# Universidad de las Ciencias Informáticas

**"Facultad 4"**


# Título: “Diseño e implementación del módulo Novedades del Sistema de Gestión Penitenciaria de la República Bolivariana de Venezuela.”

Trabajo de Diploma para optar por el título de

Ingeniero Informático

**Autores:** Yusleidys Fuentes Puig

Annaliet Parra Pérez

**Tutor:** Ing. Angel Luis Lecuona Rodríguez

Ciudad de La Habana, Mayo de 2010

**DECLARACIÓN DE AUTORÍA**

Declaramos que somos las únicas autoras de este trabajo y autorizamos a la facultad 4 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 _____.

"Yusleidys Fuentes Puig" "Annaliet Parra Pérez"

_____ _____

"Ing. Angel Luis Lecuona Rodríguez"

_____

**DATOS DE CONTACTO**

Angel Luis Lecuona Rodríguez

- Graduado en 2008 de la carrera Ingeniería en Ciencias Informática en la UCI con Título de Oro.
- Opta actualmente por la categoría docente de Instructor.
- Se ha vinculado a los proyectos productivos desde el año 2005 en el rol de Diseñador y Jefe de módulo.
- Ha impartido cursos de capacitación a funcionarios del MIJ durante el despliegue del proyecto SIGEP en Venezuela.
- Ocupó el cargo de Jefe de Colectivo de la asignatura Inteligencia Artificial en el presente curso.

**AGRADECIMIENTOS**

*A nuestro tutor por ayudarnos en todo momento y ser sobre todo nuestro amigo.*

*A Lizandra por brindarnos su experiencia y estar siempre al tanto de nosotras.*

*A nuestros compañeros, amigos y profesores por los conocimientos que aportaron para el desarrollo de este trabajo, en especial a Eduardo.*

*A la universidad por darnos la oportunidad de convertirnos en lo que somos.*

*A Leydis y Taylen por ser nuestras hermanas durante estos 5 años.*

*A todos muchas gracias…*

**DEDICATORIA**

*Para quienes han convertido el significado de sus vidas, en objetivo de las nuestras…*

*Nuestros padres.*

**RESUMEN**

El Sistema Penitenciario Venezolano no registraba los reclusos por cada centro y por tanto ningún dato de los mismos, afectando el funcionamiento de este, por lo que se le asigna a la Universidad de las Ciencias Informáticas la informatización de todo este proceso. Una vez desarrollado y desplegado el software en su primera versión surge la necesidad de agregarle otras funcionalidades como la gestión de las novedades que podrían ocurrir en los centros penitenciarios.

Para darle cumplimiento a las funcionalidades esperadas se decidió realizar un estudio sobre las tecnologías y herramientas a utilizar, además de documentar todo el proceso de diseño e implementación de las mismas.

**PALABRAS CLAVE**

Novedad, SIGEP, Arquitectura, Diseño, Implementación.

**TABLA DE CONTENIDOS**

## INTRODUCCIÓN

El Sistema Penitenciario venezolano fue objeto de alguna de las transformaciones que tuvieron lugar en la República Bolivariana de Venezuela después de tomar en febrero de 1999 la presidencia Hugo Rafael Chávez Frías. Este es un sector que presentaba significativos problemas en Venezuela con altos índices de corrupción y violencia, además del descontrol existente, creaba un caos político en el país.

La falta de seguimiento y control en el sistema existente provocaba que no se tuviera claridad en el registro de los reclusos por cada centro penitenciario, influyendo en esto el hecho de que todos sus expedientes se llevaban en papel, por tanto muchos datos de los reclusos y sus sanciones no eran procesados, lo cual dificultaba el buen funcionamiento de los centros penitenciarios y la atención adecuada de los internos, teniendo esto un reflejo en el desconocimiento de tribunales y familiares sobre la situación de los reclusos.

Como fruto del convenio de colaboración Cuba-Venezuela, en el marco de la Alternativa Bolivariana para los Pueblos de Nuestra América (ALBA), se establece un convenio de trabajo para la Humanización del sistema penitenciario venezolano, que incluye la informatización de sus procesos, siendo esta última tarea de la Universidad de las Ciencias Informáticas, de manera que se logre en gran medida que los problemas expuestos anteriormente sean resueltos paulatinamente.

En junio del 2006 se crea en la universidad un grupo de desarrollo para crear una solución de software de amplio alcance que toma como nombre, Sistema de Gestión Penitenciaria (SIGEP), que logra en julio del 2008 responder a los requisitos capturados inicialmente.

El propio desarrollo del sistema impulsó a que las áreas a informatizar crearan o mejoraran su modelo de funcionamiento una vez que fuera desplegado el software de manera satisfactoria, surge entonces la necesidad de agregarle otras funcionalidades al mismo debido a cambios estructurales y funcionales que se le realizaron a los centros penitenciarios y al conocimiento de las potencialidades que podía brindar un software como el que estaban utilizando. Como parte de las nuevas funcionalidades identificadas se plantea la necesidad de controlar de manera eficiente y efectiva el proceso de recogida de información referente a las ocurrencias o novedades detectadas en los centros penitenciarios, dígase fugas, huelgas, y otros incidentes de relevancia.

La incorporación de esta nueva funcionalidad implica que la misma se acople perfectamente al sistema ya implementado para que brinde los servicios y funciones necesarios, por lo que debe ser desarrollada sobre la misma arquitectura utilizando las tecnologías definidas por el proyecto.

A partir de esta situación, se ha definido como problema a resolver que *el Sistema de Gestión Penitenciaria necesita agregar nuevas funcionalidades que permitan gestionar las novedades sin quebrar la arquitectura ni incumplir con los requisitos definidos por el proyecto.*

El problema planteado, enmarca su ***objeto de estudio*** en el Proceso de Control Penal perteneciente al Sistema Penitenciario venezolano, teniendo como ***campo de acción*** el Módulo Novedades del Sistema de Gestión Penitenciaria de la República Bolivariana de Venezuela.

Para la solución del problema el equipo se plantea los siguientes objetivos:

Objetivo general:

- Desarrollar el módulo Novedades del SIGEP a partir del análisis de los requisitos de software utilizando las tecnologías y herramientas definidas por el proyecto.

Objetivos específicos:

- Diseñar el módulo Novedades del SIGEP.
- Implementar el módulo Novedades del SIGEP.
- Integrar el módulo Novedades al SIGEP.

Las tareas planificadas para dar cumplimiento a los objetivos son:

- Caracterización y descripción de las tecnologías y herramientas definidas en la arquitectura del SIGEP.
- Conocer con profundidad los requisitos de software correspondiente al módulo Novedades.
- Elaboración del Diagrama Entidad-Relación.
- Ajustar la base de datos del SIGEP de acuerdo a las necesidades del módulo a desarrollar.
- Diseño de la solución de software para los requisitos relacionados con el proceso de gestionar

novedades.

- Implementación del diseño realizado relacionado con el proceso de gestionar novedades.
- Realización de las pruebas unitarias.
- Integración del módulo Novedades al SIGEP.

# 1. FUNDAMENTACIÓN TEÓRICA

## 1.1 Introducción

En el presente capítulo se abordan algunos conceptos a los cuales se harán referencia en todo el contexto del trabajo como es el caso de Sistema Penitenciario, Seguridad y Custodia y otros, todos de importancia para el entendimiento del trabajo realizado. Se brinda información acerca de los Procesos de Ingreso de las novedades según el tipo y del Proceso Registrar y Modificar novedades. Se analizan las tecnologías, herramientas y tendencias de desarrollo que se utilizarán durante el trabajo, en el marco del desarrollo del Sistema de Gestión Penitenciaria (SIGEP).

## 1.2 Sistemas penitenciarios

Un sistema no es más que un conjunto de reglas o principios sobre una materia racionalmente enlazados entre sí o se pudiera decir que además es un conjunto de elementos que relacionados entre sí ordenadamente contribuyen a determinado objeto.

Al asignar al régimen penitenciario la calificación de Sistema penitenciario se hace referencia a un conjunto de normas, procedimientos y dependencias dispuestas por el Estado para la ejecución del régimen penitenciario, entre los que se encuentran además los principios, programas, recursos humanos, dependencias e infraestructura que se encuentran relacionadas y destinadas a este régimen.

Manuel Osorio, creador del diccionario de Ciencias Jurídicas, Políticas y Sociales, asocia el Sistema Penitenciario con régimen penitenciario, definiendo éste régimen como: "conjunto de normas legislativas o administrativas encaminadas a determinar los diferentes sistemas adoptados para que los penados cumplan sus penas. Se encamina a obtener la mayor eficacia en la custodia o en la readaptación social de los delincuentes. Esos regímenes son múltiples, varían a través de los tiempos; y van desde el aislamiento absoluto y de tratamiento rígido hasta el sistema de puerta abierta con libertad vigilada". (OSORIO, 1997)

Cada gobierno define la estructura de su sistema penitenciario de acuerdo a la legislación y las condiciones reales que posee. En el caso específico de la República Bolivariana de Venezuela, tal sistema está constituido por la legislación vigente, los métodos que se emplearán para lograr su funcionamiento, las diferentes dependencias encargadas de su aplicación, los equipos de trabajo y la infraestructura carcelaria.

### 1.3 Algunas soluciones existentes relacionadas con los Sistemas Penitenciarios

Para el desarrollo del SIGEP se hizo un análisis de las características de los principales sistemas informáticos implantados en la región los que fueron valorados como posible solución o parte de la solución para realizar este trabajo, se analizaron teniendo en cuenta las funcionalidades que brindan. Estos sistemas tienen en común que permiten mantener identificado a cada uno de los individuos que ingresan al Sistema Penitenciario, registrando sus datos personales, rasgos físicos y señas particulares, entre otros aspectos de interés. Seguidamente se mencionaran algunos de estos sistemas informáticos estudiados.

**Sistema Automatizado para el Control del Recluso (SACORE) (KNIGHT 2005)**

Surge para dar cumplimiento a la Orden 43/99 del Vice Ministro Primero del Ministerio del Interior de Cuba y tiene las siguientes características:

- Garantiza respuestas inmediatas a las solicitudes de información de los diferentes órganos e instituciones del estado como son: Jefatura del MININT, Ministerio de Justicia, Tribunales, Fiscalías, MINED, INDER, FMC, MINFAR.
- Recoge prácticamente la totalidad de la información de los reclusos en todas las especialidades.
- Tiene más de 200 reportes impresos.
- Permite la recuperación dinámica a partir de una solicitud de búsqueda.
- Los partes que se emiten son obtenidos de forma automatizada.
- Permite el traslado automático de todos los datos del recluso al nivel nacional.

**Establecimiento Penitenciario Virtual en Red (España)**

Sitio web del proyecto: http://sourceforge.net/projects/epvnet

Establecimiento Penitenciario Virtual en Red (EPVNET) es un proyecto de software libre para la gestión informatizada de Centros Penitenciarios.

El proyecto EPVNET es un software de gestión para la gerencia de los internos y de los empleados en cualquier centro penitenciario basado en la legislación penal española que garantiza el control de

movimientos, de comunicaciones y de informes. Comenzó como un proyecto aislado en el año 2002 para dar una solución de gestión en el centro penitenciario de Valencia. Para el 2004 se convirtió en un proyecto de software libre hospedado en sourceforge.net bajo licencia GNU/GPL. Es una aplicación web desarrollada en PHP4 y PL/SQL, con PostgreSQL como gestor de base de datos.

**Sistema Automatizado NEOTEC (Bolivia)**

Sitio web del proyecto: http://www.neotec.cc/solutions/i3j/index_es.html

NEOTEC software realizado por una empresa de consultoría y desarrollo de software en La Paz, Bolivia, que asegura la gestión para los centros penitenciarios ofreciendo los siguientes beneficios:

- Certeza de la identidad del prisionero, visitante y trabajador de la penitenciaría.
- Registro de la huella digital en la entrada y salida de personas para evitar usurpación de identidad.
- Registro de entradas y salidas.
- Procedimiento de autorización de salidas de reclusos.
- Registro de visitas a reclusos.
- Registro de comportamiento incluyendo estudio, trabajo y sanciones.
- Supervisión de libertad condicional y beneficios extramuros.
- Sistema integrado nacional.

Es un sistema basado en la Web y tiene una base de datos centralizada. La información de los prisioneros está disponible a las instituciones del sistema penitenciario y es posible otorgar acceso a las cortes, la policía y la fiscalía. Esta información incluye el registro de los reclusos, movimientos, visitas a reclusos, compañeros de celda, comportamiento, trabajo estudio, monitoreo de libertad condicional y otros elementos.

Una vez analizados los sistemas informáticos existentes, el equipo de desarrollo de software del proyecto, a partir de la estructura actual del sistema penitenciario venezolano, identificó las áreas que el SIGEP informatizaría y se conformaron los sistemas, subsistemas y módulos que este contendría. Seguidamente

se mencionará el subsistema Seguridad y Custodia, donde están comprendidas las novedades, que son el motivo del presente trabajo.

### 1.4 Seguridad y Custodia

La seguridad penitenciaria en el ámbito operacional maneja dos modalidades, la llamada seguridad interna y la externa:

- **Seguridad interna:** se ubica dentro de las paredes del edificio de reclusión propiamente dicho, incluido el cinturón interior de seguridad y las murallas que delimitan la zona considerada de alta seguridad, donde la circulación está restringida a personas y a vehículos no oficiales y donde el personal de seguridad que cubre este servicio está autorizado a disparar si el caso lo requiere. Dentro de la seguridad interna se encuentran consideradas áreas críticas, tales como: la planta de luz, calderas, depósito de armas, vestidores del personal de vigilancia, aduana, dormitorios para internos considerados de máxima peligrosidad. Contempla también el adecuado desarrollo de las diferentes actividades y conductas propias de la población de internos en general, así como de la confianza y tranquilidad de quienes por cualquier motivo se encuentren dentro del perímetro del establecimiento.
- **Seguridad externa:** se refiere a la protección que en toda institución penitenciaria se requiere establecer, previniendo posibles ataques desde el exterior, está delimitada entre otras cosas por: marcas visibles que señalan impedimentos al libre acceso de personas y vehículos que lleguen del exterior y que por algún motivo pretenden entrar a la institución. Contempla toda el área circunvecina a la institución como zona de circulación prohibida. Para cumplir con esta indicación se requiere la implementación de rondines y vigilancia permanentes.

Para el caso del sistema penitenciario venezolano, cuando se habla de seguridad y custodia, se hace referencia a todo el proceso de protección al interno, al personal, al control de todos los hechos que atentan contra la seguridad del interno y del Establecimiento Penitenciario, así como al control de las requisas.

### 1.4.1 Novedades

El término novedades dentro de un sistema penitenciario responde al control diario de todos los sucesos recientes, noticias o hechos ocurridos en los distintos Establecimientos Penitenciarios y su información a los órganos superiores. Esto se encuentra establecido en todas las entidades. La información diaria a realizar contempla entre otros los siguientes elementos:

- Situación de los internos (Cuantos deben de existir, presentes, hospitalizados, pase, etc.).
- Hechos de Contingencia ocurridos:
  - Agresiones.
  - Motines.
  - Huelgas.
  - Lesiones.
  - Fugas.

Ante estos hechos de contingencia debe existir en los distintos Establecimientos Penitenciarios la señal para la aplicación de los planes de contingencia, el cual estipula la participación y empleo de las fuerzas y medios de cada establecimiento en el restablecimiento de la situación.

Para dar solución al problema planteado se decide implementar el módulo Novedades dentro del subsistema de Seguridad y Custodia del SIGEP, como se muestra en la Figura 1.1.

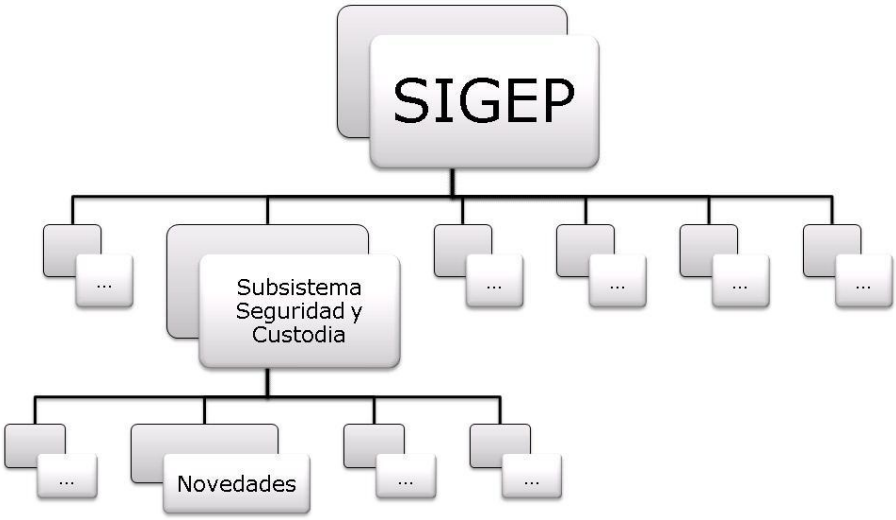


**Figura 1.1 Ubicación del módulo novedades dentro del SIGEP.**

El proceso gestionar novedades muestra todas las novedades existentes, además permite registrar las novedades según el tipo seleccionado debido a que todas estas tienen diferentes datos. Se pueden modificar, eliminar y observar algunos detalles así como filtrar por tipo de novedad y por fecha de registro.

**Proceso Registrar Novedad**

El registro de una novedad se encarga de recoger toda la información relacionada con las novedades que ocurren en el establecimiento penitenciario teniendo en cuenta el tipo de novedad seleccionada.

Los tipos de novedades posibles en el sistema penitenciario venezolano, según la legislación vigente son túneles, secuestros, heridos, motines, huelgas, recapturas, autosecuestros y muertos.

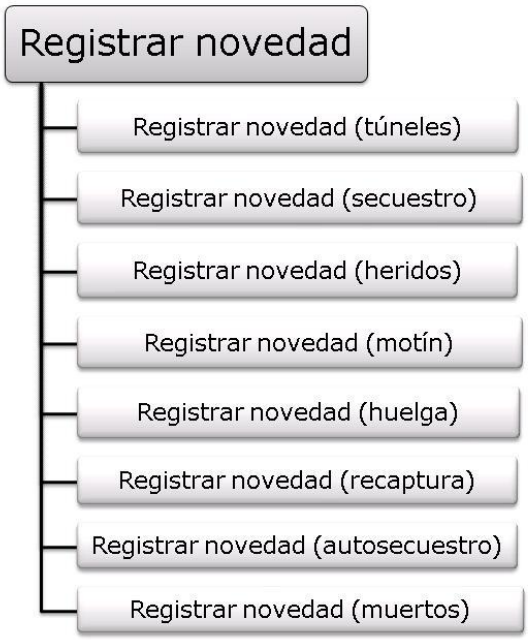


**Figura 1.2 Funcionalidad registrar novedad, según el tipo.**

**Proceso Modificar Novedad**

El proceso modificar novedad está dirigido a garantizar posibles cambios que se le pueda realizar a información ya existente de alguna de las novedades que han sido registradas en la aplicación.

**Proceso Consultar Novedad**

El proceso Consultar novedades garantiza la información de los detalles de cada una de las novedades que han sido registradas según el tipo de esta.

**Proceso Eliminar Novedad**

El proceso Eliminar novedades permite eliminar una o varias novedades que han sido seleccionadas.

## 1.5 Tecnologías y herramientas utilizadas en el desarrollo

Para desarrollar la solución de software relacionada con el proceso de gestionar novedades haciendo uso de la arquitectura y las tecnologías definidas por el proyecto, fue necesario el estudio de todas las herramientas utilizadas anteriormente en el SIGEP. A continuación serán descritas brevemente las

herramientas y tecnologías más significativas, como son la metodología de desarrollo, los patrones de diseño de software, el paradigma de programación, la plataforma de desarrollo, los frameworks, el gestor de base de datos, el entorno integrado de desarrollo y las bibliotecas.

### 1.5.1 Metodología de Desarrollo

El proyecto SIGEP desde sus inicios utilizó como metodología de desarrollo de software a Rational Unified Process (RUP) debido a:

- Ser un proyecto de alta complejidad.
- Contar con un equipo de trabajo que no superaba las 60 personas y tener poca experiencia.
- Ser conveniente garantizar la calidad del proceso desde sus inicios.
- Ganar en organización.

En este trabajo de diploma se adopta la metodología de desarrollo anteriormente predeterminada por el proyecto.

RUP es un proceso que se caracteriza por ser: (JACOBSON, 2004):

- **Dirigido por casos de uso.** Los casos de uso describen los requisitos funcionales del sistema desde la perspectiva del usuario y se usan para determinar el alcance de cada iteración y el contenido de trabajo de cada persona del equipo de desarrollo.
- **Centrado en la arquitectura**. La arquitectura permite ganar control sobre el proyecto para manejar su complejidad y controlar su integridad. Hace posible la reutilización a gran escala y provee una base para la gestión del proyecto.
- **Iterativo e incremental.** Se divide en 4 fases: Inicio, Elaboración, Construcción y Transición, y cada una de ellas se divide en iteraciones. En cada iteración se trabaja en un número de disciplinas haciendo énfasis en algunas de ellas. Algunas de las disciplinas propuestas por RUP son:
    - Modelado del negocio
    - Requisitos

- Análisis y Diseño
- Implementación
- Pruebas.

RUP propone flujos de trabajo en los que se definen las secuencias de actividades, quienes las deben desarrollar y los artefactos a generar.

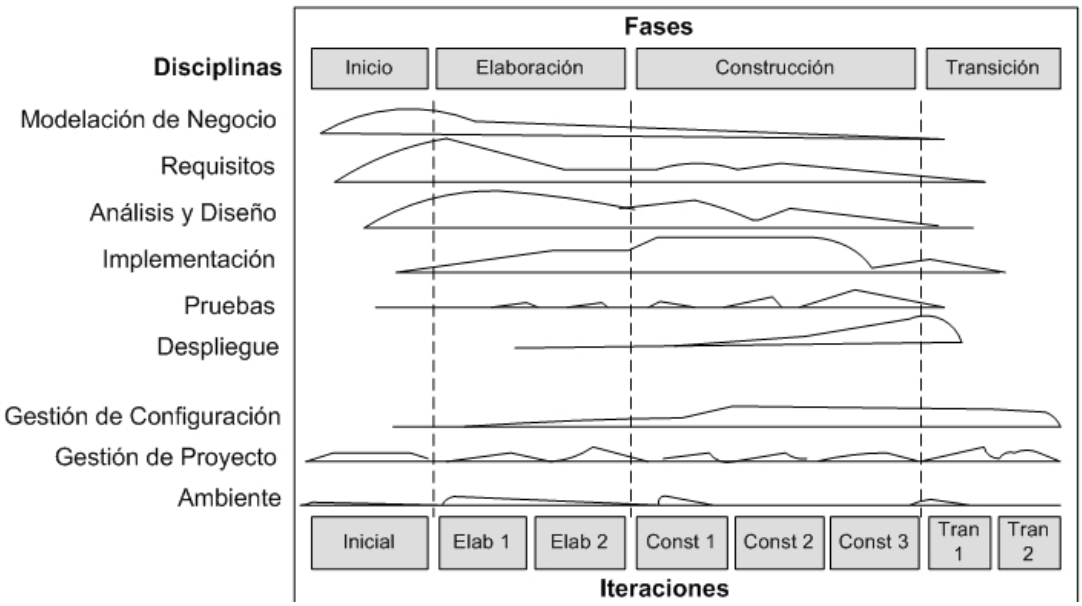


**Figura 1.3 Fases y flujos de trabajo de RUP.**

Utiliza el Lenguaje Unificado de Modelado (Unified Modeling Lenguaje, UML), el cual los autores han seleccionado para visualizar, especificar y construir los artefactos del sistema automatizado a definir.

### 1.5.2 Herramienta de modelado

Sitio web oficial: http://www.visual-paradigm.com/

La herramienta de modelado utilizada en el proyecto SIGEP fue Visual Paradigm. Esta herramienta de diseño soporta todos los diagramas UML, diagramas SysML y el diagrama de entidad-relación además produce documentación del sistema en formato PDF, HTML y formatos de MS Word. Los desarrolladores pueden diseñar documentación del sistema con una plantilla de diseño así como los analista de sistemas

pueden estimar las consecuencias de los cambios con los diagramas de análisis de impacto, tales como la matriz y el diagrama de análisis.

Visual Paradigm ofrece dos modalidades: el UML Edition y Smart Development Environment (SDE) para la integración con IDEs de desarrollo como el Eclipse, muy factibles en la realización de ingeniería directa e inversa y en la generación de la base de datos a partir de diagramas entidad relación y ORM, además de generar código Java.

### 1.5.3 Patrones de diseño de software

Los Patrones de Diseño son modelos de trabajo enfocados a dividir un problema en partes de modo que sea posible abordar cada una de ellas por separado para simplificar una solución. Un patrón de diseño es un conjunto de reglas que describen como afrontar tareas y solucionar problemas que surgen durante el desarrollo de software.

La descripción de un patrón de diseño de forma general esta dado por:

**Nombre:** frase que encierra la característica fundamental del patrón.

**Problema:** descripción del problema típico que el patrón resuelve.

**Solución:** especificación de cómo resolver el problema. Incluye la descripción de los elementos que componen el patrón y cómo se relacionan entre sí.

**Implicaciones:** define cuándo y cómo el patrón debe ser usado y los beneficios y problemas que implican su uso.

Los patrones de diseño más usados en la solución del problema son descritos a continuación según el libro Buenas Prácticas de Ángel Castiglia:

- **Data Access Object (DAO):** Centraliza todo el acceso a datos en una capa independiente, aislándolo del resto de la aplicación. Sus principales beneficios son que reduce la complejidad de los objetos de negocio al abstraerlos de la implementación real de la comunicación con la fuente de datos y que permite una migración más fácil de fuente de datos.
- **Facade (Fachada):** Simplifica el acceso a un conjunto de objetos proporcionando uno, llamado fachada, que los clientes pueden usar para comunicarse con el conjunto. Reduce el número de

objetos con los que tiene que interactuar un cliente proporcionando un menor acoplamiento y facilitando el cambio de componentes sin afectar a sus clientes.

- **Strategy (Estrategia):** Se utiliza cuando se necesitan diferentes variantes de un algoritmo o cuando una clase define muchos comportamientos los cuales se manifiestan como definiciones condicionales múltiples de sus operaciones. Este patrón propone encapsular los algoritmos en diferentes clases, eliminando las costosas definiciones de comportamiento multicondicionales. Brinda gran flexibilidad para futuras extensiones y tiene como desventaja, que puede producir una gran explosión en el número de objetos del sistema.
- **Controller (Controlador):** Este patrón propone asignar la responsabilidad de controlar el flujo de eventos de un sistema, a clases específicas llamadas controladores. Los controladores no ejecutan las tareas sino que las delegan en otras clases, con las que mantiene un modelo de alta cohesión.
- **Model-View-Controller (Modelo-Vista-Controlador, MVC):** Divide una aplicación en tres componentes, el modelo, las vistas y los controladores. El modelo contiene la funcionalidad y los datos del sistema, las vistas muestran información al usuario y los controladores manejan la interacción del usuario. Las vistas y controladores comprenden la interfaz de usuario. MVC permite crear diferentes vistas para el mismo modelo incluso en tiempo de ejecución.
- **Front-Controller (Controlador Frontal):** El patrón propone utilizar un controlador como el punto inicial de contacto para manejar las peticiones del usuario en una aplicación. El controlador maneja el control de peticiones, incluyendo la invocación de los servicios de seguridad como la autentificación y autorización, la elección de una vista apropiada, el manejo de errores, y el control de la selección de estrategias de creación de contenido.

### 1.5.4 Paradigma de programación.

Como paradigma de programación se utilizó la Programación Orientada a Objetos (POO). Para el estudio de la POO, se recurrió al artículo *Evolución de los lenguajes de programación ¿Por qué cambiarse a la Programación Orientada a Objetos?,* de Gerson Johan Samaniego Rodríguez y el libro *Introducción a la OOP*, del autor Francisco Morero, de ambas bibliografías se redactó un pequeño resumen con las caracteristicas principales de este paradigma.

La POO es una extensión de los lenguajes de alto nivel estructurados que trata de representar de una forma más sencilla el modelo del mundo real. Intenta resolver principalmente los problemas de la Ingeniería del Software como: portabilidad, reusabilidad, mantenibilidad, entre otros. Para ello se basa en características claves como el encapsulamiento, la herencia, el polimorfismo, y el desarrollo orientado primero hacia el qué, y luego hacia el cómo (interfaces).

El elemento principal de la POO es la Clase, la cual representa abstractamente un elemento del mundo real o una unidad que agrupa propiedades y funcionalidades comunes. Una instancia de una clase es un Objeto el cual tiene atributos y métodos. Los métodos modifican los atributos o prestan algún servicio a otros objetos.

### 1.5.5 Plataforma de desarrollo

La dirección del proyecto SIGEP decidió utilizar Java 2 Enterprise Edition (J2EE) como plataforma de desarrollo, por lo que para la conformación de este trabajo fue necesario estudiarla por medio de libros como *Programacion Java Server con J2EE Edition,* escrito por un colectivo importante de autores.

La plataforma de desarrollo J2EE, según la bibliografía utilizada, es una de las mejores soluciones para satisfacer las demandas en el mundo empresarial, ya que especifica la infraestructura para gestionar sus aplicaciones y los servicios API [1]para construir sus aplicaciones.

La plataforma J2EE es esencialmente un entorno distribuido aplicación-servidor, un entorno Java que ofrece:

- Un conjunto de varios API de extensión Java para construir aplicaciones. Estos API definen un modelo de programación para aplicaciones J2EE.
- Una infraestructura de periodo de ejecución para albergar y gestionar aplicaciones. Este es el periodo de ejecución en el que residen sus aplicaciones.

[1] API, del inglés *Application Programming Interface (Interfaz de programación de aplicaciones)*: Es el conjunto de funciones y procedimientos (o métodos si se refiere a programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

Las aplicaciones que puede desarrollar con estos dos elementos pueden ser programas para controlar páginas Web o componentes para implementar transacciones complejas de bases de datos, o incluso applets [2] de Java, todos ellos distribuidos por la red.

### 1.5.6 Frameworks

Los frameworks son un conjunto de componentes destinados a la construcción de un determinado tipo de aplicaciones de manera generalista, con interfaces bien definidas que interactúan entre sí para cumplir una tarea. La Gang of Four [3]define un framework como "un conjunto de clases que constituyen un diseño reutilizable para un tipo específico de aplicaciones".

En el SIGEP los frameworks que se utilizaron fueron Spring, para la capa de presentación y negocio, e Hibernate para la capa de acceso a datos. Para el aprendizaje de estos frameworks fueron muy útiles los libros Spring in Action, de los autores Craig Walls y Ryan Breidenbach, Hibernate in Action de Christian Bauer y Gavin King, e Introducción a Hirbernate, escrito por Francesc Rosés Albiol.

**Framework Spring**

Spring es un framework de código abierto, fue creado para abordar la complejidad del desarrollo de aplicaciones empresariales. Spring hace posible el uso de JavaBeans [4]para conseguir cosas que antes sólo eran posibles realizar con Enterprise JavaBeans. Sin embargo, la utilidad de Spring no se limita a un desarrollo del lado del servidor. Cualquier aplicación Java se puede beneficiar de Spring en términos de simplicidad, capacidad de prueba y articulación flexible.

---

[2] Applets: Programa escrito en lenguaje de programación Java que se pueden incluir en una página HTML.

[3] GoF, del inglés *Gang of Four*: Es el nombre con el que se conoce a los autores del libro *Design Patterns*, referencia en el campo del diseño orientado a objetos. La 'Banda de los cuatro' se compone de los autores: Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides.

[4] JavaBeans: Son un modelo de componentes creado por Sun Microsystems para la construcción de aplicaciones en Java. Definidos como "componentes de software reutilizables que se puedan manipular visualmente en una herramienta de construcción".

Spring promueve el bajo acoplamiento a través de una técnica conocida como inyección de dependencias (DI). Cuando esta inyección de dependencia se aplica, los objetos reciben pasivamente sus dependencias en lugar de crearlas o buscar los objetos dependientes por si mismos.

El framework Spring se compone de siete módulos bien definidos. Cuando se toma en su conjunto, estos módulos dan todo lo que necesita para desarrollar aplicaciones empresariales, lo que no implica que deban ser usados todos, sino los necesarios para cada aplicación.

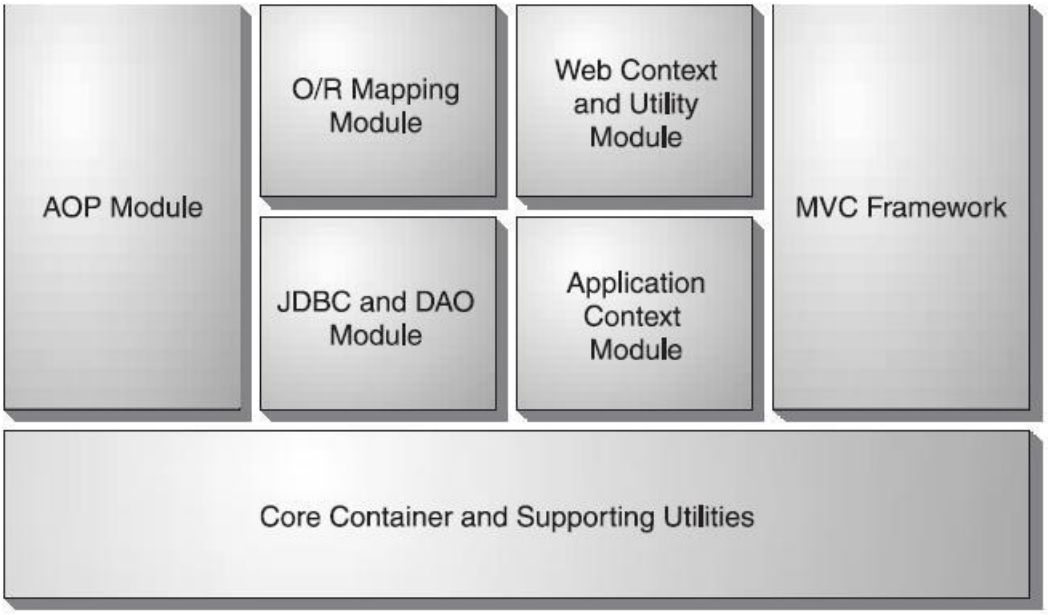


**Figura 1.4 Módulos de Spring Frameworks.**

La estructura de los módulos de Spring, tiene como base el Core que proporciona las funcionalidades fundamentales de Spring. Este módulo sirve de base al resto de los módulos, ya que sobre el se desarrollan los demás.

De estos módulos se utilizan para el desarrollo del SIGEP los siguientes:

- **Módulo de integración de mapeo Objeto-relacional (ORM[5]):** Spring proporciona el módulo de ORM para aquellos que prefieren usar un mapeo objeto-relacional, la herramienta directamente sobre JDBC. Spring no trata de aplicar su propia solución ORM, pero proporciona ganchos a varios frameworks ORM más conocidos, como Hibernate, Java Data Objects (JDO), e iBATIS SQL Maps. La gestión de transacciones de Spring soporta cada uno de estos marcos ORM, así como Java Database Connectivity (JDBC).
- **Módulo MVC de Spring:** Spring viene con un framework Modelo / Vista / Controlador (MVC) con todas la funciones para la construcción de aplicaciones web. Aunque Spring se puede integrar fácilmente con otros frameworks MVC, como Struts, Spring framework MVC hace uso de IoC [6]para establecer una separación clara de los objetos de negocio de la lógica de controlador.

**Framework Hibernate**

Hibernate es un entorno de trabajo que tiene como objetivo facilitar la persistencia de objetos Java en bases de datos relacionales y al mismo tiempo la consulta de estas bases de datos para obtener objetos.

Hibernate se encarga de enlazar el modelo relacional y el objetual de manera que trabaja desde Java como corresponde. Usando el modelo objetual. Además expresa las condiciones de las consultas objetualmente y se encarga de transformar una consulta al dialecto SQL que corresponda.

### 1.5.7 Gestor de Base de Datos

En el libro Diseño de Bases de Datos, de la autoría de la Licenciada Rosa María Mato García, se define y describe lo que constituye un gestor de bases de datos:

Un Gestor de Base de Datos es el software que permite la utilización y/o la actualización de los datos almacenados en una (o varias) base(s) de datos por uno o varios usuarios desde diferentes puntos de vista y a la vez, se denomina sistema de gestión de bases de datos (SGBD).

[5] ORM, del inglés *Object Relacional Mapping (Mapeo relacional de objetos)*: Persistencia automática y transparente de objetos de una aplicación en una base de datos relacional utilizando meta datos que describen la correspondencia entre el objeto y las tablas de la base de datos.

[6] Inversión de control (IoC): Conjunto de técnicas y patrones de diseño de software que invierte el control del flujo de un sistema. El control es invertido en comparación con el modelo tradicional de interacción expresado en una serie de llamadas consecutivas a procedimientos.

El objetivo fundamental de un SGBD consiste en suministrar al usuario las herramientas que le permitan manipular, en términos abstractos, los datos, o sea, de forma que no le sea necesario conocer el modo de almacenamiento de los datos en la computadora, ni el método de acceso empleado.

El gestor de base de datos a utilizar para el desarrollo y despliegue del SIGEP es Oracle, el cual proporciona un adecuado soporte para la réplica de los datos, necesario en el SIGEP, para la comunicación desde los servidores de los centros penitenciarios al servidor del Centro de Datos, que almacena toda la información del sistema a nivel nacional, en donde se utiliza Oracle Enterprise Edition v10.2.0.3 y en cada centro penitenciario Oracle Standard Edition 10.2.0.3 que es más ligero y la licencia es menos costosa.

Seguidamente se ofrecen más características de este gestor de bases de datos que permitieron definirlo como el más adecuado para el desarrollo del SIGEP, teniendo en cuenta además que este fue el gestor propuesto por el cliente del software.

**Oracle**

Es un sistema de gestión de base de datos relacional, 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 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.

Oracle como software privativo tiene la desventaja de los altos precios de sus licencias y del soporte técnico, además de los elevados requisitos de hardware que necesita, lo que lo convierte en un gestor típico de sistemas informáticos de grandes compañías o instituciones gubernamentales.

### 1.5.8 Entorno integrado de desarrollo

El entorno integrado de desarrollo (IDE[7]), que se escogió para desarrollar el proyecto SIGEP fue Eclipse, debido a las múltiples facilidades que esta herramienta brinda. Dichas facilidades son descritas en el documento *Introducción a la plataforma Eclipse* las que se mencionan a continuación.

**Eclipse**

Eclipse es una poderosa herramienta que permite integrar diferentes aplicaciones para construir un IDE, es un proyecto de desarrollo de software de código abierto, que está dividido en tres partes: el proyecto Eclipse Proyect, Eclipse Tools, y Eclipse Technology Proyect.

El Eclipse Proyect está subdividido a su vez en tres sub-proyectos que son la propia plataforma, Java Development Tool (JDT) y Plugin Development Enviroment (PDE).

Mediante Eclipse se puede crear diversas aplicaciones como son sitios Web, programas Java, C++ y Enterprise Java Beans donde su principal aplicación es JDT, herramienta para crear aplicaciones en Java.

Otras aplicaciones pueden ser integradas a eclipse en forma de plugins[8], que son reconocidos automáticamente por Eclipse al iniciar el mismo. A pesar de ser gratis y de código abierto, sus plugins pueden no serlo dadas las características de su licencia.

Algunos plugins que trae incluida la plataforma son: Ant, Compare, Core, CVS, Debug, Help, JDT, Jface, Releng, Scripting, Search, SWT, Text, UI, Update, Team, WebDAV.

### 1.5.9 Bibliotecas

En ciencias de la computación, una biblioteca es un conjunto de subprogramas utilizados para desarrollar software. Las bibliotecas contienen código y datos, que proporcionan servicios a programas

[7] IDE, del inglés *Integrated Development Environment*: Programa compuesto por un conjunto de herramientas para un programador. Entorno de programación que ha sido empaquetado como un programa de aplicación consistente en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica GUI.

[8] Plugin: Aplicación informática que interactúa con otra aplicación para aportarle una función o utilidad específica y es ejecutada por la aplicación principal.

independientes, es decir, pasan a formar parte de éstos. El SIGEP utilizó como bibliotecas JasperReports para la generación de reportes y Dojo toolkit para la programación en la capa de presentación.

Para el estudio de estas bibliotecas se utilizó como bibliografía *JasperReports for Java Developers*, escrito por David R. Heffelfinger, el trabajo de investigación científica *Reportes con JasperReport, IReport y JFreeChart en Aplicaciones Empresariales en Java*, elaborado por un conjunto de estudiantes pertenecientes al proyecto SIGEP, además de utilizar el trabajo de diploma *Desarrollo de una Librería de Componentes JavaScript*[9] *basado en Dojo Toolkit*, de la autoría de Lilian Teresa Castro Mecías y Rolando Bermúdez Peña, este último miembro del equipo de desarrollo del SIGEP. Seguidamente son descritas ambas bibliotecas.

**JasperReports**

JasperReports es una biblioteca de código abierto de clases de Java diseñado para ayudar a los desarrolladores con la tarea de añadir capacidades de reportes para las aplicaciones Java. Dado que no es una herramienta independiente, no puede ser instalado por su cuenta. Por el contrario, está incrustado en las aplicaciones de Java mediante la inclusión de esta biblioteca en el CLASSPATH de la aplicación.

JasperReports permite la generación de reportes en varios formatos como PDF, RTF, HTML, XML y XLS, a partir de una plantilla XML. Jasper Reports permite representar información diversa de distintas maneras, pudiéndose incluir en los reportes texto, tablas, imágenes y gráficos.

El SIGEP utiliza esta librería para brindar información estadística que apoye la toma de decisiones de la Dirección General de Custodia y Rehabilitación del Recluso y para la generación de documentos oficiales que emiten los centros penitenciarios.

**Dojo toolkit**

Dojo es una herramienta de código abierto DHTML (Dynamic HTML) escrito en JavaScript destinado a facilitar el rápido desarrollo de JavaScript o de las aplicaciones basadas en Ajax[10] y sitios web. Su idea es

[9] JavaScript: Lenguaje de Script para páginas Web, basado en la sintaxis de Java.

[10] Ajax del inglés ***A****synchronous* ***J****avaScript* ***A****nd* ***X****ML* (JavaScript asíncrono y XML): Es un framework de aplicación Javascript para crear aplicaciones de colaboración en tiempo real que se ejecutan en el navegador.

la de abstraer al desarrollador de las complejidades del DHTML y de las discrepancias existentes entre navegadores, que hacen que el código JavaScript a utilizar sea diferente. Dojo provee gran cantidad de funcionalidades las cuales lo dividen en tres grandes proyectos, Dojo Core, Dijit y DojoX.

Dijit y DojoX están basadas en la sólida base que Dojo Core brinda. Dijit incorpora un conjunto de widgets [11]con los cuales se pueden crear amigables interfaces en muy poco tiempo, y escribiendo casi o ningún código JavaScript. DojoX es donde se desarrollan extensiones para Dojo Toolkit, sirve de cuna para nuevas ideas y pruebas de funcionalidades adicionales para el Toolkit principal.

### 1.6 Conclusiones

Con el análisis desarrollado en este capítulo, después de haber descrito de forma general las características el proceso Novedades se puede llegar a la conclusión que es de vital importancia la informatización de este procesos para el Sistema Penitenciario venezolano, utilizando las tecnologías ya definidas por el proyecto en sus anteriores versiones, debido a que contienen tendencias recientes de la programación, como la Programación Orientada a Aspectos [12]y la inyección de dependencias y además los frameworks Spring como framework de aplicación e Hibernate como framework de soporte para la capa de acceso a datos que son considerados de referencia en su ámbito.

[11] Widget: Componente gráfico con el cual el usuario interactúa.

[12] Programación orientada a aspectos (AOP): Paradigma de programación cuya intención es permitir una adecuada modularización de las aplicaciones y posibilitar una mejor separación de conceptos.

# 2. ARQUITECTURA DEL SISTEMA

## 2.1 Introducción

En este capítulo se abordarán temas como las principales características del sistema y algunos conceptos e importancia de estos. Se describe la arquitectura que utiliza el SIGEP, sobre la cual se desarrolla el módulo Novedades, además de abordar un breve análisis de las capas por las que está compuesta. Se detalla el flujo de actividades a realizar para dar solución al diseño e implementación de un módulo en el SIGEP.

## 2.2 Arquitectura del SIGEP

La arquitectura de software esta definida como “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 2000).

La arquitectura del SIGEP se basó en Arquitectura Base sobre la Web (ArBaWeb) la cual orienta el diseño de las capas lógicas a través de una propuesta de diseño base; organiza la forma de codificar según las propuestas de convenciones o estándares de códigos y recursos; y brinda una estructura física para soportar el código, creando así un esqueleto base y además propone mecanismos de colaboración entre los componentes integrados en ella.

Además presenta una arquitectura de tres capas lógicas fundamentales: Capa de Presentación, Capa de Lógica de Negocio, y Capa de Acceso a Datos. Dichas capas están separadas lógica y estructuralmente y se encuentran bien delimitadas una de la otra.


**Figura 2.1 Arquitectura en tres capas.**

**Capa de Presentación**

Esta capa se limita a la navegabilidad y a gestionar todos aquellos aspectos relacionados con la lógica de presentación de la aplicación. En ArBaWeb esta capa descansa sobre una capa de servicios de negocio, por lo que no contiene lógica de negocio, sino simplemente lo concerniente a aspectos de presentación. Esta arquitectura permite que se pueda elegir más de una capa de presentación en la misma aplicación, sin impactar en las capas arquitectónicas inferiores, las cuales no deberían tener conocimiento sobre la capa superior.

La capa de presentación se encarga de cuestiones como:

- Navegabilidad del sistema, mapa de navegación, etc.
- Formateo de los datos de salida: Resolución del formato más adecuado para la presentación de resultados. Está relacionado directamente con la internacionalización de la aplicación
- Internacionalización: Los textos, etiquetas, y datos en general a presentar se obtendrán de uno u otro fichero de recursos en base al idioma preferido del navegador del usuario. En base a esta condición se ven afectadas las representaciones numéricas, las validaciones sobre los datos de entrada y otros aspectos relativos al idioma del usuario remoto.
- Validación de los datos de entrada, en cuanto a formatos, longitudes máximas, etc.
- Interfaz gráfica con el usuario.

Esta capa web es la responsable de tratar con las interacciones del usuario y obtener los datos que pueden ser mostrados en un formato determinado.

Normalmente está compuesta por tres tipos de objetos:

**Controladores:** Son responsables de procesar las entradas del usuario en forma de peticiones HTTP, invocando las funcionalidades necesarias, expuestas por la capa de servicios de negocio y devolviendo un modelo requerido para ser mostrado.

**Modelo:** Contienen los datos resultantes de la ejecución de la lógica de negocio, los cuales deberían ser mostrados en la respuesta.

**Vistas:** Son responsables de mostrar el modelo en la respuesta de la petición. La forma de mostrar el modelo podrá ser de diferentes tipos de vistas, por ejemplo, archivos JSP(Java Server Page), HTML, PDF, documentos de Excel, etcétera. Las vistas no son responsables de modificar los datos o incluso de obtener los datos; estas simplemente sirven para mostrar los datos del modelo que han sido suministrados por un controlador.

La combinación de estos objetos, es la aplicación el patrón arquitectónico llamado MVC (Model View Controller), que en el caso del SIGEP se aplica con el framework MVC de Spring.

**Capa de Lógica de Negocio**

En la capa de lógica de negocio es donde se implementan todas aquellas reglas obtenidas a partir del análisis funcional del proyecto. Esta capa es completamente independiente de cualquiera de los aspectos relacionados con la presentación de la aplicación y a los mecanismos de persistencia empleados en la capa de acceso a datos. Cuando la capa de negocio requiera recuperar o persistir entidades o cualquier conjunto de información, lo hará siempre apoyándose en los servicios que ofrezca la capa de acceso a datos para ello. De esta forma, la sustitución del motor de persistencia no afecta lo más mínimo a esta parte del sistema.

A pesar de usar la arquitectura que define ArBaWeb, los objetos de dominio pueden tener lógica de negocio. Sin embargo, la capa de servicios de negocio tiene un rol importante. En esta capa radican los objetos de negocio o Business Objects (DEEPAK ALUR 2003). Los objetos de negocio separan los datos y la lógica de negocio usando un modelo de objetos.

Estas son las funcionalidades que presenta esta capa:

**Lógica de negocio específica de procesos de negocios:** A veces es más oportuno para los objetos de dominio contener lógica de negocio aplicable a muchos casos de uso específicos. Sin embargo, existen casos de usos específicos que son realizados en la capa de servicios de negocios. Pero se propone que en esta definición de arquitectura los objetos de dominio no presenten ningún tipo de lógica de negocio, sino que esta responsabilidad recaiga sobre los objetos de negocio, permitiendo usar a los objetos de dominio como objetos de transferencia que se mueven entre las capas arquitectónicas de la aplicación.

**Puntos de entrada muy bien definidos para las operaciones de negocio implementadas:** Los objetos de negocio brindan las interfaces usadas por la capa de presentación.

**Control de transacciones:** Aunque a veces la lógica de negocio pueda ser movida a objetos de dominio, el control de transacciones no debería. Sino que las políticas transaccionales de la aplicación deberían ser planteadas al nivel de los objetos de negocio.

**Ejecución de restricciones de seguridad:** Las restricciones de seguridad en esta capa es en los puntos de entradas a la capa media, es decir en los objetos de negocio.

**Capa de Acceso a Datos**

La capa de acceso a datos es la encargada de persistir las entidades que se manejan en el negocio, el acceso a los datos almacenados, la actualización, entre otros, aunque puede ofrecer servicios relacionados con la persistencia o recuperación de información más complejos. La capa de acceso a datos encapsula toda la lógica de almacenamiento, independizando al resto del sistema del mecanismo de persistencia.

Los objetos de acceso a datos (DAO) 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.

Las interfaces de los DAOs contienen básicamente los siguientes tipos de métodos:

**Métodos para descubrir:** Localizan los objetos almacenados para ser usados por la capa de servicios de negocio.

**Métodos para persistir o salvar:** Hacen persistentes a los objetos transitorios.

**Métodos para eliminar:** Eliminan a los objetos guardados en el medio de almacenamiento (generalmente una base de datos).

**Métodos para conteos y otras funciones agregadas:** Devuelven los resultados de operaciones que son más eficientes implementarlas usando funcionalidades de la base de datos (procedimientos almacenados, etcétera.) que iterar sobre los mismos objetos.

## 2.3 Actividades en el diseño e implementación de un módulo del SIGEP

### 2.3.1 Diseño del dominio

Usamos modelo de dominio cuando determinamos que el flujo de información es difuso, dudoso, inexacto, con múltiples orígenes o solo contiene eventos o sucesos, cuando existe la imposibilidad de determinar subsistemas por el exceso de interconexiones, cuando hay un solapamiento de responsabilidades y además es difícil establecer reglas de funcionamiento.

En un modelo de dominio se captura los tipos más importantes de objetos que existen o los eventos que suceden en el entorno donde estará el sistema. El diseño del dominio constituye una entrada principal a las restantes actividades de diseño. Además se identifican los nomencladores y las clases del dominio se definen en el paquete *domain* del módulo.

La definición del dominio, en especial de las entidades persistentes sirve como una primera aproximación al diseño definitivo del modelo de datos.

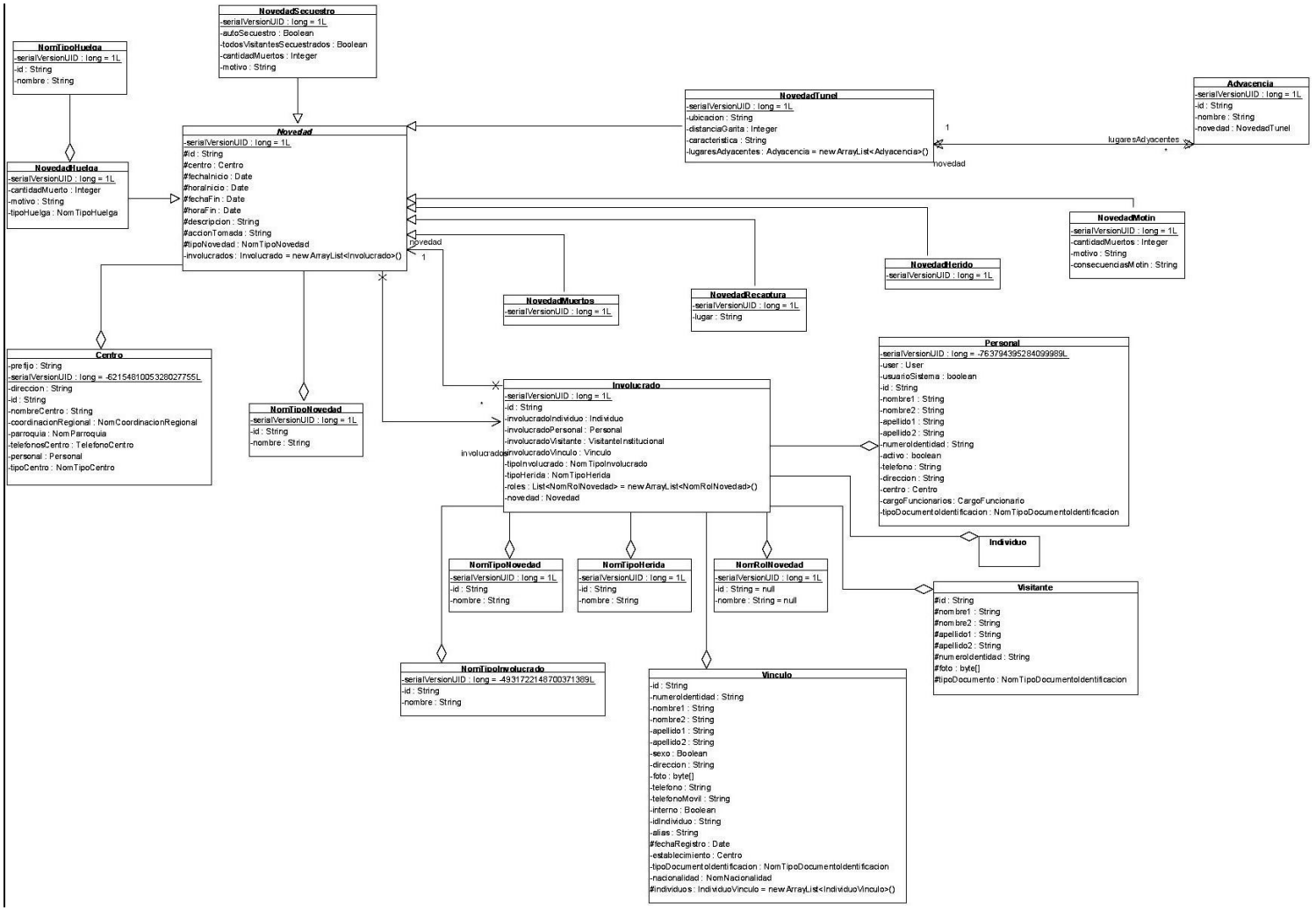


**Figura 2.2 Diagrama de clases del dominio.**

### 2.3.2 Diseño del Modelo de Datos

El diseño de la base de datos debe almacenar información y permitir a los usuarios recuperarla y actualizarla en base a sus peticiones. La utilización de estos objetos de la base de datos es analizada detenidamente y se decide siempre que implique mayor rendimiento en el acceso a los datos y un acoplamiento mínimo al gestor de base de datos.

En esta actividad obtendremos como resultado el modelo de datos de cada módulo. Esta estructura tiene que garantizar que el almacenamiento y recuperación de la información ocurra de manera adecuada y que se cumplan las restricciones identificadas, a través de la integridad referencial.

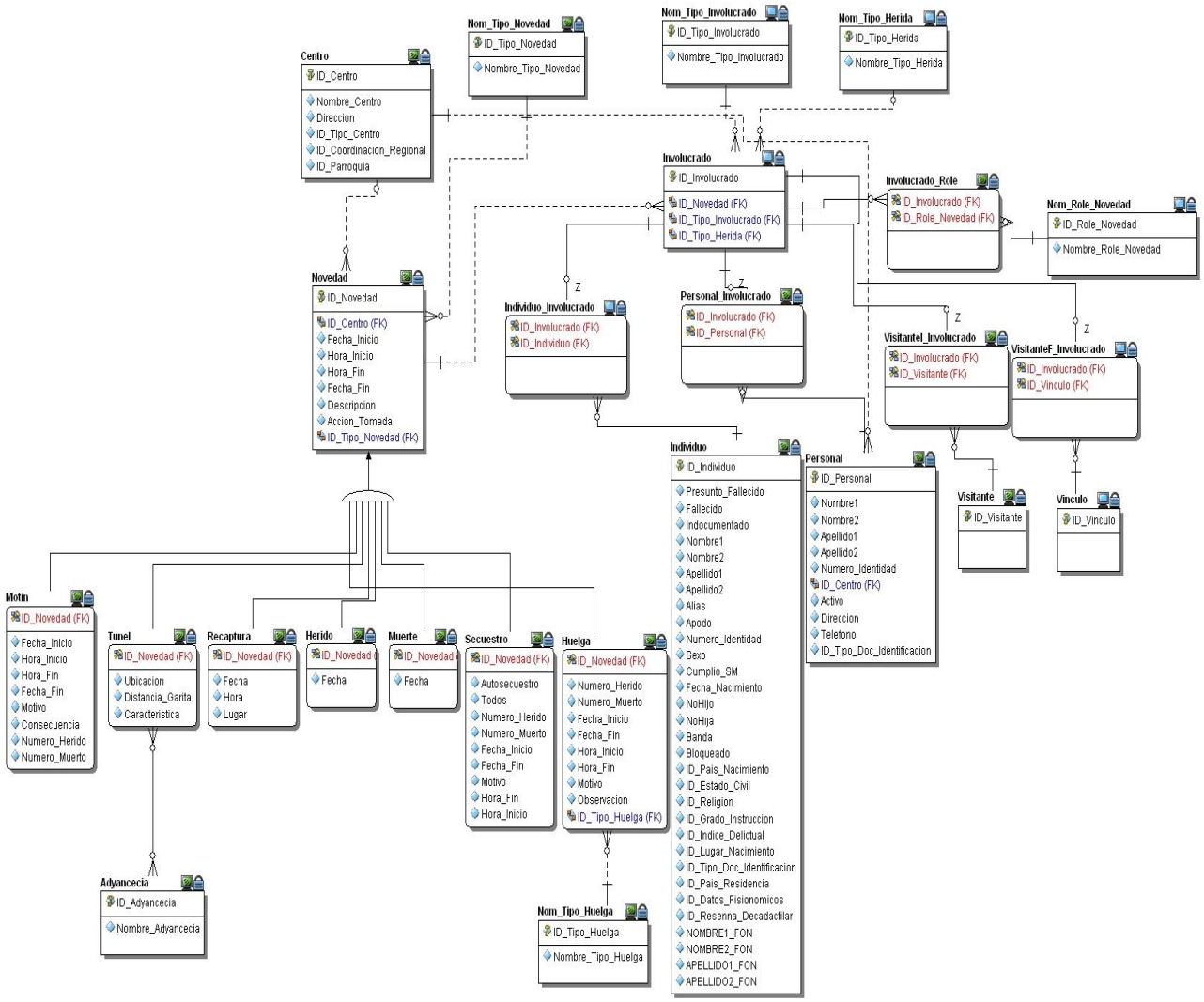


**Figura 2.3 Diagrama del modelo de datos.**

### 2.3.3 Diseño de la arquitectura en tres capas

En el diseño de la capa de presentación se definen las vistas y el flujo de navegación, por consiguiente las peticiones del usuario y los controladores encargados de atenderlas. Para el diseño de los controladores generalmente es necesario diseñar otros componentes que cumplen funciones en el flujo de Spring-MVC como son validadores (*validator*), objetos de respaldo (*command*[13]) y *property editors*. Luego se definen componentes del cliente, como son clases JavaScript, imágenes, entre otros.

El diseño de la capa de negocio contiene las clases necesarias para cubrir las funcionalidades definidas para el módulo, las cuales son expuestas a la capa de presentación mediante la fachada, que es la responsable de conocer a los verdaderos objetos de negocio, los managers. Los objetos de negocio se definen de acuerdo al agrupamiento lógico de funcionalidades que respondan a procesos de negocio. En esta capa se definen los eventos que pueden ser lanzados o escuchados, las excepciones a lanzar y la manera en que van a ser tratadas y mostradas al usuario.

Muy ligado al diseño de la capa de negocio está el diseño de la capa de acceso a datos, debido a que las funcionalidades de esta capa surgen como necesidades de la capa de negocio. Ambas capas se comunican mediante las interfaces de la capa de acceso a datos, las cuales definen las funcionalidades necesarias relacionadas con la persistencia y recuperación de datos del medio de almacenamiento. El diseño de esta capa requiere el mínimo acople posible al gestor de base de datos, sin desaprovechar las potenciales que ofrece Oracle. Este desacoplamiento de la aplicación y la base de datos, es posible con el uso del framework ORM Hibernate.

En la figura 2.4 se muestra un diagrama de clases donde se evidencia la interacción de las tres capas.

[13] Command: Clases utilizadas para guardar datos procedentes de las vistas, utilizando el patrón Command.

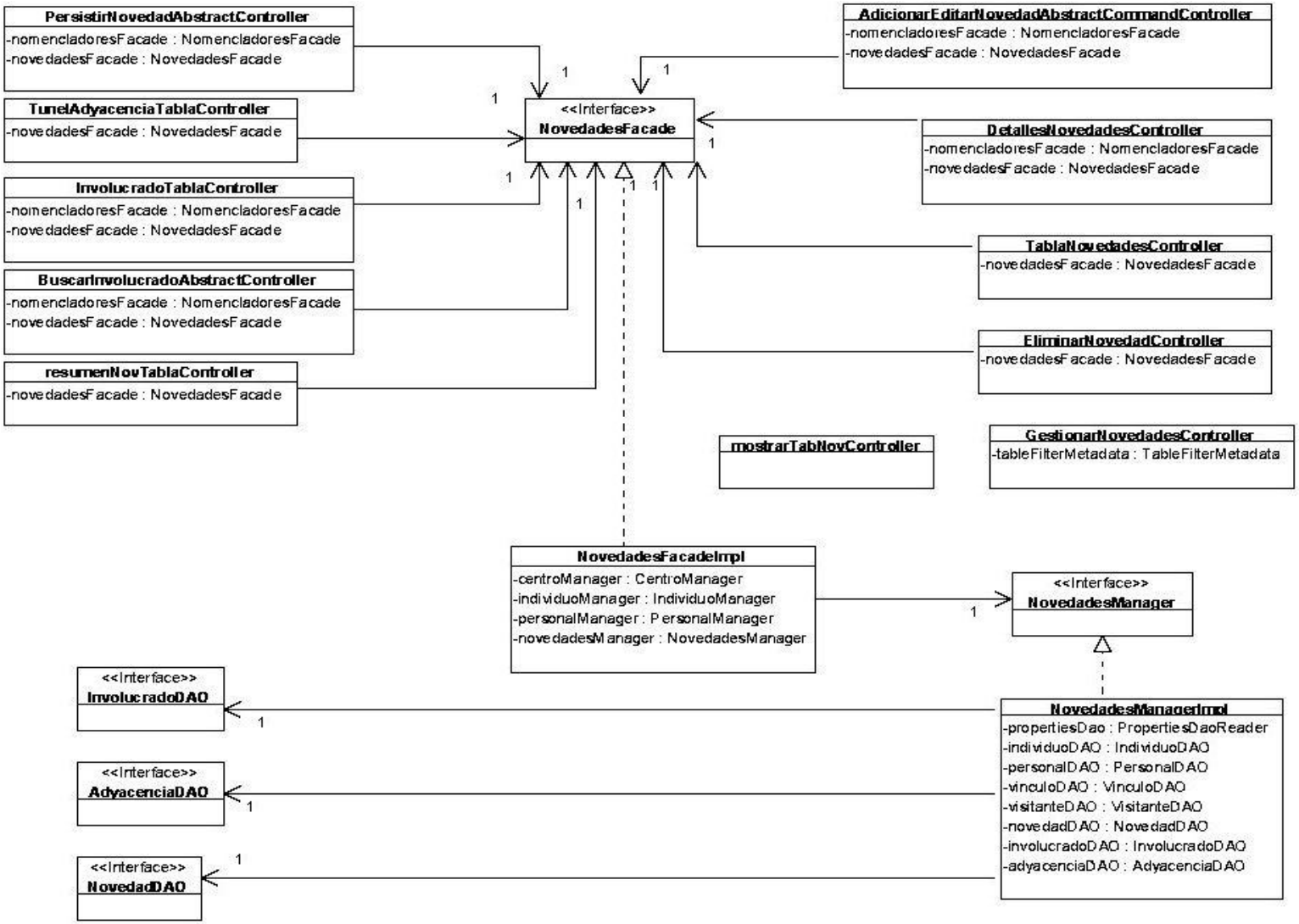


**Figura 2.4 Relaciones de las clases de las diferentes capas.**

### 2.3.4 Implementación de las entidades del dominio

Las entidades del dominio por lo general tienen muy poco comportamiento, en ellas solo se implementan los métodos más usados comúnmente por la mayoría de las entidades, como son equals(), hasCode() y toString().

En la implementación de las entidades del dominio se deben definir los atributos de cada entidad, para que queden listas en un buen por ciento para implementaciones reusables.

### 2.3.5 Implementación de las interfaces de los managers

Las interfaces que representan un conjunto de funcionalidades de la lógica de negocio de un módulo se denominan *managers*.

Se implementan los métodos definidos para cada una de las interfaces de los *managers*, ajustándose a las funcionalidades previstas. Cada método que se implemente tiene que verificar la integridad de los datos e informar a la capa de interfaz cualquier eventualidad a través de las excepciones definidas. Si fuese necesario se implementa la publicación y manipulación de eventos.

Los *managers* implementados son configurados en el fichero de configuración del contexto de Spring de la capa de negocio del módulo el que corresponden. Este fichero se encuentra en el paquete *configuration* y tiene por nombre: ***sigep-[subsistema]-[módulo]-business-context.xml***.

#### 2.3.5.1 Pruebas unitarias a los managers

Por cada *manager* se debe realizar una prueba de caja blanca para verificar cada método implementado. Estas pruebas se benefician de la técnica de inyección de dependencias de Spring y se basan en el framework de pruebas *JUnit*[14].

En el SIGEP, las clases de prueba de los *managers* terminarán con el nombre de la interfaz a la que prueban seguidos de “Test”, estas se definen en el paquete ***manager.impl.test*** del módulo al que pertenecen. Las mismas deben definir la dirección física de los ficheros de configuración del contexto de

[14] JUnit: Framework utilizado para realizar pruebas unitarias de aplicaciones Java. Creado por Erich Gamma y Kent Beck .Constituye un estándar para pruebas en la plataforma Java y de referencia para otras.

Spring donde se encuentra ubicado el objeto que están probando y sus dependencias, ya sean implementaciones falsas o reales, por lo que estas clases deben heredar de la clase ***org.springframework.test.AbstractDependencyInjectionSpringContextTests*** de Spring.

### 2.3.6 Implementación de la interfaz de la fachada

La interfaz de la *fachada* no tiene ninguna lógica de negocio, solo sabe que clase es responsable da cada funcionalidad, es decir, donde radica la información. Aunque las peticiones las desempeñan las otras clases, la *fachada* sirve de intermediaria para simplificar el acceso de la capa de presentación a la capa de negocio, reduciendo el número de objetos con los que tiene que interactuar esta.

En el SIGEP las interfaces que representan las fachadas que agrupan las funcionalidades del negocio de los módulos (se basa en el patrón de diseño Facade) terminarán con la palabra “Facade”. Las implementaciones de estas fachadas comenzarán con el nombre de la interfaz correspondiente y terminaran con la palabra “Impl”.

Para la implementación de la interfaz de la *fachada* solo tiene que conocerse en cuales *managers* se encuentran implementadas las funcionalidades necesarias para comunicarlas a la capa de interfaz.

### 2.3.7 Implementación de la capa de acceso a datos

En la implementación de la capa de acceso a datos se crean los ficheros de mapeo de Hibernate (hbm.xml) y se implementan los objetos de acceso a datos (DAO), por lo cual es necesario utilizar el plug-in de Eclipse Hibernate Tools, herramienta que aumenta considerablemente la productividad en esta actividad. Los ficheros *hbm* se colocan en el paquete ***dao.impl.map***.

En el SIGEP las interfaces que representan las operaciones sobre los objetos de acceso a datos, correspondientes al patrón de diseño Data Access Object terminan con la palabra “DAO”. Las implementaciones reales de estas interfaces comienzan con el nombre de la interfaz correspondiente y terminan con la palabra “Impl” y son configurados en el fichero ***sigep-[subsistema]-[módulo]-dataaccesscontext.xml***, donde se inyecta el sessionFactory [15]de Hibernate.

[15] Session Factory: objeto que configura al framework Hibernate a partir de los ficheros de mapeo (hbm.xml) y el dialecto a utilizar, según el gestor de base de datos.

Dichas implementaciones se centran la utilización de los APIs *Criteria* y *Example* del Framework Hibernate. Estos APIs son una poderosa herramienta para la creación de complejas consultas de forma relativamente fácil pero, en caso de que no satisfagan las necesidades del implementador, se usa el HQL (Lenguaje de consultas de Hibernate), que permite mayor flexibilidad y, en último caso, SQL. La utilización de las APIs mencionadas y del HQL permite escribir código más sencillo, transparente y tolerante al cambio.

Las implementaciones falsas de las interfaces DAO, sustituyen a las reales en caso de que aún no estén implementadas, permitiendo simular respuestas a peticiones que se hagan desde capas superiores. Los *mocks,* como se les llama a las implementaciones falsas, comienzan con el nombre de la interfaz correspondiente y terminan con la palabra “Mock”.

#### 2.3.7.1 Pruebas unitarias a los *DAOs*

Las pruebas unitarias que se le aplican a los *DAOs* son similares a las pruebas efectuadas a los *managers.* En el SIGEP estas clases de prueba de los DAOs comienzan con el nombre de la interfaz correspondiente y terminan con la palabra “Test” y se definen en el paquete ***dao.impl.test*** del módulo al que pertenecen, además heredan de una clase de Spring, ***org.springframework.test.AbstractTransactionalDataSourceSpringContextTests***. Al heredar de esta clase, los casos de prueba se ejecutarán en un contexto de transacciones a las que se puede hacer *rollback*, o *commit* en dependencia de los intereses de cada prueba. De esta forma se pueden probar las funcionalidades de acceso a datos sin afectar los datos en la base de datos.

### 2.3.8 Implementación de los controladores

Las clases controladoras manejan el flujo web de la aplicación, estas implementan directa o indirectamente ***org.springframework.web.servlet.mvc.Controller*** para insertarse en el flujo de Spring-MVC. Se encargan de la comunicación con la capa de negocio a través de su fachada, validan los datos de entrada de la aplicación y formatean lo datos de salida, mostrando las vistas correspondientes.

En el SIGEP los controladores se configuran en el fichero del contexto de Spring correspondiente a la capa de presentación: ***sigep-[subsistema]-[módulo]-servlet.xml*** y su nombre debe terminar con la palabra “Controller”.

### 2.3.9 Implementación de las páginas JSP

Las páginas JSP construyen documentos HTML con los datos que reciben de las clases controladoras asociadas a ellas. Para programar las JSP se utilizan principalmente las etiquetas de Spring y la biblioteca de etiquetas JSTL (Java Standard Tag Library). En el SIGEP el diseño gráfico de estas se realiza mediante el uso de estilos que radican en un fichero *CSS* [16]definido para el proyecto, donde aparecen todos los estilos de la aplicación.

Las páginas JSP se colocan en la carpeta ***/WEB-INF/jsps/[subsistema]/[modulo]/***, los nombres asignados a estos ficheros deben de alguna manera clara y breve relacionarse o describir el fin para el cual han sido destinadas.

### 2.3.10 Implementación de la lógica en el cliente

Para la implementación de la lógica en el cliente se utiliza el lenguaje JavaScript, con el cual se programan las validaciones en el cliente, así como el comportamiento de componentes gráficos como calendarios, tablas, botones y pantallas emergentes de error o información al usuario, además de lanzar peticiones al servidor usando AJAX y JSON-RPC.

Se debe crear un archivo .js para cada JSP que necesite de código JavaScript. En ese archivo se deberán programar todas las funciones y objetos que se necesite en la JSP. Estos archivo JS (JavaScript) deben ser ubicados en la carpeta ***WebContent/js/scripts/[nombre_del_módulo]***.

En el SIGEP se utilizan los componentes de la biblioteca JavaScript de componentes gráficos y DojoToolkit (Dojo 0.42) los cuales se localizan en el módulo *widget* de Dojo.

## 2.4 Conclusiones

De manera general en este capítulo se realizó un análisis de la arquitectura del SIGEP que contribuye a una mejor comprensión de la solución desarrollada. En cada actividad del flujo se ejemplificó el uso de los frameworks y tecnologías mencionados en el capítulo 1 y la manera de integrarse para lograr el diseño y la implementación del módulo Novedades.

---

[16] CSS, del inglés *Cascading Style Sheets (Hojas de estilo en cascadas)*: Lenguaje formal utilizado para definir la presentación de un documento estructurado escrito en HTML o XML.

# 3. DISEÑO E IMPLEMENTACIÓN DEL SISTEMA

## 3.1 Introducción

En este capítulo se le da solución al módulo Novedades de acuerdo a su diseño e implementación. Se utiliza como lógica de presentación en el capítulo la secuencia de actividades realizadas, en correspondencia con las definidas para la realización de un módulo dentro del SIGEP representando solamente una muestra de las funcionalidades distintivas de este módulo.

## 3.2 Análisis de las funcionalidades

Desde el punto de vista de las funcionalidades descritas para el modulo Novedades en la documentación generada por la captura de requisitos resultó que las funcionalidades a implementar son:

- Registrar novedades según el tipo.
- Eliminar novedades.
- Modificar novedad.
- Detalles de novedades.

Para facilitar el trabajo de los usuarios de la aplicación, se decide utilizar asistentes que guíen paso a paso en las funcionalidades donde se ejecutan registros.

## 3.3 Diseño del dominio

El modelo de dominio del módulo Novedades que se muestra en la Figura 3.1 representa el diagrama del dominio, la descripción de las entidades y los nomencladores más significativos.

El módulo Novedades tiene como principal elemento los tipos de novedades que se pueden registrar en el sistema. Además, aporta nomencladores útiles para la programación de la interfaz como NomRolNovedad que contiene los posibles comportamientos de un involucrado dentro de una novedad.

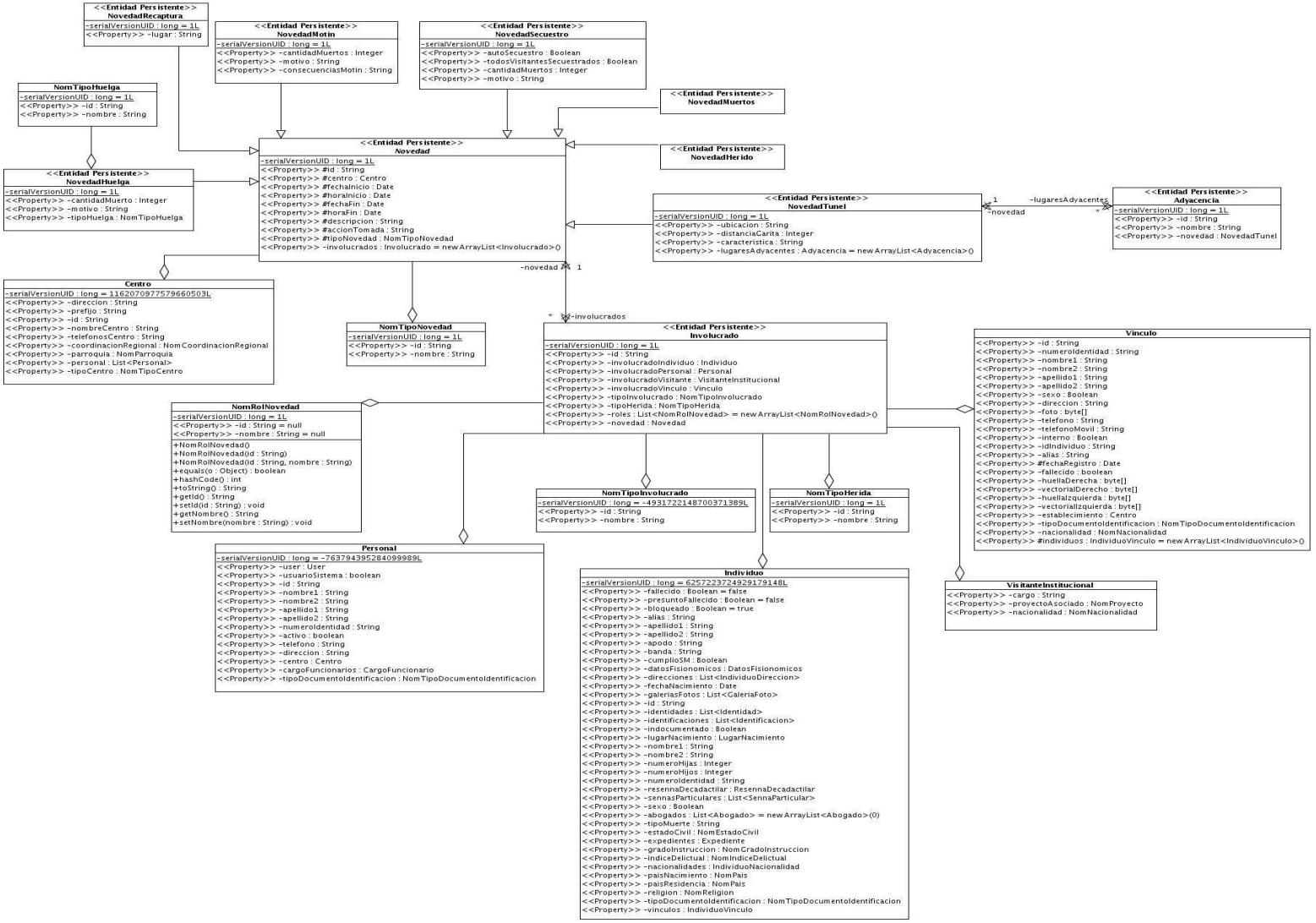


**Figura 3.1 Diagrama de clases persistentes del dominio.**

### 3.4 Diseño del modelo de datos

El modelo de datos del módulo Novedades se representa la figura 2.3. El modelo relativo al módulo Novedades se refleja en la Figura 2.5 “Diagrama entidad relación del módulo Novedades”.

En el modelo de datos de Novedades se observa la representación de las entidades persistentes que responden al registro de una novedad en un centro penitenciario. Para que este registro sea efectivo se debe tener en cuenta el tipo de novedad a la que hacemos referencia y los datos del individuo que se encuentra involucrado en la misma. Para ilustrar la validación a ejecutar se tomara como referencia una novedad de tipo Huelga (clase NovedadHuelga). Esta novedad puede ser realizada por cualquier tipo de individuo que esté involucrado en la misma (clase NomTipoInvolucrado), donde este involucrado puede tener diferentes roles debido a su comportamiento ante esta novedad (clase NomRolNovedad).

Así el módulo Novedades aporta al modelo de datos del SIGEP toda la jerarquía de novedades, las tablas que nomenclan por cada tipo de novedad y los tipos de involucrados que tiene relación con ella.

### 3.5 Diseño e implementación

En las figuras 3.2, 3.3 y 3.4 se muestran dos diagramas de secuencia y un diagrama de clases de estereotipos web respectivamente, correspondientes a los casos de uso Registrar y Modificar Novedades, los que constituyen los casos de uso más significativos del módulo Novedades.

Para explicar el flujo de información, según la arquitectura definida por el SIGEP y descrita en el capítulo anterior, se escogió como ejemplo este caso de uso, haciendo referencia al diseño de las clases a implementar, la descripción de las mismas, así como su interrelación a través de métodos específicos.

El diagrama de secuencia representa la interacción de un conjunto de objetos en una aplicación a través del tiempo y se modela para cada método de la clase. Para una mejor comprensión del caso de uso Modificar Novedad, este fue expresado en dos diagramas de secuencia, uno que comprende el flujo hasta que se muestran los datos de la novedad a modificar (Figura 3.2) y otro que parte de la aceptación por parte del usuario de los cambios realizados y se procede a persistir la novedad. Este último es el mismo para el caso de uso Registrar Novedad una vez que se van a persistir los datos de la novedad a registrar (Figura 3.3).

El diagrama de clases de estereotipos web permite representar la interacción existente entre los componentes de la capa de presentación, incluyendo páginas servidoras y clientes, formularios, archivos java script, clases, entre otros. En la figura 3.4 se detalla todo el flujo de información en la capa de presentación, según la arquitectura del SIGEP y utilizando el patrón Modelo-Vista-Controlador, del caso de uso seleccionado.

En la figura 3.2 y 3.3 aparecen las clases principales que intervienen en la funcionalidad registrar o modificar una novedad desde la capa de presentación partiendo de la clase controladora hasta la capa de acceso a datos.

En los Anexos 1 y 2 aparecen los diagramas de secuencia y diagramas de clases de estereotipos web de los restantes casos de uso del módulo Novedades.

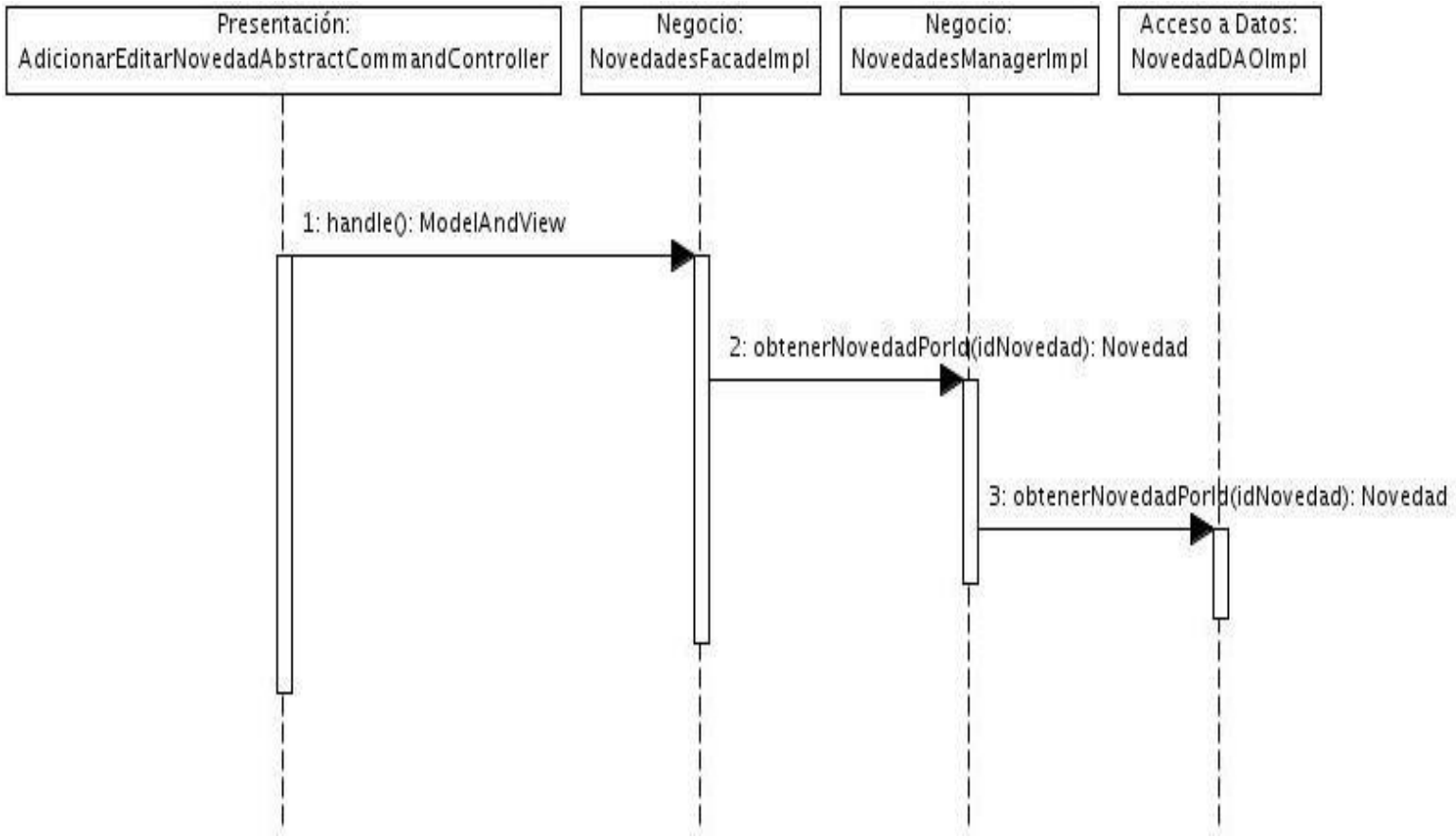


**Figura 3.2 Diagrama de secuencia del caso de uso Modificar Novedad.**

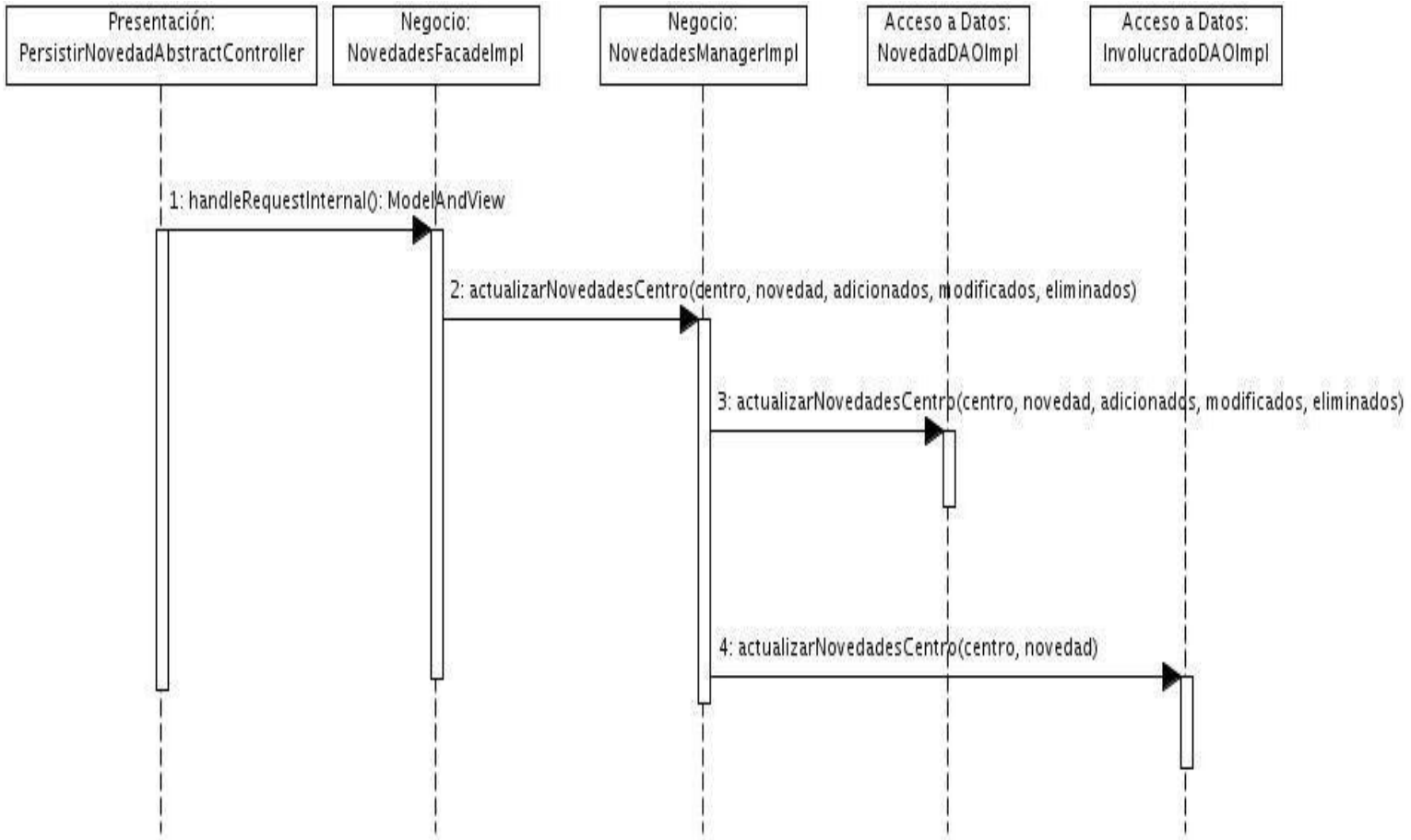


**Figura 3.3 Diagrama de secuencia de los casos de uso Registrar y Modificar Novedad**

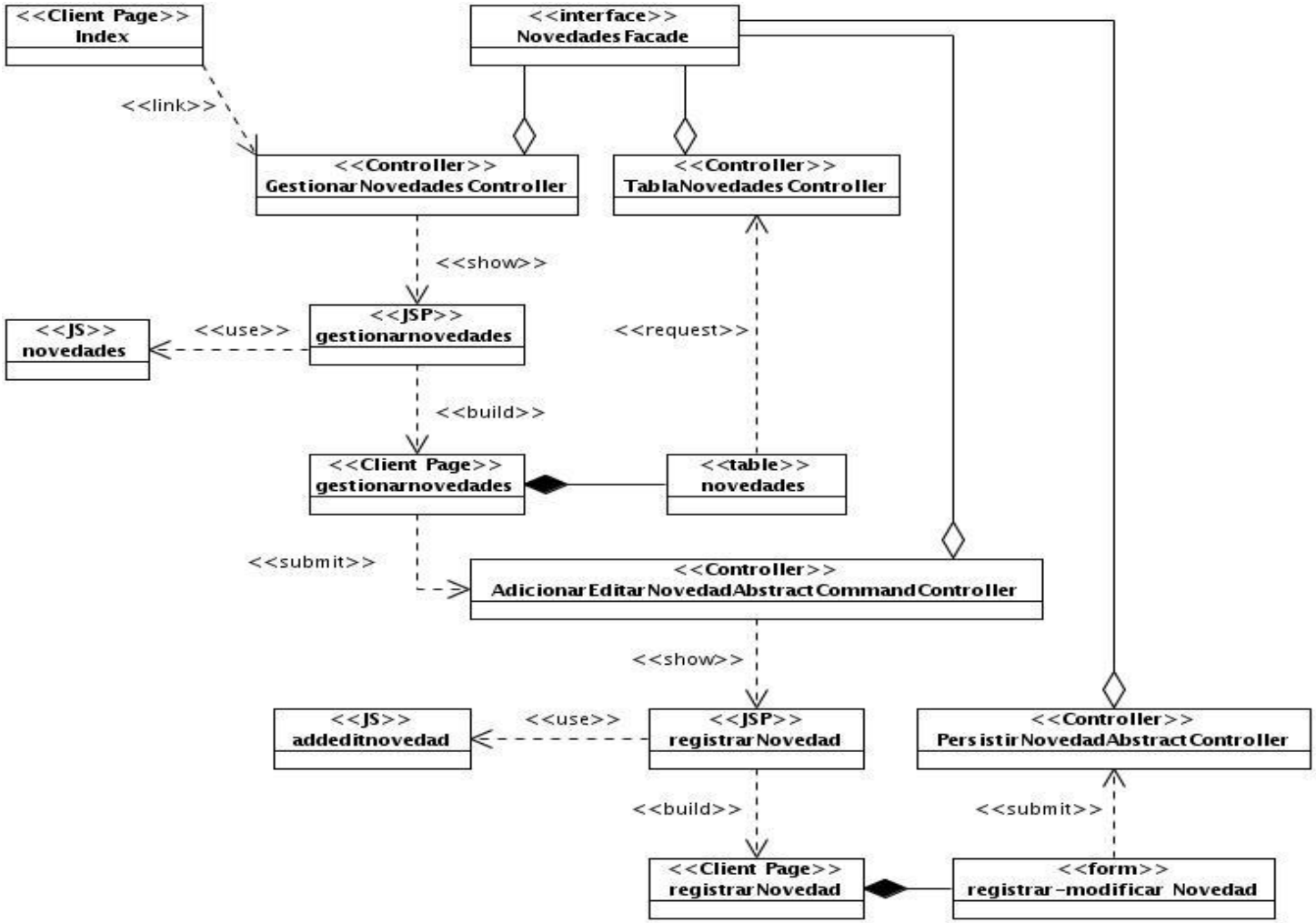


**Figura 3.4 Diagrama de clases de estereotipos WEB de los casos de uso Registrar y Modificar Novedad.**

### 3.6 Descripción de las Clases e Interfaces.

*NovedadesFacade:* Es una interfaz que contiene las funcionalidades particulares del móduloNovedades, que sirven de fachada a la Capa de Presentación.

*NovedadesFacadeImpl:* Esta clase implementa la interfaz NovedadesFacade. En ella no se implementa ninguna lógica de negocio, simplemente se limita a delegar las responsabilidades a la clase Manager correspondiente.

*NovedadesManager:* Es una interfaz que contiene las funcionalidades relacionadas con una novedad realizada en un centro penitenciario.

*NovedadesManagerImpl:* Esta clase implementa la interfaz *NovedadesManager.* Se encarga de implementar algunas funcionalidades para obtener la cantidad de novedades de un penado, mostrar una lista con todas las novedades realizadas hasta ese momento entre otras.

*NovedadDAO:* Interfaz que contiene las funcionalidades específicas relacionadas con la entidad Novedad dentro de la Capa de accesos a Datos.

*NovedadDAOImpl:* Esta clase implementa la interfaz NovedadDAO. Se encarga de implementar las funcionalidades brindadas tales como obtener un listado de novedades según el tipo de centro, contar la cantidad de novedades tipo túnel, entre otras.

*InvolucradoDAO:* Interfaz que contiene las funcionalidades específicas relacionadas con la entidad Involucrado dentro de la Capa de accesos a Datos.

*InvolucradoDAOImpl:* Esta clase implementa la interfaz *InvolucradoDAO*. Se encarga de implementar las funcionalidades brindadas tales como obtener un listado y contar la cantidad de novedades según el individuo que la ejecute.

*GestionarNovedadesController:* Clase controladora responsable de procesar los datos de una novedad gestionada por el usuario y mostrar el modelo en la respuesta de la petición.

*PersistirNovedadAbstractController:* Esta clase controladora es la encargada de validar todos los datos que suministra el usuario sobre una novedad invocando las funcionalidades necesarias para persistirla.

*AdicionarEditarNovedadAbstractCommandController:* Esta clase controladora es responsable de devolver los datos necesarios para adicionar o editar una novedad.

*gestionarNovedades.jsp:* Este archivo es el encargado de construir la vista gestionar novedades, que muestra la tabla con todas las novedades.

*registrarNovedad.jsp:* Este archivo se encarga de construir la vista registrar-modificar novedades, que muestra el formulario con los campos para registrar o modificar una novedad.

*NovedadCommand:* Esta clase es utilizada para guardar datos procedentes de la vista *registrarNovedad.jsp*, utilizando el patrón Command definido en los patrones GoF.

### 3.7 Aspectos relevantes del flujo básico Registrar-Modificar Novedad.

Las funcionalidades registrar y modificar son muy similares, por lo que utilizan las mismas clases y la mayoría de los métodos también coinciden, solo varía la forma de mostrar la página, que en caso de registrar, el formulario mostrado está vacío como se observa en la figura 3.5, y si es modificar, la vista muestra los datos de la novedad seleccionada con posibilidad de cambiarlos lo que se evidencia en la figura 3.6. Una vez llenos los campos tanto para registrar como modificar, se acepta y se persisten los datos. De modo general, así se desarrolla este flujo, a continuación se detallará parte de este proceso destacando lo más novedoso por cada uno de las componentes que intervienen en el mismo en las capas establecidas en la arquitectura del SIGEP.

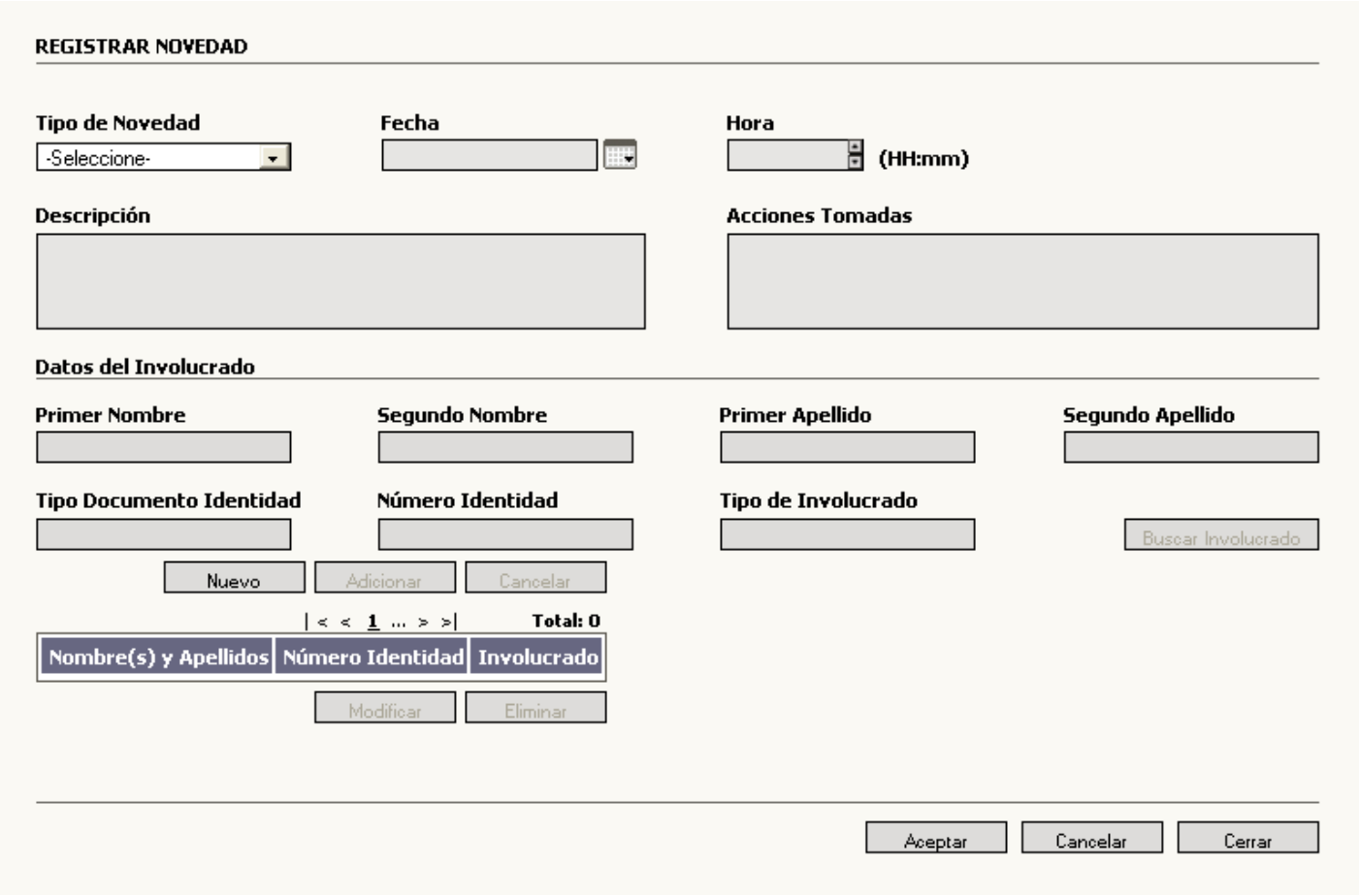


**Figura 3.5 Interfaz gráfica de Registrar Novedad.**

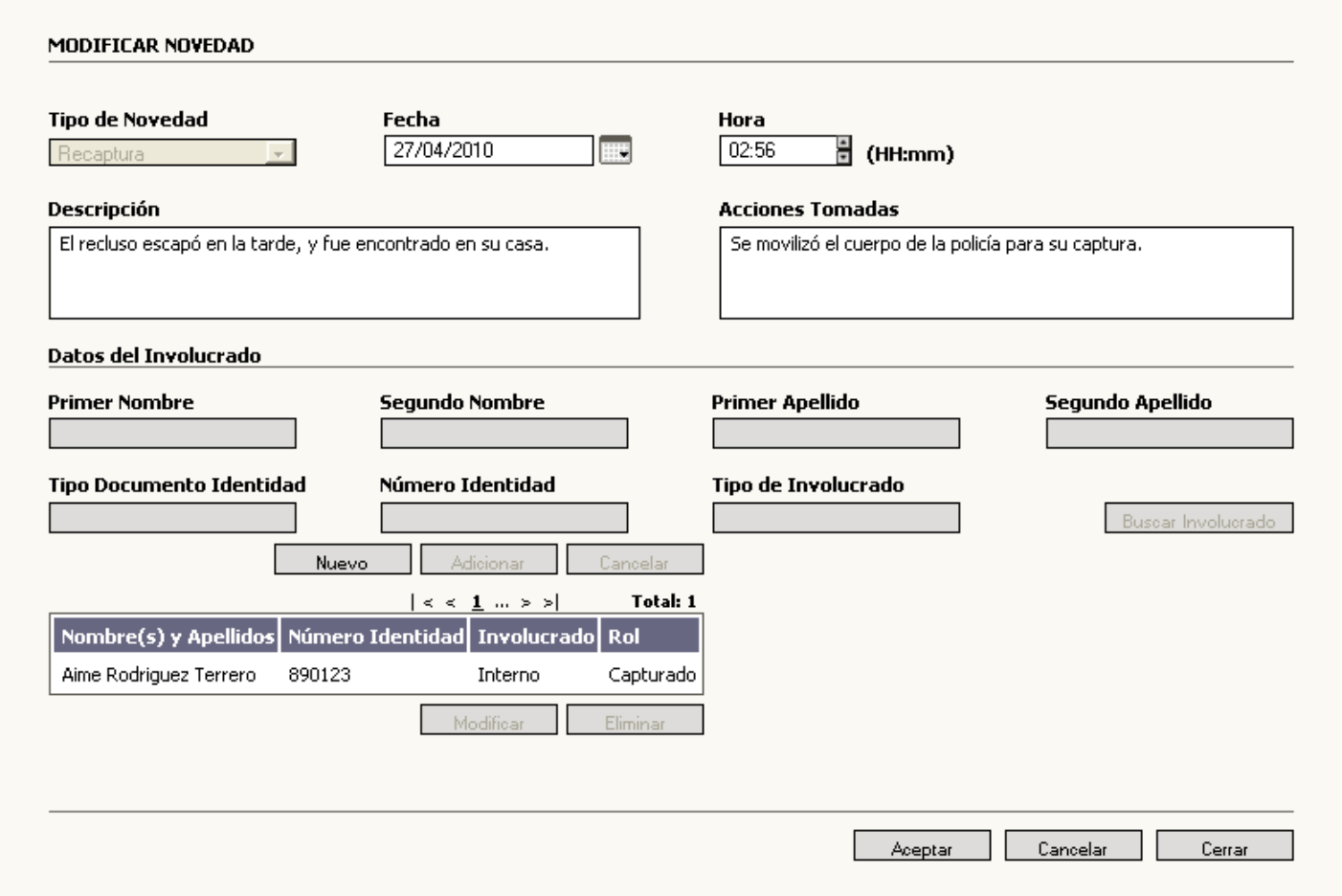


**Figura 3.6 Interfaz gráfica de Modificar Novedad.**

Una vez que el usuario interactúa con la vista principal del módulo, creada por el archivo *gestionarNovedades.jsp*, al pulsar el botón registrar o modificar se lanza un llamado que es atendido por el archivo *novedades.js* y este envía una petición que será resuelta por el archivo *sigep-seguridadcustodia-novedades-servlet.xml*.

Este archivo es el encargado de encapsular todos los beans de la capa de presentación. Utilizando Spring MVC se configuran componentes como Controladores, Handler Mappings, View Resolvers, utilizados para conformar la lógica de esta capa. En la sección del Handler Mappings se realiza la asignación de los patrones de URL a los objetos del controlador.

```
<bean id="simpleUrlHandler"
      class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
      <property name="mappings" ref="novedadesURLMappings"></property>
      <property name="interceptors">
            <list>
                  <ref bean="openSessionInViewInterceptor" />
            </list>
      </property>
 </bean>

<bean name="novedadesURLMappings"
class="vnz.sigep.administracion.seguridad.util.urlagrupation.MappingsAgrupation">
      <property name="name">
            <value>novedades</value>
      </property>
      <property name="fatherKey">
            <value>Seguridad Custodia</value>
      </property>
      <property name="pattern">
            <value>/novedades</value>
      </property>
      <property name="agrupations">
            <list> <ref bean="gestionarnovedadesUrlAgrupation"/> </list>
      </property>
</bean>

<bean name="gestionarnovedadesUrlAgrupation"
      class="vnz.sigep.administracion.seguridad.util.urlagrupation.URLAgrupation">
      <property name="name">
            <value>gestionar</value>
      </property>
      <property name="pattern">
            <value>/gestionar</value>
      </property>
      <property name="mappings">
            <props>
                  <prop key="/adicionarnovedad.htm, Adicionar una novedad">
                        adicionarEditarNovedadAbstractCommandController
                  </prop>
                  <prop key="/editarnovedad.htm, Editar una novedad">
                        adicionarEditarNovedadAbstractCommandController
                  </prop>
                        ...
            </props>
      </property>
</bean>
```

Luego se inyecta una instancia de la clase creada al atributo *novedadesFacade* del controlador:

```
<prop key="/adicionarnovedad.htm, Adicionar una novedad">
      adicionarEditarNovedadAbstractCommandController
</prop>
            . . .
<bean id="adicionarEditarNovedadAbstractCommandController"
      class="vnz.sigep.seguridadcustodia.novedades.web.controller.AdicionarEditarNo
vedadAbstractCommandController">
      <property name="novedadesFacade">
            <ref bean="novedadesFacade" />
      </property>
      <property name="nomencladoresFacade">
            <ref bean="nomencladoresFacade" />
      </property>
</bean>
```

Las clase controladora *AdicionarEditarNovedadAbstractCommandController.java* maneja el flujo web de la aplicación para esta funcionalidad, esta clase hereda de otra clase de Spring ***org.springframework.web.servlet.mvc.AbstractCommandController*** que acepta varios parámetros desde la petición que se realiza en la vista, los cuales son encapsulados en un objeto, que seria *novedadCommand*, donde los atributos de los archivos jsp según el tipo de novedad se machean con los del command, con objetos javascript, usando el formato JSON para el intercambio de información entre el cliente y el servidor.

Este controlador recibe como dato el identificador de una novedad, que en caso de no tener valor, significaría que la petición realizada es para registrar una nueva novedad, mostrándose la vista *registrarNovedad* con los campos del formulario vacíos.

Para resolver la vista retornada por el controlador, en el archivo *sigep-seguridadcustodia-novedades-servlet.xml* se construye la URL de la vista a mostrar, agregándole un prefijo con la localización dentro de los paquetes y un sufijo con su extensión.

```
<!-- Configurando los View Resolvers -->

<bean id="beanNameViewResolver"
      class="org.springframework.web.servlet.view.BeanNameViewResolver">
      <property name="order">
            <value>1</value>
      </property>
</bean>

<bean id="jstlViewResolver"
      class="vnz.sigep.common.global.util.CustomInternalResourceViewResolver">
      <property name="order">
            <value>2</value>
      </property>
      <property name="viewClass"
            value="org.springframework.web.servlet.view.JstlView" />
      <property name="prefix"
            value="/WEB-INF/jsps/seguridadcustodia/novedades/">
      </property>
      <property name="suffix">
            <value>.jsp</value>
      </property>
</bean>
```

Si este identificador tuviese algún valor, entonces la petición seria para modificar una novedad y el formulario necesitaría mostrar todos los datos de la misma, los que se obtienen a través del método *obtenerNovedadPorId(String idNovedad)* declarado en la interfaz *NovedadesFacade.java.*

Las fachadas de un módulo se basan en el patrón Facade para permitir una clara división entre las capas arquitectónicas, es por esta razón que se inyectan los beans de los managers responsables de implementar todas las funcionalidades de la fachada, en el fichero de configuración del contexto de Spring, *sigep-seguridadcustodia-novedades-business-context.xml* para la capa de negocio.

```
<bean id="novedadesFacade"
      class="vnz.sigep.seguridadcustodia.novedades.facade.impl.NovedadesFacadeImpl"
>
      <property name="novedadesManager">
            <ref bean="novedadesManager" />
      </property>
      <property name="centroManager">
            <ref bean="centroManager" />
      </property>
      <property name="individuoManager">
            <ref bean="individuoManager" />
      </property>
      <property name="personalManager">
            <ref bean="personalManager" />
      </property>
</bean>
```

Cada método de negocio definido en las interfaces de los managers, cuando se ejecuta, es envuelto dentro de un contexto transaccional. Para la definición de las transacciones sobre los métodos de negocio se utilizó la programación orientada a aspectos, garantizando la ejecución de un conjunto de acciones lógicas o que se reviertan en el caso de alguna anomalía (excepción). Las transacciones se aplican de forma declarativa en los ficheros de configuración del contexto de Spring.

```
<!-- Transacciones -->

<aop:config>
      <aop:advisor pointcut="execution(* *..*NovedadesManagerImpl.*(..))"
            advice-ref="NovedadesMgrAdvice" />
</aop:config>

<tx:advice id="NovedadesMgrAdvice">
      <tx:attributes>
            <tx:method name="*" rollback-for="java.lang.Exception" />
      </tx:attributes>
</tx:advice>
```

La interacción del negocio con la capa de persistencia se realizará mediante el uso de los DAOs, es por esto que en el fichero *sigep-seguridadcustodia-novedades-business-context.xml* se inyectan los beans de los DAOs encargados de encapsular la lógica de acceso a datos utilizados en el manager novedadesManager.

Los DAOs implementados se configuran en el fichero *sigep-seguridadcustodia-novedades-dataaccess-context.xml*. En la configuración de estos objetos se inyecta el sessionFactory de Hibernate como se muestra en el siguiente ejemplo:

```
<bean id="novedadDAO"
      class="vnz.sigep.seguridadcustodia.novedades.dao.impl.NovedadDAOImpl">
      <property name="sessionFactory">
            <ref bean="sessionFactory" />
      </property>
</bean>
```

La clase *NovedadDAOImpl.java* extiende de la clase *AbstractBaseDAO.java* e implementa de la interface *NovedadDAO.java*. La clase *AbstractBaseDAO.java* implementa las operaciones de acceso a datos genéricas de una entidad como persistir, eliminar, obtener todos entre otras que son utilizadas por todos los DAOs del módulo. Además la interfaz *NovedadDAO.java* te permite adicionar algunos métodos más específicos, como por ejemplo *getNovedadesByCentro (final Centro centro)* el cuál te proporciona un listado de novedades según el centro que se provee como parámetro.

En la implementación de la capa de acceso a datos se crean los ficheros de mapeo de Hibernate (hbm.xml) para la conexión con la base de datos, en los que se mapean todos los atributos de las clases con los de las tablas de la Base Datos, así como las relaciones existentes entre las entidades. Se observa a continuación una imagen del archivo *Novedad.hbm.xml*.

```
<property name="fechaFin" type="date">
      <column name="FECHA_FIN" length="7" />
</property>

<many-to-one name="tipoNovedad"
      class="vnz.sigep.common.global.domain.NomTipoNovedad" fetch="select">
      <column name="ID_TIPO_NOVEDAD" length="20" not-null="true" />
</many-to-one>
```

Una vez realizado todos los métodos correspondientes de los DAOs y los mapeos en los archivos hbm, logramos encapsular la persistencia de los objetos de dominio tanto para registrar una novedad, como mostrar datos de una ya existente a los que sea necesario realizar alguna modificación, completándose así, el flujo de actividades registrar y modificar que propone la arquitectura del SIGEP.

### 3.8 Análisis de resultados de validaciones con el cliente.

El módulo Novedades del SIGEP ha sido verificado en dos ocasiones, en pruebas de aceptación con usuarios finales y especialistas funcionales y en prueba piloto en un ambiente real en el Centro Penal la Mínima de Carabobo del estado Valencia. Este módulo pertenece a la segunda versión del SIGEP que se encuentra desplegada en el centro mencionado y en la Dirección General de Servicios Penitenciarios de la República Bolivariana de Venezuela.

En las pruebas de aceptación realizadas se detectaron una serie de no conformidades por parte del cliente, las que se mencionan a continuación:

- Agregar a las pantallas huelga, secuestro, autosecuestro, motín los roles mediador e instigador.
- Cuando se regresa a la pantalla de Novedades, luego de haber registrado o modificado una novedad, la pantalla se abre lentamente, haciendo un barrido de arriba abajo.
- Cuando se entra a la pantalla de modificar una novedad, los botones Aceptar y Cancelar no funcionan.
- La novedad "huelga de hambre" debe llevar cantidad de muertos.

Todas las no conformidades fueron resueltas, lo que permitió que en la prueba piloto los resultados fueran satisfactorios y el cliente quedara satisfecho con el trabajo realizado.

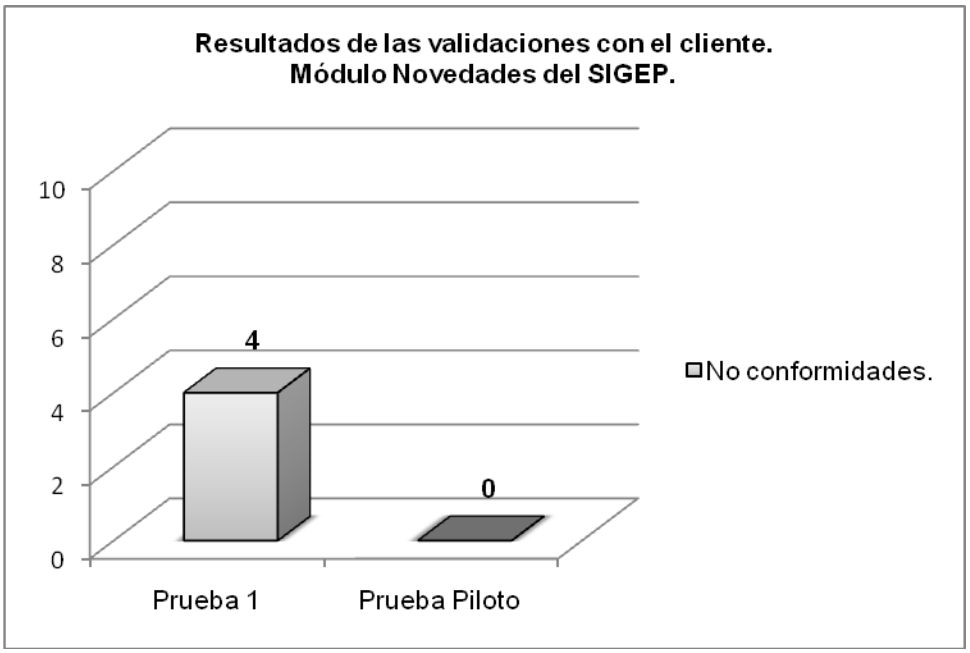


**Figura 3.7 Resultados de las pruebas realizadas al módulo Novedades.**

## 3.9 Conclusiones

Una vez realizado todo el proceso de diseño e implementación de la solución de software para los requisitos relacionados con el proceso de gestionar novedades, siguiendo el flujo de trabajo definido para el SIGEP, podemos afirmar que se cumplieron todas las tareas propuestas para el desarrollo de este capítulo. Apoyándonos en la elaboración de los diagramas correspondientes para el diseño y por consiguiente la creación de todas las clases requeridas para la implementación, se utilizaron patrones de diseño para propiciar la mejor solución acorde a las especificidades del proyecto, permitiendo la integración satisfactoria del módulo Novedades al SIGEP.

## CONCLUSIONES

Con el presente trabajo de diploma se obtuvo como resultado la puesta en marcha del módulo Novedades del SIGEP, dando cumplimiento a los objetivos propuestos para la solución del problema planteado. Las tareas trazadas para cumplir los objetivos, tuvieron excelentes resultados que fueron evidenciados en las pruebas realizadas, donde solo se detectaron un mínimo de no conformidades que fueron resueltas de inmediato, posibilitando la grata aceptación por parte del cliente, lo que demuestra que la aplicación cumple satisfactoriamente con los requisitos de software estudiados, utilizando las tecnologías y metodologías definidas por el proyecto.

## RECOMENDACIONES

Se recomienda monitorear la eficiencia del módulo Novedades, al trabajar con los volúmenes de datos reales, ya que hasta el momento sólo ha sido probado con volúmenes de datos inferiores a los del entorno de utilización.

Se recomienda agregar funcionalidades para la generación de reportes, con el objetivo de brindar a los usuarios finales la información gestionada por el módulo.

## BIBLIOGRAFÍA

**Albiol, Francesc Rosés.** *Introducción a Hibernate.* 2003.

**Bauer, Christian y King, Gavin.** *Hibernate in Action.* Greenwich : MANNING.

**Benavides Zaila, Yadira, Gomez Correa, Juan Carlos y Rivero Duharte, Franklin. 2005-2006.** *Reportes con jasperreport, ireport y jfreechart en Aplicaciones Empresariales en Java.* Ciudad de La Habana : s.n., 2005-2006.

**Castiglia, Angel.** *Buenas Prácticas.* Buenos Aires, Argentina : s.n.

**Castro Mecías, Lilian Teresa y Bermúdez Peña, Rolando. 2008.** *Desarrollo de una Librería de Componentes javascript basado en Dojo Toolkit.* Ciudad de La Habana : s.n., 2008.

**Garcia, Lic. Rosa María Mato.** *Diseño de bases de datos.* 1999.

**Garrido, Juan Salvador Castejón.** *Arquitectura y diseño de sistemas web modernos.* 1, Murcia : s.n., 2004.

**Heffel, David R.** *Jasperreports for Java Developers.* Birmingham, Mumbai : Packt Publishing, 2006.

*Introducción a la plataforma Eclipse.* S.l. : Proyecto Pasantía LEAD.

**Jacobson, Ivar**.; **Booch, Grady** y otros. *El Proceso Unificado de Desarrollo de Software.Enterprise Manager.* June 2001.

**KNIGHT, T. C. H.** *Sistema Informativo de la Dirección de Establecimientos Penitenciarios*, 2005.

**Lanvin, Daniel Fernández.** *Definición de una arquitectura software para el diseño de aplicaciones web basadas en tecnología Java-J2EE.* Oviedo : s.n.

**Larman, Craig.** *UML y Patrones.*

**Morero, Francisco. 1999-2000 .** *Introducción a la OOP.* S.l. : Grupo EIDOS, 1999-2000 .

**Osorio, Manuel.** *Diccionario de Ciencias Jurídicas, Políticas y Sociales*. Heliasta,1997.

**Rodríguez, Gerson Johan Samaniego.** *Evolución de los lenguajes de programación ¿Por qué cambiarse a la Programación Orientada a Objetos?* Universidad de los Andes : s.n.

RUP. *“Rational Unified Process”.* 2003.

**Walls, Craig and Breidenbach, Ryan.** *Spring in Action.* Estados Unidos de América : Manning Publications Co., 2005.

**Williamson, Alan, et al.** *Programacion Java Server con J2EE Edicion.*

**ANEXO 1: Descripción de los requisitos funcionales más significativos a través de los casos de uso del sistema (CUS) utilizando el formato informal:**

1. **CUS_Registrar Novedad.**

**Escenario principal de éxito:**

El usuario indica en el sistema que desea añadir una nueva novedad. El sistema en correspondencia con el tipo de novedad seleccionada, muestra los datos relacionados con ella y muestra además una lista de personas que pudieran estar involucrados. El sistema valida y guarda todos los datos.

**Escenarios alternativos:**

El sistema aún no tiene al involucrado en la novedad a registrar y le muestra al usuario que debe registrarlo. El usuario indica que desea añadir un involucrado. El sistema permite añadir los datos del involucrado.

Los datos introducidos son inválidos. El sistema señala al usuario los errores detectados y permite al usuario corregirlos.

2. **CUS_Modificar Novedad.**

**Escenario principal de éxito:**

El usuario indica que desea modificar la novedad seleccionada. El sistema muestra todos los datos y permite la realización de algunos cambios, valida los datos y guarda las transformaciones.

**Escenarios alternativos:**

Los datos introducidos son inválidos. El sistema muestra al usuario los errores detectados y permite al usuario corregirlos.

3. **CUS_Consultar Novedad.**

**Escenario principal de éxito:**

El usuario indica que desea ver los detalles de una novedad seleccionada. El sistema muestra una nueva ventana con todos los datos referentes a esa novedad.

4. **CUS_Eliminar Novedad.**

**Escenario principal de éxito:** El usuario indica que desea eliminar una novedad seleccionada. El sistema le permite al usuario eliminar la novedad.

## ANEXO 2: DIAGRAMAS DE SECUENCIA

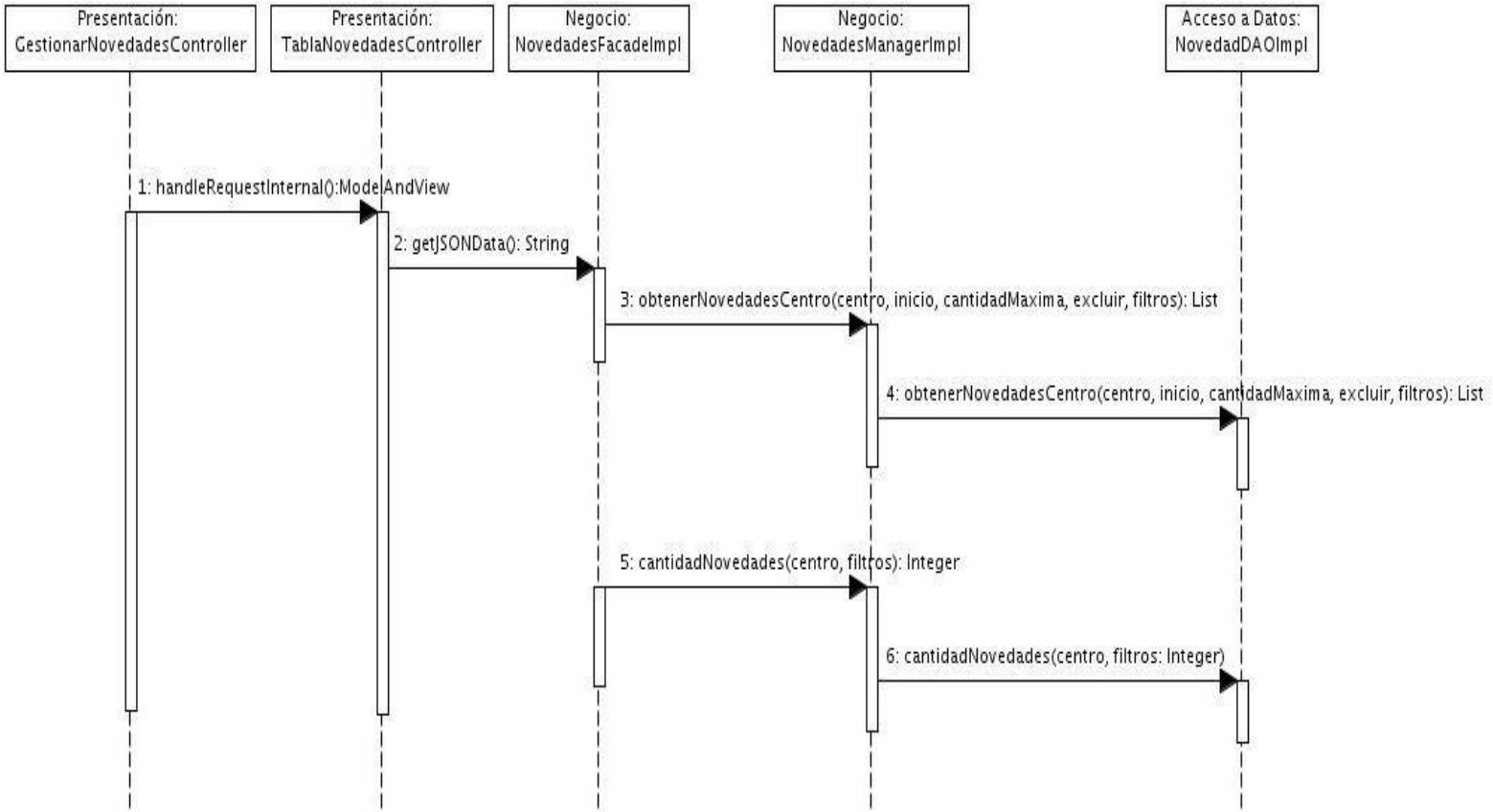


**Figura-Anexo 2.1 Diagrama de secuencia del caso de uso Gestionar Novedades.**

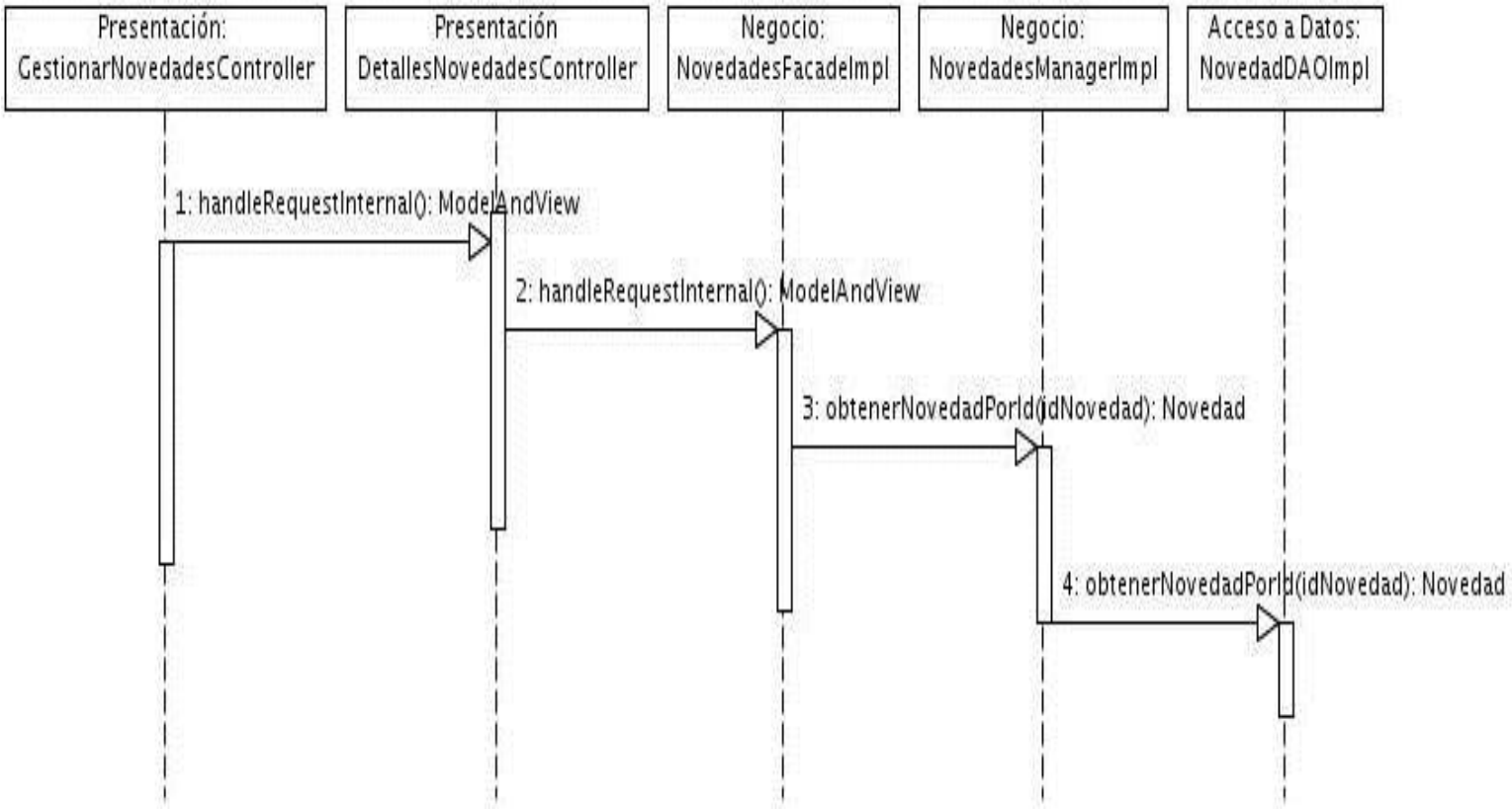


**Figura-Anexo 2.2 Diagrama de secuencia del caso de uso Consultar Novedades.**

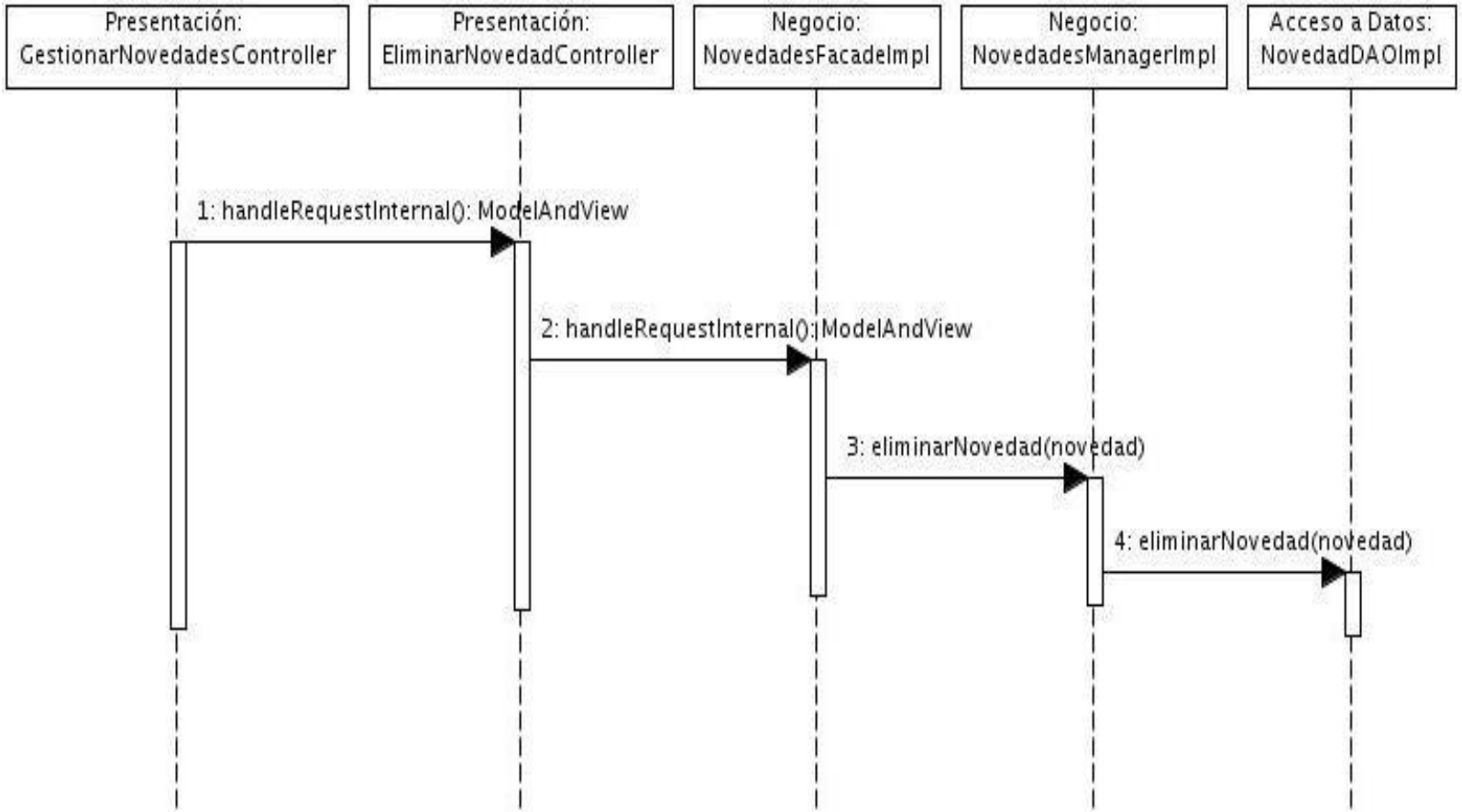


**Figura-Anexo 2.3 Diagrama de secuencia del caso de uso Eliminar Novedades.**

## ANEXO 3: DIAGRAMAS DE CLASES DE ESTEREOTIPOS WEB

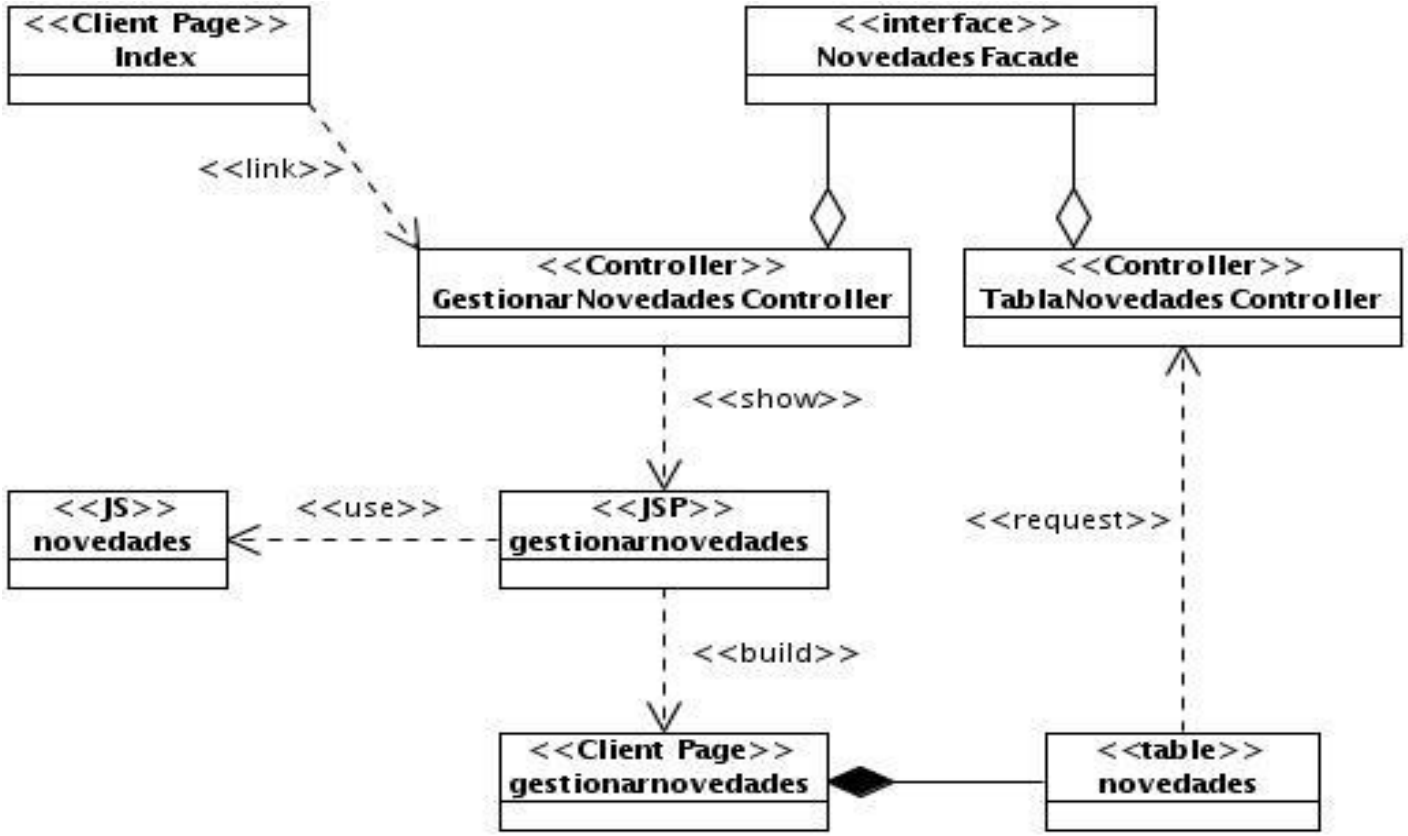


**Figura-Anexo 3.1 Diagrama de clases de estereotipos web del caso de uso Gestionar Novedad.**

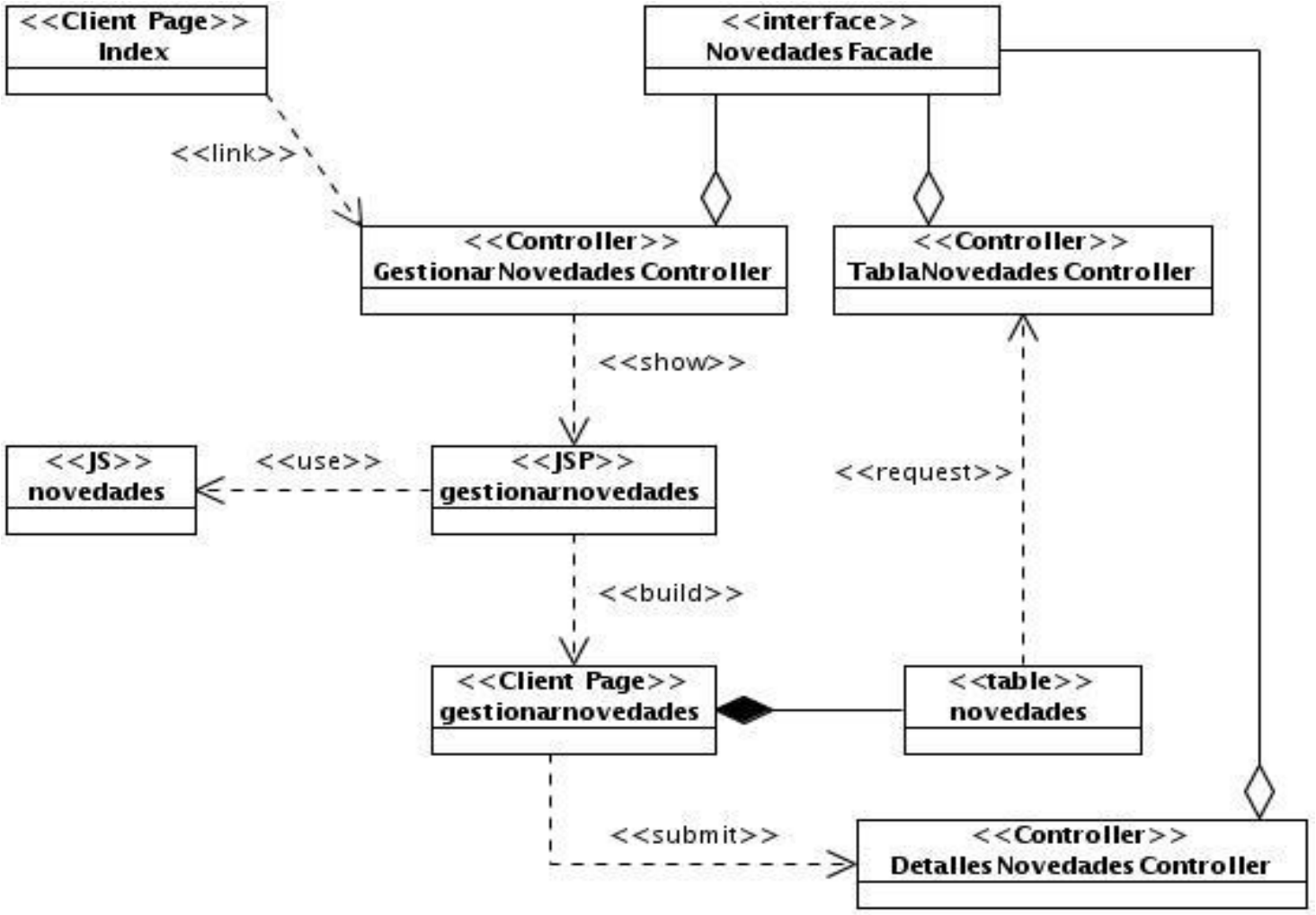


**Figura-Anexo 3.2 Diagrama de clases de estereotipos web del caso de uso Consultar Novedad.**

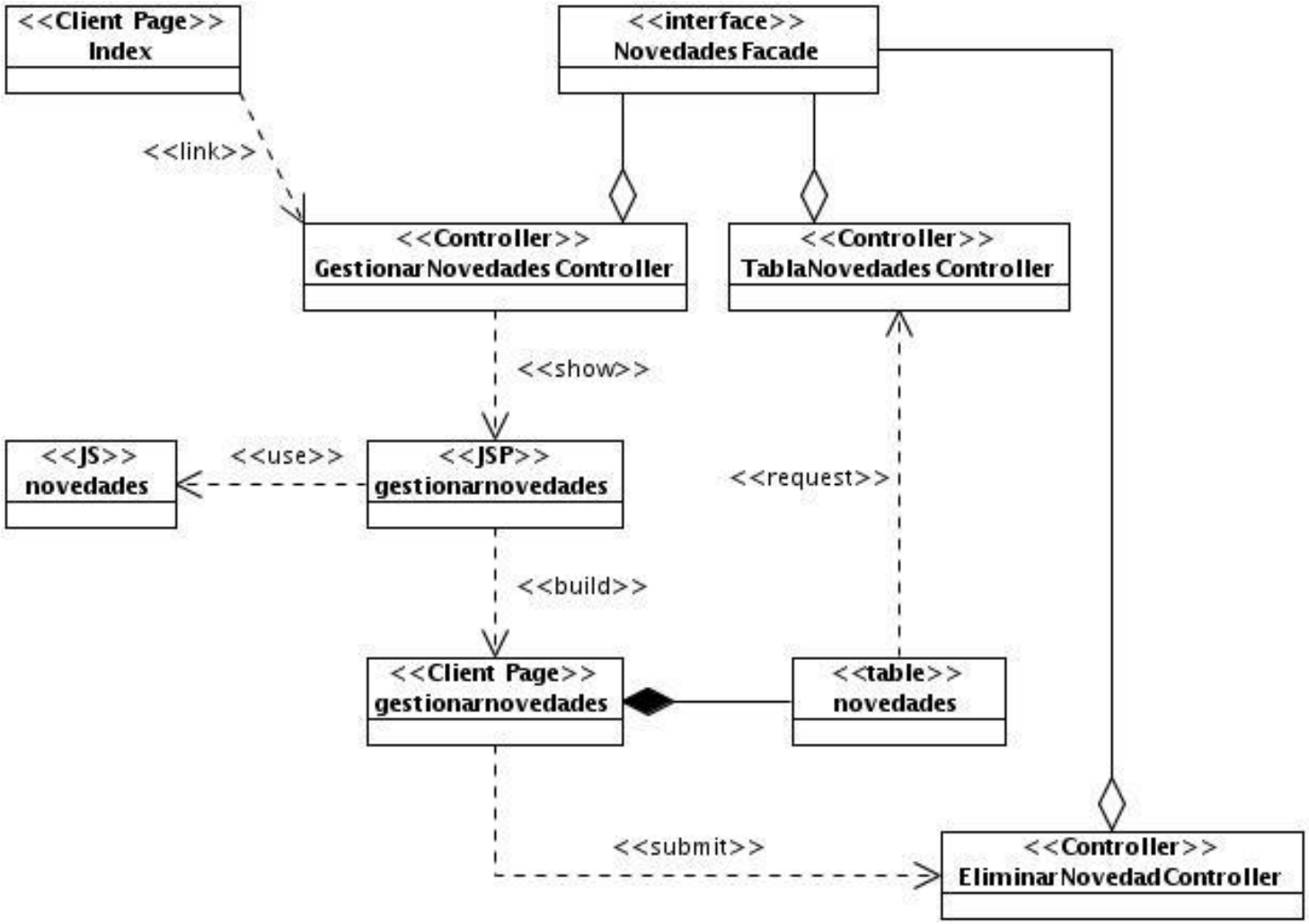


**Figura-Anexo 3.3 Diagrama de clases de estereotipos web del caso de uso Eliminar Novedad.**

## ANEXO 4: IMÁGENES DE LA APLICACIÓN

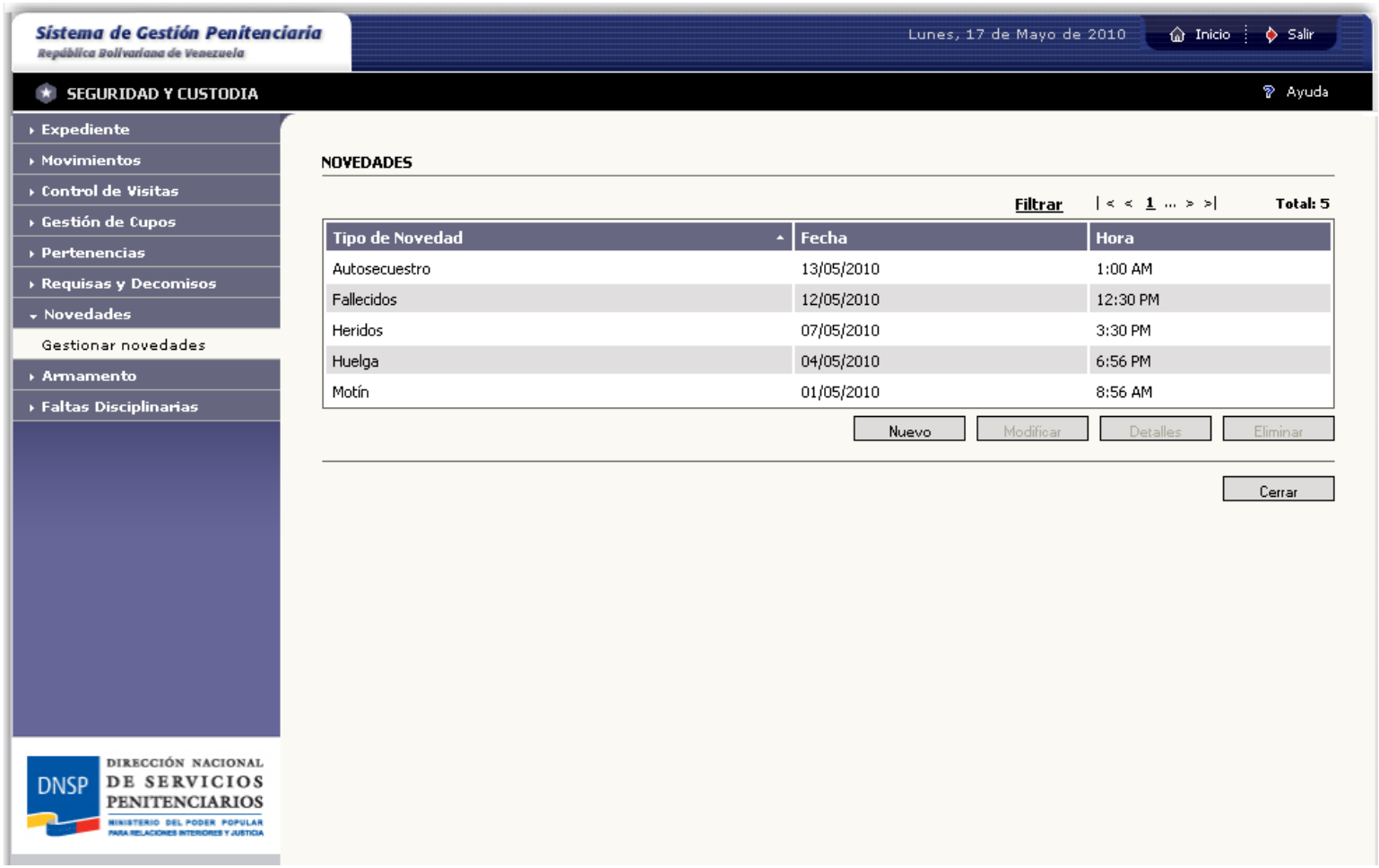


**Figura-Anexo 3.1 Interfaz gráfica de Gestionar Novedades.**


**Figura-Anexo 3.2 Interfaz gráfica de Consultar Novedad.**

## GLOSARIO DE TÉRMINOS

**A**

**Aplicación informática:** Es un tipo de programa informático diseñado para facilitar al usuario la realización de un determinado tipo de trabajo.

**Atributos:** Son las características o propiedades que definen a un objeto.

**C**

**Caso de uso:** Fragmentos de funcionalidad que el sistema ofrece para aportar un resultado de valor para sus actores.

**Clase:** Es una abstracción sobre un conjunto de objetos que tienen en común estructura y comportamiento.

**D**

**Dato:** Información en un formato que pueda ser procesado por un ordenador.

**E**

**Entidad de Dominio:** Es una clase que representa conceptos reales del negocio.

**H**

**HTML:** Hyper Text Mark Language (Lenguaje de Marcas de Hipertexto). Es el lenguaje de marcado predominante para la construcción de páginas Web.

**HTTP:** Hypertext Transfer Protocol (Protocolo de transferencia de hipertexto). Es el protocolo usado para intercambiar archivos (texto, gráfica, imágenes, sonido, video y otros archivos multimedia) en la World Wide Web.

**I**

**Interface:** Es un conjunto de constantes y métodos a los que no se les da implementación, también se puede ver como una clase que solo cuenta con métodos abstractos.

**J**

**Java:** Es un lenguaje de programación de alto nivel orientado a objetos.

**JDBC:** Java Database Connectivity, es un API que permite la ejecución de operaciones sobre base de datos desde el lenguaje de programación Java.

**JDO:** Java Data Objects (Objetos de datos de Java), es una definición de interfaz basada en la persistencia de objetos para el lenguaje Java, que describe el almacenamiento, consulta y recuperación de objetos de almacenes de datos.

**JDT:** Java Development Tool (Herramientas de desarrollo Java), es conjunto de plug-ins que añaden las capacidades de un IDE con todas las funciones de Java para la plataforma Eclipse.

**JSP:** Java Server Pages (Páginas servidoras de java), ofrece una forma sencilla para crear páginas web dinámicas que son independientes de la plataforma e independiente del servidor.

**JSON:** JavaScript Object Notation (Notación de objetos de Javascript), formato que representa a los objetos de Javascript.

**M**

**Método:** Son operaciones que pueden modificar el estado de un objeto o simplemente obtener datos sobre el mismo.

**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.

**O**

**Objeto:** Es el resultado de la instanciación de una clase.

**Objetos persistentes:** Son los objetos que van a contener la información persistente de la aplicación, en este caso los objetos persistentes serán los objetos de dominio.

**P**

**Patrón:** Son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces.

**Penado:** Persona del sexo femenino o masculino que se encuentra en un centro penitenciario en cumplimiento de una sanción firme de privación de libertad.

**PDE:** Plug-in Development Environment ofrece herramientas para crear, desarrollar, probar, limpiar, construir y desplegar plug-ins de Eclipse, fragmentos, características, actualizar sitios y productos de RCP.

**Programación:** Se define como el proceso de creación de un programa de computadora, mediante la aplicación de procedimientos lógicos.

**Pruebas unitarias:** Pruebas realizadas a las partes más atómicas del sistema. Verifican el comportamiento de sólo una clase o método, etcétera, que es consecuencia de una decisión de diseño.

**R**

**Régimen intramuros:** Establecimientos penitenciarios cerrados dónde cumplen sanción los individuos privados de libertad o los individuos que se encuentran bajo una medida cautelar privativa de libertad.

**Régimen extramuros:** Centros penitenciarios abiertos, en estos centros solo se ingresa por otorgamiento de régimen abierto, libertad condicional, suspensión condicional de le ejecución de la pena y suspensión condicional del proceso.

**S**

**Servlet:** Es una clase Java que ofrece funciones suplementarias al servidor.

**SIGEP:** Sistema de Gestión Penitenciaria.

**Software:** Es el conjunto de los programas de cómputo, procedimientos, reglas, documentación y datos asociados que forman parte de las operaciones de un sistema de computación.

**T**

**Toolkit:** Es una biblioteca de utilidades, las cuales incorporan componentes de interfaz de usuario (IU) o widgets.

**U**

**UML:** Unified Modeling Language (Lenguaje unificado de modelado), se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software.

**URL:** Uniform Resource Locutor (Locutor Uniforme de Recursos). Secuencia de caracteres de acuerdo a un formato estándar que se usa para nombrar recursos como documentos e imágenes en Internet según su localización.

**X**

**XHTML:** Es el lenguaje de marcado pensado para sustituir a HTML como estándar para las páginas Web. En su versión 1.0, XHTML es solamente la versión XML de HTML, por lo que tiene, básicamente, las mismas funcionalidades, pero cumple las especificaciones, más estrictas, de XML.

**XML:** Extensible Markup Language (lenguaje de marcas extensible). Propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo, etc.

1