Universidad de las Ciencias Informáticas
Facultad 4

Título: Diseño e implementación de las capas de Negocio y Acceso a datos de los módulos Salidas Transitorias y Traslados Interpenal del SIGEP

Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas

Autor(es): Alejandro Expósito Álvarez
Yunieski Escobar Requejo

Tutor: Ing. Javier López del Castillo Caymares

Ciudad de La Habana, Mayo 2008

DECLARACIÓN DE AUTORÍA

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

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

<Alejandro Expósito Alvarez> **<Yunieski Escobar Requejo>**

_____ _____

Firma del Autor Firma del Autor

Ing. Javier Lopez del Castillo Caymares

_____

Firma del Tutor

El ing. Javier López del Castillo Caymares es graduado en el 2007 de la carrera Ingeniería en Ciencias Informáticas en la Universidad de las Ciencias Informáticas.
Luego de su graduación fue ubicado en la misma universidad como profesor de programación de la facultad # 4.
Actualmente pertenece al proyecto SIGEP.

AGRADECIMIENTOS

Agradecemos a todas las personas que ayudaron a confeccionar este trabajo, a nuestro tutor, Javier por sus profundas revisiones, a Hermes por su ayuda siempre oportuna y en general a todos los que aportaron su granito de arena.

Alejandro y Yunieski

DEDICATORIA

Dedico este trabajo a todos mis compañeros y amigos, a mi familia en especial a mis padres, hermanos y mi hijo, que han sido mi inspiración, a ellos también dedico mi título.

Yunieski Escobar Requejo

Dedico este trabajo y mi título, principalmente a mi madre por haberme impulsado siempre a ser una persona mejor. A mi padre que aunque ya no se encuentra conmigo estoy seguro de que se sentiría orgulloso. A mis tíos Freddy y Maria por el apoyo brindado durante todos mis años de carrera. A mi primo Robertico por regañarme cuando no sabía programar. A mis hermanos Erick, Saylí y Teté. A mis compañeros por hacer de mis años de estudio universitario inolvidables. A toda mi familia y amigos.

Alejandro Expósito Álvarez

RESUMEN

Debido a la grave situación del Sistema Penitenciario venezolano se decide crear un sistema para la gestionar toda la información referente al mismo, este se denomina Sistema de Gestión Penitenciaria (SIGEP) y constituye la aplicación que se encargará de informatizar todo el recorrido de los penados por el Sistema Penitenciario venezolano.

Como parte de esta solución se hace necesario informatizar los traslados interpenal que se encargarán de controlar el traslado de un penado de un centro penitenciario a otro y las salidas transitorias que se encargarán de controlar todas las salidas del penado que impliquen el retorno al centro.

Para esto se diseñaron e implementaron las salidas transitorias y los traslados interpenal. Para desarrollar está solución se tuvieron en cuenta patrones de diseño y las herramientas, tecnologías y convenciones que fueron definidas en la arquitectura del SIGEP.

PALABRAS CLAVE
SIGEP, Sistema Penitenciario, Seguridad y Custodia, Traslado Interpenal, Salida Transitoria.

A principios del año 2005 se realizó un censo nacional sobre la situación judicial de la población penitenciaria. Para el censo se elaboró una encuesta que recogía información sobre: datos del interno, datos sobre el delito, datos sobre el proceso judicial y sobre la condición física de la infraestructura penitenciaria. El censo permitió contar con un acercamiento a la realidad del momento en que fue aplicado; sin embargo, esto no resolvió las necesidades en materia de gestión, información, comunicación y apoyo a la toma de decisiones de la Dirección General de Servicios Penitenciarios (DGSP). La situación de las prisiones en Venezuela es alarmante, el hacinamiento, la pésima alimentación, las malas condiciones higiénicas y sanitarias, la drogadicción, el envilecimiento sexual y el retraso procesal son características que conforman un régimen cloacal. Para resolver este problema el gobierno venezolano ha formulado un proyecto de Humanización del Sistema Penitenciario, como parte de este se ha creado el Sistema de Gestión Penitenciaria (SIGEP) para dar respuesta a las necesidades planteadas anteriormente. Este sistema se encargará de ayudar a la toma de decisiones estratégicas de la Dirección General de Servicios Penitenciarios (DGSP), además contribuirá a que se respeten los derechos de los internos, su actividad de rehabilitación y reinserción, y a la gestión administrativa de los penales y de la Dirección General.

Una parte importante del SIGEP es gestionar la planificación de traslados interpenal partiendo de órdenes de los tribunales o de la dirección general, previamente registradas. De igual manera controlará la ejecución de los mismos, registrando la salida, y llegada al Centro destino. Además el sistema debe permitir la planificación de las salidas transitorias, estas son las salidas que implican un retorno del privado de libertad al establecimiento. Entre estas se encuentran los traslados a hospital, a tribunal y los permisos especiales que puede recibir un interno, amparado por la ley. Además se debe gestionar la ejecución de las salidas planificadas y el retorno de los individuos. A partir de esta situación se ha definido el siguiente

PROBLEMA A RESOLVER:

¿Cómo diseñar e implementar las capas de lógica de negocio y acceso a datos de los módulos Salidas Transitorias y Traslados Interpenal del SIGEP?

Teniendo en cuenta el problema planteado se define el siguiente

OBJETO DE ESTUDIO:

Especificación de requisitos funcionales de los módulos Traslados Interpenal y Salidas Transitorias, arquitectura definida para el SIGEP y los flujos de trabajo definidos para el Diseño e Implementación de las capas de negocio y acceso a datos.

Por lo que se especifica el siguiente

CAMPO DE ACCIÓN:

Diseño e implementación de las capas de Negocio y Acceso a Datos de los módulos Salidas Transitorias y Traslados Interpenal del SIGEP, se plantea el siguiente

OBJETIVO GENERAL:

Diseñar e implementar las capas de Negocio y Acceso a Datos de los módulos que automaticen los procesos de Salidas Transitorias y Traslados Interpenal en el SIGEP.

TAREAS A CUMPLIR:

- Realizar el diseño de la solución y realizar los diagramas necesarios teniendo en cuenta los requisitos especificados para el módulo.
- Implementar la solución utilizando las herramientas definidas en la arquitectura.
- Realizar pruebas unitarias y de integración.
- Documentar la solución.

Actualidad e importancia del trabajo

Debido a todos los problemas que presenta actualmente el sistema penitenciario venezolano, el Sistema de Gestión Penitenciaria es de vital importancia ya que resuelve lo referente a la gestión de la información del mismo.

En cuanto a Salidas Transitorias y Traslados Interpenal se logra facilitar y agilizar el trabajo a los Funcionarios de Seguridad y Custodia ya que las operaciones referentes a estos se llevaban de manera manual. Además se aumenta considerablemente el control sobre los movimientos realizados por los privados de libertad.

Métodos y herramientas para resolver el problema:

Para resolver el problema se han utilizado un grupo de herramientas y tecnologías que permiten darle solución al problema planteado, utilizando como lenguaje de programación para desarrollar, Java, bajo la tecnología JEE y como framework arquitectónico base a Spring. Además se usa como framework de acceso a datos Hibernate. Eclipse como Entorno Integrado de Desarrollo (IDE) con los plugins WTP para trabajar con proyectos web, Spring IDE para

trabajar con Spring, Hibernate Tools para trabajar con Hibernate y para el modelado Visual Paradigm.

DESCRIPCIÓN MÁS DETALLADA DEL PROBLEMA.

Seguridad y Custodia:

Es el área que permitirá controlar las actividades de custodia a los reclusos, tanto dentro del penal como fuera de este. También es la encargada del control del inventario del armamento, municiones y los medios técnicos asignados a los custodios para el cumplimiento de sus funciones dentro de los establecimientos penitenciarios. Permite mantener un control sobre las capacidades de los establecimientos penitenciarios. Además aquí se lleva un control sobre las visitas que se reciben en los establecimientos penitenciarios, tanto las relacionadas con los reclusos como de carácter administrativo. En este documento se va a tratar específicamente lo relacionado con los traslados interpenal y las salidas transitorias.

BREVE DESCRIPCIÓN DE LOS PROCESOS.

Traslado Interpenal:

Es el traslado de un interno de un centro penitenciario a otro, previamente amparado por una orden judicial.

A continuación se describen brevemente los procesos involucrados con los traslados interpenal.

Planificación de Traslado Interpenal.

El proceso del negocio, se inicia una vez que el tribunal correspondiente envía una orden de traslado al establecimiento penitenciario donde se encuentra recluido el interno a trasladar. Una vez recibida dicha orden, el Coordinador de Seguridad y Custodia procede a fijar una fecha y hora en la que el interno deberá ser trasladado por los Custodios responsables de dicho traslado, los cuales serán nombrados en el momento de la ejecución.

Ejecución de Traslado Interpenal.

El proceso de Ejecución de Traslado Interpenal se inicia llegada la fecha y hora en la cual ha sido planificado el traslado que se va a ejecutar. En este momento, el Coordinador de Seguridad y Custodia asigna uno o varios custodios que tendrán la misión de acompañar al interno que será trasladado, al establecimiento penitenciario destino.

El Custodio que se encuentra de guardia en el establecimiento, registra en el libro de novedades el nombre de los custodios que se responsabilizarán con el traslado y los datos del transporte en que serán trasladados. Una vez que el interno es recibido en el establecimiento destino, los custodios lo notifican al establecimiento penitenciario origen y se procede a registrar el egreso. En el establecimiento destino, se registra el ingreso del interno trasladado.

Salida Transitoria: Es la salida que implica un retorno del privado de libertad al establecimiento, esta puede ser un traslado a hospital, a tribunal o uno de los permisos especiales especificados en la ley.

A continuación se describen brevemente los procesos involucrados con salidas transitorias.

Planificación de Salida Transitoria.

El proceso de planificación de una salida transitoria se inicia una vez que el tribunal correspondiente a la causa de un interno, envía una orden de salida transitoria al establecimiento penitenciario donde este se encuentra recluido. Si la orden del tribunal no establece una fecha y hora para la salida, el Coordinador de Seguridad y Custodia es el encargado de establecerlas.

Ejecución de Salida Transitoria.

El proceso de ejecución de Salida Transitoria se inicia llegada la fecha y hora en que debe ser ejecutada la salida transitoria planificada previamente. En este momento, el Coordinador de Seguridad y Custodia determina los custodios que se responsabilizarán con la tarea de acompañar al interno al lugar a donde debe ser trasladado, y luego regresarlo al establecimiento penitenciario.

En caso de tratarse de un traslado a hospital por urgencia médica, no se parte de una planificación previa, se ejecuta dicha salida cuando el médico del establecimiento la considere imprescindible para la salud del interno y el director lo apruebe.

En el momento de la captura de requisitos, los procesos descritos anteriormente se llevaban de manera manual en cada una de las prisiones de Venezuela por lo que la información era difícil de consultar y no estaba organizada correctamente. Además el proceso se hacía trabajoso y podía tomar bastante tiempo. Resultaba complicado realizar consultas específicas de información como por ejemplo, mostrar todos los traslados planificados dado un rango de fechas.

Como parte de la captura de requisitos realizada en Venezuela por parte de los analistas de sistema del proyecto SIGEP se obtuvieron los requisitos funcionales, la descripción de las funcionalidades a desarrollar y un prototipo no funcional de los módulos. Estos artefactos son el punto de partida para el diseño e implementación de los módulos Salidas Transitorias y Traslados Interpenal.

Funcionalidades para Traslados Interpenal.

Se presenta una breve descripción de las funcionalidades.

Funcionalidad	Planificar Traslado Interpenal
Actor	FSC
Breve descripción	Se planifica un nuevo traslado interpenal para un individuo o grupo de

	individuos previamente seleccionado(s) a partir de las órdenes registradas en el sistema. El Sistema brinda la posibilidad de introducir los datos del traslado interpenal y adicionar o eliminar individuos a dicho traslado antes de ser registrado.
Pre - condiciones	El FSC se ha identificado y autenticado ante el sistema. El FSC ha seleccionado una o varias órdenes de traslado interpenal.
Post- condiciones	La planificación del traslado interpenal queda correctamente registrada en el sistema.

Funcionalidad	Cancelar Traslado Interpenal
Actor	FSC
Breve descripción	Se selecciona un traslado interpenal de los que ya se encuentran planificados y se indica que se desea cancelarlo. El sistema registra los motivos de cancelación del mismo.
Pre - condiciones	El FSC se ha identificado y autenticado ante el sistema. El FSC ha seleccionado un traslado interpenal de los que se encuentran planificados.
Post- condiciones	El traslado interpenal planificado queda cancelado correctamente. Este no aparecerá más en la lista de traslados interpenal planificados.

Funcionalidad	Actualizar Traslado Interpenal
Actor	FSC
Breve descripción	Se selecciona un traslado interpenal de los que ya se encuentran planificados, se indica que se desea modificar sus detalles y si se desea adicionar y/o eliminar individuos, el sistema permite la entrada de los nuevos datos y registra la actualización del traslado interpenal.
Pre - condiciones	El FSC se ha identificado y autenticado ante el sistema. Se ha seleccionado un traslado interpenal planificado previamente.
Post- condiciones	El traslado interpenal planificado es modificado correctamente.

Funcionalidad	Consultar Traslado Interpenal
Actor	FSC
Breve descripción	Se selecciona un traslado interpenal y se indica que se desean ver los detalles del mismo, el sistema muestra los detalles
Pre - condiciones	El FSC se ha identificado y autenticado ante el sistema. Se ha seleccionado un traslado interpenal previamente planificado.
Post- condiciones	Los detalles del traslado interpenal planificado son consultados correctamente

Funcionalidad	Consultar historial de Traslados Interpenal planificado.
Actor	FSC
Breve descripción	Se muestran todos los traslados interpenal que han sido planificados en el centro. Adicionalmente el Sistema permite especificar un período de tiempo válido para filtrar los resultados.
Pre - condiciones	El FSC se ha identificado y autenticado ante el sistema.
Post- condiciones	El FSC consulta el historial de traslados interpenal planificados en el centro penitenciario.

Funcionalidad	Registrar ejecución de Traslado Interpenal.
Actor	FSC
Breve descripción	Se selecciona un traslado interpenal y se indica que se desea registrar la ejecución del mismo, se introducen los datos de la ejecución y se registra la misma.
Pre - condiciones	El FSC se ha identificado y autenticado ante el sistema. Se ha seleccionado un traslado interpenal previamente planificado.
Post- condiciones	La ejecución del traslado interpenal queda correctamente registrada en el sistema.

Funcionalidad	Registrar conclusión de Traslado Interpenal.
Actor	FSC
Breve descripción	Se selecciona un traslado interpenal que está en ejecución y se registra la

	confirmación de que ha concluido.
Pre - condiciones	El FSC se ha identificado y autenticado ante el sistema. Se ha seleccionado un traslado interpenal en estado de ejecución.
Post- condiciones	La conclusión del traslado interpenal queda correctamente registrada en el sistema.

Funcionalidades para Salidas Transitorias.

Se presenta una breve descripción de las funcionalidades.

Funcionalidad	Cancelar Salida Transitoria
Actor	FSC
Breve descripción	Se selecciona una salida transitoria de las que ya se encuentran planificadas y se indica que se desea cancelarla. El sistema registra los motivos de cancelación de la misma.
Pre - condiciones	El FSC se ha identificado y autenticado ante el sistema. Se ha seleccionado una de las salidas transitorias planificadas previamente.
Post- condiciones	La orden de la salida transitoria planificada queda cancelada correctamente.

Funcionalidad	Actualizar Salida Transitoria
Actor	FSC
Breve descripción	Se selecciona una salida transitoria para modificar sus detalles y si se desea adicionar y/o eliminar individuos a partir de las órdenes de salida transitoria registradas en el sistema.
Pre - condiciones	El FSC se ha identificado y autenticado ante el sistema. Se ha seleccionado una salida transitoria planificada previamente.
Post- condiciones	La salida transitoria planificada es modificada correctamente

Funcionalidad	Consultar Salida Transitoria
Actor	FSC
Breve descripción	Se selecciona una salida transitoria y se indica que se desean ver los detalles de la misma, el sistema muestra los detalles.

Pre - condiciones	El FSC se ha identificado y autenticado ante el sistema. Se ha seleccionado una salida transitoria planificada previamente.
Post- condiciones	Los detalles de la salida transitoria planificada son consultados correctamente

Funcionalidad	Consultar historial de Salidas Transitorias
Actor	FSC
Breve descripción	Se muestran todas las salidas transitorias que han sido planificadas en el centro. Adicionalmente el Sistema permite especificar un período de tiempo válido para filtrar los resultados.
Pre - condiciones	El FSC se ha identificado y autenticado ante el sistema.
Post- condiciones	El FSC consulta el historial de salidas transitorias planificadas en el centro penitenciario.

Funcionalidad	Registrar ejecución de Salida Transitoria
Actor	FSC
Breve descripción	Se selecciona una orden de salida transitoria y se indica que se desea registrar la ejecución de la misma, se introducen los datos de la ejecución y se registra la misma.
Pre - condiciones	El FSC se ha identificado y autenticado ante el sistema. Se ha seleccionado una salida transitoria planificada previamente.
Post- condiciones	La ejecución de la salida transitoria queda correctamente registrada en el sistema.

Funcionalidad	Registrar conclusión de Salida Transitoria
Actor	FSC
Breve descripción	Se selecciona una salida transitoria que está en ejecución y se registra la confirmación de que ha concluido.
Pre - condiciones	El FSC se ha identificado y autenticado ante el sistema. Se ha seleccionado una salida transitoria en estado de ejecución.
Post- condiciones	La conclusión de la salida transitoria queda correctamente registrada en el sistema.

Funcionalidad	Registrar ejecución de traslado urgente a hospital.
Actor	FSC
Breve descripción	Se indica que desea registrar la ejecución de un traslado urgente a hospital. Se introducen los datos y los individuos que serán trasladados. El sistema registra el traslado urgente a hospital.
Pre - condiciones	El FSC se ha identificado y autenticado ante el sistema.
Post- condiciones	La ejecución de la salida transitoria queda correctamente registrada en el sistema.

TÉNICAS, MÉTODOS Y HERRAMIENTAS UTILIZADAS.

ARQUITECTURA DEFINIDA PARA EL SIGEP

La arquitectura del SIGEP está regida por una arquitectura de tres capas lógicas bien definidas. Cada capa puede ser modificada tanto como sea necesario sin provocar cambios en las demás. Una capa no tiene dependencias con la capa superior, cada capa depende solamente de la fachada que le permite comunicarse con la capa inmediata inferior. Esta dependencia entre capas es normalmente a través de fachadas, asegurando que el acoplamiento sea el más bajo posible y la abstracción del funcionamiento de la capa inferior, sea casi total.

Seguidamente se analizará en más detalles cada una de las capas propuestas anteriormente comenzando de abajo hacia arriba.

Capa de Acceso a Datos

En la arquitectura del SIGEP está definido que en la capa de acceso a datos se va a implementar el patrón DAO, por lo tanto, toda la lógica de acceso a datos de cada una de las entidades, se va a encapsular en clases que llevarán por nombre [NombreEntidad]DAOImpl, las cuales, implementarán sus interfaces correspondiente, donde se define el comportamiento de estas.

Esta interfaz debe heredar de una interfaz donde se definieron las operaciones básicas de (Crear, Leer, Actualizar y Borrar). La definición de esta interfaz es la siguiente:

```
package vnz.sigep.core.dataaccess.dao; import java.io.Serializable;
import java.util.List;
public interface BaseDAO<T, ID extends Serializable> {
        T findById(ID id, boolean lock);
        T persist(T entity);
        void delete(T entity);
        List<T> findAll();
        List<T> findByExample(T exampleInstance, String... excludeProperty);
}
```

Esta interfaz tiene métodos como persist (encargado de registrar o actualizar una instancia del objeto persistente) y delete (encargado de eliminar la instancia del objeto persistente de la Base de Datos). Además para implementar el DAO se debe especificar el tipo de dato de la clase persistente y el tipo de dato del identificador. Básicamente lo que se hace es separar operaciones primarias para el acceso a datos de las operaciones de este tipo relacionadas con el negocio de esa entidad en particular. Sin embargo si una entidad sólo necesitara operaciones básicas, es necesario hacer de todas formas una interfaz para ella, de lo contrario se lanzará una excepción. El código de la clase que implementa a BaseDAO es el siguiente:

```
package vnz.sigep.core.dataaccess.dao.impl;

public abstract class AbstractBaseDAO<T, ID extends Serializable> extends
HibernateDaoSupport implements BaseDAO<T,ID> {
	private Class<T> persistentClass;

	protected AbstractBaseDAO(Class<T> persistentClass) {
		this.persistentClass = persistentClass;
	}

	public Class<T> getPersistentClass() {
		return persistentClass;
	}

	public T findById(ID id, boolean lock){
		// Se implementa utlizando el HibernateTemplate
	}
	public T persist(T entity) {
		getHibernateTemplate().saveOrUpdate(entity);
		return entity;
	}

	public void delete(T entity) {
		getHibernateTemplate().delete(entity);
	}

	public List<T> findAll() {
		// Se implementa utlizando el HibernateTemplate
	}
	public List<T> findByExample(final T exampleInstance, final String... excludeProperty) {
		// Se implementa utlizando el HibernateTemplate
	}
	protected List<T> findByCriteria(final Criterion... criterion) {
		// Se implementa utlizando el HibernateTemplate
	}
	protected T findUniqueByCriteria(final Criterion... criterion) {
		// Se implementa utlizando el HibernateTemplate
```

```
        }
}
```

Existen algunas cosas interesantes en esta implementación, primero esta clase hereda de HibernateDAOSupport la cual, como se dijo anteriormente provee un HibernateTemplate para acceder al API de Hibernate. Nótese también que tiene un atributo de tipo Class, que representa el tipo de clase de la entidad persistente. La implementación concreta debe heredar de la clase AbstractBaseDAO, la cual le da acceso al HibernateDAOSupport para interactuar con el API de Hibernate y además implementar la interfaz concreta de su entidad.

Capa de Servicios de Negocio

Responsabilidad: Manejar la lógica de negocio perteneciente al módulo y definir los métodos de clases que serán envueltos en contextos transaccionales. Las políticas transaccionales de la aplicación serán planteadas sobre los objetos de negocio en el fichero de configuración de Spring perteneciente a cada módulo.

Componentes de la capa de Servicios de Negocio

En esta capa radican dos componentes fundamentales:

Las Entidades de dominio, las cuales contendrán solamente sus atributos y los métodos de acceso a los mismos, de esta manera, pueden moverse verticalmente por las capas de la aplicación, como contenedores de datos, sin ninguna otra responsabilidad.

Los manejadores o Managers, Son clases donde se ha encapsulado la lógica de negocio correspondiente a entidades de dominio.

Por cada módulo se definió una o más fachadas en caso de que se requiera que agrupen los métodos de negocio implementados en los manejadores o Managers que serán explicados más adelante. La fachada de un módulo se basa en el patrón Facade para permitir una clara división entre las capas arquitectónicas. Los Managers son las clases que se especializan en un conjunto de funcionalidades que representan el negocio sobre una o varias entidades. Los Managers son las únicas clases en la aplicación que tendrán lógica de negocio mientras que las fachadas se limitarán solamente a agrupar las funcionalidades para ser expuestas a capas superiores.

Capa de Presentación

Esta capa será fina y no contendrá lógica de negocio, sino simplemente lo concerniente a aspectos de presentación, por ejemplo, el código para manipular las interacciones web. Esta

capa puede variar en complejidad en el transcurso del desarrollo de la aplicación, sin embargo ninguna de las capas inferiores debe ser afectada por estos cambios.

PATRÓN DE DISEÑO.

Un patrón de diseño es una solución a un problema de diseño. Para que una solución sea considerada un patrón debe poseer ciertas características. Una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores. Otra es que debe ser reusable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias.

Singleton:

Se utiliza para garantizar que una clase solo tiene una única instancia, proporcionando un punto de acceso global a la misma.

En el SIGEP se utiliza a través del framework Spring, pues este tiene como opción por defecto que todos los beans que se declaran en el fichero de configuración cumplan con este patrón.

Fachada:

Este patrón pertenece al grupo de patrones estructurales de GoF y tiene como objetivo simplificar el acceso a un conjunto de objetos proporcionando uno que todos los clientes pueden usar para comunicarse con el conjunto. Una fachada, conoce qué componentes son responsables de qué peticiones y así, es capaz de delegarlas a los componentes apropiados.

En el SIGEP se utilizan las fachadas para crear un punto único de acceso entre dos capas, de esta manera se define el punto de entrada a cada nivel respectivamente, así se simplifican las dependencias obligando a cada uno de los objetos a comunicarse con el nivel que está por debajo a través de ellas.

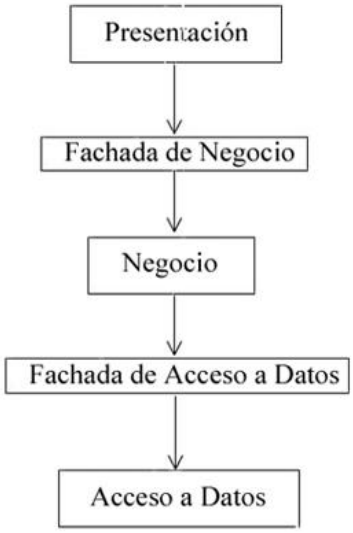

Figura 1: Representación del patrón Fachada

DAO (Data Access Object)

Este patrón pertenece al grupo de patrones de integración de CORE y se usa para abstraer y encapsular la lógica de acceso a datos en una clase. El Objeto de Acceso a Datos abstrae a las clases del negocio de la implementación de acceso a datos y habilita el acceso transparente a la fuente de datos. El objeto de negocio también delega las operaciones de cargar y persistir en los DAO, solo se encargara de pedirle a estos los métodos correspondientes.

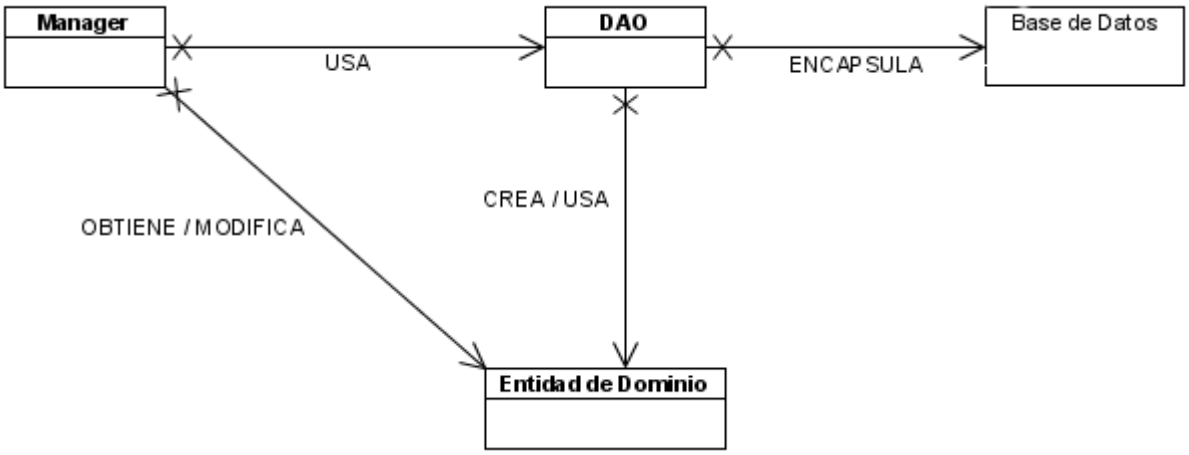

Figura 2: Representación del Patrón DAO

En la aplicación hay un DAO por cada entidad de dominio, este se nombrara [nombre entidad]DAO en el que se implementa toda la lógica de acceso a datos referente a la entidad.

TECNOLOGÍAS:

Java:

Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros.

El lenguaje Java se creó con cinco objetivos principales:

1. Debería usar la metodología de la programación orientada a objetos.
2. Debería permitir la ejecución de un mismo programa en múltiples sistemas operativos.
3. Debería incluir por defecto soporte para trabajo en red.
4. Debería diseñarse para ejecutar código en sistemas remotos de forma segura.
5. Debería ser fácil de usar y tomar lo mejor de otros lenguajes orientados a objetos, como C++.

Una de las principales características de este lenguaje es que es orientado a objetos, esta se refiere a un método de programación y al diseño del lenguaje. Cuando se utiliza la Programación Orientada a Objetos (POO) se diseña el software de manera que los distintos tipos de datos que use estén unidos a sus operaciones. Así, los datos y el código (funciones o métodos) se combinan en entidades llamadas objetos. Un objeto puede verse como un paquete que contiene el “comportamiento” (el código) y el “estado” (datos). Otra característica es la independencia de la plataforma, esto significa que programas escritos en Java pueden ejecutarse igualmente en cualquier tipo de hardware independientemente del sistema operativo que se esté usando. Para ello, se compila el código fuente escrito en lenguaje Java, para generar un código conocido como “bytecode” (específicamente Java bytecode), instrucciones máquina simplificadas específicas de la plataforma Java. Esta pieza está “a medio camino” entre el código fuente y el código máquina que entiende el dispositivo destino. El bytecode es ejecutado entonces en la máquina virtual (VM), un programa escrito en código nativo de la plataforma destino (que es el que entiende su hardware), que interpreta y ejecuta el código. Además, se suministran librerías adicionales para acceder a las características de cada dispositivo (como los gráficos, ejecución mediante hebras o threads, la interfaz de red) de forma unificada. Una ventaja que tiene Java sobre lenguajes como C++ es el recolector de basura, en C++ el programador se encuentra con la carga de tener que administrar la memoria solicitada dinámicamente de forma manual. En Java, este problema potencial es evitado en gran medida por el recolector automático de basura. El programador

determina cuándo se crean los objetos y el entorno en tiempo de ejecución de Java es el responsable de gestionar el ciclo de vida de los objetos.

Framework:

Según la “pandilla de cuatro” o en inglés “gang of four” (GoF) “un framework es un conjunto de clases que cooperan entre sí para lograr un diseño reusable para una clase especifica de software” (3). Un framework provee una guía arquitectónica al dividir el diseño en clases abstractas y definir sus responsabilidades y colaboraciones. El desarrollador adapta el framework a una aplicación específica al heredar y utilizar instancias de las clases del mismo.

Spring:

Spring es un framework muy popular en estos días que a diferencia de otros existentes para trabajar con J2EE interviene en todas las capas de la aplicación. A pesar de que Spring Framework no obliga a usar un modelo de programación en particular, se ha popularizado en la comunidad de programadores en Java al considerársele una alternativa y substituto del modelo de Enterprise JavaBean. Por su diseño el framework ofrece mucha libertad a los desarrolladores en Java y soluciones muy bien documentadas y fáciles de usar para las prácticas comunes en la industria. Spring Framework hizo que aquellas técnicas que resultaban desconocidas para la mayoría de programadores se volvieran populares en un periodo muy corto de tiempo, el ejemplo más notable es la inyección de dependencias. En el año 2004, Spring disfrutó de unas altísimas tasas de adopción y al ofrecer su propio framework de programación orientada a aspectos (aspect-oriented programming, AOP) consiguió hacer más popular su paradigma de programación en la comunidad Java. Spring tiene muchas ventajas como por ejemplo:

Spring es ligero tanto en su tamaño como a en los recursos que consume. La mayor parte de este puede ser distribuido en un jar que pesa solo un poco más de 2.5 MB.

Spring es un contenedor si se mira el hecho de que este contiene y maneja el ciclo de vida y la configuración de los objetos de aplicación. En Spring el desarrollador declara como cada uno de los objetos de aplicación debería ser creado, como deberían ser configurados y como deberían estar asociados entre ellos.

Spring provee una estructura consistente para el acceso a datos que se integra ya sea con Java Data Base Connectivity o con otros frameworks tales como Hibernate o Ibatis.

Spring incluye un poderoso y altamente configurable framework MVC.

La inyección de dependencias que radica en resolver las dependencias de cada clase (atributos) generando los objetos cuando se arranca la aplicación y luego inyectarlos en los demás objetos que los necesiten a través de métodos set o bien a través del constructor, pero estos objetos se instancian una vez, se guardan en una factoría y se comparten por todas las clases. (1)

Hibernate:

Hibernate es un potente framework de mapeo objeto/relacional y servicio de consultas para Java. Es la solución ORM (Object-Relational Mapping) más popular en el mundo Java. Permite diseñar objetos persistentes que podrán incluir polimorfismo, relaciones, colecciones, y un gran número de tipos de datos. De una manera muy rápida y optimizada se puede generar BBDD en cualquiera de los entornos soportados: Oracle, DB2, MySQL y otros.

Sus principales características son:

• Permite expresar consultas utilizando SQL nativo o consultas basadas en criterios.

• Puede operar proporcionando persistencia de una manera transparente para el desarrollador.

•Se integra con el paradigma de POO de una manera natural: herencia, polimorfismo, asociación, agregación y composición y el API de colecciones de Java.

• Permite inicialización perezosa (lazy) de objetos y colecciones.

• Proporciona el lenguaje HQL el cual provee una independencia del SQL de cada base de datos, tanto para el almacenamiento de objetos como para su recuperación.

• Permite utilizar los diversos tipos de generación de identificadores que proporcionan los sistemas gestores de bases de datos así como generación independiente de la base de datos, incluyendo identificadores asignados por la aplicación o claves compuestas. (2)

JUnit

JUnit es un framework de código abierto para pruebas en java originalmente escrito por Kent Beck y Erich Gamma. Con JUnit se logra ejecutar pruebas automatizadas de manera fácil, permite evaluar si el funcionamiento de uno o varios métodos se comporta como se espera, en función de algún valor de entrada se evalúa el valor de retorno para verificar que sea el esperado. Se integra con muchos IDEs como NetBeans, BlueJ, IntelliJ, JBuilder, Eclipse y otros.

HERRAMIENTAS EMPLEADAS:

Eclipse 3.3:

Eclipse es un Entorno de Desarrollo Integrado (IDE) de código abierto. Existen otras aplicaciones que pueden ser integradas a eclipse en forma de plugins, siendo reconocidos automáticamente al iniciarse éste, uno de los principales es el JDT (Java Development Toolkit) que se utiliza para desarrollar aplicaciones en java, también existen plugins para C++, php y otros. Eclipse corre sobre varios sistemas operativos incluyendo Windows y Linux.(6)

Spring IDE:

Spring IDE es un plugin para eclipse que sirve como interfaz de usuario gráfica para la configuración de los archivos usados por Spring Framework. Permite el completamiento de etiquetas, valores de atributos y elementos en estos archivos de configuración.

Muestra un árbol con todos los proyectos de Spring y sus archivos de configuración. Permite hacer búsquedas de beans en dichos archivos. Muestra gráficamente los beans y sus dependencias.

Presenta un editor gráfico con completamiento y validaciones para los archivos de definiciones del flujo web usado por el Spring Web Flow. (10)

Hibernate Tools:

Hibernate Tools es un plugin para Eclipse utilizado para trabajar con el framework Hibernate.

Este plugin tiene las siguientes características disponibles:

• Editor de mapeos: Es un editor para los archivos de mapeos XML de Hibernate, permitiendo auto completamiento y sintaxis resaltada.

• Consola: La perspectiva de consola de Hibernate permite configurar conexiones a base de datos, provee visualización de clases y sus relaciones. Admite además ejecutar preguntas en formato del lenguaje de preguntas de Hibernate (HQL) interactivamente contra la base de datos y mostrar los resultados.

• Ingeniería inversa: Es su característica más importante, permitiendo generar las clases del modelo de dominio y los archivos de mapeos de Hibernate a partir de una base de datos.

• Asistentes: Presenta asistentes como: generador de archivos de configuración rápidamente, configuración de la consola, entre otros. (9)

WTP (Web Tools Platform)

La plataforma de herramientas web de Eclipse o Eclipse Web Tools Platform (WTP) provee varias APIs para desarrollo de aplicaciones sobre la web y JEE. Estas incluyen editores gráficos, de código fuente para una variedad de lenguajes, además de herramientas y APIs para permitir el despliegue, ejecución y prueba de aplicaciones.

Se integra con servidores web dentro de Eclipse como ambiente de ejecución de primera clase para aplicaciones web. También incluye la configuración de servidores y su asociación con los proyectos web. (7)

Subclipse:

Subclipse es un plugin para Eclipse que le permite integrarse con la herramienta para el control de versiones Subversion, permitiendo operaciones de sincronización, actualización, entre otras. Dispone de una vista de comparación entre el recurso local y el remoto que puede ser de gran ayuda en caso de que exista conflicto entre las versiones de estos. Muestra una vista del historial de versiones de los recursos con un conjunto de atributos de las acciones realizadas sobre cada uno.

Es un plugin muy útil para el desarrollo colaborativo, es decir, para situaciones en las que se encuentran un conjunto de desarrolladores trabajando sobre el mismo proyecto. (8)

Apache Tomcat

Apache Tomcat es un contenedor de Servlet usado en la implementación de referencia oficial para las tecnologías Java Servlet y Java Server Pages (JSP). Es desarrollado en un ambiente participativo y abierto.

Apache Tomcat es usado en numerosas aplicaciones web de gran escala y críticas en diversas industrias y organizaciones que se referencian en su sitio oficial.

La versión 5.x es implementada a partir de las especificaciones Servlet 2.4 y JSP 2.0.

FLUJOS DE TRABAJO

Para realizar el diseño e implementación de los módulos se han propuesto un conjunto de actividades a seguir por los roles de Diseñador, Implementador de lógica de negocio e Implementador de acceso a datos, a este conjunto de actividades se les denomina flujo de trabajo.

Flujo de trabajo diseño:

- Analizar los artefactos de entrada recibidos (Descripción de funcionalidades y prototipo)
- Identificar las clases de dominio.
- Identificar las clases persistentes.
- Refinar el modelo de datos.
- Realizar diagrama de clases de las entidades de dominio.
- Definir las interfaces de las Fachadas, los Managers y los DAOs.

- Realizar diagramas de clases en los que se representen las Fachadas, los Managers y los DAOs.
- Realizar diagramas de secuencias de las funcionalidades.
- Declarar las Fachadas, los Managers y los DAOs.

Flujo de trabajo de Implementador de lógica de negocio

- Declarar todas las entidades de dominio.
- Implementar las fachadas y configurarlas en el xml de configuración de Spring.
- Implementar los Managers y configurarlos en el xml de configuración de Spring.
- Realizar pruebas unitarias a la implementación de los Managers.

Flujo de trabajo Implementador de acceso a datos

- Crear ficheros de mapeo.
- Implementar los DAOs y configurarlos en el xml de configuración de Spring.
- Realizar pruebas unitarias a la implementación de los DAOs.

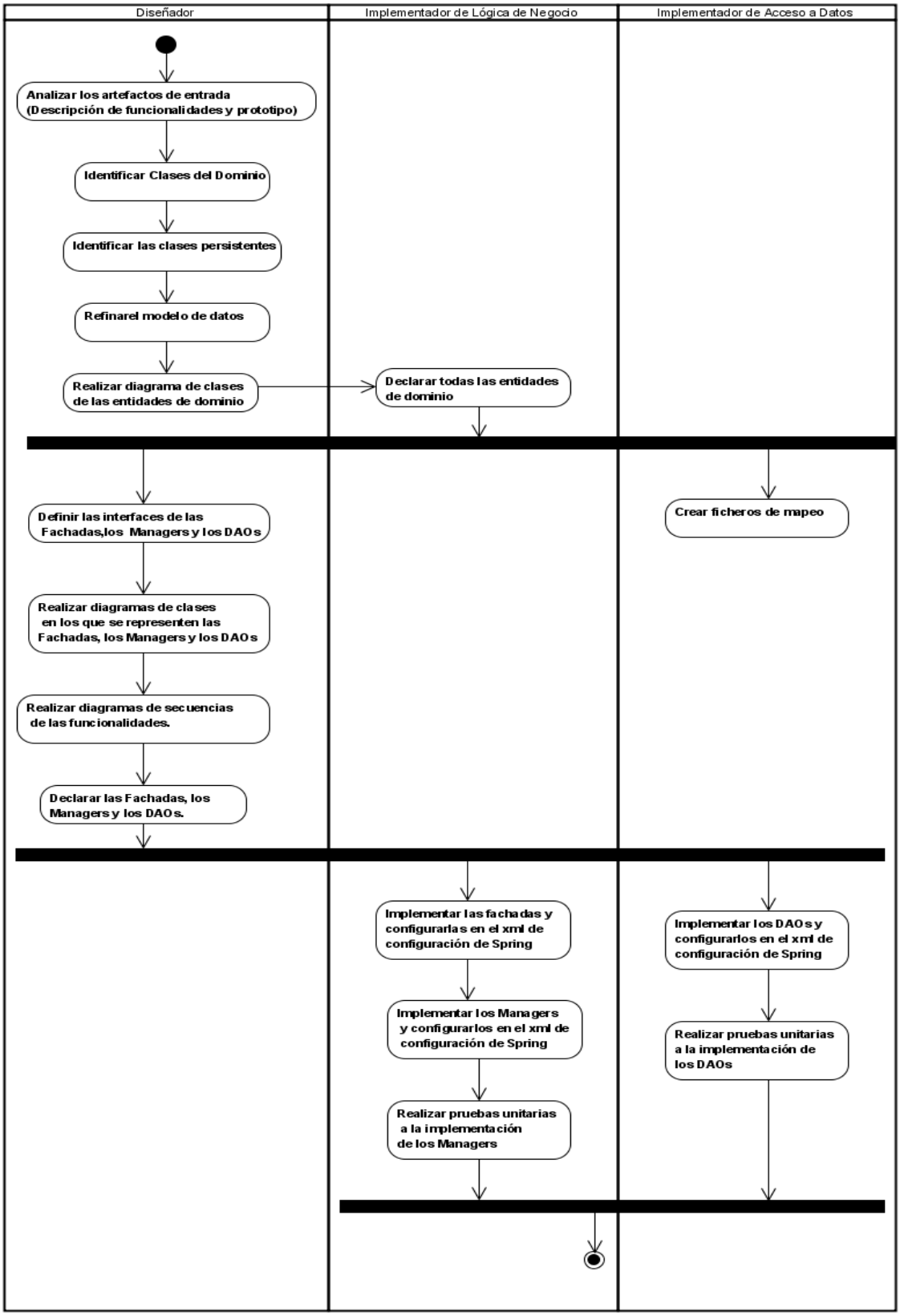

Figura 3: Diagrama de Actividad de los Flujos de Trabajo.

DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

Clases del Dominio: En el diseño de la solución se definieron las clases de dominio teniendo en cuenta las tablas de la Base de Datos y otros factores. Estas clases no contendrán ninguna lógica de negocio, de esta manera pueden servir como objetos de transferencia y moverse por todas las capas de la aplicación.

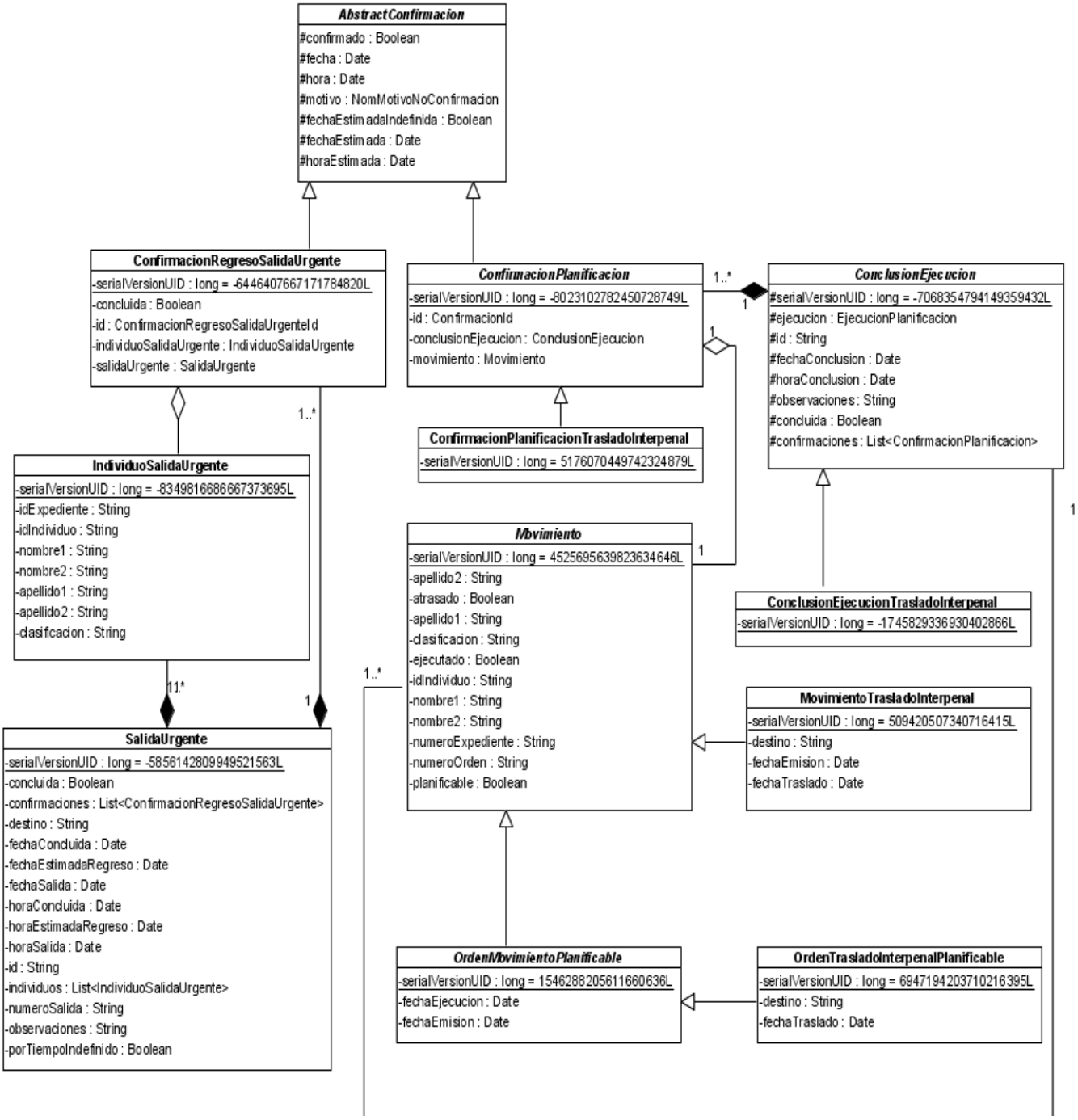

Figura 4: Diagrama de clases del dominio (parte 1)

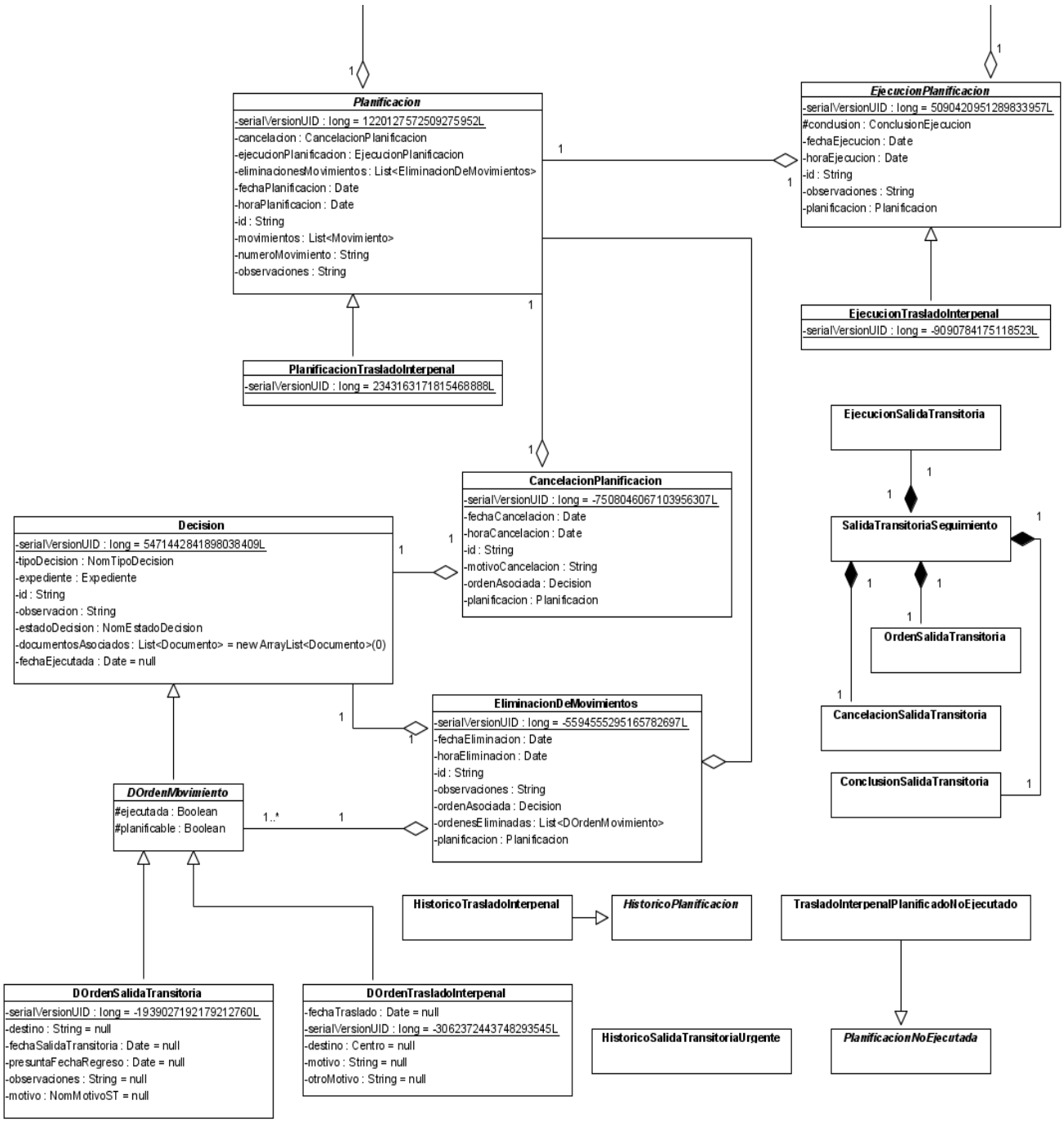

Figura 5: Diagrama de clases del dominio (parte 2)

Clases del Negocio: La lógica del negocio se implementa en las clases llamadas “Manager”, estas se comunican con la capa de presentación a través de las fachadas. Este diseño se realizó teniendo en cuenta los patrones estudiados.

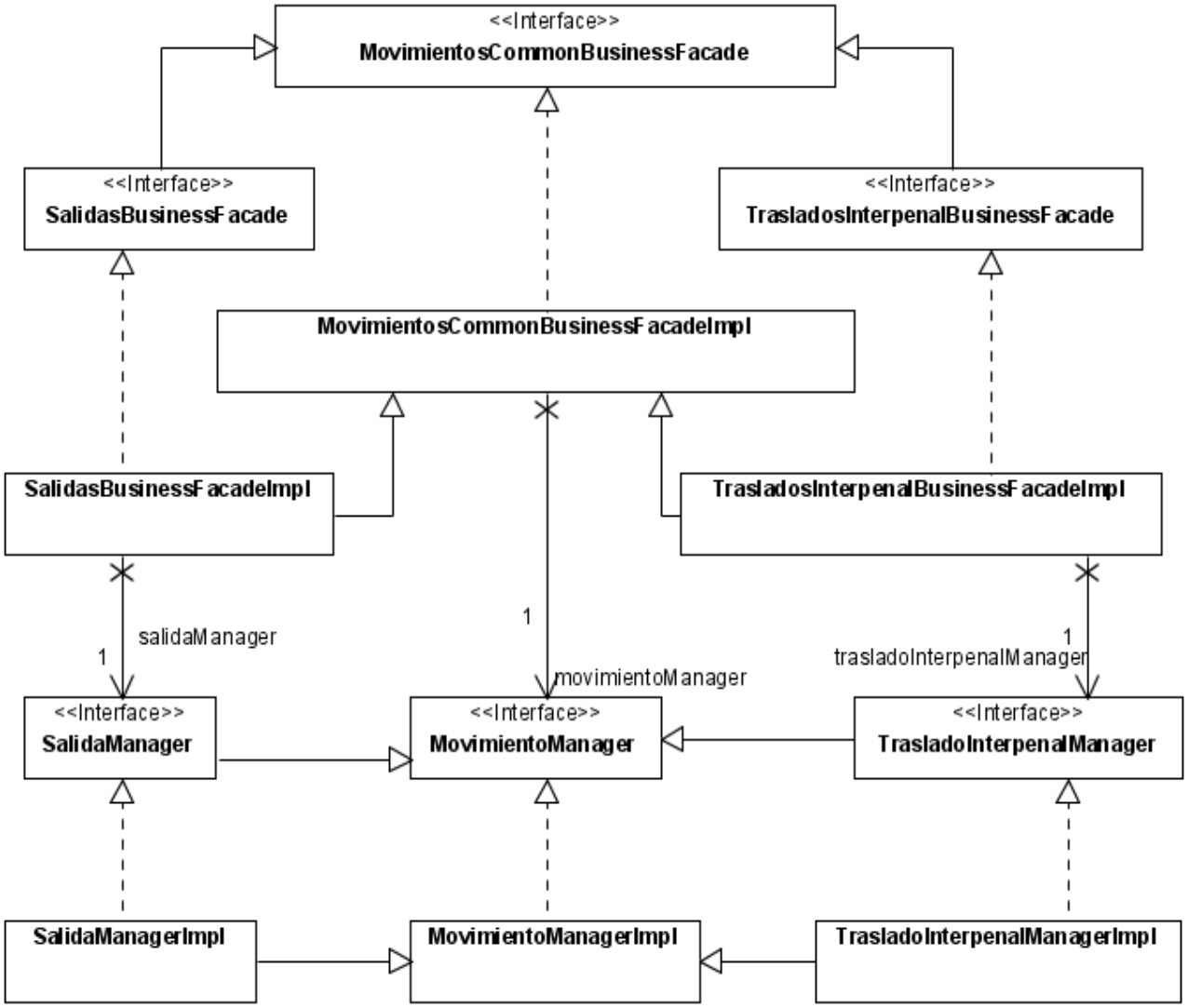

Figura 6: Diagrama de clases del negocio

Clases Acceso a Datos: En las clases de acceso a datos se ha declarado un DAO por cada clase de dominio, estas clases son las encargadas de toda la lógica de acceso a datos. La interacción de esta capa con la capa de negocio es a través de una fachada que termina en DataAccesFacade, de esta manera se cumple con el patrón Fachada.

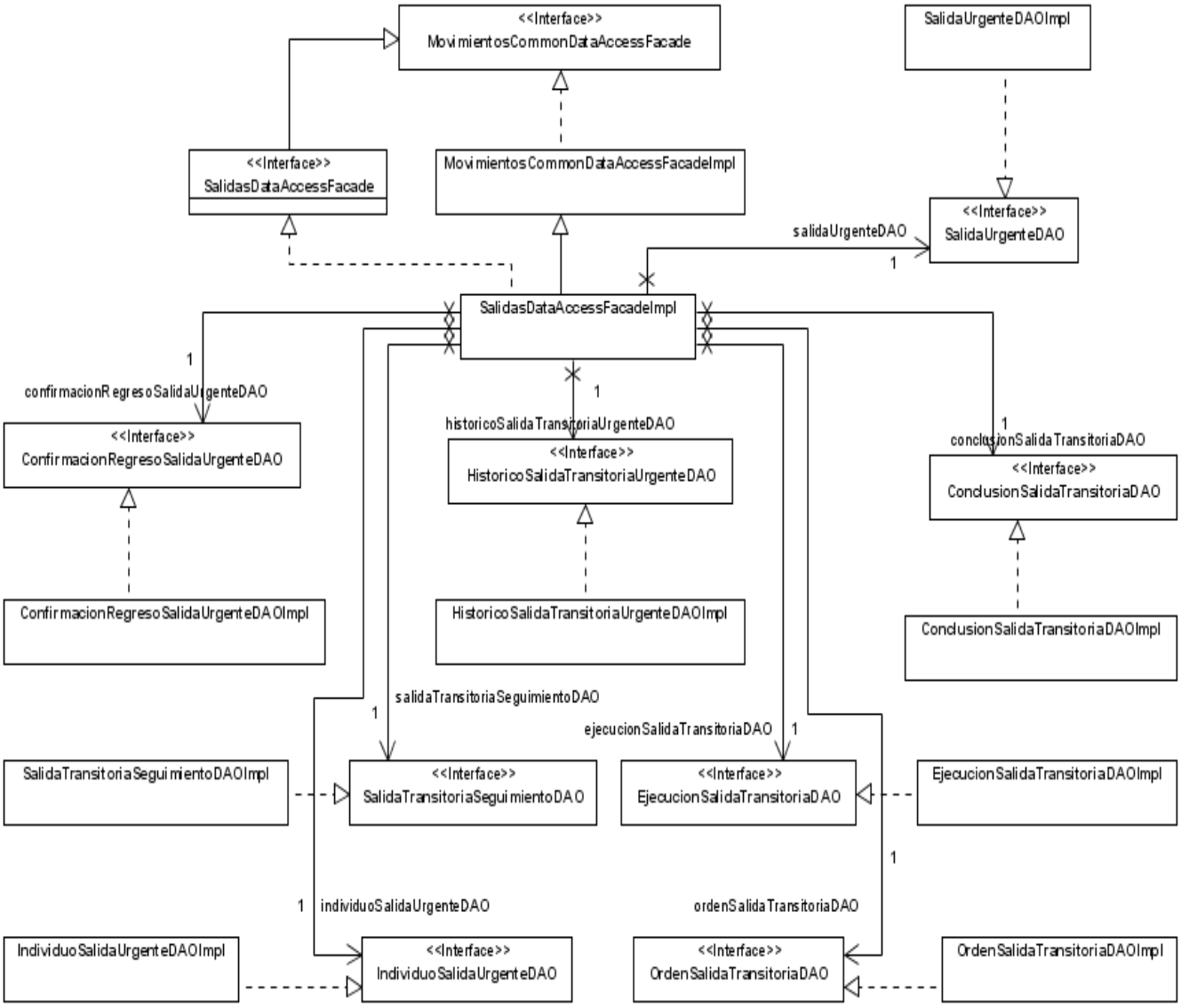

Figura 7: Diagrama de clases de acceso a datos (Parte1)

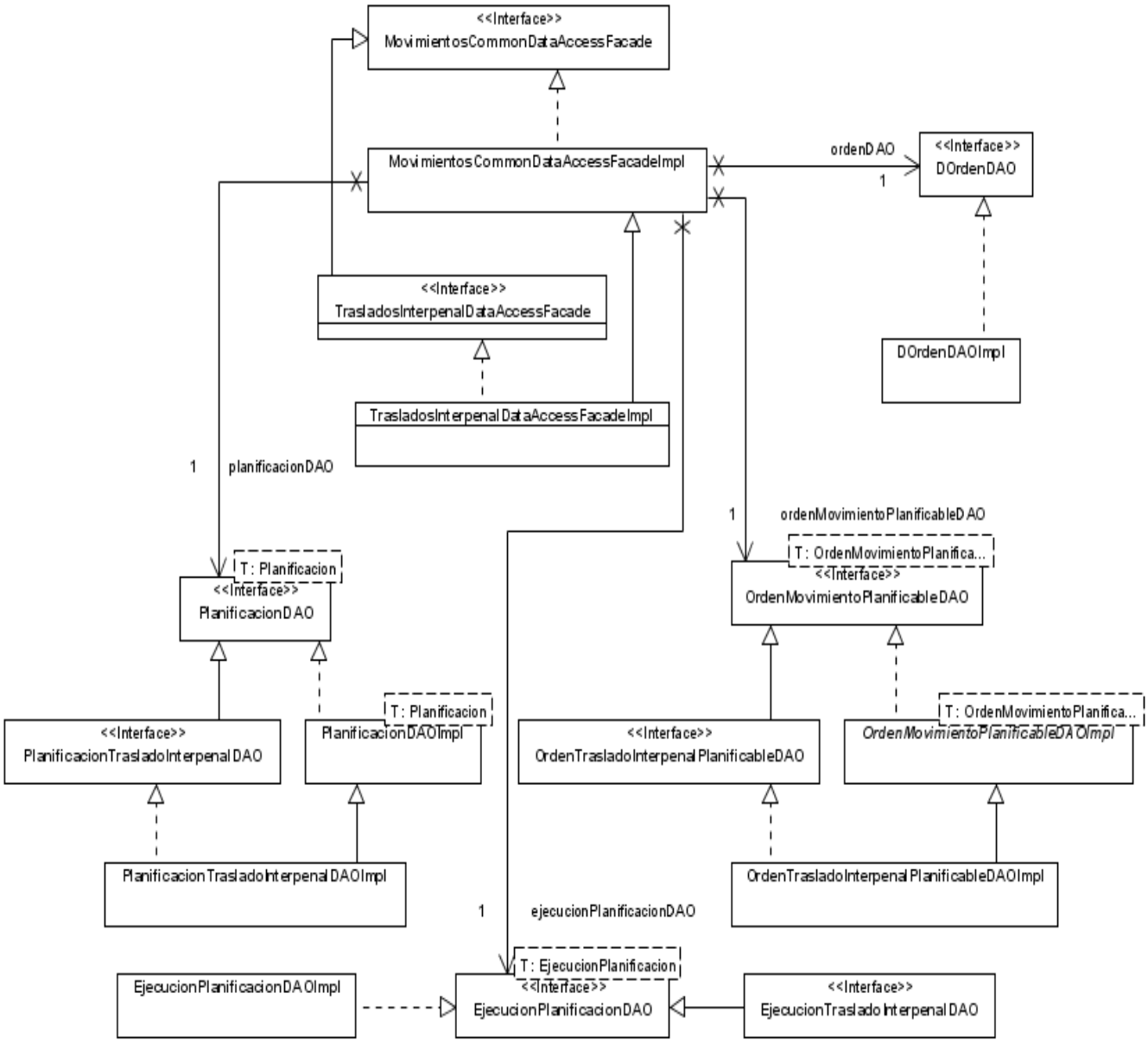

Figura 8: Diagrama de clases de acceso a datos (Parte2)

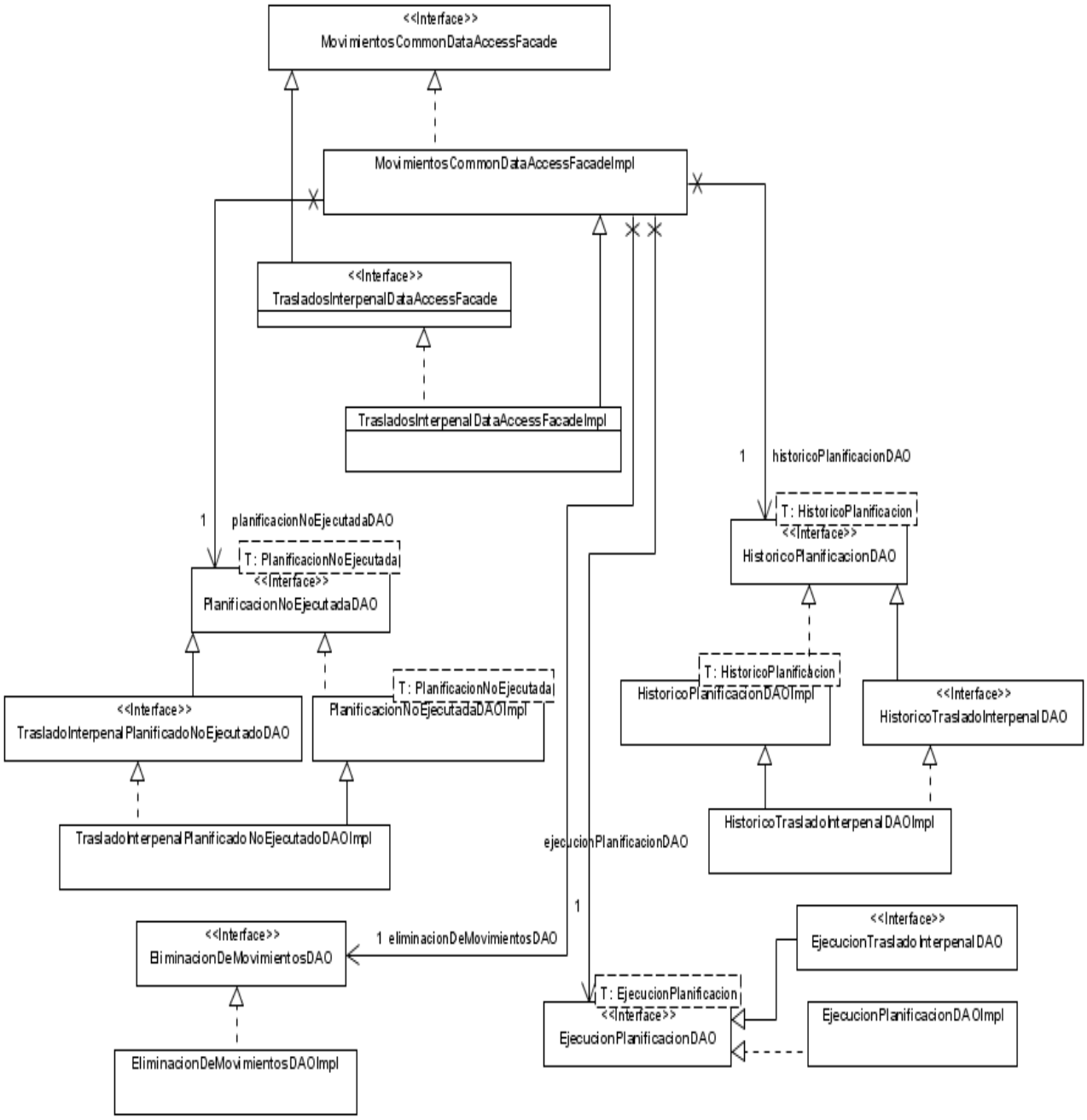

Figura 9: Diagrama de clases de acceso a datos (Parte3)

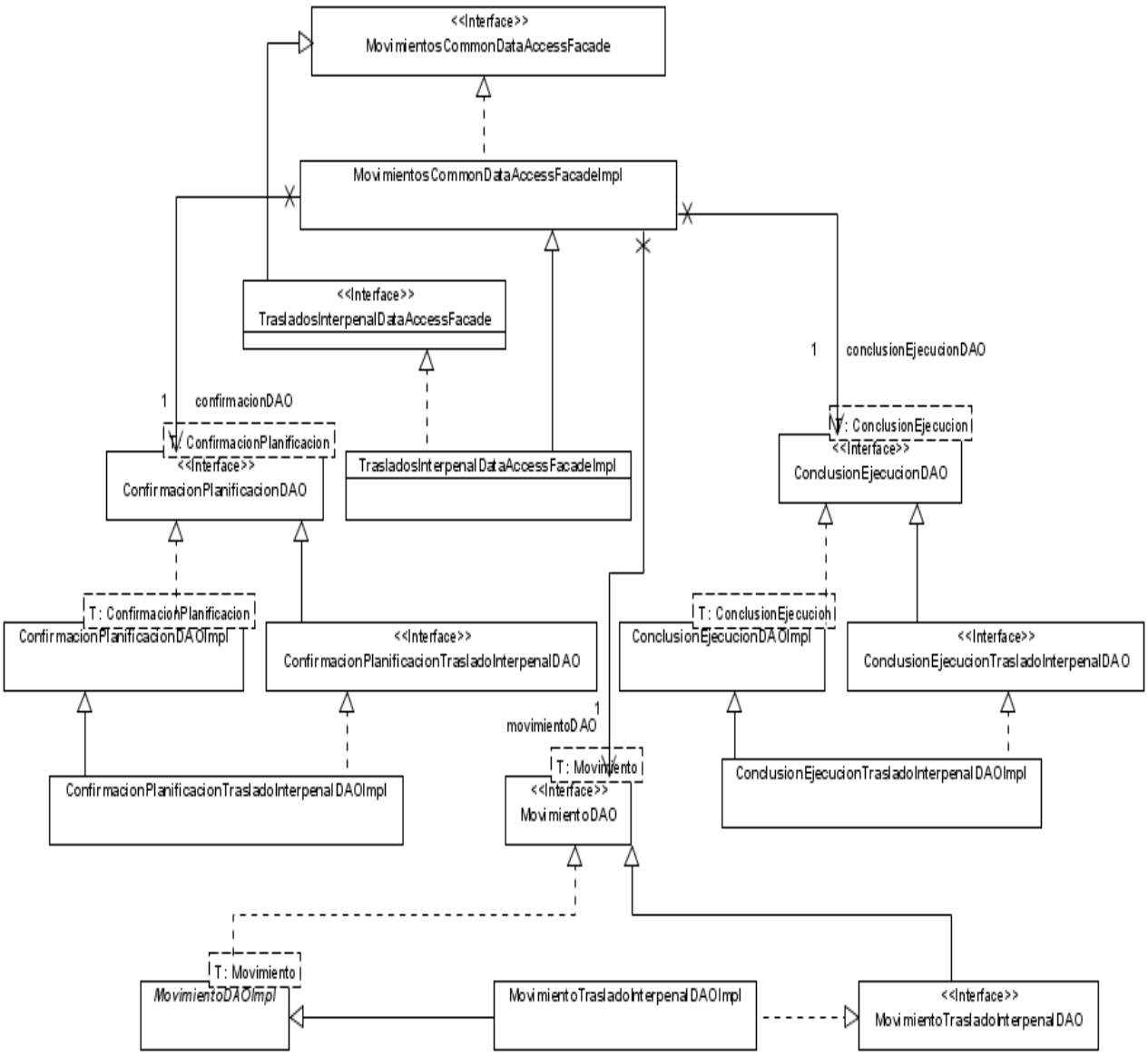

Figura 10: Diagrama de clases de acceso a datos (Parte4)

Descripción de la funcionalidad Planificar Traslado Interpenal:

Diagrama de Clases: En el siguiente diagrama se representan las clases que intervienen en la funcionalidad mencionada.

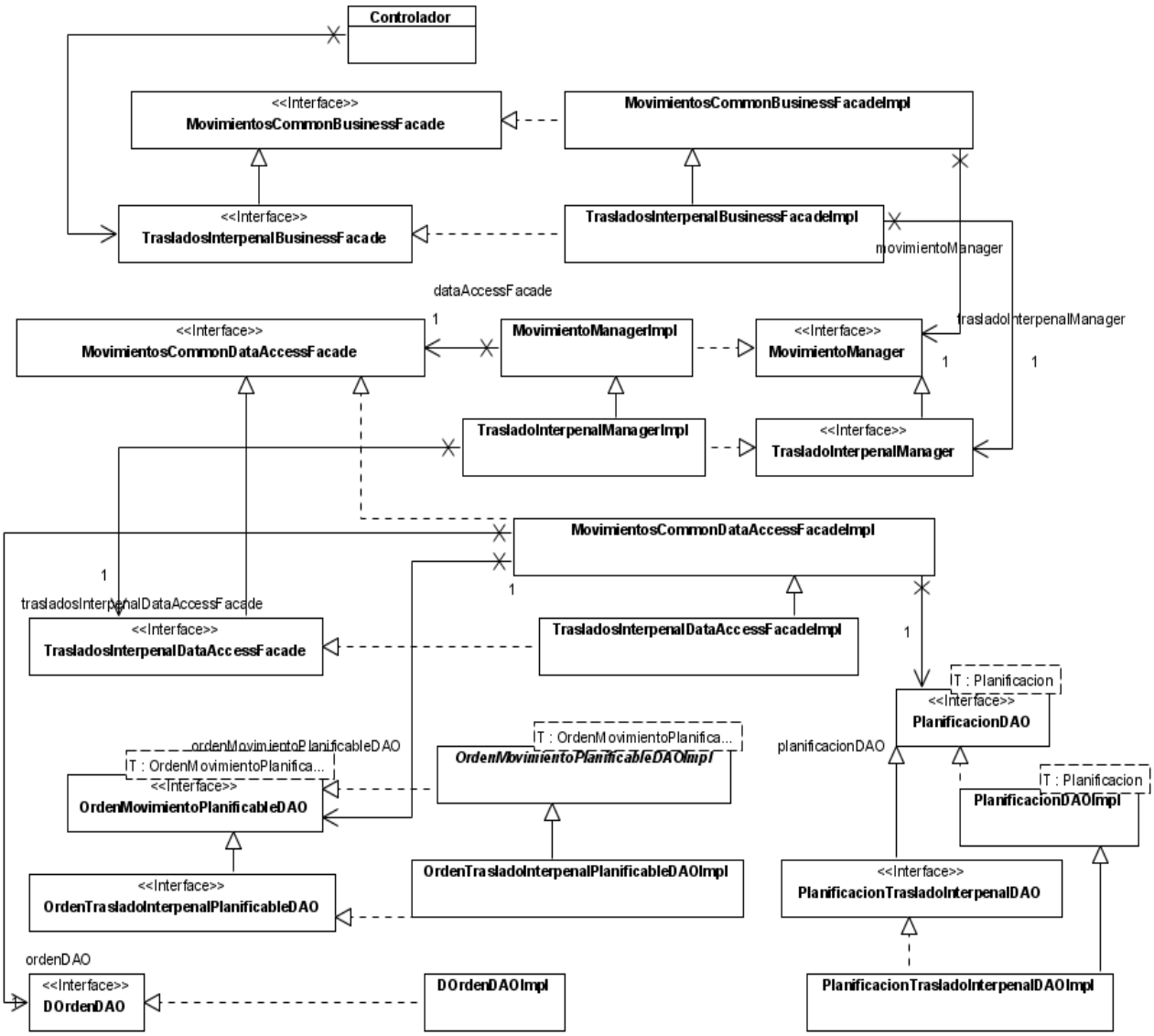

Figura 11: Diagramas de clases planificar traslado interpenal

En el diagrama de clases representado en la figura 11 se representan las clases que intervienen en la funcionalidad planificar Traslado Interpenal y las relaciones entre ellas.

Controlador: Clase que representa la capa de presentación.

MovimientosCommonBusinessFacade: Interfaz que expone a la capa de presentación las funcionalidades comunes tanto para Salidas Transitorias como para Traslados Interpenal.

MovimientosCommonBusinessFacadeImpl: Clase que implementa las funcionalidades definidas en la interfaz "*MovimientosCommonBusinessFacade*". En esta no se implementa ninguna lógica de negocio, sólo se llaman los métodos necesarios de la clase manager correspondiente.

TrasladosInterpenalBusinessFacade: Interfaz de la capa de lógica de negocio del módulo Traslados Interpenal, que contiene las funcionalidades relacionadas con el modulo brindadas a la capa de presentación. Hereda de la interfaz "*MovimientosCommonBusinessFacade*".

TrasladosInterpenalBusinessFacadeImpl: Clase que implementa las funcionalidades definidas en la interfaz "*TrasladosInterpenalBusinessFacade*". En esta no se implementa ninguna lógica de negocio, sólo se llaman los métodos necesarios de la clase manager correspondiente.

MovimientoManager: Interfaz que define los métodos de lógica de negocio comunes a los módulos de Traslados Interpenal y Salidas transitorias.

MovimientoManagerImpl: Clase que implementa la lógica de negocio de los métodos definidos en la interfaz "*MovimientoManager*".

TrasladoInterpenalManager: Interfaz que define las funcionalidades de lógica de negocio especificas para los traslados interpenal. Hereda de la interfaz "*MovimientoManager*".

TrasladoInterpenalManagerImpl: Interfaz que implementa los métodos de lógica de negocio definidos en la interfaz "*TrasladoInterpenalManager*". Hereda de la clase "*TrasladoInterpenalManagerImpl*".

MovimientosCommonDataAccessFacade: Interfaz que expone las funcionalidades de la capa de acceso a datos comunes para Traslados Interpenal y Salidas Transitorias.

MovimientosCommonDataAccessFacadeImpl: Clase que implementa los métodos definidos en la interfaz "*MovimientosCommonDataAccessFacade*". En esta clase no se implementa la lógica de acceso a datos, sólo se accede a los métodos de los DAOs correspondientes.

TrasladosInterpenalDataAccessFacade: Interfaz que expone los métodos de acceso a datos específicos de Traslados Interpenal. Hereda de la interfaz "*MovimientosCommonDataAccessFacade*".

TrasladosInterpenalDataAccessFacadeImpl: Clase que implementa los métodos definidos en la interfaz "*TrasladosInterpenalDataAccessFacade*". Hereda el comportamiento de la clase "TrasladosInterpenalDataAccessFacadeImpl". En esta clase no se implementa la lógica de acceso a datos, sólo se accede a los métodos de los DAOs correspondientes.

OrdenMovimientoPlanificableDAO: Interfaz que expone los métodos relacionados con el acceso a datos de la entidad de negocio "*OrdenMovimientoPlanificable*", esta entidad no está mapeada con una tabla de la base de datos, se genera dinámicamente a partir de una consulta.

OrdenMovimientoPlanificableDAOImpl: Clase que implementa los métodos definidos en la interfaz "*OrdenMovimientoPlanificableDAO*".

OrdenTrasladoInterpenalPlanificableDAO: Interfaz que expone los métodos relacionados con el acceso a datos de la entidad de negocio "*OrdenTrasladoInterpenalPlanificable*", esta entidad no

está mapeada a una tabla de la base de datos, se genera dinámicamente a partir de una consulta. Hereda de "*OrdenMovimientoPlanificableDAO*".

OrdenTrasladoInterpenalPlanificableDAOImpl: Clase que implementa los métodos definidos por la interfaz "*OrdenTrasladoInterpenalPlanificableDAO*" hereda de "*OrdenMovimientoPlanificableDAOImpl*".

PlanificacionDAO: Interfaz que expone los métodos relacionados con el acceso a datos de la entidad persistente "*Planificacion*" que mapea con la tabla de la base de datos "*Planificacion*".

PlanificacionDAOImpl: Clase que implementa los métodos definidos por la interfaz "*PlanificacionDAO*".

PlanificacionTrasladoInterpenalDAO: Clase que expone los métodos relacionados con el acceso a datos de la entidad persistente "*PlanificacionTrasladoInterpenal*" que está mapeada con la tabla de la base de datos "*TrasladoInterpenal*". Hereda de "*PlanificacionDAO*".

PlanificacionTrasladoInterpenalDAOImpl: Clase que implementa los métodos definidos por la interfaz "*PlanificacionTrasladoInterpenalDAO*". Hereda de "*PlanificacionDAOImpl*".

DOrdenDAO: Interfaz que expone los métodos relacionados con el acceso a datos de la entidad persistente "*Decision*" que mapea con la tabla de la base de datos "*DECISION*".

DOrdenDAOImpl: Clase que implementa los métodos definidos por la interfaz "*DOrdenDAO*".

Diagrama de secuencia

A continuación se muestra un diagrama de secuencia que muestra el flujo a seguir para planificar un traslado interpenal, dicho flujo cumple con todas las pautas definidas en la arquitectura base del SIGEP. De esta manera se puede ver como se aplica el patrón Fachada para separar las capas y el patrón DAO para separar la lógica de acceso a datos.

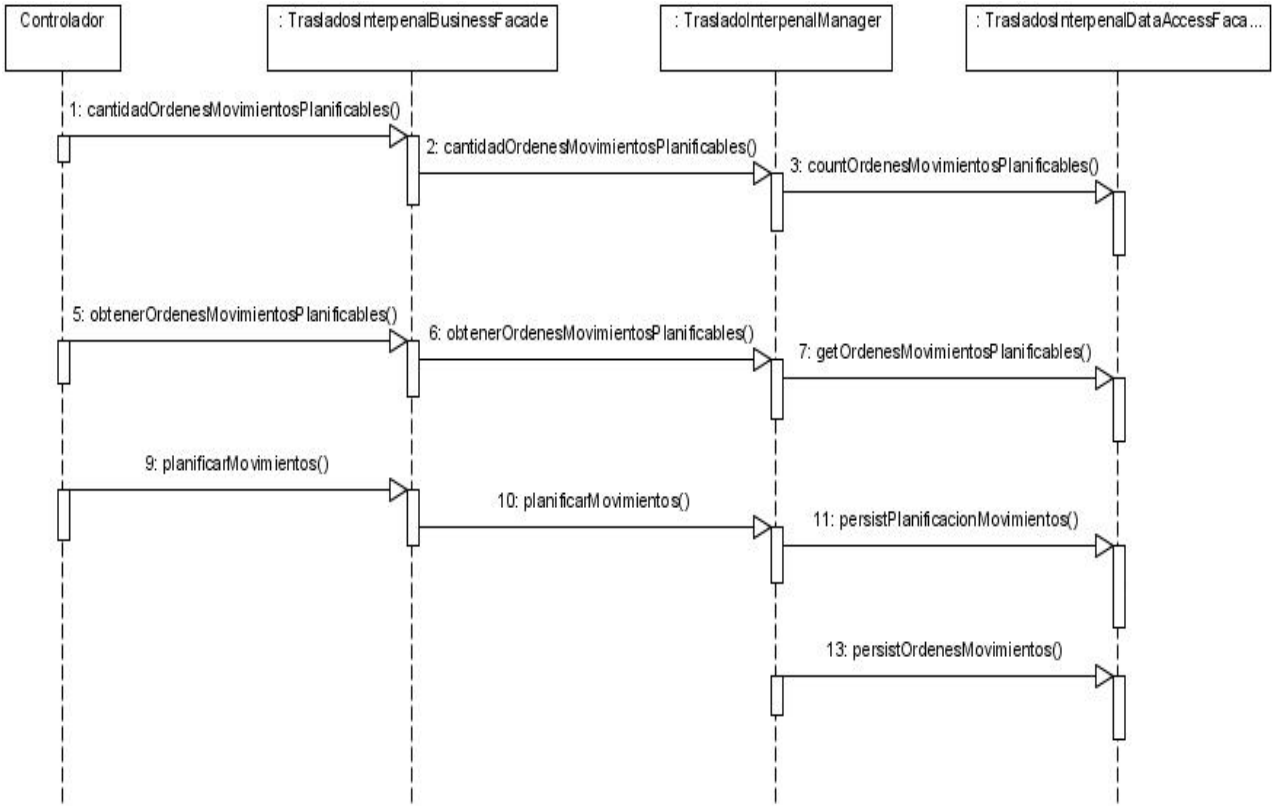

Figura 12: Diagrama de secuencia de la funcionalidad planificar traslado interpenal (parte1)

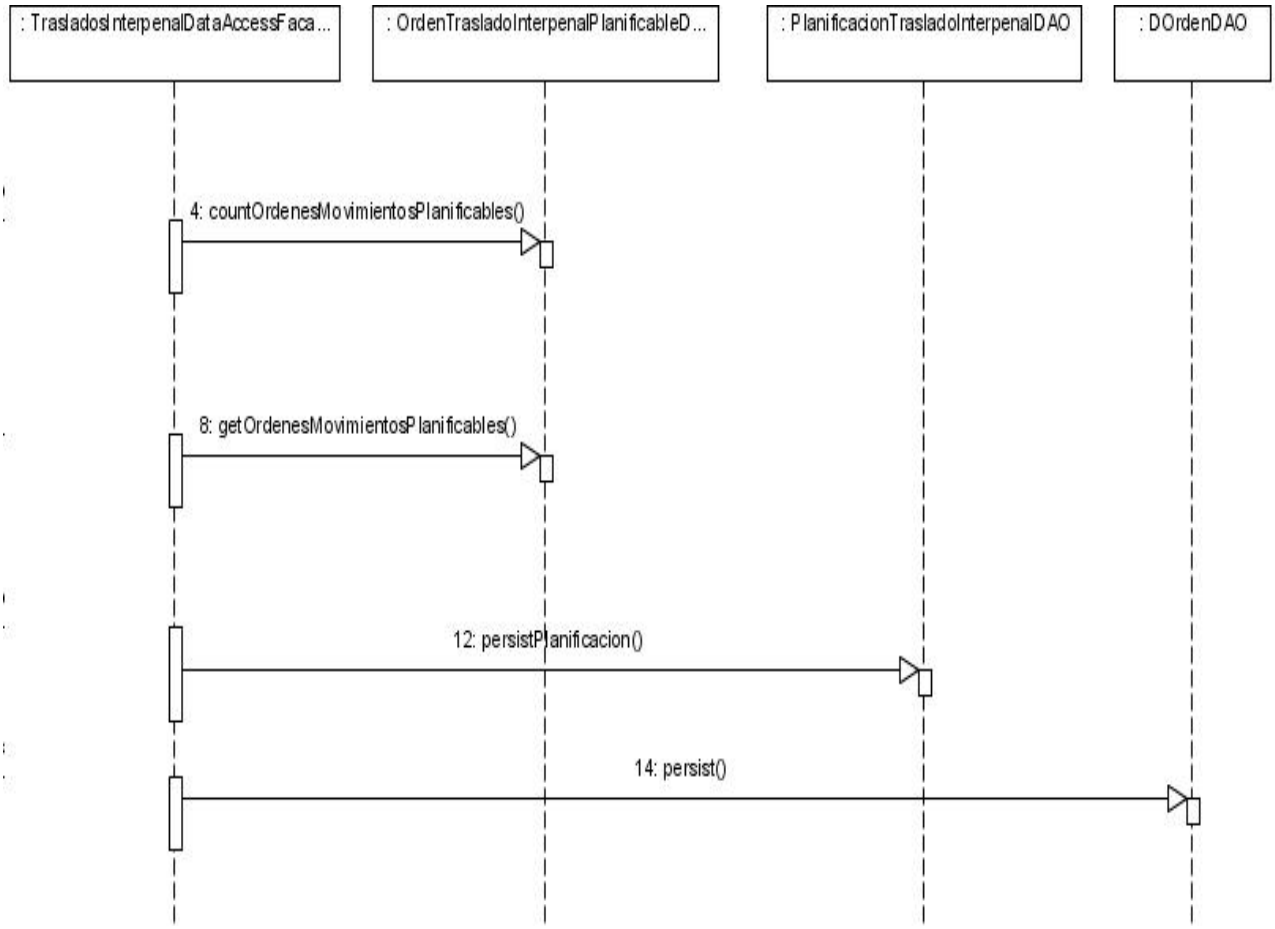

Figura 13: Diagrama de secuencia de la funcionalidad planificar traslado interpenal (parte2)

Descripción de la funcionalidad Registrar conclusión de Salida Transitoria

Diagrama de clases: Aquí se representan las clases que intervienen en la funcionalidad mencionada.

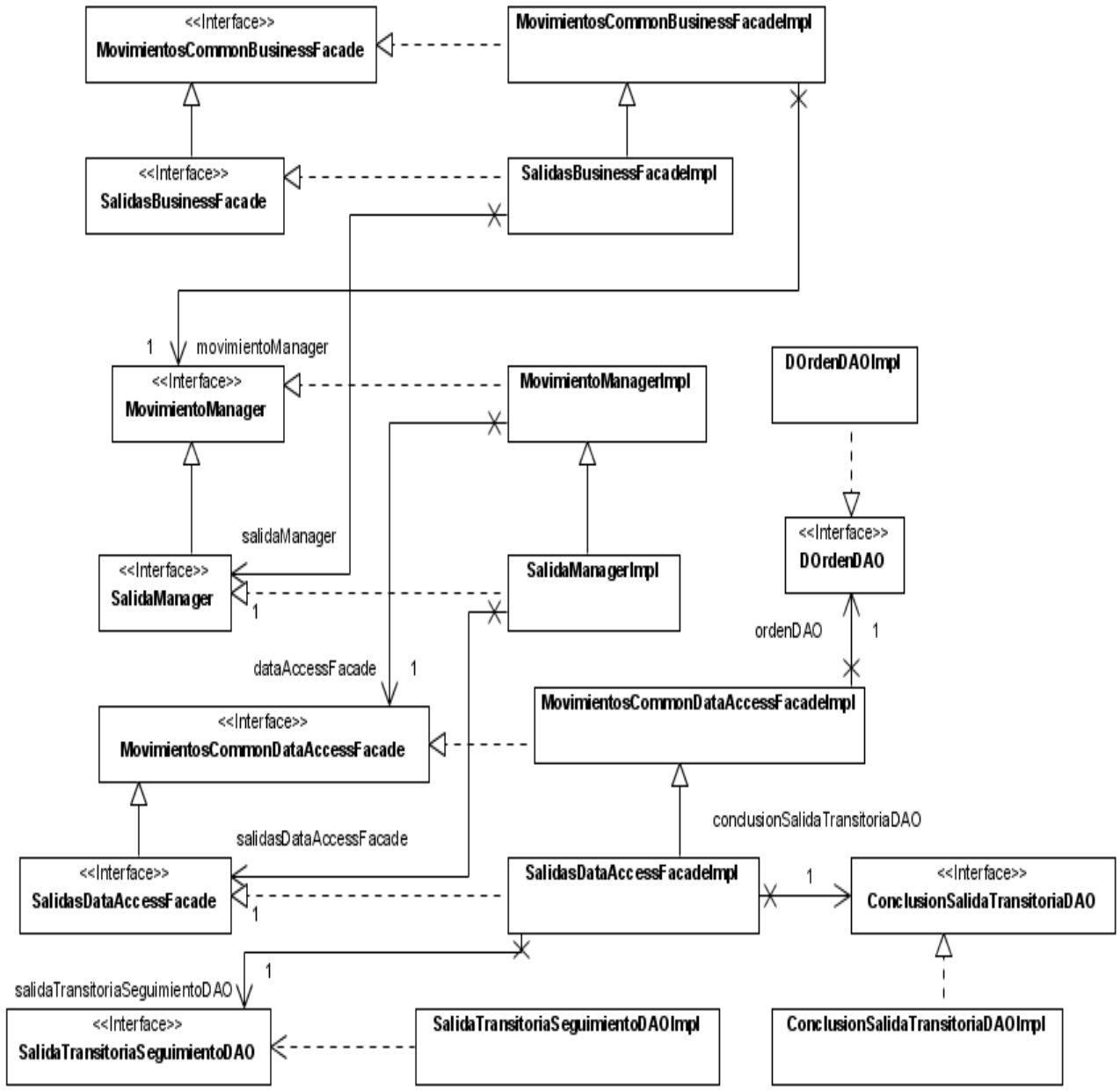

Figura 14: Diagrama de clases registrar conclusión de salida transitoria.

En el diagrama de clases representado en la figura 13 se encuentran las clases que intervienen en la funcionalidad planificar Traslado Interpenal. En este diagrama, se omitirán las clases que ya fueron explicadas anteriormente.

SalidasBusinessFacade: Interfaz de la capa de lógica de negocio del módulo Salidas Transitorias, que contiene las funcionalidades relacionadas con el modulo brindadas a la capa de presentación. Hereda de la interfaz "*MovimientosCommonBusinessFacade*".

SalidasBusinessFacadeImpl: Clase que implementa las funcionalidades definidas en la interfaz "*SalidasBusinessFacade*". Esta clase no implementa lógica de negocio.

SalidasDataAccessFacade: Interfaz que expone los métodos de acceso a datos específicos de Salidas Transitorias. Hereda de la interfaz "*MovimientosCommonDataAccessFacade*".

SalidasDataAccessFacadeImpl: Clase que implementa los métodos definidos en la interfaz "*SalidasDataAccessFacade*". Hereda de la clase "*TrasladosInterpenalDataAccessFacadeImpl*".

SalidaTransitoriaSeguimientoDAO: Interfaz que expone los métodos de acceso a datos de la entidad persistente "*SalidaTransitoriaSeguimiento*" la cual mapea con la tabla de la base de datos "SALIDA_TRANS_SEG".

SalidaTransitoriaSeguimientoDAOImpl: Clase que implementa los métodos definidos en la interfaz "*SalidaTransitoriaSeguimientoDAO*".

ConclusionSalidaTransitoriaDAO: Interfaz que expone los métodos de acceso a datos de la entidad persistente "*ConclusionSalidaTransitoria*" la cual mapea con la tabla de la base de datos "*CONCLUSION_STRAN*".

ConclusionSalidaTransitoriaDAOImpl: Clase que implementa los métodos definidos en la interfaz "*ConclusionSalidaTransitoriaDAO*".

Diagrama de secuencia: Se muestra la interacción entre las clases que intervienen en la funcionalidad.

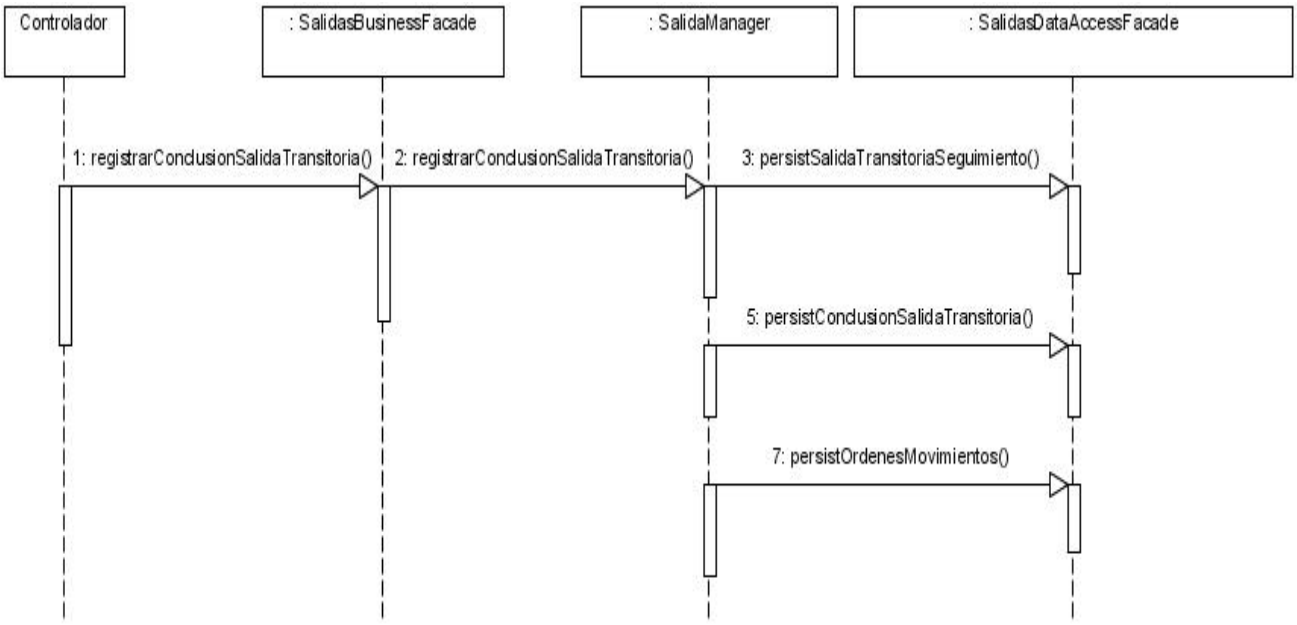

Figura 15: Diagrama de secuencia registrar conclusión salida transitoria (parte 1)

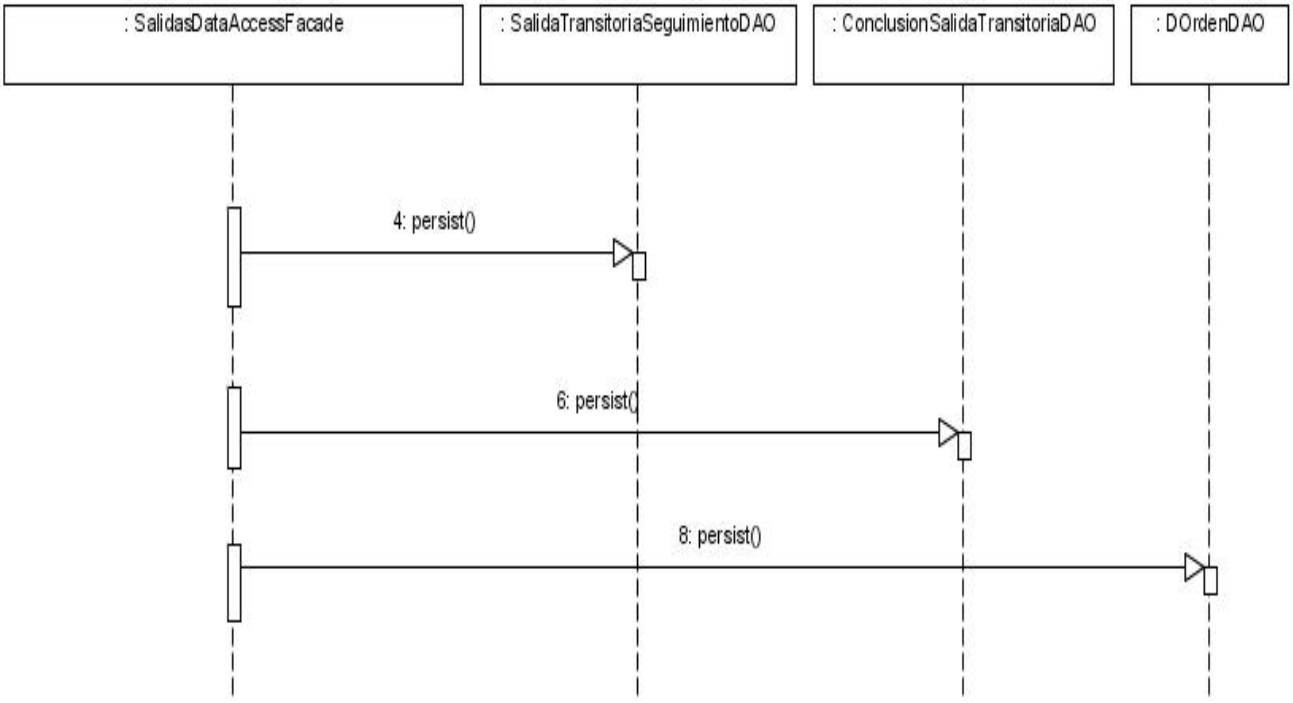

Figura 16: Diagrama de secuencia registrar conclusión salida transitoria (parte 2)

Modelo físico de datos

Aunque el modelo de datos lo realiza el diseñador de base de datos, como parte del flujo de trabajo se tiene que este debe ser revisado por el diseñador, el cual puede sugerirle cambios al diseñador de base de datos.

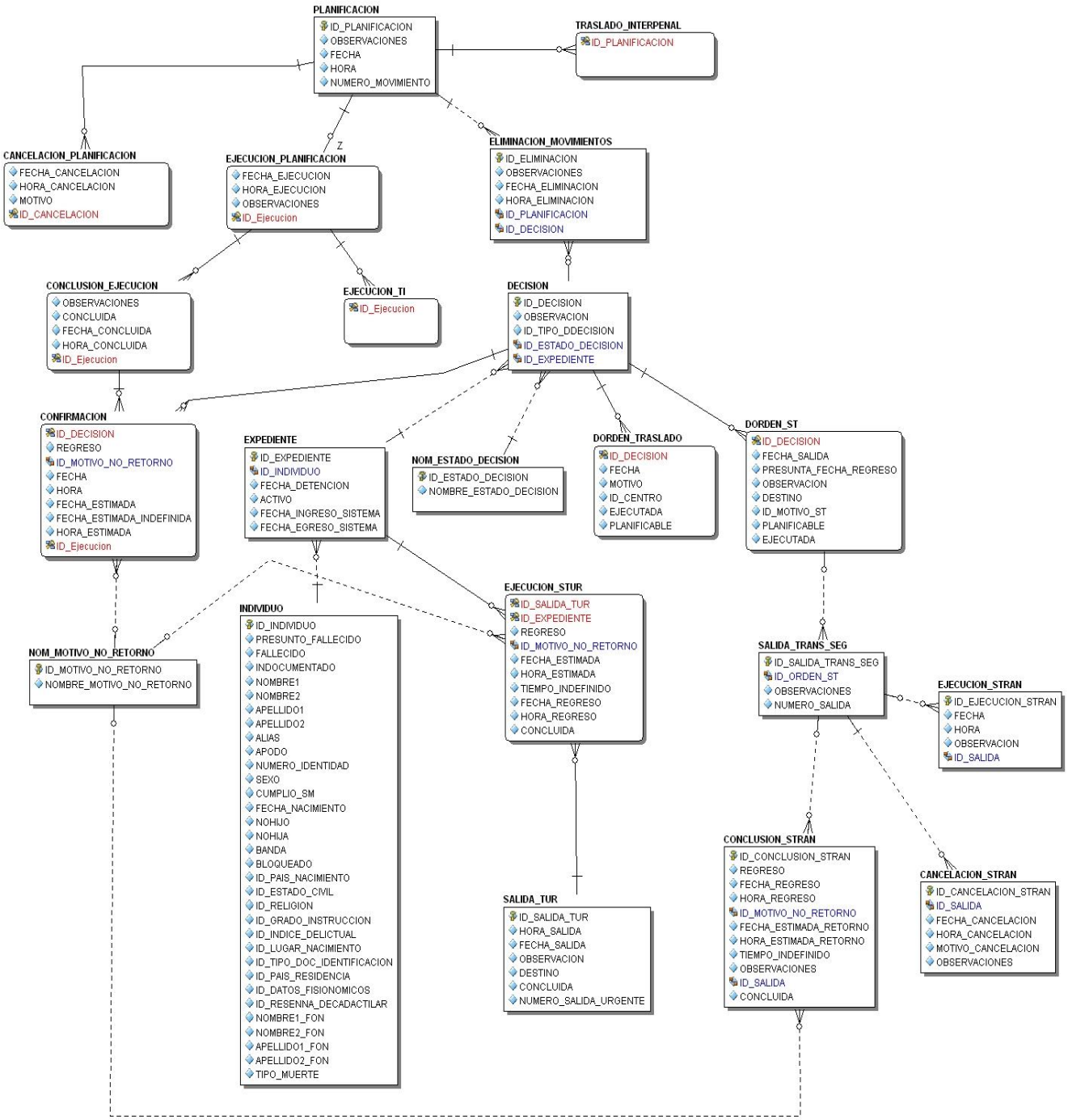

Figura 13: Modelo físico de datos

Se realizó el diseño de los módulos propuestos teniendo en cuenta los requisitos que fueron capturados en Venezuela durante el análisis y teniendo en cuenta patrones de diseño.

Se implementó la solución utilizando las herramientas y convenciones definidas en la arquitectura.

Se realizaron pruebas unitarias y de integración para comprobar que las capas estaban bien acopladas y que todas las funcionalidades trabajaban de manera correcta.

Se realizó la documentación de la solución.

Se ha cumplido el objetivo planteado contribuyendo de esta manera al mejoramiento del Sistema Penitenciario Venezolano. El sistema ya fue aprobado por el cliente y se encuentra funcionando en una prueba piloto en Venezuela.

Se recomienda explotar ampliamente el software para identificar situaciones o escenarios que no hayan sido tenidos en cuenta cuando se concibió la captura de requisitos en Venezuela.

1. **Walls, Craig.** *Spring in Action Second Edition.* s.l. : Manning Publications Co., 2008.

2. **Bauer, Christian y Gavin, King.** *Hibernate in Action.* s.l. : Manning Publications Co., 2005.

3. **Kaisler, Stephen H.** *Software Paradigms.* s.l. : John Wiley & Sons, Inc, 2005.

4. **Larman, Craig.** *UML y Patrones. Introducción al análisis y diseño orientados a objetos.* s.l. : Prentice Hall, 1999.

5. **González, Luis Alberto Pimentel y Rivero, Iósev Pérez.** *"ArBaWeb: ARQUITECTURA BASE SOBRE LA WEB".* Ciudad de La Habana : s.n., 2008.

6. [En línea] www.eclipse.org.

7. [En línea] http://www.eclipse.org/webtools/.

8. [En línea] http://subclipse.tigris.org/.

9. [En línea] http://www.hibernate.org/255.html.

10. [En línea] http://springide.org/blog/.

GLOSARIO

FSC: Funcionario de Seguridad y Custodia. Trabajador del área de Seguridad y Custodia.