Universidad de las Ciencias Informáticas

“Facultad 4”

Título: “Diseño e implementación de las capas de negocio y acceso a datos de los módulos Planificación y Ejecución de Visitas Familiares”

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

Autores: Ariosky Areces Gonzalez

Iskael Diaz Marquez

Tutora: Ing. Yanet Vega Miniet

Ciudad de la Habana, Mayo de 2008

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo y autorizamos al proyecto SIGEP de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

_____ _____

Ariosky Areces Gonzalez Iskael Diaz Marquez

_____

Ing. Yanet Vega Miniet

DATOS DE CONTACTO

Ing. Yanet Vega Miniet

- Profesor Adiestrado.
- Graduada en el 2007 de Ingeniero en Ciencias Informáticas de la Universidad de las Ciencias Informáticas (UCI), Título de Oro, promedio 5.06 puntos.

AGRADECIMIENTOS

Les agradecemos a todas las personas que de una forma u otra hicieron posible la realización de este trabajo, en especial a nuestra tutora Yanet por su gran ayuda y paciencia.

DEDICATORIA

A mami y papi por el cariño y el apoyo que me han brindado, por la confianza que siempre me han tenido, por enseñarme que en la vida los sueños se pueden hacer realidad cuando se buscan con voluntad y dedicación.

A mi hermanito para que continúe estudiando con esfuerzo y sacrificio.

A mi novia por apoyarme todo este tiempo y por brindarme su amor, eres mi vida.

A mis hermanos de la vida por estar siempre a mi lado haciéndome creer que siempre se puede un poquito más.

A todos aquellos que de una forma u otro me han apoyado y han creído en mí, para que a pesar de todo siga adelante.

Ariosky

A mi mamá y mi papá que me han apoyado, por la dedicación que me han brindado siempre, por su esfuerzo porque yo sea alguien en la vida, por educarme y ser mis mejores maestros, por confiar siempre en mí, por ser mi inspiración y por su amor.

A mis hermanas por estar siempre a mi lado, por darme su apoyo y cariño, por ayudarme en todo lo que he necesitado.

A todos los hermanos que no tengo pero que los considero como si fuesen mis hermanos, por haberme apoyado y por compartir conmigo incondicionalmente en las buenas y en las malas.

En fin a todas las personas que de una forma u otra hicieron posible que se realizara mi sueño.

Iskael

RESUMEN

Como parte del proceso de modernización del sistema penitenciario venezolano se ha concebido el Sistema de Gestión Penitenciaria (SIGEP) como una solución de software que viene a apoyar los procesos operativos, tácticos y estratégicos de la Dirección General de Custodia y Rehabilitación del Recluso (DGCRR). Dentro de este sistema se desarrollaron los módulos para la planificación y el control de la ejecución de las visitas familiares que se realizan en los establecimientos penitenciarios.

El presente trabajo aborda el diseño e implementación de dichos módulos, el cual se ajusta a los lineamientos arquitectónicos establecidos para el desarrollo del SIGEP, utilizando como *framework* base a Spring e Hibernate para el acceso a datos.

Se describe el flujo de trabajo de los roles que intervienen en el diseño e implementación de las capas de negocio y acceso a datos. Se describe la arquitectura sobre la cual se ha diseñado e implementado la solución así como las herramientas utilizadas.

PALABRAS CLAVE

SIGEP, DGCRR, Visitas Familiares, Planificación, Ejecución, Diseño, Implementación

TABLA DE CONTENIDOS

RESUMEN _____ *II*

PALABRAS CLAVE _____ *III*

INTRODUCCIÓN _____ *1*

DESARROLLO _____ *3*

Descripción detallada del problema _____ **3**

Solución de software propuesta _____ **5**

Técnicas, herramientas y métodos utilizados _____ **11**

Metodología de desarrollo _____ 11

Ambiente de desarrollo _____ 13

Modelo de diseño de la solución propuesta _____ **18**

CONCLUSIONES _____ *35*

RECOMENDACIONES _____ *36*

BIBLIOGRAFÍA _____ *37*

GLOSARIO _____ *38*

INTRODUCCIÓN

A partir de la aprobación de la Constitución de la República Bolivariana de Venezuela se comienzan a llevar a cabo una serie de reformas en pos de resolver los principales problemas de la población venezolana, surgiendo así el proyecto Humanización del Sistema Penitenciario, que integra atención a la salud de los individuos privados de libertad, asesoría especializada y un sistema informático para gestionar y automatizar los procesos penitenciarios: Sistema de Gestión Penitenciaria (SIGEP).

Dentro de las áreas a informatizar se encuentra la de Seguridad y Custodia, que se encarga de controlar las actividades de custodia a los reclusos y del orden interior de los establecimientos penitenciarios, estimando conveniente priorizar dentro de esta área una serie de procesos que irían resolviendo los problemas más apremiantes de la población penal. Entre estos procesos, se encuentran la planificación y la ejecución de visitas familiares a las cuales todo individuo privado de libertad tiene derecho, respetando lo reglamentado en las leyes vigentes, basado en un calendario general que permita elaborar una planificación previa de las visitas familiares, así como el control de la ejecución de las mismas. Sin embargo, un estudio realizado permitió identificar que esto no se cumple a cabalidad en el actual Sistema Penitenciario.

Para ello se han tomado una serie de medidas por parte de la DGCRR, las cuales permiten realizar el control y chequeo de las visitas familiares a los penales, llevando a cabo un seguimiento minucioso de las acciones que se realizan en estos desde el momento en que se planifica la visita familiar hasta que se ejecuta y concluye la misma.

En la definición del sistema se obtuvieron los requisitos del proceso Visitas Familiares que dentro del área de Seguridad y Custodia tendrá el objetivo de mantener el control de la planificación y ejecución de las visitas a los individuos privados de libertad. Una vez determinados estos requisitos de software surge la necesidad de obtener un diseño de la solución que permita la posterior implementación, por lo que se define como el problema a resolver:

¿Cómo desarrollar las funcionalidades necesarias para diseñar e implementar las capas de lógica de negocio y acceso a datos de la planificación y ejecución de visitas familiares en los establecimientos penitenciarios, cumpliendo los requisitos establecidos y según la arquitectura definida en el proyecto SIGEP?

Basado en lo anterior, se ha definido como Objeto de Estudio el proceso de desarrollo de software de los módulos Planificación y Ejecución de Visitas Familiares y como campo de acción el diseño e implementación de las capas de negocio y acceso a datos de dichos módulos.

Para dar solución al problema se propone como objetivo:

Diseñar e implementar las capas de lógica de negocio y acceso a datos de los módulos del SIGEP que automaticen los procesos de Planificación y Ejecución de Visitas Familiares del Sistema Penitenciario venezolano, según los requisitos y la arquitectura establecidos en el proyecto.

Para dar cumplimiento a este objetivo es necesario realizar una serie de tareas que se relacionan a continuación:

1. Realizar el diseño de la solución que de respuesta a los requisitos especificados con la utilización de patrones de diseño y cumpliendo con la tecnología y la arquitectura definida para el SIGEP.
2. Implementar la solución diseñada.
3. Realizar pruebas unitarias y de integración.

La solución propuesta permitirá a los establecimientos penitenciarios (Internados Judiciales, Centros Penitenciarios, Centros de Tratamiento Comunitario) y otras sedes (Coordinaciones Regionales y Unidades Técnicas de Apoyo al Sistema Penitenciario), recopilar y controlar la información operativa que se genera con las visitas familiares en estos centros. Esta solución manejará datos sobre la población penal y auditará los procesos legales para garantizar un cumplimiento justo de las visitas familiares. La solución estará diseñada para ofrecer al personal servicios en línea que faciliten la ejecución de sus tareas, permitirá a la DGCRR poseer una visión global del sistema y por tanto tomar decisiones en base a una información íntegra y precisa, mejorando el desempeño operacional de la misma.

Para el desarrollo del SIGEP se ha considerado el Proceso Unificado de Desarrollo (RUP) como metodología de referencia. El marco de desarrollo describe los artefactos de RUP elegidos para la construcción del sistema por cada una de las disciplinas y durante las fases que prevé dicha metodología. (ARIAS 2006).

“El SIGEP se desarrollará bajo la tecnología *Java Enterprise Edition* utilizando como *framework* arquitectónico base a *Spring Framework*. Como gestor de base de datos se utilizará Oracle 10G por petición del cliente del software. Como *framework* de persistencia se utilizará *Hibernate*. En la capa de presentación se utilizará como tecnología *Spring MVC* bajo las adaptaciones realizadas al *framework* para el proyecto”. (PIMIENTEL 2006)

DESARROLLO

Descripción detallada del problema

El Reglamento de Ley de Régimen Penitenciario establece el derecho de los individuos a recibir visitas familiares. (BETANCOURT 2007) Para garantizar este derecho el área de Seguridad y Custodia debe planificar y ejecutar las visitas familiares según se describe a continuación.

Planificación de Visitas Familiares

La misión del proceso Planificación de Visitas Familiares es garantizar el derecho que tiene todo individuo sometido a una medida privativa de libertad a recibir visitas de acuerdo a lo establecido en los reglamentos y leyes vigentes en esta materia. El alcance del mismo es elaborar el calendario general para mantener el control de las visitas familiares a realizar y diseñar el plan de realización de las mismas, el cual consiste en planificar los turnos en los cuales recibirán la visita los internos recluidos en el centro, a partir de la clasificación de los mismos y lo que establece el reglamento.

La planificación de visitas familiares está formada por dos importantes actividades a realizar:

Elaborar Calendario General de Visitas:

El Calendario General de Visitas es elaborado por un Funcionario de la Dirección del Establecimiento Penitenciario. El objetivo del mismo es organizar el proceso de visitas en un marco de tiempo previamente establecido. El calendario tiene un período de vigencia y dentro de este período, se encuentran los días específicos en que corresponde la visita. A su vez, incluye cada qué tiempo se repetirán estos días y el horario destinado a la visita dentro de cada día especificado. El calendario lleva la aprobación del Director del Establecimiento Penitenciario y es publicado por el Funcionario de la Dirección encargado de elaborarlo.

Diseñar Plan de Realización de Visitas:

Para la elaboración del Plan de Realización de visitas se toma como base el Calendario General, teniendo en cuenta el calendario seleccionado se asignan los turnos de visita registrados en el mismo a los grupos de visitas familiar, los cuales están integrados por individuos penados de libertad que posean la misma clasificación del calendario en cuestión.

Ejecución de Visitas Familiares.

La misión del proceso Ejecución de Visitas Familiares es garantizar que se cumplan todas las medidas y disposiciones de seguridad reglamentadas en las normas y leyes vigentes. El mismo debe garantizar el registro e identificación de visitantes, el control de visitantes en el Establecimiento Penitenciario, preparar al individuo antes de la visita y el control de los visitantes al terminar la misma.

El proceso se inicia una vez que se presenta un visitante al establecimiento penitenciario. El Funcionario de Seguridad y Custodia (FSC), una vez que ha verificado que el visitante no posee objetos prohibidos en el centro, procede a registrar en el libro de novedades los datos del visitante, el nombre del interno al cual visitará y la fecha y hora de entrada. Una vez que la visita ha concluido, el Funcionario de Seguridad y Custodia, registra en el mismo libro, cualquier observación importante con respecto al desarrollo de la visita, así como la fecha y hora de salida.

En caso de que el visitante haya mostrado una mala conducta durante su visita al centro, el Funcionario de Seguridad y Custodia podrá retirarle el permiso para realizar visitas nuevamente.

La ejecución de visitas familiares está formada por cuatro tareas de suma importancia para el correcto desarrollo de la misma, estas son:

Verificar que el nombre del visitante aparezca en el Listado de Visitantes Autorizados:

El Custodio Asistencial busca y verifica que el nombre del visitante esté en el Listado de Visitantes Autorizados, el cual debe haber sido previamente enviado por Tratamiento. En el listado que tiene el Custodio Asistencial, está el nombre del individuo y una relación de las personas que él autorizó para que lo visitara.

Registro e identificación de visitantes:

En este proceso se verifica la identidad del visitante a través de su documento de identidad y de su huella dactilar. Además, se le toma una fotografía y se registran sus datos generales. Si es un menor de edad, se verifica la correspondencia entre su representante legal y la partida de nacimiento; en caso de existir cualquier problema en alguna de las actividades anteriores, el Establecimiento Penitenciario toma las medidas correspondientes.

Control de visitantes al terminar la visita:

Al terminar el proceso de visita los visitantes recogen las pertenencias que habían sido entregadas al iniciarse la misma, son identificados nuevamente por las autoridades correspondientes y se les autoriza la salida del Establecimiento Penitenciario.

Registrar datos del visitante no autorizado.

El FSC registra los datos personales (nombre, número de cédula o pasaporte, teléfono, nombre del interno) del visitante no autorizado y lo relaciona con el nombre del individuo al cual quiere visitar. Se prohíbe la entrada del visitante al Establecimiento Penitenciario.

Actualmente en Venezuela cada centro penitenciario ejecuta los subprocesos de Planificación y Ejecución de Visitas Familiares de forma diferente, no se hace una planificación que responda a los verdaderos intereses de los individuos sometidos a una medida privativa o restrictiva de libertad en un centro penitenciario. Los datos de los visitantes que se registran a la entrada no son los mismos en todos los centros penitenciarios, en muchas ocasiones no se verifica la identidad de los visitantes que vienen a realizar la visita familiar, esto imposibilita mantener un control adecuado y seguimiento de los visitantes que reiteradamente realizan la visita al centro penitenciario. La información que se maneja sobre las visitas familiares en los centros de reclusión se lleva de manera manual o no se recoge en su totalidad, la información de estas visitas se archiva en montones de papel, o simplemente dejan de existir una vez concluida la visita. Conllevando esto a una serie de problemas y dificultades que hacen necesario la informatización del proceso de visitas familiares los cuales son (ARIAS 2006) :

1. No se garantiza el derecho de los individuos a recibir visitas familiares según su clasificación y lo establecido en la legislación.
2. No se garantizan las medidas de seguridad establecidas durante el desarrollo de la visita familiar, por ejemplo, que el individuo no reciba visitas de personas con una influencia negativa en su personalidad y conducta.
3. La información sobre las visitas familiares que maneja cada centro penitenciario no es la misma y se recoge de manera manual, por lo que se hace difícil consultarla y emitir partes respecto a ellas, impidiendo la toma de decisiones en función de la realidad objetiva de cada centro penitenciario.

Solución de software propuesta

Según se ha explicado anteriormente, en adición a las medidas tomadas desde la DGCRR para resolver la crítica situación del Sistema Penitenciario, se ha decidido desarrollar un sistema informático que apoye los principales procesos a nivel de centro penitenciario. Respecto al proceso de Planificación y Ejecución de Visitas Familiares el sistema debe:

1. Contribuir con la transformación y modernización del Sistema Penitenciario Venezolano, que propicie el rescate progresivo de la disciplina en los establecimientos penitenciarios y la

adecuada organización y funcionamiento de los sistemas de trabajo durante la planificación y ejecución de las visitas familiares.

2. Garantizar el derecho de cada individuo a recibir las visitas en los períodos establecidos por la legislación, según su clasificación.
3. Contribuir a la prevención y enfrentamiento de hechos ilícitos, como la presencia de objetos prohibidos dentro del centro penitenciario y el tráfico de armas, drogas, entre otros.
4. Generar información para los niveles de mando superiores del desarrollo de las visitas familiares.

Para ello es necesario que el sistema cumpla los siguientes requisitos funcionales:

Tabla 1. Resumen de la funcionalidad 1.

Funcionalidad -1	Gestionar Calendarios de Visita Familiar
Actor	FSC
Breve Descripción	Esta funcionalidad muestra todos los calendarios de visita familiar que estén registrados en el sistema y brinda la posibilidad de adicionar, eliminar o modificar un calendario de los que ya se encuentran registrados.
Pre-Condiciones	El funcionario se ha autenticado ante el sistema.
Post-Condiciones	Se ha consultado, adicionado, modificado o eliminado un Calendario de Visita Familiar.

Tabla 2. Resumen de la funcionalidad 2.

Funcionalidad-2	Gestionar Turnos de Visita Familiar
Actor	FSC
Breve Descripción	Esta funcionalidad permite mostrar todos los turnos relacionados con un calendario de visita, así como modificar, eliminar y adicionar un nuevo turno al calendario seleccionado.
Pre-Condiciones	El funcionario se ha autenticado ante el sistema. El funcionario debe haber seleccionado un Calendario de Visita Familiar
Post-Condiciones	Se ha consultado, adicionado, modificado o eliminado un Turno de Visita Familiar.

Tabla 3. Resumen de la funcionalidad 3.

Funcionalidad-3	Gestionar Grupos de Visita Familiar
Actor	FSC
Breve Descripción	Esta funcionalidad permite mostrar todos los grupos de visita que han sido registrados en el Sistema para cada una de las clasificaciones existentes, así como modificar, eliminar y/o adicionar un nuevo grupo de visita familiar.
Pre-Condiciones	El FSC se ha autenticado ante el sistema.
Post-Condiciones	Se ha consultado, adicionado, modificado o eliminado un Grupo de Visita Familiar.

Tabla 4. Resumen de la funcionalidad 4.

Funcionalidad-4	Planificar Visitas Familiares
Actor	Funcionario de Seguridad y Custodia
Breve Descripción	Esta funcionalidad muestra los datos principales del Calendario seleccionado, además brinda la posibilidad de asociar los Grupos de Visita Familiar que pertenezcan a la misma clasificación del Calendario seleccionado con los Turnos de Visita Familiar definidos para el mismo, y viceversa.
Pre-Condiciones	El FSC se ha autenticado ante el sistema. El funcionario de Seguridad y Custodia debe haber seleccionado un Calendario de Visita Familiar.
Post-Condiciones	Se han planificado las visitas familiares para el calendario seleccionado.

Tabla 5. Resumen de la funcionalidad 5.

Funcionalidad-5	Histórico de Visitas Familiares Realizadas
Actor	FSC
Breve Descripción	Esta funcionalidad muestra un histórico de todas las visitas familiares realizadas en el penal, y brinda la posibilidad de poder consultar los detalles de una visita familiar y/o eliminar una de las visitas presentes en el histórico.
Pre-Condiciones	El FSC se ha autenticado ante el sistema.
Post-Condiciones	Se ha consultado, adicionado, modificado o eliminado un Grupo de Visita Familiar.

Tabla 6. Resumen de la funcionalidad 6.

Funcionalidad-6	Consultar Visitantes no Autorizados
Actor	FSC
Breve Descripción	Esta funcionalidad le permite al FSC, consultar todos los visitantes que han sido declarados como no autorizados a entrar al penal, así como la posibilidad de devolverles la autorización para que puedan efectuar visitas.
Pre-Condiciones	El FSC se ha autenticado ante el sistema.
Post-Condiciones	Se actualiza el listado de Visitantes no Autorizados a entrar en el penal.

Tabla 7. Resumen de la funcionalidad 7.

Funcionalidad-7	Registrar entrada de Visitante
Actor	FSC
Breve Descripción	Esta funcionalidad le permite al FSC registrar la entrada de un visitante a la Visita Familiar recogiendo todos sus datos en caso de que la visita esté permitida, o registrar la denegación de la visita registrando los detalles de la misma, los cuales incluyen los datos del visitante.
Pre-Condiciones	El FSC se ha autenticado ante el sistema.
Post-Condiciones	Queda registrada la entrada del visitante a penal.

Tabla 8. Resumen de la funcionalidad 8.

Funcionalidad-8	Registrar salida del Visitante
Actor	FSC
Breve Descripción	Esta funcionalidad le permite al FSC registrar la salida de un visitante que se encontraba dentro del penal realizando una Visita Familiar. Se registran los datos de la salida y se brinda la posibilidad de retirarle el permiso a realizar visitas nuevamente si ha mostrado una conducta incorrecta.
Pre-Condiciones	El FSC se ha autenticado ante el sistema. El FSC debe haber seleccionado uno de los visitantes que esté en el listado de las visitas en ejecución.
Post-Condiciones	Queda registrada la salida del visitante del penal.

Técnicas, herramientas y métodos utilizados

Metodología de desarrollo

El Proyecto Técnico del SIGEP (ARIAS 2006) establece como metodología de referencia RUP. Esta metodología define “un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas de software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyecto”. (JACOBSON, IDFSDF 2004)

RUP es un proceso:

- Iterativo e incremental, lo cual permite dividir el proyecto en pequeños subproyectos para desarrollarlo en distintas etapas e iteraciones que resultan en un incremento del producto. (JACOBSON, IDFSDF 2004)
- Dirigido por casos de uso, el cual es uno de los métodos más utilizados y efectivos para reflejar los requisitos. Estos no sólo sirven para especificar los requisitos, ellos son los encargados de guiar el ciclo de vida del proyecto. (JACOBSON, IDFSDF 2004)
- Centrado en la arquitectura, la arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente. RUP se desarrolla mediante iteraciones, comenzando por los CU relevantes desde el punto de vista de la arquitectura. (JACOBSON, IDFSDF 2004)
- RUP propone además flujos de trabajo en los que se definen las secuencias de actividades, quienes las deben desarrollar y los artefactos a generar. En el marco del proyecto SIGEP se generarán los siguientes artefactos en las disciplinas de Diseño e Implementación, en las cuales se centra este trabajo: modelo de datos, diagramas de clases por capas arquitectónicas, diagramas de secuencia y código fuente.

En el proyecto SIGEP se ajustaron los flujos de trabajo propuestos por RUP a las necesidades del proyecto. Específicamente se definieron las siguientes actividades para los roles Diseñador, Programador de Lógica de Negocio y Programador de Acceso a Datos, los cuales cubren el alcance del presente trabajo.

Diseñador (BENAVIDES 2006)

1. A partir del Modelo Conceptual que se refleja en la Especificación de los requisitos funcionales y el diseño relacional de la Base de Datos (BD) se establecen las entidades de diseño, los nombres de estas, así como los atributos de cada una de ellas, relacionados con campos en tablas en la BD.
2. Diseñar las funcionalidades de la fachada del módulo y de los managers. Identificar los servicios que la capa de negocio debe prestar para dar respuesta al caso de uso.
3. Diseñar las interfaces de acceso a datos. Definir las interfaces que manejarán el ciclo de vida de las entidades persistentes.
4. Detallar el flujo de interacción entre las clases en cada una de las capas lógicas de la aplicación, principalmente la capa de negocio y acceso a datos.
5. Modelar y mostrar colaboración entre paquetes y subsistemas del diseño.
6. Generar la documentación final de la disciplina de Diseño.

Programador de Lógica de Negocio (DELGADO 2006)

1. Crear Managers Falsos para el trabajo en paralelo de la capa superior y su utilización en la etapa de pruebas.
2. Configurar los Manager en los XML correspondientes para su utilización.
3. Implementar y configurar la Fachada del Módulo para su funcionamiento.
4. Implementar los Managers reales que permitirán dar funcionalidad a la aplicación.
5. Configurar los Managers para su uso en la aplicación.
6. Re implementación de las entidades de dominio.

Programador de Acceso a Datos

1. Compatibilizar Objetos con la Base de Datos.
    a. Generar ficheros de mapeo.
    b. Chequear tipos de datos.
    c. Chequear relaciones (objetos y tablas).
    d. Validar los ficheros de mapeo (hbm.xml).
2. Implementar las interfaces de los DAO (Data Access Object).
3. Probar la implementación de los DAO.
    a. Implementar y ejecutar las pruebas de unidad.
    b. Declarar los beans de los DAOs en el dataaccess-context.xml.

Ambiente de desarrollo

El ambiente de desarrollo es el conjunto de herramientas, *frameworks* y tecnologías, versiones a usar y su integración, que intervienen en un proceso de desarrollo de software.

A continuación se describe el ambiente de desarrollo del SIGEP, exponiendo sus características y ventajas para poder tener un mayor conocimiento de sus prestaciones y valorar su utilidad:

Java. Lenguaje de programación.

Java es una tecnología orientada al desarrollo de software con la cual se puede realizar cualquier tipo de programa. Las características principales que ofrece Java son:

- Orientado a objetos: Java fue diseñado como un lenguaje orientado a objetos desde el principio. Los objetos agrupan en estructuras encapsuladas tanto sus datos como los métodos (o funciones) que manipulan esos datos. (*CARACTERISTICAS DE JAVA*)
- Distribuido: Java proporciona una colección de clases para su uso en aplicaciones de red, que permiten abrir *sockets* y establecer y aceptar conexiones con servidores o clientes remotos.(*CARACTERISTICAS DE JAVA*)
- Interpretado y compilado a la vez: Java es compilado, en la medida en que su código fuente se transforma en una especie de código máquina. Por otra parte, es interpretado, ya que se puede ejecutar directamente sobre cualquier máquina a la cual se hayan portado el intérprete y el sistema de ejecución en tiempo real (run-time).(*CARACTERISTICAS DE JAVA*)
- Robusto: Java fue diseñado para crear software altamente fiable. Para ello proporciona numerosas comprobaciones en compilación y en tiempo de ejecución. (*CARACTERISTICAS DE JAVA*)
- Seguro: Dada la naturaleza distribuida de Java, la seguridad es necesidad de vital importancia por lo que presenta barreras de seguridad en el lenguaje y en el sistema de ejecución en tiempo real. (*CARACTERISTICAS DE JAVA*)
- Independiente de la arquitectura de hardware: Java está diseñado para soportar aplicaciones que serán ejecutadas en los más variados entornos de red, desde Unix a Windows NT, pasando por Mac y estaciones de trabajo, sobre arquitecturas distintas y con sistemas operativos diversos. (*CARACTERISTICAS DE JAVA*)

J2EE

J2EE define un estándar para el desarrollo de aplicaciones empresariales multicapa, fue diseñada por Sun Microsystems. J2EE simplifica las aplicaciones empresariales basándolas en componentes modulares y estandarizados, proveyendo un conjunto completo de servicios a estos componentes, y manejando muchas de las funciones de la aplicación de forma automática, sin necesidad de una programación compleja.

Eclipse. Entorno de desarrollo integrado (IDE)

Eclipse es un entorno de desarrollo integrado de código abierto independiente de una plataforma:

- Está desarrollado en Java y su principal uso es como IDE para Java. El soporte para desarrollo en Java es provisto por un componente enchufado o *plugin*, pero además están disponibles

plugins para otros lenguajes, como C/C++, Cobol, C#. (GALLARDO 2003) En principio permite ejecutar un programa sobre cualquier plataforma. Es una plataforma extensible de código abierto para desarrollar herramientas.

- La arquitectura de *plugins* de Eclipse permite, además de integrar diversos lenguajes sobre un mismo IDE, introducir otras aplicaciones accesorias que pueden resultar útiles durante el proceso de desarrollo como: herramientas UML, editores visuales de interfaces, ayuda en línea para librerías, etc.

Spring Framework

Es un *framework* de código abierto para el desarrollo de aplicaciones para la plataforma Java. La primera versión fue escrita por Rod Johnson, quien lo lanzó primero con la publicación de su libro *Expert One-on-One Java EE Design and Development.*

Es el más popular y el más ambicioso de todos los *framework* de peso ligero. Es el único *framework* que interviene en todas las capas arquitectónicas de una aplicación JEE. Además está diseñado para facilitar una flexibilidad arquitectónica. (JOHNSON 2005)

Los principales valores de Spring, según Rod Johnson, se pueden resumir en: No es agresivo, provee un modelo consistente de programación, ayuda a promover la reusabilidad de código, facilita el diseño Orientado a Objetos (OO) en aplicaciones JEE, permite la extracción de valores de configuración desde el código java a archivos XML o archivos de propiedades, está diseñado a fin de que las aplicaciones lo usen para que las pruebas sean lo más fácil posible, S*pring* hace de soluciones existentes un uso más fácil, dentro de una arquitectura consistente.

Hibernate

Hibernate es un *framework* objeto/relacional y un generador de sentencias sql. Te permite diseñar objetos persistentes que podrán incluir polimorfismo, relaciones, colecciones, y un gran número de tipos de datos. Hibernate se integra en cualquier tipo de aplicación justo por encima del contenedor de datos. Hibernate genera las sentencias SQL y libera al desarrollador del manejo manual de los datos que resultan de la ejecución de dichas sentencias, manteniendo la portabilidad entre todas las bases de datos con un ligero incremento en el tiempo de ejecución, ofrece también un lenguaje de consulta de datos llamado **HQL** (*Hibernate Query Language*), al mismo tiempo que una API para construir las consultas programáticamente (conocida como *"Criteria"*). (*Relational Persistence for Java*)

JUnit

JUnit es un conjunto de clases (*framework*) que permite realizar la ejecución de clases Java de manera controlada, para poder evaluar si el funcionamiento de cada uno de los métodos de la clase se comporta como se espera. Es utilizado como método para las pruebas unitarias.

Visual Paradigm

Visual Paradigm es una herramienta CASE (Computer Aided Software Engineering) profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Este software ayuda a una más rápida construcción de aplicaciones de calidad y a un menor coste. Permite construir todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Es muy útil en la ingeniería inversa, generación de código, importación desde Rational Rose, exportación/importación XMI(XML de Intercambio de Metadatos), generador de informes, editor de figuras, integración con MS Visio, plug-in, integración con el IDE Eclipse y otros.

ER/Studio

Es una herramienta de modelado de datos fácil de usar y multinivel para el diseño y construcción de bases de datos a nivel físico y lógico. Direcciona las necesidades diarias de los administradores de bases de datos, desarrolladores y arquitectos de datos que construyen y mantienen aplicaciones de bases de datos grandes y complejas.

Oracle

Oracle es un sistema de gestión de base de datos relacional (o RDBMS por el acrónimo en inglés Relational Data Base Management System), fabricado por Oracle Corporation. Se considera uno de los sistemas de bases de datos más completos. Es un sistema gestor de base de datos robusto, tiene muchas características que nos garantizan la seguridad e integridad de los datos; que las transacciones se ejecuten de forma correcta, sin causar inconsistencias; ayuda a administrar y almacenar grandes volúmenes de datos; estabilidad, escalabilidad y es multiplataforma.

Arquitectura de software

Existen muchas definiciones sobre arquitectura de software:

“La Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución. (IEEE 1995)

“La arquitectura de software, tiene que ver con el diseño y la implementación de estructuras de software de alto nivel. Es el resultado de ensamblar un cierto número de elementos arquitectónicos de forma adecuada para satisfacer la mayor funcionalidad y requerimientos de desempeño de un sistema, así como requisitos no funcionales, como la confiabilidad, escalabilidad, portabilidad, y disponibilidad” (KRUCHTEN)

“Como un edificio, un sistema de software es una única entidad, pero al arquitecto del software y a los desarrolladores les resulta útil presentar el sistema desde diferentes perspectivas para comprender mejor el diseño. Estas perspectivas son vistas del modelo del sistema. Todas juntas representan la arquitectura” (JACOBSON, IVAR, GRADY BOOCH, JAMES RUMBAUGH, 1999)

En el presente trabajo se ha utilizado la arquitectura definida en el Documento de Arquitectura de Software para el proyecto SIGEP (PIMIENTEL 2006) : El cual presenta una arquitectura base para: orientar el diseño de las capas lógicas a través de una propuesta de diseño base; organizar la forma de codificar según las propuestas de convenciones o estándares de códigos y recursos; brindar una estructura física para soportar el código, creando así un esqueleto base; y proponer mecanismos de colaboración entre los componentes integrados en ella.

Esta arquitectura está regida por una arquitectura de tres capas lógicas fundamentales bien definidas: Capa de acceso a datos, Capa de servicios de negocio y Capa de presentación. Otro concepto tratado es: “objetos persistentes del dominio (entidades)”, este conjunto de objetos puede llegar a considerarse otra capa lógica que puede presentar este tipo de arquitectura. No es una arquitectura distribuida aunque pudiera adaptarse sin mucho esfuerzo a un sistema que lo requiera. Cada capa puede ser modificada tanto como sea posible sin provocar un impacto en las demás capas. Una capa no es consciente de lo que le ocurre a la capa superior, su dependencia es puramente con la capa inmediata inferior. Esta dependencia entre capas es normalmente entre interfaces, asegurando que el acople sea el más bajo posible. (PIMIENTEL 2006)

Con el objetivo de organizar la aplicación se definieron como unidades de organización subsistemas y módulos. Un módulo encapsula un conjunto de funciones que debe realizar el sistema, las cuales son agrupadas por tener características muy similares y se definen en la etapa de diseño. Un subsistema se refiere a un conjunto de módulos que por razones de similitud o de perseguir objetivos comunes, son agrupados. (PIMIENTEL 2006)

Se crean mecanismos de colaboración los cuales se dividen en: funcionalidad común y funcionalidad especializada, como estrategias para establecer la forma de comunicación entre los componentes del software (módulos y subsistemas) (PIMIENTEL 2006)

Con el fin de lograr un lenguaje común y uniforme en toda la aplicación se especifican convenciones de nombres y estándares de código para los distintos recursos de la aplicación. (PIMIENTEL 2006)

Define un orden de carga de los módulos de la aplicación respetado la dependencia entre dichos módulos. (PIMIENTEL 2006)

Modelo de diseño de la solución propuesta

Los artefactos generados en el flujo de trabajo Requisitos: Descripción de funcionalidades, Prototipo de Interfaz de Usuario, Modelo de Dominio y Diccionario de Datos constituyen las entradas, en SIGEP, al flujo de trabajo Diseño.

"Diseño es el proceso de aplicar distintas técnicas y principios con el propósito de definir un dispositivo, proceso, o sistema, con los suficientes detalles como para permitir su realización física" (E.S.TAYLOR 1959) El propósito del Diseño es definir los objetos software y cómo colaboran entre ellos para satisfacer los requisitos. (LARMAN 2002)

Durante el diseño de software se presentan problemas que se repiten en situaciones particulares, para resolverlos se proponen soluciones que han demostrado ser exitosas resolviendo problemas similares y que son reusables en diferentes circunstancias: los patrones de diseño. En la solución propuesta se han aplicado patrones:

“Patrón es una pareja de problema/solución con un nombre y que es aplicable a otros contextos, con una sugerencia sobre la manera de usarlo en situaciones nuevas”(LARMAN 1999)

En el desarrollo del presente trabajo fueron utilizados los patrones generales para la asignación de responsabilidades (GRASP, por sus siglas en inglés):

“Los patrones ***GRASP*** describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Constituyen un apoyo para la enseñanza que ayuda a entender el diseño de objeto esencial y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable.” (SAAVEDRA)

Bajo acoplamiento: El Bajo Acoplamiento es un patrón evaluativo que el diseñador aplica al juzgar sus decisiones de diseño. Estimula asignar una responsabilidad de modo que su colocación no incremente el acoplamiento tanto que produzca los resultados negativos propios de un alto acoplamiento.

Alta Cohesión: Asignar una responsabilidad de modo que la cohesión siga siendo alta. Representa una Clase con responsabilidades moderadas en un área funcional, colaborando con otras para

concretar tareas. Diseño más claro y comprensible. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme.

Experto: Es un principio básico que suele utilizarse en el diseño orientado a objetos. Con él no se pretende designar una idea oscura ni extraña; expresa simplemente la "intuición" de que los objetos hacen cosas relacionadas con la información que poseen. Ofrece una analogía con el mundo real.

Creador: El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento, se debe buscar una clase de objeto que agregue, contenga y realice otras operaciones sobre este tipo de instancias.

Con la utilización de estos patrones y el análisis de las funcionalidades expuestas con anterioridad se generan las clases de los módulos de planificación y ejecución de visitas familiares respectivamente, estas clases son las representadas a continuación en los diagramas de clases de dominio, diagramas de clases del negocio y diagramas de clases de acceso a datos.

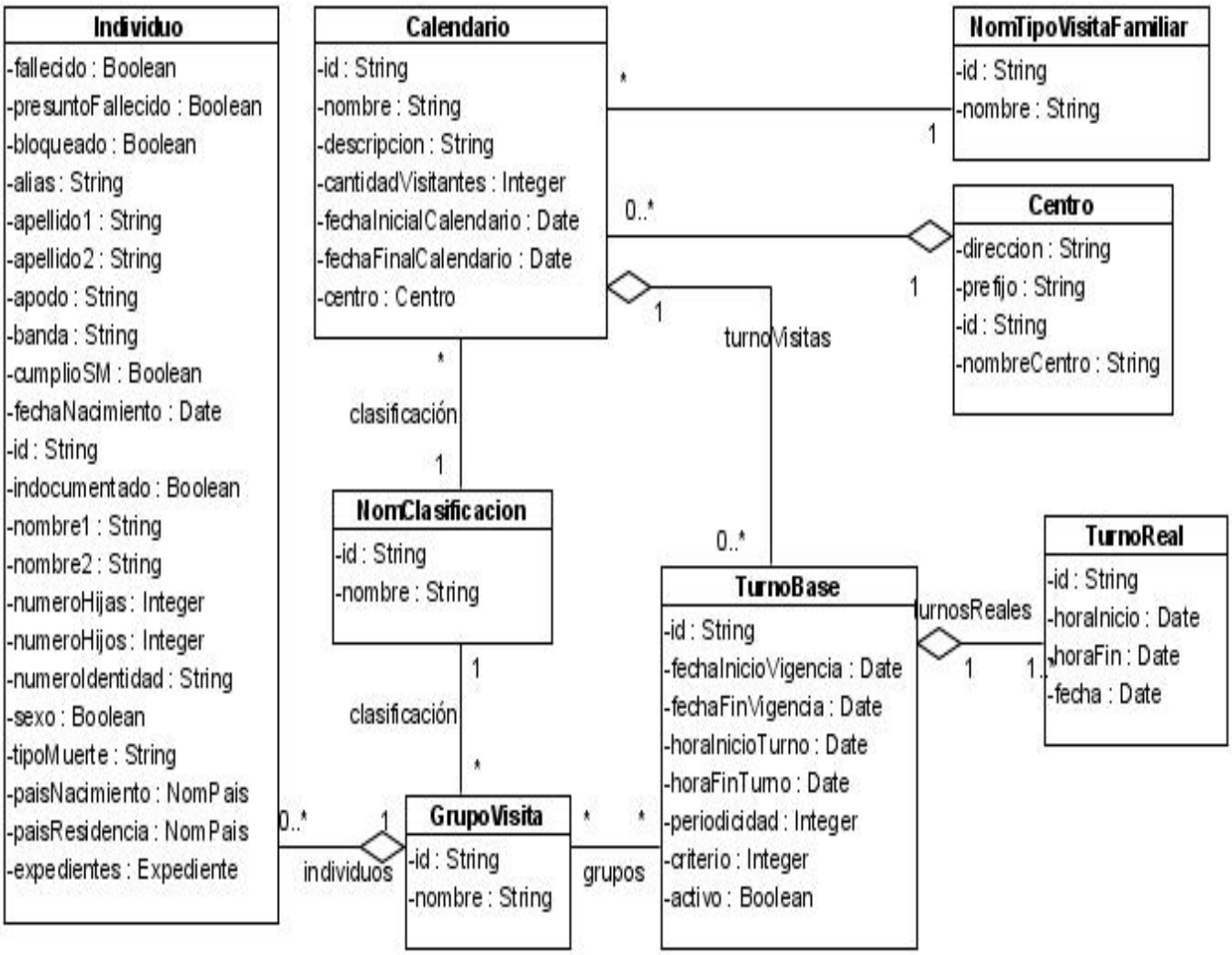

Figura 1. Diagrama de clase del Dominio de: Planificación de Visitas Familiares

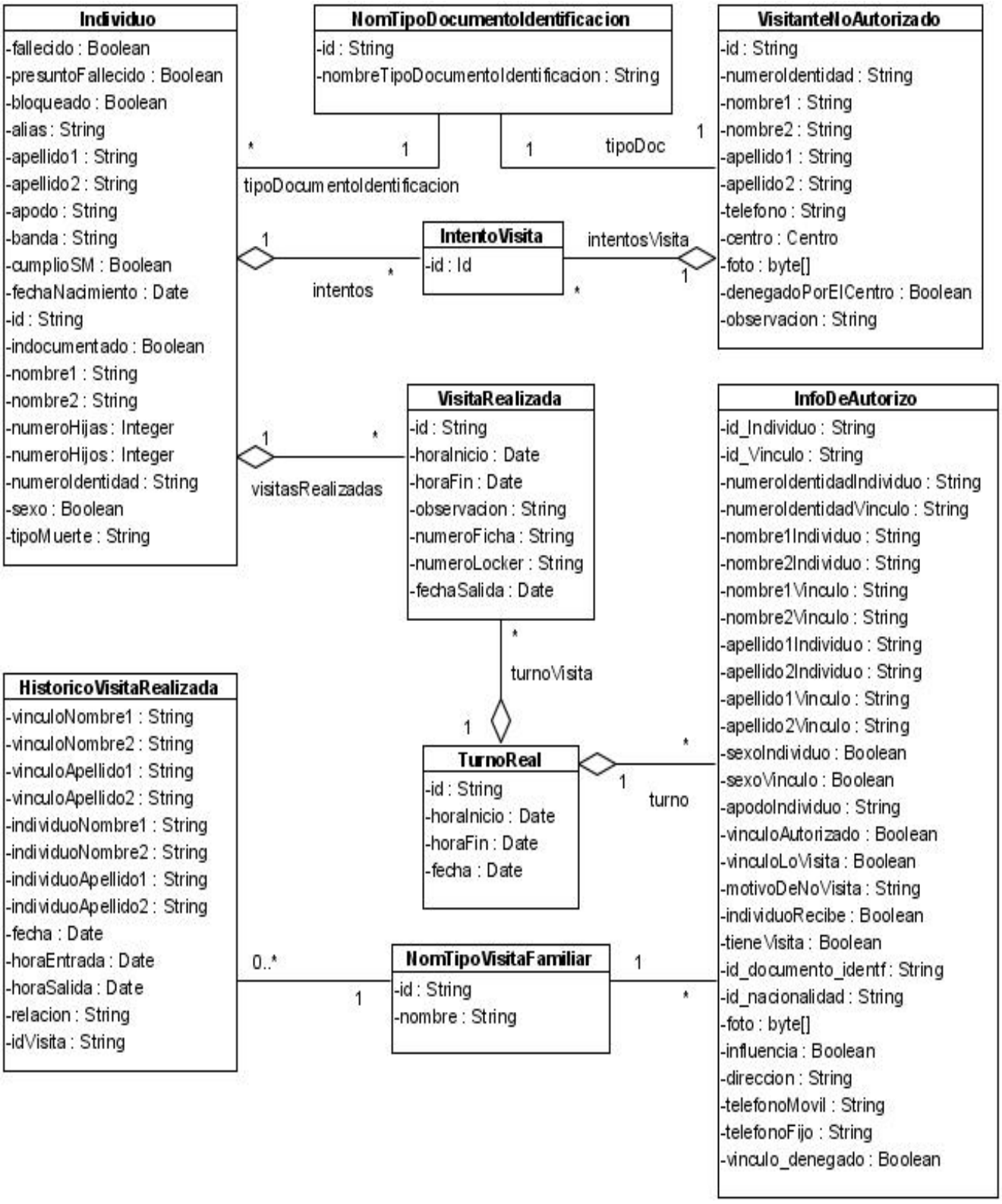

Figura 2. Diagrama de clase del Dominio de: Ejecución de Visitas Familiares.

En las figuras 3 y 4 están representados los diagramas de la capa de lógica de negocio de planificación y ejecución de visitas familiares respectivamente, en los cuales se han utilizado además de los patrones antes mencionados el patrón *Facade* o Fachada.

***Facade* (Fachada):** provee una interfaz unificada sencilla que haga de intermediaria entre un cliente y una interfaz o grupo de interfaces más complejas. Reduce la dependencia entre clases, ofrece un punto de acceso al resto de clases, si estas cambian o se sustituyen por otras sólo hay que actualizar la clase Fachada sin que el cambio afecte a las aplicaciones cliente. Fachada no oculta las clases sino que ofrece una forma más sencilla de acceder a ellas, en los casos en que se requiere se puede acceder directamente a ellas.

El mismo se pone de manifiesto en VisitasFamiliaresFacade la cual es una interfaz que unifica todas las funcionalidades de los módulos en cuestión, haciendo que los módulos sean más fáciles de usar y brindando un bajo acoplamiento en la capa de negocio. La función fundamental de la interfaz VisitasFamiliaresFacade es agrupar las funcionalidades que serán expuestas a las capas superiores y delegar dichas funcionalidades hacia la interfaz VisitaFamiliarManager, que declara los métodos de la lógica de negocio. VisitaFamiliarManagerImpl es la clase donde se implementa toda la lógica de negocio de los módulos planificación y ejecución de visitas familiares, además es donde se hacen las peticiones a la capa de acceso a datos mediante los DAOs.

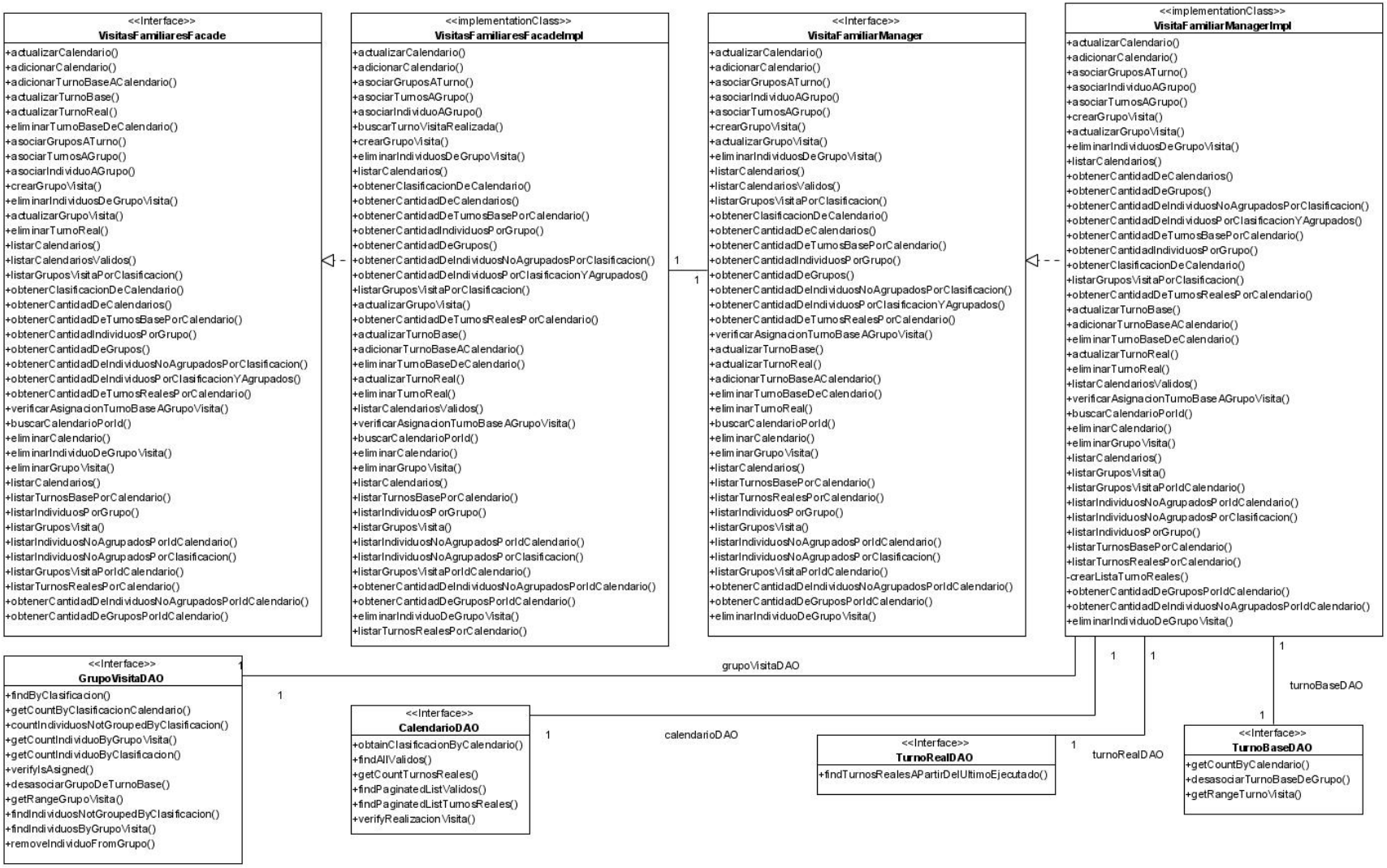

Figura 3. Diagrama de clases del Negocio de: Planificación de Visitas Familiares.

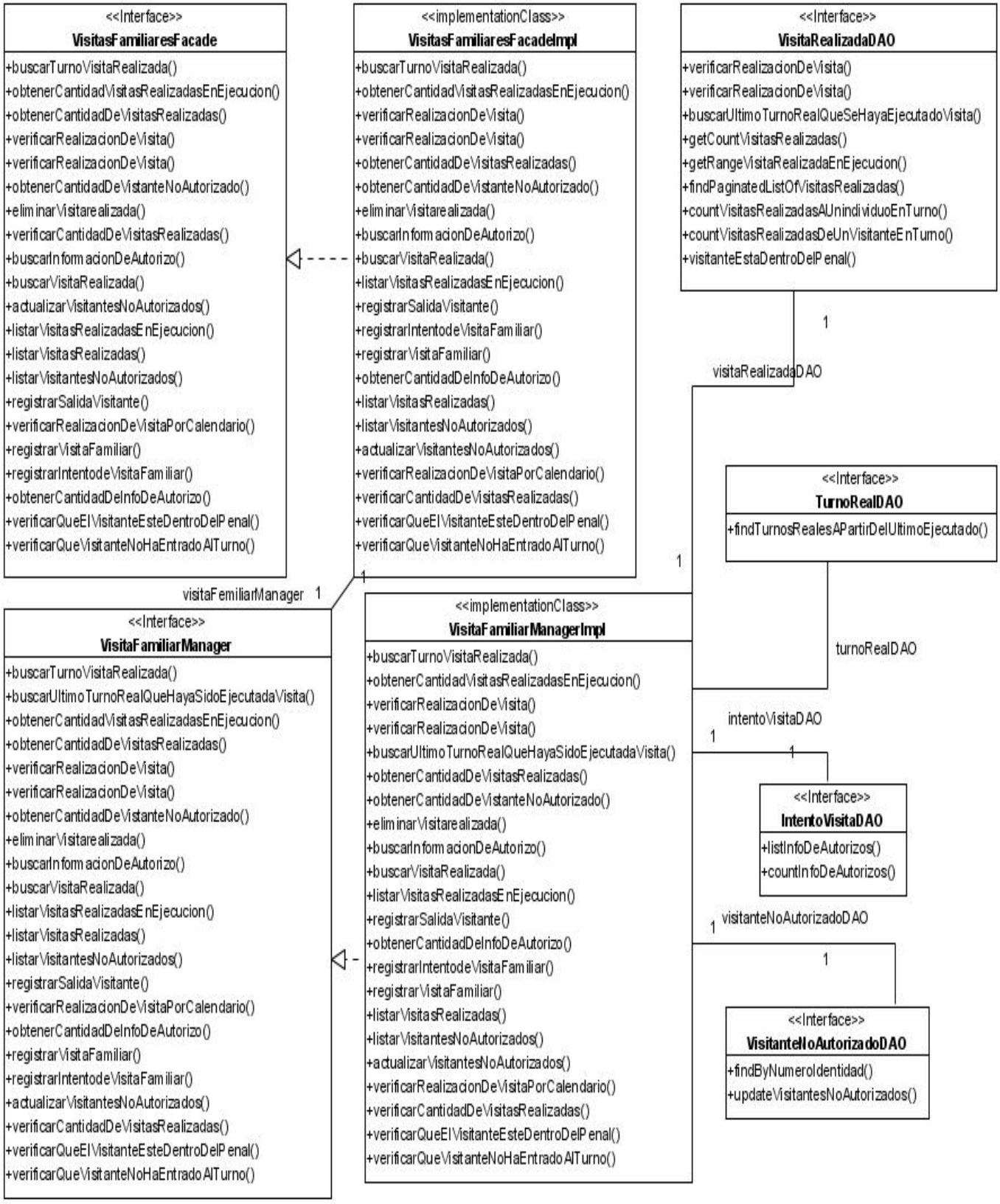

Figura 4. Diagrama de clases del Negocio de: Ejecución de Visitas Familiares.

En las figuras 5 y 6 se muestran los diagramas de la capa de Acceso a datos de los módulos de planificación y ejecución de visitas familiares respectivamente donde se utilizó el patrón de diseño DAO.

DAO: El patrón DAO es una solución que suministra una interfaz común entre la aplicación y uno o más dispositivos de almacenamiento de datos, surge con el objetivo de abstraer y encapsular los mecanismos de obtención y recuperación de datos, convirtiendo los accesos a fuentes de datos en una capa aparte.

Debido a esto si las fuentes de datos cambian tanto de proveedor como de tipo, los componentes de negocios siguen utilizando las mismas interfaces y no deben ser cambiados, ya que lo único que se cambiará es la implementación respectiva a la fuente de datos nueva.

La interacción del negocio con la capa de persistencia se realizará mediante el uso de interfaces. El término de Objeto de Acceso a Datos o en inglés, Data Access Object (DAO) es ampliamente usado en el desarrollo de software.

Los DAOs encapsulan la persistencia de los objetos de dominios, proveen la persistencia de los objetos transitorios y las actualizaciones de los objetos existentes en la base de datos. Las implementaciones de los DAOs estarán disponibles para los objetos (típicamente para los objetos de negocio) haciendo uso de la inyección de dependencias con los objetos de negocio y las instancias de los DAOs, configurada en el contenedor de inversión de control de Spring Framework. Tanto las interfaces como las implementaciones de la capa de acceso a datos extienden de BaseDAO y AbtractBaseDAO respectivamente, interfaces predefinidas en la arquitectura del proyecto las cuales contienen funcionalidades comunes para los DAOs.

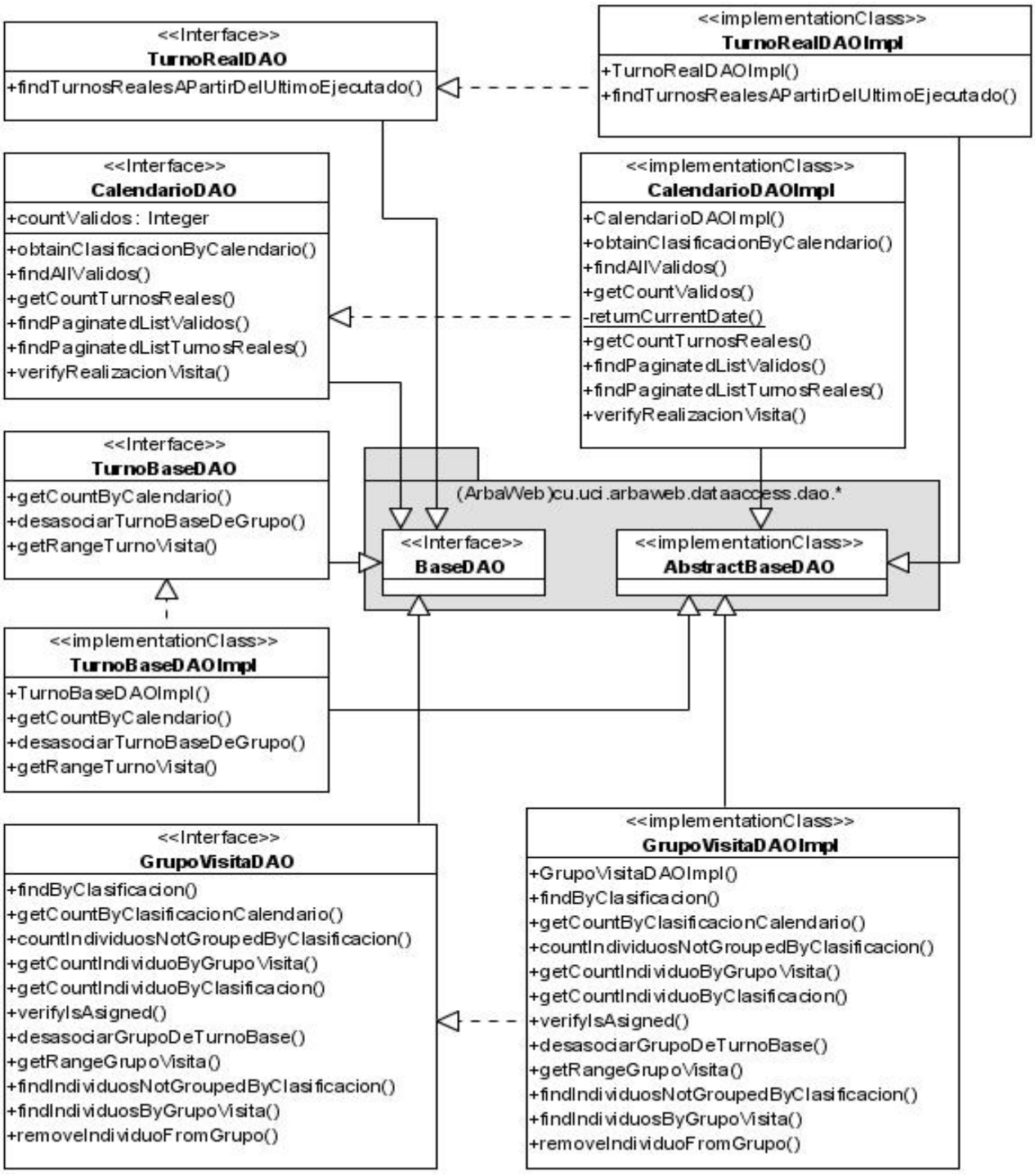

Figura 5. Diagrama de clase de Acceso a Datos: Planificación de Visitas Familiares.

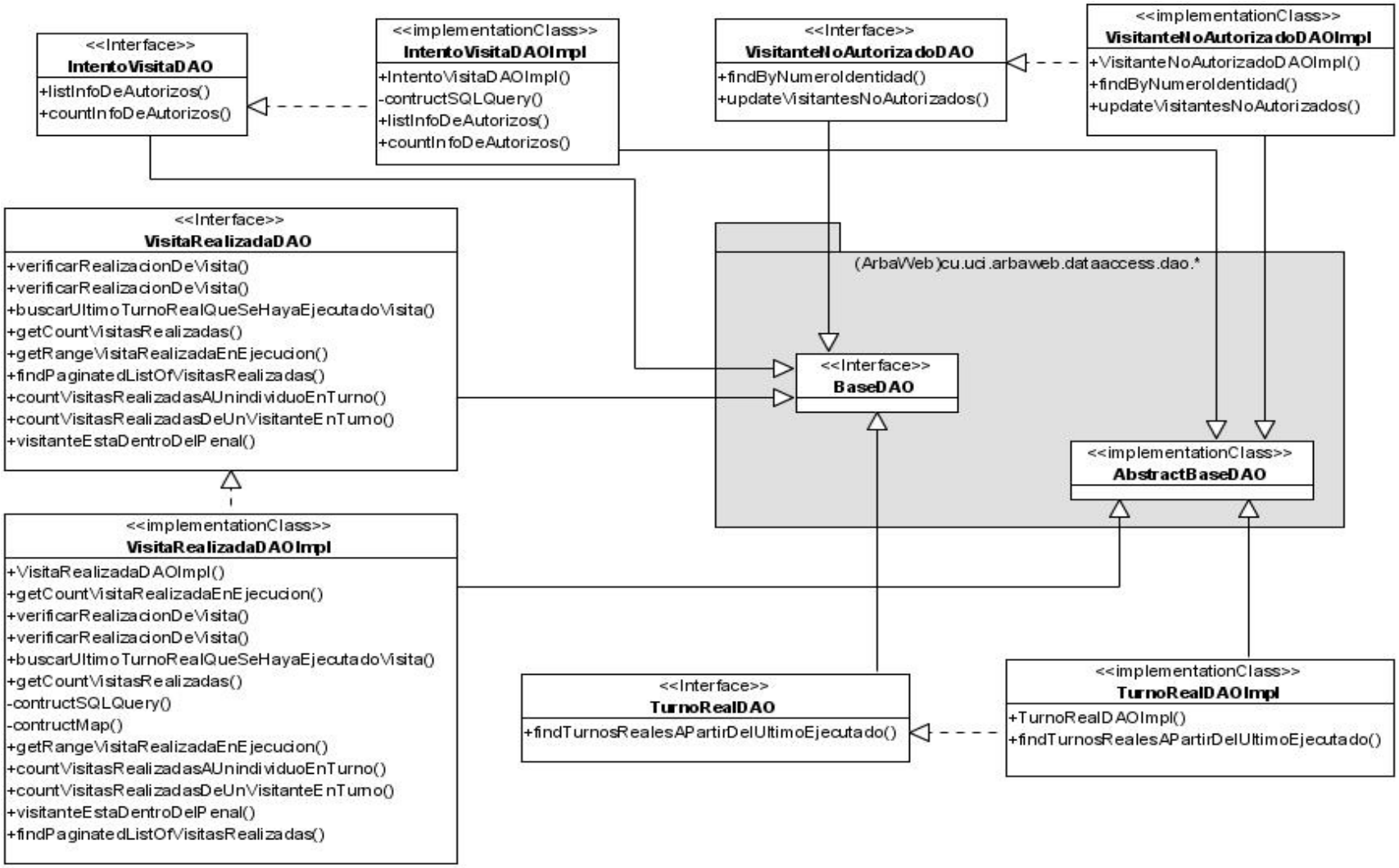

Figura 6. Diagrama de clase de Acceso a Datos: Ejecución de Visitas Familiares.

Diagramas de Secuencia

En los siguientes diagramas de secuencias se muestran las interacciones entre objetos, ordenadas en secuencia temporal durante un escenario concreto. Como los requisitos en cuestión tienen varios flujos se crea un diagrama de secuencia para cada uno de ellos.

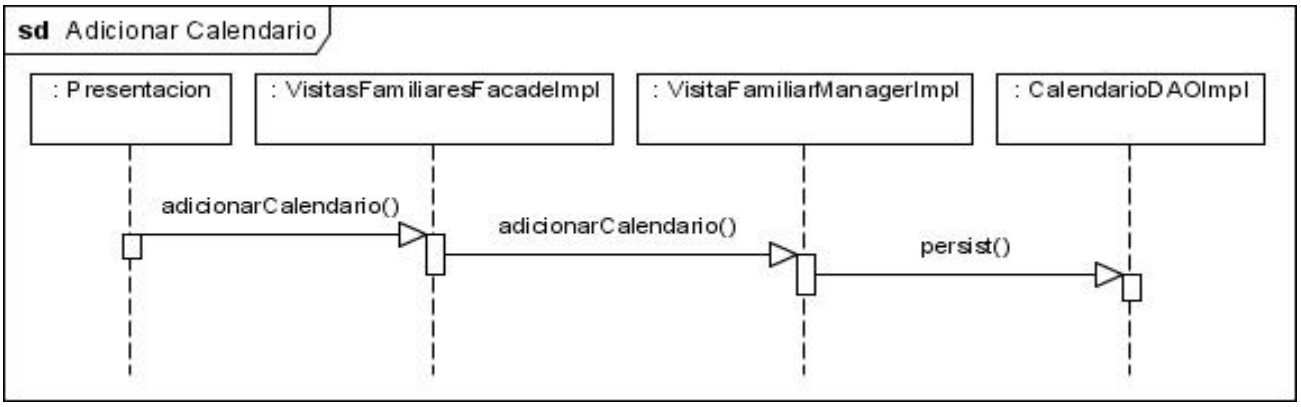

Figura 7. DS: Adicionar Calendario.

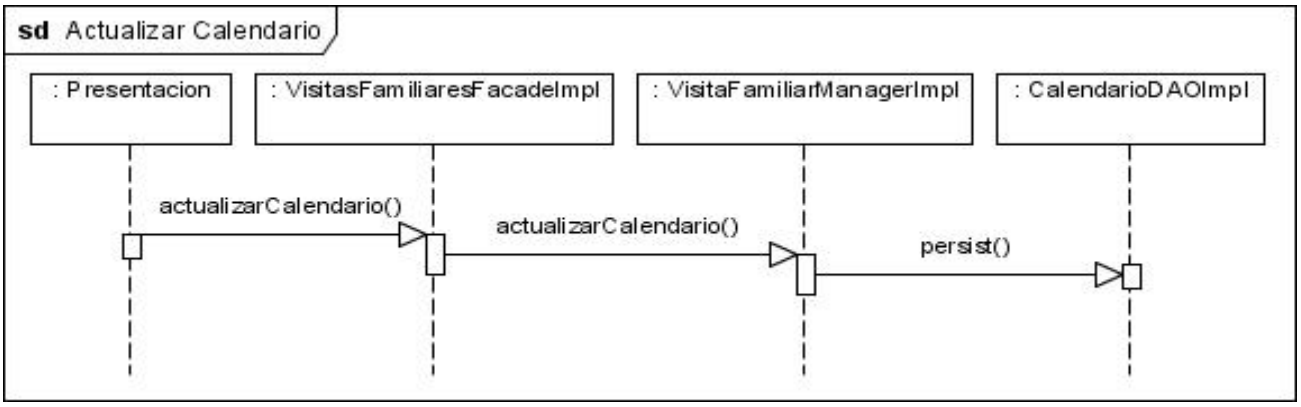

Figura 8. DS: Actualizar Calendario.

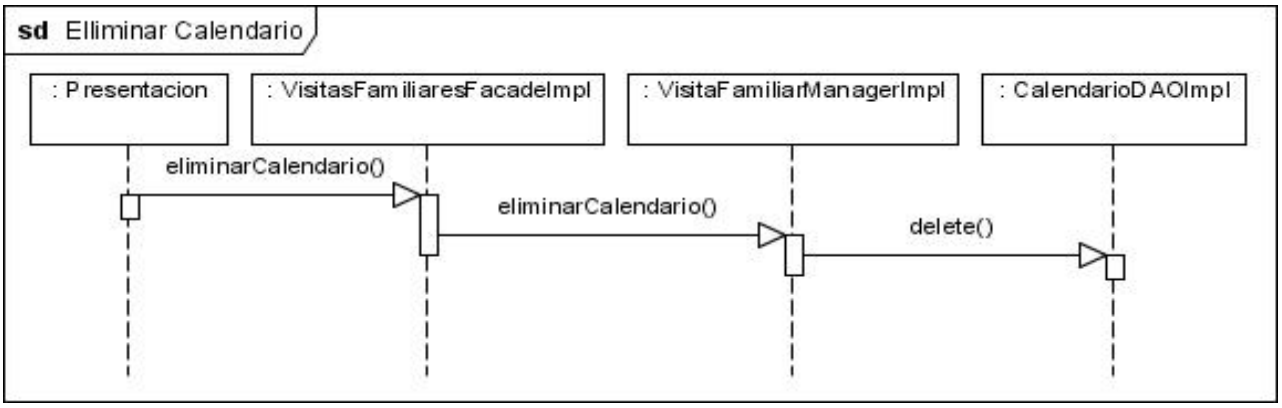

Figura 9. DS: Eliminar Calendario.

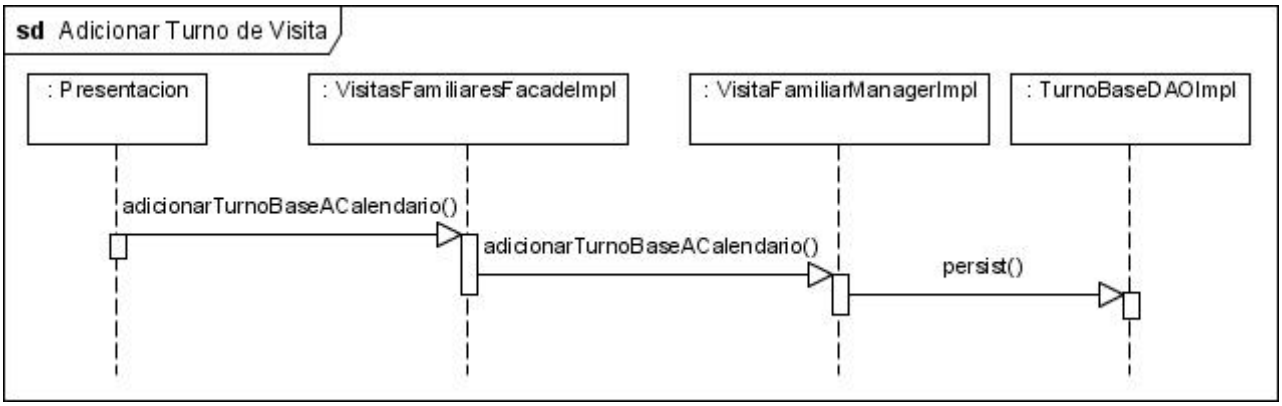

Figura 10. Adicionar Turno de Visita.

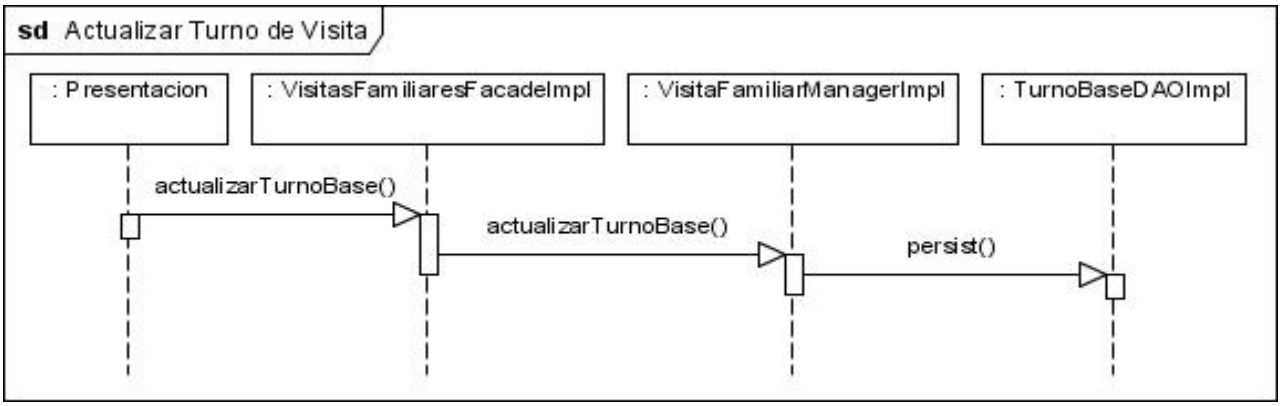

Figura 11. DS: Actualizar Turno de Visita.

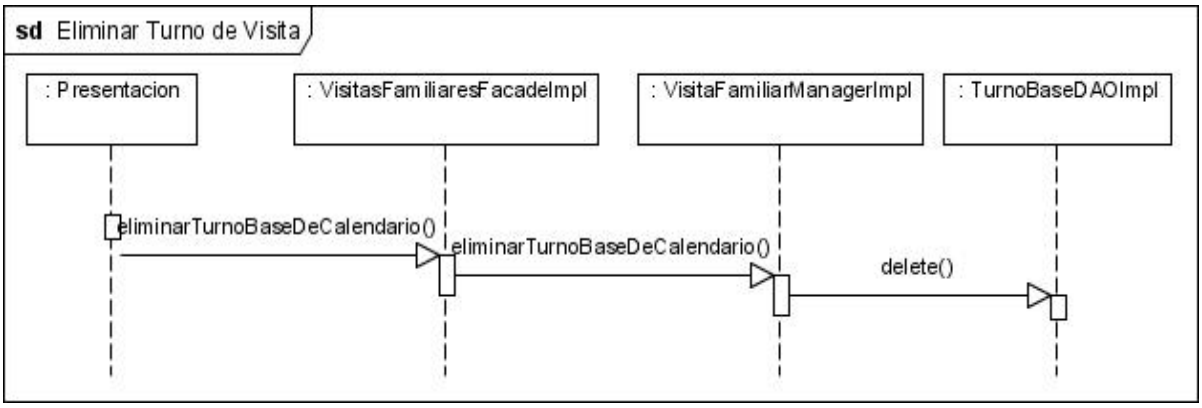

Figure 12 DS: Eliminar Turno de Visita.

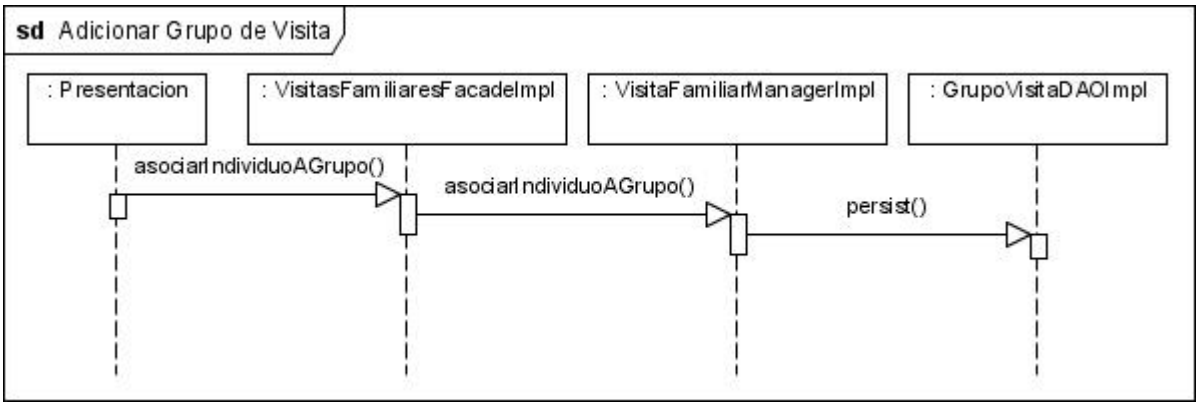

Figura 13. DS: Adicionar Grupo de Visita.

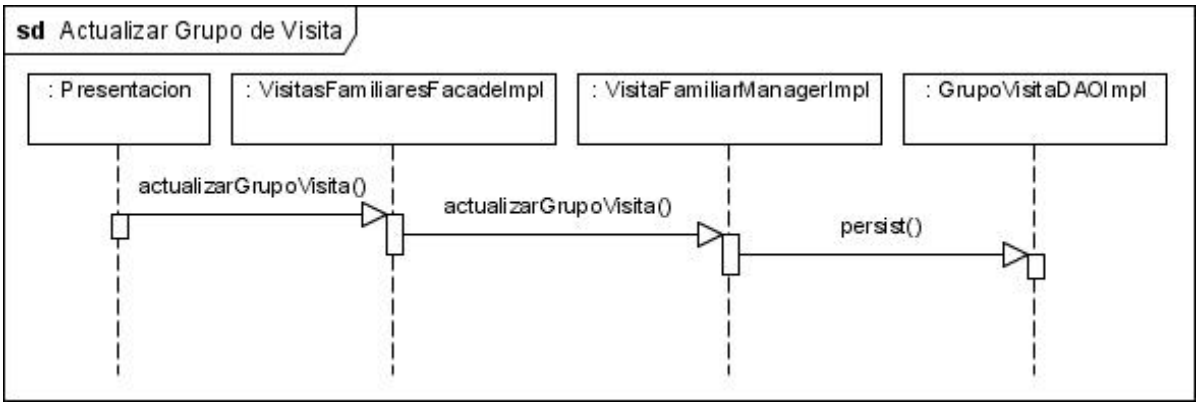

Figura 14. DS: Actualizar Grupo de Visita.

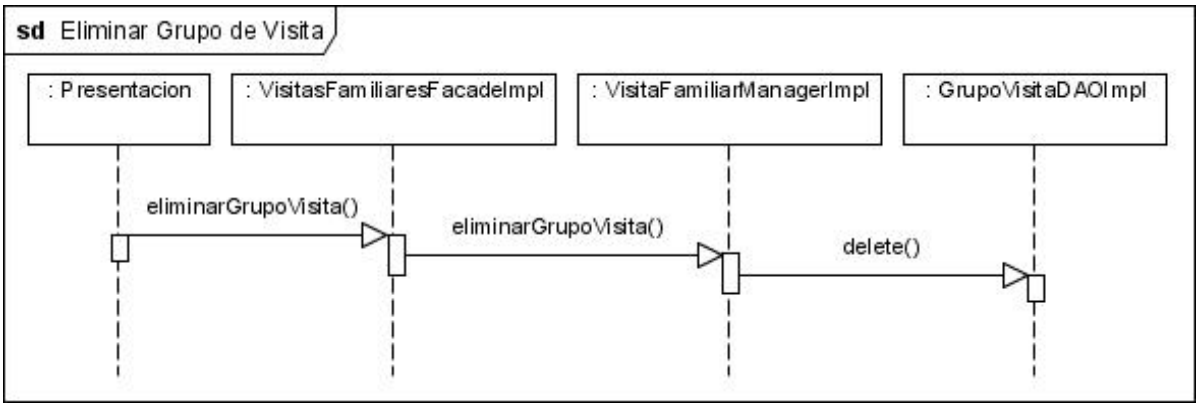

Figura 15. DS: Eliminar Grupo de Visita.

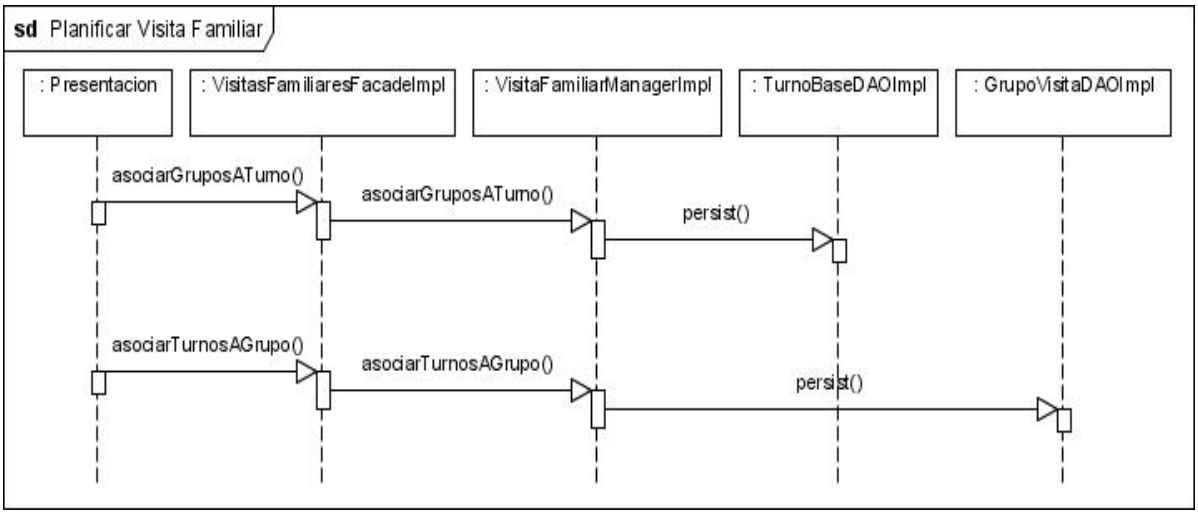

Figura 16. DS: Planificar Visita Familiar.

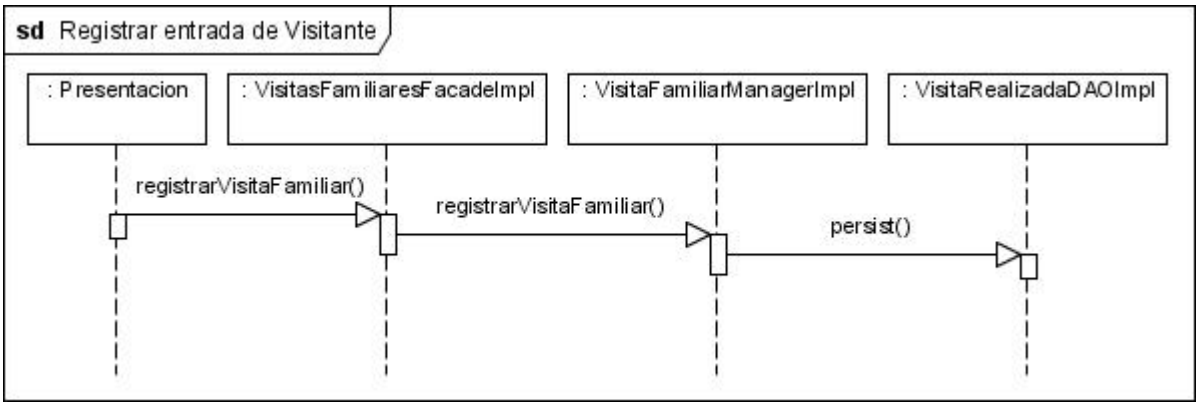

Figura 17. DS: Registrar entrada de Visitante.

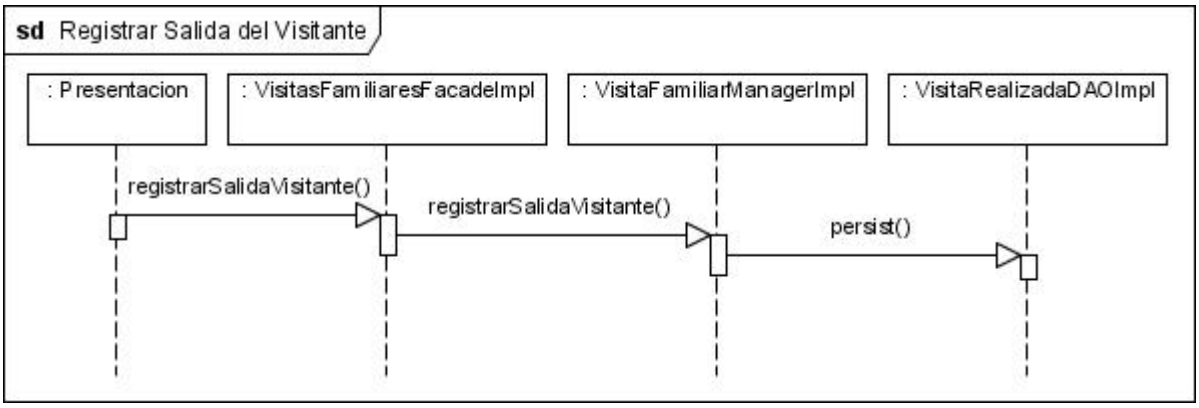

Figura 18. DS: Registrar Salida Visitante.

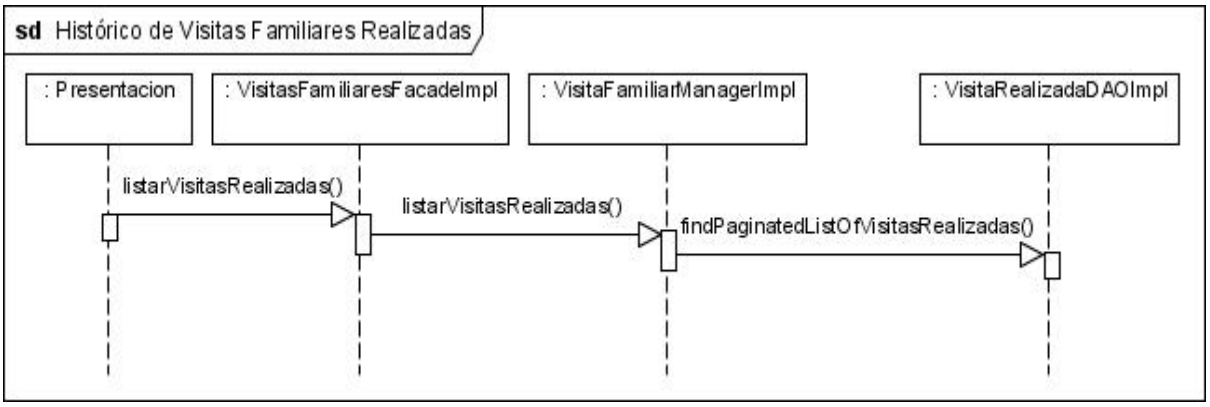

Figura 19. DS: Histórico de Visitas Familiares Realizadas.

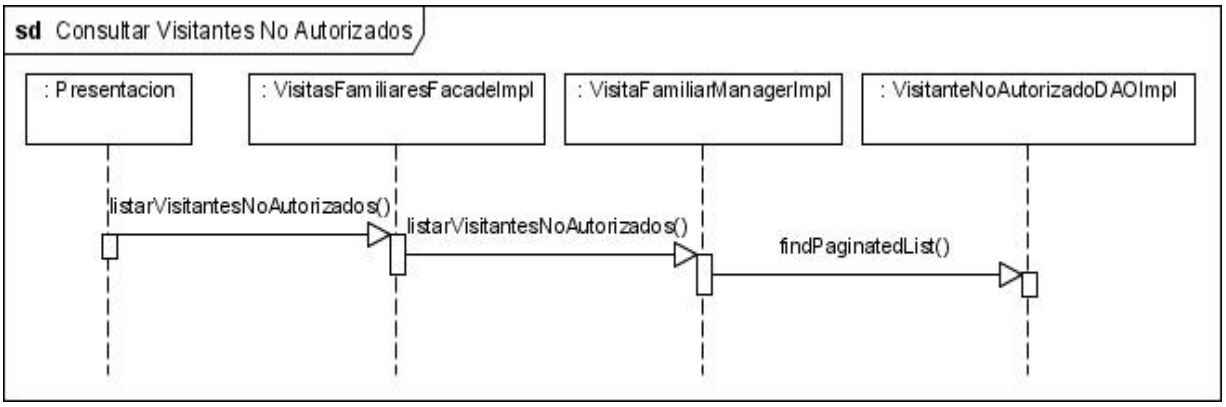

Figura 20. DS: Consultar Visitantes No Autorizados.

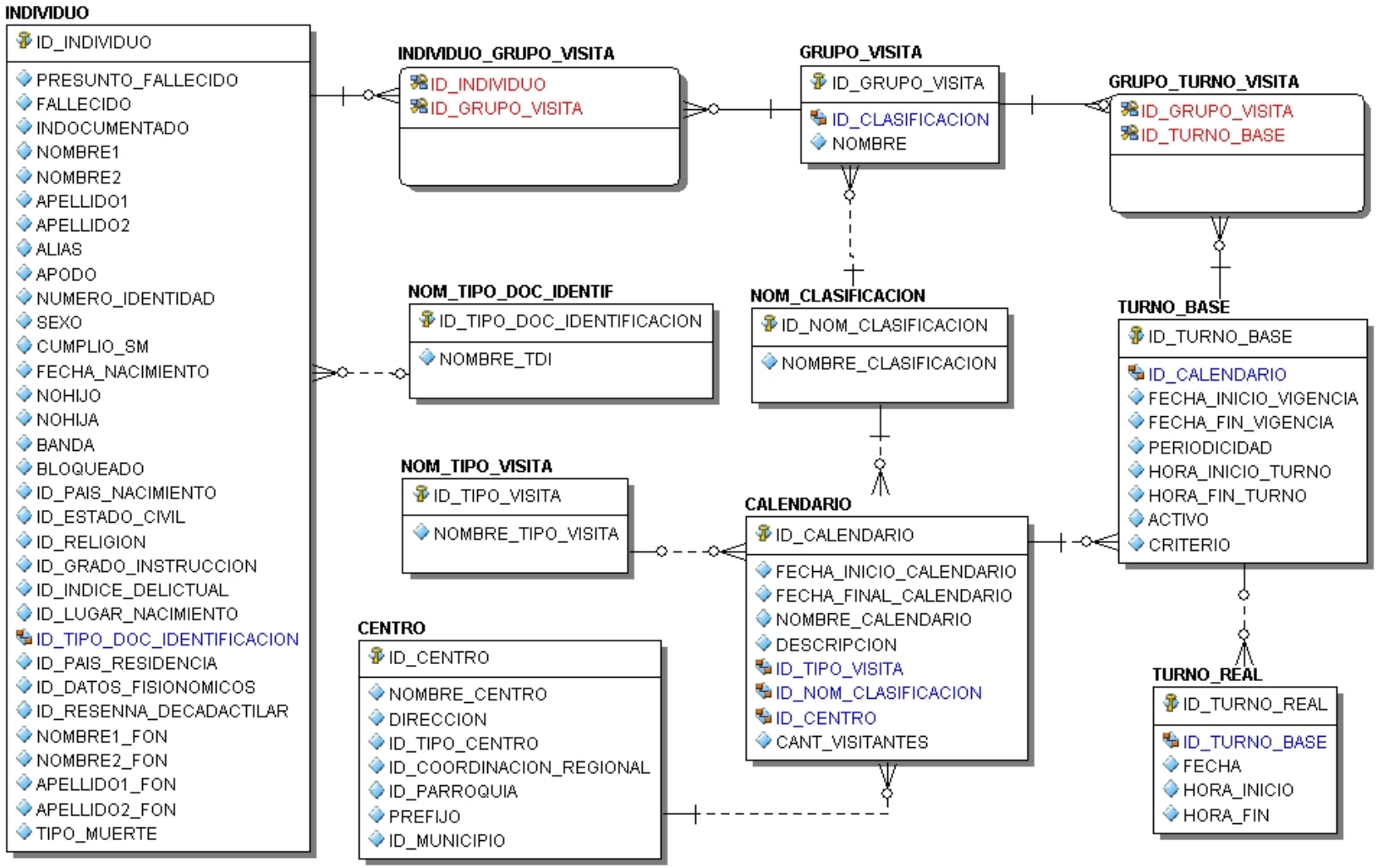

Figura 21. Modelo de datos de Planificación de Visitas Familiares.

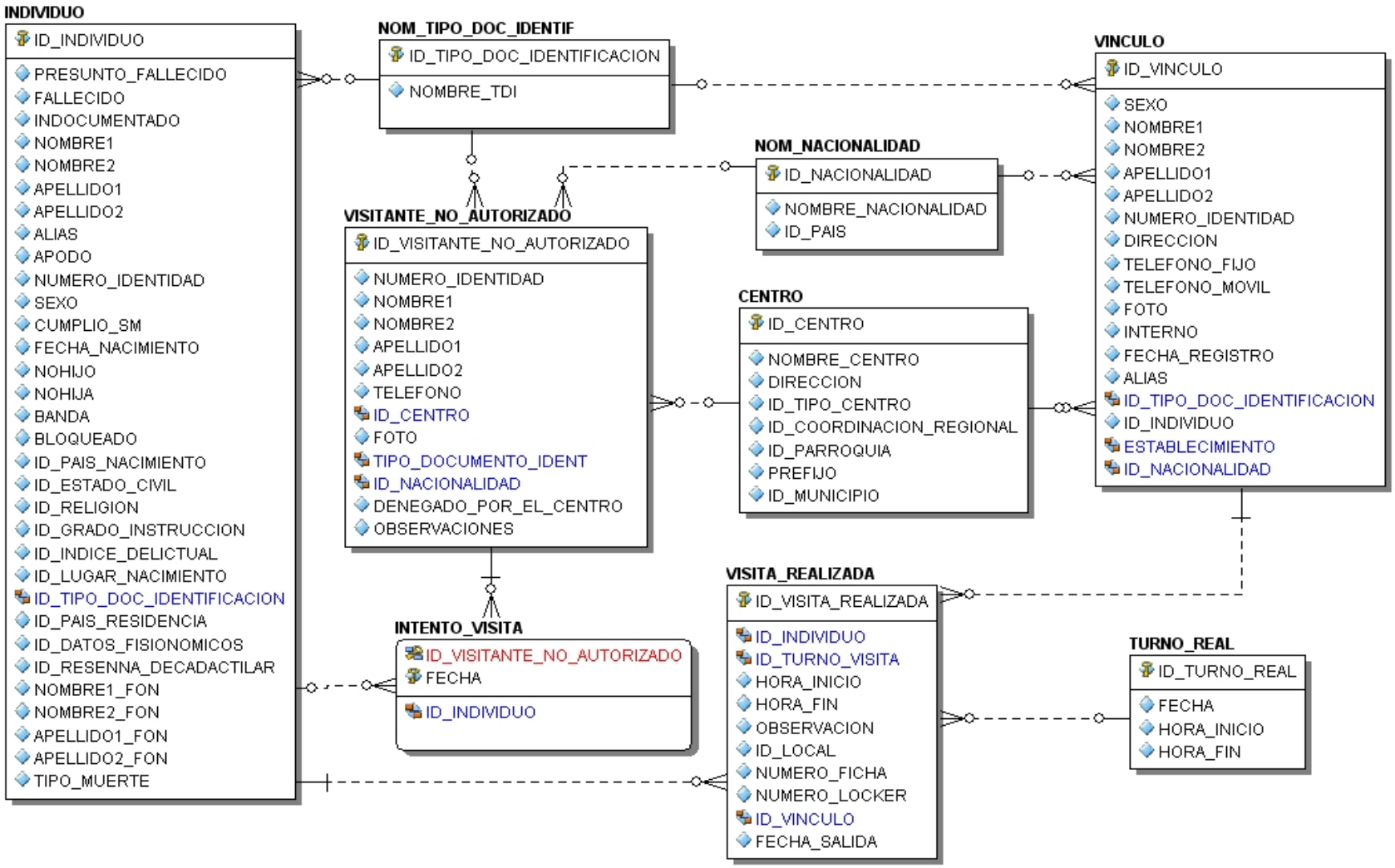

Figura 22. Modelo de datos de Ejecución de Visitas Familiares.

CONCLUSIONES

Se diseñaron las capas de Negocio y Acceso a Datos de los módulos Planificación y Ejecución de visitas familiares, utilizando para ello los diagramas de clases e interacción.

Se diseñó el modelo de datos de acuerdo con las entidades involucradas en la solución de los módulos.

Se implementó la solución del problema usando los artefactos brindados por el diseño con las herramientas, tecnologías y convenciones de código definidas en la arquitectura del proyecto.

Se logró cumplir con lo requerido por el cliente pues los módulos están probados y aceptados por el mismo y se encuentran funcionando en un penal piloto en la República Bolivariana de Venezuela.

.

RECOMENDACIONES

Realizar pruebas bajo condiciones donde haya mayor volumen de registros de datos.

Valorar el uso de estos módulos en sistemas penitenciarios desarrollados bajo la plataforma JEE con características similares que no cuenten con esta utilidad.

De igual manera el flujo de trabajo seguido podría servir durante el diseño y la implementación de la capa de negocio y acceso a datos de sistemas desarrollados bajo la plataforma JEE.

BIBLIOGRAFÍA

ARIAS, A. Proyecto Técnico de Asesoría Especializada, Colaboración Médica Odontológica, Comunicación Institucional y Solución Tecnológica para apoyar la modernización del Sistema Penitenciario de la República Bolivariana de Venezuela UCI-DGCRR, 2006: 95.

BENAVIDES, Y. Flujo de Trabajo: Diseñador Versión 1.0, 2006: 6.

BETANCOURT, A. Custodia y Seguridad: Procesos Elementales del Negocio. Coordinación y Ejecución de Visitas Familiares., Universidad de las Ciencias Informáticas, 2007: 50.

CARACTERISTICAS DE JAVA. Disponible en: http://www.cica.es/formacion/JavaTut/Intro/carac.html

DELGADO, J. C. Flujo de Trabajo: Programador de lógica de Negocio Versión 1.0, 2006: 7.

E.S.TAYLOR An Interim Report on Engineering Design, Massachusetts Institute of Technology, 1959.

GALLARDO, D. E. B., ROBERT MCGOVERN. Eclipse In Action: A Guide for Web Developers, 2003.

IEEE IEEE Std 1471-2000 IEEE Recommended Practice for Architectural Description of Software-Intensive Systems -Description, 1995: 42-50.

JACOBSON, I. El Proceso Unificado de Desarrollo de Software. Empresa Poligráfica de Holguín, 2004: 435.

JACOBSON, I., GRADY BOOCH, JAMES RUMBAUGH, El Proceso Unificado de Desarrollo de Software. México: Addison-Wesley, 1999: 57.

JOHNSON, R. Professional Java Development with the Spring Framework, 2005: 0764574833.

KRUCHTEN, P. "Architectural Blueprints--The 4+1 View Model of Software Architecture".

LARMAN, C. UML Y PATRONES. 1999. 189 p. 0-13-748880-7

--- UML y patrones, Prentice Hall, 2002: 520.

PIMIENTEL, L. Documento de arquitectura del SIGEP, Universidad de las Ciencias Informáticas, 2006.

Relational Persistence for Java. Disponible en: http://hibernate.bluemars.net

SAAVEDRA, J. PATRONES GRASP (Patrones de Software para la asignación General de Responsabilidad). Disponible en: http://jorgesaavedra.wordpress.com/category/patrones-grasp/]

GLOSARIO

1. **SIGEP:** Sistema de Gestión Penitenciaria.
2. **DGCRR:** Dirección General de Custodia y Rehabilitación del Recluso
3. **IDE:** Integrated Development Enviroment (Ambiente Integrado de Desarrollo).
4. **POJO:** Es el acrónimo de Plain Old Java Object. Clase del lenguaje de programación Java. Este nombre se les da a las clases que no son de algún tipo especial (EJBs, Java Beans, etcétera) y no cumplen ningún otro rol ni implementan alguna interfaz especial.
5. **Plugins:** Un plugin o plug-in, es una aplicación que interactúa con otra para agregarle una funcionalidad específica y es ejecutada por la aplicación principal. En el caso particular de Eclipse no son más que un conjunto de clases que permiten hacerlo más extensible.
6. **Socket**: queda definido por una dirección IP, un protocolo y un número de puerto.
7. **Framework**: estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado