



APLICACIÓN WEB PARA DESPLEGAR INFORMACIÓN DE DISTRIBUCIÓN DE RUTAS DE BUSES
EMPRESARIALES
DRUBU

PAOLA ANDREA OSPINA GONZÁLEZ
Paolira1235@gmail.com



UNIVERSIDAD DEL VALLE
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN
SANTIAGO DE CALI
2011

APLICACIÓN WEB PARA DESPLEGAR INFORMACIÓN DE DISTRIBUCIÓN DE RUTAS DE BUSES
EMPRESARIALES
DRUBU

PAOLA ANDREA OSPINA GONZÁLEZ
Paolira1235@gmail.com

Trabajo de grado para optar el título
De Ingeniera de Sistemas

Director:
Jan Andrés Restrepo Cárdenas, Ing.

Codirector
Carlos Mauricio Gaona Cuevas, Ph. D.



UNIVERSIDAD DEL VALLE
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN
SANTIAGO DE CALI
2011

AGRADECIMIENTOS

A Dios porque gracias a sus promesas puedo cumplir con este ciclo en mi vida.

A mi madre por ser incondicional, fuerte, por sus consejos, por sus enseñanzas de vida y por ser quien me impulsa a hacer todo lo que hago en mi vida

A mis hermanas por su confianza, palabras de apoyo y porque siempre, así sea en la distancia, sacan lo mejor de mí, me ayudan a poner los pies en la tierra.

A Jan, Sr Director! Gracias por ser mi amigo incondicional y creer en mí, gracias por ayudarme en todo y estar siempre allí para escucharme, explicarme, gracias por ser tú!

A Juan Fernando, amor gracias por ser mi maestro, por tu paciencia y por calmarme en mis momentos de desespero, por darme mi espacio y por acompañarme, por darme valor y simplemente por estar ahí, gracias porque en todo este proceso tu apoyo siempre ha sido incondicional.

A mis amigos, por su compañía, consejos y ayudas.

Al profesor Gaona, por su asesoría y colaboración.

NOTA DE ACEPTACIÓN

El presente trabajo de grado fue aprobado en cumplimiento de los requisitos expedidos por la institución para optar el título de:

INGENIERA DE SISTEMAS

Director: Jan Andres Restrepo, Ing.

Codirector: Mauricio Gaona, Ph.D

Evaluador: Oswaldo Solarte

Tabla de contenido

RESUMEN	3
INTRODUCCIÓN	4
1. DEFINICIÓN DEL PROYECTO	5
1.1 PROBLEMA DE INVESTIGACIÓN.....	5
1.1.1 Planteamiento del problema.....	5
1.2. OBJETIVOS.....	5
1.2.1. <i>Objetivo general</i>	5
1.2.2. <i>Objetivos específicos</i>	5
1.3. ALCANCE	6
1.4. JUSTIFICACIÓN	6
2. MARCO DE REFERENCIA	7
2.1 MARCO CONCEPTUAL.....	7
2.1.1 <i>Ingeniería Web</i>	7
2.1.2 <i>RIA</i>	8
2.1.3 <i>AJAX</i>	8
2.1.4 <i>Mashup</i>	10
2.1.5 <i>eXtreme Programming</i>	13
2.1.6 <i>Symfony</i>	14
2.1.7 <i>Doctrine</i>	15
2.1.8 <i>SIG (Sistemas de Información Geográfica)</i>	15
2.1.9 <i>Cartografía Digital</i>	17
2.1.10 <i>API YOURS</i>	20
2.1.11 <i>Gosmore</i>	21
2.1.12 <i>Nominatim</i>	21
2.1.13 <i>OpenLayers</i>	21
2.1.14 <i>OpenStreetMap</i>	23
2.1.15 <i>Panoramio</i>	24
2.1.16 <i>Web 2.0</i>	24
2.1.17 <i>Git</i>	25
2.2 ESTADO DEL ARTE	25
2.2.1 <i>Routing maps</i>	26
2.2.2 <i>Mapaspublicar.com</i>	26
2.2.3 <i>Palma Routing</i>	27
3. DESARROLLO DEL SISTEMA	28
3.1. RECOLECCIÓN DE LA INFORMACIÓN	29
3.2. DESARROLLO DEL SOFTWARE	29
3.2.1 <i>Definición de las historias de usuario</i>	29
3.2.2 <i>Diseño de la arquitectura del sistema</i>	30
3.2.3 <i>Diseño de la base de datos del sistema</i>	32
3.2.4 <i>Detalles de la implementación</i>	33
4. RESULTADOS	40
5. CONCLUSIONES Y TRABAJOS FUTUROS	42

5.1. CONCLUSIONES.....	42
5.2. TRABAJOS FUTUROS.....	43
BIBLIOGRAFÍA	44

Índice de figuras

Figura 1: Conceptos, metodologías y tecnologías usadas en DRUBU	7
Figura 2: Árbol de tecnologías Ajax.....	9
Figura 3: Diferencia entre el modelo clásico y Ajax	10
Figura 4: Arquitectura de la tecnología Mashup	12
Figura 5: funcionamiento de la metodología XP en práctica	13
Figura 6: Comparación entre tipo de datos raster y vectorial	16
Figura 7: Modelo Holístico de un SIG funcional	17
Figura 8: Coordenadas geográficas	18
Figura 9: Sistema de coordenadas UTM	19
Figura 10: proyección geográfica	19
Figura 11: Aplicaciones que usa el Api YOURS	20
Figura 12: Ejemplo de funcionamiento de OpenLayers	23
Figura 13: Funcionamiento de Routing Maps	26
Figura 14: Captura del funcionamiento de mapaspublicar.com	27
Figura 15: Funcionamiento de palma Routing	28
Figura 16: Vista lógica de la aplicación.....	32
Figura 17: Modelo de datos gestión de rutas y buses	32
Figura 18: Modelo de datos gestión de usuarios generado por sfDoctrineGuardplugin.....	33
Figura 19: Distribución de módulos del proyecto	33
Figura 20: lista de datos	34
Figura 21: Creación de bus	35
Figura 22: Creación de ruta	36
Figura 23: Indicaciones y popup.....	39
Figura 24: Datos de la ruta	39
Figura 25: Capas de datos de la aplicación	41

Índice de tablas

Tabla 1: Diferencias entre un portal web y Mashup	15
Tabla 2: Actividades del proyecto	33
Tabla 3: Formato usado para las Historias de usuario	34
Tabla 4: Hardware usado durante el desarrollo del proyecto	35

RESUMEN

El sistema “DRUBU”, Distribución de rutas de buses empresariales es una aplicación Web que permite crear, consultar y ubicar rutas de sistemas de transporte empresariales con información contextual sobre un mapa digital de la ciudad de Santiago de Cali, ayudando a plasmar información plana de forma gráfica, rápida y precisa con fácil acceso al ser un aplicativo web, usando como base de interacción el concepto de *Mashup*, aplicaciones web híbridadas.

Para su desarrollo fue necesario abordar diversos conceptos sobre cartografía, geodesia, SIG (Sistemas de Información Geográfica), servicios online de mapas viales y otras tecnologías actuales de tipo web.

Después de solidificar la base teórica y tener una visión amplia sobre las diversas tecnologías actuales, tanto de servicios como herramientas, se decidió realizar la implementación usando el lenguaje de programación PHP, usar el *framework Symfony* para el manejo del patrón de desarrollo Modelo/Vista/Controlador, usar los servicios online *OpenStreetMap* para la visualización de mapas y *Gosmore* para el enrutamiento, y consolidar la interacción del usuario y servicios a través de los APIs implementadas en JavaScript de *OpenLayers* y *Panoramio*, proveedor de fotografías geoposicionadas. Todo esto permite entregar al usuario una aplicación tipo RIA mejorando su experiencia de uso.

INTRODUCCIÓN

En la actualidad se ha hecho común la necesidad de obtener información de forma más amigable, práctica y ajustada al entorno que nos rodea, requiriéndose interconectar datos para obtener más información que permita cumplir con las expectativas ó alcanzar nuevos objetivos impuestos.

Los nuevos sistemas cartográficos y de despliegue de información geográfica urbana a través de mapas disponibles en Internet, permiten obtener una gran fuente de información, la cual puede “mezclarse” con otros sistemas y así ofrecer una nueva perspectiva de lo que ésta puede proporcionar, dando a los usuarios la posibilidad de tener una experiencia más palpable de su entorno.

Este tipo de concepto se conoce como *Mashup* cuyo fundamento se basa en la mejor manera de mostrar información correlacionando los datos que se encuentran a su disposición.

Este documento estructurado en cinco capítulos describe todos los procesos teóricos y prácticos que se realizaron para el desarrollo del trabajo de grado: El primer capítulo “Definición del proyecto” define el problema y su posible solución, los objetivos, la justificación, el alcance del proyecto. El segundo capítulo “Marco de referencia”, presenta el marco teórico y el marco conceptual. El tercer capítulo “Diseño del sistema” explica la metodología de desarrollo usada y muestra de una manera detallada los resultados obtenidos en cada una de sus etapas. En el cuarto capítulo “Resultados” se detallan los resultados obtenidos de la aplicación. El quinto capítulo “Conclusiones y trabajos futuros” presenta las conclusiones y trabajos futuros del proyecto.

1. DEFINICIÓN DEL PROYECTO

1.1 Problema de Investigación

1.1.1 Planteamiento del problema

La obtención de información siempre ha sido una necesidad en la vida cotidiana, cada vez se exige que ésta sea de forma veraz, asequible y oportuna. Esto no es una excepción para el manejo de rutas de buses, las cuales se llevan de forma manual, no automatizada, de cambio constante y no enfocado directamente al cliente de las rutas, ocasionando que estos obtengan información adicional a través del boca a boca, generando algunas veces confusión y redundancia de esfuerzos.

Uno de los casos más particulares surge cuando los usuarios de las rutas poseen la intención de abordar una ruta diferente a la habitual, con el objetivo de dirigirse a un lugar de la ciudad diferente al de su destino cotidiano. Bajo esta circunstancia surge la necesidad de obtener información extra para transportarse, generalmente se busca personas cercanas con experiencia y/o conocimiento de las rutas que en la mayoría de los casos son otros usuarios dejando como última opción dirigirse a la oficina encargada de las rutas y consultar la información que poseen ellos incluyendo el mapa físico de distribución de rutas.

También surgen otros inconvenientes arraigados en la misma gestión de las rutas cuando estas cambian constantemente y se vuelve tedioso realizar los cambios e igualmente presentar dicha información la cual no carece de efectividad real.

1.2. Objetivos

1.2.1. Objetivo general

Desarrollar una aplicación Web, la cual permita desplegar información sobre distribución de rutas buses empresariales en la ciudad de Cali.

1.2.2. Objetivos específicos

- Determinar que conceptos de la Web 2.0 son los más adecuados para el diseño e implementación de la aplicación.

- Identificar las APIs de *OpenStreetMap* para ser usadas en la presentación de información geográfica.
- Recolectar y analizar los datos que contienen los sistemas de información sobre usuarios y el sistema de buses para estandarizarla y usarla en la aplicación de buses.
- Generar un módulo para el ingreso y manejo de datos por parte del usuario de forma gráfica.
- Generar un módulo para el despliegue de información usando conceptos de *Mashup*.
- Validar la aplicación para un caso práctico.

1.3. Alcance

Diseño, desarrollo e implementación de la aplicación Web para desplegar información sobre la distribución de rutas de buses.

La aplicación funcionará en un ambiente web aplicando conceptos de Web 2.0 permitiendo la actualización dinámica de la información.

Se abordaron los siguientes aspectos: Gestión de usuarios, gestión de buses, gestión y consulta de rutas

1.4. Justificación

La Aplicación Web para desplegar información sobre la distribución de rutas de buses empresariales, DRUBU, fue concebida debido a la creciente ola de uso de medios digitales en actividades diarias, entre estas la generación y consulta de rutas de transporte para la cual se encuentran muy pocas opciones: las que se encuentran en la web no son específicamente para el lugar del caso de estudio o las que si son para Colombia no poseen información enriquecida o son software privativo por el que hay que pagar un licenciamiento muy alto o no son web sino que funcionan para Hardware de Geoposicionamiento global haciendo necesario que el usuario adquiriera ese tipo de equipos (Garmin, Siemens, Nokia, NavMan, Tomtom, etc.). Teniendo en cuenta el mercado y las necesidades del usuario este proyecto propone mejorar el estado actual de presentar, almacenar y asignar las rutas de transporte que poseen las empresas, usando mapas geográficos de la ciudad de Cali enfocándose en la presentación de información para que sea más amigable al usuario final.

2. MARCO DE REFERENCIA

2.1 Marco conceptual

Para el desarrollo de este proyecto, se manejaron conceptos teóricos que se describen en la Figura 1, en ésta se plasma un esquema general de los conceptos, metodologías y tecnologías usados para la ejecución y finalización de la aplicación:

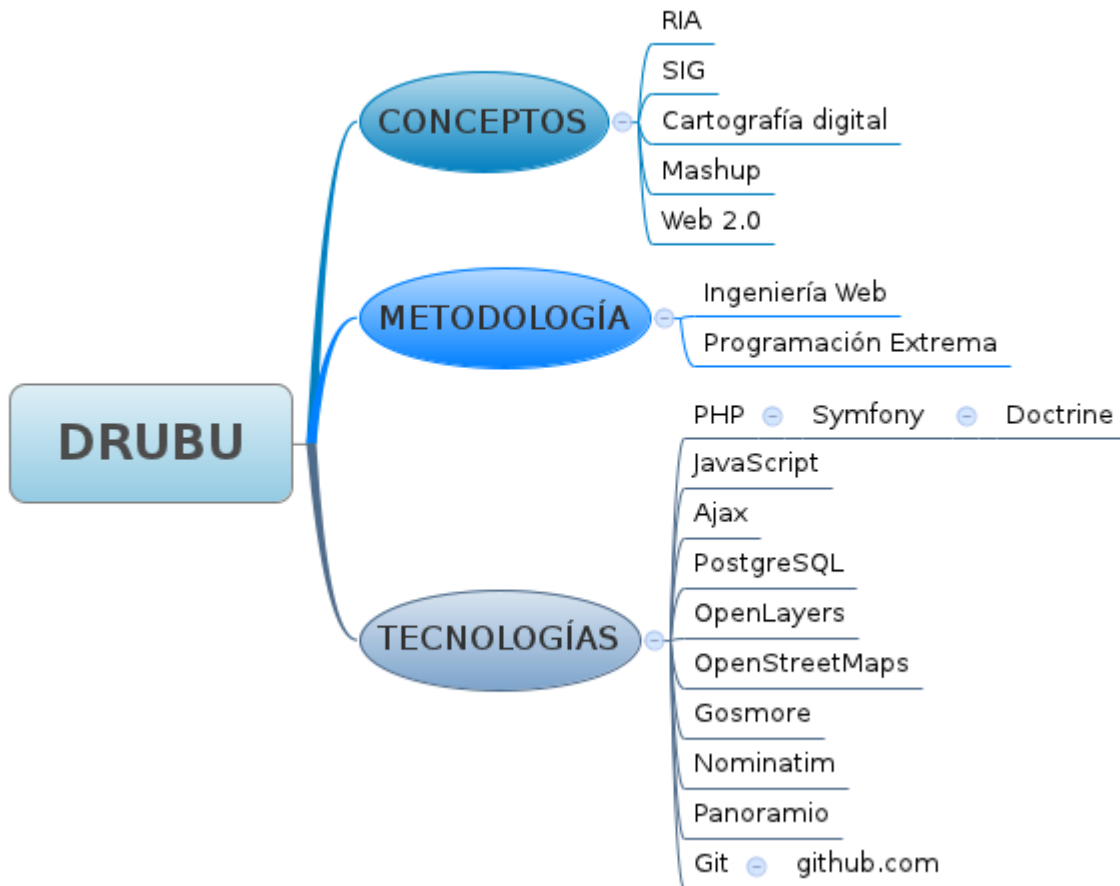


Figura 1: Conceptos, metodologías y tecnologías usadas en DRUBU

2.1.1 Ingeniería Web

La Ingeniería de la Web es la aplicación de metodologías sistemáticas, disciplinadas y cuantificables al desarrollo eficiente, operación y evolución de aplicaciones de alta calidad en la *World Wide Web*.

En este sentido, la Ingeniería de la Web hace referencia a las metodologías, técnicas y herramientas que se utilizan en el desarrollo de Aplicaciones Web complejas y de gran

escala, en las que se apoya la evaluación, diseño, desarrollo, implementación y evolución de dichas aplicaciones [1].

La Ingeniería de la Web es multidisciplinar y aglomera contribuciones de diferentes áreas como: arquitectura de la información, ingeniería de hipermedia/hipertexto, ingeniería de requisitos, diseño de interfaz de usuario, usabilidad, diseño gráfico y de presentación, diseño y análisis de sistemas, ingeniería de software, ingeniería de datos, indexado y recuperación de información, testeo, modelado y simulación, despliegue de aplicaciones, operación de sistemas y gestión de proyectos.

2.1.2 RIA

Rich Internet Applications (Aplicaciones Ricas de Internet) es un nuevo tipo de aplicación con más ventajas sobre las tradicionales aplicaciones Web. Esta surge como una combinación de las ventajas que ofrecen las aplicaciones Web y las aplicaciones de escritorio tradicionales [2].

Normalmente en las aplicaciones Web, hay una recarga continúa de páginas cada vez que el usuario pulsa sobre un enlace produciendo un tráfico muy alto entre el cliente y el servidor. Otra de las desventajas de las aplicaciones Web tradicionales es la poca capacidad multimedial que poseen de forma innata, para ver un vídeo se debe usar un programa externo o ajeno para su reproducción (plugins).

En cambio, en los entornos RIA, no se producen recargas de página, ya que desde el principio se carga lo necesario y sólo se produce comunicación con el servidor cuando se necesitan datos externos como los de una Base de Datos o archivos externos.

2.1.3 AJAX

AJAX, acrónimo de *Asynchronous JavaScript And XML (JavaScript* asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (*Rich Internet Applications*). Éstas se ejecutan en el cliente, es decir, en el navegador de los usuarios y mantiene comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre la misma página sin necesidad de recargarla [3]. Esto significa aumentar la interactividad, velocidad y usabilidad en la misma.

Ajax se utilizó como término por primera vez en el artículo “*Ajax: A New Approach to Web Applications*”¹, hasta ese momento la nueva vertiente de aplicaciones Web que estaba surgiendo carecía de término normalizado. Según lo expuesto “AJAX no es una tecnología en sí misma, en realidad, se trata de varias tecnologías independientes que se unen en nuevas y sorprendentes formas”.

Las distintas tecnologías que conforman Ajax son:

- XHTML y hojas de estilos en cascada (CSS) para el diseño que acompaña a la información.
- *Document Object Model* (DOM) accedido con un lenguaje de scripting por parte del usuario, especialmente implementaciones *ECMAScript* como *JavaScript* y *Jscript*, para mostrar e interactuar dinámicamente con la información presentada.
- El objeto *XMLHttpRequest* para intercambiar datos asincrónicamente con el servidor web. En algunos *frameworks* y en algunas situaciones concretas, se usa un objeto *iframe* en lugar del *XMLHttpRequest* para realizar dichos intercambios.
- XML es el formato usado comúnmente para la transferencia de vuelta al servidor, aunque cualquier formato puede funcionar, incluyendo HTML pre-formateado, texto plano, JSON y hasta EBML.

En el siguiente esquema se muestra como se agrupan las tecnologías anteriormente mencionadas aplicadas en *Ajax*:

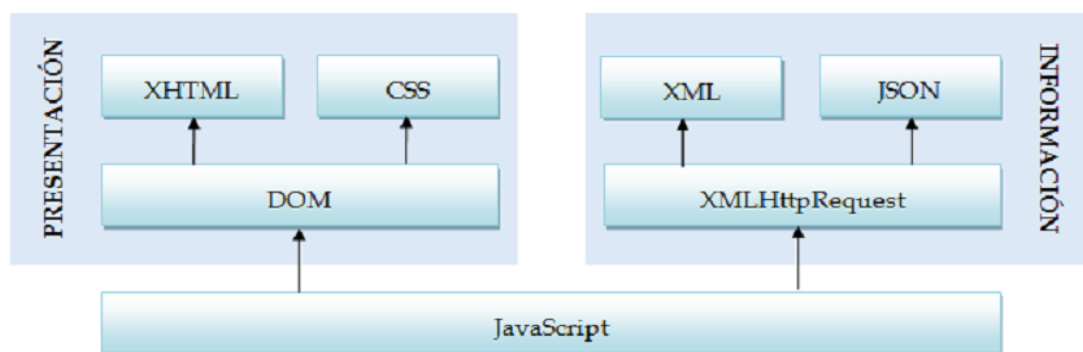


Figura 2: Árbol de tecnologías Ajax

¹James, J. (febrero de 2005). *Ajax: A New Approach to Web Applications*. Recuperado el octubre de 2009, de <http://adaptivepath.com/ideas/essays/archives/000385.php>

La aparición de *Ajax* ha sido una gran novedad a la hora de realizar páginas Web, puesto que es una visión totalmente diferente al uso que existía hasta entonces. En el siguiente gráfico se presentan las diferencias en el funcionamiento entre las aplicaciones Web tradicionales y aquellas en las que se ha introducido *Ajax*:



Figura 3: Diferencia entre el modelo clásico y Ajax

Lo que *Ajax* pretende es solucionar la sobrecarga que se produce si se realizan muchas peticiones al servidor, provocando que el cliente soporte continuas y aparentes largas esperas para obtener lo que espera.

2.1.4 Mashup

Se puede definir el término *Mashup* como una aplicación Web, o sitio Web, que combina a la perfección el contenido de más de una fuente en una experiencia integrada. Es decir, un sitio o aplicación Web que utiliza los servicios que ofrecen otras páginas y los integra, formando un único sitio. Es lo que podría llamarse una aplicación o sitio Web híbrido [4]. Como se puede comprobar, la característica fundamental es que no es una página estática en la que el contenido lo actualiza su propio dueño, sino que se renueva constantemente al ritmo que lo hacen aquellos sitios de los que se extrae la información.

Las principales ventajas y desventajas que se presentan al usar *Mashup* en una aplicación o sitio web, son:

Ventajas

Para los proveedores:

- I+D y marketing.
- Publicidad para la marca.
- Incremento de ventas.
- Ingresos por publicidad directa e indirecta.

Para los desarrolladores:

- Aplicaciones con mucho nivel.
- utilización de pocos recursos.
- Interés del público.
- tecnología accesible.
- No es sólo para programadores.

Para los usuarios

- Ocio y diversión.
- Ahorro de tiempo y dinero.
- Acceso a nuevos y mejores servicios.
- “Los usuarios son lo más importante en la Web 2.0”.

Desventajas

- Anuncios no deseados.
- Problemas con la propiedad intelectual.
- Limitadores tecnológicos.
- Dependencia de los proveedores.

Algunas diferencias entre los Portales y la de *Mashups* son las siguientes:

	Portal	Mashup
Clasificación	Madura (antigua), extensión del modelo tradicional de aplicaciones web.	Nueva, Web 2.0
Filosofía	El enfoque se resume en generación de código y agregación de fragmentos.	Adopta un enfoque más básico, agregando el contenido sin preocuparse del enmarcado.
Dependencias de Contenido	Agrega contenido orientado a la presentación (HTML, WML, etc.)	Puede operar con contenido en XML y también contenido orientado a la presentación.
Dependencias de Ubicación	La agregación del contenido toma lugar en el servidor.	La agregación del contenido puede tomar lugar tanto en el servidor como en el cliente.
Modelo de eventos	Modelo de Lectura y Actualización definido por un API de <i>portlet</i>	Las operaciones de CRUD están basadas en un arquitectura tipo REST (<i>representational state</i>

	específico (dependiente del <i>vendedor</i>).	<i>transfer</i>), pero no hay un API formal.
Estándares	Se observan estándares como <i>JSR 168</i> , <i>JSR286</i> y <i>WSRP</i> , los cuales gobiernan su comportamiento; sin embargo, la implementación de la tecnología depende del vendedor.	Estos pueden utilizar XML-RPC, SOAP (que requiere de programación) o intercambio XML basado en REST (mediante HTTP y no SOAP). Generalmente se utiliza RSS y Atom para el intercambio de información o fuentes de contenido.

Tabla 1: Diferencias entre un portal web y un Mashup [5]

Para utilizar *Mashup* en un sitio Web, se siguen normalmente tres acciones básicas [6]:

Fuente de Datos: Esta técnicamente se refiere a un “*Web Service*”, una API, un RSS o una página web bien estructurada, los cuales se comportan como los proveedores de información, al ser relacionada a la Internet puede estar ubicada en cualquier servidor del mundo.

Sitio Mashup: Es la aplicación Web integradora, encargada de realizar la mezcla y correlación de información y a través de una interface más atractiva presenta la nueva información.

- **Browser o navegador:** Es el componente usado por el usuario para acceder al sitio *Mashup*.

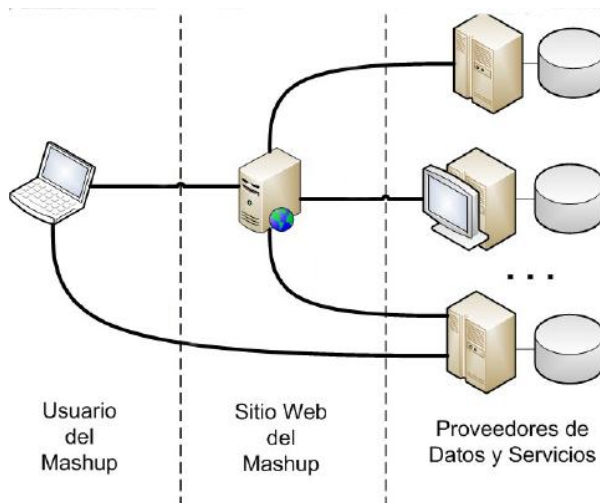


Figura 4: Arquitectura de la tecnología *Mashup* [6]

Actualmente se pueden identificar tres tipos:

- **Mashups de consumidor:** Son los más conocidos, combinan muchas fuentes externas en una sola interfaz: mapas + fotos + videos.
- **Mashups de datos:** Mezclan información similar de diferentes fuentes de datos (búsquedas, compras, noticias).
- **Mashups de negocio:** mezclan fuentes de datos internas y externas que integran en una sola interfaz. Son colaborativos.

2.1.5 eXtreme Programming

La programación extrema o *eXtreme Programming* (XP) es un enfoque de la ingeniería de software formulado por Kent Beck, autor del primer libro sobre la materia, *Extreme Programming Explained: Embrace Change* (1999) [7]. Es la más destacada de los procesos ágiles de desarrollo de software. Al igual que éstos, la programación extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad. Los defensores de XP consideran que los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable del desarrollo de proyectos. Creen que ser capaz de adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto es una aproximación mejor y más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios en los requisitos. Se puede considerar la programación extrema como la adopción de las mejores metodologías de desarrollo de acuerdo a lo que se pretende llevar a cabo con el proyecto, y aplicarlo de manera dinámica durante el ciclo de vida del software.



Figura 5: funcionamiento de la metodología XP en práctica [8]

Las características fundamentales del método son:

- Desarrollo iterativo e incremental.
- Pruebas unitarias continuas
- Programación en parejas
- Frecuente interacción del equipo de programación con el cliente o usuario
- Refactorización del código
- Propiedad del código compartida
- Simplicidad en el código

2.1.6 Symfony

Symfony es un *framework* diseñado para facilitar y optimizar la construcción de aplicaciones Web con PHP, reuniendo un conjunto de herramientas que facilitan esta tarea. Fundamentalmente *Symfony* toma las mejores características de otros *frameworks* como *Rails* e incorpora ideas propias, haciéndolo estable y muy bien documentado [9].

Este *framework* hace uso del patrón tradicional MVC o Modelo Vista Controlador, separando la lógica de negocio que representa la información con la que trabaja la aplicación. La lógica del servidor que se encarga de controlar y coordinar todos los elementos de la aplicación y transformar las peticiones del cliente en operaciones sobre el modelo y la presentación o vista de la aplicación que se encarga de transformar la información obtenida por el modelo, en resultados visibles para el cliente en las páginas Web a las que acceden los usuarios.

Algunas características de *Symfony* son:

- El código es más legible e incluye comentarios de *phpDocumentor*.
- Facilita la creación de sesiones restringidas mediante la gestión y autenticación de credenciales.
- La gestión de la caché reduce el ancho de banda utilizado y la carga del servidor.
- Hace uso de *helpers* que simplifican el tamaño del código necesario en la presentación y facilitan la interacción con *Ajax*.

2.1.7 Doctrine

Doctrine es un ORM para PHP 2.2.3 y posterior. Además de todas las ventajas que conlleva un ORM, uno de sus puntos fuertes es su lenguaje DQL (*Doctrine Query Language*) inspirado en el *HQL de Hibernate* [10].

Cuando se trabaja con Doctrine, se necesita informar a su motor interno cual es el modelo de la aplicación, para ello se puede hacer ingeniería inversa de la base de datos existente, o si se empieza la aplicación desde cero, crear el modelo en la sintaxis específica que propone Doctrine y luego generar toda la base de datos.

2.1.8 SIG (Sistemas de Información Geográfica)

“Un SIG es un sistema de hardware, software y procedimientos diseñados para auxiliar en la captura, administración, manipulación, análisis, y presentación de datos u objetos referenciados espacialmente llamados comúnmente datos espaciales u objetos espaciales” (Harmon & Anderson, 2003)

Los sistemas de información geográfica utilizan la información geográfica (base de datos georeferenciada) almacenada en un sistema de información para demostrar su efectividad, especialmente en la resolución de problemas espaciales, servir de soporte para la toma de decisiones y ayudar en la planificación [11].

La razón fundamental para utilizar un SIG es la gestión de información espacial. El sistema permite separar la información en diferentes capas temáticas y las almacena independientemente, permitiendo trabajar con ellas de manera rápida y sencilla, facilitando la posibilidad de relacionar la información existente a través de la topología de los objetos, con el fin de generar otra nueva que no se podría obtener de otra forma.

Las principales cuestiones que puede resolver un SIG son las siguientes:

- Localización, preguntar por las características de un lugar concreto.
- Condición, el cumplimiento o no de unas condiciones impuestas al sistema.
- Tendencia, comparación entre situaciones temporales o espaciales distintas de alguna característica.
- Rutas, cálculo de rutas óptimas entre dos o más puntos.
- Pautas, detección de pautas espaciales.
- Modelos, generación de modelos.

Existen dos formas de almacenar los datos SIG:

- **Tipo de dato Raster:** Es cualquier tipo de imagen digital representada en mallas, este formato divide el espacio en celdas regulares donde cada una de ellas representa un único valor.
- **Tipo de dato Vectorial:** Este tipo de datos guarda las características geográficas como vectores, tal y como indica su nombre.

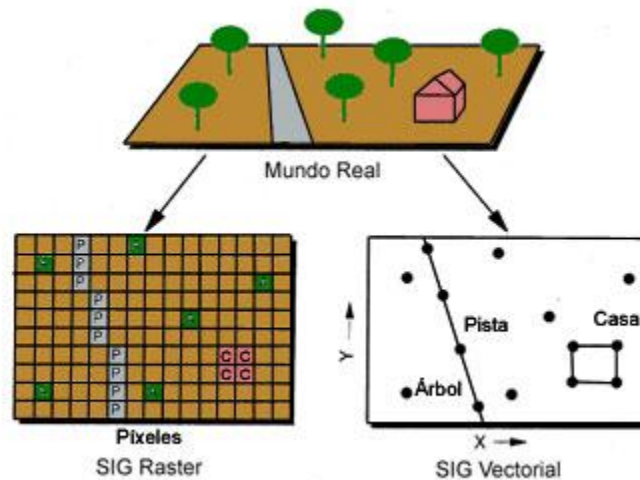


Figura 6: Comparación entre tipo de datos *raster* y vectorial [12]

Una base de datos SIG puede contener la información en cualquiera de estos formatos o en una combinación de los dos. A partir de esa base de datos, se crean mapas según las necesidades del usuario. Además de producir y almacenar mapas, un SIG puede procesarlos, ya que los datos del mundo real están almacenados en formato digital, a diferencia de los mapas de papel convencional

Un SIG no solamente es un software instalado en un equipo de cómputo, sino que es un conjunto de componentes resultado del proceso de diseño, creación y manejo de éstos:

- Tecnología (programas y equipos, software y hardware)
- Ideas
- Personal
- Red
- Datos
- Métodos

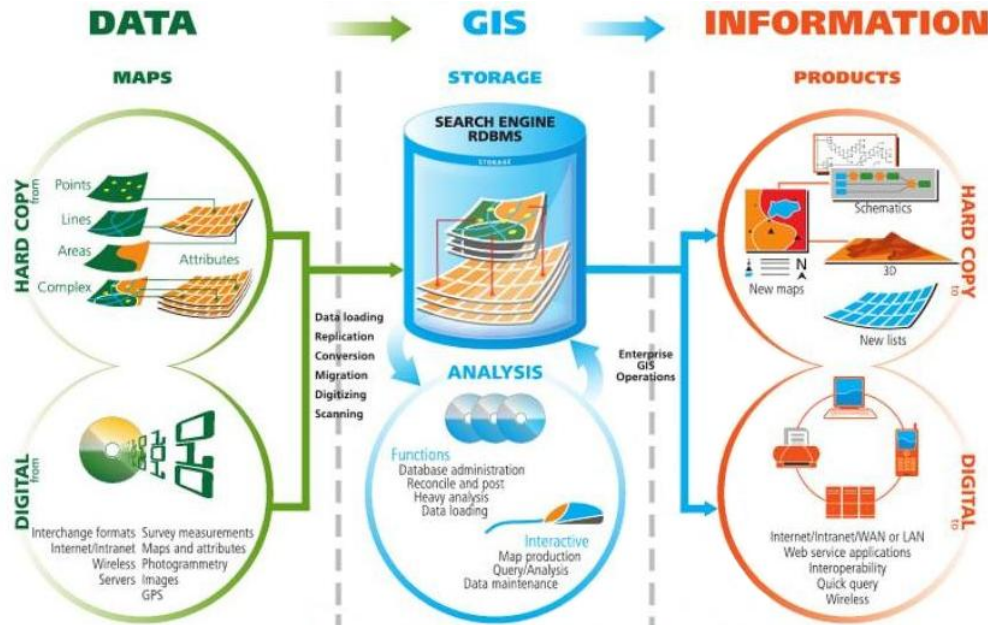


Figura 7: Modelo Holístico de un SIG funcional [13]

En la figura anterior se observa un modelo holístico de un SIG funcional, el cual transforma los datos a través del análisis en información útil. En el centro se observa que el SIG almacena datos espaciales, lleno de información de atributos vinculados lógicamente (desde la izquierda), en una base de datos de almacenamiento SIG, donde las funciones de análisis se controlan de forma interactiva por un operador humano para generar los productos de información necesaria (que se muestra a la derecha).

2.1.9 Cartografía Digital

La cartografía es una disciplina que integra ciencia, técnica y arte, que trata de la representación de la Tierra sobre un mapa o representación cartográfica, sin embargo, la Cartografía digital es un sistema georeferenciado que contiene información detallada sobre las condiciones físicas y la infraestructura de un determinado territorio, también se puede decir que es la parte de la cartografía que se ocupa del tratamiento y la representación de datos mediante números o caracteres de un repertorio finito, además de ser la base para la implementación de los SIG [11].

2.1.9.1 Sistema de coordenadas²

Un sistema de coordenadas es una creación artificial para permitir la definición analítica de la posición de un objeto o fenómeno. Se trata de un conjunto de valores y puntos que

²Olaya Victor, versión 1,0, sistemas de información geográfica, 12 enero 2011, capítulo 3, pág 41-49

permiten definir unívocamente la posición de cualquier punto de un espacio n -dimensional. Desde un punto de vista matemático, todos los sistemas de coordenadas son admisibles y el único motivo para seleccionar uno u otro es la conveniencia que permite cada uno de ellos.

Los sistemas de coordenadas más utilizados para representar la superficie de la Tierra Son:

- El **Sistema de coordenadas geográficas** [SCG] utiliza una superficie esférica tridimensional para definir las localizaciones sobre la superficie terrestre. En este sistema, cualquier punto sobre la superficie terrestre se determina con dos ángulos medidos desde los centros de la Tierra, conocidos como latitud y longitud. La latitud mide el ángulo entre cualquier punto y el Ecuador. La longitud mide el ángulo a lo largo del Ecuador desde cualquier punto de la Tierra.

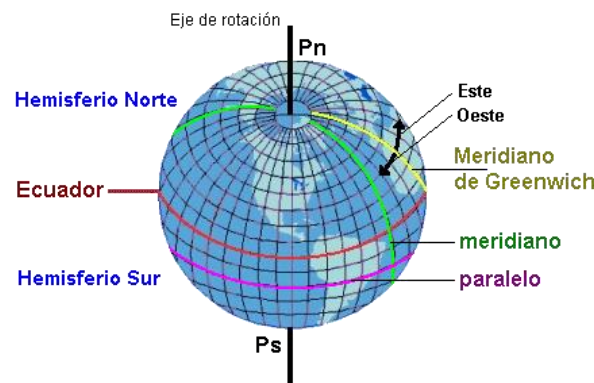


Figura 8: Coordenadas geográficas [14]

- **Sistema de Coordenadas Universal Transversal de Mercator** (UTM) es un sistema de coordenadas basado en la proyección cartográfica transversa de *Mercator*³, que se construye como la proyección de *Mercator* normal, pero en vez de hacerlo por la tangente al Ecuador, se la hace tangente a un meridiano. A diferencia del sistema de coordenadas geográficas, las magnitudes en el sistema UTM se expresan en metros únicamente al nivel del mar que es la base de la proyección del elipsoide de referencia. La proyección de *Mercator* es una proyección cilíndrica y por tanto, resulta de la proyección de una esfera (la Tierra) en un cilindro tangente a un meridiano central, en particular al Ecuador de la Tierra.

En coordenadas UTM una posición se define por tres elementos: el huso en el que se encuentra, las coordenadas E (eje horizontal) y N (eje vertical), y el hemisferio en el que se encuentra.

³La **proyección de Mercator** es un tipo de proyección cartográfica cilíndrica, ideada por **Gerardus Mercator** en 1569, para elaborar planos terrestres [15].

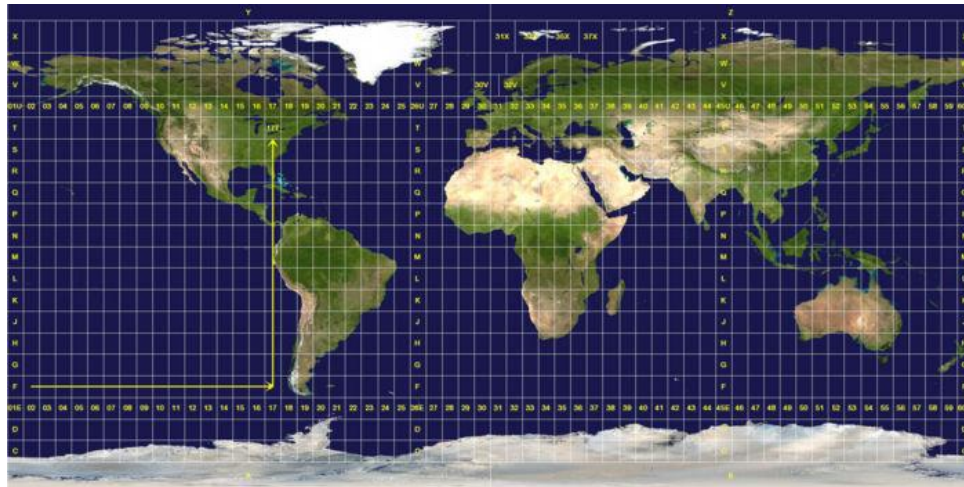


Figura 9: Sistema de coordenadas UTM [16]

- Sistema de coordenadas proyectadas.** Para representar la totalidad de la superficie terrestre sin ningún tipo de distorsión⁴, un mapa debe tener una superficie esférica porque un mapa plano no puede representar con exactitud la superficie redondeada de la Tierra, excepto en áreas muy pequeñas en las que la curvatura es despreciable. El sistema de transformación es lo que se conoce como proyección cartográfica. Cuando una superficie esférica se transfiere a un plano modifica su geografía y la distorsiona, pero existen muchas transformaciones que mantienen una o varias de las propiedades geométricas del globo. La clasificación de las proyecciones es compleja, pues hay diferentes criterios teniendo en cuenta las propiedades geométricas o la superficie de la que derivan. La gran diversificación de proyecciones se debe al hecho que se debe minimizar la distorsión dentro de la zona que se pretende cartografiar y de esa manera mantener la exactitud del posicionamiento.



Figura 10: proyección geográfica [17]

⁴Conocido también como error de deformación.

2.1.10 API YOURS

El objetivo de YOURS (**Y**et **a**nother **O**pen**S**treet**M**ap **R**oute **S**ervice) es utilizar los datos de *OpenStreetMap* para hacer un sitio web de enrutamiento utilizando tantas aplicaciones *Open Source* como sea posible [18].

Algunas de sus características son:

- Generar rutas más rápidas o más cortas en modos diferentes:
 - Utilizando todos los caminos disponibles para autos, bicicletas y peatones.
 - Utilizando sólo rutas/redes ciclistas nacionales, regionales o locales para bicicletas.
- Permite arrastrar, colocar y mover *waypoints*.
- Geolocalización: Búsqueda de calles y nombres de lugares para determinar sus coordenadas.
- Descargar rutas en archivos como GPX, por lo que se puede usar en dispositivos GPS para hacer *trackback*.
- Zoom automático a la ubicación del usuario basado en geolP⁵.

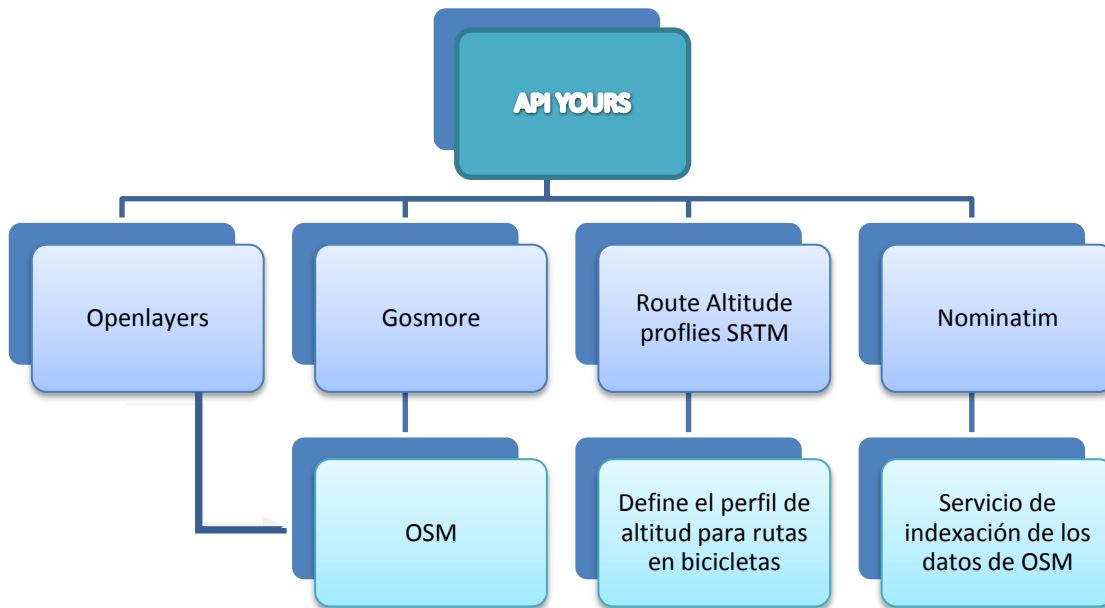


Figura 11: Aplicaciones que usa el Api YOURS

⁵Georeferenciación por la dirección IP.

2.1.11 Gosmore

Es un motor de enrutamiento y visualización de datos XML para archivos *.osm* de *OpenStreetMap* [20] como el caso de “planet.osm” u otros de *Geofabrik* o *Cloudmade*.

Las Características principales que provee *Gosmore* [19]:

- Ventana de visualización de mapas en 2D (proyección *Mercator*), adicionalmente vista en 3D (como ave en vuelo).
- Búsqueda incremental de todas las etiquetas. Los resultados se ordenan desde la más cercana a la más alejada.
- Enrutamiento con soporte de restricciones.
- Obtiene ubicación actual de un receptor GPS (a través de *gpsd* usando Linux).
- Posibilidad de agregar puntos de interés y trazar caminos que se almacenen en archivos tipo *osm*.
- Capacidad para cambiar automáticamente mapas sobre la marcha, para poder acceder a todo el planeta.

2.1.12 Nominatim

Es una herramienta creada para buscar datos de *OpenStreetMaps* por su nombre, dirección y generar direcciones de síntesis de los puntos de OSM (geocodificación inversa) [20]. Se puede encontrar en <http://nominatim.openstreetmap.org>. *Nominatim* se utiliza comúnmente como una de las fuentes en el sistema de búsqueda de la página principal *OpenStreetMap* y en los sitios web de *MapQuest Open Initiative*⁶.

2.1.13 OpenLayers

Es una librería *Open Source* creada por la compañía estadounidense **MetaCarta**.

Provee un API de *JavaScript* que permite la incorporación fácil de mapas en una página web o aplicación sin importar la fuente, ya que separa las herramientas de mapas de las herramientas de datos, de manera que cualquiera de ellas pueda trabajar sobre diferente fuente de datos [21].

Openlayers actualmente tiene soporte para las capas de OGC (*Open Geospatial Consortium*) WMS (*Web Map Service Interface Standard*), WFS (*Web Feature Service*), provee controles de: Zoom panorámico, barra zoom, controles del ratón, capa de

⁶ <http://open.mapquest.co.uk/>

comunicación o “leyenda,” *permalink*, marcadores, ventanas emergentes, características de objetos, barra de escala, iconos, y selección de capas entre otros eventos.

En el siguiente aparte de código se muestra cómo funciona OpenLayers

```
//Llamar a OpenLayers API
<script src="http://openlayers.org/api/OpenLayers.js"></script>
<script type="text/javascript">
function init() {
    map = new OpenLayers.Map( 'map' );
    // se crean los servidores (Metacarta,Google,Bing, Yahoo, OpenStreetMaps)
    server = new OpenLayers.Layer.WMS( "OpenLayers WMS",
        "http://labs.metacarta.com/wms/vmap0",
        {layers: 'basic'} );
    server_google = new OpenLayers.Layer.Google( "Google",
        {type: G_HYBRID_MAP } );
    server_bing = new OpenLayers.Layer.VirtualEarth("Bing");
    server_yahoo = new OpenLayers.Layer.Yahoo("Yahoo");
    server_osm = new OpenLayers.Layer.OSM ("OSM");
    //se crean capas por cada servidor
    map.addLayers([server, server_google, server_bing, server_yahoo]);
    //Se pone la capa del servidor de osm por defecto
    map.setBaseLayer(server_osm);
    //Se crea el control del mouse para ubicar la posición de éste en el mapa
    map.addControl(new OpenLayers.Control.MousePosition());
    //Se ubica un centro del mapa
    map.setCenter(new OpenLayers.LonLat(0, 0), 2);
    //Habilita el cambio de capas
    map.addControl( new OpenLayers.Control.LayerSwitcher() );
}
</script>
```

Este es el resultado obtenido del anterior código:

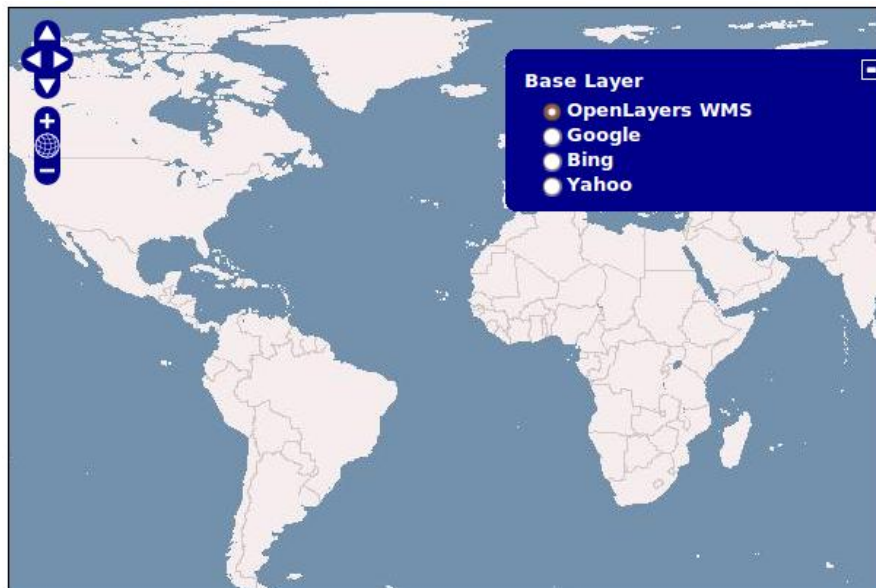


Figura 12: Ejemplo de funcionamiento de OpenLayers

2.1.14 OpenStreetMap

Es un proyecto creado por el inglés *Steve Coast* en el año 2006. Está enfocado en crear y ofrecer datos geográficos libres, básicamente son mapas libres y editables [22]. Su origen es debido a que la mayoría de mapas existentes y asequibles en internet tienen restricciones legales ó técnicas para su uso, evitando que cualquier persona los pueda utilizar a su libre elección de forma creativa o productiva. Además poseen licencias de uso que no permiten realizar corrección de errores, añadir nuevos datos o emplear esos mapas en determinados modos sin pagar por ellos. *OpenStreetMap* por su carácter y filosofía cuenta con el apoyo de una gran comunidad de usuarios que pueden subir sus trazas o coordenadas desde un GPS o cualquier aparato de georeferenciamiento para crear locaciones nuevas o corregir datos existentes.

Se trata de un proyecto con bastante impacto a nivel mundial, cada día se añaden aproximadamente 25.000 Km de carreteras y caminos nuevos, con casi un total de 60'000.000 Km viales y cuenta con más de 300.000 usuarios registrados. El tamaño de la base de datos del mundo "*planet.osm*" es de más de 170 GB con un incremento diario de aproximadamente 10 MB, esto último gracias a permisos concedidos de poder "calcar" mapas de *Bing maps* y *Yahoo! maps* y el apoyo de entidades del estado de diferentes países, los cuales han donado su cartografía para reproducirla en OSM.

OpenStreetMap utiliza una estructura de datos topológica. Los datos se almacenan en el datum WGS84⁷ lat/lon de proyección de *Mercator*. Los elementos básicos de la cartografía OSM son los listados a continuación:

- Nodos, son puntos que recogen una posición geográfica dada.
- Vías (ways), son una lista de nodos que representa una poli línea o polígono.
- Relaciones, son grupos de nodos, caminos y otras relaciones a las que se pueden asignar determinadas propiedades.
- Etiquetas (tags), se pueden asignar a nodos, caminos o relaciones y constan de una clave (key) y de un valor (ej. highway=trunk).

OSM utiliza *Mapnik*⁸ para generar *quadtiles* proyectadas en EPSG: 900913. Permitiendo usar *OpenLayers* para mostrar estas teselas⁹ de una manera compatible con las de otros proveedores de mapas.

2.1.15 Panoramio

Panoramio es un servicio inaugurado en el 2005 por Joaquín Cuenca y Eduardo Manchón, creciendo vertiginosamente alcanzando en junio de 2007 más de 5 millones de fotografías. Ese mismo mes fue adquirido por Google para seguir enriqueciendo sus programas como Google Earth [24].

Panoramio es un sitio web dedicado a exhibir fotografías de lugares o paisajes que los usuarios crean y georefencian, su objetivo es permitirles a los usuarios de este programa aprender más sobre una zona específica del mapamundi, observando las fotografías que otros usuarios han tomado ahí. Además provee un API¹⁰ implementada en JavaScript que suministra elementos de interfaces de usuario fáciles de utilizar y posibilidades de búsqueda para poder mostrar las fotos de Panoramio en un sitio web.

2.1.16 Web 2.0

La Web 2.0 comprende aplicaciones web, que permiten realizar aplicaciones con contenido altamente interactivo, actualización transparente al usuario y estéticamente mucho más atractivas que lo habitual en aplicaciones Web. Lo anterior se consigue con una mezcla de tecnologías como JavaScript, HTML, CSS, XML, HTTP, HTTPS y tecnologías de

⁷WGS84 son las siglas en inglés de **World Geodetic System 84** (que significa Sistema Geodésico Mundial 1984).[23]

⁸Mapnik [en línea], versión 0.5.1, Programa informático, <http://www.mapnik.org/>

⁹Tesela o celda: es la forma como se divide el espacio

¹⁰<http://www.panoramio.com/api/widget/api.html>, Reproductor API de Panoramio - Muestra las fotos de Panoramio en tu propia web

servidor como PHP, Symfony entre otras. Si a lo anterior le agregamos algunas características como redifusión de contenidos mediante la combinación de sistemas de redes sociales, como FOAF y XFN, blogs, los wikis o las etiquetas (folksonomias), que fomentan la colaboración y el intercambio ágil de información entre los usuarios, tendremos lo que algunos autores denominan aplicaciones Web 2.0 [RIBE, 2007] ¹¹.

En el Caso de DRUBU, se usaron los conceptos de RIA, *Mashup*, y tecnologías Api YOURS, Ajax, *Panoramio*, *OpenStreetMaps*, *OpenLayers* y *Symfony* explicadas en los puntos anteriores del marco teórico. En las siguientes secciones se muestran como estas permitieron el desarrollo del trabajo.

2.1.17 Git

Git es un proyecto *Open Source* cubierto por la *GNU General Public License v2*. Originalmente escrito por *Linus Torvalds* y mantenido por *Junio C. Hamano*.

Es un sistema de control de versiones diseñado para manejar proyectos muy grandes con velocidad y eficiencia, pero igual de apropiado para repositorios pequeños; es especialmente popular con la comunidad *open source*, sirviendo como plataforma de desarrollo para proyectos como el Kernel *Linux*, *Ruby on Rails*, *WINE* o *X.org*.

Git entra la categoría de herramientas de manejo de código fuente distribuido. Cada directorio de trabajo de Git es un repositorio completo con historial y capacidades totales de seguimiento de revisiones, independiente de acceso de red o un servidor central. Aun así, Git es extremadamente rápido y eficiente con el espacio¹².

Para Drubu se usó Github, el cual es un proyecto en línea de alojamiento con Git. Incluye visualizador de código fuente, editor en línea, wikis, entre otros. La ubicación del código fuente con guía de instalación de la aplicación es: <https://github.com/juanchopx2/drubu>.

2.2 Estado del Arte

El uso de *Mashup* es muy reciente. Aunque en el pasado han existido aplicaciones Web que usan información de otros sitios, estas lo hacían de forma poco elegante, sin estandarizar y sin opción de reuso.

¹¹**Ribes Xavier**. El valor de los metadatos y de la inteligencia colectiva. "Telos. Cuadernos de Comunicación e Innovación", n. 73 (2007).

¹²**Briano F**, página web, <http://picandocodigo.net/2008/git-sistema-de-control-de-versiones-distribuido/>, mayo de 2011

Con la aparición de tecnologías Web como *JavaScript*, DOM, Ajax, CSS, *Web Services*, ambientes de desarrollo para web, APIs de aplicaciones online, *toolkits* y *frameworks* de Ajax, unido al avance de los fabricantes en crear y actualizar sus navegadores Web, obteniéndose mejores velocidades y mejor desempeño, más la creciente tendencia de penetración del acceso a internet por medio de banda ancha, ha llevado al surgimiento y adopción de aplicaciones *Mashup*. Algunos de los casos particulares de sitios *Mashup* son:

2.2.1 Routing maps

Desarrollada por el grupo de investigación "*Sistemas de Optimización Aplicada*" (SOA) de la Universidad de Valencia, **RoutingMaps** es una herramienta informática de características profesionales que permite la resolución de problemas reales de flotas de vehículos capacitados (CVRP - *Capacited Vehicle Routing Problem*), cálculo, optimización y gestión de rutas, además este software integra conjuntos de funciones de interés para la gestión logística de los procesos de aprovisionamiento y distribución de mercancías [25].



Figura 13: Funcionamiento de Routing Maps

2.2.2 Mapaspublicar.com

Mapaspublicar.com [26] permite a sus usuarios ubicar fácilmente establecimientos de todo tipo, los cuales se encuentran referenciados en el sitio web de **paginasamarillas.com**. Según la necesidad se puede ubicar directamente la dirección sobre el mapa o por un criterio de búsqueda sobre la razón social de lo que se requiere; como resultado despliega una lista de establecimientos que en simultanea los georeferencia en el mapa y al seleccionar alguno este es señalado con un globo de información contextual sobre el mapa mostrando información extra sobre el establecimiento.

Igualmente el sitio posee los siguientes enlaces:

- *Sitios de Interés*: Muestra los sitios de interés en el mapa que se está observando.
- *Cómo Llegar*: Permite ver la distancia entre dos puntos, ingresando una dirección de partida y una de destino.
- *Imprimir*: Permite al usuario imprimir la vista de su búsqueda para así tener una guía portátil y ligera de desplazamiento.



Figura 14: Captura del funcionamiento de mapaspublicar.com

2.2.3 Palma Routing

Cálculo de rutas con pgRouting, OpenStreetMap y OpenLayers. Es un proyecto de fin de carrera de Roshni Budhrani de la Universidad Abierta de Cataluña. Palma Routing es una aplicación que calcula (usando diferentes algoritmos: A*, Dijkstra, Shooting Star) y muestra el camino a seguir entre dos puntos sobre un mapa en un navegador, como particularidad, el mapa a mostrar corresponde a la de la ciudad de Palma de Mallorca, España, y se establece como puntos de destino para las rutas los atractivos turísticos o puntos de interés de la ciudad

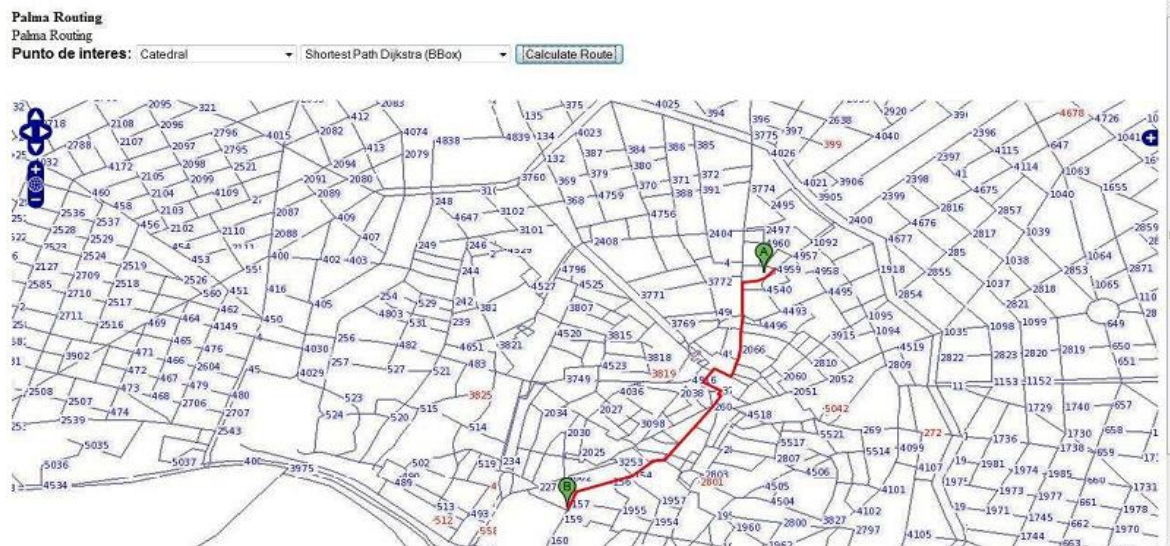


Figura 15: Funcionamiento de palma Routing [27]

3. DESARROLLO DEL SISTEMA

Las actividades desarrolladas durante el proyecto fueron las siguientes:

Actividad	Tarea
1. Recolección de información	• Consultas web • Consulta de material bibliográfico • Consulta de proyectos posiblemente relacionados
2. Análisis y diseño de la aplicación	• Análisis del problema • Definición de requerimientos • Utilización de técnicas de diseño orientado a objetos
3. Selección de tecnologías	• Consultas web • Análisis de las necesidades tecnológicas
4. Implementación	• Desarrollo de la propuesta consignada en las historias de usuario y los elementos de diseño.
5. Pruebas	• Diseño de plan de pruebas. • Aplicación del conjunto de pruebas.

6. Diseño de un caso práctico	• Desarrollo de un caso práctico de rutas empresariales usando la aplicación construida.
7. Análisis de resultado	• Análisis de la herramienta y del caso práctico
8. Presentación de resultados	• Sustentación pública. • Exhibición de la aplicación implementada. • Exhibición del caso práctico.

Tabla 2 Actividades del proyecto

3.1. Recolección de la Información

En esta etapa inicial del proyecto se realizó la recolección del material bibliográfico necesario para el desarrollo de la aplicación entre ellos se destaca:

- *Pro web 2.0.Mashups Remixing Data and Web Services [4].*
- *PHP web 2.0 Mashup projects [29].*
- *A Gentle Introduction to symfony 1.4 [9].*
- *Wiki de OpenStreetMap [28]*
- *OpenLayers documentation [30].*
- *Understanding Mashup development [32].*
- *Mashup man: an online innovator uses an ingenious fusion of imagery and databases to present information in exciting new ways. (THE BEAT) [33].*
- *QMix, Match, Rediscovery: A Mashup Experiment of Knowledge Organization in an Enterprise Environment [34]*

Esta actividad se realizó a lo largo del proyecto y permitió solidificar la base teórica y tener una visión amplia sobre las diversas tecnologías actuales, tanto de servicios como herramientas

3.2. Desarrollo del software

3.2.1 Definición de las historias de usuario

Ver Anexo 1: Historias de usuario

En esta etapa se crearon las historias de usuario. Estas permitieron determinar los requerimientos, las interfaces y el esquema general del proyecto.

A continuación se presenta el formato de las historias de usuario realizado:

Nombre	Nombre de la historia de usuario
Identificador	Identificador de la historia de usuario
Actores	Roles que interactúan con la historia de usuario
Iteración	Iteración en la que el cliente desea que se implemente la historia de usuario.
Tiempo estimado	Estimación realizada por el grupo de desarrollo del esfuerzo necesario para implementar la historia de usuario
Dependencia	Dependencia con otras historias de usuario establecida por el personal de desarrollo
Descripción	Texto explicativo de la historia de usuario
Observaciones	Notas aclaratorias de las actividades descritas de la historia de usuario

Tabla 3: Formato usado para las historias de usuario

3.2.2 Arquitectura del sistema

La arquitectura del sistema identifica la estructura lógica y física que regirá la organización de la aplicación, define de manera abstracta los componentes que llevan a cabo alguna tarea, sus interfaces y la comunicación entre ellos.

Debido a que uno de los objetivos primordiales de desarrollar la herramienta como aplicación Web era separar la lógica de negocios de la lógica de diseño con el fin de extrapolar la arquitectura por capas de las tradicionales aplicaciones de software al mundo de la Web y conseguir una aplicación más estructurada y ágil, se empleó el patrón Modelo-Vista-Controlador (MVC), lo cual da como resultado tres vistas arquitecturales que se presentan a continuación:

3.2.2.1 Vista física

La vista de despliegue o física comprende la topología del sistema, las conexiones físicas y las lógicas, describiendo cada uno de los nodos de la red.

Las características para el servidor Web y el servidor de Base de datos de la aplicación son:

Software

- Sistema operativo basado en *nix.
- Servidor HTTP ((preferiblemente Apache HTTP Server 2.2).
- Sistema manejador de base de datos (*PostgreSQL* 8.4).

- Framework de desarrollo *Symfony* 1.4.
- Motor de enrutamiento *Gosmore*.
- Manejador de versiones Git.

Hardware

Las diferentes características de hardware de los equipos utilizados durante el desarrollo del proyecto se describen a en la siguiente tabla:

Procesador	Memoria RAM	Disco duro
AMD Phenom II X4 925 2.8 Ghz	2,00 GB	80 GB
Intel® Core2 Quad Q6600 a 2,4 GHz	3,5 GB	160 GB
Intel® Pentium M 1,86 GHz	1,5 GB	12GB

Tabla 4: Hardware usado durante el desarrollo del proyecto

Las características a nivel de software para aplicación cliente son:

- Sistema operativo GNU/Linux o Microsoft Windows (2000, XP, Vista, 7).
- Navegador compatible Gecko o Internet Explorer versión 7 o superior.

3.2.2.2 Vista Lógica

En esta vista se presentan las principales características del diseño de las diferentes capas de software utilizadas para la implementación del sistema

- **Capa de interfaz de usuario:** Esta capa provee la interfaz con la que interactúa el usuario; Se encarga de generar el código *JavaScript* que será enviado al navegador de este. Esta capa se implementó utilizando la librería de *OpenLayers* y maneja todo lo relacionado con la presentación y gestión de la información sobre el mapa digital de *OpenStreetMap*.
- **Capa de distribución:** Esta capa tiene a su cargo el manejo de las peticiones de la aplicación Web y de generar las respuestas en el formato JSON. Para el desarrollo del presente proyecto se utilizó Apache, versión 2.2.
- **Capa de lógica del negocio:** Esta capa se comunica con la Capa de interfaz de usuario, para recibir las peticiones y entregar los resultados de las operaciones. De igual manera se comunica con la Capa de acceso a datos para la recuperación y almacenamiento. Esta capa se implementó la versión 5.2 del lenguaje de programación PHP utilizando el *framework* de desarrollo Web *Symfony*, en su versión 1.4 [9].

- **Capa de acceso a los datos:** Representa como se gestionan los datos en la base de datos mediante el manejo de las peticiones al sistema manejador de base de datos (DBMS, por sus siglas en inglés). Esta capa seguirá el diseño de los patrones de la técnica de programación mapeo objeto-relacional (ORM, por sus siglas en inglés), para garantizar la persistencia de los datos en el sistema. El mapeo objeto relacional será manejado con *Doctrine* el cual está integrado con *Symfony*.

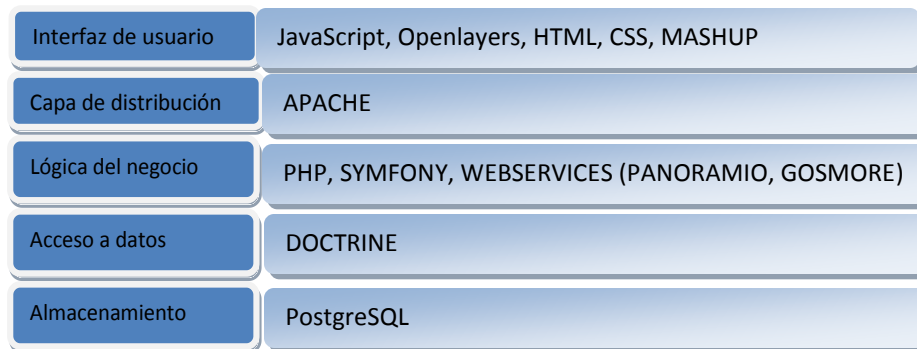


Figura 16: Vista lógica de la aplicación

3.2.3 Diseño de la base de datos del sistema

La base de datos del sistema cuenta con tres tablas para su funcionamiento y ocho tablas generadas por el plugin de *Symfony*, *sfDoctrineGuardplugin* [31], para la gestión de usuarios. No es necesario un modelo complejo debido a que esta herramienta ejecuta la mayor parte de su lógica en el lado del cliente y la mayoría de comunicación realizada con el servidor es para guardar, consultar y recuperar información de las rutas.

En las siguientes figuras se presenta dicho modelo.

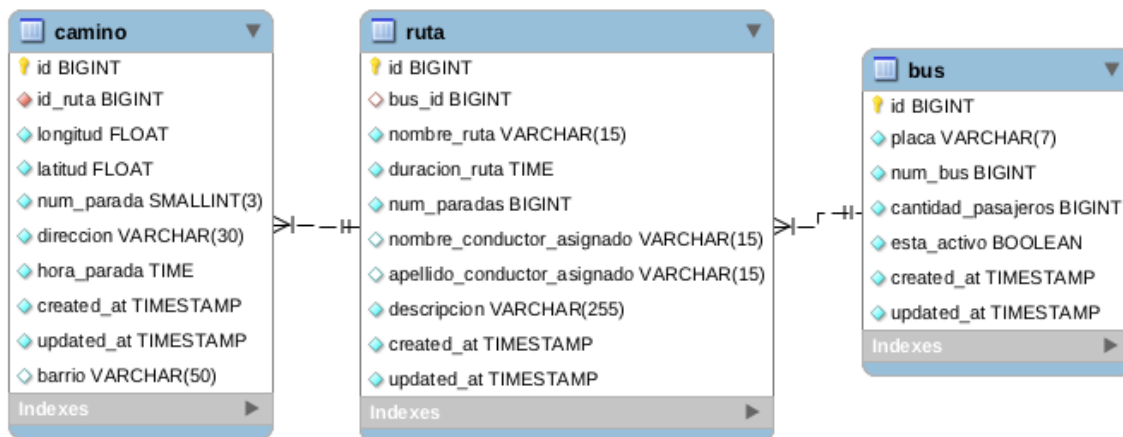


Figura 17: Modelo de datos gestión de rutas y buses

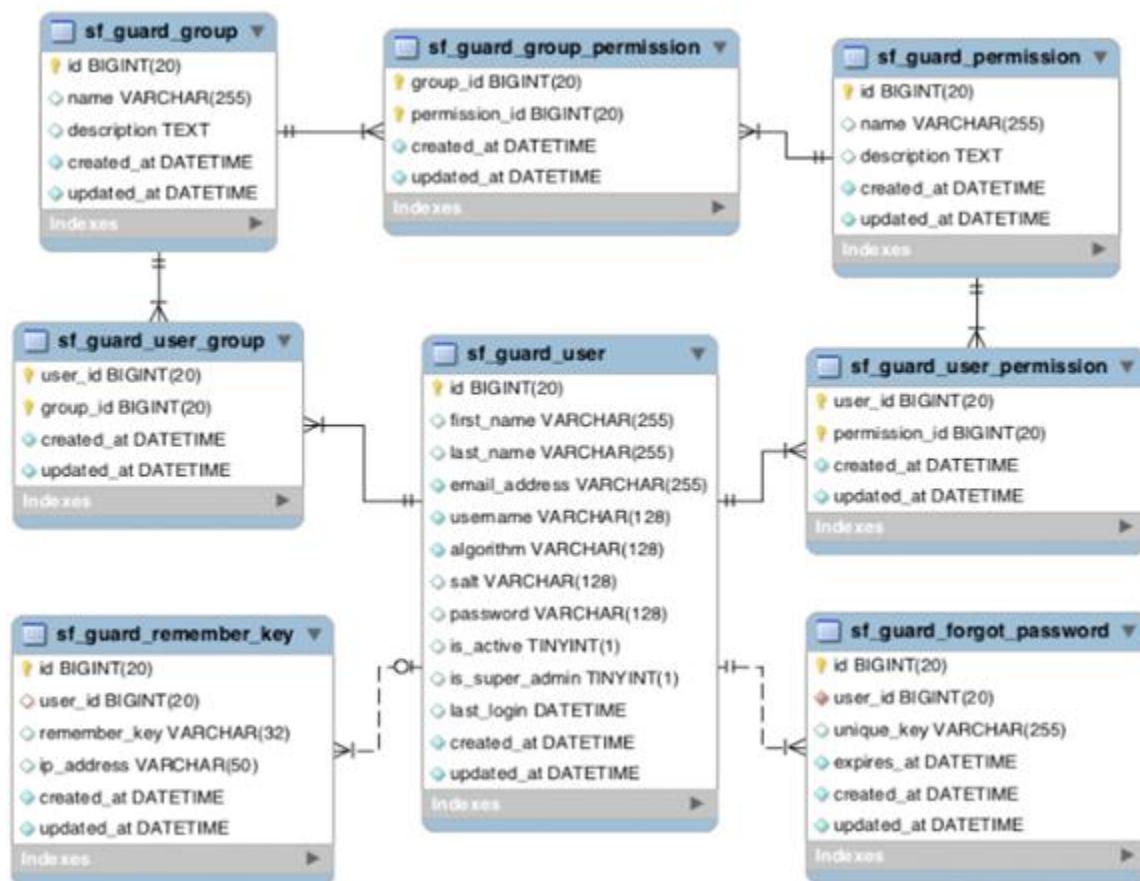


Figura 18: Modelo de datos gestión de usuarios generado por *sfDoctrineGuardplugin*

3.2.4 Detalles de la implementación

El desarrollo de este proyecto fue dividido en dos aplicaciones: **backend** y **frontend**. En el *backend* se encuentran los módulos relacionados con la gestión de la información solo accesible por los usuarios administradores. En el *frontend* se encuentra el módulo con el que interactúa el usuario visualizador.

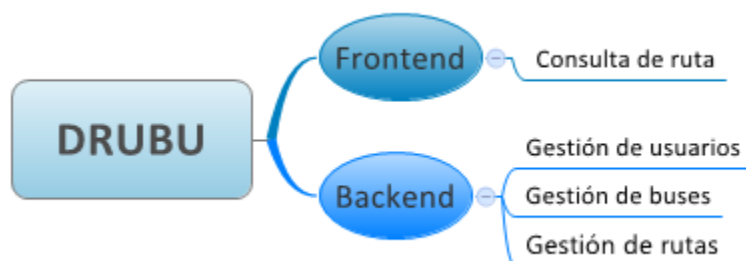


Figura 19: Distribución de módulos del proyecto

3.2.4.1 Backend

3.2.4.1.1 Módulo de gestión de usuarios

Este módulo es generado por el plugin de *Symfony sfDoctrineGuardPlugin*, el cual brinda el modelo (usuario, grupo y objetos de permisos) y los módulos (*backend* y *frontend*) para asegurar una correcta configuración.

Este plugin provee autenticación y seguridad, significa que los usuarios tendrán que iniciar una sesión/autenticarse para acceder a ciertas áreas (seguridad). Diferentes usuarios pueden tener diferentes 'privilegios' (autorización). De esta manera se asegura la aplicación para diferentes tipos de usuarios.

sfDoctrineGuardPlugin añade cuatro módulos al proyecto:

1. *sfGuardAuth*: Módulo de autenticación.
2. *sfGuardGroup*: Módulo de gestión de grupos de usuarios.
3. *sfGuardPermission*: Módulo de gestión de privilegios para acceder a diferentes áreas del proyecto.
4. *sfGuardUser*: Módulo de gestión de usuarios.

3.2.4.1.2 Módulo de gestión de buses

El módulo de gestión de buses permite al usuario administrador ingresar información de la flota de buses con la que cuenta la empresa:

- Placa del bus
- Número interno de bus
- Cantidad de pasajeros
- Habilitado para el servicio.

 [Gestión de usuarios](#) | [Gestión de buses](#) | [Gestión de rutas](#)

Buses List

☐	Id	Placa	Num bus	Cantidad pasajeros	Esta activo	Created at	Updated at	Actions
☐	1	vcj-701	103	25	✓	May 15, 2011 7:51 PM	May 15, 2011 7:51 PM	Edit Delete
☐	2	vch-168	105	35	✓	May 15, 2011 7:52 PM	May 15, 2011 7:52 PM	Edit Delete

2 results

Choose an action ▾

go

New

Figura 20: lista de datos



 [Gestión de usuarios](#) | [Gestión de buses](#) | [Gestión de rutas](#)

New Buses

Placa	
Num bus	
Cantidad pasajeros	
Esta activo	☒

 [Back to list](#)

Figura 21: Creación de bus

3.2.4.1.3 Módulo de gestión de rutas

Para la gestión de rutas fue necesario subdividir el trabajo en dos partes: Gestión de información y dibujado de la ruta sobre el mapa.

En la gestión de información, el usuario administrador completa los datos necesarios para una ruta:

- Nombre de la ruta.
- Cantidad de paradas.
- Tiempo que dura el recorrido.
- Nombre del conductor asignado, si lo hay.
- Bus asignado.
- Descripción general de la ruta.
- Paradas de la ruta:
 - Número de parada.
 - Dirección.
 - Hora de parada.
 - Barrio.
 - Longitud y Latitud (estos valores se obtienen del punto posicionado sobre el mapa).

Para desarrollar el sub-módulo de dibujado de la ruta sobre el mapa se utilizó la librería *OpenLayers*, la cual permite crear capas sobre el mapa de *OpenStreetMap* y sobre ella poner los puntos de parada y dibujar la ruta. Además, ayuda a la captura de la posición de

los puntos para obtener de estos la longitud y latitud para poder ser enviados al motor de enrutamiento, *Gosmore*, encargado de hacer el cálculo de la ruta.

Capturas de pantalla y apartes del código fuente para este módulo:

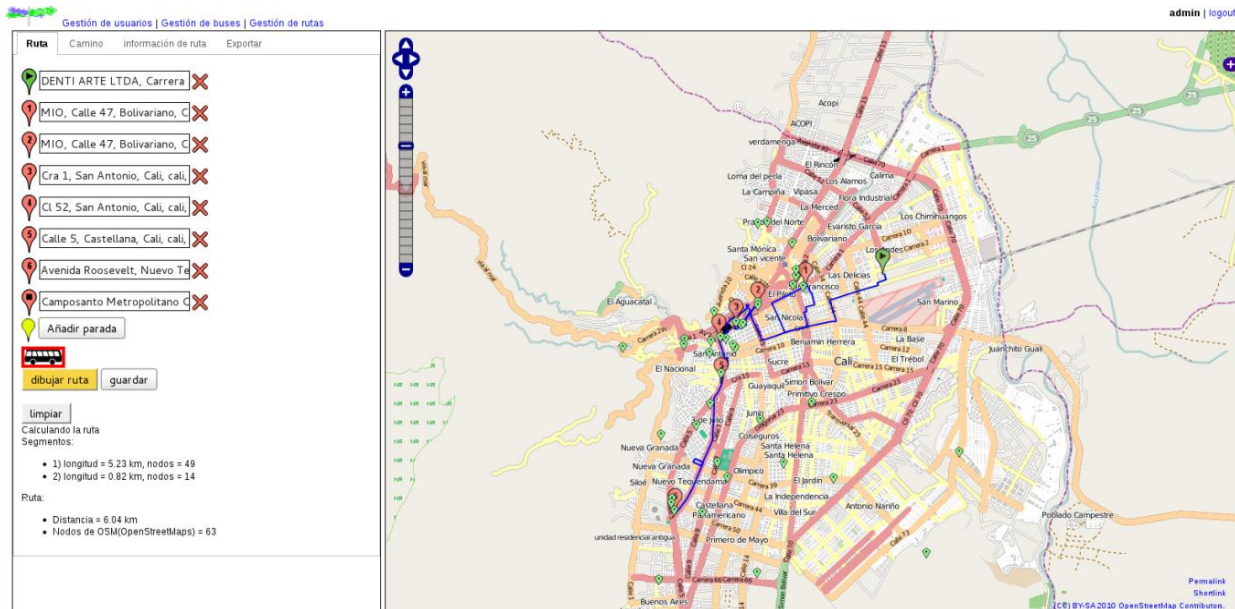


Figura 22: Creación de ruta

Creación del mapa y la ruta: en el archivo IndexSucess.php

```
//Iniciando el mapa
function mapInit(){
    map = createMap("map");

    map.dataLayer = new OpenLayers.Layer("Data", { "visibility": false });
    map.dataLayer.events.register("visibilitychanged", map.dataLayer,
    toggleData);
    map.addLayer(map.dataLayer);
    yourLayers.Layer.events.register("visibilitychanged", yourLayers.Layer,
    toggleRouting);
    //añadiendo las capas al map
    map.addLayers([yourLayers.Layer, yourLayers.Markers]);
    var $_GET = {};
    document.location.search.replace(/\??(?:([^=]+)=([^&]*))&?)/g,
    function() {
        $_GET[arguments[1]] = arguments[2];
    });
    var centre = new OpenLayers.LonLat(parseFloat ($_GET["lon"]),
    parseFloat ($_GET["lat"]));
    if ($_GET["zoom"])
```

```

        setMapCenter(centre, parseFloat ($_GET["zoom"]));
    else {
//ubicando el mapa en la posición (long, lat)
        var long = -76.519;
        var lati = 3.444;
        var z = 14;
        var centro = new OpenLayers.LonLat(long, lati);
        setMapCenter(centro, z);
    }
//actualiza la ubicación del mapa
    updateLocation();
//activación eventos del mouse
    map.events.register("moveend", map, updateLocation);
    map.events.register("changelayer", map, updateLocation);
    handleResize();
    // configurando la barra de rute
    jQuery("#nav_header").tabs();
    // Inicializando la información necesaria para el los waypoints
    YoursInit (yourLayers, map, "route_via", "status", "directions",
"feature_info",
    SelectVehicle, SelectAdjective, updateLocation);
// Cargar el JSON de panoramio para cargar las fotos
    OpenLayers.ProxyHost = "/cgi-bin/proxy.cgi?url=";
    url = "http://www.panoramio.com/map/get_panoramas.php";
    parametros = {
        \'order\':\'popularity\',
        \'set\':\'full\',
        \'from\':0,
        \'to\':50,
        \'minx\': -76.5691,
        \'miny\': 3.3685,
        \'maxx\': -76.4673,
        \'maxy\': 3.5001,
        \'size\':\'thumbnail\'
    }
}

```

Función ubicada en Yours.js encargada de añadir un *waypoint* (o punto de parada) a la lista.

```

this.addWaypoint = function(id) {
    if (id == "from") {
        id = 0;
    } else if (id == "to" || id === undefined) {
        id = this.Waypoints.length;
    }
    // Crea el waypoint
}

```

```

        var waypoint = new Yours.Waypoint(this);
//tipo de waypoint diferente al inicio ó fin
        waypoint.type = "via";
        // añade el waypoint al arreglo de Waypoints
        this.Waypoints.splice(id, 0, waypoint);
        // Corrige el tipo de waypoint añadido
        switch (id) {
            case 0:
                this.Start.type = "via";
                this.Start = this.Waypoints[0];
                this.Start.type = "from";
                break;
            case this.Waypoints.length - 1:
                this.End.type = "via";
                this.End.draw(); // Redraw using 'via' marker
                this.End = this.Waypoints[this.Waypoints.length - 1];
                this.End.type = "to";
                break;
        }
        // Reenumera todos los waypoints empezando por el último
        for (var i = id; i < this.Waypoints.length; i++) {
            var wp = this.Waypoints[i];
            wp.position = i;
            wp.draw();
        }
        // devuelve el waypoint insertado
        return waypoint;
    };

```

Función encargada de dibujar el punto de parada en el mapa.

```

this.draw = function() {
    if (this.lonlat !== undefined) {
        // borra el marker anterior, si existe
        if (this.marker !== undefined) {
            this.route.Markers.removeFeatures([this.marker]);
            this.marker.destroy();
        }

        /* Crea un marker y lo añade a la capa de marcadores */
        this.marker = new OpenLayers.Feature.Vector(
            new OpenLayers.Geometry.Point(this.lonlat.lon, this.lonlat.lat),
            {waypoint: this, image: this.markerUrl()}
        );
        this.route.Markers.addFeatures([this.marker]);
        /* Actualiza la ubicación del marker añadido */
        if (this.name === undefined && this.route.autoroute) {

```

```

        var that = this;
        Yours.NominatimLookup(Yours.lookupMethod.coordToName, "", that,
this.route.map, this.update);
    }
};

```

En la siguiente imagen se muestra la ventana emergente del marcador número 3 mostrando la foto de la ubicación de la parada, en este caso el CAM. En la parte izquierda se encuentra las indicaciones del camino que se debe seguir para la ruta.

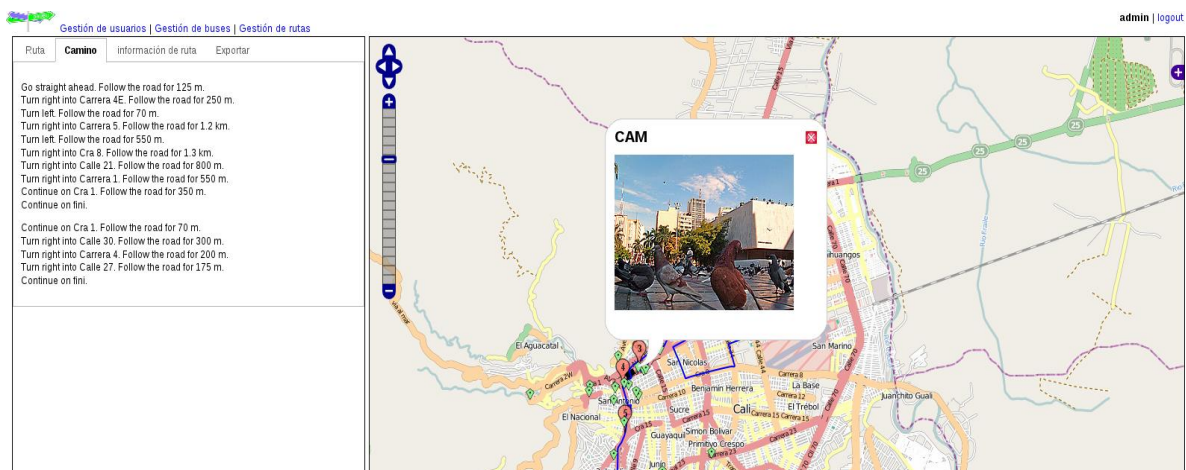


Figura 23: Indicaciones y popup

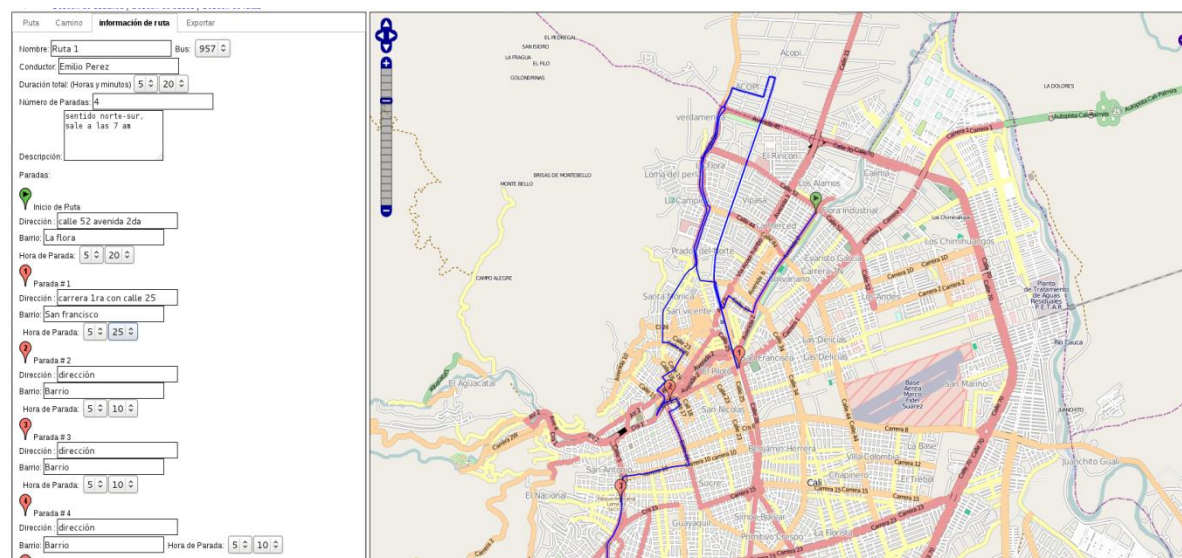


Figura 24: Datos de la ruta

Se generaron seis archivos *javascript*:

yours.js: Contiene toda la configuración de *Openlayers* para usarse con *OpenstreetMap* hace las búsquedas de la locación de un punto con *Nominatim*, posee funciones de configuración de ruta que luego serán llamadas en *routing.js*,

map.js: Llama a las funciones de *yours* encargadas de la inicialización del mapa (creación de capas, manejo de eventos sobre el mapa, etc.).

routing.js: Es el encargado de crear los DOM usando las funciones de *yours.js*. Maneja los waypoints (añade, edita, reorganiza y borra) y la Ruta (estado, configuración, dibujado y eventos).

site.js: Es usado cuando el usuario hace *scrolls/zooms* ya que captura la información para parsearla un href si se da click sobre los links en el mapa (*permalink, shortlink*)

panoramio.js: Contiene configuración y llamado REST¹³ a Panoramio.

drubu.js: se encuentran las funciones propias para el proyecto DRUBU (guardar, editar y borrar ruta, etc.).

Representational State Transfer) o REST es una técnica de arquitectura software para sistemas hipermedia distribuidos como la World Wide Web

3.2.4.2 Frontend

3.2.4.2.1 Módulo de consulta de ruta

Para acceder a este módulo no es necesario estar registrado.

Éste módulo también usa el API de *OpenLayers* para desplegar mapas geográficos digitales con la información de las rutas y sus respectivas paradas.

Aquí el usuario escoge la ruta a visualizar y ésta se despliega en la parte derecha de la aplicación mostrándola sobre el mapa de *OpenStreetMap* de la ciudad de Cali. Además, si el usuario lo desea puede generar un archivo en formato PDF con los datos de las paradas ruta y las indicaciones del camino a seguir.

4. RESULTADOS

Como resultado se obtuvo una herramienta web que tiene como objetivo ser moderna, atractiva en interactiva ideada para los usuarios que necesiten consultar fácilmente las rutas de buses ofrecidas por una empresa, evitando obtener información errónea y tardía ya que DRUBU es centralizada y gestionado por el personal encargado del transporte.

¹³Representational State Transfer. es una técnica de arquitectura software para sistemas hipermedia distribuidos como la World Wide Web

Esta herramienta web ha sido implementada usando la metodología de desarrollo XP y bajo el enfoque Orientado a Objetos.

Entre las tecnologías utilizadas está la librería de *JavaScript OpenLayers* para el desarrollo e interacción con el *Mashup* y visualización de la información sobre el servicio de mapas web *OpenStreetMap*. Por el lado del servidor, la aplicación fue desarrollada utilizando el lenguaje de programación PHP en conjunto con el *framework* de desarrollo *Symfony*, el cual provee características importantes como la generación automática de la estructura de directorios inicial del proyecto, gestión sencilla de las sesiones de usuario, uso y gestión de la caché para la aplicación y abstracción del modelo de datos por el ORM *Doctrine*, integrado al *framework*. En el lado del servidor también fue necesario compilar, instalar y usar *Gosmore* como motor para enrutamiento en mapas viales.

PostgreSQL fue el manejador de base de datos elegido para el almacenamiento de la persistencia ya que permite concurrencia gracias al sistema de *Acceso concurrente multiversión* (MVCC, por sus siglas en inglés). *PostgreSQL* permite que mientras un proceso escribe en una tabla, otros puedan acceder a la misma sin necesidad de ser bloqueados.

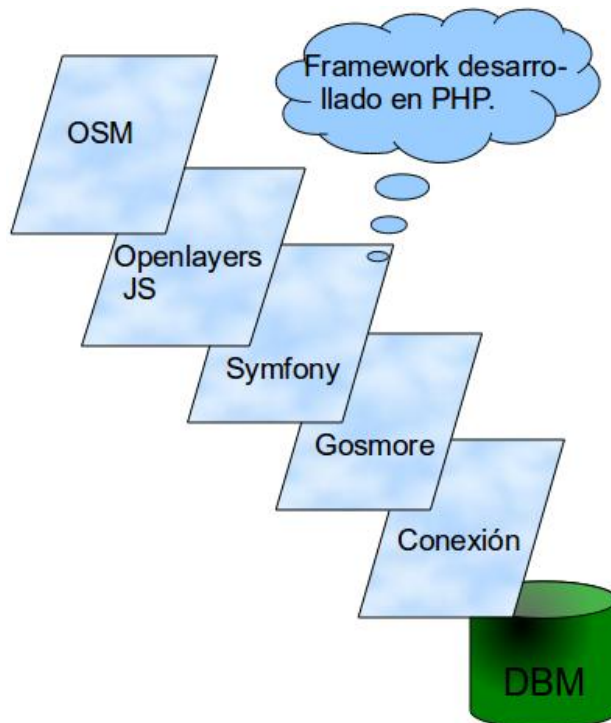


Figura 25: Capas de datos de la aplicación

Las interfaces de usuario se diseñaron con base a las características que deben poseer los entornos RIA, utilizando tecnologías *Ajax*, los cuales permiten simular dentro de una aplicación Web todas las ventajas que ofrece una aplicación de escritorio combinadas con

las de una aplicación Web, además la comunicación con el servidor únicamente se produce para la obtención de datos persistentes en la base de datos con el fin de evitar una continua recarga de páginas ya la sobrecarga de procesos en el lado del servidor.

La aplicación consta de cuatro módulos contruidos con base a los requerimientos planteados al inicio y adecuados a partir de las diferentes sugerencias realizadas por el director y codirector del proyecto y enfocado a los usuarios finales de la aplicación. A continuación se lista:

- **Gestión de usuario:** Este módulo fue generado por el plugin de *Symfony sfDoctrineGuard* el cuál provee autenticación y seguridad.
- **Gestión de buses:** Este es el encargado de gestionar los datos de los buses con los que cuenta la empresa.
- **Gestión de ruta:** Divido en dos sub-módulos uno de ellos es el encargado del CRUD de los datos sobre las rutas y el otro se ocupa de dibujar los puntos y la ruta sobre el mapa digital.
- **Consulta de ruta:** Permite la visualización de una ruta sobre el mapa.

5. CONCLUSIONES Y TRABAJOS FUTUROS

5.1. Conclusiones

- Se desarrolló una aplicación Web que a través del uso de RIA y basándose en el enfoque Web 2.0, permitió incorporar características de aplicaciones de escritorio, ejecutar diferentes operaciones desde el navegador a través de una interfaz simple, intuitiva y agradable al usuario que permitiera la realización de consultas de rutas.
- DRUBU es una herramienta que facilita la centralización de datos evitando a los usuarios confusiones, ya que el recorrido de la ruta se muestra dibujada sobre un mapa y en un texto.
- El uso de varias tecnologías y herramientas de desarrollo integradas demandando un tiempo importante de investigación para obtener un óptimo funcionamiento entre estas.
- Los *Mashups* son parte importante de la denominada Web 2.0, ya que son aplicaciones de alto nivel, utilizan pocos recursos, su tecnología es accesible y no es sólo para programadores ya que la creatividad y el *know-how* es lo principal.

- La implementación del proyecto mediante *Symfony*, permitió que se desarrollaran tareas secundarias de manera rápida, de modo que se contara con más tiempo para dedicarse a las tareas y actividades más importantes para lograr el propósito de la aplicación.
- La utilización de técnicas Ajax fue indispensable para posibilitar la obtención de datos desde el servidor en procesos como crear, guardar, abrir o borrar, evitando que se hiciera una recarga completa del área de trabajo, de modo que se logaran efectos que anteriormente sólo se alcanzaban mediante las herramientas de escritorio.
- El uso de tecnologías de software libre (*symfony, JavaScript, Gosmore, OpenStreetMaps, Php, PostgreSQL*) fue un factor fundamental en el desarrollo del proyecto, ya que otorga tener acceso a potentes herramientas y bibliotecas que además de facilitar algunos aspectos del desarrollo, al combinarlas permitió generar una aplicación robusta. En estas tecnologías, el hecho de que su código fuente se encuentre a disposición permite la creación de comunidades de usuario entorno a ellas, produciendo así una recurso excepcional para encontrar información, el cual fue vital para la realización del proyecto.

5.2. Trabajos futuros

- Implementar un módulo el cuál permita sugerir rutas cercanas a un punto dado que el usuario situé sobre el mapa.
- Habilitar el registro de los usuarios visualizadores en la aplicación para que estos puedan guardar puntos de interés y rutas cercanas a estos.
- Adaptar la visualización de la aplicación navegadores y resoluciones para *smartphones*.
- Utilizar las opciones de geolocalización que incluyen los *smartphones*, para recomendar al usuario rutas cercanas o más adecuadas partiendo de su ubicación exacta.

BIBLIOGRAFÍA

- [1] **Nieto-Santisteban**, María, Ingeniería Web. Construyendo Web Apps., *Universidad de Extremadura. Departamento de Informática*, <http://eisc.univalle.edu.co/materias/WWW/pdf/ingenieria-web-1.pdf>, consultado abril 6 de 2009.
- [2] **Loosly C**, Rich Internet Applications: *Design, Measurement, and Management Challenge*, http://eisc.univalle.edu.co/materias/WWW/RichInternet_5.pdf, consultado abril de 2009.
- [3] Ajaxian, AJAX, en línea, <http://www.ajaxian.com/>, consultado abril de 2009.
- [4] **Yee R**, *Pro web 2.0. Mashups Remixing Data and Web Services*, Apress, febrero de 2008.
- [5] **Pravia C**, ¿Qué es un Mashup?, artículo en línea, <http://blog.internexo.com/2008/02/qu-es-un-mashup.html>, abril de 2009.
- [6] **López J**, Mashup sumando en la red, artículo en línea, <http://www.ibercajalav.net/img/mashups.pdf>, abril de 2009.
- [7] Extreme Programming, en línea, <http://www.extremeprogramming.org/>, abril de 2009.
- [8] **Escribano G**, Introducción al EXTREME PROGRAMMING, <http://www.dsi.uclm.es/asignaturas/42551/trabajosAnteriores/Presentacion-XP.pdf>, abril de 2009.
- [9] **Potencier, Fabien**. Chapter 1 - Introducing Symfony. *A gentle introduction to symfony*, en línea, http://www.symfony-project.org/gentle-introduction/1_4/en/01-Introducing-Symfony, Julio de 2010.
- [10] Doctrine : Introduction What is Doctrine?, en línea, <http://www.doctrine-project.org/projects/orm/1.2/docs/manual/introduction/en>, agosto de 2010

- [11] **Olaya V**, *Sistemas de información geográfica*, versión 1.0, última revisión el 12 enero 2011
- [12] Sistemas de Información Geográfica y Teledetección, en línea, http://concurso.cnice.mec.es/cnice2006/material121/unidad2/td_sig.htm, consultado febrero 2011.
- [13] **Tomlinson R**, *Thinking about GIS: geographic information system planning for managers*, third edition, ISBN 9781589481589, ESRI, Inc. 2007, pág 2.
- [14] **Gonzalvo R**, Meridianos, Paralelos, Coordenadas geográficas, en línea, <http://entendiendolageografia.blogspot.com/2010/11/meridianos-paralelos-coordenadas.html>, consultado febrero 2011.
- [15] Wikipedia, Proyección de Mercator, en línea, http://es.wikipedia.org/wiki/Proyecci%C3%B3n_de_Mercator, consultado febrero 2011
- [16] Wikipedia, Sistema de coordenadas universal transversal mercator, en línea, http://es.wikipedia.org/wiki/Sistema_de_Coordenadas_Universal_Transversal_de_Mercator, consultado febrero 2011
- [17] **Romero J**, Unidad 1 La Tierra y su representación geográfica, en línea, <http://www.juanjoromero.es/blog/2008/07/u1-la-tierra-y-su-representacion-geografica/>, consultado febrero 2011
- [18] the main wiki project page for YOURS, en línea, <http://wiki.openstreetmap.org/wiki/YOURS>, consultado diciembre 2010
- [19] Gosmore, en línea, <http://wiki.openstreetmap.org/wiki/Gosmore>, consultado diciembre de 2011
- [20] Nominatim, en línea, <http://wiki.openstreetmap.org/wiki/Nominatim>, consultado febrero de 2011.
- [21] OpenLayers Wiki, en línea, <http://trac.openlayers.org/wiki> , agosto de 2010
- [22] Wikipedia, OpenStreetMap, en línea, <http://es.wikipedia.org/wiki/OpenStreetMap>, consultado diciembre 2010
- [23] Wikipedia, WGS84, en línea, <http://es.wikipedia.org/wiki/WGS84>, consultado febrero 2011.

- [24] Wikipedia, Panoramio, en línea, <http://es.wikipedia.org/wiki/Panoramio>, consultado mayo 2011
- [25] Routing Maps, en línea, <http://soa.iti.es/software-para-la-optimizacion-de-rutas-y-gestion-de-flotas>, consultado enero de 2011
- [26] Publicar S.A. - Mapaspublicar.com, en línea, <http://www.mapaspublicar.com/>, consultado abril de 2009.
- [27] **Roshni Budhrani**, Trabajo de fin de carrera, *Palma Routing: Cálculo de rutas con pgRouting, OpenStreetMap y OpenLayers*, b_roshni@uoc.edu, mayo 2010
- [28] **Chow Shu-Wai**, *PHP web 2.0 Mashup projects*, Packt publishing, septiembre de 2007 ISBN 978-1-847190-88-8
- [29] Wiki de OpenStreetMap, en línea, http://wiki.openstreetmap.org/wiki/Main_Page, consultado febrero 2011
- [30] OpenLayers Documentation, en línea, <http://docs.openlayers.org/>, consultado febrero de 2011
- [31] Symfony Project, sfDoctrineGuardPlugin 5.0, en línea, <http://www.symfony-project.org/plugins/sfDoctrineGuardPlugin>, consultado noviembre de 2010
- [32] **Yu J, Benatallah B, Casati F and Daniel F.** *Understanding Mashup development*. En: IEEE internet Computing. 5, (12)2008, p. 44.
- [33] **Leone R.** *Mashup man: an online innovator uses an ingenious fusion of imagery and databases to present information in exciting new ways.*(THE BEAT). En: American Journalism Review. 28.6 (Dec 2006): p.10.
- [34] **Meza J and Zhu.** *QMix, Match, Rediscovery: A Mashup Experiment of Knowledge Organization in an Enterprise Environment*. En: International Journal of Knowledge Management. 1, (4)2008, p. 65.