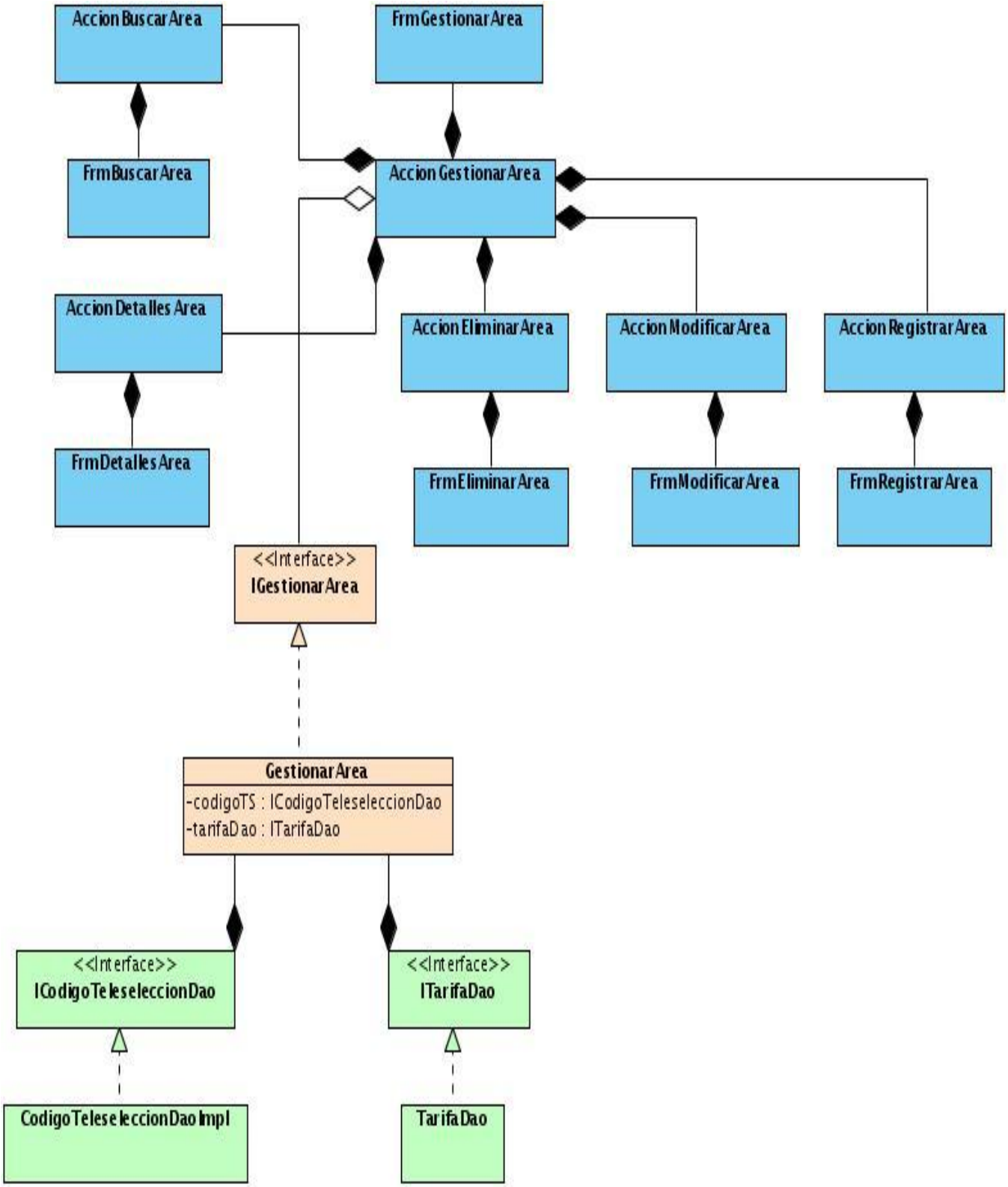


Figura 4. Diagrama de Clases del Diseño. Caso de Uso" Gestionar Área"

3.2.1.1.3 Caso de Uso "Gestionar Tarifa"

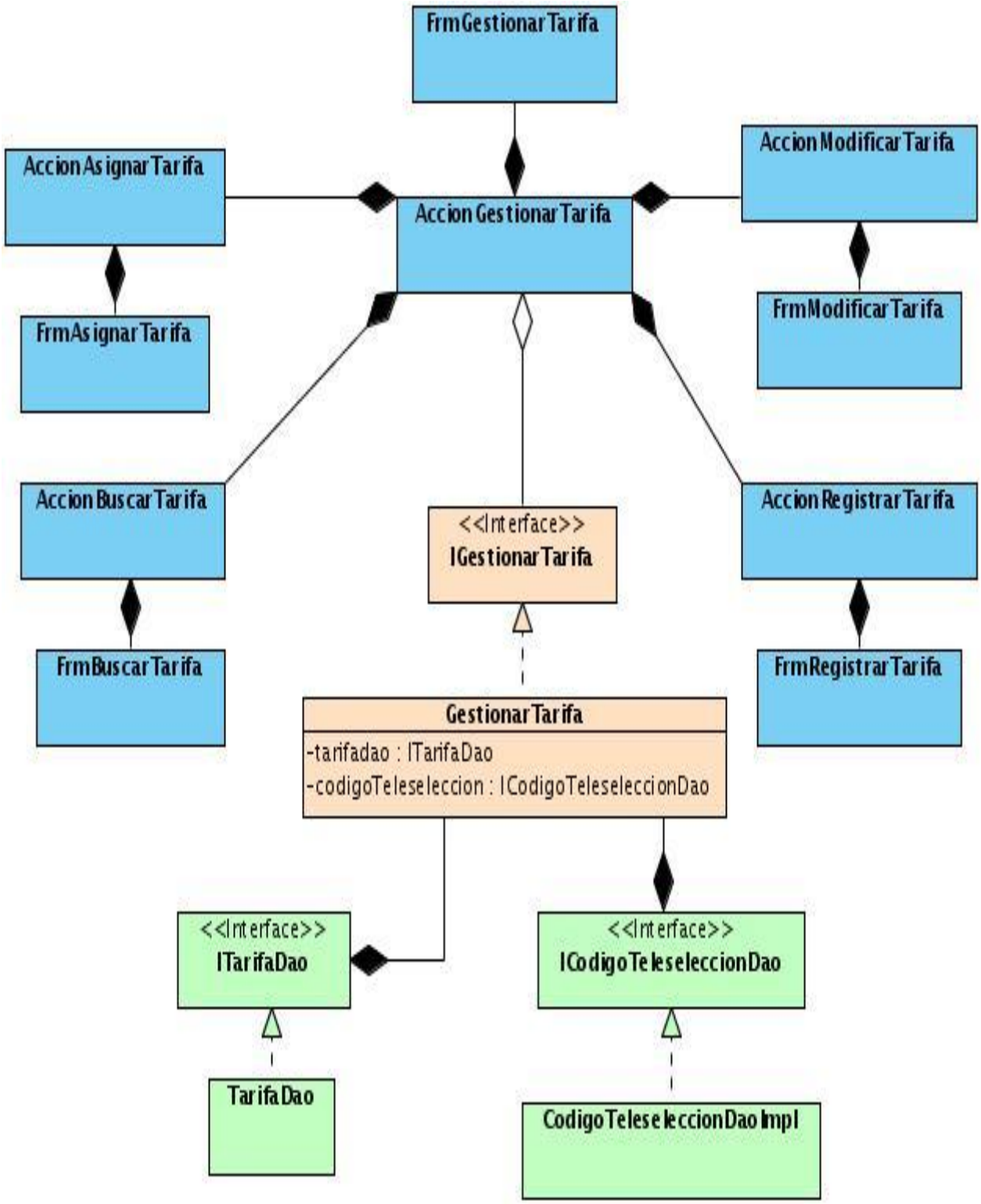


Figura 5. Diagrama de Clases del Diseño. Caso de Uso" Gestionar Tarifa"

3.2.1.1.4 Caso de Uso" Gestionar Forma de Cobro"

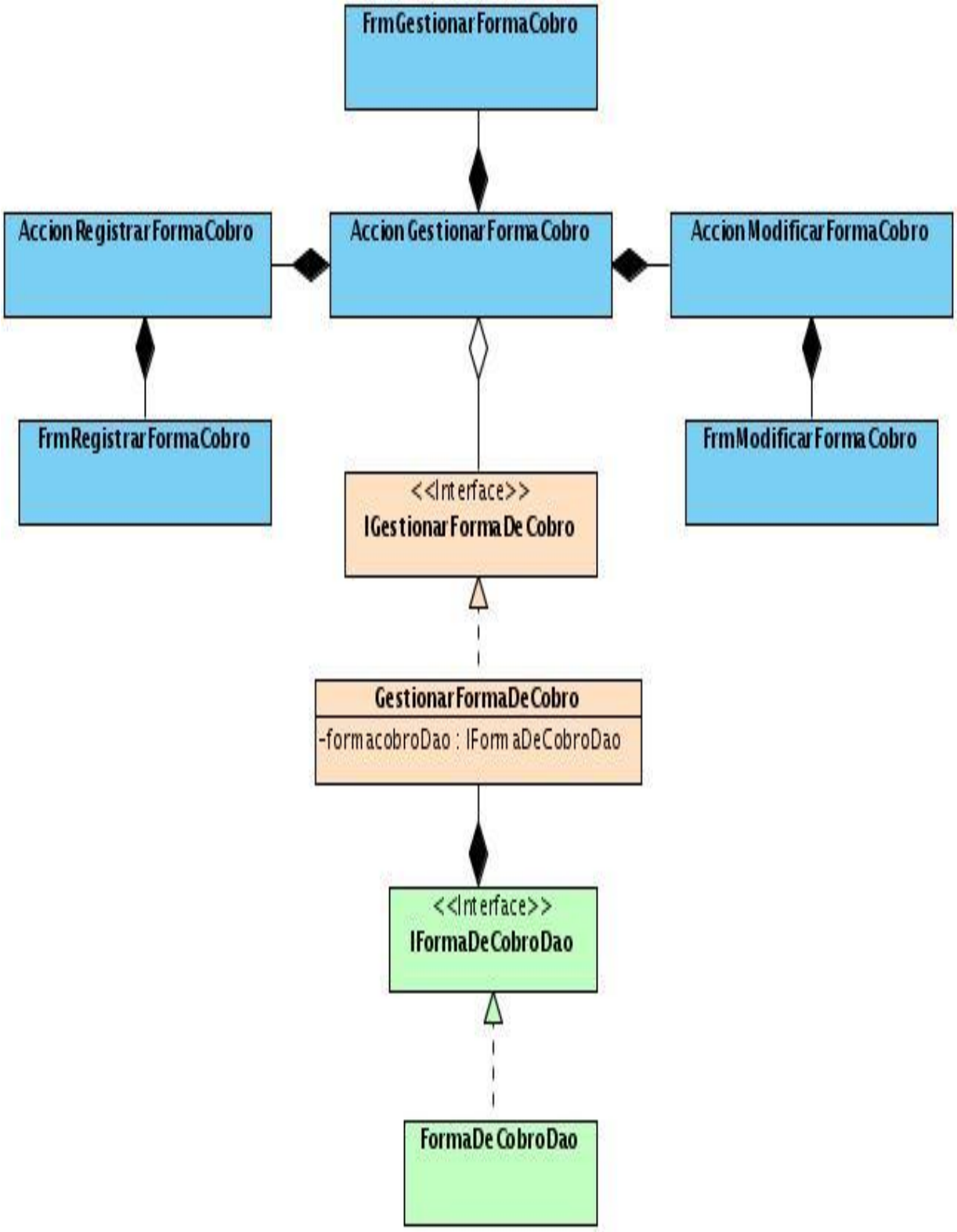


Figura 6. Diagrama de Clases del Diseño. Caso de Uso" Gestionar Forma de Cobro"

3.2.1.1.5 Caso de Uso" Gestionar Fechas Especiales"

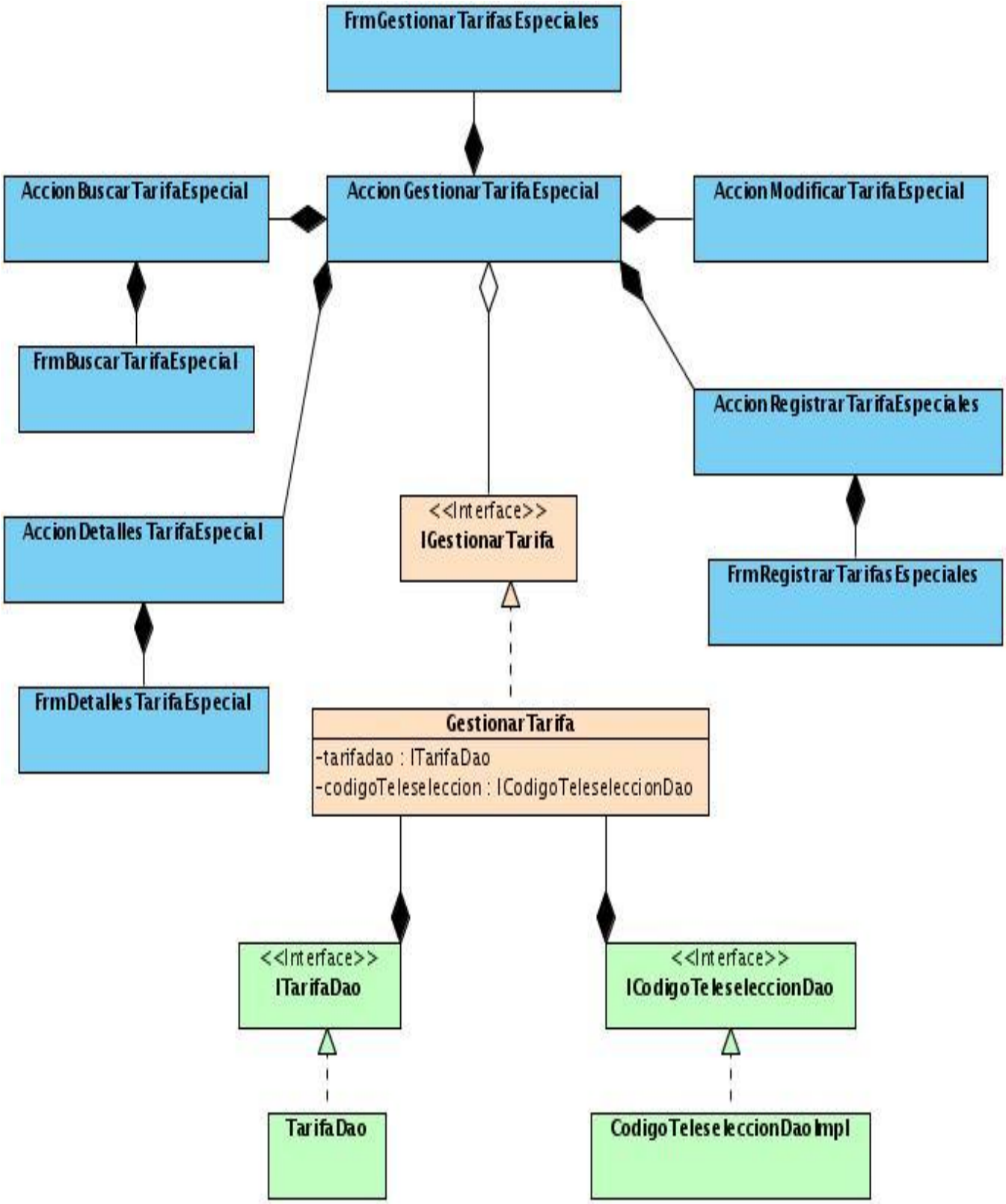


Figura 7. Diagrama de Clases del Diseño. Caso de Uso" Gestionar Fechas Especiales"

3.2.1.1.6 Casos de Uso" Notificar Consumo de Presupuesto y Bloquear código de cuenta verificable o extensión"

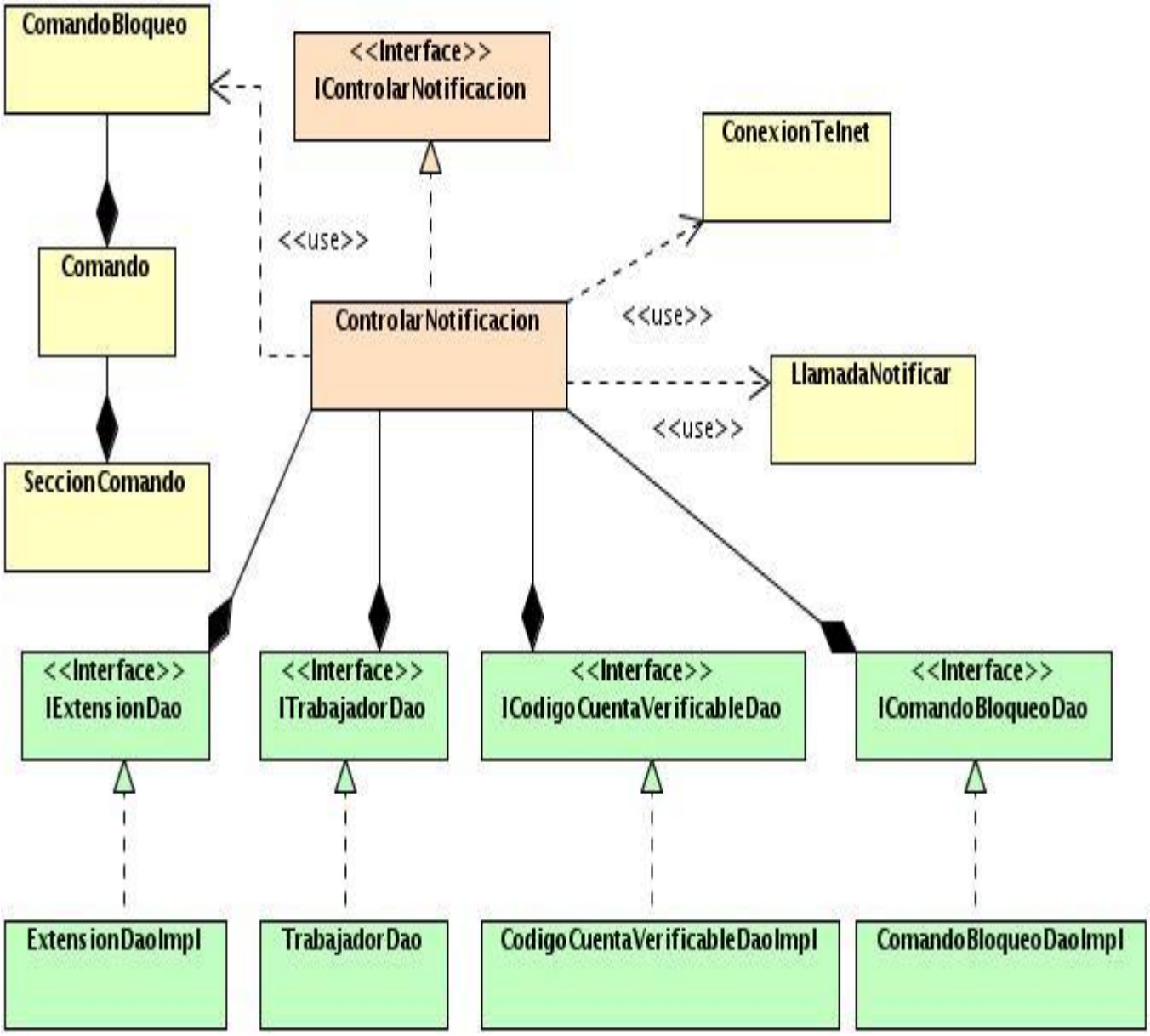


Figura 8. Diagrama de Clases del Diseño¨ Notificar Consumo de Presupuesto y Bloquear código de cuenta verificable o extensión ¨

3.3 Patrones

Es una solución a un problema en un contexto, codifica conocimiento específico acumulado por la experiencia en un dominio. Un sistema bien estructurado está lleno de patrones. Cada patrón describe un problema que ocurre una y otra vez en nuestro ambiente, y luego describe el núcleo de la solución a ese problema, de tal manera que se puede usar esa solución un millón de veces más, sin hacer jamás la misma cosa dos veces. (16)

Categorías de patrones:

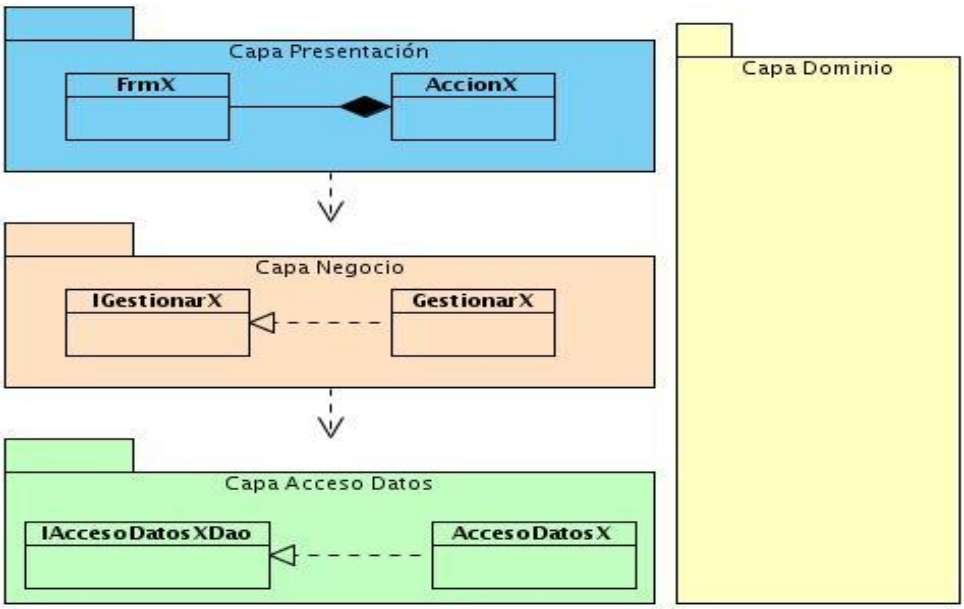
- **Patrones arquitecturales**: Aquellos que expresan un esquema organizativo estructural fundamental para sistemas software.
- **Patrones de diseño**: Aquellos que expresan esquemas para definir estructuras de diseño (o sus relaciones) con las que construir sistemas software.
- **Idiomas**: Patrones de bajo nivel, específicos para un lenguaje de programación o entorno concreto.

Patrón de Arquitectura propuesto.

El patrón arquitectónico aplicado es el de n capas *(Layers),* pues el sistema estará estructurado específicamente en cuatro capas: la capa de presentación, capa de negocio, capa de acceso a datos y la capa de dominio. En esta estructura la capa inferior proporciona servicios a la superior. Las capas se tratarán de forma independiente, pues cada una encapsula un aspecto concreto del sistema.

Este patrón es importante porque simplifica la comprensión y la organización del desarrollo del sistema, reduciendo las dependencias de forma que las capas más bajas no son conscientes de ningún detalle o interfaz de las superiores.

La arquitectura del sistema puede verse representada en la siguiente figura donde la capa de presentación contiene los formularios y las acciones que controlan los formularios. La capa de negocio contiene las clases que manejan la lógica del negocio; la capa de acceso a datos, contiene las clases que se relacionan con la base de datos y la capa de dominio contiene las clases entidad que son utilizadas por cualquiera de las clases de las restantes capas.



Patrón de diseño

Un patrón de diseño es:

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

Clasificación de patrones de diseño

Las categorías en las que se dividen los patrones de diseño son las siguientes:

- **De Creación**: Abstraen el proceso de creación de instancias de objetos. Ayudan a hacer a un sistema independiente de cómo se crean, se componen y se representan sus objetos.
- **Estructurales**: Tratan con la composición de clases u objetos. Se ocupan de cómo las clases y objetos son utilizados para componer estructuras de mayor tamaño.
- **De Comportamiento**: Caracterizan el modo en que las clases y objetos interactúan y se reparten la responsabilidad. Atañen a los algoritmos y a la asignación de responsabilidades entre objetos.

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

- **Singleton**

Problema: Se necesitan mantener los datos llamadas en espera y las llamadas a notificar autenticadas en la aplicación.

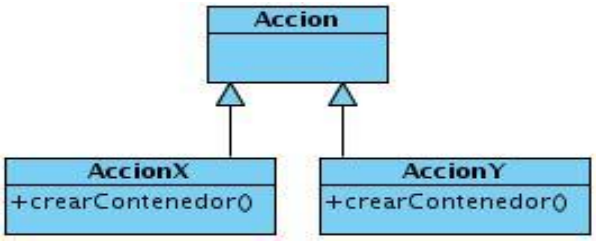
Solución: El patrón Singleton garantiza que, en todo momento, solo existe un objeto de una clase en particular.

- **Factory Method**

Problema: Se necesita crear formularios con características diferentes entre ellos.

Solución: El patrón Factory Method garantiza la creación de objetos concretos.

Implementación:



- **Inyección de Dependencias**

Problema: Se necesitan crear objetos y componentes reutilizables.

Solución: La inyección de dependencias garantiza la inyección a cada objeto de los objetos necesarios según las relaciones plasmadas en un fichero de configuración.

Implementación: Se delega en Spring la creación de los objetos.

```
<bean id="beanacciongestionartarifa" class="presentacion.accion.AccionGestionarTarifa">
   <property name="accionRegistrarTarifa" ref="beanaccionregistrartarifa"/>
   <property name="accionBuscar" ref="beanaccionbuscartarifa"/>
   <property name="accionModificar" ref="beanaccionmodificartarifa"/>
   <property name="gestionarTarifa" ref="beangestionartarifa"/>
   <property name="container" ref="FrmGestionarTarifa" />
</bean>
```

```
private AccionRegistrarTarifa accionRegistrarTarifa;
private AccionBuscarTarifa accionBuscar;
private AccionModificarTarifa accionModificar;
private IGestionarTarifa gestionarTarifa;
private List<Horario> lista;
```

- **Inversión de Control**

Se utiliza para delegar en otro componente, un framework por ejemplo, la responsabilidad de crear las instancias necesarias en lugar de crearlas nosotros mismos.

```
<bean id="FrmGestionarTarifa" factory-bean="beanacciongestionartarifa" factory-method="crearContenedor">
```

3.4 Diseño de Base de Datos

3.4.1 Modelo Lógico de Datos

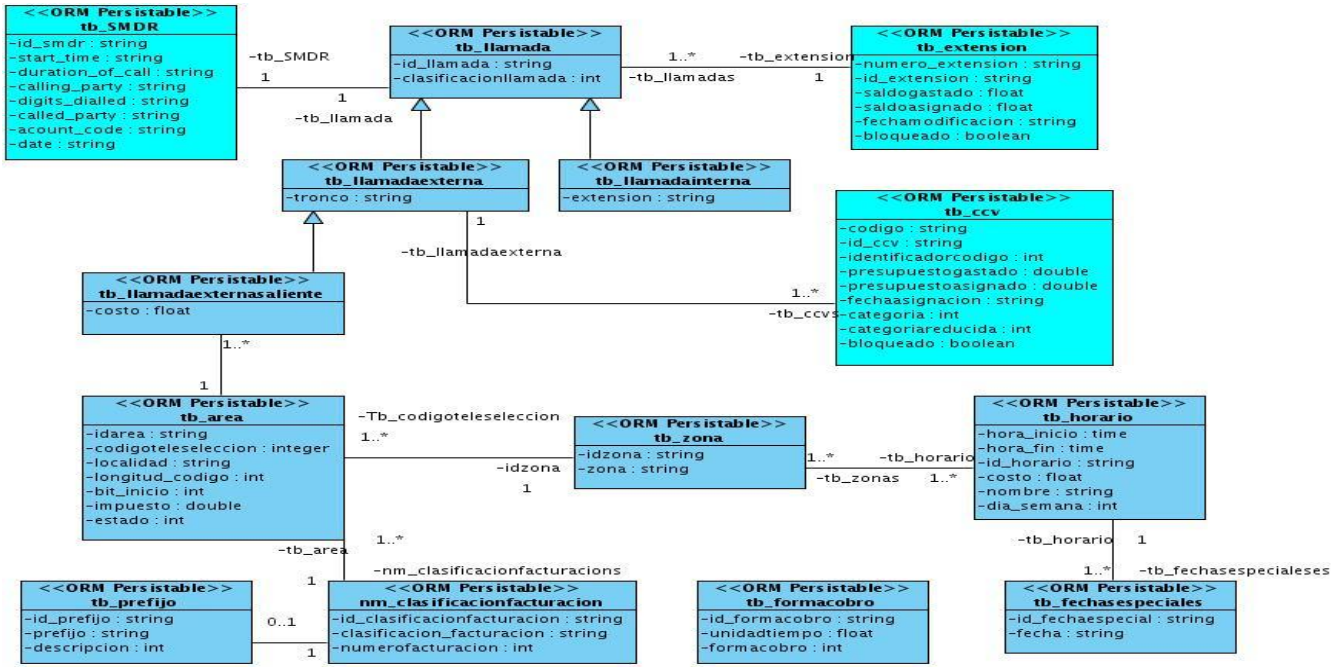


Figura 9. Modelo Lóogico de Datos

Leyenda:

Modulo Facturación

Pertenecen a otro modulo del proyecto

3.4.2 Modelo Físico de Datos del Sistema

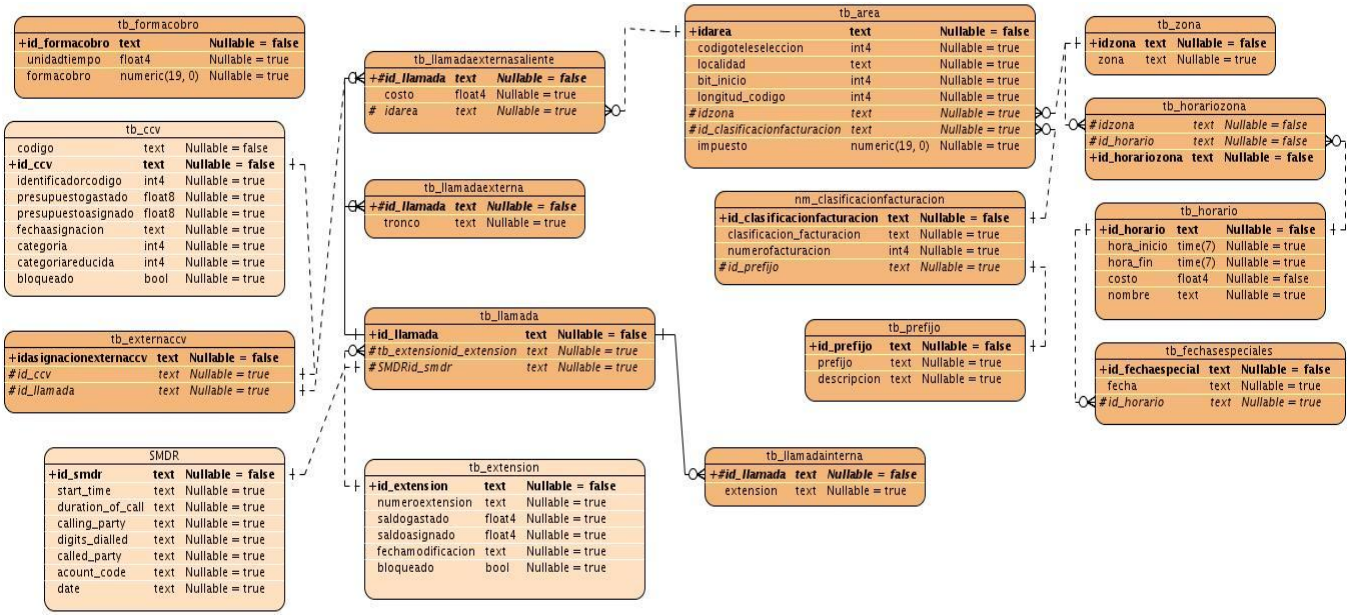


Figura 10. Modelo Físico de Datos

Leyenda:

Modulo Facturación

Pertenecen a otro modulo del proyecto

3.5 Diagrama de Despliegue

El Diagrama de Despliegue es un tipo de diagrama del Lenguaje Unificado de Modelado que se utiliza para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes. Los elementos usados por este tipo de diagrama son nodos, componentes y asociaciones.

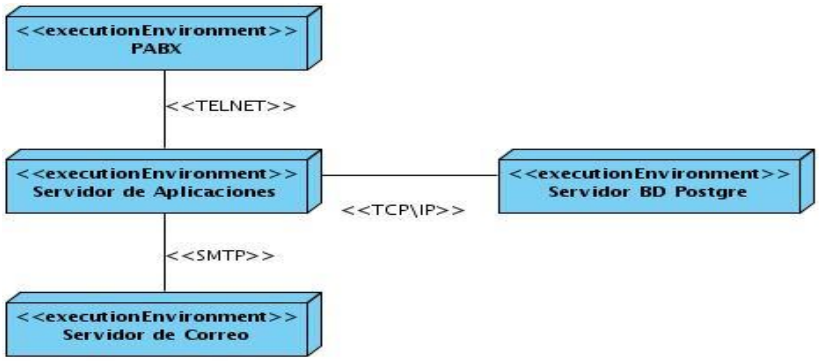


Figura 11. Diagrama de Despliegue

3.6 Diagrama de Componentes

Un diagrama de componentes representa cómo un sistema de software es dividido en componentes y muestra las dependencias entre estos componentes. Los componentes físicos incluyen archivos, cabeceras, bibliotecas compartidas, módulos, ejecutables, o paquetes. Los diagramas de Componentes prevalecen en el campo de la arquitectura de software pero pueden ser usados para modelar y documentar cualquier arquitectura de sistema. Son utilizados para modelar la vista estática y dinámica de un sistema. Muestra la organización y las dependencias entre un conjunto de componentes.

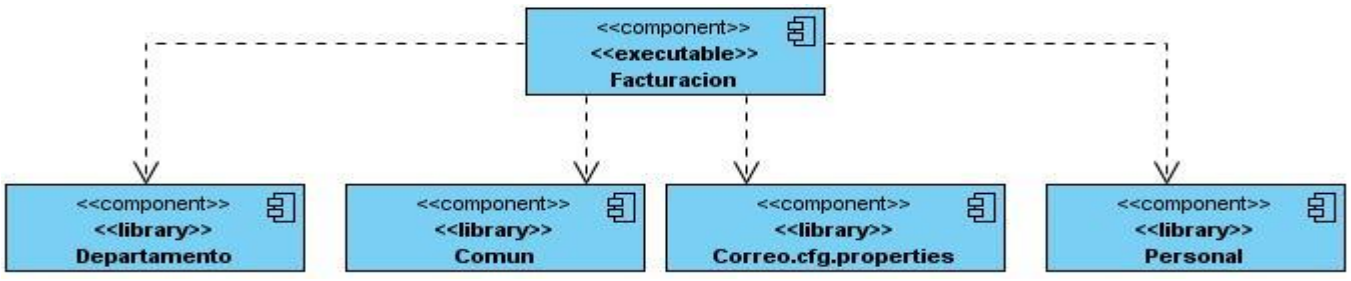


Figura 12. Diagrama de Componentes

A continuación se describen los componentes:

Facturación: Es el punto de salida de la aplicación.

Común: Contiene todas las clases que son usadas por varios de los componentes de la aplicación.

Departamento: Contiene clases que permiten a los usuarios controlar todas las operaciones que se realizan sobre los departamentos.

Personal: Contiene clases que permiten a los usuarios controlar todas las operaciones que se realizan con el personal, ya sean usuarios o trabajadores.

Correo.cfg.properties: Contiene los parámetros de configuración para el envío de correo electrónico.

3.7 Conclusiones

En este capítulo se modela el diagrama de clases del diseño, el modelo lógico de datos y el modelo físico de datos. También se obtiene el modelo de despliegue, que describe las configuraciones sobre las cuales deberá implementarse el sistema y el diagrama de componentes que ilustra los componentes de software que se usarán para construir el sistema.

CAPÍTULO 4: ESTUDIO DE LA FACTIBILIDAD

4.1 Introducción

La estimación y planificación del proyecto es un paso importante que no se debe obviar en la realización del mismo. Tiene como objetivo estimar con cierto grado de certeza los recursos necesarios para el desarrollo del proyecto, ya sean recursos de hardware, software, esfuerzo, tiempo, costo y en base a esto tomar la mejor decisión, de si se decide continuar con el desarrollo del software o no. En el presente capítulo se realizará un estudio de factibilidad para la realización del sistema propuesto mediante una estimación de tamaño, esfuerzo y planificación necesaria para llevar a cabo el mismo.

4.2 Método de Estimación por puntos de Casos de Uso

Con esta técnica existe la posibilidad de predecir el tamaño de un sistema a partir de las características de sus requisitos, expresados en los casos de uso. Se realiza mediante la asignación de pesos a un cierto número de factores que lo afectan de complejidad técnica y ambiente para finalmente, contabilizar el tiempo total estimado para el proyecto a partir de esos factores.

4.3 Planificación

4.3.1 Cálculo de Puntos de Casos de Uso sin Ajustar

Se tiene 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.

Tipo	Descripción	Peso	Cant * 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 interfaz	2	2*2

Complejo	basada en texto.		
	Una persona que interactúa con el sistema mediante una interfaz gráfica.	3	3*1
Total UAW			7

Tabla 5. Para calcular el Factor de Peso de los Actores sin ajustar (UAW)

Tipo	**Descripción**	**Peso**	**Cant * Peso**
Simple	El caso de uso contiene de 1 a 3 transacciones.	5	4*5
Medio	El caso de uso contiene de 4 a 7 transacciones.	10	3*10
Complejo	El caso de uso contiene más de 8 transacciones.	15	0*15
Total UUCW			50

Tabla 6. Para calcular el Factor de Peso de los Caso de Uso sin ajustar (UUCW)

Luego:

UUCP= 7 + 50

UUCP= 57 (Factor de Peso de los Casos de Uso sin ajustar)

4.3.2 Cálculo de Puntos de Casos de Uso Ajustados

Se tiene la siguiente ecuación:

UCP= UUCP * TCF * 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.

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	2	3
T4	Procesamiento interno complejo.	1	5	1
T5	El código debe ser reutilizable.	1	4	5
T6	Facilidad de instalación.	0.5	1	2
T7	Facilidad de uso.	0.5	3	2.5
T8	Portabilidad.	2	5	5
T9	Facilidad de cambio.	1	3	3
T10	Concurrencia.	1	5	5
T11	Incluye objetivos especiales de seguridad.	1	5	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				34.5

Tabla 7. Para calcular Factor de complejidad técnica (TCF)

Para calcular TCF:

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

TCF = 0.6 + 0.01 *34.5

TCF =0.945 (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 de objetos.	1	4	4
E4	Capacidad del analista líder.	0.5	4	2
E5	Motivación.	1	5	5
E6	Estabilidad de los requerimientos.	2	3	6
E7	Personal part-time.	-1	2	-2
E8	Dificultad del lenguaje de programación.	-1	3	-3
Total EF				13.5

Tabla 8. Para calcular el Factor Ambiente (EF)

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)

Luego calculando los puntos de Caso de Uso Ajustados quedaría:

UCP= UUCP * TCF * EF

UCP= 57 * 0.945 * 0.995

UCP= 53.5956 (Puntos de Casos de Uso ajustados)

4.3.3 Cálculo de Esfuerzo del Flujo de Trabajo Implementación

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

- CF = 20 horas-hombre (si Total EF ≤ 2)
- CF = 28 horas-hombre (si Total EF = 3 ó Total EF = 4)
- CF = abandonar o cambiar proyecto (si Total EF ≥ 5)

Total EF = Cant EF < 3 (entre E1 –E6) + Cant EF > 3 (entre E7, E8)

Como el total contabilizado es 3 se utiliza el factor de conversión 28 horas-hombre/Punto de Casos de Uso.

EF= 3 + 0

EF= 3

E = UCP * CF

E = 53.5956 * 28 horas-hombre

E = 1500,6768 horas-hombre (Esfuerzo estimado en horas-hombre)

4.3.4 Cálculo del Esfuerzo Total del Modulo Facturación

Actividad	% Esfuerzo	Valor Esfuerzo
Negocio	10%	250.1128 horas-hombre
Diseño	30%	750.3384 horas-hombre
Implementación	60%	1500,6768 horas-hombre
Total	100%	2501.128 horas-hombre

Tabla 9. Cálculo del esfuerzo total de todo el proyecto

EL esfuerzo total (ET) del proyecto sería:

ET = 2501.128 Horas-Hombres.

ET = 17.894 Mes-Hombres.

El tiempo de desarrollo (TD) del modulo Facturación sería:

TD = ET / CH

TD = 17.894 Mes-Hombres **/** 3 Hombre

TD = 5.10 Mes

Donde:

CH: Cantidad de Hombres.

Teniendo un hombre, se estima que el tiempo de desarrollo del Modulo Facturación sería aproximadamente **17** meses. Para un equipo de **3** personas el proyecto tiene una duración de **5** meses y **10** días.

4.3.5 Cálculo del Costo Total del Modulo Facturación

El costo total del proyecto (CT) sería:

CT = ET * CHM / CH

CT = 17.894 Mes-Hombres *** 86** Hombre-Mes **/ 1** Hombre

CT = 1538.88

Donde:

CHM: Costo Hombre-Mes.

Asumiendo que el costo por Hombre-Mes es $86 se estima que el costo total del proyecto sería $1538.88 aproximadamente.

4.4 Beneficios Tangibles e Intangibles

4.4.1 Tangibles

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

El Módulo Facturación del Sistema para la Gestión Integral de los Costos de llamadas en Pizarras Telefónicas no es un producto utilizado con fines comerciales u otros intereses similares. Su desarrollo permitirá facturar todas las llamadas externas salientes que sean registradas en la PABX, así llevar con control del consumo del presupuesto asignado y en caso de que este sea sobrepasado notificar mediante un servidor de correo y bloquear el código de cuenta verificable asignado a un abonado o la extensión de la cual se efectuó la llamada. El beneficio principal que reporta el sistema es contar con una solución informática que permita facturar de manera eficiente las llamadas realizadas desde la red

interna hacia la red pública, es decir, las llamadas externas salientes, además el ahorro que reporta el poseer un software que permita controlar que no se exceda el saldo asignado.

4.4.2 Intangibles

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:

- Se contara con una aplicación flexible, dinámica, fácil de usar y de interfaz agradable que permitirá la facturación de las llamadas telefónicas externas salientes.
- Disminución de los gastos, debido a que no hay que comprar el software.

4.5 Análisis de Costos y Beneficios.

El Módulo Facturación del Sistema 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 facturación de las llamadas telefónicas externas salientes.

4.6 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 módulo Facturación que permitió valorar qué tan factible sería la realización del mismo.

CONCLUSIONES GENERALES

En el presente trabajo de diploma se detallaron y describieron cada uno de los elementos y procesos que intervienen en la facturación de las llamadas, para poder obtener una mejor comprensión de cómo se lleva a cabo todo este proceso. Se determinó la metodología para el desarrollo de software y fueron propuestas las herramientas óptimas para la implementación del sistema. Además, se analizaron e implementaron los patrones de diseño y estilos arquitectónicos; logrando una aplicación con un diseño robusto y flexible.

Con el desarrollo del Módulo Facturación del Sistema para la Gestión Integral de Costos de Llamadas en Pizarras Telefónicas (SGIC - PBX) se dio solución a los siguientes problemas:

- Inexistencia de un mecanismo que proporcione notificaciones automáticas al sobrepasar un presupuesto asignado por concepto de facturación de servicios.
- Imposibilidad para realizar un análisis periódico del consumo real y el presupuesto planificado para cada área o departamento.
- Sobre consumo del presupuesto asignado para las diferentes áreas o departamentos.

Por lo antes expuesto se concluye que los objetivos propuestos para el presente trabajo de diploma han sido cumplido de forma satisfactoria.

Recomendaciones.

RECOMENDACIONES

A continuación se listan las recomendaciones en vistas de posibles mejoras al sistema:

- Realizar pruebas para verificar la eficiencia de las funcionalidades realizadas.
- Implementar la funcionalidad que permita importar y exportar las tarifas telefónicas configuradas.

Referencias.

REFERENCIAS

1. Free Download Manager. [Online] [Cited: octubre 23, 2009.] http://www.freedownloadmanager.org/es/downloads/Llame_Compa%C3%B1ero_de_la_Contabilidad_27591_p/.

2. Freedown Load Manager. [Online] [Cited: octubre 23, 2009.] http://www.freedownloadmanager.org/es/downloads/PbxTools_PhoneJournal_2221_p/.

3. **Letelier, Patricio.** Portal Desarrollo de Software. [Online] [Cited: noviembre 2, 2009.] https://pid.dsic.upv.es/C1/Material/Documentos%20Disponibles/Forms/DispForm.aspx?ID=96&Source=https%3A%2F%2Fpid.dsic.upv.es%2FC1%2FMaterial%2Fdefault.aspx&RootFolder=%2FC1%2FMaterial%2FDocumentos%20Disponibles

4. **Comeau, Greg.** Zator. [Online] [Cited: noviembre 3, 2009.] http://www.zator.com/Cpp/E1_2.htm.

5. Manual de Java. [Online] [Cited: noviembre 3, 2009.] http://www.manual-java.com/manualjava/caracteristicas-java.html.

6. Laptop. [Online] [Cited: noviembre 4, 2009.] http://dev.laptop.org/~edsiper/byteofpython_spanish/ch01s02.html.

7. Freedown Load Manager. [Online] [Cited: noviembre 4, 2009.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%5Bcuenta_de_Plataforma_de_Java_14715_p/.

8. Kiokea.net. [Online] [Cited: noviembre 4, 2009.] http://es.kioskea.net/telecharger/telecharger-3866-postgresql.

9. **Pecos, Daniel.** PostgreSQL & MySQL. [Online] [Cited: 11 22, 2009.] http://danielpecos.com/docs/mysql_postgres/index.html.

10. Guia Ubuntu. [Online] [Cited: noviembre 5, 2009.] http://www.guia-ubuntu.org/index.php?title=NetBeans.

11. Atenas. [Online] [Cited: noviembre 5, 2009.] http://www.atenas.cult.cu/rl/informatica/manuales/sl/introduccion_al_SL/eclipse.html.

12. Comunidad de Programadores. [Online] [Cited: noviembre 5, 2009.] http://www.lawebdelprogramador.com/temas/enlace.php?idp=3545&id=44&texto=java.

13. Hibernate. [Online] [Cited: noviembre 10, 2009.] http://www.hibernate.org/.

14. *Conferencia Ingenieria de Software: Flujo de Trabajo de Requerimientos.* **Universidad de las Ciencias Informaticas. 2008.** 2008.

Referencias.

15. *Conferencia Ingenieria de Software: Arquitectura y Patrones de Diseño*. **Universidad de las Ciencias Informaticas. 2008.** 2008.

ANEXOS

1.1 Descripción de Casos de Uso

1.1.1 Descripción del Caso de Uso ¨Gestionar Tarifa Telefónica¨

Caso de Uso:	Gestionar Tarifa Telefónica
Actores:	Administrador PBX
Resumen:	El caso de uso inicia cuando el administrador de la PBX desea gestionar la tarifa telefónica. De esta forma puede registrar una nueva tarifa, modificar una tarifa, eliminar una tarifa y mostrar detalles de una tarifa. En dependencia de la opción que escoja realizará algunos pasos para terminar la acción de la opción que escogió.
Precondiciones:	El usuario debe estar autenticado como Administrador PBX.
Referencias	RF1.1, RF1.2, RF1.3, RF1.4, RF1.5
Prioridad	Crítica

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. Selecciona la opción que desea realizar: a) Registrar nueva tarifa telefónica b) Modificar tarifa telefónica c) Eliminar tarifa telefónica d) Mostrar detalles de la tarifa telefónica	2. Dependiendo de la opción seleccionada muestra la interfaz correspondientes: a) Registrar nueva tarifa telefónica, ir a la sección “Registrar nueva tarifa”. b) Modificar tarifa telefónica, ir a la sección "Modificar tarifa telefónica". c) Eliminar tarifa telefónica, ir a la sección "Eliminar tarifa telefónica". d) Mostrar detalles de la tarifa telefónica, ir a la sección “Detalles de la tarifa telefónica”.

Flujo Normal de Eventos	
Sección “Registrar nueva tarifa telefónica”	
1 Introduce los datos de la nueva tarifa telefónica para realizar su registro: • Nombre • Hora Inicio • Hora Fin • Costo	2 Verificar la existencia del tipo de tarifa introducida.

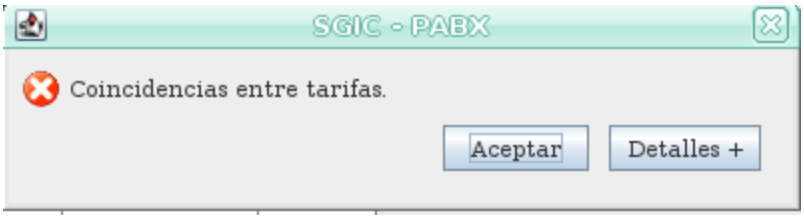
• Día de la semana	
	3 Verifica completitud de los campos mostrados
	4 Registrar datos de la nueva tarifa telefónica

Prototipo de Interfaz

Flujo Alterno 2.1

Acción del Actor	Respuesta del Sistema
	2.1 El tipo de tarifa introducida existe, emite un mensaje informativo

Prototipo de Interfaz



Flujo Alterno 3.1

Acción del Actor	Respuesta del Sistema
	3.1 Campos vacíos, muestra un mensaje informativo.

Prototipo de Interfaz

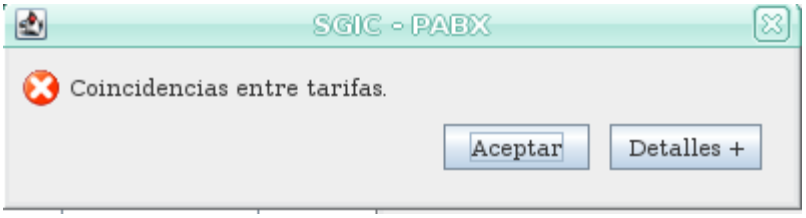
Sección "Modificar tarifa telefónica"	
Acción del Actor	**Respuesta del Sistema**
1. Selecciona el tipo de tarifa que desea modificar	2. Visualiza todos los datos del tipo de tarifa seleccionada.
3. Modifica los datos que desea actualizar: a) Nombre b) Hora Inicio c) Hora Fin d) Costo e) Día de la semana	4. Verifica la información introducida.
	5. Verifica completitud de los datos introducidos.
	6. Modifica los datos de la tarifa.

Prototipo de Interfaz

Flujo Alterno 4.1	
Acción del Actor	**Respuesta del Sistema**
	4.1. El nuevo tipo de tarifa introducida corresponde con uno existente, emite un

	mensaje informativo.

Prototipo de Interfaz



Flujo Alterno 5.1

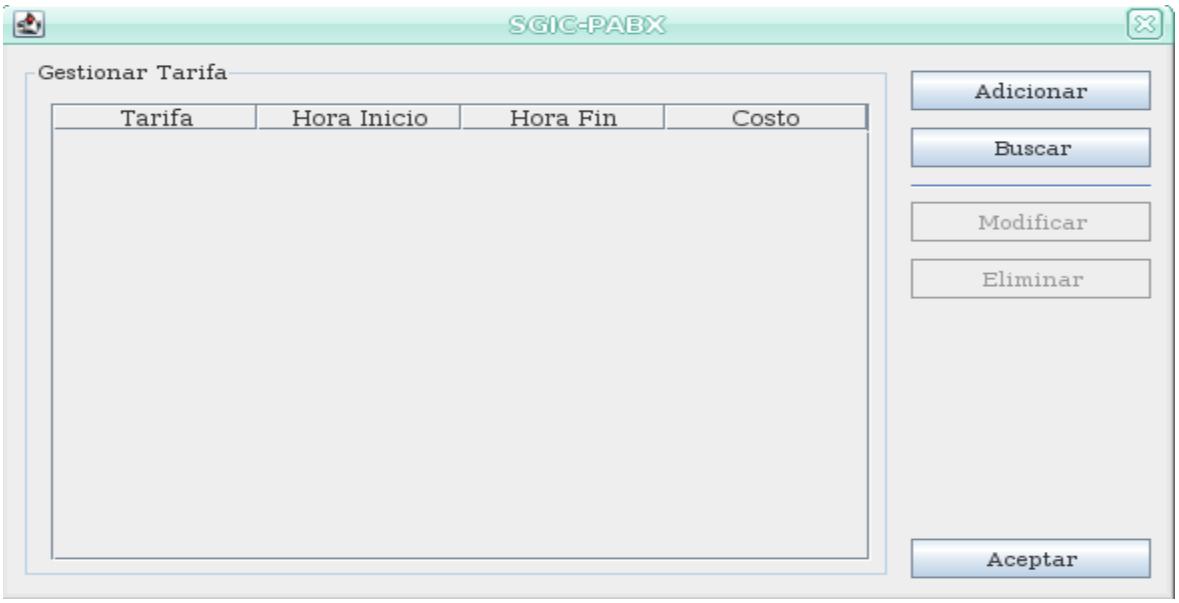
Acción del Actor	Respuesta del Sistema
	5.1. Campos vacíos, muestra un mensaje informativo.

Prototipo de Interfaz

Sección "Eliminar tarifa telefónica"

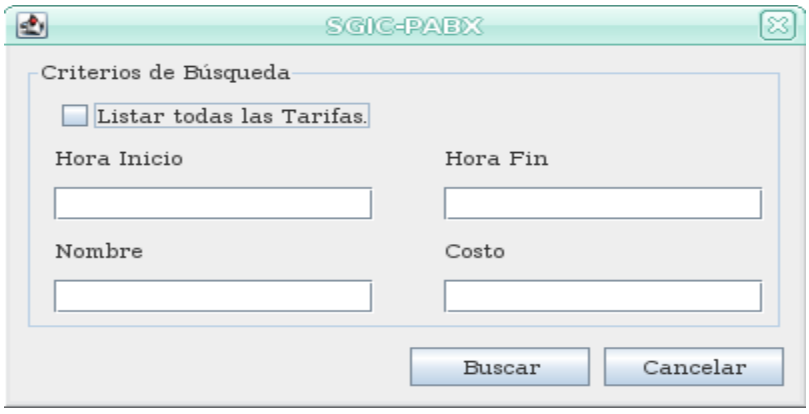
Acción del Actor	Respuesta del Sistema
1. Selecciona el tipo de tarifa a eliminar	2. Solicita confirmación para proceder con la eliminación.
3. Confirma eliminación	4. Elimina los datos de la tarifa telefónica

Prototipo de Interfaz



Sección "Mostrar detalles tarifa telefónica"	
Acción del Actor	**Respuesta del Sistema**
	1. Muestra los datos de la(s) tarifa(s) existente(s)

Prototipo de Interfaz



Poscondiciones	Fue adicionada, modificada, eliminada o detallada una tarifa telefónica.

1.1.2 Descripción del Caso de Uso ¨Facturar llamada telefónica¨

Caso de Uso:	Facturar llamada telefónica
Actores:	PBX
Resumen:	Calcula el costo de las llamadas telefónicas salientes, ya sean locales, nacionales o internacionales.
Precondiciones:	Debe existir al menos una llamada saliente.
Referencias	RF 2
Prioridad	Crítica

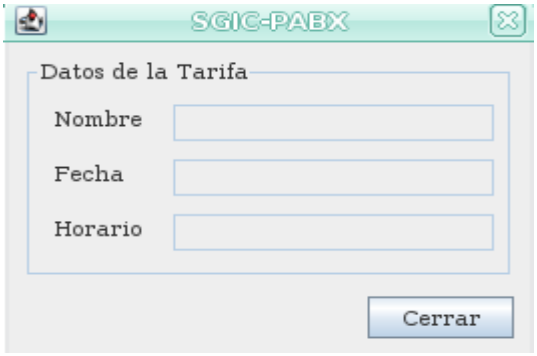
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
	1- Toma la llamada externa saliente.
	2- Toma de la llamada externa saliente el número marcado, la duración y si posee ccv.
	3- Determina del número marcado su código de teleselección.
	4- Se ubica el área a la que pertenece dicho código de teleselección.
	5- Se obtiene la zona a la que pertenece el área.
	6- Se obtienen las tarifas correspondientes a dicha zona.
	7- Se selecciona la tarifa en la que se encuentra ubicada la hora en que se realizo la llamada.
	8- Se calcula el costo que debe abonar.
	9- Si la llamada esta asociada a un código de cuenta verificable, ir a la sección ¨Descontar saldo al código de cuenta verificable¨. Si la llamada esta asociada a una extensión, ir a la sección ¨Descontar saldo a extensión telefónica¨.
Sección ¨Descontar saldo al código de cuenta verificable ¨	
	9.1 Se busca el saldo gastado hasta el momento por este código de cuenta verificable.
	9.2 Se suma el saldo gastado con el costo de la llamada.

	9.3 Se actualiza el saldo gastado por dicho código de cuenta verificable.
Sección ¨ Descontar saldo a extensión telefónica ¨	
	9.1 Se busca el saldo gastado hasta el momento por esta extensión.
	9.2 Se suma el saldo gastado con el costo de la llamada.
	9.3 Se actualiza el saldo gastado por dicha extensión.
Prototipo de Interfaz No aplica	
Poscondiciones	No aplica

1.1.3 Descripción del Caso de Uso ¨Gestionar Fechas Especiales ¨

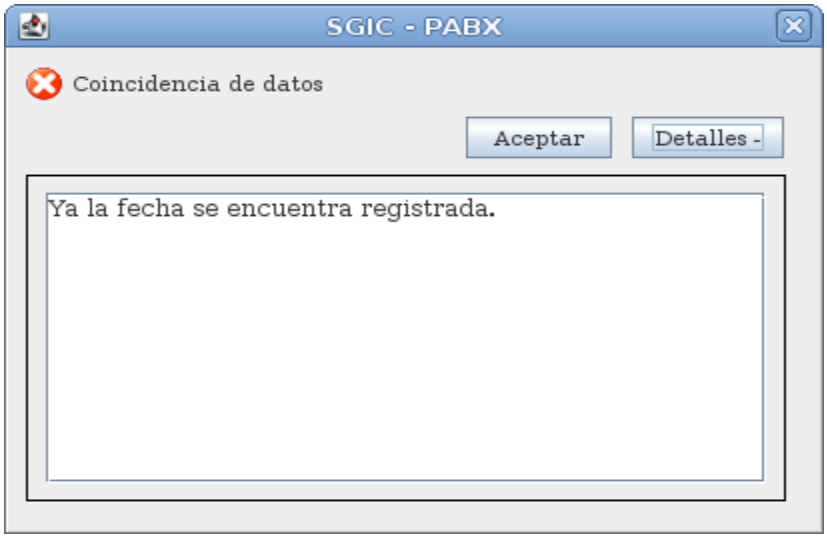
Caso de Uso:	Gestionar Fechas Especiales
Actores:	Administrador PBX
Resumen:	El caso de uso inicia cuando el administrador de la PBX desea gestionar fechas especiales. De esta forma puede registrar una nueva fecha especial, mostrar detalles de una fecha especial o eliminar una fecha especial. En dependencia de la opción que escoja realizará algunos pasos para terminar la acción de la opción que escogió.
Precondiciones:	El usuario debe estar autenticado como Administrador PBX.
Referencias	RF5.1, RF 5.2, RF 5.3
Prioridad	Crítica

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. Selecciona la opción que desea realizar: a) Registrar nueva fecha especial. b) Mostrar detalles fecha especial. c) Eliminar fecha especial	2. Dependiendo de la opción seleccionada muestra la interfaz correspondientes: a) Registrar nueva fecha especial, ir a la sección “Registrar nueva fecha especial”. b) Mostrar detalles de fecha especial, ir a la sección "Mostrar detalles fecha especial". c) Eliminar fecha especial, ir a la sección "Eliminar fecha especial".
Flujo Normal de Eventos	
Sección “Registrar nueva fecha especial”	
1. Introduce los datos de la nueva fecha especial para realizar su registro: a) Nombre. b) Fecha. c) Horario.	2. Verificar la existencia de la fecha especial introducida.
	3. Verifica completitud de los campos mostrados
	4. Registrar datos de la nueva fecha especial.
Prototipo de Interfaz	



Flujo Alterno 2.1	
Acción del Actor	**Respuesta del Sistema**
	2.1 El tipo de fecha especial introducida existe, emite un mensaje informativo

Prototipo de Interfaz

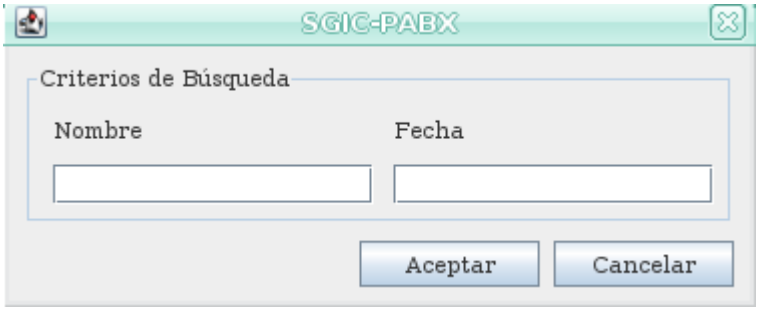


Flujo Alterno 3.1	
Acción del Actor	**Respuesta del Sistema**
	3.1 Campos vacíos, muestra un mensaje informativo.

Prototipo de Interfaz

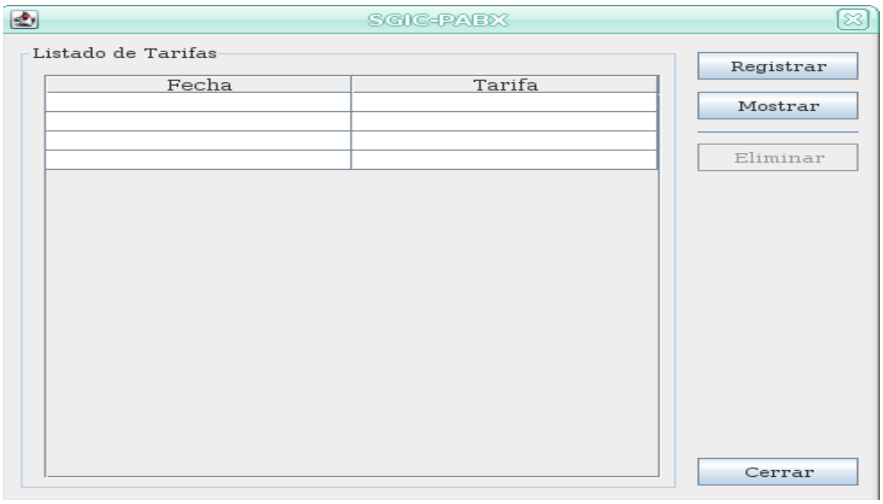
Sección “Mostrar detalles fecha especial”	
Acción del Actor	**Respuesta del Sistema**
1. Introduce los criterios de búsqueda.	2. Realiza la búsqueda.
	3. Muestra los datos de la(s) fecha(s) especial(s) existente(s)

Prototipo de Interfaz



Sección “Eliminar fecha especial”	
Acción del Actor	**Respuesta del Sistema**
1. Selecciona la fecha especial a eliminar.	2. Solicita confirmación para proceder con la eliminación.
3. Confirma eliminación.	4. Elimina los datos de la fecha especial

Prototipo de Interfaz



Poscondiciones	Fue adicionada, detallada eliminada una fecha especial.