UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 2

SISTEMA DE MAPIFICACIÓN WEB

Trabajo de Diploma

Presentado para optar por el título de Ingeniero en Ciencias Informáticas.

Autor: Reynier González Tejeda

Tutor: Ing. Elieser Bello Ross
Ing. Michel Arias Arias

Ciudad de la Habana, Cuba. Enero de 2009.

"*Año del 50 aniversario del triunfo de la Revolución*"

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis y reconozco 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 _____.

_____

Reynier González Tejeda

Autor

_____ _____

Elieser Bello Ross Michel Arias Arias

Tutor Tutor

Dedicatoria.

A mis padres por enseñarme a ser el hombre que soy, todo lo que pueda lograr en este mundo se los debo a ustedes.

Agradecimientos.

A papi, por enseñarme a ser mejor cada día y no conformarme con la mediocridad.

A mami, por darme todo el cariño y la dedicación que una madre puede dar.

A Eli, por apoyarme siempre y darme tanta ternura.

A mis amigos, que estén lejos o cerca nunca los olvido.

A Rammel, por todas las horas que nos quemamos las pestañas.

A mis tutores Elieser y Michel, sin ellos esto no hubiera sido posible.

Resumen.

Se conoce como información geográfica a aquella que describe lugares, objetos o eventos referenciados sobre la superficie terrestre, que tienen coordenadas o regiones definidas explícitamente, de forma que pueden ser ubicados en un mapa. La labor de los órganos de seguridad, entiéndase policía o cualquier otra institución de esta índole, constantemente demanda el uso de información de este tipo para el cumplimiento de sus funciones.

Las dependencias policiales de la República Bolivariana de Venezuela aún no cuentan con las ventajas que ofrecen las tecnologías de la informática y las comunicaciones. Continuamente se ven en la necesidad de manipular y consultar la información geográfica de sus áreas de responsabilidad, la ubicación de sus recursos, el desarrollo de los delitos que ocurren y el resultado de las medidas que se toman para combatirlos, lo cual se convierte en un proceso engorroso y se obtienen resultados poco fiables.

Con este fin nace la idea de construir un Sistema de Mapificación Web para ser usado por las dependencias policiales de la República Bolivariana de Venezuela, que permitirá la representación y procesamiento de información geográficamente referenciada. Además incluirá la capacidad de crear mapas temáticos, que serán usados para estudios estadísticos policiales.

El Sistema de Mapificación Web tiene como finalidad mostrar información relevante acerca de la ocurrencia y evolución de los actos delictivos en un área geográfica determinada, contribuyendo así a que las dependencias policiales de la República Bolivariana de Venezuela brinden un mejor servicio y tomen mejores decisiones para el bienestar y la seguridad del pueblo.

Índice.

Introducción.

Se conoce como información geográfica a aquella que describe lugares, objetos o eventos referenciados sobre la superficie terrestre, que tienen coordenadas o regiones definidas explícitamente, de tal forma que pueden ser ubicados en un mapa.

La labor de los órganos de seguridad, entiéndase policía o cualquier otra institución de esta índole, constantemente demanda el uso de información geográfica para el cumplimiento de sus funciones. La necesidad de controlar sus áreas de responsabilidad, así como la ocurrencia y evolución de los delitos en cada una de ellas y la planificación de tácticas y estrategias para combatirlos, son algunos ejemplos de acciones que requieren el uso de tecnología capaz de ofrecer información visual inmediata y detallada.

Las dependencias policiales de la República Bolivariana de Venezuela aún no cuentan con las ventajas que ofrecen las tecnologías de la informática y las comunicaciones. Sus acciones se basan en los métodos tradicionales, de mapas físicos marcados con pines. Continuamente se ven en la necesidad de manipular y consultar la información geográfica de sus áreas de responsabilidad, la ubicación de sus recursos, el desarrollo de los delitos que ocurren y el resultado de las medidas que se toman para combatirlos, que de la forma tradicional no permite que se obtengan mejores resultados en el cumplimiento de sus funciones. Por solo poner un ejemplo, en un mapa de una ciudad un pin puede marcar el área equivalente a varias cuadras. Este margen de error parece pequeño, pero puede significar la diferencia entre la vida y la muerte de una persona. Además, los mapas físicos marcados con pines o lápiz no reflejan patrones o tendencias que pueden ser fácilmente determinados mediante medios digitales.

Ante el análisis de la **situación problémica** expuesta, se plantea como **problema científico** la necesidad que presentan las dependencias policiales de la República Bolivariana de Venezuela de administrar, mostrar y procesar información geográficamente referenciada de sus áreas de cobertura.

El **objeto de esta investigación** son los sistemas de información geográfica (SIG), siendo el **campo de acción** los sistemas de información geográfica web como apoyo a los órganos policiales.

Para darle solución al problema planteado se definió como **objetivo general** desarrollar un sistema de información geográfico web (Sistema de Mapificación Web) para las dependencias policiales de la República Bolivariana de Venezuela.

Los **objetivos específicos** que se derivan de este objetivo general son:

- Seleccionar la tecnología SIG a emplear en el desarrollo del sistema.
- Definir la arquitectura del Sistema de Mapificación Web.
- Obtener una versión funcional que posibilite el procesamiento de información geográficamente referenciada.

Las **tareas** a desarrollar para dar respuesta a las interrogantes y cumplimentar los objetivos de la investigación son las siguientes:

- Valorar las posibles tecnologías SIG a utilizar.
- Determinar las herramientas, frameworks y la metodología a emplear en el desarrollo del sistema.
- Diseñar e implementar la arquitectura del Sistema de Mapificación Web.
- Implementar las funcionalidades que permitan el procesamiento de información geográficamente referenciada.

Para dar solución satisfactoria al problema expuesto anteriormente y cumplir con los objetivos propuestos es necesaria una profunda comprensión de los procesos de análisis y tratamiento de información geográfica que ocurren en el entorno en que se enmarca el problema. Es imprescindible un conocimiento previo de las necesidades concretas que presentan las dependencias policiales de la República Bolivariana de Venezuela, las cuales al final de este trabajo serán transformadas en funcionalidades del Sistema de Mapificación Web. La principal fuente de esta información está constituida por el Trabajo de Diploma presentado por Michel Arias Arias y Ailín Alarcón Ferrá para optar por el título de Ingeniero en Ciencias Informáticas. El presente trabajo constituye una continuación del mencionado anteriormente, y está caracterizado por la reutilización y el mejoramiento de sus resultados.

El producto final de la investigación será un Sistema de Mapificación Web con la capacidad de procesar información referenciada geográficamente, para ser utilizado por las dependencias policiales de la República Bolivariana de Venezuela.

El presente documento está compuesto por tres capítulos:

En el **Capítulo 1 "Fundamentación Teórica"** se exponen los fundamentos generales que sirven de soporte teórico en la solución del problema. Se analizan las herramientas y lenguaje de programación a utilizar para el desarrollo y además se plantea la metodología a emplear en el proceso de creación del sistema.

En el **Capítulo 2** "**Diseño del Sistema**" se diseña la solución propuesta, se muestran los diagramas de clases que representan la arquitectura del Sistema de Mapificación Web aplicando los patrones arquitectónicos y de diseño seleccionados.

En el **Capítulo 3** "**Implementación de la Solución**" se representa el diagrama de componentes que detalla la forma en que se estructura el sistema, reflejando la transformación de los elementos del modelo del diseño en términos de componentes, ficheros que contienen código fuente, ejecutables, etc. así como las dependencias entre ellos.

Capítulo 1. Fundamentación Teórica.

1.1 Introducción.

En este capítulo se detalla la situación problémica existente en las dependencias policiales de la República Bolivariana de Venezuela que hacen necesario este trabajo. Además se exponen conceptos relacionados con los sistemas de información geográfica y las herramientas que facilitan el desarrollo de aplicaciones basadas en SIG, y se definen las tecnologías y metodología de desarrollo de software a utilizar para el diseño e implementación del Sistema de Mapificación Web.

1.2 Seguridad Ciudadana.

La seguridad es una premisa necesaria para el funcionamiento de la sociedad y uno de los principales criterios para asegurar la calidad de vida de la población. Constituye un derecho de todo ser humano y es la facultad de toda persona a desenvolverse dentro de una sociedad libre de amenazas que atenten contra su vida, integridad física, psíquica o cultural.

Se puede identificar como seguridad ciudadana a la acción integrada que desarrolla un país, con la colaboración de la ciudadanía, destinada a asegurar su convivencia pacífica y la erradicación de la violencia.

El universo de la seguridad ciudadana comprende tanto aquello que la amenaza como lo que la protege; de un lado está la violencia, criminalidad nacional e internacional, y del otro lado está el quehacer de todas las instituciones estatales y de la sociedad civil relacionada con la promoción y protección de la misma.

La inseguridad ciudadana se ha convertido en uno de los grandes desafíos de las sociedades contemporáneas. El impacto del fenómeno sobre la calidad de la vida de los ciudadanos obliga a los gobiernos nacionales y locales y a los sectores organizados de la sociedad a diseñar esquemas alternativos a los existentes que, siendo en su cometido de disminuir los niveles de inseguridad, no sacrifiquen el avance de la Democracia y el respeto por los Derechos Humanos y las Garantías Ciudadanas. (1)

1.2.1 Seguridad Ciudadana en Venezuela.

La inseguridad ciudadana es uno de los problemas más preocupantes en la población de la República Bolivariana de Venezuela. La cantidad de delitos que se cometen en esta nación de América del Sur ha originado gran incertidumbre, teniendo en cuenta la situación política vivida en estos últimos años, generada por una oposición que busca crear caos para tratar de llegar al poder, lo cual ha influido para que el gobierno tome cartas en este asunto con la decisión y firmeza que se requiere para solventar el problema de la inseguridad. (2)

Venezuela encara hoy día el desafío de adecuar sus estructuras administrativas al nuevo proceso socio-político que vive el país, marcado por los principios de justicia social y respeto a los derechos y garantías constitucionales, con clara orientación hacia la inclusión de todos los ciudadanos en el goce y ejercicio pleno de los mismos.

En ese sentido el Ministerio del Poder Popular para Relaciones Interiores y Justicia, conjunto con los órganos de seguridad ciudadana, han tomado medidas para el combate y prevención del delito, así como asegurar una convivencia pacífica en el país.

1.2.2 Actividad Policial en Venezuela.

Existen en Venezuela distintos organismos destinados a garantizar la seguridad ciudadana de la población. Entre ellos se cuentan el Cuerpo de Investigaciones Científicas, Penales y Criminalísticas (CICPC), el Cuerpo de Bomberos y Administración de Emergencias de Carácter Civil, la Organización de Protección Civil y Administración de Desastres, y la policía municipal, estatal y nacional.

La actividad policial, como parte de la administración del Estado, puede concebirse como "*la disposición de una fuerza organizada para el mantenimiento del orden público mediante la vigilancia (aspecto preventivo) y la aprehensión de los infractores a los fines de imposición de una sanción (aspecto represivo), sanción a cargo de la propia instancia policial, de otras dependencias administrativas o de una instancia jurisdiccional*". (3)

Uno de los objetivos principales para garantizar la seguridad ciudadana en Venezuela es fortalecer las políticas, estrategias, lineamientos y directrices que conlleven a la automatización del control de las acciones policiales, a fin de proveer de tecnología de información y comunicación que permita lograr el cumplimiento de sus tareas sobre una plataforma tecnológica de avanzada. (4)

De esta situación se derivan ideas que conllevan a la creación de sistemas informáticos como el Sistema de Gestión Policial (SIGEPOL), y el Centro de Tratamiento y Análisis de Información para la Seguridad Ciudadana (CTAISC), con la finalidad de ayudar a coordinar las acciones de los organismos de seguridad ciudadana.

Con este fin nace también la idea de diseñar e implementar un Sistema de Mapificación Web para ser usado por las dependencias policiales de la República Bolivariana de Venezuela, que permitirá la representación y procesamiento de información geográficamente referenciada. Además incluirá la capacidad de crear mapas temáticos, que pueden ser usados para estudios estadísticos policiales.

El Sistema de Mapificación Web tiene como finalidad mostrar información relevante acerca de la ocurrencia y evolución de los actos delictivos en un área geográfica determinada, contribuyendo así a que las dependencias policiales de la República Bolivariana de Venezuela brinden un mejor servicio y tomen mejores decisiones para el bienestar y la seguridad del pueblo.

1.3 Sistema de Información Geográfico (SIG).

Un sistema de información geográfico (SIG) es una integración de hardware, software y datos con la capacidad de mostrar, almacenar, administrar y controlar todo tipo de información geográficamente referenciada. Permiten visualizar, analizar e interpretar los datos en forma de mapas, reportes, gráficas y tablas, que pueden revelar relaciones, tendencias y patrones. (5)

Los SIG han constituido durante los últimos años una potente herramienta de trabajo de uso masivo para la creación y gestión de información espacial. Un SIG puede definirse como un sistema computarizado capaz de contener y procesar datos descriptivos de un lugar de la superficie terrestre. Los datos son organizados básicamente como una serie de capas, con una base de datos de atributos asociados (5). Estos sistemas permiten la entrada, almacenamiento, manipulación, análisis, modelación, recuperación, representación y salida eficiente de datos espaciales y de sus atributos, de acuerdo a especificaciones y requerimientos concretos (6). La tecnología SIG puede integrar datos geográficos con la fotografía aérea y los reconocimientos en el terreno.

Un SIG debe tener las siguientes funciones:

- Funciones de entrada y salida de datos.
- Funciones de gestión de datos (modificar, eliminar, etc.).
- Funciones de análisis y consulta.

El SIG es una herramienta de gran importancia en el análisis y la toma de decisiones. Aporta soluciones a diferentes problemas que frecuentemente requieren de un rápido acceso a información de varios tipos, que sólo pueden estar relacionadas por geografía o distribución espacial. Sólo la tecnología SIG permite almacenar y manipular información usando geografía, analizar patrones, relaciones, y tendencias en la información, todo con el interés de contribuir a la toma de mejores decisiones y a la realización de estudios más precisos.

Existen varias clasificaciones de SIG, de acuerdo al modo en que representan los datos. Ninguno es superior a otro, sino que están optimizados para propósitos específicos.

1.3.1 SIG Vectorial.

Los SIG vectoriales son aquellos que para la descripción de los objetos geográficos utilizan vectores.

En este modelo la información se representa a través de puntos, líneas y polígonos almacenados como una colección de coordenadas (x,y). A estos objetos de dibujo se les puede asociar las diversas capas de información que se relacionan con el modelo espacial generado a través de puntos y líneas:

- Los puntos se reducen a pares de coordenadas latitud-longitud o x-y, que marcan la posición de lo que es modelado sobre la superficie de la tierra. Así, los pozos, fuentes, manantiales, puntos contaminados, etc. pueden quedar representados con esta estructura vectorial.

- Las líneas son una serie ordenada de posiciones unidas por segmentos rectos. Permiten modelar carreteras, ríos, curvas de nivel, etc.

- Los polígonos son líneas cerradas que delimitan superficies. Son empleados para modelar vegetaciones, suelos, geologías, montes, provincias, países, etc.

Este tipo de SIG es extremadamente útil para describir características discretas, pero menos útil para describir características de variación continua.

Fig. 1.1 SIG Vectorial.

1.3.2 SIG Ráster.

Utiliza una malla rectangular de pequeñas celdas, denominadas píxel. Cada píxel recibe un número como representación a su valor temático. Este número porta la información necesaria para modelar un aspecto del medio. Dado que la malla es regular (el tamaño del píxel es constante) y que conocemos la posición en coordenadas del centro de una de las celdas, se puede decir que todos los píxeles están georreferenciados.

El modelo de datos ráster es especialmente útil cuando tenemos que describir objetos geográficos con límites difusos o de medios muy variables.

Fig. 1.2 SIG Ráster.

1.3.3 SIG Orientado a Objetos.

El SIG orientado a objetos es un modelo de datos que ha surgido en estos últimos años y plantea un cambio en la concepción de la estructura de la base de datos geográficos. Mientras que el modelo de datos vectorial y ráster estructura sus datos en capas, este intenta organizar la información en objetos geográficos y sus relaciones, estos objetos son agrupados en clases y son sometidos a una serie de procesos.

Este modelo introduce un carácter dinámico a la información incluida en el sistema, por ello es aconsejable usarlo en situaciones en que la naturaleza de los objetos que se modelan cambia constantemente en el tiempo y/o el espacio.

La ventaja fundamental que permite esta estructura de datos frente a las demás es la forma dinámica en que se representan los datos. Es decir, a partir de una serie de parámetros establecidos en el comportamiento de los objetos geográficos, se puede simular su evolución futura, lo que constituye un gran avance si se trabaja en entornos en los que se requiere simulación de situaciones potenciales.

1.3.4 Algunas aplicaciones de los SIG.

Administrar, regular, controlar y planificar las acciones que se desarrollan en un territorio determinado constituye una tarea muy compleja, por lo cual los SIG son utilizados en distintos sectores como una herramienta de apoyo y consulta. A continuación se brindan algunas aplicaciones de estos en distintas esferas de la actividad humana:

- **Agricultura**: Para el monitoreo y manejo de parcelas y granjas, estudios regionales y nacionales.
- **Estudios Ambientales:** Para evitar la degradación ambiental pues contribuyen a determinar zonas de riesgo, ya sea por desertificación y deforestación, y para realizar estudios climatológicos.
- **Epidemiología y Salud:** Asiste en la ubicación de enfermedades relacionadas con factores ambientales y actividades humanas.
- **Turismo:** Para la realización y manejo de facilidades y atracciones.
- **Estudios sociales:** Análisis de dinámicas demográficas.
- **Instituciones de Seguridad Ciudadana:** Para visualizar en un área geográfica lo que se requiera atender; como la ubicación de eventos que son informados al centro por organismos o personas que requieran del apoyo de los órganos de seguridad; la ubicación donde ha ocurrido un determinado incidente, los sitios referenciados geográficamente como estaciones de bomberos y policías, hospitales y vehículos en servicio.

1.4 Tecnología SIG a utilizar.

Existen varias compañías y proyectos dedicados al desarrollo de tecnología SIG. Estos ofrecen una amplia colección de herramientas que hacen más rápido y eficiente el desarrollo de aplicaciones que utilicen SIG. Entre ellos podemos encontrar a ESRI, MapServer y MapInfo.

1.4.1 ESRI.

ESRI (Enviromental Systems Research Institute) es una de las empresas dedicadas al desarrollo de sistemas de información geográfica. Sus productos estrellas ArcGIS y ArcView han alcanzado gran popularidad entre las empresas que utilizan tecnología SIG, y sus formatos se han extendido y estandarizado. Provee gran número de aplicaciones para la captura, edición, análisis, tratamiento, diseño, publicación e impresión de información geográfica. Estas aplicaciones se engloban en familias temáticas como ArcGIS Desktop para el tratamiento de mapas digitales en aplicaciones de escritorio, ArcGIS Server para la

publicación y gestión de información geográfica sobre la web, y ArcGIS Movil para la captura y administración de datos geográficos en dispositivos móviles.

1.4.2 MapServer.

MapServer es una plataforma de código abierto para la publicación de datos espaciales y aplicaciones de cartografía interactiva para la web. Originalmente desarrollado a mediados del año 1990 en la Universidad de Minnesota, MapServer es liberado bajo una licencia estilo MIT (una de las licencias open source del Instituto Tecnológico de Massachusetts), y funciona en las principales plataformas (Windows, Linux y Mac OS X).

MapServer es ahora un proyecto de OSGeo (*Open Source Geospatial Foundation*), y es mantenido por un número creciente de desarrolladores de todo el mundo. Cuenta con el apoyo de un diverso grupo de organizaciones que financian el equipamiento y el mantenimiento, dentro de OSGeo. Es administrado por el Comité Directivo del Proyecto MapServer, compuesto por desarrolladores y otros colaboradores.

Las principales ventajas de MapServer consisten en ser multiplataforma, soportar los principales formatos vectoriales y ráster, y ser muy configurable, además de ser un proyecto de código abierto lo cual lo convierte en una opción muy atractiva para los desarrolladores. Sin embargo, según sus propios directivos, MapServer aún no es un SIG con funciones completas.

1.4.3 MapInfo.

MapInfo es uno de los líderes mundiales en el desarrollo de tecnología SIG. Ofrece software, datos, herramientas en línea y servicios profesionales para ayudar a las organizaciones a obtener un mayor valor de localización basada en la información y tomar decisiones más rentables. Más de 7000 empresas y organizaciones gubernamentales de todo el mundo, lo que representa casi cada sector de la industria, dependen de tecnología y conocimientos de MapInfo para alcanzar sus objetivos de negocio. Sus especialistas en la industria ayudan a estas organizaciones a aprovechar la ubicación de inteligencia para satisfacer diversas necesidades, desde gestión de activos y planificación de la red, a la selección de sitios, gestión de riesgos y aplicaciones móviles. Las soluciones de MapInfo están disponibles en múltiples idiomas a través de una red de socios estratégicos y canales de distribución en 60 países de todo el mundo.

Los diferentes productos de MapInfo han sido diseñados para que los usuarios visualicen y examinen los datos desde una perspectiva geográfica, superponiéndolos en mapas digitales con diferentes niveles. De esta forma, es posible utilizar la información geográfica de una empresa (por ejemplo, ciudades, nombres de calles, códigos postales) para descubrir modelos de comportamiento y tendencias que resultan difíciles de identificar de otra manera.

MapInfo ofrece una amplia gama de soluciones y software para PC e Internet, herramientas de desarrollo de aplicaciones y datos.

- **MapInfo MapBasic:** Entorno de desarrollo de aplicaciones para MapInfo Professional.
- **MapInfo MapX:** Una forma fácil y económica de añadir funcionalidades de gestión de mapas a otras aplicaciones y soluciones.
- **MapInfo MapXtreme 2005:** Es la novedad principal del entorno de desarrollo basado en ubicaciones. Permite crear aplicaciones personalizadas sobre la plataforma .NET
- **MapInfo SpatialWare:** Herramienta de gestión de información para almacenar, gestionar y manipular los datos basados en ubicación.
- **MapInfo MapXtreme Java Edition:** Potente servidor de mapas para Internet basado en Java, para implementaciones globales de las aplicaciones de mapas.

1.4.4 Mapinfo MapXtreme Java Edition.

Es un servidor de gestión de mapas para la visualización geográfica y la toma de decisiones, que permite el desarrollo y la implantación de aplicaciones con rapidez y bajos costos. Ofrece las siguientes características:

- Compatible con los principales servidores web y de aplicaciones.
- Compatible con MapInfo SpatialWarea (producto de almacenamiento de datos espaciales).
- Brinda un Framework completo, aplicaciones de ejemplo y una gran documentación diseñada para acelerar el desarrollo de aplicaciones.

- Fácil de personalizar, los Java Beans pueden ser usados en entornos de desarrollo visual.
- Es multiplataforma.
- Para la gestión de los mapas propone:
    - Mapas temáticos. Permite sombreado, gráficos de tarta y de barra, etc.
    - Gestión de objetos. Permite guardar, intersectar o borrar objetos como puntos, líneas y polígonos, etc.
    - Capa modificable. Permite dibujar objetos personalizados.
    - Búsquedas. Permite localizar recursos rápidamente en un mapa, ejemplo: calle, código postal, etc.
    - Control de capas. Permite gestionar múltiples capas, colores de las capas de datos, niveles de zoom, visibilidad y estilos de etiquetas.
    - Selecciones espaciales. Permite trabajar con datos espaciales de una región específica, ejemplo una provincia.
    - Tratamiento de imágenes Ráster. Permite visualizar imágenes .tif, escaneadas y vía satélite.

Mapinfo MapXtreme Java Edition es un producto desarrollado completamente en Java, que permite desplegar aplicaciones en cualquier sistema operativo. Incluye los siguientes componentes:

- API de MapJ y bibliotecas de clases Java.
- MapXtreme JavaBeans.
- Manager y Java Web Application Builder.
- Biblioteca de etiquetas personalizadas JSP.
- Máquina virtual de Java 2 (1.4.1).
- Ejemplos de aplicaciones.
- Muestra de datos.
- Conjunto de documentación de Java MapXtreme (incluye la guía del desarrollador y la documentación del API de MapJ).

MapXtreme Java Edition ofrece una solución fácil de usar, multiplataforma, de alta calidad y gran rendimiento.

1.5 Metodología de Desarrollo de Software.

Una metodología no es más que un conjunto de técnicas, normas y disposiciones que ayudan a desglosar el desarrollo de un software en tareas más sencillas, fáciles de realizar y controlar. Según Jacobson y Rumbaugh, es la definición del conjunto de actividades que guían los esfuerzos de las personas implicadas en el proyecto, a modo de plantilla que explica los pasos necesarios para terminar el proyecto (7).En un proyecto de desarrollo de software la metodología define Quién debe hacer Qué, Cuándo y Cómo debe hacerlo.

No existe una metodología de software universal. Las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigen que el proceso sea configurable

En la actualidad existen varias metodologías utilizadas en el desarrollo de software:

- OPEN.
- MÉTRICA 3.
- Rational Unified Process (RUP).

1.5.1 Rational Unified Process (RUP).

La metodología RUP es el resultado de varios años de desarrollo y uso práctico en el que se han unificado técnicas de desarrollo y de trabajo de muchas metodologías utilizadas por los clientes. La versión que se ha estandarizado vio la luz en 1998 y se conoció en sus inicios como Proceso Unificado de Rational 5.0, de ahí las siglas con las que se identifica a este proceso de desarrollo.

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 creados, modificados y usados por las actividades. Pueden ser modelos, elementos dentro del modelo, código fuente y ejecutables.

- **Flujos de actividades (“Cuándo”):** Secuencia de actividades realizadas por trabajadores y que produce un resultado.

El ciclo de vida de RUP presenta tres características fundamentales: dirigido por casos de uso, centrado en la arquitectura, iterativo e incremental.

- **Dirigido por casos de uso:** Los casos de uso reflejan lo que los usuarios futuros necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requerimientos. A partir de aquí los casos de uso guían el proceso de desarrollo ya que los modelos que se obtienen, como resultado de los diferentes flujos de trabajo, representan la realización de los casos de uso (cómo se llevan a cabo).
- **Centrado en la arquitectura:** La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente.
- **Iterativo e Incremental:** RUP propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros. Por ejemplo, una iteración de elaboración centra su atención en el análisis y diseño, aunque refina los requerimientos y obtiene un producto con un determinado nivel, pero que irá creciendo incrementalmente en cada iteración.

Es práctico dividir el trabajo en partes más pequeñas o mini proyectos. Cada mini proyecto es una iteración que resulta en un incremento. Las iteraciones hacen referencia a pasos en los flujos de trabajo, y los incrementos, al crecimiento del producto. Cada iteración se realiza de forma planificada es por eso que se dice que son mini proyectos.

RUP divide el proceso de desarrollo en ciclos, teniendo un producto funcional al final de cada ciclo, cada ciclo se divide en fases que finalizan con un hito donde se debe tomar una decisión importante, las cuatro fases que incluye RUP son:

- **Inicio:** el objetivo en esta etapa es determinar la visión del proyecto.
- **Elaboración:** en esta etapa el objetivo es determinar la arquitectura óptima.
- **Construcción:** en esta etapa el objetivo es obtener la capacidad operacional inicial.
- **Transición:** el objetivo es llegar a obtener el release del proyecto.

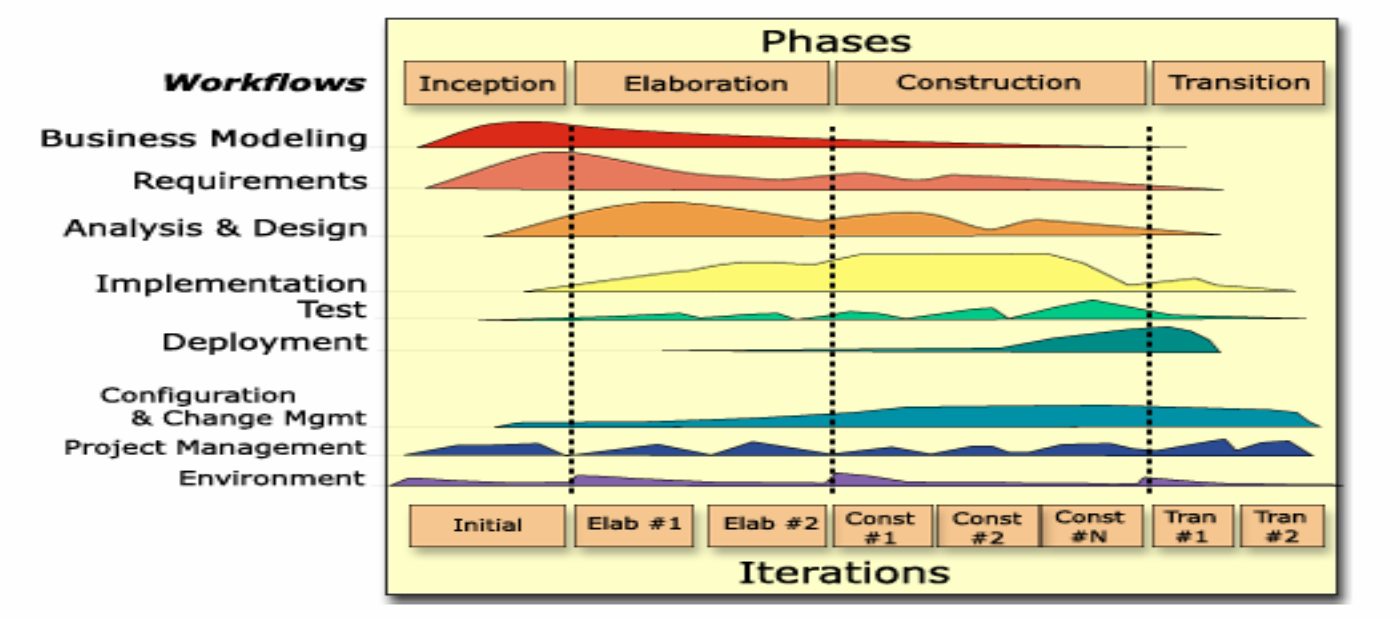

Fig. 1.3: RUP en dos dimensiones.

1.6 Plataforma de Software a utilizar.

Una plataforma de software es un conjunto de programas y librerías usadas para ejecutar aplicaciones; en su forma más simple consiste únicamente de un sistema operativo, entorno de programación, o más comúnmente una combinación de ambos. Una excepción notable es el lenguaje de programación Java, que usa una máquina virtual independiente del sistema operativo para leer el código compilado, conocido en la jerga de Java como bytecode. Dos de las plataformas más conocidas y extendidas entre los programadores son .NET y Java.

1.6.1 La Plataforma Java.

La plataforma Java es el nombre de un entorno o plataforma de desarrollo creada por Sun MicroSystems. Se ejecuta sobre otra plataforma hardware/software y provee:

- El lenguaje de programación Java.
- La Máquina Virtual Java (JVM).
- La Interfaz de Programación de Aplicaciones (API).

Hoy en día esta plataforma ha evolucionado en concordancia con el avance tecnológico y se ha convertido en una de las plataformas de programación más usadas por los desarrolladores. Su principal ventaja es que su entorno de desarrollo es independiente de la plataforma sobre la que se trabaje, es decir, sus aplicaciones son funcionales tanto en Linux, Unix, Solaris, Power/Mac, como en Windows, brindando una compatibilidad a nivel de fuentes, código intermedio y bibliotecas de clases.

La implementación en dicha plataforma proporciona un compilador de Java para traducir código fuente al conjunto de instrucciones de la máquina virtual de Java, un intérprete para ejecutar las instrucciones de la máquina virtual y una implementación de la API de Java.

Actualmente existen distintas ediciones o especificaciones de la plataforma Java como por ejemplo:

- **J2ME (Java 2 Edición Micro):** Orientada a entornos de limitados recursos, como teléfonos móviles, PDAs (Personal Digital Assistant o Asistente Personal Digital), etc.
- **J2SE (Java 2 Edición Estándar):** Para entornos de gama media y estaciones de trabajo. Aquí se sitúa al usuario medio en una PC de escritorio.
- **J2EE (Java 2 Edición Empresarial):** Orientada a entornos distribuidos empresariales o de Internet.

Hasta la fecha, la plataforma Java ha atraído a más de cinco millones de desarrolladores de software. Se utiliza en los principales sectores de la industria de todo el mundo y está presente en un gran número de dispositivos, equipos y redes.

La versatilidad y eficiencia de la tecnología Java, la portabilidad de su plataforma y la seguridad que aporta, la han convertido en la tecnología ideal para su aplicación a redes. De portátiles a centros de datos, de consolas de juegos a súper equipos científicos, de teléfonos móviles a Internet, Java está en todas partes.

Más de 4.500 millones de dispositivos utilizan la tecnología Java:

- 800 millones de computadoras personales.
- 1.500 millones de teléfonos móviles y otros dispositivos de mano (fuente: Ovum).
- 2.200 millones de tarjetas inteligentes.
- Sintonizadores, impresoras, web cams, juegos, sistemas de navegación para automóviles, terminales de lotería, dispositivos médicos, cajeros de pago en aparcamientos, etc.

1.6.2 Lenguaje de programación Java.

Java es un lenguaje de programación, independiente de la plataforma, creado por Sun MicroSystems. Alcanzó su madurez con la popularización de Internet y es en cierta manera el heredero legítimo de C++, eliminando la mayoría de sus complejidades como por ejemplo: la herencia múltiple y la creación de punteros.

Java presenta características que lo convierten en un lenguaje seguro, estándar y de alto nivel, algunas de las principales características se muestran a continuación:

- **Orientado a Objetos:** Java implementa la tecnología básica de C++ con algunas mejoras y elimina algunas características para mantener el objetivo de la simplicidad del lenguaje. Soporta las tres particularidades de la orientación a objetos: encapsulación, herencia y polimorfismo. Se ha impuesto como el paradigma de los lenguajes de programación orientados a objetos.
- **Distribuido:** Java se ha construido con extensas capacidades de interconexión TCP/IP. Existen librerías de rutinas para acceder e interactuar con protocolos como http y ftp. La característica de ser distribuido es debido a que proporciona las librerías y herramientas para que los programas puedan distribuirse, es decir, que se corran en varias máquinas, interactuando.
- **Interpretado:** Se traduce el código fuente a un código intermedio (bytecode), que es interpretado por La Maquina Virtual de Java, lo cual permite que se pueda ejecutar en cualquier sistema operativo.

- **Robusto:** Java realiza verificaciones en busca de problemas tanto en tiempo de compilación como en tiempo de ejecución. La comprobación de tipos en Java ayuda a detectar errores lo antes posible en el ciclo de desarrollo.
- **Seguro:** No se permite el acceso ilegal a la memoria ya que no se trabaja con punteros.
- **Portable:** En lugar de compilarse a código nativo de la máquina, los programas de Java se traducen al formato bytecode que es el mismo en cualquier sistema operativo. Estos bytecodes son procesados directamente por la Máquina Virtual de Java que es dependiente de la máquina en uso. La portabilidad radica en que se pueden programar las aplicaciones solo una vez y ejecutar en cualquier plataforma. Cuando se compila un programa Java en una plataforma Windows/Intel, se obtiene la misma salida compilada (o los mismos bytecodes) que en un sistema Macintosh o Unix.
- **Altas prestaciones:** No se pierde tiempo optimizando código que no se ejecutará.
- **Multihilo:** Permite la ejecución de varias tareas a la vez.
- **Dinámico:** No conecta todos los módulos que comprende una aplicación hasta el tiempo de ejecución.

La mayoría de los lenguajes de programación se caracterizan por ser interpretados o compilados, lo que determina la manera en que serán ejecutados en una computadora. Java tiene la característica de ser al mismo tiempo compilado e interpretado. El compilador es el encargado de convertir el código fuente de un programa a bytecodes, que son independientes de la plataforma en que se trabaje y que es ejecutado por el intérprete de Java, que forma parte de la Máquina Virtual de Java.

La programación en Java permite el desarrollo de aplicaciones tanto bajo el esquema Cliente/Servidor como de aplicaciones distribuidas, que lo hace capaz de conectar dos o más computadoras ejecutando tareas simultáneamente.

1.6.3 La Interfaz de Programación de Aplicaciones (API).

La API de Java está formada por una amplísima jerarquía de clases ya desarrolladas que ofrecen un gran abanico de posibilidades al programador, cubriendo una gran cantidad de

aspectos relacionados con el desarrollo de software en general. La API es vastísima, está organizada en paquetes (packages) ordenadas por temas. La nomenclatura de los paquetes es uniforme y ayuda a categorizar las clases.

El J2SE permite la utilización de todos estos packages en el desarrollo de programas Java y el JRE (Java Runtime Environment) permite la ejecución de aplicaciones que usan cualquiera de las clases del API. Para cada especificación de la plataforma existe una API.

1.6.4 La Máquina Virtual Java (JVM).

Recibe este nombre porque es una máquina imaginaria que se implementa emulando por software una máquina real. Es un programa ejecutable para una plataforma específica, capaz de interpretar y ejecutar el bytecode, el cual es generado por el compilador del lenguaje Java. Su misión principal es la de garantizar la portabilidad de las aplicaciones Java.

Las tareas principales de la JVM son las siguientes:

- Reservar espacio en memoria para los objetos creados.
- Liberar la memoria no usada (garbage collector).
- Asignar variables a registros y pilas.
- Llamar al sistema huésped para ciertas funciones, como los accesos a los dispositivos.
- Vigilar el cumplimiento de las normas de seguridad de las aplicaciones Java.

En la JVM se encuentra el motor que en realidad ejecuta el programa Java y es la clave de muchas de las características principales de Java, como la portabilidad, la eficiencia y la seguridad.

1.6.5 ¿Por qué elegir Java?

Java ha sido mejorado, ampliado y probado por una comunidad especializada de más de 5 millones de desarrolladores, la mayor y más activa del mundo. Gracias a su versatilidad, eficiencia y portabilidad, Java se ha convertido en un recurso inestimable ya que permite a los desarrolladores:

- Desarrollar software en una plataforma y ejecutarlo en prácticamente cualquier otra.
- Crear programas para que funcionen en un navegador web y en servicios web.
- Desarrollar aplicaciones para servidores como foros en línea, tiendas, encuestas, procesamiento de formularios HTML, etc.
- Combinar aplicaciones o servicios que usan el lenguaje Java para crear servicios o aplicaciones totalmente personalizados.
- Desarrollar potentes y eficientes aplicaciones para teléfonos móviles, procesadores remotos, productos de consumo de bajo coste y prácticamente cualquier tipo de dispositivo digital.

Por todas las características anteriormente mencionadas, es que se propone el empleo de esta plataforma que integra todo lo necesario para que el desarrollo de la aplicación en cuestión sea favorable, sin costes elevados y ajustado a cronogramas.

1.7 Frameworks a utilizar.

En el desarrollo de software, un framework o marco de trabajo es una estructura de soporte definida sobre la cual un proyecto puede ser organizado y desarrollado. Puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre aplicaciones para ayudar a desarrollar y unir los diferentes componentes de un proyecto. Representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio. Los frameworks son diseñados con la intención de facilitar y agilizar el desarrollo de software, permitiendo a los diseñadores y programadores pasar más tiempo identificando requerimientos de software que tratando con los tediosos detalles de bajo nivel de proveer un sistema funcional.

1.7.1 Prototype.

Prototype es un framework que facilita el desarrollo de aplicaciones web con JavaScript y AJAX. Su autor original es Sam Stephenson, aunque las últimas versiones incorporan código e ideas de muchos otros programadores. A pesar de que incluye decenas de utilidades, la librería es compacta y está programada de forma muy eficiente. La principal particularidad de este framework, es que acelera y facilita el acceso a funciones de JavaScript, ganando un

código ordenado y legible. Con la utilización de Prototype se gana en facilidad y agilización del proceso de implementación de este tipo de aplicaciones, permite reutilizar código y promueve buenas prácticas de desarrollo, a la vez que resuelve problemas comunes de compatibilidad entre navegadores. Recientes encuestas lo categorizan como el framework más popular para programar en JavaScript.

1.7.2 Spring Framework.

El Spring Framework, más conocido simplemente como Spring, es un framework de código abierto para el desarrollo de aplicaciones para la plataforma Java, aunque también hay una versión para la plataforma .NET, Spring.net.

A pesar de que Spring Framework no obliga a usar un modelo de programación en particular, se ha popularizado en la comunidad de programadores en Java al considerársele una alternativa al modelo de Enterprise JavaBean. Este framework ofrece a los desarrolladores en Java mucha libertad, soluciones muy bien documentadas y fáciles de usar para las prácticas comunes en la industria, además de un diseño y arquitectura sólidos y confiables, basados en las mejores prácticas y patrones de programación aceptados en la actualidad, como son el patrón MVC y el patrón Inyección de Dependencia. También incluye amplias capacidades de integración con otros frameworks como JSP, Velocity, XSLT y Struts.

Spring está compuesto por siete módulos principales que colaboran y brindan una amplia gama de funcionalidades fáciles de emplear, que garantizan una arquitectura robusta para cualquier proyecto que tenga su basamento en él. La flexibilidad con que está diseñado e implementado permite escoger y utilizar sólo los módulos necesarios para el desarrollo de la aplicación e ignorar el resto.

- **Módulo núcleo (Core Container and Supperting Utilities):** Contiene la Fábrica de Clases (BeanFactory) que es el corazón de toda aplicación sobre Spring. BeanFactory aplica la Inversión de Controles (Invertion of Control o IoC), específicamente la Inyección de Dependencias (Dependency Injection), para separar la configuración de la aplicación y sus especificaciones de dependencias entre objetos de la lógica de negocio de la aplicación.

- **Módulo AOP (Aspect-Orient Programming):** Brinda soporte a la Programación Orientada a Aspectos, con la que es posible manejar y solucionar problemáticas como las auditorías y las transacciones de datos con la Base de Dato o entre capas

de abstracción dentro de la aplicación, manteniendo la legibilidad y limpieza en el código que define la lógica de negocio de la aplicación en cuestión.

- **Módulos ORM y JDBC Abstraction and DAO:** Brindan soporte a la gama de funcionalidades y problemáticas comunes en el trabajo con Bases de Datos y abstraen al programador de las acciones básicas de intercambio con la Base de Dato permitiendo que sea posible mantener el código de acceso a datos limpio y simple. Además, ORM implementa mecanismos de enganches para lograr compatibilidad e integración con otros frameworks que soportan el mapeo de objetos relacionales como Hibernate e iBatis.

- **Módulo MVC (Model – View – Controller):** Brinda soporte al desarrollo de aplicaciones web basando el intercambio de los clientes (dígase navegadores web) con el servidor, en la filosofía que sigue el patrón arquitectónico del mismo nombre. Esta facilidad también emplea la IoC para lograr separar limpiamente la lógica de controlador de los objetos de negocio, también permite unir declarativamente, parámetros de una petición a objetos de negocio. Además incorpora un mecanismo de validación de formularios muy útil en la comprobación de los datos provenientes del cliente. (10)

1.7.3 DWR (Direct Web Remoting).

DWR es una librería de código abierto que permite incorporar tecnología AJAX a las aplicaciones WEB desarrolladas en Java. Permite que el código del navegador web use funciones Java que se ejecutan en el servidor como si se ejecutaran desde el navegador. DWR está separado en 2 partes principales: Por un lado existe una parte que consiste en código JavaScript que se encarga de la comunicación con el servidor y de la actualización dinámica de la página. Por otro lado, en el servidor hay un servlet que recibe los requests provenientes de este código JavaScript, los procesa y devuelve la respuesta.

DWR genera dinámicamente código JavaScript basado en las clases de Java. Así, el programador puede acceder a las clases java desde el cliente a través de JavaScript.

Este framework tiene una larga lista de usuarios y es usado por muchos proyectos como el sitio web de la cadena de tiendas Wal-Mart y el sitio de reservaciones de vuelos de la American Airlines, por poner algunos ejemplos.

DWR presenta un gran numero de características y funcionalidades que lo hacen atractivo para los desarrolladores de aplicaciones web con Java, como la conversión de virtualmente cualquier tipo de estructura de datos entre Java y JavaScript (cadenas, listas, objetos e incluso archivos binarios cargados o descargados), manejo de excepciones, avanzada protección CSRF (Cross Site Request Forgery), a la vez que permite profunda integración con varias tecnologías Java del lado del servidor como Spring Framework.

La utilización de bibliotecas como DWR tiene grandes beneficios sobre la implementación de frameworks AJAX propios. El principal es la portabilidad. Generalmente el JavaScript utilizado tiene funciones que son dependientes de los navegadores y que, por lo tanto, generan cierta incompatibilidad entre plataformas. El uso de un motor AJAX permite abstraer estas complicaciones y utilizar las funciones provistas por el framework dejando que este se ocupe de las cuestiones particulares de cada navegador.

1.7.4 Ibatis.

Es un marco de trabajo especializado en dar soporte a las operaciones y problemáticas más comunes referentes a las interacciones entre el acceso a datos de una aplicación y la base de datos correspondiente. Resulta un componente de software encargado de traducir entre objetos y registros, basado en capas, situado entre la lógica de negocio y la capa de origen de datos. iBATIS asocia objetos de modelo (JavaBeans) con sentencias SQL o procedimientos almacenados mediante ficheros descriptores XML, simplificando la utilización de bases de datos. Está compuesto de dos paquetes complementarios pero independientes: IBatis Data Access Objects (DAO), que implementa la capa de abstracción (ibatis-dao.jar) e iBatis SQL MAPS, que implementa la capa de persistencia (ibatis-sqlmap.jar). IBatis Data Access Objects (DAO) oculta los detalles de la capa de persistencia y proporciona un API común para todas las aplicaciones. iBatis Data Mapper proporciona un modo simple y flexible de mover los datos entre los objetos Java y la base de datos relacional, se tiene toda la potencia de SQL sin una línea de código JDBC (Java DataBase Connectivity). (11) Para su uso no requiere de grandes esfuerzos de aprendizaje, consta de buena documentación y es un proyecto de código abierto. Ofrece un sistema de mapeo directo en ficheros XML. Proporciona mecanismos para el acceso a procedimientos almacenados, con la debida consideración por parte del programador de no implementar lógica de negocio en los mismos.

1.8 Herramientas de Desarrollo.

Las herramientas de desarrollo de software permiten a los desarrolladores la creación de aplicaciones para sistemas concretos. Las más comunes incluyen técnicas de soporte para la detección de errores de programación, la generación automática de código, entre otras características que facilitan y agilizan el desarrollo de soluciones de software. Frecuentemente incluyen también códigos de ejemplo, notas técnicas y documentación de soporte.

1.8.1 Visual Paradigm.

Visual Paradigm es una poderosa herramienta profesional para el modelado de sistemas y soluciones de software usando UML. Soporta todo el ciclo de desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite crear diagramas de clases y otros artefactos necesarios a lo largo del proceso de desarrollo de un software.

Se integra fácilmente con varios IDEs de programación como Eclipse y Visual Studio .Net. Brinda la posibilidad de documentar los artefactos generados, sin la necesidad de utilizar herramientas externas. Posibilita la generación semiautomática de código a partir de los diagramas, también se puede llevar a cabo la ingeniería inversa para refinar los modelos. Está disponible en muchas plataformas. Acelera el desarrollo del software completo obteniendo productos de calidad reduciendo costos y riesgos.

1.8.2 Eclipse.

Eclipse es un entorno de desarrollo integrado de código abierto, multiplataforma basado en Java. Fue desarrollado originalmente por IBM, pasando luego a constituir la Comunidad Eclipse, una organización independiente que promueve el código abierto y un conjunto de servicios y productos suplementarios. En sí mismo Eclipse es un marco y un conjunto de servicios para construir un entorno de desarrollo a partir de componentes conectados (plug-in). Eclipse contiene una serie de perspectivas que proporcionan funcionalidades y facilidades para el desarrollo de un tipo específico de tarea.

La característica clave de Eclipse es la extensibilidad. Eclipse es una gran estructura formada por un núcleo y muchos plug-ins que van conformando la funcionalidad final. La forma en que los plug-ins interactúan es mediante interfaces o puntos de extensión; así, las nuevas aportaciones se integran sin dificultad ni conflictos. Está compuesto de tres

subproyectos: la Plataforma Eclipse, la Java Development Tool y el Plug-in Development Environment. El éxito de la Plataforma Eclipse depende de cómo sea capaz de admitir una amplia gama de herramientas de desarrollo para reproducir lo mejor posible las herramientas existentes en la actualidad. (12)

En el desarrollo de la aplicación se incluyen un conjunto de plug-ins que amplían sus funcionalidades como plataforma de desarrollo de aplicaciones sobre la web. Los plug-ins más importantes son los siguientes.

- Web Tool Platform (WTP): Para soportar todas las funciones necesarias para desarrollar aplicaciones web sobre J2EE.
- Spring IDE: Para facilitar el uso sobre los beans y xml definidos por Spring Framework.
- Subeclipse: Para permitir de una forma mucho más ágil y cómoda el desarrollo colaborativo de software en un equipo de desarrolladores.

1.9 Conclusiones.

En este capítulo se sustentó teóricamente el trabajo realizado y se describieron las características de la situación problémica que lo hacen necesario. Además se establecieron las bases para el desarrollo de la aplicación, al explicarse las tecnologías sobre las que estará soportado. Se realizó un análisis detallado de las mismas, y de los beneficios que aportaría cada una al desarrollo de la solución propuesta, lo cual quedó plasmado en el contenido de este capítulo.

De las tecnologías SIG presentadas durante el capítulo, se seleccionó para el desarrollo del Sistema de Mapificación Web el MapInfo MapXtreme para Java, pues además de ser multiplataforma y compatible con los principales servidores web y de aplicación, presenta un framework completo, gran cantidad de documentación y aplicaciones de ejemplo, es fácil de usar y altamente configurable, de alta calidad y gran rendimiento.

Se escogió como metodología de desarrollo de software a RUP (Rational Unified Process) y como herramienta de modelado a Visual Paradigm. La aplicación estará soportada sobre la Plataforma Java. Los Frameworks a utilizar serán Spring, Ibatis, DWR y Prototype, sobre el entorno de desarrollo integrado (IDE) Eclipse 3.4.1.

Capítulo 2. Diseño del Sistema.

2.1 Introducción.

En este capítulo se presenta el diseño del Sistema de Mapificación Web. Se describe la arquitectura base del sistema y se presenta el diagrama de paquetes que se propone para estructurar los elementos del diseño. Se muestran además los diagramas de clases de cada uno de estos paquetes divididos por capas, y la descripción de las principales clases del diseño.

En el contenido se muestra el diseño elaborado con antelación y como este ha sido refinado y mejorado, utilizando las potencialidades de los frameworks presentados en el capítulo anterior, obteniéndose así una propuesta más robusta y eficiente.

2.2 Arquitectura.

La arquitectura es el esqueleto o base de una aplicación. En esta aparecen los artefactos más significativos, para establecer un esquema de cómo deben ser los próximos artefactos a construir, o sea, es puramente un semi-molde al que se deben ajustar sin muchos cambios los restantes artefactos a construir en la aplicación o solución.

Establecer las bases sólidas que soporten la construcción, escalabilidad, correcto funcionamiento y mantenimiento de una aplicación informática, requiere un estudio detallado de los patrones seguidos a lo largo del desarrollo de software, las buenas prácticas de programación, el entorno de desarrollo de la misma, las prestaciones que ha de satisfacer y los requerimientos necesarios para su puesta en marcha. La descripción de la arquitectura define los principales aspectos tomados en cuenta en el diseño del sistema.

2.3 Valoración de la opción de arquitectura Cliente-Servidor de la propuesta anterior.

Para el desarrollo de una aplicación web utilizando MapInfo MapXtreme se proponen tres tipos de configuración de la arquitectura: Cliente Ligero, Cliente Pesado y Cliente Intermedio, que se diferencian entre si en la forma en que los datos son manejados por el cliente y el servidor.

La arquitectura Cliente Ligero es la opción más común de las tres, puesto que el cliente sólo necesita un navegador web para acceder a la aplicación y el uso de plug-ins no es requerido. El mapa constituye una imagen generada en el servidor, el cual recibe toda la responsabilidad de la aplicación, lo cual garantiza un tránsito de red mínimo y una carga de trabajo en el cliente prácticamente nula.

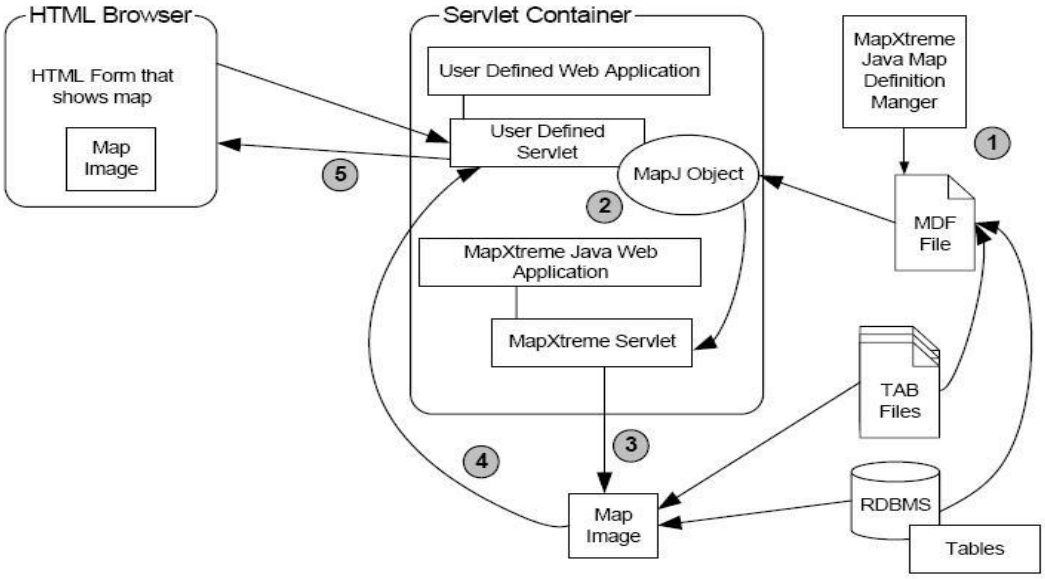

Fig. 2.1. Arquitectura Cliente Ligero.

En la opción Cliente Pesado la lógica de negocio y los datos de la aplicación residen en el lado del cliente, generalmente mediante la descarga de un Applet de Java, el cual recibe toda la carga de trabajo. Esta configuración permite utilizar herramientas visuales equivalentes a las utilizadas en aplicaciones de escritorio y los mapas son generados a una velocidad mucho mayor, pero tiene como desventaja que genera un tráfico de red muy superior y un incremento de los requerimientos del cliente, quien debe ejecutar toda la lógica de negocio y necesita soporte para Java y plug-ins.

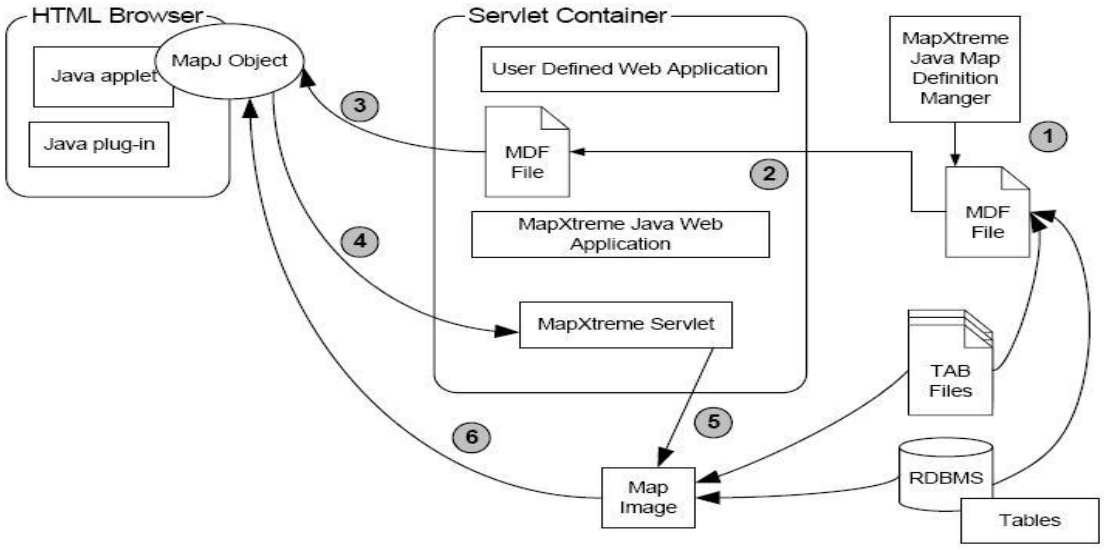

Fig. 2.1. Arquitectura Cliente Pesado.

La configuración Cliente Intermedio combina las características de las dos anteriores. El cliente descarga un Applet de Java que le permite utilizar herramientas visuales, aunque la generación del mapa y la lógica de negocio ocurren en el servidor. Esto trae consigo una disminución de los requerimientos del cliente, ya que el grueso de la carga de trabajo tiene lugar en el servidor, pero aún así se requiere soporte para Java y plug-ins para el navegador.

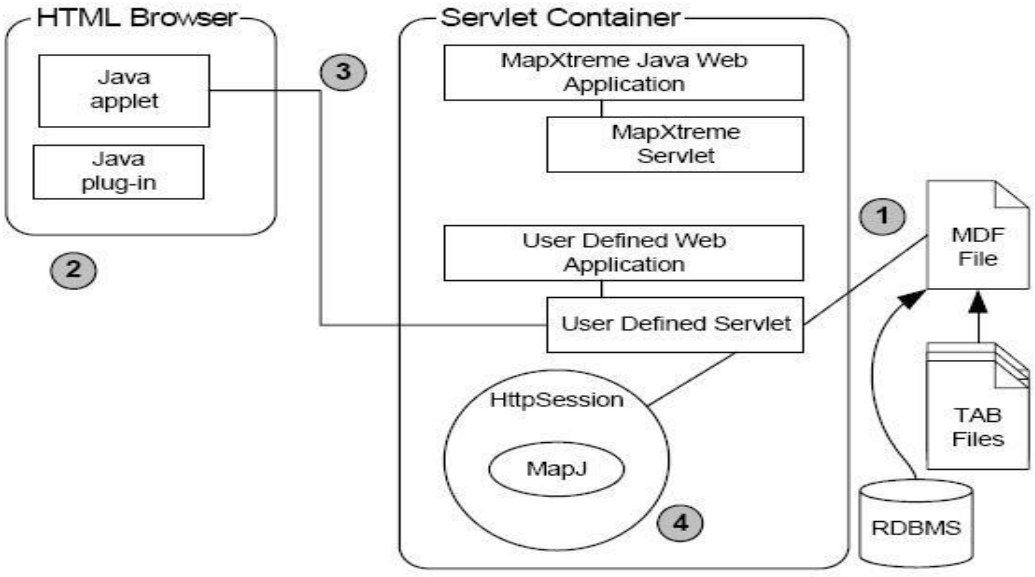

Fig. 2.3. Arquitectura Cliente Intermedio.

En la propuesta anterior se seleccionó la configuración Cliente Ligero, la cual constituye la más idónea a aplicar en el Sistema de Mapificación Web, puesto que permite ser usada por cualquier navegador sin necesidad de otros requerimientos como plug-ins o soporte para Java. Esta arquitectura aporta ventajas significativas como un tráfico de red mínimo, un volumen de trabajo en el cliente casi inexistente y un control completo de la aplicación en el lado del servidor. Las razones antes expuestas son suficientes para ratificar esta opción de configuración como la más propicia a utilizar en el sistema.

2.4 Valoración del diseño anterior.

En el diseño y especificación de la estructura global del sistema hay que señalar algunos desaciertos en los patrones y abstracciones empleados, que lejos de ofrecer un marco de construcción robusto, puede introducir desventajas en la mantenibilidad y flexibilidad del sistema, así como constituir un freno ante la escalabilidad del mismo.

Al analizar la arquitectura anterior se evidencia que no presenta una distribución de las clases en capas de forma explicita, se puede identificar en algunas el nivel al que pertenecen mientras otras tienen un alcance un tanto difuso. Esto trae como consecuencia una pobre capacidad de reutilización y adaptación, a la vez que genera confusión en la distribución de las tareas para su implementación. El reparto de responsabilidades es una práctica ineludible para conseguir flexibilidad, escalabilidad y claridad en el código.

2.5 Presentación del nuevo diseño.

La nueva propuesta de arquitectura del Sistema de Mapificación Web se corresponde con el patrón arquitectónico Layers (Capas). Está conformada por Objetos del Dominio (Domain) y las capas de Presentación, Negocio y Acceso a Datos. La utilización de frameworks especializados permite realizar abstracciones y definir de una forma precisa la separación entre las capas.

2.5.1 Modelo en Capas.

El patrón de arquitectura en capas (Layers) tiene como objetivo fundamental dividir la aplicación en capas lógicas atendiendo a las responsabilidades de las clases. En este tipo de arquitectura, a cada nivel se le asigna una tarea simple, lo que permite un diseño escalable, que puede ampliarse con facilidad en caso de que las necesidades aumenten. Cada capa o nivel esta constituido por un conjunto de clases acopladas entre sí. Cada capa, para

completar su funcionamiento, delega tareas a la capa que se encuentra por debajo en la jerarquía, brindándole los datos necesarios para la realización de las mismas y solicitando de ella la información requerida. Una de las ventajas de este patrón es que el desarrollo se lleva a cabo en niveles, y en caso de que sobrevenga algún cambio, este sólo afectaría al nivel requerido sin influir en los demás.

A continuación se muestra cómo está estructurada la arquitectura del sistema y posteriormente se explican cada una de sus partes.

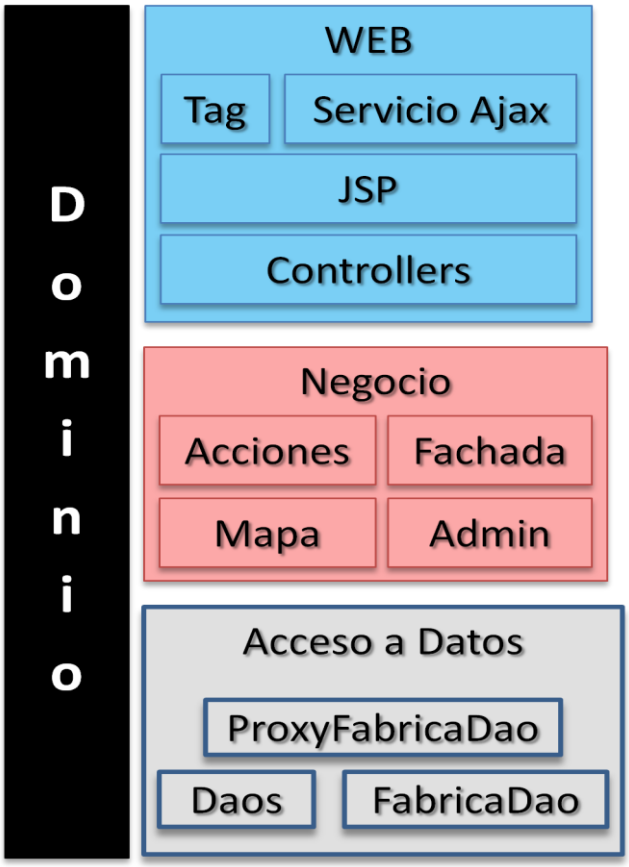

Fig. 2.4. Arquitectura en Capas.

Para lograr una mejor comprensión se comenzará la explicación por la capa más alta en la jerarquía, siguiendo el flujo de las acciones que se desencadenan con una petición del usuario, pero para ello se hace necesario conocer la razón por la cual las clases del dominio (DOMAIN), se representa verticalmente en la figura 2.4.

2.5.2 Dominio.

El dominio esta formado por un conjunto de clases que representan de manera abstracta objetos del mundo real. Las mismas contienen información que el sistema debe manipular a través de las capas, dándole en cada una de ellas el uso necesario. Dado que las demás capas se van a encargar de hacer cumplir las restricciones del negocio, el dominio constituye el soporte para la transferencia de datos desde el acceso a datos hasta la presentación y viceversa.

2.5.3 Capa de Presentación.

La capa de presentación descansa sobre una capa de servicios de negocio. Esto significa que esta capa será fina y no contendrá lógica de negocio, sino simplemente lo concerniente a aspectos de presentación, por ejemplo, el código para manipular las interacciones web.

La base de la presentación son las JSP (Java Server Pages), que consisten en una tecnología Java capaz de generar contenido dinámico para la web, en forma de páginas HTML. Las JSP permiten la utilización de código Java mediante scripts. Además es posible utilizar algunas acciones JSP predefinidas mediante etiquetas. Estas etiquetas pueden ser enriquecidas mediante la utilización de Librerías de Etiquetas (TagLibs o Tag Libraries) externas e incluso personalizadas.

Las etiquetas personalizadas (Tags) son marcas que indican al navegador lo que debe mostrar y cómo mostrarlo. Las ventajas que proveen son eliminar código repetitivo, potenciar la reutilización y acelerar el desarrollo de las páginas. En la capa de presentación se utiliza una librería de Tags que tiene como función encapsular el código HTML y JavaScript que dispara las peticiones al servidor, el cual utiliza el Spring Web MVC Framework para atender dichas peticiones.

Spring Web MVC Framework, es una implementación del patrón arquitectónico Modelo Vista Controlador (MVC). En el módulo MVC que implementa Spring se define al DispatcherServlet como el controlador frontal de la aplicación, quien es el encargado de atender todas las peticiones que se le hagan al servidor. El componente responsable de manipular la petición en el MVC de Spring es un Controller.

Para que el DispatcherServlet logre identificar qué Controller manipulará la petición actual, este se auxilia del HandlerMapping, El HandlerMapping es el encargado de mapear los patrones de URL con el objeto Controller asociado. Una vez que el DispatcherServlet conoce

el objeto Controller, le delega la responsabilidad de atender la petición. El Controller realiza la función para la que fue diseñado y devuelve un objeto ModelAndView al DispatcherServlet con el objeto View o el nombre lógico del mismo. Si el objeto ModelAndView contiene el nombre lógico del View, entonces el DispatcherServlet consulta al ViewResolver para localizar al objeto View que visualizará la respuesta (response) al cliente.

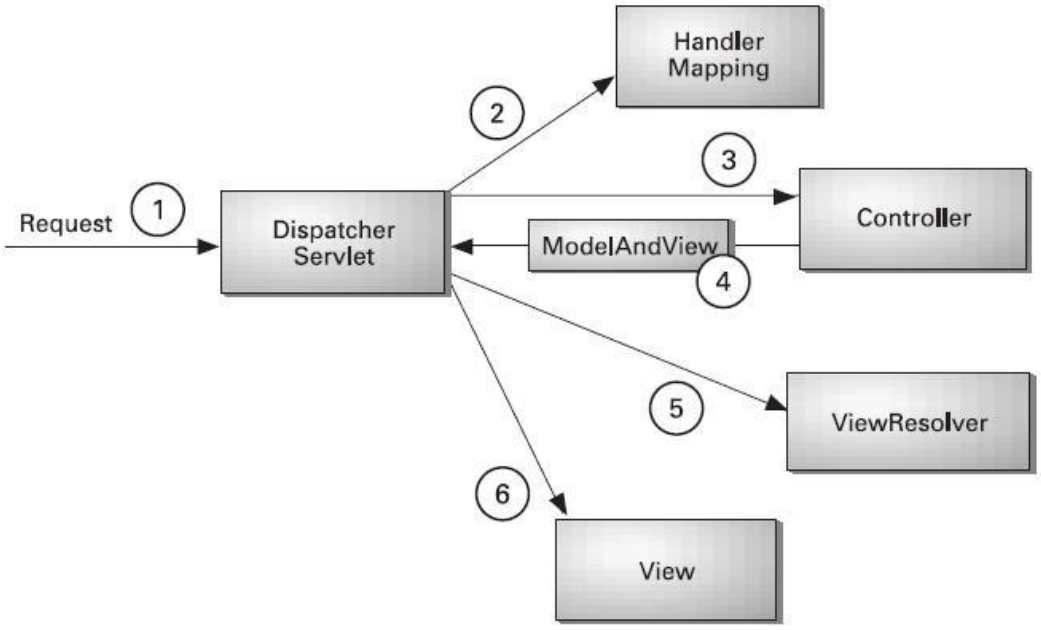

Fig. 2.5. Spring MVC.

En su módulo MVC, Spring implementa una serie de controladores personalizados que facilitan la construcción de nuevos controladores. Entre todos ellos se escogió el controlador AbstractCommandController para la creación del controlador MapaCommandController, que es la base de todos los Controllers utilizados en la aplicación.

Se escogió el AbstractCommandController por su capacidad de recibir en la petición una serie de parámetros y transformarlos en atributos de un objeto (command), que encapsula los datos de la petición.

El Controller personalizado MapaCommandController hereda las funcionalidades del AbstractCommandController y además incluye la referencia a la interfaz IMapa que representa el comportamiento de un mapa real. Este Controller tiene también la capacidad de crear el objeto command, y transformarlo en la acción que atenderá la petición utilizando la clase FabricaAcciones. El controlador solo conoce de las acciones que implementan el

método **ejecutar (IMapa mapa),** de la interfaz IAccionVisual o IAccionInformativa respectivamente, lo cual garantiza un bajo acoplamiento entre clases. Las acciones y la interfaz IMapa son las interfaces de comunicación entre las capas de presentación y negocio.

A partir de las funcionalidades expuestas por la interfaz de la capa de negocio, se construye el modelo correspondiente a la vista que se ha de mostrar finalmente al usuario. Los mismos se van a auxiliar de clases validadoras que chequearán que los objetos del dominio posean la información correcta y necesaria para poder realizar el proceso del negocio en que están inmersos.

- **Modelo:** Estos objetos contienen los datos resultantes de la ejecución de la lógica de negocio, los cuales deberían ser mostrados en la respuesta.

- **Vistas:** Estos objetos son responsables de mostrar el modelo en la respuesta de la petición. Las vistas no son responsables de modificar los datos o incluso de obtener los datos; estas simplemente sirven para mostrar los datos del modelo que han sido suministrados por un controlador.

El empleo de AJAX posibilita efectuar modificaciones a las páginas HTML mediante peticiones asincrónica al servidor Web y así obtener actualizaciones dinámicas del contenido. Con el empleo de la librería DWR, se obtienen las abstracciones concebidas como Servicio AJAX, responsable de complementar solicitudes hechas por el usuario, a través de llamadas a métodos o funciones de invocación remota y traducción de tipos de variables no primitivos del lenguaje Java. Es elegante y sencilla la interacción desde el código JavaScript de la página cargada por el navegador Web con las funcionalidades del negocio expuestas por la Fachada.

2.5.4 Capa de Negocio.

Las clases del negocio implementan la lógica de negocio de la aplicación. Son las que en realidad ejecutan las acciones y controlan el flujo de eventos e información del sistema. Están divididas en cuatro grupos: Mapas, Acciones, Fachadas y Admins. A continuación se explica el funcionamiento general de cada uno.

El Mapa constituye el núcleo del sistema. Solo existe uno por cada sesión. Tiene la tarea de encapsular toda la información con que se está trabajando. Es la representación digital del mapa que el usuario observa y guarda toda la información de este en un objeto de la clase

MapJ del framework MapXtreme. Todas las acciones que el usuario ejecuta sobre el mapa tienen su base en este objeto.

Las Acciones son las interfaces definidas para dar soporte a las funcionalidades del Sistema de Mapificación Web. Sus implementaciones son dos clases abstractas, que representan los dos tipos de operaciones que se pueden efectuar con el mapa: actuar sobre él (AccionVisual) o consultar su información (AccionInformativa). Estas clases, se convierten en la vía de comunicación entre la capa de negocio y la de presentación (implementación del Patrón Facade). Además, sus hijas encapsulan no solamente la información de la petición hecha por el usuario, sino que también definen la forma de ejecutar dicha orden (implementación del Patrón Command).

Las acciones se crean dinámicamente una vez que las peticiones son atendidas por el Controller, utilizando la clase BeanFactory que proporciona el Spring Framework. La clase encargada de decir al Controller el tipo de acción que debe contestar a la petición hecha por el usuario es la FabricaAcciones (implementación del Patrón Factory). Esta utiliza un XML donde se relacionan las peticiones con las acciones que deben ejecutarlas. Además, en el caso de las acciones informativas, guardan la vista que debe ser mostrada al usuario. Para optimizar el rendimiento se crea solo una instancia de la clase FabricaAcciones para toda la aplicación (implementación del Patrón Singleton).

En Fachada se encuentran las interfaces de comunicación que brindan soporte a la capa de presentación para las peticiones asincrónicas realizadas con DWR (implementación del Patrón Facade). Es una solución de diseño que busca mantener un bajo acoplamiento. Estas delegan su labor a los Admin especializados en gestionar la información que necesitan.

El Admin agrupa la lógica necesaria para responder a las llamadas asincrónicas. Esta separado de la capa de presentación por la Fachada, lo cual garantiza que no se exponga lógica de negocio, lo que representaría un serio problema de seguridad.

2.5.5 Capa de Acceso a Datos.

La capa de Acceso a Datos es la encargada de la comunicación con la base de datos, permitiendo abstraer y separar la lógica del negocio de la aplicación de la forma en que se almacenan físicamente los datos. La implementación de esta capa se basa en el ampliamente utilizado patrón de persistencia DAO (Data Access Object, Objeto de Acceso a Datos).

Los DAOs son las clases responsables de hacer persistir los objetos del dominio, constituyen una abstracción entre la capa de persistencia (base de datos) y la capa de negocio, y encapsulan la forma en que ambas se relacionan. Su uso posibilita ocultar completamente los detalles de implementación de la fuente de datos a sus clientes. Las implementaciones de los DAOs se instancian mediante el uso de la clase FabricaDao (implementación del patrón AbstractFactory), y son accedidas por los objetos del negocio mediante una instancia de la clase ProxyFabricaDao (implementación del patrón Proxy).

La capa de Acceso a Datos esta soportada sobre el framework Ibatis, que ofrece un sistema de mapeo directo en ficheros XML y proporciona mecanismos para el acceso a procedimientos almacenados, disminuyendo drásticamente la cantidad de código necesario para comunicar con la base de datos y separando por completo el código Java de las sentencias SQL y los llamados a procedimientos almacenados.

2.6 Diagrama de clases del diseño.

El diagrama general de clases del diseño representa la interacción entre las clases de las diferentes capas por las que está integrado el Sistema de Mapificación Web. Se ha utilizado como elemento característico el color definido en la figura 2.1 para identificar las clases pertenecientes a cada capa. Las clases representadas con el color verde indican las clases que proporcionan los frameworks utilizados y se representan para mayor comprensión del diagrama.

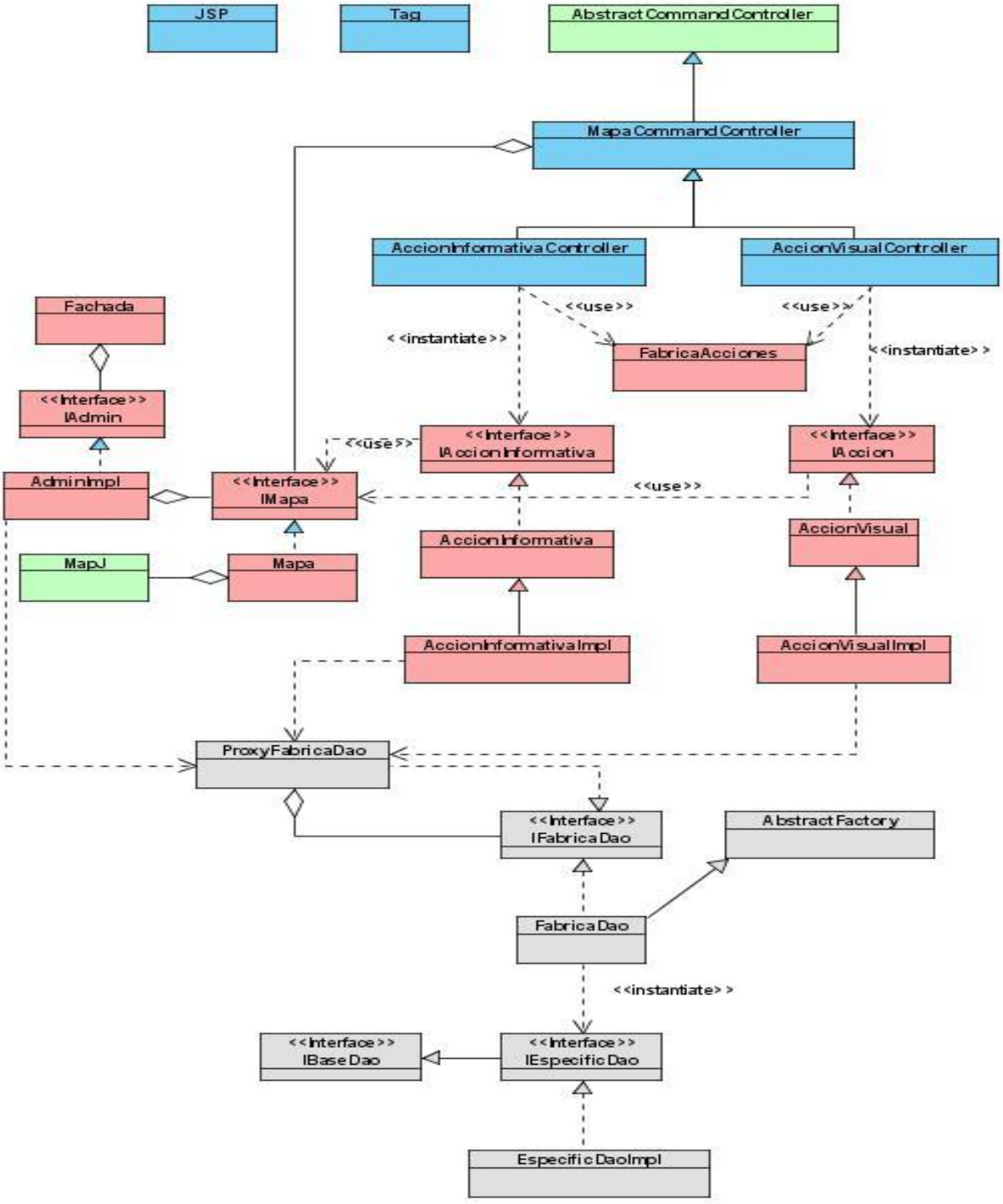

Fig. 2.6. Diagrama de clases del diseño.

2.7 Diagrama de Paquetes del diseño.

Los diagramas de paquetes se usan para organizar el sistema en elementos más pequeños y fáciles de interpretar. Estos muestran cómo un sistema está dividido en agrupaciones

lógicas mostrando las dependencias entre esas agrupaciones. Un paquete de diseño es una colección de clases, relaciones, realizaciones de casos de usos, diagramas y otros paquetes. Es usado para estructurar el modelo de diseño mediante su división en partes más pequeñas y agrupar elementos relacionados de dicho modelo con propósitos organizacionales.

El diagrama de paquetes mostrado a continuación fue la propuesta inicial diseñada para la evolución del sistema. Es un diagrama que garantiza la alta cohesión de las estructuras y componentes que lo conforman. Para elaborarlo, el criterio escogido fue agrupar los casos de uso requeridos para dar soporte a un determinado proceso de negocio.

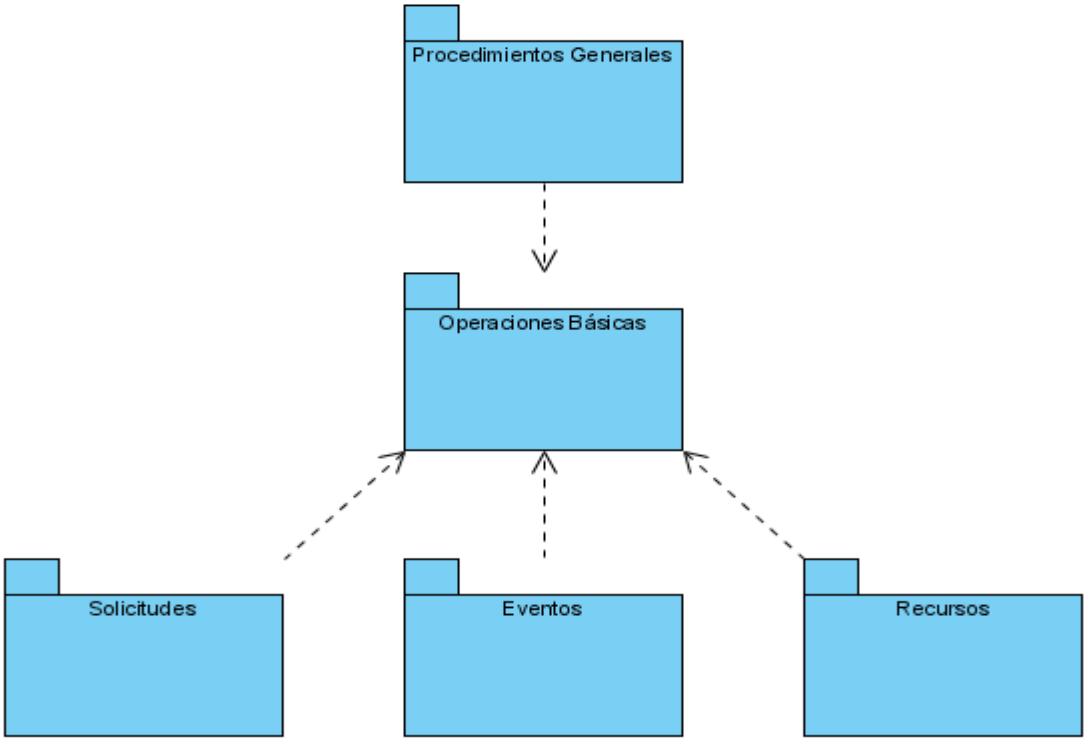

Fig. 2.7. Diagrama de paquetes de la propuesta anterior.

Para la nueva propuesta se decidió agrupar las clases atendiendo a su funcionalidad. Esto debe garantizar un entendimiento mas detallado de la organización de las clases y de la arquitectura propuesta. Para disponer las colaboraciones entre los paquetes, se analiza la relación de uso que se establece entre uno y otro; el paquete A depende de B cuando invoca los procedimientos o métodos que se encuentran en B.

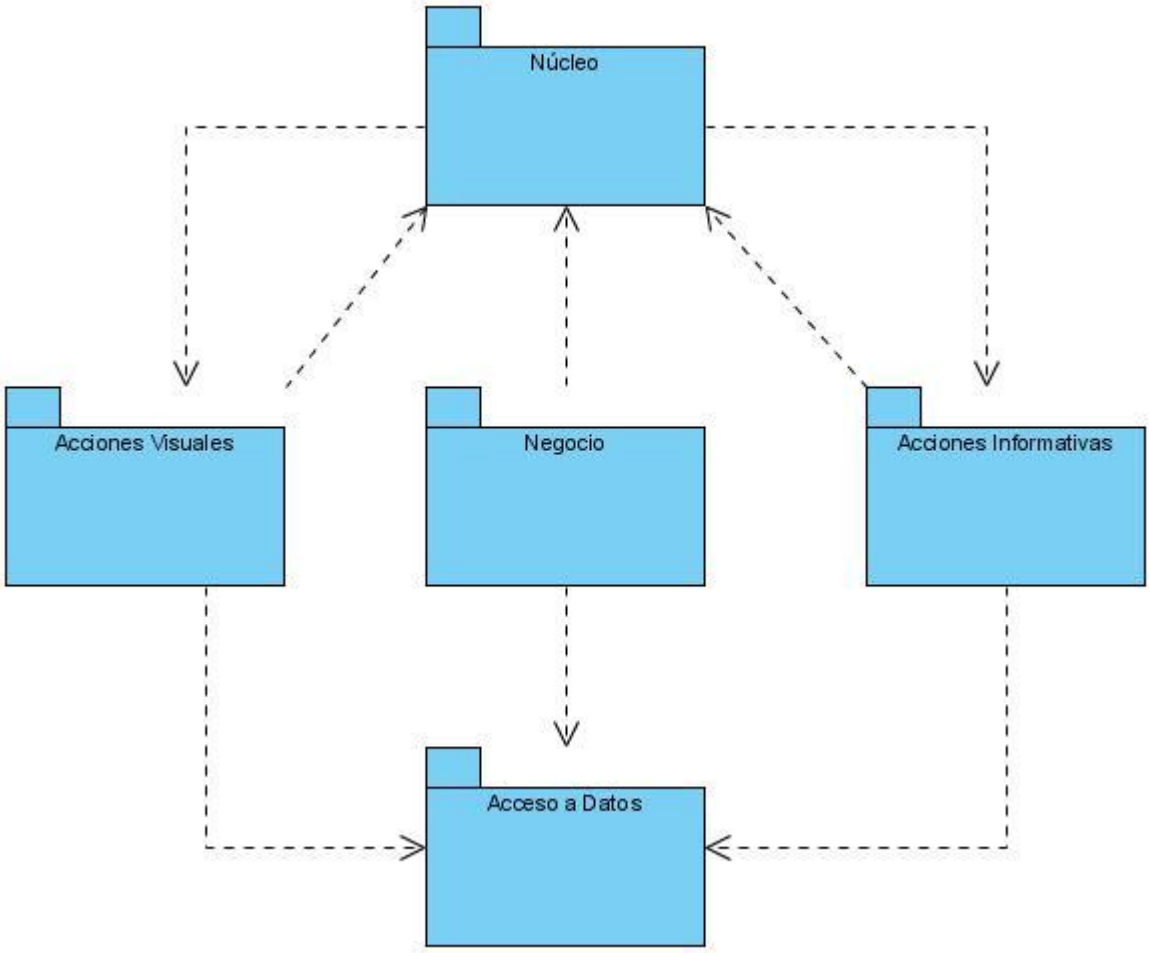

Fig. 2.8. Diagrama de paquetes de la nueva propuesta.

A continuación se describe brevemente la funcionalidad y funcionamiento de cada clase. En la descripción sólo se especificarán los atributos y métodos principales. En cada clase, por cada atributo definido, existirán los métodos **get** y **set** que aunque no se describan estarán presentes.

2.7.1 Paquete Núcleo

El paquete Núcleo es, como su nombre lo indica, el paquete principal del Sistema de Mapificación Web. Funciona como cerebro y corazón del sistema, ya que no solo es el encargado de atender a las peticiones y delegar responsabilidades para su cumplimiento, sino que gestiona la información concreta del mapa mostrado al usuario.

Dicho paquete contiene la clase principal de la aplicación, la clase Mapa, así como su interfaz de comunicación con los demás paquetes del sistema, la interfaz IMapa. La

funcionalidad de este paquete es gestionar la información del mapa que se muestra al usuario. Para esto se utiliza la clase MapJ proporcionada por el framework de MapXtreme.

El paquete Núcleo también contiene la clase encargada de generar las acciones que responden a las peticiones del usuario, la clase FabricaAcciones, así como las clases encargadas de atender dichas peticiones y delegarlas a las acciones, la clase MapaCommandController y sus clases descendientes: AccionVisualController y AccionInformativaController.

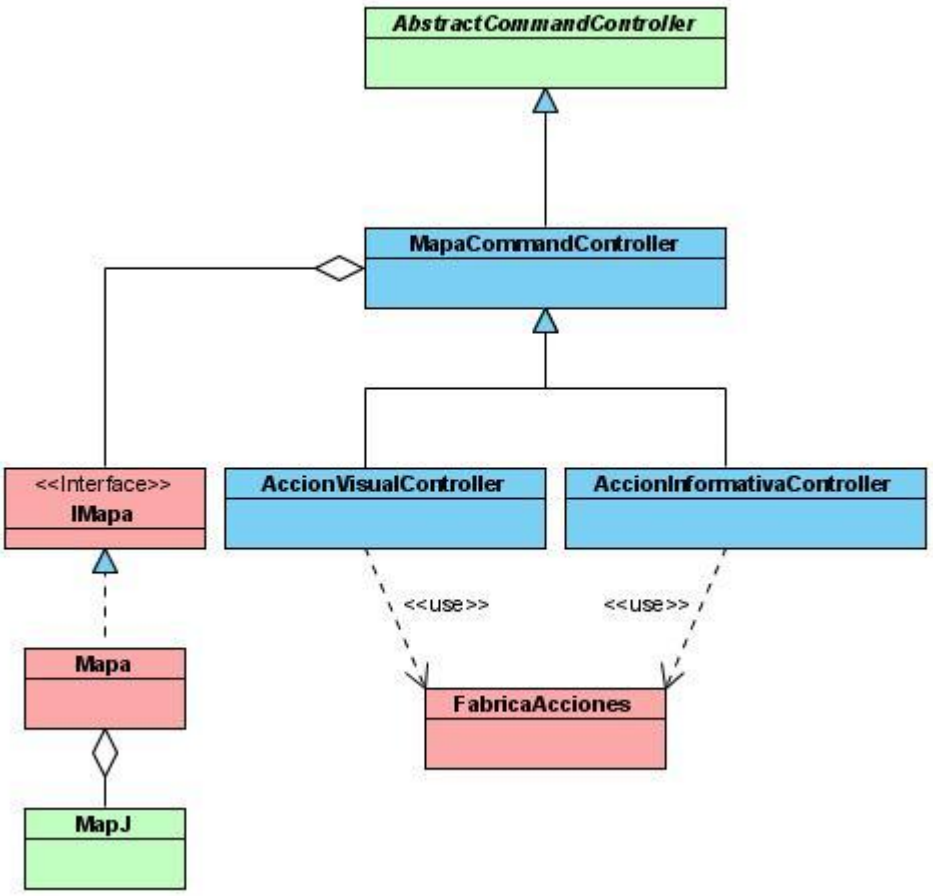

Fig. 2.9. Diagrama de clases del paquete Núcleo.

Clase Mapa

Propósito: Representa el mapa real que el usuario observa en su navegador y almacena todos los datos del mismo, al actuar como contenedor del Objeto de la clase MapJ del framework MapXtreme. Implementa la interfaz IMapa, que es la interfaz de comunicación entre las capas de negocio y presentación, ya que expone los métodos de acceso al mapa real.

Atributos:

- MapJ mapJ: Objeto de la clase MapJ que almacena la información del mapa mostrado al usuario.

Métodos:

- public double getFactorZoom (): Devuelve el factor de acercamiento del mapa que se muestra al usuario.
- public String getDirectorioSalva (): Devuelve la dirección física del directorio donde se guardan los mapas en el servidor.
- public String getTipoImagen (): Devuelve el tipo de imagen con que se muestra el mapa al usuario (gif, jpg, etc).
- public MapJ getMapJ (): Devuelve el objeto MapJ que almacena la información del mapa.
- public String getUrlServidorMapxtreme (): Devuelve la URL del servidor de MapXtreme.
- public String getMapaInicial (): Devuelve el mapa por defecto que se muestra al usuario.
- public List<Capa> getCapas (): devuelve un listado con los nombres y visibilidad de las capas que se muestran en el mapa.
- public List<String> getColumnasCapa (String capa): Devuelve los nombres de las columnas o atributos de una capa dado su nombre.

- public boolean isInicializado (): Devuelve un valor booleano que indica si el mapa fue inicializado.
- public void setInicializado (boolean inicializado): Permite cambiar el valor que indica si el mapa fue inicializado.
- public AbstractLayer getCapa (String capa): Devuelve una capa del mapa dado su nombre.

Clase FabricaAcciones

Propósito: Es la clase encargada de crear dinámicamente las acciones como respuesta a las peticiones del usuario.

Atributos:

- Map<String, String> repositorio: Objeto de la clase HashMap, pares clave-valor, que relaciona cada petición que el usuario puede realizar con la acción que la atiende y la vista final que debe mostrarse.

Métodos:

- public Class<?> getAccion(String key): devuelve una acción dada la petición del usuario.
- public String getVista(String key): devuelve una vista dada la petición del usuario.

2.7.2 Paquete Acciones Visuales.

Este paquete recoge las clases de la aplicación que permiten al usuario realizar acciones sobre el mapa. Todas las operaciones que ocasionan cambios en el mapa, visibles o no, están definidas en clases contenidas en este paquete. Incluye además las etiquetas personalizadas que desencadenan las peticiones.

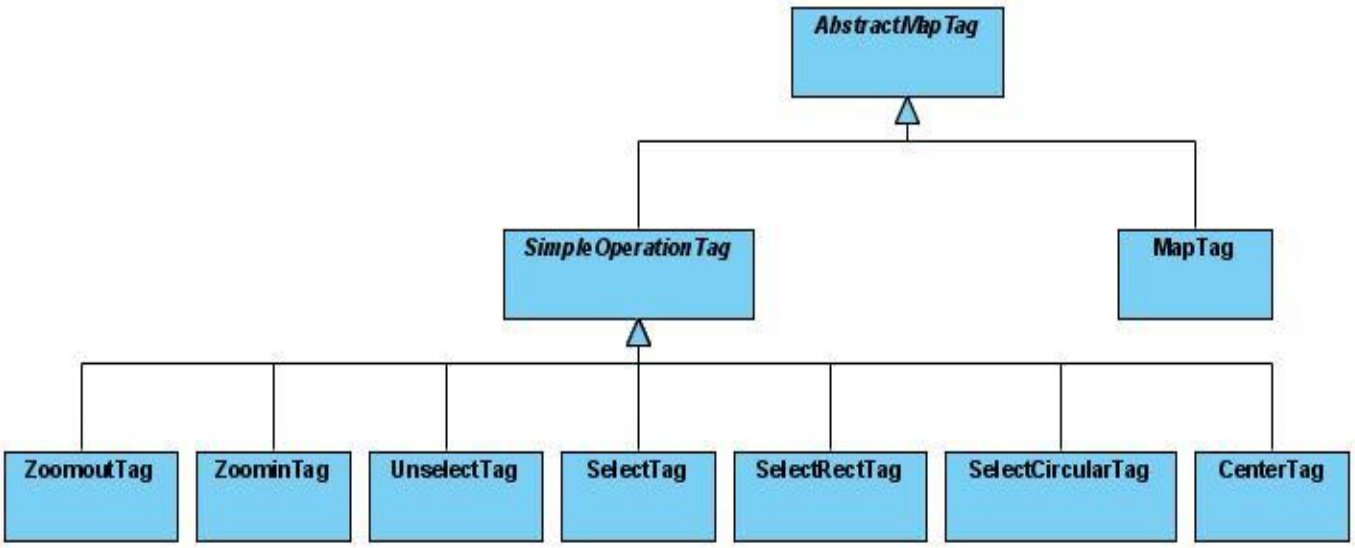

Fig. 2.10. Diagrama de la Capa de Presentación del paquete Acciones Visuales.

Clase MapTag

Propósito: Es la clase encargada de crear físicamente la imagen del mapa que el usuario puede observar en su navegador.

Métodos:

- public int doStartTag (): genera el código HTML y JavaScript necesario para crear la imagen del mapa.

Clase CenterTag

Propósito: Es la clase que define la etiqueta o tag que representa la acción de recentrar el mapa. Define el código HTML y JavaScript que premite al usuario realizar la petición de re centrar el mapa.

Métodos:

- public int doStartTag (): genera el código HTML y JavaScript necesario para realizar la petición de re centrar el mapa.

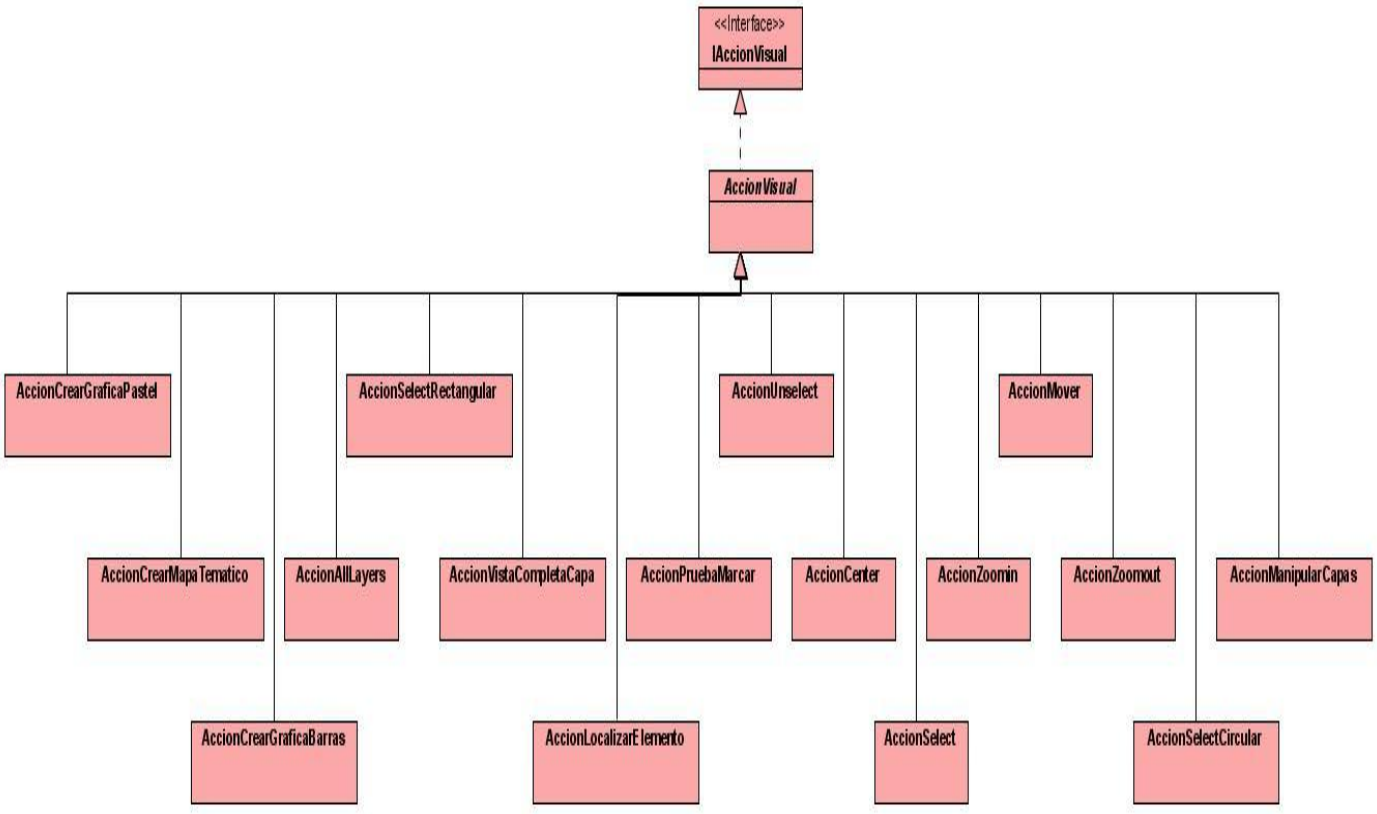

Fig. 2.11. Diagrama de la Capa de Negocio del paquete Acciones Visuales.

Clase AccionInit

Propósito: Es la clase encargada de inicializar el mapa cuando se invoca por primera vez.

Métodos:

- public boolean ejecutar (IMapa mapa): inicializa el mapa utilizando un conjunto de capas ubicadas en el servidor.

Clase AccionCenter

Propósito: Define las operaciones necesarias para centrar el mapa a partir de un punto indicado. Las nuevas coordenadas del centro del mapa son las del punto.

Atributos:

- double y: representa la coordenada y del nuevo centro.
- double x: representa la coordenada x del nuevo centro.

Métodos:

- public boolean ejecutar (IMapa mapa): cambia las coordenadas del centro del mapa a las seleccionadas por el usuario.

Clase AccionCrearMapaTematico

Propósito: Es la clase encargada de crear mapas temáticos.

Atributos:

- int cantRangos: representa la cantidad de rangos en que deben dividirse los valores.
- int distribución: representa el método en que se distribuirán los valores por rango.
- String nombreTema: nombre de la nueva capa que contendrá el mapa temático.
- String capa: nombre de la capa de la cual se obtendrán los valores para crear el mapa temático.
- String columna: nombre del atributo de estudio, del que se desea crear un mapa temático.
- String color1: color que representa el rango con los valores más bajos.
- String color2: color que representa el rango con los valores más altos.

Métodos:

- public boolean ejecutar (IMapa mapa): crea el mapa temático con los datos especificados por el usuario.

2.7.3 Paquete Acciones Informativas.

Este paquete encapsula las clases de la aplicación que permiten obtener información del mapa que se muestra al usuario.

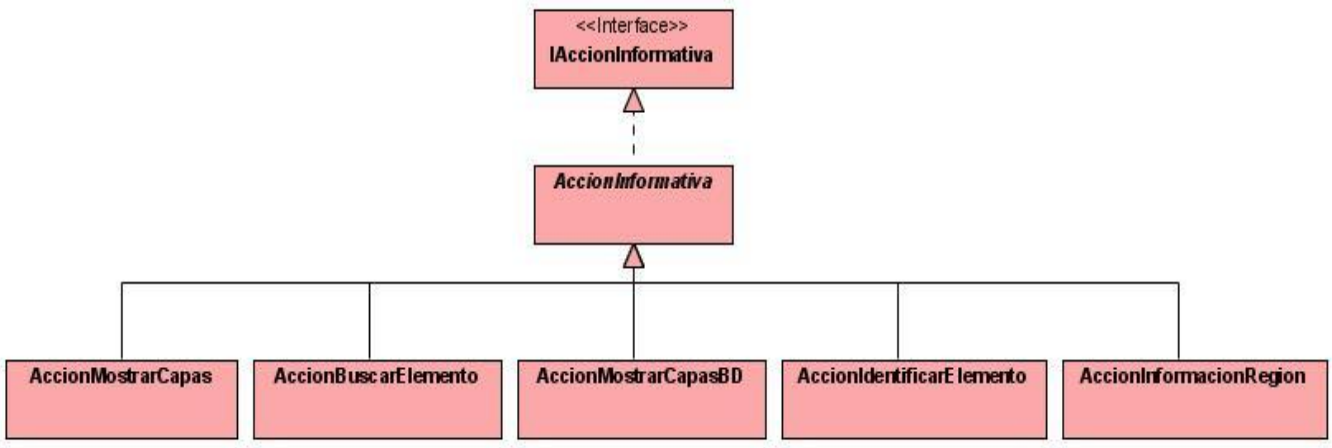

Fig. 2.12. Diagrama de clases del paquete Acciones Informativas.

Clase AccionInformacionRegion

Propósito: Es la clase encargada de obtener toda la información de una región del mapa definida por el usuario.

Atributos:

- double y: coordenada y de la región de la cual se desea obtener información.
- double x: coordenada x de la región de la cual se desea obtener información.

Métodos:

- public Map<String, Object> ejecutar (IMapa mapa): obtiene todos los datos de las capas del mapa en las coordenadas seleccionadas.

Clase AccionMostrarCapas

Propósito: Es la clase encargada de mostrar una lista de las capas abiertas en el mapa.

Métodos:

- public Map<String, Object> ejecutar (IMapa mapa): obtiene los nombres de todas las capas abiertas en el mapa.

2.7.4 Paquete Negocio.

Este paquete agrupa las clases de la aplicación cuya funcionalidad es dar soporte a las llamadas asincrónicas invocadas por el framework DWR, que solicitan información del mapa que se muestra al usuario.

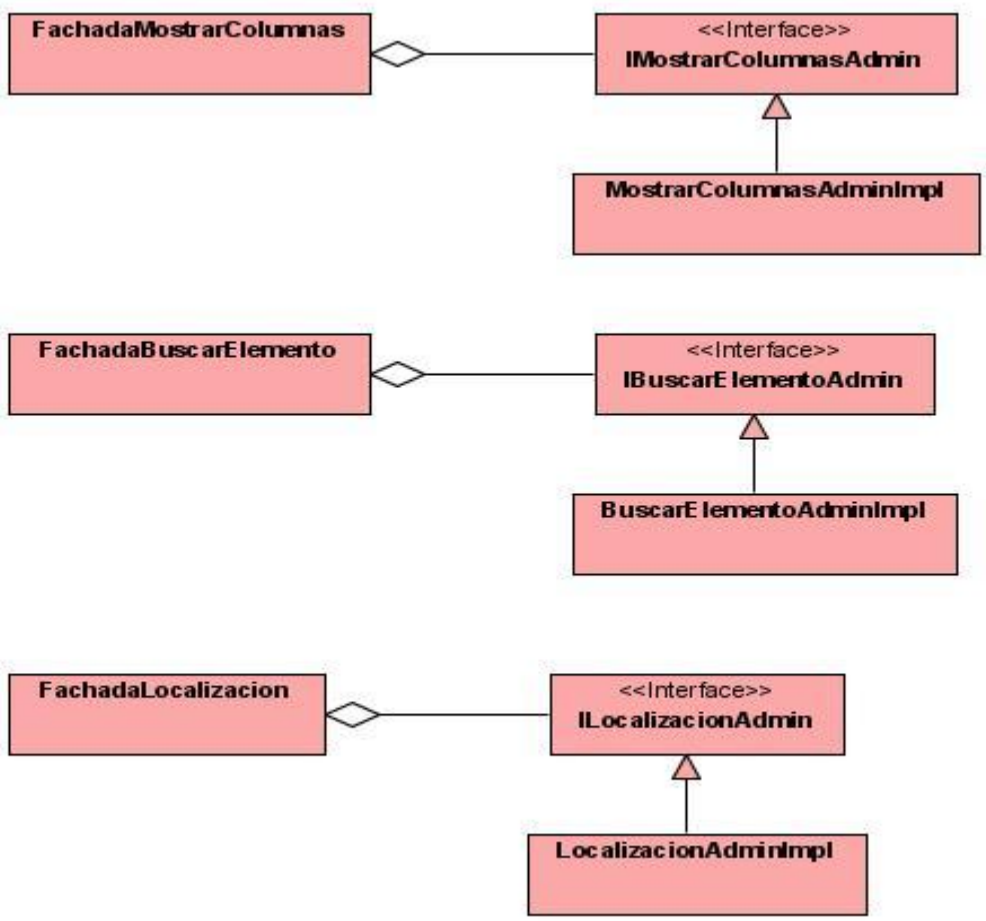

Fig. 2.13. Diagrama de clases del paquete Negocio.

Clase LocalizacionAdminImpl

Propósito: Es la clase encargada de obtener la información que representa la localización del mapa que se muestra al usuario, coordenadas del centro del mapa y zoom.

Atributos:

- IMapa mapa: interfaz que expone los métodos de acceso al mapa digital, el objeto de la clase MapJ del framework MapXtreme.

Métodos:

- public double centroX (): obtiene la coordenada x del centro del mapa actual.
- public double centroY (): obtiene la coordenada y del centro del mapa actual
- public double zoom (): obtiene el valor de zoom del mapa actual.

Clase BuscarElementoAdminImpl

Propósito: Es la clase encargada de buscar en la capa del mapa actual seleccionada por el usuario los elementos que coincidan con un criterio de búsqueda.

Atributos:

- IMapa mapa: interfaz que expone los métodos de acceso al mapa digital, el objeto de la clase MapJ del framework MapXtreme.

Métodos:

- public buscarElemento(String capa, String columna,String elemento): busca en la capa seleccionada los elementos que coincidan con el criterio de búsqueda. El parámetro columna representa la información o atributo de la capa seleccionada que se debe comparar, en caso de no definir ninguna, se comparará con todos los atributos de la capa. El parámetro elemento representa el criterio de búsqueda.

2.7.5 Paquete Acceso a Datos.

Este paquete contiene las clases necesarias para la comunicación de la aplicación con la base de datos. Esta soportado sobre el framework Ibatis.

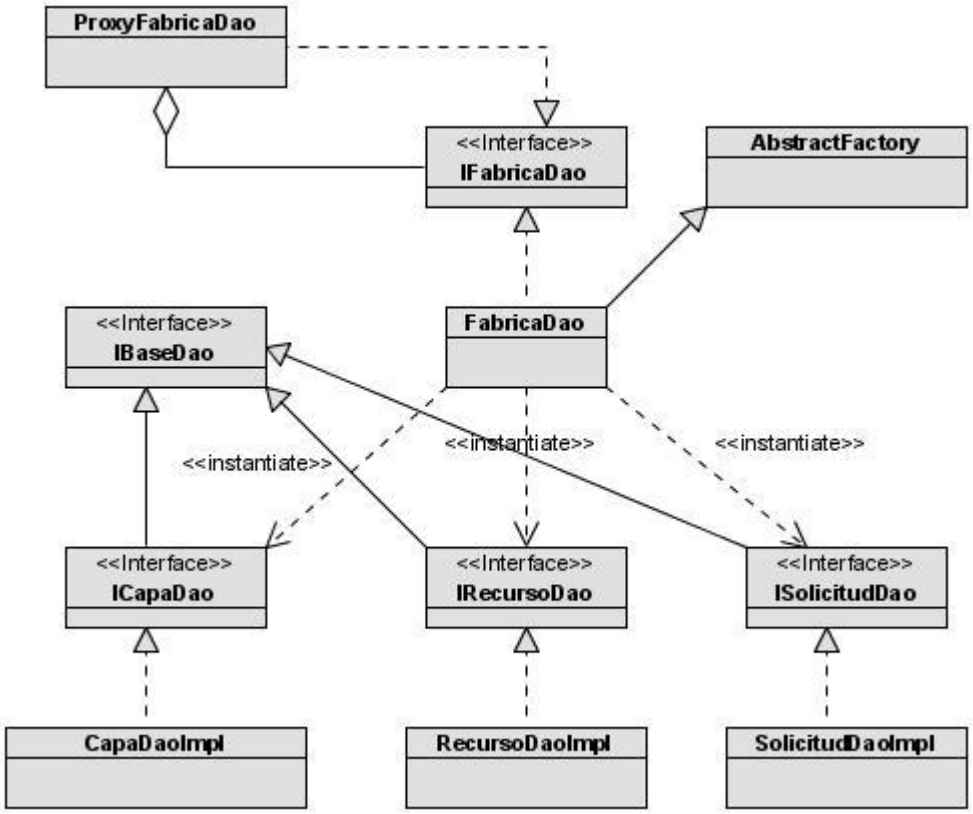

Fig. 2.14 Diagrama de clases del paquete Acceso a Datos.

Clase ProxyFabricaDao

Propósito: Es la clase encargada de la exponer las funcionalidades del acceso a datos a las clases del negocio que las necesiten.

Atributos:

- IFabricaDao fabricaDao: interfaz que expone la fábrica de instanciación de los DAOs.

Métodos:

- public ICapaDao getCapaDao(): Devuelve una instancia de la clase que implementa la interfaz ICapaDao.
- public IRecursoDao getRecursoDao(): Devuelve una instancia de la clase que implementa la interfaz IRecursoDao.

- public ISolicitudDao getSolicitudDao(): Devuelve una instancia de la clase que implementa la interfaz IRecursoDao.

Clase FabricaDao

Propósito: Es la clase encargada de instanciar los DAOs (Objetos de Acceso a Datos).

Métodos:

- public ICapaDao getCapaDao(): Devuelve una instancia de la clase que implementa la interfaz ICapaDao.
- public IRecursoDao getRecursoDao(): Devuelve una instancia de la clase que implementa la interfaz IRecursoDao.
- public ISolicitudDao getSolicitudDao(): Devuelve una instancia de la clase que implementa la interfaz IRecursoDao.

Clase CapaDaoImpl

Propósito: Proporcionarle una implementación concreta a la interfaz ICapaDao que define las posibles acciones sobre una capa a nivel de acceso a datos.

Métodos:

- public List<Capa> seleccionarTodos(): Devuelve un listado de todas las capas existentes en la base de datos.
- public Capa seleccionarPorId(String idEntidad): Devuelve una capa de la base de datos dado su identificador.

2.8 Comparación con la propuesta anterior.

Se hace necesaria una comparación con la anterior arquitectura, para verificar la evolución del diseño y el aumento de los niveles de flexibilidad, adaptabilidad, claridad y reutilización. Para esto se utilizará un diagrama de clases de una de las funcionalidades básicas propuesto en la anterior solución y su equivalente en el nuevo diseño.

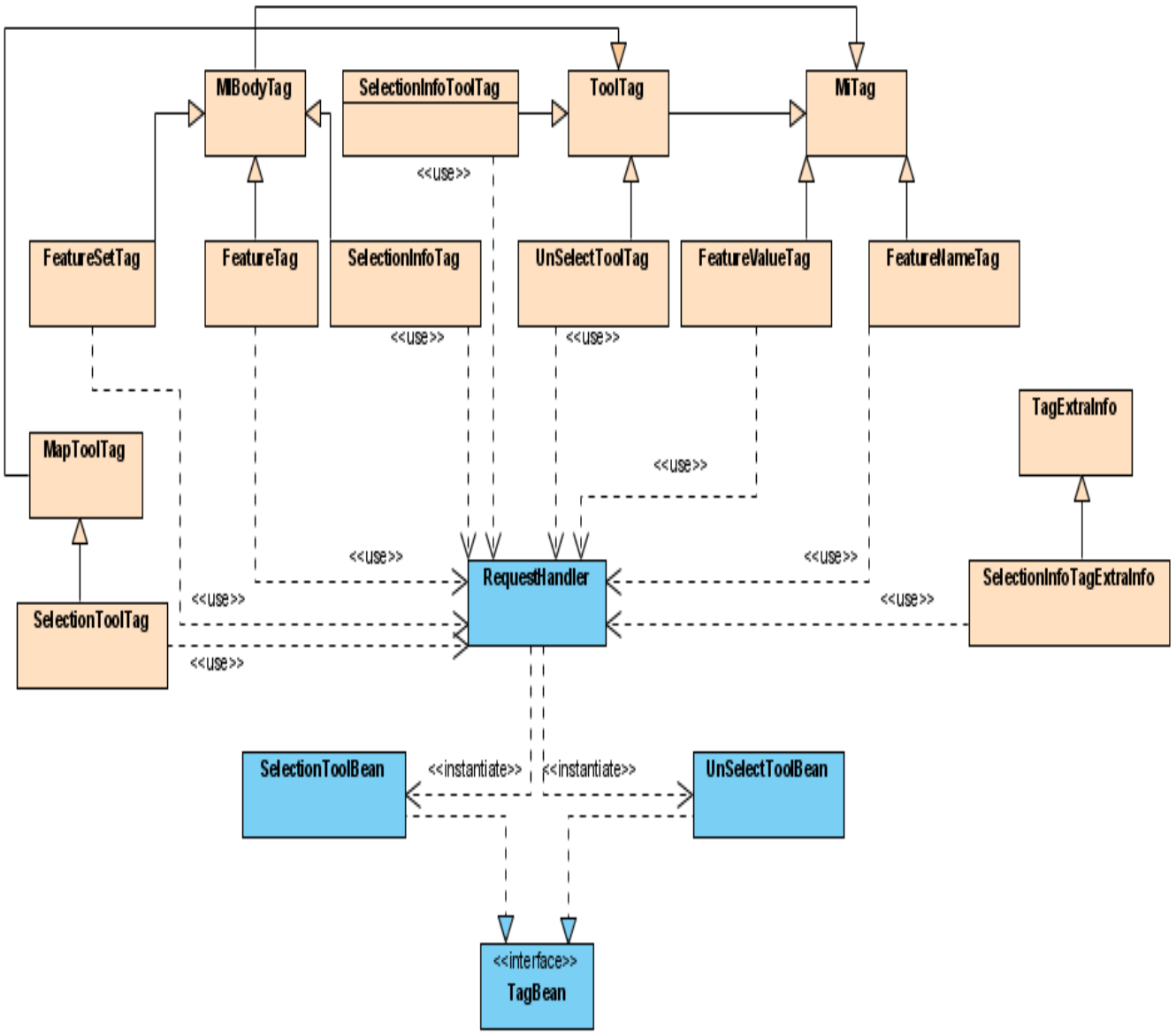

Fig. 2.15. Diagrama de clases de la funcionalidad Seleccionar Región (propuesta anterior).

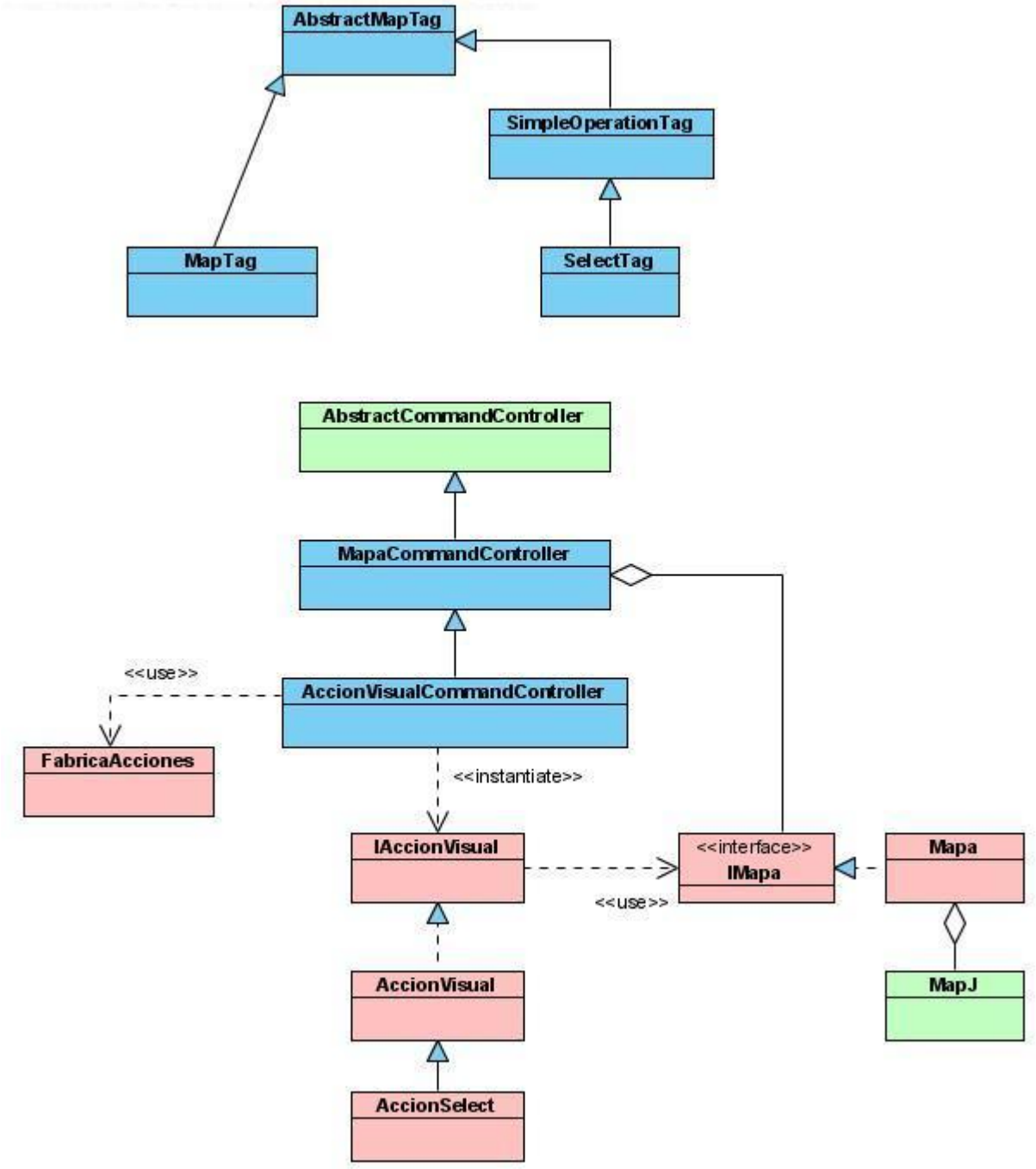

Fig. 2.16. Diagrama de clases de la funcionalidad Seleccionar Región (nueva propuesta).

Como se puede observar al comparar los diagramas anteriores, la nueva propuesta presenta un diseño más claro y legible, y la separación entre las capas de presentación y acceso a datos es evidente. La práctica de programar orientado a interfaces y la utilización del patrón inyección de dependencia cuya implementación brinda el framework Spring, garantiza un

muy bajo acoplamiento entre las clases, potenciando la flexibilidad y reutilización de los componentes.

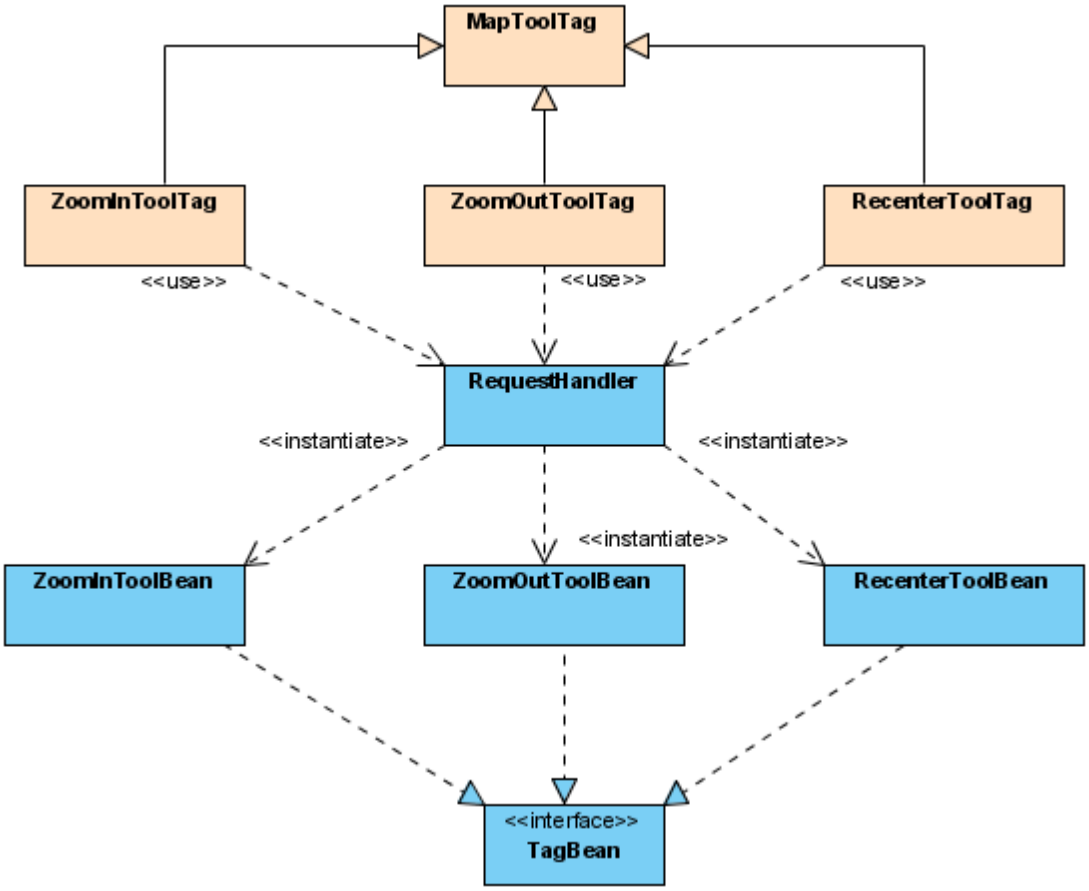

Fig. 2.17. Diagrama de clases de las funcionalidades Acercar, Alejar, Centrar (propuesta anterior).

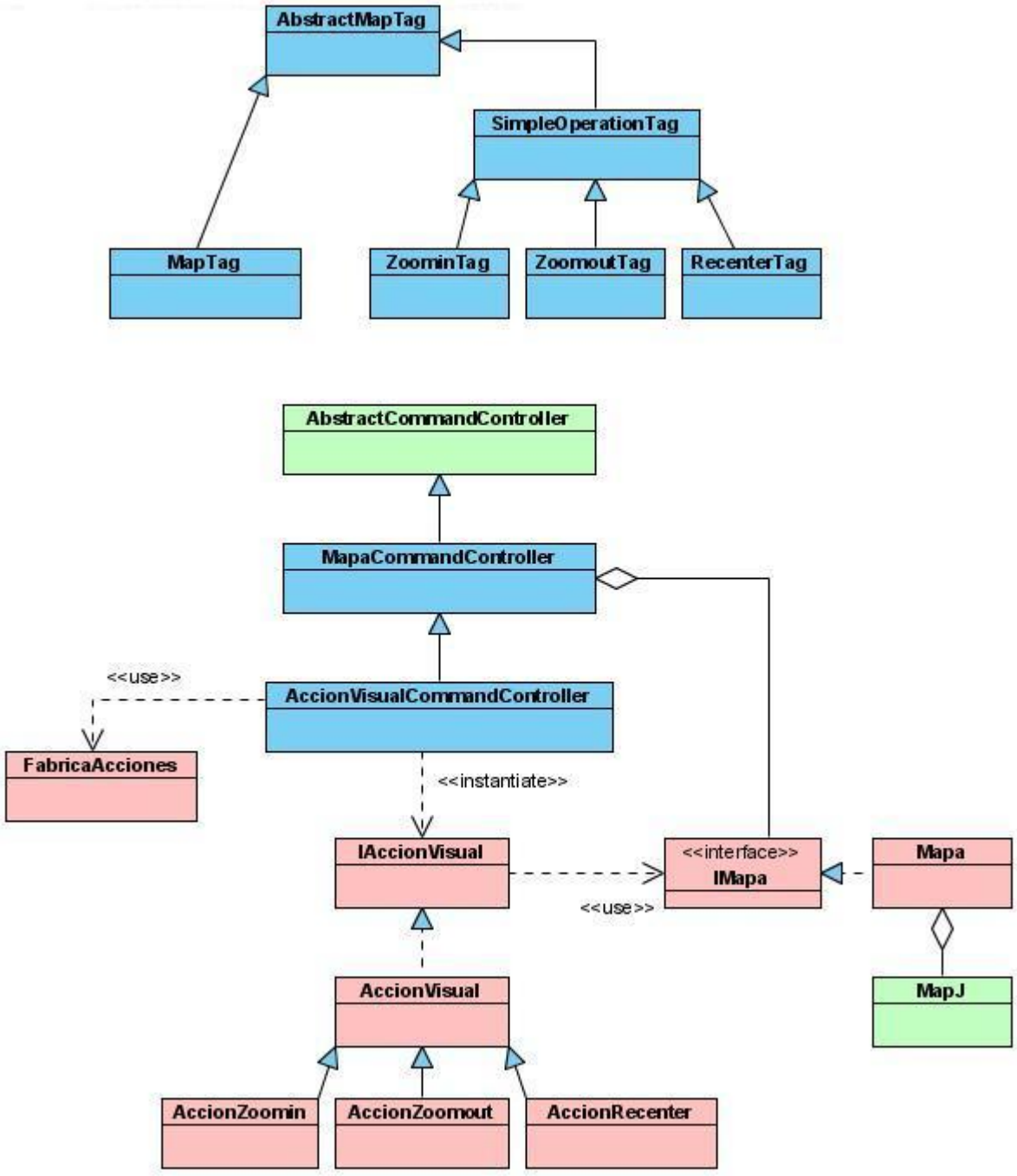

Fig. 2.18. Diagrama de clases de las funcionalidades Acercar, Alejar, Centrar (nueva propuesta).

La comparación de los diagramas de clases que representan las funcionalidades Acercar, Alejar y Centrar, demuestra que el nuevo diseño presenta un nivel de reutilización mucho más alto que el de la propuesta anterior. El número de clases a implementar para cada nueva funcionalidad se reduce drásticamente en el nuevo diseño, trayendo como

consecuencia una disminución de la cantidad de código, un aumento de la claridad y legibilidad del mismo. Además de esto, se gana en flexibilidad y escalabilidad, y se facilita el futuro mantenimiento del mismo.

2.9 Conclusiones.

En este capítulo se describió la arquitectura del Sistema de Mapificación Web, se presentó el diagrama de paquetes para el diseño propuesto, así como los diagramas de clases de cada una de las capas correspondientes a estos y se realizó una descripción de las principales clases del diseño. Se efectuó una comparación con la propuesta de diseño anterior donde se puso de manifiesto una evolución de las características de flexibilidad, claridad y escalabilidad, obtenidas como resultado del uso de los frameworks y patrones de diseño.

Capítulo 3. Implementación.

3.1 Introducción.

En este capítulo se representan en términos de componentes los elementos del modelo del diseño definidos en el capítulo anterior. Se presentan los diagramas con los componentes principales que conforman el Sistema de Mapificación Web y se realiza una breve descripción de cada componente, especificando su propósito y contenido.

3.2 Modelo de Implementación.

Los componentes son la parte física y reemplazable de un sistema. Están compuestos por un conjunto de interfaces y proporcionan la realización de dicho conjunto. Se usan para modelar los elementos físicos que pueden hallarse en un nodo, por lo que empaquetan elementos como clases, colaboraciones e interfaces. Son independientes entre ellos y tienen su propia estructura e implementación. Tienen relaciones de traza con los elementos del modelo que implementan.

3.3 Diagrama de Componentes.

Los diagramas de componentes se utilizan para modelar la vista estática de un sistema. Muestran la organización y las dependencias lógicas entre un conjunto de componentes software, sean éstos: código fuente, librerías, binarios, ejecutables, etc. El uso más importante de estos diagramas es mostrar la estructura de alto nivel del modelo de implementación.

A continuación se muestra el diagrama de componentes correspondiente al Sistema de Mapificación Web.

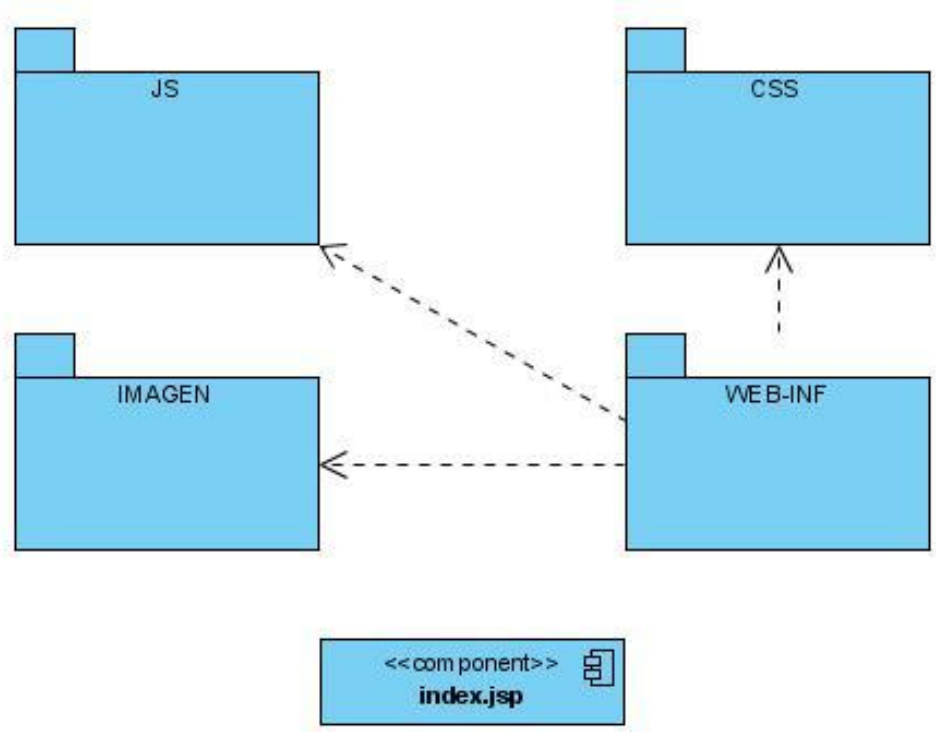

Fig. 3.1. Diagrama de componentes.

3.4 Descripción de los Componentes

a) Subsistema CSS.

Propósito:

Contenedor físico de los ficheros que contienen código CSS u hojas de estilos.

Contenido:

- colorPicker.css
- estilos.css
- form-validation.css
- menu-bar.css
- menu-item.css

- modal-message.css

b) Subsistema IMAGEN.

Propósito:

Contenedor físico de las imágenes de la aplicación.

Contenido:

- allLayers.gif
- center.gif
- centers.gif
- close.jpg
- disk.gif
- fondo_windows.jpg
- gif-loading.gif
- grafica.gif
- graficas.gif
- h.png
- identElem.gif
- identElems.gif
- info.gif
- infos.gif
- layers.gif
- mapaTematico.gif
- mapaTematicos.gif

- menu_strip_bg.jpg
- menu_strip_bg_gray.gif
- menu_strip_down_arrow.gif
- menu_strip_down_arrow.png
- menu_strip_separator.gif
- menu_strip_separator-gray.gif
- menu-bar-gradient.jpg
- menu-bar-gradient-gray.gif
- menu-bar-right-arrow.gif
- menu-bar-right-arrow.png
- mover.gif
- movers.gif
- open.gif
- position.png
- print.gif
- select.gif
- selectCircular.gif
- selectCirculars.gif
- selectRect.gif
- selectRects.gif
- selects.gif
- unselect.gif

- vistaCompletaCapa.gif
- window_cerrar.JPG
- window_sup.JPG
- zoomin.gif
- zoomins.gif
- zoomout.gif
- zoomouts.gif

c) Subsistema JS.

Propósito:

Contenedor físico de los ficheros que contienen código JavaScript.

Contenido:

- asincronox.js
- buscarElementos.js
- colorPicker.js
- crearGraficas.js
- funciones.js
- livevalidation.js
- manipularcapas.js
- menu-for-applications.js
- mouseEvents.js
- movermapa.js
- parametros.js

- prototype-1.6.0.2.js
- rubberband.js
- toolbar.js
- validacion.js
- ajax.js
- ajax-dynamic-content.js
- modal-message.js
- move-modal.js
- windows.js

d) Subsistema WEB-INF.

Propósito:

Contenedor físico de los ficheros que conforman el núcleo de la aplicación.

Contenido:

- Subsistema Classes
- Subsistema JSP
- Subsistema LIB
- dwr.xml
- web.xml

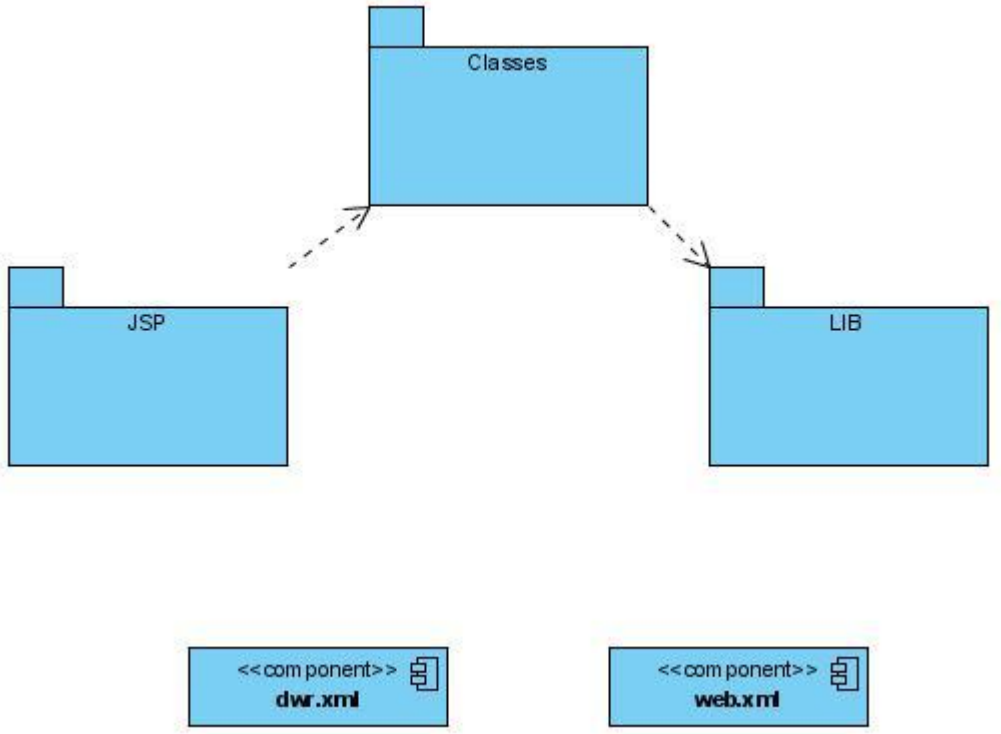

Fig. 3.2. Diagrama de componentes del subsistema WEB-INF.

d.1) Subsistema Classes.

Propósito:

Contenedor físico del compilado de cada una de las clases de la aplicación y los ficheros de configuración, archivos con extensión CLASS y XML.

Contenido:

- IBaseDao.class
- ICapaDao.class
- IEventoDao.class
- CapaDaoImpl.class
- RecursoDaoImpl.class
- SeguridadDaoImpl.class

- SolicitudDaoImpl.class
- IRecursoDao.class
- ISeguridadDao.class
- ISolicitudDao.class
- IFabricaDao.class
- FabricaDao.class
- ProxyFabricaDao.class
- Capa-SqlMap.xml
- Recurso-SqlMap.xml
- Seguridad-SqlMap.xml
- Solicitud-SqlMap.xml
- sqlMapConfig.xml
- BuscarSolicitudParametro.class
- MultipleConnectionsPoolsDataSource.class
- IUsersDetailsPool.class
- UserPool.class
- UsersDetailsPoolImpl$UserRolMapping.class
- UsersDetailsPoolImpl.class
- ListaAtributosTypeHandler.class
- TraductorErrorSql.class
- accesodatos-config.xml
- app-config.xml

- controller-config.xml
- dominio-config.xml
- fabricas-config.xml
- mapa-config.xml
- negocio-config.xml
- Atributo.class
- Capa.class
- CategoriaRecurso.class
- Elemento.class
- Evento.class
- Filtro.class
- LocalizacionInfo.class
- Motivo.class
- Organismo.class
- Propiedad.class
- Recurso.class
- Solicitud.class
- TablaInformacion.class
- Tema.class
- TipoRecurso.class
- TipoSolicitud.class
- Usuario.class

- AccionInformativa.class
- AccionVisual.class
- FabricaAcciones.class
- IAccionInformativa.class
- IAccionVisual.class
- AccionBuscarElemento.class
- AccionIdentificarElemento.class
- AccionInformacionRegion.class
- AccionMostrarCapas.class
- AccionMostrarCapasBD.class
- AccionInit.class
- AccionAllLayers.class
- AccionCenter.class
- AccionLocalizarElemento.class
- AccionManipularCapas.class
- AccionMover.class
- AccionPruebaMarcar.class
- AccionSelect.class
- AccionSelectCircular.class
- AccionSelectRectangular.class
- AccionUnselect.class
- AccionVistaCompletaCapa.class

- AccionZoomin.class
- AccionZoomout.class
- AccionCrearGrafica.class
- AccionCrearGraficaBarras.class
- AccionCrearGraficaPastel.class
- AccionCrearMapaTematico.class
- IAutenticarAdmin.class
- IBuscarElementoAdmin.class
- IBuscarRecursosAdmin.class
- ILocalizacionAdmin.class
- IMostrarColumnasAdmin.class
- AutenticarAdminImpl.class
- BuscarElementoAdminImpl.class
- BuscarRecursosAdminImpl.class
- LocalizacionAdminImpl.class
- MostrarColumnasAdminImpl.class
- FachadaBuscarElemento.class
- FachadaBuscarRecursos.class
- FachadaLocalizacion.class
- FachadaMostrarColumnas.class
- IMapa.class
- MapaImpl.class

- MostrarMapaCommandController.class
- AccionInformativaController.class
- AccionVisualController.class
- AbstractMapTag.class
- CenterTag.class
- MapTag.class
- SelectCircularTag.class
- SelectRectTag.class
- SelectTag.class
- SimpleOperationTag.class
- UnselectTag.class
- ZoominTag.class
- ZoomoutTag.class
- ZoominValidator.class

d.2) Subsistema JSP.

Propósito:

Contenedor físico de las interfaces de usuario, archivos con extensión JSP.

Contenido:

- error.jsp
- abrirCapas.jsp
- administrarCapas.jsp
- buscarElemento.jsp

- buscarProcedimientosGenerales.jsp
- buscarRecursos.jsp
- buscarSolicitud.jsp
- crearGraficas.jsp
- crearMapaTematico.jsp
- detallesEvento.jsp
- detallesRecursos.jsp
- detallesSolicitud.jsp
- eventoInformadoOrganismo.jsp
- eventoInformadoPersona.jsp
- indicarRecursos.jsp
- indicarSolicitudRecursos.jsp
- velocidadRecurso.jsp
- vistaCompletaCapa.jsp
- include.jsp
- index.jsp
- identificarElemento.jsp
- informacionRegion.jsp

d.3) Subsistema LIB.

Propósito:

Contenedor físico de las librerías utilizadas, archivos con extensión JAR.

Contenido:

- acegi-security-1.0.4.jar
- app-spring-1.1.jar
- cglib-nodep-2.1_3.jar
- classes12.jar
- clibwrapper_jiio.jar
- commons-beanutils.jar
- commons-codec.jar
- commons-collections.jar
- commons-dbcp.jar
- commons-digester.jar
- commons-io.jar
- commons-lang.jar
- commons-logging.jar
- commons-pool.jar
- concurrent.jar
- dwr-2.0.1.jar
- ibatis-2.3.0.677.jar
- jai_codec.jar
- jai_core.jar
- jai_imageio.jar
- jdom.jar

- jserverc.jar
- jstl.jar
- jta.jar
- log4j-1.2.14.jar
- miappletsup.jar
- micsys.jar
- midtds.jar
- mifonts.jar
- mijsptaglibs.jar
- mijts.jar
- mistyles.jar
- miutil.jar
- mlibwrapper_jai.jar
- mmjclient.jar
- mmjcommon.jar
- mxj.jar
- mxjclient.jar
- mxjdevsup.jar
- mxjdtds.jar
- mxjgeom.jar
- mxjiusdp.jar
- mxjloc.jar

- mxjmanager.jar
- mxjoradp.jar
- mxjRásterdp.jar
- mxjtabdp.jar
- mxjwmsdp.jar
- rjsclient.jar
- rjscommon.jar
- spring-beans.jar
- spring-context.jar
- spring-core.jar
- spring-dao.jar
- spring-ibatis.jar
- spring-jdbc.jar
- spring-web.jar
- spring-webmvc.jar
- standard.jar
- xalan.jar
- xercesImpl.jar
- xml-apis.jar
- xmlParserAPIs.jar

d.4) Componente dwr.xml.

Propósito:

Fichero XML de configuración del framework DWR para las llamadas asincrónicas en la aplicación.

d.5) Componente web.xml.

Propósito:

Descriptor de despliegue para la aplicación Web.

e) Componente index.jsp.

Propósito:

Constituye el punto de entrada a la aplicación.

3.5 Conclusiones.

Los diagramas confeccionados en este capítulo, representan en términos de componentes y subsistemas de implementación, los elementos del modelo de diseño obtenidos en el capítulo anterior y establecen las dependencias entre unos y otros.

Conclusiones.

El presente trabajo finaliza dando cumplimiento al objetivo general trazado de desarrollar un Sistema de Mapificación Web con la capacidad de procesar información referenciada geográficamente.

Se efectuó un estudio de las herramientas y tecnologías de desarrollo que se utilizaron en la confección de la solución. Se definió un diseño que modela el sistema y brinda soporte a todos los requisitos funcionales y no funcionales. Un aspecto de gran utilidad en la confección y concepción del diseño fue la comprensión y utilización de patrones, que permitieron aplicar buenas prácticas y brindar soluciones probadas a problemas comunes.

Se estructuró un diagrama de componentes que representa la implementación del sistema en términos de componentes, quedando construida una aplicación capaz de garantizar rapidez y efectividad en los procesos de tratamiento y análisis de la información geográficamente referenciada que se llevan a cabo en las dependencias policiales de la República Bolivariana de Venezuela.

Recomendaciones.

Continuar la implementación del Sistema de Mapificación Web.

Ampliar las capacidades del Sistema de Mapificación Web, incorporando funciones avanzadas de búsqueda y técnicas de reconocimiento de patrones.

Utilizar las funcionalidades básicas y la arquitectura del Sistema de Mapificación Web como base para el desarrollo de otros sistemas en los que se necesite manipular o procesar información geográfica.

Referencias Bibliográficas.

1. González, Patricia. International Development Research Centre. [Online] 2003. [Cited: octubre 27, 2008.] http://www.idrc.ca/uploads/user-S/105250556802segundo_folleto.pdf.
2. MIJ. Reportajes.htm. 2006. nº.
3. Gabaldón, Luis Gerardo. Cfr. p. 116. 1.
4. Escamillo, J. C. Ministerio del Poder Popular para Relaciones de Interiores y Justicia [Online] [Cited: octubre 27, 2008.] http://www.mpprij.gob.ve/spip.php?article=2071
5. Aronoff, Stanley. 1989. Geographic information systems : a management perspective. Ottawa : s.n., 1989.
6. Valenzuela, C. 1989. Introducción a los Sistemas de Información Geográfica. Santa Fé de Bogotá. Colombia. : Subdirección de Docencia e Investigación. (CIAF), 1989.
7. Jacobson, I.; Booch, G. y Rumbaugh, J.; "El Proceso Unificado de Desarrollo de software". 2000.
8. Larman, Craig. Introducción al análisis y diseño orientado a objetos. UML y Patrones. Tomo I.
9. Patrones de Diseño en aplicaciones Web con Java J2EE. Ciberaula. [Online] [Cited: octubre 29, 2008.]
10. C. Walls, R. Breidenbach. Spring in Action. s.l. : Manning Publications, 2005.
11. Clinton Begin, Brandon Goodin, Larry Meadors. iBatis in Action. 2007. Manning Publications.
12. Direct Web Remoting. [Online] [Cited: octubre 30, 2008.] http://directwebremoting.org/.
13. García Puebla, Ivón. Gestor de Contenidos. Eclipse: una herramienta profesional al alcance de todos. [Online] Agosto 2, 2005. [Cited: Octubre 31, 2008.]

Bibliografía.

1. Alarcón Ferrá, Ailín and Arias Arias, Michel. 2007. *SISTEMA DE GESTIÓN PARA CENTROS DE EMERGENCIAS DE SEGURIDAD CIUDADANA 171. Módulo de Mapificación Web.* Universidad de las Ciencias Informáticas. 2007. Tesis.
2. Departamento de Ingeniería Topográfica y Cartografía UPM. [Online] ESCUELA TÉCNICA SUPERIOR DE INGENIEROS EN TOPOGRAFÍA, GEODESIA Y CARTOGRAFÍA. [Cited: octubre 27, 2008.] http://www.geo.upm.es/ven/resources/glossary.html.
3. Pitney Bowes MapInfo. [Online] [Cited: octubre 27, 2008.] http://www.mapinfo.com/location/integration/.
4. ESRI. [Online] [Cited: enero 12, 2008.] http://www.esri.com/.
5. MapServer. [Online] [Cited: enero 13, 2008.] http://mapserver.org/.
6. Sun MycroSystems. [Online] [Cited: octubre 27, 2008.] http://www.sun.com/.
7. Open Source Initiative. [Online] [Cited: enero 12, 2008.] http://www.opensource.org/licenses/mit-license.php/.
8. Spring Source Community. [Online] [Cited: enero 12, 2008.] http://www.springsource.org/.
9. Ibatis. [Online] [Cited: enero 12, 2008.] http://www.ibatis.apache.org/.
10. Prototype. [Online] [Cited: enero 12, 2008.] http://www.prototypejs.org/.

Glosario de Términos.

- **Ajax (*Asynchronous JavaScript And XML*):** Técnica de desarrollo web para crear aplicaciones interactivas. Permite el intercambio asíncrono entre cliente y servidor.
- **CSRF (Cross Site Request Forgery):** Tipo de ataque malicioso a un sitio web basado en explotar la confianza que estos tienen con sus usuarios.
- **Enterprise JavaBeans:** Una de las API que forman parte del estándar de construcción de aplicaciones empresariales J2EE. Su objetivo es dotar al programador de un modelo que le permita abstraerse de los problemas generales de una aplicación empresarial (transacciones, persistencia, seguridad, etc.) y centrarse en el desarrollo de la lógica de negocio en sí.
- **IDE (Entorno Integrado de Desarrollo):** Programa compuesto por un conjunto de herramientas para el programador.
- **Java Beans:** Modelo de componentes para la construcción de aplicaciones en Java. Consiste en encapsular varios objetos simples en uno más complejo. Para funcionar como JavaBean, una clase debe obedecer ciertas convenciones sobre nomenclatura de métodos, construcción, y comportamiento.
- **Mapa temático:** Mapa que representa fenómenos abstractos o temas específicos de tipo cuantitativo o cualitativo de alguna variable.
- **MapJ:** Clase del framework MapXtreme encargada de contener la información del mapa digital que se muestra al usuario.
- **OSGeo (Open Source Geospatial Foundation):** Organización no gubernamental sin ánimo de lucro cuya tarea es promover el desarrollo colaborativo de tecnologías y datos geoespaciales.
- **Plug-in:** Aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy especifica.
- **Request:** Orden que se envía a un servidor por medio de una aplicación y porta información acerca de la misma.

- **Servlet:** Objeto que se ejecuta en un servidor o contenedor JEE, especialmente diseñado para ofrecer contenido dinámico desde un servidor web.