SIGIEM: Sistema de Información Geográfica Integrado, Energético y Medioambiental

UNIVERSIDAD DE CASTILLA-LA MANCHA
ESCUELA SUPERIOR DE INFORMÁTICA

INGENIERÍA
EN INFORMÁTICA

PROYECTO FIN DE CARRERA

SIGIEM: Sistema de Información Geográfica Integrado, Energético y Medioambiental

David Antonio Pérez Zaba

Febrero, 2013

UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INFORMÁTICA

Departamento de Tecnologías y Sistemas de Información

PROYECTO FIN DE CARRERA

SIGIEM: Sistema de Información Geográfica Integrado, Energético y Medioambiental

Autor: David Antonio Pérez Zaba

Director: Dr. Manuel Ángel Serrano Martín

Febrero, 2013

David Antonio Pérez Zaba

Ciudad Real – España

E-mail: dpzaba@gmail.com
Web site: `http://sigiem.uclm.es`

TRIBUNAL:

Presidente:

Vocal 1:

Vocal 2:

Secretario:

FECHA DE DEFENSA:

CALIFICACIÓN:

PRESIDENTE	VOCAL 1	VOCAL 2	SECRETARIO
Fdo.:	Fdo.:	Fdo.:	Fdo.:

Resumen

En los últimos años los Sistema de Información Geográfica (SIG) se han convertido en una de las herramientas más importantes tanto a nivel de investigación como a nivel empresarial, debido fundamentalmente a que la mayoría de la información que se maneja hoy en día tiene asociada una componente geográfica.

Gracias a las nuevas tecnologías se han descubierto nuevas posibilidades y se ha acercado esta herramienta a un público más amplio y diverso, haciendo que hoy en día los SIG sean elementos de consumo.

En este documento se recoge un estudio de las tecnologías y estándares web pertinentes y por otro lado, las herramientas relacionadas con los SIG para mostrar cómo ambas ideas se pueden unir con el fin de construir nuevas herramientas.

Como objetivo principal, en este Proyecto Fin de Carrera, se pretende desarrollar y poner en marcha un SIG que permita visualizar información de energía y medio ambiente de Castilla-La Mancha. De manera que se tratará su extensibilidad y su capacidad de conectar y mostrar información en tiempo real de sensores que se encuentran distribuidos geográficamente.

Índice general

Índice de tablas

Índice de figuras

Índice de listados

Listado de acrónimos

SIGIEM	Sistema de Información Geográfica Integrado, Energético y Medioambiental
PFC	Proyecto Fin de Carrera
GNU	GNU is Not Unix
VGI	Volunteered Geographic Information
CLM	Castilla-La Mancha
JCCM	Junta de Comunidades de Castilla-La Mancha
UCLM	Universidad de Castilla-La Mancha
ESI	Escuela Superior de Informática
IGN	Instituto Geográfico Nacional
CNIG	Centro Nacional de Información Geográfica
IDE-CLM	Infraestructura de Datos Espaciales de Castilla-La Mancha
IDEE	Infraestructura de Datos Espaciales de España
IDR	Instituto de Desarrollo Regional
Libelium	Libelium Comunicaciones Distribuidas
WMS	Web Map Service
CSW	Catalogue Service
UML	Unified Modeling Language
MVC	Modelo Vista Controlador
CPD	Centro de Procesamiento de Datos
JSON	JavaScript Object Notation
BSON	Binary JSON
RDBMS	Relational Database Management System
BDD	Behavior-driven development
TDD	Test-driven development
DDD	Domain-driven development
XP	Extreme Programming

API	Application Programming Interface
SQL	Structured Query Language
ORM	Object-Relational Mapping
LOC	líneas de código
CRUD	Create, Read, Update and Destroy
REST	Representational State Transfer
HTML5	HyperText Markup Language versión 5
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
CSS3	Cascading Style Sheets versión 3
W3C	World Wide Web Consortium
Ajax	Asynchronous Javascript And XML
CSRF	Cross-Site Request Forgery
XSS	Cross-Site Scripting
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
DNS	Domain Name System
URL	Uniform Resource Locator
URI	Uniform Resource Identifier
DOM	Document Object Model
XML	Extensible Markup Language
SIG	Sistema de Información Geográfica
GIS	Geographic information system
OSGeo	Open Source Geospatial Foundation
OGC	Open Geospatial Consortium
QGIS	Quantum GIS
GDAL	Geospatial Data Abstraction Library
OGR	OGR Simple Feature Library
OSM	OpenStreetMap
GPS	Global Positioning System
LGPL	GNU Lesser General Public License

Agradecimientos

A mis padres y a mis hermanos por hacer que sea la persona que soy. A mis amigos de Toledo y a los amigos que he hecho en Ciudad Real, tanto dentro como fuera de la carrera, por aguantarme tal y como soy.

A Manuel, director de este proyecto, por darme la oportunidad de comenzar mi carrera profesional y demostrar de lo que soy capaz.

David

A bú, a los nonos y a mis padres

Capítulo 1

Introducción

Un Sistema de Información Geográfica (SIG), integra hardware, software, gestión, captura, análisis y visualización de todas las formas posibles de información geográficamente referenciada. Permiten ver, entender, interpretar y visualizar datos de muchas maneras revelando relaciones, patrones y tendencias en forma de mapas y gráficos. Además, ayudan a responder preguntas y problemas rápidamente debido a que muestran los datos de una forma entendible y fácilmente compartible [ESR].

Los sistemas de información geográfica se han convertido en los últimos años en una de las herramientas de trabajo más importantes a nivel de investigación y a nivel empresarial. Tanto es así que, se calcula que aproximadamente un 70 % [Ola12] de la información manejada hoy en día está georreferenciada, información que tiene asignada una posición geográfica e información adicional relativa a esa localización. De esta forma, la geografía ha pasado de ser un ámbito particular que guardaba cierta relación con algunos campos concretos, a ser fundamental en muchas otras disciplinas.

Gracias a esto, los SIG son las herramientas más importantes y los responsables de que la elaboración cartográfica haya evolucionado radicalmente en unas décadas. Gracias a las nuevas tecnologías se han descubierto nuevas posibilidades y se ha acercado esta herramienta a un público más amplio y diverso, redefiniendo así la disciplina.

De hecho hoy en día los SIG han pasado a ser elementos de consumo como pueden ser: los navegadores Global Positioning System (GPS), que se utilizan para guiarse durante un trayecto, Google Earth [gooa] que introdujo la visualización 3D del planeta Tierra a usuarios no especializados, Google Maps [goob] u OpenStreetMap [osm], miles de aplicaciones basadas en estas dos herramientas que permiten presentar información de todo tipo como puede ser Foursquare [fou] o múltiples aplicaciones para encontrar viviendas en alquiler.

Con este Proyecto Fin de Carrera (PFC) se pretende desarrollar una aplicación web moderna y usable que permita presentar información sobre Castilla-La Mancha (CLM), integrando información de distintas fuentes y haciendo que se pueda expandir fácilmente con nueva información (haciendo especial hincapié en la información medio ambiental y energética). Además también se pretende mostrar información (y alertas) en tiempo real de sensores que se encuentren dispersos geográficamente.

Para ello la aplicación web se desarrollará utilizando *Ruby on Rails* y las últimas tecnologías y estándares web, haciendo que tenga una correcta visualización en multitud de dispositivos (ayudándose del framework *Bootstrap*). También usará la cartografía de OpenStreetMap (a través de los servicios de CloudMade), una biblioteca *Javascript* (*Leaflet*) para la visualización de la información en el mapa, un almacenamiento de datos eficiente utilizando *MongoDB* y se desarrollará una Application Programming Interface (API) que de acceso a los servicios web para la comunicación con los sensores. Por último, para poder mostrar y comprobar el correcto funcionamiento de la aplicación, se desarrollará un emulador que simulará la comunicación y el comportamiento de los sensores.

Este proyecto se desarrollará dentro de un proyecto de I+D en el grupo de investigación Alarcos de la Escuela Superior de Informática (ESI) de Ciudad Real, y estará financiado por la Junta de Comunidades de Castilla-La Mancha (JCCM) y la Universidad de Castilla-La Mancha (UCLM).

1.1. Estructura del documento

La estructura que sigue este documento se ajusta a la normativa vigente de la ESI para la titulación de Ingeniería en Informática [nor07].

A continuación se listan los capítulos y la finalidad de cada uno de ellos:

Capítulo 1: Introducción

En este primer capítulo se ha realizado una introducción breve sobre la importancia de los SIG, su evolución en los últimos años y el contexto bajo el que se desarrolla este PFC.

Capítulo 2: Objetivos

En este capítulo describe de forma general el objetivo principal del proyecto y se concretan los objetivos específicos. También se presentan las limitaciones que se han tenido a la hora de desarrollar este Sistema de Información Geográfica como se había planteado en un principio.

Capítulo 3: Estado del arte

Muestra el trabajo de investigación y documentación realizado que se ha realizado antes de comenzar el desarrollo del proyecto, con el fin de recopilar y aprovechar técnicas e ideas utilizadas en proyectos similares.

Capítulo 4: Método de trabajo

Se detallan todas las herramientas utilizadas en la realización del proyecto, tanto software como hardware. Además se explica la metodología utilizada y el proceso de desarrollo.

Capítulo 5: Resultados

Muestra el trabajo que se ha realizado, la evolución que ha seguido y el método de desarrollo, algunos resultados obtenidos mediante las técnicas aplicadas y una estimación del coste que ha supuesto el desarrollo de SIGIEM.

Capítulo 6: Conclusiones y propuestas

En el este último capítulo, se realiza una valoración de los objetivos alcanzados, también se realiza una serie de propuestas de trabajo futuro para mejorar algunos aspectos de SIGIEM. Por último se realiza una valoración personal del autor sobre la experiencia durante el desarrollo de este Proyecto Fin de Carrera.

Capítulo 2

Objetivos

En este capítulo se describe el objetivo general y se concretan los objetivos específicos del proyecto y las limitaciones que se han presentado.

2.1. Objetivo general

Desarrollar e implantar un Sistema de Información Geográfica (SIG) con información medio ambiental y energética de Castilla-La Mancha (CLM).

Esta herramienta debe poder añadir nueva información geográfica y manejar información en tiempo real de sensores distribuidos geográficamente. Deberá mostrar alertas si la información que envían los sensores no cumple unos límites establecidos. Y deberá ser fácilmente adaptable para poder integrar datos de otras áreas y temáticas.

La aplicación desarrollada debe ser usable y adaptarse correctamente para su visualización en dispositivos móviles o grandes pantallas.

2.2. Limitaciones

La hipótesis inicial para desarrollar SIGIEM era:

- La información que se desea mostrar debe proceder de una fuente de datos oficial y fiable y para ello se utilizarían datos procedentes del Instituto Geográfico Nacional (IGN) [ign] y de la Infraestructura de Datos Espaciales de Castilla-La Mancha (IDE-CLM) [idea].
- Utilizar sensores de la empresa Libelium Comunicaciones Distribuidas [lib] para transmitir información en tiempo real de algunos puntos de CLM.

2.2.1. Limitaciones con los datos

El IGN dispone de un amplio catálogo de servicios web y de datos de toda España que pone a disposición de los navegantes de cuatro formas:

- Consulta y visualización: permite visualizar, analizar y consultar la información geográfica disponible. Además, permite la visualización de datos procedentes de fuentes externas usando servicios web estándar.

  El software que se utiliza para mostrar la cartografía no dispone de los datos o no permite de manea cómoda ni automática consultar información de regiones específicas y la última actualización del portal fue el 15 de Octubre de 2011. Por lo tanto se descartó.

- Busqueda de datos: agrupa los productos cartográficos en dos grupos.

  - Datos: dispone de mapas históricos topográficos, modelos digitales del terreno, límites municipales en varias escalas, catálogo de nombres de pueblos, etc.

    Al no ser puntos de interés ni sobre la temática de medio ambiente ni energético, se descartaron.

  - Servicios web: muestra varios servicios que utilizan el estándar del OGCWeb Map Service (WMS) y un servicio que utiliza Catalogue Service (CSW).

    Como se indica en la web, los servicios que utilizan el estándar WMS « su objetivo es poder visualizar información geográfica para un área determinada a través de internet. Se encuentra organizado en una o más capas, que pueden visualizarse, ocultarse y combinarse con capas de otros servicios WMS. Opcionalmente se pueden consultar la información asociada a la capa, la leyenda si existe y los atributos disponibles en un punto » [ign]. Una vez estudiado el estándar se descartó puesto que no se puede consultar información de una región, si no de cada coordenada geográfica de cada capa.

    Y el otro servicio que anuncian (CSW) ya no está disponible en la red.

- Centro de descargas: este apartado nos redirige al Centro Nacional de Información Geográfica (CNIG) en el que podemos optar por descargar las vías de carreteras, límites geográficos de los municipios y topónimos de toda España.

  Como Sistema de Información Geográfica Integrado, Energético y Medioambiental está enfocado a Castilla-La Mancha y a mostrar los recursos energéticos y medio ambientales no se utilizó.

- Compra: ofrece a través de tiendas físicas situadas en España comprar un Atlas Nacional de España, Libros y publicaciones técnicas del IGN, fotografías aéreas, cartografía impresa, etc.

Esta forma de acceso físico a los datos se descartó puesto que se deseaba automatizar la tarea y tener acceso a la fuente de dichos datos.

El otro origen de información oficial y más específica de Castilla-La Mancha es el IDE-CLM, que es un subconjunto de la Infraestructura de Datos Espaciales de España (IDEE) [ideb]. Pero ninguno de los servicios que ofrecía están activos actualmente por lo que se descartó tras comprobarlo.

Así que tras estas dos grandes limitaciones se contactó personalmente con el Instituto de Desarrollo Regional (IDR) [idr], que son los que habían desarrollado el IDE-CLM. Tras varios intentos de comunicación con algunos miembros del equipo (muchos miembros ya no forman parte de la UCLM y otros no tenían conocimientos sobre el IDE-CLM) se pudo contactar con personal de I+D adecuado el cual comunicó que los datos de ese proyecto ya no estaban en sus manos, puesto que la JCCM se lo había concedido a una empresa privada y que por lo tanto no tenían acceso a los datos que habían recogido.

Tras esto, se contactó con algunos ministerios de España y con algunas consejerías de Castilla-La Mancha, hasta que la consejería de Agricultura de Castilla-La Mancha indicó unos enlaces donde descargar algunos datos geográficos de interés para este proyecto.

Debido a todas estas limitaciones sobre la obtención de información geográfica, se acortó dramáticamente el tiempo del que se disponía para poder desarrollar SIGIEM de forma adecuada y cómoda.

2.2.2. Limitaciones con los sensores

También se ha tenido problemas con el uso de los sensores que se preveían utilizar.

La empresa encargada de enviarlos e implantarlos no ha cumplido la fecha de entrega. Y debido a que el proyecto SIGIEM debe realizarse en un tiempo acotado (el contrato tiene una duración de siete meses), se decidió seguir adelante, dejándolo abierto de forma que se puedan implantar más adelante.

2.3. Objetivos específicos

El objetivo principal descrito anteriormente, implica una serie de objetivos específicos que serán descritos brevemente a continuación:

2.3.1. Objetivos académicos

- **Aprendizaje de *Ruby on Rails***

  Para desarrollar la aplicación se utilizará el lenguaje de programación *Ruby* y el framework de desarrollo web *Rails*, como se explica en el capítulo 4.

  Esto requiere un aprendizaje y entrenamiento previo para poder realizar el proyecto adecuadamente. Además del aprendizaje de otras herramientas y bibliotecas que faciliten la gestión y desarrollo del proyecto.

- **Aprendizaje de *Javascript***

  Para desarrollar una aplicación moderna y usable, será necesario el uso del lenguaje de programación *Javascript*. Además se utilizará una biblioteca para facilitar el manejo la información geográfica que se mostrará en el navegador del cliente.

  Esto requiere un aprendizaje y entrenamiento previo para poder realizar el proyecto adecuadamente. Además del aprendizaje de frameworks y otras bibliotecas que faciliten su utilización.

- **Estudio de los fundamentos de los SIG**

  Para tratar y manejar correctamente los datos, mapas e información geográfica se necesita conocer los fundamentos básicos de los SIG. Además de conocer, la relación de estos sistemas con las tecnologías web.

- **Estudio de estándares y herramientas de los SIG**

  Como se debe almacenar información geográfica, deberán conocerse los estándares que existen actualmente para codificar esta información y escoger el que mejor se adapte a los requisitos.

  Además también deberán estudiarse herramientas y bibliotecas para la transformación, filtrado, limpieza y gestión de los datos geográficos.

- **Aprendizaje de bases de datos *NoSQL***

  Se debe realizar un estudio y aprendizaje de este nuevo paradigma de almacenamiento de datos. En concreto las bases de datos orientadas a documentos, de forma que ayuden en el almacenaje y la extracción eficiente de los datos.

2.3.2. Objetivos funcionales

- **Acceso eficiente en grandes conjuntos de datos almacenados**

  La herramienta debe aceptar nuevos datos geográficos y estos archivos pueden llegar a ocupar varios *MiB*, además debe hacer frente al constante envío de datos de los sensores.

  Para ello se debe diseñar un esquema de almacenaje de datos que permita su acceso de forma rápida, además de utilizar un sistema de gestión de base de datos que favorezca estas características.

- **Comunicación con los sensores**

  Para la comunicación con los sensores, que se podrán encontrar distribuidos geográficamente y podrán ser de distintos tipos, deberá escogerse el mejor método de comunicación de forma que sea independiente de la plataforma, sistema y forma en la que se encuentren conectados a la red.

- **Control y registro de alertas de los datos que envían los sensores**

  Uno de los aspectos más importantes debido al constante envío de los datos de los sensores es su control. Ya que si sobrepasan unos límites establecidos individualmente para cada uno de ellos, se deberá registrar esa actividad y mostrar una alerta en tiempo real a los administradores de la herramienta.

- **Emulador de sensores**

  Para comprobar que el proyecto desarrollado funciona correctamente y debido a las limitaciones que se han comentado (ver sección 2.2.2), se debe desarrollar un emulador (configurable) que simule el envío de datos de los sensores.

- **Aplicación usable y atractiva**

  Debido a que la aplicación estará disponible para el público general es muy importante que sea atractiva y fácil de utilizar. Además para los gestores de esta herramienta también debe facilitarse su tarea.

  Para ello se utilizarán las últimas tecnologías y estándares web como: HyperText Markup Language versión 5 (HTML5), Cascading Style Sheets versión 3 (CSS3), Asynchronous Javascript And XML (AJAX), etc y algunos frameworks que faciliten la aplicación de estos.

  Al ser una aplicación web existen multitud de dispositivos desde los que se puede acceder, por lo que se deberá desarrollar de forma que se adapte a la mayoría de estos dispositivos posibles.

- **Despliegue de la aplicación**

  Al terminar el desarrollo de este Proyecto Fin de Carrera, la aplicación desarrollada debe desplegarse en un servidor de la Universidad de Castilla-La Mancha (UCLM).

  Para ello habrá que conocer las herramientas adecuadas para hacerlo, asegurar un nivel de calidad de la aplicación y preparar esta máquina para producción.

Capítulo 3

Estado del arte

En este capítulo se muestra el trabajo de investigación y documentación que se ha realizado sobre la temática abordada, antes de comenzar con el desarrollo del Proyecto Fin de Carrera. En primer lugar, se hace una introducción a los Sistema de Información Geográfica y posteriormente a las tecnologías web más utilizadas actualmente. De forma, que se pueda recopilar ideas y aprovechar técnicas utilizadas en proyectos similares.

3.1. Sistema de Información Geográfica

Según [Tom90] un SIG es un elemento que permite «analizar, presentar e interpretar hechos relativos a la superficie terrestre». Otro punto de vista según [SE90] es que es un «sistema de información diseñado para trabajar con datos referenciados mediante coordenadas espaciales o geográficas».

Estas definiciones hoy en día, debido a la gran evolución que han tenido los SIG, se quedan cortas. Al igual que se ha pasado del papel al ordenador, los mapas se han trasladado a los SIG. Pero un SIG no sólo es la nueva forma de cartografía, de la misma forma que no invalida las formas anteriores.

Un Sistema de Información Geográfica (SIG), integra hardware, software, gestión, captura, análisis y visualización de todas las formas posibles de información geográficamente referenciada. Permiten ver, entender, interpretar y visualizar datos de muchas maneras revelando relaciones, patrones y tendencias en forma de mapas y gráficos. Además, ayudan a responder preguntas y problemas rápidamente debido a que muestran los datos de una forma entendible y fácilmente compartible [ESR].

De forma esquematizada, por lo tanto un GIS debe permitir:

- Lectura, edición, almacenamiento y en general gestión de datos espaciales.
- Análisis de los datos. Desde consultas sencillas hasta la construcción de complejos modelos.
- Generación de resultados como informes, gráficos o mapas.

Por lo tanto un SIG es un elemento complejo que engloba una serie de otros elementos

conectados y cada uno de estos desempeña una función distinta (ver sección 3.1.2).

3.1.1. Componentes de un SIG

Los SIG son sistemas complejos que integran una serie de elementos interrelacionados. Una forma habitual de entenderlos es observarlos como si estuviesen formados por tres subsistemas (ver figura 3.1):

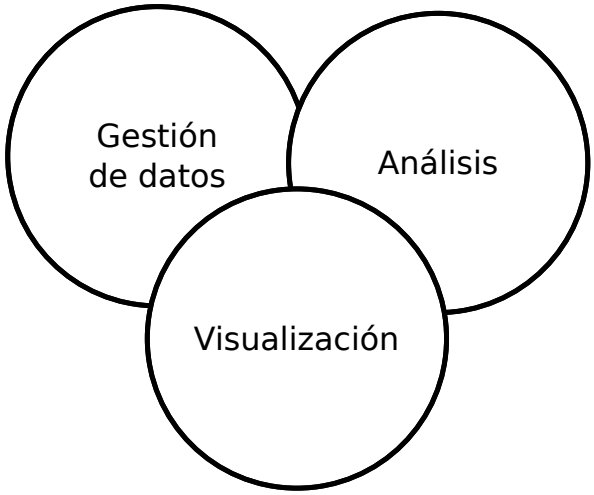

Figura 3.1: Subsistemas de un SIG

- Subsistema de datos: encargado de las operaciones de entrada, salida y gestión de datos. Permite a los otros subsistemas tener acceso a los datos y trabajar en base a ellos.
- Subsistema de visualización y creación de cartografía: crea representaciones de los datos y permite interactuar con estos.
- Subsistema de análisis: métodos y procesos para el análisis de datos geográficos

Otra forma de ver los SIG es fijándose en los elementos de definen su disciplina (ver figura 3.2):

Datos

Son necesarios para hacer que el resto de componentes puedan realizar su trabajo, por lo que resulta obligatorio conocer exhaustivamente los datos y su naturaleza. Es necesario conocer las características fundamentales de los datos geográficos (forma y propiedades), cómo se gestionan y almacenan estos en un entorno digital.

De la misma forma que aumenta el volumen de los datos, lo hacen también sus orígenes y la forma en la que puede recogerse la información geográfica. Por lo cual es de vital importancia saber cómo integrar datos de distintas fuentes y entender cómo afecta esto a las características de dichos datos.

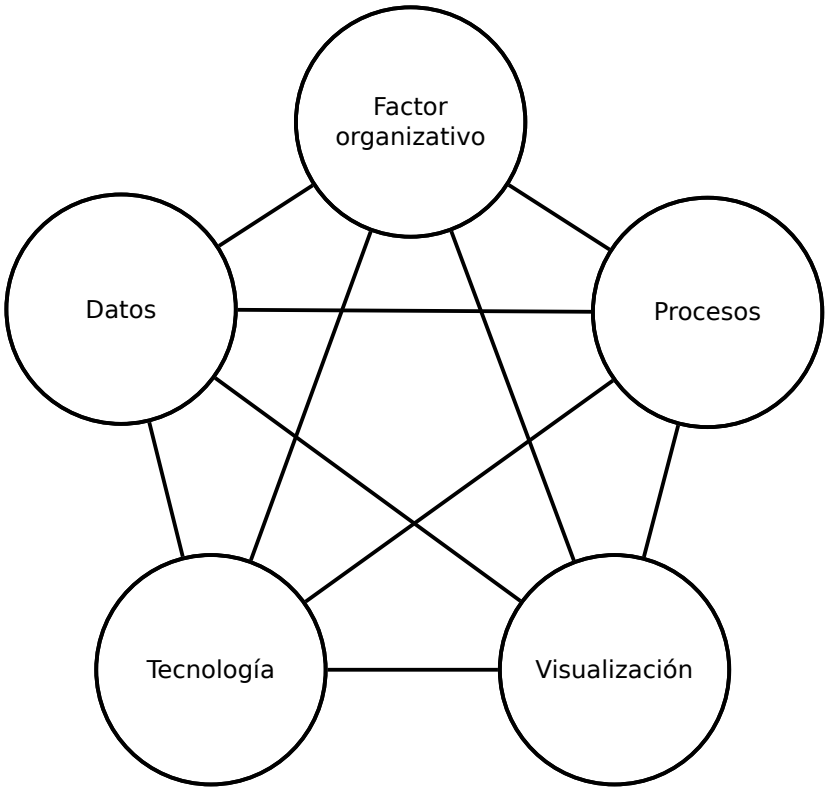

Figura 3.2: Elementos de los SIG

Otro elemento que está cobrando mucha importancia en los últimos días es la calidad de los datos.

Procesos

Debido a que los ordenadores tienen una gran capacidad de cálculo, los SIG siempre incorporan una serie de formulaciones que permiten obtener resultados y análisis de datos espaciales. Estos procesos pueden ser muy sencillos o muy complejos y pueden aplicarse a en general o específicamente a un campo.

La ventaja de incorporar todos los procesos en una herramienta es que se pueden automatizar tareas y gracias a la capacidad de cómputo que se dispone hoy en día, se pueden obtener resultados que no podrían ser obtenidos de otra forma.

Visualización

Cualquier tipo de información puede representarse de forma gráfica, lo que facilita la interpretación de dicha información. Debido a que la información geográfica tiene una naturaleza visual, esta es la forma principal de trabajar con esta información.

Al contrario que un mapa es de naturaleza gráfica, en los SIG se trabaja con datos principalmente de tipo numéricos, por lo que deben ser capaces generar representaciones gráficas a partir de estos, además de generar cartografía clásica, con métodos de diseño cartográfico e impresión de mapas.

Tecnología

Este elemento incluye tanto el hardware como el software utilizado. El hardware representa el elemento físico del SIG y forma la plataforma sobre la que se trabaja. Hoy en día no es necesario grandes computadoras, es suficiente con la mayoría de los ordenadores personales, pero no sólo se incluyen estas herramientas, sino también una serie de periféricos para tareas más concretas como tabletas digitalizadoras, plotters y navegadores GPS.

El software es el encargado de operar y manipular los datos y existen tanto herramientas clásicas, que permiten visualizar, gestionar y analizar datos geográficos, hasta herramientas más especializadas que se centran en alguno de estos campos.

Factor organizativo

Los sistemas SIG requieren una correcta organización y coordinación entre sus distintos elementos. Este factor organizativo ha ido ganando importancia a medida que los SIG han ido evolucionando y se han convertido en sistemas más complejos.

Tienen especial importancia la relación entre las personas que forman el sistemas y la relación que tienen los datos con todos los otros elementos, ya que de una forma u otra interactúan con estos, lo que ha propiciado los intentos de estandarización de los datos.

De echo, hoy en día los SIG ya no se utilizan de forma individual ni por personas con un único perfil. Un ejemplo de ellos es lo que se conoce como Volunteered Geographic Information (VGI) [Goo07], lo que ha sido posible gracias a la popularización y bajo coste de los navegadores GPS y la llegada de la Web 2.0.

3.1.2. Clasificación de los SIG

Los SIG son herramientas muy versátiles y útiles desde muchos puntos de vista. Se les que puede sacarse mucho provecho según el enfoque adoptado. Para clasificar los diferentes ámbitos en los que se aplican, no sólo se debe atender al ámbito en sí, sino también a la funcionalidad que van a desarrollar y según otros aspectos relativos a los elementos que componen el SIG.

Algunos aspectos que permiten clasificarlos son [Ola12]:

- Procesos principales de interés.
- Tipo fundamental de datos con los que trabaja.
- Volumen de variables y datos empleados.
- Precisión de trabajo necesaria.
- Tipo de usuarios esperados.
- Complejidad de las operaciones y tareas desarrolladas.

- Medida en que los SIG responden a las necesidades existentes.

También es importante, para poder clasificar correctamente los SIG, el papel que desempeña. Algunos de los papeles más importantes son:

- Herramienta modeladora: mediante la ejecución de análisis espaciales, puede modelarse realidades geográficas complejas.
- Herramienta para la toma de decisiones: en muchos campos donde se aplican los SIG requieren en algún momento tomar una decisión en función de algunas variables. Representan una utilización de la capacidad de modelar, en la interpretación de los resultados es clave.
- Herramienta para la difusión de información geográfica: debido a la capacidad de exponer los datos geográficos de forma óptima a un público muy amplio, especialmente con las tecnologías web.
- Herramienta centralizadora: es un sistema apto para coordinar las tareas de un equipo de trabajo. Es especialmente útil en grandes empresas, en las que existen varios usuarios con distintos intereses de información y de esta forma proporciona un elemento central para el acceso a la información y garantiza que ese acceso de realiza de forma correcta.

3.1.3. Áreas de aplicación

Algunas de las principales áreas en las que se aplican los SIG son [Ola12]:

Gestión recursos naturales

Es uno de los campos donde se llevan aplicando desde sus inicios, de hecho los primeros SIG se crearon debido a las necesidades de este área. Los SIG son excepcionales en este campo, permitiendo la gestión de la información sobre los distintos recursos naturales y la explotación de gran cantidad de datos (tanto vectoriales como de tipo ráster) de este área de aplicación.

Gestión de riesgos

Los riesgos pueden ser de origen natural o generados por actividades humanas, estos se pueden analizar en un SIG, tratando de evaluar la probabilidad de que se produzcan episodios problemáticos o estudiando su distribución. Los datos empleados en este campo son habitualmente ráster (debido al uso de variables continuas), aunque algunos fenómenos se recogen mejor con los vectoriales, como los incendios, inundaciones o avalanchas.

Ecología

En este campo, los SIG aportan grandes ventajas y herramientas para conocer el estado de poblaciones o comunidades bajo estudio y para estudiar su comportamiento.

Además, también se utilizan para la toma de datos en campo, su posterior creación de cartografía y el manejo de esta.

Las funciones más utilizadas en este ámbito son las de contenido visual: visor de datos, preparación cartográfica, etc. Pero también se utilizan procesos de análisis para buscar información acerca de las poblaciones estudiadas.

Negocios y marketing

Toda actividad de mercado debe estudiarse sobre el espacio en el que tiene lugar para maximizar sus resultados y minimizar la inversión a realizar. Los SIG permiten estudiar la distribución de la competencia, localizar zonas óptimas para una tienda o predecir la evolución de un negocio en función de los de su entorno.

Ciencias sociales

Provee a las ciencias sociales las herramientas para extraer conclusiones y elaborar hipótesis sobre las relaciones entre individuos o grupos. Sobre todo se realizan estadísticas (geoestadística) en las que la componente espacial desarrolla un papel fundamental para entender las relaciones entre diversos individuos de una comunidad.

Planificación

Se utilizan fundamentalmente para la toma de decisiones, puesto que al tener todos los factores geográficos en cuenta, ayudan al estudio para escoger el mejor lugar para realizar una actividad, maximizando beneficios y minimizando posibles impactos negativos.

Militar

La información espacial es fundamental para la actividad militar y el análisis espacial para el estudio de distintos escenarios que proveen los SIG se han utilizado desde sus inicios. Cabe recalcar que el cuerpo militar es uno de los más importantes creadores de cartografía, por lo que además también necesitan las capacidades de edición de datos, digitalización, etc.

3.1.4. Modelos de representación

Existen dos modelos de representación de datos en los SIG: vectorial y ráster. Estos dos formatos se distinguen por la manera en la que almacenan la información geográfica y por su representación.

Raster

En este modelo, la zona que se recoge se divide de forma sistemática en una serie de unidades mínimas (denominadas celdas) y para cada una de estas se recoge la información pertinente que la describe. Aunque la malla de celdas puede contener información sobre variables, lo habitual es que se trate de una única variable (sólo un valor para cada celda).

La división sistemática en celdas según algún patrón, es muy interesante de cara al análisis, puesto que le otorga una relación implícita entre las celdas, siendo contiguas entre sí y sin solaparse.

Aunque en el ejemplo de la figura 3.3, la unidad mínima es un cuadrado, puede tomarse cualquier figura geométrica [DB86], e incluso cada celda podría no tener el mismo tamaño (aunque no son habituales en los SIG).

Vectorial

En este modelo no existen unidades fundamentales en las que se divide la zona recogida, sino que recoge las características y variabilidad de esta mediante formas geométricas. La forma de este modelo, se codifica de modo explícito, a diferencia del raster donde venía implícito en la estructura de la malla.

Utilizando puntos, líneas y polígonos es capaz de modelar el espacio geográfico si se asocia con una serie de valores. Todas estas formas geométricas pueden reducirse en última instancia a puntos, de esta forma, una línea es un conjunto de puntos interconectados en un orden determinado y los polígonos son líneas cerradas. Por lo tanto, todo elemento del espacio queda determinado por una serie de puntos que determinan sus propiedades espaciales y una serie de valores asociados.

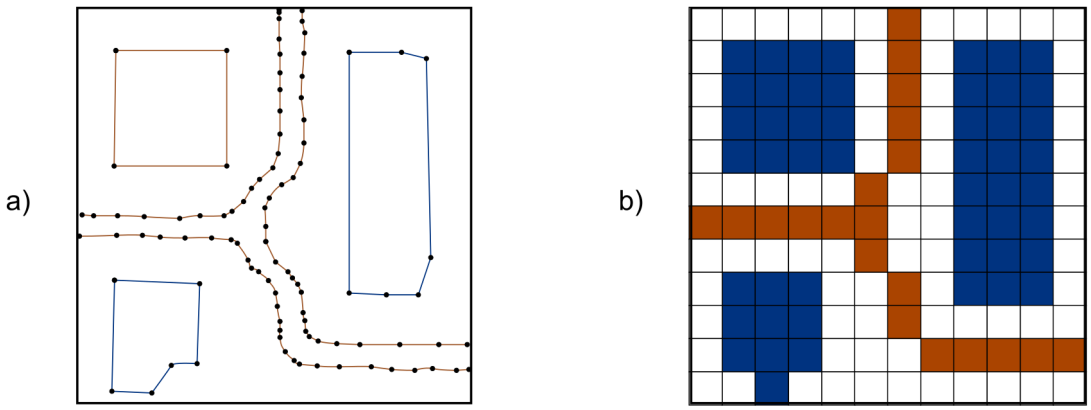

Figura 3.3: Comparación entre los modelos de representación vectorial (a) y ráster (b). Figura tomada de [Ola12]

Diferencias entre el modelo raster y el vectorial

Es fácil intuir que existen grandes diferencias entre estos dos modelos (ver figura 3.3) y que cada uno posee sus ventajas e inconvenientes. En los inicios de los SIG hubo una tendencia a separar estos modelos, de tal forma que si se trabajaba con uno, no se podía trabajar con el otro. Aunque, hoy en día siguen existiendo algunos conflictos, casi todos los SIG permiten trabajar con los dos.

A continuación se presentan las principales diferencias entre estos dos modelos [Ola12]:

- El modelo raster hace énfasis en la características del espacio que recoge (qué y cómo), mientras que el vectorial prioriza la localización.
- El modelo raster tiene una precisión limitada por el tamaño de su unidad mínima (celdas), de esta forma las características del espacio más pequeñas que una celda no pueden ser recogidas y de igual forma, tampoco la variación espacial dentro de una misma celda.
- El número de elementos a almacenar es, en general, muy superior en el modelo raster. Debido a que se divide tola la superficie recogida en las mismas unidades, independientemente de la necesidad de la complejidad de la variable en cada punto o de la necesidad de estudiarla con más o menos detalle.
- El desarrollo de algoritmos de análisis para modelos vectoriales es más difícil, debido a las irregularidades espaciales que presenta. Al contrario que con el modelo raster, que debido a la regularidad de las mallas raster hacen sencillos implementar algoritmos e incluso aquellos que hace uso de combinaciones de varias capas.

3.2. La Web

La Web es una de las múltiples formas de compartir información hoy en día a través de Internet. En ella se observa un gran crecimiento, sobre todo en las últimas décadas, de cantidad de documentación e información almacenada. La comunicación que presenta está basada en el protocolo Hypertext Transfer Protocol (HTTP), mediante el cual se define la sintaxis y la semántica que utilizan elementos de software de toda la arquitectura web para sus transacciones comunicativas. En la Web existen dos roles distinguidos: por un lado el servidor que proporciona documentos y el cliente que los pide y consume. El servidor es pues el computador que aloja la información que el cliente necesita y demanda.

El funcionamiento de esta comunicación se lleva a cabo de una forma específica. Por un lado el servidor Web se encuentra a la espera de cualquier petición realizada por parte del cliente. Tras ello, al detectarlas, las procesa y responde rápidamente. Todas estas transacciones de información son bidireccionales, es por ello que el papel del cliente es clave para poder establecer una forma de crear Web que resulte útil al cliente.

Los clientes envían y reciben los datos necesarios al servidor y desde el mismo, independientemente del navegador que estén utilizando, esto significa que sus navegadores deben de ser capaces de interpretar la información pertinente, al igual que el servidor debe entender las peticiones. La tecnología cliente-servidor permite que en el caso de que lo que sirva al navegador sea una página estática, lo que el usuario va visualizar no tiene por qué ser exactamente lo que vería si estuviese en el lado del servidor, ya que lo que realmente sucede cuando se navega es que se descarga una copia de la respuesta del servidor carpeta del lado del cliente y se visualiza con las capacidades de éste.

Además la comunicación a través de Internet necesita que cada computador tenga una dirección Internet Protocol (IP). Para realizar una petición a un servidor Web, bastaría con utilizar la dirección IP que dicho servidor tenga asignada. Para facilitar esa tarea, se hace uso de un sistema de nombres de dominio Domain Name System (DNS), que permiten, utilizando un nombre conocido (en lenguaje prácticamente natural y legible para el ser humano), obtener la dirección IP de servicios Web. Es posible que más de una dirección Web corresponda con un mismo servidor por lo que se juega con la eficiencia de las comunicaciones.

La Web, a pesar de que en los inicios se ideó para la transmisión de documentos de texto, actualmente permite otros contenidos como vídeos, imágenes o audio. Así mismo, temas como la mensajería han incorporado este tipo de contenidos creando una mejor experiencia de usuario. Todo ello es posible gracias al protocolo HTTP, que unifica la forma de comunicarse. A pesar de ello, la forma en la que el contenido se muestra al cliente no es siempre uniforme. De todas formas se debe destacar que existe la posibilidad de que cada navegador muestre de una forma diferente un mismo contenido Web, todo depende de las tecnologías utilizadas por cada uno de ellos.

Por ello se puede resumir que la Web tiene dos puntos de vista: *server-side* y *client-side*, que estos términos se utilizan para diferenciar *quién* realiza el procesamiento de la información (el software del servidor Web o el software cliente) y que aunque la Web tradicional usa sobre todo tecnología del lado del servidor, la Web actual requiere un gran trabajo por parte del software del cliente.

3.2.1. Tecnologías utilizadas en el servidor

El servidor Web es el encargado de escuchar peticiones HTTP y servir contenido a los clientes. Para que esto sea posible, son necesarios como mínimo dos elementos:

- **Computador servidor:** proporciona lo necesario para obtener el acceso a la red y al almacenamiento de información necesarios para servir el contenido que se requiera. Los servidores Web son computadores dedicados que poseen características especiales para obtener con el mínimo hardware posible el mejor rendimiento.
- **Software servidor:** el software servidor se encuentra instalado y ejecutándose continuadamente en el computador cliente. La Programación del lado del servidor consiste en el procesamiento de una petición de un usuario mediante la ejecución de un script en el servidor Web para generar páginas HTML dinámicamente como respuesta.

En los inicios de la Web, los servidores se limitaban a recibir peticiones de páginas HTML. Este tipo de documentos son páginas estáticas con un contenido fijo y almacenado en un medio de almacenamiento secundario. Los navegadores Web interpretan este lenguaje de etiquetado y generan una representación visual que da finalmente lugar a las páginas Web tal y como se muestran en los navegadores. Las transacciones de información que se han comentado se muestran en la figura 3.4.

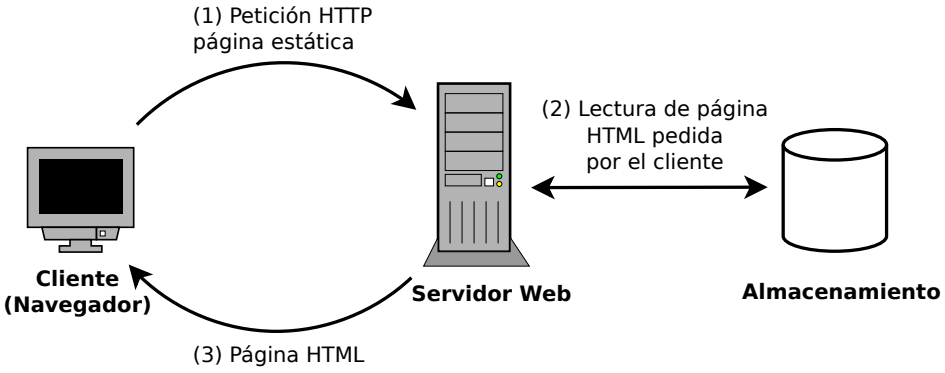

Figura 3.4: Flujo de información en la Web estática

En la actualidad, y sobre todo desde hace unos años, se ha llegado a un punto en el cual las páginas Web son en gran parte dinámicas. Esto significa que, una vez recibida la petición de un cliente, el servidor genera el contenido HTML en tiempo real, para posteriormente enviarlo de la forma tradicional. Este tipo de contenido dinámico ofrecen una mejor interactividad con

el usuario ya que presentan más ventajas y posibilidades. La figura 3.5 muestra el flujo de datos en este caso.

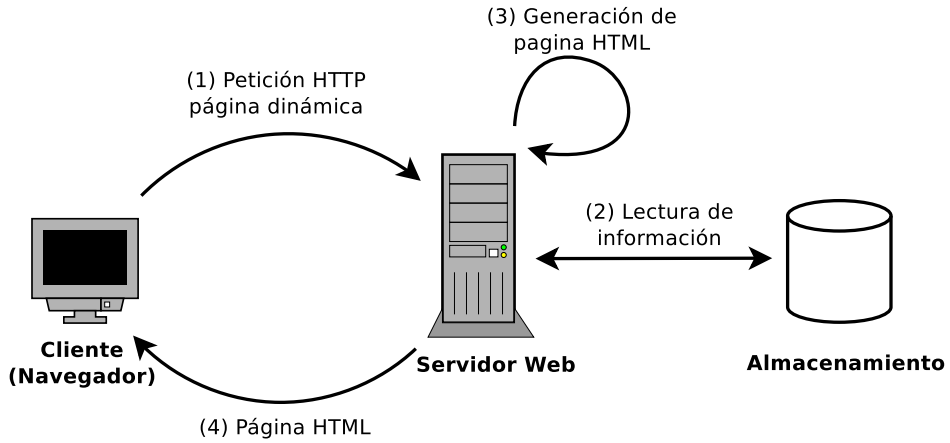

Figura 3.5: Flujo de información en la Web dinámica

Los servidores Web deben ser capaces de interpretar un lenguaje específico para poder generar contenido de forma dinámica y producir así el HTML correspondiente. Para que la información se muestre como el cliente la desea y sea realmente dinámica, es necesario que además exista un elemento de almacenamiento o fuente de información que permita obtener los datos necesarios de una manera eficaz.

Almacenamiento de datos

Actualmente, casi todas las aplicaciones web necesitan de alguna forma almacenar datos, esto puede observarse desde dos perspectivas:

Almacenamiento permanente

Para que se guarden de forma permanente los datos, se suelen utilizar bases de datos. Existen muchas alternativas que implementan este mecanismo, algunas alternativas libres son: *MySQL, PostgreSQL, MongoDB, CouchDB*, ...

Junto a este tipo de almacenamiento, se suele utilizar balanceadores de carga, si el sistema tiene un tráfico elevado, técnicas de replicación, para asegurar la información en caso de fallos. Pero

Almacenamiento temporal

Se utiliza en los sistemas con un tráfico elevado y con necesidades de una latencia baja, no es posible que se utilicen únicamente un almacenamiento persistente. Al igual que en las computadoras, existe un nivel de jerarquía de memoria, en la Web también se pueden intercalar elementos entre las peticiones de los usuarios y el almacenamiento persistente de una aplicación.

Al ser un almacenamiento temporal, no se puede guardar información crítica. Si se desea asegurar que la información almacenada en estos, se transmita al almacenamiento persistente se deben definir unos mecanismos de replicación. Una de las alternativas más utilizada es *Memcached* [mem].

3.2.2. Tecnologías utilizadas en el cliente

El término «programación en el lado del cliente» o *client-side-scripting* se utiliza para hacer referencia al código que se ejecuta en el navegador del lado del cliente y que permite que el usuario lo observe y trate con él. Típicamente, se utilizan tres lenguajes para ello: HTML, CSS (Cascading Style Sheets) y *Javascript*.

HTML

El lenguaje HTML es un lenguaje de marcado predominante para la elaboración de contenido Web que es utilizado para construir modelos de datos que se muestran en los navegadores de los clientes. Es el estándar en la programación Web y todos los navegadores lo soportan. Aunque desde un principio fue construido con la intención de permitir el formateo del texto, en la actualidad se tiende a delegar esta responsabilidad hasta el extremo en las hojas de estilos CSS.

Como HTML es un lenguaje de marcas, no posee características como la generación iterativa de elementos o la respuesta a eventos, es por ello que se vale de otro lenguaje, que queda en el lado del servidor o en *Javascript*. En su última versión, HTML 5 incorpora elementos para la creación de contenido Web multimedia, como se nombraba con anterioridad, de tipo audio, vídeo etc.

CSS

Es un lenguaje de hojas de estilo. Se usa para definir la presentación de un documento estructurado que se encuentre escrito en HTML. Este lenguaje para la presentación de documentos utilizado principalmente en la Web para modificar la apariencia de las páginas, no necesita modificar los archivos HTML asociados.

En comparación con otros estilos que son embebidos en el propio HTML, este lenguaje posee distintas ventajas como:

- Independencia entre el modelo de datos y la presentación.
- Compartición de estilos entre diferentes páginas.
- Adaptación del estilo al medio de visualización dependiendo de las características de este.
- Disminución del consumo de red debido a que no es necesario repetir estilos para elementos de un mismo tipo.

Javascript

El principal lenguaje utilizado para el lado del cliente. Es soportado por la gran mayoría de navegadores modernos que interpretan los scripts incluidos en el código HTML. Se trata pues de un lenguaje interpretado derivado de *ECMAScript*, orientado a objetos, basado en prototipos, imperativo y dinámico. En los últimos años se ha comenzando a utilizar con otros fines como la programación del lado del servidor, con bibliotecas como *Node.js* o la programación de widgets de escritorio, como en *Gnome Shell*.

Asynchronous Javascript And XML (AJAX) [Mel08] es una técnica de programación y desarrollo Web que utiliza *Javascript* y que cada vez se está haciendo más importante. Esta técnica se utiliza con el fin de realizar peticiones a un servidor y así modificar partes de una página, sin necesidad de llevar a cabo una carga completa de la página en cuestión. AJAX es una tecnología asíncrona puesto que los datos se cargan en segundo plano pero esto no tiene por qué ser así siempre, es decir, se pueden realizar peticiones síncronas y utilizar el formato JSON para el envío y recepción de datos.

Al tratarse de una técnica de programación, se basa en los siguientes elementos:

- HTML y CSS para representar la información de la página.
- DOM para interactuar con la información de la página.
- XML para el intercambio de información.
- Objeto *XMLHttpRequest* para la comunicación asíncrona.
- *Javascript* con el fin de poder unir los elementos anteriores.

Gran cantidad de sitios Web utilizan hoy en día *Javascript* para mejorar la experiencia de usuario y permitir una interacción completa. Es posible entonces generar sitios y contenido Web completamente dinámicos, que proporcionan una sensación de fluidez y continuidad durante toda la navegación.

HTML 5 y Javascript Es la tecnología predominante y demandada por numerosas e importantes compañías. Con la aparición de HTML 5 [Pil10], el potencial de *Javascript* se ha visto enormemente incrementado. Los nuevos elementos como `video`, `audio`, `canvas` y *WebGl* ofrecen posibilidades mucho más competitivas. El último ejemplo nombrado, permite que se pueda dibujar en una parte de la página, utilizando sentencias que son básicas. Sobre estos principios, se han construido bibliotecas para trabajar con gráficos 2D [lea] y 3D [3dj], haciendo que sea una completa alternativa a *Flash* (ver figura 3.6).

Figura 3.6: Ejemplo de las nuevas posibilidades que otorga HTML 5 [3dj]

3.2.3. Frameworks para el desarrollo Web

Para poder implementar aplicaciones web complejos, se recurre a gran cantidad de frameworks web. Incluyen bibliotecas para las funciones más comunes en la mayoría de los sistemas, como por ejemplo: uso de plantillas, gestión de las bases de datos y de sesiones. Además, estos proporcionan adaptabilidad al desarrollador, puesto que se pueden modificar según sus necesidades.

Generalmente, casi todos los frameworks de desarrollo web implementan el patrón MVC [Ree79], lo que proporciona una serie de ventajas:

- Fácil reutilización de código fuente.
- Desacoplamiento entre la interfaz, la lógica de la aplicación y la información almacenada.
- Fácil implementación de métodos de acceso a la información.

El comportamiento corriente que se obtiene al utilizar este tipo de frameworks con el patrón MVC, es el siguiente:

- El usuario interactúa con la aplicación web, realizando alguna petición.
- El controlador asociado a esa petición convierte esa petición en una acción del dominio.
- El controlador notifica normalmente al modelo de datos la acción realizada, generando cambios en él.

- El controlador redirige el flujo de información hacia una vista, notificándola de los cambios producidos en el modelo utilizado.
- Se muestra el resultado de la operación al usuario.

Algunos de los frameworks para el desarrollo web más utilizados, divididos según el lenguaje a utilizar son:

- *PHP: CakePHP*, *Symfony*, *Yii* y *Zend*.
- *Python: Django*, *Zope*, *Pylons* y *TurboGears*.
- *Ruby: Rails*, *Merb* y *Sinatra*.
- *Javascript: Backbone.js*, *Cappuccino*, *Google Web Toolkit*, *jQuery* y *YUI*.
- *.NET: ASP.NET MVC*, *Nancy* y *MonoRail*.

3.2.4. Representational State Transfer

Representational State Transfer (REST), es un estilo de arquitectura de software diseñado para sistemas hipermedia distribuidos como la Web, iniciado por Roy Thomas Fielding [Fie00].

Algunas definiciones útiles para su comprensión son:

Estilo de arquitectura

Conjunto coordinado de restricciones que controlan el funcionamiento y características de los elementos de la arquitectura y permiten las relaciones de unos elementos con otros.

Los elementos que forman una arquitectura son :

- **Componente:** unidad abstracta de instrucciones software y estados internos que proporcionan una transformación de los datos a través de su interfaz.
- **Conector:** mecanismo abstracto, que hace posible la comunicación entre los componentes, además de permitir también su coordinación y cooperación.
- **Dato:** elemento de información que se transfiere dese o hacia un componente mediante un conector.

Su motivación al desarrollar REST fue crear un modelo de arquitectura que describiese cómo debería comportarse la Web, persiguiendo unos objetivos concretos [res]:

- Escalabilidad de los componentes de interacción.
- Generalidad de las interfaces.
- Independencia en el desarrollo de componentes.
- Sistemas intermedios para mejorar la seguridad, reducir el tiempo de respuesta y encapsular los sistemas.

Restricciones

Para conseguir los objetivos citados anteriormente, deben cumplirse una serie de restricciones:

Cliente - servidor

Separar la competencia del cliente y el servidor. De esta manera se mejora la portabilidad y la escalabilidad, puesto que se simplifican componentes de ambas partes al no tener que implementar funcionalidades de la otra.

Sin estado

Cada petición del cliente debe contener toda la información necesaria para que el servidor la comprenda y no se necesite mirar datos almacenados sobre el contexto de comunicación. Por lo tanto, el estado se guarda íntegramente en el cliente.

Esta restricción mejora la eficiencia, porque hace más fácil recuperar el sistema debido a errores parciales, escalabilidad, porque al no almacenar los estados entre peticiones los componentes pueden liberar los recursos más rápidamente y visibilidad, puesto que el servidor no tiene que ocuparse de comprobar otros datos antes de atender la petición.

La principal desventaja que supone esta restricción es que puede empeorar el funcionamiento de la red, debido al incremento de tráfico al enviar datos repetidos con cada petición.

Caché

Las respuestas a una petición deberán poderse etiquetarse como *cacheable* o *no cacheable*. De tal forma, que si una respuesta es *cacheable*, entonces se da permiso a la caché para reutilizar la respuesta más tarde si se hace una petición equivalente.

Con esta restricción se evitan peticiones duplicadas al servidor, mejorando así la eficiencia y escalabilidad, así como el tiempo medio de espera de la comunicación.

La principal desventaja, es que si los datos de la caché difieren de los del servidor puede inducir en un mal comportamiento de la aplicación.

Interfaz uniforme

Esta es una de las restricciones novedosas que aporta REST. La principal característica que lo distingue es el énfasis de usar una interfaz uniforme entre los componentes. De esta forma se simplifica la arquitectura general del sistema y la visibilidad de las interacciones.

La principal desventaja, es que al utilizar una interfaz uniforme puede degradar la eficiente, puesto que la información que se transfiere está de una forma estándar, no según las necesidades de la aplicación. Aún así, el interfaz que propone REST está diseñado para ser eficiente con datos hipermedia. Las restricciones que propone para la interfaz son:

- Identificación de recursos.
- Manipulación de recursos a través de sus representaciones.
- Mensajes autodescriptivos.
- Hipermedia como el motor del estado de la aplicación.

Sistema de capas

Este sistema permite tener una arquitectura compuesta por capas jerárquicamente ordenadas, limitando el comportamiento de los componentes (no pueden ver más allá de la capa con la que interactúan) y facilitan la escalabilidad.

Usando el sistema de capas, se pueden simplificar componentes al mover funcionalidades frecuentes hacia sistemas intermedios. Esto también permite un mejor balanceo de carga del sistema al utilizar distintas redes y centros de procesamiento.

La principal desventaja que presenta es que se añade retardos al procesar los datos, debido a la adición de cabeceras de cada capa.

Código bajo demanda

Esta restricción es opcional y consiste en permitir a los clientes tener la funcionalidad de descargar y ejecutar código (en forma de *applets* o *scripts*). La principal desventaja es que reduce la visibilidad y puede influir en la seguridad del sistema.

Elementos arqutectónicos

Datos

La clave de la abstracción que presenta REST es el recurso, cualquier información puede ser un recurso (documento, imágen, sevicio, ...). De tal forma, que cuando un usuario intenta realizar una actividad remota, no accede a un servicio, sino a un recurso.

Los componentes de REST se comunican transfiriendo respresentaciones de un recurso en un formato que se ajusta a un conjunto de tipos estándares [res]. Estos tipos serán seleccionados dinámicamente basándose en las capacidades del receptor y la naturaleza del recurso.

REST utiliza un identificador de recurso para reconocer cada recurso involucrado en una interacción con otro componente. Los conectores de REST proporcionan una interfaz genérica para manipular y acceder a los valores de dichos recursos independientemente del software que se utilice para gestionar la petición.

Representación

En REST los componentes realizan acciones en un recurso utilizando una representación del mismo. Es decir, una captura del estado actual o previsto de un recurso. Una representación no es nada más que una secuencia de bytes más un metadato para describir estos bytes.

Por lo tanto, lo que un cliente solicita al servidor no es un recurso, sino la representación de un recurso. Un recurso es el término abstracto que define a la información que se va a acceder, pero la representación es lo que se transmite al cliente.

Conectores

Estos conectores, utilizan el principio de encapsulación presentando una interfaz abstracta para el acceso a los recursos, separando competencias y ocultando mecanismos de comunicación e implementaciones subyacentes. REST utiliza varios tipos de conectores, los principales son: cliente, servidor y caché. Y los secundarios: resolver y túnel.

Todas las interacciones entre los distintos elementos en REST son sin estado. Cada petición contiene toda la información necesaria para que un conector pueda comprenderla. Esta restricción supone:

- Eliminar de los conectores la necesidad de retener el estado de la aplicación entre dos peticiones. Lo que reduce el uso de recursos y mejora la escalabilidad.
- Permite el procesamiento paralelo de las interacciones.
- Permite comprender y ver a un sistema intermedio una petición.
- Obliga a que en todas las peticiones se incluya información sobre si la petición puede ser *cacheable* o no.

La Web como implementación de REST

Un ejemplo donde se aplica toda esta teoría, es la World Wide Web:

- La Web está compuesta por recursos, cada sitio o aplicación web puede considerarse uno.
- Cada recurso tiene un identificador único global, su URI.
- Mediante HTTP y dada una URL se puede operar sobre estos recursos. La operación se realiza con los verbos del protocolo HTTP y con las cabeceras *Content-Type* y *Accept* se especifican las representaciones que entienden tanto el cliente como el servidor.

  La equivalencia de los verbos de HTTP con las operaciones que harían sería:

  - GET: obtener un recurso.
  - POST: crear, operar sobre un recurso.
  - DELETE: borrar un recurso.
  - PUT: actualizar o cambiar estado de un recurso.
- La representación se especifica mediante el *Mime-Type*. Debido a que la mayoría de estos son estándares (*json, xml*, ...) se facilita la interoperabilidad.

Capítulo 4

Método de trabajo

En este capítulo se presenta la metodología y proceso de desarrollo que se ha seguido para realizar este proyecto y las todas las herramientas tanto software como hardware que se han utilizado.

4.1. Metodología de desarrollo

Se ha elegido para la realización de este Proyecto Fin de Carrera la metodología de **prototipado evolutivo** [Dav92] por los siguientes motivos:

- Tiene una planificación sencilla.
- No tienen porqué conocerse todos los requisitos antes de comenzar el desarrollo y además pueden ir variando con el tiempo.
- En cada iteración se realiza una implementación parcial que cubre alguno de los requisitos conocidos, de maneara que se pueden ir descubriendo el resto de requisitos paulatinamente.
- Debido a las iteraciones se reduce el riesgo y aumenta la probabilidad de desarrollar lo que se desea, además de ver con cada una de estas iteraciones un progreso tangible [BD91].

Es un modelo de ciclo de vida basado en incrementos (ver figura 4.1 [CV]), ya que asume que los requisitos están sujetos a cambios continuos, de manera que con cada iteración evoluciona de acuerdo a la realimentación y nuevos requisitos detectados en cada una de sus versiones. A diferencia de otro tipos de prototipado, en este desarrollo no se desecha el código fuente generado.

Para que se de un refinamiento del prototipo y pueda evolucionar en cada iteración debe estar involucrado totalmente el cliente (en este caso es el director del proyecto).

4.1.1. BDD como proceso de desarrollo

Según su creador Dan North [Nor06], Behavior-driven development (BDD) es un proceso de desarrollo de software basado en los principios de TDD. Combina las técnicas de Test-

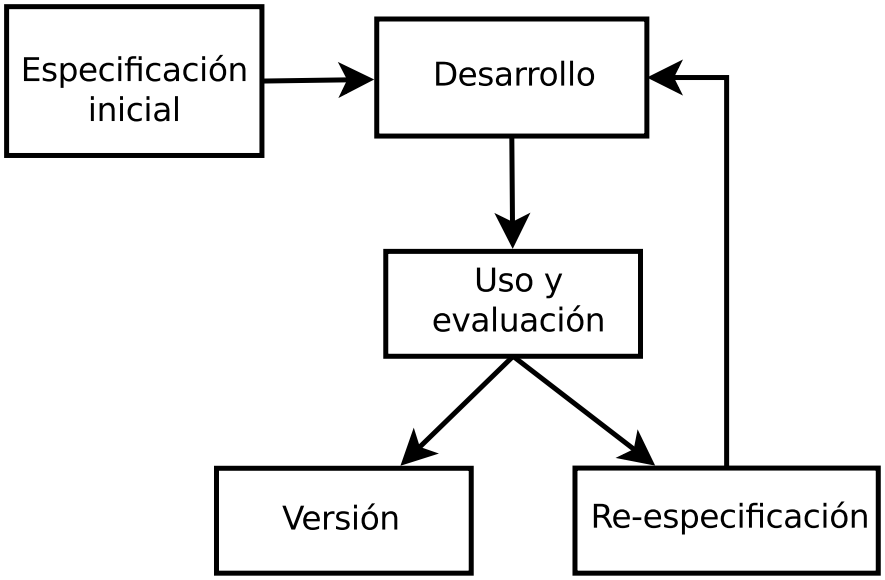

Figura 4.1: Prototipado evolutivo

driven development (TDD) junto con las ideas del diseño y análisis orientado a objetos, y el diseño dirigido por la lógica (DDD). Y de esta forma, trata de proporcionar herramientas compartidas entre los analistas y los desarrolladores de software.

Algunas de los problemas con los que se encontraba Dan North al aplicar TDD eran:

- Desconocimiento de la parte con la que comenzar en el desarrollo.
- Qué se debía probar y qué no.
- Cuánto hay que probar.
- Qué nombre usar en las pruebas.
- Entender porqué fallan las pruebas.

Para explicar los principios de BDD, es necesario explicar los de TDD (ver figura 4.2). De forma resumida, tras haber definido una lista de requisitos, se debe [Bec03]:

- Elegir un requisito de la lista inicial.
- Escribir una prueba.
- Verificar que no se supera la prueba anterior.
- Escribir la implementación mínima que supere la prueba.
- Ejecutar todas las pruebas, para comprobar que el conjunto funciona correctamente.
- Refactorizar el código, para que se mantenga limpio y fácil de entender y no exista código duplicado.

- Actualizar los requisitos, eliminar el requisito implementado de la lista y añadir algún otro si se ha obtenido como resultado de esta iteración y comenzar de nuevo con el ciclo de desarrollo.

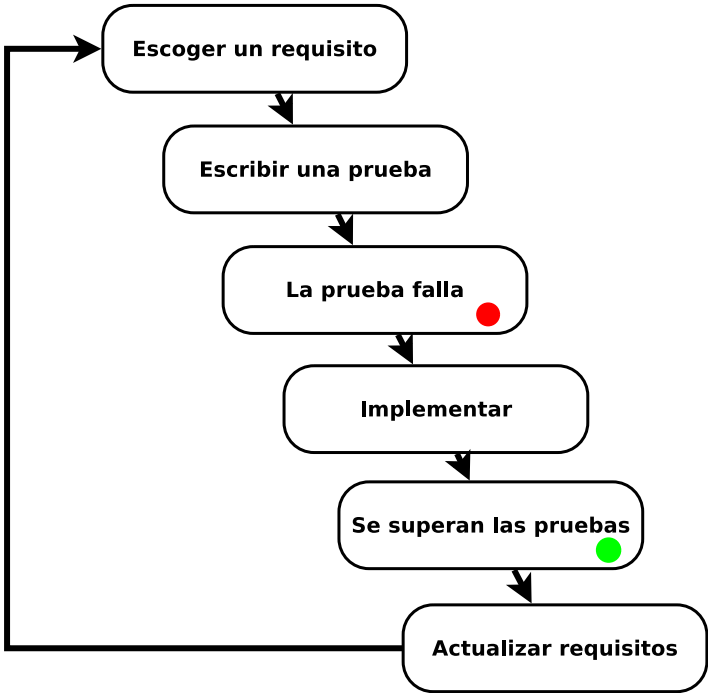

Figura 4.2: Ciclo de desarrollo de TDD

El ciclo de desarrollo de TDD no es muy concreto, no especifica a qué nivel de abstracción se deben realizar las pruebas, ni cómo. BDD especifica que las pruebas deben especificarse en función del comportamiento de la unidad a probar (el comportamiento deseado), acercándose de esta forma a los requisitos de negocio.

Aplicación de BDD

Se ha escogido BDD debido a que se deseaba utilizar una práctica de programación derivada de las metodologías ágiles, en la que se escribiesen las pruebas primero y estuviese enfocada en el comportamiento que debía tener la aplicación (especificado en los requisitos).

Además, gracias al framework *Rspec* de *Ruby* se ha podido aplicar de manera muy práctica y satisfactoria. Y junto con *Capybara*, una biblioteca que simula el comportamiento de los usuarios interactuando con la aplicación web se ha considerado una de las mejores formas de aportar al proyecto fiabilidad y calidad.

Aún así, no siempre se ha podido aplicar esta técnica. Debido a que en algunos momentos no se conocía la forma o tecnología en la que se debía codificar algún requisito y no se sabía tampoco el comportamiento de respuesta que daría el sistema. En estos casos se

ha utilizado Spike [Lef11], término que viene del XP, que de forma resumida se utiliza en los casos en los que se necesita un conocimiento previo que no se posee y para obtenerlo se realiza una investigación hasta encontrarlo. En estos casos las pruebas se han realizado posteriormente.

4.2. Herramientas utilizadas

4.2.1. Herramientas hardware

Máquina de producción

Esta máquina (virtual) se encuentra ubicada en el Centro de Procesamiento de Datos (CPD) de la Escuela Superior de Informática (ESI) de Ciudad Real.

- Máquina virtual:

  RAM: 1GiB

  Procesador: Intel Xeon E5440 a 2.83 GHz

Máquinas de desarrollo

Algunas de las características más relevantes de las máquinas que se han utilizado en el desarrollo son:

- Portatil personal: DELL Vostro 1000

  RAM: 4 GiB

  Procesador: AMD Turion 64 X2 Mobile technology TL60 a 1.8 GHz

  Tarjeta gráfica: ATI Radeon Xpress 1150

- Máquina del laboratorio de Alarcos: HP Z600

  RAM: 24 GiB

  Procesador: Intel Xeon E5620 a 2.4 GHz

  Tarjeta gráfica: Nvidia Quadro 4000

4.2.2. Herramientas software

Sistemas Operativos

En todos los equipos que se mencionan en el apartado de Hardware (ver 4.2.1) se ha utilizado **Ubuntu** [ubu] como sistema operativo. Se ha elegido esta distribución de GNU/Linux frente a otras debido a su facilidad de instalación, rendimiento, estabilidad, y porque se tenía un conocimiento previo.

Se han utilizado varias versiones de este sistema:

- En la máquina de producción (servidor) se ha instalado Ubuntu Server 12.04.
- En la máquina del laboratorio Ubuntu Desktop 12.04.
- En la máquina personal Ubuntu Desktop 12.10.

Lenguajes de programación

Se debe diferenciar entre el lenguaje que se ejecuta en el servidor, y el que se ejecuta en el cliente:

- En el servidor: toda la programación del proyecto se ha realizado en **Ruby** [ruba]. Aunque se utiliza alguna biblioteca que está escrita en C.

  Se escogió el lenguaje *Ruby* (versión 1.9) debido a que es un lenguaje totalmente libre [rubb], flexible, expresivo y que facilita la legibilidad del código. Todas estas características hacen que se pueda utilizar de forma natural y efectiva, además posee un buen framework para el desarrollo web: *Rails* (ver 4.2.2).

  Por otro lado, la elección de *Ruby* también se presentó como un reto personal, debido a que no había trabajado con él en anteriores ocasiones pero lo consideraba adecuado para este proyecto.

- En el cliente: se ha utilizado **Javascript** para poder generar una aplicación dinámica y con efectos, haciendo que esta sea más usable y llamativa.

  También se ha utilizado JQuery [jqu] que es un framework de *Javascript* muy extendido que facilita algunas operaciones como las animaciones, creación de contenido, etc.

Base de datos

Se ha utilizado **MongoDB** [10g] la cual es una base de datos **NOSQL!** orientada a documentos. A continuación se detallan algunas de sus principales características.

Es flexible puesto que almacena los datos en formato JavaScript Object Notation (JSON) (que serializa a BSON). Esto le ortorga un modelo de datos mucho más rico sin los problemas que de asignar los tipos de datos de un lenguaje específico, además como no tiene un esquema fijo hace que sea mucho más fácil de modificar posteriormente que con un esquema de un RDBMS tradicional.

También es fácil de usar al poseer muchas de las características de los RDBMS tradicionales como consultas dinámicas, ordenación, actualizaciones, inserciones condicionales, generación de índices secundarios, etc. Otra ventaja es que se instala con una configuración por defecto haciendo que se pueda utilizar instantáneamente y por lo tanto haciendo más fácil también su instalación y mantenimiento.

Otro aspecto destacable es que se diseñó para ser escalable y de alto rendimiento puesto que al almacenar los datos relacionados en un documento, las consultas pueden ser mucho más rápidas que en un RDBMS donde los datos se almacenan separados en varias tablas, las cuales deben ser consultadas para formar una respuesta. Por otro lado también hace que sea más fácil la escalabilidad lineal de la base de datos aplicando automáticamente la técnica de *Sharding*, de este modo se pueden añadir más máquinas a un cluster haciendo que incremente la capacidad sin tener que parar el sistema en ningún momento.

Control de versiones

Entre los distintos sistemas de control de versiones se ha escogido uno distribuido, en concreto **Git** [git].

Se ha elegido este frente a otros debido a que además de ser código abierto, está muy extendido entre los desarrolladores que utilizan *Ruby* como lenguaje de programación y la mayoría de las bibliotecas que se han utilizados también usan este sistema.

Para alojar el proyecto se ha elegido **Bitbucket** [bit].

Herramientas para el desarrollo

- **GNU Emacs:** [ema] es un editor muy potente, completamente libre, muy popular y caracterizado por su extensibilidad. Se ha utilizado la versión 23.4.1 con una serie de configuraciones que pueden ser encontradas en `http://projects.dpzaba.com/emacs_config/overview`
- **Firefox:** [fir] es una navegador web libre desarrollado por Mozilla que implementa los estándares web actuales y algunos en estado de borrador y que posee multitud de complementos (se ha utilizado la versión nightly 20.0). Para poder depurar el código *Javascript* que se ejecuta en el cliente, se ha utilizado la consola web que este navegador provee además de la extensión Firebug.

Herramientas de testing

Todas las herramientas utilizadas son bibliotecas de *Ruby* disponibles como gemas.

- **Rspec:** [rsp] es una herramienta de test para el lenguaje de programación Ruby. Nació bajo el paradigma BDD y está diseñado para hacer el desarrollo dirigido por pruebas productivo y divertido.
- **Capybara:** [cap] framework para pruebas de aceptación que permite simular el comportamiento de los usuarios en aplicaciones web.
- **Spork:** [spo] es un servidor de pruebas que soporta cualquier framework de Ruby. Es rápido y estable debido a que hace una copia del entorno (permitiendo el uso de cualquier módulo y meta programación) cada vez que se ejecutan los tests.

- **Guard:** [gua] es una herramienta de línea de comandos que permite manejar los eventos de las modificaciones de un sistema de archivos.

  Se utiliza junto con Spork, de manera que se crea un servidor de integración continua local.

- **Factory Girl:** [fac] biblioteca de *Ruby* que proporciona una nueva sintaxis más cómoda de definición de objetos para su utilización en las pruebas.

- **SimpleCov:** [sim] biblioteca de *Ruby* (compatible con la versión 1.9) para analizar y generar una documentación sobre la cobertura de las pruebas. Es compatible con cualquier framework de test de Ruby, se configura rápidamente y el resultado del análisis es claro y con un diseño muy cuidado.

- **Vega:** [veg] es una plataforma multiplataforma de código abierto para probar la seguridad de aplicaciones web. Ayuda a encontrar vulnerabilidades como por ejemplo SQL Injection y Cross-Site Scripting (XSS). Además es extensible mediante una API en lenguaje Javascript.

Servidor web

Se ha elegido utilizar **Apache** [apa], en concreto la versión 2.2.22. Es un servidor HTTP de código abierto, seguro, extensible y eficiente que se ha convertido en el más usado del mundo.

Para que este servidor web pueda trabajar con *Ruby*, se ha escogido el módulo **Phusion Passenger** Open Source [pas] (versión 3.0.13) que es muy recomendado en la comunidad de desarrollo de *Ruby* y Rails.

Documentación y gráficos

- **LaTeX:** [lat] sistema de edición de textos de alta calidad, normalmente se utiliza en los documentos científicos o técnicos. Para seguir la normativa de redacción de este documento, se ha utilizado la clase **arco-pfc** [Ali12], disponible en `https://bitbucket.org/arco_group/arco-pfc`.

- **BibTeX:** [bib] herramienta utilizada para dar formato a la lista de referencias de este documento.

- **JabRef:** [jab] software libre para la gestión bibliográfica que utiliza *BibTeX* como formato nativo.

- **CLOC:** [cloa] herramienta de linea de comandos libre, se utiliza para contar las líneas de código del proyecto.

- **Dia:** [dia] programa libre para la creación de diagramas. Es el utilizado para crear los diagramas de este documento.

- **GIMP:** [gim] programa libre de edición de imágenes digitales. Se ha usado para retocar y crear todas las imágenes que se utilizan en el proyecto.
- **Inkscape:** [ink] editor libre y multiplataforma de gráficos vectoriales que usa estándares de la W3C. Se ha utilizado principalmente para la creación del logo de la aplicación desarrollada.
- **Quick Sequence Diagram Editor:** [sde] es una herramienta de código abierto que permite crear diagramas de secuencia UML mediante descripciones textuales de los objetos y mensajes, utilizando una sintaxis muy fácil.

Software GIS

- En entorno de escritorio: se ha utilizado **Quantum GIS (QGIS)** [qgi] para el procesado y estudio de los datos geográficos. Se ha elegido frente a otras herramientas similares debido a que es software libre bajo la licencia LGPL, es multiplataforma, está mantenido por el OSGeo, utiliza las mismas herramientas para procesar datos geográficos que utiliza este Proyecto Fin de Carrera (GDAL y OGR) además de soportar un amplio número de formatos de archivos vectoriales y raster.
- En entorno Web: se ha utilizado **CloudMade** [clob], esta empresa creada por uno de los fundadores de OpenStreetMap (OSM) proporciona una plataforma con varios productos y servicios para desarrollar aplicaciones que utilizan información geográfica. Los servicios y productos que han sido utilizados son:
  - Leaflet [lea]: es una biblioteca de código abierto en Javascript para mapas interactivos. Se diseñó haciendo hincapié en tres aspectos fundamentales: simplicidad, rendimiento y usabilidad. Funciona en la mayoría de los navegadores modernos y utiliza las últimas características que ofrecen los estándares web HTML5 y CSS3. Está ampliamente extendida, tiene una buena documentación y es fácilmente extensible mediante plugins.
  - Style Editor: es una aplicación web que permite personalizar completamente el estilo de los mapas. Permite cambiar el color, tamaño, puntos de interés y visibilidad de los distintos elementos que aparecen en el mapa.
  - Servicio de mapas: ofrece un alojamiento de la cartografía de OSM además de las distintas personalizaciones que se pueden hacer mediante Style Editor.

Bibliotecas

Debido a su gran número y que algunas ya han sido mencionadas y explicadas en los apartados anteriores, sólo se citarán las más importantes. Para conocer todas las dependencias que tiene este Proyecto Fin de Carrera y qué versión se ha utilizado de cada una de las bibliotecas vea el archivo `Gemfile` (apéndice E).

- **Rails:** [raib] es un framework de código abierto para el desarrollo web con Ruby, sigue la arquitectura Modelo Vista Controlador (MVC) y está enfocado en la productividad y felicidad de los programadores, permitiendo escribir buen código. Se basa en dos principios fundamentales:
  - No te repitas: las definiciones sólo deben hacerse una vez, dado que todos los componentes de Rails están integrados.
  - Convención antes que configuración: sólo hace falta definir la configuración que no es convencional en Rails.
- **OGR:** [ogr] es una biblioteca y herramienta de línea de comandos escrita en C++ que permite leer y escribir en muchos formatos vectoriales. Forma parte de la biblioteca GDAL y se utiliza para transformar los datos geográficos que importa la herramienta a un estándar común.

Capítulo 5

Resultados

En este capítulo se muestra el trabajo realizado, la evolución y desarrollo del proyecto. Se definen los requisitos y las iteraciones de la metodología utilizada, además de los diagramas más interesantes para comprender la arquitectura y la implementación esencial para explicar algunas funcionalidades.

5.1. Especificación de requisitos

Tras el primer mes, el cual se dedicó al estudio de las herramientas, estándares y tecnologías a utilizar se obtuvieron una serie de requisitos.

Como se explica en el capítulo 2, el objetivo de este proyecto es desarrollar e implantar un Sistema de Información Geográfica (SIG) con información medio ambiental y energética de Castilla-La Mancha. Y debido a que no se tenían todos los requisitos definidos al principio y fueron cambiando durante el desarrollo (ver capítulo 4) se eligió la metodología de prototipado evolutivo.

A continuación se listan los requisitos iniciales:

- La aplicación debe ser una aplicación web y estar accesible desde Internet.
- La aplicación debe ser capaz de almacenar datos de forma persistente.
- La pantalla principal deberá mostrar un mapa en el que se pueda superponer la información guardada.
- La aplicación deberá gestionar usuarios con distintos roles.
- Los usuarios deben poder seleccionar los datos que se visualizarán en el mapa.
- Los administradores de la aplicación podrán subir nueva información geográfica.
- Los administradores de la aplicación deberán visualizar si ocurren alertas en tiempo real.

5.2. Iteraciones realizadas

A continuación se muestra un diagrama de las iteraciones (ver figura 5.1) y se enumeran los hitos que se consiguieron durante el desarrollo de SIGIEM:

1. **Prueba de concepto de *Rails* con *MongoDB*** - 9/7/2012.

   Antes de embarcarse en un proyecto de envergadura como SIGIEM, se realizó un proyecto paralelo para comprobar el correcto funcionamiento de *Rails* utilizando *MongoDB* como base de datos a través del ORM *MongoMapper*.

   Se reemplazó el módulo *Active Record* de *Rails*, puesto que este es el encargado del acceso a las bases de datos de tipo SQL. Y tras el éxito de este proyecto paralelo, se comenzó el desarrollo de este SIGIEM.

2. **Prototipo de la interfaz de usuario y prueba de uso de *Leaflet*** - 12/7/2012.

   Tras el logro de utilizar *Rails* con *MongoDB* se creó un prototipo de la interfaz de usuario para ver cual podría ser el aspecto del proyecto. Se utilizó un menú desplegable (utilizando *Bootstrap* [boo]) y se realizó la primera toma de contacto con la biblioteca *Leaflet*, para mostrar un punto en el mapa.

3. **Inicio de sesión de usuarios** - 28/8/2012.

   Se consiguió que los usuarios pudiesen iniciar y cerrar sesión en la aplicación mediante un formulario. En estos momentos el modelo de los usuarios todavía no estaba completamente definido.

4. **Gestión de usuarios** - 6/9/2012.

   En este momento se podía listar los usuarios, eliminar a un usuario y editar la información del usuario que ha iniciado sesión.

5. **Gestión de categorías** - 17/9/2012.

   Se creó el modelo de las categorías en la base de datos y la vista para acceder a este recurso. De esta manera de podían listar las categorías existentes, modificar una categoría o incluso borrar una.

6. **Interfaz de usuario con el menú de categorías** - 25/9/2012.

   Se integró en el menú desplegable de la interfaz de usuario existente con las categorías que existían en la base de datos. De tal forma que si se borraba una, desaparecía del menú y las nuevas aparecían.

7. **Utilización de OGR** - 6/11/2012.

   Se realizó un estudio de herramientas para trabajar con información geográfica, entre muchas posibilidades se eligió el estudio de la biblioteca *GDAL* hasta encontrar la herramienta *OGR* que contiene. Se realizaron varias pruebas de concepto con la herramienta hasta dar con la solución que se ha implementado.

8. **Gestión de datos** - 26/11/2012.

   Tras este hito se había creado la clase de la capa de modelo de los datos y las vistas necesarias para su gestión. De tal forma que se podían listar los datos almacenados, añadir un dato nuevo y borrar datos existentes.

9. **Mostrar información en el mapa a través de una petición AJAX** - 3/12/2012.

   Se consiguió realizar la primera petición AJAX. Se realizó de la forma que *Rails* aconseja (Unobtrusive Javascript) consiguiendo que se mostrase por primera vez en el mapa valores de los datos almacenados en la base de datos.

10. **Gestión de dispositivos** - 10/12/2012.

    Se creó la clase del modelo de los dispositivos y las interfaces de la capa de la vista para poder gestionarlos. De esta forma se podían crear dispositivos, eliminarlos y modificar sus valores. Fue la primera vez que se trató con formularios anidados (nested forms) en *Rails*.

11. **Emulador** - 17/12/2012.

    Para comprobar el correcto funcionamiento de la aplicación se desarrolló un emulador. Este es capaz de comunicarse con la aplicación de la misma forma que lo harían los sensores, mediante una petición POST del protocolo Hypertext Transfer Protocol (HTTP) [htt], gracias a la API que utiliza SIGIEM.

    En este momento, se podía generar datos aleatorios para el sensor de un dispositivo y posteriormente enviarla.

12. **Registro y aviso de las alertas** - 20/12/2012.

    El último hito fue realizar el control y visualización de las alertas. Puesto que después de cada envío de datos de los sensores, hay que comprobar que no se ha sobrepasado o rebajado los límites aceptados que establece cada alerta a su sensor.

    También se realizó una petición AJAX para comprobar las alertas registradas y que de esta forma no se recargue la página web entera que visualiza el usuario. Haciendo que el usuario pueda visualizar en tiempo real si se han producido alertas en algún sensor.

    Además también se desarrolló un controlador y sus respectivas vistas, para que permitiese borrar las alertas que ya se hayan visualizado.

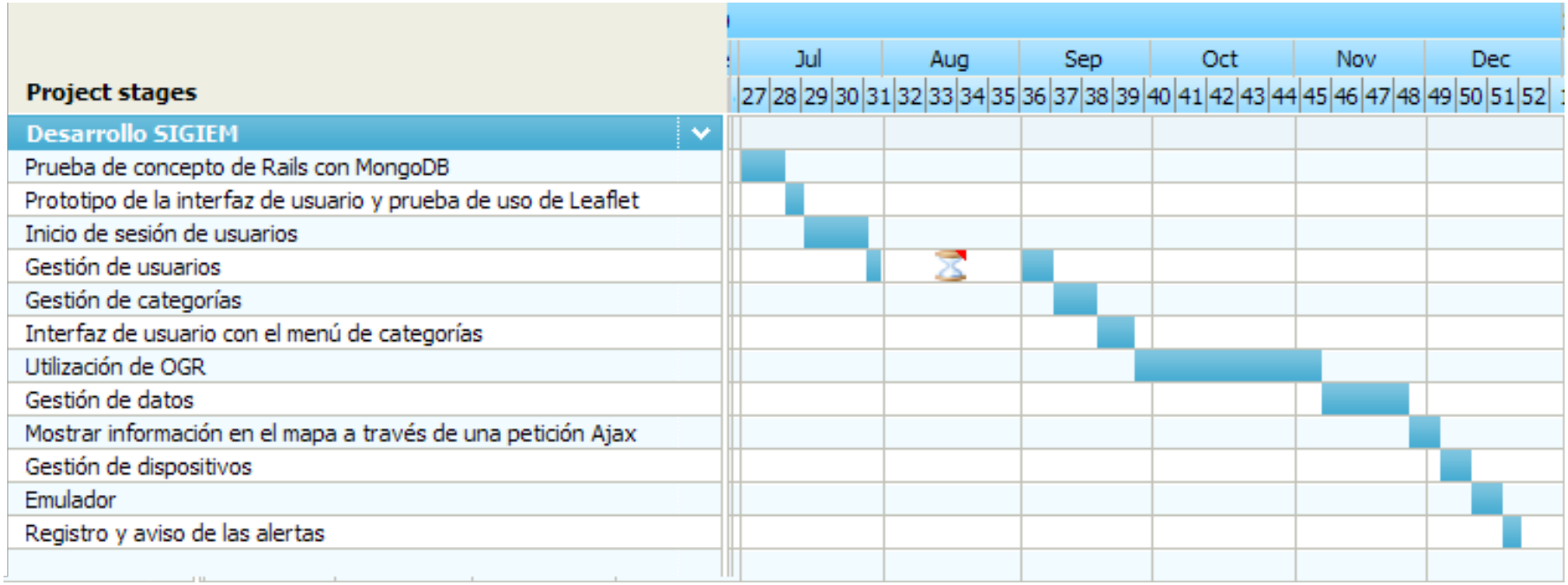

Figura 5.1: Diagrama de las iteraciones en el tiempo

5.3. Casos de uso

Tras el análisis de requisitos inicial y las posteriores modificaciones que se hicieron de los mismos al añadir o eliminar alguno, se obtiene el siguiente diagrama de casos de uso (ver la figura 5.2).

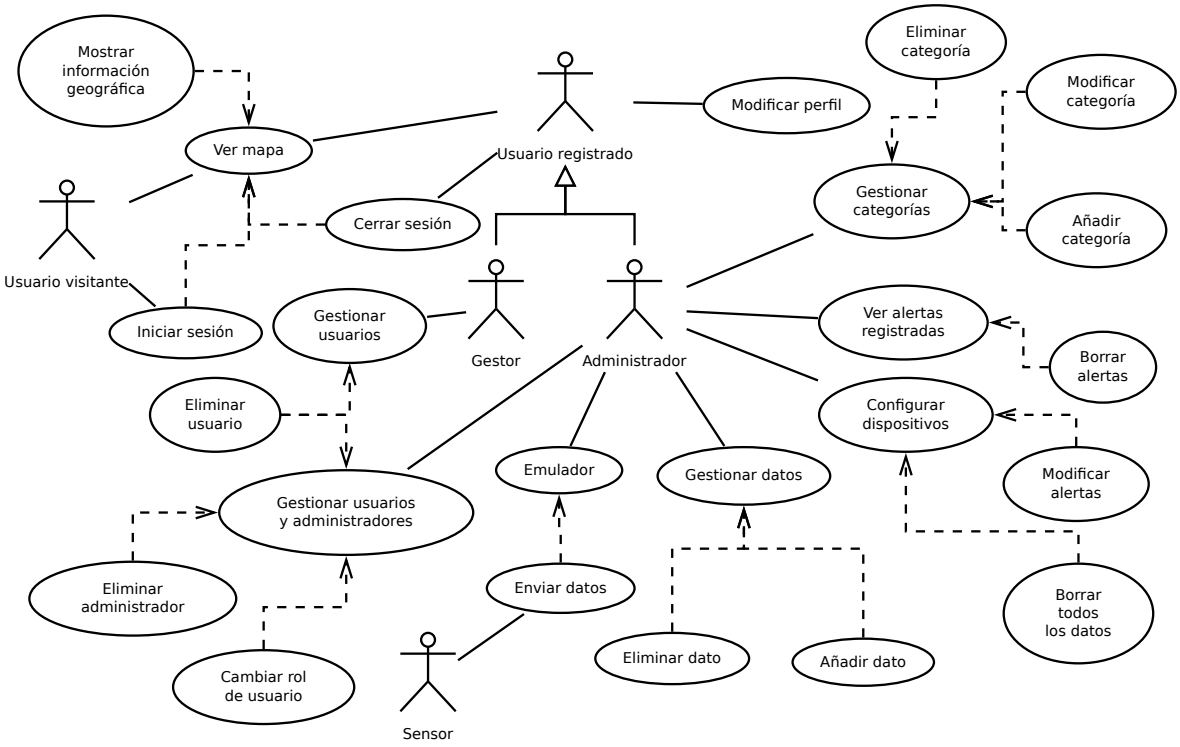

Figura 5.2: Diagrama de casos de uso de SIGIEM

A continuación se detallan los casos de uso principales:

Ver mapa

Todos los usuarios de la aplicación pueden visualizar el mapa y visualizar los elementos del menú que desean.

Se puede observar el diagrama de secuencia de la acción de *Mostrar información geográfica* en la figura 5.3.

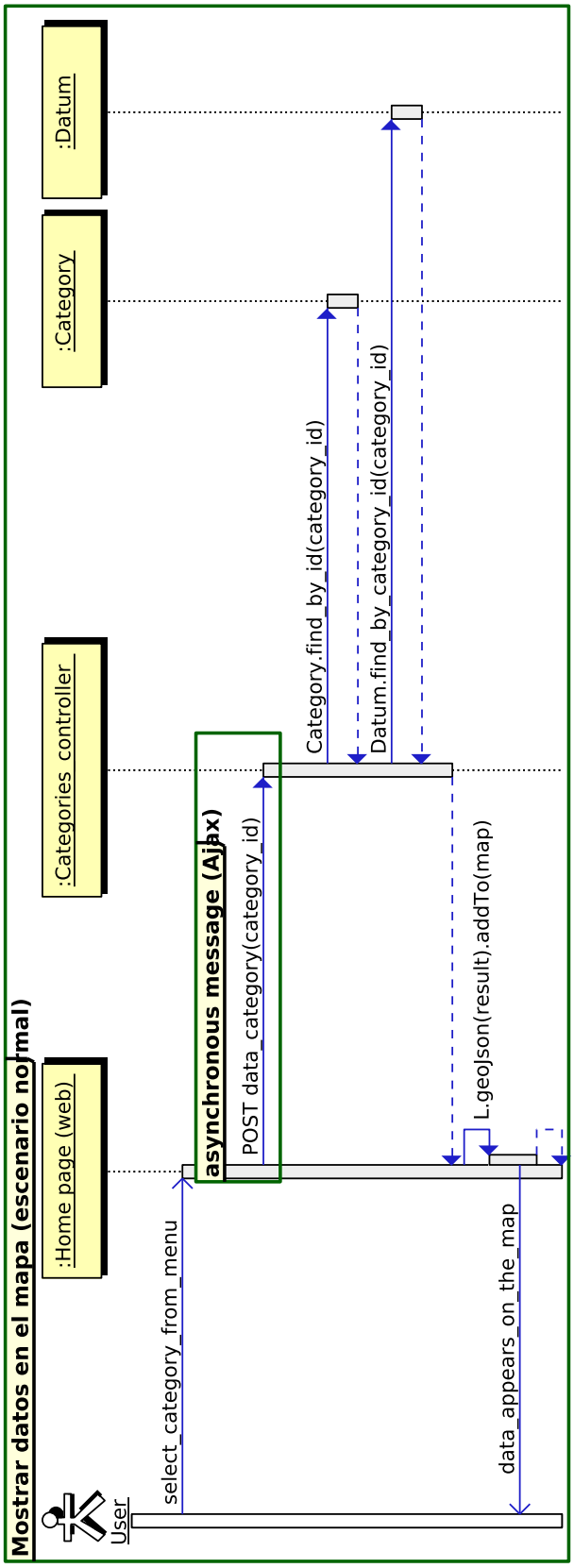

Figura 5.3: Diagrama de secuencia: *Mostrar información geográfica*

Iniciar sesión

Un usuario que no haya iniciado sesión anteriormente, podrá introducir sus datos para iniciar sesión en el sistema.

Modificar perfil

Un usuario que ya haya iniciado sesión puede modificar su perfil. Puede cambiar su nombre, email y contraseña.

Gestionar usuarios

Un usuario gestor, puede listar los usuarios con rol de usuario común que se encuentren registrados en el sistema. Opcionalmente también podrá eliminarlos y cambiar el rol de estos a gestor.

Gestionar usuarios y administradores

Un usuario administrador, puede listar todos los usuarios que se encuentren registrados en el sistema. Opcionalmente también podrá eliminarlos y cambiar el rol de estos.

Gestionar datos

Un usuario administrador, puede listar los datos geográficos almacenados. Opcionalmente también podrá añadir nuevos datos y eliminar alguno ya existente.

Se puede observar el diagrama de secuencia de la acción de *Añadir datos* en la figura 5.4.

Configurar dispositivos

Los usuarios administradores, podrán listar los dispositivos que hayan enviado información al sistema. Opcionalmente también podrán cambiar los valores de las alertas de cada sensor del que dispongan o desactivarlas individualmente.

Gestionar categorías

Los usuarios administradores, podrán listar todas las categorías del sistema. Opcionalmente también podrán añadir nuevas categorías, modificar la jerarquía y nombre de estas y eliminar las ya existentes. Estas categorías serán las que se visualicen como menú en el mapa.

Ver alertas registradas

Los usuarios administradores, podrán listar las alertas registradas en el sistema. Opcionalmente también podrán eliminar individualmente cada alerta ya ocurrida, o incluso eliminar todas las alertas registradas de cada sensor.

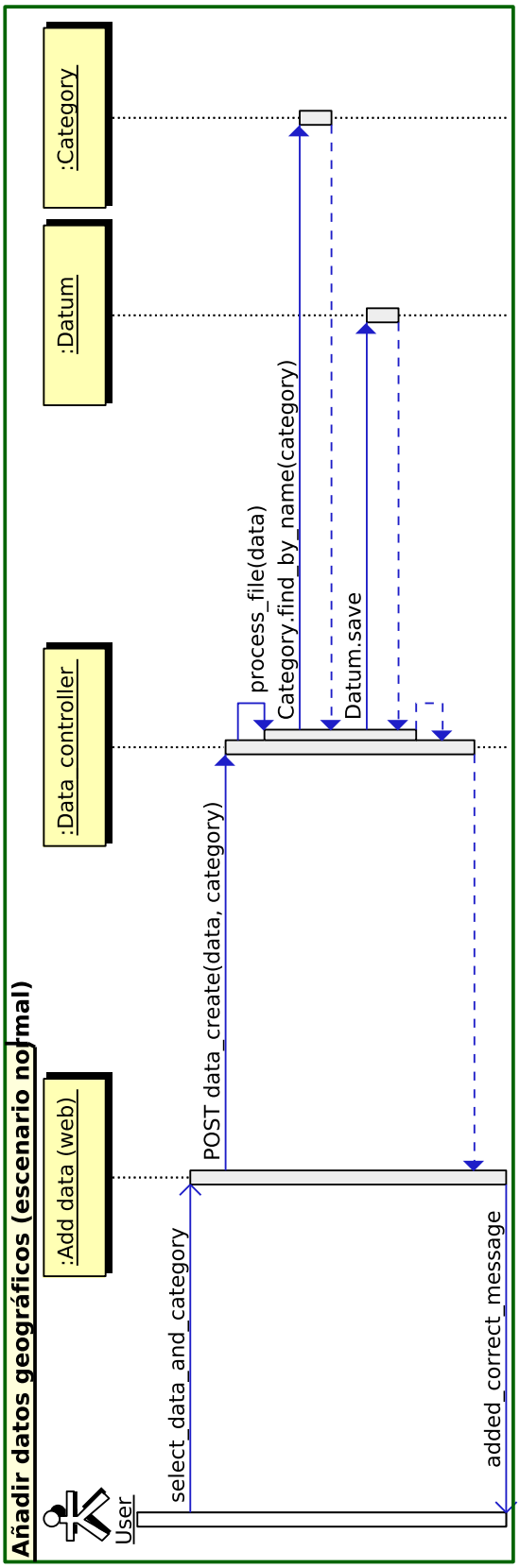

Figura 5.4: Diagrama de secuencia: *Añadir datos geográficos*

Emulador

Los usuarios administradores, pueden acceder al apartado del emulador. Podrán elegir el nombre del dispositivo y el nombre del sensor del que desean generar datos aleatoriamente. Opcionalmente, una vez generados estos datos aleatorios podrán enviarlos al sistema.

Enviar datos

Los sensores y el emulador, podrán enviar datos al sistema de una forma específica.

Se puede observar el diagrama de secuencia de la acción de *Enviar datos* en la figura 5.5.

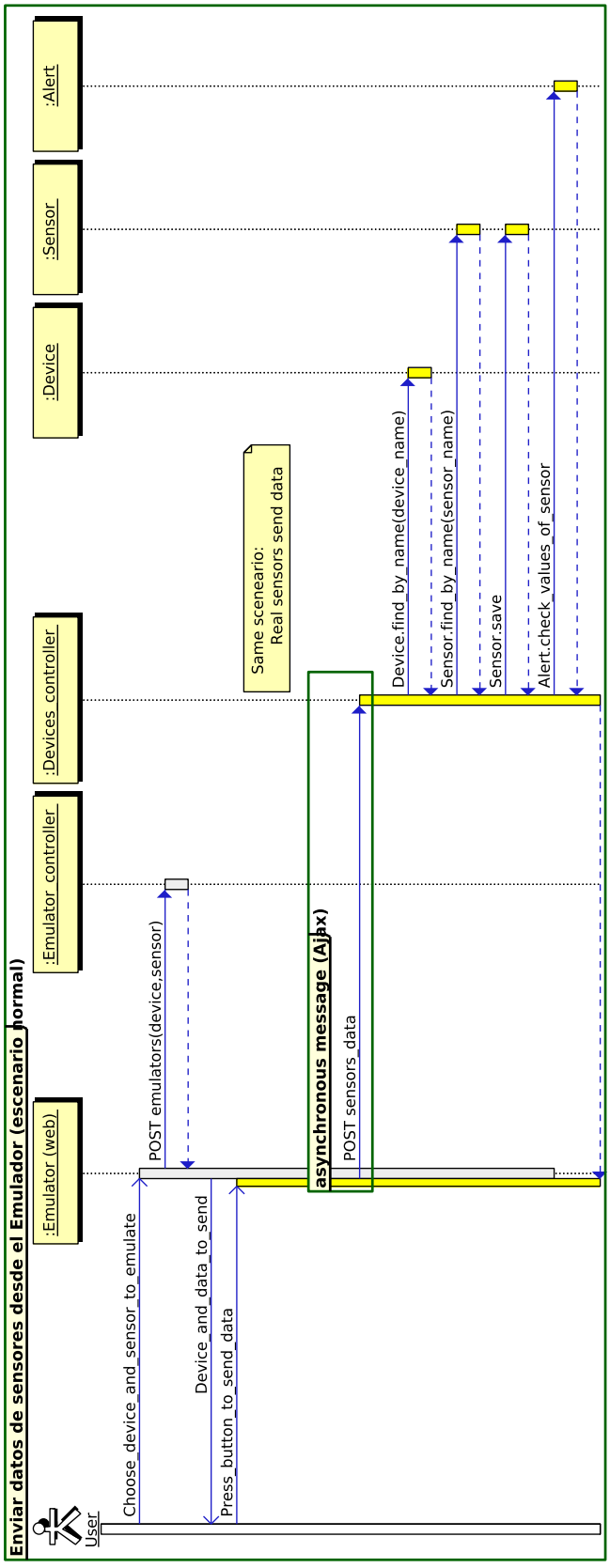

Figura 5.5: Diagrama de secuencia: *Enviar datos*

5.4. Diseño

En este apartado se describen las clases y otros módulos que hay en cada capa del sistema. Se ha utilizado una arquitectura separada en tres capas debido al uso del patrón Modelo Vista Controlador (MVC). A continuación se muestra un diagrama general de la arquitectura de SIGIEM (ver figura 5.6).

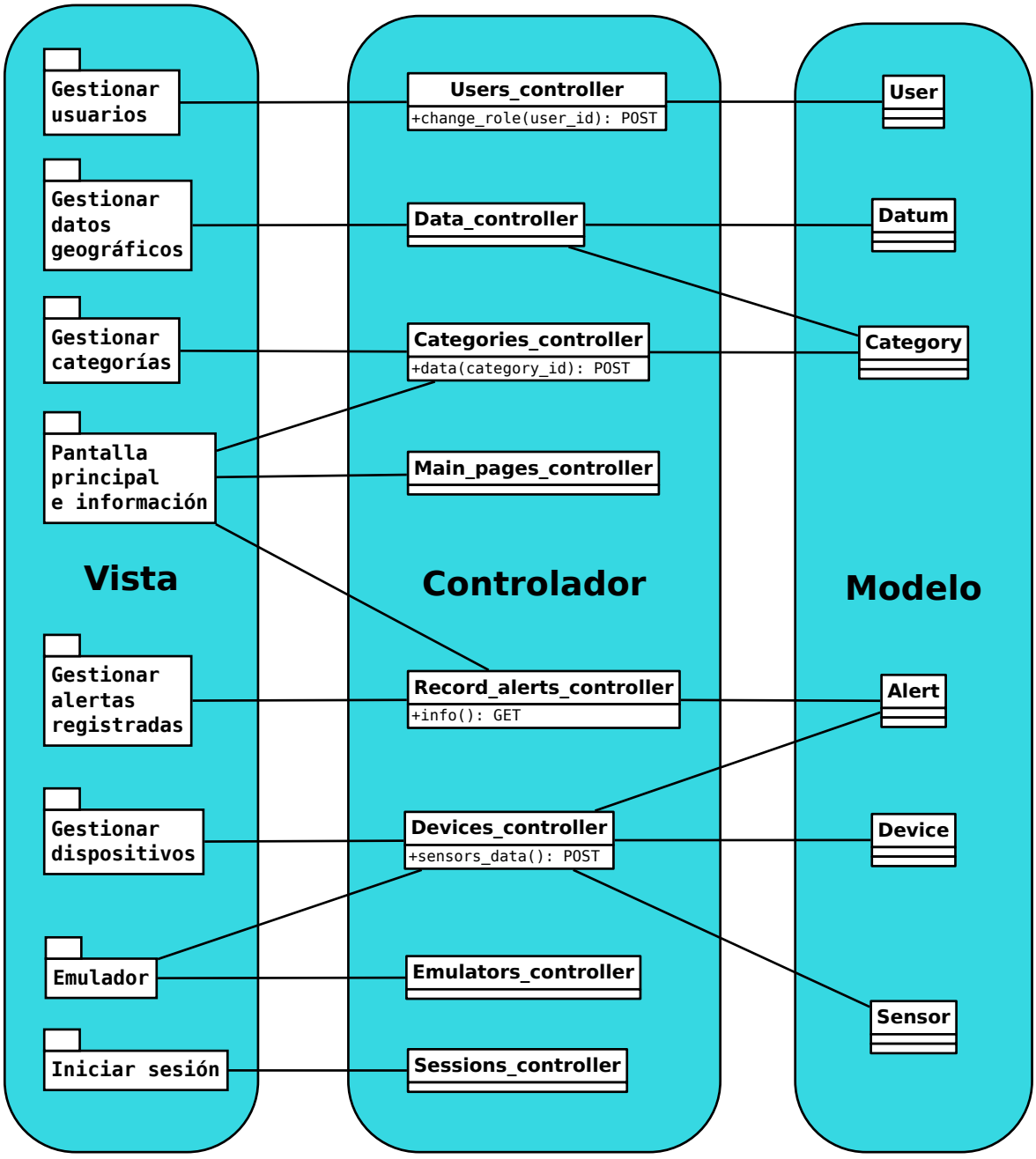

Figura 5.6: Diagrama de clases de SIGIEM

El diseño de la aplicación, ha quedado condicionado y favorecido por el uso de patrones. A continuación, se hará una breve introducción a estos y se mostrará cómo se han aplicado en la sección 5.5.

- **MVC:** para separar la arquitectura en tres capas y de esta forma, reduciendo el acoplamiento al mantener alejada la implementación de la interfaz de usuario, de la lógica de negocio y de los modelos.
- **Fachada:** se utiliza para reducir la complejidad del sistema, minimizando las comunicaciones y dependencias entre distintos módulos.
- **Factoría abstracta:** principalmente, se utilizará para el acceso a la base de datos, favoreciendo que la aplicación se independiente de su tipo (se utilizará un ORM para el acceso a la base de datos).
- **Delegación:** derivado de utilizar el patrón MVC, se utilizará para evitar acoplamientos, dando un mayor grado de abstracción entre las capas.

5.4.1. Acciones comunes

Es importante recalcar que la mayoría de las clases tienen el mismo número de métodos y se llaman de la misma forma. Por eso, se han excluido en el diagrama de la figura 5.6.

Esto se debe a la aplicación del principio REST [Fie00], que se ha explicado en el estado del arte. Así, de esta forma los controladores se orientan hacia las operaciones CRUD, fundiéndose con los recursos y de esta forma generan URLs más limpias. Un ejemplo de esto, aplicado al recurso usuario se puede ver en la tabla 5.4.1.

Verbo HTTP	Acción	URL REST	Utilización
GET	index	`/users`	Muestra todos los usuarios
GET	new	`/users/new`	Devuelve el HTML para crear un nuevo usuario
POST	create	`/users`	Crea un nuevo usuario
GET	show	`/users/:id`	Muestra un usuario específico
GET	edit	`/users/:id/edit`	Devuelve el HTML para editar un usuario
PUT	update	`/users/:id`	Actualiza un usuario concreto
DELETE	destroy	`/users/:id`	Elimina un usuario concreto

Tabla 5.1: Equivalencia en *Rails* de las acciones, rutas y peticiones al utilizar REST para el recurso usuario

5.4.2. Modelo

En esta capa se encuentran las clases que representan la información que el sistema procesa y almacena.

En *Rails* se pueden encontrar en la carpeta `app/models` y provee y permite el uso de métodos de validación, métodos que respondan a eventos durante el ciclo de vida de un objeto y métodos específicos de cada modelo para encapsular comportamientos.

A continuación se realiza una breve descripción de cada clase:

- **User**

  Representa a los usuarios que se almacenan en la base de datos. Contiene los atributos y requisitos que debe tener un objeto de este tipo para poder ser almacenado.

  Algunos de los elementos más importantes son:

  - El almacenamiento de la clave de usuario es cifrado, utilizando el módulo ActiveModel de *Rails* [act].
  - Se valida el email mediante una expresión regular.
  - Crea a cada usuario de un token de seguridad único para identificarle al iniciar sesión en el sistema.

- **Category**

  Representa a las categorías que se almacenan en la base de datos. Contiene los atributos y requisitos que debe tener un objeto de este tipo para poder ser almacenado.

  Algunos de los elementos más importantes son:

  - Una categoría puede tener asociado un dato (un objeto de tipo Datum).
  - Si se añade una categoría y la categoría a la que pertenece no existe, se crea antes esta categoría.
  - La longitud de los nombres de las categorías pueden tener como máximo 30 caracteres.

- **Datum**

  Representa a los datos geográficos que se almacenan en la base de datos. Contiene los atributos y requisitos que debe tener un objeto de este tipo para poder ser almacenado.

  Algunos de los elementos más importantes son:

  - Debe estar asociado obligatoriamente a una categoría (objeto de tipo Category).
  - El formato de los datos geográficos almacenados debe seguir el estándar *Geojson* [HB08].

- **Device**

  Representa a los dispositivos que se almacenan en la base de datos. Contiene los atributos y requisitos que debe tener un objeto de este tipo para poder ser almacenado.

  Algunos de los elementos más importantes son:

  - Debe tener obligatoriamente una coordenada geográfica válida (latitud y longitud).
  - Debe tener asociado al menos un sensor (un objeto de tipo Sensor).

- **Sensor**

  Representa a los sensores de los dispositivos que se almacenan en la base de datos. Contiene los atributos y requisitos que debe tener un objeto de este tipo para poder ser almacenado.

  Algunos de los elementos más importantes son:

  - Tiene un campo que puede identificar el tipo de sensor.
  - Debe pertenecer obligatoriamente a un dispositivo (objeto de tipo Device).
  - Puede tener una alerta asociada (un objeto de tipo Alert).
  - Almacena todos los valores que se mandan al sistema.

- **Alert**

  Representa a las alertas de cada sensor de un dispositivo que se almacenan en la base de datos. Contiene los atributos y requisitos que debe tener un objeto de este tipo para poder ser almacenado.

  Algunos de los elementos más importantes son:

  - Debe pertenecer a un sensor obligatoriamente (un objeto de tipo Sensor).
  - Debe tener al menos un límite superior o inferior válido.
  - Almacena todas las alertas que sobrepasen el intervalo definido por los límites.

5.4.3. Controlador

En esta capa se encuentran los archivos encargados de definir el comportamiento de la aplicación.

En un proyecto *Rails* se puede encontrar en la carpeta `app/controllers` y son los que controlan el flujo de las peticiones que realizan los usuarios de la aplicación, se encargan del filtrado y autenticación de estas peticiones, además también definen y crean el almacenamiento de las sesiones de cada usuario y también se encargan de controlar todas las posibles excepciones que se puedan dar al utilizar la aplicación.

Por defecto en *Rails* se utiliza la siguiente nomenclatura: `nombre_modelo_en_plural` seguido de `_controller`. Por ejemplo para el modelo `Day`, el controlador por defecto sería `Days_controller`. De esta forma, no hay que indicar ni configurar nada para las rutas de acceso a cada método del controlador, puesto que lo hace de acuerdo a REST.

Los ficheros que se encuentran en esta capa son:

- **Main_pages_controller**

  Se encarga del encaminamiento de las páginas principales de la aplicación: la página principal donde se muestra el mapa y el acerca de.

- **Sessions_controller**

  Clase que autoriza el acceso de los usuarios registrados.

- **Users_controller**

  Sólo se puede acceder a los métodos de esta clase si el usuario es un administrador. Es la clase encargada de controlar el flujo y realizar las acciones para listar los usuarios, eliminar un usuario, modificar los datos de un usuario, cambiar el rol de un usuario o crear uno nuevo.

  También se encarga de comprobar que el usuario que accede al método de modificación de la información almacenada de un usuario, es el mismo usuario rechazando cualquier otro intento.

- **Categories_controller**

  Sólo se puede acceder a los métodos de esta clase si el usuario es un administrador. Es la clase encargada de controlar el flujo y realizar las acciones para listar las categorías, eliminar una categoría, modificar una categoría o crear una nueva.

  También se encarga de atender las peticiones AJAX que se realizan desde la página principal (viendo el mapa) al seleccionar un elemento del menú. Comprueba que se solicita una categoría que existe y devuelve la información geográfica asociada, la cual será mostrada en el mapa de forma instantánea.

- **Data_controller**

  Sólo se puede acceder a los métodos de esta clase si el usuario es un administrador. Es la clase encargada de controlar el flujo y realizar las acciones para listar los datos geográficos almacenados, añadir nuevos datos o eliminar uno ya existente.

  Es la encargada de procesar la información geográfica que se desea añadir, comprobando que es un formato válido, descomprimir los documentos si el formato lo requiere y transformar los datos al sistema de coordenadas correcto y al formato correcto.

- **Devices_controller**

  Sólo se puede acceder a los métodos de esta clase si el usuario es un administrador. Es la clase encargada de controlar el flujo y realizar las acciones para listar los dispositivos que han enviado datos al sistema, añadir o modificar las alertas que tienen asociadas cada uno de los sensores de los que dispone o eliminar todos los datos almacenados de este dispositivo.

  También se encarga de atender las peticiones AJAX que se realizan desde el emulador o desde cada uno de los dispositivos (físicos) que se conecten al sistema. Los datos que se reciben se añadirán a los ya existentes del dispositivo o se creará uno nuevo si no existe. Está diseñado, para que las coordenadas geográficas de los dispositivos puedan variar y de esta forma, permitir que se muevan los dispositivos físicos.

- **Record_alerts_controller**

  Sólo se puede acceder a los métodos de esta clase si el usuario es un administrador. Es la clase encargada de controlar el flujo y realizar las acciones para listar las alertas registradas, borrar alguna alerta registrada, o todas las asociadas a un sensor.

  También se encarga de atender las peticiones AJAX que se realizan desde la página principal, para controlar en tiempo real si se ha producido alguna alerta en algún sensor o no.

- **Emulators_controller**

  Sólo se puede acceder a los métodos de esta clase si el usuario es un administrador. Es la clase encargada de controlar el flujo y crear la información aleatoria del emulador, que posteriormente se podrá enviar.

  La información que genera está acotada, para que el dispositivo se encuentre en las coordenadas geográficas de Castilla-La Mancha, además de poder generar datos que hagan saltar las alertas del sensor que envía la información.

Vista

En esta capa es la encargada de mostrar los datos de la capa de modelo y la forma de interactuar con estos, en general se encuentran los elementos de generar la interfaz de usuario.

En un proyecto *Rails* puede encontrarse en la carpeta `app/views` y `app/assets` y utiliza por defecto un sistema de plantillas bastante completo. Esto hace que una vez definida la plantilla principal de la aplicación, sólo haya que escribir las partes que cambian de esta. Además gracias al uso de *partials*, que son fragmentos de código en archivos separados escritos en *Ruby* y en HTML, favorecen la buena práctica de no repetir código. Además, el sistema de plantillas también permite la utilización de plantillas anidadas.

Los ficheros que se encuentran en este apartado son:

- **Archivos HTML**

  Son los archivos elaborados con el lenguaje HyperText Markup Language (HTML), utilizados para describir y definir la estructura del documento que se va a presentar.

  Archivos html.erb

  Son los archivos que *Rails* utiliza para generar código HTML dinámicamente. Por lo tanto, en estos archivos también se puede encontrar código escrito en *Ruby*.

- **Archivos CSS**

  Son los encargados de dar el estilo y diseño necesario a la interfaz de usuario. Se utiliza el lenguaje Cascading Style Sheets (CSS) para definir como será la presentación de cada documento, permitiendo separar la estructura del documento de su presentación.

- **Archivos Javascript**

  Son archivos de código fuente escritos en *Javascript*, un dialecto del estándar *ECMAScript* [ecm], que se utilizan principalmente del lado del cliente y hacen que se mejore la interfaz de usuario y la interacción con esta. También se utiliza para realizar peticiones al servidor de forma asíncrona mediante Asynchronous Javascript And XML (AJAX).

  Entre este tipo de archivos, se encuentra la biblioteca utilizada para visualizar el mapa (*Leaflet* [lea]) y el framework para facilitar el desarrollo con *Javascript* (*jQuery* [jqu]).

  Archivos js.erb

  Son los archivos que *Rails* utiliza para generar código *Javascript* dinámicamente. Por lo tanto, en estos archivos también se puede encontrar código escrito en *Ruby*. Se utilizan en las peticiones *Ajax* que se hacen al servidor.

5.5. Implementación

Como ya se ha explicado, el desarrollo del proyecto está dividido en varias iteraciones. En cada una de estas iteraciones se han ido añadiendo nuevas funcionalidades hasta completar la primera versión de SIGIEM. También entre algunas iteraciones se han ido implementando mejoras y refactorizando código de funcionalidades ya implementadas.

En cada iteración, se elegía la funcionalidad a implementar y se inicializaban las pruebas de integración a realizar. Si se desarrollaba un controlador además, para la realización de las pruebas se utilizaba una biblioteca (*Capybara*) para simular el comportamiento de un usuario utilizando la aplicación.

Por norma general (como se ha explicado en la sección 4.1.1), antes de comenzar a desarrollar alguna funcionalidad de un modelo o controlador en cada iteración, se creaba una prueba. Una vez que esta prueba fallaba (al utilizar integración continua se ejecutaban automáticamente) se procedía a codificar esa funcionalidad. Este ciclo se repetía hasta que se completase la funcionalidad del modelo o controlador que se estaba desarrollando en esa iteración.

Aunque siempre se han intentado realizar las pruebas antes que el código, en algunas situaciones no se ha podido aplicar esta técnica, normalmente debido a la falta de conocimiento de alguna tecnología, poco uso del proceso de desarrollo o a la falta de conocimiento de las respuestas que debía tener el sistema ante una situación. En estos casos, las pruebas se realizaron posteriormente.

Gracias al lenguaje *Ruby* y a los frameworks utilizados como *Rails*, *Rspec* el código resultante es sencillo y claro, además de ayudar a utilizar buenas prácticas de desarrollo. De hecho muchos de los patrones de diseño que hay que implementar en otros lenguajes, son soportados de forma nativa por *Ruby* [FM08].

La jerarquía de directorios para la implementación de SIGIEM que se ha seguido es la que propone *Rails*.

Algunas de las buenas prácticas utilizadas son:

- **Código limpio y simple:** se ha tratado de hacer que el código sea simple, fácil de entender, que exprese la idea y la intención para la que se ha creado. Un ejemplo se encuentra en el listado 5.1.

```
1  describe "add category with valid information" do
2
3    before do
4      fill_in "Nombre", with: new_name
5      fill_in "Categoria a la que pertenece", with:
           new_parent
6    end
7
8    it "should create a new category" do
9      number_of_categories=Category.count
10     click_button :add_category
11     Category.count.should == number_of_categories+1
12   end
```

Listado 5.1: Extracto de la clase de pruebas `categories_controller_spec`

- **Refactorización de código:** siempre que se ha podido se ha tratado de refactorizar el código. Aunque debido al tiempo acotado para el desarrollo del proyecto, seguramente se pueda realizar una refactorización más profunda.
- **Pruebas automáticas:** el software desarrollado se ha probado automáticamente (ver sección 5.7.2) mediante pruebas de integración. Además se ha utilizado integración continua durante todo el desarrollo del proyecto, lo que ha hecho posible que al añadir funcionalidad nueva no se modificara el comportamiento de funcionalidades creadas anteriormente.
- **Uso de estándares abiertos:** gracias al uso de estándares web (*HTML,CSS*), la aplicación está soportada por la mayoría de los navegadores web que existen, además de facilitar el desarrollo de algunas funcionalidades.

  También el uso de estándares de codificación de los SIG permite la integración de distintas fuentes de datos y el uso futuro de los datos recogidos por esta aplicación.
- **Convención antes que configuración**: es un principio que utiliza *Rails* por defecto, lo que ha permitido no tener que configurar nada para realizar la aplicación web, aparte de la utilización de *MongoDB* (lo cual no es una configuración convencional de *Rails*).

- **Técnica REST:** [Fie00] se ha seguido debido a que se utiliza por defecto en el encaminamiento de *Rails*. Se ha intentado seguir con todos los recursos, pero ha habido en algunos que no se ha aplicado completamente. Un ejemplo de su uso, se puede ver en el listado 5.2.

```
1      users GET    /users(.:format)          users#index
2            POST   /users(.:format)          users#create
3  new_user GET    /users/new(.:format)      users#new
4 edit_user GET    /users/:id/edit(.:format) users#edit
5       user PUT    /users/:id(.:format)      users#update
6            DELETE /users/:id(.:format)      users#destroy
```

Listado 5.2: Rutas del recurso usuario siguiendo REST

- **Patrones de diseño:** como ya se ha comentado anteriormente, muchos de los patrones son soportados de forma nativa por *Ruby*, a continuación se citan brevemente algunos de los utilizados:
  - **Modelo Vista Controlador:** gracias a *Rails* se propone este patrón por defecto como arquitectura a seguir para desarrollar la aplicación web (ver figura 5.7).

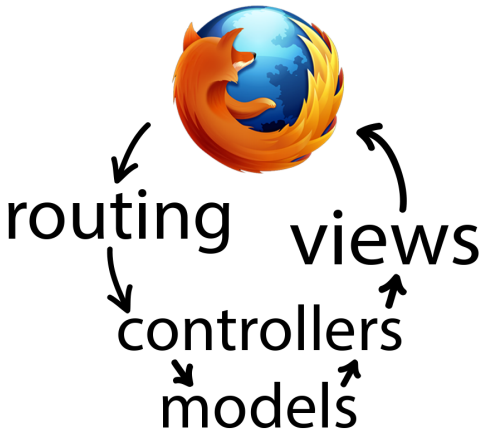

Figura 5.7: Arquitectura MVC que utiliza *Rails*. El logo de Mozilla Firefox es propiedad de Mozilla Foundation [moz]

  - **Iterator:** se ha utilizado cada vez que se ha manejado un objeto de tipo `Array`, o con los métodos que se han pasado a elementos de tipo *Integer* como `times` o `upTo`. En general con cualquier objeto de tipo *Enumerate*.
  - **Singleton:** se ha usado cada vez que se ha trabajado con los valores `true`, `false` o `nil`, puesto que en *Ruby* son instancias singleton de sus respectivas clases `TrueClass`, `FalseClass` y `NilClass`.

- **Active Record:** [Fow03] es un patrón arquitectural, que de nuevo, al usar *Rails* se utiliza por defecto. Este proporciona una abstracción en el acceso a la base de datos mediante el uso objetos del lenguaje de programación.

  En el listado 5.3 se puede ver cómo en la línea 4 se llama al método `destroy` del objeto usuario y este automáticamente genera una operación de borrado en la base de datos.

- **Delegación de funcionalidad:** se utiliza en la comunicación entre las clases controladoras y las clases de los modelos. Por ejemplo al llamar al método `destroy` del controlador de usuarios, este a su vez llama al método `destroy` del modelo usuario (ver listado 5.3).

```
1 class UsersController < ApplicationController
2     def destroy
3         user=User.find_by_id(params[:id])
4         user.destroy
5         flash[:success]="Usuario eliminado
              correctamente."
6         redirect_to users_path
7     end
```

Listado 5.3: Método `destroy` de la clase `Users_controller`

- **Factoría abstracta:** se utiliza por defecto en *Rails* en el acceso a la base de datos, puestro que gracias a su módulo *Active Record* el acceso a la información almacenada es independiente de los distintos tipos de bases de datos que se puede utilizar.

  También se ha utilizado este patrón al utilizar la biblioteca *FactoryGirl*, utilizada fundamentalmente en las pruebas, se puede ver un ejemplo en el listado 5.4. Este código crea un objeto usuario independientemente de la base de datos que se esté utilizando.

```
1 FactoryGirl.define do
2   factory :user do
3
4     sequence(:name){ |n| "User Test #{n}"}
5     sequence(:email){ |n| "user_#{n}@user.com"}
6     password "daviddavid"
7     password_confirmation "daviddavid"
```

Listado 5.4: Extracto del fichero `factories.rb` en el que se define a un usuario

5.5.1. Pruebas

Un apartado muy importante de la implementación han sido las pruebas automáticas que se han realizado, en la cuales se han seguido las mismas buenas prácticas que para la aplicación.

Se han implementado un total de 325 pruebas (ver figura 5.8), las cuales ocupan 1640 líneas de código (LOC) y dan una proporción con respecto al código de la aplicación de $1:2$. Es decir, por cada línea de código de la aplicación se han escrito dos de pruebas.

```
17:12:11 - INFO - Running all specs
Running tests with args ["--drb", "-f", "progress", "-r", "/var/l
ib/gems/1.9.1/gems/guard-rspec-1.2.1/lib/guard/rspec/formatters/n
otification_rspec.rb", "-f", "Guard::RSpec::Formatter::Notificati
onRSpec", "--out", "/dev/null", "--failure-exit-code", "2", "spec
"]...
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................

Finished in 54.55 seconds
325 examples, 0 failures


Randomized with seed 35671

Done.

17:13:07 - INFO - Starting Spork for RSpec
```

Figura 5.8: Ejecución de todas las pruebas implementadas

Las pruebas realizadas cubren el $91{,}11\,\%$ del código de la aplicación, lo cual asegura un cierto grado de calidad de la aplicación. Se puede ver en detalle la explicación de la cobertura para cada clase de prueba abriendo el fichero situado en `coverage/index.html` y un resumen de esta documentación en el Anexo D.

Se ha realizado un fichero de prueba por cada controlador y para cada modelo. Se pueden encontrar en los directorios `spec/request` y `spec/models` respectivamente, los ficheros de prueba se llaman igual que el controlador o el modelo seguido de `_spec`. Además también se ha creado un fichero `spec/factories.rb` en el cual se definen los objetos a utilizar en las pruebas, de manera que no se tenga que repetir código y sea independiente de los distintos tipos de base de datos.

También se ha creado el directorio `spec/data_input`, donde se almacenan datos geográficos válidos y no válidos y en diferentes formatos para su utilización en las pruebas automáticas.

Las pruebas que se realizan a los controladores, utilizan la biblioteca *Capybara* para simular el comportamiento de los usuarios al interactuar con la aplicación. De esta manera se puede describir el comportamiento de un usuario indicando los enlaces y botones en los que se pulsa, los valores de los formularios que rellena, los elementos que se seleccionan o los ficheros que se desean enviar al sistema. Se puede ver un ejemplo de este tipo de pruebas en el listado 5.5.

```
1  #coding: utf-8
2  require 'spec_helper'
3
4  describe "User Controller," do
5
6    subject{ page}
7
8    describe "New user :" do
9
10     before{ visit registrate_path}
11
12     it{ should have_selector 'title', text: "Registro"}
13     it{ should have_selector 'h1', text: "Registrate"}
14
15     describe "should show a form" do
16
17       it{ should have_selector 'form'}
18       it{ should have_selector 'label', text: "Nombre"}
19       it{ should have_selector 'label', text: "Email"}
20       #...
21       it{ should have_selector 'input', type: 'submit', value:
             :registrar}
22     end
23
24     describe "with valid information," do
25
26       before do
27         fill_in "Nombre", with: "David"
28         fill_in "Email", with: "david@dpzaba.com"
29         fill_in "Contrasena", with: "dpzaba"
30         fill_in "Repita la contrasena", with: "dpzaba"
31       end
32
33       it "user created (in BBDD)" do
```

```
34          number_of_users_before=User.count
35          click_button :registrar
36          number_of_users_aftr=User.count

38          number_of_users_aftr.should eq number_of_users_before+1
39        end

41        it "should show a message" do
42          click_button :registrar
43          should have_selector('div.alert.alert-success',
44                               text: "Bienvenido")
45        end
```

Listado 5.5: Extracto de las pruebas del controlador de usuarios

Sin embargo, en las pruebas realizadas a los modelos, no se ha utilizado este tipo de bibliotecas. En este caso, se comprueba que el objeto contiene los campos que se desea y que cumple todas las validaciones y requisitos que se definen en la clase del modelo. Se puede ver un ejemplo de este tipo de pruebas en el listado 5.6.

```
1  require 'spec_helper'

3  describe Category do

5    before do
6      @category=FactoryGirl.create(:category)
7    end

9    subject{ @category}

11   it{ should be_valid}

13   describe "should respond to" do
14     it { should respond_to :name}
15     it { should respond_to :parent}
16   end

18   describe "name:" do

20     it "is too long" do
21       @category.name="a"*31
```

```
23        should_not be_valid
24      end

26      it "is duplicate in DB (isn't case sensitive)" do
27        category_duplicate=@category.dup
28        category_duplicate.name.upcase!

30        category_duplicate.should_not be_valid
31      end

33      it "is duplicate in DB" do
34        category_duplicate=@category.dup

36        category_duplicate.should_not be_valid
37      end
38    end #name
```

Listado 5.6: Extracto de las pruebas al modelo de las categorías

5.5.2. Licencia

El código desarrollado, se ha liberado utilizando la licencia *GNU General Public License v3.0* [gpl].

5.6. Test de seguridad

Debido a que SIGIEM se encontrará expuesto a Internet es muy importante tener en cuenta la seguridad de esta aplicación. Debido a que no se dispone de suficiente tiempo como para realizar un análisis exhaustivo se ha realizado una pequeña batería de pruebas con la herramienta *Vega*.

Se debe recalcar que gracias a *Rails* [raic] se han podido evitar muchas de las principales fuentes de fallos de seguridad como: Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), etc.

También, debido a que se utiliza un Sistema de Gestión de Base de Datos **NOSQL!**, el cual todavía no es tan común como los basados en SQL y que se realizan las consultas a través de un ORM (*MongoMapper*) se evitan muchos ataques dirigidos hacia la base de datos.

Todas las pruebas realizadas se han ejecutado contra el servidor local (*Thin* [thi]) y la cantidad total de test realizados supera los 4000 (ver figura 5.9).

Estos test se han realizado dos veces, una sin introducir ningún parámetro adicional y otra añadiendo una *cookie* de administrador de SIGIEM. Para que el texto no se haga demasiado

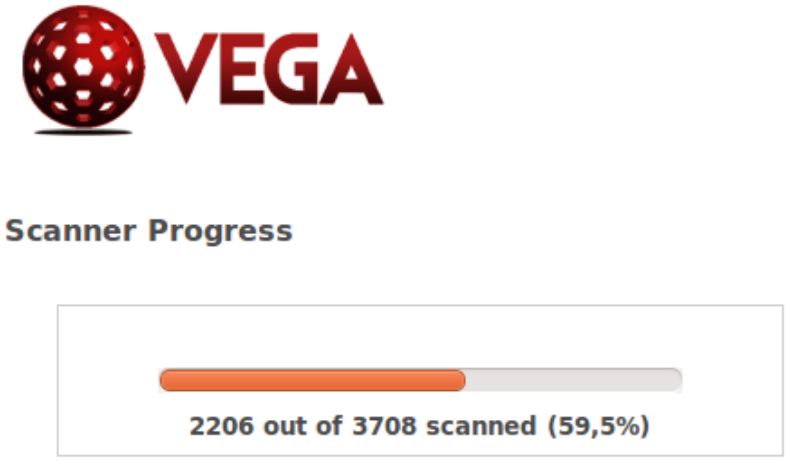

Figura 5.9: Progreso del escáner Vega

extenso, se pueden observar los resultados de los test en el Anexo C. Sólo añadir que se han encontrado los mismos tipos fallos (ver C.3) en los dos casos, sin que en ningún caso sean de gran importancia.

5.7. Estadísticas

En esta sección se muestran algunas estadísticas, resultado del desarrollo del proyecto. De manera que se pueda estimar un esfuerzo del trabajo realizado.

5.7.1. Código de aplicación

Al comienzo del desarrollo del proyecto se dio un importante incremento del esfuerzo, ya que se tuvo que estudiar todas las tecnologías a utilizar, estándares web y estándares de codificación y acceso a información de los Sistema de Información Geográfica y eso ha supuesto alrededor de un mes de aprendizaje. Aún así, cada día que se trabaja con estas tecnologías y herramientas se aprenden nuevas facetas y nuevas maneras de trabajar.

A continuación se muestra una tabla con el número de líneas de código del proyecto divididas entre los distintos componentes:

Componente	Líneas	LOC	Clases	Métodos	Métodos/Clases	LOC/Método
Controladores	651	466	10	47	4	7
Modelos	251	165	7	12	1	11
Helpers	109	81	0	15	0	3
Total	**1011**	**849**	**17**	**74**	**4**	**29**

Tabla 5.2: Estadísticas de lineas de código

En la vista se puede hacer una aproximación al número de líneas de código, puesto que en este apartado se genera tanto código *HTML*, como *Javascript* dinámicamente. Una aproximación es la de la tabla 5.3.

Componente	Número archivos	LOC
CSS	6	205
HTML	30	500
Javascript	8	165
Total	**44**	**870**

Tabla 5.3: Aproximación a las lineas de código de la vista

5.7.2. Código de pruebas

También se incluye otra tabla con información sobre las líneas de código de las pruebas realizadas, siguiendo también la división en módulos.

Componente probado	Líneas	LOC
Modelo	605	427
Comportamiento	1726	1213
Total	**2331**	**1640**

Tabla 5.4: Estadísticas de lineas de código de las pruebas

Viendo estas dos tablas (tabla 5.2 y 5.4), se observa que se obtiene una proporción de test de 1:2, es decir, por cada línea de código de la aplicación se han escrito dos de pruebas.

Estas estadísticas, han sido generadas automáticamente gracias a la utilidad *Rake* [rak] que utiliza por defecto *Rails*. Pero por defecto *Rake* no muestra el número de líneas de código de la capa de la vista, ni código *Javascript*, ni *CSS* y para ello se ha utilizado otra herramienta: *CLOC*.

5.7.3. Cobertura de las pruebas

Puede observarse en las figuras del Anexo D la cobertura de las pruebas realizadas, dando una cobertura media total superior al 90 % (ver figura 5.10). A modo de resumen se muestra la tabla 5.5.

Figura 5.10: Cobertura media total del proyecto

Módulo	Cobertura
Modelos	100 %
Controladores	86.67 %
Helpers	97.73 %
Total	**91.11 %**

Tabla 5.5: Resumen de las estadísticas de cobertura de código

La documentación sobre la cobertura de las pruebas ha sido generada por la biblioteca (*SimpleCov* [sim]). Puede encontrarse toda la documentación generada en la carpeta `coverage` en el repositorio de SIGIEM y si se desea volver a generar toda esta documentación puede hacerse ejecutando todas las pruebas de nuevo.

5.8. Estimación de costes

Se ha trabajado durante más de siete meses en este proyecto, pero sólo durante seis a tiempo completo (en total alrrededor de 980 horas) y todos los recursos necesarios para la construcción de SIGIEM han sido descritos en el apartado 4.2.

Teniendo en cuenta sólo los siete meses que se ha estado contratado y considerando como coste de infraestructura sólo los dos equipos que se han utilizado en el desarrollo (ya que el servidor, es compartido) y que el sueldo como contratado con la categoría CUARTA-O Proyecto de I+D, tipo CUARTA-O-II es de 990.76 €al mes, se puede hacer una estimación del coste total (ver tabla 5.6).

Recurso	Cantidad	Coste
Dell Vostro 1000	1	450 €
Monitor HP ZR24w	1	350 €
Workstation HP Z600	1	3000 €
Horas trabajadas	980	6935 €
Total		**10735 €**

Tabla 5.6: Estimación del coste del proyecto

Capítulo 6

Conclusiones y propuestas

En este capítulo se realiza un valoración de los objetivos que se han alcanzado. También se realiza una serie de propuestas de trabajo futuro para mejorar algunos aspectos de SIGIEM como son: la funcionalidad, la seguridad y el soporte. Por último, se hace una valoración personal del autor sobre la experiencia durante el desarrollo de este Proyecto Fin de Carrera.

6.1. Objetivos alcanzados

Tras haber realizado el análisis del capítulo de Resultados, se puede observar y confirmar que se han cumplido satisfactoriamente los objetivos propuestos, tanto el objetivo general como los objetivos específicos de este proyecto (ver capítulo 2).

El proyecto SIGIEM es capaz de mostrar información geográfica de Castilla-La Mancha sobre la temática de medio ambiente y energía. Además, para los administradores de la aplicación también es capaz de mostrar los sensores que han enviado datos y mostrar si estos han generado o generan alguna alerta.

Al comienzo del proyecto se tuvo que realizar un largo periodo de aprendizaje tanto del lenguaje de programación *Ruby* como del framework de desarrollo web *Rails*. Se ha de mencionar que gracias a este aprendizaje previo y a la elección de estas herramientas, se ha podido realizar el trabajo de forma exitosa y atractiva.

Gracias al uso de un sistema de gestión de base de datos **NOSQL!** se pudo tener un esquema flexible de datos, lo que ha permitido que cambios producidos en los requisitos no hayan supuesto mucho esfuerzo al modificarlo. También ha hecho posible que se diseñe un sistema capaz de almacenar grandes conjuntos de datos y tener un acceso eficiente a los mismos.

El estudio de los estándades y herramientas que se utilizan con los SIG permitió la elección de un estándar de codificación de datos (*Geojson* [HB08]) muy cercano al modelo de datos utilizado para su almacenamiento y para su tratamiento a la hora de ser visualizados.

La comunicación con los sensores mediante la creación de una API para unos servicios web, hace que sea totalmente independiente tanto de los dispositivos que se utilicen para enviar datos, como del software que se utilice para generar las peticiones.

Debido al hecho del envío de datos por parte de los sensores, otro objetivo específico era su control y registro, el cual también se ha llevado a cabo satisfactoriamente de forma que los administradores de la aplicación puedan observar las irregularidades de estos datos en tiempo real, gracias al uso de AJAX.

Para comprobar que en SIGIEM la comunicación con los sensores y el registro de alertas funcionaban correctamente, se creó un emulador para simular el envío de datos desde los mismos. Este fue uno de los requisitos que no estaba contemplado inicialmente, pero gracias al cual se pudo comprobar el funcionamiento del sistema.

Otro objetivo que se deseaba cumplir era desarrollar una aplicación que tuviese un cierto grado de calidad. Esto se ha conseguido gracias al uso de los últimos estándades web que han permitido realizar un diseño atractivo, usable y adaptable a distintos dispositivos. Y el gran número de pruebas de aceptación desarrolladas, han permitido avanzar con seguridad durante la etapa de desarrollo, asegurando que el comportamiento ya implementado no dejaba de funcionar debido a las nuevas funcionalidades que se iban incorporando.

Por último, se deseaba que el sistema estuviese accesible a todo el mundo a través de Internet por lo que se realizó un despliegue en el CPD de la Escuela Superior de Informática de Ciudad Real utilizando Apache como servidor web y un script en *Ruby* para automatizar la tarea de la puesta en marcha del sistema. Se puede acceder a este mediante la dirección `sigiem.uclm.es`.

A continuación, se muestra el estado de los objetivos planteados inicialmente, tanto los funcionales (ver tabla 6.1), como los académicos (ver tabla 6.2).

Objetivo	**Estado**
Mostrar información geográfica de Castilla-La Mancha	√
Acceso eficiente a los datos almacenados	√
Comunicación con los sensores	√
Control y registro de alertas de los datos que envían los sensores	√
Emulador de sensores	√
Aplicación usable y atractiva	√
Despliegue de la aplicación	√

Tabla 6.1: Resumen de los objetivos funcionales alcanzados

Objetivo	Estado
Aprendizaje de *Ruby on Rails*	√
Aprendizaje de *Javascript*	√
Aprendizaje de bases de datos *NoSQL*	√
Aprendizaje de los fundamentos de los SIG	√
Estudio de estándares y herramientas de los SIG	√

Tabla 6.2: Resumen de los objetivos académicos alcanzados

6.2. Propuestas de trabajo futuro

Aunque se han cubierto los objetivos principales de este proyecto, se han detectado una serie de mejoras que se consideran interesantes para el futuro de la herramienta. Además, como en estos momentos se encuentra en producción hay una serie de comprobaciones y mantenimiento que deberían realizarse.

Algunas de estas posibles mejoras son:

- **Mejora del diseño**

  Aunque se ha alcanzado un diseño consistente y aceptable, debido a la limitación de tiempo no se han podido realizar todos los detalles que se tenían en mente.

  Además, también habría que probarlo en muchos dispositivos y navegadores debido a las diferentes características existentes entre ellos para ver que funciona y se visualiza correctamente. Otro posible detalle sería la utilización de iconos propios para mostrar los sensores o puntos en el mapa.

- **Ampliación del catálogo de datos**

  Debido a las limitaciones que se han mencionado en la sección 2.2 y a la dificultad de encontrar otras fuentes fiables, no se dispone de muchos datos para utilizar.

  Otra forma de mostrar datos sería a través del estándar Web Map Service (WMS), aunque estos servicios deberían ser sólo de Castilla-La Mancha y de igual forma que los datos, no son fáciles de encontrar. Esta posible mejora, está parcialmente desarrollada pero no se incluye en la versión de producción.

  También podría plantearse la posibilidad de utilizar datos que sean *Open Data* y estén estructurados de alguna forma que sea comprensible por una máquina. En España, existe una iniciativa oficial por parte del gobierno [ope], sobre la que habría que investigar y ver si es posible utilizarla.

- **Monitorización**

  Una vez puesta en producción la herramienta, habría que monitorizar tanto la aplicación como el servidor. Controlar los accesos al servidor, recursos que se utilizan, peticiones que atiende, tiempos de respuesta, etc.

  Otro ejemplo sería controlar el número de accesos a la base de datos y observar si la cache que posee *Rails* es suficiente para proporcionar un buen rendimiento o es necesario utilizar otra como *Memcached* [mem].

- **Test de rendimiento**

  Derivado del punto anterior sobre monitorización surge la idea de realizar pruebas de rendimiento de la aplicación.

  Además, puesto que *Rails* dispone de un módulo [raia] para ejecutarlas fácilmente, se podría detectar las partes que utilizan más tiempo, más memoria, etc. Para ello habría que estudiar los diferentes benchmarks y métricas que se utilizan para medir el rendimiento web.

- **Mejorar la gestión de alertas**

  En este momento la aplicación permite gestionar las alertas individualmente de cada sensor. Se podría añadir la opción de gestionar todas las alertas de los sensores del mismo tipo. De tal forma que, se podrían gestionar a la vez todas las alertas de los sensores que miden la temperatura.

  Esta opción se consideró desde el principio (está contemplado en el modelo de datos), pero como no era uno de los objetivos principales no se ha desarrollado por completo.

- **Mejora de la API**

  Otra mejora importante que se podría realizar es la incorporación de un token de seguridad a la hora de establecer una comunicación entre los sensores y SIGIEM a través de la API web. De esta forma, podrían validarse los dispositivos que envían información y tener estadísticas de estos.

- **Refactorización de código y mejora de las pruebas**

  Aunque siempre que se ha podido, se ha tratado de refactorizar el código de manera que sea más legible y pueda mantenerse mejor, debido al tiempo acotado para el desarrollo del proyecto hay muchas partes que pueden ser mejoradas.

  De igual forma, con el avance del proyecto, se han ido obteniendo conocimientos que al principio no se tenían, por lo que tanto el código como las pruebas escritas al inicio, pueden mejorarse considerablemente.

6.3. Conclusión personal

Realizar el Proyecto Fin de Carrera es la etapa más importante de los estudios de Ingeniería en Informática puesto que permite demostrar de forma práctica todos los conocimientos adquiridos durante estos años de estudio y trabajo.

Personalmente, tuve la suerte de poder realizar este proyecto contratado en un proyecto de investigación en el grupo Alarcos, haciendo que además de ser el proyecto de mayor envergadura que he desarrollado, se mezclase con mi primera experiencia laboral. Uno de los aspectos que más me llamó la atención fue el poder desarrollar este proyecto desde el principio hasta tener una primera versión pública que se pueda utilizar.

Empezar este proyecto supuso para mí varios retos. Tuve que adquirir conocimientos de Sistemas de Información Geográfica: proyecciones y sistemas de coordenadas, modelo raster y vectorial, estándares comunes y cómo se utilizan estos sistemas en la web. Además tenía unos conocimientos muy básicos de desarrollo web por lo que tuve que dedicar inicialmente mucho tiempo a su aprendizaje, aunque cada día que continúo desarrollando aprendo cosas nuevas y trato de mejorar las realizadas anteriormente.

La decisión de utilizar el lenguaje de programación *Ruby* fue una apuesta arriesgada de la que no me arrepiento ya que he descubierto un gran lenguaje que no había probado anteriormente, muy potente y que me encanta. También supuso para mí un gran esfuerzo y adaptación en el desarrollo, el intentar utilizar un proceso de desarrollo basado en pruebas (aunque haya habido casos en los que no he sabido aplicarla correctamente, puesto que todavía me queda mucho aprendizaje), pero gracias a esto creo que he conseguido un software de mayor calidad y que cumple con las funcionalidades deseadas.

Otro aspecto del que estoy orgulloso, es que todo el proyecto se ha desarrollado con software libre o de código abierto, sin ninguna herramienta privativa y que será compartido bajo los mismos principios.

En general, creo que la experiencia que he adquirido me vendrá muy bien en mi futuro laboral (puesto que ahora mismo la mayoría de sistemas que se realizan son web o tienen alguna interacción con elementos web) y considerando que tenía un tiempo acotado para trabajar en el proyecto (el contrato fue de siete meses y hay que descontar el mes de verano y las vacaciones de Navidad) estoy contento con el trabajo realizado y espero que el sistema desarrollado sea utilizado y de ayuda para muchas personas.

ANEXOS

Anexo A

Manual de usuario

A.1. Vista principal

Al acceder a SIGIEM a través de `sigiem.uclm.es` se muestra la pantalla principal de la aplicación (ver figura A.1). En esta se muestra, un mapa como elemento central, un menú superior con las secciones accesibles de la aplicación (registro, acerca de, inicio de sesión) y otro menú situado a la izquierda del mapa con todas las categorías que se pueden consultar.

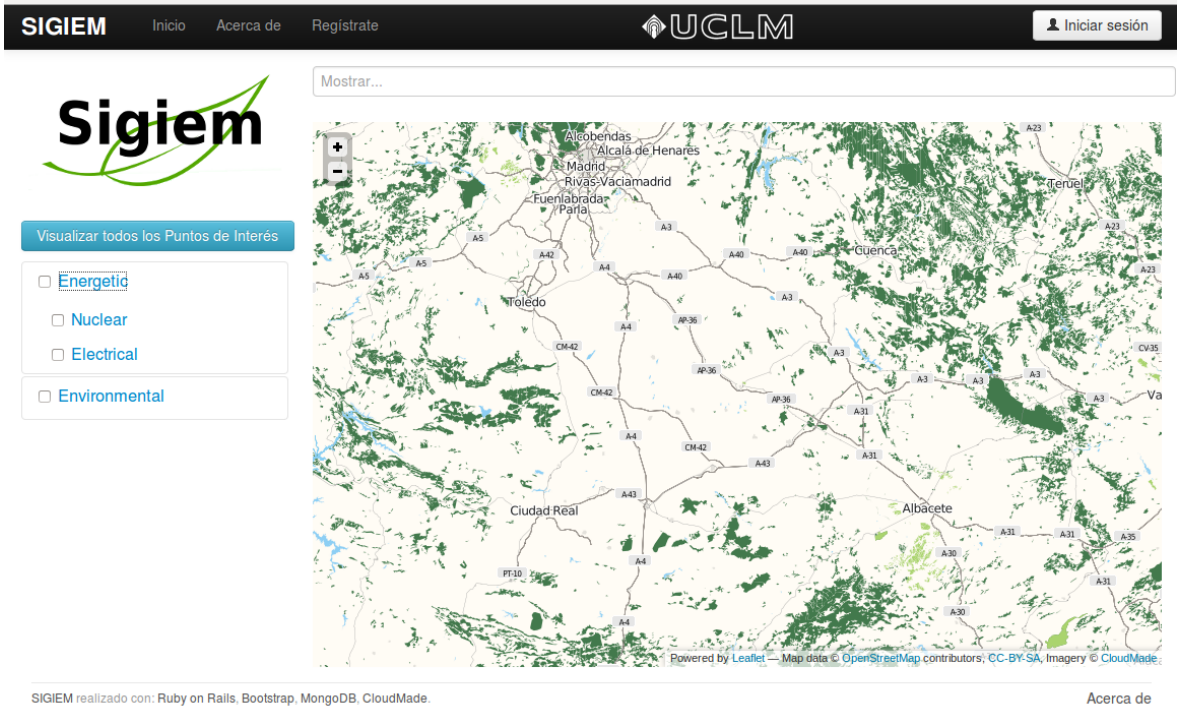

Figura A.1: Vista principal de la aplicación

Si se selecciona alguna de las categorías del mapa que están a la izquierda, podrán verse en el mapa dibujados los resultados. También puede desplazarse por el mapa y hacer zoom para visualizar mejor los elementos mostrados y seleccionarlos para obtener más información. Puede encontrarse un ejemplo en las figuras A.2 y A.3.

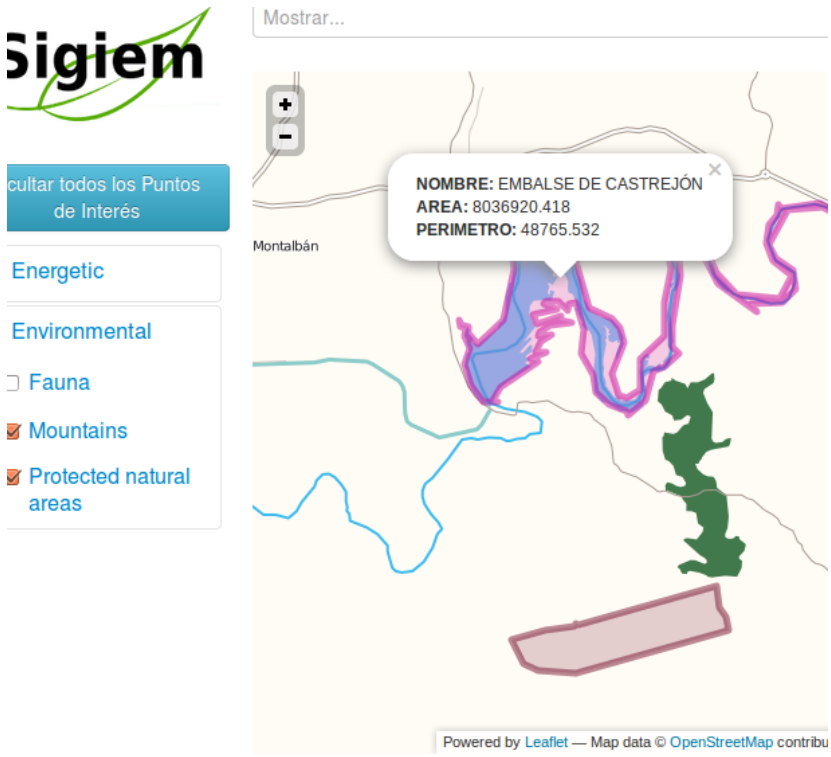

Figura A.2: Vista principal de la aplicación mostrando datos de embalses de Toledo

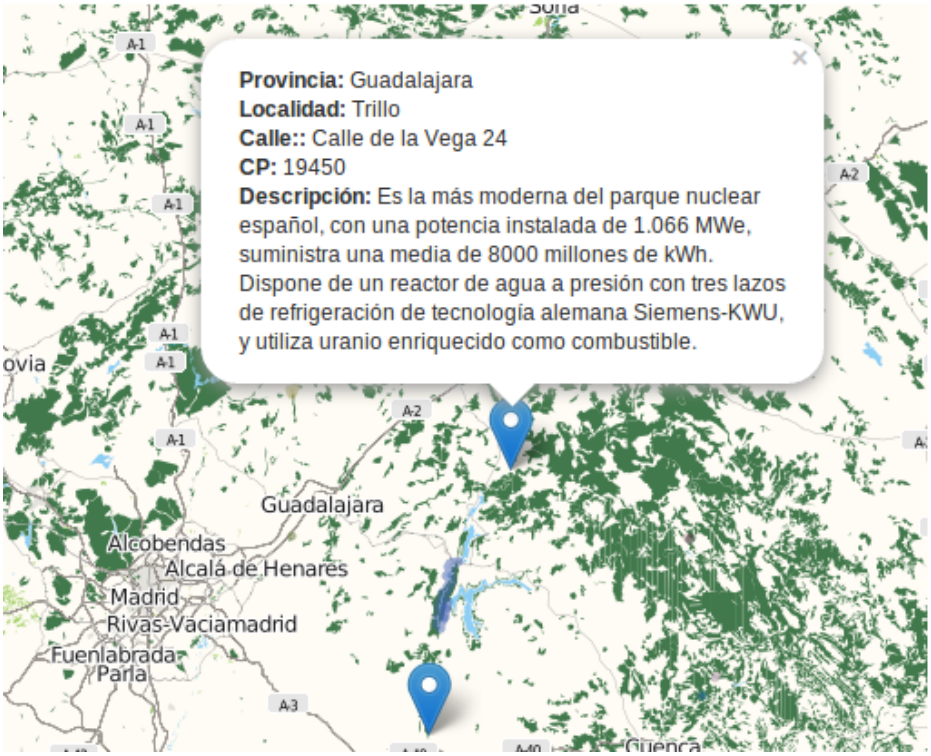

Figura A.3: Vista principal de la aplicación mostrando datos de la central nuclear de Trillo

A.2. Registro

Para darse de alta en el sistema el usuario debe estar registrado. Para llevar a cabo esta acción debe situarse en la pantalla *Regístrate*. Puede encontrar un ejemplo del aspecto de esta pantalla en la Figura A.4.

Figura A.4: Visualización de la pantalla de registro

Tras el registro sus datos quedarán almacenados en el sistema y podrá acceder con posterioridad al mismo, valiéndose del email y contraseña elegidos.

Además, si ha llegado a esta pantalla y usted ya está registrado, observe que en la parte inferior de la misma se ofrece la opción de *Iniciar sesión*. Si desea más información acerca de ello, por favor lea la sección A.3.

A.3. Iniciar Sesión

Una vez haya realizado el registro en el sistema podrá acceder al mismo utilizando su email y contraseña. Tras rellenar los campos pertinentes con estos datos debe pulsar el botón *Entrar*. La pantalla que le proporciona estas acciones es la de *Iniciar Sesión* cuya apariencia se observa en la figura A.5.

Figura A.5: Visualización de la pantalla de inicio de sesión

Además, si ha llegado a esta pantalla sin estar registrado, observe que en la parte inferior de la misma se ofrece la opción *crear una*, que hace referencia precisamente al registro. Si desea más información acerca de ello, por favor lea la sección A.2.

A.4. Menú de la aplicación

Una vez que se haya iniciado sesión (ver sección A.3), se visualizará un menú situado en la parte superior derecha de la aplicación. El contenido de este menú variará dependiendo del rol del usuario en la aplicación. Puede ver un ejemplo de menú para un usuario administrador en la figura A.6.

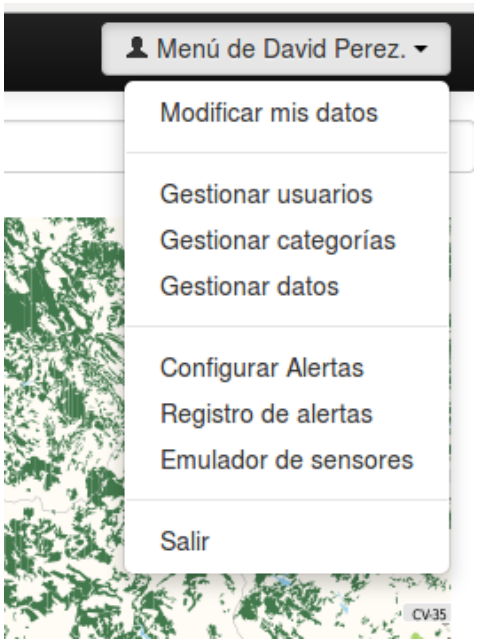

Figura A.6: Vista del menú de un usuario administrador

A.5. Edición de datos de un usuario

Todo usuario registrado en la aplicación, puede modificar sus datos cuando lo desee en la pantalla de *Editar datos*, a la cual se puede acceder a través de la opción *Modificar mis datos* del menú superior derecho. Puede verse un ejemplo en la figura A.7.

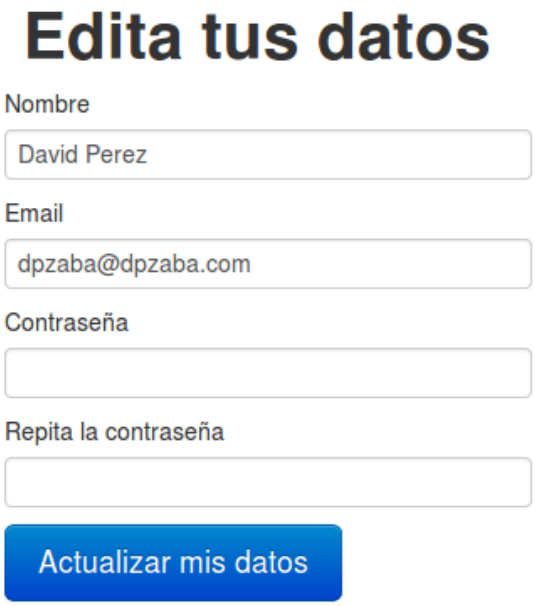

Figura A.7: Vista del formulario para editar los datos de un usuario

A.6. Gestión de usuarios

Los usuarios administradores, pueden acceder a la pantalla de gestión de usuarios. En esta se presentan todos los usuarios de la aplicación y permite borrarlos, o gestionar su rol en la aplicación. Puede ver un ejemplo de esta pantalla en la figura A.8.

Figura A.8: Vista principal para la gestión de usuarios

A.7. Gestión de categorías

Los usuarios administradores, pueden acceder a la pantalla de gestión de categorías. En esta se presentan todos las categorías de la aplicación y permite borrarlas, modificarlas o añadir nuevas. Puede ver un ejemplo de la pantalla principal para gestionar las categorías en la figura A.9 y otro ejemplo al añadir una categoría en la figura A.10.

Figura A.9: Vista principal para la gestión de categorías

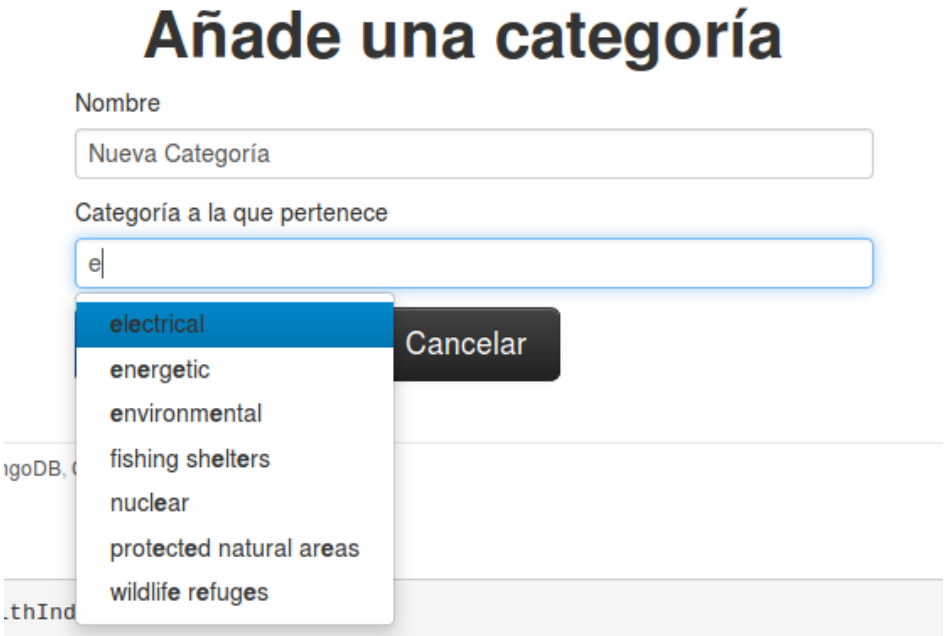

Figura A.10: Visualización de la pantalla para añadir una categoría

A.8. Gestión de datos geográficos

Los usuarios administradores, pueden acceder a la pantalla de gestión de datos geográficos. En esta se presentan todos los datos almacenados en la aplicación. Se pueden añadir nuevos datos o borrar los existentes.

Un ejemplo de la pantalla principal para gestionar los datos geográficos en la figura A.11 y otro ejemplo al añadir unos datos geográficos a una categoría en la figura A.12.

Para más información acerca de las categorías, vaya a la sección A.7.

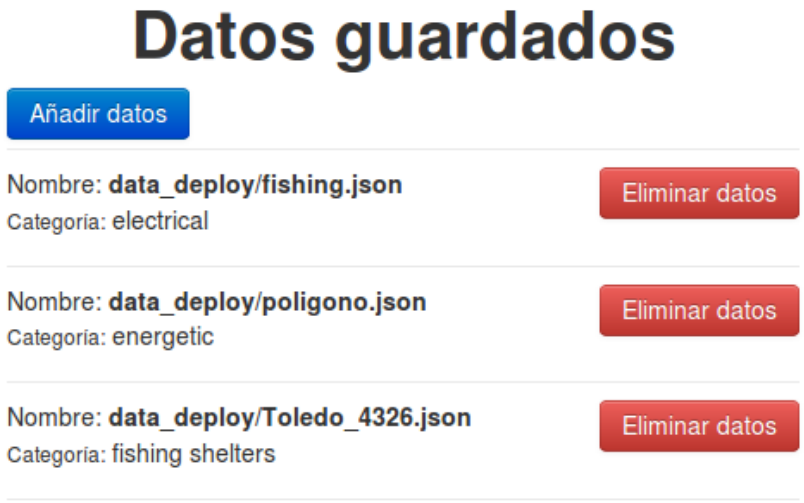

Figura A.11: Vista principal para la gestión de datos geográficos

A.9. Emulador

Los usuarios administradores, pueden acceder a la pantalla del emulador. En esta se muestra un formulario para introducir los datos del dispositivo y sensor de los que se desean generar datos aleatorios. Una vez introducidos el nombre del dispositivo y del sensor a simular, debe presionarse el botón *Generar datos*. Puede encontrarse un ejemplo de esta pantalla en la figura A.13.

Una vez generados los datos aleatorios, se muestran en la pantalla (ver figura A.14), dando la opción de poder enviarlos al sistema o recargar la página para generar otros datos.

Figura A.12: Visualización de la pantalla para añadir nuevos datos geográficos

Figura A.13: Vista principal del emulador para generar datos aleatorios

Figura A.14: Vista del emulador para enviar los datos generados al sistema

A.10. Gestión de alertas

Los usuarios administradores, pueden acceder a la pantalla de las alertas registradas. En esta se muestra una lista con todos los sensores que han producido alguna alerta e información relativa a esta. Puede encontrarse un ejemplo de esta pantalla en la figura A.15.

Figura A.15: Vista de las alertas registradas

A.11. Gestión de dispositivos

Sólo pueden acceder a este recurso los usuarios administradores. En esta pantalla se podrá borrar todos los datos almacenados de cada dispositivo, o configurar las alertas de cada sensor. Puede ver un ejemplo de esta pantalla en la figura A.16.

Figura A.16: Vista principal de la configuración de los dispositivos

A.12. Mensajes del sistema

La aplicación muestra un gran número de mensajes para informar al usuario sobre las acciones que realiza. Tanto mensajes generales sobre acciones realizadas (ver figura A.17) para recordar al usuario la acción realizada, como errores que se han producido al realizar alguna petición (ver figura A.18).

También es capaz de mostrar que se ha producido un error, al rellenar alguno de los formularios de la aplicación (ver figura A.19).

Los usuarios administradores, también podrán visualizar las alertas que se han producido desde la pantalla principal de la aplicación (ver figura A.20).

Figura A.17: Mensaje de información al usuario

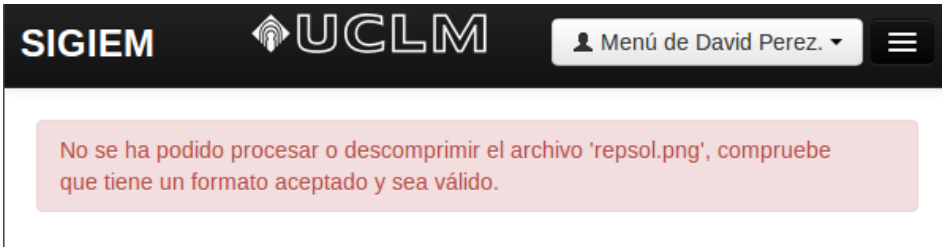

Figura A.18: Mensaje de error al usuario

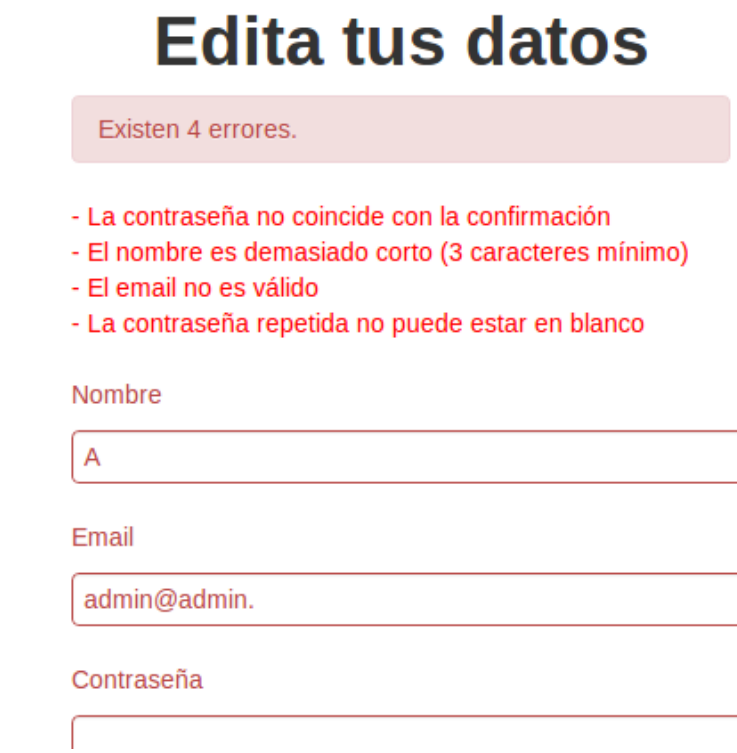

Figura A.19: Mensajes de error de un formulario

Figura A.20: Mensaje de alerta de los sensores

A.13. Adaptación de la interfaz

SIGIEM es capaz de adaptarse a multitud de pantallas, tanto pantallas con una resolución muy alta (1920x1200) como a dispositivos móviles con resoluciones bajas (480x800). Puede ver una imagen de la aplicación, en una pantalla de un móvil con una resolución de 480x800 en la figura A.21.

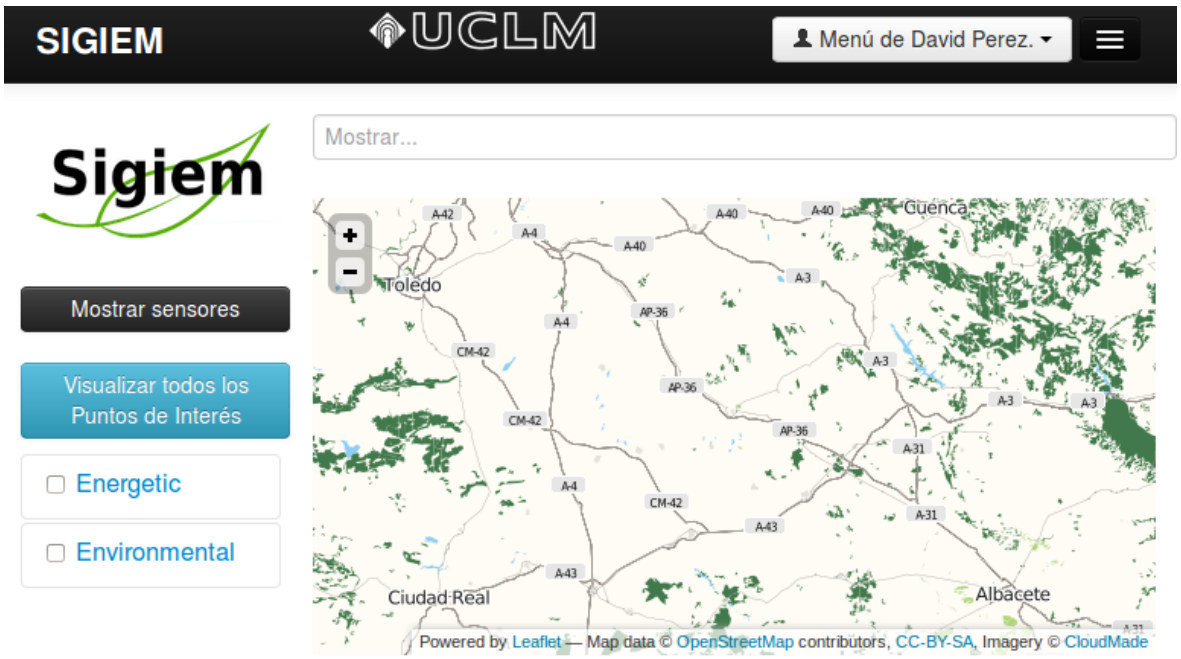

Figura A.21: Vista de la pantalla principal de la aplicación, en una pantalla con una resolución de 480x800

A.14. Acerca de

Se puede consultar más información sobre SIGIEM en la sección de *Acerca de* (ver figura A.22).

Figura A.22: Visualización de información sobre SIGIEM

Anexo B

Manual de instalación y despliegue

El manual de instalación y despliegue se hará para el sistema operativo *Ubuntu*, en concreto la versión 12.04. Aunque, los pasos realizados serán casi idénticos si se utilizase *Debian* o alguna otra distribución *GNU/Linux* derivada de alguna de estas.

B.1. Instalación de las herramientas necesarias

B.1.1. Lenguaje de programación

```
1   sudo apt-get install ruby1.9.3
```

B.1.2. Bibliotecas utilizadas

OGR

La herramienta utilizada para transformar datos geográficos es *OGR* [ogr], que se encuentra dentro de la biblioteca *GDAL*. Para instalarla se debe ejecutar:

```
1   sudo apt-get install gdal-bin
```

Dependencias del proyecto

Primero instalamos el gestor de dependencias *Bundler* [bun]:

```
1   sudo gem install bundler
```

Gracias al archivo *Gemfile* (ver Anexo E), que describe todas las dependencias del proyecto y al sistema de paquetes de *Ruby*, sólo es necesario ejecutar el siguiente comando:

```
1   sudo bundle install
```

Esto bajará todas las bibliotecas que se utilizan (si no se tienen ya instaladas) y en la versión especifica que se necesitan para el proyecto.

B.1.3. Base de datos

Para instalar la base de datos *MondoDB* [10g], sólo es necesario ejecutar:

```
sudo apt-get install mongodb
```

Y si se desea acceder al a consola de *MongoDB* para gestionarla hay que ejecutar:

```
mongo
```

Interfaz gráfica para la gestión de MongoDB

Si se desea, se pueden utilizar algunas aplicaciones con interfaces gráficas que ayudan a gestionar la base de datos. Hay muchas aplicaciones disponibles, pero dos de las mejores que he encontrado, gratuitas y de código abierto son:

mViewer [mon]

Hay que bajarse la versión correspondiente al sistema operativo utilizado desde `https://github.com/Imaginea/mViewer`, descomprimir el contenido y situarse en esa carpeta para ejecutar:

```
./start_mViewer.sh
```

De esta forma, se puede acceder a la interfaz gráficas desde un navegador web en la dirección `http://localhost:8080`.

Humongous [Pan]

Se puede utilizar como otra biblioteca del proyecto para tenerla siempre disponible. Para su instalación hay que añadirla al fichero *Gemfile* e instalar las dependencias de nuevo o ejecutar:

```
gem install humongous
```

Y al ejecutar el siguiente comando, será accesible desde un navegador web en la dirección `http://localhost:9000`.

```
humongous
```

B.1.4. Control de versiones

El sistema de control de versiones utilizado ha sido *Git* [git], para instalarlo se debe ejecutar:

```
sudo apt-get install git git-core
```

B.1.5. Servidor web

El servidor web utilizado es *Apache* [apa], para instalarlo hay que ejecutar:

```
sudo apt-get install apache2
```

Para poder ejecutar *Ruby* en *Apache* se necesita instalar *Phusion Passenger*:

```
sudo gem install passenger
```

Para enlazarlo con *Apache* es necesario ejecutar el siguiente código y seguir las instrucciones que se indican:

```
sudo passenger-install-apache2-module
```

B.2. Puesta en marcha

B.2.1. Obtener la aplicación

Lo primero que hay que haces es tener la aplicación en un directorio, para ello debe bajarse del repositorio donde está alojada (Bitbucket [bit]).

```
1  git clone https://dpzaba@bitbucket.org/dpzaba/sigiem.git
```

Si ya tuviésemos la aplicación descargada en nuestro sistema, lo único que habría que hacer sería obtener la última versión disponible en el repositorio:

```
1  git pull -u
```

B.2.2. Lanzar el servidor

Hay que crear un *VirtualHost* de *Apache* en el puerto 80 (o en el deseado). Para ello debemos modificar (con permisos de administrador) el archivo que *Apache* crea por defecto en `/etc/apache2/sites-available/default`, y dejarlo de la forma en que se muestra en el listado B.1.

```
1  <VirtualHost *:80>
2    ServerAdmin admin@email.com
3
4    ServerName sigiem.uclm.es
5    DocumentRoot /home/david/sigiem/sigiem/public/
6
7    <Directory /home/david/sigiem/sigiem/public>
8      Options MultiViews
9      AllowOverride all
10   </Directory>
11
12   ErrorLog ${APACHE_LOG_DIR}/error.log
13
14   # Possible values include: debug, info, notice, warn, error,
        crit,
15   # alert, emerg.
16   LogLevel warn
17
18   CustomLog ${APACHE_LOG_DIR}/access.log combined
```

```
20 </VirtualHost>
```

Listado B.1: Configuración del host virtual de *Apache* para SIGIEM

Como se observa en B.1, se debe indicar la ruta donde se ha descargado el proyecto y el nombre del servidor.

Aunque por defecto el servicio del servidor web debe arrancar sólo, depende de la distribución *GNU/Linux* que utilicemos, así pues para asegurarse de que el servidor está arrancado ejecutamos:

```
1 sudo service apache2 start
```

Por último si no arranca de inmediato, se puede ejecutar el siguiente comando para que se reinicie el *host*.

```
1   touch /home/david/sigiem/tmp/restart.txt
```

Anexo C

Resultados del análisis de Vega

Scan Alert Summary

High	(None found)
Medium	(1 found)
Local Filesystem Paths Found	1
Low	(10 found)
Form Password Field with Autocomplete Ena	5
Directory Listing Detected	3
Email Addresses Found	2
Info	(302 found)
Interesting Meta Tags Detected	271
Blank Body Detected	28
HTTP Error Detected	3

Figura C.1: Resultado de Vega sin configuración adicional

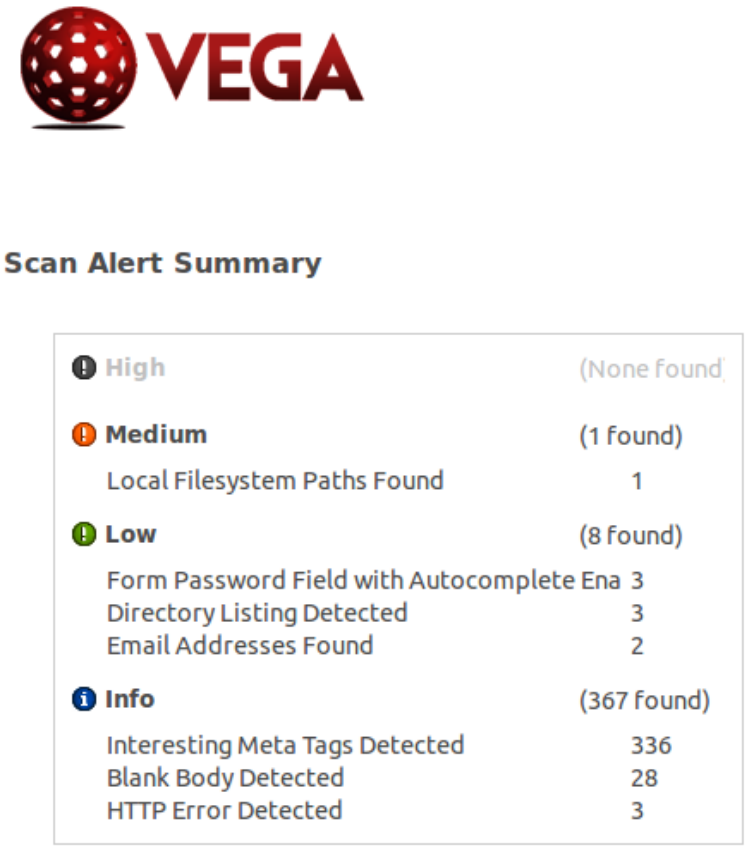

Figura C.2: Resultado de Vega utilizando una cookie de administrador

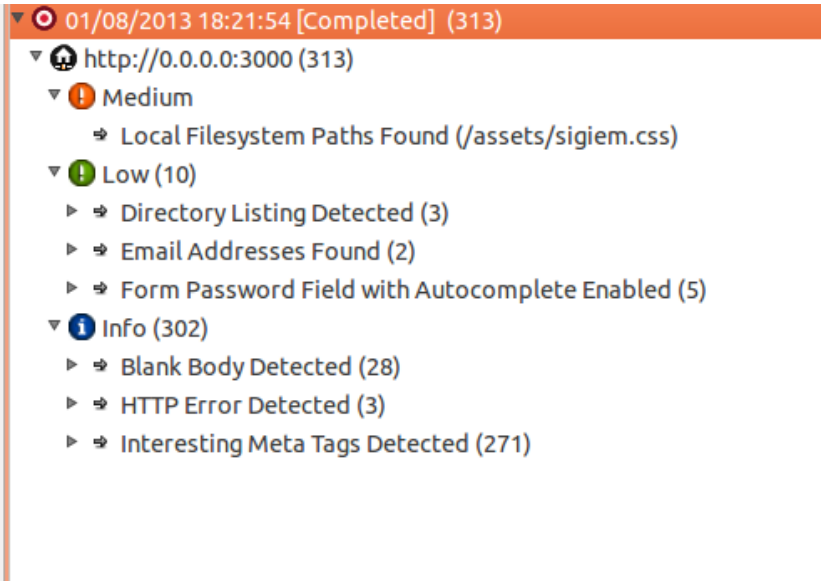

Figura C.3: Tipos de fallos encontrador por Vega

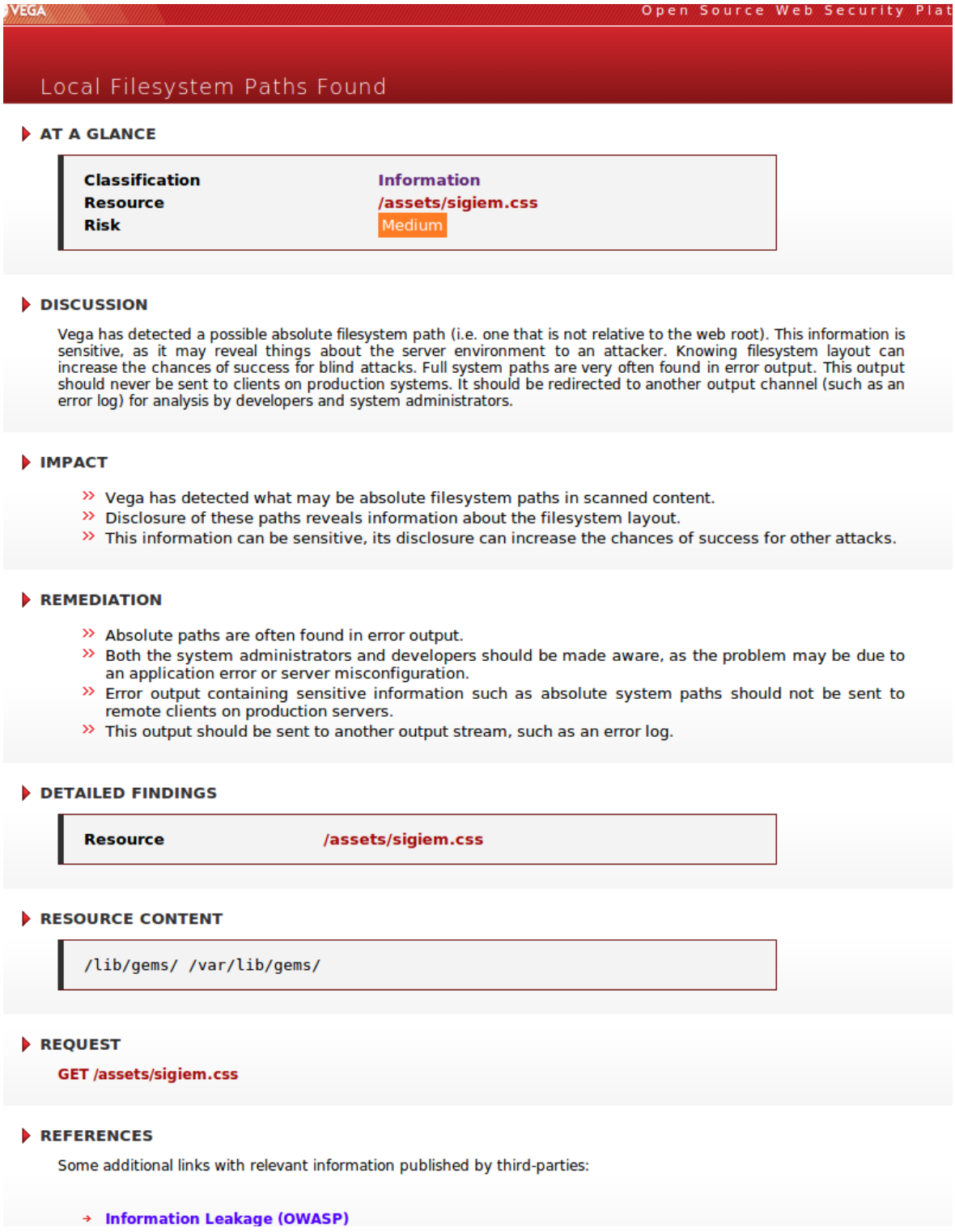

Figura C.4: Explicación y recomendación de Vega ante los fallos encontrados

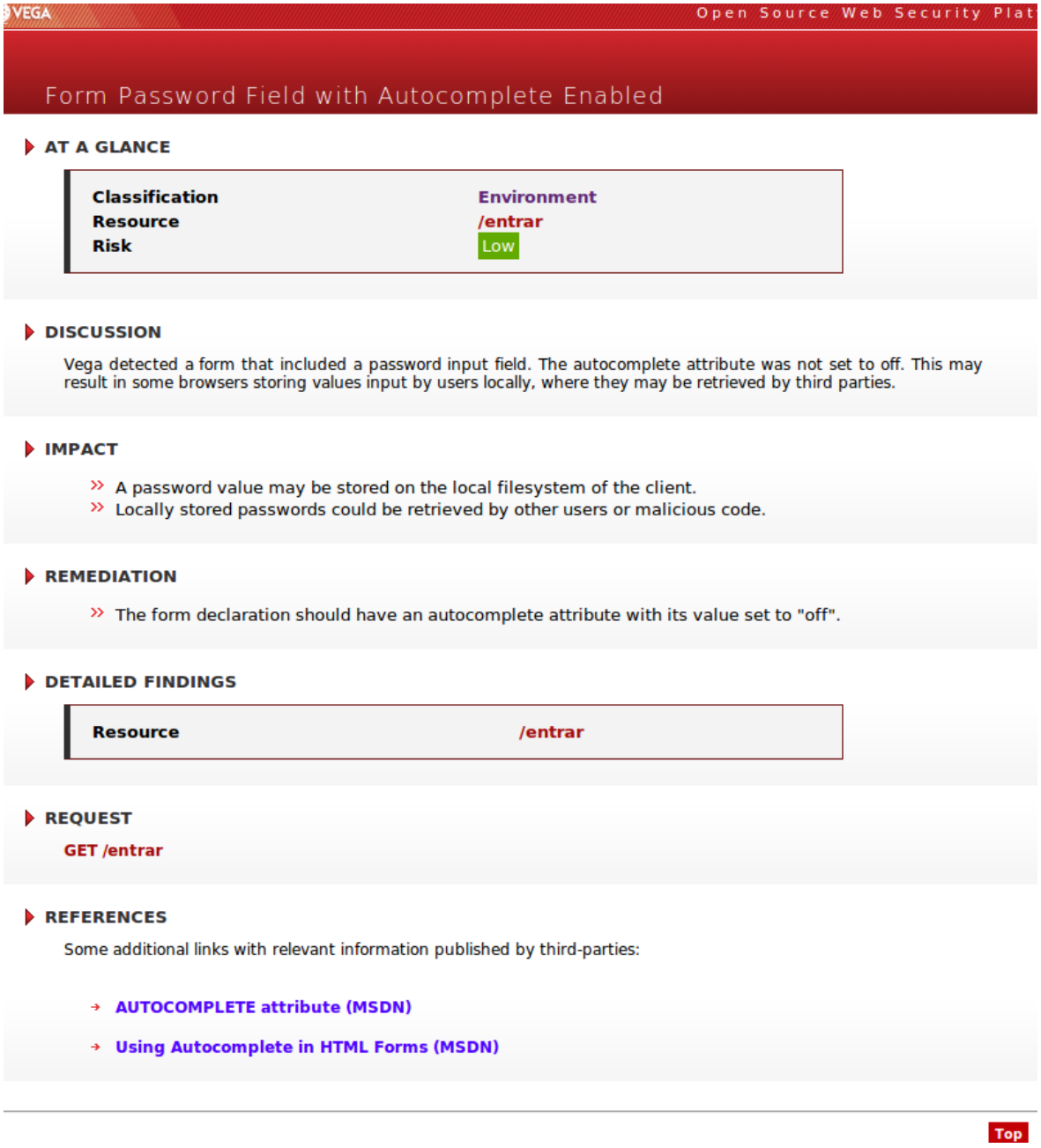

Figura C.5: Explicación y recomendación de Vega ante los fallos encontrados

Anexo D

Cobertura de las pruebas

Generated **5 minutes ago**

Models (100.0% covered at 514.17 hits/line)

7 files in total. **132** relevant lines. **132** lines covered and **0** lines missed

Search:

File	% covered	Lines	Relevant Lines	Lines covered	Lines missed	Avg. Hits / Line
app/models/alert.rb	**100.0 %**	51	27	27	0	3378.8
app/models/category.rb	**100.0 %**	32	18	18	0	85.7
app/models/datum.rb	**100.0 %**	42	18	18	0	14.5
app/models/device.rb	**100.0 %**	48	25	25	0	76.8
app/models/sensor.rb	**100.0 %**	24	14	14	0	19.4
app/models/user.rb	**100.0 %**	40	23	23	0	23.0
app/models/wms_server.rb	**100.0 %**	14	7	7	0	1.0

Showing 1 to 7 of 7 entries

Generated **8 minutes ago**

All Files (77.52%) **Controllers (86.67%)** **Models (100.0%)** **Mailers (100.0%)** **Helpers (97.73%)** **Libraries (8.08%)** **Plugins (100.0%)**

Controllers (86.67% covered at 52.39 hits/line)

10 files in total. **330** relevant lines. **286** lines covered and **44** lines missed

Search:

File	% covered	Lines	Relevant Lines	Lines covered	Lines missed	Avg. Hits / Line
app/controllers /record_alerts_controller.rb	27.59 %	56	29	8	21	0.3
app/controllers /data_controller.rb	85.54 %	168	83	71	12	3.2
app/controllers /devices_controller.rb	91.18 %	127	68	62	6	415.7
app/controllers /categories_controller.rb	94.87 %	86	39	37	2	12.9
app/controllers /emulators_controller.rb	95.65 %	49	23	22	1	4.0
app/controllers /wms_servers_controller.rb	96.3 %	53	27	26	1	7.2
app/controllers /users_controller.rb	97.44 %	73	39	38	1	5.9
app/controllers /application_controller.rb	100.0 %	5	4	4	0	1.0

Generated **7 minutes ago**

Helpers (97.73% covered at 136.69 hits/line)

7 files in total. **44** relevant lines. **43** lines covered and **1** lines missed

Search:

File	% covered	Lines	Relevant Lines	Lines covered	Lines missed	Avg. Hits / Line
app/helpers/main_pages_helper.rb	85.71 %	15	7	6	1	41.4
app/helpers/application_helper.rb	100.0 %	3	1	1	0	1.0
app/helpers/categories_helper.rb	100.0 %	18	6	6	0	125.5
app/helpers/data_helper.rb	100.0 %	2	1	1	0	1.0
app/helpers/sessions_helper.rb	100.0 %	48	21	21	0	688.8
app/helpers/users_helper.rb	100.0 %	21	7	7	0	98.1
app/helpers/wms_servers_helper.rb	100.0 %	2	1	1	0	1.0

Showing 1 to 7 of 7 entries

Anexo E

Archivo Gemfile

```
source 'https://rubygems.org'

gem 'rails', '3.2.9'

#mongodb
gem 'mongo_mapper', github: "jnunemaker/mongomapper"
gem 'mongo'
gem 'bson_ext'

#bootstrap
gem 'bootstrap-sass'

#bcrypt for authentication (has_secure_password)
gem 'bcrypt-ruby'

#pagination for resources
gem 'will_paginate'
gem 'bootstrap-will_paginate'

#internationalize the app
gem 'rails-i18n'

#json
gem 'json'

#ruby library for reading and writing zip files
gem 'rubyzip'

gem 'jquery-rails'

gem 'thin'
```

```
33 group :development, :test do
34   gem 'rspec-rails', '2.10.0'
35   gem 'guard-rspec'
36 end

38 #integration test
39 group :test do
40   #integration test
41   gem 'capybara', '1.1.2'

43   #make available open pages while test
44   gem 'launchy'

46   #factories
47   gem 'factory_girl_rails'

49   #spork, speeding up tests
50   gem 'spork'

52   #spork with guard
53   gem 'guard-spork'

55   #gems only for linux
56   gem 'rb-inotify'
57   gem 'libnotify'

59   #test coverage
60   gem 'simplecov', :require => false
61 end

63 group :assets do
64   gem 'sass-rails',   '~> 3.2.3'
65   gem 'coffee-rails', '~> 3.2.1'
66   gem 'uglifier', '>= 1.0.3'
67 end
```

Listado E.1: Archivo Gemfile con todas las bibliotecas utilizadas en el proyecto

Anexo F

GNU Free Documentation License

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

   "This License" refers to version 3 of the GNU General Public License.

   "Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

   "The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

   To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

   A "covered work" means either the unmodified Program or a work based on the Program.

   To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright

law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

   The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

   A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

   The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

   The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any

non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

   You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

   a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

   b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

   c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

   d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

   A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

   You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

   a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A “User Product” is either (1) a “consumer product”, which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, “normally used” refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such

uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered

work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automa-

tically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

   You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

    Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

    An “entity transaction” is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party’s predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey,

or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

    If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

    Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

    The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit

to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>

Copyright (C) <textyear>  <name of author>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
```

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

```
<program>  Copyright (C) <year>  <name of author>

This program comes with ABSOLUTELY NO WARRANTY; for details type ‘show w’.
This is free software, and you are welcome to redistribute it
under certain conditions; type ‘show c’ for details.
```

The hypothetical commands `show w` and `show c` should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see `http://www.gnu.org/licenses/`.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read `http://www.gnu.org/philosophy/why-not-lgpl.html`.

Referencias

[10g] Inc. 10gen. MongoDB. url: http://www.mongodb.org/.

[3dj] Three.js - Javascript 3D library. url: http://mrdoob.github.com/three.js.

[act] ActiveModel::SecurePassword::ClassMethods. url: http://api.rubyonrails.org/classes/ActiveModel/SecurePassword/ClassMethods.html.

[Ali12] David Villa Alises. Clase arco-pfc, 2012. url: https://bitbucket.org/arco_group/arco-pfc.

[apa] Apache HTTP Server. url: https://httpd.apache.org.

[BD91] E. Bersoff y A. Davis. Impacts of Life Cycle Models of Software Configuration Management. *ACM*, páginas 104–118, 1991.

[Bec03] Kent Beck. *Test-Driven Development by Example*. Addison-Wesley Professional, 2003.

[bib] BibTeX. url: http://www.bibtex.org.

[bit] Bitbucket. url: https://bitbucket.org.

[boo] Bootstrap. url: http://twitter.github.com/bootstrap/.

[bun] Bundler. url: http://gembundler.com.

[cap] Capybara. url: https://github.com/jnicklas/capybara.

[cloa] CLOC: Count Lines of Code. url: http://cloc.sourceforge.net/.

[clob] CloudMade. url: http://cloudmade.com/.

[CV] Mahil Carr y June Verner. Prototyping and Software Development Approaches.

[Dav92] Alan M. Davis. Operational Prototyping: A new Development Approach. *IEEE Software*, página 71, 1992.

[DB86] B. Diaz y S. Bell. *Spatial Data Processing using Tesseral Methods*. 1986.

[dia] Dia. url: http://live.gnome.org/Dia.

[ecm] Standard ECMA-262. url: http://www.ecma-international.org/publications/standards/Ecma-262.htm.

[ema] GNU Emacs. url: https://www.gnu.org/software/emacs.

[ESR] ESRI. What is GIS? url: http://www.esri.com/what-is-gis.

[fac] Factory Girl. url: https://github.com/thoughtbot/factory_girl.

[Fie00] Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, 2000.

[fir] Mozilla Firefox. url: https://www.mozilla.org/es-ES/firefox/.

[FM08] David Flanagan y Yukihiro Matsumoto. *The Ruby Programming Language*. O'Reilly Media, 2008.

[fou] Foursquare. url: https://es.foursquare.com/.

[Fow03] Martin Fowler. *Patterns of enterprise application architecture*. 2003.

[gim] GIMP. url: http://www.gimp.org/.

[git] Git. url: http://git-scm.com.

[gooa] Google Earth. url: https://www.google.com/earth.

[goob] Google Maps. url: https://maps.google.com.

[Goo07] M. Goodchild. Citizens as sensors: the world of volunteered geography, 2007. url: http://www.esf.edu/nysgisconf/2008/Goodchild_VGI2007.pdf.

[gpl] GNU General Public License v3.0. url: https://www.gnu.org/licenses/gpl.html.

[gua] Guard. url: https://github.com/guard/guard.

[HB08] Allan Doyle et all Howard Butler, Martin Daly. The GeoJSON Format Specification, Junio 2008. url: http://www.geojson.org/geojson-spec.html.

[htt] RFC 2616 - Hypertext Transfer Protocol - HTTP/1.1. url: https://tools.ietf.org/html/rfc2616.

[idea] Infraestructura de Datos Espaciales de Castilla-La Mancha. url: http://ide.jccm.es.

[ideb] Infraestructura de Datos Espaciales de España. url: www.idee.es.

[idr] Instituto de Desarrollo Regional. url: http://www.idr-ab.uclm.es.

[ign] Instituto Geográfico Nacional. url: http://www.ign.es.

[ink] Inkscape. url: http://inkscape.org/.

[jab] JabRef. url: http://jabref.sourceforge.net/.

[jqu] jQuery. url: http://jquery.com.

[lat] LaTeX. url: http://www.latex-project.org/.

[lea] Leaflet. url: http://leafletjs.com/.

[Lef11] Dean Leffingwell. *Agile Software Requirements: Lean Requirements Practices for Teams, Programs and the Enterprise*. 2011.

[lib] Libelium Comunicaciones Distribuidas. url: http://www.libelium.com.

[Mel08] J. Mellado. *Ajax*. 2008.

[mem] Memcached. url: http://memcached.org/.

[mon] mViewer. url: http://imaginea.com/mviewer.

[moz] Mozilla Style guide. url: https://www.mozilla.org/en-US/styleguide/identity/firefox/branding/.

[Nor06] Dan North. Introducing BDD. *Better Software*, 2006. url: http://dannorth.net/introducing-bdd.

[nor07] Normativa del Proyecto Fin de Carrera, Noviembre 2007. url: http://webpub.esi.uclm.es/archivos/89/NormativaPFC2007.

[ogr] OGR Simple Feature Library. url: http://www.gdal.org/ogr.

[Ola12] Victor Olaya. *Sistemas de Información Geográfica*. 2012.

[ope] Catálogo de Información Pública de la Administración General del Estado. url: http://datos.gob.es/datos/.

[osm] OpenStreetMap. url: http://www.openstreetmap.org/.

[Pan] Bagwan Pankaj. Humongous. url: https://github.com/bagwanpankaj/humongous.

[pas] Phusion Passenger. url: https://www.phusionpassenger.com/.

[Pil10] M. Pilgrim. *HTML5: up and running*. 2010.

[qgi] Quantum GIS (QGIS). url: http://www.qgis.org/.

[raia] Performance Testing Rails Applications. url: http://guides.rubyonrails.org/performance_testing.html.

[raib] Ruby on Rails. url: http://rubyonrails.org/.

[raic] Ruby On Rails Security Guide. url: http://guides.rubyonrails.org/security.html.

[rak] Ruby Make. url: http://rake.rubyforge.org/.

[Ree79] T. Reenskaug. Models-views-controllers. Technical report, Xerox PARC, 1979.

[res] REST: Representational State Transfer. url: http://bibing.us.es/proyectos/abreproy/11247/fichero/Memoria%252F8-Representational+State+Transfer+%28REST%29.pdf.

[rsp] Rspec. url: http://rspec.info.

[ruba] Ruby. url: http://www.ruby-lang.org/es/about/.

[rubb] Ruby License. url: http://www.ruby-lang.org/es/about/license.txt.

[sde] Quick Sequence Diagram Editor. url: http://sdedit.sourceforge.net.

[SE90] J. Star y J. Estes. *Geographic Information Systems: An Introduction*. 1990.

[sim] SimpleCov. url: https://github.com/colszowka/simplecov.

[spo] Spork. url: https://github.com/sporkrb/spork.

[thi] Thin. url: http://code.macournoyer.com/thin/.

[Tom90] C.D. Tomlin. *Geographic information systems and cartographic modelling*. 1990.

[ubu] Ubuntu. url: http://www.ubuntu.com/.

[veg] Vega. url: http://subgraph.com/products.html.

Este documento fue editado y tipografiado con LaTeX
empleando la clase **arco-pfc** que se puede encontrar en:
`https://bitbucket.org/arco_group/arco-pfc`