UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 6

Desarrollo de una API para brindar servicios de mapas en la Línea de Productos de Software Aplicativos SIG.

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas.

AUTOR

Rayner Barroso Armas.

TUTORES

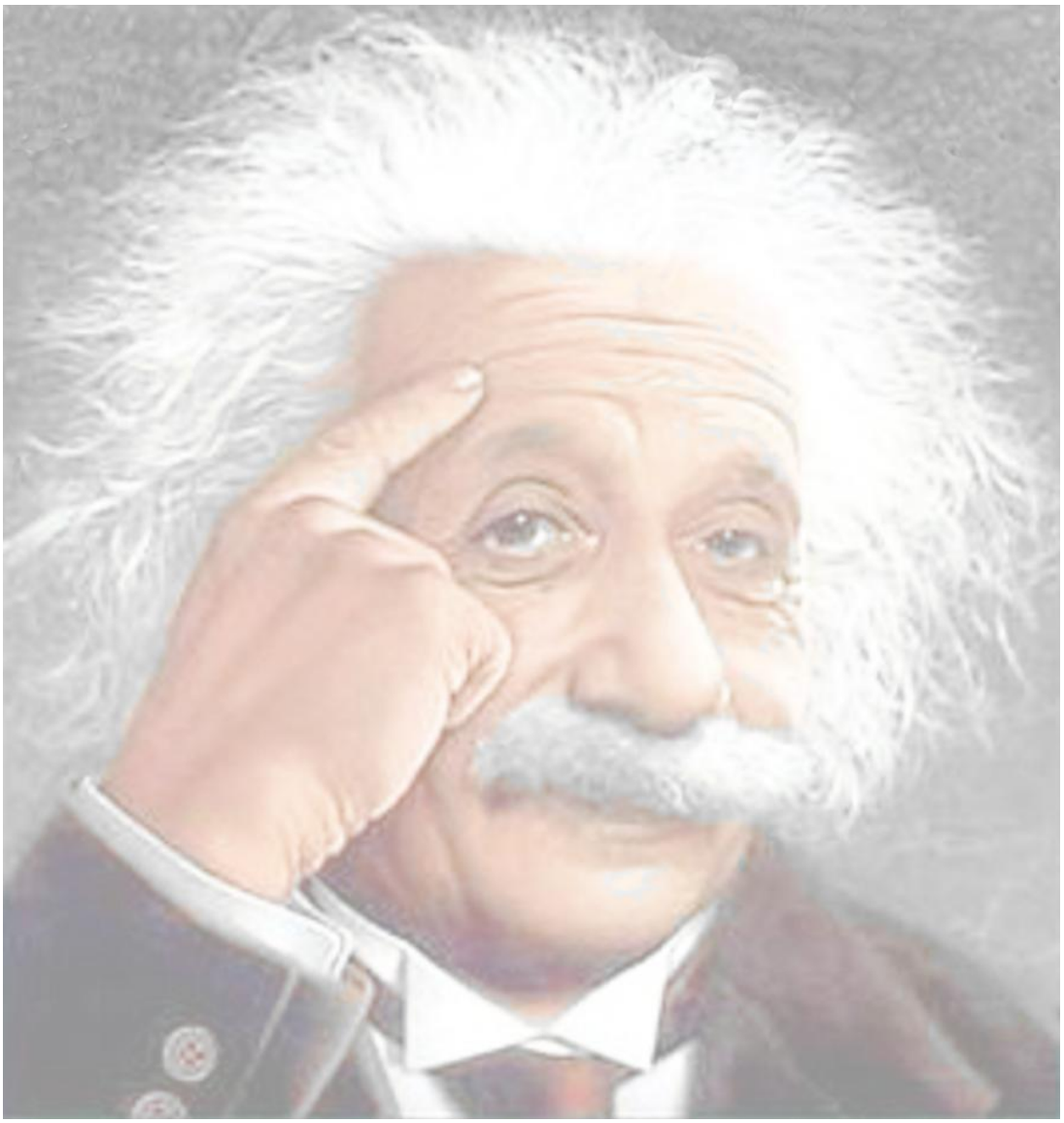
Ing. Adrián Gracia Águila.

Ing. Alain León Companioni.

Ciudad de La Habana 2013

"Año 55 de la Revolución"

PENSAMIENTO



"Si buscas resultados distintos, no hagas siempre lo mismo."

A. Einstein

DECLARACIÓN DE AUTORÍA

Declaro ser el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas hacer uso del mismo en su beneficio.

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

Autor:

_____

Rayner Barroso Armas

Tutores:

_____ _____

Ing. Adrian Gracia Águila. Ing. Alain León Companioni.

DATOS DE CONTACTOS

Nombre y Apellidos del Tutor: Ing Adrian Gracia Águila.

Institución: Universidad de las Ciencias Informáticas

E-mail: agracia@uci.cu .

Ingeniero en Ciencias Informáticas, graduado en Universidad de las Ciencias Informáticas en el año 2009. Actualmente es el líder de la LPS "Aplicativos SIG" Dpto. de Geoinformática. GEySED.

Nombre y Apellidos del Tutor: Ing. Alain León Companioni.

Institución: Universidad de las Ciencias Informáticas

E-mail: acompanioni@uci.cu

Ingeniero en Ciencias Informáticas, graduado la UCI en el año 2010.

OPINIÓN DEL USUARIO DEL TRABAJO DE DIPLOMA

AGRADECIMIENTOS

No es tarea fácil poner en papel a todos los que de una manera u otra estuvieron a mi lado durante este tiempo y mucho menos expresarles cuanto agradezco que así lo hicieran.

Me siento inmensamente dichoso y agradezco de la forma más grandiosa posible a mis padres que siempre creyeron en mí.

A mi hermano, que tuvo que hacer un montón de trabajo el solo y yo pude ayudarle bien poco por estar enfrascado en el proceso de esta tesis.

A mi cuñada Lianet que siempre estuvo pendiente de todo lo que ocurría y a mi sobrinita Zanney que es la más linda del mundo.

A mis tutores Adrian y Alain, que sin ellos nada hubiera salido, porque lucharon conmigo en todo momento, gracias hermanitos.

Un agradecimiento especial a Yenysell, que bastante la molesté pidiéndole hojas durante el proceso y sobre todo porque intercedió por mí con cuanta persona se enteraba que era tribunal u oponente.

Quisiera expresar mi más sincero agradecimiento a todos los profesores que aportaron su grano de arena en la preparación de quien soy hoy en día.

A todos mis amigos que nunca me dieron la espalda y siempre tuvieron una sonrisa en sus rostros a la hora de contar con ellos y hablo de Iván y Made, que bastante los mortifiqué en este tiempo, Angelito, Nilmar, Enrique, Javier, Yusma y Mayde, a Roque que me ayudo muchísimo con el tema de la investigación.

En general a todos mis amigos, discúlpenme que no los mencioné uno a uno, si lo hiciera sería infinita la lista. A todos siempre los tendré en mi corazón.

DEDICATORIA

Este trabajo lo quiero dedicar a mi familia, en especial a mis padres por ser siempre mi apoyo y mi faro guía, a mi hermano y mi cuñada que siempre estuvieron pendientes de todo, a mi sobrinita por ser la más linda del mundo, a mis abuelos que siempre me alentaron a que siguiera y no me cansara, a tirar hacia adelante, y a mis amigos que les toco la peor parte que fue soportarme todo este tiempo.

A todos muchas gracias y ojalá se sientan orgullosos de mí.

RESUMEN

En la actualidad, existe una tendencia a que la información geográfica sea manejada digitalmente para de esta forma hacer más fácil la interacción con la información y agilizar el proceso de la toma de decisiones. En la Universidad de las Ciencias Informáticas (UCI), específicamente en la Línea de Productos de Software (LPS) Aplicativos SIG, se desarrolla un fuerte trabajo en el desarrollo de Sistemas de Información Geográfica (SIG).

Sin embargo dicha LPS presenta deficiencias para gestionar la seguridad sobre los servicios de mapas que brinda, además de que no todos los servicios brindados cumplen con los estándares del Open Geospatial Consortium (OGC).

En este trabajo y para darle solución a los problemas existentes dentro de la LPS, se crea un sistema, específicamente una Interfaz de programación de aplicaciones (API por sus siglas en Ingles) para brindar servicios de mapas.

Se utiliza la metodología RUP, como Framework de desarrollo Symfony 2, como servidor de mapas se utiliza MapServer, como servidor de base de datos Apache 2 y como Sistema Gestor de Base de Datos se utiliza PostgreSQL con extensión PostGis.

Palabras claves:

Servicios de mapas, Servicios Web OGC, SIG, API

Índice

Índice de figuras

Índice de tablas

INTRODUCCIÓN

En la actualidad, la informática ocupa un rol importante, tanto en la vida diaria de las personas, como en el desarrollo de las empresas. El avance de las Tecnologías de la Información y la Comunicación (TIC) ha brindado grandes progresos en diferentes sectores de la actividad humana como: la medicina, la biología, la ingeniería, la industria, la investigación científica, la gestión empresarial, entre otros.

Gracias al surgimiento de la Internet y el desarrollo actual de las tecnologías computacionales y de comunicación, la sociedad se ha desarrollado vertiginosamente, aumentando considerablemente tanto el flujo como el volumen de la información circulante. Por ello se hace cada vez más necesario representar de manera eficiente los datos; siendo fundamental el desarrollo de sistemas y servicios que gestionen adecuadamente la información.

Entre las numerosas herramientas y sistemas que se han venido desarrollando en los últimos tiempos, se encuentran los Sistemas de Información Geográfica (SIG). Estos sistemas, desde su surgimiento y hasta la fecha, han evolucionado en correspondencia con el propio desarrollo de las TIC. Los SIG, tienen su origen entre las décadas del sesenta y el ochenta, conociéndose el Sistema de Información Geográfica de Canadá desarrollado por Roger Tomlinson en 1962, como la primera utilización real de los SIG.

En los primeros SIG desarrollados, toda la información se almacenaba en una única computadora, ejecutándose el SIG también en ella. Luego, estos sistemas se fueron perfeccionando y ganando en cuanto a conectividad dentro de la empresa, en intranets, hasta que finalmente, surgen los enfoques orientados a la Web para intentar satisfacer las demandas de toda la sociedad.

Un SIG es un sistema utilizado para almacenar, manipular y representar información referenciada geográficamente, donde la representación de dicha información está estrechamente relacionada con su posición geográfica. Estas tecnologías se han vuelto tan populares en los últimos años, que actualmente son consideradas como herramientas esenciales para el uso efectivo de la información geográfica, así como para resolver problemas de gestión y planificación.

Cuba, actualmente se encuentra en pos de la digitalización y automatización de cuanta información y proceso sea posible, aplicándose en diferentes esferas de la sociedad como son los servicios, la educación, la salud, la gestión económica y las investigaciones, entre otros. La implementación de sistemas que capturan, procesan y representan la información de una forma más práctica y entendible es uno de los mayores aportes de las TIC, con lo que contribuyen al ahorro de recursos y tiempo del país, influyendo así positivamente en el desarrollo socio-económico del mismo.

La Universidad de las Ciencias Informáticas (UCI) ha sido desde sus inicios la mayor protagonista de los logros informáticos del país. Debido a que en Cuba existen pocas empresas que se dediquen al desarrollo de software, la UCI ocupa un lugar importante en la producción de software a nivel nacional.

En la UCI existen diversos Centros de Desarrollo de Software, los cuales están integrados por diferentes Proyectos Productivos. Tal es el caso del Centro de Desarrollo Geoinformática y Señales Digitales (GEYSED) el cual se dedica al desarrollo de productos, servicios y soluciones informáticas en el procesamiento de Señales Digitales y la Geoinformática. Particularmente en el Departamento de Geoinformática están los proyectos dedicados a la realización de aplicaciones SIG, destacando la Línea de Productos de Software (LPS) Aplicativos SIG, la cual hace uso de GeneSIG como plataforma de desarrollo y como servidor de mapas MapServer.

GeneSIG es una plataforma soberana desarrollada por un grupo de especialistas de la UCI en conjunto con especialistas de GEOCUBA y las FAR la cual posee numerosos servicios orientados a la web. Aunque dicha plataforma utiliza PHP/Mapscript, que es una librería de MapServer que permite la construcción de mapas, solo utiliza la librería antes mencionada y no implementa ningún servicio de los que posee MapServer, dando como resultado, que en sus soluciones, los servicios brindados no cumplan con los estándares establecidos por el *Open Geospatial Consortium* (OGC).

Por su parte, MapServer, a pesar de tener implementados numerosos servicios y cumplir con muchos de los estándares del OGC, no se ocupa de gestionar la seguridad en los mismos y tampoco presta el servicio *Web Feature Service Transactional* (WFS-T). Este servicio, además de que permite interactuar, realizar consultas y recuperación de elementos geográficos en los mapas servidos por el estándar *Web Map Service* (WMS) del OGC, permite además, la edición de los mismos, o sea, la creación, eliminación y actualización de los elementos geográficos del mapa.

De lo anteriormente expuesto surge como ***problema a resolver*** ¿Cómo centralizar el acceso a los servicios de mapas brindados en la Línea de Productos de Software Aplicativos SIG a través de Mapserver, cumpliendo con los estándares del OGC?

Para dar solución al problema planteado surge el ***objetivo general*** Desarrollar una Interfaz de Programación de Aplicaciones (API por sus siglas en inglés) para centralizar los servicios de mapas brindados en la LPS Aplicativos SIG a través de Mapserver.

Una vez identificado el problema, se hace necesario enfocar la investigación en los servicios Web, lo cual constituye el ***objeto de estudio*** de la misma; definiéndose así como ***campo de acción*** los servicios de mapas orientados a la Web brindados en la LPS Aplicativos SIG.

Partiendo del objetivo general de este trabajo, surge la siguiente ***idea a defender:*** Contar con una API que brinde servicios de mapas, permitirá la estandarización y centralización de los mismos dentro de la LPS Aplicativos SIG.

Para la organización del trabajo se proponen los siguientes **objetivos específicos**:

1. Definir el marco teórico de la investigación.
2. Diseñar una API para brindar servicios de mapas orientados a la Web dentro de la LPS Aplicativos SIG para contribuir a la centralización y estandarización de los servicios brindados dentro de la antes mencionada LPS.
3. Implementar el sistema propuesto.
4. Validar el sistema mediante pruebas de software.

Para dar cumplimiento a los objetivos trazados, se definen las siguientes **tareas de investigación:**

1. Definición de los principales conceptos asociados al objeto de la investigación así como a su campo de acción.
2. Análisis de la información existente relacionada con APIs para brindar servicios Web.
3. Análisis de la información existente relacionada con APIs para brindar servicios de mapas orientados a la Web.
4. Selección de las principales herramientas que se ajustan a las necesidades de la investigación para el desarrollo del sistema propuesto.
5. Identificación de las principales funcionalidades del sistema.
6. Diseño del sistema propuesto.
7. Creación de los artefactos para la fase implementación.
8. Implementación del sistema propuesto.
9. Validación de los resultados obtenidos mediantes pruebas para comprobar el cumplimiento de las funcionalidades del sistema.

En la investigación se emplearon diferentes métodos científicos.

Métodos teóricos:

1. **Histórico – lógico:** Se utiliza para el trabajo recopilatorio sobre el proceso de desarrollo de las API, los servicios de mapas y su evolución.
2. **Análisis – síntesis:** Se emplea con el objetivo de analizar las tecnologías y herramientas para el desarrollo de las API y los servicios de mapas Web Feature Service Transactional (WFS-T), así como las metodologías existentes para la construcción de los artefactos para la fase ingenieril de este sistema los IDS y, extraer los elementos más importantes relacionados con el objeto de estudio.

3. **Modelación**: Se utiliza para representar por medio de diagramas el proceso de desarrollo de la API propuesta, teniendo como resultado un mejor entendimiento de la posible solución a implementar.

Métodos empíricos:

1. **Entrevista:** Individuales y colectivas: con los jefes de proyecto de la LPS Aplicativos SIG para comprender la situación real del problema existente, así como las opiniones y sugerencias de los mismos.

Muestreo:

Población: Se utilizará en la investigación al jefe de la LPS y seis especialistas de la misma.

Muestra: Como muestra se tomará al jefe de la Línea y a 4 especialistas de la misma.

Unidad de Estudio: Como unidad de estudio se tomará al jefe de la Línea.

Técnica de muestreo: Dentro de las técnicas de muestreo existentes, se utilizará la técnica de muestreo intencional debido a que consiste en que el buen juicio posibilitará escoger los integrantes de la muestra, por lo que el investigador selecciona explícitamente los elementos que son representativos o con posibilidades de brindar mayor información.

Posibles resultados:

- API para brindar servicios de mapas en la LPS Aplicativos SIG.

El trabajo está estructurado en tres capítulos, los cuales se describen a continuación:

- **Capítulo 1.** En este capítulo se hace referencia a los conceptos asociados a la investigación, se caracteriza el objeto de estudio, se exponen ejemplos de las soluciones existentes relacionadas con el problema a resolver, se definen las diferentes tecnologías que se utilizarán para el desarrollo del sistema y se define la situación problemática que existe actualmente en la LPS Aplicativos SIG.
- **Capítulo 2.** En este capítulo, se describe la solución propuesta para realizar la implementación correspondiente, se analiza tanto el sistema como el negocio, se dejan reflejados los actores, trabajadores, casos de uso, además de las descripciones de los requisitos funcionales y no funcionales y a su vez se modelan los diagramas correspondientes**.**
- **Capítulo 3.** En este capítulo se plantea la construcción de la solución propuesta en el Capítulo 2 a través de los flujos de trabajo Análisis y diseño, Modelo de implementación y Pruebas, generando cada uno de ellos los artefactos correspondientes.

CAPÍTULO 1.

1.1 Introducción.

En este capítulo se abordan diferentes conceptos asociados al problema, que son útiles para un mejor entendimiento de la información, se profundiza en el objeto de estudio. Se realiza un estudio del estado del arte de algunas herramientas y tecnologías usadas actualmente, se describen algunas de las soluciones existentes en el mundo, enmarcándose en la situación problemática, se hace la selección de las herramientas a utilizar para el desarrollo de la solución y se caracteriza el estado actual en la LPS Aplicativos SIG.

1.2 Conceptos asociados al sistema.

1.2.1 Definición de mapa.

El concepto de mapa proviene del término latín *mappa*. Se trata de una representación gráfica y métrica de una porción de territorio sobre una superficie bidimensional, que por lo general suele ser plana, aunque también puede ser esférica como en el caso de los globos terráqueos (1), a estas características se suman que el mapa puede ser además analógico (cuando es impreso sobre papel) o digital (codificado en cifras, almacenado en un ordenador y presentado en una pantalla) es la representación de cualquier estructura de datos usada para reflejar cartográficamente una variable espacial, nominal o cuantitativa, independientemente del modelo de datos utilizado que puede ser vectorial o ráster(2).

De manera que, se define un mapa como la representación gráfica de una porción de territorio en una superficie plana, los mapas poseen propiedades métricas y pueden ser analógicos o digitales, y su estructura de datos puede ser vectorial o ráster.

1.2.2 Servicios de mapas orientados a la Web (WMS).

La primera experiencia relevante en esta dirección se encontró en el año 1959, cuando Waldo Tobler define los principios de un sistema denominado MIMO (Map In - Map Out), con el único fin de aplicar los ordenadores al campo de la cartografía. Este sistema permitía establecer los principios básicos para la creación de datos geográficos, su codificación, análisis y representación dentro de un sistema informatizado. Estos son los elementos fundamentales del software que integra un Sistema de Información Geográfica, los cuales aparecen en todas las aplicaciones desarrolladas desde ese momento (3).

Los servicios WFS (*Web Feature Service*), ofrecen una interfaz de comunicación que permite interactuar con los mapas servidos por el estándar WMS (*Web Map Service*), como por ejemplo, editar la imagen que nos ofrece el servicio WMS o analizar la imagen siguiendo criterios geográficos. Para realizar estas operaciones se utiliza el lenguaje GML (*Geography Markup Language*) que deriva del XML7 (*Extensible Markup Language*), que es el estándar a través del que se transmiten las órdenes (4).

Los Servicios Web surgieron por la necesidad de estandarizar la comunicación entre distintas plataformas y lenguajes de programación, permiten que servicios y software sin importar plataforma o geografía puedan ser combinados para cumplir un objetivo. El servicio *Web Map Service* (WMS), es una especificación de este tipo de tecnología, con el aporte de que produce mapas de datos referenciados espacialmente de forma dinámica a partir de información geográfica. Los mapas creados por WMS se generan normalmente en un formato de imagen como PNG, GIF o JPEG y constituyen componentes claves en la puesta en marcha de los llamados Sistemas de Información Geográfica (SIG) (5).

El *Web Map Service* (WMS) constituye un estándar para la publicación de cartografía en Internet, el cual a través de una interfaz simple HTTP muestra datos referenciados espacialmente, contenidos en una o más bases de datos espaciales. Los mapas generados por un Servicio WMS se producen de manera dinámica y al ser normalmente generados en formatos como PNG, GIF o JPEG pueden ser desplegados en cualquier navegador web o en interfaces que lo soporten. Los datos ofrecidos en los mapas pueden estar dos formatos fundamentalmente: formato ráster y formato vectorial

Modelo Ráster.

En el modelo ráster el espacio es discretizado en pequeños rectángulos o cuadrados, de forma que el tamaño que tienen estos elementos es fundamental y determina la resolución. Utiliza una única primitiva muy similar al punto, el píxel, contracción de las palabras inglesas: *picture element*. Una malla de puntos de forma cuadrada o rectangular que contiene valores numéricos representa las entidades cartográficas y sus atributos a la vez. Los modelos lógicos menos complejos son los basados en el modelo conceptual ráster, en buena medida porque la geo-referenciación y la topología son implícitas a la posición - columna y fila - del píxel en la malla. Cada atributo temático es almacenado en una capa propia.

La precisión de la geo-referenciación en el modelo ráster está sesgada conceptualmente por la porción del territorio que representa el píxel, la cual es la unidad de medida lineal y superficial mínima del sistema. Además a veces no se especifica cómo está georreferenciada la celda, respecto a su ángulo superior izquierdo o a su ángulo inferior izquierdo o respecto a su centro. El modelo conceptual ráster tiene serias limitaciones conceptuales en la precisión de la referenciación, con un margen de error equivalente a la mitad de la base y de la altura del píxel.

Modelo Vectorial.

Las representaciones se centran en la precisión de localización de los elementos geográficos sobre el espacio y donde los fenómenos a representar son discretos, es decir, de límites definidos. Cada una de estas geometrías está vinculada a una fila en una base de

datos que describe sus atributos. Para modelar digitalmente los datos vectoriales se utilizan tres elementos geométricos: el punto, la línea y el polígono.

Puntos: Los puntos se utilizan para las entidades geográficas que mejor pueden ser expresadas por un único punto de referencia. En otras palabras: la simple ubicación. Por ejemplo, las ubicaciones de los pozos, picos de elevaciones o puntos de interés. Los puntos transmiten la menor cantidad de información de estos archivos y al utilizarlos no son posibles las mediciones. También se pueden utilizar para representar zonas a una escala pequeña. Por ejemplo, las ciudades en un mapa del mundo estarán representadas por puntos en lugar de polígonos.

Líneas o polilíneas: Las líneas unidimensionales o polilíneas son usadas para rasgos lineales como ríos, caminos, ferrocarriles, rastros, líneas topográficas o curvas de nivel. De igual forma que en las entidades puntuales, en pequeñas escalas pueden ser utilizadas para representar polígonos. En los elementos lineales puede medirse la distancia.

Polígonos: Los polígonos bidimensionales se utilizan para representar elementos geográficos que cubren un área particular de la superficie de la tierra. Estas entidades pueden representar lagos, límites de parques naturales, edificios, provincias, o propiedades del suelo. Por ejemplo: los polígonos transmiten la mayor cantidad de información en archivos con datos vectoriales y en ellos se pueden medir el perímetro y el área.

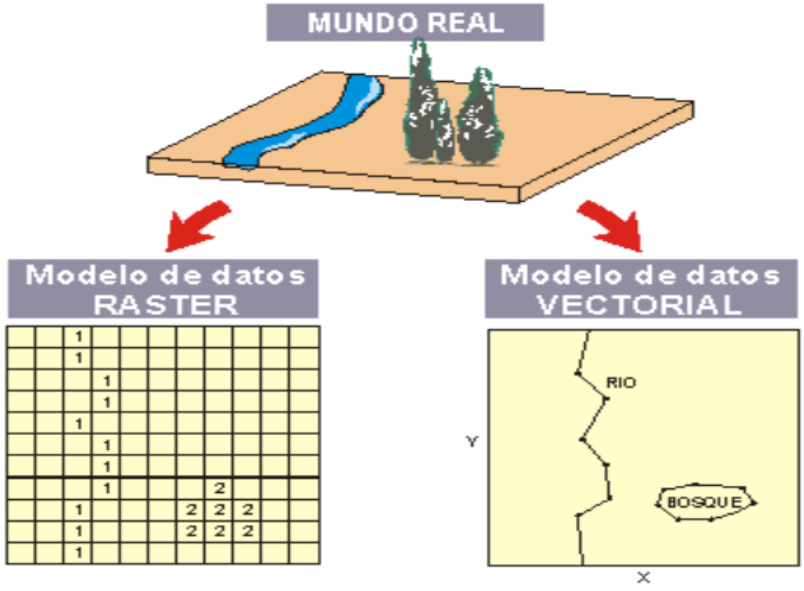


Figura 1. Representación Ráster y Vectorial. (http://*gabrielortiz.com*)

1.2.3 Sistema de Información Geográfica (SIG).

(SIG o GIS, en su acrónimo inglés) es una integración organizada de hardware, software y datos geográficos diseñado para capturar, almacenar, manipular, analizar y desplegar en todas sus formas la información geográficamente referenciada con el fin de resolver problemas complejos de planificación y gestión.

También puede definirse como un modelo de una parte de la realidad referido a un sistema de coordenadas terrestre y construido para satisfacer unas necesidades concretas de información. En el sentido más estricto, es cualquier sistema de información capaz de integrar, almacenar, editar, analizar, compartir y mostrar la información geográficamente referenciada.

En un sentido más genérico, los SIG son herramientas que permiten a los usuarios crear consultas interactivas, analizar la información espacial, editar datos, mapas y presentar los resultados de todas estas operaciones (6).

Los SIG son herramientas necesarias para superar la visión sectorial y consolidar una comprensión integral del territorio, mediante la interacción de las dimensiones ambiental, cultural, económica, social, espacial entre otras. Los SIG desempeñan un papel fundamental en la representación y análisis de la información geográfica, debido a que proveen los medios necesarios para la captura, organización, manipulación y uso de la información. Esto ha permitido que la aplicación de los SIG sea muy diversa (7).

Un Sistema de Información Geográfica (SIG) queda definido entonces como una integración de hardware, software, datos georreferenciados y personal, con el objetivo de capturar, almacenar, manipular, analizar y mostrar la información geográficamente referenciada en pos de resolver problemas complejos de planificación y gestión.

1.2.4 API (Interfaz de Programación de Aplicaciones).

Según Francis y Dominique, en su libro La alquimia de las multitudes, una API se define como: “puertas que los creadores de los programas abren de forma voluntaria para permitir que otros creadores puedan apropiarse de los elementos que interesen de dichos programas y añadir sus propios servicios.” (8).

Otra definición de API la brindan Jorge y Marcell, en su libro Fundamentos de la Telemática, donde queda definida como: “un conjunto de convenios de llamada en programación que definen cómo se invoca un servicio a través de la aplicación.” (9).

Partiendo de los conceptos anteriores, API se define como un conjunto de procedimientos, protocolos, herramientas y funciones que definen una interfaz a un servicio, donde las diferentes funciones pueden ser llamadas a través de varias aplicaciones.

Ejemplos de API.

- **Google Maps:** a través de su acceso a "API", permite ponerle datos e información útil sobre sus mapas, y presentarlos con ciertas búsquedas o funciones personalizadas, desde nuestra propia aplicación.
- **Paypal:** con su "API", permite hacer operaciones de pagos electrónicos usando nuestro propio sistema web, sin necesidad de acceder/operar en la web de Paypal.

- **Twitter:** ha permitido el desarrollo de un gran número de sistemas alternativos y servicios web que operan a través de su "API".
- **UPS y DHL:** (operadoras logísticas internacionales) brindan acceso a sus sistemas desde APIs, para permitirles a los sitios de comercio electrónico poder calcular el costo de envío de los productos vendidos según ciertos parámetros.
- **Facebook Connect:** cede a través del "API" ciertos datos para registrar automáticamente usuarios en otros sitios web, dándoles la posibilidad de registrarse y loguearse con sus propias cuentas de Facebook.

1.3 Objeto de estudio.

1.3.1 Caracterización de la situación actual dentro de la LPS Aplicativos SIG.

La Universidad de las Ciencias Informáticas (UCI), está compuesta por diferentes facultades, las cuales se especializan en diferentes áreas del conocimiento. En cada una de estas facultades existen Centros de Desarrollo de Software, que a su vez están conformados por diferentes Proyectos Productivos, teniendo en su mayoría, como objetivo común, la producción de software.

En el Centro de Desarrollo Geoinformática y Señales Digitales (GEYSED) de la Facultad 6, el cual se dedica al desarrollo de productos, servicios y soluciones informáticas orientadas al procesamiento de Señales Digitales y la Geoinformática. Específicamente en el Departamento de Geoinformática, se encuentran los proyectos productivos dedicados a la realización de Sistemas de Información Geográfica (SIG), destacando la Línea de Productos de Software (LPS) Aplicativos SIG, la cual utiliza GeneSIG como plataforma de desarrollo y se especializa en la implementación y desarrollo de aplicaciones SIG orientadas a la Web.

La LPS ha ido ganando en experiencia y prestigio, y ya cuenta con varios productos en su haber, como son por ejemplo: **SIGMIC**: *Sistema de Información Geográfica para el Ministerio de la Informática y las Comunicaciones*, **SIGSALUD**: *Sistema de Información Geográfica para la Salud*, **SIGRutas**: *Sistema de gestión y análisis de Información Geográfica asociado a las rutas de la UCI en las tres provincias habaneras*, entre otros.

Dentro de la LPS, también se han dado los primeros pasos brindando servicios de mapas en proyectos como por ejemplo Genética Médica y además se han brindado servicios de mapas utilizando Mapserver en proyectos como SIGUCI, pero estos servicios presentan dificultades, pues no cumplen con los estándares establecidos por el OGC, entre ellos con el *Web Feature Service Transactional* (WFS-T) que es un estándar establecido por el OGC, el cual se encarga de la edición del mapa, así como la creación y eliminación de los elementos del mismo.

Otra dificultad presentada en los servicios brindados dentro de la LPS, es que los mismos no permiten al usuario realizar tematizaciones por colores, gráficos de barra o de pastel, o por

símbolos proporcionales. Además los servicios ofrecidos a partir de MapServer, presentan problemas de seguridad, pues este no gestiona la seguridad de los servicios que brinda.
Actualmente la LPS se encuentra trabajando a fin de erradicar estas dificultades para poder brindar servicios de mapa orientados a la web de forma estandarizada y gestionando de manera eficiente la seguridad de los mismos.

1.3.2 Servicios Web (SW).

Recurso de software que se ejecuta en un servidor Web remoto, en respuesta a la solicitud hecha por un cliente. Los servicios Web son equivalentes a cualquier aplicación que corre en un equipo local, sólo que la información necesaria para llevar a cabo una tarea específica es enviada al servidor y el resultado de esa tarea, devuelto al usuario, ambos en la forma de contenido Web (10).
García y Munilla, definen a los SW como: aplicaciones modulares que se pueden publicar, ubicar e invocar desde cualquier punto de red o desde el interior de una red local, basados en estándares abiertos de Internet. No es necesario que el proveedor y el usuario de un SW tengan el mismo sistema operativo y utilicen el mismo lenguaje de programación, dado que se basan en estándares aceptados plenamente por la industria como, *Extensible Markup Language* (XML), *HyperText Markup Language* (HTML) y *Simple Mail Transfer Protocol* (SMTP, Protocolo para la transferencia de Correo) (11).
En el libro ASP.Net, describe a los SW como: una biblioteca de clases accesible vía HTTP (Protocolo de transferencia de hipertexto). Con una interfaz descrita en XML, de forma estándar e independiente del lenguaje de implementación subyacente (12).
Partiendo de los conceptos planteados anteriormente, un SW se define como: un conjunto de aplicaciones que utilizan estándares para el intercambio de comunicación entre sistemas de cualquier plataforma con el objetivo de lograr una integración entre ellas. El objetivo del intercambio de datos entre dichas aplicaciones o tecnologías es el de ofrecer servicios.

SOAP (*Simple Object Access Protocol*).

Es un protocolo de la capa de aplicación para el intercambio de mensajes basados en XML sobre redes de computadoras. Básicamente es una vía de transmisión entre un SOAP transmisor y un SOAP receptor, pero los mensajes SOAP deben interactuar con un conjunto de aplicaciones para que se pueda generar un "diálogo" a través de mensajes SOAP. Un mensaje SOAP es la unidad fundamental de una comunicación entre nodos SOAP (13).
SOAP es básicamente un paradigma de una sola vía pero con la ayuda de las aplicaciones se puede llegar a crear patrones complejos. SOAP básicamente está constituido por:

- Un marco que describe el contenido del mensaje e instrucciones de proceso.
- Un conjunto de reglas para representar los tipos de datos definidos.
- Convenciones para representar llamadas a procedimientos remotos y respuestas.

REST (*Representational State Transfer*).

Rest es un estilo de arquitectura de software para sistemas hipermedias distribuidos tales como la *World Wide Web* (14). Existen otras definiciones válidas de REST como la que plantea que es un conjunto de principios para el diseño de redes, utilizado comúnmente para definir una interfaz de transmisión sobre HTTP. REST para su funcionamiento, utiliza un conjunto de estándares tales como HTML, URL, XML, GIF, JPG y tipos MIME (15).

Los principios de REST son:

- Escalabilidad de la interoperabilidad con componentes.
- Generalidad de interfaces.
- Puesta en funcionamiento independiente.
- Compatibilidad con componentes intermedios.

REST o SOAP.

Cuando se implementan servicios Web, la selección del estándar de comunicación que se utilizará, constituye un factor muy importante para alcanzar el éxito de dichos servicios. Por esta razón es necesario tener en cuenta las ventajas, desventajas, seguridad, flexibilidad de cada uno para entonces elegir el más adecuado al entorno.

La Figura 2. Muestra los niveles de popularidad de los diferentes protocolos para el desarrollo de servicios web.

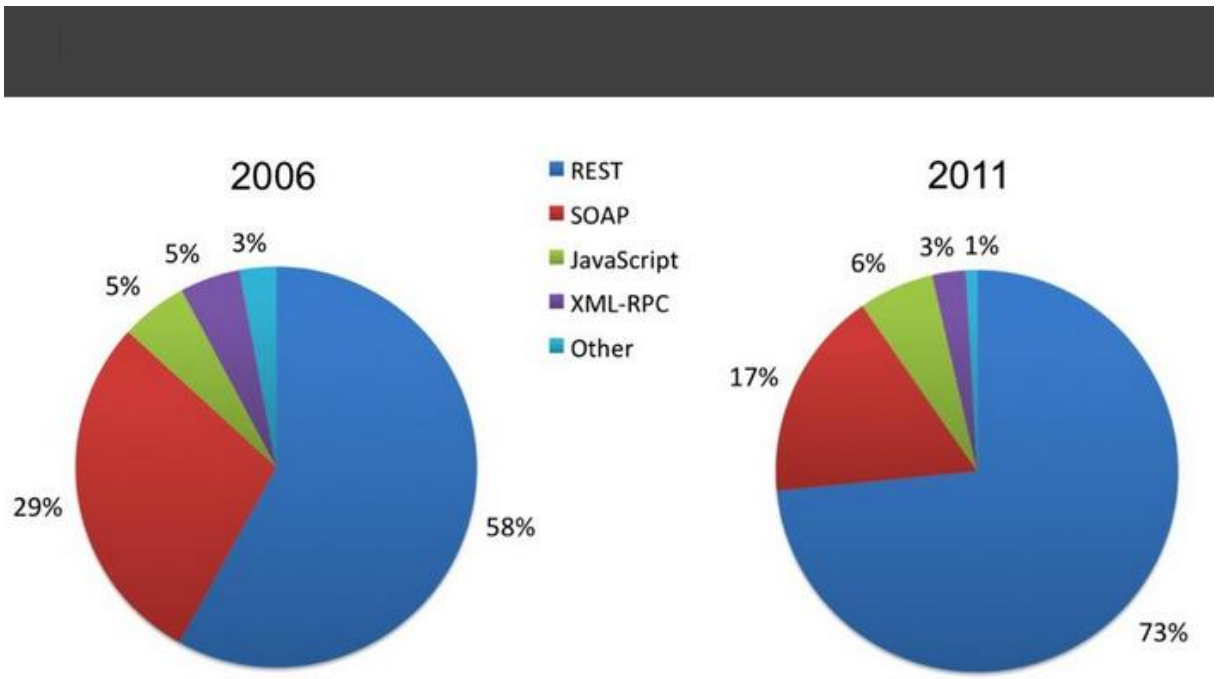


Figura 2. Uso de protocolos para servicios Web (http://programableweb.com**)**

Al observar la imagen, se ve claramente que la mayor competencia está entre REST y SOAP y como en el período 2006 - 2011 la popularidad de REST aumenta en un 15%. Actualmente un gran número de diseñadores de servicios web plantean que SOAP es complejo, por lo que están comenzando a diseñar servicios web basados en REST para servir grandes volúmenes de datos.

Gigantes como Amazon y Google, implementan sus servicios usando REST, y en el caso de Google, el uso de este estilo devino en un proceso de mejora de sus servicios, tras anunciar la descontinuación de su API SOAP a partir del 31 de agosto de 2011, cuando sus especialistas afirmaron, según un comunicado oficial en el Blog de Desarrolladores de Google (*The official Google Code blog*) que: "...dicha API fue disminuyendo su uso considerablemente durante los dos últimos años...".

SOAP constituye la solución adecuada cuando se trata de dominios aislados, donde el entorno de implantación tiene bien definido un dominio. Por esta razón cuando se necesita un servicio web en un entorno empresarial es muy recomendado el uso de SOAP, pero cuando el número de usuarios es muy grande es necesario emplear una estrategia diferente para la interoperabilidad de los sistemas que no poseen la misma API.

A partir del estudio realizado de las dos tendencias, anteriormente mencionadas, finalmente se selecciona REST como estilo arquitectónico para el desarrollo de la solución propuesta.

1.4 Estándares OGC para los servicios de mapa en la Web.

El *Open Geospatial Consortium* (OGC) es un consorcio global de empresas, universidades, organizaciones públicas e individuos unidos por el interés de crear estándares para la informática espacial. Las raíces de la organización se encuentran en la frustración de desarrolladores del SIG GRASS quienes cada día tenían el problema de convertir datos espaciales de un formato a otro. En los siguientes años fundaron la *Open Geospatial Foundation* (OGF) y un año más tarde el OGC. Desde el inicio, el OGC trata sobre código abierto aunque en esos años nadie hablaba de Software Libre. Los iniciadores de la fundación OGF eran los mismos que desarrollaron las primeras versiones del software SIG de código abierto GRASS porque siempre tenían el problema de formatos incompatibles. El OGC crea estándares técnicos para la interoperabilidad de software geomántica (16).

Entre los estándares establecidos por el OGC se encuentran:

1.4.1 Catalogue Service for Web (CSW).

El servicio de catálogo estandariza la forma en que se publican los metadatos de todo tipo de información geográfica tanto la publicada a través de otros estándares como cualquier recurso geográfico (fotografías escaneadas, mapas en papel, etc.). Este servicio es la pieza

clave en una IDE para ofrecer a los usuarios la posibilidad de descubrir qué información publican los diferentes actores: empresas y sobre todo administraciones públicas.

Su objetivo es poder visualizar Información Geográfica. Proporciona una representación, una imagen del mundo real para un área requerida. Esta representación puede provenir de un fichero de datos de un SIG, un mapa digital, una ortofoto, una imagen de satélite.

Está organizada en una o más capas, que pueden visualizarse u ocultarse una a una. Se puede consultar cierta información disponible y las características de la imagen del mapa. Esta especificación del OGC permite superponer visualmente datos vectoriales, ráster, en diferente formato, con distinto Sistema de Referencia y Coordenadas y en distintos servidores, siempre devolviendo al usuario final una imagen rasterizada en un formato ampliamente usado como PNG, JPG o GIF.

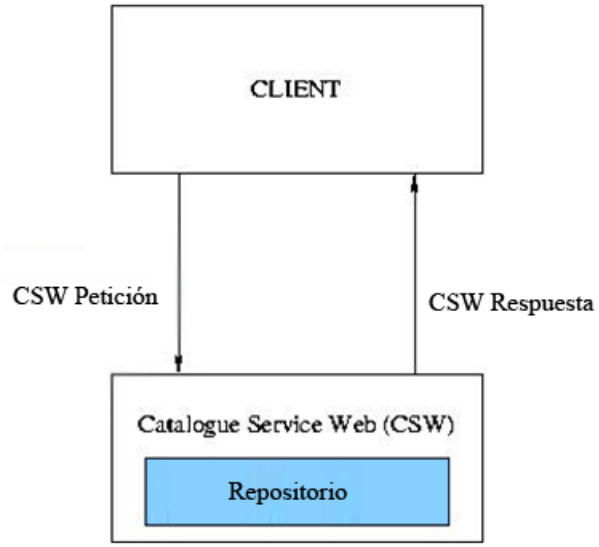


Figura 3. Petición y respuesta en CSW. (4).

Las operaciones que se definen en este estándar OGC son 7, siendo cuatro obligatorias y tres opcionales.

- **GetCapabilities** (Obligatoria): Solicitud de las características del servicio.
- **DescribeRecord** (Opcional): Informa acerca de la estructura de los registros de metadatos.
- **GetDomain** (Opcional): Permite a los usuarios consultar los valores permitidos de un parámetro o propiedad determinado.
- **GetRecordsById** (Obligatoria): Obtiene los registros de metadatos de los recursos catalogados mediante los identificadores de dichos registros.
- **GetRecord** (Obligatoria): Envía una consulta al catálogo (*query*) y devuelve todos los registros de metadatos de los recursos catalogados que satisfacen los requisitos de la consulta.

- **HarvestRecord** (Opcional): Mediante la ubicación de un recurso de metadatos el servicio analiza el recurso y crea o modifica los registros de metadatos de su catálogo. El objetivo es registrar y modificar registros de metadatos de forma automática.
- **Transaction** (Opcional): Permite insertar, actualizar y borrar registros del catálogo de metadatos.

1.4.2 Web Feature Services (WFS).

Permite acceder y consultar todos los atributos de un elemento espacial, como un río, una calle o una ciudad, representado en modo vectorial, con una geometría descrita por un conjunto de coordenadas. Habitualmente los datos proporcionados están en formato GML (otro estándar OGC). Un WFS permite no sólo visualizar la información tal y como permite un WMS, sino también consultarla y descargarla libremente.

Para apoyar la transacción y procesamiento de consultas, las operaciones que se definen son:

- **GetCapabilities:** Un servicio web debe ser capaz de describir sus capacidades. En concreto, se debe indicar que tipos de entidades se pueden atender y qué operaciones son compatibles con cada tipo de entidad.
- **DescribeFeatureType:** Describe la estructura del tipo de objeto geográfico pedido.
- **GetFeature:** Un servicio web debe ser capaz de atender una solicitud para recuperar las instancias de la entidad. Además, el cliente debe ser capaz de especificar las propiedades de la característica para ir a buscar y debe ser capaz de restringir la consulta espacialmente y no espacialmente.
- **GetGmlObject:** Un servicio web puede ser capaz de atender una solicitud para recuperar las instancias de elementos atravesando XLinks que se refieren a sus identificadores XML. Además, el cliente debe ser capaz de especificar si los XLinks anidados en datos de elementos devueltos también deben ser recuperados.
- **Transaction:** Un servicio web puede ser capaz de procesar solicitudes de transacciones de servicios. La petición de la transacción está compuesta de operaciones que modifican las características, es decir crear, actualizar y eliminar operaciones en las características geográficas.
- **LockFeature:** Un servicio web puede ser capaz de procesar una solicitud de bloqueo en una o más instancias de un tipo de característica para la duración de una transacción. Esto asegura que las transacciones serializables son compatibles.

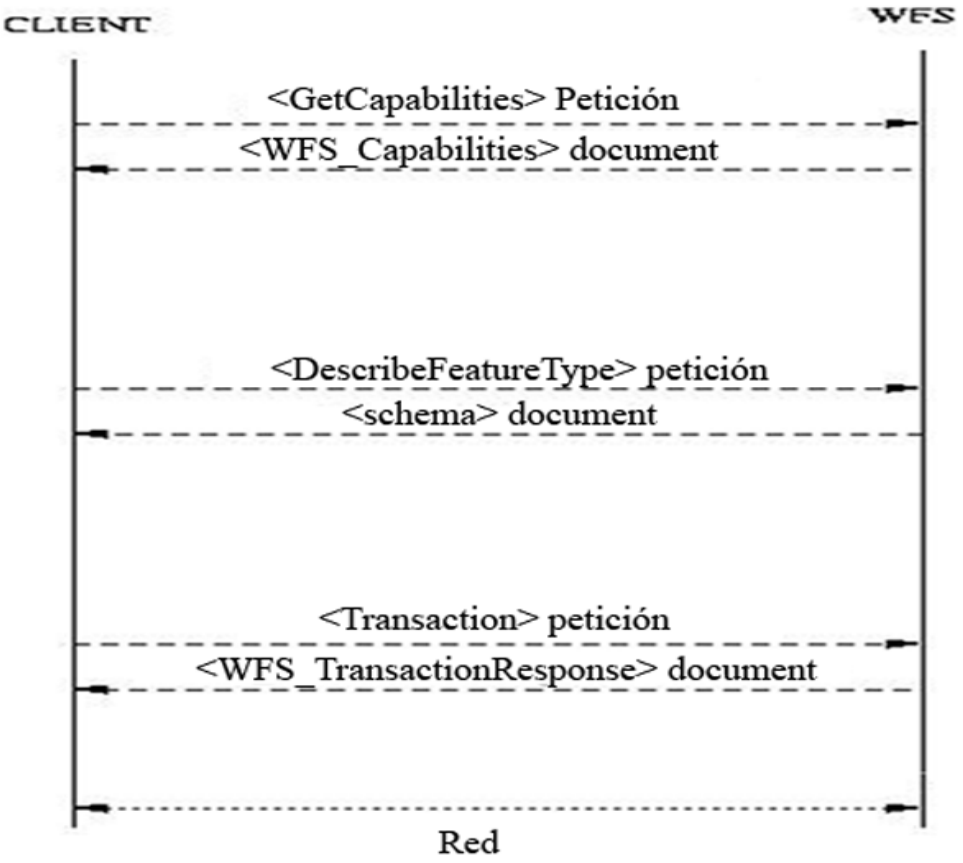


Figura 4. Diagrama de Protocolo WFS. (http://live.osgeo.org)

1.4.3 Web Feature Service Transactional (WFS-T).

Este servicio permite hacer consultas y recuperación de elementos geográficos, así como la edición, creación, eliminación y actualización de estos elementos geográficos del mapa.

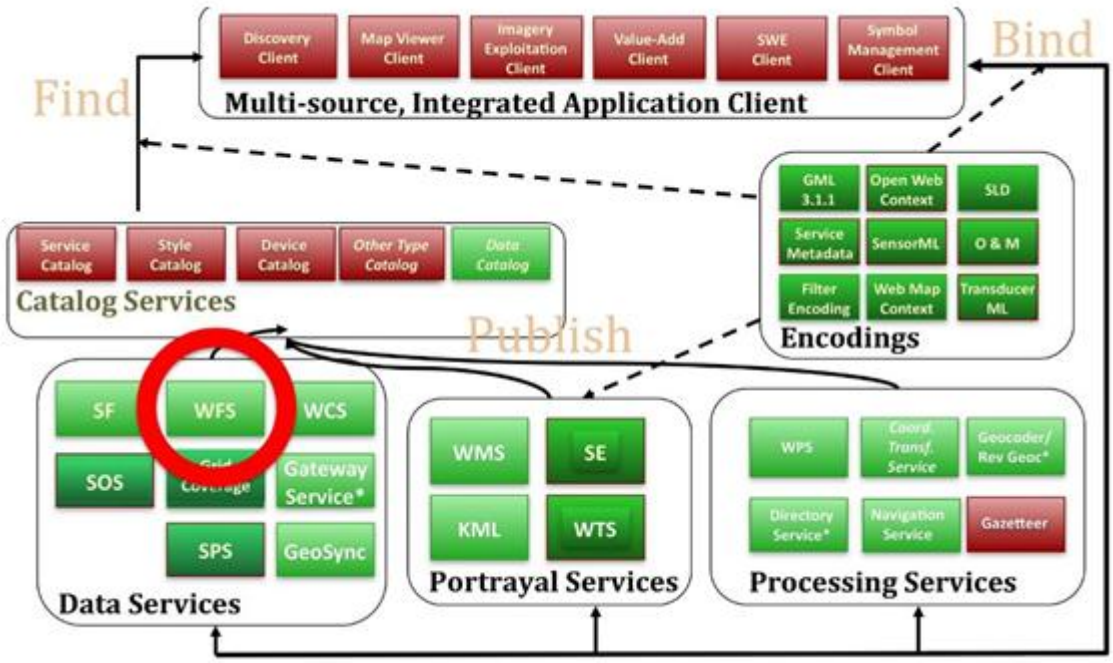


Figura 5. Interface WFS Estándar. (http://live.osgeo.org)

Web Feature Service brinda servicios como:

- Soporta entidades (*features*) simples y complejas.
- Describir los campos de atributos disponibles por entidades (DescribeFeatureType).
- Consultar una colección para un subconjunto de entidades basado en un filtro proporcionado (GetFeature).
- Añadir, editar o borrar entidades (Transaction).

Web Feature Service define estos métodos:

- **GetCapabilities**: Un servicio web debe ser capaz de describir sus capacidades. En concreto, se debe indicar qué tipos de entidades que pueden atender y que operaciones son compatibles con cada tipo de entidad.
- **DescribeFeatureType**: Un servicio web debe poder, previa solicitud, para describir la estructura de cualquier tipo de entidad que puede dar servicio.
- **GetFeature**: Un servicio web debe ser capaz de atender una solicitud para recuperar las instancias de entidad. Además, el cliente debe ser capaz de especificar que propiedades de la característica para ir a buscar y debe ser capaz de restringir la consulta espacialmente y no espacialmente.
- **Transaction**: Un servicio web puede ser capaz de solicitudes de transacciones de servicios. A petición de la transacción está compuesta de operaciones que modifican las características, es decir crear, actualizar y eliminar operaciones en las características geográficas.

- **LockFeature**: Un servicio web puede ser capaz de procesar una solicitud de bloqueo en una o más instancias de un tipo de característica para la duración de una transacción. Esto asegura que las transacciones serializables son compatibles.

1.4.4 Web Coverage Service (WCS).

Este servicio permite acceder a datos ráster sin postproceso. Es decir, permite acceder a los datos reales en formatos comprimidos o sin comprimir y de una única banda o multiespectrales. De este modo es posible publicar modelos digitales del terreno (MDT) o imágenes satelitales mediante estándares.

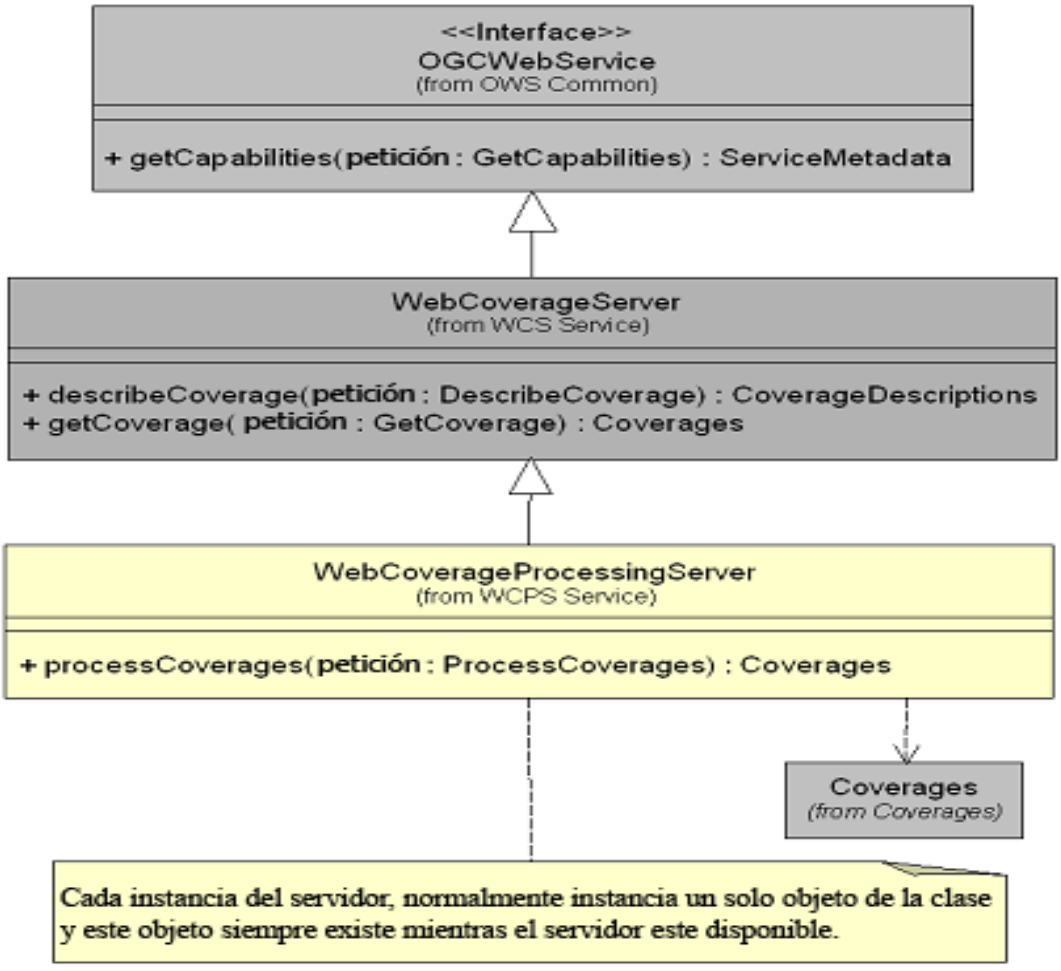


Figura 6. Diagrama UML de la Interfaz WCP. (http://opengeospatial.org)

1.4.5 Styled Layer Descriptor (SLD).

Es un perfil del servicio WMS, por el que se define una codificación que permite definir por parte del usuario personalizaciones en la simbología y los colores de los componentes del mapa. El SLD da respuesta a los usuarios que necesitan personalizar la representación visual del mapa, a través de una norma que define un lenguaje que puedan entender cliente y servidor.

Elementos del *Styled Layer Descriptor* (SLD):

Cabecera de archivo SLD.

- Layer.
- NamedLayer.
- Styles.
- UserLayer.
- UserStyles.
- FeatureTypeStyles.
- Rules.
- Simbolization.

1.4.6 Web Map Context (WMC).

Son documentos XML que contienen toda la información necesaria para mostrar un conjunto de mapas para un área seleccionada. Estos pueden provenir de uno o más servicios de mapas del OGC (WMS) y mostrar la composición de mapa dentro de una determinada área de interés.

Los Documentos *Web Map Context* (WMC) se pueden generar, guardar, reutilizar y cambiar dentro y entre las aplicaciones de mapeo que implementan los estándares OGC WMC. Los Documentos WMC se pueden comparar a los "proyectos" o "espacios de trabajo" en las aplicaciones convencionales de SIG de escritorio, pero son estandarizados y, por tanto compatibles entre diferentes paquetes de software.

1.5 Tecnologías a utilizar.

En este epígrafe se realizará la selección de las tecnologías y herramientas a utilizar en la implementación de la solución.

1.5.1 Servidores de mapas de código abierto.

En los últimos tiempos, el empleo del software libre ha crecido de manera sin precedentes, de igual manera también se ha incrementado el uso de herramientas SIG. Entre los servidores de mapas de código abierto existentes en el mundo se pueden mencionar: Map Guide Open Source (17), GeoServer (18), Deegree.(19), UMN MapServer (20). Para el desarrollo del presente trabajo fue escogido MapServer por poseer una serie de características que son necesarias para el desarrollo del sistema propuesto, además de ser el servidor empleado por la LPS Aplicativos SIG. Dichas características son:

- Salidas cartográficas avanzadas.
- Ejecución de la aplicación y dibujo de elementos según la escala.
- Generación automática de los elementos de un mapa (barra de escala, mapa de referencia y leyenda).
- Soporte a los lenguajes de scripting y ambientes de desarrollo más populares.PHP, Python, Perl, Ruby, Java y .NET.

- Soporte multiplataforma: Linux, Windows, Mac OS X, Solaris, entre otros.
- Múltiples formatos de datos vectoriales y ráster: Shapefile de ESRI.

1.5.2 Servidores Web.

Existen numerosos servidores web entre los que se pueden mencionar: Microsoft IIS (21), Java System Web Server (22), y Apache 2 (23). Se escogió Apache2 porque además de estar definido su uso por la LPS, además presenta un grupo de características que se mencionan a continuación:

- Es un servidor web HTTP de código abierto, que soporta varias plataformas, como son: Unix (GNU/Linux, BSD, etc.), Macintosh y Microsoft Windows, entre otras.
- Es una plataforma robusta y potente para el desarrollo y la distribución de aplicaciones orientadas a la web.

1.5.3 Lenguajes de programación de código abierto.

Entre los varios lenguajes de programación existentes de código abierto están: ASP.NET (24), Perl.(25), PHP5 (26), entre otros. Para el desarrollo del presente trabajo se escoge este último, pues es el lenguaje utilizado dentro de la LPS, y posee las siguientes características:

- Es un lenguaje multiplataforma.
- Facilita la conexión con un gran número de bases de datos como PostgreSQL, MySQL y Oracle, entre otros.
- Es extensible a través de la creación de módulos y bibliotecas externas.

1.5.4 Sistema Gestor de Base de Datos.

En el mundo existen numerosos sistemas para la gestión de bases de datos entre ellos se pueden mencionar: Oracle, DB2, PostgreSQL, MySQL, MS SQL Server entre otros. En el presente trabajo se ha elegido **PostgreSQL** con extensión **PostGIS**, pues además de ser multiplataforma y de código abierto, poseen una gran cantidad de ventajas, fundamentalmente porque existe un módulo integrado en **PostGIS**, que añade la capacidad de almacenamiento/recuperación según la especificación SFS (*Simple Features Specification*) del consorcio internacional *Open GeoSpatial* (OGC), lo cual es ideal para esta solución, es decir permite almacenar y manejar objetos geográficos.

PostgreSQL con extensión **PostGIS**, funciona como un conjunto, uniendo la características de **PostgreSQL** como: funciona en la mayoría de los Sistemas Operativos más utilizados, entre ellos Linux y Windows; utiliza principalmente SQL como lenguaje de consulta a la base de datos; admite varios lenguajes como Java, Perl, Python, Ruby, C/C++, etc., y su lenguaje nativo (PL/PGSQL). Y las características de **PostGIS**, que permiten el uso de objetos GIS, al mismo tiempo, posee soporte para índices GiST basados en R-Tree y funciones básicas para el análisis de objetos GIS.(27).

Con **PostGIS** se pueden usar todos los objetos que aparecen en la especificación OpenGIS como puntos, líneas, polígonos, multilíneas, multipuntos y coleccione geométricas. **PostGIS** es además, la alternativa de software libre más avanzada y de alto rendimiento, otra de las características que lo hace imprescindible en esta investigación es la integración de muchas de sus funciones con el servidor de mapas MapServer que se utiliza para el desarrollo de la aplicación.(28).

1.5.5 Metodologías de desarrollo de software.

Las metodologías de desarrollo de software surgen ante la necesidad de utilizar una serie de procedimientos, técnicas, herramientas y soporte documental a la hora de desarrollar un producto software. Dichas metodologías pretenden guiar a los desarrolladores al crear un nuevo software, pero los requisitos de un software a otro son tan variados y cambiantes, que ha dado lugar a que exista una gran variedad de metodologías para la creación del software.(29)

Existen numerosas metodologías entre las están Programación Extrema o XP (*Extreme Programming*), RUP (*Rational Unified Process*), entre otras. Para el desarrollo del presente trabajo fue utilizada RUP, debido a que el equipo de desarrollo está familiarizado con la misma, es una metodología mediante la cual se generan los artefactos necesarios para el correcto diseño del sistema. La misma presenta las siguientes características.(29):

RUP es un proceso y 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, estos son captados cuando se modela el negocio y representados a través de los requerimientos. A partir de aquí los casos de uso guían el proceso de desarrollo ya que los modelos que obtenidos como resultado de

los diferentes flujos de trabajo, representan la realización de los casos de uso (cómo se llevan a cabo).

- **Dirigido por casos de uso:** Los casos de uso reflejan lo que los usuarios futuros necesitan y desean, estos son captados cuando se modela el negocio y representados a través de los requerimientos. A partir de aquí los casos de uso guían el proceso de desarrollo ya que los modelos que obtenidos 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 sea desarrollada en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros.

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 toma 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.

1.5.6 El Lenguaje Unificado de Modelado (UML).

UML es un "lenguaje" para el modelado orientado a objetos. UML se usa para definir un sistema de software; para detallar los artefactos en el sistema; para documentar y construir. UML se puede usar en una gran variedad de formas para soportar una metodología de desarrollo de software (como el Proceso Unificado de Rational) pero no especifica en sí mismo qué metodología o proceso usar (30).

Debido a las características anteriormente mencionadas, se escoge UML como lenguaje de modelado pues presenta un conjunto de herramientas que permiten modelar (analizar y diseñar) sistemas orientados a objetos, con la modelación de los de artefactos durante las primeras fases del ciclo de vida del software, los implementadores, en fases posteriores tendrán mayor dominio comprensión sobre qué es lo que se debe implementar, permitiendo tanto al cliente como a los desarrolladores tener una representación real del alcance y factibilidad que puede llegar a tener el producto.

1.5.7 Herramienta Case (Computer Aided Software Engineering).

Las herramientas CASE brindan soporte para desarrollar y mantener software. Son herramientas individuales que ayudan al desarrollador de software o administrador de proyecto durante una o más fases del desarrollo de software (31).

Para el desarrollo del presente trabajo se selecciona la herramienta CASE Visual Paradigm.

Visual Paradigm.

Es una herramienta CASE profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML (32).

Características:

- Soporte de UML versión 2.1.
- Diagramas de Procesos de Negocio.
- Ingeniería inversa.
- Modelo a código, diagrama a código.
- Diagramas de flujo de datos.
- Soporte ORM.
- Generación de bases de datos.
- Transformación de diagramas de Entidad-Relación en tablas de base de datos.
- Distribución automática de diagramas.

1.5.8 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 (33).

Para el desarrollo del presente trabajo se selecciona **NetBeans** en su versión 7.2.1**.** NetBeans es una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java, pero se utiliza para desarrollar en varios lenguajes de programación (34).

Características:

- NetBeans es un Entorno de Desarrollo Integrado (IDE) de código abierto y gratuito.

- Es multiplataforma, haciendo posible su uso en diversos sistemas operativos.
- Brinda las herramientas necesarias para el desarrollo en lenguajes como Java, XML, HTML, PHP, Java Script y es compatible con Symfony2.
- Presenta aplicaciones para la detección y tratamiento de errores.
- Tiene una amplia comunidad de desarrollo y soporte.

1.5.9 Framework de desarrollo.

En el desarrollo del presente trabajo se escoge como framework de desarrollo Symfony2, debido a que, además de ser *Open-Source*, es un framework rápido, flexible y fácil de aprender que permite a los desarrolladores construir aplicaciones webs más mantenibles (35).

Entre sus características están:

- **Alto rendimiento**: Symfony2 ha sido desarrollado teniendo en cuenta el rendimiento como mayor prioridad, por lo que es uno de los frameworks más rápidos, hasta 3 veces más rápido que Symfony 1.4 o Zend Framework 1.10 y consume la mitad de la memoria.
- **Usabilidad avanzada**: Es un framework fácil de utilizar gracias a que cuenta con una API de desarrollo muy intuitiva.
- **Extensible**: Symfony2 se construye a base de bundles (plugins en Symfony 1).
- **Flexible**: Gracias a que Symfony2 cuenta con un micro-kernel basado en un contenedor de inyección de dependencia y un manejador de eventos es muy fácil configurarlo a voluntad.
- **Construido para desarrolladore**s: Symfony2 proporciona las herramientas que en gran medida mejoran la productividad de los desarrolladores, como la famosa barra de depuración web, soporte nativo de entornos, páginas detalladas de errores entre otras cosas.
- **Construido en base a otros grandes framework**s: Symfony2 tomó lo mejor de los conceptos de otros frameworks de desarrollo como Django, Spring y Ruby on Rails. También aprovecha componentes de Zend Framework y de Doctrine.
- **Listo para usar**: Symfony2 cuenta con todas las características que el desarrollador de aplicaciones web necesita. También proporciona seguridad integrada y promueve el desarrollo web utilizando buenas prácticas.

1.6 Análisis de soluciones anteriores.Google Maps.

Google Maps es uno de los servicios de mapas que brinda Google, el cual ofrece imágenes de mapas e incluso fotos satelitales del mundo entero, permite mover y hacer zoom a la imagen obtenida. El usuario puede crear una lista de puntos para saber cómo llegar a su destino, calculando el tiempo necesario y la distancia recorrida entre las ubicaciones.

También ofrece la posibilidad de que cualquier propietario de una página Web integre muchas de sus características a su sitio de forma gratuita, pero si se pretende hacer uso comercial de su servicio, tiene que pagarlo, de esta forma se le incluyen un conjunto de nuevas funcionalidades (36).

Google Maps no funciona siguiendo el estándar WMS, sino que funciona a través de su propia API, mostrando las imágenes que ya tiene generadas y georreferenciadas, troceadas a las diferentes escalas y almacenadas en sus servidores. El proceso es verdaderamente rápido, pues, monta la imagen por cuadriculas, cuando se desplaza el mapa solo tiene que añadir las cuadriculas adyacentes en vez de cargar toda la imagen.

Google Maps cuenta con las siguientes funciones:

- **Paneo de los mapas:** permite realizar paneo del mapa haciendo clic y arrastrando el puntero del ratón, obteniendo de esta forma los mapas adyacentes.
- **Zoom:** permite realizar zoom del mapa mediante la rueda de desplazamiento del ratón, ampliando o reduciendo el mapa.
- **Imágenes por satélite:** permite obtener imágenes satelitales con datos de mapa superpuestos, y realizar las funciones de zoom y paneo.
- **Rutas detalladas:** permite introducir una dirección y Google Maps se encargará de señalarla y de trazar el itinerario.
- **Street View:** permite ver imágenes de las calles y desplazarte por ellas.

A pesar de que la API de Google brinda numerosos servicios, esta no ofrece una solución completa a la problemática en cuestión, pues dicha API no permite a los usuarios realizar tematizaciones, personalizar sus búsquedas, ni permite la edición de los mapas brindados; además de que, para poder utilizarla con fines de negocio, se debe pagar por ella, o sea no es libre.

1.6.2 Deegree.

Es libre, estable, potente y fácil de usar. Deegree es el conjunto más completo de implementaciones de estándares del *Open Geospatial Consortium* (OGC) en software libre y abierto, abarcando desde Servicios Transaccionales ***Web Feature Service*** a la visualización de datos tridimensionales a través de *Servicios Web Terrain*, y muchos más. Es una solución de Sistemas de Información Geográfica e Infraestructuras de Datos Espaciales (IDE) basada tanto en Web, como desktop. Está compuesto de un conjunto Interfaces de Aplicación (APIs) Java y un potente mapeo objeto-relacional para esquemas espaciales simples y complejos. Deegree también proporciona un conjunto de *Web Services* estándar de mapas, entidades (features) y servicios de catálogo, así como sensores y servicios de procesamiento (19).

Características de Deegree:

- **Web Map Service.**
  - Contenidos de capas flexible.
  - Soporta y utiliza definición de estilos (SLD 1.0).
  - Capacidad de generación de gráficos (tartas, barras, líneas).
  - Fuentes de datos: Todos los Servicios Web comunes OGC (WMS, WFS, WCS), PostgreSQL/PostGIS, Oracle Spatial, y arbitrariamente sentencias SQL se pueden utilizar para crear contenidos de capas WMS.
  - Muy estable, incluso para grandes escalas.
  - Soporta HTTP GET, HTTP POST y peticiones de información de geometrías (*features*).
  - Certificado de soporte OGC.
- **Web Feature Service.**
  - Soporta entidades (*features*) simples y complejas.
  - Transformación de coordenadas para más de 3000 Sistemas de Referencia de Coordenadas.
  - Soporte de formatos de salidas flexibles.
  - Soporte de directiva INSPIRE.
- **Web Coverage Service.**
  - Soporta peticiones HTTP GET y HTTP POST.
  - Fuentes de datos: imágenes (tif, png, jpeg, gif, bmp); GeoTIFF; ECW files; Oracle GeoRaster.
  - Alta velocidad de acceso para coberturas grandes.
- **Catalogue Service-Web.**
  - Fuentes de datos: PostgreSQL-Database; Oracle-Database.
  - Peticiones soportadas: GetCapabilities; DescribeRecord; GetRecordById; GetRecords; Transaction - Insert, Update, Delete; Harvesting.
- **Web Perspective View Service.**
  - Fuentes de datos: remote/local-WMS, remote/local-WFS, local-WCS, Postgres/PostGIS, Oracle Spatial.
  - Modelos de elevación que pueden ser de datos vectoriales o ráster.
  - Peticiones: Get3DFeatureInfo, GetView.

Por otra parte DEEGREE aunque ofrece servicos estandarizados, tampoco permite tematizar por colores, gráficos de barra o de pastel, ni gestiona la seguridad de sus servicios, por lo que no ofrece solución a la situación problemática en cuestión.

1.6.3 Open Street Map.

OpenStreetMap facilita los datos en bruto para su descarga desde su propia página web. Estos pueden ser modificados para cada proyecto así como presentados con estilos de renderizado personalizados.

- Se puede producir mapas de carreteras, también para la creación de mapas de senderismo, mapas de vías ciclables, mapas náuticos, mapas de estaciones de esquí, etc.
- Se usan en aplicaciones para el cálculo de las rutas óptimas para vehículos y peatones.
- Muestra cartografía mediante mapas en línea con diferentes estilos de renderizado y visualización.
- El editor on-line Potlatch. Por su sencillez es el editor principal que se encuentran los contribuidores en la misma página del proyecto. Esta programado bajo tecnología Adobe Flash.
- El editor off-line JOSM. Es una aplicación de escritorio que el usuario se descarga y ejecuta directamente en ordenador. Está basado en tecnología Java y es ultiplataforma. Actualmente es el editor más avanzado que posee la comunidad por su versatilidad y características, y en el cual están involucradas en su desarrollo un mayor número de personas.
- El editor off-line Merkaartor. Este editor multiplataforma posee un cuidado entorno grafico haciendo uso de la biblioteca Qt.

Open Street Map se especializa en mostrar información sobre calles, rutas, viales entre otros, pero no muestra otras entidades, como construcciones, estados, países, etc., por esta razón no ofrece solución al problema en cuestión.

1.6.4 Infraestructuras de Datos Espaciales de la República de Cuba (IDERC).

La Infraestructura de Datos Espaciales de la República de Cuba (IDERC) abarca las políticas, tecnologías, estándares y recursos humanos necesarios para la efectiva recolección, administración, acceso, entrega y utilización de los datos espaciales a nivel nacional en función de la toma de decisiones económicas, políticas y sociales, y del desarrollo sostenible (37).

La IDERC brinda un conjunto de datos como son: Distribución Político-Administrativa, Hidrografía, Puntos Poblados, Asentamientos urbanos, Elevaciones, Viales y Nombres Geográficos de la república de Cuba. Posee además un grupo de servicios tales como el Diccionario Geográfico, Nombres Geográficos y un Visor de Mapas accesible mediante mapas interactivos.

1.7 Conclusiones parciales.

En el presente capítulo se abordaron los elementos fundamentales relacionados con los servicios de mapas orientados a la Web, así como los estándares establecidos por el OGC, que comprenden los conceptos básicos asociados al objeto de estudio lo que posibilitó tener una mejor visión del problema.

Partiendo de el estudio de soluciones existentes de APIs para brindar servicios de mapas a nivel internacional, se puede afirmar que los mismos no brindan una solución completa al problema existente dentro de la LPS Aplicativos SIG, dicho estudio, además, sirve de apoyo a la investigación para lograr un mejor entendimiento de estos sistemas y al análisis de la situación problemática antes planteada.

Después de analizar la plataforma de desarrollo utilizada dentro de la LPS y de los sistemas y servicios desarrollados hasta el momento, es posible afirmar, que no se satisfacen en su totalidad las necesidades actuales existentes dentro de la LPS Aplicativos SIG, lo que trae consigo que sea necesario la implementación de un sistema que dé respuesta a todas las necesidades existentes dentro de la mencionada LPS.

CAPÍTULO 2.

2.1 Introducción.

En el presente capítulo se enumeran los requisitos funcionales (RF) y los requisitos no funcionales (RNF) del sistema, el modelo de dominio del mismo y las descripciones del sistema y los casos de uso (CU), lo cual contribuirá a la comprensión del sistema de forma general y se presentan los diagramas correspondientes y el modelo de clases del sistema.

2.2 Modelo de Dominio.

En un modelo de dominio se capturan los tipos más importantes de objetos que existen o los eventos que suceden en el entorno donde estará el sistema, trayendo consigo la ventaja de ayudar a usuarios, clientes y desarrolladores a utilizar un vocabulario común para entender el contexto en que se desarrollará el sistema.

Diagrama de clases del Modelo de Dominio.

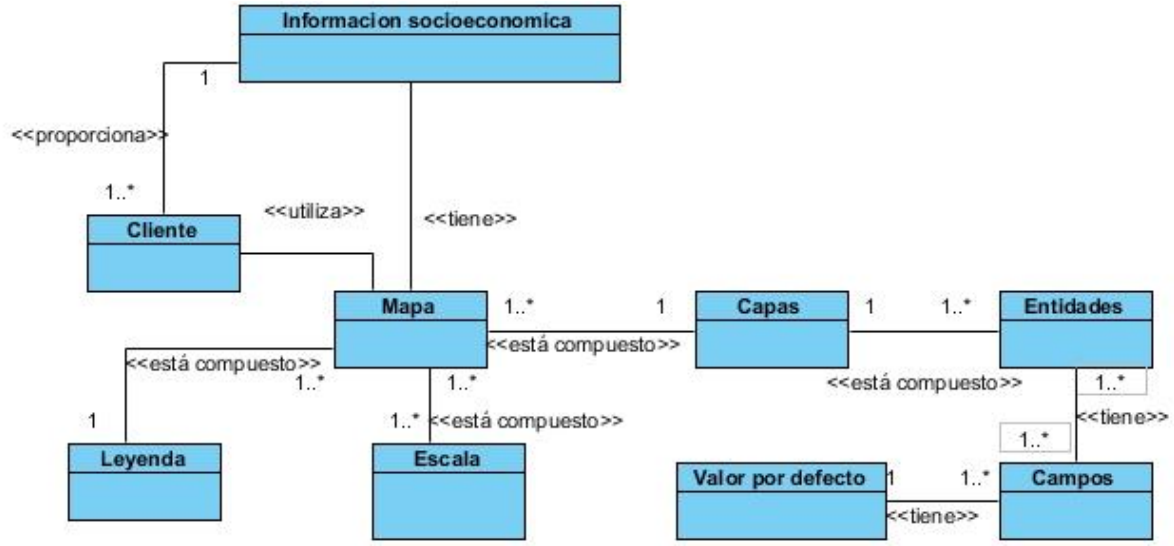


Figura 7. Modelo de Dominio.

2.2.1 Definición de las clases del Modelo de Dominio.

En este epígrafe se explica en que consistes cada una de las clases del Modelo de Dominio.

- **Cliente:** se refiere a una aplicación o sistema que tiene los privilegios para interactuar con las funcionalidades del sistema.
- **Mapa:** cartografía de una localización referenciada geográficamente la cual posee capas, leyenda y escalas.
- **Capas:** agrupación de un grupo de entidades de un mismo tipo en un mapa.
- **Escala:** es la proporción respecto a la realidad a la que están representados los distintos objetos en un mapa.
- **Leyenda:** es una lista explicativa que define los símbolos utilizados en un mapa.
- **Entidades:** se refiere a los distintos elementos representados en el mapa, países, municipios, ríos, lagos, calles, construcciones entre otros.

- **Campos:** se refiere a los atributos de la entidad a la que hace referencia.
- **Información socioeconómica:** Es un conjunto organizado de datos procesados referentes al aspecto social y económico de cualquier lugar de interés del país.

2.3 Levantamiento de requisitos.

Un requisito, es una descripción de las necesidades y aspiraciones que surgen respecto a un producto determinado. El objetivo principal de la definición de los requisitos es identificar lo que se necesita para el funcionamiento correcto del producto La definición y descripción de estos, ayuda a que exista una mayor comunicación entre el cliente y el equipo de desarrollo. Existen dos tipos de requerimientos, o requisitos: los requisitos funcionales y los no funcionales.

2.3.1 Requisitos funcionales.

Los requisitos funcionales son declaraciones de los servicios que debe proporcionar el sistema, describe lo que este debe hacer.

Para el desarrollo de la solución propuesta, el sistema debe cumplir con los siguientes requisitos funcionales.

RF 1. Autenticar usuario: el sistema debe ser capaz de validar las credenciales de un usuario.

RF 2. Adicionar aplicación: el sistema debe permitir adicionar una aplicación.

RF 3. Modificar aplicación: el sistema debe permitir modificar una aplicación.

RF 4 Eliminar aplicación: el sistema debe permitir eliminar una aplicación.

RF 5 Mostrar aplicación: el sistema debe ser capaz de mostrar la lista de las aplicaciones.

RF 6. Adicionar rol: el sistema debe permitir adicionar un rol.

RF 7. Modificar rol: el sistema debe permitir modificar un rol.

RF 8. Eliminar rol: el sistema debe permitir eliminar un rol.

RF 9. Mostrar rol: el sistema debe ser capaz de mostrar la lista de roles de la aplicación.

RF 10. Adicionar usuario: el sistema debe permitir adicionar un usuario.

RF 11. Modificar usuario: el sistema debe permitir modificar un usuario.

RF 12. Eliminar usuario: el sistema debe permitir eliminar un usuario.

RF 13. Mostrar usuario: el sistema debe ser capaz de mostrar la lista de usuarios de la aplicación.

RF 14. Adicionar capa: el sistema debe permitir adicionar una capa.

RF 15. Modificar capa: el sistema debe permitir modificar una capa.

RF 16. Eliminar capa: el sistema debe permitir eliminar una capa.

RF 17. Mostrar capa: el sistema debe ser capaz de mostrar la lista de capas de la aplicación.

RF 18. Adicionar campo: el sistema debe permitir adicionar un campo.

RF 19. Modificar campo: el sistema debe permitir modificar un campo.

RF 20. Eliminar campo: el sistema debe permitir eliminar un campo.

RF 21. Mostrar campo: el sistema debe ser capaz de mostrar la lista de campos de la aplicación.

RF 22. Adicionar acción: el sistema debe permitir adicionar una acción.

RF 23. Modificar acción: el sistema debe permitir modificar una acción.

RF 24. Eliminar acción: el sistema debe permitir eliminar una acción.

RF 25. Mostrar acción: el sistema debe ser capaz de mostrar la lista de acciones de la aplicación.

RF 26. Adicionar valor por defecto: el sistema debe permitir adicionar un valor por defecto.

RF 27. Modificar valor por defecto: el sistema debe permitir modificar un valor por defecto.

RF 28. Eliminar valor por defecto: el sistema debe permitir eliminar valor por defecto.

RF 29. Mostrar valor por defecto: el sistema debe ser capaz de mostrar los valores por defecto.

RF 30. Adicionar entidad: el sistema debe permitir adicionar una entidad.

RF 31. Modificar entidad: el sistema debe permitir modificar una entidad.

RF 32. Eliminar entidad: el sistema debe permitir eliminar una entidad.

RF 33. Mostrar entidad: el sistema debe ser capaz de mostrar el listado de las entidades de la aplicación.

RF 34. Asignar rol: el sistema debe ser capaz de asignar rol a los usuarios de una aplicación.

RF 35. Asignar usuarios: el sistema debe ser capaz de asignar usuarios a una aplicación.

RF 36. Asignar capas: el sistema debe ser asignar las capas a un mapa de la aplicación.

RF 37. Asignar entidades: el sistema debe ser capaz de asignar las entidades a la aplicación.

RF 38. Mostrar capacidades: el sistema debe ser capaz de mostrar las capacidades del servicio de mapas.

RF 39. Mostrar mapa: el sistema debe ser capaz de mostrar los mapas de la aplicación.

RF 40. Mostrar leyenda: el sistema debe ser capaz de mostrar la leyenda del mapa solicitado en la aplicación.

RF 41. Mostrar información: el sistema debe ser capaz de mostrar la información de las entidades de una aplicación.

RF 42. Describir las capas de un mapa: el sistema debe ser capaz de mostrar la descripción de las capas de una aplicación.

RF 43. Describir tipos de objeto: el sistema debe ser capaz de mostrar los tipos de objetos existentes en una capa de una aplicación.

RF 44. Tematizar por colores: el sistema debe ser capaz de, a partir de un indicador dado, tematizar por colores la capa seleccionada en función del parámetro dado.

RF 45. Tematizar por gráfica de pastel y gráfica de barras: el sistema debe ser capaz de, a partir de dos o más parámetros dados, tematizar en función de dichos parámetros, la información de la capa seleccionada en graficas de pastel o de barras.

RF 46. Tematizar por símbolos proporcionales: el sistema debe ser capaz de a partir de un parámetro dado, tematizar por objetos relacionales la información de la capa seleccionada.

2.3.2 Requisitos no funcionales.(RNF).

Los requisitos no funcionales son restricciones de los servicios, cualidades y ofrecidas por el sistema y surgen en función de las necesidades del usuario. Estas propiedades están en función de brindar al producto, usabilidad, rapidez, confiabilidad y atractivo.

Usabilidad.

RNF 1. El sistema esta creado para ser utilizado por personas con conocimientos básicos de informática.

Rendimiento.

RNF 2. El sistema debe menos de 5 segundos a la hora de procesar la información y dar respuesta a las peticiones de los usuarios.

Confidencialidad.

RNF 3. La información manejada por el sistema debe estar protegida ante el acceso no autorizado y la divulgación.

Disponibilidad.

RNF 4. El sistema debe estar disponible todo el tiempo para los usuarios autorizados.

RNF 5. El período entre fallos recuperables, como por ejemplo fallos en el servidor no debe exceder las 24 horas.

Restricciones de diseño

RNF 6. El producto de software final debe diseñarse sobre una arquitectura modelo -vista-controlador.

Interfaces de hardware

Para las computadoras clientes:

RNF 7. Se requiere que tengan una tarjeta de red.

RNF 8. Al menos 128 MB de memoria RAM.

RNF 9. Procesador a 512 MHz como mínimo.

Para los servidores.

RNF 10. Se requiere una tarjeta de red.

RNF 11. El Servidor de Mapas y el servidor Web tengan como mínimo 2 GB de RAM y 40 GB de disco duro.

RNF 12. El Servidor de Base de Datos tenga como mínimo 2GB de RAM y 40GB de disco duro.

RNF 13. Procesador a 3 GHz como mínimo.

La construcción de la aplicación funcionará bajo los conceptos de arquitectura cliente-servidor. Por tanto el servidor del usuario final debe tener como requerimientos mínimos de software:

Para las PCs clientes:

- Un navegador como Mozilla Firefox, Zafari u otro navegador que cumpla con los estándares W3C.
- Sistema operativo: GNU/Linux, Windows y Mac OS.

Para los Servidores:

- Sistemas operativos GNU/Linux Ubuntu Server 11.04. en adelante.
- Servidor Web Apache 2.0 o superior, con módulo PHP 5 configurado con la extensión pgsql incluida.
- PostgreSQL como Sistema Gestor de Base de Datos.
- PostGis como extensión de PostgreSQL como soporte de datos espaciales.
- MapServer 5.2.2 o superior, con extensión PHP/Mapscript.

2.4 Descripción del sistema.

2.4.1 Descripción de los actores.

Un actor no es parte del sistema en desarrollo, es un agente externo que interactúa con el mismo en pos de obtener un resultado esperado. El sistema cuenta con los actores que se especifican a continuación:

Actores	Descripción
Usuario	Es la persona que utiliza, según el nivel de acceso que tenga, las funcionalidades.
Administrador	Es la persona que gestiona los niveles de acceso y los permisos de los diferentes usuarios del sistema.

Tabla 1. Descripción de los actores.

2.4.2 Diagrama de casos de uso del sistema.

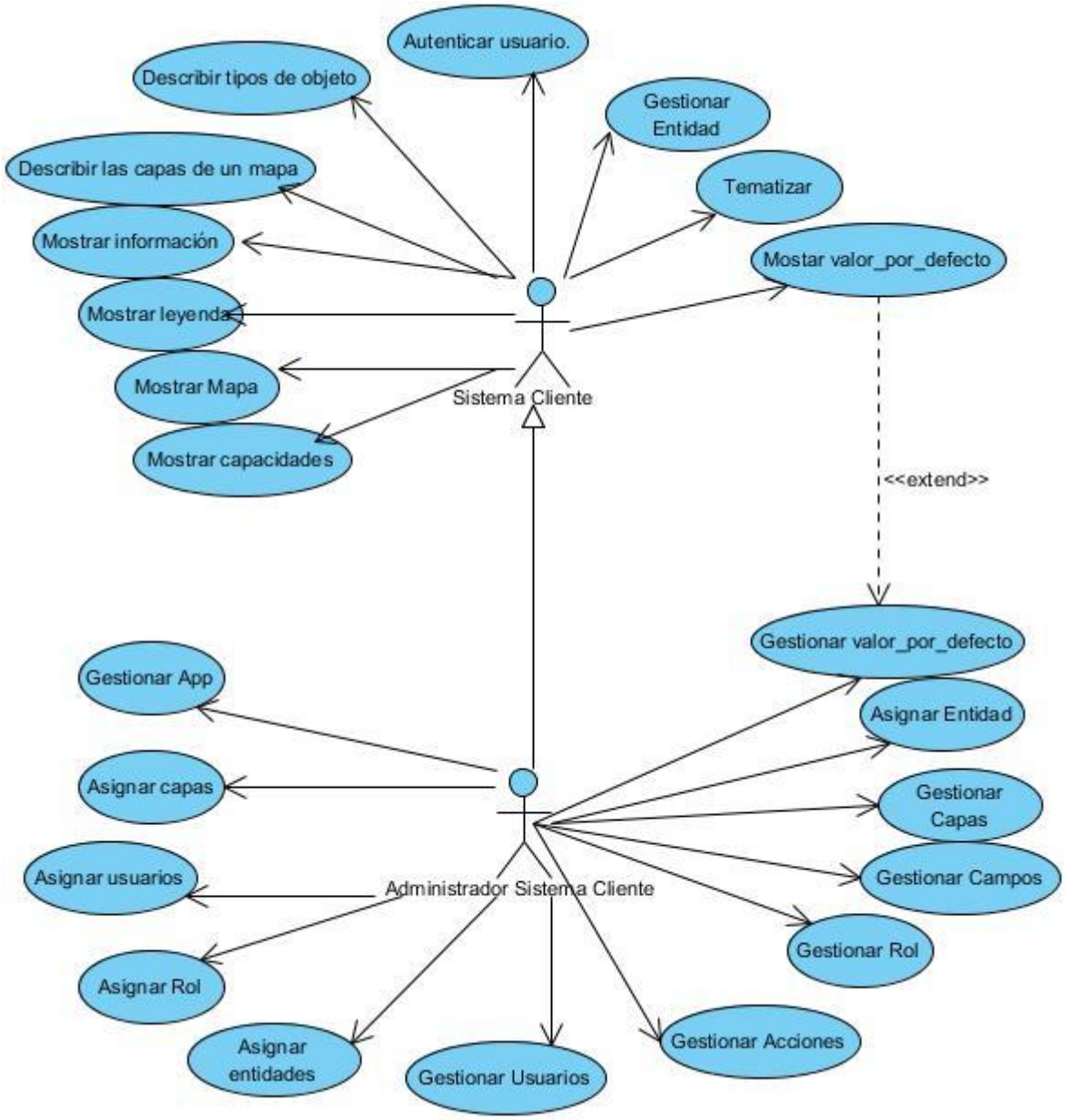


Figura 8. Diagrama de Casos de Uso del Sistema.

2.4.3 Descripción textual de los casos de uso del sistema.

En este epígrafe se describen detalladamente los CU presentes en el Diagrama de Casos de Uso del Sistema. Se situará el propósito general de los CU, el actor que le da inicio al mismo, así como, las precondiciones y las postcondiciones para su funcionamiento.

CU-Autenticar Usuario.

Caso de Uso:	Autenticar usuario
Actores.	Usuario (inicia)
Resumen:	El CU inicia cuando el Usuario desea autenticarse en el sistema, para ello debe introducir su usuario y contraseña y el servicio le permite realizar

	este proceso. El CU finaliza cuando el usuario queda autenticado.
Referencias	RF. 1
Prioridad	Crítica

Flujo normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El CU inicia cuando el actor solicita el servicio de autenticación, enviando los datos: • Usuario • Contraseña	2. El sistema comprueba que los datos son válidos. **3.** Genera y devuelve un token de acceso, para consumir los demás servicios y termina el caso de uso.
Flujos Alternos	
Flujo alterno	
	3. Devuelve el mensaje de Acceso denegado
Postcondiciones	**El usuario queda autenticado.**

Tabla 2. Descripción textual del CU Autenticar Usuario.

CU-Mostrar Mapa

Caso de Uso.	Mostrar mapa.
Actores.	Usuario (inicia).
Resumen.	El caso de uso inicia cuando el usuario solicita un mapa.
Referencias.	RF. 39**.**
Prioridad.	Crítica.

Flujo normal de eventos.	
1. El CU inicia cuando el actor solicita un mapa.	2. El sistema comprueba que el usuario tenga acceso al mapa solicitado y que la petición esté formulada correctamente. 3. El sistema devuelve el mapa solicitado en el formato de imagen solicitado y termina el caso de uso.
Flujos Alternos	
	1 El sistema devuelve el mensaje de error: Acceso Denegado.
	2 El sistema devuelve el mensaje de error: Petición Incorrecta.

Postcondición.	El sistema devuelve el mapa solicitado como una imagen.

Tabla 3. Descripción textual del CU Mostrar Mapa.

CU-Gestionar Entidad

Caso de Uso.	Gestionar entidad.
Actores.	Usuario (inicia).
Resumen.	El caso de uso inicia cuando el usuario solicita acceso a una entidad.
Referencias.	RF. 30, RF. 31, RF. 32, PF. 33**.**
Prioridad.	Crítica.
Flujo normal de eventos.	
Sección Insertar	
1. El CU inicia cuando el actor solicita insertar una entidad.	2. El sistema comprueba que el usuario tenga privilegios para realizar la acción y que la petición esté formulada correctamente. 3. El sistema inserta la entidad y termina el caso de uso.
Flujos Alternos	
	2 El sistema devuelve el mensaje de error: Acceso Denegado.
	3 El sistema devuelve el mensaje de error: Petición Incorrecta.
Postcondición.	El sistema inserta la entidad solicitada satisfactoriamente.
Sección Actualizar	
4. El CU inicia cuando el actor solicita actualizar una entidad.	5. El sistema comprueba que el usuario tenga privilegios para realizar la acción y que la petición esté formulada correctamente. 6. El sistema actualiza la entidad y termina el caso de uso
Flujos alternos	
	5- El sistema devuelve el mensaje de error: Acceso Denegado.
	6-El sistema devuelve el mensaje de error:

	Petición Incorrecta.
Postcondición.	El sistema actualiza la entidad solicitada satisfactoriamente.
Sección Eliminar	
7. El CU inicia cuando el actor solicita eliminar una entidad.	8. El sistema comprueba que el usuario tenga privilegios para realizar la acción y que la petición esté formulada correctamente. 9. El sistema elimina la entidad y termina el caso de uso
Flujos alternos	
	8- El sistema devuelve el mensaje de error: Acceso Denegado.
	9- El sistema devuelve el mensaje de error: Petición Incorrecta.
Postcondición.	El sistema elimina la entidad solicitada satisfactoriamente.
Sección Mostrar	
10. El CU inicia cuando el actor solicita mostrar una entidad.	11. El sistema comprueba que el usuario tenga privilegios para realizar la acción y que la petición esté formulada correctamente 12. El sistema muestra la entidad y termina el caso de uso.
Flujos alternos	
	11- El sistema devuelve el mensaje de error: Acceso Denegado.
	12- El sistema devuelve el mensaje de error: Petición Incorrecta.
Postcondición.	El sistema muestra la entidad solicitada satisfactoriamente.

Tabla 4. Descripción textual del CU Gestionar Entidad.

2.5 Conclusiones.

El Modelo de Dominio, permitió ofrecer una visión de los tipos más importantes de objetos que existen o los eventos que suceden en el entorno donde se desarrollarán los servicios del sistema. Constituyó además, la base para identificar los requisitos funcionales con los que debe cumplir la solución propuesta. Se determinaron las restricciones con las que deben cumplir los servicios del sistema gracias la definición los requisitos no funcionales. Los artefactos modelados durante el Análisis y el Diseño constituyen la base de la implementación exitosa del producto.

CAPÍTULO 3.

3.1 Introducción.

En el presente capítulo se realiza la construcción de la solución propuesta. Se representa la arquitectura de la misma así como los principales artefactos del diseño de la solución propuesta tales como: diagrama de clases de casos de uso, modelo de datos, modelo de despliegue y modelo de implementación entre otros. Se describen además los patrones de diseño utilizados en el desarrollo de la solución.

3.2 Arquitectura del sistema.

El patrón de arquitectura de las aplicaciones de software Modelo Vista Controlador (MVC), es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos. El patrón MVC se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página, el controlador es el Sistema de Gestión de Base de Datos y el modelo es el modelo de datos (38).

- **Modelo:** Encapsula los datos y las funcionalidades y es independiente de cualquier representación de salida y/o comportamiento de entrada.
- **Vista:** Muestra la información al usuario y obtiene los datos del modelo. Pueden existir múltiples vistas del modelo. Cada vista tiene asociado un componente controlador.
- **Controlador:** Reciben las entradas, usualmente como eventos que codifican los movimientos o pulsación de botones del ratón, pulsaciones de teclas, entre otros. Los eventos son traducidos a solicitudes de servicio para el modelo o la vista. El usuario interactúa con el sistema a través de los controladores (39).

Las Vistas y los Controladores conforman la interfaz de usuario. Un mecanismo de propagación de cambios asegura la consistencia entre la interfaz y el modelo. La separación del modelo de los componentes vista y del controlador permite tener múltiples vistas del mismo modelo. Si el usuario cambia el modelo a través del controlador de una vista, todas las otras vistas dependientes deben reflejar los cambios. Por lo tanto, el modelo notifica a todas las vistas siempre que sus datos cambien. Las vistas, en cambio, recuperan los nuevos datos del modelo y actualizan la información que muestran al usuario (39).

Ventajas del MVC:

- Clara separación entre interfaz, lógica de negocio y de presentación.
- Sencillez para crear distintas representaciones de los mismos datos.
- Reutilización de los componentes.

3.3 Patrones de diseño.

3.3.1 Patrones para signar responsabilidades (GRASP).

Los patrones GRASP describen los principios fundamentales del diseño para la asignación de responsabilidades. Constituyen un apoyo para la enseñanza que ayuda a entender el diseño de objeto esencial y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable. Estos patrones no compiten con los patrones de diseño sino que sirven de guía para encontrar los patrones de diseño (40)

Para la construcción del sistema propuesto se concretó utilizar los siguientes patrones GRASP:

Patrón Experto: Para determinar que clase debe asumir una responsabilidad a partir de la información que posee. Además permite conservar el encapsulamiento, ya que los objetos se valen de su propia información para realizar lo que se le oriente.

Ventajas:

- Se conserva el encapsulamiento de la información, debido a que los objetos utilizan su propia información para llevar a cabo las tareas. Normalmente, esto conlleva a un bajo acoplamiento, lo que da lugar a sistemas más robustos y más fáciles de mantener.
- Se distribuye el comportamiento entre las clases que contienen la información requerida, por lo tanto, se estimula las definiciones de clases más cohesivas y "ligeras" que son más posibles de entender y mantener (40)

En la solución se evidencia el uso de este patrón en la inclusión de Doctrine para mapear la base de datos. Este último, genera las clases para la gestión de la información de las tablas de dicha base de datos con las responsabilidades debidamente asignadas según el patrón Experto. Cada una de estas clases cuenta con un conjunto de funcionalidades que las convierte en expertas de la información de la tabla a la que representa. Además la aplicación de este patrón permite mover parte de la lógica del negocio hacia las clases modelo evitado así la sobrecarga del controlador.

Patrón Creador: Para guiar la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El diseño bien asignado permitirá soportar un bajo acoplamiento, una mayor claridad, el encapsulamiento y la reutilización. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento.

En Symfony la capa del controlador contiene el código que une la lógica del negocio con la presentación. Este cuenta con varios componentes entre los que se identifican las acciones que son los métodos que, utiliza el modelo. En la aplicación en cuestión, estas acciones se

ejecutan en las clases “ActionController”, “AppController” y “EntityController” entre otros, de los módulos “Action”, “App” y “Entity” respectivamente. En las acciones mencionadas se crean los objetos que representan las entidades evidenciando que son “creadoras” de dichas clases.

Patrón Alta cohesión: Este patrón define que una clase tiene responsabilidades moderadas en un área funcional y colabora con otras clases para llevar a cabo las tareas. Una clase con alta cohesión tienen un número relativamente pequeño de métodos, con funcionalidad altamente relacionada y no realiza mucho trabajo. Colabora con otros objetos para compartir el esfuerzo si la tarea es extensa. Además es ventajosa porque es relativamente cómodo de mantener, entender y reutilizar.

Ventajas:

- Se incrementa la claridad y facilita la comprensión del diseño.
- Se simplifican el mantenimiento y las mejoras (40)

Una de las características principales del Framework Symfony es la organización del trabajo en cuanto a la estructura del proyecto, lo cual permite crear y trabajar con clases con una alta cohesión posibilitando que a estas clases le sean asignadas responsabilidades estrechamente relacionadas, para que realicen un trabajo excesivo. Por ejemplo, las clases “ActionController”, “AppController” y “EntityController” contienen responsabilidades estrechamente relacionadas, encargadas de controlar las acciones de sus respectivas plantillas. Esto hace posible que el software sea flexible a cambios sustanciales con efecto mínimo.

Patrón Bajo acoplamiento: Este patrón asigna una responsabilidad para mantener el bajo acoplamiento que no es más que tratar de que una clase no dependa de muchas otras, así esa clase no tendrá muchas dependencias, proporcionando la reutilización. Este patrón no puede verse separado del patrón Experto y del patrón Alta Cohesión.

Ventajas:

- No afectan los cambios en otros componentes.
- Fácil de entender de manera aislada.
- Conveniente para reutilizar (40)

El patrón Bajo Acoplamiento determina que una clase no dependa de muchas otras clases, lo que posibilita que no se afecten por cambios en otros componentes. Un ejemplo de cómo Symfony implementa este patrón es que las clases “ActionController”, “AppController” y “EntityController” heredan solamente de Controller para lograr un bajo acoplamiento de clases.

Patrón Controlador: Para manejar y controlar los eventos del sistema. Con la utilización de este patrón se logra separar la lógica de negocio de la capa de presentación. De esta

manera se consigue un mayor control sobre el sistema y se favorece la reutilización de código.

Las clases “ActionController”, “AppController” y “EntityController” son clases controladoras del sistema y su función es servir de enlace entre la vista y el modelo. Precisamente es esta la evidencia de la existencia del patrón Controlador en el proyecto, aunque es común que este se implemente asignando el control a una sola clase, de este modo se asegura que no se sobrecargue el controlador. Symfony implementa además el patrón *Front Controller* (Controlador Frontal), en el cual existen varias clases controladoras que forman un flujo para atender las peticiones del usuario y de otras clases.

Patrón No Hables con Extraños: Se asigna la responsabilidad a un objeto directo del cliente para que colabore con un objeto indirecto, de modo que el cliente no necesite saber nada del objeto indirecto.

3.3.2 Patrones GOF (Gang Of Four, en Inglés).

Patrón Fachada: Patrón estructural que trata de simplificar la interfaz entre dos sistemas o componentes de software ocultando un sistema complejo detrás de una clase que funciona como pantalla o fachada. La idea principal es la de ocultar todo lo posible la complejidad de un sistema, el conjunto de clases o componentes que lo definen, de forma que solo se ofrezca un punto de entrada al sistema cubierto por la fachada. En la solución en cuestión se aprecia la entrada al sistema en app.php. Su uso aísla los posibles cambios que se puedan producir en alguna de las partes.

Patrón Cadena de Responsabilidad: La cadena de responsabilidad se encarga de evitar el acoplamiento del remitente de una petición a su receptor, dando a más de un objeto la posibilidad de manejar la petición.

3.4 Modelo de Diseño.

En el modelo de diseño se describe la realización física de los casos de uso centrándose en cómo los requisitos funcionales y no funcionales, junto con las restricciones relacionadas con el entorno de implementación, tienen un impacto en el sistema a considerar.

3.4.1 Diagrama de clases del Diseño.

El Diagrama de Clases del Diseño es uno de los artefactos que genera el Modelo de Diseño, en el cual se muestran un conjunto de clases, interfaces y colaboraciones, así como las relaciones entre sí. Se realiza un diagrama por caso de uso identificado. Estos diagramas son importantes pues visualizan, especifican y documentan los modelos estructurales, lográndose una muestra amplia y confiable del sistema previo a su implementación.

Diagrama de Clases del Diseño del Caso de Uso Gestionar Usuario.

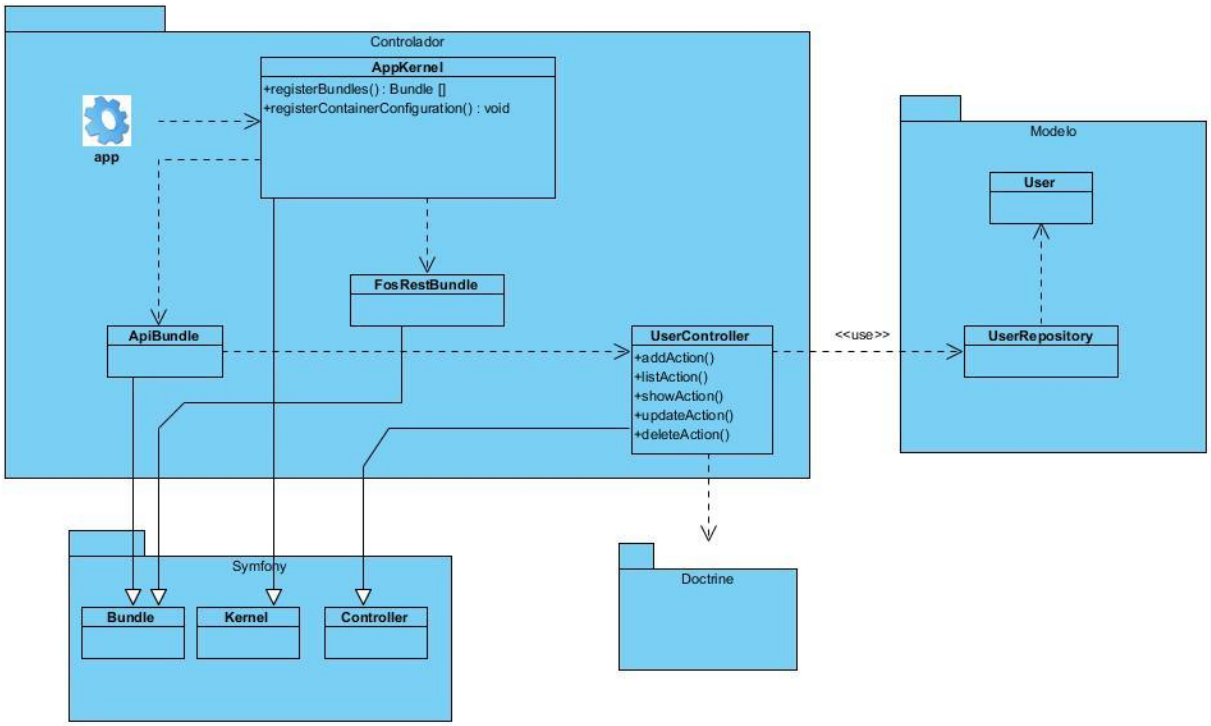


Figura 9. Diagrama de clases del Diseño del Caso de Uso Gestionar Usuario.

En la Figura 9 se muestra como el usuario accede al sistema a través de la página servidora "app", Las clases dentro del paquete Controlador se encuentran "FosRestBundle", "ApiBundle" y "UserController". "FosRestBundle", hereda de la clase "Bundle", propia de Symfony y en ella se encuentran implementadas varias herramientas para brindar los servicios siguiendo el estándar REST. "ApiBundle", contiene las clases propias de la solución en cuestión y es quien realiza una instancia a "UserController", clase en la que se implementan las acciones:

- agregar usuario (***addAction()***).
- listar usuarios (***listAction()***).
- mostrar usuario (***showAction()***).
- actualizar usuario (***updateAction()***).
- eliminar usuario (***deleteAction()***).

Estas acciones se llevan a cabo según las peticiones del usuario. En las clases del paquete modelo se implementan las acciones relacionadas con la gestión de la información física en la base de datos

Diagrama de Clases del Diseño del Caso de Uso Mostrar Mapa.

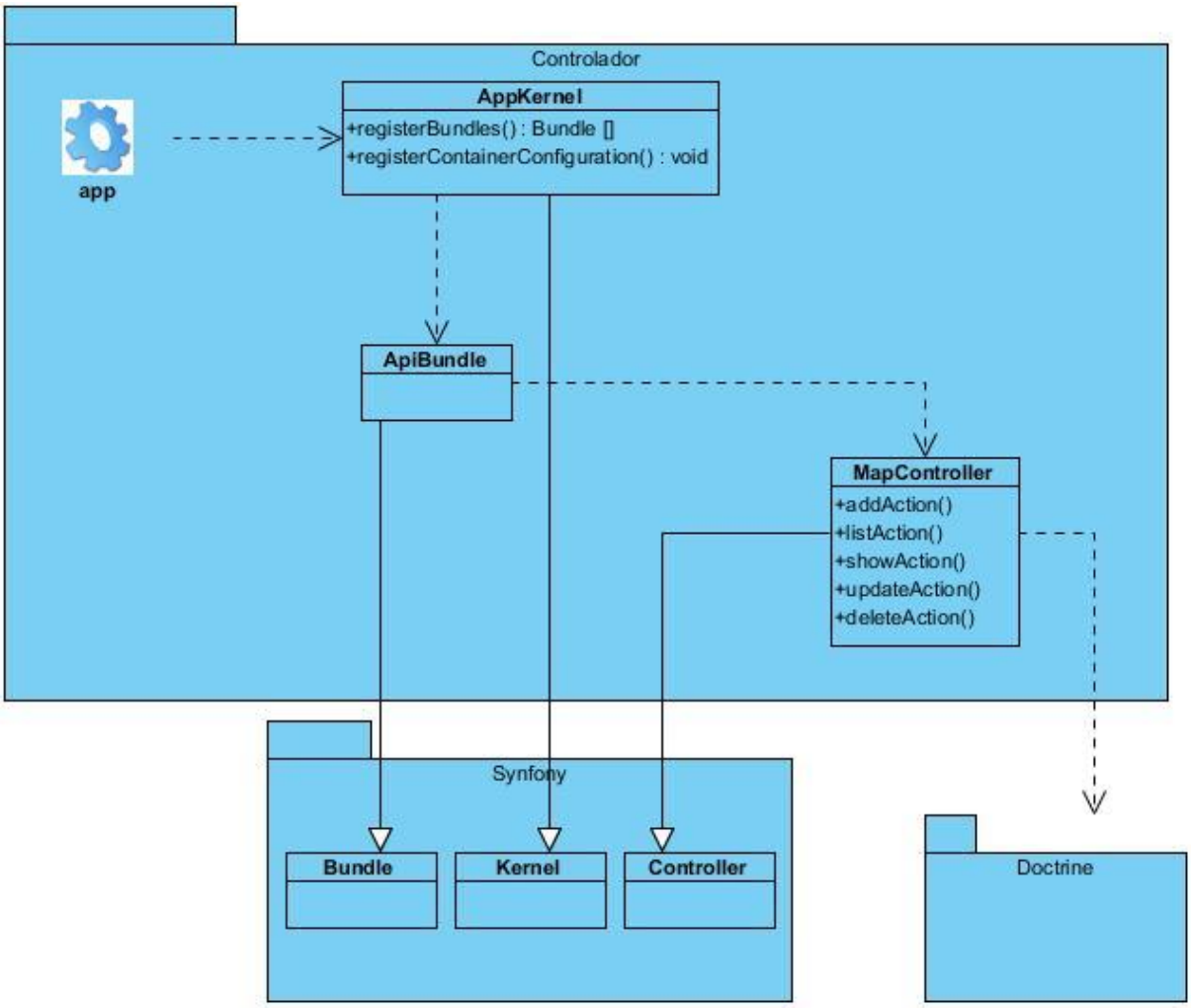


Figura 10. Diagrama de clases del diseño del Caso de Uso Mostrar Mapa.

En el CU Mostrar Mapa el usuario accede al sistema a través de la pagina servidora “app”, envía la petición de un mapa y el “AppKernel”, haciendo una instancia a la clase “ApiBundle”, inicializa el “MapController” que es el encargado de realizar la acción solicitada por el usuario.

Diagrama de Clases del Diseño del Caso de Uso Gestionar Entidad.

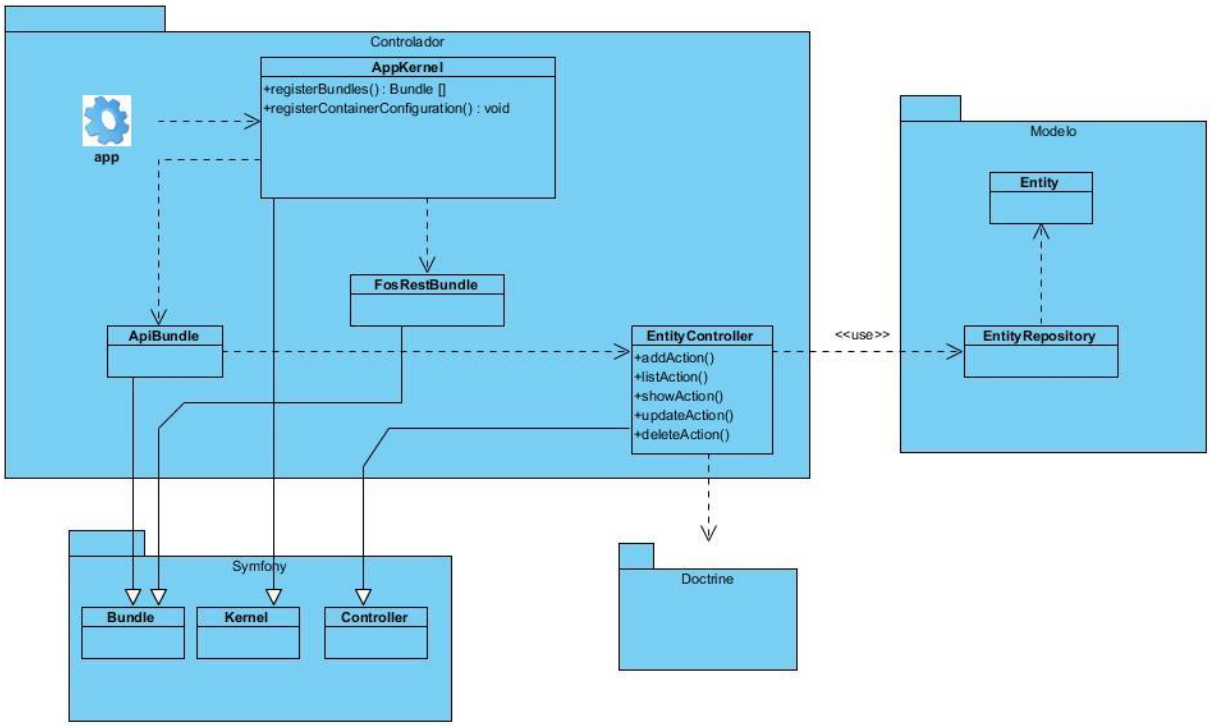


Figura 11. Diagrama de clases del diseño del Caso de Uso Gestionar Entidad.

En la Figura 11 se muestra como el usuario gestiona las entidades. Entre las clases dentro del paquete Controlador se encuentran “FosRestBundle”, “ApiBundle” y “EntityController”. “FosRestBundle”, hereda de la clase “Bundle”, propia de Symfony y en ella se encuentran implementadas varias herramientas para brindar los servicios siguiendo el estándar REST. “ApiBundle”, contiene las clases propias de la solución en cuestión y es quien realiza una instancia a “UserController”, clase en la que se implementan las acciones propias para la gestión de las entidades:

- agregar entidad (***addAction()***).
- listar entidad (***listAction()***).
- mostrar entidad (***showAction()***).
- actualizar entidad (***updateAction()***).
- eliminar entidad (***deleteAction()***).

3.5 Modelo de Datos.

Un Modelo de Datos se puede definir como un conjunto de conceptos, reglas y convenciones que permiten aplicar una serie de abstracciones a fin de describir y manipular los datos que se desean almacenar en la base de datos. A continuación se presentan el modelo entidad-relación de la bases de datos utilizada para el manejo de información.

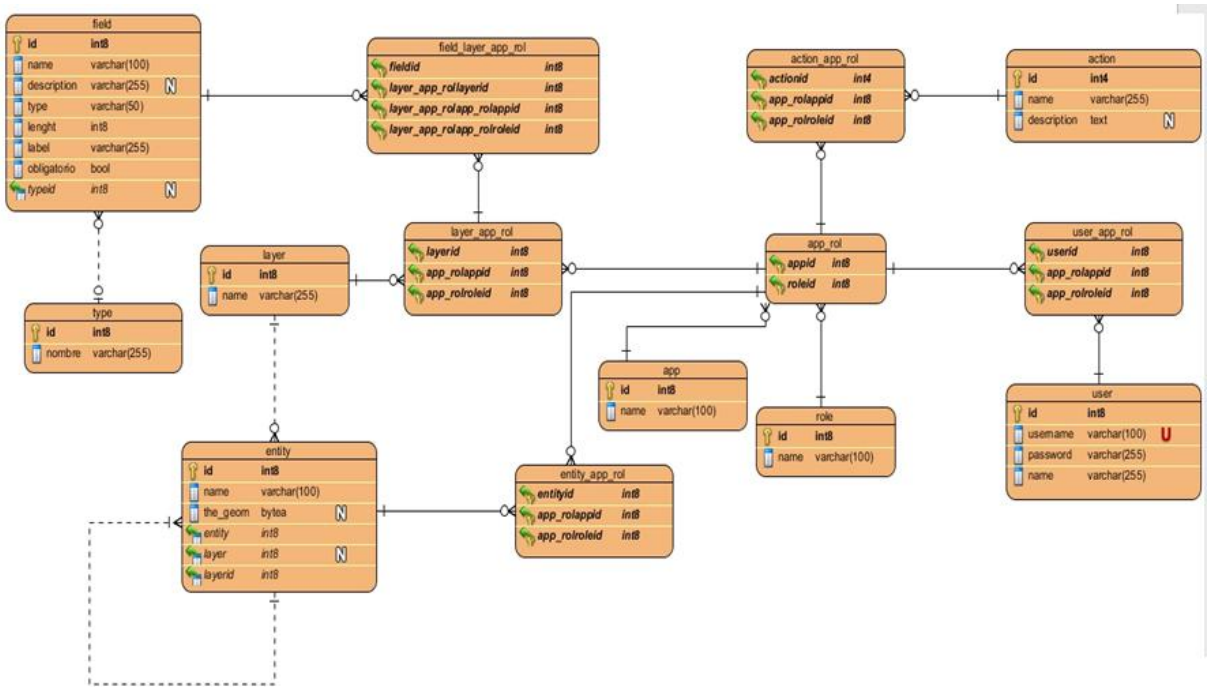


Figura 12. Modelo de Datos.

La solución en cuestión intenta resolver dos problemas fundamentales: brindar servicios de mapas para cualquier aplicación cliente y gestionar la seguridad de dichos servicios.

Para tratar el problema de la seguridad se emplean las siguientes entidades del Modelo de Datos previamente obtenido (Ver Figura 12.):

- **app:** en esta tabla se guarda la información de las aplicaciones clientes (identificador y nombre).
- **user:** es la encargada de almacenar la información de los usuarios (identificador, nombre de usuario, contraseña y nombre).
- **role:** se encarga se guardar el role que puede asumir el usuario.
- **action:** almacena las acciones que puede realizar cada role de la aplicación (app).
- **app_role:** guarda la relación entre: “role” y “app”, es decir: que roles tiene una aplicación y que aplicaciones tiene un role.
- **user_app_role**: guarda la relación entre: “app_role” y “user”, es decir: el role que posee un usuario en una aplicación.
- **action_app_role**: es la relación entre: “app_role” y “action”, esta entidad guarda la información referente a que acciones posee un role en una aplicación.

Para bridar los servicios de mapas se utilizan las siguientes entidades del Modelo de Datos:

- **layer**: esta entidad guarda la información de las diferentes capas de un mapa (identificador y nombre).
- **entity**: en esta entidad se almacena la información de las entidades del mapa [identificador, nombre, datos geográficos o posición en el mapa (the_geom), nombre

de capa a la que pertenece (*layer*), identificador de la capa a la que pertenece (*layerid*)]

- **field**: esta entidad almacena la información de los campos (identificador, nombre, descripción, tipo, longitud, identificador de tipo).
- **tipo**: esta entidad almacena la información sobre los tipos de campos (identificador y nombre).
- **layer_app_rol**: almacena la relación entre: “app_role” y “layer”, es decir: a que capas tiene acceso un role en una aplicación.
- **field_layer_app_role**: esta entidad guarda la relación entre: “layer_app_role” y “field”, es decir, qué campos tiene la capa a la que accede un role en una aplicación.
- **entity_app_role**: en esta entidad se almacena la relación entre: “app_role” y “entity”, es decir a que entidades tiene acceso un role en una aplicación.

3.6 Modelo de Despliegue.

En el Modelo de Despliegue se muestra la configuración de los nodos de procesamiento en tiempo de ejecución, los cuales representan objetos físicos con recursos computacionales. Se utiliza para capturar los elementos de configuración del procesamiento y las conexiones entre esos elementos. Estas conexiones se etiquetan con un estereotipo que expresan el tipo de conector o protocolo de comunicación o de red utilizado.

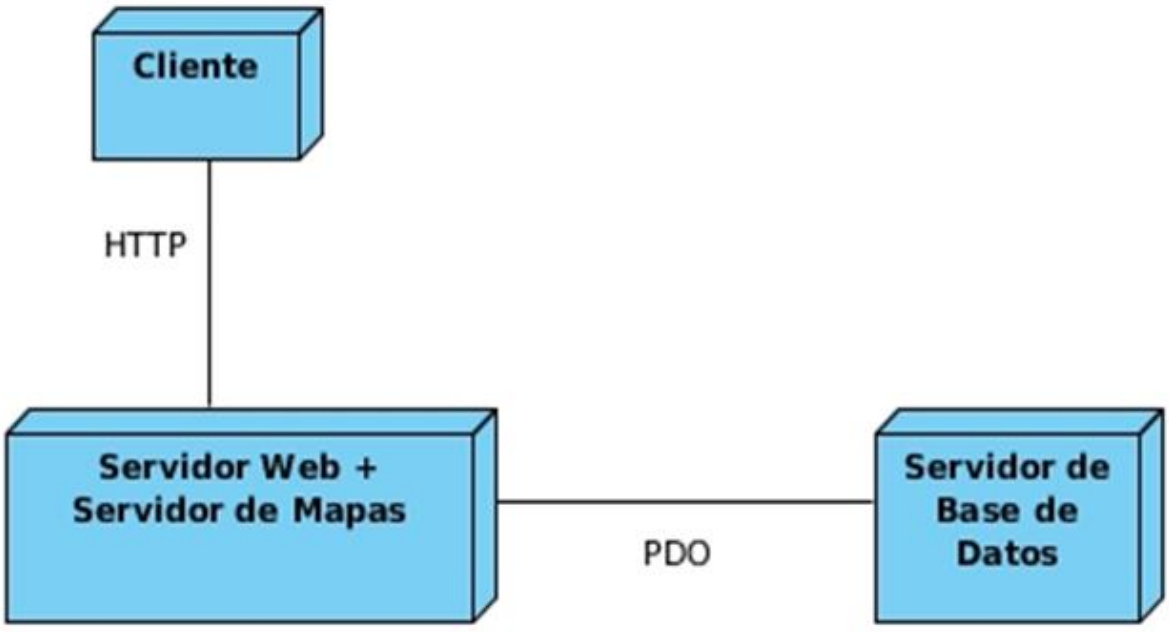


Figura 13. Diagrama de Despliegue.

El Diagrama de Despliegue está compuesto por los nodos necesarios para obtener la información necesaria brindada mediante los SW de las funcionalidades de gestión del API para brindar servicios de mapas. Para ello, cuenta con un servidor Web Apache y un servidor de Mapas, que están conectados a un servidor de base de datos PostgreSQL con

extensión PostGIS, mediante el protocolo PDO (del inglés Portable Distributed Objects), donde su principal función es, la de realizar la solicitud de los datos a la base de datos. Los usuarios podrán acceder a estos servicios mediante una computadora cliente que le permitirá el consumo de los mismos, implementados utilizando el protocolo REST, que va a permitir la conexión al servidor web a través del protocolo HTTP.

3.7 Modelo de implementación.

El modelo de implementación describe como los elementos del modelo de diseño se implementan en términos de componentes. Describe también cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados y cómo dependen los componentes unos de otros (41)

3.7.1 Diagrama de Componentes .

Los Diagramas de Componentes muestran un conjunto de elementos del modelo de implementación tales como: componentes, subsistemas de implementación y sus relaciones. Muestran la organización y las dependencias lógicas entre un conjunto de componentes de software, sean éstos: componentes de código fuente, librerías, binarios o ejecutables. Los SW implementados para las funcionalidades de gestión del sistema propuesto, se encuentran agrupados en paquetes que ayudan a un mejor entendimiento de la estructura de los mismos.

A continuación se muestra un diagrama de componentes genérico para el sistema (Ver Figura 14.).

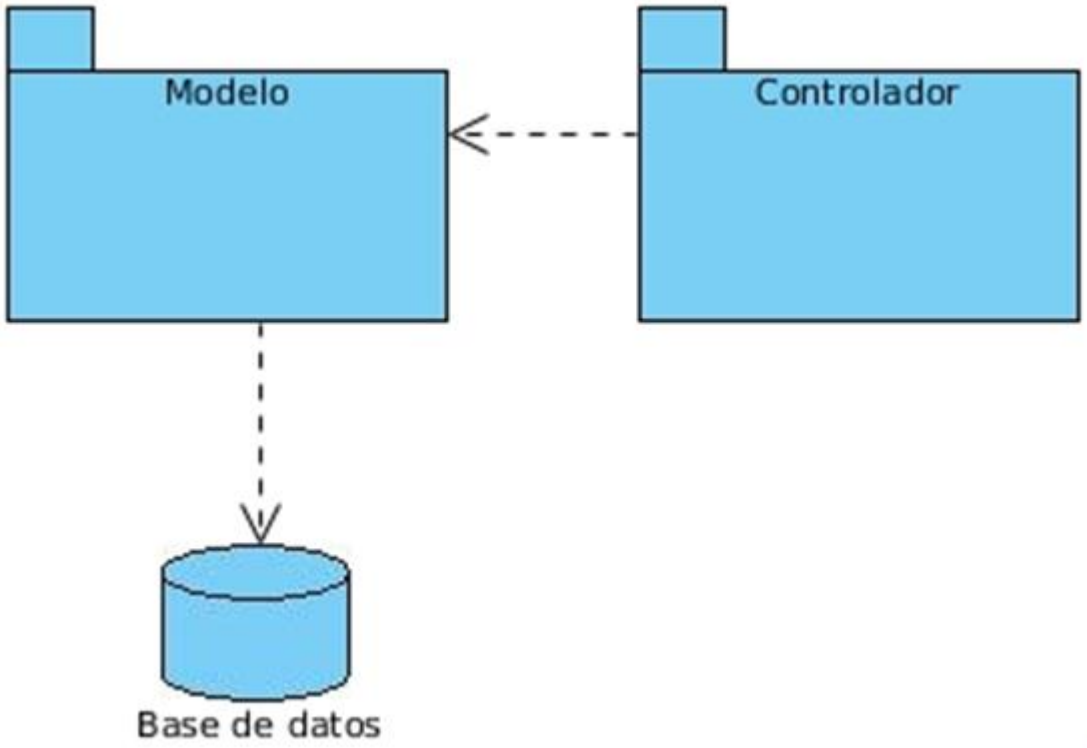


Figura 14.Diagrama de Componentes.

Diagrama de componentes del Caso de Uso Gestionar Usuario.

Para una mejor comprensión del diagrama se expone una vista detallada de los paquetes Modelo y Controlador, así como los componentes que contiene cada paquete para el CU Gestionar Usuario.

Vista Detallada: Paquete Modelo.

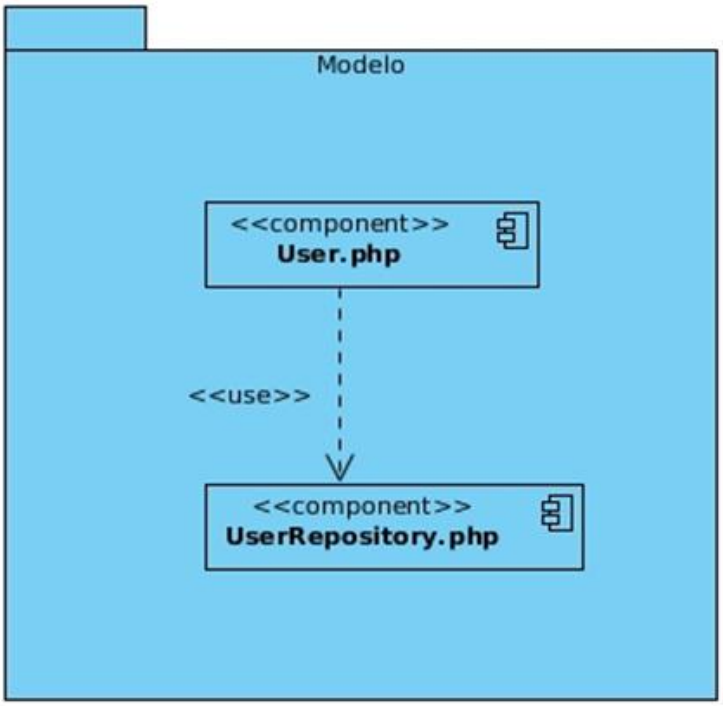


Figura 15. Vista detallada del paquete Modelo Gestionar Usuario.

Vista Detallada: Paquete Controlador.

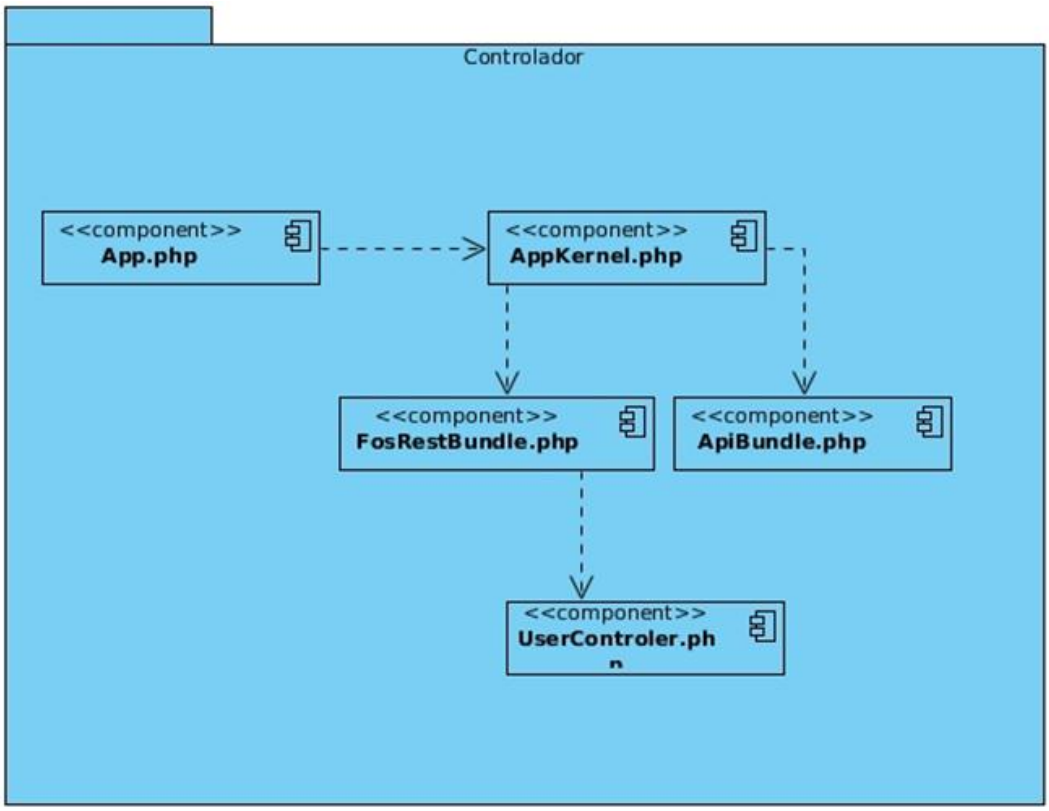


Figura 16. Vista detallada del paquete Controlador Gestionar Usuario.

Diagrama de Componentes del Caso de Uso Mostrar Mapa.

A continuación se muestran la vista detallada del paquete Controlador del CU Mostrar Mapa.

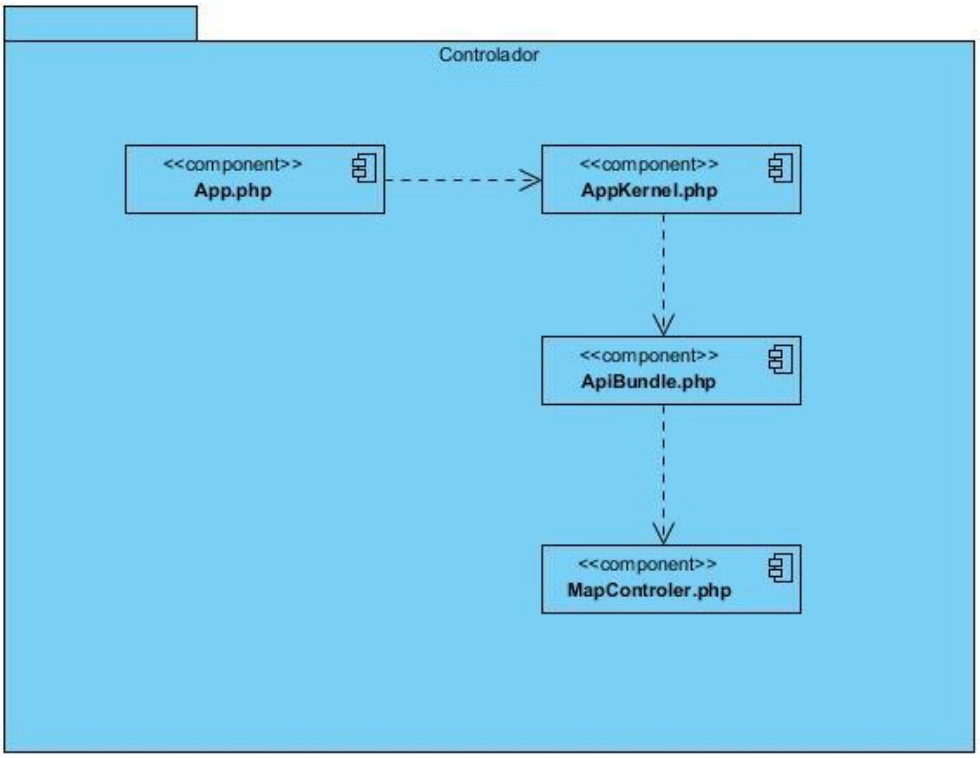


Figura 17. Vista detallada del paquete Controlador Mostrar Mapa.

Diagrama de Componentes del Caso de Uso Gestionar Entidad.

Para una mejor comprensión del diagrama se expone una vista detallada de los paquetes Modelo y Controlador, así como los componentes que contiene cada paquete para el CU Gestionar Entidad.

Vista Detallada: Paquete Modelo Gestionar Entidad.

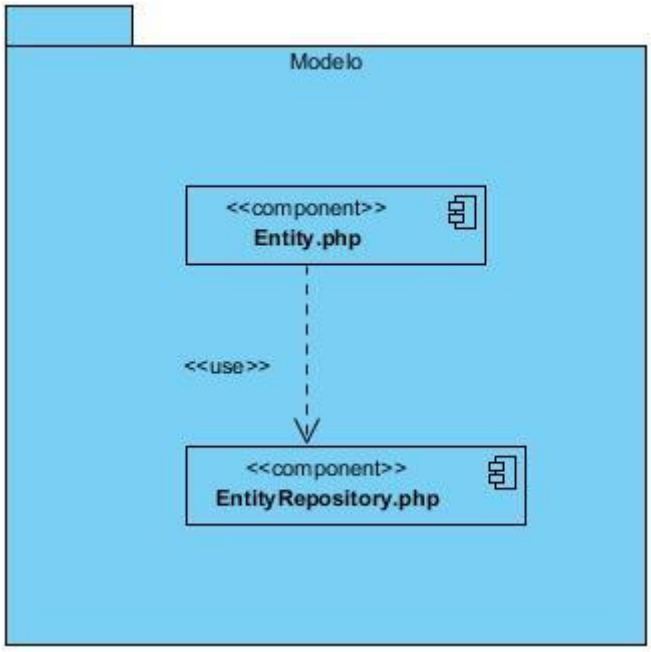


Figura 18. Vista detallada del paquete Modelo Gestionar Entidad.

Vista Detallada: Paquete Controlador Gestionar Entidad.

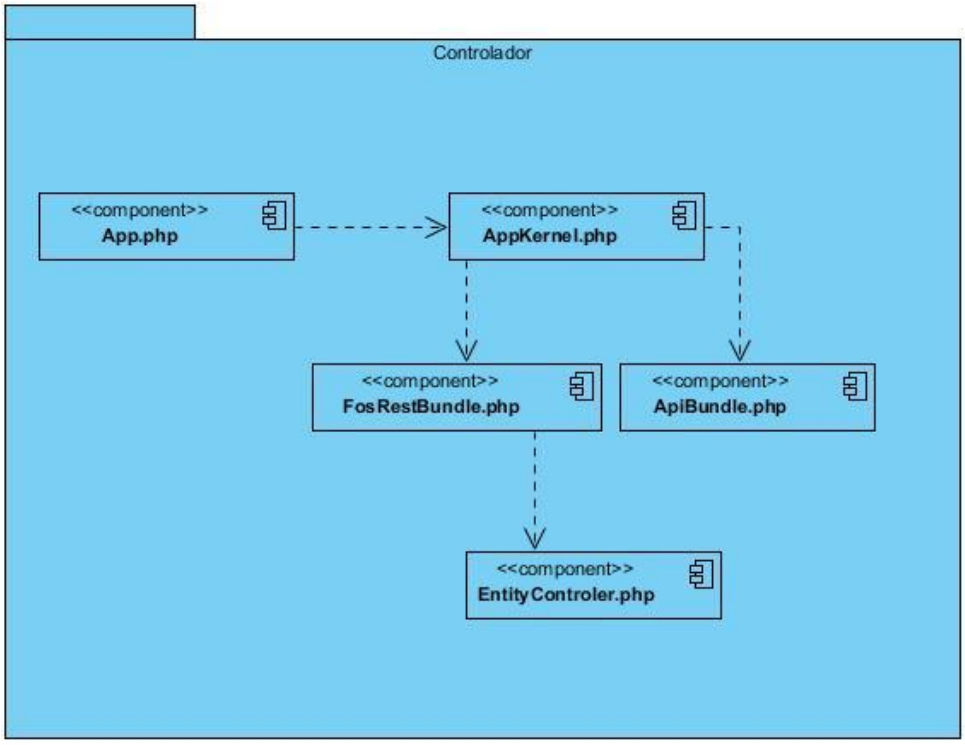


Figura 19. Vista detallada del paquete Controlador Gestionar Entidad.

3.8 Pruebas.

3.8.1 Tipos de Pruebas.

Para la comprobación del correcto funcionamiento del sistema, se seleccionó dentro de los niveles de prueba el de Pruebas Funcionales, ya que estas se centran en las funciones, las entradas y salidas. Su objetivo fundamental es asegurar el trabajo apropiado de los requisitos funcionales, incluyendo la entrada de datos, procesamiento y obtención de resultados.

Según Pressman, estas pruebas se realizan aplicando las pruebas de caja negra ejecutando cada caso de uso, flujo de caso de uso, o función, usando datos válidos e inválidos. De acuerdo con Pressman, las pruebas funcionales pueden realizarse en base a dos enfoques principales:

- **Prueba de Caja Blanca:** En la prueba de caja blanca, se analiza la estructura lógica del programa y para cada alternativa que pueda presentarse, los datos de prueba ideados conducirán a ella.
- **Prueba de Caja Negra:** En la prueba de la caja negra, los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta.

Método de prueba.

El método de caja negra se refiere a las pruebas que se llevan a cabo sobre la interfaz del software. Estos pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene.

Para realizar las pruebas se elaboran las Descripciones de Casos de Prueba (DCP) que ayudan a que los probadores se cercioren del buen funcionamiento de los servicios implementados.

3.8.2 Diseño de Casos de Prueba de Caja Negra.

El Método de la Caja Negra se centra en los requisitos fundamentales del software y permite obtener entradas que prueben todos los requisitos funcionales del programa.

Con este tipo de pruebas se intenta encontrar:

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a las bases de datos externas.
- Errores de rendimiento.
- Errores de inicialización y terminación.

Métodos empleados en las pruebas de caja negra.

- Partición Equivalente.

- Análisis de valores límite (AVL).
- Prueba de la tabla ortogonal.
- Adivinando el error.

Dentro de estos métodos se eligió el método **Partición Equivalente** y **Análisis de Valores Límite**.

Partición Equivalente: Según Pressman, Partición Equivalente es el método de prueba de caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar.

Análisis de valores límite (AVL): De acuerdo con Pressman, el Análisis de Valores Límite es una técnica de diseño de casos de prueba que completa a la partición equivalente. En lugar de seleccionar cualquier elemento de una clase de equivalencia, el AVL lleva a la elección de casos de prueba en los extremos de la clase. En lugar de centrarse solamente en las condiciones de entrada, el AVL obtiene casos de prueba también para el campo de salida

3.8.3 Diseño de pruebas.

Un caso de prueba se diseña según las funcionalidades descritas en los casos de usos. Este diseño se elabora previo a la realización de las pruebas exploratorias, o de interfaz como también pueden llamarse, se parte de la descripción de los casos de usos del sistema, como apoyo para las revisiones. Cada planilla de caso de prueba recoge la especificación de un caso de uso, dividido en secciones y escenarios, detallando las funcionalidades descritas en él y describiendo cada variable que recoge el caso de uso en cuestión además quedan plasmadas las revisiones realizadas al caso de prueba; así como un registro de todo aquello que no satisface y corresponde a la calidad del software.

En el proceso de prueba en cuestión se hace uso del método Caja Negra. Para detallar el caso de uso se utiliza una tabla, donde se desglosa esta funcionalidad en secciones y a su vez estas en escenarios, para hacer más fructífera la ejecución de las pruebas. Esta tabla contiene los campos:

- **Nombre de la sección:** Se especifica el nombre de la sección [SC 1: Nombre de la sección].
- **Escenarios de la sección:** Se especifican los escenarios de cada sección [EC 1.1: Nombre del Escenario].
- **Descripción de la funcionalidad:** Se describe brevemente la funcionalidad del escenario.

Diseño de Casos de prueba del Caso de Uso Autenticar Usuario.

El siguiente es un ejemplo donde se detalla el caso de uso **Autenticar Usuario** del sistema en cuestión.

- **Nombre del CU:** Autenticar Usuario.
- **Descripción general:** El CU inicia cuando el Usuario solicita el servicio de autenticación enviando los datos: Usuario y Contraseña. El CU finaliza cuando el usuario queda autenticado.
- **Condiciones de ejecución:** El sistema debe permitir autenticar un usuario.

Secciones a probar en el caso de uso.

- Autenticar usuario.

Nombre de la sección.	**Escenarios de la sección.**	**Descripción de la funcionalidad**	**Flujo central**
SC 1: Autenticar Usuario.	EC 1. Autenticar usuario de forma correcta.	Esta funcionalidad permite validar las credenciales de un usuario brindándole o no acceso a los servicios del sistema.	• El sistema comprueba que los datos son válidos. • Genera y devuelve un token de acceso, para consumir los demás servicios y termina el caso de uso.
	EC 1.2.Autenticar usuario de forma incorrecta.	El sistema no autentica al usuario debido a que hay un error en los datos insertados o se han dejado campos vacios.	• El sistema detecta que los datos insertados no son validos o que alguno de los campos está vacío. • Devuelve el mensaje de Acceso denegado.

Tabla 5. Diseño del Caso de Prueba de Caja Negra del Caso de Uso: Autenticar Usuario.

A partir de esta descripción se detallan las variables que se encuentran en toda la interfaz asociadas al caso de uso que se le estaba diseñando el caso de prueba. Esta descripción está compuesta por campos tales como:

- **No.**: Se enumera todos los campos o variable, descrito en el caso de uso.
- **Nombre de campo**: Se especifica el nombre del campo de entrada.
- **Clasificación**: Se especifica la clasificación del componente [ejemplo: campo de texto, lista de selección, entre otros].
- **Puede ser nulo**: Se especifica si el campo puede ser nulo o no, para ello solo se puso SI o NO.

- **Descripción**: Se describen brevemente los datos que debían introducirse.

La siguiente tabla muestra un ejemplo de la descripción de variables del caso de uso Gestionar Usuario del Sistema en cuestión.

NO.	Nombre del campo	Clasificación.	Valor nulo.	Descripción.
1	DatoUsuario	Campo de Texto	NO	Debe introducir el nombre de usuario.
2	DatoPassword	Campo de Texto	NO	Debe introducir la contraseña del usuario.

Tabla 6. Descripción de Variables del Caso de Uso Autenticar Usuario.

Esta descripción posibilita que se ejecute una matriz de datos, donde se evalúa y se prueba la validez de cada uno de los datos introducidos en el sistema, específicamente en la sección que se estará probando. Utilizando un juego de datos válidos e inválidos se identificará el empleo de la técnica de partición de equivalencia y análisis de valores límites.

La matriz de datos contiene los siguientes aspectos:

- **Escenario**: Se especifica el nombre del escenario.
- **Variables** [1, 2,…, n]: Se especifica el nombre de la variable, o el número según la tabla de descripción de variables y en su celda correspondiente se indicó el valor del dato [V (Válido), I (Inválido), N/A (No Aplica)].
- **Respuesta del sistema**: Se escribe el resultado que se esperaba al realizar la prueba.
- **Resultado de la prueba**: Se escribe el resultado que se obtuvo al realizar la prueba. (Satisfactorio o No Satisfactorio).
- **Flujo central**: Se expone el flujo central del caso de uso al que se le estaba diseñando el caso de prueba.

A continuación, un ejemplo de la matriz de datos del caso de uso Autenticar Usuario del Sistema en cuestión.

Matriz de Datos

SC 1. Autenticar Usuario.

ID del escenario	Escenario	V1 DatoUsuario.	V2 DatoPassword.	Respuesta del sistema	Resultado de la prueba
EC 1.1	Autenticar usuario de	V/rayner	V/rayner1		Satisfactoria.

	forma correcta.				
EC 1.2	Autenticar usuario de forma incorrecta.	V/rayne	V/rayner1	Devuelve mensaje de error	Satisfactoria
		V/rayner	V/rainel	Devuelve mensaje de error	Satisfactoria

Tabla 7. Matriz de Datos Autenticar Usuario.

3.9 Conclusiones parciales.

En este capítulo se realizó la construcción de la solución propuesta para el API para brindar servicios de mapas en la LPS Aplicativos SIG, donde se desarrollaron los principales artefactos a llevar a cabo en el proceso de diseño e implementación para la construcción del sistema. Mediante la realización del diagrama de clases se obtuvo una muestra amplia y confiable del sistema previo a su implementación, la construcción del Modelo de Despliegue, describe cómo se encuentra distribuida física y lógicamente la arquitectura del sistema.

La elaboración del Modelo de componentes, permitió especificar los componentes establecidos para el desarrollo de la aplicación y la relación entre ellos. Mediante los casos de prueba se validó el software, verificando el correcto funcionamiento del sistema, comprobándose que no existían errores en las funciones operativas.

.

CONCLUSIONES GENERALES.

Luego de concluido todo el proceso de construcción del sistema propuesto: API para brindar servicios de mapas dentro de la LPS Aplicativos SIG, se arriban a las siguientes conclusiones:

- Según el problema de la investigación unido a los requisitos de la Universidad para el desarrollo de la presente investigación, resulta una necesidad la implementación de una API puesto que las soluciones estudiadas no cumplen las condiciones antes mencionadas.
- La correcta selección de la metodología de desarrollo, el lenguaje de modelado y la herramienta CASE utilizadas propició la correcta representación del sistema a través de los diferentes modelos.
- El sistema desarrollado brinda servicios de mapas orientados a la Web de forma centralizada y cumplen con los estándares establecidos por el OGC.
- Luego de la aplicación de las pruebas de caja negra se detectaron y posteriormente resolvieron las distintas no conformidades encontradas en el proceso, dando a lugar un sistema libre de no conformidades.

RECOMENDACIONES.

- Se recomienda integrar la presente solución con el producto Catálogo de Mapas desarrollado por el Departamento Geoinformática.
- Se recomienda utilizar la API para la realización de futuros proyectos dentro de la LPS Aplicativos SIG.
- Se recomienda realizar el manual de usuario para la solución presentada.

REFERENCIAS BIBLIOGRÁFICAS.

1. GRACIA JOAQUIN. <<IngenieroSoftware>>. [Consultado el: 15 de Diciembre de 2012]. Disponible en: http://www. Ingenierosoftware.com/analisisydiseno/casosdeuso.php.

2. BELMONTE, S. and NUÑEZ.V. Desarrollo de modelos hidrológicos con herramientas SIG. 2009, [En Línea]. Disponible en: http://www.geofocus.rediris.es/2006/Informe2_2006.pdf.

3. Historia de los SIG. [En Línea]. Consultado el: 15 de Diciembre de 2012]. Disponible en: http://svn.osgeo.org/osgeo/book/es/Fundamentos/Historia/Historia.tex.

4. CONSORTIUM.OPEN.GOSPATIAL. Web map server implementation specification. 2006, [En Línea]. Consultado el:]. Disponible en: Disponible en: http://opengeospatial.org.

5. CARLOS.E.ARCANCEL. Web Map Services:. Revista Ciencia e Innovación Tecnológica Número 126. 2008, p.

6. BOLSTAD.P. Gis Fundamentals. A first text on Geographic Information Systems, Second Edition. White Bear Lake. 2005, p.

7. PUMAIN.DENISE. HyperGeo, [En Línea]. Citado en: 6 de Diciembre de 2012]. Disponible en: http://www.hypergeo.eu/article.php3?id_article=265.

8. FRANCIS.P and DOMINIQUE.P. La alquimia de las multitudes, [En Línea]. Paidos Ibérica, SA, 2009. 978-84-493-2196-2. Disponible en: https://docs.google.com/gview?url=http://www.elboomeran.com/upload/ficheros/obras/primer_cap_alquimia.pdf&chrome=true.

9. ARGUIÑA.J and MIRALLES.M. Fundamentos de Telemática., [En Línea]. Valencia. Universidad Politécnica de Valencia, 2005., Consultado el: 15 de Diciembre de 2012.]. Disponible en: http://books.google.com.cu/books?id=zq1r6ed9NlYC&pg=PP5&lpg=PP5&dq=fundamentos+de+telem%C3%A1tica+jorge+l%C3%A1zaro+laporta+marcel+miralles+agui%C3%B1iga&source=bl&ots=mpVtDp7-16&sig=utjj_ngY2cucwZrCScRIo6jAaYk&hl=es&sa=X&ei=idNLUdvcJcXh4AOlgoHoAw&ved=0CCoQ6AEwAA.

10. PUENTES, S. Facultad de Biología, Universidad de La Habana. 2005. Disponible en: http://fbio.uh.cu/sites/bioinfo/glosario.html.

11. GARCIA, I. and MUNILLA.E E-business Colaborativo, Fundación Confemetal. ISBN 84-95428-98-9..

12. DEWIT, O. *Asp.net Programación Web con Visual Studio y Web Matrix*, s.l.Ediciones ENI, 2003.

13. IBM.GLOBAL.SERVICES. Providing prevemptive protection for critical enterprise servers, [En Línea]. Consultado el: 18 de Enero de 2013]. Disponible en: http://www.w3.org/TR/2007/REC-soap12-part1-20070427/.

14. FIELDING.ROY, T. Architectural Styles and the Design of Network-based Software Architectures, [En Línea]. Consultado el: 18 de enero de 2013]. Disponible en: http://roy.gbiv.com/pubs/dessertation/top.htm.

15. POLANCO.GASCON, H. Estudio de un IDS Open Source frente a herramientas de análisis y explotación de vulnerabilidades., Universidad Carlos III de Madrid, 2010, p.

16. CHRISTL, A. La definición de Abierto en OGC, OSGeo y OSM. XIII convención y feria internacional Informática 2009, La Habana, Cuba, 2009.

17. MAP.GUDIE.OPEN.SOURCE.PROJECT Info Sheet. [En Línea]. Consultado el: 7 de Diciembre de 2012]. Disponible en: http://www.osgeo.org/mapguide.

18. WELCOME-GEOSERVER. [En Línea]. Consultado en: 7 de Diciembre de 2012]. Disponible en: http://geoserver.org.

19. DEEGREE. OSGeo-Live 5.0 Documentation, [En Línea]. Consultado el: 7 de Diciembre de 2012]. Disponible en: http://www.osgeo.org/mapguide.

20. BIENVENIDOS.A.MAPSERVER. MapServer 6.0.1 documentation, [En Línea]. Consultado el: 20 de Diciembre de 2012]. Disponible en: http://mapserver.org/es/index.html.

21. CARACTERISTICAS de II S 6.0. [En Línea]. Consultado el: 8 de Diciembre de 2012]. Disponible en: http://technet.microsoft.com/es-es/library/cc740244(WS.10).aspx.

22. SUN.JAVA.SYSTEM. *Notas de la versión de Sun Java System web Server 7.0 Actualización 1*, [En Línea]. Consultado el: 9 de Diciembre de 2012]. Disponible en: http://docs.oracle.com/cd/E19146-01/820-2674/6ne8s2442/index.html.

23. HTTPD Servidor web apache2. [En Línea]. Consultado el: 9 de Diciembre de 2012]. Disponible en: http://doc.ubuntu-es.org/HTTPD_Servidor_web_Apache2.

24. GRUPO.DANYSOFT. *Las Novedades de ASP.NET*, [En Línea]. Consultado el: 9 de Diciembre de 2012]. Disponible en: http://www.danysoft.com/free/aspnet.pdf

25. BARQUERO.CHAVES, B. I. and MENDEZ.RODRIGUEZ.W. *Características del lenguaje Perl 5.0 y su aplicación como herramienta de desarrollo en la elaboración de un Servidor Web*, [En Línea]. 2003. [Consultado el: 12 de Diciembre de 2012]. Disponible en: http://www.di-mare.com/adolfo/cursos/2007-2/pp-Perl.pdf.

26. *PHP Introducción Manual*. [En Línea]. Consultado el: 12 de Diciembre de 2012]. Disponible en: http://php.net/manual/es/intro.pdo.php.

27. QUIÑONES.E. *Introducción a Postgres*, [En Línea]. Consultado el: 12 de Diciembre de 2012]. Disponible en: http://www.apesol.org.pe.

28. POSTGIS. [En Línea]. Consultado el: 14 de Diciembre de 2012]. Disponible en: http://www.postgis.refractions.net/documentation/postgis-spanish.pdf.

29. CARRILLO, I., PEREZ.R. ,RODRIGUEZ.A. Metodología de desarrollo del Software, [En Línea]. Consultado el: 7 de Enero de 2013]. Disponible en: http://riunet.upv.es/bitstream/handle/10251/11867/Memoria.pdf?sequence.

30. ZAMUDIO, L., RAMIREZ.L, .ALVAREZ.M, .REODRIGUEZ.A. Teoría y Administración de Base de Datos, [En Línea]. Consultado el: 7 de Enero de 2013]. Disponible en: http://www.oocities.org/es/avrrinf/tabd/Foro/Foro_UML.htm.

31. Introducción a Herramientas CASE y System Architect. [En Línea]. 1995. [Consultado el 7 de Enero de 2013]. Disponible en: http://users.dsic.upv.es/asignaturas/eui/mtp/doc-practicas/intro_case_SA.pdf.

32. Visual Paradigm for UML. [En Línea]. Consultado el: 8 de Enero de 2012]. Disponible en:http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p.

33. Herramientas para el desarrollo del sistema. [En Línea]. Consultado el: 8 de Enero de 2013]. Disponible en: http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/hernandez_s_ja/capitulo4.pdf.

34. Bienvenidos a NetBeans Portal de IDE Java de Código Abierto. [En Línea]. Consultado el: 15 de Enero de 2013]. Disponible en: http://netbeans.org/index_es.html.

35. Introducción a Symfony2. [En Línea]. Consultado el: 15 de Enero de 2013]. Disponible en: http://parasitovirtual.wordpress.com/2011/02/03/introduccion-a-symfony2/

36. AYUDA.DE.GOOGLE. Acerca de Google Maps, [En Línea]. 2010. [Disponible en: http://maps.google.es/support/bin/answer.py?hl=es&answer=7060.

37. SECRETARIA.EJECUTIVA.DE.LA.IDERC. Infraestructura de Datos Espaciales de la República de Cuba, [En Línea]. 2005. [Consultado el: 29 del Enero de 2013]. Disponible en: http://www.iderc.co.cu/.

38. BUENMASTER. [En Línea]. Consultado el: 29 de Enero de 2013]. Disponible en: http://www.buenmaster.com/?a=536.

39 MVC. [En Línea]. Consultado el: 23 de Enero de 2013]. Disponible en: http://msdn.microsoft.com/es-es/library/bb972251.aspx#M19.

40. IAMRMAN, C. UML y patrones: Introducción al análisis y programación orientada a objetos. Pearson. 2003. p. 8420534382.

41. SUAZO, C. Modelo de Implementación, [En Línea]. Consultado el: 20 de Diciembre de 2012]. Disponible en: http://www.slideshare.net/joshell/diagramas-uml-componentes-y-despliegue.

ANEXOS.

Diagrama de clases del diseño del CU Gestionar Aplicación.

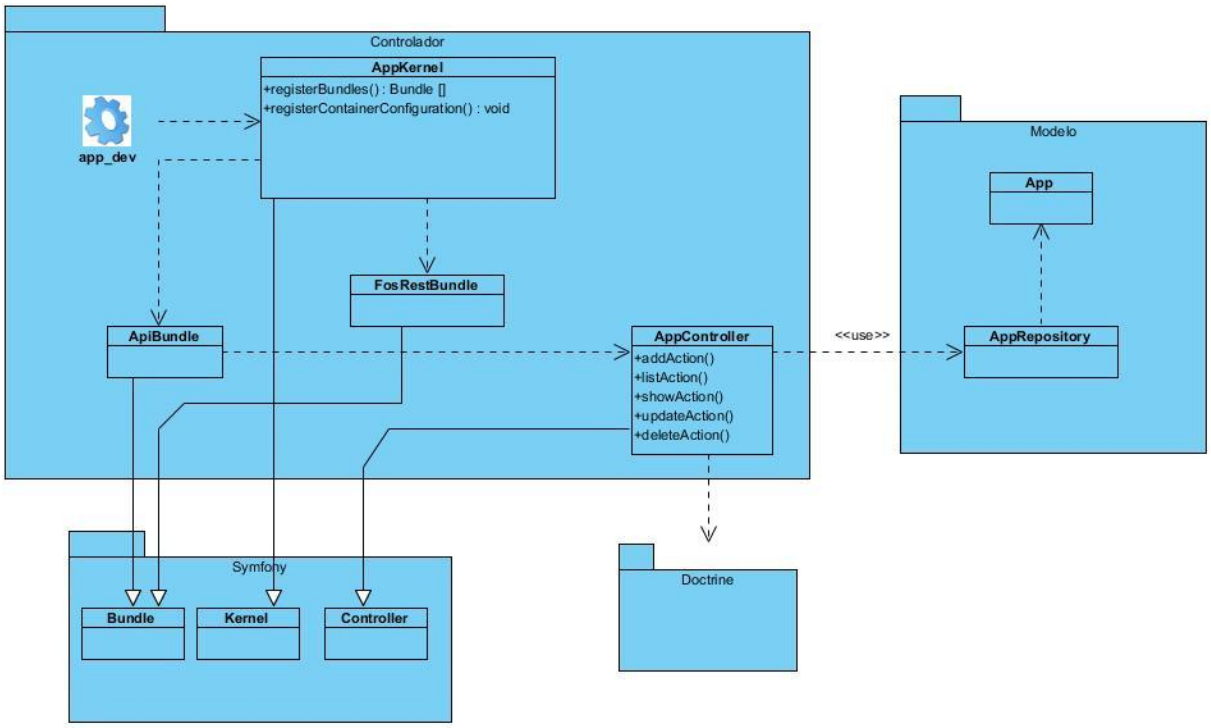


Diagrama de clases del diseño del CU Gestionar Role.

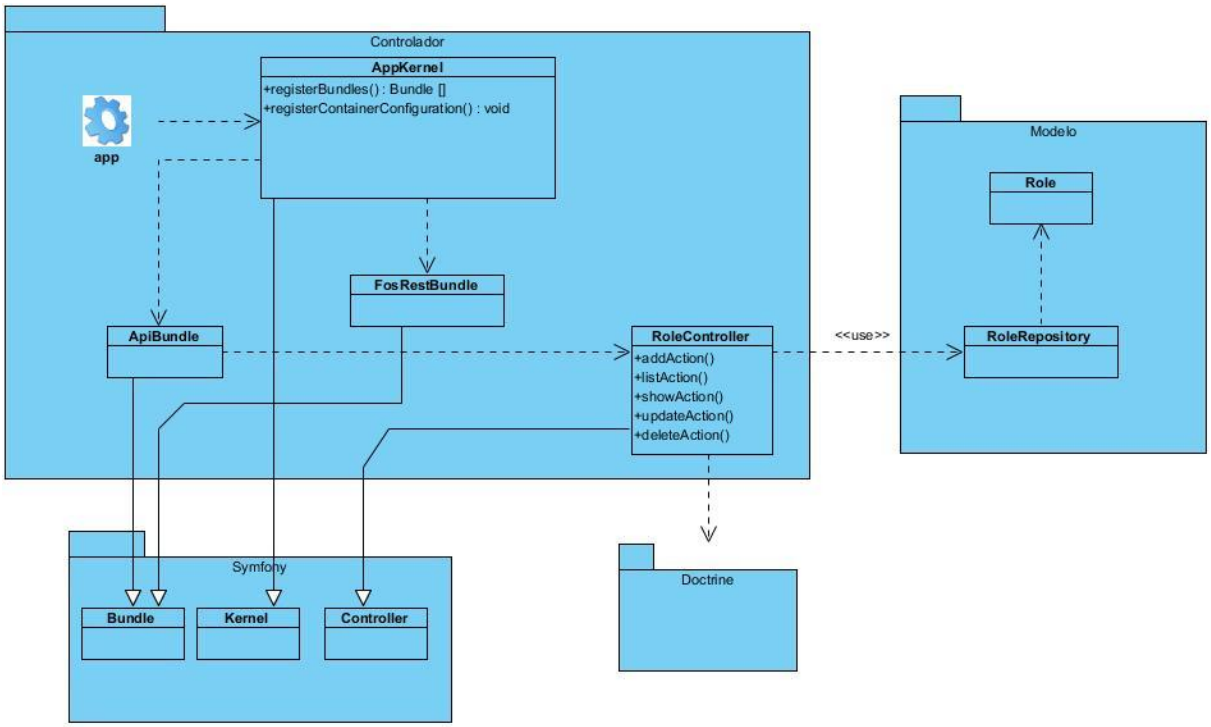


Diagrama de clases del diseño del CU Gestionar Capas.

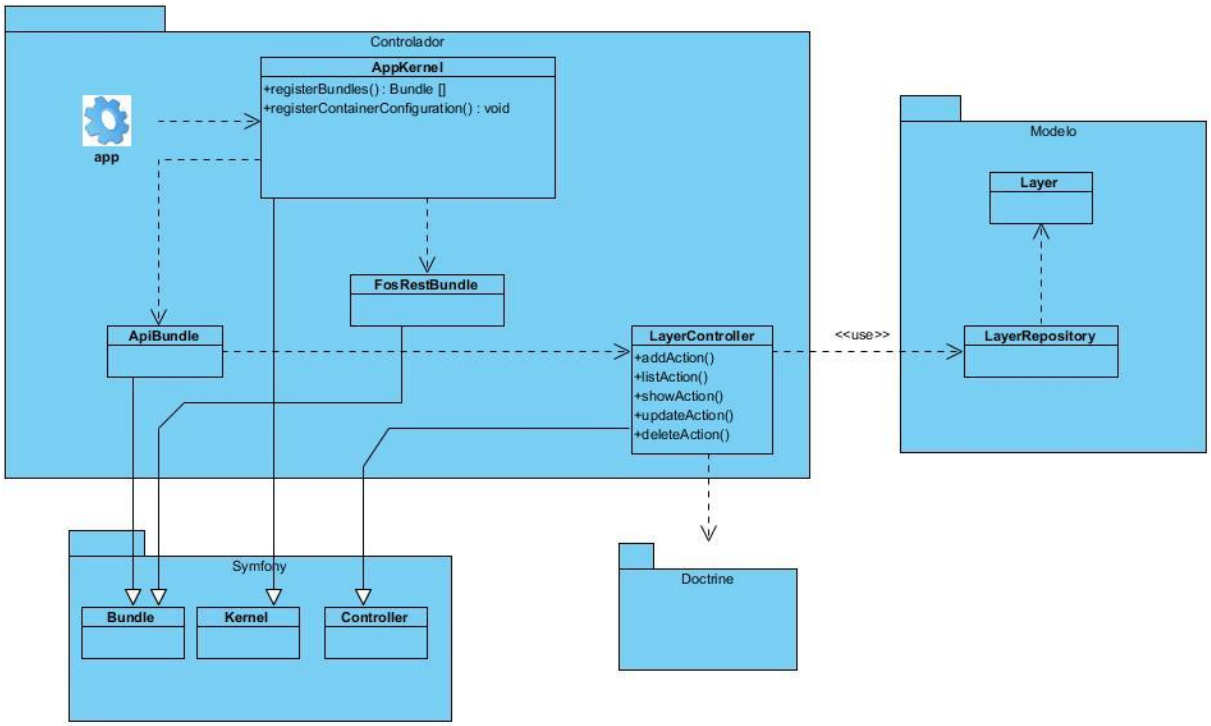


Diagrama de clases del diseño del CU Gestionar Campos.

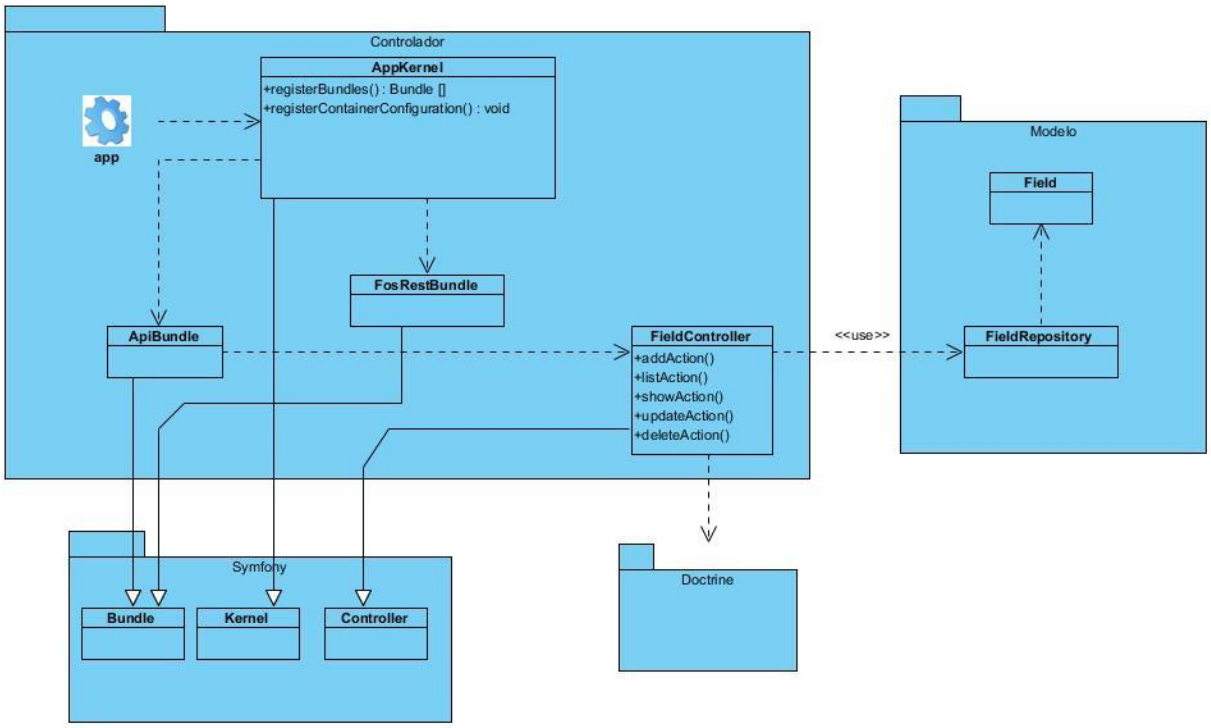
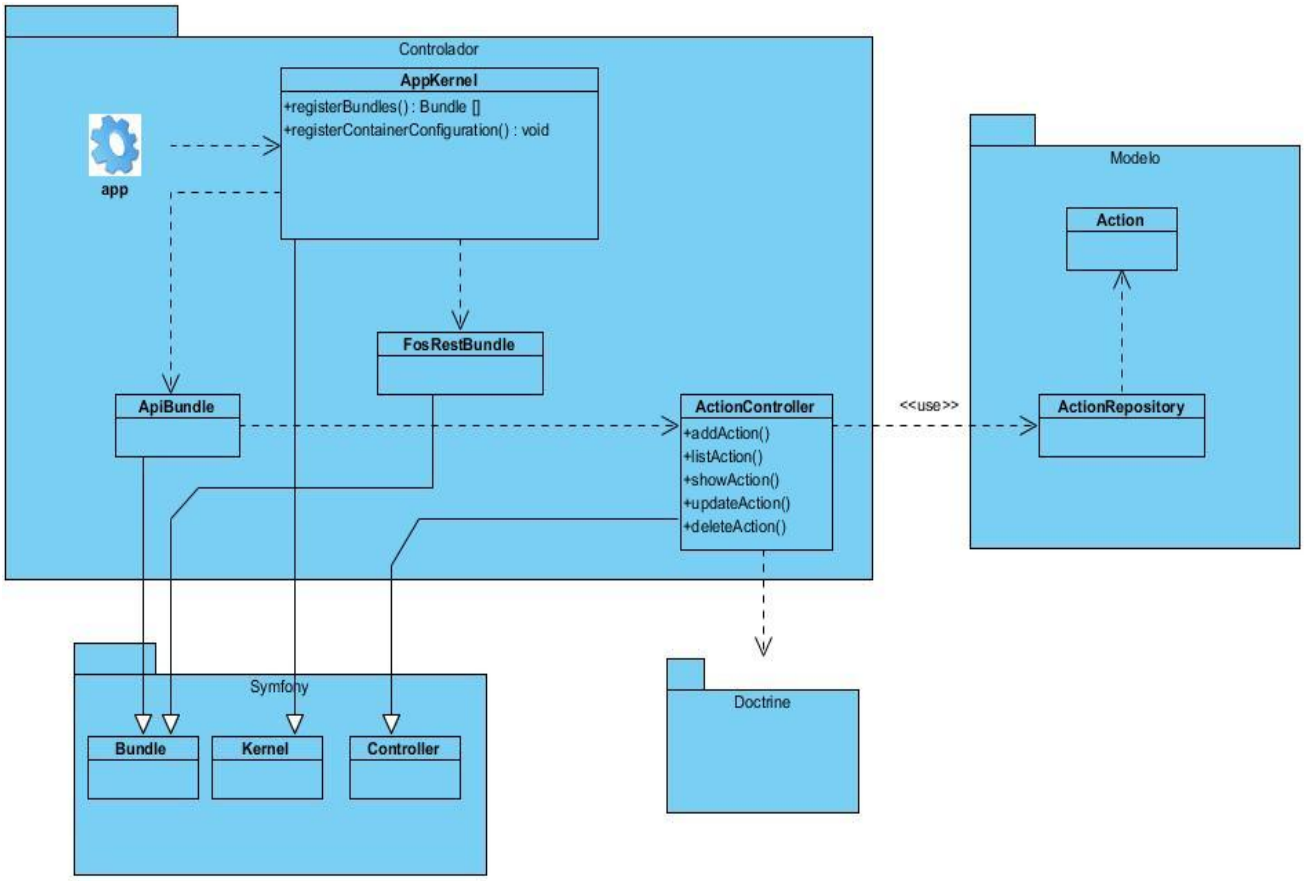
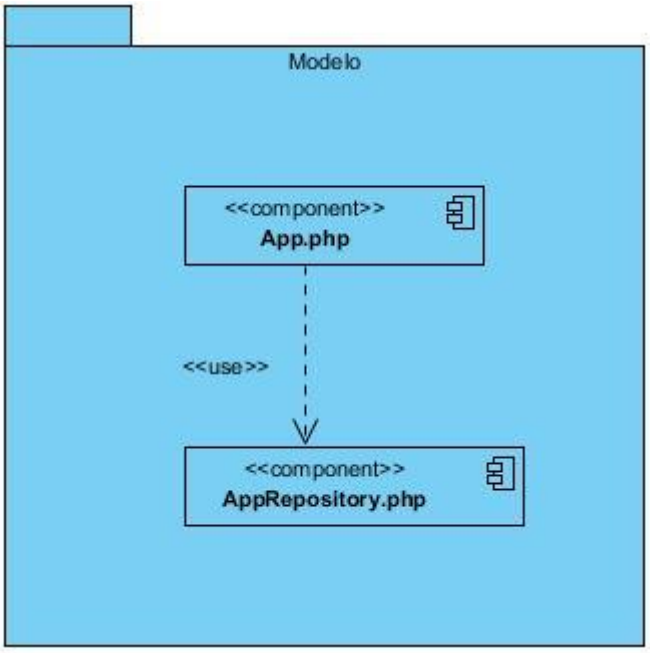


Diagrama de clases del diseño del CU Gestionar Acciones.

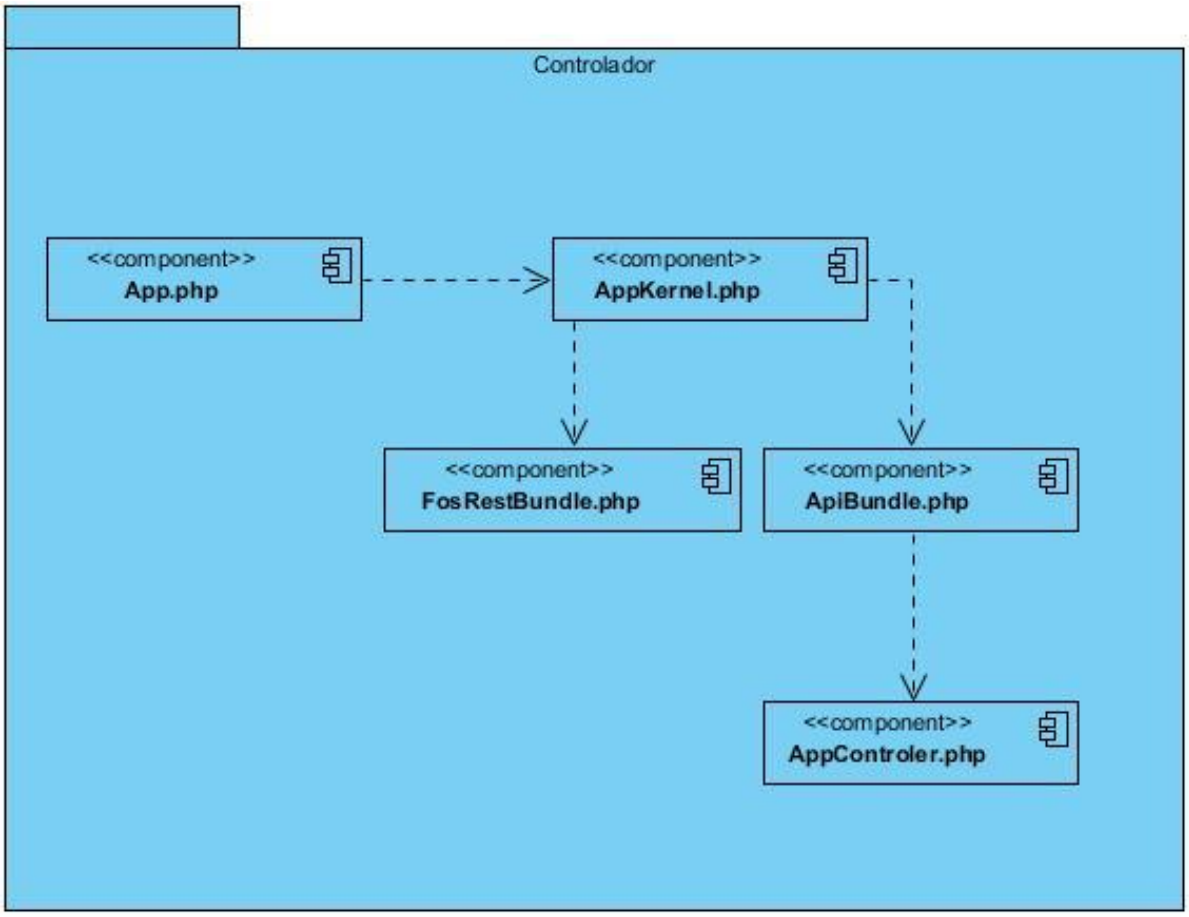

Controlador
AppKernel
+registerBundles() : Bundle []
+registerContainerConfiguration() : void
app
FosRestBundle
ApiBundle
ActionController
+addAction()
+listAction()
+showAction()
+updateAction()
+deleteAction()
<<use>>
Modelo
Action
ActionRepository
Symfony
Bundle
Kernel
Controller
Doctrine

Diagramas de Componentes

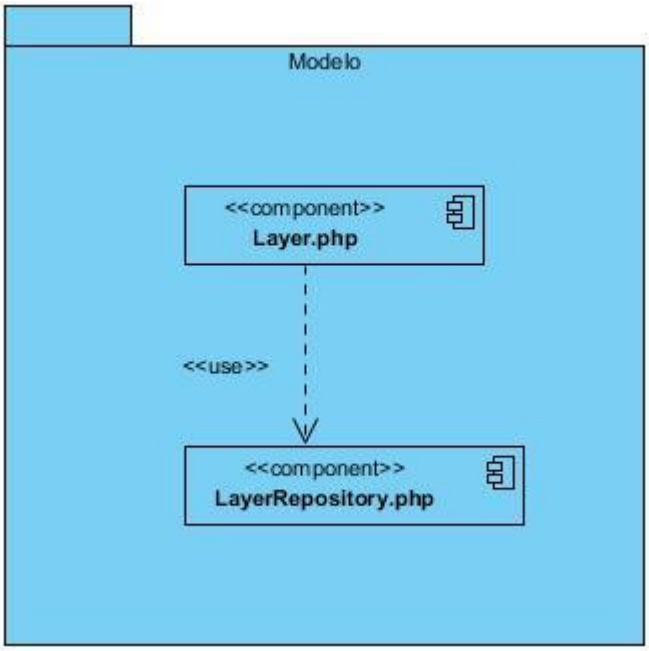
Vista Detallada: Paquete Modelo Gestionar Aplicación.



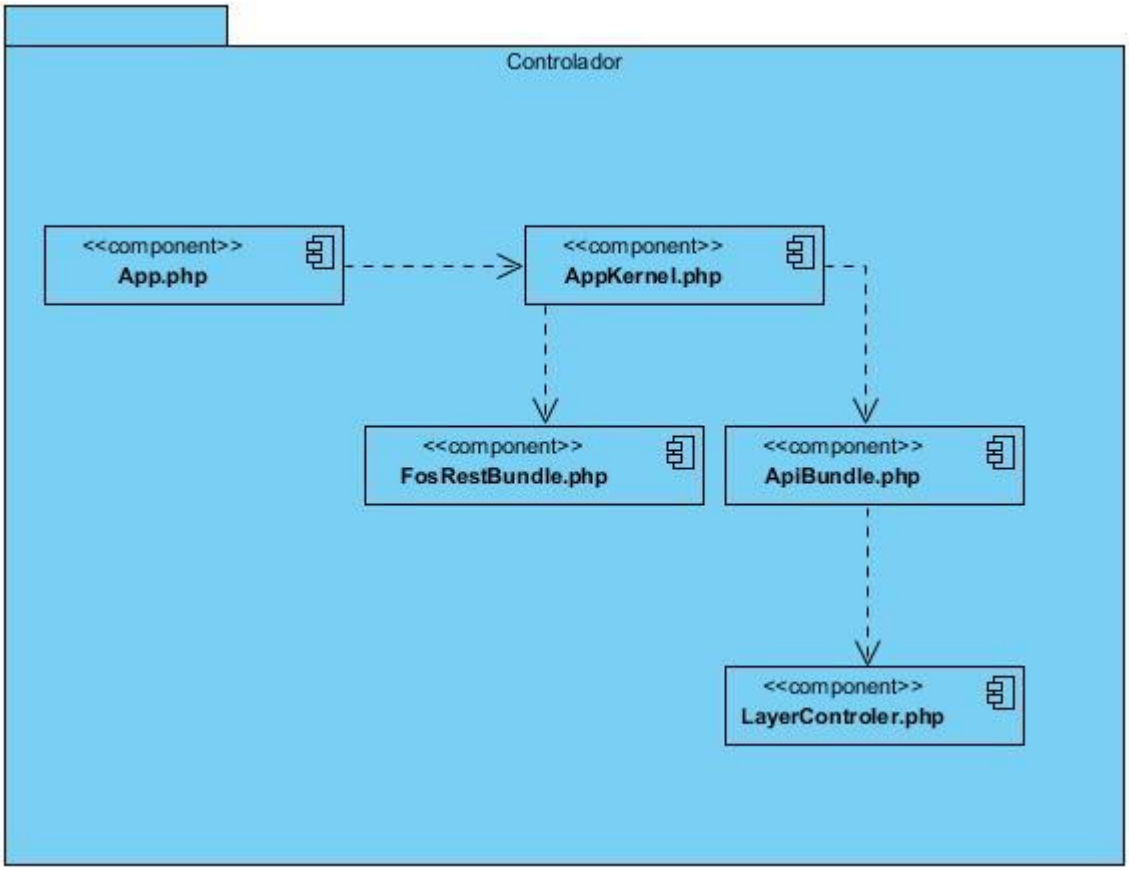
Vista Detallada: Paquete Controlador Gestionar Aplicación.



Vista Detallada: Paquete Modelo Gestionar Capa.



Vista Detallada: Paquete Controlador Gestionar Capas.



Glosario de términos.

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

Framework: Un framework, es una estructura de soporte definida mediante la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Servidor web: Es un programa que implementa el protocolo HTTP. Este protocolo está diseñado para transferir lo que llamamos hipertextos, páginas web o páginas HTML: textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de música.

XML: Siglas de *Extensible Markup Language*, lenguaje desarrollado por el W3 Consortium para permitir la descripción de información contenida en el WWW a través de estándares y formatos comunes, de manera que tanto los usuarios de Internet como programas específicos (agentes) puedan buscar, comparar y compartir información en la red.

HTML (*Hyper Text Mark-up Language*): Lenguaje desarrollado por el CERN que sirve para modelar texto y agregarle funciones especiales (como hipervínculos). Es la base para la creación de páginas Web tradicionales.

HTTP (*Hypertext Transfer Protocol*): Protocolo de Transferencia de Hipertexto utilizado en la WWW para transmitir las páginas de información entre el programa navegador y el servidor. Se destaca que el HTTP seguro es un protocolo HTTP mejorado con funciones de seguridad con clave simétrica.

Doctrine: Librería para el mapeo objeto relacional. Permite realizar las consultas mediante un lenguaje propio llamado DQL. (*Doctrine Query Language*).