Universidad de las Ciencias Informáticas

Facultad 4

Título: Diseño e implementación del acceso a datos del proyecto TeleBanca.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autores: Yanisleidys Armas de Armas

Edisbel Vázquez Barrios

Tutor: Ing. Yuliesky Torres Iglesias

Junio 2007

…” ¿Por qué esta magnífica tecnología científica,

que ahorra trabajo y nos hace la vida más fácil,

nos aporta tan poca felicidad?

La repuesta es está, simplemente: porque aún

no hemos aprendido a usarla con tino…”

Albert Einstein

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas (UCI) a hacer uso del mismo en su beneficio.

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

Yanisleidys Armas de Armas

_____

Edisbel Vázquez Barrios

_____

Ing. Yuliesky Torres Iglesias

_____

DATOS DE CONTACTO

Ing. Yuliesky Torres Iglesias
Graduado de Ingeniero Informático en el 2005 y profesor instructor con dos años de experiencia docente en la Universidad de las Ciencias Informáticas (UCI). Actual jefe del Proyecto TeleBanca.

AGRADECIMIENTOS

A todos nuestros familiares que nos han apoyado, a nuestros amigos y compañeros de aula. A todos nuestros profesores que cumplieron con su objetivo de formar profesionales para nuestra sociedad.

Las gracias también a nuestro comandante en jefe, Fidel Castro Ruz por darnos esta Revolución Socialista Cubana, y a esta última por habernos permitido alcanzar los conocimientos científicos y culturales que hemos alcanzado.

Y esa es la vida, a cada paso un escalón, cada escalón un sueño, y en cada sueño la posibilidad de hacerlos realidad. Hoy, gracias a todos los que nos han ayudado, hemos realizado uno.

Con todo el amor que puede guardar
el corazón de seres agradecidos.

DEDICATORIA

...A nuestros padres y hermanos que son los responsables de que hoy estemos aquí, a los que nos brindaron su apoyo para que se vieran realizados nuestros sueños y a aquellos que aunque no están presenten les hubiese gustado disfrutar como nosotros de este inolvidable y único momento…

RESUMEN

El auge de las nuevas tecnologías, han proporcionado a las empresas y organizaciones, nuevas opciones y herramientas que son utilizadas en la actualidad. Un ejemplo de ello lo son las aplicaciones Web para gestionar y automatizar todos los procesos que se lleven a cabo. Las aplicaciones Web prácticamente se han convertido en la base que provee facilidades y ventajas para la gestión de información, marketing, prestación de servicios, administración, entre otras.

El proyecto de Banca Telefónica, centrará su actividad en la posibilidad de desarrollar una aplicación la cual soportará la gestión de pago de servicios, la prestación de información a través de un Centro de llamadas y la personalización de las tarjetas necesarias para la utilización de este servicio. Esta aplicación brinda facilidades para la prestación de servicios de pago, control de los mismos así como la recuperación de información haciendo que todo este proceso sea realizado de una forma mas eficaz.

Por lo tanto necesita de un subsistema automatizado que realice el acceso a datos de esta aplicación, el cual debe permitir la interacción rápida y segura con las fuentes de datos como servidores de base de datos y debe automatizar la interacción y comunicación con los Web Services de los bancos asociados al proyecto.

En este documento se describen las etapas de análisis, diseño e implementación del subsistema antes mencionado, haciendo uso Lenguaje Unificado de Modelación (UML: *Unified Modeling Language*) y su herramienta por excelencia el *Rational Rose*.

TABLA DE CONTENIDOS

INTRODUCCIÓN

En las empresas cubanas de servicios, con el rápido incremento de sus clientes cada año, se hace cada vez más necesario de un sistema automatizado que controle, regule y registre los pagos de estos servicios, así como brindar otros muy necesarios para aumentar el confort de sus clientes. Hoy en día, son los clientes los que tienen que personarse en las oficinas territoriales de cada empresa, o esperar que un trabajador de la empresa se persone en su residencia, para efectuar el pago de los servicios prestados mensualmente, son ejemplos los servicios telefónicos de ETECSA (Empresa de Telecomunicaciones de Cuba), servicios hidráulicos, pago de multas, y servicios de gas licuado. Esta gestión se hace en la mayoría de las empresas de forma manual, y muy pocas son las que se apoyan en sistemas automatizados primarios. Esto trae como consecuencia que la gestión de estos procesos se haga con muy poca eficiencia y ocurran frecuentes atrasos en los pagos de estos servicios, lo que en muchos casos ocasiona el retiro del servicio y las sanciones a clientes. Es por esto que surge una nueva entidad para gestionar de forma conjunta los procesos de pago de servicios, la empresa TeleBanca, cuyo objetivo es funcionar como una banca telefónica donde los clientes registrados pueden efectuar los pagos, de manera electrónica, a las empresas que le brindan sus servicios por medio de llamadas telefónicas, brindándole al cliente un mayor confort y la posibilidad de efectuar sus pagos en cualquier momento; esta es la razón del surgimiento del proyecto productivo TeleBanca, cuyo objetivo es desarrollar una solución Web automatizada para gestionar todos los procesos en la empresa TeleBanca.

En esta tarea, se necesita de un subsistema automatizado para manipular la transferencia de información y la comunicación con los gestores de bases de datos que estén asociados a la aplicación Web; este subsistema es la capa de Acceso a Datos y su ausencia trae como consecuencia que los desarrolladores del sistema deban implementar en otros componentes o capas, las funcionalidades del acceso a fuentes de datos, empleando en esta actividad mucho tiempo de desarrollo y esfuerzo. Además, al tener que implementar estas funcionalidades exclusivas del acceso a fuentes de datos en otras capas, se afecta la independencia del sistema y la transparencia, por lo que cualquier cambio futuro en la implementación de estas funcionalidades implica cambiar gran cantidad de código en otras capas, lo que supone un gasto de esfuerzo adicional.

De modo general, en el trabajo que desarrollamos se persigue el siguiente objetivo:

- Desarrollar un subsistema automatizado para simplificar, flexibilizar y organizar el acceso a fuentes de datos en el proyecto productivo TeleBanca.

A partir del análisis del objetivo general se derivan los siguientes objetivos específicos:

- Construir la base de la arquitectura de la capa de Acceso a Datos.
- Proporcionar una base de arquitectura sólida y estable para su diseño e implementación.
- Implementar los elementos del diseño en términos de elementos de implementación.

Cómo se puede observar, los objetivos específicos se limitan a enunciar las ideas esenciales para la construcción del subsistema final y profundizar en el estudio de las funcionalidades requeridas para el desarrollo del mismo.

Para dar cumplimiento a estos objetivos se plantean las siguientes tareas:

- Investigar sobre las características y ventajas de la implementación de las capas de acceso a fuentes de datos en los sistemas de software.
- Investigar sobre el desempeño y funcionalidades que debe cumplir una capa de acceso a fuentes de datos en una aplicación.
- Estudiar de las técnicas, herramientas, componentes y modelos de arquitectura que se utilizan en las capas de Acceso a fuentes de datos en soluciones informáticas.
- Investigar las tendencias, tecnologías y lenguajes de programación usados para el diseño e implementación de capas de acceso a datos.
- Diseñar la arquitectura de la capa de acceso a fuentes de datos.
- Implementar los componentes desarrollados.

Estructura de los capítulos

El presente documento se estructura en cuatro capítulos y anexos, que incluyen todo lo relacionado con el trabajo investigativo realizado, así como también el diseño y la implementación del subsistema propuesto para la realización del mismo.

El Capítulo I *Fundamentación Teórica* Incluye el estado del arte del tema a tratar, y el análisis de de las tendencias, tecnologías, metodologías y software usados en la actualidad para resolver el problema. Se incluye además una breve descripción de las herramientas propuestas para la implementación y modelación.

El Capítulo II *Características del sistema* describe el objeto de estudio, problema y situación problémica, objeto de automatización, propuesta de sistema, el entorno de trabajo en que se desarrolla el sistema, requerimientos funcionales y no funcionales y los casos de uso del sistema.

El Capítulo III *Análisis y Diseño del Sistema* incluye los diagramas de interacción y diagrama de clases, además de la descripción de las clases, abarca el Diseño de la base de datos el cual contiene el Diagrama de Modelos de Datos y la descripción de las tablas.

El Capítulo IV *Implementación* incluye el diagrama de despliegue y el diagrama de componentes.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción

El vertiginoso auge de las nuevas tecnologías que empezaron con el surgimiento de Internet, han brindado a las empresas y organizaciones, opciones y herramientas que son explotadas con gran intensidad en la actualidad. Una de ellas es la utilización de aplicaciones Web para gestionar y automatizar sus procesos. Estas herramientas constituyen un elemento de extrema importancia ya que brindan sus ventajas en casi todos los aspectos: marketing, administración, prestación de servicios, gestión de información y publicidad; y las aplicaciones Web prácticamente se han convertido en la base que provee todas estas comodidades y ventajas.

Cada vez son más las organizaciones y empresas que notan la gran importancia y flexibilidad de las aplicaciones Web, de las cuales una de sus principales características es que la gran parte de las veces implican relaciones con fuentes y orígenes de datos, o sea, es de extrema importancia almacenar y acceder a la información crítica para la organización y gestionar eficientemente los procesos propios de la aplicación. Las tecnologías de Bases de Datos son el elemento por excelencia usado por estas aplicaciones dirigidas a la gestión de procesos empresariales; que aunque llevan muchos años de utilización, por su importancia constituyen un punto clave a la hora de diseñar un sistema.

Conjuntamente con este auge de las aplicaciones Web, ha surgido un nuevo modelo de arquitectura, el modelo multicapa, el cual se ha convertido actualmente en el patrón de arquitectura por excelencia en el desarrollo de software para la gestión empresarial ya que en este modelo de arquitectura, cada capa compone un subsistema en el que se ubican clases con responsabilidades propias; donde sus objetivos son facilitar la reutilización de código y componentes gracias a la herencia y encapsulamiento, que brindan los lenguajes orientados a objetos. Esto brinda grandes beneficios a las aplicaciones que utilizan esta arquitectura, como las siguientes:

- Aislamiento de la lógica del negocio en componentes separados reutilizables en otras capas o aplicaciones.
- Asignación de recursos a cada una de las capas por separado, brindando la posibilidad de desarrollarlas en paralelo.

Pero la principal ventaja que tiene esa arquitectura es que puede cambiarse la capa de presentación o incluso llevarse a una nueva plataforma sin que cambie la capa del negocio. Además el acceso a datos

también puede ser trasladado a una base de datos distinta, a otro modelo de base de datos o a otro servidor sin influir en la capa de negocio. La propia capa de negocio también puede a su vez ser reemplazada sin influir estos cambios significativamente en las capas de datos y presentación.

Dentro de esta arquitectura, la capa de acceso a datos juega un papel muy importante ya que es la encargada de persistir las entidades que se manejan en el negocio, la recuperación de los datos almacenados y la actualización.

1.1 Aplicaciones empresariales y el acceso a datos

Actualmente se ha evidenciado un gran uso de las aplicaciones para desarrollar sistemas automatizados de apoyo a la gestión empresarial. Las empresas se han beneficiado enormemente de las ventajas de comunicación de Internet; ofreciendo servicios útiles a sus clientes, proveedores, agentes comerciales y usuarios en general, tanto internos como externos. Para lograr estos objetivos, los desarrolladores de software deben apoyarse un una lógica de negocio escalable, robusta y reutilizable. Durante los últimos años la programación orientada a objetos ha emergido como la principal metodología para el desarrollo de software que cumple con estos requisitos.

Como característica de las aplicaciones desarrolladas en los últimos tiempos, se evidencia el uso de lenguajes de programación orientada a objetos, los cuales han ayudado a que los sistemas de gran tamaño sean fáciles de entender, de fácil depuración y más rápidos a la hora de introducir nuevos cambios; otra característica de las aplicaciones empresariales actuales es que se relacionan con bases de datos, principalmente relacionales; son altamente escalables, pueden soportar más carga de trabajo sin necesidad de modificar el software solamente con desplegar mas servidores; no dejan de prestar servicio, por lo que siempre están disponibles a los clientes; no todos los usuarios tienen acceso a la misma información, por lo que la seguridad es una característica importante; tienen una alta integración porque permiten integrar otras aplicaciones construidas con otras tecnologías de desarrollo. La arquitectura que se utiliza principalmente para su desarrollo es la multicapa, donde se evidencia una separación clara entre sus capas, y donde la capa de acceso a datos siempre está presente.(1)

En el ámbito internacional, los catálogos online de productos son una muestra del beneficio que proveen las aplicaciones empresariales; estos permiten transformar la empresa en una vidriera virtual rompiendo la frontera física existente y permitiendo la introducción en nuevos mercados. En este tipo de aplicación, el usuario puede administrar completamente el catálogo de productos en el sitio de su empresa; puede

agregar nuevos productos, modificar los existentes o dar de baja aquellos que ya no necesite que estén publicados. Otra de sus funcionalidades es la de poder manejar categorías y agrupar los productos. También tienen un sistema de carrito de compra donde cada usuario puede hacer sus pedidos y enviar el mismo por correo electrónico. Ejemplos de catálogos online tenemos a Mayorco (www.mayorco.com) y CHITA'S (www.chitasweb.com.ar).

Otra de las aplicaciones de más amplio uso en la actualidad, son las aplicaciones Web para la automatización de CallCenters, llamados también Web-CallCenters; que no es más que la posibilidad que se le brinda a las empresas de poder atender a sus clientes de una forma no presencial, vía telefónica, permitiendo reducir costos por no tener que acondicionar grandes espacios para atención presencial, reduciendo además tiempos de atención a clientes y permitiendo automatizar parte de las actividades de una manera mucho más fácil(2). Ejemplo de la puesta en práctica de este tipo de tipo de aplicación lo tenemos en la empresa española Iberrail, que gestiona varias agencias de viajes y ofrece productos y servicios profesionales, entre los que se destacan sus ofertas hoteleras. Hasta hace poco tiempo, esta empresa operaba de la forma tradicional, las agencias de viajes contactaban con el CallCenter y las confirmaciones de reservación se hacían vía fax. Con el objetivo de adquirir una ventaja competitiva y lograr una mayor expansión hacia otros mercados, esta empresa decidió modernizar su CallCenter. El objetivo principal de este proyecto era abrir un nuevo canal de venta que redujera los costos y sirviera para atraer nuevos clientes. El proyecto fue desarrollado en JAVA JSP por ser un entorno Web y utilizando un servidor Web Linux; esta elección fue tomada teniendo en cuenta que esta plataforma permitía un mejor aprovechamiento de la infraestructura existente en la empresa y permitía un mejor aprovechamiento de la tecnología existente; fue necesario un servidor de aplicaciones sólido, rápido y estable que permitiese evolucionar y migrar hacia posibles plataformas futuras limpiamente y sin esfuerzo; el servidor Web Linux fue la elección que les brindo esta opción, cumpliendo estos requisitos. El proyecto desarrollado aloja dentro del servidor Web, un servidor de bases de datos PostgreSQL, esta elección fue hecha tomando en cuenta que durante las pruebas de desarrollo, este servidor de bases de datos les demostró su fiabilidad y su factibilidad a migrar a plataformas futuras. En los primeros 3 meses de funcionamiento, el proyecto piloto de la empresa ha marcado unos resultados muy superiores a los previstos en principio.(3)

Otro ejemplo relevante lo muestra WebHispania, empresa española de servicios informáticos, la cual ha desarrollado un sistema automático basado en Web Services para CallCenters, llamado NetCenter, este sistema permite el acceso a bases de datos MySQL, MS SQL Server y Oracle, gestionando toda la

información, comunicación y posibilitando la gestión de todo tipo de solicitudes, ya sean de información, administrativas y facturación. Es un sistema multicapa que permite la completa integración de operadores, agentes y responsables, guardando en la base de datos todas las solicitudes, comunicaciones y documentos para su posterior acceso y consulta.

En nuestra universidad son varios los proyectos productivos que tienen como tarea desarrollar aplicaciones Web, en este caso podemos mencionar al proyecto productivo Prisiones, un proyecto de desarrollo Web que utiliza el lenguaje de programación Java.

Para el acceso a datos, han utilizado JDBC (Java Database Connectivity), un API (Application Program Interface) que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java, que funciona independientemente del sistema operativo donde se ejecute o de la base de datos a la cual se accede.
Este proyecto productivo, teniendo en cuenta los requerimientos de su aplicación, decidió utilizar una herramienta de mapeo objeto relacional para optimizar el acceso a datos, la elección hecha fue Hibernate teniendo en cuentas las facilidades que les brinda.

Otro punto que se tuvo en cuenta a la hora de definir la arquitectura de acceso a datos de este proyecto fue la utilización del framework Spring, ya que este les permite simplificar el código de Hibernate considerablemente. La integración entre estos frameworks se beneficia de la misma infraestructura genérica de transacciones y la jerarquía de excepciones DAO (Data Access Object) que Spring usa para JDBC, haciendo fácil la integración de diferentes metodologías de persistencia en la aplicación de ser necesario.

La arquitectura propuesta extiende básicamente los beneficios de esta integración, logrando una configuración completa y escalable, aplicando los conceptos de estos dos frameworks a la aplicación en particular. Se logra, además, una reducción considerable de código en la capa de acceso a datos debido a la reutilización de funcionalidad común para todos los Objetos de Acceso a Datos y la creación de clases de utilidad. Otra característica de la arquitectura propuesta es la definición del flujo de trabajo que debe seguir el equipo de desarrollo para la capa persistente de la aplicación.

1.2 Capas de Acceso a Datos. Tendencias actuales

La capa de acceso a datos es la parte lógica, dentro de la arquitectura de sistemas multicapa que se encarga de implementar aquellas clases encargadas de realizar todas las operaciones con las bases de datos dentro de la aplicación. Contiene las clases que interactúan con la base de datos; y estas clases se implementan como una necesidad de mantener la cohesión y la alta especialización que ayudan a reducir la dependencia entre las otras clases y demás capas de la arquitectura, además de maximizar la flexibilidad de la aplicación.(4)

La evolución que han tenido las nuevas tecnologías ha cambiado significativamente la manera en que se desarrollan las aplicaciones. Las que se construían hace 10-20 años, con arquitecturas muy simples, de una sola capa y apoyándose de un solo gestor de bases de datos relacional, deben ahora conectarse con otros sistemas, producir y consumir datos desde las más disímiles fuentes. Los servicios de datos de más alto nivel, como la generación de informes son cada vez más usados hoy en día.(5)

En la actualidad, el uso de las bases de datos y de las capas de acceso a datos que se encargan de gestionar la comunicación con ellas, resultan un instrumento de información muy valioso y que puede ser aprovechado efectivamente en la generación de ventas y utilidades. Tener y administrar bases de datos con clientes, implica un problema de información, el cual genera consideraciones de almacenamiento, seguridad y uso. Ante estos problemas aparecen procesos y tecnologías nuevas que buscan suplir las necesidades de manejo de información en las empresas.

Para solucionar estos problemas de conectividad y accesibilidad a disímiles fuentes de datos como servidores de bases de datos relacionales y ficheros XML, han surgido nuevas herramientas y tecnologías como el mapeo objeto relacional, más conocido por su nombre en inglés, Object-Relational Mapping, que es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en la base de datos; con esta nueva técnica, se logra que el desarrollador se independice y se abstraiga completamente del tipo de gestor de bases de datos. Esta tendencia o forma de trabajar tiene el inconveniente de no poder aprovechar la potencia de los motores de base de datos, dado que para realizar consultas entre conjuntos de entidades o tablas es necesario

hacerlo desde el lenguaje de programación, puesto que se carece de utilidades como la sentencia JOIN de SQL.(6)

1.3.1 Herramientas de mapeo Objeto-Relacional

Una herramienta de mapeo objeto relacional, es un framework que propone una nueva forma de modelar los datos, y que permite solucionar la diferencia que existe entre los paradigmas de la Programación orientada a objetos y el modelo Relacional. El modelo Relacional trata con relaciones y conjuntos por lo cual tiende a ser de carácter matemático, mientras que el modelo de la Programación orientada a objetos trata con objetos, su carácter y las asociaciones entre ellos; los problemas entre estos dos modelos surgen en el momento de acceder a la fuentes de datos y tratar de almacenar los objetos obtenidos de la consulta; entre estos problemas podemos citar:

- *Reglas de Acceso*. En el modelo relacional los atributos pueden ser accedidos y modificados a través de operadores relacionales predefinidos, mientras que en el modelo orientado a objetos, se permite que cada clase defina la forma en que serán alterados sus atributos.
- *Ataduras de Esquemas.* Los objetos del modelo orientado a objetos, no siguen ningún esquema puesto que son definidos por el programador, mientras que las tablas en el modelo relacional, deben seguir el esquema entidad-relación.
- *Identificador único.* Las llaves primarias de una fila tienen generalmente una forma de poder representarse como texto visible, mientras que los objetos no requieren un identificador único externamente visible.
- *Estructura y Comportamiento*. El modelo orientado a objetos se concentra primordialmente en asegurar que la estructura del código es legible, reutilizable y segura, mientras que el modelo relacional enfatiza en el comportamiento que el sistema tendrá una vez en producción, sea eficiente, adaptable y rápido. El modelo relacional enfatiza que la forma en que los usuarios finales perciben el comportamiento del sistema es mucho más importante.

Las herramientas de mapeo objeto relacional se encargan no solo de manejar y plantear una solución a las diferencias expuestas, sino que además buscan reducir susceptiblemente el código necesario para llevar a cabo las operaciones de persistencia y recuperación de objetos. Proporcionan además interfaces más simples para el manejo de objetos a través de su propio lenguaje de consulta, y proveen al

programador de configuraciones que le permiten optimizar los tiempos de respuesta en sus correspondientes aplicaciones.

Existen varias herramientas de mapeo objeto relacional que se usan constantemente en el desarrollo Web, como por ejemplo Hibernate, Oracle Toplink y JDO; estas herramientas han evolucionado desde su aparición en los años 70 y su uso se ha expandido entre los distintos programadores que buscan una solución para la persistencia de datos, cuando se manejan grandes cantidades de información; son por tanto un tema actual en la programación Web y proponen un modelo que permite separar aún más la lógica del negocio de los objetos en sí mismos, por tanto se considera que este tema puede aportar mucho para posteriores desarrollos, en la medida que fomenta un nuevo estilo de trabajo, respaldado por grandes empresas como Oracle o Sun, y a la vez por iniciativas de código libre como Hibernate, que actualmente es la herramienta de mapeo objeto relacional que más aceptación y mas reputación ha ganado en el mundo del desarrollo de software. Se ha consolidado como el producto de código libre líder en el mercado gracias a sus prestaciones, documentación y estabilidad. Hibernate ha logrado esta gran aceptación en el mundo del desarrollo de aplicaciones gracias a que:

- Es probablemente la herramienta de mapeo objeto relacional más popular en el 2005, es open-source, lanzado bajo la licencia GNU.
- Brinda persistencia para Java Beans, que no son más que clases java con los métodos de acceso sets y gets y un constructor vacío, es decir las clases persistentes no necesitan implementar ninguna interfaz especifica.
- Brinda soporte para modelos de negocios bien granulados, por ejemplo: Tipos de datos definidos por el usuario, relaciones entre objetos o colecciones de objetos, herencia, tipos de datos compuestos, identificadores compuestos, etc.
- Es posible utilizar la herramienta en entornos manejados, o en entornos no manejados (aplicaciones de escritorio, de consola, etc).
- Los ficheros de mapeo de la herramienta son flexibles e intuitivos, lo que aumenta su legibilidad y por ende la productividad de los programadores.
- Soporta diferentes estrategias para mejorar el rendimiento de las aplicaciones, por ejemplo: soporta diferentes estrategias para cargar objetos y relaciones como la carga lenta de colecciones, permite monitorear el rendimiento de la capa de persistencia, entre otras.

- Tiene integrado un poderoso lenguaje de consulta, HQL (Hibernate Query Language) basado en SQL (Structured Query Language) pero con una semántica orientada a objetos.
- Existen numerosas herramientas para el desarrollo con Hibernate, lo que indiscutiblemente ahorra una cantidad considerable de tiempo durante el desarrollo de la aplicación.

1.3.2 Bases de Datos Orientadas a Objetos

Actualmente, ha surgido una nueva tendencia para optimizar el almacenamiento, gestión y acceso a datos utilizado por las aplicaciones; el uso de bases de datos orientadas a objetos, OODB (*Object Oriented Data Base*) y los gestores de bases de datos orientados a objetos, OODBMS (*Object Oriented Data Base Management System).*

Las bases de datos orientadas a objetos son aquellas cuyo modelo de datos está orientado a objetos, almacenan y recuperan objetos en los que se almacena su estado y comportamiento. Su origen se debe a que en los modelos clásicos de datos, como el modelo relacional, existen problemas para representar cierta información, puesto que aunque permiten representar gran cantidad de datos, las operaciones que se pueden realizar con ellos son bastante simples. Las clases utilizadas en un determinado lenguaje de programación orientado a objetos son las mismas clases que serán utilizadas en una base de datos que utiliza el mismo modelo; de tal manera, que no es necesaria una transformación del modelo de objetos, como la utilización de una herramienta de mapeo objeto relacional, para ser utilizado por un gestor de bases datos orientado a objetos. Estos tipos de bases de datos surgen para evitar los problemas que se generan al tratar de representar información, aprovechar las ventajas del paradigma orientado a objetos en el campo de las bases de datos y para evitar transformaciones entre el modelo relacional y el modelo orientado a objetos.(7)

Mientras que en una base de datos relacional los datos a almacenar son representados en tablas, en una orientada a objetos los datos se almacenan como objetos; que no son más que entidades identificables unívocamente que describen tanto el estado como el comportamiento de una entidad del mundo real. El estado de un objeto es descrito mediante atributos mientras que su comportamiento es definido mediante métodos. Las características de una base de datos orientada a objetos son:

- La forma de identificar objetos es mediante un identificador único para cada objeto. Generalmente este identificador no es accesible ni modificable para el usuario, que es un modo de aumentar la integridad de entidades y la integridad referencial.

- Encapsulamiento: cada objeto contiene y define procedimientos o métodos y la interfaz mediante la cual se puede acceder a él y manipularlo. La mayoría de los gestores de bases de datos orientadas a objetos permiten el acceso directo a los atributos incluyendo operaciones definidas por el propio sistema gestor de bases de datos del mismo modelo, los cuales leen y modifican los atributos para evitar que el usuario tenga que implementar una cantidad considerable de métodos cuyo único propósito sea el de leer y escribir los atributos de un objeto. Generalmente, los sistemas gestores de bases de datos orientados a objetos, permiten al usuario especificar qué atributos y métodos son visibles en la interfaz del objeto y cuales pueden invocarse desde afuera.

Las ventajas que ofrecen las bases de datos orientadas a objetos son muchas, las más importantes son:

- Mayor capacidad de modelado. El modelado de datos orientado a objetos permite modelar de una manera mucho más fiel. Esto se debe a que un objeto permite encapsular tanto un estado como un comportamiento, puede almacenar todas las relaciones que tenga con otros objetos y que los objetos pueden agruparse para formar objetos complejos.
- Ampliación. Esto se debe a que se pueden construir nuevos tipos de datos a partir de los ya existentes. Permite la agrupación de propiedades comunes de diversas clases e incluirlas en una superclase y permite además la reusabilidad de clases, lo que repercute en una mayor facilidad de mantenimiento y un menor tiempo de desarrollo.
- Mayores prestaciones. Los sistemas gestores de bases de datos orientadas a objetos proporcionan mejoras significativas de rendimiento con respecto a los relacionales.

Estas ventajas influyen significativamente a la hora de vincular un gestor de bases de datos orientadas a objetos con aplicaciones desarrolladas en lenguajes de programación que utilizan el mismo modelo, como el hecho de no utilizar herramientas de mapeo objeto relacional para el acceso a datos.

1.3.3 Bases de Datos Objeto Relacionales

Otra tendencia actual para lograr optimizar el acceso a datos, es el uso de las Bases de Datos Objeto Relacionales, ORDB (*Object Relational Data Base*) y los sistemas gestores de Bases de Datos Objeto Relacionales, ORDBMS (*Object Relational Data Base Managment System*). El término Base de Datos Objeto Relacional, se usa para describir una base de datos que ha evolucionado desde el modelo

relacional hacia otra más amplia que incorpora conceptos del paradigma orientado a objetos. Por tanto, un Sistema de Gestión Objeto-Relacional contiene ambas tecnologías: relacional y orientada a objetos.

En los últimos años, las grandes corporaciones de software del mundo como Microsoft, Oracle, IBM y Sybase han incursionado en el desarrollo de sistemas gestores de bases de datos objeto-relacionales, llevando al mercado nuevas versiones de sus viejos productos con esta nueva tecnología. Recientemente, muchas empresas han estado usando los sistemas de bases de datos relacionales para aplicaciones no tradicionales debido a la demanda de almacenamiento de imágenes y objetos multimedia en la base de datos, consecuentemente, estos objetos y las operaciones que implican su recuperación, se han vuelto mucho más complejas, ejemplos de datos complejos son imágenes, objetos multimedia y objetos 3D. Debido a esto muchas personas piensan que el futuro de los gestores de bases de datos serán sistemas puramente orientados a objetos. Inicialmente esta idea resulto muy prometedora, sin embargo estos sistemas han sido incapaces de llenar las expectativas, es por esto que surge la nueva tecnología que encierra los dos conceptos: el orientado a objetos y el relacional.

La tecnología objeto-relacional brinda grandes ventajas a la hora de desarrollar aplicaciones. Una idea básica de estas ventajas es que el usuario pueda crear sus propios tipos de datos, para ser utilizados en aquella tecnología que permita la implementación de tipos de datos predefinidos. Además, los ORDBMS permiten crear métodos para esos tipos de datos. Con ello, este tipo de ORDBMS hace posible la creación de funciones usando tipos de datos definidos por el usuario, lo que proporciona flexibilidad y seguridad.

El principal objetivo que se tiene en cuenta a la hora de diseñar un ORDBMS es lograr mantener los beneficios de ambos modelos, el orientado a objetos y el relacional; como son la escalabilidad y el soporte para tipos de datos complejos. Los ORDBMS emplean un modelo de datos que trata de incorporar las características del modelo orientado a objetos a los tradicionales sistemas gestores relacionales.

Los sistemas gestores basados en esta tecnología permiten importantes mejoras en muchos aspectos con respecto a los gestores relacionales tradicionales; estos sistemas gestionan tipos de datos complejos con un esfuerzo mínimo y albergan parte de la aplicación en el servidor de base de datos. Permiten almacenar datos complejos de una aplicación dentro de la base de datos sin necesidad de forzar los tipos de datos tradicionales. Son compatibles en sentido ascendente con las bases de datos relacionales tradicionales,

tan familiares a multitud de usuarios. Es decir, se pueden pasar las aplicaciones sobre bases de datos relacionales al nuevo modelo sin tener que rescribirlas. Adicionalmente, se pueden ir adaptando las aplicaciones y bases de datos para que utilicen las funciones orientadas a objetos.

Durante muchos años ha habido innumerables debates sobre si la siguiente generación de la tecnología de bases de datos sería exclusivamente orientada a objetos o basada en SQL con extensiones orientadas a objetos. Los partidarios de esta segunda opción argumentan varias razones para demostrar que el modelo objeto relacional dominará:

- Las bases de datos objeto-relacionales son compatibles en sentido ascendente con las bases de datos relacionales actuales. Esto les brinda la posibilidad a los usuarios y desarrolladores de migrar su gestor de bases de datos de forma sencilla y sin complicaciones.
- Las primeras bases de datos orientadas a objetos puras no admitían las capacidades estándar de consulta de las bases de datos SQL: esto puede ser un problema importante cuando surgen necesidades que no se han previsto en el diseño original. En realidad, una de las razones principales por las cuales las empresas adoptaron las bases de datos relacionales tan rápidamente fue su capacidad para crear consultas y manejar sentencias SQL tradicionales.

1.4 Importancia de la capa de Acceso a Datos en las aplicaciones

Con el auge que han tenido las aplicaciones Web en los últimos años, cada vez son más las necesidades de los clientes de tener soporte para la información que manejan estas aplicaciones, como gestores de bases de datos; actualmente, los equipos de desarrollo, en su mayoría, escogen arquitecturas multicapa para el desarrollo de aplicaciones Web, de modo que utilicen una capa de acceso a datos que encapsule todas las interacciones con la base de datos. Así, se simplifican las llamadas desde la capa de presentación, y permite encapsular la lógica de acceso a datos, siendo transparentes las operaciones de la capa de acceso a datos a la capa de lógica del negocio.

Esta es la capa desde donde se implementan los componentes que interactúan con la base de datos y donde se encapsula el acceso a datos con estos componentes de una manera fácil, para lograr un rendimiento óptimo. Esto permite que las demás capas, componentes e interfaces no interactúen directamente con el servidor de bases de datos y así excluir las complejidades del acceso a datos en el código fuente de estos. Las capas de acceso a datos, por lo general, se desarrollan para interactuar con

las bases de datos de la manera más rápida posible, con alta eficiencia y completamente transparente para las otras capas o componentes de la aplicación.

Las capas de acceso a datos juegan un importante papel dentro de la arquitectura de las aplicaciones, ellas proveen muchas ventajas claves entre las que tenemos:

- Separación entre la capa de negocio y los gestores de bases de datos. Los componentes de la capa de acceso a datos implementan aquellas funciones especializadas en como acceder a los diferentes servidores de bases de datos y brindar esos datos, en un formato adecuado, a la capa de negocio de la aplicación. Este nivel de aislamiento protege a los componentes de las otras capas, específicamente de la capa de negocio, de los cambios potenciales que puedan ocurrir en el servidor de bases de datos.
- Mejoras en el mantenimiento de la aplicación. Encapsula toda la gestión de acceso a datos en una sola capa, que ofrece la ventaja de reutilizar componentes, reduciendo la cantidad de código fuente a desarrollar y mantener.

1.5 Necesidad de la capa de Acceso a Datos en el proyecto TeleBanca

Teniendo en cuenta la arquitectura elegida para el desarrollo del proyecto TeleBanca, que es una arquitectura multicapa, se hace necesario el uso de una capa de acceso a datos por la importancia antes mencionada. Este proyecto, que incluye una estrecha relación con una base de datos bastante amplia en información y entidades, tiene la gran necesidad de gestionar de forma eficiente, rápida y segura esta información crítica para su funcionamiento. Si se prescindiera de este subsistema, se tendría que implementar en otras capas, como la de negocio o presentación, las funcionalidades exclusivas de una capa de acceso a datos, esto trae como consecuencia una enorme pérdida de tiempo de desarrollo y esfuerzo, además de generar innumerables complicaciones a la hora de darle mantenimiento o realizar actualizaciones en el código de la aplicación.

1.6 Lenguajes, herramientas y tecnologías utilizadas

1.6.1 Lenguajes y tecnologías

CSharp (C#)

Es el lenguaje de programación orientado a objetos que Microsoft ha desarrollado para su plataforma .NET. Está construido especialmente para adaptarse al framework y aprovechar al máximo todas sus

características. Fue diseñado desde 0 con vistas a ser utilizado en .NET, por lo que no cuenta con elementos heredados de versiones anteriores e innecesarias en la plataforma .NET. El lenguaje combina los mejores elementos de múltiples lenguajes de amplia difusión como C++, Java, Visual Basic o Delphi. De hecho, su creador Anders Heljsberg fue también el creador de muchos otros lenguajes y entornos como Turbo Pascal, Delphi o Visual J++. La idea principal detrás del lenguaje es combinar la potencia de lenguajes como C++ con la sencillez de lenguajes como Visual Basic, y que además la migración a este lenguaje por los programadores de C, C++ y Java sea lo más rápida posible.

Entre sus características más notables tenemos su sencillez, eficacia y su seguridad a la hora de manipular tipos de datos. Permite desarrollar aplicaciones rápidamente y mantiene la expresividad y elegancia de los lenguajes de tipo C. Visual Studio .NET admite Visual C# con un editor de código completo, plantillas de proyecto, diseñadores, asistentes para código, un depurador eficaz y fácil de usar, además de otras herramientas. La biblioteca de clases .NET Framework ofrece acceso a una amplia gama de servicios de sistema operativo y a otras clases útiles y adecuadamente diseñadas que aceleran el ciclo de desarrollo de manera significativa.

C# es el único lenguaje que ha sido diseñado específicamente para ser utilizado en la plataforma .NET. Programar usando C# es mucho más sencillo e intuitivo que hacerlo con cualquiera de los otros lenguajes soportados por la plataforma. Por esta razón, se suele decir que es el lenguaje nativo de .NET. Entre las características de este lenguaje se encuentran:

- Sencillez: Elimina muchos elementos de otros lenguajes que son innecesarios en la plataforma .NET.
- Modernidad: Incorpora elementos útiles para el desarrollo de aplicaciones como nuevas instrucciones.
- Orientado a objetos: Es más puro en tanto que no admite ni funciones, ni variables globales, sino que todo el código y datos han de definirse dentro de definiciones de tipos de datos, lo que reduce problemas por conflictos de nombres y facilita la legibilidad del código. Soporta encapsulamiento, herencia y polimorfismo, que son las características esenciales del modelo orientado a objetos.
- Gestión automática de memoria: Tiene a su disposición el recolector de basura del CLR (Common Language Runtime). No es necesario incluir instrucciones de destrucción de objetos.

- Eficiente: Pueden marcarse regiones de código como inseguras y podrán usarse en ellas punteros, lo que puede resultar vital para situaciones donde se necesite una eficiencia y velocidad procesamiento muy grande.

Teniendo en cuanta estas características del lenguaje, y valorando las ventajas que brinda en comparación con otros lenguajes de la plataforma, los líderes del proyecto TeleBanca tomaron la decisión de usarlo como el lenguaje de desarrollo de la aplicación; además de ser el lenguaje nativo de la plataforma .NET y ser el que mejor explota sus funcionalidades.

SQL

SQL (Structured Query Lenguaje), Lenguaje de Consultas Estructurado, es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones sobre las mismas. Reúne las características del álgebra y el cálculo relacional permitiendo lanzar consultas con el fin de recuperar información de una base de datos de una forma sencilla.(8)

El SQL es un lenguaje de acceso a bases de datos que explota la flexibilidad y potencia de los sistemas relacionales permitiendo gran variedad de operaciones sobre los mismos. Es un lenguaje de alto nivel que gracias a su fuerte base teórica y su orientación al manejo de conjuntos de registros permite una alta productividad en codificación. Permite fundamentalmente dos modos de uso:

- Un uso interactivo, destinado principalmente a los usuarios finales avanzados u ocasionales, en el que las diversas sentencias SQL se escriben y ejecutan en línea de comandos, o un entorno semejante.
- Un uso integrado, destinado al uso por parte de los programadores dentro de programas escritos en cualquier lenguaje de programación anfitrión. En este caso el SQL asume el papel de sub-lenguaje de dato.

XML

XML (Extensible Markup Language), significa lenguaje de marcas generalizado. Es un lenguaje usado para estructurar información en un documento o en general en cualquier fichero que contenga texto, como por ejemplo ficheros de configuración de un programa o una tabla de datos. Ha ganado muchísima popularidad en los últimos años debido a ser un estándar abierto y libre. El XML fue propuesto en 1996, y

la primera especificación apareció en 1998. Desde entonces su uso ha tenido un crecimiento acelerado, que se espera que continúe durante los próximos años; hoy en día es un lenguaje por excelencia usado en casi todas las aplicaciones para representar datos en ficheros o para guardar datos en ficheros de configuración de aplicaciones.(9)

XML representa una manera más avanzada de representar la información, cuya principal novedad consiste en permitir compartir los datos con los que se trabaja a todos los niveles, por todas las aplicaciones y soportes. XML juega un papel muy importante en el mundo informático actual, ya que es el lenguaje que permitirá compartir la información de una manera segura, fiable y fácil.
Este lenguaje de etiquetas ofrece múltiples ventajas a la hora de representar información, entre ellas podemos mencionar:

- Es extensible, por lo que una vez diseñado el documento XML y puesto en producción, igual es posible extenderlo con la adición de nuevas etiquetas de manera de que los antiguos consumidores de la vieja versión todavía puedan entender el nuevo formato.
- Si un tercero decide usar un documento creado en XML, es sencillo entender su estructura y procesarlo. Mejora grandemente la compatibilidad entre aplicaciones.

Los desarrolladores del lenguaje tuvieron unos objetivos bien marcados a la hora de diseñarlo, como por ejemplo:

- Es directamente utilizable sobre Internet y aplicaciones Web.
- Es soportado por una amplia variedad de aplicaciones.
- Es prácticamente fácil implementar editores y procesadores de código XML.
- El número de características opcionales en XML es mínimo.
- Los documentos XML son fácilmente legibles por los desarrolladores y razonablemente claros.
- Los documentos XML son fáciles de crear.

Pero la principal característica del lenguaje XML es que es un estándar extensible, multiplataforma y basado en texto para la representación de información de una manera estructurada y legible; que ha sido un lenguaje clave en el desarrollo de Servicios Web.

Microsoft .NET

Microsoft .NET es el conjunto de nuevas tecnologías en las que Microsoft ha estado trabajando durante los últimos años, cuyos principales objetivos fueron inicialmente:

- Mejorar sus sistemas operativos.
- Mejorar su modelo de componentes COM.
- Obtener un entorno específicamente diseñado para el desarrollo y ejecución del software, en forma de servicios que puedan ser tanto publicados, como accedidos a través de Internet de forma independiente del lenguaje de programación, modelo de objetos, sistema operativo y hardware utilizados, tanto para desarrollarlos como para publicarlos.

Éste entorno es lo que se denomina la plataforma .NET, y los servicios antes mencionados son a los que se denomina servicios Web. Para el desarrollo y ejecución de aplicaciones en este nuevo entorno tecnológico, Microsoft proporciona el conjunto de herramientas conocido .NET Framework SDK, que incluye compiladores de lenguajes como C#, Visual Basic.NET, C++ y JScript.NET específicamente diseñados para crear aplicaciones para este framework.

El corazón de la plataforma.NET es el CLR (Common Language Runtime), que es una aplicación similar a una máquina virtual que se encarga de gestionar la ejecución de las aplicaciones escritas para ella. A estas aplicaciones les ofrece numerosos servicios que facilitan su desarrollo y mantenimiento y favorecen su fiabilidad y seguridad. Entre estos servicios los más importantes son:

- Modelo de programación consistente y sencillo, completamente orientado a objetos.
- Eliminación del temido problema de compatibilidad entre las DLL.
- Ejecución multiplataforma.
- Ejecución multi-lenguaje, hasta el punto de que es posible hacer cosas como capturar en un programa escrito en C# una excepción escrita en Visual Basic.NET que a su vez hereda de un tipo de excepción escrita en Cobol.NET.
- Recolección de basura.
- Aislante de memoria entre procesos y comprobaciones automáticas de seguridad de tipos en las conversiones.
- Soporte multi-hilo.

- Gestión del acceso a objetos remotos que permite el desarrollo de aplicaciones distribuidas de manera transparente a la ubicación real de cada uno de los objetos utilizados en las mismas.
- Seguridad avanzada, hasta el punto de que es posible limitar los permisos de ejecución del código en función de su procedencia (Internet, red local, CD-ROM, etc.), el usuario que lo ejecuta o la empresa que lo creó.
- Interoperabilidad con código preexistente, de manera que es posible utilizar con facilidad cualquier librería de funciones u objetos COM y COM+ creados con anterioridad a la aparición de la plataforma .NET.
- Adecuación automática de la eficiencia de las aplicaciones a las características concretas de cada máquina donde se vaya a ejecutar.

ADO.NET

ADO.NET es lo más reciente en una extensa línea de tecnologías de acceso a bases de datos que comenzó hace varios años con la interfaz de programación de aplicaciones (API) de la conectividad abierta de base de datos, ODBC (Open Database Connectivity). Se ha diseñado siguiendo específicamente unas directrices más generales y menos orientadas a la base de datos que reúne todas las clases que permiten el manejo de datos. Estas clases representan los objetos que contienen datos y que muestran las capacidades normales de las bases de datos: índices, ordenación, vistas. Aunque ADO.NET es la solución definitiva para las aplicaciones de base de datos de la tecnología .NET, destaca por un diseño global que no se centra tanto en las bases de datos como su antecesor el modelo ADO.

ADO.NET proporciona acceso coherente a orígenes de datos como Microsoft SQL Server, así como a otros expuestos mediante OLE DB y XML. Las aplicaciones para usuarios que comparten datos pueden utilizar esta nueva tecnología para conectar a estos orígenes de datos y recuperar, manipular y actualizar los datos. Otra de sus características es que separa limpiamente el acceso a datos de la manipulación de datos y crea componentes que pueden ser usados conjuntamente o por separado. ADO.NET incluye proveedores de datos de .NET Framework para conectarse a una base de datos, ejecutar comandos y recuperar resultados. Esos resultados se procesan directamente o se colocan en un objeto DataSet con el fin de exponerlos al usuario para un propósito específico, junto con datos de varios orígenes, o de utilizarlos de forma remota entre niveles. El objeto DataSet de ADO.NET también puede utilizarse

independientemente de un proveedor de datos de .NET Framework para administrar datos que son locales de la aplicación o que proceden de un origen XML.(10)

1.6.2 Herramientas utilizadas

Microsoft SQL Server 2000

Microsoft SQL Server 2000, es un sistema RDBMS (Relational Database Management System), este potente gestor de bases de datos de alto rendimiento, es capaz de soportar millones de registros por tabla con un interfaz intuitiva y con herramientas de desarrollo integradas como Visual Studio, además, incorpora un modelo de objetos totalmente programable, SQL-DMO, con el que es posible desarrollar cualquier aplicación que manipule componentes de SQL Server. Que sea muy intuitivo en su administración o instalación, no significa que sea fácil, una mala instalación, una base de datos mal creada, mal diseñada o una mala administración puede hacer que las aplicaciones tengan un rendimiento escaso o deficiente. Transact-SQL es el lenguaje que emplea este gestor para las peticiones entre el cliente y el servidor. Es un lenguaje exclusivo de SQL Server, pero basado en el lenguaje SQL estándar, utilizado por casi todos los tipos de bases de datos relacionales que existen. SQL Server otorga a los administradores una herramienta potencialmente robusta, provista de las funcionalidades suficientes que le permiten mantener un óptimo nivel de seguridad en la utilización de los recursos del sistema y de la base de datos.

Microsoft Visual Studio .NET 2005

Microsoft Visual Studio .net 2005 es un IDE (Integrated Development Environment), un entorno de desarrollo integrado compuesto por un conjunto de herramientas para el desarrollo de aplicaciones y la programación en uno o varios lenguajes de programación. Es la última versión de Microsoft de este ambiente de desarrollo sacado al mercado en el 2002.

Esta última versión del ambiente de desarrollo, integra nuevas facilidades para el programador como soportar los nuevos lenguajes C# y Visual Basic .NET; desarrollados igualmente por Microsoft. Fue

desarrollado para la implementación de aplicaciones Windows, aplicaciones Web, utilizando la nueva tecnología de Microsoft ASP. NET y los servicios Web XML. Entre las nuevas ventajas que ofrece, se destaca su integración con el .NET Framework 2.0. Provee además un conjunto de herramientas que ofrecen grandes beneficios para el desarrollo individual y para los equipos de desarrollo de software.
Las ventajas que ofrece con relación a las versiones anteriores son entre otras:

- Mayor productividad de los desarrolladores y obtención de resultados mucho más rápido.
- Desarrollo de aplicaciones Windows, aplicaciones Web dinámicas y aplicaciones empresariales en un menor tiempo de desarrollo.
- Mayor eficiencia en la comunicación y colaboración dentro del equipo de desarrollo.

Entre las novedades que tiene, las más destacadas son:

- Depurador configurable para ignorar código fuente de terceros, y que solo procese el código fuente que nos interesa.
- Soporte para el SQL Server 2005. Brinda la posibilidad de desarrollar componentes para el gestor de bases de datos SQL Server 2005, incluyendo la administración de la base de datos utilizada.
- Permite la definición parcial de clases, es decir, una misma clase en más de un fichero de código fuente.
- Permite el uso de plantillas en el desarrollo Web (Master Pages), se personaliza una página como plantilla y luego puede ser aplicado ese formato a las páginas de la solución Web.
- Nueva herramienta de configuración que permite la edición del fichero de configuración WebConfig, que permite la configuración fácil de las aplicaciones Web sin la necesidad de editar directamente el archivo XML de configuración.

Este entorno fue seleccionado como la herramienta de desarrollo del proyecto productivo TeleBanca por las grandes ventajas que ofrece en relación con otros entornos de desarrollo, además de ofrecer una mejor integración y facilidad de utilización de otras herramientas que el cliente tenía en su infraestructura, como servidores de bases de datos SQL Server 2000. Otro aspecto que se tuvo en cuenta para su elección fue que el cliente estuvo de acuerdo en seleccionarlo para el desarrollo de la aplicación.

1.6.3 Lenguajes de modelado

UML

El Lenguaje Unificado de Modelado UML (Unified Modeling Language) es la herramienta usada en la descripción y construcción de software reconocida por la industria como un estándar. Es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Se usa para entender, diseñar, configurar, mantener y controlar la información sobre los sistemas a construir.

UML capta la información sobre la estructura estática y el comportamiento dinámico de un sistema. Este lenguaje de modelado pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar. UML es una notación con la cual se construyen sistemas por medio de conceptos orientados a objetos. Esta prescribe un conjunto de notaciones y diagramas estándares, y describe la semántica esencial de lo que estos diagramas y símbolos significan.

Las características más generales de UML son:

- Tecnología de orientación a objetos.
- Viabilidad en la corrección de errores.
- Desarrollo incremental e iterativo.

Proceso Unificado de Rational (RUP)

RUP es un proceso de desarrollo de software de forma tal que se asignan tareas y responsabilidades cuyos objetivos son asegurar la producción de software de calidad dentro de plazos y presupuestos predecibles. Dirigido por casos de uso, centrado en la arquitectura, iterativo (mini-proyectos) e incremental (versiones).

Conclusiones

En este capítulo se han expresado las tendencias en el mundo informático actual hacia el desarrollo de nuevos componentes y herramientas que brindan soporte a la hora de realizar el acceso a datos en los proyectos de desarrollo de software, como las herramientas de mapeo objeto relacional y los gestores de bases de datos orientados a objetos y objeto relacionales. Se han expresado las principales características de los lenguajes y herramientas usadas en el proyecto productivo TeleBanca, así como las tecnologías en las cuales se ha apoyado el equipo de desarrollo. Se ha valorado el estado actual de aplicaciones similares a nivel internacional, nacional y en la universidad.

Valorando las grandes ventajas que brinda la plataforma de desarrollo de Microsoft, y teniendo en cuenta las potencialidades del ambiente de desarrollo Visual Studio .NET, se escogen estas herramientas y tecnologías como el soporte para desarrollar una aplicación Web eficiente y que cumpla con todos los requisitos expresados por el cliente, ya que la combinación de estas tecnologías brindan la posibilidad de obtener, actualizar y gestionar toda la información que utiliza el proyecto de una forma eficiente y rápida.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

Introducción

En este capítulo se describe el objeto de estudio, el problema y la situación problémica; se hace un análisis crítico de cómo se ejecutan actualmente estos procesos, las causas que originan la situación problémica y las consecuencias, así como se realiza la propuesta del sistema y se analizan los requerimientos funcionales y no funcionales. Se describen los procesos de negocio que serán objeto de automatización y se describen los sistemas automatizados que existen en la empresa para la que se desarrolla el sistema.

2.1 Objeto de estudio

En la actualidad, la mayoría de las empresas utilizan grandes cantidades de información y cada vez es mayor su necesidad de manipular y gestionar estos datos. Las aplicaciones informáticas son la solución que le han brindado a estas empresas las facilidades de gestionar la información de una manera fácil, rápida y segura; combinando la integración empresarial, automatizando los procesos de negocio de la empresa y los servicios que estas brindan.

Casi todas las aplicaciones dirigidas a gestionar o automatizar procesos de negocio en empresas y organizaciones, incluyen una relación estrecha con fuentes de datos con el objetivo de guardar aquella información útil y crítica de la empresa o la que esta necesita para su funcionamiento. En el caso especial de este proyecto, se estudia la necesidad de automatizar los procesos de pago de servicios a empresas y organismos, y la necesidad de un subsistema dentro de la aplicación para automatizar los procesos de acceso a fuentes de datos en el proyecto productivo TeleBanca.

2.1.1 Problema y situación problémica

Problema Científico

En las empresas cubanas de servicios no existe ningún sistema automatizado que controle, regule y registre los pagos de servicios.

Como desarrollar un subsistema automatizado para simplificar, flexibilizar y organizar el acceso a fuentes de datos mediante la implementación de una capa de Acceso a Datos en el proyecto productivo TeleBanca.

Situación problémica

En las empresas cubanas de servicios, con el rápido incremento de sus clientes cada año, se hace cada vez más necesario de un sistema automatizado que controle, regule y registre los pagos de estos servicios, así como brindar otros muy necesarios para aumentar el confort de sus clientes. Hoy en día, son los clientes los que tienen que personarse en las oficinas territoriales de cada empresa, o esperar que un trabajador de la empresa se persone en su residencia, para efectuar el pago de los servicios prestados mensualmente, son ejemplos los servicios telefónicos de ETECSA (Empresa de Telecomunicaciones de Cuba), servicios hidráulicos, pago de multas, y servicios de gas licuado. Esta gestión se hace en la mayoría de las empresas de forma manual, y muy pocas son las que se apoyan en sistemas automatizados primarios. Esto trae como consecuencia que la gestión de estos procesos se haga con muy poca eficiencia y ocurran frecuentes atrasos en los pagos de estos servicios, lo que en muchos casos ocasiona el retiro del servicio y las sanciones a clientes. Es por esto que surge una nueva entidad para gestionar de forma conjunta los procesos de pago de servicios, la empresa TeleBanca, cuyo objetivo es funcionar como una banca telefónica donde los clientes registrados pueden efectuar los pagos, de manera electrónica, a las empresas que le brindan sus servicios por medio de llamadas telefónicas, brindándole al cliente un mayor confort y la posibilidad de efectuar sus pagos en cualquier momento; esta es la razón del surgimiento del proyecto productivo TeleBanca, cuyo objetivo es desarrollar una solución Web automatizada para gestionar todos los procesos en la empresa TeleBanca.

En esta tarea, se necesita de un subsistema para manipular la transferencia de información y la comunicación con los gestores de bases de datos que estén asociados a la aplicación Web; este subsistema es la capa de Acceso a Datos y su ausencia trae como consecuencia que los desarrolladores del sistema deban implementar en otros subsistemas, las funcionalidades del acceso a fuentes de datos, empleando en esta actividad mucho tiempo de desarrollo y esfuerzo. Además, al tener que implementar estas funcionalidades exclusivas del acceso a fuentes de datos en otras capas, se afecta la independencia del sistema y la transparencia, por lo que cualquier cambio futuro en la implementación de estas funcionalidades implica cambiar gran cantidad de código en otras capas, lo que supone un gasto de esfuerzo adicional.

2.2 Objeto de automatización

Se desea automatizar los procesos de transferencia y gestión de información con el sistema gestor de bases de datos asociado al proyecto y automatizar además la comunicación y la transferencia de información con el Web Service del Banco; se desea automatizar también los procesos de elaboración de reportes personalizados para su posterior consulta por el personal autorizado.

Teniendo en cuenta los argumentos planteados hasta el momento, se propone este proyecto de análisis, diseño e implementación de un subsistema, en este caso la capa de acceso a datos del proyecto TeleBanca, como una solución a los problemas planteados anteriormente.

2.3 Información que se maneja

La información que se maneja son los datos de la Banca telefónica; nombre, dirección, correo electrónico, teléfono, entre otros; los datos de los bancos asociados a esta, identificador del banco, su nombre, dirección URL del Web Service de dicho Banco; se manejan los datos de los usuarios con acceso al sistema, identificador del usuario, su nombre, contraseña encriptada, rol al que pertenece en el sistema; se guardan los datos de los roles: identificador del rol y su descripción; se maneja la información referente a las funcionalidades: descripción, nombre en el menú del sistema, etc; se guarda y maneja demás la relación entre estos; se manejan y guardan las acciones que estos usuarios realizan dentro del sistema: identificador del usuario, funcionalidad, fecha en que realizo la acción en el sistema y descripción de la acción realizada; se gestionan además los datos de las sucursales de cada banco: el numero de la sucursal y su nombre; se maneja la información relacionada con los datos de las tarjetas emitidas por el banco a sus clientes: su identificador, número de pin, nombre del propietario, sus apellidos, numero de la sucursal a la que pertenece, fecha de impresión de la tarjeta, el estada en que se encuentra la tarjeta, entre otros datos; se gestiona la información referente a los lotes de las tarjetas, se guardan los datos de las transacciones realizadas: identificador de la transacción, traza, identificador del usuario, identificador de la tarjeta, importe pagado e identificador del servicio, entre otros datos; se maneja la información de los servicios que brinda la banca telefónica así como sus datos y relación entre estos, se maneja la información referente a las matrices impresas en las tarjetas y las por imprimir, se guarda y maneja la información referente a la configuración del sistema, los datos de cada transacción realizada así como los errores emitidos por el sistema.

2.4 Propuesta de sistema

Este subsistema ha sido diseñado para brindar al proyecto productivo TeleBanca, una solución que gestione los procesos de comunicación e intercambio de información con los gestores de bases de datos utilizados por el proyecto; además de automatizar la interacción y comunicación con los Web Services de los bancos asociados al proyecto.

La información crítica para el funcionamiento del sistema estará guardada en una base de datos: la configuración del sistema, los datos de los usuarios, los datos de las tarjetas, la información referente a los servicios que brinda la banca telefónica igualmente estarán guardados en dicha base de datos. A esta información solo tendrán acceso los usuarios autenticados en el sistema, de acuerdo a los roles y funcionalidades que tengan cada uno.

Para el acceso a la información almacenada en la base de datos, se han utilizado componentes que brindan todas las facilidades de recuperación, actualización y eliminación de datos. Teniendo en cuenta las características de la comunicación entre el subsistema y los gestores de bases de datos, el subsistema ha sido dividido en 4 módulos:

- **Módulo Gestión Bancaria.** Es el módulo encargado de gestionar las interacciones con los bancos asociados, es el que gestiona las transacciones realizadas en el sistema y por su importancia es el módulo de más peso dentro de la arquitectura del sistema.
- **Módulo Gestión Históricos.** Es el encargado de administrar aquellas tablas de la base de datos que guardan informaron que ha sido deshabilitada o eliminada de otras tablas de la base de datos.
- **Módulo Gestión Usuarios.** Es el módulo encargado de gestionar las autenticaciones de los usuarios en el sistema, de administrar y registrar las acciones que realizan estos usuarios una vez dentro del sistema. Es el encargado además de almacenar las funcionalidades que tienes los usuarios y sus roles.
- **Módulo Gestión y Mantenimiento de la Banca Telefónica.** Se encarga de almacenar la información de configuración del sistema y de registrar las acciones realizadas en el mantenimiento del sistema.

2.5 Especificaciones de requerimientos del sistema

2.5.1 Requerimientos funcionales

Los requisitos funcionales están dirigidos a guiar el proceso de desarrollo del software para lograr un mayor entendimiento entre los clientes y el equipo de desarrollo, o sea todas las ideas que los clientes, usuarios y miembros del equipo de proyecto tengan acerca de lo que debe hacer el sistema, deben ser analizadas como posibles requisitos funcionales.

Oficialmente están reconocidas como definiciones de requisitos funcionales las siguientes:

- Condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo.
- Condición o capacidad que tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente.
- Una representación documentada de una condición o capacidad como las antes mencionadas.(11)

Requisitos funcionales de la capa de acceso a datos:

1. Gestionar las acciones de los usuarios. El sistema debe ser capaz en todo momento de obtener, eliminar, insertar y actualizar la información sobre las acciones que hacen los usuarios en el sistema.
2. Gestionar la información Banca Telefónica. El sistema debe ser capaz de obtener, eliminar, insertar y actualizar los datos de la Banca Telefónica.
3. Gestionar la información de los Bancos. El sistema debe ser capaz de obtener, eliminar, insertar y actualizar los datos del Banco.
4. Gestionar datos de servicios. El sistema debe ser capaz de obtener, eliminar, insertar y actualizar los datos de los servicios que brinda la Banca Telefónica.
5. Gestionar la relación entre los servicios y datos. El sistema debe ser capaz de obtener, eliminar, insertar y actualizar la relación entre los datos que poseen los servicios y los servicios que brinda la Banca Telefónica.
6. Gestionar servicios. El sistema debe ser capaz de obtener, eliminar, insertar y actualizar los servicios que brinda la Banca Telefónica.

7. Gestionar configuración. El sistema debe ser capaz de obtener, eliminar, insertar y actualizar la configuración del sistema.
8. Gestionar entidad. El sistema debe ser capaz de obtener, eliminar, insertar y actualizar la información de las entidades del sistema.
9. Gestionar errores. El sistema debe ser capaz de obtener, eliminar, insertar y actualizar la información referente a los errores que genera el sistema.
10. Gestionar funcionalidades. El sistema debe ser capaz de obtener, eliminar, insertar y actualizar la información de las funcionalidades a las que pueden acceder los usuarios en el sistema.
11. Gestionar tarjetas eliminadas. El sistema debe ser capaz de obtener, eliminar, insertar y actualizar los datos de las tarjetas eliminadas del sistema.
12. Gestionar entidades eliminadas. El sistema debe ser capaz de obtener, eliminar, insertar y actualizar la información de las entidades eliminadas en el sistema.
13. Gestionar lotes. El sistema debe ser capaz de obtener, eliminar, insertar y actualizar los datos de los lotes del sistema.
14. Gestionar matriz. El sistema debe ser capaz de obtener, eliminar, insertar y actualizar los datos de las matrices de cada tarjeta guardadas en el sistema.
15. Gestionar notificación. El sistema debe ser capaz de obtener, eliminar, insertar y actualizar los datos de la notificación al sistema.
16. Gestionar pago complejo. El sistema debe ser capaz de obtener, eliminar, insertar y actualizar la información del los pagos complejos realizados en el sistema.
17. Gestionar país. El sistema debe ser capaz de obtener, eliminar, insertar y actualizar los datos de cada país al que pertenecen las tarjetas guardadas en el sistema.
18. Gestionar reclamación. El sistema debe ser capaz de obtener, eliminar, insertar y actualizar la información de las reclamaciones hechas por los clientes.
19. Gestionar roles. El sistema debe ser capaz de obtener, insertar, eliminar y actualizar los datos de los roles que juegan los usuarios dentro del sistema.
20. Gestionar sucursales. El sistema debe ser capaz de obtener, insertar, eliminar y actualizar los datos de las sucursales a las que pertenecen las tarjetas guardadas en el sistema.
21. Gestionar tarjeta. El sistema debe ser capaz de obtener, insertar, eliminar y actualizar la información de las tarjetas de los clientes.

22. Gestionar tema. El sistema debe ser capaz de obtener, insertar, eliminar y actualizar la información de los temas para el servicio de información a clientes.
23. Gestionar transacciones. El sistema debe ser capaz de obtener, insertar, eliminar y actualizar la información de las transacciones realizadas dentro del sistema.
24. Gestionar usuario. El sistema debe ser capaz de obtener, insertar, eliminar y actualizar la información de los usuarios del sistema.
25. Autenticar con Web Service. El sistema debe ser capaz de conectarse al Web Service del Banco y autenticarse con este.
26. Gestionar Conciliaciones. El sistema debe ser capaz de conectarse al Web Service del Banco y enviar la información referente a las conciliaciones en un periodo, es decir, enviar las conciliaciones de las tarjetas creadas, activas y canceladas.

2.5.2 Requerimientos no funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe cumplir. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable.

Requerimientos de Software

1. El sistema debe estar instalado en un sistema operativo Windows XP o superior; debe disponer además del framework .NET versión 2.0 o superior.

Requerimientos de Hardware

1. El sistema debe disponer como mínimo de una cantidad de memoria física de 256 MB y la velocidad mínima del procesador requerida es de 800 MHz

Requerimientos de Seguridad

1. Confidencialidad. La información referente a los usuarios, las transacciones, las entidades y todos los demás datos manejados por el sistema, deben estar protegidos contra accesos no autorizados y divulgación.
2. Integridad. La información manipulada por el sistema debe estar protegida contra la manipulación no autorizada, alteración o destrucción en todo momento.

3. Disponibilidad. El sistema debe garantizar en todo momento el acceso a los datos por parte de aquellos usuarios autorizados, la información debe estar disponible a tiempo completo para aquellos usuarios que así lo requieran.

2.6 Definición de casos de uso del sistema

Un caso de uso puede ser definido como una secuencia de acciones, incluyendo variaciones, que el sistema puede ejecutar y que produce un resultado observable de valor para un actor que interactúa con el sistema. A continuación la definición de los actores del sistema; las descripciones de los casos de uso, los diagramas de cada caso de uso, los casos de uso por paquetes y los casos de uso expandidos puede consultarse en el **Anexo 1**.

Actores	Justificación
Capa de negocio del sistema TeleBanca(BusinessLayer)	Es el encargado de iniciar las operaciones de selección, actualización e interacciones con el sistema.
Web Services de los Bancos asociados (WSB)	Es el Sistema externo del Banco, una aplicación Web que Brinda servicios que utiliza la capa de Acceso a Datos.

Conclusiones

A partir del análisis profundo de los problemas planteados, y la necesidad de acceder a la información guardada en los gestores y bases de datos; se evidencia la necesidad de automatizar los procesos de acceso y gestión de datos, así como de comunicación con aplicaciones externas por parte del proyecto productivo TeleBanca. En este capítulo se ha explicado la propuesta del subsistema, los requerimientos funcionales y no funcionales; se han definido los casos de uso del sistema, llegando a la conclusión de las grandes ventajas que el subsistema brindaría al proyecto productivo.

CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA

Introducción

En este capítulo se realizara el análisis y el diseño del subsistema. El propósito del análisis es lograr una comprensión más precisa de los requisitos, refinarlos y estructurarlos; analizar con profundidad los requisitos funcionales desde el punto de vista de los desarrolladores y proporcionar una visión general del sistema. En el análisis se profundiza además en el dominio de la aplicación para una mayor comprensión del problema y modelar la solución. En la etapa de diseño se toma la decisión de cómo va a ser el subsistema, o sea, se define una arquitectura robusta para el subsistema. A través de esta fase se adapta el diseño para que coincida con el entorno de implementación y los requisitos no funcionales, es decir resolver el problema del cómo se va a realizar el subsistema.

3.1 Análisis del sistema

3.1.1 Definición del modelo de análisis.

Un modelo conceptual es una representación de conceptos en un dominio del problema. Además de descomponer el espacio del problema en unidades comprensibles (conceptos), la creación de este modelo contribuye a esclarecer la terminología o nomenclatura del dominio. Puede verse como un modelo que define la comunicación entre los términos importantes y como se relacionan entre sí. En UML ilustramos el modelo conceptual como un diagrama de clases.

Modelo de clases del análisis

DIAGRAMA DE CLASES DE ANÁLISIS

Módulo Gestión Bancaria

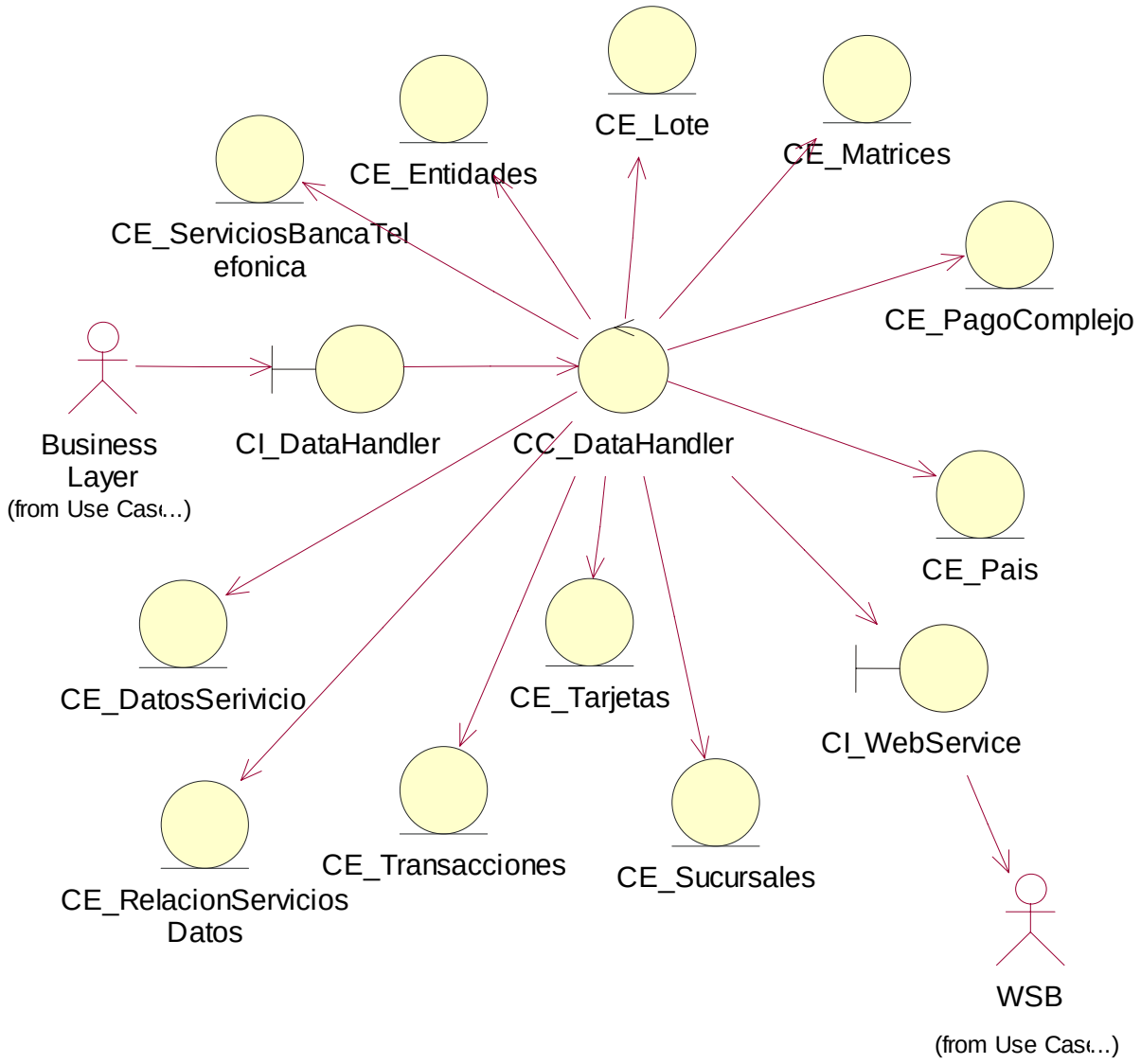


DIAGRAMA DE CLASES DE ANÁLISIS

Módulo Gestión Histórico

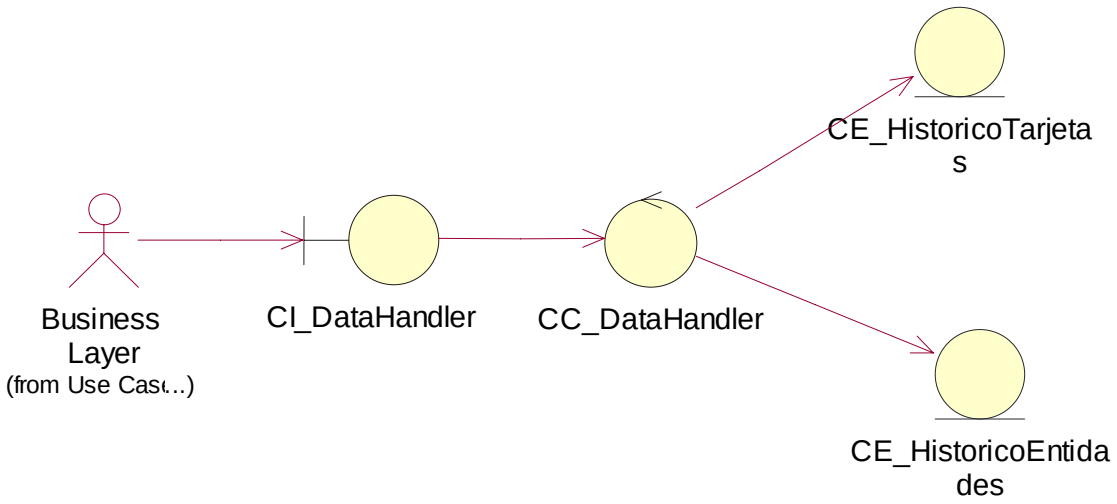


DIAGRAMA DE CLASES DE ANÁLISIS

Módulo Gestión Usuario

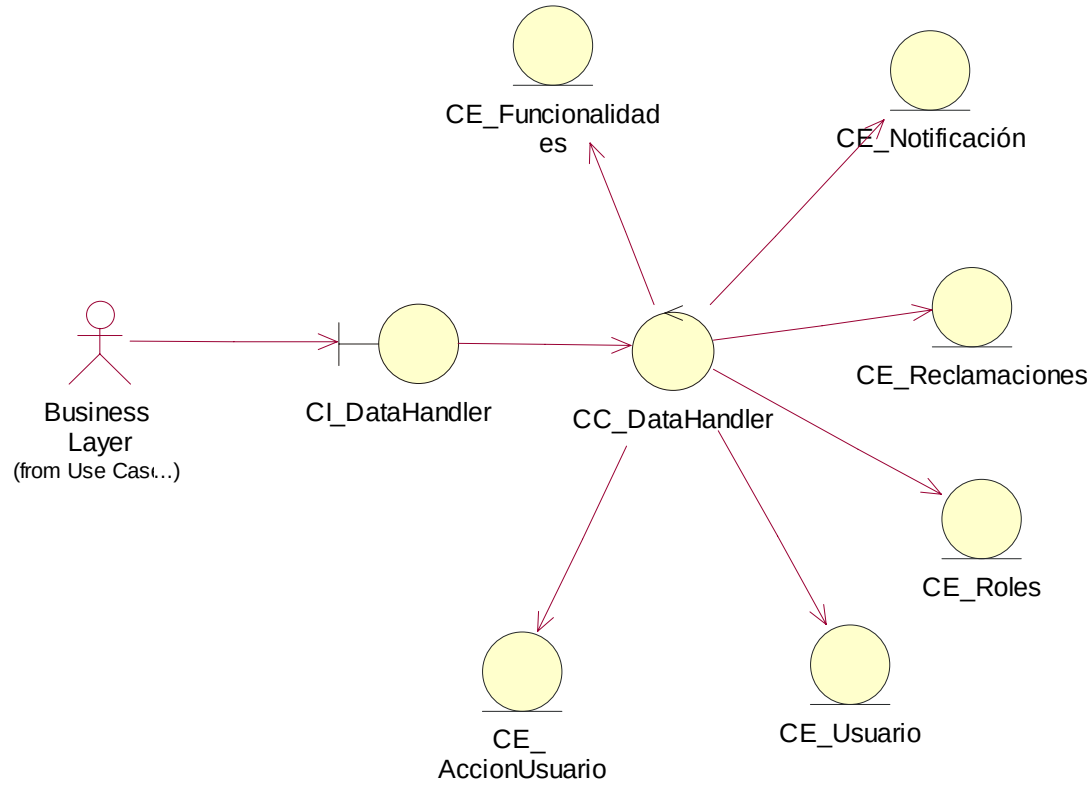
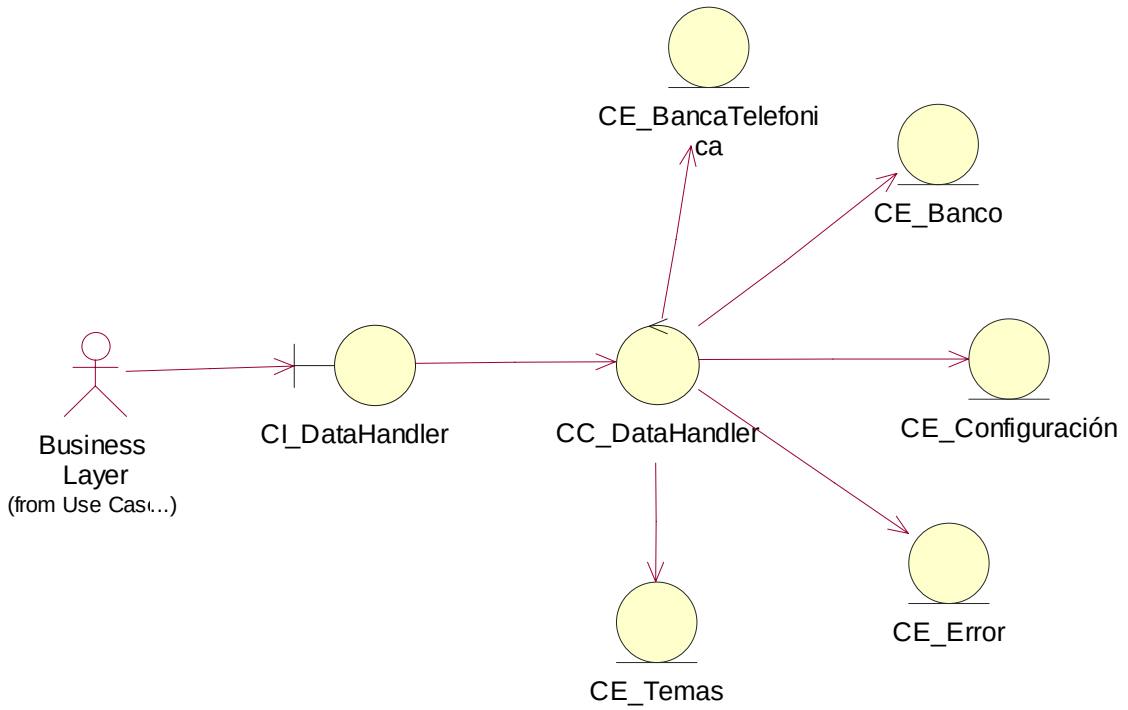


DIAGRAMA DE CLASES DE ANÁLISIS

Módulo Gestión y Mantenimiento de la Banca Telefónica



3.2 Diseño

En este epígrafe se desarrollaran los diagramas de interacción mediante los diagramas de secuencia. También se definen los diagramas de clases obtenidos del refinamiento del modelo conceptual del análisis así como la descripción de las clases. Y finalmente, la creación de los diagramas para diseñar la base de datos.

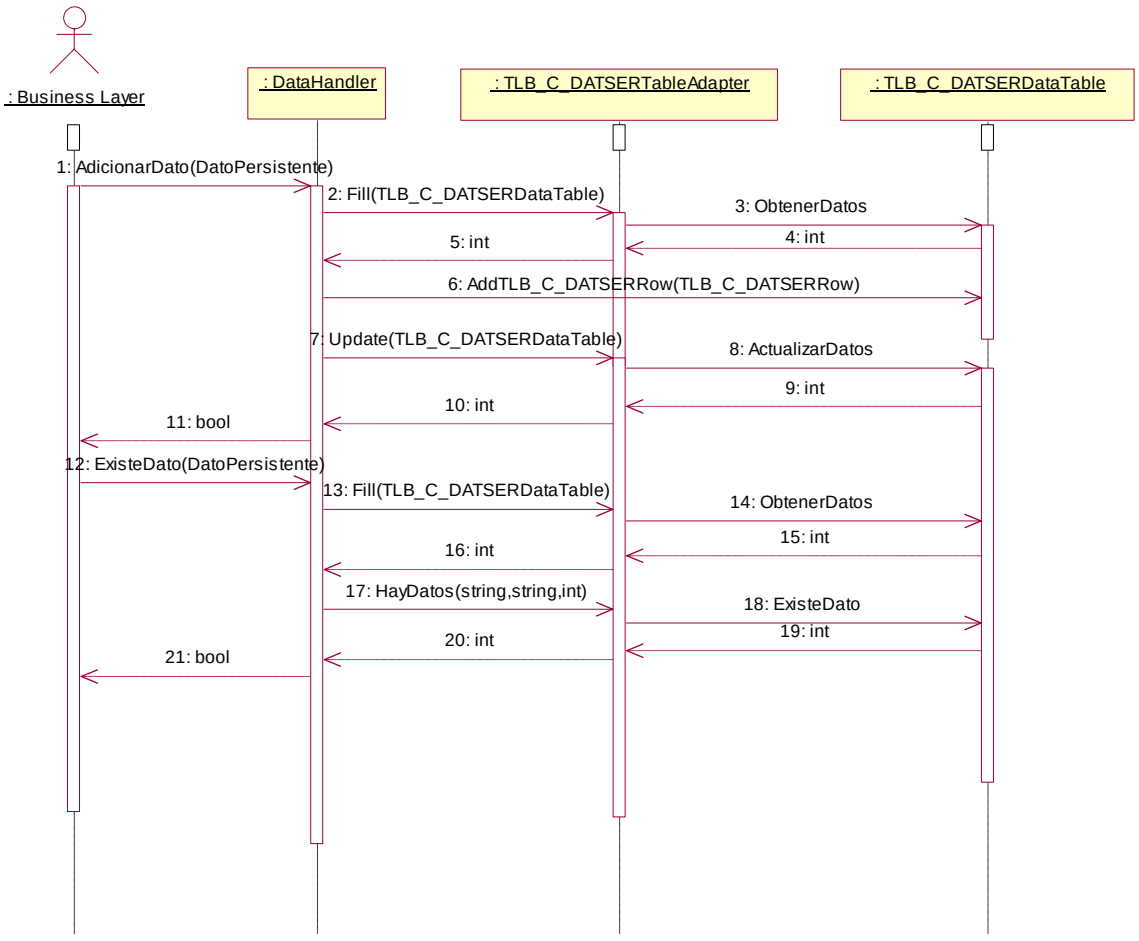
3.2.1 Diagramas de Interacción

Un diagrama de interacción muestra una interacción, que consta de un conjunto de objetos y sus relaciones, incluyendo los mensajes que se pueden enviar entre ellos. En la notación de UML, los diagramas de secuencia y los diagramas de colaboración son llamados diagramas de interacción y son uno de los tipos de diagramas que se utilizan para modelar los aspectos dinámicos de los sistemas.

Seleccionamos el tipo de diagrama de secuencia para construir los diagramas de interacción, ya que los diagramas de secuencias destacan el ordenamiento de los mensajes, es decir, muestra como los objetos se comunican entre ellos en un espacio de tiempo secuencial. Se han desarrollado diagramas de secuencia para todos los casos de uso de cada paquete, a continuación se muestran los más importantes por módulos y el resto puede consultarse en el **Anexo 2**.

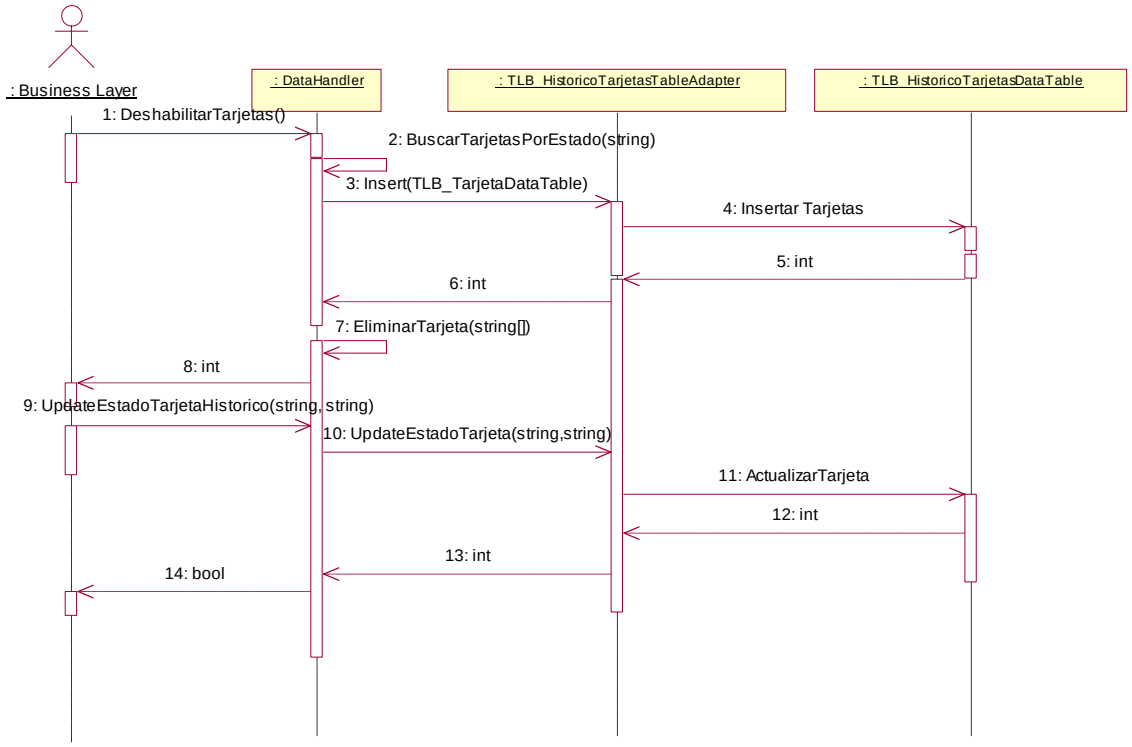
- **Módulo Gestión Bancaria.**

Diagrama de Secuencia para el flujo AdicionarDato y ExisteDato del CU_Gestionar DatosSerivicio



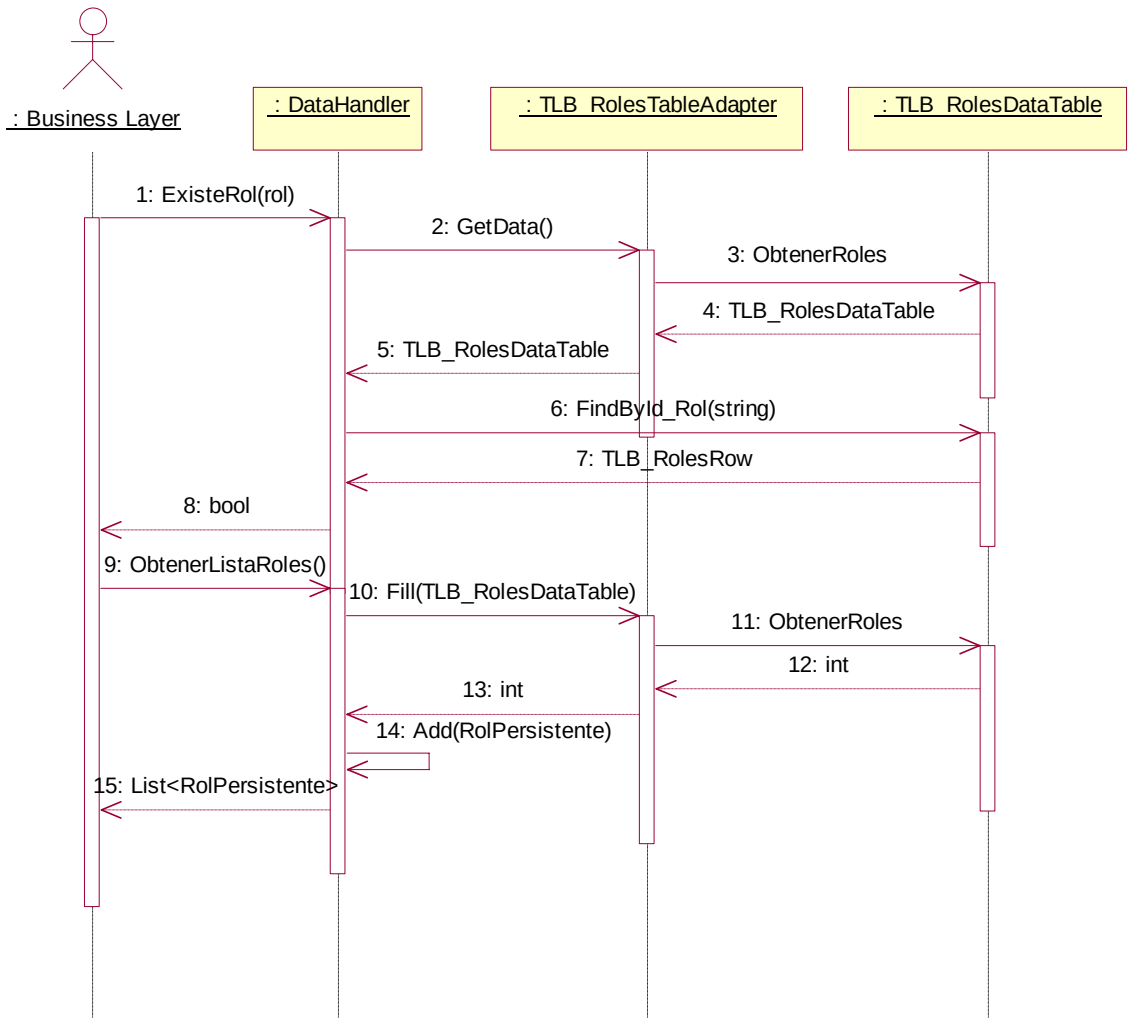
- **Módulo Gestión Histórico.**

Diagrama de Secuencia para el flujo DeshabilitarTarjetas y UpdateEstadoTarjetaHistorico del CU_Gestionar HistoricoTarjetas



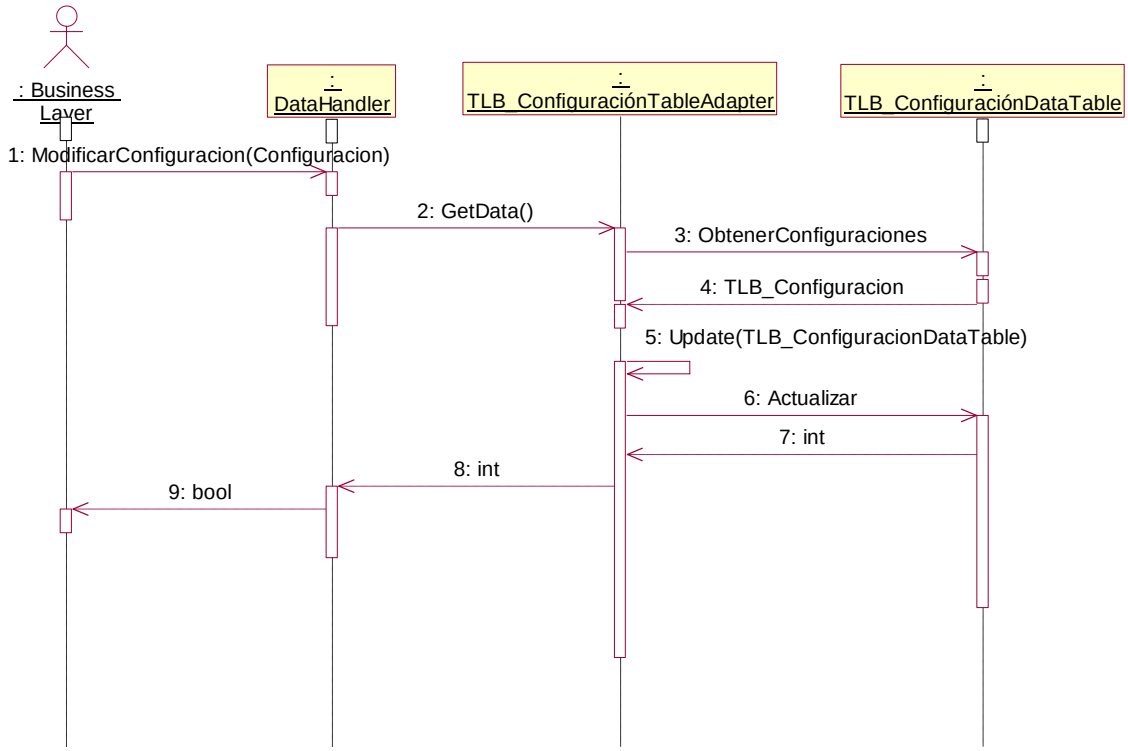
- **Módulo Gestión Usuario.**

Diagrama de Secuencia para el flujo ExisteRol y ObtenerListaRoles del CU_Gestionar Roles



- **Módulo Gestión y Mantenimiento de la Banca Telefónica.**

Diagrama de Secuencia para el flujo ModificarConfiguracion del CU_Gestionar Configuración



3.2.2 Diagramas de Clases

Los diagramas de clases tienen gran importancia en el momento de entender las relaciones entre clases involucradas en los casos de uso. A continuación aparecen los diagramas de clases más descriptivos de cada módulo, el resto puede consultarse en el **Anexo 3**.

Subdiagrama #1 de clases del diseño. Módulo Gestión Bancaria

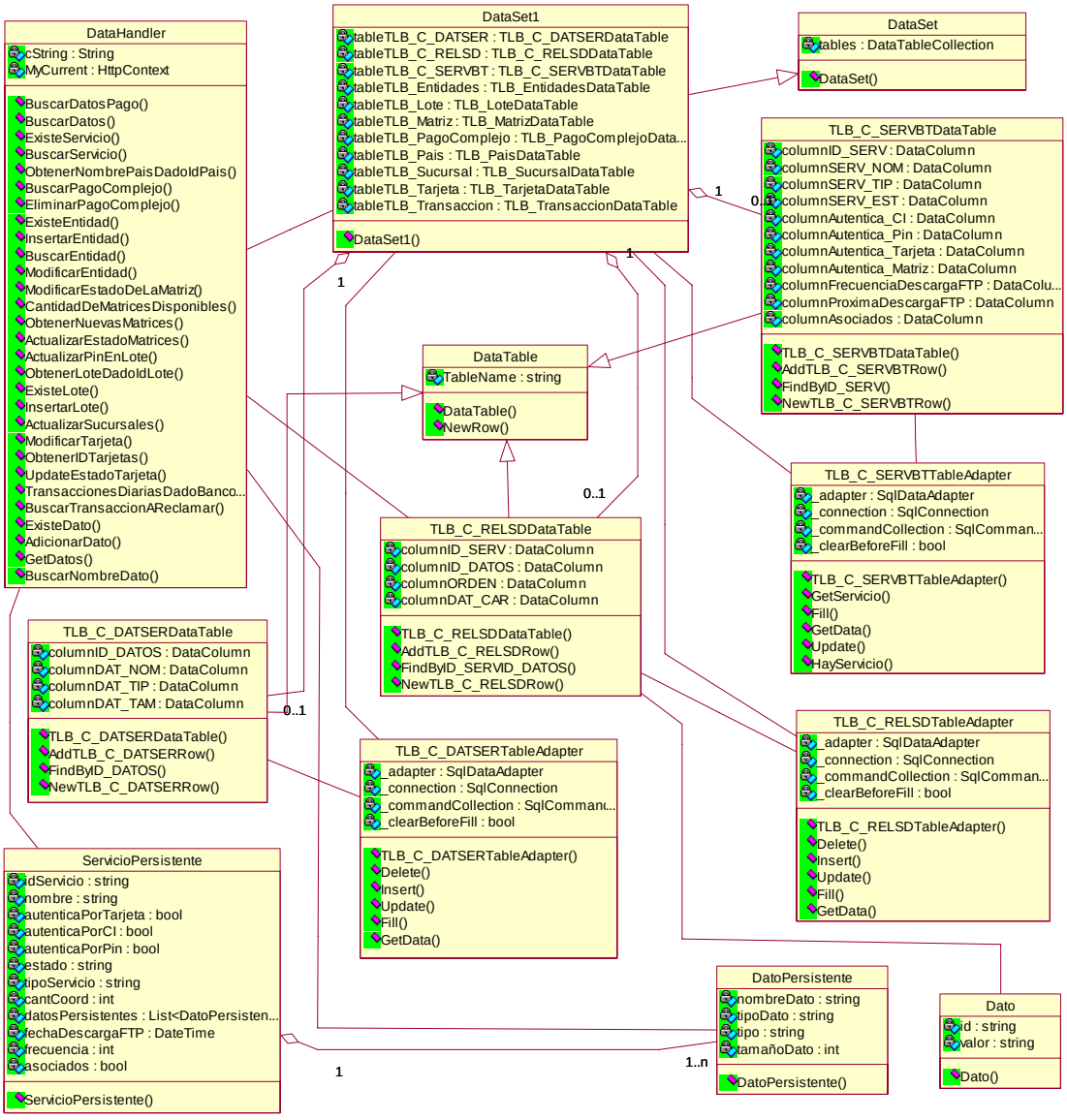
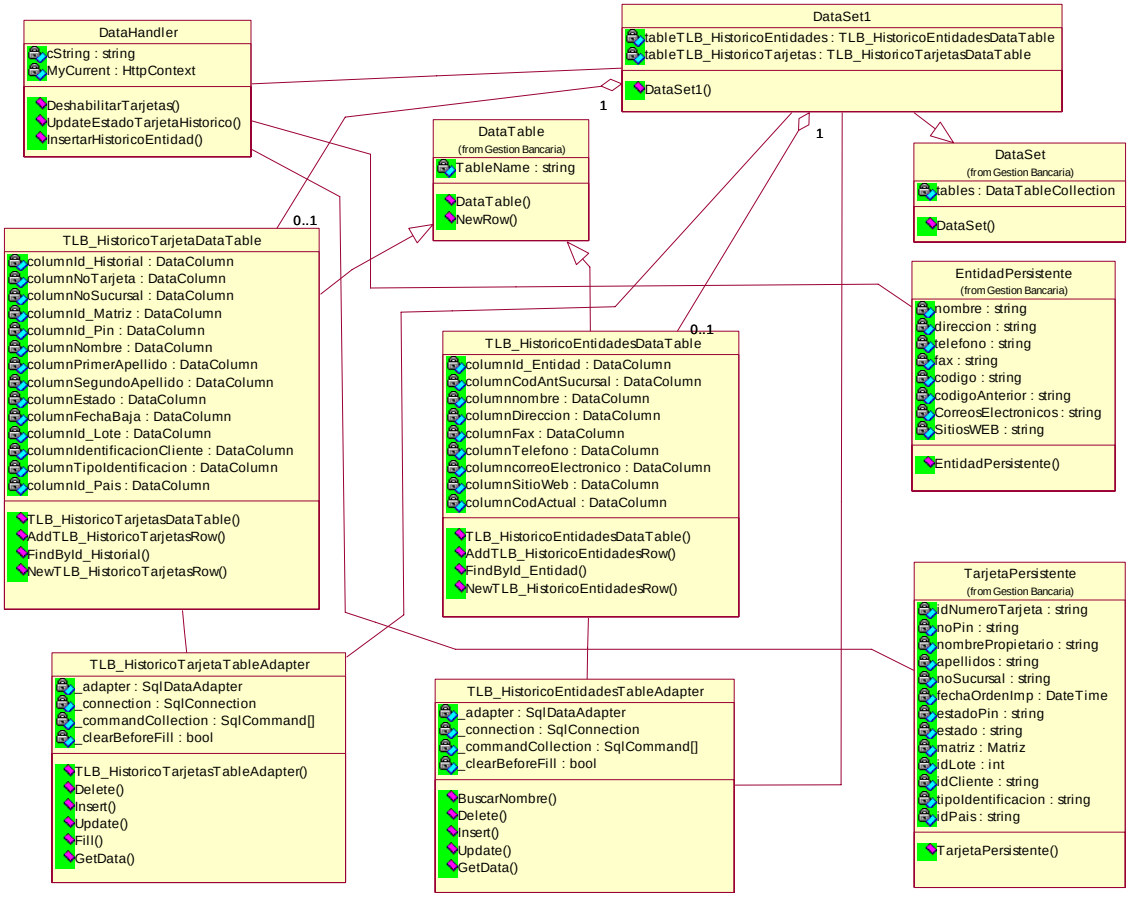
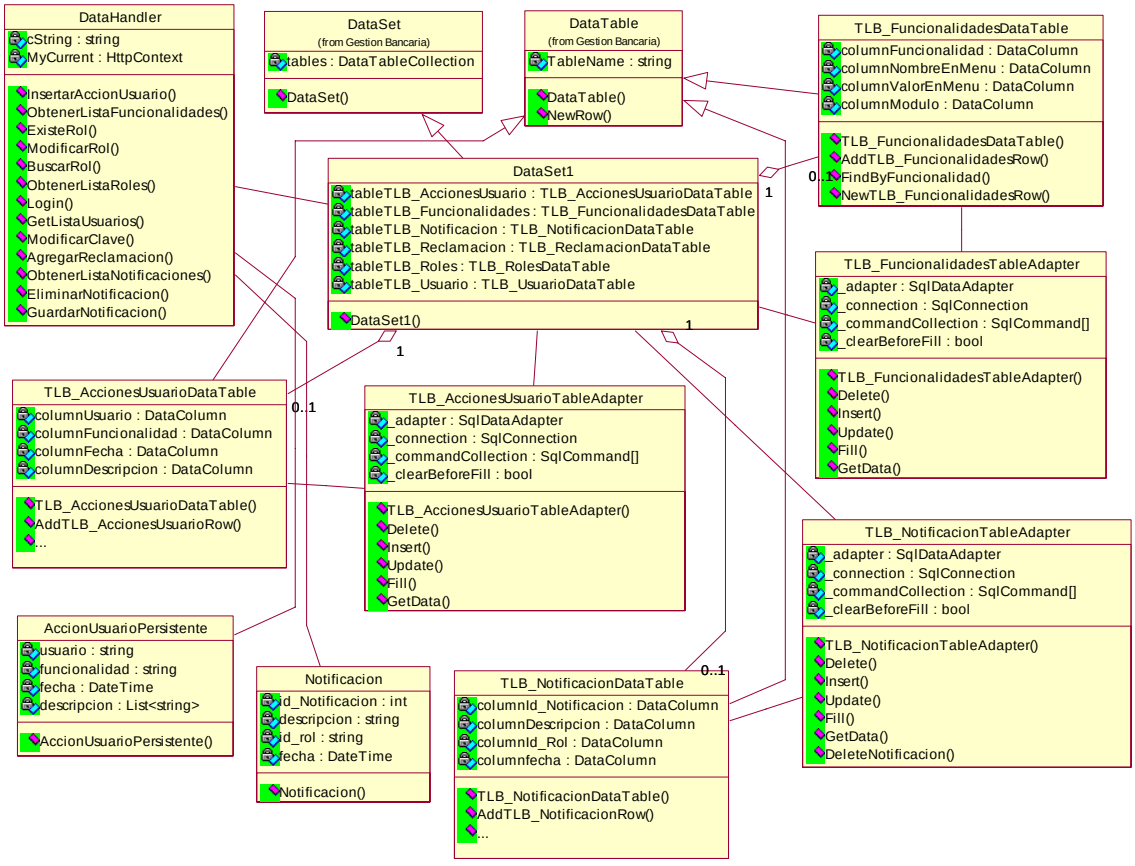


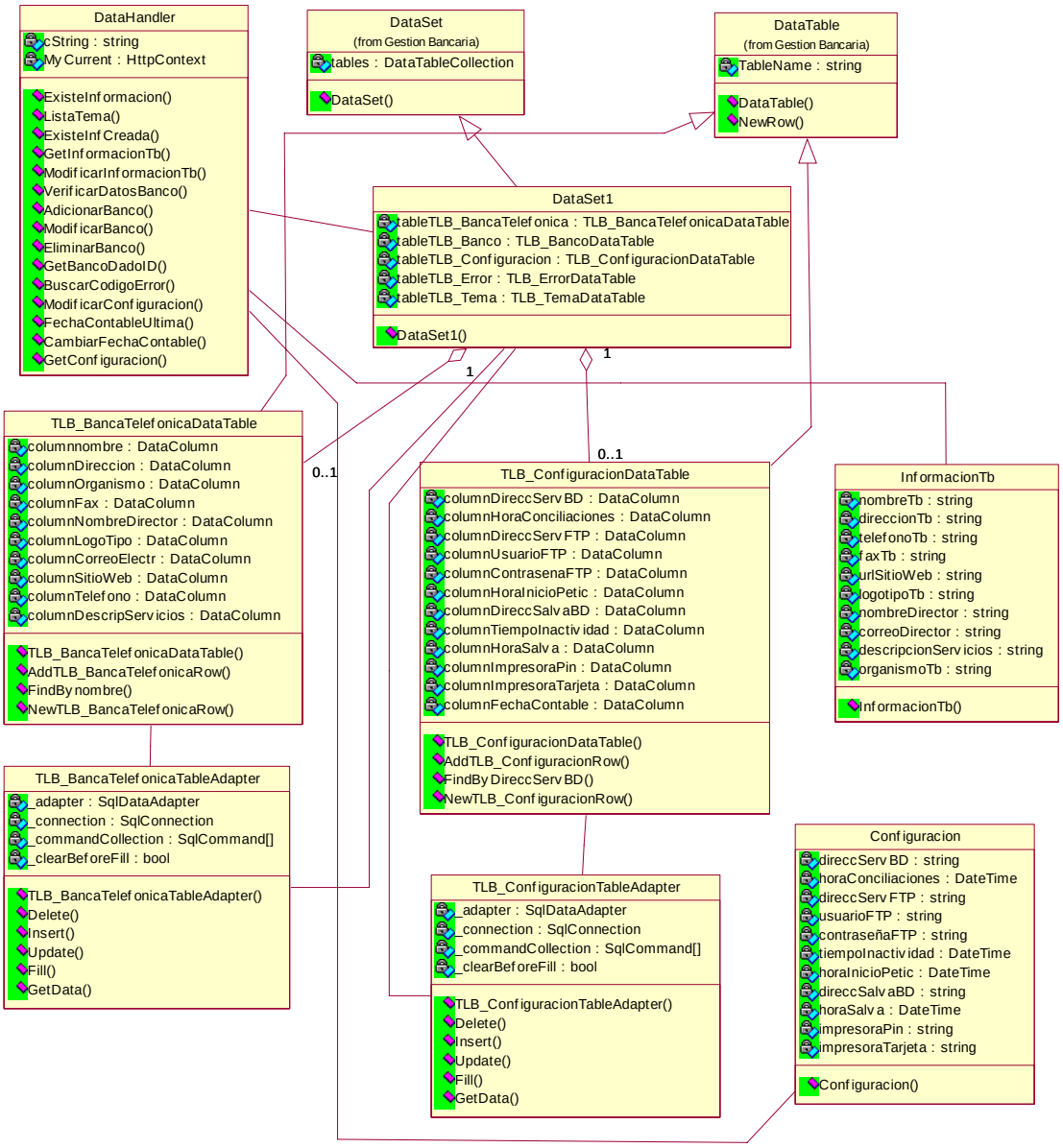
Diagrama de clases del diseño. Módulo Gestión Histórico



Subdiagrama #1 de clases del diseño. Módulo Gestión Usuarios



Subdiagrama #1 de clases del diseño. Módulo Gestión y Mantenimiento de la Banca Telefónica



3.2.3 Descripción de las Clases

La descripción de las clases ayudan a comprender mucho mejor las responsabilidades de las mismas y al mismo tiempo tener una visión más exacta del diseño del sistema. A continuación la descripción de las clases más significativas dentro del diseño del subsistema y las demás descripciones pueden consultarse en el **Anexo 4**.

Nombre: DataHandler	
Tipo de clase (controladora)	
Atributo	**Tipo**
cString	String
MyCurrent	HttpContext
Para cada responsabilidad:	
Nombre:	BuscarDatosPago(string codServ)
Descripción:	Obtiene los datos de pago de un servicio dado el identificador del servicio y devuelve la lista de datos de pagos.
Nombre:	BuscarDatos(string codServ,string[] tiposDato)
Descripción:	Obtiene los tipos de datos especificados dados el identificador del servicio y el arreglo de tipo de dato, devolviendo una lista de datos del servicio.
Nombre:	ExisteServicio(string NombreServicio, string idServ)
Descripción:	Verifica la existencia en la base de datos de un servicio dado el identificador y el nombre del servicio.
Nombre:	BuscarServicio(string nombServ)
Descripción:	Obtiene un servicio dado su nombre.
Nombre:	ObtenerNombrePaisDadoIdPais(string IdPais)
Descripción:	Obtiene el nombre de un país dado su identificador.
Nombre:	BuscarPagoComplejo(string idServicio, string idCliente)
Descripción:	Obtiene una lista de pagos complejo dado el identificador del cliente y el identificador del servicio.
Nombre:	EliminarPagoComplejo(int id)
Descripción:	Elimina un pago complejo en la base de datos dado su identificador.

Nombre:	ExisteEntidad(string nombre)
Descripción:	Verifica la existencia de una entidad dado su nombre.
Nombre:	InsertarEntidad(EntidadPersistente entidadP)
Descripción:	Inserta una entidad en la base de datos dado la entidad.
Nombre:	BuscarEntidad(string nombre)
Descripción:	Obtiene una entidad dado el nombre de la entidad.
Nombre:	ModificarEntidad(EntidadPersistente entidadP)
Descripción:	Modifica una entidad en la base de datos dado la entidad.
Nombre:	ModificarEstadoDeLaMatriz(int id, string nuevoEstado)
Descripción:	Modificar el estado de una matriz en la base de datos dado su identificador y el nuevo estado.
Nombre:	CantidadDeMatricesDisponibles()
Descripción:	Buscar la cantidad de matrices disponibles en la base de datos.
Nombre:	ObtenerNuevasMatrices()
Descripción:	Obtiene las nuevas matrices y devuelve la lista de matrices desocupadas.
Nombre:	ActualizarEstadoMatrices(List<Matriz> lm)
Descripción:	Actualizar el estado de las matrices dado la lista de matrices.
Nombre:	ActualizarPinEnLote(int idLote, string idUsuarioPin, DateTime dtValor)
Descripción:	Actualiza el estado de los pines de un lote, el usuario que imprimió los pines y la fecha, dado el identificador del lote y los parámetros del usuario y la fecha.
Nombre:	ObtenerLoteDadoIdLote(int idLote)
Descripción:	Obtiene el lote de la base de datos dado el identificador del lote.
Nombre:	ExisteLote(string idUsuario, DateTime dtValor)
Descripción:	Verifica la existencia de un lote dado el identificador del operador que imprimió el lote y la fecha de impresión.
Nombre:	InsertarLote(LotePersistente lote)
Descripción:	Inserta un lote en la base de datos dado el lote y devuelve el identificador del lote.
Nombre:	ActualizarSucursales()
Descripción:	Actualiza las sucursales en la base de datos y devuelve el nombre del banco a la que pertenece la sucursal actualizada.

Nombre:	ModificarTarjeta(string numero, TarjetaPersistente t)
Descripción:	Modifica la tarjeta en la base de datos dado el número de la tarjeta y la tarjeta.
Nombre:	ObtenerIDTarjetas()
Descripción:	Obtiene la lista con los identificadores de las tarjetas de la base de datos.
Nombre:	UpdateEstadoTarjeta(string IdTarjeta, string estado)
Descripción:	Actualiza el estado de las tarjetas dado el identificador de la tarjeta y el estado.
Nombre:	BuscarTransaccionAReclamar(string traza, DateTime f)
Descripción:	Obtiene la transacción dado la fecha de la transacción y la traza de la transacción.
Nombre:	ExisteDato(DatoPersistente dato)
Descripción:	Verifica la existencia de datos en la base de datos.
Nombre:	AdicionarDato(DatoPersistente dato)
Descripción:	Adiciona datos en la base de datos.
Nombre:	GetDatos()
Descripción:	Obtiene una lista de datos de la base de datos.
Nombre:	BuscarNombreDato(int idDato)
Descripción:	Obtiene el nombre del dato dado el identificador del dato.
Nombre:	DeshabilitarTarjetas()
Descripción:	Obtiene las tarjetas que se encuentran en estado deshabilitada en la tabla tarjetas y las adiciona a la tabla TLB_HistoricoTarjetas, después las elimina de la tabla tarjetas y devuelve la cantidad de tarjetas deshabilitadas.
Nombre:	BuscarTarjetasPorEstado(string estado)
Descripción:	Obtiene la lista de tarjetas dado un estado determinado.
Nombre:	UpdateEstadoTarjetaHistorico(string IdTarjeta, string estado)
Descripción:	Actualiza el estado de las tarjetas dado el identificador de la tarjeta y el estado.
Nombre:	InsertarHistoricoEntidad(EntidadPersistente entidadP)
Descripción:	Inserta una entidad en la base de datos.
Nombre:	InsertarAccionUsuario(AccionUsuarioPersistente accionUsuario)
Descripción:	Inserta las acciones de los usuarios en la base de datos dado la acción del usuario.
Nombre:	ObtenerListaFuncionalidades()
Descripción:	Obtiene la lista de funcionalidades de la base de datos.

Nombre:	ExisteRol(string rolP)
Descripción:	Verifica la existencia de un rol dado el identificador del rol.
Nombre:	ModificaRol(RolPersistente rolP)
Descripción:	Modifica el rol de la base de datos dado el identificador del rol.
Nombre:	BuscarRol(string nombreRol)
Descripción:	Obtiene los roles de la base de datos dado el nombre del rol.
Nombre:	ObtenerListaRoles()
Descripción:	Obtiene la lista de roles de la base de datos.
Nombre:	Login(string usuario, string pass)
Descripción:	Autenticarse dado el usuario y la contraseña.
Nombre:	GetListaUsuarios()
Descripción:	Obtiene la lista de usuarios que se encuentra en la base de datos.
Nombre:	ModificarClave(string usuario, string nuevacontrasenna)
Descripción:	Modifica la clave dado el usuario y la nueva contraseña.
Nombre:	AgregarReclamacion(ReclamacionTransaccion r)
Descripción:	Insertar reclamación dado la reclamación en la base de datos.
Nombre:	ObtenerListaNotificaciones()
Descripción:	Obtiene la lista de notificaciones de la base de datos.
Nombre:	EliminarNotificacion(int idNotif)
Descripción:	Elimina la notificación dado el identificador de la notación.
Nombre:	GuardarNotificacion(Notificacion n)
Descripción:	Guarda la notificación dado la notificación y devuelve el identificador de la notificación.
Nombre:	ExisteInformacion(int id_Tema)
Descripción:	Verifica la existencia de información en un tema dado el identificador del tema.
Nombre:	ListaTema()
Descripción:	Obtiene una lista con el nombre de los temas que se encuentran en la base de datos.
Nombre:	ExisteInfCreada()
Descripción:	Verifica la existencia de información en la base de datos.
Nombre:	GetInformacionTb()

Descripción:	Obtiene la información de la banca telefónica que se encuentra en la base de datos.
Nombre:	ModificarInformacionTb(InformacionTb informacionTb)
Descripción:	Modifica la información de la banca telefónica que se encuentra en la base de datos.
Nombre:	VerificarDatosBanco(string nombre, string webServices, string numBanco, string abreviatura)
Descripción:	Verifica la existencia de los datos del banco y los devuelve.
Nombre:	AdicionarBanco(BancoPersistente bancoP)
Descripción:	Adiciona los datos del banco en la base de datos.
Nombre:	ModificarBanco(BancoPersistente bancoP)
Descripción:	Modifica los datos del banco dado el identificador del banco.
Nombre:	EliminarBanco(string numBanco)
Descripción:	Elimina los datos del banco dado el número del banco.
Nombre:	GetBancoDadoID(string ID_Banco)
Descripción:	Obtiene el banco dado el identificador del banco.
Nombre:	BuscarCodigoError(string traza)
Descripción:	Obtiene la descripción del error dado la traza.
Nombre:	ModificarConfiguracion(Configuracion Cconf)
Descripción:	Modifica la configuración dado la configuración.
Nombre:	FechaContableUltima()
Descripción:	Obtiene la última fecha contable que se encuentra en la base de datos.
Nombre:	CambiarFechaContable(DateTime pFecha)
Descripción:	Modifica la fecha contable dado la fecha contable.
Nombre:	GetConfiguracion()
Descripción:	Obtiene la configuración que se encuentra en la base de datos.

Nombre: DataSet1	
Tipo de clase (controladora)	
Atributo	**Tipo**
tableTLB_C_DATSER	TLB_C_DATSERDataTable
tableTLB_C_RELSD	TLB_C_RELSDDataTable

tableTLB_C_SERVBT	TLB_C_SERVBTDataTable
tableTLB_Entidades	TLB_EntidadesDataTable
tableTLB_Lote	TLB_LoteDataTable
tableTLB_Matriz	TLB_MatrizDataTable
tableTLB_PagoComplejo	TLB_PagoComplejoDataTable
tableTLB_Pais	TLB_PaisDataTable
tableTLB_Sucursal	TLB_SucursalDataTable
tableTLB_Tarjeta	TLB_TarjetaDataTable
tableTLB_Transaccion	TLB_TransaccionDataTable
tableTLB_HistoricoEntidades	TLB_HistoricoEntidadesDataTable
tableTLB_HistoricoTarjetas	TLB_HistoricoTarjetasDataTable
tableTLB_AccionesUsuario	TLB_AccionesUsuarioDataTable
tableTLB_Funcionalidades	TLB_FuncionalidadesDataTable
tableTLB_Notificacion	TLB_NotificacionDataTable
tableTLB_Reclamacion	TLB_ReclamacionDataTable
tableTLB_Roles	TLB_RolesDataTable
tableTLB_Usuario	TLB_UsuarioDataTable
tableTLB_BancaTelefonica	TLB_BancaTelefonicaDataTable
tableTLB_Banco	TLB_BancoDataTable
tableTLB_Configuracion	TLB_ConfiguracionDataTable
tableTLB_Error	TLB_ErrorDataTable
tableTLB_Tema	TLB_TemaDataTable
Para cada responsabilidad:	
Nombre:	DataSet()
Descripción:	Constructor de la clase padre
Nombre:	DataSet1()
Descripción:	Constructor de la clase

3.3 Diseño de la Base de Datos

Las bases de datos necesitan la definición de una estructura que le permitan almacenar datos, reconocer el contenido y recuperar la información. En nuestro subsistema la base de datos es extensa y el contenido que se almacena en ella es referente a los servicios que brinda la Banca Telefónica, la configuración del sistema y la información referente a los usuarios y clientes de la empresa. Para una mejor comprensión se ha dividido el diseño de la base de datos en 4 diagramas, uno por cada módulo del sistema:

- **Gestión Bancaria.** Modela las tablas de la base de datos que gestiona el módulo bancario.
- **Gestión Históricos.** Modela el diseño de las tablas que gestiona el módulo Histórico.
- **Gestión Usuario.** Modela el diseño de las tablas de la base de datos que guardan información referente a los usuarios del sistema, sus roles dentro del sistema y sus funcionalidades.
- **Gestión y Mantenimiento de la Banca Telefónica.** Modela el diseño de las tablas gestionadas por el módulo de mantenimiento.

3.3.1 Diagramas del Modelo de Datos

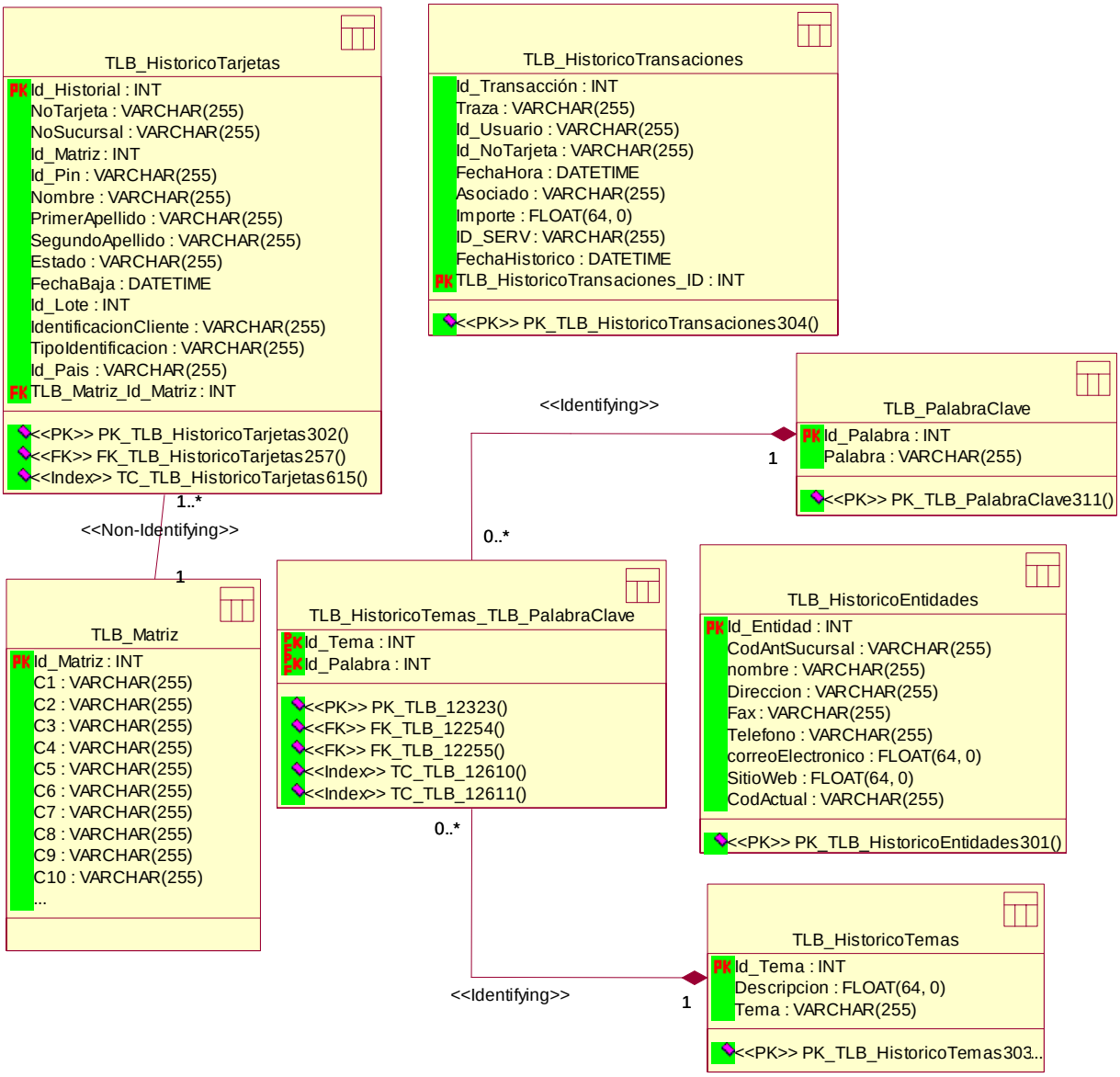
El modelo de datos de las clases persistentes es un mecanismo para representar la información de aquellas clases que formarán parte de la base de datos como tablas. A continuación los diagramas del modelo de datos dividido por módulos para una mejor comprensión.

MODELO DE DATOS

Módulo Gestión Bancaria

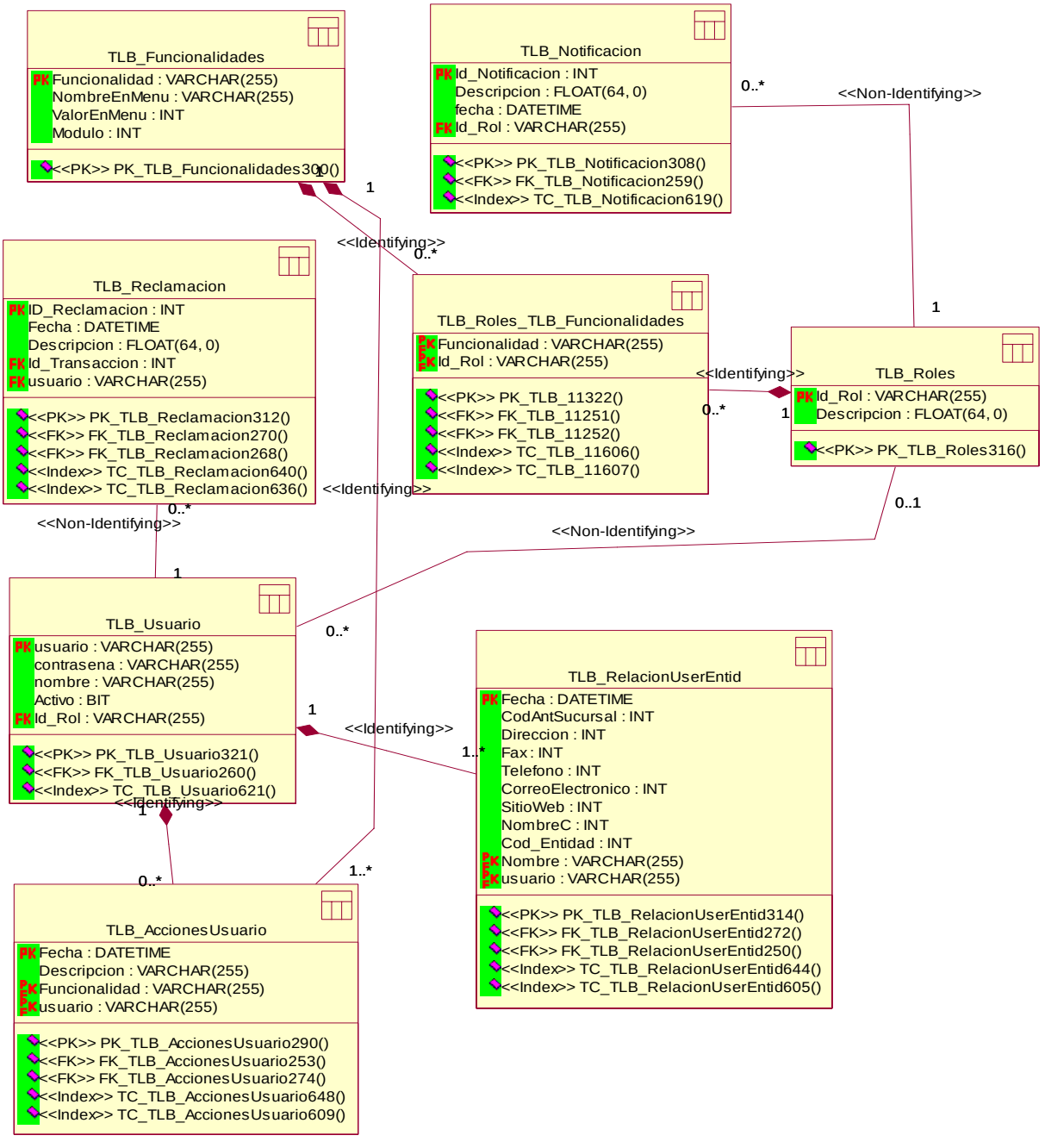
MODELO DE DATOS

Módulo Gestión Histórico



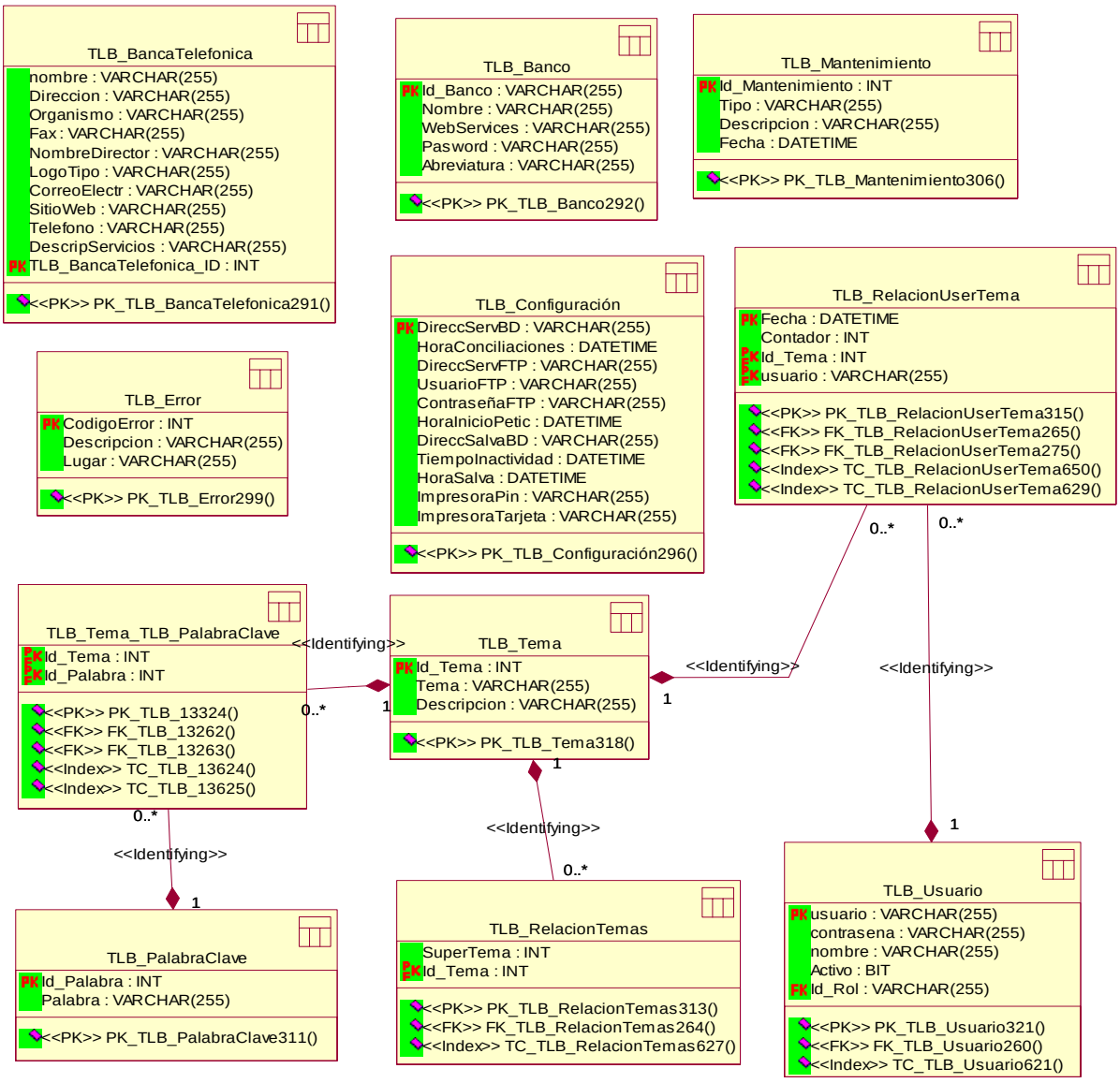
MODELO DE DATOS

Módulo Gestión Usuario



MODELO DE DATOS

Módulo Gestión y Mantenimiento de la Banca Telefónica



3.3.2 Descripción de las tablas

A continuación se describen las principales tablas de la base de datos del sistema, las demás descripciones pueden consultarse en el **Anexo 5.**

Nombre: TLB_C_DATSER		
Descripción: Registra los datos de cada servicio		
Atributo	**Tipo**	**Descripción**
ID_DATOS	Smallint	Identificador de los datos
DAT_NOM	VarChar(20)	Nombre del dato
DAT_TIP	VarChar(15)	Tipo del dato
DAT_TAM	Smallint	Tamaño del dato

Nombre: TLB_C_RELSD		
Descripción: Guarda la relación entre los datos y los servicios.		
Atributo	**Tipo**	**Descripción**
ID_SERV	VarChar(2)	Identificador del servicio
ID_DATOS	Smallint	Identificador de los datos
Orden	Smallint	Orden en que se mostraran los datos
DAT_CAR	VarChar(15)	Característica del dato

Nombre: TLB_C_SERVBT		
Descripción: Guarda los datos de los servicios que brinda la Banca Telefónica.		
Atributo	**Tipo**	**Descripción**
ID_SERV	VarChar(2)	Identificador de servicio
SERV_NOM	VarChar(20)	Nombre del servicio
SERV_EST	VarChar(2)	Estado del servicio (AC y NA)
SERV_TIP	VarChar(2)	Tipo de servicio (01 y 02)

Autentica_CI	Bit	Si se autentico el carnet de identidad
Autentica_Pin	Bit	Si se autentico el Pin
Autentica_Tarjeta	Bit	Si se autentico la tarjeta
Autentica_Matriz	Smallint	Cantidad de coordenadas de la matriz que deben verificarse
FrecuenciaDescargaFTP	Smallint	Frecuencia de Descarga del FTP
ProximaDescargaFTP	DateTime	Fecha de la próxima descarga del FTP

Nombre: TLB_Tarjeta		
Descripción: Registra los datos de la tarjeta.		
Atributo	**Tipo**	**Descripción**
Id_NoTarjeta	Varchar(10)	Identificador de número de tarjeta
No_Pin	Varchar(4)	Número de pin
Nombre	Varchar(30)	Nombre del propietario de la tarjeta
PrimerApellido	Varchar(30)	Primer apellido del propietario de la tarjeta
SegundoApellido	Varchar(30)	Segundo apellido del propietario de la tarjeta
NoSucursal	Varchar(10)	Número de la sucursal
FechaOrdenImp	DateTime	Fecha en que se imprime la tarjeta
EstadoPin	Varchar(10)	Estado del pin
Estado	Char(1)	Estado actual de la tarjeta
Id_Matriz	Int	Identificador de matriz
Id_Lote	Int	Identificador de lote
IdentificacionCliente	Varchar(50)	Identificador del propietario
TipoIdentificacion	Varchar(50)	Tipo de identificación del propietario
Id_Pais	Varchar(3)	Identificador del país

Nombre: TLB_Transaccion		
Descripción: Registra los datos de transacción.		
Atributo	**Tipo**	**Descripción**
Id_Transaccion	Int	Identificador de transacción

Traza	Varchar(13)	El consecutivo de la transacción en el banco
Id_Usuario	Varchar(15)	Identificador de usuario
Id_NoTarjeta	Varchar(10)	Identificador de número de tarjeta del propietario.
FechaHora	DateTime	Fecha y Hora de la transacción
Asociado	Varchar(20)	Asociado a pagar
Importe	Money	Importe de la transacción
ID_SERV	Varchar(2)	Identificador de servicio

3.4 Tratamiento de errores

En el diseño del subsistema para el acceso a datos del proyecto TeleBanca, se ha tenido en cuenta la idea de que las excepciones sean relanzadas en algunos casos hacia las funciones de la capa de negocio que utilizan las funcionalidades del sistema; en otros casos las excepciones son manejadas directamente por el subsistema. Han tenido un manejo especial las excepciones relacionadas con el servidor de base de datos y las excepciones lanzadas por el Web Service del Banco, principalmente las excepciones de no conectividad con el gestor de bases de datos y con el servicio Web, donde siempre el tratamiento a seguir ha sido captar las excepciones lanzadas por defecto, personalizar los mensajes en un lenguaje amigable para el usuario del sistema y guardarlas en un fichero temporal para su posterior almacenamiento en la base de datos. Las excepciones de otra naturaleza como errores en la entrada de datos, son relanzadas hacia las funciones de la capa de negocio donde se sigue un tratamiento de acuerdo con los procesos de negocio establecidos en el proyecto. No se han tenido en cuenta las excepciones de interfaz de usuario ya que el subsistema no cuenta con una interfaz para el usuario del sistema, quien utiliza las funcionalidades del subsistema para el acceso a datos es la capa de negocio del proyecto TeleBanca.

3.5 Seguridad

Dada la sensibilidad de los datos que se manejan en el entorno de las finanzas, se hace necesario un mecanismo que garantice la seguridad y la autenticidad de los datos que se manejan en la aplicación, la autenticación de usuarios se realiza en capas superiores de la aplicación como la capa de negocio, para esto se cuenta con un servidor de bases de datos para los usuarios. El principal mecanismo para la seguridad de la información que se maneja en el subsistema de acceso a datos es la autenticación con los Web Services del banco, para lo que se cuenta en la base de datos con una contraseña encriptada para cada Web Service, y antes de realizar alguna operación con estos se verifica la autenticación. De esta

forma, se protege el acceso a la información y la integridad de los datos que maneja la capa de acceso a datos.

Conclusiones

Con la culminación de la etapa de análisis, logramos una primera aproximación conceptual, ya que se logra una mayor comprensión de los requisitos a este nivel pues se realiza el modelo conceptual de clases, por lo tanto, al iniciarse la fase de diseño podemos lograr un mayor nivel de especificación con el fin de cubrir todos los requisitos funcionales y no funcionales considerando además el entorno de implementación del subsistema. Al culminar la fase de diseño se realizaron los diagramas de interacción mediante los diagramas de secuencia, gracias a todos estos artefactos generados en esta etapa se logro hacer referencia de forma general al tratamiento de errores del subsistema. Con la creación de todos los elementos que constituyen esta etapa podemos tener una idea mucho más exacta del subsistema propuesto.

CAPÍTULO 4: IMPLEMENTACIÓN

Introducción

En este capítulo se comienza con el resultado del diseño y se implementa el sistema en términos de componentes. La mayor parte de la arquitectura del sistema es capturada durante el diseño, siendo el propósito principal de la implementación el desarrollar la arquitectura y el sistema como un todo. De forma más específica, los propósitos de la implementación son:

- Distribuir el sistema asignando componentes ejecutables a nodos en el diagrama de despliegue.
- Implementar las clases y subsistemas encontrados durante el diseño. En particular, las clases se implementan como componentes de fichero que contienen código fuente.
- Probar los componentes individualmente, y a continuación integrarlos compilándolos y enlazándolos en uno o más ejecutables, antes de ser enviados para ser integrados y llevar a cabo las comprobaciones de sistema.

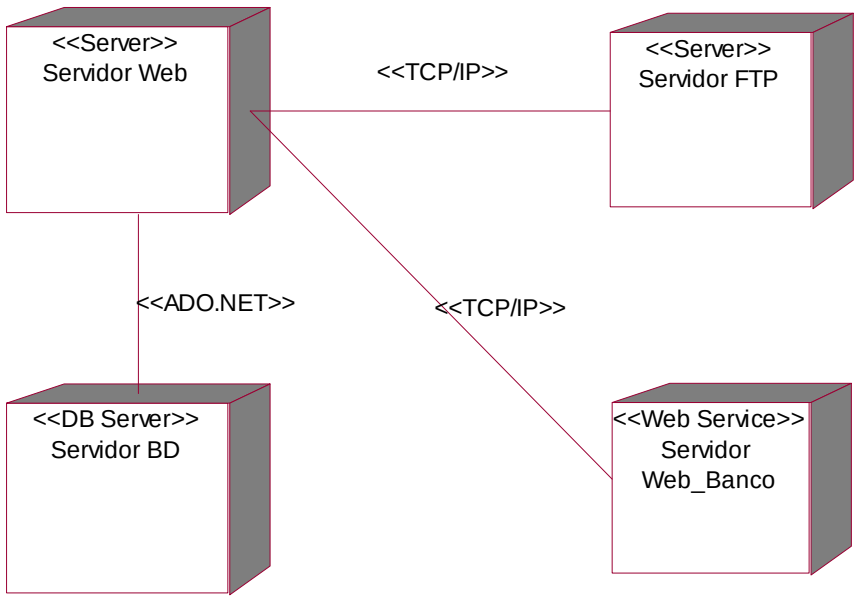
4.1 Implementación

En la fase de implementación se comienza con los resultados obtenidos de la etapa de diseño, es decir, describe como los elementos del modelo de diseño se implementan en términos de componentes y se organizan de acuerdo a los nodos específicos del modelo de despliegue. El modelo de implementación está conformado por los diagramas de despliegue y componentes los cuales describen los componentes a construir y su organización, y la dependencia entre nodos físicos en los que funcionara el subsistema.

4.1.1 Diagrama de Despliegue

Los Diagramas de Despliegue muestran la disposición física de los distintos nodos que componen un sistema y la distribución de los componentes sobre dichos nodos. La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación.

DIAGRAMA DE DESPLIEGUE



4.1.2 Diagramas de Componentes

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. Muestran las opciones de realización incluyendo código fuente, binario y ejecutable. Los componentes representan todos los tipos de elementos software que entran en la fabricación de aplicaciones informáticas. Las relaciones de dependencia se utilizan en los diagramas de componentes para indicar que un componente utiliza los servicios ofrecidos por otro componente.

DIAGRAMA DE COMPONENTES

General

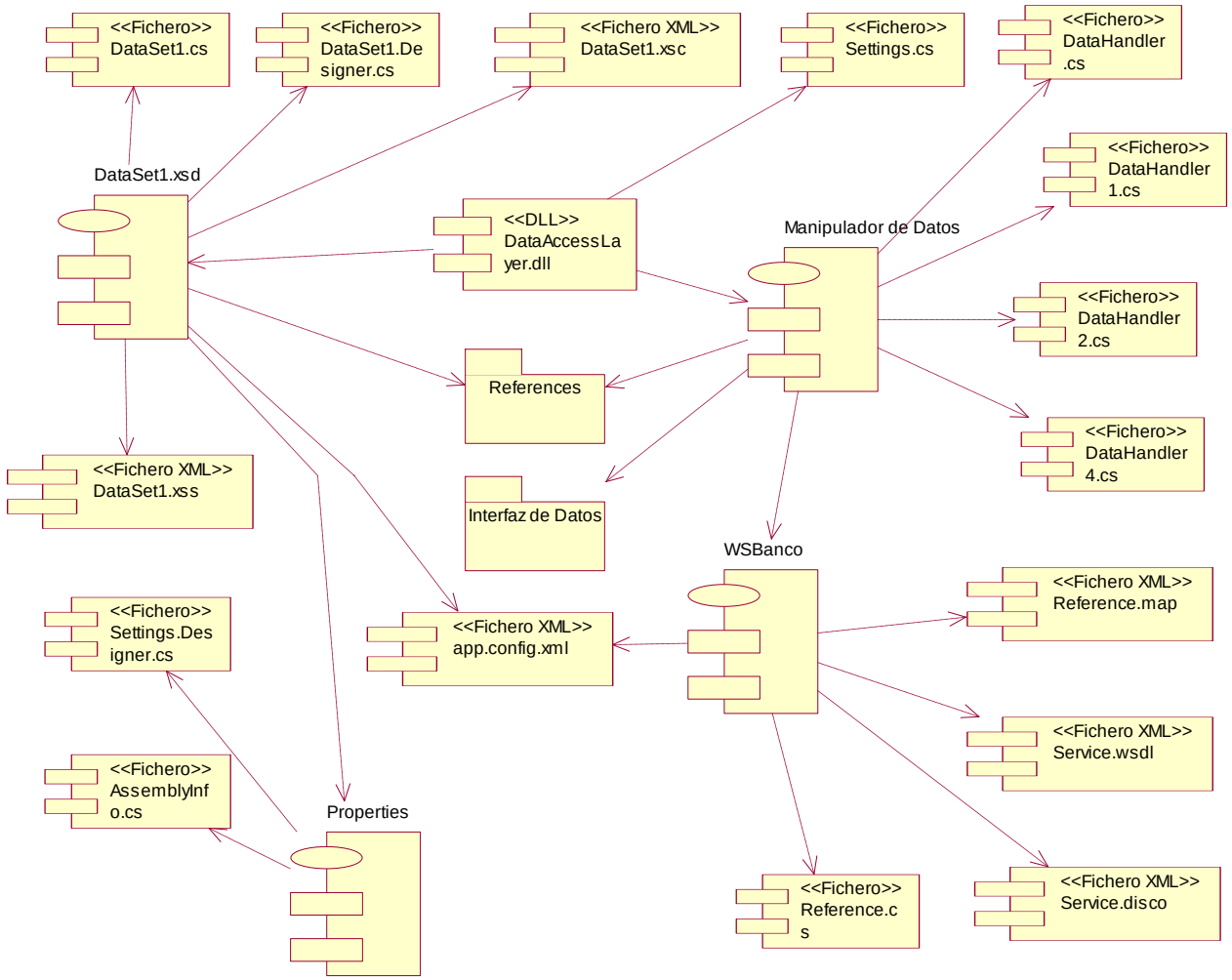


DIAGRAMA DE COMPONENTES

Referencias

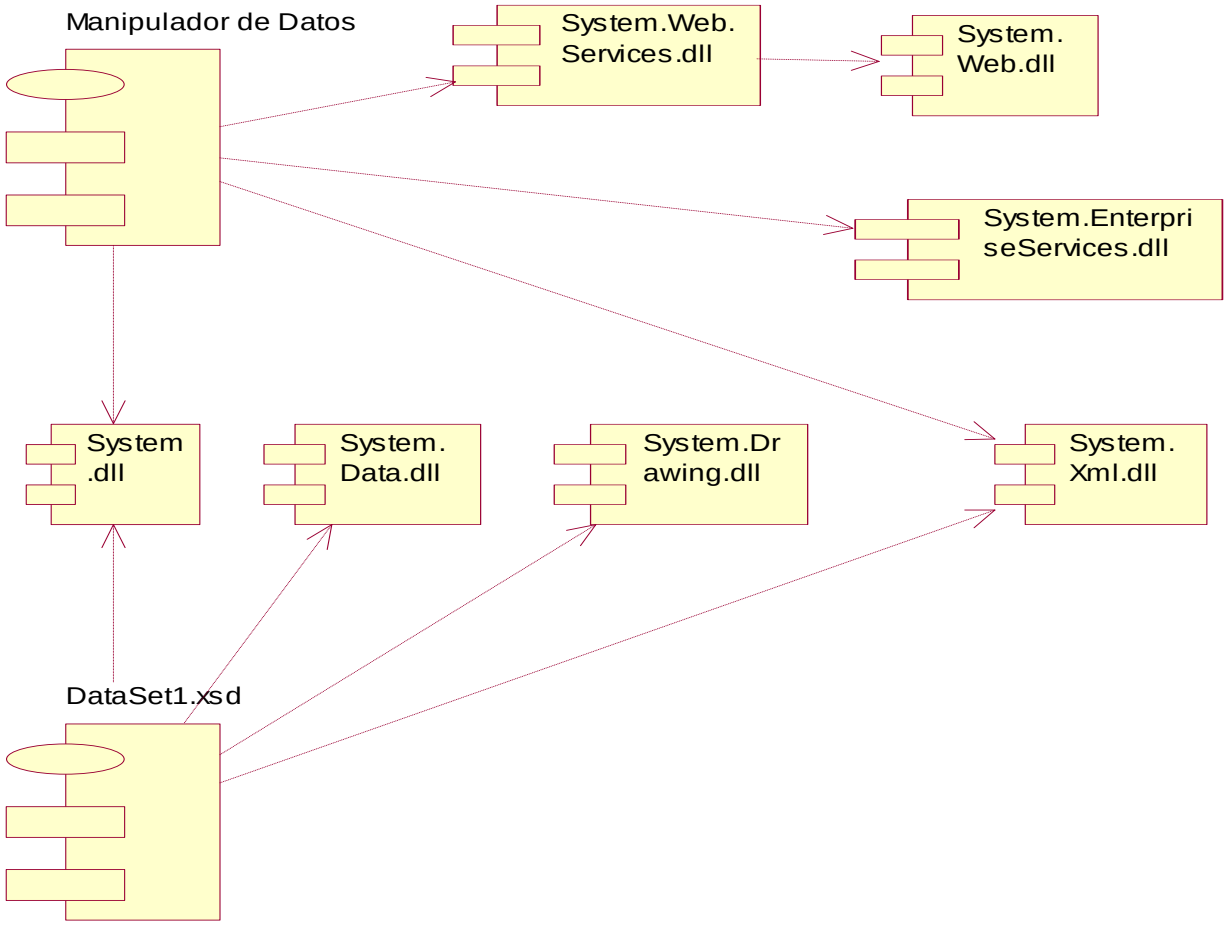
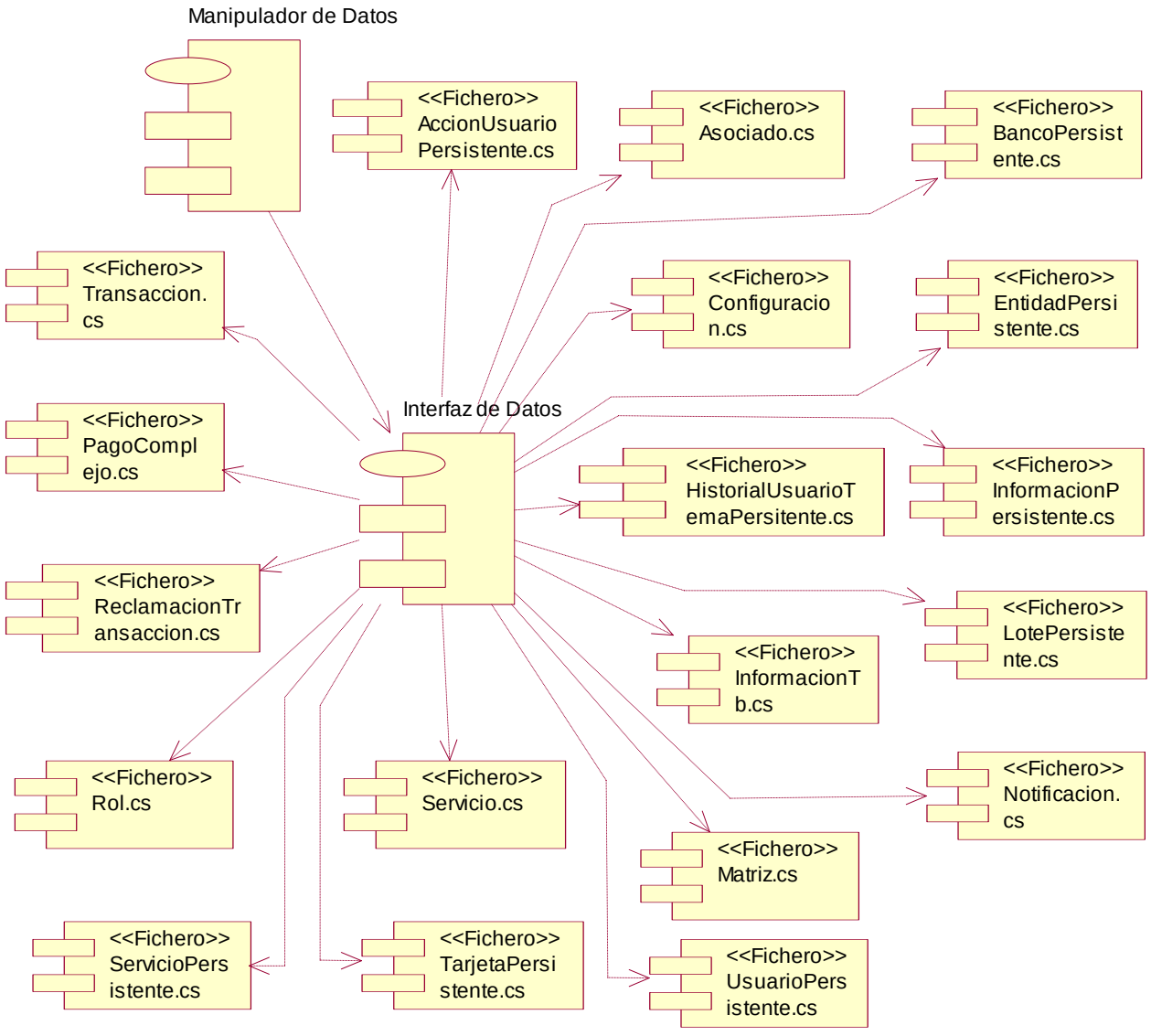


DIAGRAMA DE COMPONENTES

Interfaz de Datos



Conclusiones

En este capítulo se ha analizado el flujo de trabajo de implementación del subsistema planteado para la solución del problema, se ha elaborado el diagrama de despliegue del sistema y el diagrama de componentes; básicamente, se ha llegado a la conclusión de que en el resultado principal del flujo de trabajo de implementación, es el modelo de implementación, el cual incluye los elementos:

- Subsistemas de implementación y sus dependencias, interfaces y contenidos.
- Componentes, incluyendo componentes fichero y bibliotecas dinámicas, y las dependencias entre ellos.
- La vista de arquitectura del modelo de implementación, incluyendo los elementos arquitectónicamente significativos.

El modelo de implementación es la entrada principal de las etapas de prueba que siguen a la implementación. Más concretamente, durante la etapa de prueba cada construcción generada durante la implementación es sometida a pruebas de integración, y posiblemente también a pruebas de sistema.

CONCLUSIONES

Al finalizar este proyecto se pueden llegar a las siguientes conclusiones:

1. Profundizar nuestros conocimientos en el funcionamiento e implementación de las capas de acceso a fuentes de datos en los sistemas de software.
2. Especificar los requisitos necesarios para el desarrollo de la capa de acceso a datos.
3. Determinar los casos de uso que cumplan con los requisitos encontrados.
4. Desarrollar los casos de uso encontrados.
5. Obtener un diseño de la base de datos
6. Implementar los componentes desarrollados.
7. Documentar lo desarrollado usando la metodología de RUP.

Este proyecto nos ha dado la posibilidad como autores:

1. Conocer con mayor detalle el funcionamiento de la tecnología .Net y los servicios Web.
2. Aumentar el conocimiento del Proceso Unificado de Desarrollo de Software.
3. Realizar un pequeño aporte en el desarrollo de este gran proyecto productivo TeleBanca.

RECOMENDACIONES

Con el objetivo de disponer de una aplicación que realice el acceso a fuentes de datos de forma automatizada a todos los procesos que se llevan a cabo en el proyecto productivo TeleBanca, recomendamos lo siguiente:

1. Mantenerse actualizado en cuanto a las nuevas tecnologías para el desarrollo del acceso a fuentes de datos.
2. Darle continuidad al ciclo de desarrollo.
3. Realizar las pruebas de caja blanca al sistema.

Continuar con la investigación para lograr mejoras en versiones futuras.

BIBLIOGRAFÍA

1. PERMUY, F. B. *Introducción al Desarrollo de Aplicaciones Empresariales.* Diciembre 2006, Disponible en: http://www.tic.udc.es/~fbellas/teaching/pfc3/IntroAplEmp.pdf.

2. AGUILAR, J. V. *¿Qué es un call center?.* Disponible en: http://www.gestiopolis.com/canales/demarketing/articulos/61/callcenter.htm.

3. CORPORATION, I. *Iberrail combina su call center con el canal web, de la mano de Servisa.* Disponible en: www.servisa.com.

4. FRANCIA, J. *Desarrollo de una Aplicación en tres Capas con VS .NET.* Disponible en: http://www.microsoft.com/spanish/msdn/comunidad/mtj.net/voices/art140.asp.

5. HERNÁNDEZ, O. *Acceso a datos de próxima generación.* Disponible en: http://www.es-asp.net/articulos-asp-net/1748.aspx.

6. ARMSTRONG, D. *.NET Application Architecture: the Data Access Layer.* Disponible en: http://www.simple-talk.com/dotnet/.net-framework/.net-application-architecture-the-data-access-layer.

7. DEVARAKONDA, R. S. *Object-Relational Database Systems - The Road Ahead.* Disponible en: www.acm.org/crossroads/xrds7-3/ordbms.html.

8. SOLANO, A. *SQL Server 2000.* Disponible en: http://www.netveloper.com/contenido2.aspx?IDC=64_0_.

9. GALLINAS, A. F. G. Y. R. B. *Acceso a Datos Relacionales Bajo un Entorno XML*. Disponible en: http://vecpar.fe.up.pt/xata2005/papersfinal/48.pdf.

10. ESPOSITO, D. *ADO.NET para el programador de ADO.* Disponible en: http://www.microsoft.com/spanish/msdn/articulos/archivo/180501/voices/adonetdev.asp.

11. LEFFINGWELL, D. *Features, Use Cases, requirements, Oh My!* 2000, Disponible en: Http://www.rational.com/media/whitepapers/featucregom.pdf.

12. PRESSMAN, R. *Ingeniería del Software. Un enfoque práctico.* Editado por: Hill, M. 2002, Disponible en: http://bibliodoc.uci.cu/pdf/reg02689.pdf.

13. SCHMULLER, J. *Aprendiendo UML en 24 Horas.* Editado por: Education, P. 2000, Disponible en: http://bibliodoc.uci.cu/pdf/reg00004.pdf.

14. JAMES RAMBAUGH, I. J., GRADY BOOCH. *El Lenguaje Unificado de Modelado. Manual de Referencia.* Editado por: Wesley, A. 1998, Disponible en: http://bibliodoc.uci.cu/pdf/reg03050.pdf.

15. WENDY BOGGS, M. B. *UML with Rational Rose 2002.* Editado por: Sybex. 2002, Disponible en: http://bibliodoc.uci.cu/pdf/0782140173.pdf.

16. INFORMATICAS, U. D. L. C. *Modelación de casos de uso del sistema*. En *IGSW1*. 2004.

17. INFORMÁTICAS, U. D. L. C. *Flujo de Análisis y Diseño. Modelo de Análisis*. En IGSW1. 2005.

18. INFORMÁTICAS, U. D. L. C. *Flujo de trabajo Análisis & Diseño. (Modelo de diseño)*. En IGSW1. 2005.

ANEXOS

"[Insertar anexos]"

GLOSARIO

- **ADO.NET:** Es una evolución del modelo de acceso a datos de ADO (ActiveX Data Objects) que controla directamente los requisitos del usuario para programar aplicaciones escalables. Se diseñó específicamente para el Web, teniendo en cuenta la escalabilidad, la independencia y el estándar XML. Utiliza algunos objetos ADO, como Connection y Command, y también agrega objetos nuevos. Algunos de los nuevos objetos clave de ADO.NET son DataSet, DataReader y DataAdapter.
- **Acceso a datos:** El nivel de gestión de acceso a datos es la porción de la arquitectura que proporciona intercambio de datos en tiempo real (o casi) entre los diversos sistemas responsables del flujo de datos. Las tecnologías que pueden aparecer en la arquitectura de acceso a datos son: OLE DB, ADO.NET y XML.
- **Actor*:*** Alguien o algo, fuera del sistema o negocio que interactúa con el sistema o negocio.
- **API:** Es una interfaz de programación de aplicaciones. Es un conjunto de especificaciones de comunicación entre componentes software. Se trata del conjunto de llamadas al sistema que ofrecen acceso a los servicios del sistema desde los procesos y representa un método para conseguir abstracción en la programación. Uno de sus principales propósitos consiste en proporcionar un conjunto de funciones de uso general. De esta forma, los programadores se benefician de las ventajas de la API haciendo uso de su funcionalidad, evitándose el trabajo de programar todo desde el principio.
- **ASP.NET (Active Server Pages)**: Es una tecnología del lado servidor de Microsoft para páginas web generadas dinámicamente, que ha sido comercializada como un anexo a Internet Information Server (IIS).
- ***Base de Datos (BD*):** Conjunto de datos interrelacionados, almacenados con carácter más o menos permanente en la computadora, puede ser considerada una colección de datos variables en el tiempo.
- **CLR (Common Language Runtime):** Lenguaje común de tiempo de ejecución, es el núcleo de la plataforma .NET. Es el motor encargado de gestionar la ejecución de las aplicaciones para ella

desarrolladas y a las que ofrece numerosos servicios que simplifican su desarrollo y favorecen su fiabilidad y seguridad.

- **COM (Component Object Model):** Modelos de Objetos basados en Componentes, estas tecnologías, originarias de Microsoft, sirven para el diseño de componentes, en realidad de Objetos programables, y la base de OLE tanto en sus variantes OLE DB (DataBase) como OLE DS (Directory Services) y ActiveX.
- **DAO (Data Access Object):** Objeto de Acceso a Datos, es un componente de software que suministra una interfaz común entre la aplicación y uno o más dispositivos de almacenamiento de datos, tales como una Base de datos o un archivo.
- **DLL (Dynamic Link Library):** Biblioteca de vínculos dinámicos, es un archivo que contiene funciones que se pueden llamar desde aplicaciones u otras DLL. Los desarrolladores utilizan las DLL para poder reciclar el código y aislar las diferentes tareas. Las DLL no pueden ejecutarse directamente, es necesario llamarlas desde un código externo.
- **Framework Spring:** Es una plataforma Open Source de desarrollo de aplicaciones para la plataforma Java. Diseñado en módulos, con funcionalidades específicas y consistentes con otros módulos, te facilita el desarrollo de funcionalidades específicas. Dentro de las ventajas que nos ofrece, nos encontramos con que elimina la necesidad de usar distintos y variados tipos de ficheros de configuración y mejora la práctica de programación.
- **Gestores de bases de datos:** conjunto de herramientas que suministra a todos (administrador, analistas, programadores, usuarios) los medios necesarios para describir, recuperar y manipular los datos almacenados en la BD, manteniendo la seguridad, integridad y confidencialidad de los mismo.
- **Herramientas de Mapeo objeto relacional (Object-Relational mapping):** Es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos. En la práctica esto crea una base de datos orientada a objetos virtual, por sobre la base de datos relacional. Esto posibilita el uso de las características propias de la orientación a objetos (básicamente herencia y polimorfismo). Hay paquetes comerciales y de uso libre disponibles que desarrollan el mapeo

relacional de objetos, aunque algunos programadores prefieren crear sus propias herramientas ORM.

- **Hibernate:** Es un marco de trabajo Java que proporciona mecanismos de mapeo objeto/relacional para definir cómo se almacenan, eliminan, actualizan y recuperan los objetos Java. Además, Hibernate ofrece servicios de consulta y recuperación que pueden optimizar los esfuerzos de desarrollo. Por último, Hibernate reduce el esfuerzo necesario para convertir hojas de resultados de la base de datos relacional en gráficos de objetos Java. Hibernate es un servicio de persistencia y consulta objeto/relacional de alto rendimiento y muy poderoso. Nos permite desarrollar clases persistentes siguiendo el idioma de orientación a objetos - incluyendo asociación, herencia, polimorfismo, composición y colecciones. Hibernate permite expresar consultas en su propia extensión portátil SQL (HQL), así como en SQL nativo, o con una API orientada a objetos.
- **HQL (Hibernate Query Language):** Es un lenguaje para el manejo de consultas a la base de datos. Este lenguaje es similar a SQL y es utilizado para obtener objetos de la base de datos. El uso de HQL nos permite usar un lenguaje intermedio que según la base de datos que usemos y el dialecto que especifiquemos será traducido al SQL dependiente de cada base de datos de forma automática y transparente.
- **Java Beans:** Interfaz para la programación en Java desarrollado por Sun Microsistems, que permite la descomposición de una tarea en bloques o Beans. Para funcionar como una clase JavaBean, una clase debe obedecer ciertas convenciones sobre nomenclatura de métodos, construcción, y comportamiento. Estas convenciones permiten tener herramientas que puedan utilizar, reutilizar, sustituir, y conectar JavaBeans.
- **JDBC (Java Database Connectivity):** es un API que describe o define una librería estándar para acceso a fuentes de datos, principalmente orientado a Bases de Datos relacionales que usan SQL (Structured Query Language). No solo provee un interfaz para acceso a motores de bases de datos, sino que también define una arquitectura estándar, para que los fabricantes puedan crear los drivers que permitan a las aplicaciones Java el acceso a los datos.
- **Licencia GNU:** La Fundación para el Software Libre (FSF - Free Software Foundation) está dedicada a eliminar las restricciones de uso, copia, modificación y distribución del software. Promueve el desarrollo y uso del software libre en todas las áreas de la computación.

Específicamente, la Fundación pone a disposición de todo el mundo un completo e integrado sistema de software llamado GNU. La mayor parte de este sistema está ya siendo utilizado y distribuido.

- **Linux:** Es un sistema operativo multi-usuario, multi-tarea que corre en muchas plataformas, incluyendo Intel. Linux tiene una buena interoperabilidad con otros sistemas operativos, incluyendo los de Apple, Microsoft y Novell. Soporta una variada gama de software, incluyendo el sistema de ventanas X y redes TCP/IP.
- **.NET Framework:** El .NET Framework de Microsoft es un ambiente para desarrollar, implementar y ejecutar servicios Web XML y otras aplicaciones. Consiste de tres partes principales: el tiempo de ejecución del lenguaje común, los tipos de Framework, y ASP.NET. Como una infraestructura complementaria, el .NET Compact Framework de Microsoft, es un conjunto de interfaces de programación que permiten a los desarrolladores apuntar a dispositivos móviles como teléfonos inteligentes.
- **.NET Framework SDK (Software Development Kit):** El Kit de desarrollo de software (SDK) de Microsoft .NET Framework versión 2.0 incluye herramientas, documentación y ejemplos que necesitan los desarrolladores para escribir, generar, probar e implementar aplicaciones de .NET Framework en plataformas x86.
- **Open-Source:** Código abierto (del inglés open source) es el término con el que se conoce al software distribuido y desarrollado libremente.
- **ORDBMS (Object Relational Database Management System):** Usa modelos de datos que trata de combinar las características del modelo Orientado a Objetos con el modelo Relacional. Toda la información de la base de datos está guardada en tablas pero algunas notaciones tabulares pueden tener una estructura de datos abstracta. Soporta una forma de SQL llamada SQL3. Tiene un modelo relacional porque los datos están guardados en forma de tablas con filas y columnas pero el modelo relacional tiene que ser modificado para soportar las características clásicas de programación orientada a objeto.
- **PostgreSQL:** Es un servidor de base de datos relacional libre, liberado bajo la licencia BSD. Es una alternativa a otros sistemas de bases de datos de código abierto como MySQL, así como sistemas propietarios como Oracle o DB2.

- **Programación orientada a objeto:** La Programación Orientada a Objetos es una metodología de diseño de software y un paradigma de programación que define los programas en términos de "clases de objetos", objetos que son entidades que combinan estado (es decir, datos) y comportamiento (esto es, procedimientos o métodos). La programación orientada a objetos expresa un programa como un conjunto de estos objetos, que se comunican entre ellos para realizar tareas.

- **RDBMS (Relational Database Management System):** Una base de datos relacional está compuesta de muchas relaciones en forma de listas de dos dimensiones y columnas que tienen filas. La forma en que la información esta presentada al usuario y al programador es conocida como la vista lógica de la base de datos. La información guardada en el disco de la computadora es llamada vista interna.

- **Software (componentes lógicos, programas, software):** Programas o elementos lógicos que hacen funcionar un ordenador o una red, o que se ejecutan en ellos, en contraposición con los componentes físicos del ordenador o la red.

- **SQL (Structured Query Language):** Es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones sobre las mismas. Usa características del álgebra y el cálculo relacional permitiendo lanzar consultas con el fin de recuperar información de interés de una base de datos, de una forma sencilla.

- **Transact-SQL:** Es el lenguaje de programación de SQL Sever, mediante el cual se pueden realizar muchas operaciones relacionadas con él sin programarlas en un lenguaje de programación, lo que simplifica el código y gana en rapidez, se ejecuta dentro del SQL Sever y es código compilado.

- **URL (Uniform Resource Locator):** Localizador Uniforme de Recursos. Es una secuencia de caracteres, de acuerdo a un formato estándar, que se usa para nombrar recursos, como documentos e imágenes en Internet, por su localización. Es además la cadena de caracteres única con la cual se asigna una dirección a cada uno de los recursos de información disponibles en Internet.