Universidad de las Ciencias Informáticas

Facultad 6

Sistema Informático para la Gestión de Información de los Recursos Materiales y Programas- Proyectos de las Coordinaciones Regionales de Prevención del Delito.

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

Autor(es): Wilmar Barban Otaño

Danoy Alonso Llerena

Tutor(es): Ing. Aidacelys López Díaz

Ing. Luis Enrique Ramírez Noy

Noviembre, 2009

"El que no posee el don de maravillarse ni de entusiasmarse más le valdría estar muerto, porque sus ojos están cerrados"

Albert Einstein

DECLARACIÓN DE AUTORÍA

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

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

Wilmar Barban Otaño

_____

Firma del Autor

Danoy Alonso Llerena

_____

Firma del Autor

Ing. Aidacelys López Díaz

_____

Firma del Tutor

Ing. Luis Enrique Ramíres Noy

_____

Firma del Tutor

DATOS DE CONTACTO

Tutores:

Ing. Aidacelys López Díaz

Universidad de las Ciencias Informáticas, Habana, Cuba.

Email: alopezd@uci.cu

Ing. Luis Enrique Ramírez Noy

Universidad de las Ciencias Informáticas, Habana, Cuba.

Email: noy@uci.cu

AGRADECIMIENTOS

En primer lugar, agradecerle al Comandante en Jefe Fidel Castro por crear este magnífico proyecto.

A nuestros padres, hermanos y novias por ser guías e inspiración en todo momento.

A nuestros tutores por apoyarnos y guiarnos.

A un amigo en especial Roniel Hernández Cuervo que nos ayudo muchísimo, creo que sin su apoyo no hubiera sido posible este trabajo.

A Liusmila Nieto Cervantes, Andrés Ballester Marsal y en general a todos los profes del proyecto por su ayuda incondicional.

A todos nuestros compañeros del proyecto que de una manera y otra nos ayudaron y compartieron momentos con nosotros.

A todos nuestros amigos que nos ayudaron y apoyaron en el transcurso de todos estos años en especial a Alex Tablada Álvarez y Dayana de la Mella Reus.

A todos los que de una forma u otra contribuyeron y nos apoyaron en el desarrollo de este trabajo.

A todos muchas gracias.

DEDICATORIA

Dedico esta tesis a mis padres María del Carmen Llerena y Francisco Alonso Alonso por educarme, guiarme y ponerme siempre en el camino correcto…

A mi novia y hermana por confiar en mí y siempre apoyarme…

Danoy

Deseo dedicar el presente trabajo a mis padres Walter Barban Fonseca y María Elena Otaño Piñero, a mi hermano, mi novia y a mi familia en general por su apoyo, confianza y guía…

Wilmar

RESUMEN

La presente investigación surge en el marco de trabajo del Proyecto: “Solución Integral para el Perfeccionamiento del Sistema de Prevención del Delito de la República Bolivariana de Venezuela” aprobado en la Novena Comisión Mixta Cuba-Venezuela del Convenio de colaboración Cuba-Venezuela; más conocido como “Prevención del Delito de la República Bolivariana de Venezuela”. En esta investigación se desarrollará una aplicación web empresarial con el objetivo de resolver la problemática de cómo mejorar la gestión de la información referente a los Recursos materiales y Programas – Proyectos de las Coordinaciones Regionales y la Dirección General de Prevención del Delito. Para ello los autores desempeñarán los principales roles definidos por RUP en cada fase, además de que se utilizarán herramientas y tecnologías de la plataforma J2EE que exponen tendencias de la programación web actual y otras herramientas como apoyo a la aplicación.

PALABRAS CLAVE

SIGIRMPP, Spring, Hibernate, DGPD, CR, RUP, Java.

TABLA DE CONTENIDOS

INTRODUCCIÓN

Con el triunfo del Comandante Hugo Chávez, la Revolución Bolivariana prioriza la agenda social, generando condiciones para la inclusión. La salud, la educación, el empleo, las viviendas, la seguridad ciudadana entre otros indicadores, comienzan a favorecer a los sectores más pobres y excluidos, siendo estas variables macro sociales elementos que impactan positivamente en la seguridad ciudadana y la prevención del delito. [1]

En tal sentido el Ministerio del Poder Popular para Relaciones Interiores y Justicia (MPPRIJ), atendiendo a su misión de garantizar la seguridad ciudadana, mediante la formulación de políticas dirigidas al resguardo de la paz pública, desarrollo territorial equilibrado y la estabilidad de la nación; promueve la formulación y funcionamiento del proyecto “Solución Integral para el Perfeccionamiento del Sistema de Prevención del Delito de la República Bolivariana de Venezuela”, partiendo de la premisa de que la seguridad ciudadana es una condición necesaria para el desarrollo humano, la cual es fundamental para desenvolverse en la vida cotidiana, con el menor nivel de amenaza a los derechos, la integridad personal y el goce de los bienes; así como también con el fin de brindar a las coordinaciones regionales una herramienta que mejore su desempeño, para de esta manera garantizar una asistencia de mayor calidad al ciudadano.

La Dirección General de Prevención del Delito (DGPD) está adscrita al Viceministerio de Seguridad Ciudadana perteneciente al Ministerio del Poder Popular para Relaciones Interiores y Justicia de la República Bolivariana de Venezuela. Este organismo posee 22 Coordinaciones Regionales (CR); 18 adscritas al MPPRIJ, ubicadas en los Estados: Anzoátegui, Aragua, Barinas, Bolívar, Carabobo, Cojedes, Falcón, Guárico, Lara, Mérida, Miranda, Monagas, Portuguesa, Táchira, Trujillo, Yaracuy y Zulia y en el Distrito Capital; además de cuatro que se han creado de forma descentralizada en los estados Delta Amacuro, Nueva Esparta, Sucre y Vargas.

Entre las funciones de la Dirección General de Prevención del Delito se encuentran formular, coordinar y evaluar políticas, además de los programas y proyectos relacionados con la prevención de la violencia y la criminalidad; también se encarga de promover y ejecutar políticas de Estado en materia de investigaciones criminológicas y de la formación del personal para el desarrollo de los programas dirigidos hacia este sentido. Al mismo tiempo, se encarga de promover la participación de las comunidades en los

programas de prevención de la violencia y coordinar las acciones y actividades dirigidas a la prevención de la criminalidad en conjunto con instituciones públicas y privadas.

Las CR están conformadas por un equipo de profesionales y técnicos (Coordinador General, secretaria, trabajador social, administradores, entre otros funcionarios) que se encargan de la atención e implementación de los programas de prevención del delito, como el Programa de Educación y El Proyecto de la Comunidad. La DGPD propone una serie de programas y proyectos acorde a las necesidades de cada región, grupo o problema a enfrentar; que las CR deben ejecutar en sus áreas de acción; a la vez que las CR pueden realizar sus propios proyectos, los cuales deben ser aprobados previamente por la DGPD. Es necesario destacar que los programas generan actividades diversas las cuales son orientadas a cada CR por la supervisora de la DGPD correspondiente a ese estado, cada región adapta las actividades según su realidad, por lo que las CR deben rendir cuenta de su ejecución generando un informe mensual que contiene todo lo realizado en el período, así como otra información demostrando sus logros y desempeño.

También manejan recursos materiales limitados (bienes muebles y bienes materiales) que le son entregados para el cumplimiento de sus actividades, asignados por la DGPD, la gobernación u otras instituciones. Estos recursos son controlados por la DGPD a través de un código asignado a cada recurso; por lo que todas las CR deben enviar un informe del inventario de sus recursos materiales cuando se lo solicitan. De los mismos se gestionará un registro de control, las solicitudes de materiales y actas de entregas al recibir los mismos, entre otros datos de interés.

Con los recursos existentes se realizan diversas operaciones, inserción de nuevos materiales, desincorporación de los mismos, entre otros; los cuales son dirigidos por el administrador de la CR**.**

En este contexto las CR no cuentan con una manera efectiva de gestionar la información referente a los programas – proyectos y al control de los recursos materiales. Esta información es hoy manejada y archivada de forma manual, cada CR crea sus propios formatos para la realización de sus informes lo que no garantiza la estandarización de la información y dificulta el análisis de la misma que cada supervisora debe hacer de los estados que atiende. Es importante destacar además que cada supervisora atiende varios estados(un estado tiene una CR); es decir, posee la misma información en diferentes formatos

además de los datos propios de cada coordinación, dificultando el intercambio y adecuado flujo de información entre la CR y la DGPD, lo cual afecta la ejecución de sus procesos internos y la efectividad en el logro de resultados dirigidos al desarrollo de acciones que contribuyan a la prevención de la violencia, el fortalecimiento de la convivencia ciudadana y la paz, para alcanzar la calidad de vida adecuada en las comunidades venezolanas; por ello se plantea como **problema a resolver**: ¿Cómo contribuir a mejorar la gestión de la información referente a los recursos materiales y los programas-proyectos que manejan las Coordinaciones Regionales y la Dirección General de Prevención del Delito?

El problema trazado se enmarca en el siguiente **objeto de estudio:** Proceso de Gestión de información de las Coordinaciones Regionales de Prevención del Delito de la República Bolivariana de Venezuela y **el campo de acción** es el Proceso de Gestión de información asociada a los recursos materiales y los programas- proyectos realizados por las Coordinaciones Regionales de Prevención del Delito.

Para dar solución al problema planteado, esta investigación tiene como **objetivo general:** Desarrollar un sistema informático que gestione la información referente a los Recursos materiales y Programas – Proyectos de las Coordinaciones Regionales y la Dirección General de Prevención del Delito.

A partir de este objetivo general se trazaron los siguientes **objetivos específicos:**

1. Identificar las funcionalidades que debe cumplir el sistema informático.
2. Diseñar las clases del sistema informático.
3. Implementar el sistema informático.

Los que se cumplirán a través de las siguientes **tareas de la investigación:**

1. Estudio del negocio.
2. Definición de los requisitos funcionales del sistema informático.
3. Definición de los requisitos no funcionales del sistema informático.
4. Investigación del estado del arte, las herramientas y tecnologías existentes para el desarrollo del sistema informático.
5. Selección de las herramientas y tecnologías para el desarrollo del sistema informático.
6. Realización de los diagramas de clases del diseño correspondiente al sistema informático.
7. Realización de los diagramas de secuencia del diseño correspondiente al sistema informático.

8. Realización de diseño de la base de datos del sistema informático.
9. Realización de los diagramas de componentes del sistema informático.
10. Implementación de los componentes del sistema informático.
11. Realización de pruebas a nivel de desarrollador del sistema informático.

A continuación se describe de manera resumida el contenido que se expone en cada capítulo:

Capítulo 1 Fundamentación Teórica: Este capítulo comprende el estudio del estado del arte acerca de los elementos fundamentales a tener en cuenta para la solución del problema planteado, se definen los principales conceptos en relación con el objeto de estudio, la metodología de desarrollo, lenguaje de programación, y la plataforma de desarrollo.

Capítulo 2 Características del Sistema: Este capítulo describe la propuesta de solución para el problema a resolver. Se presentan los Diagramas de Casos de Uso del Sistema con sus especificaciones. Además se describen los requisitos funcionales y no funcionales, así como se describen los patrones de casos de usos utilizados.

Capítulo 3 Diseño del Sistema: En este capítulo se describe el diseño de la aplicación, se muestran los patrones de diseño empleados, los diagramas de clases y de interacción. Además se detallan los patrones y estilos arquitectónicos utilizados, así como el modelo de diseño de la base de datos (MER).

Capítulo 4 Implementación del Sistema: En este capítulo se presentan los diagramas de componentes que se definieron durante la implementación de la aplicación y se muestran además algunas imágenes de esta. También, muestran las pruebas realizadas sobre la aplicación para comprobar sus funcionalidades.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En el presente capítulo se abordan algunos conceptos imprescindibles para el desarrollo del Sistema Informático para la Gestión de Información de los Recursos Materiales y Programas - Proyectos de las Coordinaciones Regionales de Prevención del Delito. (SIGIRMPP). En él se analizan las tecnologías, herramientas y tendencias de desarrollo a utilizar en el diseño e implementación del sistema.

1.2 Sistema de Gestión de información.

En la era de la información y de la explosión de sus tecnologías, se vive la etapa en la que la humanidad ha alcanzado un desarrollo imprevisible. En este contexto, debe entenderse que las tecnologías de la información y las telecomunicaciones no son más que un medio para transmitir, gestionar datos y conocimiento. Sin embargo uno de los principales problemas de la información es su exceso, es necesario invertir mucho tiempo en ella; por lo que debe ser gestionada en disímiles ocasiones.

La gestión de la información es el proceso de analizar y utilizar la información que se ha recabado y registrado; permitiendo tomar decisiones documentadas. Esta se caracteriza por una visión más amplia de las posibilidades reales de una organización para resolver determinada situación o arribar a un fin determinado.

Un sistema de información puede definirse como un conjunto de componentes interrelacionados que recolectan (o recuperan), procesan, almacenan y distribuyen información para apoyar la toma de decisiones y el control de una organización. También pueden ayudar a los directivos y al personal al análisis de problemas, la visualización de cuestiones complejas y la creación de nuevos productos. [2] Estos poseen una enorme importancia en el incremento de la capacidad organizacional frente al cambio del entorno. La voluntad de lograr un sistema de información útil, que permita obtener una ventaja competitiva, implica la posibilidad de ofrecer múltiples, frecuentes, oportunas y relevantes informaciones.

La necesidad actual de tomar decisiones en poco tiempo para poder hacer frente a la agresividad del entorno hace necesaria la inmediatez de la información procesada con una gran dosis de veracidad,

fenómeno del cual no se encuentra exento el MPPRIJ. Sin embargo para realizar un sistema de gestión de información es necesario definir una metodología que guíe el desarrollo del mismo.

1.3 Metodologías de desarrollo de software.

Todo desarrollo de software es riesgoso y difícil de controlar, pero si no se lleva una metodología de por medio, se obtendrán clientes insatisfechos con el resultado y profesionales frustrados. La ausencia de una metodología en el desarrollo de un proyecto de software garantiza con seguridad también la ausencia de calidad.

Una metodología de desarrollo de software consiste en un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar un software.

Tener metodologías diferentes para aplicar de acuerdo con el proyecto que se desarrolle resulta imprescindible teniendo en cuenta las necesidades cambiantes que tiene el entorno de desarrollo actual y el acelerado progreso de la informática a nivel mundial resulta una idea interesante. Estas metodologías pueden involucrar prácticas tanto de metodologías ágiles como de metodologías tradicionales.

Las metodologías pueden ser clasificadas considerando su filosofía de desarrollo en **metodologías ágiles o ligeras** y **metodologías pesadas o tradicionales**.

1.3.1 Metodologías ágiles o ligeras:

Se caracterizan por ser incrementales (Entregas pequeñas de software, con ciclos rápidos); cooperativo (Cliente y desarrolladores trabajan juntos constantemente con una cercana comunicación); sencillo (El método en sí mismo es simple, fácil de aprender y modificar); estar bien documentadas y ser adaptables (Permite realizar cambios de último momento). Sus elementos clave son: poca documentación, simplicidad, análisis como una actividad constante, diseño evolutivo, integraciones y testeos diarios. Entre algunas de las metodologías ágiles de desarrollo de software se encuentran: Adaptive Software Development (ASD), Agile Unified Process (AUP), Essential Unified Process (EssUP), Feature Driven Development (FDD), Lean Software Development (LSD), Open Unified Process (OpenUP), Programación Extrema (XP), Scrum, entre otras. [3]

1.3.2 Metodologías pesadas.

Teniendo en cuenta la filosofía de desarrollo de las metodologías, aquellas con mayor énfasis en la planificación y control del proyecto, en especificación precisa de requisitos y modelado, reciben el apelativo de Metodologías Tradicionales o Pesadas. [4]

Estas imponen una disciplina de trabajo sobre el proceso de desarrollo del software, con el fin de conseguir un software más eficiente. Para ello, se hace énfasis en la planificación total de todo el trabajo a realizar y una vez que está todo detallado, comienza el ciclo de desarrollo del software. Se centran especialmente en el control del proceso, mediante una rigurosa definición de roles, actividades, artefactos, herramientas y notaciones para el modelado y documentación detallada. Además, las metodologías tradicionales no se adaptan adecuadamente a los cambios, por lo que no son apropiadas cuando se trabaja en un entorno, donde los requisitos no pueden predecirse o bien pueden variar. Algunas de estas metodologías son: RUP (Rational Unified Procces), MSF (Microsoft Solution Framework), Win-Win Spiral Model, Iconix, entre otras.

1.3.3 Selección de la metodología

El acelerado desarrollo de software en la actualidad y la necesidad de que los proyectos sean concluidos exitosamente siendo un producto de gran valor para los clientes, generan grandes cambios en las metodologías adoptadas por los equipos para cumplir sus objetivos, puesto que unas se adaptan mejor que otras al contexto del proyecto brindando mejores ventajas. Debido a ello es de vital importancia la selección de una metodología que ajustada en un equipo cumpla con sus metas, y satisfaga mas allá de las necesidades definidas al inicio del proyecto.

En el momento de seleccionar una metodología para aplicar en la construcción de un sistema es necesario tener en cuenta las características del proyecto y del equipo y ser adaptada al contexto del mismo. Una de las características principales a tener en cuenta es la complejidad del sistema a desarrollar, es decir, es necesario valorar la complejidad del proceso a automatizar, la cantidad de requisitos que deben ser implementados en el sistema y la complejidad y cantidad de información que se maneja en el proceso. La complejidad puede ser alta, media o baja en dependencia de las características antes mencionadas. En el caso del SIGIRMPP por las características propias que presenta el proyecto y

por encontrarse orientado a un organismo oficial como el MPPRIJ, el colectivo de autores acuerda utilizar el Rational Unified Process (RUP) como metodología de desarrollo del software.

1.3.4 ¿Por qué utilizar RUP para desarrollar el SIGIRMPP?

No existen dos proyectos de desarrollo de software que sean iguales. Cada uno tiene prioridades, requerimientos, y tecnologías muy diferentes. Sin embargo, en todos los proyectos, se debe minimizar el riesgo, garantizar la predictibilidad de los resultados y entregar software de calidad superior a tiempo. Rational Unified Process, es una plataforma flexible de procesos de desarrollo de software que ayuda brindando guías consistentes y personalizadas de procesos para todo el equipo de proyecto.

RUP describe cómo utilizar de forma efectiva reglas de negocio y procedimientos comerciales probados en el desarrollo de software para equipos de desarrollo de software, conocidos como “mejores prácticas”. Captura varias de las mejores prácticas en el desarrollo moderno de software en una forma que es aplicable para un amplio rango de proyectos y organizaciones. Es una guía de cómo utilizar de manera efectiva UML. Provee a cada miembro del equipo fácil acceso a una base de conocimiento con guías, plantillas y herramientas para todas las actividades críticas de desarrollo. Además crea y mantiene modelos, en lugar de enfocarse en la producción de una gran cantidad de papeles de documentación.

1.3.5 Rational Unified Process (RUP)

Su objetivo es asegurar la construcción de sistemas de software de alta calidad que satisfagan las necesidades de los usuarios finales y clientes cumpliendo con los cronogramas y presupuestos previstos. Como RUP es un proceso, en su modelación define como sus principales elementos:

Trabajadores (“quién”)	Define el comportamiento y responsabilidades (rol) de un individuo, grupo de individuos, sistema automatizado o máquina, que trabajan en conjunto como un equipo. Ellos realizan las actividades y son propietarios de elementos.
Actividades (“cómo”)	Es una tarea que tiene un propósito claro, es realizada por un trabajador y manipula elementos.

Artefactos (”qué”)	Productos tangibles del proyecto que son producidos, modificados y usados por las actividades. Pueden ser modelos, elementos dentro del modelo, código fuente y ejecutables.
Flujo de actividades (“Cuándo”)	Secuencia de actividades realizadas por trabajadores y que produce un resultado de valor observable.

RUP se caracteriza por ser:

• **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 o flujos de trabajo, haciendo énfasis en algunas de ellas. Las disciplinas propuestas por RUP son: Modelamiento del negocio, Requerimientos, Análisis y diseño, Implementación, Prueba (Testeo), Instalación, Administración del proyecto, Administración de configuración y cambios, además de Ambiente. Cada iteración añade funcionalidades al producto de software o mejora las existentes.

En el anexo 1 se muestran los flujos de trabajo y las iteraciones que define RUP por las cuales debe transitar el desarrollo de un producto software. Además RUP identifica una serie de roles para los trabajadores que realizan las actividades de cada flujo, dentro de los principales roles se encuentran: analista, diseñador, arquitecto, implementador, jefe de proyecto y probador. [5] Los flujos de trabajo proponen la realización de un gran número de actividades y la elaboración de un amplio conjunto de artefactos, que usualmente los proyectos de desarrollo de software se comprometen a desarrollar pero que no realizan en su totalidad debido a la carencia de tiempo y personal. RUP indica que al inicio del proyecto se realice una adecuación de cada flujo de trabajo de manera que se produzcan solo los artefactos y se realicen las actividades que tienen un propósito dentro del proyecto. A continuación se

describen las disciplinas de mayor interés de acuerdo a los roles desempeñados por los autores para la realización del trabajo de diploma.

Disciplina de Modelamiento del negocio

En esta disciplina se describen los procesos de negocio, identificando quiénes participan y las actividades que requieren automatización; es decir persigue comprender la estructura y la dinámica de la organización en la cual se va a implantar el sistema, los problemas actuales de la organización e identificar las mejoras potenciales, así como asegurar que los consumidores, usuarios finales y desarrolladores tengan un entendimiento común de la organización. Es de suma importancia destacar que debido a la falta de estandarización de la información y a que en las CR y en la DGPG no se pueden definir claramente los procesos, los actores ni los trabajadores que intervienen en el negocio; no es posible realizar un modelo del negocio. Por tanto el colectivo de autores decide realizar un **Modelo del Domino**; el cual constituye el principal artefacto generado en esta disciplina a partir del desempeño de los autores en el rol de analistas del negocio.

Un **Modelo del Domino** captura los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan las "cosas" que existen o los eventos que suceden en el entorno en el que trabaja el sistema. [6] El objetivo del modelado del dominio es comprender y describir las clases más importantes dentro del contexto del sistema. El Modelo de Dominio permite comprender los conceptos que utilizan los usuarios, los conceptos con los que trabajan y con los que deberá trabajar nuestra aplicación.

Disciplina Requerimientos

“La parte más difícil en la construcción de sistemas de software es decir precisamente que construir. Ninguna otra parte puede perjudicar tanto el trabajo final si es realizada de forma errónea.” [7]

En esta disciplina los principales roles desempeñados son el de analista de sistema, especificador de casos de usos y el arquitecto. Además se generaran como artefactos la **Especificación de Requisitos**, el **Modelo de Casos de Usos** y las **Realizaciones de los Casos de Usos**.

La **Especificación de Requisitos** contiene:

- ✓ Los **Requisitos Funcionales:** son capacidades o condiciones que el sistema debe cumplir. Estos se mantienen invariables sin importar con que propiedades o cualidades se relacionen.
- ✓ Los **Requisitos No Funcionales:** son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido y confiable.

El **Modelo de Casos de Usos** es un modelo del sistema que contiene actores, casos de uso y sus relaciones.

Las **Realizaciones de los Casos de Usos** son reseñas textuales del caso de uso. Normalmente tienen el formato de una nota o un documento relacionado de alguna manera con el caso de uso, y explica los procesos o actividades que tienen lugar en el caso de uso.

Disciplina Análisis y Diseño

El objetivo de la disciplina Análisis y Diseño es transformar los requisitos de software en un diseño del sistema, desarrollar una arquitectura robusta y adaptar el diseño al ambiente de implementación. La adecuación del SIGIRMPP para la disciplina contempla como artefactos a generar el **Diagrama de Clases del diseño**, **los Diagramas de Secuencias** y el **Modelo de Datos** desempeñando como roles fundamentales el de diseñador y el de arquitecto.

Un **Diagrama de Clases del Diseño** muestra un conjunto de interfaces, colaboraciones y sus relaciones. Incluye la siguiente información: clases, asociaciones y atributos, interfaces, con sus operaciones y constantes, métodos, navegabilidad y dependencias. A diferencia del Modelo del Dominio, un Diagrama de Clases de Diseño muestra definiciones de entidades software más que conceptos del mundo real.

Un **Diagrama de Secuencia** es una representación que muestra, en determinado escenario de un caso de uso, los eventos generados por actores externos, su orden y los eventos internos del sistema. Contiene detalles de implementación del escenario, incluyendo los objetos y clases que se usan para implementar el escenario, y mensajes intercambiados entre los objetos.

El **Modelo de Datos** describe la representación lógica y física de los datos persistentes utilizados por la aplicación. En casos de aplicaciones que usen sistemas gestores de bases de datos relacionales, este modelo debe incluir representaciones para procedimientos almacenados, entre otros.

Disciplina Implementación

En esta disciplina se define la organización del código en subsistemas de implementación. Su objetivo principal es la implementación del diseño, además de la realización de pruebas unitarias a los componentes y la integración de los resultados del trabajo de implementadores individuales en un sistema ejecutable. Como resultado de las actividades realizadas por el implementador se obtiene el software listo para ser probado por terceros, con todas las funcionalidades implementadas y si fuese necesario **Diagramas de Componentes**.

1.4 Unified Modeling Language(UML)

Es un "lenguaje de modelado" para especificar o para describir métodos o procesos. Se utiliza para definir un sistema, para detallar los artefactos en el sistema y para documentar y construir. Se puede aplicar en el desarrollo de software entregando gran variedad de formas para dar soporte a una metodología de desarrollo de software (tal como RUP), pero no especifica en sí mismo qué metodología o proceso usar. No obstante, es necesario además de modelar el software, definir las herramientas y tecnologías que deben utilizarse para la confección del software. [8]

1.5 Programación orientada a objetos.

La Programación Orientada a Objetos (POO) es un paradigma de programación que expresa un programa como un conjunto de objetos que colaboran entre ellos para realizar tareas. Un objeto es una entidad provista de un conjunto de propiedades o atributos y de comportamiento o funcionalidad que representa a los objetos, procesos o conceptos del mundo real. Junto a su definición se relacionan un conjunto de claves dentro de su significado como lo son: **clase** (definición de las propiedades y comportamiento de un conjunto de objetos concretos) e **instancia** (lectura de estas definiciones y la creación de un objeto a partir de ellas).

La POO no rompe con el paradigma secuencial, estructural y otros anteriores, sino que contiene elementos de las mejores prácticas de estos y agrega nuevos beneficios. Ejemplo de ello, utilizado con frecuencia en el SIGIRMPP es la reutilización y la extensión del código. Además, permite agilizar el desarrollo de software y facilitar el trabajo en equipo y el mantenimiento del software. [9] En sus pocas décadas de desarrollo, la POO ha demostrado ser imprescindible para el manejo de la complejidad creciente de los sistemas informáticos actuales. En el SIGIRMPP se utiliza como lenguaje Java, exponente puro de este paradigma.

1.6 Plataforma de desarrollo

J2EE

Es un estándar de la industria para desarrollar aplicaciones empresariales portables, robustas, escalables y seguras utilizando Java como lenguaje. J2EE define una arquitectura para desarrollar aplicaciones distribuidas complejas utilizando un modelo multicapas. La lógica de aplicación está dividida en componentes de acuerdo a su función. Estos pueden estar instalados en diferentes nodos en correspondencia con la capa a la que pertenezcan. Componentes de la capa cliente se ejecutan en la máquina cliente, los componentes web y de la capa de negocio se ejecutan en el servidor de aplicaciones y otros componentes en sistemas legados o servidores de bases de datos.

J2EE consiste en:

• **Una especificación:** define los requisitos que debe cumplir una implementación de un producto J2EE.

• **Un Modelo de Programación:** guía de diseño para desarrollar aplicaciones J2EE.

• **Una plataforma:** conjunto de APIs[1], tecnologías y herramientas de desarrollo.

[1]API, del inglés Application Programming Interface: 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.

• **Una implementación de referencia:** Implementación de ejemplo de los servicios brindados por la plataforma.

• **Una Suite de pruebas de compatibilidad:** certifica la compatibilidad de un producto con la especificación J2EE a través de varios tipos de pruebas.

La plataforma J2EE reduce significativamente el esfuerzo necesario por los desarrolladores, proveyendo una robusta arquitectura que de otra forma tendrían que implementar ellos mismos. De esta manera los desarrolladores se pueden concentrar solo en la implementación de componentes que satisfagan las necesidades propias del negocio. Entre las APIs de la plataforma J2EE utilizadas en el desarrollo del SIGIRMPP se encuentran: Java-Servlets, Java Server Pages y JDBC [10].

1.7 Java-Servlets

Permite extender las capacidades de un servidor de aplicaciones que es accesible a través del modelo petición-respuesta. Generalmente es utilizado en ambientes web como es el caso del SIGIRMPP. Los servlets proveen un mecanismo efectivo de interacción entre la lógica de negocio que se ejecuta en un servidor y las aplicaciones clientes. Los servlets viven en un contenedor de servlets que se ejecuta en un servidor web. Este contenedor gestiona su ciclo de vida y traduce las peticiones, que un cliente web hace a través del protocolo HTTP[2], en objetos que las encapsulan. De igual forma, el contenedor traduce las respuestas de los servlets al protocolo web correspondiente. [11]

[2]HTTP, del inglés HyperText Transfer Protocol: Es el protocolo sin estado que define la sintaxis y la semántica que utilizan los elementos de software de la arquitectura web (clientes, servidores, proxies) para comunicarse. Es un protocolo orientado a transacciones y sigue el esquema petición-respuesta entre un cliente y un servidor.

1.8 Java Server Pages (JSP)

La tecnología JSP evolucionó de Java-Servlets, de hecho, una JSP se compila en una especie de servlet que es ejecutado en un contenedor de servlets. Una JSP es un documento texto que tiene contenido estático (texto plano, HTML, XML) y elementos que determinan cómo la página será construida dinámicamente. Las JSPs separan la lógica de presentación de la lógica de aplicación, promoviendo la reutilización y la separación de roles en los equipos de desarrollo. La mayoría de las vistas que se muestran al usuario en el SIGIRMPP son documentos HTML generados a partir de la tecnología JSP. [12]

1.9 JDBC

Brinda una interfaz para el acceso a bases de datos relacionales. JDBC generaliza las funciones más comunes de acceso a los datos, abstrayendo los detalles específicos de un proveedor de determinada base de datos. El resultado es un conjunto de clases e interfaces, que pueden ser utilizadas con cualquier gestor que tenga un controlador JDBC apropiado. [13] Esta API es utilizada por la mayoría de los frameworks de persistencia como Hibernate.

1.10 Frameworks

Un framework es una mini arquitectura reusable que provee una estructura y comportamiento genéricos para una familia de abstracciones de software [14]. Es un conjunto de componentes con interfaces bien definidas que interactúan entre sí para cumplir una tarea. La GoF[3] define un framework como “un conjunto de clases que constituyen un diseño reutilizable para un tipo específico de aplicaciones”. Un framework no tiene funcionalidades de una aplicación específica, sino que las aplicaciones se construyen sobre ellos. Para usar un framework es necesario personalizarlo, extendiéndolo o componiendo las distintas instancias de sus componentes e insertando las funcionalidades específicas de la aplicación en ciertos puntos que el

[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.

framework provee con ese fin. Luego el framework funciona solo invocando las rutinas específicas de la aplicación.

En el desarrollo de aplicaciones similares al SIGIRMPP, el uso de frameworks se ha convertido en algo verdaderamente importante porque implica ahorro en tiempo de diseño y código, puesto que implementa un conjunto de patrones de diseño, lo que permite su reutilización como componentes de software y a su vez suelen implementar las partes más engorrosas y difíciles del dominio del problema permitiendo que el desarrollador se concentre en implementar las tareas propias de la aplicación. Existen varias clasificaciones para los frameworks de acuerdo a diferentes criterios: [14]

• **Frameworks de aplicación:** Proveen un amplio rango de funcionalidades típicamente usadas en una aplicación. Interviene en varias capas de la aplicación como Interfaz de Usuario, Acceso a Datos, entre otros.

• **Frameworks de dominio:** Proporciona funcionalidades para un dominio específico de aplicación.

• **Frameworks de soporte:** Se dirigen a dominios muy específicos dentro de una aplicación como manejo de memoria, reportes, entre otros.

Sobre la plataforma J2EE existen un conjunto amplio de frameworks que utilizan e integran las APIs que esta brinda. La mayoría de estos frameworks constituyen referencias para otras plataformas.

En el desarrollo de la arquitectura base del SIGIRMPP se utilizó el framework de aplicación Spring que al igual que el framework de soporte Hibernate son muy populares, por sus múltiples ventajas, en el desarrollo de aplicaciones web dentro de la plataforma J2EE. A continuación se presentan algunos aspectos relacionados con los frameworks usados en el desarrollo del sistema.

1.10.1 Framework Spring

Spring es un framework basado en la Inversión de Control[4] y en la Programación Orientada a Aspectos[5]. Se distribuye de forma libre y su código es abierto. El costo de utilizar Spring en términos de rendimiento tanto para la aplicación que se realiza como para el sistema y hardware que lo soporta es despreciable puesto que su consumo de recursos es mínimo. Es no intrusivo, o sea, las clases de una aplicación basada en Spring generalmente no dependen de las clases específicas del framework. Por estas características se clasifica como un framework ligero.

Permite configurar complejas aplicaciones a partir de componentes simples. En Spring los objetos de la aplicación se declaran en ficheros (normalmente en formato xml) y el framework se encarga de instanciarlos y configurarlos correctamente a través de la inyección de dependencias. Mediante esta técnica los objetos reciben pasivamente sus dependencias sin necesidad de crearlas o buscarlas, proporcionando un bajo acoplamiento entre los componentes de la aplicación. Spring provee soporte para la programación orientada a aspectos permitiendo separar la lógica de la aplicación de los servicios de sistemas como la auditoría y la gestión de transacciones. Esta ventaja facilita que la implementación se centre en el negocio y no en otros tipos de servicios. [15]

1.10.2 Módulos de Spring

Spring está dividido en módulos bien definidos (Figura 1), pero no obliga a hacer uso de todos ellos. Se puede elegir los módulos que se necesiten en el caso específico y buscar otras opciones cuando Spring no satisfaga los requisitos. A través de ellos brinda puntos de extensión con varios de los frameworks y bibliotecas de la plataforma J2EE como Hibernate, JMS, JMX, RMI y Jax-RPC. La estructura de los

[4]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.

[5]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.

módulos de Spring, tal como se muestran en la figura, tiene como base el Core que contiene las funcionalidades fundamentales del framework. Sobre el Core se desarrollan los restantes módulos que a su vez pueden constituir la base de otros, por ejemplo los módulos Web y ORM.

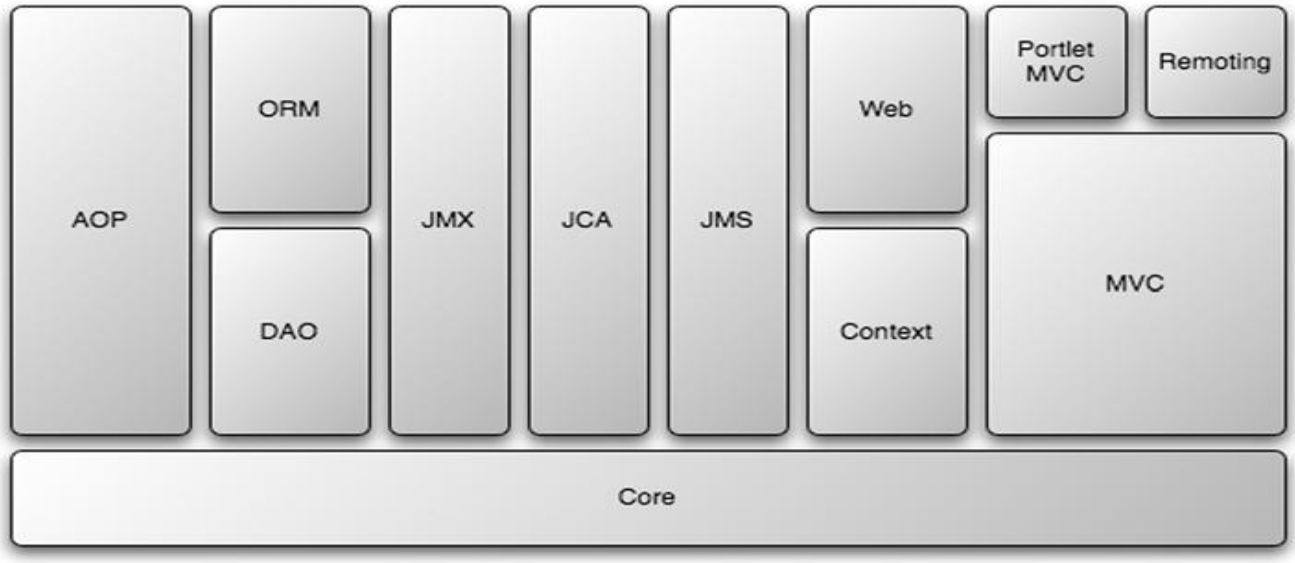


Figura 1: Módulos del Framework Spring

De todos sus módulos, los más utilizados en la implementación del SIGIRMPP son:

• **Spring-ORM:** El módulo provee una forma conveniente para construir la capa de acceso a datos basados en el patrón DAO y se integra a las soluciones ORM[6] como Hibernate, Java Persistence API, Java Data Objects, Apache OJB, JDO, iBATIS SQL Maps y Oracle's Top Link. Brinda además algunos servicios, como la integración con el mecanismo de transacciones declarativas de Spring y el manejo transparente de excepciones. Proporciona integración con varias implementaciones ORM. Existen dos formas de integración, a través de plantillas predefinidas o codificando DAOs directamente contra el API del ORM elegido. Cualquiera de las dos aproximaciones ofrece los beneficios de Spring, como ser

[6]ORM, del inglés Object Relacional Mapping: 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.

configurados a través de IoC, transaccionalidad, wrapping común para excepciones de acceso a datos y manejo de la configuración independiente de la implementación. [15]

• **Spring-MVC:** El modelo MVC es muy utilizado en la construcción de aplicaciones web. Spring se integra con frameworks de soporte MVC como Struts, JSF, WebWork y Trapestry, pero también provee su propia implementación del modelo MVC basada en el patrón Controlador Frontal. [15]

1.10.2 Framework Hibernate

Hibernate es un framework ORM para la plataforma Java; distribuido bajo los términos de la licencia GNU LGPL ha ganado popularidad como herramienta de soporte a la capa de acceso a datos en el desarrollo de aplicaciones empresariales. Provee mapeo objeto-relacional básico, y otras características sofisticadas como caché y caché distribuida, carga perezosa y ávida.

Como todas las herramientas ORM, Hibernate busca solucionar el problema de la diferencia entre dos modelos ampliamente utilizados para organizar y manipular datos: el orientado a objetos en las aplicaciones y el relacional en las bases de datos. Para lograr esto el desarrollador debe especificar a Hibernate cómo es su modelo de datos. Con esta información Hibernate permite manipular los datos desde las aplicaciones operando sobre objetos con todas las características de la POO. Hibernate convierte los datos que define Java a los que define SQL, siendo transparente esta conversión para el desarrollador. Genera además las sentencias SQL y libera al desarrollador del manejo manual de los datos que resultan de la ejecución de dichas sentencias. También ofrece un lenguaje de consulta de datos llamado HQL (Hibernate Query Language), y al mismo tiempo APIs para construir las consultas programáticamente. Hibernate elimina la necesidad de parte del código de acceso a los datos, permitiendo que el desarrollador se concentre más en la lógica de negocio. Esta reducción de código representa un aumento de la productividad y permite que la capa de acceso a datos sea más entendible y fácil de mantener. A continuación se presenta los aspectos relacionados con la persistencia de los datos y su gestión [16]

1.11 Gestor de Base de Datos

1.11.1 PostgreSQL

PostgreSQL (8.3) es un sistema de gestión de base de datos relacional orientado a objetos y libre. Este gestor se caracteriza principalmente por poseer:

Alta concurrencia

Mediante un sistema denominado MVCC (Acceso concurrente multiversión) PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Cada usuario obtiene una visión consistente de lo último a lo que se le hizo commit. Esta estrategia es superior al uso de bloqueos por tabla o por filas común en otras bases, eliminando la necesidad del uso de bloqueos explícitos.

Amplia variedad de tipos nativos

PostgreSQL provee nativamente soporte para: números de precisión arbitraria, texto de largo ilimitado, figuras geométricas (con una variedad de funciones asociadas), direcciones IP (IPv4 e IPv6), bloques de direcciones estilo CIDR[7], direcciones MAC y arrays. Adicionalmente los usuarios pueden crear sus propios tipos de datos, los que pueden ser por completo indexables gracias a la infraestructura GiST de PostgreSQL.

[7]Classless Inter-Domain Routing (CIDR Encaminamiento Inter-Dominios sin Clases) se introdujo en 1993 y representa la última mejora en el modo como se interpretan las direcciones IP. Su introducción permitió una mayor flexibilidad al dividir rangos de direcciones IP en redes separadas. De esta manera permitió un uso más eficiente de las cada vez más escasas direcciones IPv4 y un mayor uso de la jerarquía de direcciones ('agregación de prefijos de red'), disminuyendo la sobrecarga de los enrutadores principales de Internet para realizar el encaminamiento.

Por otra parte los bloques de código que se ejecutan en el servidor. Pueden ser escritos en varios lenguajes, con la potencia que cada uno de ellos da, desde las operaciones básicas de programación, tales como bifurcaciones y bucles, hasta las complejidades de la programación orientada a objetos o la programación funcional. [17] [18]

Por las características mencionadas el colectivo de autores del presente trabajo decide usar este potente motor de bases de datos, que tiene prestaciones y funcionalidades equivalentes a muchos gestores de bases de datos comerciales. Decir que es más completo que MySQL porque permite métodos almacenados, restricciones de integridad, vistas, etc. aunque en las últimas versiones de MySQL se han hecho grandes avances en ese sentido. Sin embrago este solo cuenta con el Motor de Datos y un número pequeño de utilidades, para potenciar el trabajo con PostgreSQL suele ser necesario añadir utilidades externas creadas especialmente para este motor, en el caso del presente trabajo se determinó usar EMS SQL Manager for PostgreSQL (SQL Manager 2007 for PostgreSQL 4.3.0.1).

1.11.2 EMS SQL Manager for PostgreSQL

Es una aplicación de alto desempeño para la administración y desarrollo de PostgreSQL Database Server. El programa trabaja con cualquier versión de PostgreSQL hasta la 8.4 y soporta todas las últimas características de PostgreSQL, incluyendo espacios de tablas, nombres de argumentos en funciones y más. El programa ofrece muchas herramientas poderosas para usuarios experimentados, como un Diseñador Visual de Bases de Datos, Constructor Visual de Consultas y un editor BLOB. Su interfaz gráfica es sumamente atractiva e incluye un modo guiado de trabajo. Entre sus características se encuentran la administración y navegación rápida de bases de datos, administración fácil de todos los objetos PostgreSQL, herramientas de manipulación avanzada de datos, excelentes herramientas visuales y de texto para la construcción de consultas, capacidades de exportación e importación de datos, un poderoso diseñador visual de bases de datos, un modo guiado para labores de mantenimiento y una interfaz atractiva. [19]

Independientemente del gestor de la base de datos, se hace imprescindible utilizar herramientas para la creación del sistema con el cual interactuará finalmente el usuario.

1.12 Entorno integrado de desarrollo (IDE[8])

Eclipse

Eclipse (3.5) es una plataforma libre sobre la cual se pueden acoplar herramientas de desarrollo de todo tipo mediante la implementación de plug-ins. Una de las características más importantes de Eclipse es su facilidad de extensión. El IDE Eclipse es una de las herramientas que se engloban bajo el denominado Proyecto Eclipse. El Proyecto Eclipse incorpora tanto el desarrollo del IDE Eclipse como de algunos de los plug-ins más importantes como el JDT (Java Development Tools), plug-in para el lenguaje Java, y el CDT (C-C++Development Tooling), plug-in para el lenguaje C/C++. El IDE Eclipse es gratis y de código abierto pero sus plug-ins pueden no serlo dadas las características de su licencia "no viral". [20] El sistema de plug-ins agiliza el trabajo del equipo de desarrollo en la implementación. Algunos de los plug-ins más utilizados en el SIGIRMPP son:

• **Hibernate Tools:** Constituye un conjunto de herramientas para facilitar el uso del framework Hibernate. Las principales funcionalidades que brinda son: un editor de mapeos de los metadatos de la base de datos a las clases y una consola para la ejecución de consultas HQL. Permite realizar ingeniería inversa a partir de la base de datos de las clases y de los mapeos de las entidades.

• **Spring IDE:** Agiliza el trabajo con el framework Spring, principalmente en proyectos grandes, puesto que contiene un editor XML que permite autocompletado y un validador y visor de los beans configurados. Permite la búsqueda y visualización en forma de grafo de todos los beans y sus dependencias.

[8] 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.

1.13 Bibliotecas

1.13.1 Jasper Reports

Librería gratis y de código abierto que 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. [21]

Esta librería se utiliza en el SIGIRMPP para la generación de informes de los programas, proyectos y otros.

1.13.2 Dojo Toolkit

Es una biblioteca Java Script[9]de código abierto. Resuelve problemas comunes y engorrosos en el desarrollo de aplicaciones con Java Script como, por ejemplo, la disparidad del modelo de eventos de los distintos navegadores. Provee un conjunto de componentes de interfaz gráfica de usuario como calendarios y menús, que se pueden insertar de manera sencilla en páginas HTML. Estos componentes son utilizados en las interfaces del SIGIRMPP, logrando agilizar el desarrollo al reutilizar código existente con un alto grado de terminación. [22]

1.14 Herramienta de modelado

Visual Paradigm

Visual Paradigm es una suite completa de herramientas CASE[10]. Independiente de la plataforma y dotada de una buena cantidad de productos o módulos para facilitar el trabajo durante la confección de un

[9]Java Script: Lenguaje de programación interpretado utilizado principalmente para la incorporación de comportamiento dinámico a documentos HTML.

[10]Herramienta CASE, del inglés Computer Aided Software Engineering: Aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el costo del mismo en tiempo y dinero.

software y garantizar la calidad del producto final. Algunos de los diagramas que permite generar son de casos de uso, clases, secuencia, comunicación, estado, componentes, despliegue, objetos, interacción, entidad relación, ORM, procesos del negocio y visión general.

Visual Paradigm en su modalidad **Visual Paradigm for UML 6.1 Enterprise Edition**, es muy factible para la realización de los diagramas anteriores y para generar la base de datos a partir de diagramas entidad relación y ORM; por lo que el colectivo de autores del presente trabajo decidió utilizar esta herramienta de modelado. [23]

1.15 Conclusiones.

SIGIRMPP es una aplicación web de gestión de información orientado al MPPRIJ. Basado en este hecho y en las apropiadas características que posee; se utiliza como metodología de desarrollo de software una adecuación para el proyecto de la metodología RUP. Los flujos de trabajo sobre los que se basa la realización del presente trabajo son Modelamiento del negocio, Requerimientos, Análisis y Diseño e Implementación.

Las tecnologías a utilizar contienen tendencias recientes de la programación sobre la plataforma J2EE como la programación orientada a aspectos y la inyección de dependencias. Además los frameworks utilizados se consideran de referencia en su ámbito como lo es Spring como framework de aplicación e Hibernate como framework de soporte para la capa de acceso a datos. El IDE Eclipse con su sistema de plug-ins es una herramienta provechosa pues agiliza del trabajo de implementación con estos frameworks y otros abordados en el capítulo.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

2.1 Introducción.

En el presente capítulo se describe la solución para la problemática planteada. Se presenta el modelo de dominio, con la descripción de sus clases; la especificación de los requisitos y los modelos de Casos de Uso del Sistema. Además se presentan las realizaciones de los casos de uso correspondientes y los patrones de Casos de Usos que se utilizaron.

2.2 Breve descripción del sistema.

El sistema a desarrollar a través de la presente investigación constituye un sistema de gestión de información en forma de aplicación web para la manipulación de los datos necesarios de parte de las CR y la DGPD. Para el desarrollo del mismo no fue posible identificar claramente los procesos involucrados en el negocio por lo que se realiza un modelo del dominio. Además es necesario destacar que el sistema se compone por dos subsistemas: uno perteneciente a la gestión de los programas – proyectos y el otro concerniente a los recursos materiales.

2.3 Modelo del dominio.

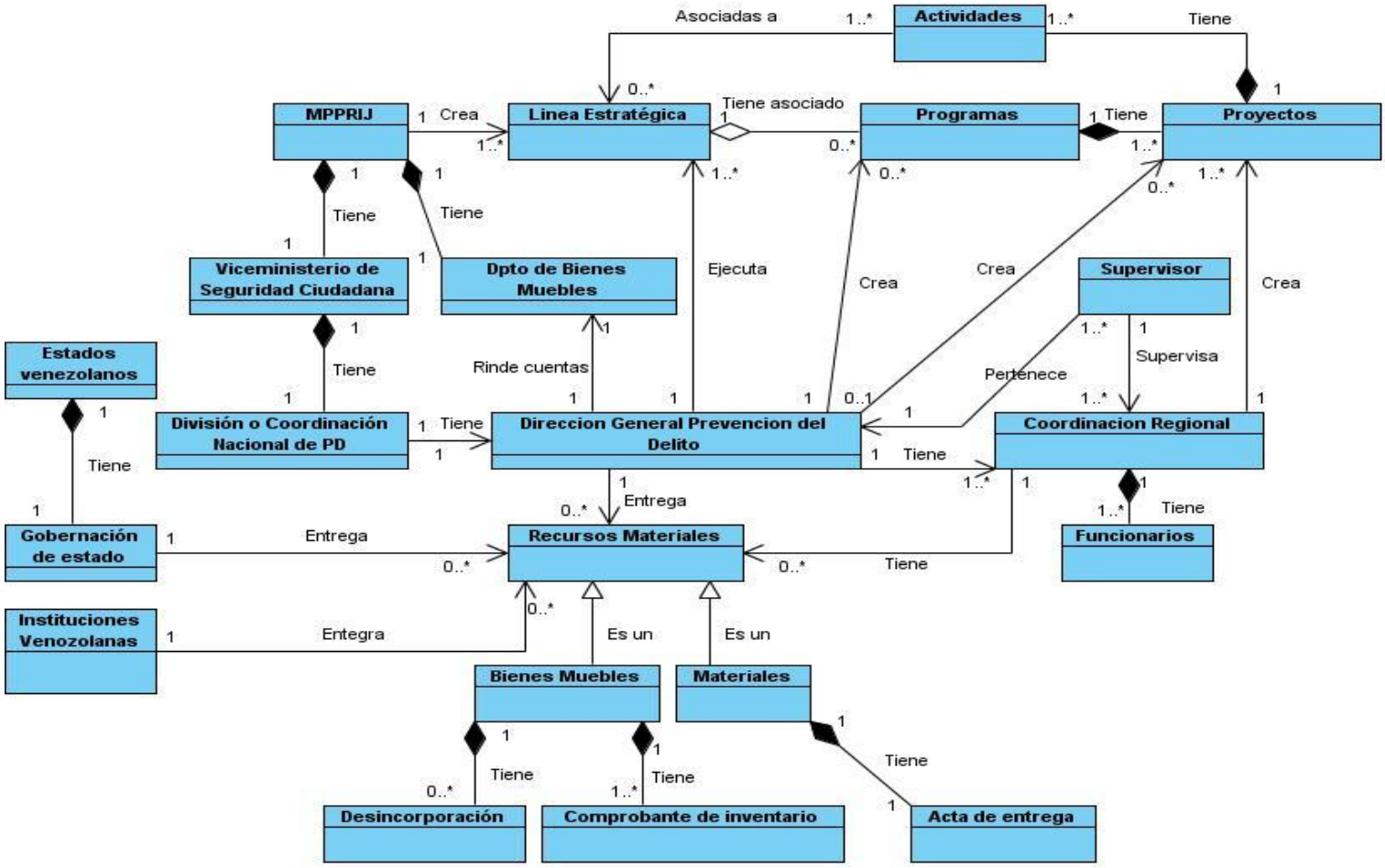


Figura 2: Modelo de Dominio.

2.3.1 Descripción de las clases

1- Clase MPPRIJ:

Ministerio del Poder Popular para Relaciones Interiores y Justicia de la República Bolivariana de Venezuela.

2- Clase Viceministerio de seguridad ciudadana:

Organismo adscrito al MIPPRIJ.

3- Clase División o Coordinación General de Prevención del Delito:

Organismo adscrito al Viceministerio de Seguridad Ciudadana.

4- Clase Dirección General de Prevención del Delito:

Adscrita al Viceministerio de Seguridad Ciudadana del Ministerio del Poder Popular para Relaciones Interiores y Justicia de la República Bolivariana de Venezuela. Esta dirección planifica la política antidelictiva, cuyas funciones versan sobre el diseño y ejecución de política, planes y programas orientados a la prevención de la violencia y la criminalidad a través del ejercicio de la corresponsabilidad entre instituciones y comunidades y del fortalecimiento de la convivencia ciudadana.

5- Clase Coordinación Regional:

Es una oficina adscrita a la Dirección General de Prevención del Delito, conformada por un equipo de profesionales y técnicos que se encargan de la atención e implementación de los programas de prevención del delito, como el Programa de Educación y El Proyecto de la Comunidad.

6- Clase Líneas Estratégicas:

La DGPR hasta el momento tiene definidas 11 líneas estratégicas para guiar el trabajo en función de la prevención del delito. Ejemplo de estas líneas son: Plan Nacional de Prevención contra la corrupción, Plan Nacional la Pornografía Infantil, Plan Nacional contra la Violencia de Género, plan contra el Consumo de Especies Alcohólicas y sustancias Estupefacientes y Psicotrópicas, etc.

7- Clase Proyectos:

Un proyecto es un conjunto de actividades con un fin específico basado en las líneas estratégicas y en programas.

8- Clase Actividades:

Acciones realizadas por las CR tales como: participar en las mesas de Seguridad Ciudadana, crear acciones preventivas, poner en marcha los convenios adquiridos en materia prevención, talleres, mítines, trabajos sociales, reuniones con las comunidades, mesas de trabajo y otras que son destinadas a incorporar a los miembros de la comunidad, en especial al sector juvenil. Otros ejemplos de actividades son capacitaciones a instituciones educativas, organismos de seguridad ciudadana y comunidades sobre la prevención de la delincuencia-violencia, consumo de drogas, lucha contra pornografía infantil, etc.

9- Clase Programa:

Un programa es el resultado de una concepción filosófica, ideológica, funcional y operativa de lo que el Estado define como su "Función Social".

10- Clase Desincorporación:

La desincorporación puede venir motivada por tres motivos: el uso, el paso del tiempo y la obsolescencia.

11 Clase Funcionario:

Trabajador o persona directamente vinculada a la CR que realiza actividades y procesos de interés para la CR.

12- Clase Material:

Elementos tangibles de carácter desechable (bate, pelotas, libros, entre otros) que se le entregan a la CR para realizar sus actividades.

13- Clase Bien mueble:

Recursos con que cuenta la comunidad para el logro de sus actividades y procesos. Estos se van desincorporando debido al uso de los mismos u otros factores. (locales, medios de transporte, entre otros.)

14- Clase Estados Venezolano:

División política geográfica de la República Bolivariana de Venezuela.

15- Clase Gobernación de Estado:

Ente gubernamental que dispone de una asamblea legislativa y un gobernador.

16- Clase Instituciones Venezolanas:

Ministerio de Educación, Salud, entre otros.

17- Clase Departamento de Bienes Muebles:

Organismo adscrito al MIPPRIJ, al cual le rinde cuenta la DGPD en lo concerniente a los recursos materiales.

18- Clase Comprobante de Inventario:

Es un documento oficial que acredita la transferencia de bienes, la entrega en uso o la prestación de servicios.

19- Clase Actas de Entrega:

Es un documento no oficial que acredita la transferencia de bienes, la entrega en uso o la prestación de servicios.

20- Clase Recursos Materiales:

Comprende los bienes muebles y materiales que le son entregados a la CR para el desarrollo de sus actividades.

21 -Clase Supervisora:

Personal que trabaja en la Dirección General de Prevención del Delito, encargado de supervisar las coordinaciones regionales.

2.4 Requisitos Funcionales

A continuación se muestran los requisitos funcionales del sistema:

- ✓ R1 Visualizar listado de materiales
- ✓ R2 Filtrar listado de materiales
- ✓ R3 Registrar entrada de materiales
- ✓ R4 Registrar salida de materiales
- ✓ R5 Actualizar material
- ✓ R6 Visualizar entradas y salidas de un material
- ✓ R7 Imprimir entradas y salidas de un material
- ✓ R8 Visualizar historial de operaciones por materiales
- ✓ R9 Filtrar historial de operaciones por materiales
- ✓ R10 Generar reporte historial de operaciones por materiales
- ✓ R11 Visualizar historial de operaciones
- ✓ R12 Filtrar historial de operaciones
- ✓ R13 Generar reporte historial de operaciones
- ✓ R14 Visualizar entrada del material
- ✓ R15 Visualizar salida del material
- ✓ R16 Actualizar entrada del material
- ✓ R17 Actualizar salida del material
- ✓ R18 Imprimir entrada del material
- ✓ R19 Imprimir salida del material
- ✓ R20 Exportar un reporte a un formato predefinido
- ✓ R21 Adjuntar Archivo
- ✓ R22 Calcular cantidad de material
- ✓ R23 Visualizar listado de bienes muebles
- ✓ R24 Filtrar listado de bienes muebles
- ✓ R25 Generar reporte del listado de bienes muebles
- ✓ R26 Registrar entrada de bienes muebles
- ✓ R27 Calcular cantidades del bien mueble
- ✓ R28 Registrar Desincorporación de bienes muebles
- ✓ R29 Actualizar Desincorporación de bienes muebles
- ✓ R30 Actualizar datos del bien mueble
- ✓ R31 Visualizar datos del bien mueble
- ✓ R32 Imprimir datos del bien mueble
- ✓ R33 Visualizar historial de operaciones por bien mueble
- ✓ R34 Filtrar historial de operaciones por bien mueble
- ✓ R35 Generar reporte historial de operaciones por bien mueble
- ✓ R36 Actualizar entrada del bien mueble
- ✓ R37 Visualizar entrada del bien mueble
- ✓ R38 Imprimir entrada del bien mueble
- ✓ R39 Visualizar Desincorporación de bienes muebles
- ✓ R40 Imprimir Desincorporación de bienes muebles

- ✓ R41 Generar informe de comprobante de inventario
- ✓ R42 Registrar nuevo proyecto
- ✓ R43 Actualizar proyectos
- ✓ R44 Visualizar datos del proyecto
- ✓ R45 Visualizar listado de proyectos
- ✓ R46 Filtrar listado de proyectos
- ✓ R47 Generar reportes de proyectos
- ✓ R48 Imprimir datos del proyecto
- ✓ R49 Adicionar proyectos de la dirección general
- ✓ R50 Registrar nuevo programa
- ✓ R51 Registrar línea estratégica
- ✓ R52 Actualizar programas
- ✓ R53 Visualizar datos del programa
- ✓ R54 Visualizar listado de programas
- ✓ R55 Filtrar listado de programas
- ✓ R56 Generar reportes de programas
- ✓ R57 Imprimir datos del programa
- ✓ R58 Asociar programa de la DGPD.
- ✓ R60 Autenticar usuario
- ✓ R61 Registrar usuario
- ✓ R62 Actualizar usuario
- ✓ R63 Eliminar usuario
- ✓ R64 Cambiar contraseña
- ✓ R65 Visualizar listado de usuarios
- ✓ R66 Registrar funcionario
- ✓ R67 Actualizar funcionario
- ✓ R68 Eliminar funcionario
- ✓ R69 Visualizar listado de funcionarios
- ✓ R70 Visualizar archivo
- ✓ R71 Eliminar archivo
- ✓ R72 Visualizar listado de adjuntos
- ✓ R73 Visualizar listado de líneas estratégicas
- ✓ R74 Actualizar línea estratégica
- ✓ R75 Eliminar línea estratégica

2.5 Requisitos No funcionales.

Los requisitos no funcionales del sistema se detallan a continuación, agrupados según las diferentes categorías que estos presentan. Es importante aclarar que los requerimientos que se exponen en este documento tributan a la tesis de arquitectura “Propuesta de arquitectura para el Sistema de Gestión de Información para las Coordinaciones Regionales de Prevención del Delito”.

RNF1 - Requerimientos de Apariencia o interfaz externa

Se deben utilizar imágenes y colores identificados con el negocio del sistema.

RNF2 - Requerimientos de Usabilidad

Los usuarios deben tener conocimientos básicos de informática por lo que el sistema debe permitir a los usuarios un acceso fácil y rápido, contando con un menú que satisfaga las necesidades de los mismos.

RNF3 - Requerimientos de soporte

Una vez terminado el desarrollo del software se deben tomar decisiones con motivo de asistir a los clientes y para lograr un mejoramiento progresivo y evolutivo, por lo que se plantean como requerimientos de soporte:

- ✓ Reinstalación de aplicaciones ante fallos.
- ✓ Instalación de nuevas versiones o actualizaciones del sistema.
- ✓ Solución de fallos de configuración de la tecnología asociada al proyecto.

RNF4 - Requerimientos de Seguridad

Confidencialidad

La autenticación será la primera acción del usuario en el sistema y consistirá en proveer un nombre de usuario único y una contraseña que debe ser de conocimiento exclusivo de la persona que se autentica.

Integridad:

Se debe garantizar que la información sensible sólo pueda ser vista por los usuarios con el nivel de acceso adecuado y que las funcionalidades del sistema se muestren de acuerdo al usuario que esté activo.

Disponibilidad

El Sistema de Gestión de Información debe estar disponible para su utilización las 24 horas del día, durante los siete días de la semana, con el menor tiempo posible de recuperación ante fallos.

RNF5 - Requerimientos Legales

La licencia para la comercialización del producto se emitirá por la Dirección de Servicios Legales de la Infraestructura Productiva, para la misma se tendrá en cuenta los componentes que se utilizaron para el desarrollo de la aplicación.

RNF6 - Requerimientos de software

Cliente

- ✓ Debe tener instalado un navegador web. Se recomienda Mozilla Firefox 3.5 o Google Chrome 4.0.223 o superior.
- ✓ Debe tener instalado Sistema Operativo: Windows XP /Vista/ 7 o GNU Linux.

Servidor

- ✓ Se debe instalar TOMCAT versión 6.0.14 o superior como servidor web y PostgreSQL versión 8.3 o superior como gestor de base de datos.
- ✓ Debe estar instalado el Java Runtime Environment (JRE) versión 1.6 o superior.
- ✓ Debe estar instalado Sistema Operativo: Windows XP /Vista/ 7o GNU Linux.

RNF7 - Requerimientos de hardware

Para el funcionamiento del sistema se requiere de máquinas con los siguientes requisitos:

- ✓ Para las PC clientes: procesador Pentium o superior, 256 MB de RAM y al menos 200 MB de capacidad del disco duro.
- ✓ Para el servidor de aplicaciones: capacidad de disco duro superior a 60 GB, microprocesador Pentium IV superior a 2.0 GHz y como mínimo 1.0 GB de RAM.
- ✓ Para el servidor de base de datos: capacidad de disco duro superior a 60 GB, microprocesador Pentium IV superior a 2.0 GHz y como mínimo 1.0 GB de RAM.

RNF8 - Requerimientos de restricciones en el diseño y la implementación

Los componentes del sistema deben desarrollarse siguiendo el principio de alta cohesión y bajo acoplamiento.

La lógica de presentación constituirá una capa independiente de la lógica de negocio, centrando su función en la interfaz de usuario y validaciones de los datos de entrada.

2.6 Patrones de Casos de usos.

Un patrón es una pareja de problema / solución con un nombre, que codifica (estandariza) buenos principios y sugerencias relacionados frecuentemente con la asignación de responsabilidades. [24] En el presente trabajo de diploma los autores se limitan a describir solo aquellos patrones que se utilizaron en la solución del problema planteado.

2.6.1 Concordancia (Commonality)

Extrae una subsecuencia de acciones que aparecen en diferentes lugares del flujo de casos de uso y es expresado por separado.

Adición

En el caso de este patrón alternativo, la subsecuencia común de casos de uso, extiende los casos de uso compartiendo la subsecuencia de acciones. Los otros casos de uso modelan el flujo que será expandido con la subsecuencia. Este patrón es preferible usarlo cuando otros casos de uso se encuentran propiamente completos, o sea, que no requieren de una subsecuencia común de acciones para modelar los usos completos del sistema. [24]

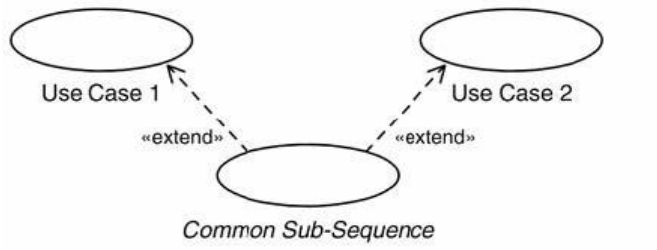


Figura 3: Patrón de Concordancia (Adición)

A continuación se muestra un fragmento del Diagrama de Casos de Usos del subsistema Recursos materiales donde se evidencia el uso de este patrón. En este diagrama el Caso de Uso **Generar reportes de bienes muebles** (Caso de Uso extendido) describe un comportamiento opcional a partir del Caso de Uso **Visualizar listado de bienes muebles** (Caso de Uso base), es decir, al visualizar el listado se puede o no escoger la opción de generar reportes.

Figura 3.1: Patrón de Concordancia (Adición)

2.6.2 CRUD (Creating, Reading, Updating, Deleting)

Este patrón se basa en la fusión de casos de uso simples para formar una unidad conceptual.

Completo

Este patrón consta de un caso de uso, llamado ***Información CRUD*** o ***Gestionar información*** modela todas las operaciones que pueden ser realizadas sobre una parte de la información de un tipo específico, tales como creación, lectura, actualización y eliminación. Suele ser utilizado cuando todos los flujos contribuyen al mismo valor del negocio, y estos a su vez son cortos y simples. [24]

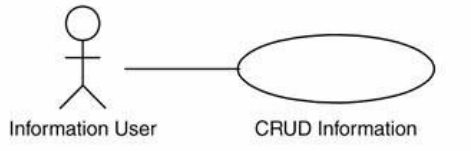


Figura 4: Patrón CRUD Completo

A continuación se muestra un fragmento del Diagrama de Casos de Usos del subsistema Recursos materiales donde se evidencia el uso de este patrón. En este Caso de Uso se incluyen las operaciones **adicionar usuarios**, **modificar usuarios**, **eliminar usuarios** y **visualizar usuarios.**

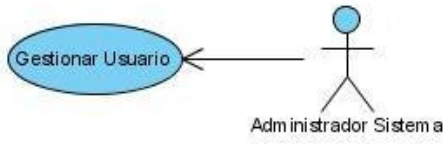


Figura 4.1: Patrón CRUD Completo

2.6.3 Múltiples actores.

Roles comunes

Puede suceder que los dos actores jueguen el mismo rol sobre el CU. Este rol es representado por otro actor, heredado por los actores que comparten este rol. Es aplicable cuando, desde el punto de vista del caso de uso, solo exista una entidad externa interactuando con cada una de las instancias del caso de uso. [24]

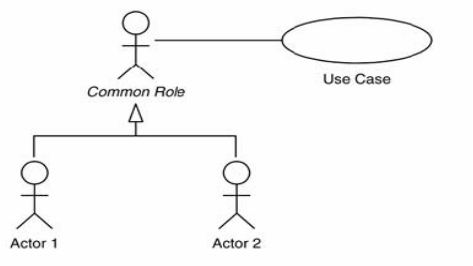


Figura 5: Patrón Múltiples Actores Roles comunes.

A continuación se muestra un fragmento del Diagrama de Casos de Usos del subsistema Programas y Proyectos donde se evidencia el uso de este patrón. En este diagrama tanto el actor **CR** como el actor **Supervisora** presentan el mismo rol sobre el Caso de Uso **Gestionar proyecto** (a modo de ejemplo), es decir, desde el punto de vista del Caso de Uso solo existe un actor (entidad externa) que interactúa con cada una de sus instancias.

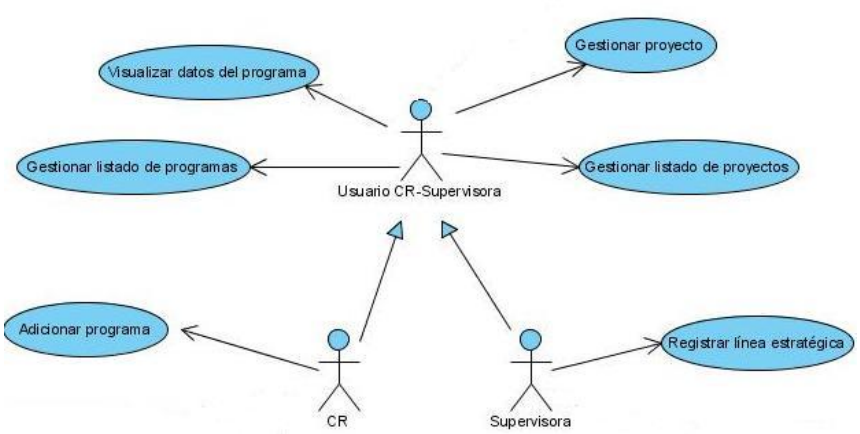


Figura 5.1: Patrón Múltiples Actores Roles comunes.

2.7 Especificación de los Casos de Uso(subsistema Programas -Proyectos)

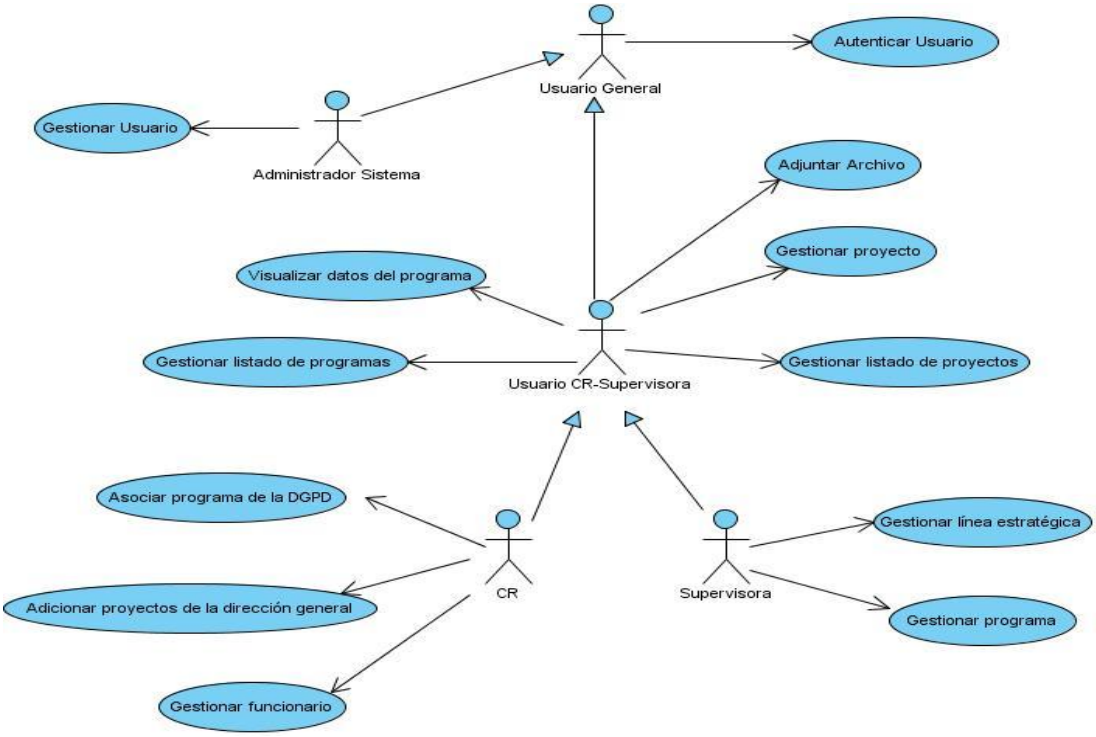


Figura 6: Diagrama de CU subsistema Programas - Proyectos.

A continuación se presentan uno de los Casos de Usos de mayor significado que cubren las principales tareas o funciones que el subsistema ha de realizar.

CUS: Gestionar proyecto.

Nombre del CU	**Gestionar proyecto**
Actores	Usuario CR-Supervisora
Propósito	Gestionar la información referente a los proyectos.
Resumen	El usuario decide gestionar proyecto, el sistema le muestra un listado con todos los proyectos existentes, además da la posibilidad de registrar un nuevo proyecto, actualizar, visualizar e imprimir datos del proyecto.
Referencias	R42, R43,R44,R48
Precondiciones	El usuario tiene que estar autenticado con el rol Usuario CR-Supervisora. Deben existir programas creados.
Poscondiciones	Queda actualizado el listado de proyecto.
Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El usuario decide gestionar proyecto.	2. El sistema muestra un listado con todos los proyectos existentes, además da la posibilidad de registrar un nuevo proyecto, actualizar, visualizar e imprimir datos del proyecto.
3. El usuario selecciona una de las opciones.	4. El sistema ejecuta alguna de las siguientes opciones en dependencia de lo seleccionado: • Si decide registrar un nuevo proyecto, ir a la sección “Registrar nuevo proyecto” • Si decide actualizar un proyecto, ir a la sección “Actualizar proyecto” • Si decide visualizar datos del proyecto, ir a la sección "Visualizar datos del proyecto".
Curso Alternativo de los eventos	
Acciones del Actor	**Respuesta del Sistema**
	2. El sistema no muestra nada pues no existe ningún proyecto y finaliza el caso de uso.
Sección “Registrar nuevo proyecto”	

Acciones del Actor	**Respuesta del Sistema**
	1. El sistema muestra un formulario con los datos que debe llenar.
2. El usuario introduce los datos.	3. El sistema verifica que no hayan introducido datos inconsistentes y además que todos los campos obligatorios estén llenos.
	4. El sistema registra el proyecto y finaliza el caso de uso.
Curso Alternativo de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
	4. Si el sistema verifica que hay datos inconsistentes y además que no todos los campos obligatorios están llenos; emite un mensaje de error.
Sección "Actualizar proyecto"	
Acciones del Actor	**Respuesta del Sistema**
	1. El sistema muestra un formulario con los datos del proyecto.
2. El usuario cambia los datos que desea.	3. El sistema verifica que no hayan introducido datos inconsistentes ni campos obligatorios vacíos.
	4. El sistema actualiza el proyecto y finaliza el caso de uso.
Curso Alternativo de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
	4. Si el sistema verifica que hay datos inconsistentes y además que no todos los campos obligatorios están llenos; emite un mensaje de error.
Sección "Visualizar datos del proyecto"	
Acciones del Actor	**Respuesta del Sistema**
	1 El sistema muestra los datos del proyecto seleccionado y da la opción de imprimir.
2. El usuario selecciona la opción de imprimir.	3. El sistema imprime los datos y finaliza el caso de uso.
Curso Alternativo de los Eventos	
Acciones del Actor	**Respuesta del Sistema**

2. El usuario no selecciona la opción y finaliza el caso de uso.	
Prioridad	Crítico

2.8 Especificación de los Casos de Uso (subsistema Recursos materiales)

Figura 7: Diagrama de CU subsistema Recursos Materiales.

A continuación se presentan uno de los Casos de Usos de mayor significado que cubren las principales tareas o funciones que el subsistema ha de realizar.

CUS: Gestionar entrada de bienes muebles.

Nombre del CU	**Gestionar entrada de bienes muebles.**
Actores	Usuario bienes y materiales.
Propósito	Gestionar la información referente a la entrada de los bienes muebles.
Resumen	Este caso de uso tiene como propósito gestionar la entrada de bienes muebles, el sistema le muestra un formulario que contiene los campos necesarios para darle entrada a los bienes muebles, además da la posibilidad de visualizar y actualizar los datos.
Referencias	R26, R27, R36, R37, R38
Precondiciones	El usuario tiene que estar autenticado con el rol de Usuario bienes y materiales
Poscondiciones	Quedan modificados los datos de la entrada de bienes muebles.
Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El usuario decide gestionar la entrada de bienes y muebles.	2. El sistema muestra un listado con los bienes muebles y da la posibilidad de registrar, actualizar y visualizar la entrada de los bienes muebles.
3. El usuario selecciona una de las opciones.	4. El sistema ejecuta alguna de las siguientes opciones en dependencia de lo seleccionado: • Si decide registrar una entrada de bien mueble, ir a la “Sección Registrar Entrada”. • Si decide modificar los datos de la entrada de un bien mueble, ir a la “Sección Actualizar Entrada”. • Si decide visualizar una entrada de un bien mueble, ir a la “Sección Visualizar Entrada”.
Curso Alternativo de los eventos	
Acciones del Actor	**Respuesta del Sistema**

	2. El sistema no muestra nada pues no existe ningún bien mueble y finaliza el caso de uso.
Sección "Registrar Entrada"	
Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
	1. El sistema muestra un formulario con los datos que debe llenar.
2. El usuario introduce los datos.	3. El sistema verifica que no hayan introducido datos inconsistentes y además que todos los campos obligatorios estén llenos.
	4. El sistema registra la entrada y muestra un mensaje informando de que se realizó la entrada correctamente, finalizando el caso de uso.
Cursos Alternativos de los eventos	
Acciones del Actor	**Respuesta del Sistema**
	4. Si el sistema verifica que hay datos inconsistentes y además que no todos los campos obligatorios están llenos; emite un mensaje de error y señala los datos incorrectos.
Sección "Actualizar Entrada"	
Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
	1. El sistema muestra un formulario con los datos de la entrada.
2. El usuario cambia los datos que desea.	3. El sistema verifica que no hayan introducido datos inconsistentes ni campos obligatorios vacíos.
	4. El sistema actualiza la entrada y finaliza el caso de uso.
Curso Alternativo de los eventos	
Acciones del Actor	**Respuesta del Sistema**
	4. Si el sistema verifica que hay datos inconsistentes y además que no todos los campos obligatorios están llenos; emite un mensaje de error y señala los datos incorrectos.
Sección "Visualizar entrada"	
Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
	1. El sistema muestra un formulario con los datos de la entrada y da la opción de imprimir.

2. El usuario selecciona la opción de imprimir.	3. El sistema imprime los datos y finaliza el caso de uso.
Curso Alternativo de los eventos	
Acciones del Actor	**Respuesta del Sistema**
2. El usuario no selecciona la opción y finaliza el caso de uso.	
Prioridad	Crítico

2.9 Conclusiones.

En el presente capítulo fueron elaborados y descritos algunos de los artefactos propuestos por RUP para el desarrollo de software, tales como el modelo de dominio, definición de los requisitos funcionales y no funcionales, diagrama de casos de uso del sistema y la descripción textual de los casos de uso del sistema. Además se explicaron los patrones de Casos de Usos utilizados para la solución del problema

CAPÍTULO 3: DISEÑO DEL SISTEMA

3.1 Introducción

En este capítulo se describe la solución del sistema a través del diseño de la aplicación, se muestran los patrones de diseño empleados y los diagramas de clases y de interacción. Además se detallan los patrones y estilos arquitectónicos utilizados, así como el modelo de diseño de la base de datos (MER). Se presentan también en este capítulo el Diagrama de Clases del diseño, los Diagramas de Secuencias y el Modelo de Datos.

3.2 Estilo arquitectónico utilizado.

"La Arquitectura del Software es la organización fundamental de un sistema formada por sus componentes, las relaciones entre ellos y el contexto en el que se implantarán, y los principios que orientan su diseño y evolución". [25] Esta es una vista estructural de alto nivel, a través de la cual se define un estilo o una combinación de estilos para la solución de un problema; concentrándose en requisitos no funcionales, convirtiéndose en esencial para el éxito o fracaso de un proyecto.

Partiendo de la existencia de múltiples estilos arquitectónicos para la realización del SIGIRMPP el colectivo de autores decidió utilizar la Arquitectura en Capas (Arquitectura en tres capas) perteneciente a los estilos arquitectónicos de Llamada y Retorno, basándose en las características y peculiaridades del sistema a desarrollar. Esta arquitectura constituye una especialización de la arquitectura cliente-servidor[11] (Ver anexo 2) donde la carga se divide en tres partes (o capas) (figura 8) con un reparto claro de funciones. El desarrollo de cada paquete del SIGIRMPP responde a un modelo multicapas donde cada

[11]Es una arquitectura de procesamientos cooperativo donde uno de los componentes pide servicios a otro. IBM define al modelo Cliente/Servidor como: "La tecnología que proporciona al usuario final el acceso transparente a las aplicaciones, datos, servicios de cómputo o cualquier otro recurso del grupo de trabajo y/o, a través de la organización, en múltiples plataformas. El modelo soporta un medio ambiente distribuido en el cual los requerimientos de servicio hechos por estaciones de trabajo inteligentes o "clientes", resultan en un trabajo realizado por otros computadores llamados servidores". [26]

capa tiene funcionalidades y objetivos precisos, así su implementación se encuentra desacoplada de la programación de cualquier otra y la comunicación con una capa inferior ocurre a través de interfaces.

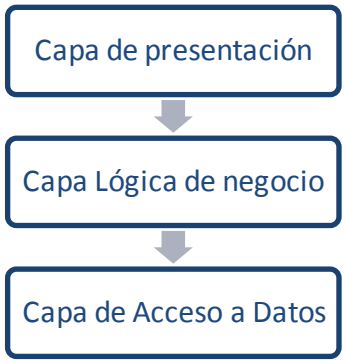


Figura 8: Modelo de Capas del SIGIRMPP

A continuación se realiza una breve descripción de cada una de las capas por las que está compuesto el sistema de acuerdo a la figura anterior.

3.2.1 Capa de Acceso a Datos

La capa de acceso a datos es la responsable de recobrar y persistir información desde y hacia la base de datos y de la comunicación con el gestor de base de datos. Esta capa contiene los objetos que encapsulan la lógica de acceso a datos (DAO) e interfaces brindadas para ser accedida desde la capa de negocio. Las implementaciones de los DAOs extienden de clases de soporte del framework Spring para el uso de este patrón usando el framework ORM Hibernate, mientras que las interfaces se mantienen independientes de Spring e Hibernate. Se encuentran además en esta capa las clases entidades que representan las clases del dominio.

3.2.2 Capa de Lógica de Negocio

La capa de negocio define e implementa las funcionalidades que responden directamente a los requisitos de manera que se conserve la integridad del sistema y de los datos. Está constituida por los servicios. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos para almacenar o recuperar datos de él.

3.2.3 Capa de Presentación

La capa de presentación define e implementa todo lo relacionado con la interfaz gráfica de usuario. Aquí residen la definición de las peticiones que el usuario puede realizar sobre la aplicación, los controladores que manejan el flujo web y la comunicación con las interfaces de la capa de negocio. Además se encuentran las vistas HTML, XML, PDF, XLS que se muestran al usuario y la implementación del comportamiento dinámico de los documentos HTML a través de Java Script. Las responsabilidades principales de la capa de presentación son: la navegabilidad del sistema, el formateo de los datos de salida, la validación de los datos de entrada y la construcción de la interfaz gráfica de usuario.

Es importante destacar que en esta capa se utiliza el patrón arquitectónico **Modelo-Vista-Controlador (MVC)** el cual separara la presentación de los datos. En el caso del SIGIRMPP, este modelo se implementa como se describe a continuación.

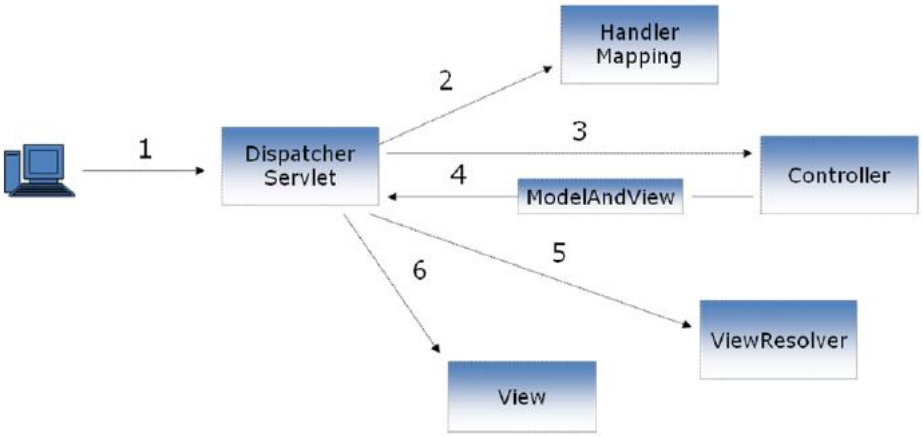


Figura 9: Implementación del patrón MVC en la Capa de presentación.

A grandes rasgos el patrón funciona de la siguiente manera:

Todas las peticiones son recibidas por el **DispatcherServlet**, este con el **HandlerMapping** identifica al controlador que procesará dicha petición, le pasa el control y obtiene como respuesta un **ModelAndView**, de este toma el nombre de la vista y utilizando el **ViewResolver** encuentra la vista física (archivo JSP), a la que le envía los datos extraídos del **ModelAndView** para que sea dibujada (rendered), resultado de lo cual obtiene el código HTML que es enviado como respuesta al cliente.

DispatcherServlet: Las peticiones de los clientes son recibidas por el DispatcherServlet del Framework Spring, quien es el punto de entrada a la implementación del patrón MVC en Spring. En esencia lo que realiza es pasar el control de las peticiones a los Controladores (**Controller**), esta clase es configurada en el archivo **web.xml**.

HandlerMapping: Permite mapear las peticiones HTTP, utilizando los URLs o partes de estos, a los controladores que las procesarán; también contiene de forma opcional interceptores que pueden ser invocados antes o después de efectuarse el mapeo. Se tienen varias opciones a utilizar, independientes o combinadas y que son configuradas en archivos XML.

Controller: Al recibir las peticiones HTTP, ejecutan código Java que realiza la lógica del sistema; manipula a los objetos DAO involucrados y decide que Vista usar para el despliegue de los resultados

ModelAndView: Un objeto de esta clase engloba los datos a mostrar (el modelo, **Model**) en un objeto de la clase **Map** y el nombre de la vista que mostrará dichos datos, este nombre es en general un alias, o sea, no apunta directamente a un archivo físico (JSP).

ViewResolver: Esta clase es utilizada para resolver a partir del nombre de la vista, el alias contenido en el **ModelAndView**, la vista física que será dibujada (rendered) con los datos que le son pasados. Se tienen varias opciones a utilizar, independientes o combinadas y que son configuradas en archivos XML.

View: Este término se refiere las vistas físicas, básicamente a los archivos JSP, PDF, XLS. [15]

3.3 Patrones de diseño de software.

Los patrones de diseño son una solución probada para un problema general de diseño, en un contexto determinado. Encierran la experiencia que programadores e ingenieros han adquirido en la solución de problemas comunes. En el rol que se desarrolla en el trabajo de diploma, ayuda a los autores a completar el diseño de una solución de manera rápida, flexible y segura.

Los patrones de diseño pueden incrementar o disminuir la capacidad de comprensión de un diseño o de una implementación, disminuirla al añadir accesos indirectos o aumentar la cantidad de código, incrementarla al regular la modularidad, separar mejor los conceptos y simplificar la descripción.

Benefician la documentación y el mantenimiento de los sistemas pues proveen una especificación de los objetos y clases y sus relaciones, así como del objetivo del diseño. Es importante notar que un patrón de diseño representa experiencia y conocimiento. No es software ejecutable por lo que, cada vez que se use, debe ser implementado nuevamente. [27] Algunos de los patrones a los que se hace referencia en este documento por su utilización en la solución técnica son:

• **Data Access Object (DAO):** Patrón de Diseño J2EE que 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. La figura 10 provee una visión de su aplicación en el sistema donde las clases de Acceso a Datos se encuentran agrupadas de la siguiente manera ***nombre_del_paquete*.DAO** donde se encuentran las clases responsables de recobrar y persistir información desde y hacia la base de datos y de la comunicación con el gestor de base de datos. Las implementaciones de los DAOs extienden clases de soporte del framework Spring para el uso de este patrón usando el framework ORM Hibernate, mientras que las interfaces se mantienen independientes de Spring e Hibernate. [28] [29]

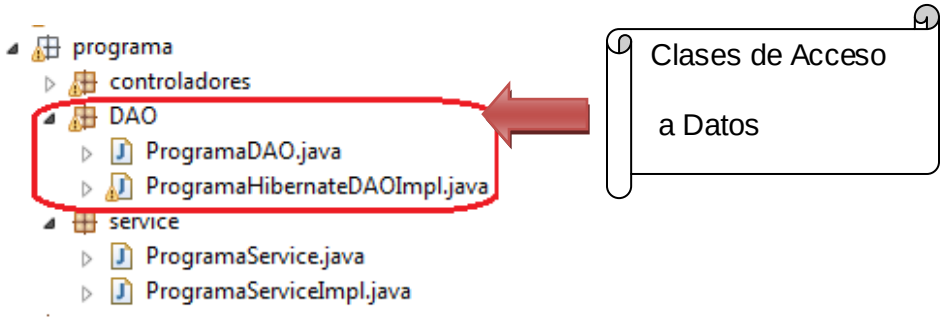


Figura 10: Ejemplo de un paquete DAO en el SIGIRMPP.

• **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. En el SIGIRMPP las clases a las cuales se les asignan estas responsabilidades se encuentran dentro de paquetes con la siguiente estructura ***nombre_del_paquete*.controladores**. A modo de ejemplo se escoge la clase **EliminarProgramaControlador.java**, la cual se puede observar en la figura 11 dentro del

paquete llamado controladores, esta clase extiende del controlador AbstractController del framework Spring y se encarga de controlar la petición de eliminar un programa de la lista de programas, manipulando los datos (el identificador del programa) para en dependencia de estos delegarlos a las clases responsables de ejecutar la acción requerida; devolviéndole luego a la vista física la JSP los datos para presentárselos al usuario. [30]

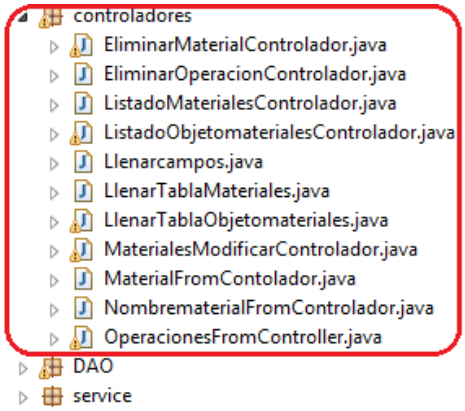


Figura 11: Ejemplo de un paquete de controladores en el SIGIRMPP.

• **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. [29] Este patrón es utilizado por Spring MVC a través del DispatcherServlet que en el SIGIRMPP se encuentra configurado en el **dispatcher-servlet.** (Ver anexo 3). Un ejemplo de cómo funciona este patrón en el sistema sería el siguiente; si el usuario decide visualizar en el navegador la página gestionar proyectos, esta petición (la URL listadoProyectos.hmt) es recibida por el **dispatcher-servlet,** este posee un mecanismo para determinar cual controlador maneja esta petición (listadoProyectosControlador en este caso), luego este controlador toma la petición y ejecuta la tarea (en realidad la delega a otras clases), devolviendo un ModelAndView al **dispatcher-servlet**, el que se encarga de despachar la petición a la Vista (listadoProyecto.jsp)

• **Dependency Injection (Inyección de dependencias):** es un patrón de diseño orientado a objetos, en el que se suministran objetos a una clase en lugar de ser la propia clase quien cree el objeto. Es una forma

de Inversión de Control, que está basada en constructores de Java. Este radica en resolver las dependencias de cada clase (atributos) generando los objetos cuando se arranca la aplicación y luego inyectarlos en los demás objetos que los necesiten a través de métodos set o bien a través del constructor, pero estos objetos se instancian una vez, se guardan en una factoría y se comparten por todos los usuarios (al menos en el caso de Spring) y evitando tener que andar extendiendo clases. Este patrón es de gran utilidad en el SIGIRMPP porque logra un bajo acoplamiento entre las clases. Un conjunto de clases que importan determinadas interfaces y no crean directamente instancias de las implementaciones de dichas interfaces. Las instancias de dichas interfaces se configuran en un fichero XML logrando un bajo acoplamiento o desacoplamiento sin necesidad de modificar el código fuente. Esto se evidencia en las clases agrupadas en los paquetes ***nombre_del_paquete*.controladores** que importan las interfaces contenidas en ***nombre_del_paquete*.service**, sin embargo, ellas no construyen directamente las instancias en su implementación, ya que se inyecta la dependencia a través de un fichero de configuración XML así como la relación que tienen con las clases que implementan estas interfaces, que no es directa pues las clases controladoras no conocen quienes implementan los servicios que ellas utilizan, todo esto determina el bajo acoplamiento entre ellas y la potencialidad de la aplicación ante modificaciones posteriores. Lo mismo sucede entre las clases de los paquetes ***nombre_del_paquete*.service** y ***nombre_del_paquete*.DAO.** [31] [32]

3.4 Diagramas de Clases del Diseño.

Un diagrama de clases del diseño describe gráficamente las especificaciones de las clases de software y de las interfaces de la aplicación. Sirve también para visualizar las relaciones entre las clases que involucran el sistema.

A continuación se describen algunos aspectos relevantes del diseño:

Para una mayor comprensión del diseño del sistema se realizaron los diagramas por casos de usos y solamente se muestra en este documento el diagrama correspondiente al CUS Gestionar Proyectos. El diseño de la aplicación por casos de usos se puede apreciar en la figura 12

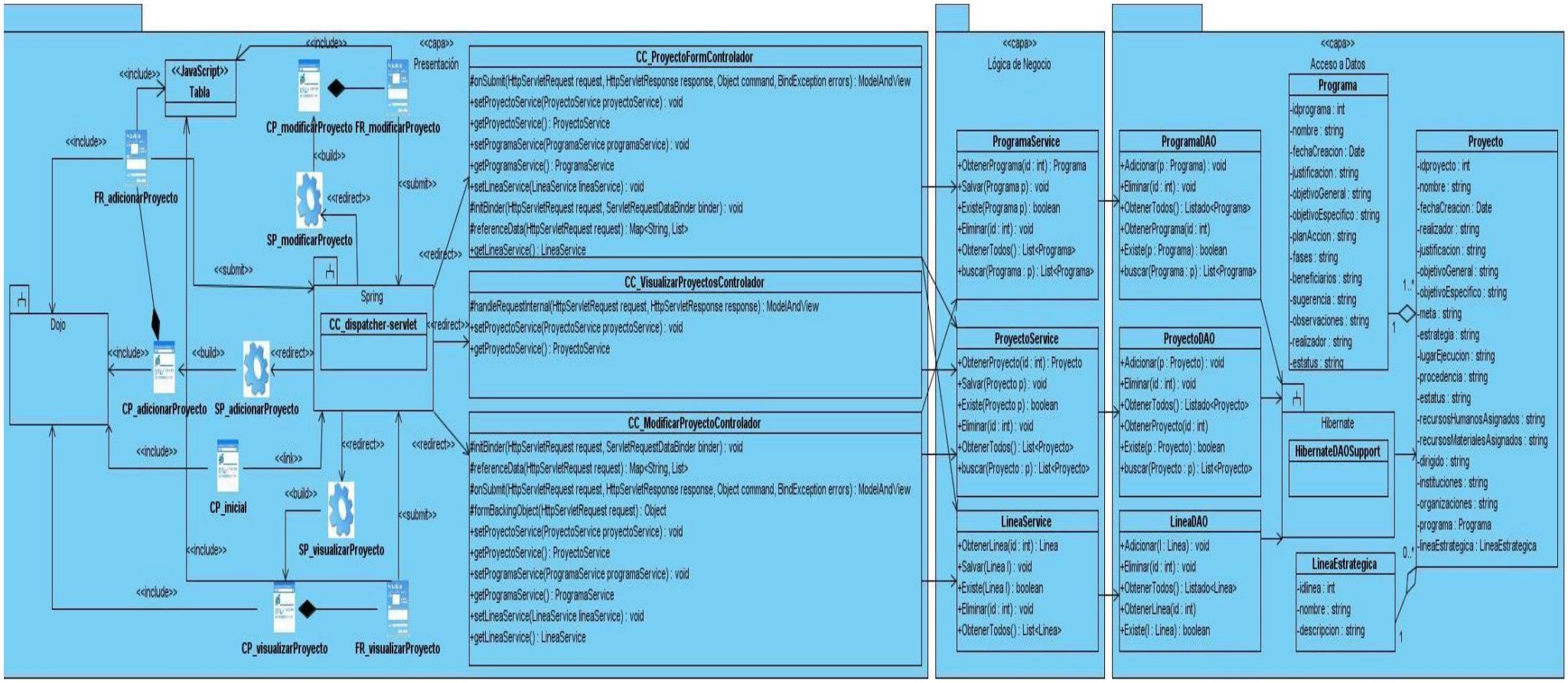


Figura 12: Diagrama de clases del diseño perteneciente al CUS Gestionar Proyecto.

Diagramas de Secuencia.

Se muestran a continuación los diagramas de secuencia correspondientes al caso de uso mencionado. La figura 13 representa la interacción entre las clases que intervienen en la Sección “Registrar nuevo proyecto” del CUS Gestionar Proyecto.

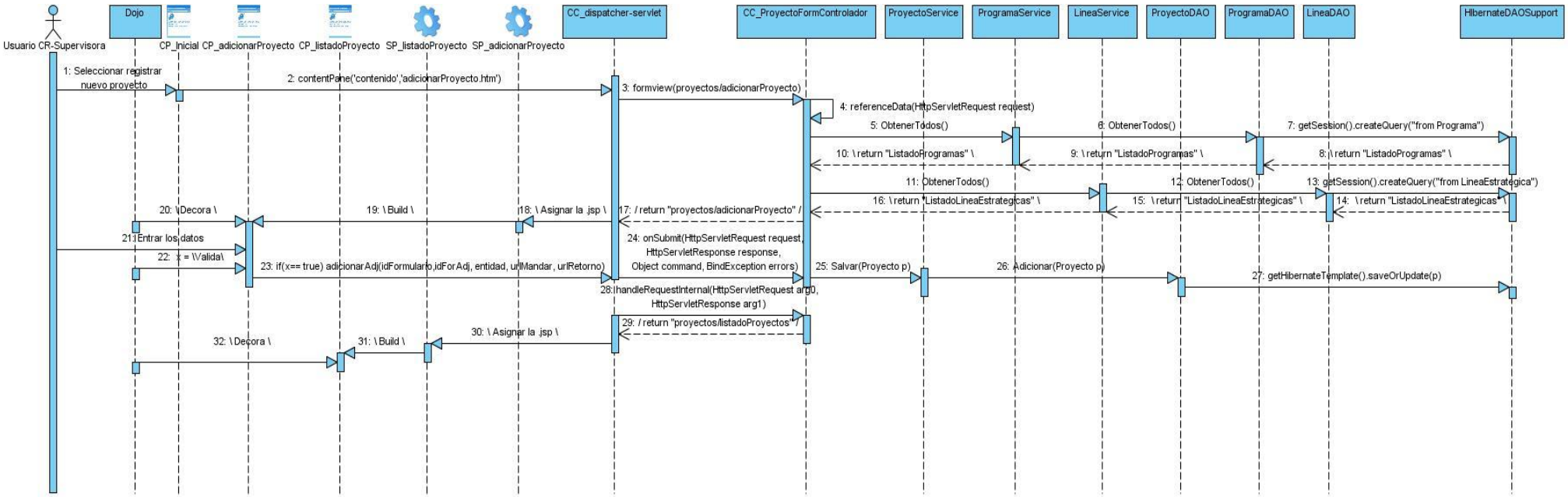


Figura 13: Diagrama de secuencia perteneciente a la sección “Registrar nuevo Proyecto” del CUS Gestionar Proyecto.

La figura 14 representa la interacción entre las clases que intervienen en la Sección “Actualizar proyecto” del CUS Gestionar Proyecto.

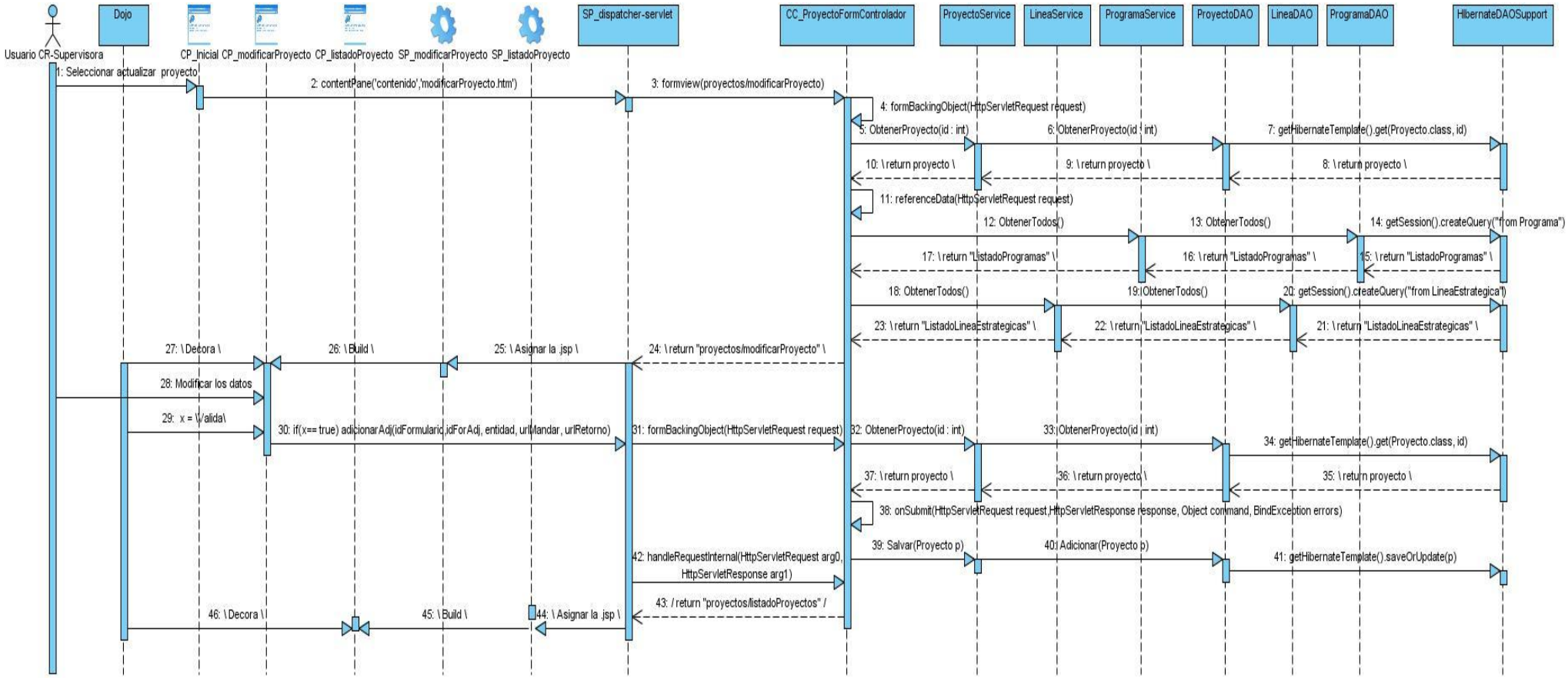


Figura 14: Diagrama de secuencia perteneciente a la sección “Actualizar Proyecto” del CUS Gestionar Proyecto.

La figura 15 representa la interacción entre las clases que intervienen en la Sección “Visualizar proyecto” del CUS Gestionar Proyecto.

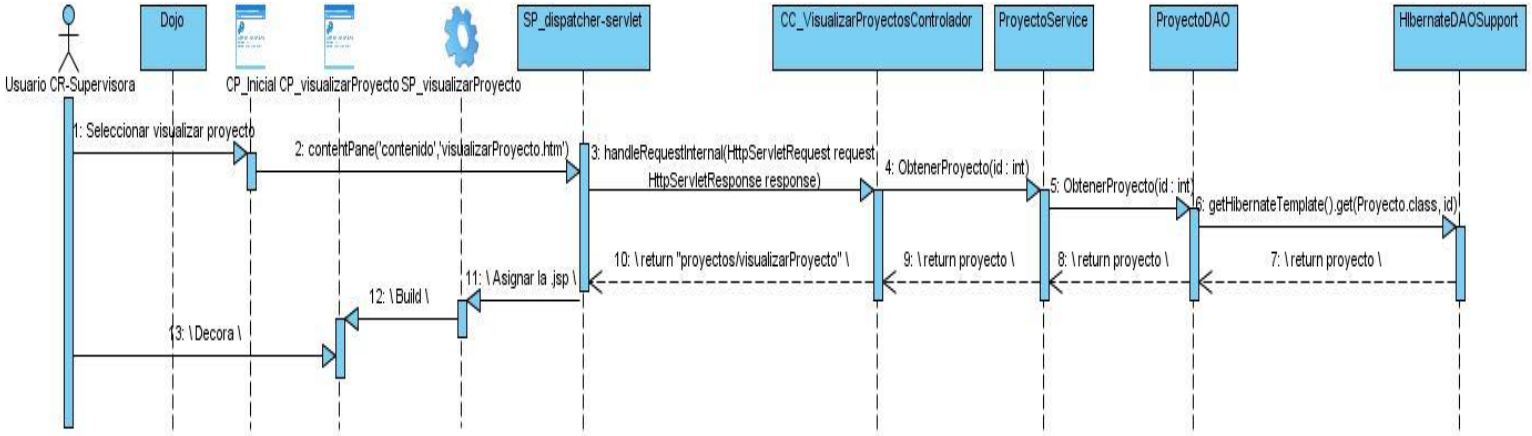


Figura 15: Diagrama de secuencia perteneciente a la sección “Visualizar Proyecto” del CUS Gestionar Proyecto.

3.5 Modelo de Datos

El Modelo de Datos es una forma usada para la descripción de una base de datos. Por lo general permite describir las estructuras de datos de la base (el tipo de datos que incluye la base y la forma en que se relacionan), las relaciones de integridad (las condiciones que los datos deben cumplir para reflejar correctamente la realidad deseada) y las operaciones de manipulación de los datos (agregado, borrado, modificación y recuperación de los datos de la base).

Normalmente todo modelo tiene una representación gráfica, para el caso de los datos el modelo más popular es el Modelo Entidad-Relación o Diagrama Entidad-Relación. Se denomina así debido a que precisamente permite representar relaciones entre entidades (objetivo del modelado de datos).Este

modelo está compuesto por entidades, atributos, relaciones, cardinalidad y llaves. En la figura 16 se representa en Modelo Entidad-Relación correspondiente al SIGIRMPP.

Es importante destacar además que la base de datos se encuentra normalizada[12] hasta la tercera forma normal ya que en ella se incluye la eliminación de todos los grupos repetidos (1ra FN), se asegura que todas las columnas que no son llave sean completamente dependientes de la llave primaria (2FN), y se elimina cualquier dependencia transitiva, es decir, se eliminan las dependencias entre las columnas que no son llave y que a su vez son dependientes de otras columnas que tampoco son llave (3FN). [33]

3.6 Conclusiones.

Como resultado de este capítulo, se representan los diagramas de clases del diseño y los diagramas de interacción del sistema correspondientes a los casos de usos descritos en el capítulo 2. Además se representa el Modelo de Datos del sistema; se especifica el estilo y los patrones arquitectónicos utilizados, así como los patrones de diseño usados.

[12] Proceso mediante el cual se transforman datos complejos a un conjunto de estructuras de datos más pequeñas, que además de ser más simples y más estables, son más fáciles de mantener. También se puede entender la normalización como una serie de reglas que sirven para ayudar a los diseñadores de bases de datos a desarrollar un esquema que minimice los problemas de lógica. Cada regla está basada en la que le antecede.

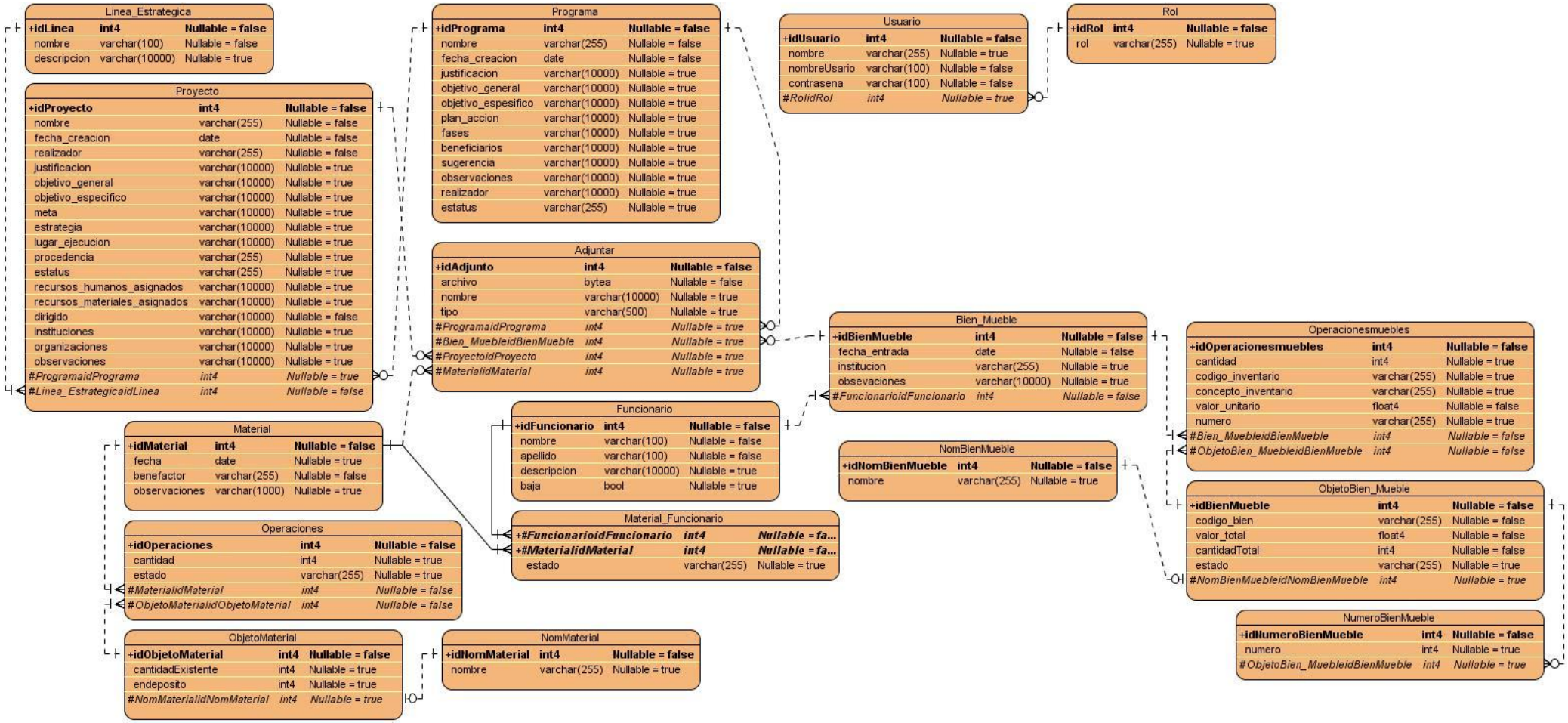


Figura 16: Modelo Entidad-Relación correspondiente al SIGIRMPP.

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBA

4.1 Introducción

En la realización de este capítulo se describe cómo fue implementado el CUS Gestionar Proyectos en términos de componentes. Además se figuran las pruebas unitarias realizadas al sistema y se muestran imágenes de la aplicación desarrollada.

4.2 Diagrama de Componentes.

Dentro del flujo de implementación se crea como artefacto el diagrama de componentes que describe los elementos físicos del sistema y sus relaciones; así como muestra las opciones de realización incluyendo código fuente, binario y ejecutable. Los componentes representan todos los tipos de elementos de software que entran en la fabricación de aplicaciones informáticas. Estos pueden ser simples archivos, paquetes, bibliotecas cargadas dinámicamente, entre otros. En la figura 17 se muestra el Diagrama de componentes correspondiente al CUS Gestionar Proyecto.

4.3 Pruebas al Sistema.

En el presente trabajo los autores se limitan a realizar pruebas a nivel de desarrollador. Tradicionalmente estas pruebas han sido consideradas solo para la prueba de unidad (enfocada a los elementos testeables más pequeños del software. Verifica que los flujos de control y de datos están cubiertos, y que ellos funcionen como se espera), aunque en la actualidad en algunos casos pueden ejecutar pruebas de integración (descubren errores o incompletitud en las especificaciones de las interfaces de los paquetes. El objetivo es tomar los componentes probados en unidad y construir una estructura de programa que esté de acuerdo con lo que dicta el diseño). Se recomienda que estas pruebas cubran más que las pruebas de unidad. [7] Como resultado de las pruebas realizadas se corrigieron errores ortográficos, campos sin validar, entre otros errores detectados en la interfaz de la aplicación, así como algunas líneas de código defectuosas.

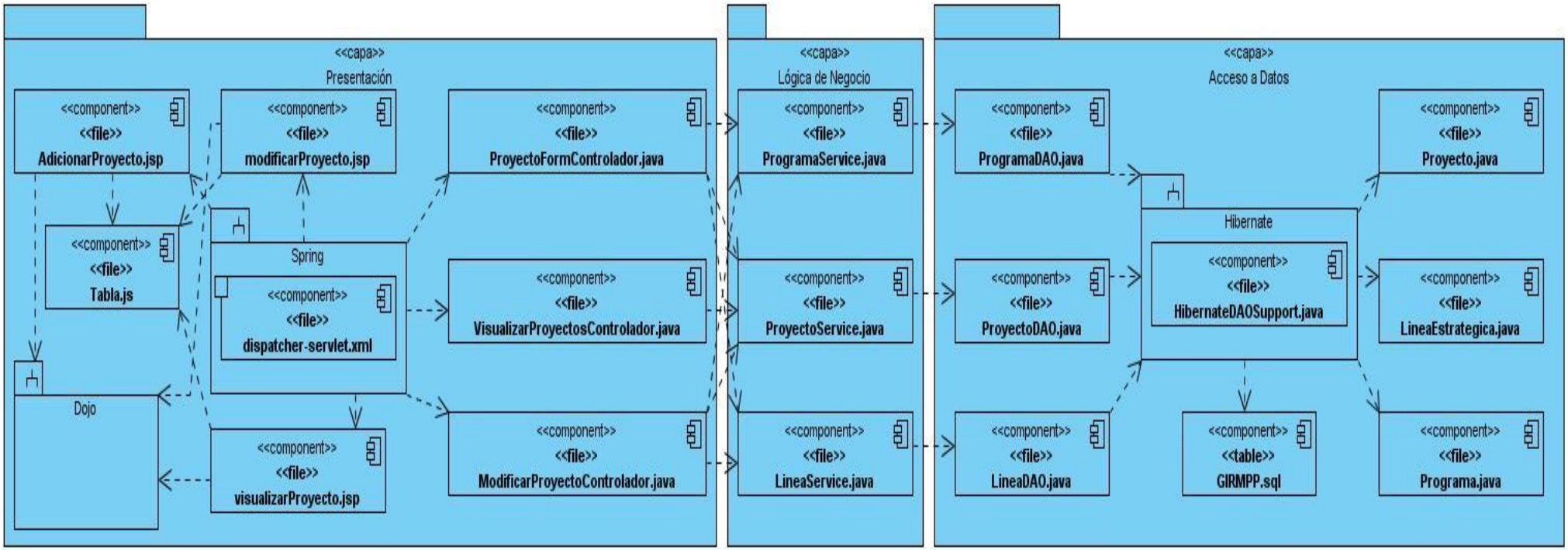


Figura 17: Diagrama de componentes correspondiente al CUS Gestionar Proyecto.

4.4 Imágenes de la aplicación.

A continuación se muestran las interfaces correspondientes al CUS Gestionar Proyectos.

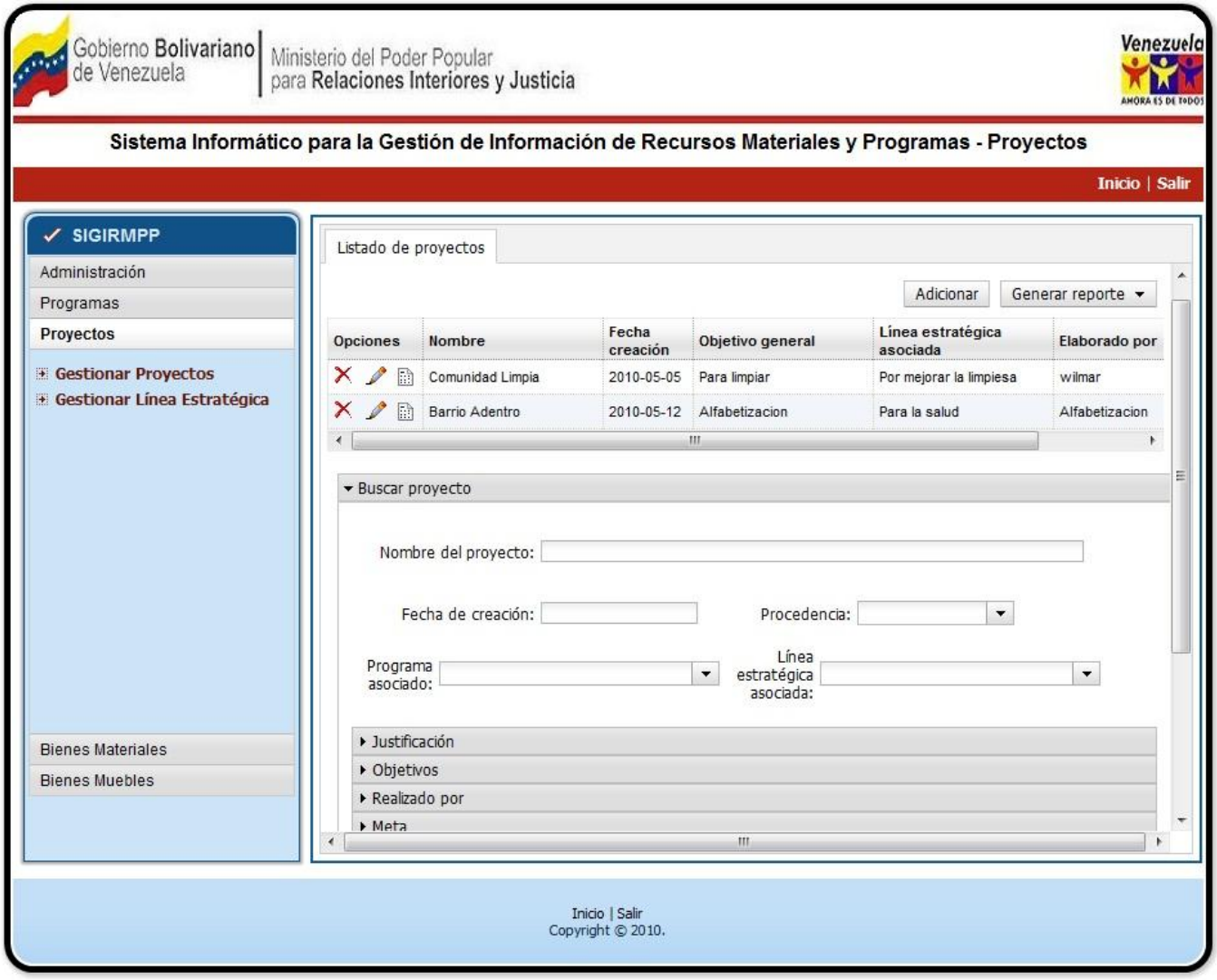


Figura 18: Imagen de la aplicación correspondiente al CUS Gestionar Proyecto.

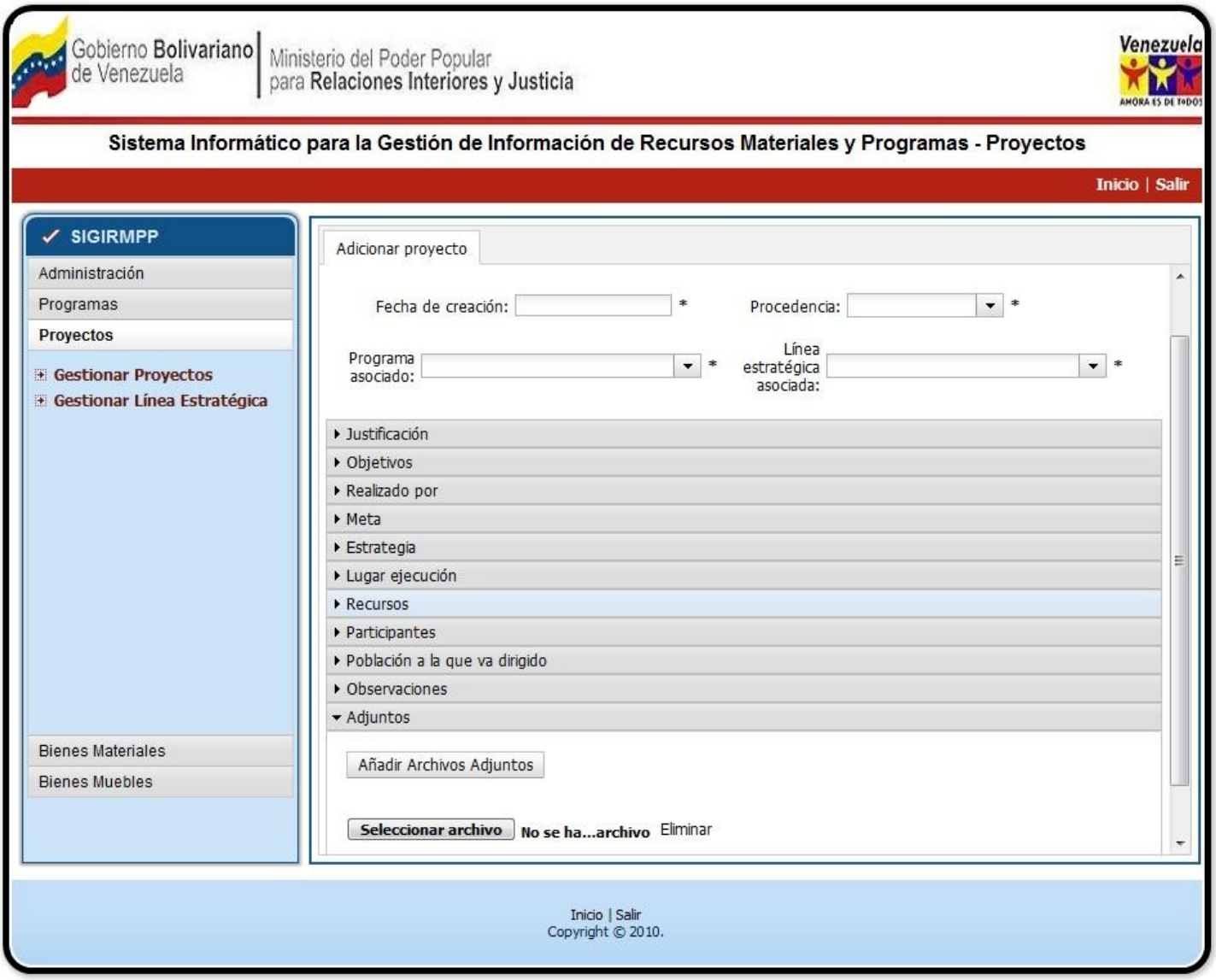


Figura 19: Imagen de la aplicación correspondiente al CUS Gestionar Proyecto.

4.5 Conclusiones

Como resultado de este capítulo se presentó el diagrama de componentes correspondiente al CUS Gestionar Proyectos, con las imágenes respectivas en la aplicación. También se mostró el resultado de las pruebas unitarias realizadas al sistema y se explicó la forma en que se ejecutaron.

CONCLUSIONES

- ✓ La correcta especificación de los requisitos funcionales posibilitó la definición de las funcionalidades del SIGIRMPP.
- ✓ La utilización de los diferentes patrones de diseño permitió diseñar una solución considerando las buenas prácticas de desarrollo de software.
- ✓ Con la implementación del SIGIRMPP se logró mejorar la gestión de la información referente a los recursos materiales y los programas-proyectos que manejan las Coordinaciones Regionales y la Dirección General de Prevención del Delito.

RECOMENDACIONES

- ✓ Se recomienda utilizar el framework Spring en su versión 3.0 para facilitar el desarrollo del sistema, disminuyendo la cantidad de código y la reusabilidad del mismo.
- ✓ Se recomienda integrar el SIGIRMPP a la “Solución Integral para el Perfeccionamiento del Sistema de Prevención del Delito de la República Bolivariana de Venezuela.” en lo referente a los módulos de Recursos Materiales y Programas – Proyectos.
- ✓ Se recomienda profundizar en el estudio de las Librerías de etiquetas o TagLibs de Spring, las cuales proporcionan el desarrollo de componentes más orientados a presentación que ayudan a simplificar las páginas. Facilitan la generación de forma dinámica de código html, siendo esta una ventaja para la reutilización de código y disminución del tiempo desarrollo.

REFERENCIAS BIBLIOGRÁFICAS

1- Ministerio del Poder Popular para Relaciones Interiores y Justicia (01 de Abril de 2009). "¿Qué es Plan Caracas Segura?" Disponible en: http://www.mij.gov.ve/index.php?option=com_content&view=article&id=108 Consultado el: (30 de noviembre del 2009).

2- Ministerio de Administraciones Públicas."Planificación de sistemas de información". Disponible en: www.csi.map.es/csi/metrica3/psiproc.pdf Consultado el: (30 de noviembre del 2009).

3- latecladeescape.com (25 de enero de 2009). "Metodologías de desarrollo del software." Disponible en: http://latecladeescape.com/w0/ingenieria-del-software/metodologias-de-desarrollo-del-software.html Consultado el: (27 de febrero del 2010).

4- Letelier, Patricio, "Proceso de desarrollo de software". Disponible en: https://pid.dsic.upv.es/C1/Material/Documentos%2520Disponibles/Introducci%C3%B3n%2520Proceso%2520de%2520Desarrollo%2520de%2520SW.doc.Consultado el: (28 de febrero del 2010).

5- Gary K. Evans. "A Simplified Approach to RUP". Disponible en: http://www.therationaledge.com/content/jan_01/t_rup_ge.html. Consultado el: (1 de marzo del 2010).

6- Sosa López, Dailyn "SISTEMA PARA EL CONTROL DEL USO DE LOS SOFTWARES EDUCATIVOS". Disponible en: http://www.eumed.net/libros/2009c/585/Descripcion%20del%20modelo%20del%20dominio.htm Consultado el: (1 de marzo del 2010).

7- Pressman, Roger; (2002). "Ingeniería de software. Un enfoque práctico."Disponible en:http://www.scribd.com/doc/6722160/Roger-S-Pressman-Ingenieria-Del-Software-Un-Enfoque-Practico. Consultado el: (2 de marzo del 2010).

8- uml.org. (1999 - 2010) "Unified Modeling Language" Disponible en: http://www.uml.org/. Consultado el: (2 de marzo del 2010).

9- Dondo, Agustín. (1999 - 2010)"Programación orientada a objetos" Disponible en: http://www.programacion.com/articulo/dondo_poo/. Consultado el: (2 de marzo del 2010).

10-osmosislatina.com (2005)"Áreas conceptuales "Disponible en: http://www.osmosislatina.com/java/componentes.htm Consultado el: (2 de marzo del 2010).

11-Vandana Pursnani. “Introducción a la Programación de Java Servlets” Disponible en:http://www.acm.org/crossroads/espanol/xrds8-2/servletsProgramming.htmlConsultado el: (2 de marzo del 2010).

12-Hall, Marty (1999- 2010) “Servlets y JSP”. Disponible en: http://www.programacion.com/java/tutorial/servlets_jsp/11/#servlets_jsp_jsp10 Consultado el: (2 de marzo del 2010).

13-Java.sun.com(1995-2010) “Lesson: JDBC Basics ” Disponible en: http://java.sun.com/docs/books/tutorial/jdbc/basics/index.html Consultado el: (2 de marzo del 2010).

14-KAISLER, STEPHEN. H. (2005) Software Paradigms. Disponible en: http://books.google.com.cu/books?id=kfGHwo2E0FYC&dq=KAISLER,+STEPHEN.+H.+Software+Paradigms&printsec=frontcover&source=bn&hl=es&ei=0nNSW1IYWA8gaSweigDw&sa=X&oi=book_result&ct=result&resnum=4&ved=0CBYQ6AEwAw#v=onepage&q=KAISLER%2C%20STEPHEN.%20H.%20Software%20Paradigms&f=false. Consultado el: (2 de marzo del 2010).

15-springframework.org. Disponible en: http://www.springframework.org. Consultado el: (2 de marzo del 2010).

16-Hibernate.org. Disponible en: http://www.Hibernate.org/ Consultado el: (2 de marzo del 2010).

17-postgresql.org (19 de enero del 2008).”PostgreSQL, Award Winning Software”. Disponible en: http://www.computerworld.com.au/article/62894/postgresql_affiliates_org_domain/ Consultado el: (5 de marzo del 2010).

18-Postgresql.org. “PostgreSQL 8.3.11 Documentation” Disponible en: http://www.postgresql.org/docs/8.3/interactive/index.html Consultado el: (5 de marzo del 2010).

19-SQLManager.net. Disponible en: http://www.sqlmanager.net/products/postgresql/manager/. Consultado el: (9 de marzo del 2010).

20-eclipse.org. Disponible en: http://www.tools.Hibernate.org/. Consultado el: (9 de marzo del 2010).

21-jasperforge.org .Disponible en: http://jasperforge.org/ Consultado el: (9 de marzo del 2010).

22-dojotoolkit.org. Disponible en: http://www.dojotoolkit.org/. Consultado el: (9 de marzo del 2010).

23-Visualparadimg.com .Disponible en :http://www.visual-paradigm.com/product/vpuml/ Consultado el: (9 de marzo del 2010).

24-Övergaard, Gunnar; Palmkvist, Karin (2004) “Use Cases: Patterns and Blueprints”. Addison Wesley. Consultado el: (15 de marzo del 2010).

25-IEEE-SA. (2004)Disponible en:http://standards.ieee.org/reading/ieee/std_public/description/se/1471-2000_desc.html Consultado el: (17 de marzo del 2010).

26-Reynoso, Carlos. (marzo del 2004) “Estilos y Patrones en la Estrategia de Arquitectura de Microsoft” Disponible en: http://www.willydev.net/descargas/prev/Estiloypatron.pdf Consultado el: (17 de marzo del 2010).

27-Larman, Craig (1999) “UML y Patrones. Introducción al análisis y diseño orientado a objetos”Disponible en:http://www.scribd.com/doc/457980/UML-y-Patrones-by-Craig-Larman Consultado el: (6 de abril del 2010).

28-aprendeenlinea.com. “Patrón DAO.” Disponible en: http://aprendeenlinea.udea.edu.co/lms/moodle/mod/resource/view.php?id=57215Consultado el: (10 de abril del 2010).

29-Rayes, Andrés. (22 de septiembre del 2003). “Diseño de aplicaciones Internet usando los Patrones de diseño J2EE”. Disponible en: http://www.programacion.com/articulo/diseno_de_aplicaciones_internet_usando_los_patrones_de_diseno_j2ee_229#corej2ee_patterns_disweb. Consultado el: (14 de abril del 2010).

30-Larman, Craig (1999) “UML y Patrones. Introducción al análisis y diseño orientado a objetos”Disponible en:http://www.scribd.com/doc/457980/UML-y-Patrones-by-Craig-Larman Consultado el: (6 de abril del 2010).

31-Fowler, Martin. (23 de enero del 2004). “Inversion of Control Containers and the Dependency Injection pattern”. Disponible en: http://martinfowler.com/articles/injection.html. Consultado el: (17 de abril del 2010).

32-Vicente, Raúl. (9 de noviembre del 2006). “Inyección de dependencias con Spring”. Disponible en: http://www.programania.net/diseno-de-software/inyeccion-de-dependencias-con-spring/. Consultado el: (20 de abril del 2010).

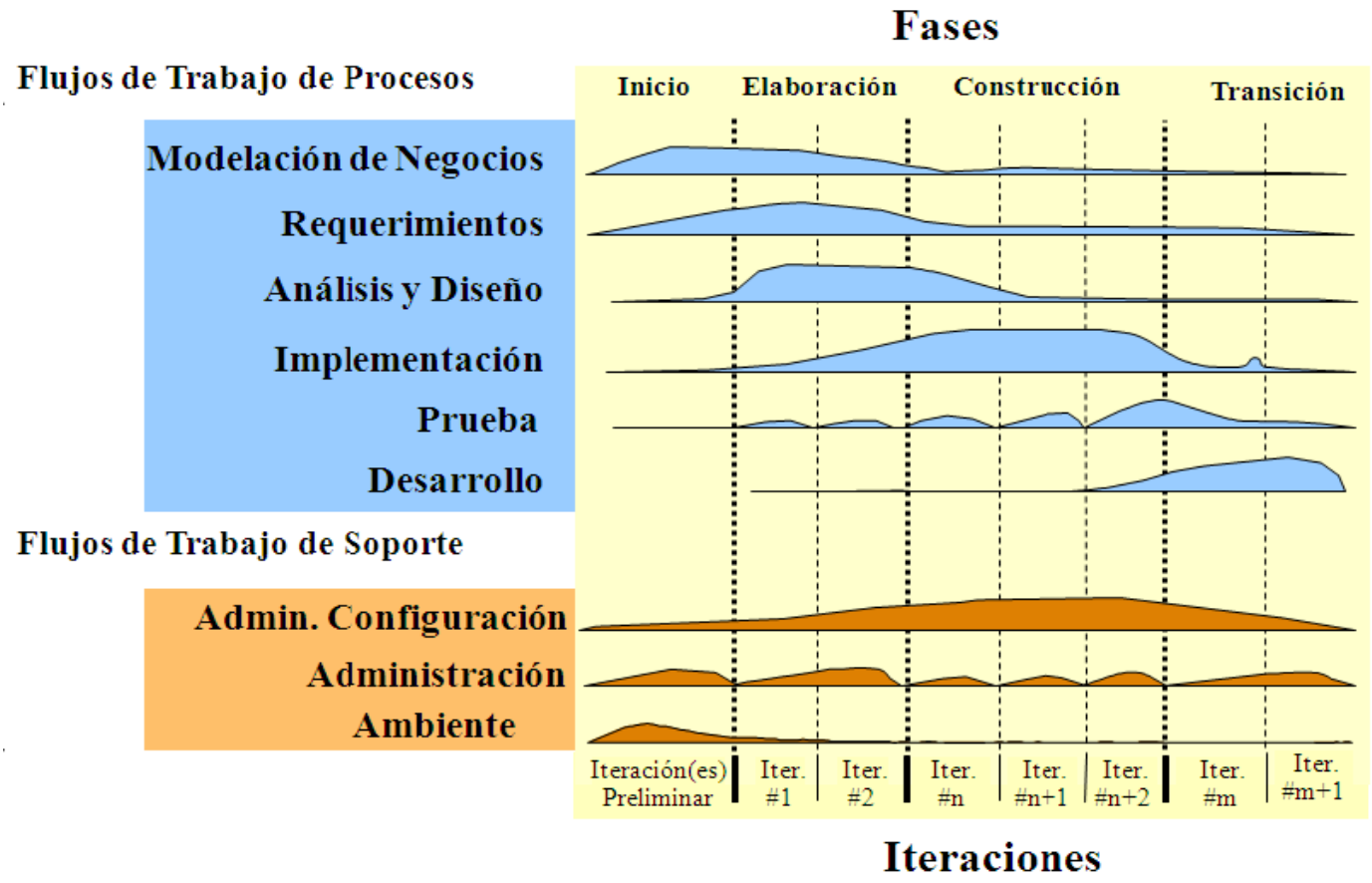
33-MySQL-hispano.com. “Normalización de bases de datos”. Disponible en: http://www.mysql-hispano.org/page.php?id=16&pag=1. Consultado el: (4 de mayo del 2010).

BIBLIOGRAFÍA

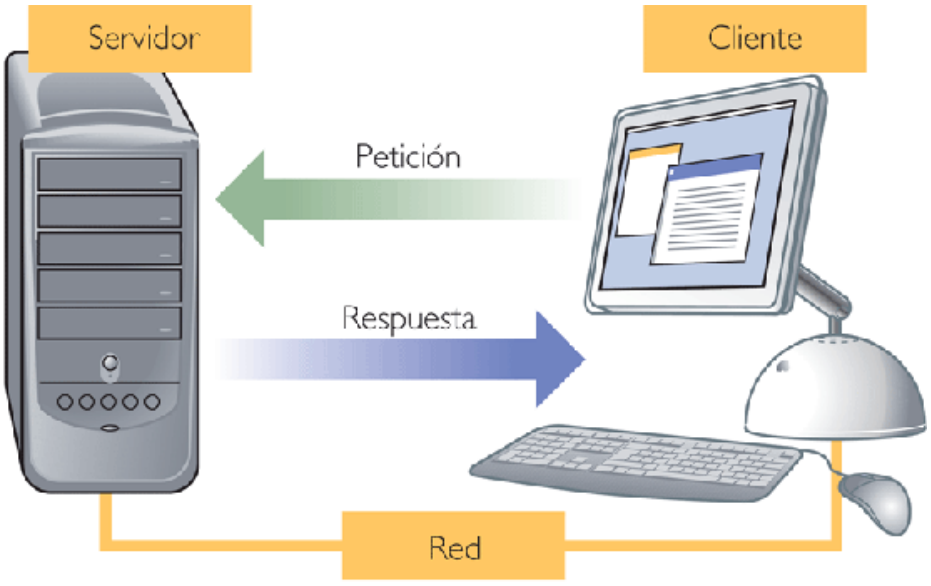
1. Booch, G. Rumbaugh, J. y Jacobson, I. (2000) “El Lenguaje Unificado de Modelado”. Addison-Wesley.
2. Jacobson, I.; Booch, G. y Rumbaugh, J. (2000) “El Proceso Unificado de Desarrollo de software”. Addison-Wesley.
3. Johson, Rod (2005)” Professional Java Development with the Spring Framework”.
4. Johson, Rod (2008)” spring-reference version 2.5.6”.
5. KAISLER, STEPHEN. H. (2005) “Software Paradigms”.
6. Larman, Craig (1999) “UML y Patrones. Introducción al análisis y diseño orientado a objetos”.
7. Navasa, M.A. Pérez, M. Sánchez.Ed. M. Sánchez. (Oct 1999.) “Aplicación de UML al desarrollo de sistemas orientados a objetos”.
8. ÖVERGAARD, Gunnar; PALMKVIST, Karin (2004) “Use Cases: Patterns and Blueprints”. Addison Wesley.
9. Pressman, Roger; Ingeniería de software. Un enfoque práctico. McGraw.Hill/Interamericana de España.
10. Stephen R. Schach. D. Mc Graw Hill. “Análisis y diseño orientado a objetos”.

ANEXOS

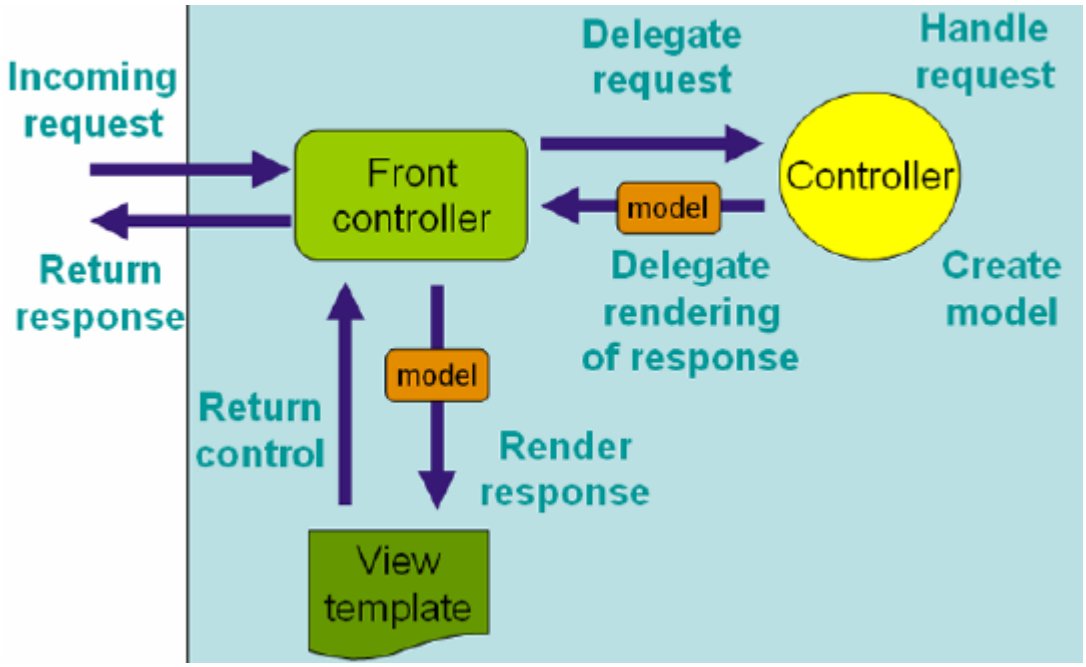
Anexo 1: Fases y flujos de trabajo en RUP



Anexo 2: Arquitectura cliente servidor.



Anexo 3: Esquema del patrón Controlador frontal en el SIGIRMPP.



GLOSARIO

API: Application Program Interface.

CR: Coordinación Regional.

DGPD: Dirección General de Prevención del Delito.

FN: Forma Normal

JDBC: Java Database Connectivity.

JMS: Java Message Service.

JMX: Java Management Extensions.

JSP: Java Server Pages.

MIPPRIJ: Ministerio del Poder Popular para las Relaciones Interiores y Justicia de La República Bolivariana de Venezuela.

ORM: Mapeo objeto-relacional.

POO: Programación Orientada a Objetos

RMI: Java Remote Method Invocation.

RUP: Rational Unified Process.

SIGIRMPP: Sistema Informático para la Gestión de Información de los Recursos Materiales y Programas – Proyectos de las Coordinaciones Regionales de Prevención del Delito.

SQL: Structured Query Language.

XML: Extensible Markup Language.