Proyecto YuppGIS

Página Web

http://yuppgis.googlecode.com/

Estudiantes

Emilia Rosa

Jorge Davison

Martín Taruselli

Germán Schnyder

Cliente

Ing. Pablo Pazos

Tutora

Ing. Raquel Sosa

Informe de Proyecto de Grado presentado al Tribunal Evaluador como requisito de graduación de la carrera Ingeniería en Computación

Montevideo, Uruguay, 2011

Resumen

YUPP es un framework de desarrollo ágil de aplicaciones web, construido en su totalidad en PHP, que permite al desarrollador crear aplicaciones completas así como disponer de prototipos funcionales en corto tiempo.

Un sistema de información geográfica (GIS) es un sistema que permite capturar, almacenar, manipular y desplegar la información geográfica. Se caracterizan por requerir de configuraciones para los diversos servicios que utilizan, ya sea acceso a mapas, servidores para la visualización de los mismos, etc.

En la actualidad no existen frameworks construidos en PHP que permitan integrar información geográfica al modelo de dominio de una aplicación. En tal sentido, se plantea la extensión del framework para soportar tipos de datos geográficos y operaciones relacionadas a ellos, y facilitando la construcción de aplicaciones que integren estas funcionalidades sin necesidad de complejas configuraciones o experiencia previa.

Para comprender la forma en que esta información se procesa en los componentes desarrollados se realiza un análisis del framework en base al código fuente dada la ausencia de documentación formal al respecto. Se decide construir diferentes componentes para el tratamiento de la información en cada capa de la arquitectura, lo que redunda en un núcleo con componentes para la visualización de UI, procesamiento de consultas geográficas a través de la interacción con la base de datos y modelo de dominio de las entidades a incorporar.

Dado que la filosofía del framework es la del desarrollo ágil, es decir, la de facilitar al desarrollador que va a construir una aplicación de las herramientas necesarias para resolver los problemas cotidianos de acceso a datos y manejo de entidades persistentes, se diseñan componentes que permitan ir sorteando cada una de las instancias necesarias para mostrar objetos geográficos representativos de elementos del mundo real.

Para la visualización, se proveen herramientas que permiten mostrar mapas y diversos controles que sirven para buscar información relacionada al mismo. Además estas herramientas son flexibles respecto a la función que realizan, lo que permite por ejemplo cambiar el tipo de mapa que se está visualizando o cambiar el comportamiento de la interfaz de usuario una vez que se obtienen los resultados de una búsqueda (mostrar elementos en el mapa, ocultarlos o listarlos en la página entre otras cosas). Es importante comprender que cada acción sobre el mapa, desde seleccionar un elemento visible hasta filtrar los que cumplen con cierta característica, dispone del soporte necesario para ejecutar comportamiento propio de la aplicación en desarrollo.

La interacción entre la visualización y la lógica de negocio de la aplicación se facilita a través de las herramientas que se brindan para tal fin, y es importante que todo esto se haga a través de estándares de comunicación de datos geográficos.

Como el framework permite utilizar varios tipos de motores de base de datos, se conserva este comportamiento también en la extensión geográfica. En caso de disponer de una instalación local de una base de datos con soporte geográfico se establece la comunicación mediante el estándar WKT, y en caso de querer interactuar con una base de datos geográfica externa al sistema, mediante SOAP y REST.

Además se establece el soporte para dos fuentes de mapas como ser *Google Maps* y *MapServer*, a través del mecanismo de comunicación WMS pudiendo alternar entre ambas opciones en cualquier momento.

Palabras Claves

GIS, PHP, Yupp, YuppGIS, Framework, Desarrollo ágil, Servidores de mapas, Base de datos

Resumen 2

Palabras Claves 3

1. Introducción 5

1.1. Introducción al Problema 5

1.2. Motivación 5

1.3. Objetivos Planteados 5

1.4. Resultados Esperados 6

1.5. Resultados obtenidos 6

1.6. Metodología de Trabajo 7

1.7. Organización del Documento 8

2. Conceptos previos para entender el trabajo 9

2.1. Introducción 9

2.1.1. Servidor de Mapas 9

2.1.2. Estándares de OGC 9

2.1.3. MapServer 9

2.1.4. GeoServer 10

2.1.5. Modelo de Datos GIS 10

2.1.6. Representación de Datos 12

2.1.7. Visualizadores de mapas 13

2.1.8. Seguridad 14

2.1.9. PHP 14

2.1.10. Framework 15

2.1.11. Desarrollo ágil 15

2.1.12. YUPP Framework 16

3. Trabajo realizado 19

3.1. Introducción 19

3.2. Requerimientos 19

3.2.1. Descripción del sistema a construir 19
3.2.2. Referencias 19
3.2.3. Requerimientos funcionales 19
3.2.4. Requerimientos no funcionales 22
3.2.5. Requerimientos de Documentación 22
3.3. Análisis 23
3.4. Diagrama de Casos de Uso 23
3.4.1. Representación de tipos geográficos 24
3.4.2. Persistencia de tipos geográficos 24
3.4.3. Operaciones básicas sobre elementos de tipo geográfico 24
3.4.4. Construcción de mapas 25
3.4.5. Gestión de capas de datos 25
3.4.6. Gestión de elementos de las capas de datos 26
3.4.7. Filtrar elementos de las capas de datos 26
3.4.8. Búsquedas de elementos sobre las capas de datos 27
3.4.9. Eventos 27
3.4.10. Visualización de capas de datos 28
3.4.11. Visualización de elementos geográficos 29
3.5. Arquitectura 30
3.5.1. Modelo de Dominio 30
3.5.2. Modelo de la Arquitectura YUPPGIS 32
3.5.3. Diseño de la Arquitectura YUPPGIS 33
4. Implementación 35
4.1. Introducción 35
4.2. Solución propuesta 35
4.2.1. Introducción 35
4.2.2. Características de la solución 35
4.2.3. Interfaz de usuario 37

4.2.4. Consideraciones generales 37
4.2.5. Configuración 37
5. Caso de Estudio 39
5.1. Introducción 39
5.2. Características del sistema 39
5.3. Comparación 42
6. Conclusiones y Trabajo Futuro 45
6.1. Introducción 45
6.2. Conclusiones 45
6.2.1. Desarrollo del Proyecto 45
6.2.2. Logros obtenidos 45
6.3. Trabajos Futuros 47
7. Glosario 48
8. Referencias 50

1. Introducción

1.1. Introducción al Problema

Yupp [1], es un framework orientado a objetos, construido en su totalidad en PHP [2] para el desarrollo ágil de aplicaciones web. El objetivo principal de Yupp es la alta productividad en el desarrollo de aplicaciones completas y la creación rápida de prototipos funcionales.

Está compuesto por dos grandes componentes: YMVC y YORM. El primero es una implementación del patrón arquitectónico Model-View-Controller [3], que simplifica la separación de intereses (*SoC*) de los distintos componentes de un sistema de información. El segundo, YORM, es una implementación orientada a objetos de clases persistentes (clases, atributos, relaciones, etc.).

1.2. Motivación

Al momento de trabajar con Sistemas de Información Geográfica se encuentran grandes dificultades, como pueden ser las configuraciones de los servicios, base de datos, acceso a mapas y servidores para la visualización de los mismos. Demasiados conceptos, siglas y pocas guías orientadas a usuarios sin experiencia, hacen que el manejo de estos sistemas sea complejo y requiera de una extensa investigación.

Los paquetes existentes en PHP orientados a trabajar con Sistemas de Información Geográfica no integran Información Geográfica y objetos del dominio, así como tampoco se integran con la información del negocio. Estos paquetes se limitan en su mayoría al manejo de la información geográfica.

Teniendo en cuenta estas dificultades, se plantea la posibilidad de extender y/o adaptar Yupp, de forma de permitir al framework soportar información geográfica, obteniendo las ventajas de productividad y funcionalidad existentes.

La idea es permitir que los usuarios puedan desarrollar sistemas geográficos de forma sencilla pudiendo hacerlo sin tener experiencia previa en el tema y sin un estudio profundo del mismo, facilitando de forma considerable el desarrollo de prototipos funcionales completos.

1.3. Objetivos Planteados

Como primer objetivo se pretende extender YUPP y sus componentes de forma de conservar el comportamiento actual y permitir además, especificar atributos de tipo geográfico para representar conceptos tales como posición, región, etc. Se podrá manejar tipos de datos geográficos de modo de poder usarlos en las definiciones de clases persistentes del modelo de dominio y en consultas a la base de datos. La manipulación de datos geográficos se podrá hacer tanto a nivel de lógica de negocio como a nivel de interfaz de usuario.

El segundo objetivo planteado, es el de integrar una capa de seguridad que permita verificar restricciones de acceso a los datos geográficos, así como permitir la creación de reglas para dichas restricciones. De esta manera el desarrollador puede filtrar la información geográfica disponible al usuario.

Como objetivo final se plantea la implementación de una aplicación, un caso de estudio, utilizando la herramienta construida YUPPGIS y poder con ello analizar ventajas y desventajas principalmente en lo que a tiempo de desarrollo y complejidad del sistema refiere.

1.4. Resultados Esperados

Se espera brindar una herramienta que contribuya en el desarrollo de aplicaciones web geográficas utilizando el framework de desarrollo YUPP y que la utilización de esta herramienta implique una mejora considerable en la integración de datos geográficos al dominio de una aplicación web no geográfica.

A su vez, se espera que esta herramienta se integre con facilidad a sistemas en producción, sin que esto implique una reestructura del modelo o rediseño de la aplicación existente.

Por último esperamos comprobar los beneficios de la herramienta construida llevando a cabo la implementación de una aplicación web basada en otra ya existente, donde esta última brinda funcionalidades geográficas, pudiendo así comparar tiempos de desarrollo y complejidad del sistema.

1.5. Resultados obtenidos

Se verá que la herramienta construida permite agregar conceptos geográficos a una aplicación YUPP de manera simple y sin necesidad de grandes esfuerzos a la hora de codificar y configurar el sistema.

Además se comprobará que los diferentes componentes que se agregan a la arquitectura de YUPP interactúan entre sí de manera correcta y sencilla, permitiendo mantener la curva de aprendizaje del framework y respetando los lineamientos del desarrollo ágil [4].

También se notará la importancia de los estándares a la hora de establecer comunicaciones entre servidores [5] y visualizadores [6] de mapas, lo que permitirá reutilizar gran cantidad de código y liberarse de complejos subsistemas de traducción y mapeo de clases.

1.6. Metodología de Trabajo

Actividad	Abr	May	Jun	Jul	Ago	Set	Oct	Nov	Dic
Estudio GIS									
Estudio YUPP									
Creación de entorno de desarrollo									
Documento de Requerimientos									
Validación de Documento de Requerimientos									
Documento de Arquitectura y Diseño del Componente									
Análisis de Requerimientos									
Documento de Arquitectura									
Diseño e Implementación del Prototipo									
Implementación del Caso de Estudio									
Documentación final									

Se planteó el cronograma de trabajo que se muestra para permitirnos trabajar en forma ordenada, utilizando una metodología de tipo cascada, teniendo visibilidad completa sobre la gestión de tareas a realizar. Además las tareas fueron realizadas en forma conjunta por los integrantes del equipo, con distinciones de responsabilidades en aspectos puntuales de la implementación pero no en las decisiones de arquitectura y diseño.

Como parte del proceso en cascada, se validó lo realizado en cada etapa por el tutor y el cliente, cada uno en el área correspondiente, lo que permitió ir teniendo una noción clara del avance del proyecto. En particular se mantuvieron reuniones asiduas con el cliente, sobre todo al principio del proyecto, para armar un panorama completo de los requerimientos, y con el tutor para cerrar correctamente cada artefacto de documentación generado y pensar la planificación de la etapa siguiente.

Como línea de trabajo en la metodología utilizada se decidió la construcción de un prototipo incremental que permitiera ir probando lo desarrollado y, para mostrar este prototipo, el avance concreto del proyecto y las perspectivas de futuro del mismo, se realizó una presentación intermedia de tipo formal en el mes de setiembre frente a tutor y cliente.

En el cronograma se aprecia cada una de las etapas de desarrollo y su duración en el tiempo, notando un fuerte impulso a la hora de comprender las herramientas, los requerimientos y el entorno a utilizar para permitir una fluidez clara a la hora de implementar el componente. Es importante notar que dicho entorno se pensó con cuidado para evitar problemas clásicos de implementación en equipos y de larga duración, por lo que se armó una máquina virtual con todas las herramientas necesarias para la correcta sincronización del equipo. Además, como se aprecia en el diagrama, en el mes de julio no se realizó tarea alguna por el periodo de exámenes que afectó a algunos miembros del equipo.

1.7. Organización del Documento

El documento se organiza en secciones con la finalidad de presentar cada una de las etapas que componen el desarrollo del componente.

En la sección “Introducción al Problema” se definen a grandes rasgos los objetivos planteados así como la motivación para alcanzarlos. Se establecen resultados esperados y se resumen los resultados obtenidos. Se muestra el calendario que abarca las diferentes etapas del desarrollo.

A continuación se revisan los conceptos previos para entender el trabajo, con el fin de comprender la problemática y las herramientas disponibles, así como una revisión del estado del arte de las diferentes tecnologías involucradas en el proyecto.

La parte central del informe refiere al trabajo realizado y detalla las diferentes etapas de construcción de la solución en base a los requerimientos planteados por el cliente; se introducen los conceptos de arquitectura y diseño que prevalecerán en la etapa de implementación.

En la sección de implementación se detallan las decisiones tomadas con el fin de respetar lo planteado en la sección anterior. Se introducen detalles técnicos y puntualizaciones específicas del lenguaje de programación aplicado a la solución.

Como prototipo de prueba de los requerimientos realizados, se introduce el análisis e implementación de un caso de estudio sugerido por el cliente. Se detallan los pasos seguidos en la construcción de un ejemplo de sistema que utiliza el componente implementado en el presente proyecto. Se releva una lista de requerimientos puntuales de dicho ejemplo. Se realiza una comparativa con la implementación sin los beneficios del componente desarrollado.

Finalmente se realiza un recuento de las lecciones aprendidas a lo largo del proyecto, se realiza una evaluación de los problemas surgidos y de las posibles soluciones, se muestran los logros obtenidos y las dificultades sorteadas, se detallan los puntos de extensión del presente proyecto y los posibles problemas que puedan plantearse en el desarrollo futuro.

2. Conceptos previos para entender el trabajo

2.1. Introducción

En este capítulo presentamos resumidamente los conceptos necesarios para comprender la terminología y la construcción de las diferentes partes del componente final. Para mayor desarrollo de cada uno de estos conceptos de recomienda leer los documentos anexos “Material de estudio de Yupp” y “Material de estudio GIS”.

2.1.1. Servidor de Mapas

Un servidor de mapas es un servidor de cartografía digital (en inglés IMS: *Internet Map Server*) que provee cartografía a través de la red tanto en modo vectorial como con imágenes [7].

La especificación estándar para estos servidores es la OGC WMS, (*Open Geospatial Consortium Web Map Service*) [8].

2.1.2. Estándares de OGC

La OGC [9] define los estándares de servicios web implementados por servidores de mapas para exponer la información geográfica [10].

Las especificaciones más importantes surgidas del OGC son:

WFS - *Web Feature Service* o Servicio de entidades vectoriales que proporciona la información relativa a la entidad almacenada en una capa vectorial (cobertura) que reúnen las características formuladas en la consulta.

WMS - *Web Map Service* o Servicio de mapas en la web, produce mapas en formato imagen a demanda para ser visualizados por un navegador web o en un cliente simple.

WCS - Web Coverage Service, interfaz estándar para proveer interoperabilidad en el acceso a “coberturas” geoespaciales (imágenes con información relacionada).

CSW - Web Catalogue Service, interfaz común para el descubrimiento, búsqueda y consulta de metadatos relacionados a datos, servicios y recursos de tipo geográfico.

Dichos servicios están definidos en base a estándares de Internet: HTTP, URL, tipos MIME y XML. Deben soportar obligatoriamente el método GET. El método POST es obligatorio para WFS pero opcional para WMS.

2.1.3. MapServer

Es un servidor de mapas en código abierto escrito en C/C++ que permite la creación de “imágenes de mapas geográficos”. Soporta numerosos estándares de la OGC, tales como WMS, WFS, WMC, WCS.

MapServer [11] es un programa CGI que se encuentra inactivo en el servidor Web (Apache/IIS).

Cuando se envía una solicitud a *MapServer*, este usa la información pasada por parámetros y un archivo de configuración (*Mapfile*) para crear una imagen del mapa solicitado. El pedido puede retornar también imágenes para las leyendas, escalas, referencias y valores pasados como variables CGI.

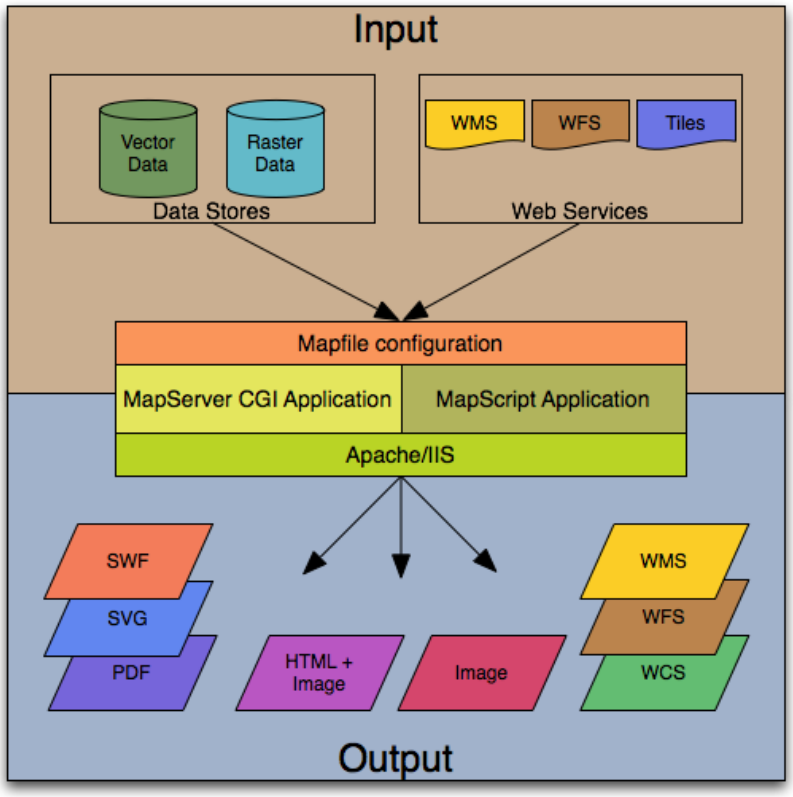

Figura 1 - Modelo de Arquitectura de MapServer

En la Figura 1 se puede ver la arquitectura de *MapServer* y como interactúa con otros componentes.

2.1.4. GeoServer

GeoServer [12] es un servidor de mapas en código abierto escrito en Java, permite a los usuarios compartir y editar datos geoespaciales. Diseñado para la interoperabilidad, publica datos de cualquier fuente de datos usando estándares abiertos.

Es una implementación del estándar *Open Geospatial Consortium Web Feature Service*, y también implementa las especificaciones de *Web Map Service* y *Web Coverage Service*.

2.1.5. Modelo de Datos GIS

OGC define un modelo de datos para objetos espaciales que son necesarios para almacenar y dibujar mapas [13] . En este caso nos interesa cubrir un mínimo de objetos necesarios para poder representar las figuras geográficas más usadas.

Cada motor de base de datos implementa las que cree necesarias, sin embargo la mayoría implementa (casi en su totalidad) el estándar de OGC como se muestra a continuación [14] [15]:

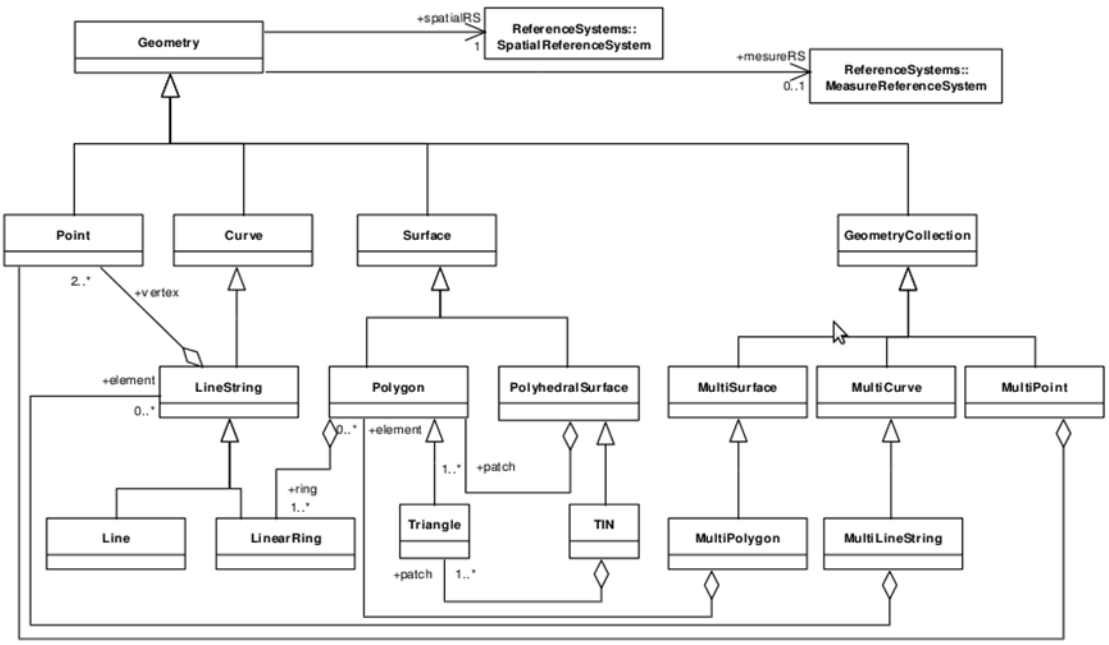

Figura 2 - Modelo de datos según estándar OGC [16]

Implementación del estándar por *PostGIS*: Point, LineString, Polygon, MultiPoint, MultiLineString, MultiPolygon, GeometryCollection [17].

Implementación del estándar por *MySQL Spatial Extensions*: Geometry, Point, Curve, LineString, Surface, Polygon, GeometryCollection, MultiPoint, MultiCurve, MultiLineString, MultiSurface, MultiPolygon [18] [19].

A continuación se muestra un esquema de lo mencionado:

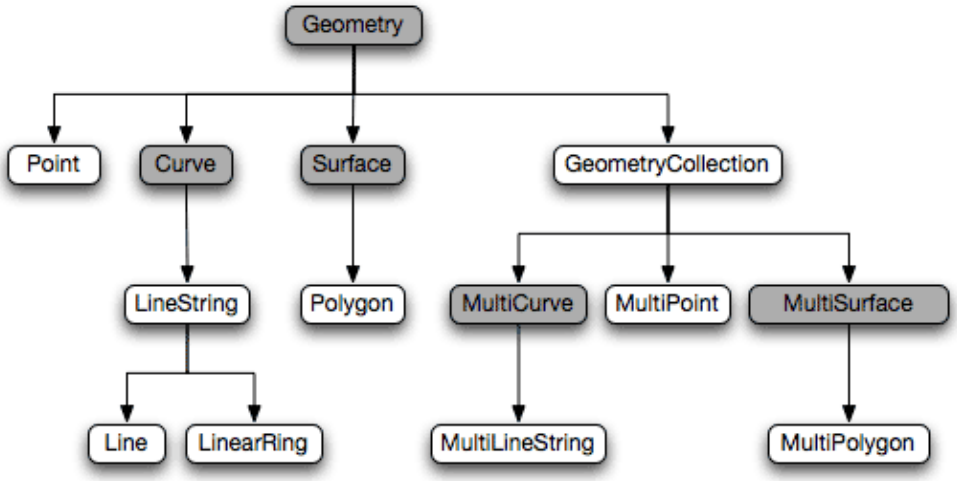

Figura 3 - Implementación del estándar por PostGIS y MySql Spatial

Como se puede observar ambos motores de base de datos representan los mismos objetos geográficos, lo cual equivale a un conjunto minimal del estándar.

2.1.6. Representación de Datos

Para la representación de datos se utilizan dos grandes lenguajes de marcas: KML y GML [20].

GML (*Geography Markup Language)* [21] describe entidades geográficas que pueden tener asociadas características geométricas (expresadas como puntos, líneas y polígonos) además de otros muchos tipos. Tiene el propósito de permitir a los usuarios crear vocabularios de objetos geográficos que pueden ser utilizados para el intercambio de datos, describir mensajes de Servicios Web Geoespaciales y expresar peticiones transaccionales.

KML (*Keyhole Markup Language)* [22], sin embargo, se utiliza para dar estilo a los *PlaceMarks* (lugares singulares) que aparecen en la aplicación *Google Earth*. De alguna forma, KML debería ser comparado con SVG, en vez de GML. Se describe así mismo como: "gramática XML y formato de fichero para modelar/almacenar entidades geográficas como puntos, líneas, imágenes y polígonos para ser mostrados en el cliente". En la figura 4 se puede apreciar el diagrama de tipos de datos que se incluyen en el lenguaje.

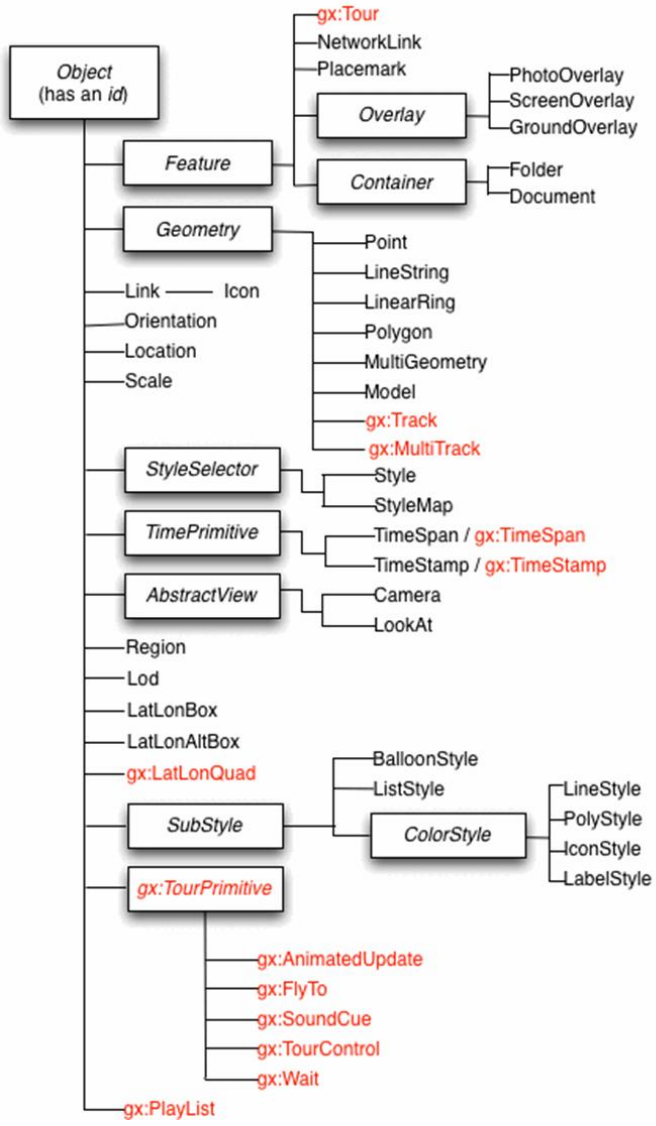

Figura 4 - Diagrama de datos de KML

La visualización geográfica incluye no sólo la presentación de los datos gráficos en el mundo, sino también el control de la navegación del usuario en el sentido de a dónde ir y dónde buscar.

Desde esta perspectiva, KML es complementaria a la mayoría de las principales normas existentes OGC como GML, WFS y WMS. Considerando que GML es un lenguaje para codificar contenido geográfico para cualquier aplicación, mediante la descripción de una gama de objetos de aplicación y sus propiedades (por ejemplo puentes, carreteras, boyas, vehículos, etc.), KML se puede utilizar para llevar el contenido de GML, y GML puede ser la información de estilo de KML.

En la actualidad, KML 2.2 utiliza ciertos elementos derivados de la geometría de GML 2.1.2 y se aprobó como estándar OGC en el 2008.

Adicionalmente, GML y KML utilizan esquemas para definir objetos que no están definidos nativamente en los lenguajes. En GML se realiza utilizando el lenguaje de esquemas de *W3C XML Schema*, mientras que KML lo hace con su propia definición de esquema de capacidades mediante el elemento KML.

A pesar de estas similitudes, los objetivos de GML y KML son bastante diferentes

2.1.7. Visualizadores de mapas

Un "Visualizador de mapas" es una herramienta de interfaz de usuario que permite realizar consultas gráficas simples sobre información geográfica; es el cliente de un servidor de mapas y se encarga de la interacción con el usuario ofreciendo una interfaz amigable y sencilla de usar.

Algunos visualizadores usan comunicación nativa o tecnología propietaria, algunos son de escritorio y otros web.

2.1.7.1. OpenLayers

OpenLayers [23] es una biblioteca de JavaScript de código abierto para mostrar mapas interactivos en los navegadores web. OpenLayers ofrece un API para acceder a diferentes fuentes de información cartográfica en la red: WMS, Mapas comerciales (*Google Maps, Bing, Yahoo,* etc.), WFS, distintos formatos vectoriales, mapas de *OpenStreetMap*, etc.

Tipos de capas soportadas por OpenLayers:

Capas Base

Capas Base son capas mutuamente excluyentes, comúnmente capas *Raster*. La capa activa determina las proyecciones disponibles (sistema de coordenadas) y niveles de zoom disponibles en el mapa.

Capas soportadas: Google, Image, KaMap, KaMapCache, MapGuide, MapServer, MultiMap, TMS, TileCache, VirtualEarth, WMS, WorldWind y Yahoo.

Capas No Base

Las capas No Base (o superposiciones) son la alternativa a las capas base. Múltiples capas de No Base pueden estar activas a la vez. Estas capas no controlan los niveles de zoom del mapa, pero puede ser activadas o desactivadas según distintos eventos. Las capas No Base siempre se muestran por encima de las capas base.

Capas soportadas: Boxes, GML, GeoRSS, Markers, PointTrack, Text, Vector y WFS.

2.1.8. Seguridad

Los servidores de mapas no ofrecen una forma de controlar el acceso, sin embargo la OGC generó el estándar *GeoXACML* [24] para la implementación del Control de Acceso en Sistemas de Información Geográficos. *GeoXACML* es una extensión de *XACML* de *OASIS*.

El concepto de seguridad sobre OGC *Web Services* sirve para la regulación de acceso a la información geoespacial. La OGC propone una arquitectura (ver figura) para dicha implementación, la cual se puede incorporar en forma de extensión, sin la necesidad de modificar los componentes ya existentes. El control de acceso provee la funcionalidad donde un usuario identificado posee permisos para invocar operaciones predefinidas sobre objetos particulares.

La arquitectura propuesta introduce el componente de control de acceso de acuerdo a la arquitectura XACML. Estos componentes son WMS-PEP el cual intercepta la comunicación entre el cliente WMS y el WMS. La autenticación de usuarios es expuesta por el componente de autenticación y la extensión GeoPDP maneja los permisos de los usuarios basado en políticas GeoXACML.

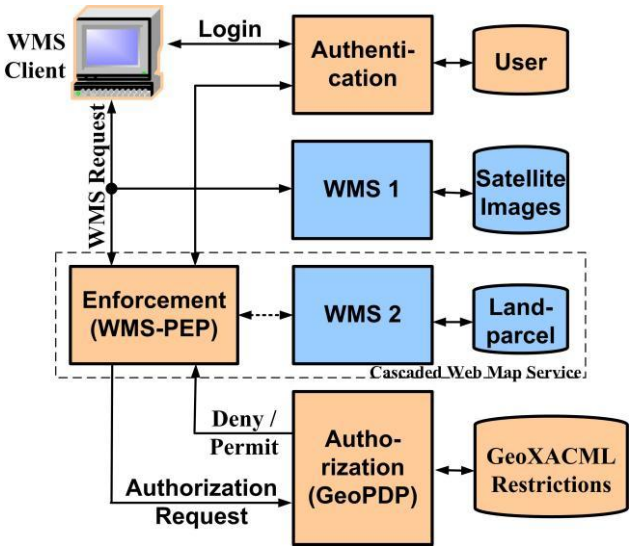

Figura 5 - Arquitectura para implementación del estándar GeoXACML

2.1.9. PHP

PHP es el acrónimo de *Hypertext Preprocessor*.

Es un lenguaje de programación de scripting del lado del servidor, rápido, con una extensa librería de funciones que cubre desde cálculos matemáticos complejos hasta tratamiento de conexiones de red.

El siguiente esquema muestra el funcionamiento de las páginas PHP:

Figura 6 - Esquema de funcionamiento de páginas PHP

2.1.9.1. Ventajas

Como característica principal se tiene que es gratuito e independiente de plataforma por lo que se presenta como una alternativa de fácil acceso para todos.

El código fuente escrito en PHP es invisible al navegador y al cliente ya que es el servidor el que se encarga de ejecutar el código y enviar el resultado en formato HTML al navegador. Esto hace que la programación en PHP sea segura y confiable.

Posee compatibilidad con la mayoría de los motores de base de datos que se utilizan en la actualidad, destacando su conectividad con *MySQL* y *PostgreSQL*.

Permite aplicar técnicas de programación orientada a objetos, no requiere definición de tipos de variables aunque sus variables se pueden evaluar también por el tipo que estén manejando en tiempo de ejecución y posee desde PHP5 manejo de excepciones.

2.1.10. Framework

Un framework es un conjunto de funcionalidades y herramientas que definen prácticas y conceptos generalizados con el fin de ayudar en la construcción de software.

En la práctica se traduce en bibliotecas para facilitar y acelerar el desarrollo así como para mejorar la calidad del software. Estas bibliotecas pueden brindar herramientas tales como generadores de vistas, servicios genéricos, clases base, wrappers varios, etc.

Lo podemos ver como una aplicación genérica parcialmente implementada.

2.1.11. Desarrollo ágil

El desarrollo ágil es una propuesta liviana para el desarrollo de software y nace como alternativa a la metodología tradicional (que podría pensarse como de tipo pesada).

Se trata de seguir pequeños consejos y mejorar así la productividad y calidad del código, todo esto mediante iteraciones de corto plazo. Los conceptos básicos de la metodología ágil se reunieron en el llamado manifiesto por el desarrollo ágil.

2.1.12. YUPP Framework

Este framework relaciona los conceptos anteriores; integra la metodología ágil mediante la implementación de bibliotecas que facilitan el desarrollo de nuevas aplicaciones, tratando de minimizar los errores y de maximizar la velocidad y performance de estas.

2.1.12.1. Modelo de información

Yupp implementa el patrón de diseño *Model-View-Controller* (MVC).

Todas las clases del modelo heredan de PersistentObject (PO), donde PO es la clase que implementa toda la lógica necesaria para la persistencia del modelo de información mediante *ORM (Object Relational Mapping),* la implementación de ORM de Yupp se llama YORM.

Yupp permite salvar un modelo en cascada (si se salva una sola instancia de una clase, se salvan también las instancias de las clases asociadas). Además permite relacionar los objetos de dos maneras: hasOne, para declarar una relación 1 a 1, y hasMany, para declarar una relación 1 a N.

2.1.12.2. Controladores

Los controladores cumplen el rol establecido en la arquitectura MVC. Principalmente contienen métodos para permitir invocaciones desde la interfaz de usuario denominados “Acciones”.

Estas acciones son las encargadas de procesar los pedidos, devolver una vista o un conjunto de datos, o redirigir hacia otras acciones (encadenamiento).

Los controladores de una aplicación se ubican en el subdirectorio "controllers" de la aplicación.

Yupp define ciertas convenciones para la nomenclatura de las clases y los métodos participantes en el flujo de ejecución de la aplicación.

Estas son:

- Nombres
  - Los nombres de los controladores deben terminar en “Controller”.
  - Los nombres de las clases siempre empiezan en mayúscula.
- Herencia
  - Todo controlador debe heredar de la clase base YuppController.
- Acciones
  - Los nombres de los métodos que son acciones deben terminar en “Action”.
  - Los nombres de los métodos siempre empiezan en minúscula.

2.1.12.3. Acciones

Las acciones son los métodos principales de los controladores y pueden tener diferentes tipos de retorno: retorno vacío, retorno de vista específica, retorno de parámetros y redirección a otra acción.

2.1.12.4. Aplicaciones

La aplicación es la unidad básica de desarrollo de Yupp Framework.

Yupp soporta varias aplicaciones a la vez, donde cada aplicación está formada por controladores, modelo, vistas y otros recursos como scripts, servicios, etc.

Cuando el desarrollador crea una nueva aplicación se crea la estructura de directorios necesaria para funcionar en Yupp, se crea el descriptor de la aplicación y se crea un controlador vacío por defecto.

2.1.12.5. Base de datos

Yupp permite que aplicación pueda tener su configuración de base de datos independiente de las demás aplicaciones. Soporta conexiones a bases de datos de tipo MySql, PostgreSQL y SQL Lite.

2.1.12.6. Templates

Yupp incorpora el concepto de templates para las vistas. El objetivo es tener centralizado en un solo lugar la lógica de la GUI encargada de mostrar una determinada entidad de negocio, de forma de utilizarla siempre que se desee mostrar en una vista.

2.1.12.7. YuppLoader

YuppLoader es la clase a través de la cual se cargan clases, interfaces y scripts, dentro de una aplicación Yupp. La carga mediante YuppLoader equivale a hacer *include_once* de un archivo en PHP, pero las ventajas de que YuppLoader resuelve las rutas de inclusión y almacena un caché de los archivos cargados por cada aplicación.

2.1.12.8. YuppConfig

YuppConfig contiene toda la configuración de Yupp Framework. Se configura la conexión a la base de datos y el modo de ejecución del framework.

Existen tres modos de ejecución:

- **desarrollo**: Al desarrollador le aparece un "escritorio" desde donde puede generar las tablas en la base de datos, ejecutar scripts de bootstrap, acceder a todos los controladores disponibles y ver estadísticas de código.
- **testing**: En el futuro servirá para cuando se quieren lanzar tests automáticos sobre la aplicación.
- **producción**: En este modo el framework se comporta como si la aplicación se fuera a instalar en un servidor y quedara pronta para ser utilizada por los usuarios.

2.1.12.9. Bootstrap

El bootstrap de Yupp se utiliza para cargar toda la información inicial que una aplicación necesita para funcionar correctamente.

Se ejecuta una sola vez luego de que un componente es instalado en el framework.

2.1.12.10. Modelo de Componentes

A continuación se presenta un diagrama de los componentes de Yupp:

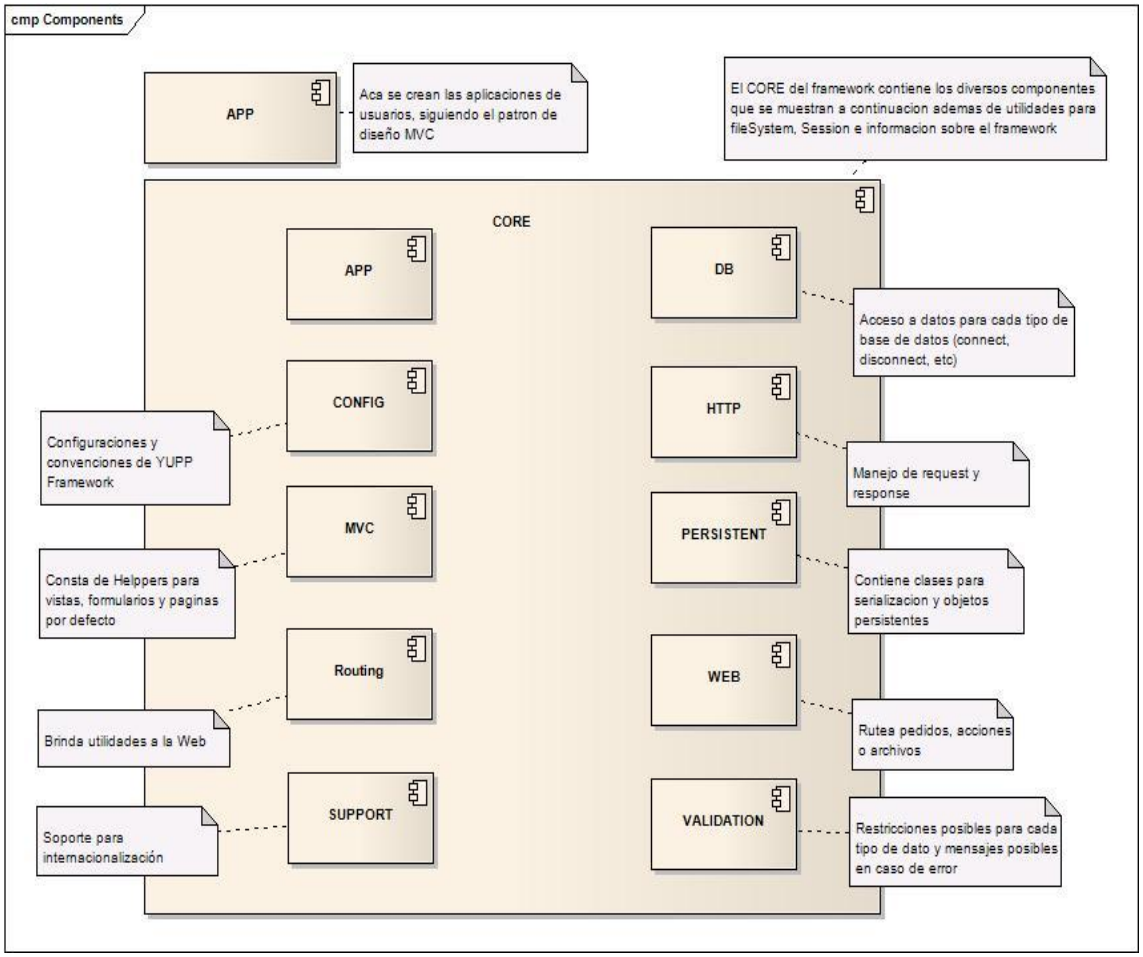

Figura 7 - Modelo de Componentes de YUPP

3. Trabajo realizado

3.1. Introducción

Este capítulo muestra el proceso llevado a cabo para obtener los insumos necesarios que permitan construir el sistema final. Con dicha finalidad es que se definen tres secciones principales, la primera referida a la etapa de relevamiento de requerimientos, la segunda referida al análisis de estos, y la tercera referida a la arquitectura del componente a construir. Para profundizar en cada una de estas etapas se recomienda la lectura de los documentos anexos relacionados.

3.2. Requerimientos

3.2.1. Descripción del sistema a construir

El componente permite manejar tipos de datos geográficos para representar conceptos tales como posición, región, ruta, etc.

Se extiende el componente YORM del framework YUPP de forma de conservar el comportamiento actual y permitir especificar atributos de tipo geográfico, de modo de poder usarlos en la definición de clases persistentes del modelo de dominio y en consultas a la base de datos.

Se provee de un conjunto de herramientas para facilitar la construcción de interfaces de usuario adecuadas al tipo de información que se incorpora al framework. Dichas herramientas siguen la línea ágil de trabajo del framework y permiten acciones tales como dibujar un mapa, agregar y quitar elementos del mismo, construir y aplicar filtros sobre los elementos, etc.

Por último, se permite restringir el acceso a la información geográfica en base a restricciones sobre los datos y a la creación de reglas para el acceso a los mismos. De esta forma es posible configurar quién accede a qué tipo de datos y qué tipo de acción puede realizar (consultar, modificar, eliminar, etc.) sobre estos.

3.2.2. Referencias

- Material de Estudio GIS
- Material de Estudio YUPP
- Propuesta de Proyecto de Grado

3.2.3. Requerimientos funcionales

3.2.3.1. Representación de tipos geográficos

Se incorporan tipos de datos básicos que permiten modelar figuras geográficas.

Los tipos de datos geográficos modelados son los siguientes: punto, línea recta, línea curva, elipse y polígono.

Se permite la definición de colecciones de los tipos de datos mencionados.

3.2.3.2. Persistencia de tipos geográficos

Se adaptan las operaciones existentes en YUPP framework para que estas sean capaces de persistir los datos geográficos definidos en el punto anterior.

3.2.3.3. Operaciones básicas sobre elementos de tipo geográfico

Se proveen las siguientes operaciones sobre elementos de tipo geográfico:

De manipulación (devuelven un nuevo objeto geográfico):

- Intersección entre dos elementos de tipo geográfico
- Unión de dos elementos de tipo geográficos
- Diferencia entre dos elementos de tipo geográficos

De consulta (devuelven un valor booleano o escalar):

- Distancia entre dos objetos de tipo geográfico en un mapa
- Área de una elemento de tipo geográfico
- Pertenencia de un elemento de tipo geográfico a otro
- Equivalencia de una colección de objetos geográficos a otro tipo geográfico (por ejemplo determinar si un conjunto de puntos forman una línea recta, una línea curva, un polígono, etc.).

Se brindan herramientas de UI que permiten trabajar con estas operaciones por lo que es posible operar con los elementos que se están visualizando en algún mapa.

3.2.3.4. Construcción de mapas

Se provee un conjunto de UI Helpers que permiten declarar los parámetros básicos de un mapa y generan la salida correspondiente a nivel de interfaz de usuario.

Se permite instanciar múltiples mapas dentro de la aplicación, donde cada mapa posee una configuración separada, de forma de poder manipular sus atributos por separado.

El visualizador de mapas a utilizar es especificado por el usuario, el cual puede elegir entre varias alternativas brindadas por OpenLayers como: Google Maps, Bing, Yahoo, etc.

En la configuración de cada mapa se puede especificar aspectos tales como tamaño del mapa, zoom y centro del mismo.

3.2.3.5. Gestión de capas de datos

Se provee un conjunto de herramientas que permiten crear capas de datos, de forma de representar colecciones de objetos geográficos.

Se permite crear múltiples capas, y cada capa puede estar asociada a uno o más mapas. A su vez cada capa puede contener elementos de otras capas, por lo que funcionan como contenedores no excluyentes.

Además cada capa cuenta con atributos predefinidos que deben ser establecidos por el usuario al momento de crearlas, como ser: nombre, descripción y palabras clave (o etiquetas); y la posibilidad de agregar atributos personalizados de forma de poder agrupar estas capas en colecciones según sus atributos.

Se brindan herramientas que permiten gestionar las asociaciones existentes entre los mapas y las capas, ya sea para asociar, desasociar, etc.

3.2.3.6. Gestión de elementos de las capas de datos

Se proveen operaciones sobre las capas que permiten agregar y quitar uno o varios objetos geográficos de una o varias capas.

Se deben especificar las capas y los elementos a agregar o quitar. Estos elementos son compartidos por todos los mapas que contengan las capas en cuestión.

Se provee a su vez de herramientas que permiten agregar y quitar elementos de una capa bajo ciertas condiciones a cumplir por el objeto geográfico (por ejemplo por el valor de uno de sus atributos). Cuando se quita/agrega un elemento se procede a quitarlo/agregarlo de todas las capas que se especifiquen.

Se brindan UI Helpers que permiten trabajar con estas operaciones por lo que es posible operar con los elementos que se estén visualizando en algún mapa.

3.2.3.7. Filtrar elementos de las capas de datos

Se provee un conjunto de operaciones sobre las capas que permiten crear filtros sobre la colección de elementos de acuerdo al valor de sus atributos de dominio.

Se debe especificar la capa a filtrar, el tipo de acción a llevar a cabo (mostrar, ocultar o delegar a alguna función del usuario) sobre los elementos que cumplan con el filtro y los valores de los atributos del filtro.

Se brinda un conjunto de UI Helpers que permiten trabajar con estas operaciones de forma de mostrar los resultados a nivel de interfaz de usuario. Por ejemplo, si se decide mostrar en un mapa los elementos con cierta característica, se construye el filtro en la capa de visualización, se realiza el filtrado en la capa lógica y se refleja el resultado en el mapa (capa de visualización).

3.2.3.8. Búsquedas de elementos sobre las capas de datos

Se provee un conjunto de operaciones sobre las capas que permite obtener un listado de elementos que cumplen con los criterios de la búsqueda.

Los criterios de búsqueda son construidos en base a los valores de los atributos de dominio de los elementos de las capas.

Esta funcionalidad es similar al filtrado salvo que en este caso se retorna la colección de elementos resultante sin ninguna consecuencia tanto a nivel de capa de visualización como de capa lógica.

Esta búsqueda se puede realizar en una o varias capas, por lo cual los elementos que se obtengan pueden pertenecer a diferentes capas.

Se brindan UI Helpers que permiten trabajar con estas operaciones y desplegar los resultados de las búsquedas en la capa de visualización.

3.2.3.9. Eventos

Se provee un conjunto de operaciones que permiten declarar una acción a llevar a cabo una vez ocurrido algún evento de capa de visualización sobre uno o varios elementos de un mapa, así como sobre las capas del mismo.

Denominamos “evento” a cualquier acción llevada a cabo mediante el mouse o el teclado. Ejemplos de eventos predefinidos son: clic, doble clic, tecla shift y clic de mouse, tecla enter, etc.

Se debe especificar la referencia al mapa, el evento y el método a invocar una vez ocurrido el mismo. El método recibe una referencia al elemento objeto del evento.

Asimismo se permite declarar eventos personalizados, y los handlers correspondientes, sobre los elementos; y siguiendo la misma lógica explicada anteriormente, estos handlers se ejecutan cuando el evento correspondiente sucede. Estos eventos, al ser personalizados, deben ser explícitamente disparados por el usuario.

Un evento, predefinido o personalizado, puede tener múltiples handlers, y todos son notificados cuando el evento sucede.

3.2.3.10. Visualización de capas de datos

Se provee un conjunto de herramientas que permiten especificar atributos de visualización para una capa de datos.

Se permite especificar un template de visualización por cada capa de datos, y un template de visualización genérico para todas las capas del mapa, que permiten mostrar los atributos de dominio.

Se provee también un conjunto de UI Helpers que permiten modificar las propiedades de visualización de las capas de datos en los mapas de acuerdo a los atributos de visualización.

En particular se brinda un UI Helper que lista las capas de datos de un mapa y permite mostrar u ocultar las mismas en la interfaz de usuario.

3.2.3.11. Visualización de elementos geográficos

Se provee un conjunto de operaciones que permiten especificar templates de visualización para los elementos de las capas de datos (atributos del modelo de dominio).

Es posible especificar un template de visualización por cada tipo de dato a mostrar en cada capa, un template de visualización genérico para todos los elementos del mapa, un template de visualización para cada tipo de dato de un conjunto de capas (o de todo el mapa), etc.

Se provee también un conjunto de UI Helpers que permiten modificar las propiedades de visualización de los elementos en los mapas de acuerdo a los atributos de visualización (ícono, color, etc.).

Se puede, mediante dichos helpers, modificar las propiedades de un elemento en particular así como las propiedades de un conjunto de elementos seleccionados o todos los elementos de alguna capa.

3.2.4. Requerimientos no funcionales

Se incorpora al framework Yupp un nuevo componente, el cual debe mantiene la línea de desarrollo ágil, poniendo especial énfasis en aspectos como mantenibilidad, extensibilidad, usabilidad y estabilidad.

El componente es compatible con la versión 0.4 (última versión) de Yupp Framework.

Es compatible también con la siguiente versión de base de datos PostgreSQL: PostgreSQL 8.4.7.Ubuntu + PostGIS 1.5.1.Ubuntu

Como visualizador de mapas se utiliza OpenLayers.

3.2.5. Requerimientos de Documentación

- Guía de instalación y configuración
    - Dedicado a los usuarios de Yupp.

Guía al usuario paso a paso a través del proceso de instalación y configuración de aquellas aplicaciones que usan el componente desarrollado.

- Archivo Léame
    - Dedicado a los usuarios de Yupp.

Es un documento en formato texto, que será de utilidad en caso de tener problemas al iniciar o usar el componente desarrollado.

- Manual de Usuario
    - Dedicado a los usuarios de Yupp.

Se brinda a los usuarios, una guía ágil para comprender y utilizar las funcionalidades del componente desarrollado mediante ejemplos y explicaciones concretas

3.3. Análisis

Se presenta a continuación el análisis de los requerimientos que se desprenden de la sección anterior, así como su descripción en base a componentes involucrados y casos de uso relacionados.

Para comprender en su totalidad cada requerimiento es necesaria una lectura previa de los documentos de Relevamiento de Requerimientos, Análisis de Requerimientos y Descripción de la Arquitectura.

Aquí se presentan en forma resumida a través del título y los casos de uso que lo componen, indicados mediante un nombre y una descripción introductoria.

3.4. Diagrama de Casos de Uso

Se presenta a continuación un diagrama con los casos de uso mas relevantes para este documento indicando la relación entre ellos y el actor bajo el rol de desarrollador.

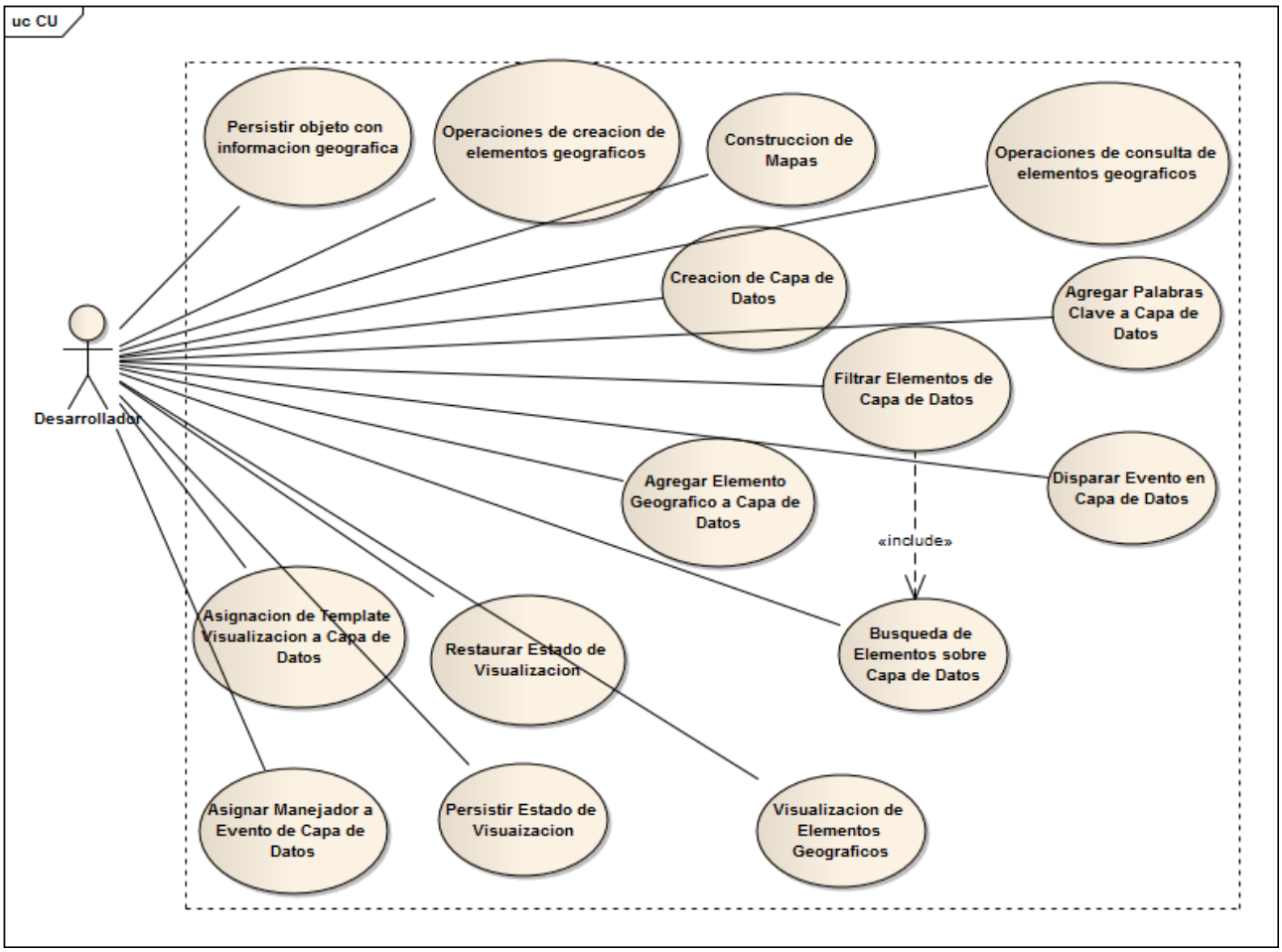

Figura 8 – Diagrama de Casos de Uso

3.4.1. Representación de tipos geográficos

Ver requerimiento R01 del documento de Análisis de Requerimientos.

3.4.1.1. Casos de uso

No tiene

3.4.2. Persistencia de tipos geográficos

Ver requerimiento R02 del documento de Análisis de Requerimientos.

3.4.2.1. Casos de uso

Nombre: Persistir objeto con información geográfica
Descripción: Este caso de uso permite al usuario persistir un objeto de dominio con atributos de tipo geográfico. Esta persistencia puede involucrar elementos de otros sistemas.

3.4.3. Operaciones básicas sobre elementos de tipo geográfico

Ver requerimiento R03 del documento de Análisis de Requerimientos.

3.4.3.1. Casos de uso

Nombre: Operaciones de creación de elementos geográficos.
Descripción: Este caso de uso permite al usuario operar con dos o más elementos de tipo geográfico para obtener un nuevo elemento geográfico. Las operaciones disponibles son: Intersección, unión y diferencia.

Nombre: Operaciones de consulta de elementos geográficos.
Descripción: Este caso de uso permite al usuario operar con dos o más elementos de tipo geográfico para obtener un valor booleano o escalar. Las operaciones disponibles son: distancia, área, pertenencia y equivalencia.

Nombre: Creación de herramienta UI
Descripción: Este caso de uso permite al usuario incluir una herramienta en la interfaz gráfica para realizar las consultas geográficas.

3.4.4. Construcción de mapas

Ver requerimiento R04 del documento de Análisis de Requerimientos.

3.4.4.1. Casos de uso

Nombre: Construcción de mapas.
Descripción: Este caso de uso permite al usuario incluir un mapa en la interfaz gráfica de acuerdo a ciertos parámetros de configuración entre los que figura el visualizador a utilizar.

3.4.5. Gestión de capas de datos

Ver requerimiento R04 del documento de Análisis de Requerimientos.

3.4.5.1. Casos de uso

Nombre: Creación de una capa de datos.
Descripción: Este caso de uso permite al usuario crear una capa de datos especificando el nombre, los atributos y las palabras claves que desee.

Nombre: Agregar palabras clave a una capa de datos existente.
Descripción: Este caso de uso permite al usuario agregar palabras clave a una capa de datos existente.

Nombre: Agregar atributos a una capa de datos existente.
Descripción: Este caso de uso permite al usuario agregar atributos a una capa de datos existente.

Nombre: Quitar palabras claves a una capa de datos existente.
Descripción: Este caso de uso permite al usuario quitar palabras clave de una capa de datos existente.

Nombre: Quitar atributos a una capa de datos existente.
Descripción: Este caso de uso permite al usuario quitar atributos de una capa de datos existente.

3.4.6. Gestión de elementos de las capas de datos

Ver requerimiento R06 del documento de Análisis de Requerimientos.

3.4.6.1. Casos de uso

Nombre: Agregar elemento geográfico a capa de datos
Descripción: Este caso de uso permite al usuario agregar un elemento con atributos geográficos a una capa de datos.

Nombre: Agregar elemento geográfico a múltiples capas de datos
Descripción: Este caso de uso permite al usuario agregar un elemento con atributos geográficos a varias capas de datos.

Nombre: Quitar elemento geográfico de una capa de datos
Descripción: Este caso de uso permite al usuario quitar un elemento con atributos geográficos de una capa de datos.

Nombre: Quitar elemento geográfico de múltiples capas de datos
Descripción: Este caso de uso permite al usuario quitar un elemento con atributos geográficos de una capa de datos.

3.4.7. Filtrar elementos de las capas de datos

Ver requerimiento R07 del documento de Análisis de Requerimientos.

3.4.7.1. Casos de uso

Nombre: Filtrar elementos de capa de datos
Descripción: Este caso de uso permite al usuario filtrar elementos con atributos geográficos especificando el criterio de filtrado y la acción que quiere llevar a cabo sobre los resultados (mostrar u ocultar).

Nombre: Aplicar acción personalizada a elementos del mapa

Descripción: Este caso de uso permite al usuario filtrar elementos con atributos geográficos especificando el criterio de filtrado y la acción que quiere llevar a cabo sobre los resultados.

Nombre: Acciones personalizadas sobre sobre elementos del mapa
Descripción: Este caso de uso permite al usuario incluir un menú de acciones posibles sobre elementos de un mapa en la interfaz gráfica.

3.4.8. Búsquedas de elementos sobre las capas de datos

Ver requerimiento R08 del documento de Análisis de Requerimientos.

3.4.8.1. Casos de uso

Nombre: Búsqueda de elementos sobre capas de datos
Descripción: Este caso de uso permite al usuario buscar elementos en una capa de datos de acuerdo a un criterio de búsqueda determinado.

Nombre: Búsqueda de elementos sobre múltiples capas de datos
Descripción: Este caso de uso permite al usuario buscar elementos en varias capas de datos de acuerdo a un criterio de búsqueda determinado.

Nombre: Búsqueda de elementos sobre capa de datos
Descripción: Este caso de uso permite al usuario buscar elementos en una o varias capas de datos en la interfaz gráfica, de acuerdo a un criterio de búsqueda determinado.

3.4.9. Eventos

Ver requerimiento R09 del documento de Análisis de Requerimientos.

3.4.9.1. Casos de uso

Nombre: Asignar manejador a evento de capa de datos

Descripción: Este caso de uso permite al usuario especificar manejadores de eventos de una capa de datos. Los eventos pueden ser predefinidos o personalizados.

Nombre: Disparar evento personalizado en capa de datos
Descripción: Este caso de uso permite al usuario disparar un evento personalizado sobre una capa de datos.

Nombre: Remover manejador de evento de capa de datos
Descripción: Este caso de uso permite al usuario remover manejadores asociados a una capa de datos.

Nombre: Asignar manejador a evento de objeto geográfico
Descripción: Este caso de uso permite al usuario especificar manejadores asociados a un elemento con atributos geográficos.

Nombre: Disparar evento personalizado en objeto geográfico
Descripción: Este caso de uso permite al usuario disparar un evento personalizado sobre un elemento con atributos geográficos.

Nombre: Remover manejador de evento de objeto geográfico
Descripción: Este caso de uso permite al usuario remover manejadores asociados a un elemento con atributos geográficos.

3.4.10. Visualización de capas de datos

Ver requerimiento R10 del documento de Análisis de Requerimientos.

3.4.10.1. Casos de uso

Nombre: Asignación template de visualización a capa de datos
Descripción: Este caso de uso permite al usuario especificar un template de visualización para elementos de cierto tipo para una capa de datos.

Nombre: Asignación template de visualización genérico para capas de datos
Descripción: Este caso de uso permite al usuario especificar un template de visualización por defecto para todos los tipos de elemento de una capa de datos.

Nombre: Visualización de capa de datos en mapa
Descripción: Este caso de uso permite al usuario elegir que capas quiere visualizar de un mapa mediante un herramienta en la interfaz gráfica.

Nombre: Persistir estado de visualización
Descripción: Este caso de uso permite al usuario persistir el estado de los filtros y herramientas graficas de forma permanente.

Nombre: Restaurar estado de visualización
Descripción: Este caso de uso permite al usuario restaura el estado de los filtros y herramientas graficas previamente persistido.

3.4.11. Visualización de elementos geográficos

Ver requerimiento R11 del documento de Análisis de Requerimientos.

3.4.11.1. Casos de uso

Nombre: Asignación template de visualización a elementos geográficos de capa de datos
Descripción: Este caso de uso permite al usuario especificar templates de visualización de un tipo de elemento en una capa de datos.

Nombre: Visualización de elementos geográficos
Descripción: Este caso de uso permite al usuario especificar los atributos de visualización a modificar. Los atributos disponibles son: icono, color y borde.

3.5. Arquitectura

Se define a continuación la arquitectura desde el punto de vista de distribución de componentes e interacción con sistemas externos. Se introduce en primera instancia el modelo conceptual del componente a través de los diferentes dominios y luego se explica la participación de cada dominio en los diferentes subsistemas.

3.5.1. Modelo de Dominio

Se describen a continuación las principales entidades participantes del componente YUPPGIS. Se consideran tres grupos disjuntos de acuerdo el rol que cumplen en el sistema.

3.5.1.1. Modelo de Capas de Datos

En este grupo se consideran aquellas entidades que permiten modelar las capas de datos como contenedores de objetos geográficos y su relación con los mapas.

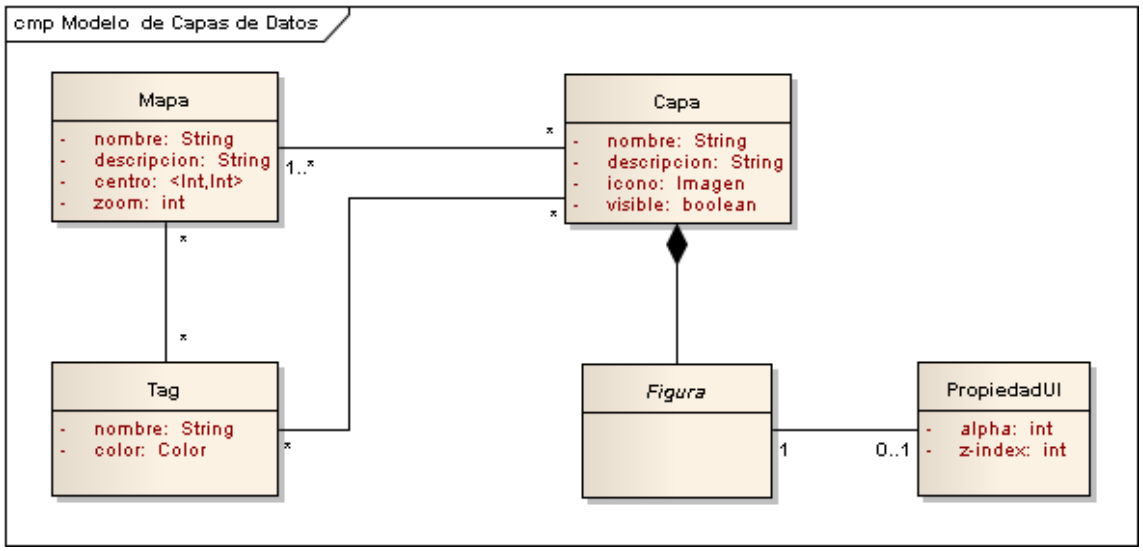

Figura 9 - Modelo de Capa de Datos

3.5.1.2. Modelo de figuras Geográficas

En este grupo se consideran aquellas entidades que permiten modelar figuras.

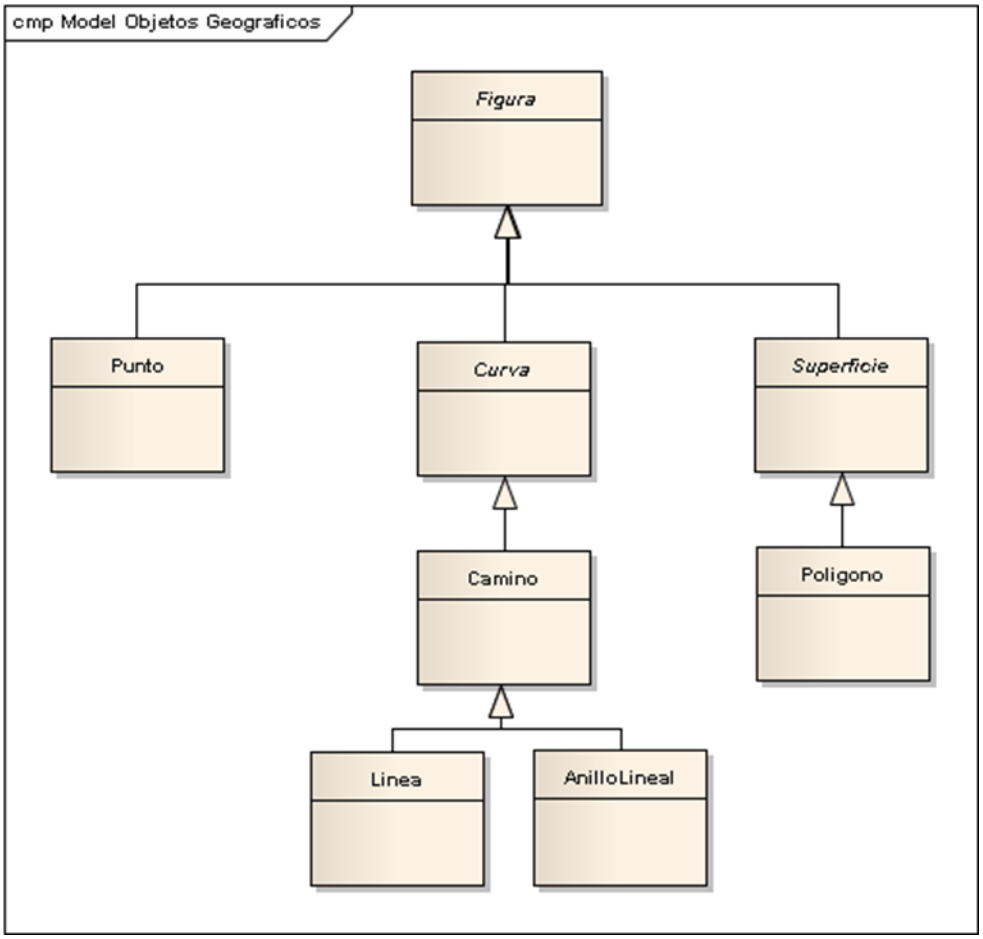

Figura 10 - Modelo de Figuras Geográficos

3.5.1.3. Modelo de Atributos Geográficos

En este grupo se consideran aquellas entidades que permiten modelar los diferentes atributos de visualización de los elementos geográficos vistos como figuras.

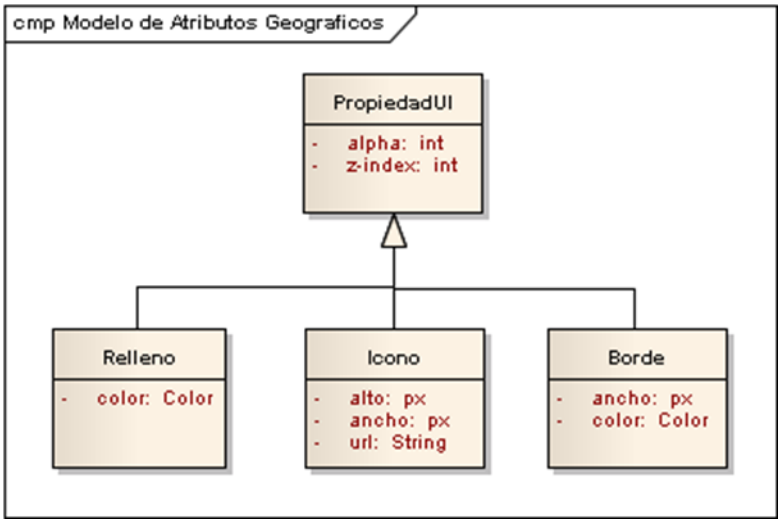

Figura 11 - Modelo de Atributos

3.5.2. Modelo de la Arquitectura YUPPGIS

Se describen los subsistemas participantes de la arquitectura del componente YUPPGIS en base al diagrama a continuación:

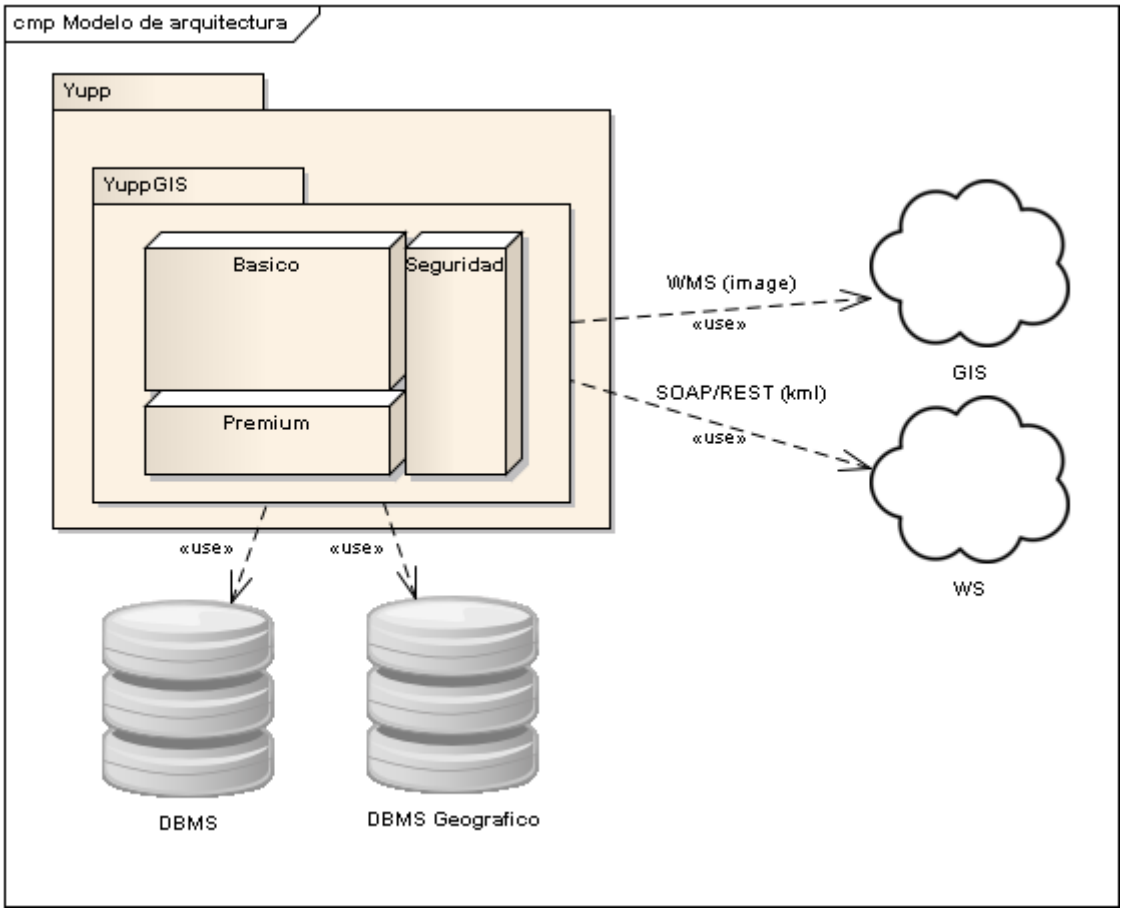

Figura 12 - Modelo de Arquitectura

YuppGis

Componente del framework Yupp, cuyo objetivo principal es permitir al programador desarrollar aplicaciones web geográficas de forma ágil. Este se comunica con un GIS a través de un servicio web WMS, posiblemente con un sistema externo a través de un webservice SOAP/REST y con un motor de base de datos Postgre con extensión PostGIS.

Este componente provee dos esquemas de funcionamiento:

Paquete Básico

Bajo este esquema de funcionamiento se soporta trabajar sin la extensión PostGIS. El sistema solamente almacena los identificadores de los objetos geográficos, proporcionados por algún sistema externo. Se restringen las funciones que son primitivas de la extensión PostGIS.

Paquete Premium

Este esquema de funcionamiento cumple con todas las funcionalidades del esquema básico, pero además permite contar con los datos geográficos de forma local y/o remota, así como con las operaciones restringidas en el esquema Básico.

GIS

Servidor de mapas encargado de generar imágenes a partir de una fuente de datos. Esta fuente de datos es una base de datos geográfica, la cual puede ser la misma que la base de datos de negocio que utiliza YuppGIS.

WS

Servicio web (SOAP/REST) externo, con el cual es posible comunicarse para obtener información geográfica en formato GML dada una petición establecida por el programador. Este servicio provee información para poder asociar un objeto del dominio de la aplicación con un objeto geográfico, siendo este el resultado obtenido de consultar al servicio web externo.

PostGis

Servidor de base de datos Postgre con extensión PostGIS, cuyo principal objetivo es almacenar información geográfica utilizada por YuppGIS.

3.5.3. Diseño de la Arquitectura YUPPGIS

Se describe la organización del framework incluyendo al componente YUPPGIS en base al diagrama a continuación:

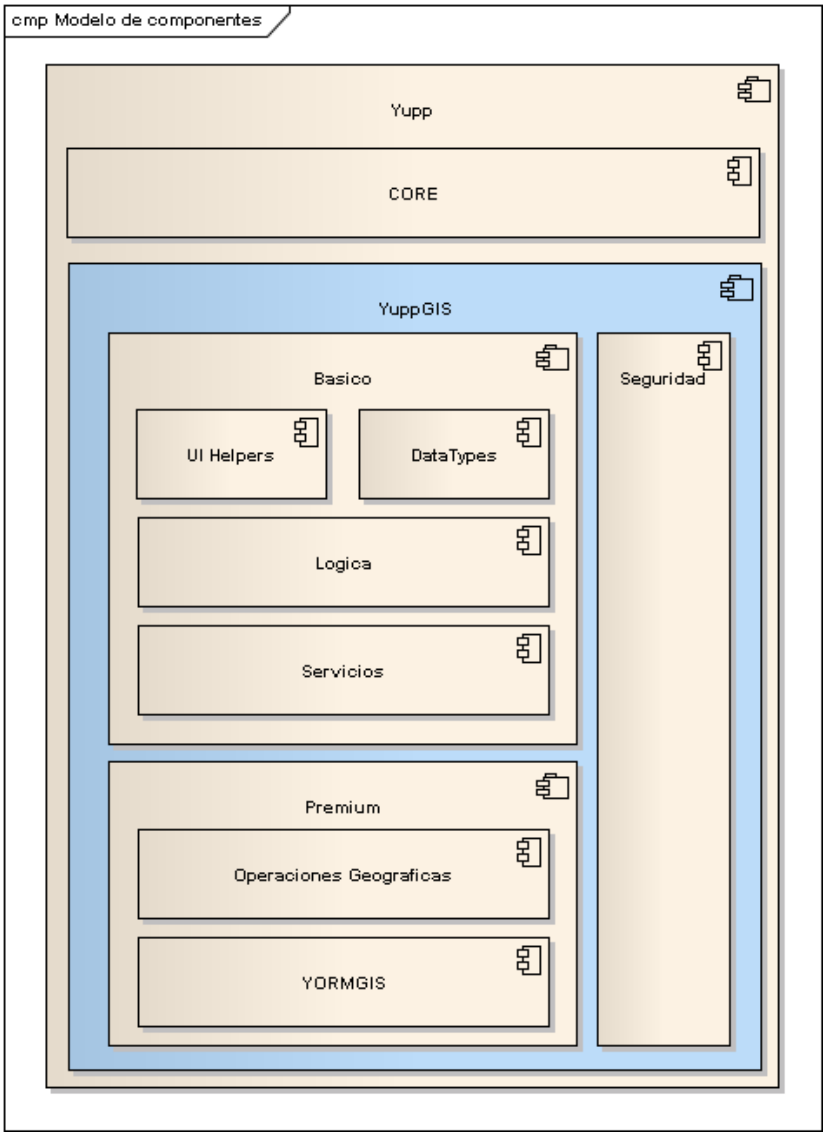

Figura 13 - Modelo de Componentes

Paquete básico

UI Helpers

Este componente consta de un conjunto de UI Helpers, los cuales permiten generar código HTML a partir de su invocación por parte del programador.

DataTypes

Este componente está conformado por contenedores de datos que son utilizados por YuppGIS para modelar los objetos geográficos. Cuenta con las definiciones de representación de Línea, Punto, Curva, Polígono, etc.

Lógica

Núcleo del componente, compuesto por controladores predefinidos y herramientas para administrar y manipular objetos geográficos.

Servicio

Este componente es el encargado de consumir servicios e interpretar los datos extraídos de los mismos. En particular, el framework YuppGIS lo utiliza para consumir servicios WMS expuestos por un GIS.

Paquete Premium

Operaciones Geográficas

Este componente es el encargado de brindar las operaciones geográficas que soporta el framework. Dichas operaciones son ejecutadas sobre una base de datos espacial.

YORMGIS

Este componente brinda funciones ORM siendo el encargado de mapear objetos del modelo de dominio junto con sus atributos geográficos a su equivalente en base de datos relacional. Permite la cargar y persistir dichos objetos desde/hacia la base de datos.

4. Implementación

4.1. Introducción

En este capítulo se describe en detalle la implementación de los componentes y subsistemas especificados en el modelo de diseño así como los requerimientos de configuración del sistema y su instalación. Se explica la solución a cada uno de los problemas a resolver y se introducen las decisiones tomadas en cada caso. Como complemento se brinda un documento anexo “Manual de Usuario” con mayor profundidad técnica que la aquí presentada.

4.2. Solución propuesta

4.2.1. Introducción

La solución considera los requerimientos explicados anteriormente, abarcando las diferentes interacciones entre los diferentes componentes, las posibilidades de visualización y los requerimientos no funcionales. Se proveen mecanismos de implantación del sistema en diversos ambientes, herramientas de configuración de las diversas opciones y posibilita la extensión de cada subsistema. Además deja en claro el rol que cumple cada componente en base a su ubicación en el sistema desarrollado y en base al ambiente en que se ejecuta.

4.2.2. Características de la solución

4.2.2.1. Esquemas de funcionamiento

El componente desarrollado provee dos esquemas de funcionamiento: Paquete Básico y Paquete Premium. Estos esquemas modelan dos tipos de realidades, la primera, cuando se cuenta con un motor de base de datos con extensión de datos geográficos, y la segunda, cuando se cuenta con un repositorio externo de datos geográficos. Dichos esquemas establecen un diseño que sigue con los lineamientos base de Yupp, estableciendo un práctico y sencillo uso de los mismos por parte del programador.

Paquete Básico

Este esquema de trabajo aplica en aquellos casos en que se cuenta con un repositorio para datos geográficos. El acceso a dicho repositorio es configurable en su totalidad por el desarrollador, permitiendo además, la construcción de un conector para tal fin. De esta forma es posible manipular datos geográficos desde un archivo, un servicio web REST, SOAP, u otros.

El paquete Básico provee un diseño que facilita el consumo de servicios web geográficos, se provee un conector por defecto, el cual consume servicios web REST que soportan KML como modelo de representación de datos geográficos.

Paquete Premium

Este esquema de trabajo aplica en aquellos casos en que se cuenta con un motor de base de datos con extensión geográfica. Bajo esta realidad se brindan al programador funcionalidades de filtrado y de creación de datos geográficos soportados por YuppGIS.

El Paquete Premium provee dos conectores de acceso a base de datos diferentes, el primero para el modelo de datos de la aplicación a desarrollar y el segundo para almacenar datos geográficos. Permite de esta forma mayor flexibilidad a la hora de trabajar con servidores de datos geográficos.

4.2.2.2. Modelos de datos

YuppGIS brinda un modelo de dominio reducido para la representación de los conceptos Mapa, Capa de datos y Etiqueta (*Tag*). Dicho modelo está diseñado para acoplarse con el modelo de dominio definido por el programador en cada aplicación.

4.2.2.3. YORMGIS

Este componente brinda funciones ORM siendo el encargado de mapear objetos del modelo de dominio, junto con sus atributos geográficos, a su equivalente en base de datos relacional. Su implementación está basada en YORM.

Su funcionalidad se divide en dos grandes aspectos:

Mapeo

Los objetos que extienden la clase GISPersistentObject son identificados por YORMGIS para realizar el mapeo correspondiente. Al ser YORMGIS una extensión de YORM, el mapeo de objetos a base de datos relacional se realiza de igual manera en todos los casos.

Un atributo geográfico se considera como una relación simple entre el objeto y el tipo del atributo geográfico, lo cual permite abstraerse del esquema de trabajo (mecanismo de almacenamiento), ya sea Paquete Premium o Paquete Básico.

En el esquema de Paquete Premium esta asociación se representa por medio de una tabla extra a la tabla del objeto, cuyo nombre debe seguir la convención declarada por YuppGIS. La convención define el formato del nombre como la concatenación entre el nombre de la tabla del objeto, el nombre del atributo y el sufijo “geo”. Dicha tabla consta de un identificador, la figura geográfica representada en WKB y las propiedades de visualización de la figura (UIProperty) representadas en formato JSON. La conversión de figura geográfica a su representación en base de datos, se realiza por medio del estándar WKT, soportado por los motores de base de datos con los cuales trabaja YuppGIS.

En cambio, en el esquema de Paquete Básico la asociación la define el programador a través de un conector a su repositorio de datos geográficos.

Consultas

En el esquema de trabajo de Paquete Premium, donde se cuenta con un motor de base de datos con extensión geográfica, se define un módulo cuyo objetivo es el procesamiento y ejecución de consultas SQL.

YuppGIS cuenta con un modelo de consultas sobre datos, el cual le ofrece al programador la generación de consultas orientadas a objetos del dominio, basadas en propiedades del ORM. De esta forma el programador se desentiende de la construcción de consultas SQL, permitiendo así la generación de consultas complejas sacando provecho de su modelo de dominio.

Dada la flexibilidad que brinda YuppGIS, bajo el esquema de Paquete Premium, de trabajar con dos conexiones a base de datos diferentes (geográfica y no geográfica), se implementa un procesador de consultas orientadas a objetos del dominio, capaz de trabajar con dichas conexiones. Uno de los principales problemas del procesador es realizar *joins* entre tablas diferentes en bases de datos diferentes y en conexiones diferentes. El procesador soluciona esto, de acuerdo a la cantidad de conexiones disponibles, ejecutando dichos *joins* en memoria y/o delegando su ejecución al motor de base de datos.

Para mayor información acerca del funcionamiento e implementación de este motor de consultas se recomienda la lectura del anexo respectivo.

4.2.3. Interfaz de usuario

YuppGIS provee un conjunto de Helpers que facilitan la visualización de mapas y filtrado a nivel de vista, basado en un plugin *jQuery* que modela el mapa y su comportamiento (para una descripción exhaustiva de las características de implementación y operaciones soportadas por este plugin revisar el anexo correspondiente).

Se destaca el Helper de visualización de mapa, el cual permite incluir un mapa en una vista de forma sencilla y en una línea de código utilizando *OpenLayers* como visualizador web. El mismo es totalmente parametrizable permitiéndole al programador una mayor posibilidad de personalización, como ser selección de tipo de capa base (*MapServer* o *Google Maps*).

Otro Helper disponible, es el que permite generar un filtrado de objetos de un mapa, siendo las opciones de filtro los atributos de los propios objetos del dominio. Estas opciones se generan de forma totalmente dinámica a través de la búsqueda de atributos de un objeto por medio de *Reflection.*

Una característica a destacar de YuppGIS, es la definición de templates personalizables por el programador a la hora de visualizar el detalle de un objeto contenido en un mapa. Los templates se definen mediante una convención heredada de Yupp, la cual consiste en que el nombre de la vista finalice con la palabra "template". A su vez YuppGIS extiende dicha convención, haciéndola más flexible, permitiendo definir templates asociados a capas de datos o a objetos de dominio, por medio de su nombre. El nombre de estos templates debe ser la concatenación del nombre del objeto y el nombre de la capa de datos o simplemente el nombre del objeto en cuestión.

4.2.4. Consideraciones generales

Servidor de Mapas

YuppGIS soporta como servidores de mapas *MapServer* y *Google Maps*, siendo el segundo más práctico a la hora de programar una aplicación. Sin embargo YuppGIS contempla y soluciona varias de las dificultades a la hora de utilizar *MapServer* como servidor de mapas, definiendo la interacción con el mismo por medio del estándar de servicio web geográfico de la OGC, WMS.

Para el desarrollo del requerimiento de seguridad a nivel de visualización de capas de un mapa, YuppGIS implementa una capa de indirección entre el visualizador web (*OpenLayers*) y el servidor de mapas (*MapServer*), la cual permite restringir el acceso a los recursos solicitados.

Serialización de objetos geográficos

Para la serialización y transporte de objetos geográficos, así como capas de datos, se utiliza el lenguaje de marcas KML. Debido su característica de definición de estilos y su simpleza, KML es el lenguaje apropiado para dicha serialización.

Eventos

YuppGIS proporciona un conjunto de herramientas para el manejo de eventos a nivel de objetos del modelo de dominio, permitiendo la notificación de eventos entre objetos. El framework detecta los elementos a observar mediante la implementación del patrón de diseño *Observer* [25] [26].

4.2.5. Configuración

Un aspecto importante de YuppGIS es la disponibilidad de personalizar la mayoría de sus características a través de un archivo de configuración, bajo el nombre de "*yuppgis_config.php"*.

Dentro de las características personalizables se encuentra el esquema de trabajo (Paquete Básico o Paquete Premium), la proyección a utilizar (SRID) para los atributos geográficos, clave para el uso de *Google Maps* y propiedades de configuración de *MapServer*, tales como capas a mostrar, URL del servidor y ubicación del archivo *Mapfile*.

4.2.5.1. Paquete Básico

Las posibles configuraciones del esquema son: el nombre del conector al repositorio, y en caso de utilizar el conector que brinda YuppGIS por defecto, se debe especificar la URL al repositorio.

4.2.5.2. Paquete Premium

En este esquema de trabajo se puede configurar la conexión a la base de datos geográfica, de forma análoga a la configuración de conexiones a bases de datos a nivel de aplicación en Yupp.

YuppGIS brinda como funcionalidad extra la generación de tablas, ya sea para la base de datos de la aplicación o la base de datos geográfica.

5. Caso de Estudio

5.1. Introducción

Uno de los objetivos del proyecto es el desarrollo de una aplicación utilizando el componente desarrollado con el fin de demostrar el potencial del mismo (caso de estudio). Para ello es necesario comparar el costo de desarrollo entre dicha aplicación y su análoga desarrollada sin YuppGIS.

Es de interés para el cliente comparar en base a una aplicación real, desarrollada en Yupp, la cual fue presentada en el "*I CONGRESO URUGUAYO DE INFRASTRUCTURA DE DATOS ESPACIALES*" con el nombre de "*Monitoreo y Control de Problemas de Salud mediante SIG*" [27].

El caso de estudio consiste en implementar un sistema de monitoreo y control de salud incorporando interacción con un sistema de información geográfica. Se considera que incluir información geo-referenciada puede ser de gran utilidad en el monitoreo y control de enfermedades en la población, contribuyendo a detectar problemas de forma temprana.

5.2. Características del sistema

Se plantea un subconjunto de los requerimientos originales de la aplicación "*Monitoreo y Control de Problemas de Salud mediante SIG*":

1. Mostrar pacientes en un mapa en base a la dirección de cada uno de ellos. Con esto se pretende visualizar a los mismos considerando la afiliación a una determinada institución asistencial, y a simple vista poder tener información de la concentración y distribución de la población a la que se atiende.
2. Cruzar información geográfica con información demográfica para enriquecer la información referente a la población cubierta al mostrar diferentes datos personales, tales como edad, sexo, datos de contacto, etc.
3. Cruzar información geográfica y demográfica con información clínica, en principio considerando enfermedades crónicas como hipertensión arterial, obesidad, asma, diabetes e insuficiencia renal.

A partir de estos requerimientos se construye un prototipo funcional utilizando YuppGIS, el cual permite:

- Mostrar la ubicación de los hogares de pacientes en un mapa.
- Acceder a información demográfica de cada paciente (nombres, apellidos, fecha de nacimiento, sexo, información de contacto y ubicación geográfica).
- Acceder a información clínica (problemas de salud y estado de control de cada uno).

Otras funcionalidades que permiten una mejor visualización de la información:

- Poder agrupar personas que viven en la misma ubicación (núcleos familiares).
- Filtros de visualización de pacientes por problemas de salud y/o nivel de control.

Dichas funcionalidades le permiten a un médico monitorear a sus pacientes, ubicando cada vivienda sobre el mapa, junto a las enfermedades de los pacientes que componen el núcleo familiar y el correspondiente nivel de control. El médico debe poder acceder a información de contacto para comunicarse con cada paciente, en caso de ser necesario.

De este modo, se permite que el médico de referencia pueda visualizar la información de cada uno de sus pacientes, tanto individualmente o en núcleos familiares. Esta capacidad de visualización por núcleos fami-

liares es de gran valor clínico, ya que permite que se puedan detectar problemas de salud que estén relacionados por las características de cada núcleo.

El prototipo desarrollado incluye una lista de cinco problemas de salud representados mediante iconografía sobre el mapa permitiendo destacar las enfermedades en el mismo. Para denotar el nivel de control de cada enfermedad se utiliza un código de tres colores, a modo de semáforo, donde “verde” significa que la enfermedad se encuentra controlada, “amarillo” significa que no se corre un riesgo inmediato y “rojo” significa que el paciente se debe controlar de forma urgente.

A continuación se presenta un diagrama de clases que representa las entidades en el prototipo de caso de estudio.

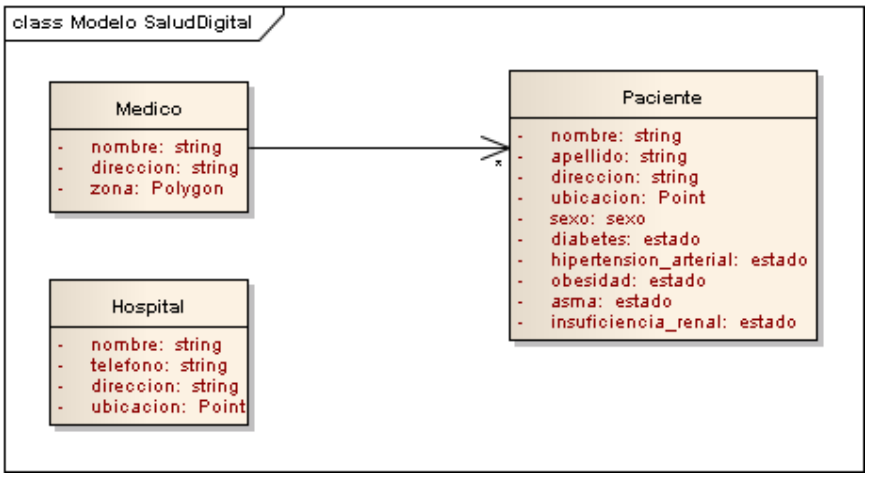

Figura 14 - Modelo de clases del caso de estudio

Se reutilizan los conceptos Capa de Datos (DataLayer) y Mapa (Map) de YuppGIS para la representación de enfermedades y mapas (de pacientes y médicos) respectivamente. Para el modelado de enfermedades, se decide representar cada enfermedad como una capa de datos, de forma de agrupar los pacientes según su enfermedad y utilizar los Helpers proporcionados por YuppGIS.

El caso de estudio está desarrollado bajo el esquema de trabajo Premium de YuppGIS, con un motor de base de datos *PostGIS*, utilizando ambas opciones de capa base para la visualización de mapas (*Google Maps* y *MapServer*) seleccionada a través del Helper correspondiente.

La implementación del caso de estudio requirió la construcción de un sistema autónomo cuyo objetivo es la geolocalización de un padrón en Montevideo, dado un nombre de calle y numero de puerta, el sistema retorna las coordenadas correspondientes al punto. Este sistema provee servicios REST para brindar las funcionalidades de: obtención de calles según un nombre y ubicación de padrones según un nombre de calle y número de puerta. Como fuente de datos se contó con información geográfica ofrecida por la *Intendencia Municipal de Montevideo* a través de su página web [28].

A continuación se muestra el diagrama de componentes para el caso de estudio, donde se puede observar los componentes a los sistemas del caso de estudio y del encargado de la geolocalización.

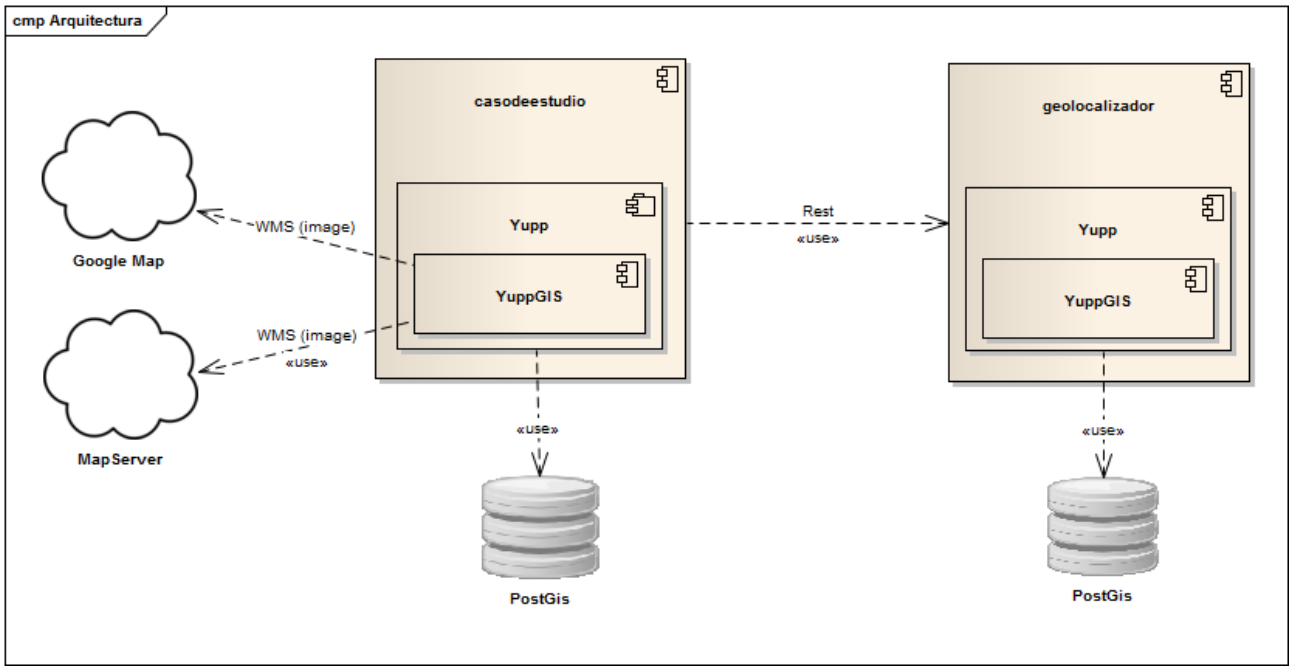

Figura 15 – Diagrama de la solución implementada para el caso de estudio.

A continuación se presentan dos capturas de pantalla del caso de estudio, la primera correspondiente a la zona de acción de un médico y la segunda corresponde a la página principal de la aplicación web.

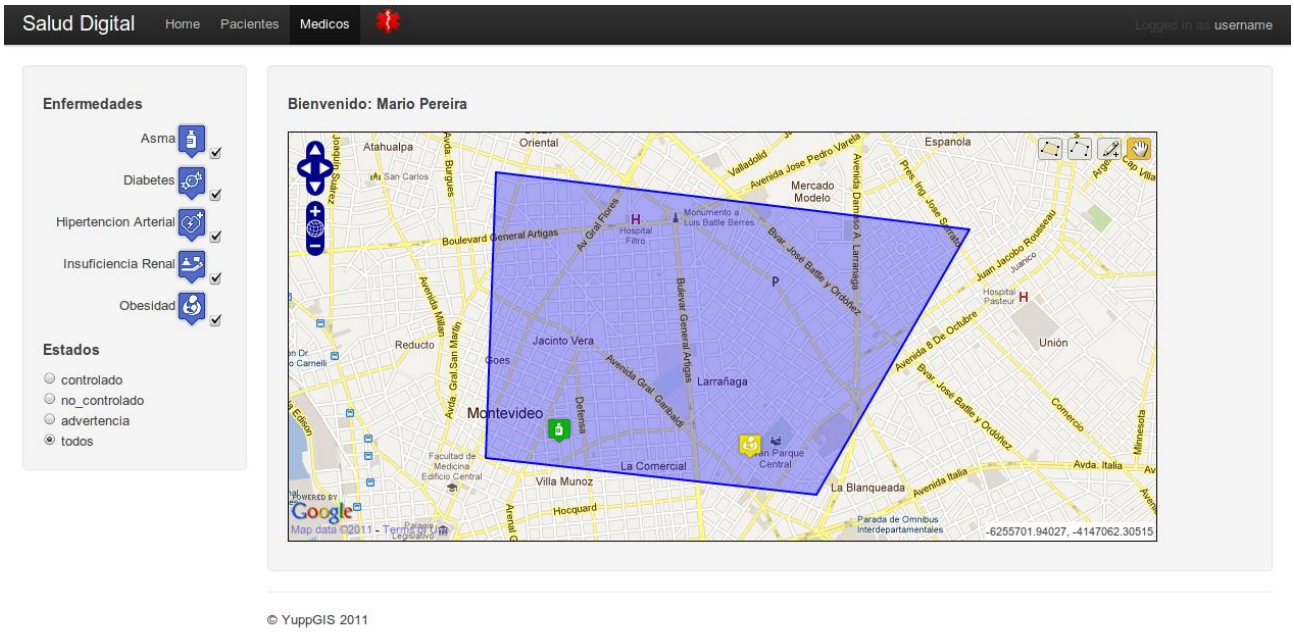

Figura 16 - Captura de pantalla de sección de un Medico.

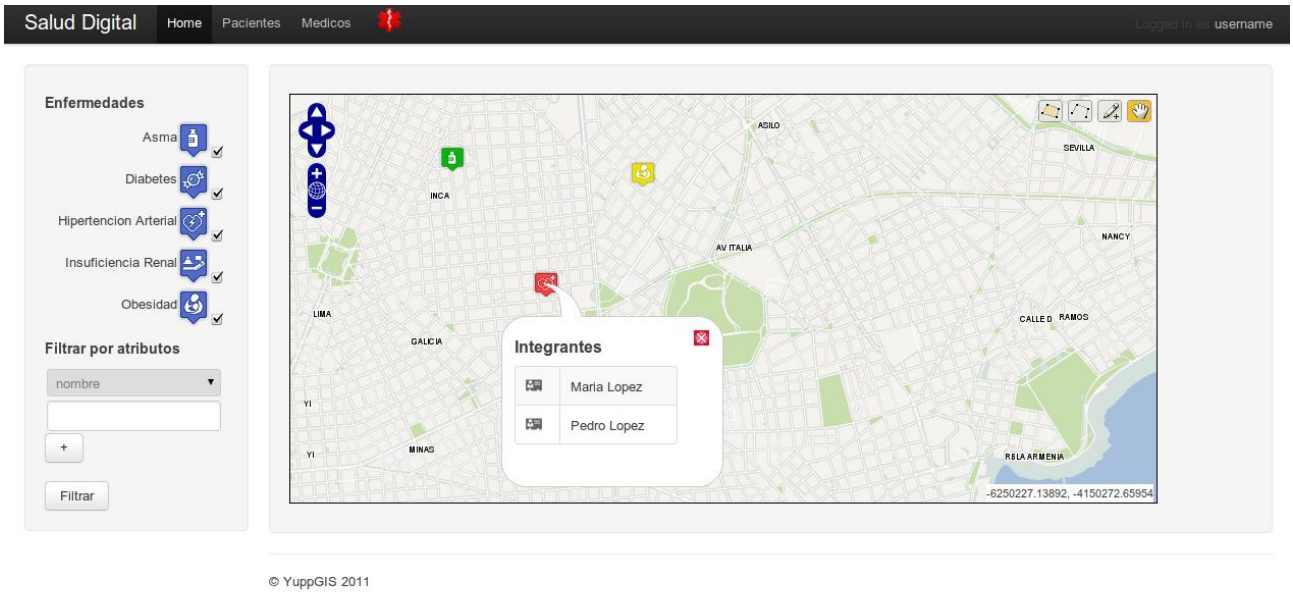

Figura 17 - Captura de pantalla de página principal.

5.3. Comparación

Durante el desarrollo del caso de estudio fue posible percibir la facilidad de creación y uso de información geo-referenciada en vistas a través de la visualización de mapas y capas de datos, así como elementos del dominio del problema presentando atributos de tipo geográfico.

Se lograron implementar las funcionalidades básicas con bajo costo de desarrollo, medido en cantidad de código escrito y tiempo requerido. Hay que tener en cuenta que en la versión original de este caso de estudio el cliente requirió de varios meses para lograr obtener un producto completamente funcional. Se destaca que una de las grandes diferencias entre dichos proyectos es la generacion de capas de datos o figuras sobre el mapa, en la version del proyecto basado solo en Yupp, se utiliza solamente MapServer para la generacion de una capa base con un conjunto de capas de datos o figuras, en cambio, en la aplicación basada en YuppGIS se utiliza simplemente MapServer para la generacion de capa base, siendo YuppGIS el que se encarga de generar las capas de datos en formato KML. De esta manera, se ahorra complejidad a la hora de configurar el servidor de mapas MapServer, y se gana flexibilidad a la hora de generar nueva capas de datos.

Se muestra a través del uso de la aplicación construida que las consultas se realizan en tiempos acordes a la necesidad del usuario, y las complejas manipulaciones geográficas no plantean demoras.

Como agregado se implementaron funcionalidades relativas a la experiencia de usuario, considerando usabilidad y aspecto, así como amigabilidad del sistema al momento de interactuar con las diferentes opciones visuales, lo cual funciona como un plus por sobre la aplicación original.

Las siguientes imágenes presentan una comparativa entre la aplicación original y el prototipo construido mediante el software CLOC [29] que utiliza como medida de comparación las líneas de código escritas:

```
> find  mapfiles/. \
>       Yupp/components/georreferencia/controllers/. \
>       Yupp/components/georreferencia/config/. \
>       Yupp/components/georreferencia/views/. \
>       Yupp/components/georreferencia/model/. | xargs cloc
      99 text files.
      32 unique files.
       3 files ignored.

http://cloc.sourceforge.net v 1.53  T=0.5 s (62.0 files/s, 9152.0 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
PHP                             30            545            196           3681
XML                              1              1             62             91
-------------------------------------------------------------------------------
SUM:                            31            546            258           3772
-------------------------------------------------------------------------------
```

Figura 18 - Resumen de cantidad de líneas obtenidas mediante CLOC del caso de estudio.

En la imagen anterior tomamos como relevante la información relacionada a cantidad de archivos y líneas de código escritas en PHP. Vemos ahora las estadísticas del caso de estudio implementado:

```
> find casodeestudio/. | xargs cloc
     200 text files.
      67 unique files.
     288 files ignored.

http://cloc.sourceforge.net v 1.53  T=0.5 s (58.0 files/s, 5838.0 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
PHP                             22            313             52           1756
CSS                              5             31             61            677
XML                              1              0              0             16
Javascript                       1              0             10              3
-------------------------------------------------------------------------------
SUM:                            29            344            123           2452
-------------------------------------------------------------------------------
```

Figura 19 - Resumen de cantidad de líneas obtenidas mediante CLOC del caso de estudio basado en YuppGIS.

Nuevamente tomamos como relevante la información relacionada al código escrito en PHP y notamos una importante disminución en el número de líneas respecto a la implementación original. Se descartan las estadísticas relacionadas a la información de estilo y datos de configuración.

Vemos ahora información comparativa de la implementación necesaria para geolocalizar datos en un mapa en base a la dirección:

```
> find FiltroCallesSvc.php ObtenerCoordenadasSvc.php \
>      FuncionesServicios.php database.php Cargador.php \
>   | xargs cloc
       5 text files.
       5 unique files.
       0 files ignored.

http://cloc.sourceforge.net v 1.53  T=0.5 s (10.0 files/s, 1164.0 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
PHP                              5            106             40            436
-------------------------------------------------------------------------------
SUM:                             5            106             40            436
-------------------------------------------------------------------------------
```

Figura 20 – Resumen de cantidad de líneas obtenidas mediante CLOC de la aplicación para geolocalización basada en Yupp.

En la imagen anterior observamos que solo se escribió código PHP para lograr el objetivo, por lo que esto constituye la totalidad de la información relevante para la comparación.

A continuación se presenta la estadística de la aplicación auxiliar para el caso de estudio basado en YuppGIS:

```
> find geolocalizacion/. | xargs cloc
      57 text files.
      34 unique files.
     140 files ignored.

http://cloc.sourceforge.net v 1.53  T=0.5 s (14.0 files/s, 290.0 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
PHP                              5             29              0             93
XML                              1              0              0             16
SQL                              1              1              2              4
-------------------------------------------------------------------------------
SUM:                             7             30              2            113
-------------------------------------------------------------------------------
```

Figura 21 - Resumen de cantidad de líneas obtenidas mediante CLOC de la aplicación para geolocalización basada en YuppGIS.

La implementación necesaria para geolocalizar datos en un mapa en base a la dirección también requiere un menor número de líneas de código escritas en PHP (se disminuye en un 80% en lo que a líneas se refiere). Esta disminución drástica de líneas de código se explica en base a la resolución de operaciones geográficas a través de los servidores de mapas de datos y a la infraestructura brindada por YuppGIS para operar con estos, lo que redunda en tener que escribir muy pocas líneas de código para realizar operaciones complejas de intersección de elementos y proyección de datos.

6. Conclusiones y Trabajo Futuro

6.1. Introducción

En este capítulo se enumeran las contribuciones del presente trabajo así como las posibles líneas de extensión y desarrollo de los diferente aspectos de la solución propuesta. Se realiza además una enumeración de logros obtenidos y dificultades sorteadas a la hora de implementar el componente diseñado.

6.2. Conclusiones

6.2.1. Desarrollo del Proyecto

El proyecto para la creación del componente YuppGIS se destaca por los objetivos cumplidos.

En primera instancia se planteó la problemática de comprender un framework desconocido y un lenguaje poco habitual para los integrantes de este equipo. La problemática de entender cómo utilizar y mejorar el framework en su versión actual acaparó los primeros meses de estudio y permitió generar un conocimiento profundo de la herramienta de forma de dar los pasos en la dirección correcta una vez comenzada la implementación.

Es posible notar a través del cronograma cómo cada etapa que comenzaba destinaba un tiempo menor en el logro de sus objetivos, y donde cada una de ellas se basaba en la anterior para construir elementos que a su vez fueran clave en la etapa siguiente. De esta forma, una vez obtenida la base del estudio del framework y del estado del arte, se pudo construir los subsistemas necesarios para conformar la base del componente final.

Para mitigar los riesgos tecnológicos fue importante la preparación de un prototipo funcional e incremental, que permitió ir incorporando las diferentes herramientas que se generaban en la implementación de forma de testear su correcto funcionamiento y facilidad de uso. Este prototipo permitió a su vez comprobar las diferentes posibilidades tecnológicas y asistir así en la toma de decisiones en este sentido.

Nos resultó fundamental el desarrollo de tests unitarios que funcionaran como una base de comprobación de las funcionalidades más complejas, y permitieran el continuo mejoramiento de código teniendo como respaldo la aceptación de dichos tests.

Se debe destacar además la ventaja de haber construido un ambiente de desarrollo homogéneo y compartido entre todos los integrantes del equipo, lo que permitió que cada uno dispusiera de un entorno preconfigurado y sincronizado con el de los demás. Esta decisión tomada al comienzo del proyecto, mitigó el riesgo de inconsistencias en cuanto a versiones de las herramientas de desarrollo y del propio código implementado, y mejoró notablemente la capacidad de paralelizar las diferentes tareas.

6.2.2. Logros obtenidos

La herramienta permite en su versión final integrar datos de diversas fuentes, lo que en un principio significó un obstáculo difícil de superar. Solo basta mencionar el hecho de que diferentes bases de datos geográficas tienen los datos en proyecciones distintas, por lo cual es necesario homogeneizar dichos valores para poder reutilizar un mismo juego de datos en diferentes visualizadores.

Se resalta también la importancia de los helpers de UI, que permiten al desarrollador interesado en incorporar datos geográficos a su aplicación disponer de filtros, listas, búsquedas, mapas y atributos de visualización de elementos sobre estos últimos de forma sencilla y sin la necesidad de tener que investigar al respecto. Y aunque el desarrollador lo ignore, todas estas construcciones se realizan en base a los estándares

manejados por los expertos, mediantes sistemas de traducción y conversión de datos, y mediante complejas consultas geográficas.

La persistencia del estado de visualización no es un tema menor, y es de destacar que se logró cumplir con el requerimiento de mantener los valores de los controles relacionados a un mapa y los elementos visibles en el mismo a pesar de sesiones y estados de la aplicación que pudiera implementar el desarrollador.

Como punto a destacar se puede mencionar la inclusión de patrones de diseño a la hora de solucionar los problemas de tipo arquitectónico, de manera de conseguir una implementación limpia y genérica para cada uno de los puntos neurales de este componente. Se puede mencionar entre otros la inclusión de una implementación propia del patrón *Observer* para manejar el requerimiento de tener eventos sobre elementos geográficos y sobre capas de datos, que a la vez queda disponible de forma de poder utilizarse en el futuro para cualquier tipo de datos que se quiera implementar.

Otro aspecto importante, y en el que hicimos especial énfasis cuando se pensó en la solución, fue que los diferentes componentes se integraran de forma natural al framework de forma de no alterar el comportamiento de aplicaciones pre-existentes. Para lograr este objetivo se tomó como prioritario el hecho de no modificar el núcleo del framework y manejar los diferentes componentes de la arquitectura desarrollada como complementos instalables por separado. Esto resultó en jerarquías de clases que permitieron heredar comportamiento y agregar nuevas funcionalidades.

Una dificultad encontrada en el camino fue la ausencia de documentación respecto al diseño y arquitectura del framework, por lo que se debió utilizar parte del tiempo disponible en generar estos documentos. Estos cumplieron una labor importante a la hora de atacar la construcción de los subsistemas desarrollados sin alterar el comportamiento existente y cumplirán una labor importante a la hora de plantearse extensiones y mantenimiento.

En cuanto a los tipos de datos utilizados, en particular los geográficos, la forma en que fueron pensados e implementados permite una fácil extensión en caso de necesitar nuevas clases, así como la posibilidad de extender fácilmente helpers y demás herramientas relacionadas (ya sean filtros, controles de búsqueda, etc.) sin preocuparse por la lógica inherente a su carácter de geográficos. Y en el caso de los no geográficos también cabe destacar que las nuevas clases implementadas guardan esta propiedad de ser fácilmente extensibles y heredables, por lo que se podría, por ejemplo, disponer de nuevas clases para representar atributos visuales en el mapa de forma rápida y sencilla.

Desde el punto de vista del usuario del framework, se pensó en la necesidad de facilitar la instalación del componente desarrollado. Para ello se utilizó la mecánica disponible de generación de tablas en bases de datos, pero con el agregado de que estas ahora pueden relacionarse con otras tablas de tipos geográfico, con lo que se logra mantener la experiencia de utilización del framework.

Por último mencionamos la construcción del caso de estudio, que permitió utilizar las herramientas desarrolladas en casos de uso reales, y obtener resultados comparativos respecto a una implementación previa de estas funcionalidades. Se pudo observar que cada requerimiento fue sencillo de implementar apoyándose en los helpers de UI para mostrar los mapas con la información, y que los requerimientos de búsqueda y filtrado fueron implementados utilizando las nuevas consultas geográficas de forma rápida y directa.

6.3. Trabajos Futuros

Una de las características fundamentales del componente desarrollado se basa en la visualización de datos geográficos en un mapa. En particular en este proyecto se utilizó el visualizador OpenLayers dado que brindaba la totalidad de las funcionalidades requeridas (y mas), aunque es de notar que se podría haber elegido otra opción, y por eso se plantea como posible extensión la habilidad de soportar otros tipos de visualizadores.

Como elemento fundamental de la interfaz gráfica se construyó un plugin para representar el mapa y sus funcionalidades, y se optó por realizarlo en el lenguaje *jQuery* [30] dado el conocimiento previo que de este se tenía. Es importante resaltar que el framework dispone de soporte para la biblioteca *Prototype*, por lo que sería fundamental permitir alternar entre ambas también en este contexto.

Los aspectos visuales se basaron principalmente en lo que el cliente requería y en el caso de estudio a construir, pero dado el estándar *kml* se pudo haber utilizado mayor cantidad de atributos gráficos para los elementos a mostrar en los mapas. De forma que también se propone como trabajo futuro enriquecer en este sentido el componente desarrollado.

Queda pendiente también disponer de mayor soporte para filtros de seguridad, permitiendo de manera sencilla construir reglas de autenticación que permitan restringir el acceso a determinado nivel de visualización de mapas (por ejemplo no poder establecer determinado zoom o región en un mapa determinado). De cualquier manera, el software construido brinda una capa de seguridad mínima que permite filtrar cada pedido y procesarlo, por lo que no deberían plantearse mayores dificultades a la hora de extenderlo.

Respecto al soporte de motores de bases de datos geográficas, actualmente disponemos de wrappers de conexión para PostgreSQL con extensión PostGIS aunque es extremadamente sencillo extender este soporte a MySql Spatial y SQLite. Dado que el framework soporta estos últimos, queda planteado para el futuro el desafío de construir los wrappers adecuados.

Otro trabajo a futuro, es la extensión e inclusión de propiedades y funciones geográficas dentro del modelo de datos geográficos, tales como validación de propiedades de la figura (por ejemplo: ¿es cerrada?) y obtención de propiedades sin necesidad de ir al motor de base de datos (por ejemplo: área, distancia, etc.). En principio, YuppGIS propone que este tipo de funciones estén disponibles solo bajo el esquema Premium, como trabajo a futuro es extender su funcionamiento al esquema Básico.

7. Glosario

Conector

Clase que encapsula la lógica para acceder a los servicios de un subsistema.

PHP

Lenguaje de programación pensado para la creación de páginas web dinámicas.

YUPP

Framework, escrito en PHP, que permite el desarrollo ágil de aplicaciones web.

ORM

Object Relational Mapping, estrategia que permite persistir objetos del dominio en una base de datos relacional.

YORM

Componente de YUPP que implementa el ORM.

SIG

Siglas de "Sistema de información geográfica" (GIS, en inglés). Denota una aplicación que permite consultar, manipular y gestionar tipos de datos espaciales.

MVC

Patrón de arquitectura de software que consiste en separar las responsabilidades del sistema en tres componentes bien definidos denominados: Model, View y Controller.

YMVC

Componente de Yupp Framework que facilita la implementación del patrón MVC.

Desarrollo ágil (Agile Development)

Metodología de desarrollo de software que enfoca el proceso de construcción de software hacia una interacción más personal con el cliente, sin tanta documentación y con entregas cada poco tiempo.

Visualizador de Mapas

Un Visualizador de Mapas es un componente que permite realizar consultas gráficas simples sobre información geográfica.

Se encarga de la interacción con el usuario (ofreciendo una interfaz amigable y sencilla de usar).

UI Helper

Clase que contiene métodos que permiten generar el código HTML requerido para la acción a llevar a cabo.

Herramienta

Clase php que contiene métodos que facilitan la construcción y el procesamiento de cierto tipo de datos.

Template de visualización

Clase php que encapsula comportamiento web y sirve para estandarizar cómo se muestra cierta información en la web.

Handler

Un handler o manejador es una operación que se ejecuta cuando sucede un evento asociado. Cada vez que el evento sucede, se notifica a todos los handlers registrados.

Elemento

Un elemento es un objeto del modelo de dominio del usuario. Puede tener o no una representación geográfica.

Observer

Patrón de diseño que permite definir una dependencia de tipo uno a muchos entre objetos de manera que cuando uno de los objetos cambia su estado, el observador se encarga de notificar este cambio a todos los otros dependientes.

WKB

Well Known Binary, formato para almacenar elementos geográficos en una base de datos.

8. Referencias

[1] **YUPP**
http://code.google.com/p/yupp/ [Mayo 2011]

[2] **PHP**
http://www.php.net/ [Mayo 2011]

[3] **MVC**
http://martinfowler.com/eaaDev/uiArchs.html [Mayo 2011]

[4] **Manifiesto por el Desarrollo Ágil de Software**
http://www.agilemanifesto.org/iso/es/manifesto.html [Abril 2011]

[5] **WMS Server**
http://mapserver.org/ogc/wms_server.html [Noviembre 2011]

[6] **WMS Client**
http://mapserver.org/ogc/wms_client.html [Noviembre 2011]

[7] **Geographic Information System Data Models**
http://library.oceanteacher.org/OTMediawiki/index.php/Geographic_Information_System_Data_Models [Noviembre 2011]

[8] **OpenGIS Web Map Service Implementation Specification**
http://www.opengeospatial.org/standards/wms [Noviembre 2011]

[9] **OGC**
http://www.opengeospatial.org/ [Setiembre 2011]

[10] **Geographic Objects**
http://www.opengeospatial.org/standards/go [Noviembre 2011]

[11] **MapServer 6.0.1 Documentation**
http://mapserver.org/ [Noviembre 2011]

[12] **GeoServer**
http://geoserver.org [Noviembre 2011]

[13] **Modelo de Referencia OGC**
http://external.opengis.org/twiki_public/pub/ILAFpublic/QueEsOpenGeospatial/Modelo_de_Referencia_OGC_ORM_Version_2(1)_Espanol.pdf [Setiembre 2011]

[14] **OGC Reference Model (ORM)**
http://www.opengeospatial.org/standards/orm [Setiembre 2011]

[15] **Simple Feature Access - Common Architecture**
http://www.opengeospatial.org/standards/sfa [Setiembre 2011]

[16] **The OpenGIS Geometry Model**
http://dev.mysql.com/doc/refman/5.0/en/opengis-geometry-model.html [Noviembre 2011]

[17] **PostGIS Geography Type**
http://postgis.refractions.net/docs/ch04.html#PostGIS_Geography [Setiembre 2011]

[18] **Simple Feature Access - SQL**
http://www.opengeospatial.org/standards/sfs [Setiembre 2011]

[19] **GIS with MySql**
http://dev.mysql.com/tech-resources/articles/4.1/gis-with-mysql.html [Setiembre 2011]

[20] **GI Markup**
http://www.galdosinc.com/archives/185 [Noviembre 2011]

[21] **GML**
http://www.opengeospatial.org/standards/gml [Noviembre 2011]

[22] **KML**
http://www.opengeospatial.org/standards/kml [Noviembre 2011]

[23] **OpenLayers**
http://openlayers.org/ [Noviembre 2011]

[24] **GeoXACML**
http://www.geoxacml.org/ [Noviembre 2011]

[25] **Observer**
http://martinfowler.com/eaaDev/uiArchs.html [Octubre 2011]

[26] **Observer Pattern**
http://www.clubdesarrolladores.com/articulos/mostrar/115-observer-pattern-patron-observador [Octubre 2011]

[27] **Trabajo FEMI Monitoreo y Control de Problemas de Salud mediante SIG**
http://www.slideshare.net/pablitox/trabajo-femi-monitoreo-y-control-de-problemas-de-salud-mediante-sig [Noviembre 2011]

[28] **Sistema de Informacion Geografica**
http://sig.montevideo.gub.uy/ [Noviembre 2011]

[29] **CLOC**
http://cloc.sourceforge.net/ [Noviembre 2011]

[30] **Jquery**
http://jquery.com/ [Noviembre 2011]