



**Universidad de Oviedo**

ESCUELA DE INGENIERÍA INFORMÁTICA

**Trabajo Fin de Grado**

**Backend de un portal de datos e  
información sobre la Tierra**

*Autor:*  
Cristian Álvarez Belaustegui

*Director:*  
Jose Emilio Labra Gayo



# Información para el lector

Los dispositivos electrónicos están cada vez más presentes en todos los ámbitos de nuestra vida. Actualmente dispositivos como *smartphones* y *tablets* juegan un papel cada vez más importante facilitando el consumo de información. Con el objetivo de facilitar la lectura de esta documentación desde este tipo de dispositivos se han introducido algunas características que se enumerarán a continuación:

- Todas las referencias a capítulos, secciones, imágenes y URLs son naveгables pulsando sobre ellas. Esto permite al lector navegar de forma sencilla y rápida a través de la documentación.
- Los diversos diagramas que se incluyen en la documentación se encuentran en un formato vectorial. Esto permite al lector ampliarlos tanto como desee para ver todos los detalles sin pérdida de calidad.



# **Agradecimientos**

A mis padres, quienes siempre me han demostrado el valor del trabajo, la dedicación y las cosas bien hechas. A mi hermana, porque su esfuerzo sirvió en gran parte como inspiración en los momentos de flaqueza. A mi novia, quien me apoyó en todo momento y sin la cual nada sería lo mismo. A mi director y a todos los miembros de WESO, por crear un ambiente de trabajo en el que hasta momentos más duros merecieron la pena.



# Resumen

El presente proyecto tiene como objetivo la creación de la infraestructura de soporte para un portal de datos abiertos. Concretamente el portal será el nuevo Land Portal, propiedad del IFAD (Fondo Internacional para el Desarrollo Agrícola) y perteneciente a la ONU (Organización de las Naciones Unidas). Este Proyecto Fin de Grado forma parte del proyecto *Rebuilding IFAD's Land-Portal - RFQ/2013/016/SC* desarrollado por el grupo de investigación WESO (Web Semantics Oviedo) y la empresa SB Consulting.

El sistema resultante está formado por dos secciones principales:

- Por un lado una sección orientada a los datos (LandBook), que permite la publicación y consulta de catálogos de datos en varios formatos como JSON, XML o RDF. Los catálogos de datos almacenados en el sistema son utilizados, entre otros, por unas visualizaciones que los presentan de forma gráfica y atractiva con el objetivo de facilitar su interpretación por parte de los usuarios.
- Por otro lado una sección social (LandDebate) que permite la creación de noticias, eventos y debates con los que fomentar la participación e implicación de los usuarios. Esta sección también alberga el blog del nuevo Land Portal.

Una parte importante del sistema es su soporte a la internacionalización en todas sus secciones. La internacionalización es completa e incluye tanto las interfaces de usuarios como el propio modelo de datos. Esta característica permite que el nuevo Land Portal sea utilizado por multitud de personas y organizaciones pertenecientes a diversos países.

El sistema está desarrollado utilizando varias tecnologías diferentes: varios lenguajes de programación (PHP, Python JavaScript), un gestor de contenidos o CMS (Drupal), un catálogo de datos (CKAN), una base de datos relacional (MySQL), una base de datos no relacional (Virtuoso) y un framework de desarrollo web (Flask).

Para la realización de este proyecto se han utilizado los conocimientos adquiridos durante los estudios del Grado en Ingeniería Informática del Software.



# Índice general

<b>1. Introducción</b>	<b>16</b>
1.1. Justificación del Proyecto . . . . .	16
1.2. Objetivos del proyecto . . . . .	16
1.3. Estudio de la situación actual . . . . .	17
1.3.1. Portal de datos del Gobierno de Estados Unidos . . . . .	17
1.3.2. Portal de datos del Gobierno Británico . . . . .	18
1.3.3. Land Matrix . . . . .	19
1.3.4. Antiguo Land Portal . . . . .	20
<b>2. Evaluación de alternativas</b>	<b>21</b>
2.1. Alternativas evaluadas . . . . .	21
2.1.1. Gestor de contenidos . . . . .	21
2.1.1.1. Joomla . . . . .	21
2.1.1.2. Wordpress . . . . .	22
2.1.1.3. Drupal . . . . .	22
2.1.2. Catálogo de datos . . . . .	23
2.1.2.1. CKAN . . . . .	23
2.1.2.2. DKAN . . . . .	23
2.1.2.3. Herramienta creada especialmente para la ocasión . . . . .	23
2.1.3. Servidor semántico . . . . .	23
2.1.3.1. Virtuoso Universal Server / Virtuoso OpenLink . . . . .	24
2.1.3.2. Stardog . . . . .	24
2.1.3.3. 4store . . . . .	24
2.1.4. Visualizador de datos enlazados . . . . .	25
2.1.4.1. Pubby . . . . .	25
2.1.4.2. Wesby . . . . .	25
2.1.5. Framework de diseño web . . . . .	26
2.1.5.1. Twitter Bootstrap . . . . .	26
2.1.5.2. ZURB Foundation . . . . .	26
2.1.5.3. Desarrollo desde cero . . . . .	26
2.1.6. Motor de búsqueda . . . . .	26
2.1.6.1. Apache Solr . . . . .	27
2.1.6.2. Elasticsearch . . . . .	27
2.1.6.3. Buscador propio del gestor de contenidos . . . . .	27
2.2. Alternativas elegidas . . . . .	28
2.2.1. Gestor de contenidos . . . . .	28
2.2.2. Catálogo de datos . . . . .	28
2.2.3. Servidor semántico . . . . .	28
2.2.4. Visualizador de datos enlazados . . . . .	29
2.2.5. Framework de diseño web . . . . .	29
2.2.6. Motor de búsqueda . . . . .	29
<b>3. Conceptos teóricos</b>	<b>30</b>
3.1. Datos abiertos . . . . .	30
3.2. Datos enlazados . . . . .	31
3.3. Datos enlazados abiertos . . . . .	31

3.3.1. Sistema de estrellas . . . . .	31
3.4. RDF Data Cube . . . . .	32
3.4.1. El modelo de cubo . . . . .	32
3.4.2. Los <i>slices</i> . . . . .	33
<b>4. Análisis</b>	<b>34</b>
4.1. Definición del sistema . . . . .	34
4.1.1. Alcance del sistema . . . . .	34
4.1.1.1. Elementos dentro del alcance del proyecto . . . . .	34
4.1.1.2. Elementos fuera del alcance del proyecto . . . . .	35
4.2. Identificación de actores del sistema . . . . .	36
4.3. Requisitos del sistema . . . . .	37
4.3.1. Especificación de los requisitos funcionales . . . . .	37
4.3.1.1. Requisitos de la sección de datos . . . . .	37
4.3.1.2. Requisitos de la sección social . . . . .	38
4.3.1.3. Requisitos de la búsqueda . . . . .	41
4.3.1.4. Requisitos del punto de entrada de datos . . . . .	42
4.3.2. Especificación de los requisitos no funcionales . . . . .	43
4.4. Identificación de subsistemas . . . . .	43
4.4.1. Descripción de las interfaces entre subsistemas . . . . .	44
4.5. Especificación de casos de uso . . . . .	45
4.5.1. Subsistema de gestión de usuarios . . . . .	46
4.5.1.1. Caso de uso “registrararse” . . . . .	46
4.5.1.2. Caso de uso “activar usuario” . . . . .	47
4.5.1.3. Caso de uso “iniciar sesión” . . . . .	47
4.5.1.4. Caso de uso “cerrar sesión” . . . . .	48
4.5.1.5. Caso de uso “ver perfil” . . . . .	48
4.5.1.6. Caso de uso “actualizar perfil” . . . . .	48
4.5.2. Subsistema de gestión de debates . . . . .	49
4.5.2.1. Caso de uso “ver debate” . . . . .	49
4.5.2.2. Caso de uso “crear debate” . . . . .	50
4.5.2.3. Caso de uso “modificar debate” . . . . .	50
4.5.2.4. Caso de uso “eliminar debate” . . . . .	51
4.5.3. Subsistema de gestión de eventos . . . . .	51
4.5.3.1. Caso de uso “ver evento” . . . . .	51
4.5.3.2. Caso de uso “crear evento” . . . . .	52
4.5.3.3. Caso de uso “modificar evento” . . . . .	52
4.5.3.4. Caso de uso “eliminar evento” . . . . .	53
4.5.4. Subsistema de gestión de noticias . . . . .	53
4.5.4.1. Caso de uso “ver noticia” . . . . .	53
4.5.4.2. Caso de uso “crear noticia” . . . . .	54
4.5.4.3. Caso de uso “modificar noticia” . . . . .	54
4.5.4.4. Caso de uso “eliminar noticia” . . . . .	55
4.5.5. Subsistema de gestión del blog . . . . .	55
4.5.5.1. Caso de uso “ver entrada” . . . . .	55
4.5.5.2. Caso de uso “crear entrada” . . . . .	56
4.5.5.3. Caso de uso “modificar entrada” . . . . .	57
4.5.5.4. Caso de uso “eliminar entrada” . . . . .	57
4.5.6. Subsistema de gestión de comentarios . . . . .	57
4.5.6.1. Caso de uso “crear comentario” . . . . .	57
4.5.6.2. Caso de uso “modificar comentario” . . . . .	58
4.5.6.3. Caso de uso “eliminar comentario” . . . . .	59
4.5.7. Subsistema de gestión de organizaciones . . . . .	59
4.5.7.1. Caso de uso “ver organización” . . . . .	59
4.5.7.2. Caso de uso “crear organización” . . . . .	60
4.5.7.3. Caso de uso “modificar organización” . . . . .	60
4.5.7.4. Caso de uso “eliminar organización” . . . . .	61
4.5.8. Subsistema de búsqueda . . . . .	61

4.5.8.1. Caso de uso “realizar una búsqueda” . . . . .	61
4.5.8.2. Caso de uso “actualizar índice de contenidos” . . . . .	62
4.5.9. Subsistema de gestión de datos . . . . .	62
4.5.9.1. Caso de uso “importar datos” . . . . .	63
4.5.9.2. Caso de uso “obtener datos” . . . . .	63
4.6. Clases preliminares del modelo . . . . .	64
4.6.1. Análisis preliminar de clases de la zona de datos . . . . .	64
4.6.1.1. Descripción de las clases . . . . .	66
4.6.2. Análisis preliminar de clases de la zona social . . . . .	68
4.6.2.1. Descripción de las clases . . . . .	70
4.7. Análisis de interfaces de usuario . . . . .	72
4.7.1. Interfaz de la zona de datos . . . . .	72
4.7.2. Interfaz de administración . . . . .	73
4.7.3. Interfaz de la zona social . . . . .	73
4.7.3.1. Vista del blog del Land Portal . . . . .	73
4.7.3.2. Vista de detalle de una entrada del blog . . . . .	73
4.7.3.3. Vista de eventos . . . . .	74
4.7.3.4. Vista de noticias . . . . .	75
4.7.3.5. Vista de debates . . . . .	76
4.7.3.6. Vista de detalle de un debate . . . . .	77
4.7.3.7. Vista de detalle de un evento y de una noticia . . . . .	79
4.7.3.8. Vista de organizaciones . . . . .	79
4.7.3.9. Vista de detalle de una organización . . . . .	80
4.7.3.10. Vista de comunidad . . . . .	80
4.7.3.11. Vista de login . . . . .	81
4.7.3.12. Vista de registro . . . . .	81
4.7.3.13. Vista de búsqueda . . . . .	81
4.8. Especificación del plan de pruebas . . . . .	85
4.8.1. Pruebas unitarias . . . . .	85
4.8.2. Pruebas de integración . . . . .	85
4.8.3. Pruebas de aceptación . . . . .	86
4.8.4. Pruebas de rendimiento . . . . .	86
<b>5. Diseño del sistema</b>	<b>87</b>
5.1. Arquitectura del sistema . . . . .	87
5.1.1. Vista del sistema . . . . .	88
5.1.1.1. Diagrama de componentes . . . . .	88
5.1.1.2. Componentes . . . . .	90
5.1.2. Vista del punto de entrada de datos . . . . .	91
5.1.2.1. Diagrama de componentes . . . . .	92
5.1.2.2. Descripción de los componentes . . . . .	94
5.1.2.3. Relacion entre los componentes . . . . .	94
5.1.2.4. Interfaces y puertos . . . . .	94
5.1.3. Vista de los módulos del gestor de contenidos . . . . .	98
5.1.3.1. Diagrama de componentes . . . . .	98
5.1.3.2. Descripción de los componentes . . . . .	100
5.1.3.3. Relación entre los componentes . . . . .	100
5.1.3.4. Interfaces y puertos . . . . .	101
5.1.4. Vista del módulo <i>landportal uris</i> . . . . .	103
5.1.4.1. Diagrama de componentes . . . . .	103
5.1.4.2. Descripción de los componentes . . . . .	105
5.1.4.3. Relación entre los componentes . . . . .	105
5.1.4.4. Interfaces y puertos . . . . .	105
5.2. Comportamiento del sistema . . . . .	108
5.2.1. Proceso de importación de datos . . . . .	108
5.2.2. Funcionamiento del framework de soporte a visualizaciones . . . . .	109
5.2.3. Llamada a las plantillas de la sección de datos . . . . .	111
5.3. Diseño del modelo de datos . . . . .	112

5.3.1.	Modelo de la zona de datos . . . . .	113
5.3.1.1.	Diagrama del modelo de datos . . . . .	113
5.3.1.2.	Elementos del modelo de datos . . . . .	115
5.3.2.	Modelo de la zona social . . . . .	120
5.3.2.1.	Diagrama del modelo de datos . . . . .	120
5.3.2.2.	Elementos del modelo de datos . . . . .	122
5.3.3.	Sistema de gestión de base de datos . . . . .	122
5.4.	Diseño de la interfaz de usuario . . . . .	123
5.4.1.	Consideraciones generales . . . . .	123
5.4.2.	Vista del blog del Land Portal . . . . .	123
5.4.3.	Vista de detalle de una entrada del blog . . . . .	123
5.4.4.	Vista de eventos . . . . .	124
5.4.5.	Vista de noticias . . . . .	124
5.4.6.	Vista de debates . . . . .	124
5.4.7.	Vista de detalle de un debate . . . . .	125
5.4.8.	Vista de detalle de una noticia y evento . . . . .	125
5.4.9.	Vista de organizaciones . . . . .	125
5.4.10.	Vista de detalle de una organización . . . . .	126
5.4.11.	Vista de login . . . . .	126
5.4.12.	Vista de registro . . . . .	126
5.4.13.	Vista del perfil de un usuario . . . . .	127
5.4.14.	Vista de búsqueda . . . . .	128
5.4.15.	Diseño adaptable . . . . .	128
<b>6.</b>	<b>Implementación del sistema</b>	<b>135</b>
6.1.	Lenguajes de programación utilizados . . . . .	135
6.2.	Herramientas utilizadas . . . . .	136
6.3.	Problemas encontrados . . . . .	137
6.3.1.	Internacionalización de datos . . . . .	137
6.3.2.	Complejidad del modelo de datos . . . . .	137
6.3.3.	Importación de datos . . . . .	138
6.3.4.	Adaptación al CMS . . . . .	138
<b>7.</b>	<b>Desarrollo de las pruebas</b>	<b>139</b>
7.1.	Pruebas de integración . . . . .	139
7.1.1.	Pruebas de la zona de datos . . . . .	139
7.1.1.1.	Desarrollo Dirigido por Pruebas . . . . .	139
7.1.1.2.	Integración continua . . . . .	140
7.1.2.	Pruebas de la zona social . . . . .	140
7.1.2.1.	Subsistema de gestión de usuarios . . . . .	141
7.1.2.2.	Subsistema de gestión de entradas del blog . . . . .	146
7.1.2.3.	Subsistema de gestión de debates . . . . .	149
7.1.2.4.	Subsistema de gestión de eventos . . . . .	152
7.1.2.5.	Subsistema de gestión de noticias . . . . .	154
7.1.2.6.	Subsistema de gestión de organizaciones . . . . .	156
7.1.2.7.	Subsistema de gestión de comentarios . . . . .	158
7.1.2.8.	Subsistema de búsqueda . . . . .	160
7.2.	Pruebas de rendimiento . . . . .	161
7.2.1.	Proceso y ámbito de las mediciones . . . . .	161
7.2.2.	Resultado de las mediciones . . . . .	161
7.2.3.	Ánalisis de los resultados . . . . .	161
7.3.	Pruebas de aceptación . . . . .	164
<b>8.</b>	<b>Planificación del proyecto</b>	<b>166</b>
8.1.	WBS . . . . .	166
8.2.	Planificación inicial . . . . .	169
8.3.	Planificación real . . . . .	170
8.4.	Valoración . . . . .	172

<b>9. Presupuesto</b>	<b>173</b>
9.1. Presupuesto de costes . . . . .	173
9.1.1. Materiales . . . . .	173
9.1.2. Trabajo personal . . . . .	174
9.1.3. Equipamiento e instalaciones . . . . .	174
9.1.4. Otros gastos . . . . .	174
9.1.5. Total . . . . .	175
9.2. Presupuesto del cliente . . . . .	175
9.2.1. Condiciones del presupuesto . . . . .	175
<b>10. Manuales del sistema</b>	<b>177</b>
10.1. Manual de instalación . . . . .	177
10.2. Manual de configuración . . . . .	177
10.3. Manual de usuario . . . . .	177
<b>11. Conclusiones y ampliaciones</b>	<b>178</b>
11.1. Conclusiones . . . . .	178
11.2. Ampliaciones . . . . .	179
11.2.1. Incluir soporte a nuevas visualizaciones en el subsistema de datos . . . . .	179
11.2.2. Incluir soporte a nuevas redes sociales para iniciar sesión en el sistema . . . . .	179
11.2.3. Facilitar el mecanismo de obtención de claves de acceso al API . . . . .	179
11.2.4. Gestión automática de los debates . . . . .	179
11.2.5. Migración del contenido desde el viejo LandPortal . . . . .	180
11.2.6. Futuros proyectos con el IFAD . . . . .	180
<b>12. Anexos</b>	<b>181</b>
12.1. Material entregado . . . . .	181
12.2. Resultados de las mediciones . . . . .	182
12.3. Installation manual . . . . .	184
12.4. Configuration manual . . . . .	185
12.4.1. Enable the <i>LandPortal</i> theme . . . . .	185
12.4.1.1. Configuring the <i>favicon</i> . . . . .	186
12.4.1.2. Configuring <i>home</i> and <i>error</i> pages . . . . .	186
12.4.1.3. Configuring the <i>login</i> redirection . . . . .	186
12.4.2. Import the <i>taxonomy terms</i> . . . . .	187
12.4.3. Configure the content types . . . . .	188
12.4.3.1. Configure the <i>Blog posts</i> . . . . .	189
12.4.3.2. Configure the <i>Debates</i> . . . . .	189
12.4.3.3. Configure the <i>Events</i> . . . . .	190
12.4.3.4. Configure the <i>News</i> . . . . .	190
12.4.3.5. Configure the <i>Organizations</i> . . . . .	190
12.4.4. Configure the search . . . . .	190
12.4.4.1. Connect to the <i>Apache Solr</i> service . . . . .	190
12.4.4.2. Set <i>Apache Solr</i> as the default search provider . . . . .	191
12.4.5. Configure the <i>WYSIWYG</i> editor . . . . .	191
12.5. User manual . . . . .	192
12.5.1. How do I start a new debate? . . . . .	192
12.5.2. How do I close a debate? . . . . .	192
12.5.3. How can I register into the portal? . . . . .	193
12.5.4. How can I moderate new registrations? . . . . .	193
12.5.5. How can I moderate articles? . . . . .	193
12.6. Tutorial: creating a custom view . . . . .	193
12.6.1. Creating the route . . . . .	194
12.6.2. Creating the model . . . . .	195
12.6.3. Creating the template . . . . .	195

# Índice de figuras

1.1. Página principal del portal de datos del Gobierno de Estados Unidos . . . . .	18
1.2. Página principal del portal de datos del Gobierno Británico . . . . .	19
1.3. Una de las visualizaciones ofrecidas por Land Matrix . . . . .	19
1.4. Página principal del antiguo Land Portal . . . . .	20
2.1. Logotipo de Joomla . . . . .	21
2.2. Logotipo de Wordpress . . . . .	22
2.3. Logotipo de Drupal . . . . .	22
2.4. Logotipo de Wesby . . . . .	25
4.1. Diagrama de subsistemas . . . . .	45
4.2. Diagrama de casos de uso del subsistema de gestión de usuarios . . . . .	46
4.3. Diagrama de casos de uso del subsistema de gestión de debates . . . . .	49
4.4. Diagrama de casos de uso del subsistema de gestión de eventos . . . . .	52
4.5. Diagrama de casos de uso del subsistema de gestión de noticias . . . . .	54
4.6. Diagrama de casos de uso del subsistema de gestión de entradas del blog . . . . .	56
4.7. Diagrama de casos de uso del subsistema de gestión de comentarios . . . . .	58
4.8. Diagrama de casos de uso del subsistema de gestión de organizaciones . . . . .	59
4.9. Diagrama de casos de uso del subsistema de búsqueda . . . . .	61
4.10. Diagrama de casos de uso del subsistema de datos . . . . .	62
4.11. Diagrama de clases preliminares de la zona de datos . . . . .	65
4.12. Diagrama de clases preliminares de la zona social . . . . .	69
4.13. <i>Mockup</i> de la vista del blog del Land Portal . . . . .	74
4.14. <i>Mockup</i> de la vista de detalle de una entrada del blog . . . . .	75
4.15. <i>Mockup</i> de la vista de eventos . . . . .	76
4.16. <i>Mockup</i> de la vista de noticias . . . . .	77
4.17. <i>Mockup</i> de la vista de debates . . . . .	78
4.18. <i>Mockup</i> de la vista de detalle de un debate . . . . .	79
4.19. <i>Mockup</i> de la vista de organizaciones . . . . .	80
4.20. <i>Mockup</i> de la vista de detalle de una organización . . . . .	81
4.21. <i>Mockup</i> de la vista de login . . . . .	82
4.22. <i>Mockup</i> de la vista de registro . . . . .	83
4.23. <i>Mockup</i> de la vista de búsqueda . . . . .	84
5.1. Diagrama de componentes del sistema . . . . .	89
5.2. Diagrama de componentes del punto de entrada de datos . . . . .	93
5.3. Diagrama de componentes del gestor de contenidos . . . . .	99
5.4. Diagrama de componentes del módulo “ <i>landportal_uris</i> ” . . . . .	104
5.5. Diagrama de actividad del proceso de importación de datos . . . . .	110
5.6. Diagrama de actividad del framework de soporte a visualizaciones . . . . .	111
5.7. Diagrama de actividad de la llamada a las plantillas en la sección de datos . . . . .	112
5.8. Diseño final del modelo de datos . . . . .	114
5.9. Diseño final del modelo de datos de la zona social . . . . .	121
5.10. Captura de la vista del blog del Land Portal (inglés) . . . . .	124
5.11. Captura de la vista de detalle de una entrada del blog (inglés) . . . . .	125
5.12. Captura de la vista de eventos (inglés) . . . . .	126

5.13. Captura de la vista de noticias (francés) . . . . .	127
5.14. Captura de la vista de debates (inglés) . . . . .	128
5.15. Captura de la vista de detalle de un debate (inglés) . . . . .	129
5.16. Captura de la vista de detalle de una noticia (francés) . . . . .	129
5.17. Captura de la vista de detalle de un evento (inglés) . . . . .	130
5.18. Captura de la vista de organizaciones (español) . . . . .	130
5.19. Captura de la vista de detalle de una organización (español) . . . . .	131
5.20. Captura de la vista de login (inglés) . . . . .	131
5.21. Captura de la vista de registro (español) . . . . .	132
5.22. Captura de la vista del perfil de un usuario (español) . . . . .	133
5.23. Captura de la vista de búsqueda (francés) . . . . .	133
5.24. Captura del diseño adaptable (inglés) . . . . .	134
 7.1. Número de observaciones entrantes frente a tiempo de procesado . . . . .	162
7.2. Número de observaciones por segundo procesadas . . . . .	163
7.3. Número de observaciones frente a memoria durante el procesado . . . . .	163
 8.1. Estructura de desglose del trabajo de la fase de análisis del sistema . . . . .	167
8.2. Estructura de desglose del trabajo de la fase de diseño del sistema . . . . .	167
8.3. Estructura de desglose del trabajo del gestor de contenidos . . . . .	167
8.4. Estructura de desglose del trabajo de la fase desarrollo de la zona social . . . . .	167
8.5. Estructura de desglose del trabajo de la fase desarrollo del Punto de Entrada de Datos . . . . .	168
8.6. Estructura de desglose del trabajo de la fase final del sistema . . . . .	168
8.7. Diagrama de Gantt con la planificación inicial del proyecto . . . . .	169
8.8. Diagrama de Gantt con la planificación real del proyecto . . . . .	171
 9.1. Presupuesto de costes del proyecto . . . . .	173
9.2. Presupuesto del cliente . . . . .	175
 12.1. User's home folder with the script files (underlined in red) . . . . .	184
12.2. Launch system installation command . . . . .	185
12.3. <i>LandPortal</i> 's favicon configuration . . . . .	186
12.4. <i>LandPortal</i> 's home and error pages . . . . .	187
12.5. User redirection configuration . . . . .	187
12.6. Taxonomy import destiny vocabulary selection . . . . .	188
12.7. Taxonomy import type and languages configuration . . . . .	189
12.8. Solr connection configuration result . . . . .	190
12.9. Solr as the default search server configuration . . . . .	191
12.10WYSIWYG text editor configuration . . . . .	192
12.11Snippet of the <i>routes.json</i> file content . . . . .	194
12.12Example of new entry in the <i>routes.json</i> file . . . . .	195
12.13Example of the model class in the <i>helloworld.php</i> file . . . . .	195
12.14Example of the template in the <i>helloworld.mustache</i> file . . . . .	196
12.15Result of the custom view . . . . .	196

# Índice de cuadros

4.1. Tabla de requisitos de la sección de datos. . . . .	37
4.2. Tabla de requisitos de la zona social del portal. . . . .	38
4.3. Tabla de requisitos de la búsqueda. . . . .	41
4.4. Tabla de requisitos del punto de entrada de datos. . . . .	42
4.5. Tabla de requisitos no funcionales. . . . .	43
5.1. Vista del punto de entrada de datos - interfaces del <i>Receiver</i> . . . . .	95
5.2. Vista del punto de entrada de datos - interfaces del <i>Router</i> . . . . .	95
5.3. Vista del punto de entrada de datos - interfaces del <i>textitController</i> . . . . .	95
5.4. Vista del punto de entrada de datos - interfaces comunes a todos los servicios. . . . .	95
5.5. Vista del punto de entrada de datos - interfaces pertenecientes al <i>Parser</i> . . . . .	96
5.6. Vista del punto de entrada de datos - interfaces del <i>servicio de SQL</i> . . . . .	97
5.7. Vista del punto de entrada de datos - interfaces del <i>ORM</i> . . . . .	97
5.8. Vista del punto de entrada de datos - interfaces del <i>servicio de RDF</i> . . . . .	98
5.9. Vista del punto de entrada de datos - interfaces del <i>servicio de CKAN</i> . . . . .	98
5.10. Vista del gestor de contenidos - interfaces del <i>CMS</i> . . . . .	101
5.11. Vista del gestor de contenidos - interfaces del <i>núcleo de Drupal</i> . . . . .	101
5.12. Vista del gestor de contenidos - interfaces de los <i>módulos contribuidos</i> . . . . .	102
5.13. Vista del gestor de contenidos - interfaces del <i>tema</i> . . . . .	102
5.14. Vista del gestor de contenidos - interfaces del módulo <i>Taxonomy dictionary</i> . . . . .	102
5.15. Vista del gestor de contenidos - interfaces del módulo <i>LandPortal URIs</i> . . . . .	103
5.16. Vista del gestor de contenidos - interfaces del módulo <i>LandbBook nodes access</i> . . . . .	103
5.17. Vista del gestor de contenidos - interfaces del módulo <i>LandPortal API auth</i> . . . . .	103
5.18. Vista del módulo <i>LandPortal URIs</i> - interfaces del módulo <i>LandPortal URIs</i> . . . . .	106
5.19. Vista del módulo <i>LandPortal URIs</i> - interfaces del componente “ <i>module</i> ”. . . . .	106
5.20. Vista del módulo <i>LandPortal URIs</i> - interfaces del componente “ <i>DatabaseHelper</i> ”. . . . .	106
5.21. Vista del módulo <i>LandPortal URIs</i> - interfaces del componente “ <i>CacheHelper</i> ”. . . . .	107
5.22. Vista del módulo <i>LandPortal URIs</i> - interfaces del componente “ <i>Model</i> ”. . . . .	107
5.23. Vista del módulo <i>LandPortal URIs</i> - interfaces del componente “ <i>Ajax</i> ”. . . . .	108
7.1. Casos de prueba del subsistema de gestión de usuarios . . . . .	142
7.2. Casos de prueba del subsistema de gestión de entradas del blog . . . . .	146
7.3. Casos de prueba del subsistema de gestión de debates . . . . .	149
7.4. Casos de prueba del subsistema de gestión de eventos . . . . .	152
7.5. Casos de prueba del subsistema de gestión de noticias . . . . .	154
7.6. Casos de prueba del subsistema de gestión de organizaciones . . . . .	156
7.7. Casos de prueba del subsistema de gestión de comentarios . . . . .	158
7.8. Casos de prueba del subsistema de búsqueda . . . . .	160
12.1. Mediciones de rendimiento del Punto de Entrada de Datos . . . . .	183

# Capítulo 1

## Introducción

### 1.1. Justificación del Proyecto

El presente proyecto tiene como objetivo la construcción de un backend para un portal de datos sobre la tierra. Este proyecto forma parte a su vez del proyecto *Rebuilding IFAD's LandPortal RFQ/2013/016/SC* desarrollado por el grupo de investigación Web Semantics Oviedo<sup>1</sup> y la empresa SB Consulting<sup>2</sup> y que cuenta como cliente con el Fondo Internacional para el Desarrollo Agrícola<sup>3</sup>, perteneciente a la Organización de las Naciones Unidas<sup>4</sup>.

El backend que se desarrollará en este proyecto será por tanto utilizado en la renovación del Land Portal<sup>5</sup>.

El Land Portal tiene como objetivo convertirse en el sitio líder a la hora de buscar información y recursos sobre todos los temas relacionados con la tierra. Tal y como Tim Davies explica en «Land Portal Strategy 2013 – 2016», página 5:

*“La visión del portal es mejorar la gestión de la tierra para beneficiar a aquellos más vulnerables y con menos derechos, a través de la transmisión de información y conocimiento.”*

Para alcanzar esta visión, el Land Portal pretende aumentar el número de países, regiones, indicadores y tópicos sobre los que almacena información, además de mejorar la visualización y reutilización de los nuevos datos y los datos ya existentes.

Por todo ello, y con el objetivo de servir como ejemplo en la transparencia de la información, el Land Portal se encuentra en una fase de crecimiento y expansión, pretendiendo convertirse en un portal de datos abiertos y contribuir a un desarrollo ágil y visible públicamente.

### 1.2. Objetivos del proyecto

El principal objetivo que se pretende cumplir en este proyecto es la construcción de un portal de datos que permita de centralizar, organizar y buscar información relacionada con la gestión y el uso de la tierra que de otra forma estaría fragmentada e inaccesible.

Dicha información procede de diversas fuentes de datos pertenecientes a gobiernos, instituciones académicas, organizaciones internacionales y organizaciones no gubernamentales como pueden ser:

---

<sup>1</sup>WESO - <http://www.weso.es/>

<sup>2</sup>SBC4D - <http://www.sbc4d.com/>

<sup>3</sup>IFAD - <http://www.ifad.org/>

<sup>4</sup>ONU - <http://www.un.org/es/>

<sup>5</sup><http://landportal.info/>

- el Banco Mundial (*WorldBank*)
- la Organización de las Naciones Unidas para la Alimentación y la Agricultura (*FAO*)
- la Organización Mundial de la Salud (*WHO*)
- el Instituto Internacional de Investigación sobre Políticas Alimentarias (*IFPRI*)
- la Organización para la Cooperación y el Desarrollo Económicos (*OECD*)

Puesto que los datos proceden de fuentes tan diversas, es importante para el portal centralizar y unificar el proceso de inserción de nuevos datos, con el fin de facilitar la colaboración de entidades externas que quieran ver sus datos reflejados en el portal y, al mismo tiempo, asegurar la calidad de los mismos, haciendo que cumplan unos estándares mínimos de calidad.

Un segundo objetivo de gran importancia para este proyecto es fomentar el diálogo, el intercambio de información y la participación de los usuarios de una forma que permita complementar, combinar y enriquecer la información presentada desde las fuentes de datos oficiales. Para ello es importante contar con un lugar en el que los usuarios puedan debatir y compartir información de una forma sencilla.

En relación con el interés por fomentar la participación de los usuarios en el portal también se pretende simplificar el método de acceso al mismo, de forma que se consiga integrar en un único punto el acceso a todas las partes del portal y al registro e inicio de sesión de los usuarios.

Por último, y como no puede ser de otra forma en un portal destinado a almacenar y presentar datos sobre las distintas regiones del mundo, es de especial interés establecer un sistema de internacionalización que permita a los usuarios visualizar la información en el idioma que prefieran y, de esta forma democratizar el acceso a toda la información expuesta.

Estos objetivos se desarrollarán en mayor detalle en el capítulo 4.

## 1.3. Estudio de la situación actual

En la actualidad, los portales de datos son una tendencia que se hace cada vez más presente en el entorno de Internet. Son varios los gobiernos y organizaciones que cuentan con un portal de datos para facilitar el acceso de los ciudadanos a todo tipo de información.

En esta sección se enumerarán algunos portales de datos abiertos que han servido como referente e inspiración a la hora de construir el nuevo Land Portal, al mismo tiempo que se repasarán las referencias y similitudes con cada uno de ellos.

### 1.3.1. Portal de datos del Gobierno de Estados Unidos

El portal de datos del Gobierno de Estados Unidos<sup>6</sup> es un referente en cuanto a la construcción de este tipo de portales. Fue publicado por primera vez en mayo de 2009 y recibió un rediseño completo en el aniversario de su primer año (el 21 de mayo de 2010). Actualmente es uno de los portales de datos con mayor cantidad de información, puesto que contiene casi 250000 conjuntos de datos. La figura 1.1 muestra la página principal del portal de datos del Gobierno de Estados Unidos.

El portal utiliza CKAN<sup>7</sup> para almacenar todos los conjuntos de datos y Drupal<sup>8</sup> como gestor de contenidos.

---

<sup>6</sup><http://www.data.gov/>

<sup>7</sup><http://ckan.org/>

<sup>8</sup><https://drupal.org/>

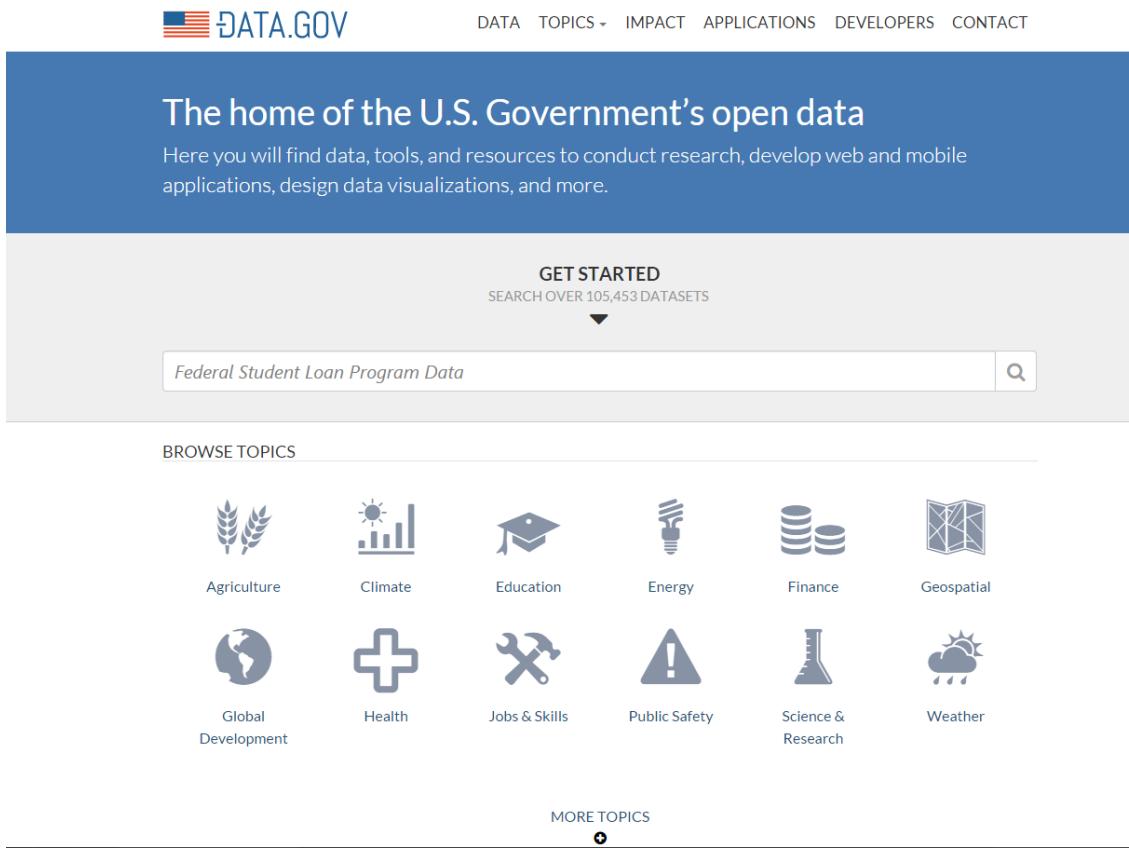


Figura 1.1: Página principal del portal de datos del Gobierno de Estados Unidos

Un apartado en el que el nuevo Land Portal intenta mejorar a este portal es la visualización de datos. El nuevo Land Portal ofrecerá visualizaciones destinadas a facilitar la interpretación de los datos por parte de los usuarios.

### 1.3.2. Portal de datos del Gobierno Británico

El portal de datos del Gobierno Británico<sup>9</sup>, al igual que el portal de datos del Gobierno de Estados Unidos, es también uno de los referentes mundiales en cuanto a portales de datos abiertos. La figura 1.2 muestra la página principal de este portal de datos.

El portal se hizo público en enero de 2010 y actualmente contiene más de 9000 conjuntos de datos procedentes de varios departamentos del Gobierno. Todos los conjuntos de datos se encuentran disponibles de forma gratuita para uso privado y comercial siempre que se atribuya su creación al Gobierno Británico.

Al igual que el portal de datos del Gobierno de Estados Unidos, este portal también utiliza CKAN y Drupal como gestor de datos y de contenido respectivamente. Como se explicará posteriormente en esta misma sección, el nuevo Land Portal también utiliza un sistema similar, debido a su probada estabilidad en sistemas similares. El aspecto del nuevo Land portal también se ha inspirado en el diseño claro y simple de este portal.

<sup>9</sup><http://data.gov.uk/>



Figura 1.2: Página principal del portal de datos del Gobierno Británico

### 1.3.3. Land Matrix

Land Matrix<sup>10</sup> es una iniciativa independiente que pretende seguir y presentar los tratados, intercambios e inversiones sobre la tierra. La primera versión beta del portal se hizo pública en abril de 2012, y en junio de 2013 el portal fue relanzado con un nuevo diseño y un catálogo de datos actualizado.

De forma similar a como sucederá en el nuevo Land Portal, los datos de Land Matrix proceden de investigadores, activistas, agencias gubernamentales, periodistas y compañías externas.

Un punto de innovación en Land Matrix es su capacidad de ofrecer a los usuarios atractivas visualizaciones de los datos que alberga (véase la figura 1.3). Esta característica, así como su soporte para la inclusión de datos de terceros en dichas visualizaciones, también formará parte del nuevo Land Portal.



Figura 1.3: Una de las visualizaciones ofrecidas por Land Matrix

<sup>10</sup><http://www.landmatrix.org/>

### 1.3.4. Antiguo Land Portal

El Land Portal<sup>11</sup> pertenece al Fondo Internacional para el Desarrollo Agrícola, que forma parte de la ONU. Fue creado en marzo de 2011 y actualmente cuenta con casi 1000 usuarios registrados y un total de 70 organizaciones diferentes. En el año 2012 el portal tuvo unos 70000 visitantes únicos, con cerca de 10000 visitas mensuales<sup>12</sup>. La figura 1.4 muestra la página de inicio del antiguo Land Portal.



Figura 1.4: Página principal del antiguo Land Portal

A pesar de no ser un portal de datos abiertos como tal, el viejo Land Portal ha sido la principal inspiración a la hora de realizar este proyecto. El nuevo portal pretende reunir y mejorar las características de los portales de datos del Gobierno Británico y del Gobierno de Estados Unidos y, al mismo tiempo, mantener el espíritu original de colaboración e intercambio de conocimientos del viejo Land Portal. Los principales puntos en los que este proyecto pretende mejorar lo ofrecido por el viejo Land Portal son los siguientes:

- Ofrecer una interfaz más amigable, en línea con la sencillez y vistosidad de los portales de datos del Gobierno Británico y el Gobierno de Estados Unidos.
- Ofrecer acceso a los conjuntos de datos procedentes de diferentes organizaciones, tanto de forma visual como a través de un API para desarrolladores y un catálogo de datos con capacidad de negociación de contenido.
- Fomentar la participación de todos los miembros de la comunidad aportando nueva información y debatiendo sobre la ya existente.

<sup>11</sup><http://landportal.info/>

<sup>12</sup>Esta información puede consultarse en [Dav13, página 3]

## Capítulo 2

# Evaluación de alternativas

En este capítulo se analizarán las posibles alternativas a los distintos componentes del sistema y posteriormente se seleccionarán las que se utilizarán para el desarrollo del sistema final.

### 2.1. Alternativas evaluadas

#### 2.1.1. Gestor de contenidos

Un gestor de contenidos (*CMS*) permite publicar, editar y modificar de forma sencilla los contenidos de una página web. La mayoría de gestores de contenidos también se encargan de la gestión de los usuarios, los roles de usuario y permisos de cada uno de ellos.

Hay multitud de gestores de contenido disponibles en el mercado. En esta sección se van a analizarán varios de ellos y posteriormente se explicará cual se ha escogido para su utilización en este proyecto.

##### 2.1.1.1. Joomla

Joomla<sup>1</sup> es un gestor de contenido de código abierto, con licencia GPL y escrito en PHP. Fue creado en 2005 como un fork de otro gestor de contenido llamado *Mambo*<sup>2</sup>.

Joomla es utilizado en sitios web de gran relevancia, como la página web de la Universidad de Harvard<sup>3</sup>.



Figura 2.1: Logotipo de Joomla

Las principales ventajas de Joomla respecto a otros gestores de contenido son su sencillez y su capacidad de extensión.

Joomla está diseñado utilizando técnicas de programación orientada a objetos y aplicando varios

---

<sup>1</sup><http://www.joomla.org/>

<sup>2</sup><http://www.mamboserver.com/>

<sup>3</sup><http://gsas.harvard.edu/>

patrones de diseño de software, lo que hace que su código esté relativamente bien formado. Por otra parte Joomla soporta cinco tipos diferentes de extensiones (componentes, plugins, plantillas, módulos e idiomas). Cada uno de estos tipos de extensión tiene un comportamiento y una finalidad diferente, lo que permite que sea posible adaptar el funcionamiento del gestor de contenidos a cada necesidad particular.

### 2.1.1.2. Wordpress

Wordpress<sup>4</sup> es un gestor de contenido de código abierto, con licencia GPLv2 y escrito en PHP. Fue creado en mayo de 2003 como un fork de otro gestor de contenido llamado *b2/cafelog*. En la actualidad Wordpress se utiliza en más de 68 millones de sitios web, entre ellos el sitio web del New York Times<sup>5</sup>, el sitio web de CNN<sup>6</sup> o el sitio web de Forbes<sup>7</sup>.



Figura 2.2: Logotipo de Wordpress

La principal ventaja de Wordpress respecto a otros gestores de contenido es su simplicidad, puesto que está orientado principalmente a la construcción de sitios orientados al blogging o a las noticias.

### 2.1.1.3. Drupal

Drupal<sup>8</sup> es un gestor de contenido de código abierto creado en el año 2001. Al igual que Joomla y Wordpress está escrito en PHP y cuenta con licencia GPLv2. Como se ha mencionado anteriormente en la sección “Estudio de la situación actual” perteneciente al capítulo 1, Drupal se utiliza en el portal de datos del Gobierno Británico<sup>9</sup> y en el portal de datos del Gobierno de Estados Unidos<sup>10</sup>.



Figura 2.3: Logotipo de Drupal

La principal ventaja de Drupal es su flexibilidad para extender y modificar su funcionamiento. En la actualidad cuenta con más de 15000 módulos disponibles. Su sistema de *hooks* permite crear módulos que responden a eventos llamados y ejecutados de forma automática por el *core* de Drupal.

---

<sup>4</sup><https://wordpress.org/>

<sup>5</sup><http://www.nytimes.com/>

<sup>6</sup><http://edition.cnn.com/>

<sup>7</sup><http://www.forbes.com/>

<sup>8</sup><https://drupal.org/>

<sup>9</sup><http://data.gov.uk/>

<sup>10</sup><http://www.data.gov/>

### 2.1.2. Catálogo de datos

Un catálogo de datos permite almacenar y organizar bajo una estructura común catálogos de datos procedentes de diversas fuentes y que pueden encontrarse en distintos formatos.

#### 2.1.2.1. CKAN

CKAN<sup>11</sup> (*Comprehensive Knowledge Archive Network*) es una plataforma de código abierto para la construcción de portales de datos y creada por la OKFN (*Open Knowledge Foundation*)<sup>12</sup> que permite publicar, buscar y organizar catálogos de datos. CKAN está escrito en Python<sup>13</sup> y utiliza PostgreSQL<sup>14</sup> como base de datos.

El funcionamiento de CKAN consiste en almacenar los catálogos de datos junto con diversos metadatos, que posteriormente son accesibles y modificables desde una interfaz web amigable para los usuarios. Además de la interfaz web CKAN también ofrece una API que permite interactuar con otras aplicaciones y servicios de terceros.

Como se ha mencionado anteriormente en la sección “Estudio de la situación actual” perteneciente al capítulo 1, CKAN se utiliza en varios portales de datos de gran envergadura, como pueden ser el portal de datos del Gobierno Británico o el portal de datos del Gobierno de Estados Unidos.

#### 2.1.2.2. DKAN

DKAN<sup>15</sup> es una plataforma basada en CKAN y Drupal para facilitar la publicación de datos. A diferencia de CKAN, DKAN está escrito utilizando el lenguaje de programación PHP.

La principal ventaja de DKAN es la estrecha integración entre el catálogo de datos (CKAN) y el gestor de contenidos (Drupal). Esta integración entre los dos componentes permite aprovechar las mejores características de cada uno de ellos. Por otra parte, esta integración también permite desplegar el catálogo de datos de una forma simple sobre una instalación de Drupal ya existente.

Un punto en contra de DKAN es su falta de madurez. DKAN fue creado en 2012 y recientemente ha alcanzado la versión 1.0. A pesar de todo, ha sido utilizado en algunos proyectos como el portal de datos de la Ciudad de Colonia y el portal de datos del Gobierno de Puerto Rico.

#### 2.1.2.3. Herramienta creada especialmente para la ocasión

La funcionalidad de un catálogo de datos podría ser implementada por una herramienta creada especialmente para este proyecto e implementada como un módulo del gestor de contenidos.

La principal ventaja de esta aproximación es que, dado que la herramienta se crearía especialmente para este proyecto, cumpliría totalmente con las necesidades del mismo. La principal desventaja radica en el esfuerzo requerido para implementar una herramienta de tal calibre, esfuerzo que ya viene solucionado por parte de otras herramientas existentes y de probada estabilidad.

### 2.1.3. Servidor semántico

Puesto que este proyecto consiste en la creación de un portal de datos enlazados abiertos, es necesario incluir un componente que se encargue de aportar la parte de datos enlazados. Estos

<sup>11</sup><http://ckan.org/>

<sup>12</sup><https://okfn.org/>

<sup>13</sup><https://www.python.org/>

<sup>14</sup><http://www.postgresql.org/>

<sup>15</sup><http://nucivic.com/dkan/>

componentes suelen llamarse servidores semánticos o *triple-stores*, puesto que almacenan los datos en formato RDF<sup>16</sup>, que modela los datos en forma de triplets.

### 2.1.3.1. Virtuoso Universal Server / Virtuoso OpenLink

Virtuoso Universal Server<sup>17</sup> es un motor de base de datos con una arquitectura híbrida que le permite ofrecer diferentes funcionalidades, que tradicionalmente han sido realizadas por diferentes productos, en un mismo componente.

Virtuoso fue creado en 1998 de la unión del middleware de acceso a datos *OpenLink* y el sistema de gestión de bases de datos relacionales *KUBL*.

Además de la capacidad para ofrecer diferentes servicios en un mismo componente, otra gran ventaja de Virtuoso es su estabilidad y rendimiento. Las principales ventajas de Virtuoso son su capacidad para ofrecer diferentes servicios desde un mismo componente y su estabilidad y rendimiento. Además Virtuoso proporciona un punto de acceso SPARQL<sup>18</sup> para consultar los datos que almacena.

Como se indica en [W3C05], la versión 6.1 de Virtuoso ha llegado a servir 15.4 billones de triplets simultáneamente (incluyendo el catálogo completo del portal de datos del Gobierno de Estados Unidos).

Virtuoso es un producto propietario, aunque tiene una versión libre con licencia GPLv2 que recibe el nombre de Virtuoso OpenLink<sup>19</sup>. Virtuoso y Virtuoso OpenLink es utilizado activamente en varios portales de datos, entre ellos el portal de datos de la Web Foundation<sup>20</sup> y la DBpedia<sup>21</sup>.

### 2.1.3.2. Stardog

Stardog<sup>22</sup> es un sistema de base de datos RDF escrito en el lenguaje de programación Java. Stardog es un producto comercial, aunque cuenta con una versión gratuita con características limitadas.

La principal ventaja de Stardog frente al resto de servidores semánticos es su capacidad para realizar inferencias sobre los datos que almacena, así como su soporte a la especificación OWL2<sup>23</sup>.

Como se menciona en [W3C05], la versión 2.1 de Stardog soporta hasta 50 billones de triplets almacenadas simultáneamente.

### 2.1.3.3. 4store

4store<sup>24</sup> es un sistema de gestión de bases de datos y un motor de consultas que almacena datos en formato RDF. 4store está escrito en ANSI C99 y cuenta con licencia GPLv3.

Las principales ventajas de 4store son su rendimiento, su escalabilidad y su estabilidad. 4store lleva siendo usado en producción en Garlik<sup>25</sup> durante 3 años. Como se puede ver en [W3C05], 4store soporta hasta 15 billones de triplets almacenadas simultáneamente.

---

<sup>16</sup><http://www.w3.org/RDF/>

<sup>17</sup><http://virtuoso.openlinksw.com/>

<sup>18</sup>SPARQL es un lenguaje para realizar consultas en grafos RDF de forma similar a cómo SQL sirve para realizar consultas en bases de datos relacionales.

<sup>19</sup><http://virtuoso.openlinksw.com/dataspace/doc/dav/wiki/Main/>

<sup>20</sup>El portal de datos de la Web Foundation ha sido también desarrollado por el grupo de investigación WESO y está disponible en <http://data.webfoundation.org/>

<sup>21</sup><http://wiki.dbpedia.org/>

<sup>22</sup><http://www.stardog.com/>

<sup>23</sup>OWL2 es un estándar publicado en 2008 por el W3C con el objetivo de construir un modelo de marcas basado en RDF y codificado en XML.

<sup>24</sup><http://4store.org/>

<sup>25</sup><http://www.garlik.com/>

Por otra parte, una posible desventaja de 4store es el limitado número de características que ofrece. 4store únicamente proporciona almacenamiento RDF y consultas SPARQL.

### 2.1.4. Visualizador de datos enlazados

Un visualizador de datos enlazados accede a los datos que se almacenan en el servidor semántico en formato RDF para mostrarlos de una forma que atractiva para el usuario y permitir navegar a través de ellos sin necesidad de realizar complejas consultas SPARQL.

#### 2.1.4.1. Pubby

Pubby<sup>26</sup> es un visualizador de datos enlazados escrito en el lenguaje Java. Pubby fue creado por Richard Cyganiak y cuenta con una licencia Apache v2.

La principal ventaja de Pubby es su facilidad de configuración y su demostrada estabilidad. Tal y como Chris Bizer explica en [Biz], originalmente Pubby fue utilizado como *frontend* de Virtuoso en DBpedia. Posteriormente Pubby fue sustituido por consumidores especializados en HTML y RDF por cuestiones de rendimiento.

La principal desventaja de Pubby es su estado de abandono. Al momento de realizar este proyecto, la última actualización de Pubby data de enero de 2011.

#### 2.1.4.2. Wesby

Wesby<sup>27</sup> es un visualizador de datos enlazados escrito en el lenguaje Scala. Es un desarrollo propio de WESO<sup>28</sup> que tiene por objetivo establecerse como alternativa que mejore lo ofrecido por Pubby.



Figura 2.4: Logotipo de Wesby

La principal ventaja de Wesby sobre Pubby es su apariencia y su rendimiento. Wesby ofrece una interfaz limpia y adaptable a diferentes dispositivos. Además, una característica ofrecida por Wesby es el soporte para la creación de visualizaciones para diferentes tipos de nodos dentro del grafo RDF. Otra ventaja de Wesby es que este construye automáticamente el esquema de URIs a partir del grafo de datos presente en el servidor semántico al que se conecta.

Una posible desventaja de Wesby respecto a Pubby es su novedad. Wesby es un proyecto relativamente nuevo, y únicamente se ha utilizado en el portal de datos de la Web Foundation, por lo que su estabilidad no está totalmente probada.

<sup>26</sup><http://wifo5-03.informatik.uni-mannheim.de/pubby/>

<sup>27</sup><https://github.com/weso/wesby>

<sup>28</sup><http://www.weso.es/>

### 2.1.5. Framework de diseño web

Un framework de diseño web facilita la tarea de diseñar la interfaz de la web del portal de forma que resulte atractiva a los usuarios.

#### 2.1.5.1. Twitter Bootstrap

Twitter Bootstrap<sup>29</sup> es un framework de diseño web desarrollado por Mark Otto y Jacob Thornton. Bootstrap vió la luz como proyecto de código abierto en el año 2011.

Una gran ventaja de Bootstrap respecto a otros frameworks de diseño web es el gran número de elementos prediseñados con el que cuenta. Otra característica interesante de Bootstrap es que facilita el desarrollo adaptable a diferentes dispositivos al utilizar una organización en filas y columnas de ancho variable.

Bootstrap es utilizado en multitud de sitios web, por lo que cuenta con una comunidad muy amplia y existe una gran cantidad de temas y documentación disponible.

Una desventaja de Bootstrap y, en general, de cualquier framework de diseño web es que obliga a realizar el desarrollo de una forma determinada para poder aprovechar sus características, desviarse del camino establecido puede provocar dificultades durante el desarrollo.

#### 2.1.5.2. ZURB Foundation

ZURB Foundation<sup>30</sup> es un framework de diseño web desarrollado por ZURB en 2011.

La principal ventaja de Foundation respecto a otros frameworks de diseño web es que permite diseñar la web orientada principalmente a dispositivos móviles y adaptada posteriormente a pantallas de un tamaño más grande.

Una característica de Foundation que produce al mismo tiempo ventajas y desventajas es que Foundation cuenta con menos elementos prediseñados que Bootstrap, algo que obliga al desarrollador trabajar más si quiere utilizar componentes no disponibles en el framework pero, al mismo tiempo, permite una mayor flexibilidad en los diseños.

#### 2.1.5.3. Desarrollo desde cero

Una tercera alternativa es realizar el diseño del portal desde cero sin utilizar ningún framework.

La principal ventaja de esta solución es la total flexibilidad para desarrollar y dar la apariencia que se deseé sin estar sujetos a las ataduras de un framework.

La principal desventaja es la cantidad de trabajo necesario para implementar y mantener una apariencia coherente a lo largo de todo el portal, además de el esfuerzo necesario para conseguir una interfaz adaptable a diferentes dispositivos.

### 2.1.6. Motor de búsqueda

Un motor de búsqueda se encarga de catalogar e indexar los contenidos de un portal, construyendo un índice inverso que permita a los usuarios encontrar la información que necesiten.

<sup>29</sup><http://getbootstrap.com/>

<sup>30</sup><http://foundation.zurb.com/>

### 2.1.6.1. Apache Solr

Apache Solr<sup>31</sup> es un motor de búsqueda desarrollado por la Apache Software Foundation<sup>32</sup>. Fue creado en el año 2004 utilizando el lenguaje Java y tiene una licencia Apache v2.

Solr utiliza la librería de búsqueda Lucene<sup>33</sup>, desarrollada también por la Apache Software Foundation. En el año 2010 los equipos de desarrollo encargados de Solr y Lucene se unieron y desde entonces es común referirse a ambos productos como *Lucene/Solr*.

Un punto a favor de Solr es su madurez y su amplia comunidad de usuarios y desarrolladores, lo que permite encontrar plugins para integrarlo en multitud de productos. Solr es utilizado en multitud de proyectos de gran envergadura como Netflix<sup>34</sup>, Apple<sup>35</sup>, MTV<sup>36</sup> o digg<sup>37</sup>. Otra característica positiva de Solr es la existencia de una API de búsqueda por HTTP. La API HTTP de Solr recibe peticiones GET y permite escoger entre varios formatos de respuesta como XML o JSON.

Un punto que puede resultar negativo de Solr frente a otros motores de búsqueda es la necesidad de definir un esquema de datos. El esquema es un fichero en formato XML que define qué estructura tendrá el índice. A pesar de esto, es posible definir elementos dinámicos que se creen bajo demanda sin tener que figurar en el esquema.

### 2.1.6.2. Elasticsearch

Elasticsearch<sup>38</sup> es un motor de búsqueda desarrollado por la Organización Elasticsearch<sup>39</sup>. Fue creado en 2010 por Shay Banon utilizando el lenguaje Java. Al igual que Solr, Elasticsearch también cuenta con una licencia Apache v2 y utiliza la librería de búsqueda Lucene.

Las principales características a favor de Elasticsearch son su capacidad para construir sistemas de búsqueda distribuidos y su arquitectura *schemaless*, lo que permite enviar documentos para que sean indexados sin necesidad de definir un esquema previamente.

El punto débil de Elasticsearch es su novedad. A pesar de ser utilizado en portales como Quora<sup>40</sup> o GitHub<sup>41</sup> no cuenta con la madurez y amplia comunidad de productos más maduros como Solr.

### 2.1.6.3. Buscador propio del gestor de contenidos

Los tres gestores de contenidos que se han analizado anteriormente cuentan con un buscador propio integrado. La principal ventaja de este buscador es la completa integración con los contenidos que forman parte del CMS. Puesto que el buscador ya forma parte del CMS, normalmente no es necesario realizar ninguna tarea de programación ni configuración para activarlo.

La gran desventaja de esta solución es la imposibilidad de indexar y buscar todos aquellos contenidos que no formen parte del CMS, algo que en este proyecto esto haría imposible buscar los contenidos pertenecientes a la sección de datos.

<sup>31</sup><http://lucene.apache.org/solr/>

<sup>32</sup><http://www.apache.org/foundation/>

<sup>33</sup><http://lucene.apache.org/>

<sup>34</sup><https://www.netflix.com/global>

<sup>35</sup><http://www.apple.com/>

<sup>36</sup><http://www.mtv.com/>

<sup>37</sup><http://digg.com/>

<sup>38</sup><http://www.elasticsearch.org/>

<sup>39</sup><https://github.com/elasticsearch>

<sup>40</sup><https://www.quora.com/>

<sup>41</sup><https://github.com/>

## 2.2. Alternativas elegidas

En esta sección se explicará qué alternativas de las anteriormente descritas se han seleccionado y las razones por las que se ha llevado a cabo dicha selección.

### 2.2.1. Gestor de contenidos

De los tres gestores de contenidos analizados para este proyecto, se ha seleccionado Drupal debido principalmente su madurez y flexibilidad.

A pesar de que los tres gestores de contenidos analizados se utilizan actualmente en multitud de proyectos, Drupal forma parte de varios proyectos con objetivos similares a este, como los portales de datos del Gobierno Británico y del Gobierno de Estados Unidos ya mencionados anteriormente. Por otra parte, una característica a tener muy en cuenta es la cantidad de módulos disponibles para extender Drupal, que en este momento supera los 15000.

Un punto importante en esta decisión es que, como Larry Garfield explica en [Gar06] Drupal utiliza el patrón arquitectónico PAC<sup>42</sup> para organizar su estructura y funcionamiento. El patrón PAC es menos conocido y utilizado que el patrón MVC<sup>43</sup>, lo que puede reflejarse en una mayor dificultad a la hora de realizar la implementación.

### 2.2.2. Catálogo de datos

Entre las tres alternativas mencionadas anteriormente, se ha seleccionado CKAN como catálogo de datos para este proyecto.

Al igual que Drupal, CKAN se utiliza en varios portales de datos con objetivos similares al que se pretende construir en este proyecto. CKAN permitirá ofrecer una vista completa de los conjuntos de datos incluidos en Land Portal.

La alternativa de utilizar una herramienta implementada especialmente para el proyecto se ha descartado por la gran complejidad que conlleva tanto la propia implementación como la integración con el gestor de contenidos y la creación de una interfaz de acceso a los datos.

### 2.2.3. Servidor semántico

Se ha seleccionado Virtuoso como servidor semántico para el nuevo Land Portal. Esta selección viene motivada por varios factores que se explicarán a continuación.

Como se ha explicado anteriormente, Virtuoso ofrece un punto de acceso SPARQL, algo que se considera indispensable en cualquier portal de datos abiertos y enlazados. Además, a diferencia de otras alternativas como Stardog, la versión libre de Virtuoso no tiene ninguna limitación en el número de conexiones que puede recibir ni en el uso de CPU, la única limitación de la versión libre es su capacidad para funcionar como una única instancia, pero esto no debería ser un problema dada la cantidad de datos que se pretende almacenar en el sistema.

Por último, al contrario que 4store, Virtuoso ofrece un API que permite almacenar y extraer datos, lo que puede resultar de utilidad a la hora de realizar el proceso de importación y enriquecimiento de datos.

<sup>42</sup>Presentation Abstraction Control - <http://en.wikipedia.org/wiki/Presentation-abstraction-control>

<sup>43</sup>Model View Controller - <http://martinfowler.com/eaaDev/uiArchs.html>

### 2.2.4. Visualizador de datos enlazados

Entre las dos alternativas para el visualizador de datos enlazados se ha seleccionado Wesby.

Esta selección viene motivada principalmente por ser Wesby un desarrollo propio del grupo de investigación WESO en el que el propio autor de este Proyecto Fin de Grado ha participado, además de por ofrecer una interfaz moderna, amigable y adaptable a diferentes dispositivos que permitirá que no desentoné respecto a las demás partes del nuevo Land Portal.

### 2.2.5. Framework de diseño web

Como framework de diseño web se ha seleccionado Bootstrap.

La principal razón para seleccionar Bootstrap frente a Foundation es que Bootstrap cuenta con una mayor cantidad de usuarios, lo que produce una mayor comunidad y permite encontrar recursos como documentación, temas, etc. más fácilmente.

El desarrollo de la apariencia del portal sin utilizar un framework de diseño web ha sido descartado debido al gran esfuerzo que requeriría en implementar un diseño moderno, vistoso, coherente y adaptativo desde cero.

### 2.2.6. Motor de búsqueda

La decisión del motor de búsqueda ha sido quizás de las más complejas. La decisión final se ha decantado en favor de Apache Solr.

A pesar de que tanto Solr como Elasticsearch son utilizados en varios proyectos importantes, la mayor madurez de Solr hace más sencillo encontrar plugins e información. En relación con la decisión del gestor de contenido, Solr cuenta con un plugin para Drupal<sup>44</sup> que lleva 7 años en desarrollo activo, por lo que su madurez y estabilidad quedan fuera de toda duda.

Debido también a la cantidad de datos con la que contará el portal tampoco parece necesario recurrir a las altas capacidades de búsqueda distribuida ofrecidas por Elasticsearch.

---

<sup>44</sup><https://drupal.org/project/apachesolr>

# Capítulo 3

## Conceptos teóricos

En este capítulo se definirán los conceptos teóricos necesarios para comprender correctamente la finalidad del sistema en construcción.

### 3.1. Datos abiertos

El concepto de datos abiertos u *Open Data* es definido por la Open Knowledge Foundation en [Fou] de la siguiente forma:

*“A piece of data or content is open if anyone is free to use, reuse, and redistribute it — subject only, at most, to the requirement to attribute and/or share-alike.”*

De la anterior definición pueden destacarse tres pilares fundamentales que marcan la diferencia entre los datos abiertos y el resto de datos, dichos puntos clave se explicarán a continuación:

- Acceso y disponibilidad. Los datos deben estar disponibles en su totalidad con un coste razonable o, preferiblemente, de forma gratuita a través de Internet. Es también importante que el formato en el que se publican los datos sea libre y modificable.
- Reutilización y redistribución. La licencia bajo la que se publican los datos no debe restringir la redistribución de los mismos ni exigir pagos o cuotas por dicha redistribución. Además, la licencia debe permitir modificar los datos e incluso cruzarlos con datos provenientes de otras fuentes.
- Participación universal. Los datos no deben poner trabas para ser accedidos por ninguna persona ni grupo de personas ni deben restringir el uso de los mismos a un ámbito de trabajo específico.

Cualquier conjunto de datos puede ser considerado un conjunto de datos abiertos si cumple con la definición anterior, aunque las principales fuentes de datos abiertos generalmente provienen de fuentes científicas o gubernamentales. En el año 2004 los ministros de ciencia de todas las naciones pertenecientes a la OECD<sup>1</sup> firmaron una declaración en la que se aboga por hacer públicos toda la información científica financiada con fondos públicos.

En cuanto a los datos abiertos procedentes de fuentes gubernamentales, como ya se ha mencionado en el capítulo 1, la tendencia de los gobiernos de ofrecer datos en forma abierta es cada vez mayor tal y como evidencian los múltiples portales de datos propiedad de la administración pública que han surgido recientemente. Algunos ejemplos son: el portal de datos del Gobierno de

<sup>1</sup>La OECD (Organización para la Cooperación y el Desarrollo Económico) es una de las organizaciones que aporta los conjuntos de datos con los que se trabaja en este proyecto.

España<sup>2</sup>, el portal de datos del Gobierno de Reino Unido<sup>3</sup> o el portal de datos del Gobierno de Estados Unidos<sup>4</sup>.

## 3.2. Datos enlazados

El concepto de datos enlazados o *Linked Data* fue acuñado por Tim Berners-Lee, director del W3C<sup>5</sup> y creador del protocolo HTTP. El concepto de datos enlazados describe un método de publicar catálogos datos estructurados de una forma en la que sea posible conectarlos con otros conjuntos de datos procedentes de diferentes fuentes.

En [Ber06], Tim Berners-Lee define los cuatro elementos fundamentales para que un conjunto de datos sea considerado un conjunto de datos enlazados:

- Usar URIs para nombrar los elementos.
- Usar URIs HTTP para permitir a las personas acceder y buscar dichos nombres.
- Cuando alguien accede a una URI, devolver la información de forma útil usando los estándares RDF o SPARQL.
- Incluir enlaces a otras URIs para facilitar el descubrimiento de más elementos.

## 3.3. Datos enlazados abiertos

El concepto de datos enlazados abiertos (*Linked Open Data*) combina los conceptos de datos abiertos (*Open Data*) y datos enlazados (*Linked Data*) que se han explicado en las secciones anteriores.

Un conjunto de datos se considera un conjunto de datos enlazados abiertos si cumple los requisitos de los datos enlazados y además se presenta bajo una licencia abierta que no impida su reutilización ni redistribución.

### 3.3.1. Sistema de estrellas

Como se menciona en [Ber06], en el año 2006 Tim Berners-Lee desarrolló un sistema de estrellas o niveles con el objetivo de concienciar (principalmente a las organizaciones gubernamentales) en el uso de datos enlazados abiertos.

A continuación se explica el significado de cada nivel de esta escala:

1. Los datos están disponibles en cualquier formato (pero manteniendo una licencia libre para ser considerados datos abiertos). Un ejemplo serían datos publicados como una imagen escaneada.
2. Los datos se encuentran en un formato estructurado y que pueda ser leído por máquinas. Por ejemplo utilizar un formato *Microsoft Excel* en lugar de una imagen escaneada.
3. Similar al anterior pero utilizando un formato libre, por ejemplo CSV, JSON o XML.
4. Cumple con todos los niveles anteriores pero además utiliza un estándar abierto de la W3C

---

<sup>2</sup><http://datos.gob.es/>

<sup>3</sup><http://data.gov.uk/>

<sup>4</sup><http://www.data.gov/>

<sup>5</sup>World Wide Web Consortium

como RDF o SPARQL. El uso de estos estándares permite que el catálogo de datos sea enlazado por otros catálogos u organizaciones.

5. Cumple con todos los niveles anteriores, pero además enlaza hacia otros catálogos de datos externos.

El portal de datos que se pretende construir en este proyecto pretende cumplir con un nivel de 5 estrellas, publicando los conjuntos de datos en formatos abiertos como RDF, XML o JSON y enlazando con los catálogos de datos de diferentes organizaciones.

## 3.4. RDF Data Cube

En la especificación del RDF Data Cube Vocabulary (recomendación de enero de 2014) [W3C14], EL W3C define el objetivo de este vocabulario de la siguiente forma:

*“There are many situations where it would be useful to be able to publish multi-dimensional data, such as statistics, on the web in such a way that it can be linked to related data sets and concepts. The Data Cube vocabulary provides a means to do this using the W3C RDF (Resource Description Framework) standard [...]”*

Puesto que uno de los objetivos del portal que se construirá en este proyecto es precisamente la publicación de datos multidimensionales en la web, la estructura del modelo de datos intentará adecuarse a la estructura descrita por el RDF Data Cube Vocabulary. El diseño final del modelo de datos puede ser visto de forma detallada en la sección “Modelo de la zona de datos” perteneciente al capítulo 5.

A modo de resumen, el RDF Data Cube pretende definir una estructura y un vocabulario basado en el estándar RDF con el que publicar conjuntos de datos multidimensionales en la web. Los conjuntos de datos son colecciones de datos con una estructura determinada. Los datos contenidos en un conjunto de datos serán de alguno de los siguientes tipos:

**Observaciones** Es la información final a la que se quiere acceder, los valores de las mediciones o los cálculos realizados.

**Información organizativa** Ayuda a encontrar una cierta observación o un conjunto de observaciones. Para encontrar una observación será necesario conocer las dimensiones bajo las que se encuentra.

**Información estructural** Ayuda a interpretar una cierta observación, por ejemplo indicando su unidad de medida o si es un valor exacto o estimado.

**Metadatos del conjunto de datos** Ayudan a describir información sobre el propio conjunto de datos, por ejemplo su publicador.

### 3.4.1. El modelo de cubo

El RDF Data Cube Vocabulary pretende crear el concepto de cubo o *hypercube* como forma de representar la estructura de la información. El cubo en el que se encuentran las observaciones se organiza en torno a un conjunto de dimensiones, atributos y medidas (todos ellos reciben el nombre de *componentes*).

**Dimensiones** Permiten localizar una observación concreta o un conjunto de observaciones. Por ejemplo una dimensión podría ser la zona geográfica sobre la que las observaciones han tenido lugar o el momento en el tiempo al que las observaciones hacen referencia.

**Medidas** Representan el fenómeno concreto observado.

**Atributos** Permiten realizar cuantificar e interpretar los valores observados. Por ejemplo un atributo podría ser la unidad de medida de una observación.

### 3.4.2. Los *slices*

El concepto de *slice* hace referencia a un subconjunto o agrupación de las observaciones existentes en el cubo.

Por ejemplo, usando los datos reales con los que se trabajará en el portal que se pretende construir en este proyecto, dadas una serie de observaciones tomadas para diversos indicadores y países a lo largo de un periodo de tiempo, podría ser interesante agrupar dichas observaciones según su indicador y el momento de tiempo en el que se han realizado. Cada uno de estos grupos representaría todas las observaciones de todos los países para un determinado indicador y momento temporal. Estos grupos reciben el nombre de *slices*.

# Capítulo 4

## Análisis

En este capítulo se realizará el análisis del sistema a construir. Las conclusiones obtenidas en este capítulo serán utilizadas en las posteriores fases del desarrollo.

### 4.1. Definición del sistema

Tal y como se ha mencionado en el capítulo de Introducción, este proyecto se encuadra dentro del proyecto *Rebuilding IFAD's LandPortal RFQ/2013/016/SC* desarrollado por el grupo de investigación WESO y la empresa SB Consulting. El cliente a quien va destinado es Fondo Internacional para el Desarrollo Agrícola, que forma parte de la Organización de las Naciones Unidas.

Conviene definir en este momento una terminología común que se va a utilizar de ahora en adelante. Se hablará de *sistema* para hacer referencia al nuevo Land Portal en su totalidad, por otra parte se hablará de *proyecto* para hacer referencia al backend del nuevo Land Portal del que es objeto la presente documentación.

#### 4.1.1. Alcance del sistema

A continuación se explicará qué partes del desarrollo del nuevo Land Portal tienen cabida en este proyecto. También se mencionarán las partes que no entrarán dentro del alcance del proyecto para que el lector pueda apreciar la complejidad del sistema en su totalidad.

##### 4.1.1.1. Elementos dentro del alcance del proyecto

- **Punto de entrada único para los datos del portal.** Como se ha explicado anteriormente los datos del portal procederán de diversas fuentes de datos y organizaciones externas. Con el fin de mantener el control sobre el tiempo y forma en la que se incluyen nuevos datos será necesario crear un punto único de entrada para los mismos. Para garantizar una cierta uniformidad y asegurar un nivel de calidad mínimo también será necesario definir un formato en el que enviar los datos hacia este punto de entrada.

Por otra parte, un componente como este tiene una gran responsabilidad en el correcto funcionamiento del portal, por lo que será necesario establecer unas medidas de seguridad que eviten la introducción de datos procedentes de orígenes no confiables.

Además será necesario insertar en una base de datos relacional los datos que lleguen al punto de entrada. Los datos que se almacenen en esta base de datos se utilizarán para la construcción del framework que da soporte a las visualizaciones de datos y que se verá a continuación.

- **Framework para proveer información con la que construir visualizaciones de datos.** Este portal de datos no se limitará a almacenar y devolver catálogos de datos, si no que también ofrecerá visualizaciones que permitan a los usuarios acceder los datos de una forma sencilla y atractiva. Como parte del proyecto se desarrollará un framework que permita proveer la información necesaria para construir las visualizaciones de datos.
- **Arquitectura para la creación de vistas personalizadas.** Además de la creación de un framework que provee la información necesaria para construir visualizaciones de datos también será necesario la creación de una arquitectura que permita incluir vistas personalizadas. Con el fin de hacer esta arquitectura lo más general y reutilizable posible se evitará hacer uso de los mecanismos que el CMS provee para la creación de vistas.
- **Mecanismo de internacionalización.** Puesto que este portal ofrecerá datos procedentes de diversas organizaciones internacionales y relativos a multitud de países y continentes distintos será necesario proveer un mecanismo de internacionalización que permita a los usuarios acceder a la información en el lenguaje que prefieran. Además será necesario que el mecanismo de internacionalización no se limite simplemente a soportar traducciones para la información estática, si no que tendrá que ir más allá y permitir la internacionalización de los propios datos.
- **Plataforma social que fomente la participación de los usuarios y complemente la información de los conjuntos de datos.** Puesto que el nuevo Land Portal pretende hacer especial énfasis en la participación de los usuarios será necesario ofrecer una plataforma en la que la comunidad pueda interactuar e intercambiar información. Bajo dicha plataforma se crearán debates para que los usuarios intercambien sus opiniones a cerca de algún tema concreto, noticias para mantener al resto de usuarios informados sobre aquellas informaciones que se consideren necesarias y eventos que tendrán lugar en una fecha concreta. Además también albergará un blog en el que el propio Land Portal coloque aquella información que considere relevante para sus usuarios.  
Dado que esta plataforma estará completamente integrada dentro del nuevo Land Portal será también necesario que cuente con un aspecto uniforme y que mantenga la línea de identidad del portal, de forma que la transición entre las diferentes partes sea transparente al usuario. Las vistas realizadas para esta parte del portal utilizarán los propios mecanismos ofrecidos por el CMS para la creación de vistas y plantillas visuales.
- **Componente de autenticación de los usuarios para usar el API.** Como se verá posteriormente la implementación del API del nuevo Land Portal queda fuera del alcance de este proyecto, aunque sí que será necesario implementar un componente que permita controlar las claves de acceso al API. La seguridad del API es un apartado muy importante, por lo que sólo deberán tener acceso aquellos usuarios que la administración desee. En relación con el punto anterior, la generación de las claves de acceso deberá realizarse de forma transparente al usuario.
- **Unificación de la búsqueda entre las diferentes partes del portal.** Con el fin de centralizar la búsqueda en una única parte del portal, será necesario unificar la búsqueda de la parte social perteneciente al CMS y de la parte de datos, cuyos datos se encuentran almacenados fuera del CMS.

#### 4.1.1.2. Elementos fuera del alcance del proyecto

- **Generación de RDF y enriquecimiento de datos.** Como se ha explicado anteriormente uno de los objetivos de este proyecto es diseñar un punto de entrada único para los datos del portal, además de implementar un mecanismo que inserte dichos datos en una base de datos relacional. Un elemento fuera del alcance de este proyecto, pero que sí se encuentra dentro del sistema real es un componente encargado de la generación de datos en formato RDF y el enriquecimiento de los mismos. Además dicho componente también almacenará los datos en Virtuoso, que fue escogido como servidor semántico tal y como se mencionó en la sección “Alternativas elegidas” del capítulo 2.

- **Importación de datos.** Siendo el sistema que se pretende construir un portal de datos, la importación de los propios datos juega un papel clave en la buena marcha del mismo. Tal y como ya se ha explicado en la sección “Objetivos del proyecto” perteneciente al capítulo 1 los datos con los que se trabajará procederán de varias y muy diversas fuentes. La importación de datos consistirá en unificar todos esos datos en un formato común y enviarlos hacia el punto de entrada de datos para que puedan ser visualizados en el portal.
- **Creación de visualizaciones.** Como se ha explicado anteriormente queda dentro del alcance del proyecto la creación de un framework que provea la información necesaria para crear visualizaciones de datos. Por la complejidad de las visualizaciones, estas quedarán fuera del alcance del proyecto y serán implementadas por un diseñador con experiencia previa en el ámbito de la visualización de datos.
- **Integración con el catálogo de datos.** En la sección “Alternativas elegidas” perteneciente al capítulo 2 de esta misma documentación, se indicó que se utilizará CKAN como catálogo de datos. La integración de CKAN con el resto del portal quedará fuera del alcance del proyecto, así como la incorporación de los datos que llegan por el punto de entrada dentro el propio catálogo.
- **Interfaz visual del portal de datos.** Anteriormente se ha mencionado que sí entrará dentro del alcance del proyecto la creación de una interfaz visual para la parte social del portal. La interfaz visual del portal de datos quedará sin embargo fuera del alcance de este proyecto por la propia necesidad de conseguir una estrecha relación con las visualizaciones de datos. A diferencia de la interfaz visual perteneciente a la plataforma social, la interfaz del portal de datos utilizará la arquitectura para la creación de vistas personalizadas y evitará la utilización de los mecanismos ofrecidos por el CMS.

## 4.2. Identificación de actores del sistema

En esta sección se identificarán todos los actores del sistema. Se consideran actores todas aquellas personas, organizaciones o elementos que toman algún papel en el sistema.

Existirán seis tipos de actores que interactúan con el sistema, dos de estos actores no serán personas, si no otros componentes de software.

**Usuarios anónimos** Los usuarios anónimos son todos aquellos usuarios del portal que no hayan iniciado sesión con una cuenta de usuario. El papel de estos usuarios en el sistema será el de consumidores de información, puesto que únicamente se les permitirá visualizar los contenidos de la zona de datos y de la zona social (exceptuando acceder a la información de los perfiles de otros usuarios registrados). Estos usuarios también podrán utilizar la búsqueda, registrar una nueva cuenta de usuario o iniciar sesión con una cuenta de usuario ya existente.

**Usuarios registrados** Los usuarios registrados son todos aquellos usuarios que tienen una cuenta de usuario en el portal y que además han iniciado sesión con ella. El papel de estos usuarios será tanto de consumidores como creadores de información, puesto que además de ver los contenidos de la zona de datos y la zona social (incluyendo acceder a la información de los perfiles de otros usuarios registrados) también podrán aportar nueva información a la zona social. Concretamente podrán crear debates, eventos y noticias, además de comentar en los debates abiertos o en las entradas del blog.

**Usuarios con acceso al API** Los usuarios con acceso al API son usuarios registrados que además cuentan con una clave de acceso al API pública del portal. Estos usuarios jugarán un papel de creadores, consumidores y difusores de información, puesto que además de todas las capacidades de los usuarios registrados también tendrán un acceso total al API que podrán utilizar para crear servicios o aplicaciones externas que se beneficien de los datos ofrecidos por el nuevo Land Portal.

**Administradores** Los administradores serán aquellos usuarios de confianza que se encarguen de mantener el funcionamiento del nuevo Land Portal. Estos usuarios tendrán principalmente un papel de moderadores de la zona social del portal. Tendrán capacidad para editar o eliminar los debates, noticias o eventos creados por otros usuarios; abrir o cerrar los debates para que el resto de usuarios puedan participar en ellos; moderar o eliminar los comentarios introducidos por otros usuarios; publicar o modificar contenido en el blog de Land Portal; gestionar el contenido del catálogo de datos y otorgar o eliminar las capacidades de administración o acceso al API del resto de usuarios registrados.

**Importadores de datos** A diferencia de los actores descritos anteriormente, los importadores de datos no serán personas, si no que serán aplicaciones creadas con el fin de insertar nuevos datos en el portal. El objetivo de estas herramientas será capturar datos provenientes de diferentes fuentes u organizaciones, transformar dichos datos a un XML Schema definido y enviarlos al punto de entrada de datos del portal.

**Visualizaciones de datos** De la misma forma que sucede con los importadores de datos, las visualizaciones serán aplicaciones creadas con el fin de extraer datos del portal. El objetivo de las visualizaciones será transformar los datos contenidos en el portal en representaciones visuales que resulten atractivas para los usuarios.

## 4.3. Requisitos del sistema

### 4.3.1. Especificación de los requisitos funcionales

A continuación se procederá a obtener, analizar y organizar los requisitos con los que contará el sistema que se construirá en este proyecto. Cabe destacar que este listado de requisitos sólo recoge los requisitos que se implementarán en este proyecto y es un subconjunto de todos los requisitos que forman parte del nuevo Land Portal.

Para hacer más sencilla la lectura, el catálogo de requisitos se dividirá en diferentes tablas dependiendo del componente al que afecten. Cada entrada de la tabla de requisitos contendrá la siguiente información:

- **Código de identificación.** El código de identificación pretende identificar a cada requisito de forma única para hacer así posible referirse a él posteriormente.
- **Nombre del requisito.** El nombre pretende introducir de forma corta y sencilla el objetivo de cada requisito.
- **Descripción del requisito.** La descripción pretende detallar cada requisito en profundidad.

#### 4.3.1.1. Requisitos de la sección de datos

De ahora en adelante la sección de datos del portal recibirá el nombre de “LandBook”. A continuación, en la tabla 4.1 se muestra el listado de requisitos pertenecientes al LandBook.

Cuadro 4.1: Tabla de requisitos de la sección de datos.

Listado de requisitos de la sección de datos		
Código	Nombre	Descripción
RLB 1	Contenido de la sección de datos	La sección de datos estará compuesta de: regiones, países, indicadores, organizaciones, catálogo de datos y widgets.
RLB 2	Acceso a la sección de datos	El sistema permitirá acceder al contenido de la sección de datos tanto a usuarios registrados como anónimos.

Continuación de la tabla 4.1		
Código	Nombre	Descripción
RLB 3	Identificación del contenido	Todo el contenido ofrecido en el LandBook tendrá una URL única.
RLB 4	Arquitectura para las vistas	Las vistas del LandBook deberán evitar el uso de los mecanismos del plantillas visuales del CMS.
RLB 4.1	Arquitectura para las vistas	Las rutas para las nuevas vistas se especificarán a través de un fichero de configuración, de forma que sea posible modificarlas sin necesidad de acceder al código fuente.
RLB 4.2	Arquitectura para las vistas	Las plantillas y modelos para las vistas se buscarán utilizando un mecanismo de convenio de nombres.
RLB 5	Integración con la zona social	Cuando se acceda a un país en el LandBook, se mostrará un enlace para ver todo el contenido relacionado con dicho país creado en la zona social..
RLB 6	Soporte a las visualizaciones	El LandBook proveerá un sistema que pueda ser utilizado por las vistas con el fin de crear visualizaciones de datos.
RLB 6.1	Soporte a las visualizaciones	El sistema permitirá devolver el valor medio de todas las observaciones existentes para una región e indicador concretos.
RLB 6.2	Soporte a las visualizaciones	El sistema permitirá devolver todas las observaciones existentes para una región e indicador concretos.
RLB 6.3	Soporte a las visualizaciones	El sistema permitirá devolver el valor medio de todas las observaciones existentes para un determinado indicador, independientemente de la región a la que hagan referencia.
RLB 6.4	Soporte a las visualizaciones	El sistema permitirá devolver una comparación del el valor medio de todas las observaciones existentes para dos indicadores concretos, independientemente de la región a la que hagan referencia.
RLB 6.5	Soporte a las visualizaciones	El sistema permitirá devolver todas las observaciones para un país e indicador concretos.
RLB 7	Soporte a la internacionalización	El LandBook permitirá mostrar la información que contiene en diferentes idiomas.
RLB 7.1	Soporte a la internacionalización	El LandBook permitirá acceder a la información en inglés.
RLB 7.2	Soporte a la internacionalización	El LandBook permitirá acceder a la información en francés.
RLB 7.3	Soporte a la internacionalización	El LandBook permitirá acceder a la información en español.

#### 4.3.1.2. Requisitos de la sección social

De ahora en adelante la sección social del portal recibirá el nombre de “LandDebate”. En la tabla 4.2 se muestra el listado de requisitos pertenecientes al LandDebate.

Cuadro 4.2: Tabla de requisitos de la zona social del portal.

Listado de requisitos de la búsqueda		
Código	Nombre	Descripción
RLD 1	Registro de usuarios	El LandDebate permitirá realizar registros de nuevos usuarios en el sistema.
RLD 1.1	Registro de usuarios	Los nuevos registros requerirán la introducción de un nombre de usuario. Este nombre de usuario será único en todo el portal.
RLD 1.2	Registro de usuarios	Los nuevos registros requerirán la introducción de una contraseña de usuario. Para evitar errores será necesario que el usuario repita la contraseña antes de completar el registro.

Continuación de la tabla 4.2		
Código	Nombre	Descripción
RLD 1.4	Registro de usuarios	Los nuevos registros requerirán la introducción del nombre real y los apellidos de la persona que se está registrando.
RLD 1.5	Registro de usuarios	Los nuevos usuarios podrán seleccionar el continente en el que se encuentran. Este campo no será obligatorio.
RLD 1.6	Registro de usuarios	Los nuevos usuarios podrán seleccionar hasta un máximo de 7 países en los que estén interesados. Este campo no será obligatorio.
RLD 1.7	Registro de usuarios	El registro de usuarios será accesible desde cualquier punto del portal.
RLD 1.8	Registro de usuarios	Los usuarios podrán registrarse en el portal utilizando sus cuentas de Twitter o Facebook.
RLD 2	Roles de usuario	Los usuarios del portal tendrán diferentes roles de usuario en función de los cuales podrán realizar diferentes tareas en el LandDebate.
RLD 2.1	Roles de usuario	El rol de usuario <i>anónimo</i> será automáticamente asignado a todos los usuarios que no se hayan registrado o no hayan iniciado sesión en el portal.
RLD 2.2	Roles de usuario	El rol de usuario <i>registrado</i> será automáticamente asignado a todos los usuarios que se hayan registrado e inicien sesión en el portal. Los usuarios con el rol <i>registrado</i> podrán crear contenido en el LandDebate y comentar en las entradas del blog y los debates.
RLD 2.3	Roles de usuario	El rol de usuario <i>administrador</i> tendrá permisos para gestionar cualquier parte del portal y otorgar roles al resto de usuarios. Todos los usuarios con rol <i>administrador</i> tendrán también el rol <i>registrado</i> automáticamente.
RLD 2.4	Roles de usuario	El rol de usuario <i>con acceso al API</i> será asignado por los administradores a aquellos usuarios registrados que deban tener una clave de acceso al API. Todos los usuarios pertenecientes a este rol también pertenecerán al rol <i>registrado</i> .
RLD 3	Inicio de sesión	Los usuarios que previamente se hayan registrado podrán iniciar sesión en el portal utilizando su nombre de usuario y contraseña.
RLD 3.1	Inicio de sesión	El formulario de inicio de sesión será accesible desde todas las partes del portal.
RLD 3.2	Inicio de sesión	Los usuarios que se hayan registrado utilizando su cuenta de Twitter o Facebook podrán iniciar sesión utilizando un botón y no necesitarán introducir su nombre de usuario ni contraseña.
RLD 3.3	Inicio de sesión	Los usuarios registrados podrán pedir una nueva contraseña para acceder al portal en caso de haber olvidado la suya.
RLD 4	Funcionamiento de los eventos	El sistema permitirá la creación de eventos que tendrán lugar en una fecha determinada.
RLD 4.1	Funcionamiento de los eventos	Durante la creación de un evento será necesario introducir su título, contenido y la fecha en la que tendrá lugar.
RLD 4.2	Funcionamiento de los eventos	Durante la creación de un evento podrán seleccionarse aquellos tópicos con los que esté relacionado.
RLD 4.3	Funcionamiento de los eventos	Durante la creación de un evento podrá incluirse una imagen que acompañe al contenido.
RLD 4.4	Funcionamiento de los eventos	Los eventos podrán ser creados por cualquier usuario que cuente con el rol de <i>registrado</i> .
RLD 4.5	Funcionamiento de los eventos	Los eventos podrán ser editados por su creador o por un usuario con rol de <i>administrador</i> .
RLD 4.5	Funcionamiento de los eventos	Los eventos sólo podrán ser eliminados por un usuario con rol de <i>administrador</i> .

Continuación de la tabla 4.2		
Código	Nombre	Descripción
RLD 5	Funcionamiento de las noticias	El sistema permitirá la creación de noticias. Las noticias presentarán información que sea de interés para los miembros del portal.
RLD 5.1	Funcionamiento de las noticias	Durante la creación de una noticia será necesario introducir su título y contenido.
RLD 5.2	Funcionamiento de las noticias	Durante la creación de una noticia será posible incluir una imagen que acompañe al contenido.
RLD 5.3	Funcionamiento de las noticias	Las noticias podrán ser creadas por cualquier usuario que cuente con el rol de <i>registrado</i> .
RLD 5.4	Funcionamiento de las noticias	Las noticias podrán ser editadas por su creador o por un usuario con rol de <i>administrador</i> .
RLD 5.5	Funcionamiento de las noticias	Las noticias sólo podrán ser eliminadas por un usuario con rol de <i>administrador</i> .
RLD 6	Funcionamiento de las entradas del blog	El sistema permitirá la creación de entradas en el blog. Las entradas en el blog representan información relevante u opiniones que se emiten desde el propio Land Portal.
RLD 6.1	Funcionamiento de las entradas del blog	Las entradas del blog sólo podrán ser creadas, editadas o eliminadas por un usuario con rol de <i>administrador</i> .
RLD 6.2	Funcionamiento de las entradas del blog	Durante la creación de una entrada del blog será necesario introducir su título y contenido.
RLD 6.3	Funcionamiento de las entradas del blog	Durante la creación de una entrada del blog será posible introducir una imagen que acompañe al contenido.
RLD 6.4	Funcionamiento de las entradas del blog	Durante la creación de una entrada del blog será posible seleccionar aquellos tópicos que se consideren relacionados con el contenido de la misma.
RLD 6.5	Funcionamiento de las entradas del blog	Cualquier usuario <i>registrado</i> podrá incluir un nuevo comentario o replicar a un comentario ya existente en una entrada del blog.
RLD 7	Funcionamiento de los debates	El sistema permitirá la creación de debates. Los debates tienen como finalidad fomentar el intercambio de ideas y la participación de los usuarios de la comunidad.
RLD 7.1	Funcionamiento de los debates	Los debates podrán ser creados por cualquier usuario <i>registrado</i> .
RLD 7.2	Funcionamiento de los debates	Los debates sólo podrán ser editados por su creador o por un usuario con rol de <i>administrador</i> .
RLD 7.3	Funcionamiento de los debates	Los debates sólo podrán ser eliminados por un usuario con rol de <i>administrador</i> .
RLD 7.4	Funcionamiento de los debates	Los debates permitirán a un <i>administrador</i> abrir o cerrar los comentarios en función de la fecha en la que el debate esté activo.
RLD 7.5	Funcionamiento de los debates	Cualquier usuario <i>registrado</i> podrá crear un nuevo comentario o responder a uno ya existente en un debate, siempre que el debate esté activo.
RLD 7.6	Funcionamiento de los debates	Durante la creación de un nuevo debate será necesario incluir su título y contenido.
RLD 7.7	Funcionamiento de los debates	Durante la creación de un nuevo debate será necesario incluir las fechas entre las que el debate estará activo.
RLD 7.8	Funcionamiento de los debates	Durante la creación de un nuevo debate será posible incluir una imagen que acompañe al contenido.
RLD 7.9	Funcionamiento de los debates	Durante la creación de un nuevo debate será posible indicar los tópicos con los que está relacionado.

Continuación de la tabla 4.2		
Código	Nombre	Descripción
RLD 7.10	Funcionamiento de los debates	Durante la creación de un nuevo debate será posible indicar las regiones con las que está relacionado.
RLD 8	Funcionamiento de las organizaciones	El sistema permitirá la creación de organizaciones.
RLD 8.1	Funcionamiento de las organizaciones	Durante la creación de una organización será necesario incluir su nombre y una descripción larga.
RLD 8.2	Funcionamiento de las organizaciones	Durante la creación de una organización será posible incluir los tópicos con los que está relacionada.
RLD 8.3	Funcionamiento de las organizaciones	Durante la creación de una organización será posible incluir los países sobre los que trabaja.
RLD 8.3	Funcionamiento de las organizaciones	Durante la creación de una organización será posible incluir los países sobre los que trabaja.
RLD 8.4	Funcionamiento de las organizaciones	Durante la creación de una organización será posible incluir los áreas sobre los que opera.
RLD 8.5	Funcionamiento de las organizaciones	Durante la creación de una organización será necesario incluir la URL de su sitio web.
RLD 8.6	Funcionamiento de las organizaciones	Las organizaciones sólo podrán ser creadas por un usuario con rol de <i>administrador</i> .
RLD 8.7	Funcionamiento de las organizaciones	Las organizaciones sólo podrán ser creadas por un usuario con rol de <i>administrador</i> .
RLD 8.8	Funcionamiento de los debates	Las organizaciones sólo podrán ser creadas por un usuario con rol de <i>administrador</i> .

#### 4.3.1.3. Requisitos de la búsqueda

En la tabla 4.3 se muestra el listado de requisitos pertenecientes a la búsqueda.

Cuadro 4.3: Tabla de requisitos de la búsqueda.

Listado de requisitos de la búsqueda		
Código	Nombre	Descripción
RBUS 1	Integración en el portal	La búsqueda deberá ser accesible desde todas las partes del portal.
RBUS 1.1	Integración en el portal	La búsqueda contará con una vista especialmente dedicada a mostrar los resultados.
RBUS 2	Integración con el LandBook	La búsqueda podrá indexar el contenido existente en el LandBook.
RBUS 2.1	Integración con el LandBook	La búsqueda podrá indexar y mostrar resultados pertenecientes a los indicadores del LandBook.
RBUS 2.2	Integración con el LandBook	La búsqueda podrá indexar y mostrar resultados pertenecientes a los países del LandBook.
RBUS 3	Integración con el LandDebate	La búsqueda podrá indexar el contenido perteneciente al LandDebate.
RBUS 3.1	Integración con el LandDebate	La búsqueda podrá indexar y mostrar resultados pertenecientes a los debates del LandDebate.
RBUS 3.2	Integración con el LandDebate	La búsqueda podrá indexar y mostrar resultados pertenecientes a los eventos existentes en el LandDebate.
RBUS 3.3	Integración con el LandDebate	La búsqueda podrá indexar y mostrar resultados pertenecientes a las noticias del LandDebate.
RBUS 3.4	Integración con el LandDebate	La búsqueda podrá indexar y mostrar resultados pertenecientes a las entradas del blog.
RBUS 3.5	Integración con el LandDebate	La búsqueda podrá indexar y mostrar resultados pertenecientes a las organizaciones presentes en el LandDebate.

Continuación de la tabla 4.3		
Código	Nombre	Descripción
RBUS 3.6	Integración con el LandDebate	La búsqueda podrá indexar y mostrar resultados pertenecientes a los comentarios creados por los usuarios en los debates.
RBUS 3.7	Integración con el LandDebate	La búsqueda podrá indexar y mostrar resultados pertenecientes a los comentarios creados por los usuarios en las entradas de blog.
RBUS 4	Personalización de los resultados	La búsqueda permitirá mostrar de diferente forma los resultados en función del tipo de contenido al que pertenezcan.
RBUS 4.1	Personalización de los resultados	Los resultados de la búsqueda mostrarán una etiqueta indicando de qué tipo de contenido se trata.
RBUS 4.2	Personalización de los resultados	La etiqueta de tipo de contenido presente en los resultados de la búsqueda permitirá acceder a todo el contenido del mismo tipo existente en el portal.
RBUS 4.3	Personalización de los resultados	Los resultados de búsqueda pertenecientes a un país del LandBook incluirán una imagen de la bandera de dicho país.
RBUS 5	Acceso al contenido	Al pulsar sobre un resultado de la búsqueda deberá cargarse el contenido completo del mismo.
RBUS 6	Priorización de resultados	La búsqueda priorizará los resultados pertenecientes al LandBook sobre los resultados pertenecientes al LandDebate.
RBUS 7	Indexación de contenido	La indexación de contenido tendrá lugar de forma periódica y automática, sin necesidad de intervención humana.
RBUS 7.1	Indexación de contenidos	La indexación de contenido podrá ser ejecutada de forma manual por un usuario con rol de <i>administrador</i> .

#### 4.3.1.4. Requisitos del punto de entrada de datos

A continuación, en la tabla 4.4 se muestra el listado de requisitos pertenecientes al punto de entrada de datos al portal.

Cuadro 4.4: Tabla de requisitos del punto de entrada de datos.

Listado de requisitos de la búsqueda		
Código	Nombre	Descripción
RPED 1	Almacenamiento de datos	El punto de datos guardará los datos que reciba en varios puntos de almacenamiento.
RPED 1.1	Almacenamiento de datos	El punto de entrada almacenará los datos en una base de datos relacional.
RPED 1.2	Almacenamiento de datos	El punto de entrada almacenará los datos en un servidor semántico, previa transformación de los mismos en formato RDF. Como se ha mencionado en la sección “Alternativas elegidas” perteneciente al capítulo 2, el servidor semántico seleccionado ha sido Virtuoso.
RPED 1.3	Almacenamiento de datos	El punto de entrada almacenará los datos en el catálogo de datos. Como se ha mencionado en la sección “Alternativas elegidas” perteneciente al capítulo 2, el catálogo de datos seleccionado ha sido CKAN.
RPED 1.4	Almacenamiento de datos	El punto de entrada podrá ser extendido con nuevos componentes de almacenamiento sin necesidad de modificar los ya existentes.
RPED 1.5	Almacenamiento de datos	Los datos que se guarden en los distintos puntos de almacenamiento serán equivalentes entre sí.

Continuación de la tabla 4.4		
Código	Nombre	Descripción
RPED 2	Integridad de los datos	El punto de entrada de datos leerá la información de los nuevos catálogos de datos en un formato determinado por un XML Schema.
RPED 2.1	Integridad de los datos	La información de nuevos catálogos de datos que llegue al punto de entrada y no sea conforme al XML Schema especificado será rechazada y no se insertará en el portal.
RPED 2.2	Integridad de los datos	La inserción de datos se hará de manera transaccional, de forma que si se produce algún fallo durante el proceso no se incluya en el portal ningún dato inconsistente.
RPED 3	Transformación de datos	El punto de entrada transformará la información que reciba a un modelo propio antes de ser exportada a los diferentes sistemas de almacenamiento.

#### 4.3.2. Especificación de los requisitos no funcionales

En la tabla 4.5 se mostrará el listado de requisitos no funcionales de este proyecto.

Cuadro 4.5: Tabla de requisitos no funcionales.

Listado de requisitos no funcionales		
Código	Nombre	Descripción
RNF 1	Interfaz del sistema	La interfaz del sistema contará con una apariencia <i>flat</i> conforme a las últimas tendencias de diseño web.
RNF 1.1	Interfaz del sistema	La interfaz del sistema contará con un diseño <i>responsive</i> que permita su adaptación a distintos tamaños de pantalla.
RNF 1.2	Interfaz del sistema	La interfaz del sistema contará con un diseño unificado a lo largo de todas las secciones que lo componen.
RNF 2	Seguridad del punto de entrada de datos	El punto de entrada de datos permitirá utilizar una lista blanca con la que restringir los orígenes de las peticiones de entrada de datos.
RNF 3	Escalabilidad del punto de entrada de datos	El punto de entrada de datos deberá ser escalable para permitir la inserción de grandes volúmenes de datos.
RNF 3.1	Escalabilidad del punto de entrada de datos	El punto de entrada de datos podrá procesar catálogos de datos con hasta 500.000 observaciones.
RNF 4	Rendimiento del LandBook	El LandBook permitirá cachear las consultas que se realicen a la base de datos con el objetivo de maximizar el rendimiento de las visualizaciones de datos.

## 4.4. Identificación de subsistemas

Una vez identificados los interesados tanto directos como indirectos y los requisitos del sistema se procederá a realizar una descomposición en subsistemas. Cada subsistema tendrá una funcionalidad única y acotada.

A continuación se detalla cada uno de los subsistemas identificados:

**Subsistema de búsqueda** El subsistema de búsqueda es el encargado de gestionar todas las búsquedas que realicen los usuarios y retornar los resultados convenientes. Además también se encargará de indexar periódicamente (o puntualmente si un administrador lo desea) los

contenidos del portal.

**Subsistema de gestión de usuarios** Este subsistema será el encargado de gestionar todas las operaciones que se realizan relacionadas con los datos de un usuario. Algunas de las tareas gestionadas por este subsistema son los registros, los inicios de sesión o la asignación de diferentes roles de usuario.

**Subsistema de gestión de debates** Este subsistema se encargará de gestionar todas las operaciones relacionadas con los debates, desde la creación y modificación de nuevos debates hasta la apertura o cierre de los debates ya existentes y la gestión de los comentarios de los usuarios.

**Subsistema de gestión de eventos** Este subsistema se encargará de gestionar todas las operaciones que guarden relación con los eventos. Las tareas típicas de este subsistema serán la creación y la edición de eventos.

**Subsistema de gestión de noticias** Este subsistema se encargará de gestionar las operaciones relacionadas con las noticias, principalmente la creación y edición de noticias.

**Subsistema de gestión del blog** Este subsistema se encargará de gestionar las tareas relacionadas con las entradas del blog. Algunas funciones de este subsistema serán la creación y edición de entradas en el blog y la moderación de los comentarios de usuarios.

**Subsistema de gestión de organizaciones** Este subsistema se encargará de gestionar las tareas relacionadas con las organizaciones, principalmente la creación, edición y eliminación de noticias.

**Subsistema de gestión de datos** Este subsistema será el encargado de gestionar los datos de la parte de datos del portal. Las funciones más representativas de este subsistema son la inserción de nuevos catálogos de datos y la provisión de información con la que crear las visualizaciones.

**Subsistema de gestión de comentarios** Este subsistema será el encargado de gestionar los comentarios de los usuarios. Las funciones más representativas de este subsistema son la creación, la modificación y la eliminación de comentarios.

Es necesario tener en cuenta que el sistema final que será el nuevo Land Portal contará con más subsistemas que no son objeto de este proyecto y, por tanto, no figuran en este análisis.

#### 4.4.1. Descripción de las interfaces entre subsistemas

Tras haber realizado la identificación de los subsistemas se describirá de qué forma se relacionarán dichos subsistemas.

Todos los subsistemas mencionados anteriormente se encontrarán en el mismo servidor web, aunque formarán parte de diferentes aplicaciones y componentes. Concretamente:

- Los subsistemas de gestión de usuarios, debates, eventos, noticias, entradas del blog y comentarios formarán parte del gestor de contenidos o CMS. A lo largo de esta documentación el conjunto de todos estos subsistemas recibirá también el nombre de “zona social”.
- El subsistema de búsqueda formará parte del buscador que, como se ha explicado en la sección Alternativas elegidas perteneciente al capítulo 2.2 sera Apache Solr.
- El subsistema de gestión de datos estará dividido entre una aplicación encargada de la inserción de datos provenientes de fuentes externas y un framework que dará soporte a las visualizaciones de datos. Por su funcionalidad, este subsistema también recibirá el nombre de “zona de datos” en esta documentación.

Los subsistemas que forman parte del gestor de contenidos (gestión de usuarios, debates, eventos, noticias, entradas del blog y comentarios) se comunicarán con el subsistema de búsqueda a través del API HTTP ofrecido por el buscador.

Por otra parte el gestor de contenidos y el subsistema de gestión de datos se comunicarán a través de una base de datos compartida, será el gestor de contenidos quien extraiga los datos necesarios de la base de datos en la que el subsistema de gestión de datos almacena la información.

En la imagen 4.1 se puede ver un esquema que representa la situación de dichos componentes y subsistemas.

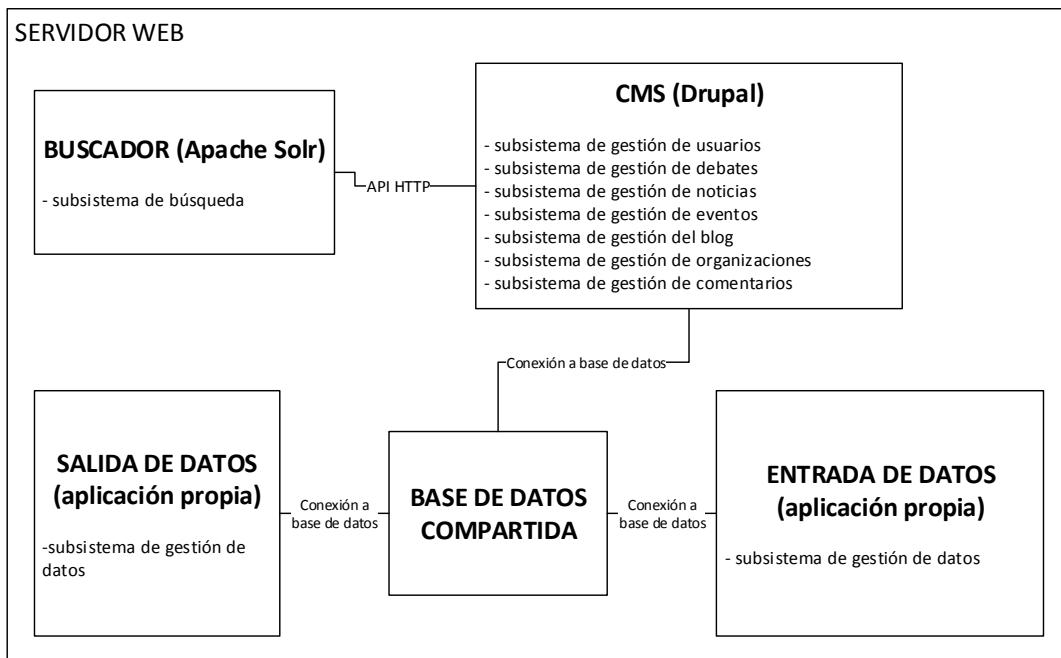


Figura 4.1: Diagrama de subsistemas

## 4.5. Especificación de casos de uso

En esta sección se detallarán los casos de uso del sistema. Para facilitar la lectura, los casos de uso y escenarios se agruparán por subsistemas.

Cada caso de uso tendrá la siguiente información:

- **Nombre.** Nombre que permite identificar al caso de uso.
- **Precondiciones.** Las precondiciones indican una situación que es de obligado cumplimiento antes del caso de uso. Por simplicidad, las precondiciones se omitirán en los casos de uso en los que no existan.
- **Descripción.** La descripción indica de forma resumida la finalidad del caso de uso.
- **Actores.** Los actores serán todos aquellos posibles participantes en el caso de uso.
- **Escenario principal.** El escenario principal indica de forma ordenada los pasos que se realizan en el caso de uso.
- **Escenarios alternativos.** Los escenarios alternativos representan variaciones o divergencias respecto al escenario principal. Por simplicidad, los escenarios alternativos se omitirán en los

casos de uso en los que no existan.

#### 4.5.1. Subsistema de gestión de usuarios

En esta sección se detallarán los casos de uso y escenarios pertenecientes al subsistema de gestión de usuarios. La figura 4.2 muestra el diagrama de casos de uso del mismo.

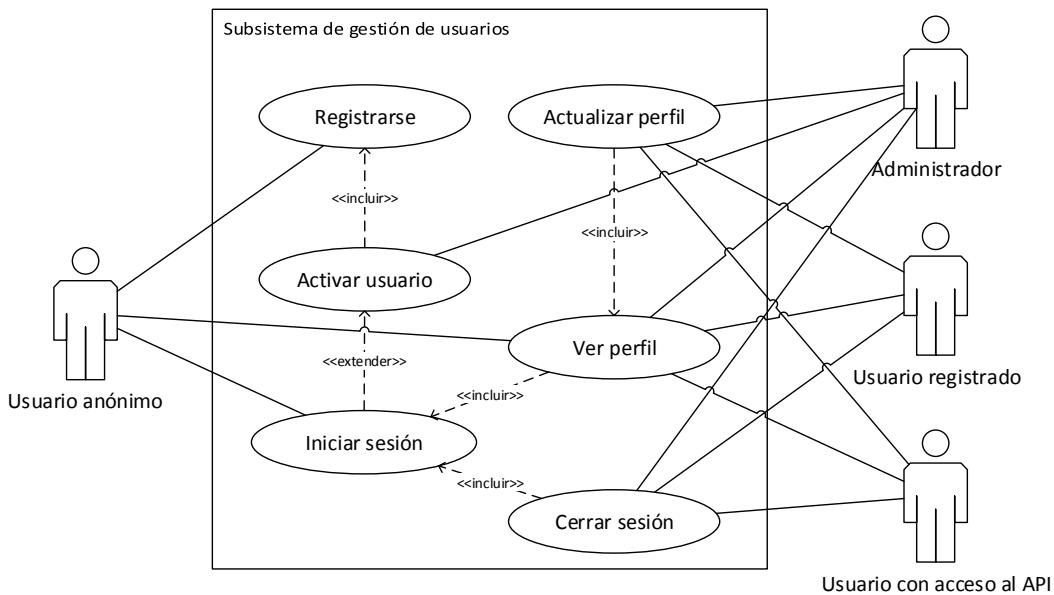


Figura 4.2: Diagrama de casos de uso del subsistema de gestión de usuarios

##### 4.5.1.1. Caso de uso “registrarse”

**Descripción** El usuario no dispone de una cuenta en el sistema y quiere crear una.

**Actores** Cualquier usuario anónimo.

##### Escenario principal

1. El usuario accede al formulario de registro.
2. Una vez en el formulario de registro, el usuario rellena todos los campos requeridos.
3. Si el usuario lo decide también puede llenar los campos opcionales.
4. Tras llenar los campos el usuario pulsa el botón de registro.
5. El sistema guarda los datos y crea la nueva cuenta de usuario con estado bloqueado.

**Escenario alternativo 1** El usuario no rellena todos los campos necesarios.

1. Cuando el usuario pulse el botón para completar el registro el sistema notificará del error.
2. Se continuará desde el punto 2 del escenario principal.

**Escenario alternativo 2** El nombre de usuario ya existe en el sistema.

1. Cuando el usuario pulse el botón para completar el registro el sistema notificará del error.
2. Se continuará desde el punto 2 del escenario principal.

**Escenario alternativo 3** El email de usuario ya existe en el sistema.

1. Cuando el usuario pulse el botón para completar el registro el sistema notificará del error.
2. Se continuará desde el punto 2 del escenario principal.

#### 4.5.1.2. Caso de uso “activar usuario”

**Descripción** El administrador activa una cuenta de un usuario para que este pueda utilizarla.

**Actores** El administrador del sistema.

**Precondiciones** La cuenta de usuario debe ser registrada previamente en el sistema.

**Escenario principal**

1. El administrador accede al panel de administración.
2. Una vez en el panel de administración, accede a la vista de usuarios.
3. El administrador selecciona el usuario o usuarios a activar.
4. Pulsa el botón correspondiente para activar los usuarios.
5. El sistema actualiza el estado de los usuarios y permite que inicien sesión en el sistema.

#### 4.5.1.3. Caso de uso “iniciar sesión”

**Descripción** El usuario accede al sistema utilizando su nombre de usuario y contraseña.

**Actores** Cualquier usuario anónimo.

**Precondiciones** La cuenta de usuario con la que se inicia sesión ha sido activada por un administrador.

**Escenario principal**

1. El usuario accede al formulario de inicio de sesión
2. El usuario introduce su nombre de usuario
3. El usuario introduce su contraseña
4. El usuario inicia sesión pulsando el botón correspondiente

**Escenario alternativo 1** El usuario no existe

1. El sistema notificará al usuario de que el nombre de usuario introducido no existe
2. Se continuará desde el paso 2 del escenario principal

**Escenario alternativo 2** La contraseña es incorrecta

1. El sistema notificará al usuario de que la contraseña introducida es incorrecta

2. Se continuará desde el paso 2 del escenario principal

#### 4.5.1.4. Caso de uso “cerrar sesión”

**Descripción** El usuario quiere cerrar su sesión en el sistema.

**Actores** Administrador, usuario registrado o usuario con capacidad de acceso al API.

**Precondiciones** Haber iniciado sesión en el sistema

##### Escenario principal

1. El usuario con sesión iniciada pulsa en el botón para cerrar sesión.
2. El sistema redirige al usuario a la vista principal con la sesión ya cerrada.

#### 4.5.1.5. Caso de uso “ver perfil”

**Descripción** El usuario quiere ver la información sobre su cuenta almacenada en el sistema.

**Actores** Administrador, usuario registrado, usuario con capacidad de acceso al API.

**Precondiciones** Haber iniciado sesión en el sistema

##### Escenario principal

1. El usuario con sesión iniciada pulsa en el botón para ver su información.
2. El sistema muestra una vista de la cuenta del usuario con todos los datos almacenados.

**Escenario alternativo 1** El usuario cuenta con capacidad de acceso al API

1. Similar al escenario principal, pero el sistema mostrará también el *token* de acceso al API.

**Escenario alternativo 2** Acceso a un perfil no propio.

1. El usuario (registrado o anónimo) pulsa en el perfil de otro usuario del sistema.
2. El sistema muestra una vista de la cuenta de usuario.
3. El sistema no muestra en ningún caso el *token* de acceso al API.

#### 4.5.1.6. Caso de uso “actualizar perfil”

**Descripción** El usuario quiere modificar la información almacenada en el sistema sobre su cuenta.

**Actores** Administrador, usuario registrado o usuario con capacidad de acceso al API.

**Precondiciones** Haber iniciado sesión en el sistema

##### Escenario principal

1. El usuario con sesión iniciada pulsa en el botón para ver su información.
2. Una vez en la vista del perfil, el usuario pulsa el botón para modificar su información.
3. El usuario modifica los datos que considere necesarios.

4. El usuario introduce su contraseña actual para asegurar su identidad.
5. El sistema actualiza la información de la cuenta de usuario.

**Escenario alternativo 1** La contraseña actual no es válida.

1. El sistema informará al usuario de su error y no modificará la información almacenada.
2. Se continua desde el punto 3 del escenario principal.

**Escenario alternativo 2** El usuario quiere modificar su contraseña.

1. El usuario debe introducir su nueva contraseña dos veces para evitar errores.
2. Se continua desde el punto 4 del escenario principal.

**Escenario alternativo 3** Las nuevas contraseñas no coinciden.

1. Las nuevas contraseñas introducidas no coinciden.
2. El sistema informa al usuario del error.
3. Se continua desde el punto 1 del escenario alternativo 2.

**Escenario alternativo 4** El nuevo correo es inválido o ya existe en el sistema.

1. El sistema informará al usuario de su error y no modificará la información almacenada.
2. Se continua desde el punto 3 del escenario principal.

#### 4.5.2. Subsistema de gestión de debates

En esta sección se detallarán los casos de uso y escenarios pertenecientes al subsistema de gestión de debates. La figura 4.3 muestra el diagrama de casos de uso del mismo.

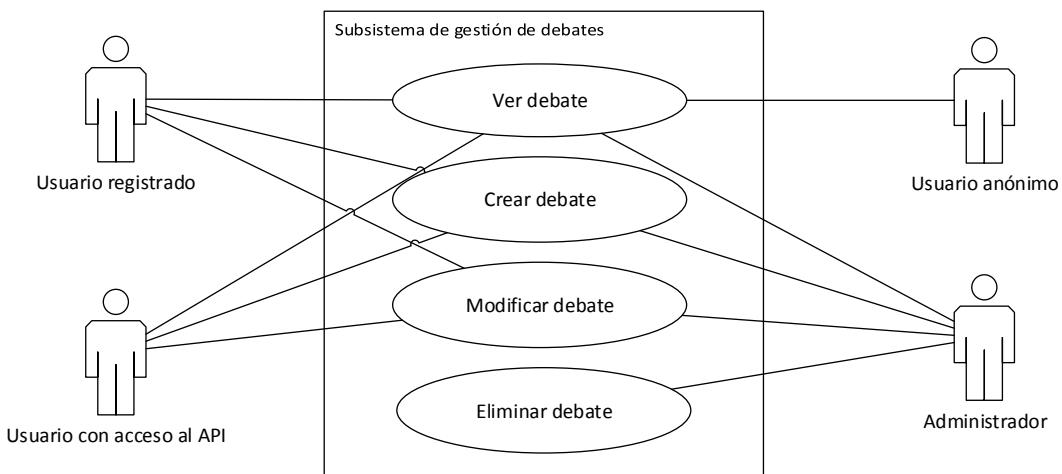


Figura 4.3: Diagrama de casos de uso del subsistema de gestión de debates

##### 4.5.2.1. Caso de uso “ver debate”

**Descripción** El usuario quiere ver el contenido y los comentarios de un debate.

**Actores** Cualquier rol de usuario registrado o no en el sistema.

#### **Escenario principal**

1. El usuario accede a la vista de los debates.
2. Una vez en la vista de debates, elige el debate del que quiere ver sus contenidos.
3. El usuario pulsa en el título del debate elegido.
4. El sistema carga la vista del debate mostrando tanto el contenido como los comentarios del mismo.

#### **4.5.2.2. Caso de uso “crear debate”**

**Descripción** El usuario quiere crear un nuevo debate para que puedan participar el resto de usuarios.

**Actores** Administrador, usuario registrado o usuario con capacidad de acceso al API.

**Precondiciones** Haber iniciado sesión en el sistema.

#### **Escenario principal**

1. El usuario accede a la vista de debates.
2. Pulsa el botón para crear un debate.
3. Rellena los campos requeridos en el formulario de creación del debate.
4. Rellena los campos opcionales que considere necesarios en el formulario de creación del debate.
5. Pulsa el botón para guardar el debate.
6. El sistema guarda el debate con estado *próximamente*.

**Escenario alternativo 1** El usuario no ha llenado todos los campos requeridos.

1. El sistema notificará al usuario de que faltan campos por llenar.
2. Se continuará desde el punto 3 del escenario principal.

#### **4.5.2.3. Caso de uso “modificar debate”**

**Descripción** Un usuario quiere modificar el contenido de un debate.

**Actores** Administrador, usuario registrado o usuario con capacidad de acceso al API.

**Precondiciones** Haber iniciado sesión en el sistema.

#### **Escenario principal**

1. El usuario accede a la vista de debates.
2. Selecciona el debate cuyo contenido desea modificar.
3. Una vez en el debate adecuado, pulsa el botón destinado a modificarlo.
4. Modifica la información que sea necesaria (excepto el estado del debate, el proceso de

modificación del estado de un debate está descrito en el escenario alternativo 1).

5. Pulsa el botón para guardar las modificaciones.
6. El sistema guarda los cambios y actualiza los contenidos del debate.

**Escenario alternativo 1** Modificar el estado de un debate.

1. El administrador accede al panel de administración.
2. Selecciona el debate que desea modificar y pulsa el botón correspondiente.
3. En la vista de edición del debate selecciona el nuevo estado que desea asignarle.
4. Se continúa desde el punto 5 del escenario principal.

**Escenario alternativo 2** No se rellenan todos los campos requeridos

1. El sistema informa al usuario del error y no modifica la información almacenada.
2. Se continúa desde el punto 4 del escenario principal.

#### 4.5.2.4. Caso de uso “eliminar debate”

**Descripción** El administrador quiere eliminar un debate.

**Actores** El administrador del sistema.

**Precondiciones** Haber iniciado sesión en el sistema.

**Escenario principal**

1. El administrador accede a la vista de administración.
2. Localiza el debate que desea eliminar y pulsa el botón correspondiente.
3. El debate será eliminado del sistema.

### 4.5.3. Subsistema de gestión de eventos

En esta sección se detallarán los casos de uso pertenecientes al subsistema de gestión de eventos. La figura 4.4 muestra el diagrama de casos de uso de dicho subsistema.

#### 4.5.3.1. Caso de uso “ver evento”

**Descripción** Un usuario quiere ver el contenido de un evento.

**Actores** Cualquier usuario registrado o no en el sistema.

**Escenario principal**

1. El usuario accede a la vista de eventos.
2. Una vez en la vista de eventos, elige el evento del que quiere ver sus contenidos.
3. Pulsa en el título del evento que quiere ver en detalle.
4. El sistema carga la vista del evento mostrando todos sus detalles.



Figura 4.4: Diagrama de casos de uso del subsistema de gestión de eventos

#### 4.5.3.2. Caso de uso “crear evento”

**Descripción** Un usuario quiere añadir un nuevo evento al sistema.

**Actores** Administrador, usuario registrado o usuario con capacidad de acceso al API.

**Precondiciones** Haber iniciado sesión en el sistema.

##### Escenario principal

1. El usuario accede a la vista de los eventos.
2. Una vez en la vista de eventos pulsa el botón correspondiente para crear un evento nuevo.
3. Rellena el formulario con los datos requeridos.
4. Si lo desea también rellena la información opcional del formulario.
5. El usuario pulsa el botón guardar, y el sistema crea el nuevo evento.

**Escenario alternativo 1** No se han llenado todos los campos requeridos.

1. El usuario no ha llenado todos los campos requeridos para crear un nuevo evento.
2. El sistema notificará al usuario de su error y no creará el nuevo evento.
3. Se continuará desde el punto 3 del escenario principal.

#### 4.5.3.3. Caso de uso “modificar evento”

**Descripción** Un usuario quiere modificar un evento del sistema.

**Actores** Administrador, usuario registrado o usuario con capacidad de acceso al API.

**Precondiciones** Haber iniciado sesión en el sistema.

**Escenario principal**

1. El usuario accede a la vista de los eventos..
2. Una vez en la vista de eventos localiza el evento que quiere modificar.
3. Cuando ha accedido al evento pulsa el botón correspondiente para modificar su contenido.
4. Rellena toda la información requerida en el formulario.
5. Si lo desea también rellena la información opcional del formulario.
6. El usuario pulsa el botón guardar y el sistema modifica la información del evento.

**Escenario alternativo 1** No se han llenado todos los campos requeridos.

1. El usuario no ha llenado todos los campos requeridos para modificar el evento.
2. El sistema notificará al usuario de su error y no modificará los datos del evento.
3. Se continuará desde el punto 4 del escenario principal.

**4.5.3.4. Caso de uso “eliminar evento”**

**Descripción** Un administrador quiere eliminar un evento del sistema.

**Actores** El administrador del sistema.

**Precondiciones** Haber iniciado sesión en el sistema.

**Escenario principal**

1. El administrador accede a la vista de administración.
2. Localiza el evento que desea eliminar y pulsa el botón correspondiente.
3. El evento será eliminado del sistema.

**4.5.4. Subsistema de gestión de noticias**

En esta sección se detallarán los casos de uso pertenecientes al subsistema de gestión de noticias. La figura 4.5 muestra el diagrama de casos de uso de dicho subsistema.

**4.5.4.1. Caso de uso “ver noticia”**

**Descripción** Un usuario quiere ver el contenido de una noticia.

**Actores** Cualquier usuario registrado o no en el sistema.

**Escenario principal**

1. El usuario pulsa en el botón que carga la vista de las noticias.
2. Una vez en la vista de noticias pulsa en el título de la noticia que quiere ver en detalle.
3. El sistema carga la vista de la noticia mostrando todos sus detalles.

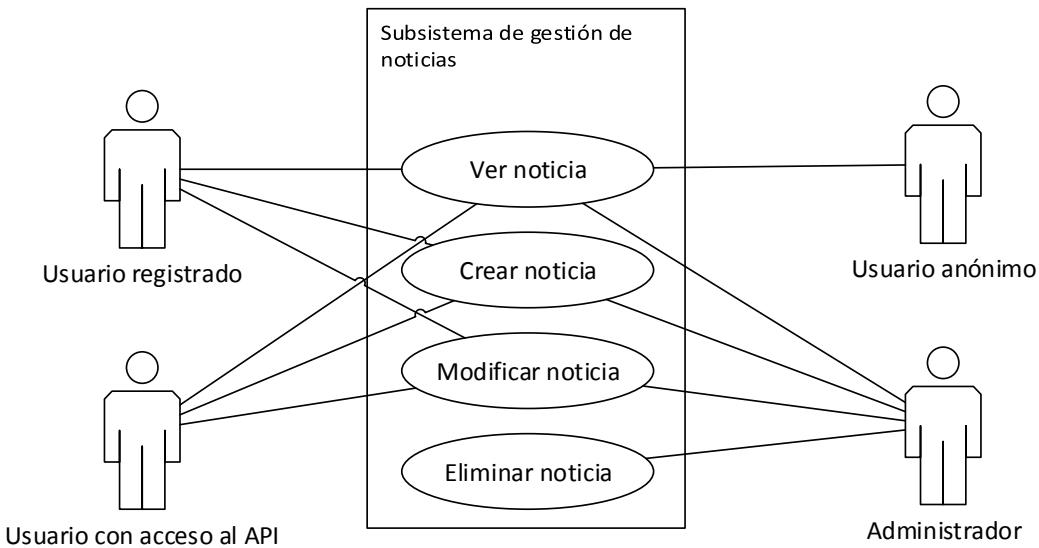


Figura 4.5: Diagrama de casos de uso del subsistema de gestión de noticias

#### 4.5.4.2. Caso de uso “crear noticia”

**Descripción** Un usuario quiere añadir una nueva noticia al sistema.

**Actores** Administrador, usuario registrado o usuario con capacidad de acceso al API.

**Precondiciones** Haber iniciado sesión en el sistema.

##### Escenario principal

1. El usuario pulsa en el botón que carga la vista de las noticias.
2. Una vez en la vista de las noticias pulsa el botón correspondiente para crear una noticia nueva.
3. Rellena el formulario con los datos requeridos.
4. Si lo desea también rellena la información opcional del formulario.
5. El usuario pulsa el botón guardar.
6. El sistema crea la nueva noticia.

**Escenario alternativo 1** No se han llenado todos los campos requeridos.

1. El usuario no ha llenado todos los campos requeridos para crear una nueva noticia.
2. El sistema notificará al usuario de su error y no creará la nueva noticia.
3. Se continuará desde el punto 3 del escenario principal.

#### 4.5.4.3. Caso de uso “modificar noticia”

**Descripción** Un usuario quiere modificar una noticia del sistema.

**Actores** Administrador, usuario registrado o usuario con capacidad de acceso al API.

**Precondiciones** Haber iniciado sesión en el sistema.

#### **Escenario principal**

1. El usuario accede a la vista de las noticias.
2. Una vez en la vista de noticias localiza la noticia que quiere modificar.
3. Cuando ha accedido a la noticia pulsa el botón correspondiente para modificar su contenido.
4. Rellena toda la información requerida en el formulario.
5. Si lo desea también rellena la información opcional del formulario.
6. El usuario pulsa el botón guardar.
7. El sistema actualiza la información de la noticia editada.

**Escenario alternativo 1** No se han llenado todos los campos requeridos.

1. El usuario no ha llenado todos los campos requeridos para modificar la noticia.
2. El sistema notificará al usuario de su error y no modificará los datos de la noticia.
3. Se continuará desde el punto 4 del escenario principal.

#### **4.5.4.4. Caso de uso “eliminar noticia”**

**Descripción** Un administrador quiere eliminar una noticia del sistema.

**Actores** El administrador del sistema.

**Precondiciones** Haber iniciado sesión en el sistema.

#### **Escenario principal**

1. El administrador accede a la vista de noticias.
2. Accede a la noticia que desea eliminar.
3. Una vez en la vista de la noticia pulsa el botón correspondiente para su eliminación.
4. La noticia será eliminada del sistema.

#### **4.5.5. Subsistema de gestión del blog**

En esta sección se detallarán los casos de uso y escenarios pertenecientes al subsistema de gestión de entradas del blog. La figura 4.6 muestra el diagrama de casos de uso del subsistema de gestión de debates.

##### **4.5.5.1. Caso de uso “ver entrada”**

**Descripción** El usuario quiere ver el contenido de una entrada del blog.

**Actores** Cualquier rol de usuario.

#### **Escenario principal**

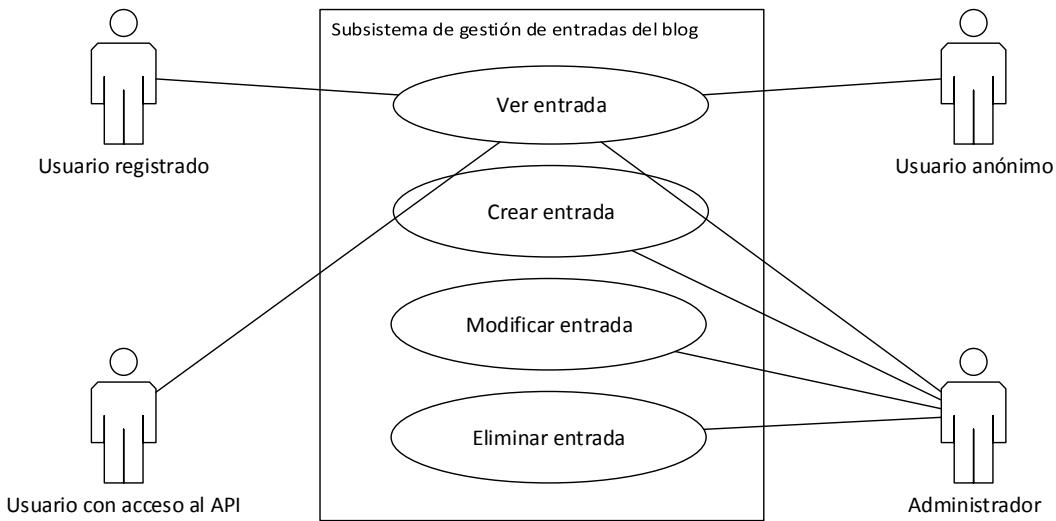


Figura 4.6: Diagrama de casos de uso del subsistema de gestión de entradas del blog

1. El usuario accede al blog del Land Portal.
2. Una vez en el blog, selecciona la entrada que quiere ver en detalle.
3. El sistema carga la vista de la entrada mostrando tanto su contenido como sus comentarios.

#### 4.5.5.2. Caso de uso “crear entrada”

**Descripción** El administrador quiere crear una nueva entrada en el blog del Land Portal.

**Actores** Administrador.

**Precondiciones** Haber iniciado sesión en el sistema.

##### Escenario principal

1. El administrador accede al blog del Land Portal.
2. Una vez en el blog, el administrador pulsa el botón para crear una nueva entrada.
3. Rellena los campos requeridos en el formulario de creación de la entrada.
4. Rellena los campos opcionales que considere necesarios en el formulario de creación de la entrada.
5. Pulsa el botón para guardar la entrada.
6. El sistema guarda la entrada y la hace visible para todos los usuarios en el blog.

**Escenario alternativo 1** El administrador no ha rellenado todos los campos requeridos.

1. El sistema notificará al usuario de que faltan campos por llenar.
2. Se continuará desde el punto 3 del escenario principal.

#### 4.5.5.3. Caso de uso “modificar entrada”

**Descripción** El administrador quiere modificar el contenido de una entrada del blog.

**Actores** El administrador del sistema.

**Precondiciones** Haber iniciado sesión en el sistema.

##### Escenario principal

1. El administrador accede al blog del Land Portal.
2. Una vez en el blog, el administrador selecciona la entrada que quiere modificar.
3. En la entrada correspondiente, el administrador pulsa el botón para editar su contenido.
4. Rellena los campos requeridos en el formulario de modificación de la entrada.
5. Rellena los campos opcionales que considere necesarios en el formulario de modificación de la entrada.
6. Pulsa el botón para guardar los cambios
7. El sistema actualiza los contenidos de la entrada del blog.

**Escenario alternativo 1** No se rellenan todos los campos requeridos

1. El sistema informa al usuario del error y no modifica la información almacenada.
2. Se continúa desde el punto 4 del escenario principal.

#### 4.5.5.4. Caso de uso “eliminar entrada”

**Descripción** El administrador quiere eliminar una entrada del blog junto con todos sus comentarios.

**Actores** El administrador del sistema.

**Precondiciones** Haber iniciado sesión en el sistema.

##### Escenario principal

1. El administrador accede a la vista de administración.
2. En la sección de contenidos localiza la entrada y pulsa el botón correspondiente para eliminarla.
3. La entrada y sus comentarios serán eliminados del sistema.

#### 4.5.6. Subsistema de gestión de comentarios

En esta sección se detallarán los casos de uso y escenarios pertenecientes al subsistema de gestión de comentarios. La figura 4.7 muestra el diagrama de casos de uso de dicho subsistema.

##### 4.5.6.1. Caso de uso “crear comentario”

**Descripción** El usuario quiere crear un nuevo comentario.



Figura 4.7: Diagrama de casos de uso del subsistema de gestión de comentarios

**Actores** Administrador, usuario registrado o usuario con capacidad de acceso al API.

**Precondiciones** Haber iniciado sesión en el sistema.

#### Escenario principal

1. El usuario accede a la entrada del blog o al debate en el que quiere crear el comentario.
2. El usuario escribe su comentario.
3. Pulsa el botón para guardar su comentario.
4. El sistema guarda el comentario y lo hace visible al resto de usuarios.

**Escenario alternativo 1** Comentario en un debate cerrado.

1. El usuario accede al debate en el que quiere crear el comentario.
2. El sistema no mostrará opciones para que el usuario cree un comentario.

#### 4.5.6.2. Caso de uso “modificar comentario”

**Descripción** El administrador desea modificar el contenido de un comentario realizado por un usuario.

**Actores** Administrador.

**Precondiciones** Haber iniciado sesión en el sistema.

#### Escenario principal

1. El usuario accede a la entrada del blog o al debate en el que se encuentra el comentario a modificar.
2. El administrador localiza el comentario y pulsa el botón modificar.
3. El administrador introduce el texto que crea conveniente y pulsa el botón guardar.
4. El sistema actualiza el contenido del comentario modificado.

#### 4.5.6.3. Caso de uso “eliminar comentario”

**Descripción** El administrador desea eliminar un comentario realizado por un usuario.

**Actores** Administrador.

**Precondiciones** Haber iniciado sesión en el sistema.

##### Escenario principal

1. El usuario accede a la entrada del blog o al debate en el que se encuentra el comentario a eliminar.
2. El administrador localiza el comentario y pulsa el botón eliminar.
3. El sistema elimina el comentario y deja de mostrarlo.

#### 4.5.7. Subsistema de gestión de organizaciones

En esta sección se detallarán los casos de uso pertenecientes al subsistema de gestión de organizaciones. La figura 4.8 muestra el diagrama de casos de uso de dicho subsistema.



Figura 4.8: Diagrama de casos de uso del subsistema de gestión de organizaciones

##### 4.5.7.1. Caso de uso “ver organización”

**Descripción** Un usuario quiere ver la información sobre una organización.

**Actores** Cualquier usuario registrado o no en el sistema.

##### Escenario principal

1. El usuario pulsa en el botón que carga la vista de las organizaciones.
2. Una vez en la vista de organizaciones pulsa en el nombre o la imagen de la organización que quiere ver en detalle.

3. El sistema carga la vista de la organización mostrando todos sus detalles.

#### 4.5.7.2. Caso de uso “crear organización”

**Descripción** Un administrador quiere añadir una nueva organización al sistema.

**Actores** El administrador del sistema.

**Precondiciones** Haber iniciado sesión en el sistema.

##### Escenario principal

1. El usuario pulsa en el botón que carga la vista de las organizaciones.
2. Una vez en la vista de las organizaciones pulsa el botón correspondiente para crear una organización nueva.
3. Rellena el formulario con los datos requeridos.
4. Si lo desea también rellena la información opcional del formulario.
5. El administrador pulsa el botón guardar.
6. El sistema crea la nueva noticia.

**Escenario alternativo 1** No se han llenado todos los campos requeridos.

1. El administrador no ha llenado todos los campos requeridos para crear una nueva organización.
2. El sistema notificará al administrador de su error y no creará la nueva organización.
3. Se continuará desde el punto 3 del escenario principal.

#### 4.5.7.3. Caso de uso “modificar organización”

**Descripción** Un administrador quiere modificar una organización existente en el sistema.

**Actores** El administrador del sistema.

**Precondiciones** Haber iniciado sesión en el sistema.

##### Escenario principal

1. El usuario accede a la vista de las organizaciones.
2. Una vez en la vista de noticias localiza la organización que quiere modificar.
3. Cuando ha accedido a la organización pulsa el botón correspondiente para modificar su contenido.
4. Rellena toda la información requerida en el formulario.
5. Si lo desea también rellena la información opcional del formulario.
6. El administrador pulsa el botón guardar.
7. El sistema actualiza la información de la organización editada.

**Escenario alternativo 1** No se han llenado todos los campos requeridos.

1. El administrador no ha llenado todos los campos requeridos para modificar la organización.
2. El sistema notificará al usuario de su error y no modificará los datos de la organización.
3. Se continuará desde el punto 4 del escenario principal.

#### 4.5.7.4. Caso de uso “eliminar organización”

**Descripción** Un administrador quiere eliminar una organización del sistema.

**Actores** El administrador del sistema.

**Precondiciones** Haber iniciado sesión en el sistema.

#### Escenario principal

1. El administrador accede a la vista de las organizaciones.
2. Accede a la organización que desea eliminar.
3. Una vez en la vista de la organización pulsa el botón correspondiente para su eliminación.
4. La organización será eliminada del sistema.

#### 4.5.8. Subsistema de búsqueda

En esta sección se detallarán los casos de uso y escenarios pertenecientes al subsistema de búsqueda. La figura 4.9 muestra el diagrama de casos de uso de dicho subsistema.

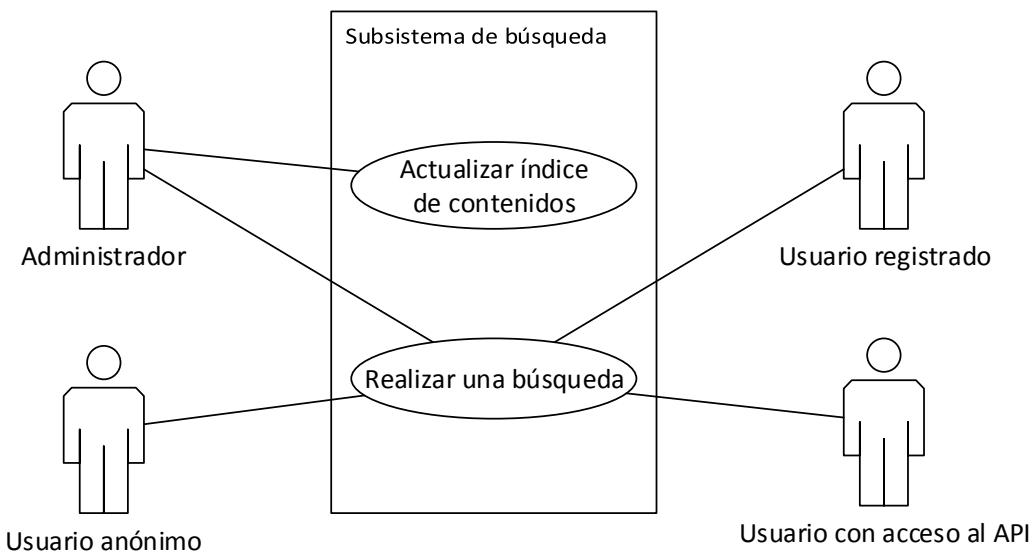


Figura 4.9: Diagrama de casos de uso del subsistema de búsqueda

#### 4.5.8.1. Caso de uso “realizar una búsqueda”

**Descripción** Un usuario busca información en el sistema.

**Actores** Cualquier rol de usuario registrado o no en el sistema.

#### Escenario principal

1. El usuario introduce un texto en el formulario de búsqueda
2. El usuario pulsa el botón de buscar
3. El sistema devuelve los resultados correspondientes

#### 4.5.8.2. Caso de uso “actualizar índice de contenidos”

**Descripción** El administrador actualiza el índice de contenidos del subsistema de búsqueda.

**Actores** El administrador del sistema.

#### Escenario principal

1. El administrador accede al panel de administración.
2. Una vez en el panel de administración, accede a las opciones de la búsqueda
3. El administrador pulsa el botón correspondiente para realizar la regeneración del índice de contenidos.
4. El sistema actualiza el índice de contenidos incluyendo los nuevos contenidos y eliminando los contenidos que ya no existan.

#### 4.5.9. Subsistema de gestión de datos

En esta sección se detallarán los casos de uso pertenecientes al subsistema de datos. La figura 4.10 muestra el diagrama de casos de uso de dicho subsistema.

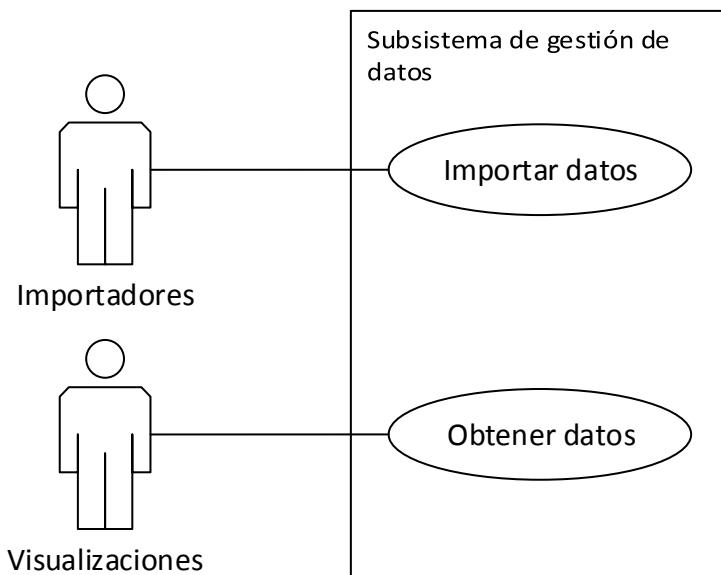


Figura 4.10: Diagrama de casos de uso del subsistema de datos

#### 4.5.9.1. Caso de uso “importar datos”

**Descripción** Un importador quiere incluir nueva información en el sistema.

**Actores** Un importador.

##### Escenario principal

1. El importador envía una petición POST al subsistema de datos incluyendo un fichero XML conforme a un XML Schema definido y con los datos a incluir en el sistema.
2. El subsistema de datos comprueba si la dirección de origen de la petición está en la lista blanca.
3. El subsistema de datos procesa la petición convirtiendo la información del fichero XML a un modelo propio.
4. El subsistema de datos persiste en la base de datos el modelo creado anteriormente.
5. El subsistema envía una respuesta 200 (*OK*) al importador.

**Escenario alternativo 1** Los datos enviados por el importador no se ajustan al XML Schema.

1. El subsistema de datos no insertará ningún dato en la base de datos.
2. El subsistema de datos envía una respuesta de error al importador.

**Escenario alternativo 2** El verbo utilizado por el importador no es el correcto (GET, PUT, etc.).

1. El subsistema de datos no insertará ningún dato en la base de datos.
2. El subsistema de datos envía una respuesta de error 405 (*Method not Allowed*) al importador.

**Escenario alternativo 3** La petición realizada por el importador no incluye el fichero XML con los datos.

1. El subsistema de datos no insertará ningún dato en la base de datos.
2. El subsistema de datos envía una respuesta de error 400 (*Bad Request*) al importador.

**Escenario alternativo 4** La petición proviene de un origen no confiable

1. El subsistema de datos aborta la petición con un código de error 403 (*Forbidden*).

#### 4.5.9.2. Caso de uso “obtener datos”

**Descripción** Una visualización pide datos para presentarlos al usuario de forma atractiva.

**Actores** Visualizaciones.

##### Escenario principal

1. La visualización lanza una petición GET al subsistema de datos.
2. El subsistema de datos devuelve un fichero JSON con los datos correspondientes a la petición realizada.
3. La visualización procesa los datos recibidos para presentarlos de forma visual.

**Escenario alternativo 1** La petición de la visualización es errónea.

1. La visualización lanza una petición al subsistema de datos.
2. El subsistema de datos responde con un código de error HTTP.

## 4.6. Clases preliminares del modelo

En esta sección se realizará un análisis preliminar de las clases del sistema. El objetivo de este análisis será mostrar una primera versión de los datos del modelo del sistema y de sus relaciones.

Para facilitar la lectura, se dividirá el análisis de clases en dos partes: una parte contendrá las clases pertenecientes a la zona social y la otra parte contendrá las clases pertenecientes a la zona de datos.

### 4.6.1. Análisis preliminar de clases de la zona de datos

A continuación se realizará el análisis de las clases pertenecientes a la zona de datos. Como se ha explicado anteriormente en la sección “Identificación de subsistemas” perteneciente al capítulo 4, la zona de datos incluye únicamente el subsistema de datos, que será el encargado de introducir datos en el sistema y devolverlos cuando sean pedidos por las visualizaciones.

La figura 4.11 muestra el diagrama de las clases preliminares de la zona datos.



Figura 4.11: Diagrama de clases preliminares de la zona de datos

#### 4.6.1.1. Descripción de las clases

**Región** Representa un país o continente del mundo. Sus atributos son los siguientes:

- **Nombre** Será el nombre de la región, por ejemplo “Europa”, “África” o “España”.
- **Código UN** Será el código numérico asignado por la División Estadística de las Naciones Unidas<sup>1</sup> para la región.

**País** Representa a un país del mundo sobre el que se almacenarán observaciones. Además de los atributos de una región, sus atributos serán:

- **ISO2** Será el código alfabético de dos letras asignado por la división estadística de las Naciones Unidas<sup>2</sup> para el país.
- **ISO3** Será el código alfabético de tres letras asignado por la división estadística de las Naciones Unidas<sup>3</sup> para el país. El código ISO3 consigue una mejor asociación con los nombres de los países que el código ISO2.
- **FAO URI** Será la URI única del país en la Ontología Geopolítica de la Organización para la Alimentación y la Agricultura de las Naciones Unidas<sup>4</sup>.
- **Continente** Representa el continente del que el país forma parte. Cada país forma parte de un continente, pero un continente puede estar compuesto de muchos países.

**Observación** Representa una medición realizada en un tiempo concreto para un país determinado.

Las observaciones son utilizadas por las visualizaciones para presentar los datos de forma visual a los usuarios. Sus atributos serán los siguientes:

- **Valor** Será el valor concreto de la observación.
- **Tiempo de referencia** Será el tiempo al que hace referencia la observación. Por ejemplo, el tiempo de referencia puede ser un año, un mes, un intervalo de años, etc.
- **Computación** Indica el tipo de computación de la que proviene la observación. Por ejemplo indica si el valor de la observación es único o un agregado de los valores durante un determinado periodo de tiempo.
- **Área de referencia** Representa el país al que la observación hace referencia. Cada observación hace referencia a un país, pero un país puede tener muchas observaciones que le hagan referencia.
- **Indicador** Representa el indicador al que pertenece la observación. Cada observación pertenece a un indicador, pero un indicador puede contener multitud de observaciones.
- **Conjunto de datos** Representa el conjunto de datos del que proviene la observación. Cada observación proviene de un único conjunto de datos, pero cada conjunto de datos incluye múltiples observaciones.

**Indicador** Representa un elemento sobre el que se realizan observaciones. Sus atributos son los siguientes:

- **Nombre** Es un nombre corto para un indicador.
- **Descripción** Es una descripción sobre lo que representa el indicador. Por ejemplo

<sup>1</sup>Este código recibe el nombre formal de “ISO 3166-1 numeric”. En [Div91] y [Sta] puede encontrarse más información sobre estos códigos.

<sup>2</sup>Este código recibe el nombre formal de “ISO 3166-1 alpha-2”. Se utiliza para hacer representar países, territorios independientes o zonas geográficas de interés especial. En [Sta] puede encontrarse más información sobre estos códigos

<sup>3</sup>Este código recibe el nombre formal de “ISO 3166-1 alpha-3”. Se utiliza para hacer representar países, territorios independientes o zonas geográficas de interés especial. En [Sta] puede encontrarse más información sobre estos códigos

<sup>4</sup>La información sobre esta ontología se encuentra disponible en [FU]

“Índice de Desarrollo Humano”<sup>5</sup>.

- **Unidad de medida** Representa la unidad de medida de las observaciones del indicador. Por ejemplo el indicador “propiedad de las tierras por parte de mujeres” podría tener una unidad de medida en porcentaje.
- **Última actualización** Indica la fecha de la última actualización del indicador o alguna de sus observaciones en el sistema.
- **Tendencia preferible** Representa si es preferible que el valor del indicador crezca o disminuya a lo largo del tiempo. Por ejemplo para el indicador “mortalidad infantil” la tendencia preferible será que disminuya.
- **Tópico** Representa el tópico (categoría) en la que se incluye el indicador. Cada indicador pertenece a un tópico, pero un tópico puede tener muchos indicadores relacionados.
- **Conjuntos de datos** Representa los conjuntos de datos en los que el indicador está presente. Un indicador puede estar presente en muchos conjuntos de datos y, al mismo tiempo, un conjunto de datos puede contener muchos indicadores diferentes.

**Tópico** Representa un categoría de indicadores que tratan sobre un mismo tema. Sus atributos son los siguientes:

- **Nombre** Es el nombre del tópico. Por ejemplo “Tierra y género” o “Propiedad de la tierra”.

**Conjunto de datos** Representa un conjunto de datos que se inserta en el sistema. Sus atributos son los siguientes:

- **Frecuencia** Indica la frecuencia con la que se actualiza el conjunto de datos. Algunas frecuencias de ejemplo pueden ser: “anual”, “trimestral”, etc.
- **Fuente** Representa la fuente de la que proviene el conjunto de datos. Cada conjunto de datos pertenece a una fuente, pero cada fuente puede contener varios conjuntos de datos.
- **Licencia** Representa la licencia bajo la que se publica el conjunto de datos. Cada conjunto de datos se publica bajo una licencia, pero cada licencia puede tener varios conjuntos de datos.

**Licencia** Representa una licencia bajo la que se publica un conjunto de datos. Sus atributos son los siguientes:

- **Nombre** Es el nombre corto de la licencia, Por ejemplo “CC BY 2.0”.
- **Descripción** Es una descripción larga sobre el funcionamiento de la licencia.
- **URL** Será la URL bajo la que se encuentra disponible la información de la licencia Por ejemplo <https://creativecommons.org/licenses/by/2.0/> para la licencia “CC BY 2.0”.
- **Republicar** Indica si la licencia permite republicar los contenidos o no.

**Fuente de datos** Representa una fuente de datos de la que se extraen conjuntos de datos. Sus atributos son los siguientes:

- **Nombre** Es el nombre de la fuente de datos. Por ejemplo “Indicadores sobre el hambre”.

**Organización** Representa una organización que provee fuentes y conjuntos de datos que se im-

---

<sup>5</sup>El Índice de Desarrollo Humano (HDI) es un indicador real mantenido por el United Nations Development Programme (UNDP) y que se encontrará en la zona de datos del portal final

portan en el sistema:

- **Nombre** Es el nombre de la organización. Por ejemplo “The Statistics Division of the FAO”<sup>6</sup>.
- **URL** Es la URL del sitio web de la organización.

#### 4.6.2. Análisis preliminar de clases de la zona social

A continuación se realizará el análisis preliminar de las clases pertenecientes a la zona social del sistema. Como se ha explicado anteriormente en la sección “Identificación de subsistemas” perteneciente al capítulo 4, la zona social engloba los subsistemas de:

- gestión de usuarios
- gestión de debates
- gestión de eventos
- gestión de noticias
- gestión de organizaciones
- gestión del blog
- gestión de comentarios

La figura 4.12 muestra el diagrama de las clases preliminares de la zona social.

---

<sup>6</sup><http://faostat.fao.org/>

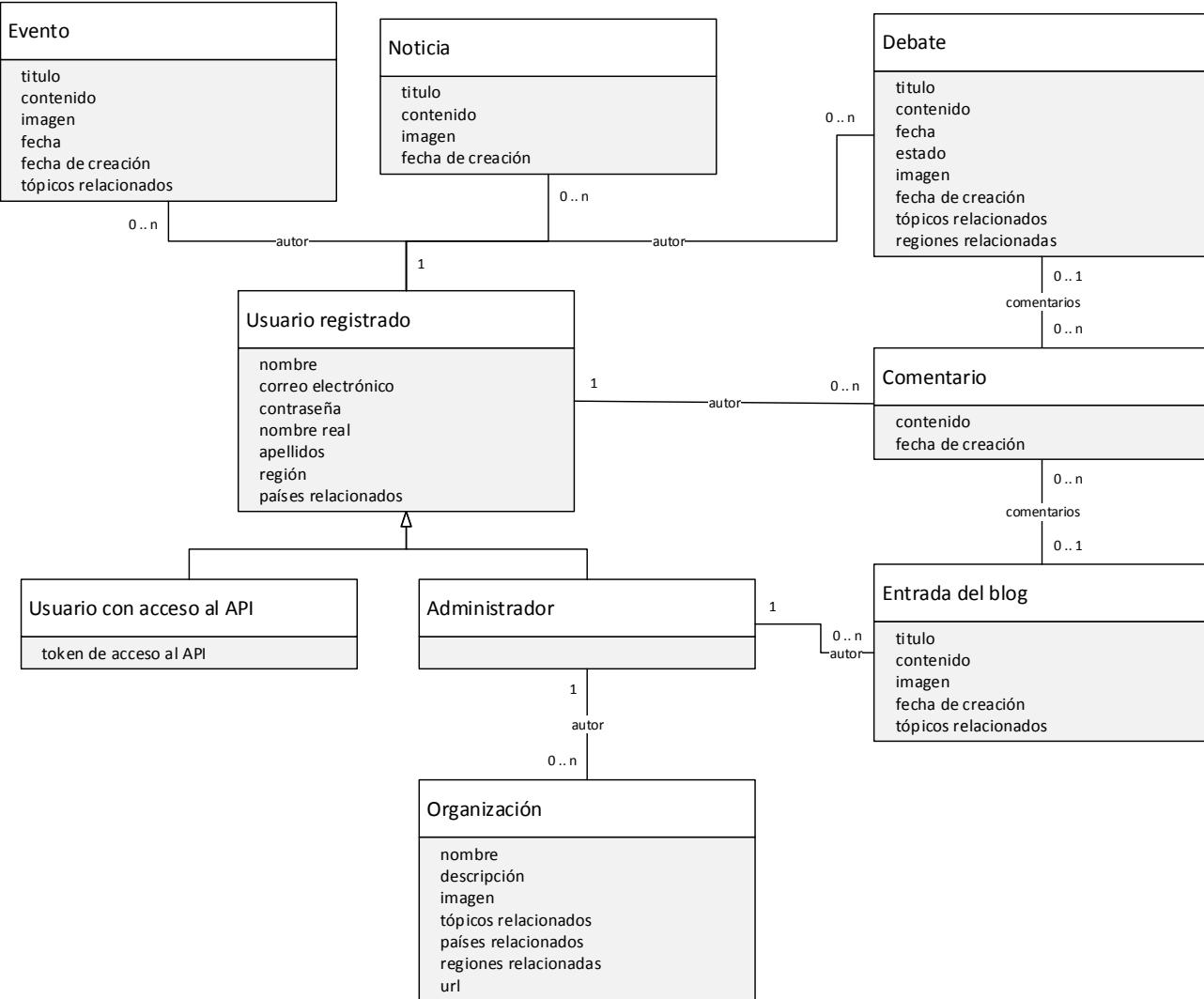


Figura 4.12: Diagrama de clases preliminares de la zona social

#### 4.6.2.1. Descripción de las clases

**Usuario registrado** Representa a los usuarios que tienen una cuenta de usuario y han iniciado sesión en el portal. Puede crear noticias, eventos, debates y realizar comentarios. Sus atributos son los siguientes:

- **Nombre** Será el nombre con el que el sistema identifica a cada usuario. El nombre de usuario debe ser único.
- **Correo electrónico** Será la dirección a la que el sistema enviará los correos electrónicos de contacto, por ejemplo: cuando el usuario solicita una nueva contraseña. El correo electrónico debe ser único.
- **Contraseña** Será utilizada para autenticar la identidad del usuario al iniciar sesión en el sistema o al modificar la información de su perfil.
- **Nombre real** Será el nombre real del usuario, podrá ser visto por otros usuarios al visitar su perfil.
- **Apellidos** Serán los apellidos reales del usuario, podrán ser vistos por otros usuarios al visitar su perfil.
- **Continente** Será el continente en el que se encuentra el usuario. Este atributo no será obligatorio y podrá estar vacío.
- **Países relacionados** Serán los países en los que el usuario tiene algún tipo de interés. Este atributo no será obligatorio y podrá estar vacío.

**Usuario con acceso al API** Representa a los usuarios registrados que tienen capacidad de acceso al API del sistema. Tienen las mismas capacidades que cualquier usuario registrado. Además de los atributos de cualquier usuario registrado también tendrán:

- **Token** Será la clave con la que el usuario accede al API. Esta clave será secreta y no podrá ser vista por ningún otro usuario.

**Administrador** Representa a los usuarios registrados con privilegios de administración en el sistema. Tendrán los mismos atributos que cualquier otro usuario registrado pero podrán crear, modificar y eliminar cualquier contenido o comentario.

**Noticia** Representa una noticia de interés en la que la fecha no es importante. Las noticias podrán ser creadas por cualquier usuario registrado. Sus atributos serán los siguientes:

- **Título** Es el título de la noticia.
- **Contenido** Es el contenido de la noticia.
- **Imagen** Es una imagen relacionada con el contenido de la noticia. Este atributo no será obligatorio y podrá estar vacío.
- **Fecha de creación** Es la fecha en la que la noticia fue creada. Este campo es almacenado automáticamente por el sistema.
- **Autor** Es el usuario registrado que crea la noticia. Cada noticia tendrá un único autor, pero cada usuario registrado podrá ser autor de múltiples noticias.

**Evento** Representa una situación o noticia que tendrá lugar en un determinado momento del tiempo. Los eventos podrán ser creados por cualquier usuario registrado. Sus atributos serán los siguientes:

- **Título** Es el título del evento.

- **Contenido** Es el contenido del evento, donde se explica toda la información necesaria.
- **Imagen** Es una imagen relacionada con el contenido del evento. Este atributo no será obligatorio y podrá estar vacío.
- **Fecha** Será la fecha en la que el evento tendrá lugar.
- **Fecha de creación** Es la fecha en la que el evento fue creado. Este campo es almacenado automáticamente por el sistema.
- **Tópicos relacionados** Serán los tópicos relacionados con el contenido del evento. Los tópicos servirán como metainformación sobre el contenido y ayudarán a enriquecer los resultados de la búsqueda.
- **Autor** Es el usuario registrado que crea el evento. Cada evento tendrá un único autor, pero cada usuario registrado podrá ser autor de múltiples eventos.

**Debate** Representa un tema u opinión en el que se quiere permitir la participación de los usuarios. Sus atributos serán los siguientes:

- **Título** Es el título del debate.
- **Contenido** Es el contenido del debate, donde se expone el tema a tratar por los usuarios.
- **Fecha** Será la fecha o periodo de fechas en las que el debate tendrá lugar.
- **Estado** Es el estado en el que se encuentra el debate. Los posibles estados son “abierto”, “cerrado” o “próximamente”.
- **Imagen** Es una imagen relacionada con el contenido del debate. Este atributo no será obligatorio y podrá estar vacío.
- **Fecha de creación** Es la fecha en la que el debate fue creado. Este campo es almacenado automáticamente por el sistema.
- **Tópicos relacionados** Serán los tópicos relacionados con el contenido del debate. Los tópicos servirán como metainformación sobre el contenido y ayudarán a enriquecer los resultados de la búsqueda.
- **Regiones relacionadas** Serán las regiones relacionadas con el debate o a las que este hace referencia. Al igual que los tópicos, las regiones servirán como metainformación sobre el contenido y ayudarán a enriquecer los resultados de la búsqueda.
- **Autor** Es el usuario registrado que crea el debate. Cada debate tendrá un único autor, pero cada usuario registrado podrá ser autor de múltiples debates.
- **Comentarios** Son los comentarios que los usuarios realizan en el debate. Cada debate podrá tener múltiples comentarios, pero cada comentario sólo podrá pertenecer a un debate.

**Entrada del blog** Representa una entrada en el blog oficial de Land Portal. Las entradas en el blog sólo pueden ser creadas por los administradores. Sus atributos son los siguientes:

- **Título** Es el título de la entrada.
- **Contenido** Es el contenido de la entrada.
- **Imagen** Es una imagen relacionada con el contenido de la entrada. Este atributo no será obligatorio y podrá estar vacío.
- **Fecha de creación** Es la fecha en la que la entrada fue creada. Este campo es almacenado automáticamente por el sistema.

- **Tópicos relacionados** Serán los tópicos relacionados con el contenido de la entrada. Los tópicos servirán como metainformación sobre el contenido y ayudarán a enriquecer los resultados de la búsqueda.
- **Autor** Es el administrador que crea la entrada. Cada entrada tendrá un único autor, pero cada administrador podrá ser autor de múltiples entradas.
- **Comentarios** Son los comentarios que los usuarios realizan en la entrada. Cada entrada podrá tener múltiples comentarios, pero cada comentario sólo podrá pertenecer a una entrada.

**Comentario** Representa la opinión que un usuario hace pública en un debate o una entrada del blog:

- **Contenido** Es el contenido del comentario.
- **Fecha de creación** Es la fecha en la que el comentario fue creado. Este campo es almacenado automáticamente por el sistema.
- **Autor** Es el usuario registrado que crea el comentario. Cada comentario tiene un autor, pero un usuario registrado puede ser autor de múltiples comentarios.

**Organización** Representa una organización que realiza alguna labor humanitaria o relacionada con la Tierra:

- **Nombre** Es el nombre de la organización.
- **Descripción** Es una descripción larga de la organización y sus objetivos.
- **Imagen** Es el logotipo de la organización.
- **Tópicos relacionados** Serán los tópicos que guardan alguna relación con los intereses de la organización. Los tópicos actuarán como metadatos que permitirán enriquecer los resultados de la búsqueda.
- **Países relacionados** Serán los países sobre los que la organización trabaja.
- **Regiones relacionadas** Serán las regiones sobre las que la organización tiene presencia.
- **url** Será la dirección del sitio web de la organización.

## 4.7. Análisis de interfaces de usuario

En esta sección se realizará una primera aproximación a la posible interfaz que tendrá el portal, detallando y explicando las vistas que la conforman. A continuación se mostrarán los *mockups* de la interfaz y se explicará cada uno de ellos.

### 4.7.1. Interfaz de la zona de datos

Como se ha mencionado en la sección “Definición del sistema” perteneciente al capítulo 4, la interfaz de zona de datos (o *LandBook*) queda fuera del alcance de este proyecto, por lo que no se mostrarán *mockups* para dicha sección.

### 4.7.2. Interfaz de administración

La interfaz de administración será directamente proveída por el gestor de contenidos que, como se ha mencionado en la sección “Alternativas elegidas” perteneciente al capítulo 2, será Drupal. Por esta razón no se mostrarán *mockups* pertenecientes a dicha interfaz.

### 4.7.3. Interfaz de la zona social

A continuación se mostrarán los *mockups* de la zona social (o *LandDebate*).

#### 4.7.3.1. Vista del blog del Land Portal

La figura 4.13 muestra el *mockup* de la vista que conformará el blog del nuevo Land Portal.

Esta pantalla mostrará las entradas del blog ordenadas de más a menos reciente. Las tres entradas más recientes se destacarán sobre el resto, e incluirán:

- Título de la entrada, enlazado a la vista de detalle de la propia entrada.
- Imagen de la entrada.
- Autor y fecha en la que la entrada fue añadida al blog.
- Tópicos relacionados. Al pulsar en un tópico, se accederá a una vista en la que se mostrarán todos los contenidos del portal relacionados con dicho tópico.
- Resumen del contenido de la entrada.

Las entradas anteriores tendrán menos énfasis en la vista, y sólo estarán representadas por su título y la fecha en la que han sido creadas. Además también se incluirá un paginador para recorrer las noticias anteriores y un enlace al *feed RSS*<sup>7</sup> donde también se encontrarán las últimas noticias.

#### 4.7.3.2. Vista de detalle de una entrada del blog

La figura 4.14 muestra el *mockup* de la vista de detalle de una entrada del blog. Esta vista incluirá:

- Título de la entrada
- Autor y fecha de creación de la entrada
- Tópicos relacionados. Como ya se ha explicado anteriormente, al pulsar sobre un tópico, el sistema cargará una vista con todos los contenidos relacionados con dicho tópico.
- Contenido de la entrada
- Imagen relacionada
- Botones destinados a compartir el contenido de la entrada en redes sociales como Twitter o Facebook.

En la zona inferior de la vista se incluirá la sección de comentarios. En esta sección los usuarios podrán participar y dar su opinión a cerca del tema tratado en la entrada del blog. Cada comentario

<sup>7</sup>El feed RSS permite suscribirse a las últimas entradas del blog utilizando un lector de feeds RSS como Feedly, Newsvibe o Digg Reader y obtener las actualizaciones de forma automática en el lector sin necesidad de acceder al portal.

The mockup shows the 'The Land Portal Hub' website at <http://beta.landportal.info>. The header includes a logo, navigation links (Home, Land Book, Land Debates, Land Library, Sitemap, English, Join Us), and a search bar. Below the header is a horizontal menu with tabs: Blog (selected), News, Events, Debates, Community, and Organisations.

The main content area displays a blog post titled "How land grabs in Africa could herald a new dystopian age of hunger" by Florent Leo, dated 12/05/2014. The post includes a small square image with a yellow circle containing the number 1. Below the post is a summary: "Africa is up for sale by the acre to the highest bidder. But how can rice exports from Ethiopia to Saudi Arabia be justified? Land grabs have grabbed global attention. It's on the agenda at the World Economic Forum this week, and as the trend for large land acquisitions accelerates, it has moved from being primarily a story about Middle Eastern petrodollars pouring into Africa, to a much more widely spread phenomenon affecting many parts of south-east Asia, such as the Philippines, as well as Latin America." A link to "Read more" is present.

Below the first post is another blog post titled "10 Joint Policy Recommendations" by Sabine Pallas, dated 12/05/2014. This post also features a small square image with a yellow circle containing the number 2. Its summary discusses the Maputo Declaration of the African Union and its impact on agriculture spending.

On the right side of the page, there is a sidebar with a yellow box containing numbered instructions:

- 1 - Last blog post including: image, title linked to the complete post, author, date, related topics, summary of content and link to the complete text. Author and topics.
- 2 - Metadata links act as a filtered search for organizations on the given topic.
- 3 - Most recent posts with the same information as above.
- 4 - Older post with just date and title linked to the complete post.
- 4 - Pagination for post archives and rss link

At the bottom of the page, there is a footer with links to Privacy policy, Terms & Conditions, FAQs, Contact Us, About, social media links (Follow us on Facebook and Twitter), and the Land Portal Partnership email address: hello@landportal.info.

Figura 4.13: Mockup de la vista del blog del Land Portal

mostrará su contenido, su autor y la fecha en la que fue creado. Los comentarios se mostrarán ordenados según su fecha de creación.

#### 4.7.3.3. Vista de eventos

La figura 4.15 muestra el *mockup* de la vista de eventos del nuevo Land Portal. Los dos eventos más recientes se destacarán sobre el resto e incluirán:

- Título del evento, enlazado a la vista de detalle del propio evento.
- Imagen del evento.
- Autor y fecha en la que el evento fue creado.
- Tópicos relacionados. Al pulsar en un tópico, se accederá a una vista en la que se mostrarán todos los contenidos del portal relacionados con dicho tópico.
- Resumen del contenido del evento.

De forma similar a como sucede en el blog, los eventos anteriores tendrán menos importancia en la vista, y únicamente se mostrarán con su título y la fecha en la que el evento tiene lugar. También es igual que el blog, se incluye un paginador y un enlace al feed RSS donde también se encontrarán los últimos eventos.

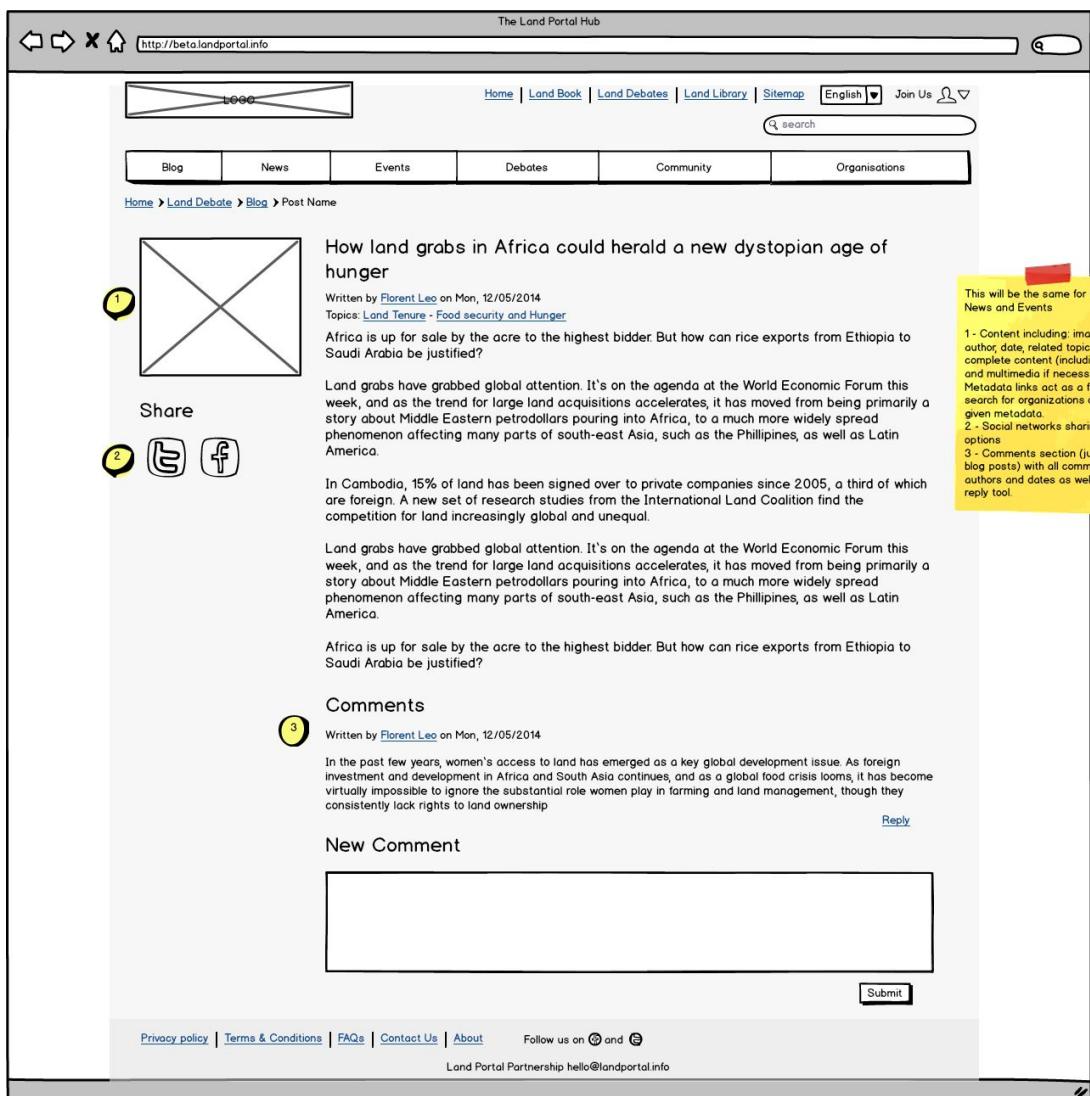


Figura 4.14: *Mockup* de la vista de detalle de una entrada del blog

Un añadido especial de esta vista es un calendario que mostrará resaltados los días en los que tiene lugar algún evento.

#### 4.7.3.4. Vista de noticias

La figura 4.16 muestra el *mockup* de la vista de noticias del nuevo Land Portal. Las cuatro noticias más recientes se destacarán sobre el resto e incluirán:

- Título de la noticia, enlazado a la vista de detalle de la propia noticia.
- Imagen de la noticia
- Resumen del contenido de la noticia.

Al igual que en la vista de eventos y del blog, las noticias anteriores sólo mostrarán su título y fecha de creación. También se incluirán, de la misma forma que las vistas anteriores, un paginador y un enlace al feed RSS con las últimas noticias.



Figura 4.15: Mockup de la vista de eventos

Una característica especial de esta vista será que en la parte izquierda de la misma se mostrarán las últimas actualizaciones de la cuenta de Twitter de Land Portal<sup>8</sup>. Para incluir este componente se utilizará la capacidad de generación de *widgets* de Twitter.

#### 4.7.3.5. Vista de debates

La figura 4.17 muestra el *mockup* de la vista de debates del nuevo Land Portal. Esta vista tiene una importancia clave dentro de la zona social, puesto que los debates serán el principal punto en el que los usuarios participen e intercambien opiniones. Los debates que no estén cerrados se destacarán sobre el resto e incluirán:

- Título del debate, enlazado a la vista de detalle del mismo.
- Autor del debate.
- Tópicos relacionados. Al pulsar en un tópico, se accederá a una vista en la que se mostrarán todos los contenidos del portal relacionados con dicho tópico.
- Resumen del contenido del debate.
- Imagen del debate.
- Fecha o fechas en las que el debate tendrá lugar.
- Estado del debate.

De forma similar a las vistas anteriores, también se incluirán los debates anteriores (en este caso los debates ya cerrados) únicamente con su título y la fecha en la que ese han creado y un enlace al *feed* RSS de la sección de debates.

<sup>8</sup>La cuenta oficial de Land Portal se puede ver en <https://twitter.com/landportal>

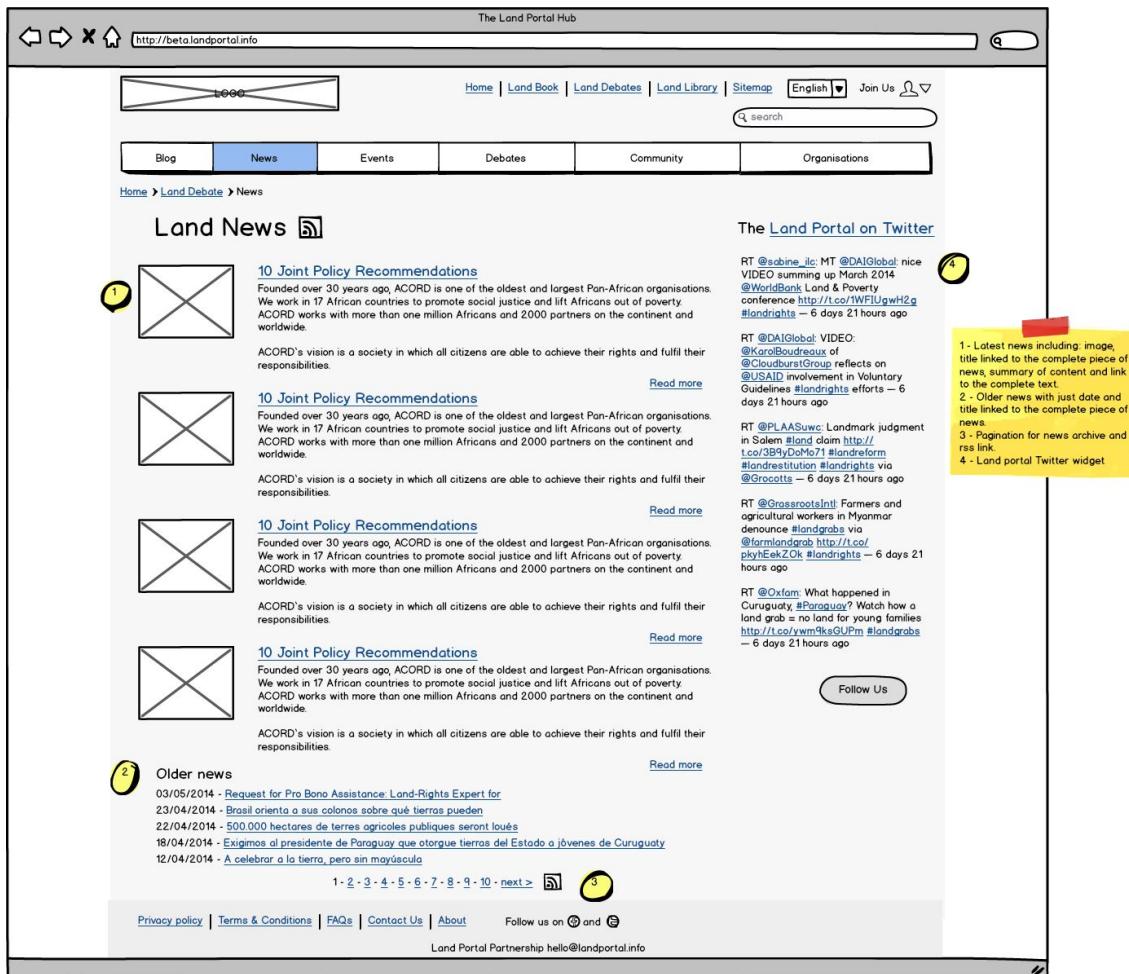


Figura 4.16: Mockup de la vista de noticias

Puesto que, como se ha dicho anteriormente, esta vista juega un papel clave fomentando la participación de los usuarios en el portal. Por ello, en la zona derecha se mostrarán los últimos comentarios que los usuarios hayan realizado en algún debate. Cada comentario incluirá:

- Fecha de creación del comentario.
- Autor del comentario.
- Título del comentario enlazado al propio comentario dentro de la vista de detalle del debate.
- Resumen del contenido del comentario.

#### 4.7.3.6. Vista de detalle de un debate

La figura 4.18 muestra el *mockup* de la vista de detalle de un debate. Al igual que la vista de debates descrita anteriormente, esta vista juega también un rol clave para fomentar la participación de los usuarios en el portal.

La vista incluirá toda la información del debate, incluyendo:

- Título del debate
- Tópicos y países relacionados

The Land Portal Hub

<http://beta.landportal.info>

Home | Land Book | Land Debates | Land Library | Sitemap | English | Join Us

Blog News Events Debates Community Organisations

Home > Land Debate > Debates

## Land Debates

**1** 3 - 10 June 2014 coming soon

**2** 25 - 30 May 2014 coming soon

**3** 18 - 28 May 2014 open

**4** 10 Joint Policy Recommendations

Facilitated by Sabine Pollos  
Topics: Land Tenure - Food security and Hunger

Founded over 30 years ago, ACORD is one of the oldest and largest Pan-African organisations. We work in 17 African countries to promote social justice and lift Africans out of poverty. ACORD works with more than one million Africans and 2000 partners on the continent and worldwide.

ACORD's vision is a society in which all citizens are able to achieve their rights and fulfil their responsibilities.

[Read more](#)

**1** 10 Joint Policy Recommendations

Facilitated by Sabine Pollos  
Topics: Land Tenure - Food security and Hunger

Founded over 30 years ago, ACORD is one of the oldest and largest Pan-African organisations. We work in 17 African countries to promote social justice and lift Africans out of poverty. ACORD works with more than one million Africans and 2000 partners on the continent and worldwide.

ACORD's vision is a society in which all citizens are able to achieve their rights and fulfil their responsibilities.

[Read more](#)

**1** 10 Joint Policy Recommendations

Facilitated by Sabine Pollos  
Topics: Land Tenure - Food security and Hunger

Founded over 30 years ago, ACORD is one of the oldest and largest Pan-African organisations. We work in 17 African countries to promote social justice and lift Africans out of poverty. ACORD works with more than one million Africans and 2000 partners on the continent and worldwide.

ACORD's vision is a society in which all citizens are able to achieve their rights and fulfil their responsibilities.

[Read more](#)

**1** Previous debates

03/05/2014 - Request for Pro Bono Assistance: Land-Rights Expert for  
23/04/2014 - Brasil orienta a sus colonos sobre qué tierras pueden  
22/04/2014 - 500.000 hectáreas de tierra agrícolas públicas serán loués  
18/04/2014 - Exigimos al presidente de Paraguay que otorgue tierras del Estado a jóvenes de Curuguaty  
12/04/2014 - A celebrar la tierra, pero sin mayúscula

[Read more](#)

**1** Privacy policy | Terms & Conditions | FAQs | Contact Us | About | Follow us on and

Land Portal Partnership hello@landportal.info

Latest comments

some hours ago in Utilisation des plates-formes en ligne pour un plus large accès aux données disponibles et pour le partage des bonnes pratiques de suivi du respect effectif des droits fonciers des femmes  
Can you tell us more about this? I'd care...

1 day ago in Land titling in Peru: what future for women's tenure security?  
Des amas de l'écran en veille AEGsavoyard...

3 days ago in Utilisation des plates-formes en ligne pour un plus large accès aux données disponibles et pour le partage des bonnes pratiques de suivi du respect effectif des droits fonciers des femmes  
What's up all, here every one is sharing...

1 week ago in How can women's land rights be secured?  
Non pas que suivantes ci-dessous pour certaines...  
Thanks for a marvelous posting! I actually...

3 weeks ago in How can women's land rights be secured?  
What's up all, here every one is sharing...

1 month ago in Utilisation des plates-formes en ligne pour un plus large accès aux données disponibles et pour le partage des bonnes pratiques de suivi du respect effectif des droits fonciers des femmes  
Thanks for a marvelous posting! I actually...

1 - Latest debates including image, title linked to the complete event description, author, related topics, date, status, summary of content and link to the complete text.  
Metadata links act as a filtered search for organizations on the given metadata.  
2 - Older debates with just date and title linked to the complete description.  
3 - Pagination for debates archive and rss link.  
4 - Latest comments on any debate, including countdown date, link to the debate and brief comment description.

Figura 4.17: Mockup de la vista de debates

- Idioma en el que tendrá lugar el debate
- Contenido del debate
- Imagen del debate
- Estado
- Autor del debate

En la parte inferior de esta vista se ubicará la sección de comentarios. El comportamiento de la sección de comentarios variará en función del estado del debate. Cuando el debate esté abierto, los usuarios podrán participar e incluir nuevos comentarios o responder a los comentarios ya existentes. Cuando el debate esté cerrado se mostrarán los comentarios existentes, pero no se permitirá añadir ningún comentario nuevo. Los comentarios se mostrarán ordenados según su fecha de creación.

Para reforzar la función social de esta vista, se incluirán también una serie de botones que permitirán compartir el debate en redes sociales como Twitter o Facebook. Al mismo tiempo, en la parte derecha de la vista se mostrarán los últimos comentarios relacionados en la red social Twitter.

Figura 4.18: *Mockup* de la vista de detalle de un debate

#### 4.7.3.7. Vista de detalle de un evento y de una noticia

Las vistas del detalle de un evento y una noticia serán similares a la vista de detalle de una entrada en el blog, el *mockup* para dicha vista puede verse en la figura 4.14.

La única diferencia de estas vistas respecto a la vista de detalle de una entrada del blog será que ni los eventos ni las noticias tendrán sección de comentarios.

#### 4.7.3.8. Vista de organizaciones

La figura 4.19 muestra el *mockup* de la vista de organizaciones. Esta vista mostrará las organizaciones existentes en el portal en forma de parrilla o *grid*, en la que cada organización mostrará su imagen y su nombre, ambos enlazados a la vista de detalle de dicha organización.

En la zona derecha de la vista se incluirán una serie de controles con los que será posible buscar organizaciones determinadas. Además, siguiendo con la tónica social del portal ya mencionada anteriormente, se mostrará un *widget* con el contenido de la comunidad de Land Portal en la red social Facebook<sup>9</sup>.

<sup>9</sup>La página oficial de Land Portal en Facebook puede verse en la siguiente dirección: <https://www.facebook.com/landportal>



Figura 4.19: Mockup de la vista de organizaciones

#### 4.7.3.9. Vista de detalle de una organización

La figura 4.20 muestra el *mockup* de la vista de detalle de una organización. Esta vista mostrará las organizaciones existentes en el portal en forma de parrilla o *grid*, en la que cada organización mostrará su imagen y su nombre, ambos enlazados a la vista de detalle de dicha organización.

Esta vista mostrará la imagen, el nombre y una descripción de la organización. Además también se incluirá un enlace a la página web de dicha organización. En la parte derecha de la vista se mostrarán los metadatos de la organización, como: el área y los países en los que opera o los tópicos relacionados. Al pulsar en cualquier término de estos metadatos el sistema cargará una vista que mostrará todo el contenido del portal relacionado con dicho término.

#### 4.7.3.10. Vista de comunidad

La vista de la comunidad será similar a la vista de organizaciones, la cual se puede ver en la figura 4.19. Las únicas diferencias respecto a la vista de organizaciones serán:

- En lugar de mostrar las organizaciones del portal, esta vista mostrará en una parrilla o *grid* los usuarios registrados en el portal.
- En la zona derecha (además del formulario de búsqueda y el *widget* de la red social Facebook) se incluirá un botón para facilitar el registro de nuevos usuarios.

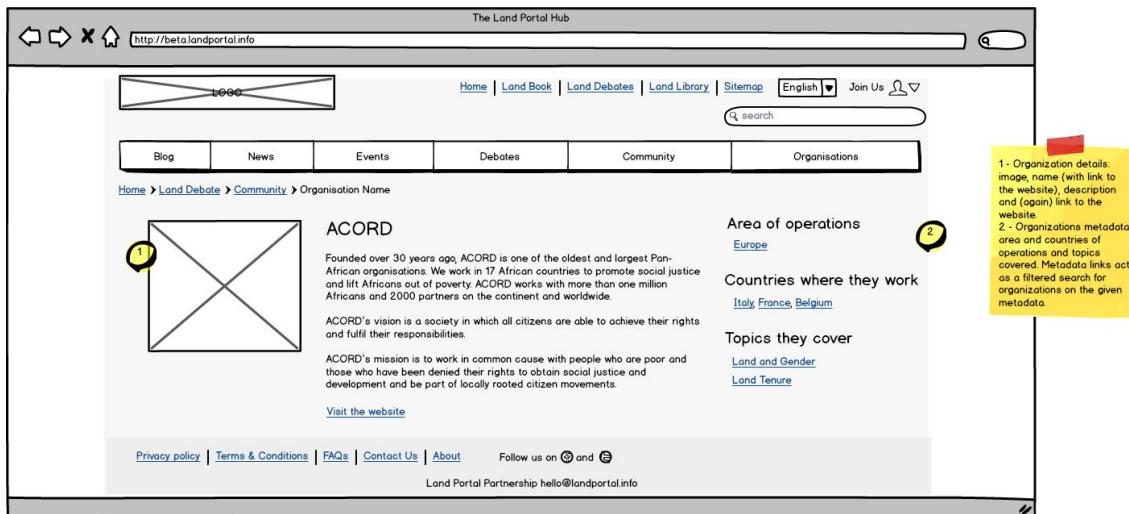


Figura 4.20: *Mockup* de la vista de detalle de una organización

#### 4.7.3.11. Vista de login

La figura 4.21 muestra el *mockup* de la vista de login. Esta vista permitirá que los usuarios inicien sesión en el portal. La vista de login contará con un formulario donde el usuario introducirá su nombre de usuario o email y su contraseña para iniciar sesión en el sistema. También existirá un enlace mediante el cual un usuario podrá solicitar una nueva contraseña en caso de haber olvidado la suya.

Además, siguiendo con la línea social del portal, se incluyen dos botones destinados a iniciar sesión en el sistema utilizando las redes sociales Twitter y Facebook.

#### 4.7.3.12. Vista de registro

La figura 4.22 muestra el *mockup* de la vista de registro. Esta vista permitirá a los usuarios crear una nueva cuenta en el sistema. La vista de registro contará con un formulario donde el usuario introducirá su nombre de usuario, su email, su nombre y apellidos y los países y continentes en los que esté interesado. Los campos obligatorios para crear una nueva cuenta de usuario están marcados con un asterisco.

El botón de registro creará la nueva cuenta en el sistema con un estado “desactivado”, a la espera de ser activada por un administrador.

#### 4.7.3.13. Vista de búsqueda

La figura 4.23 muestra el *mockup* de la vista de búsqueda. Esta vista permitirá a los usuarios buscar cualquier tipo de contenido en el portal. Al pulsar sobre cada resultado de búsqueda el usuario accederá a la vista de detalle del mismo.

Como se puede ver en la imagen, cada resultado de la búsqueda se presentará de una forma diferente y tendrá asociada una etiqueta en la que se indica el tipo de contenido al que pertenece. Al pulsar en esta etiqueta, el sistema cargará la vista correspondiente a cada tipo de contenido.



Figura 4.21: *Mockup* de la vista de login

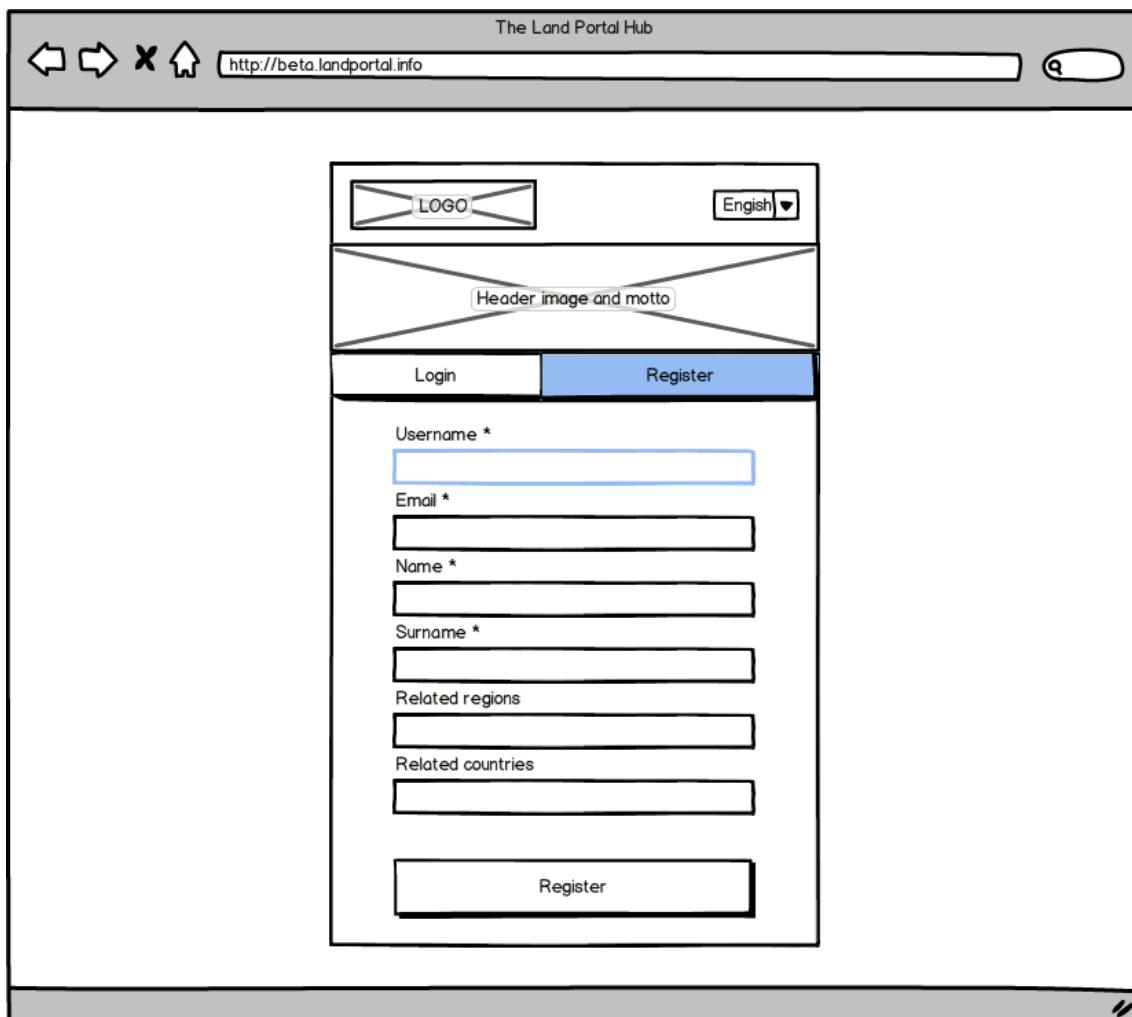


Figura 4.22: *Mockup* de la vista de registro



Figura 4.23: *Mockup* de la vista de búsqueda

## 4.8. Especificación del plan de pruebas

En esta sección se especificará el plan de pruebas que se seguirá durante el desarrollo del proyecto. Se detallará qué tipos de pruebas se realizarán, sobre qué componentes y en qué momento del desarrollo tendrán lugar.

### 4.8.1. Pruebas unitarias

Las pruebas unitarias están dirigidas a probar partes de código pequeñas y específicas. Como Martin Fowler explica en su artículo [Fowc] existen tres características fundamentales que caracterizan a las pruebas unitarias:

- Los tests se centran en una parte pequeña del sistema
- Los tests son normalmente escritos por los propios programadores utilizando alguna variante del framework *XUnit*<sup>10</sup>.
- Se espera que los tests se ejecuten de una forma rápida

Un punto clave de las pruebas unitarias es su aislamiento de otras partes del sistema. Por la estructura del subsistema de datos, cuya principal funcionalidad es insertar y extraer valores de la base de datos, sería necesario crear una gran cantidad de *mocks* que simulen el funcionamiento de la propia base de datos. Debido a esto, se ha tomado la decisión de no realizar pruebas unitarias como tal, puesto que supondría un sobreesfuerzo muy grande aislar totalmente los componentes del subsistema de la base de datos para obtener un coste muy pequeño, puesto que la propia base de datos juega un punto clave en el correcto funcionamiento de este subsistema.

Por su construcción que, como ya se ha mencionado en secciones anteriores, delega la funcionalidad principal a los mecanismos proveídos por el gestor de contenidos, las pruebas unitarias se omitirán para la zona social y los subsistemas que engloba.

### 4.8.2. Pruebas de integración

A pesar de lo mencionado anteriormente a cerca de las pruebas unitarias, existen opiniones divididas sobre lo que se considera exactamente una *unidad* en dichas pruebas. Para este proyecto, y hablando del subsistema de datos, se tomará como una *unidad* la lectura de un valor del fichero XML y su inmediata inserción en la base de datos.

Según la opinión de diferentes autores (entre ellos el propio Martin Fowler, como menciona en su artículo [Fowc]), este tipo de pruebas podrían ser consideradas pruebas unitarias. No obstante, con el objetivo de mantener la máxima fidelidad respecto a la definición de pruebas unitarias, las pruebas que se detallan a continuación han sido finalmente consideradas como pruebas de integración.

Para el desarrollo del subsistema de datos se utilizará una metodología de Desarrollo Dirigido por Pruebas (o TDD)<sup>11</sup>. Con el objetivo de facilitar esta metodología se utilizará también un servidor de integración continua.

Debido al carácter de los subsistemas de la zona social, cuya principal funcionalidad queda delegada al núcleo del CMS, las pruebas para estos componentes tendrán lugar en forma de casos de prueba que se ejecutarán manualmente durante el desarrollo.

---

<sup>10</sup>XUnit es el nombre que recibe una familia de frameworks destinados a la realización de pruebas. El primero de estos frameworks fue creado por Kent Beck para el lenguaje Smalltalk. Para más información al respecto se recomienda el artículo [Fowd].

<sup>11</sup>El desarrollo dirigido por pruebas es una metodología de desarrollo de software creada por Kent Beck en 1990. Para más información al respecto, se recomienda el libro “Test Driven Development” [Bec02]

El resultado de ambas pruebas podrá verse con mayor detalle en el capítulo 7 titulado “Desarrollo de las pruebas”.

#### 4.8.3. Pruebas de aceptación

Las pruebas de aceptación consisten en comprobar que el sistema realizado satisface las necesidades del usuario y el usuario *acepta* la solución creada. Normalmente las pruebas de aceptación son realizadas por el cliente del sistema que se construye y tienen lugar en las últimas fases del desarrollo del sistema. Las pruebas de aceptación funcionan como una verificación final de que el sistema cumple con los requisitos y funciona de una forma adecuada para los usuarios. En este proyecto las pruebas de aceptación se realizarán en forma de pruebas beta.

Las pruebas beta se realizarán durante la última fase de desarrollo. Concretamente tendrán lugar a lo largo de una semana en la que se contará con un servidor especial en el cual se desplegará la última versión estable del sistema. Este servidor estará continuamente disponible, el cliente accederá a dicho servidor, probará los diferentes aspectos del sistema y ofrecerá *feedback* a los desarrolladores con aquellos posibles fallos o cuestiones que encuentre.

Una vez recibidos los reportes será tarea del jefe de proyecto filtrarlos y separar aquellos que son fallos reales o posibles mejoras de aquellos que no lo son, y será tarea del equipo de desarrollo arreglar los fallos encontrados para obtener una versión final del sistema.

#### 4.8.4. Pruebas de rendimiento

Tal y como se ha mencionado en la sección Especificación de los requisitos no funcionales, perteneciente al capítulo 4, el punto de entrada de datos puede recibir conjuntos de datos de gran volumen. Por esto será necesario realizar pruebas específicas de rendimiento para dicho componente, con el fin de asegurar su correcto comportamiento ante diferentes entradas de datos de diferentes tamaños.

Dos puntos clave en este tipo de pruebas son: el rendimiento temporal y el rendimiento espacial. El rendimiento temporal se refiere a la velocidad con la que el componente realiza su tarea. El rendimiento espacial se refiere a la cantidad de memoria que el componente utiliza durante su funcionamiento.

El punto de entrada de datos es un componente del sistema que se ejecutará una vez cada varios meses, por lo que el rendimiento temporal del mismo no tendrá una gran importancia. El rendimiento espacial, en cambio, si tendrá una importancia alta, puesto que el consumo de memoria deberá estar controlado para evitar fallos con la llegada de grandes volúmenes de datos.

Atendiendo a la famosa cita de Donald Knuth extraída de [Knu74], las pruebas de rendimiento tendrán lugar en las últimas fases del desarrollo del punto de entrada de datos:

*“Premature optimization is the root of all evil (or at least most of it) in programming.”*

# Capítulo 5

## Diseño del sistema

En este capítulo se diseñará la forma en que se construirá el sistema, basándose en lo explicado anteriormente en el capítulo de “Análisis”.

### 5.1. Arquitectura del sistema

En esta sección se explicará en detalle el diseño estructural del sistema. Cada uno de los componentes a detallar constará de:

- Un diagrama que muestre de forma gráfica el diseño del componente y sus subcomponentes.
- Una explicación textual de los diferentes subcomponentes y las tareas que realizan.
- Una explicación textual sobre cómo se relacionan los subcomponentes entre sí.
- Descripción de forma detallada las interfaces y puertos utilizadas por los subcomponentes. Las interfaces y puertos se dividirán en tablas por componentes para hacer más sencilla su lectura. Cada entrada constará de los siguientes apartados:
  - **Interfaz.** Será el nombre que recibe el puerto/interfaz en el diagrama de componentes.
  - **Tipo.** El tipo de la interfaz/puerto será *proveído* cuando se exponga una interfaz para su uso por parte de otro componente, o *requerido* cuando se utilice una interfaz expuesta por otro componente.
  - **Tecnología.** Indica el tipo de tecnología de la interfaz/puerto.
  - **Propiedades.** Explica brevemente el objetivo de la interfaz/puerto.

Es importante destacar que esta sección se centrará en el subsistema de datos puesto que, como ya se ha mencionado en la sección Identificación de subsistemas perteneciente al capítulo 4, los subsistemas pertenecientes a la zona social (subsistema de gestión de usuarios, eventos, noticias, debates, entradas del blog y comentarios) delegan su funcionalidad en componentes ya implementados por el gestor de contenidos. De la misma forma, el subsistema de búsqueda delega su funcionalidad principal en el servidor de búsqueda.

### 5.1.1. Vista del sistema

A continuación se detallará la estructura completa del sistema que forma el nuevo *Land Portal*. A pesar de que varios de los componentes que se verán aquí quedan fuera del ámbito de este proyecto, se ha decidido incluir esta vista para proveer de un contexto que permita al lector comprender en qué parte del sistema se situarán las vistas posteriores.

#### 5.1.1.1. Diagrama de componentes

La figura 5.1 muestra el diagrama de todos los componentes que conforman el nuevo *Land Portal*.

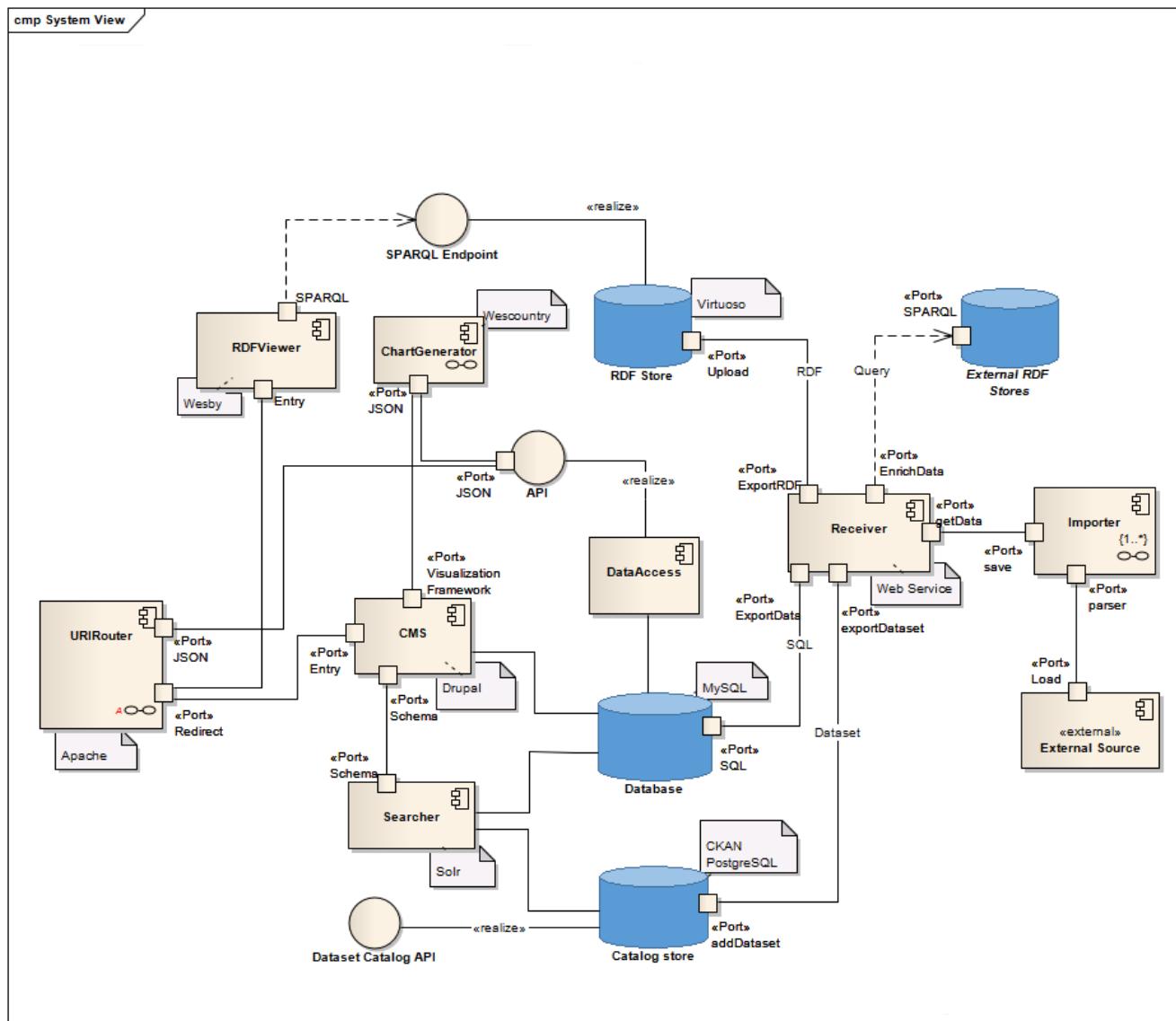


Figura 5.1: Diagrama de componentes del sistema

### 5.1.1.2. Componentes

Una vez mostrado el diagrama de componentes, se procederá a explicar el rol de cada uno de los componentes del sistema. La descripción comenzará en los componentes más internos y terminará en los componentes más externos y cercanos al usuario del sistema.

**External source** Las fuentes externas serán fuentes de datos pertenecientes a organizaciones externas. Estas fuentes proporcionan conjuntos de datos compuestos por indicadores y observaciones relativos a diferentes países y momentos del tiempo. Como ya se ha mencionado en la sección “Objetivos del proyecto” perteneciente al capítulo 1, algunas de las fuentes externas con las que se trabajará son:

- el Banco Mundial (*WorldBank*)
- la Organización de las Naciones Unidas para la Alimentación y la Agricultura (*FAO*)
- la Organización Mundial de la Salud (*WHO*)
- el Instituto Internacional de Investigación sobre Políticas Alimentarias (*IFPRI*)
- la Organización para la Cooperación y el Desarrollo Económicos(*OECD*)

**Importer** Los importadores tendrán la tarea de transformar los conjuntos de datos provenientes de las fuentes externas a un formato común con el fin de unificar y asegurar un nivel de calidad mínimo para los datos que se insertarán en el portal. Como se ha explicado en la sección “Identificación de actores del sistema” del capítulo 4, los importadores serán los actores que interactuarán con el punto de entrada de datos, enviándole a este los conjuntos de datos ya procesados.

**Receiver** El Punto de Entrada de Datos tendrá la misión de controlar todas las entradas de datos que se realicen hacia el portal. Recibirá las peticiones de los importadores y almacenará los conjuntos de datos en varios servicios diferentes, concretamente: una base de datos SQL, una base de datos RDF y un catálogo de datos. Además, el punto de entrada de datos también se encargará de enriquecer los datos antes de almacenarlos en el sistema, este enriquecimiento de datos tendrá lugar mediante la consulta de almacenamientos de RDF externos. La arquitectura del punto de entrada de datos se verá con más detalle posteriormente en la sección “Vista del punto de entrada de datos” de este mismo capítulo.

**External RDF Stores** Los almacenamientos (o *endpoints*) de RDF externos son servicios externos que contienen información variada que se utilizará para enriquecer los datos que llegan al punto de entrada. Un ejemplo de almacenamiento de RDF externo es *DBpedia*<sup>1</sup>.

**Database** La base de datos será utilizada tanto por el gestor de contenidos y el framework de visualizaciones como por el API. La base de datos será también uno de los lugares donde el punto de entrada almacene los catálogos de datos que llegan al portal. El funcionamiento y esquema de la base de datos se detallará en la sección “Diseño del modelo de datos” perteneciente a este mismo capítulo.

**Catalog store - Dataset Catalog API** El catálogo de datos será el encargado de almacenar los catálogos de datos que se utilizan en el portal acompañados de una serie de metadatos sobre su origen, creador, formato, etc. Proveerá también una interfaz que permitirá a los usuarios navegar e incluso descargar en bruto los catálogos de datos almacenados en el sistema. Como ya se explicó en la sección “Alternativas elegidas” del capítulo 2, el catálogo de datos seleccionado ha sido

---

<sup>1</sup>DBpedia contiene la información de la Wikipedia en forma de datos enlazados <http://dbpedia.org>About>

CKAN. Al igual que la base de datos, el catálogo de datos será uno de los lugares en los que el punto de entrada almacena la información que llega al portal.

**RDF Store - SPARQL endpoint** El servidor semántico almacenará los datos en formato RDF y proveerá una punto de acceso SPARQL<sup>2</sup> que podrá ser utilizado por los usuarios o por otros componentes del sistema. Este componente será uno de los lugares donde el punto de entrada almacene los catálogos de datos que recibe.

**API - DataAccess** El API será el encargado de proporcionar una interfaz de acceso a los datos almacenados en la base de datos del sistema. El API seguirá una arquitectura REST e incluirá un sistema de negociación de contenido con el que el cliente podrá seleccionar el formato en el que recibe los datos. Algunos de los formatos soportados por el API serán JSON, CSV y XML.

**Searcher** El buscador será el encargado de indexar toda la información almacenada en el portal para proveer un sistema de búsqueda amigable a los usuarios. Como se ha explicado en la sección “Alternativas elegidas” del capítulo 2, el buscador seleccionado ha sido Apache Solr.

**CMS** El gestor de contenidos tendrá una misión particularmente importante en el sistema final. Será el componente que proveerá todos los subsistemas de la zona social (véase la sección “Identificación de subsistemas”), además incluirá varios módulos que le permitirán comunicarse con otros componentes de la arquitectura. Entre estos módulos destaca el módulo encargado de proporcionar el framework de soporte a las visualizaciones. Los módulos que forman parte del gestor de contenidos podrán ser vistos con mayor detalle en la sección “Vista de los módulos del gestor de contenidos” de este mismo capítulo.

**ChartGenerator** Las visualizaciones serán las encargadas de transformar los datos en bruto almacenados en el sistema para presentarlos al usuario de una forma amigable y visual. Para su construcción las visualizaciones utilizarán los datos devueltos por un framework que forma parte del subsistema de datos y que provee la información necesaria. Las visualizaciones serán generadas por la librería *Wescoountry*.

**RDFViewer** El visualizador de datos enlazados tendrá la misión de permitir a los usuarios navegar y visualizar los datos almacenados en el servidor semántico de una forma amigable y sin necesidad de realizar consultas SPARQL de forma manual. Como se ha explicado en la sección “Alternativas elegidas” del capítulo 2 el visualizador de datos enlazados seleccionado será *Wesby*.

**URIRouter** El enrutador será el primer componente que entrará en acción del sistema ante las peticiones de los usuarios. Su misión será redirigir la petición del usuario en función de su URL hacia el componente adecuado. El enrutador utilizado será *Apache*.

### 5.1.2. Vista del punto de entrada de datos

A continuación se detallará la estructura completa del punto de entrada de datos. Este componente del sistema también es conocido como *Receiver* por ser su principal función el recibimiento y posterior almacenamiento de datos que provienen de fuentes externas al sistema.

---

<sup>2</sup>SPARQL es un lenguaje de consultas capaz de manipular datos en formato RDF. Para una mayor información al respecto véase <http://www.w3.org/TR/sparql11-overview/>

### 5.1.2.1. Diagrama de componentes

La figura 5.2 muestra el diagrama de todos los componentes que conforman el punto de entrada de datos.

El punto de entrada de datos está construido como una aplicación web que escucha peticiones del exterior y las procesa para almacenar los datos que recibe en diferentes puntos de almacenamiento.

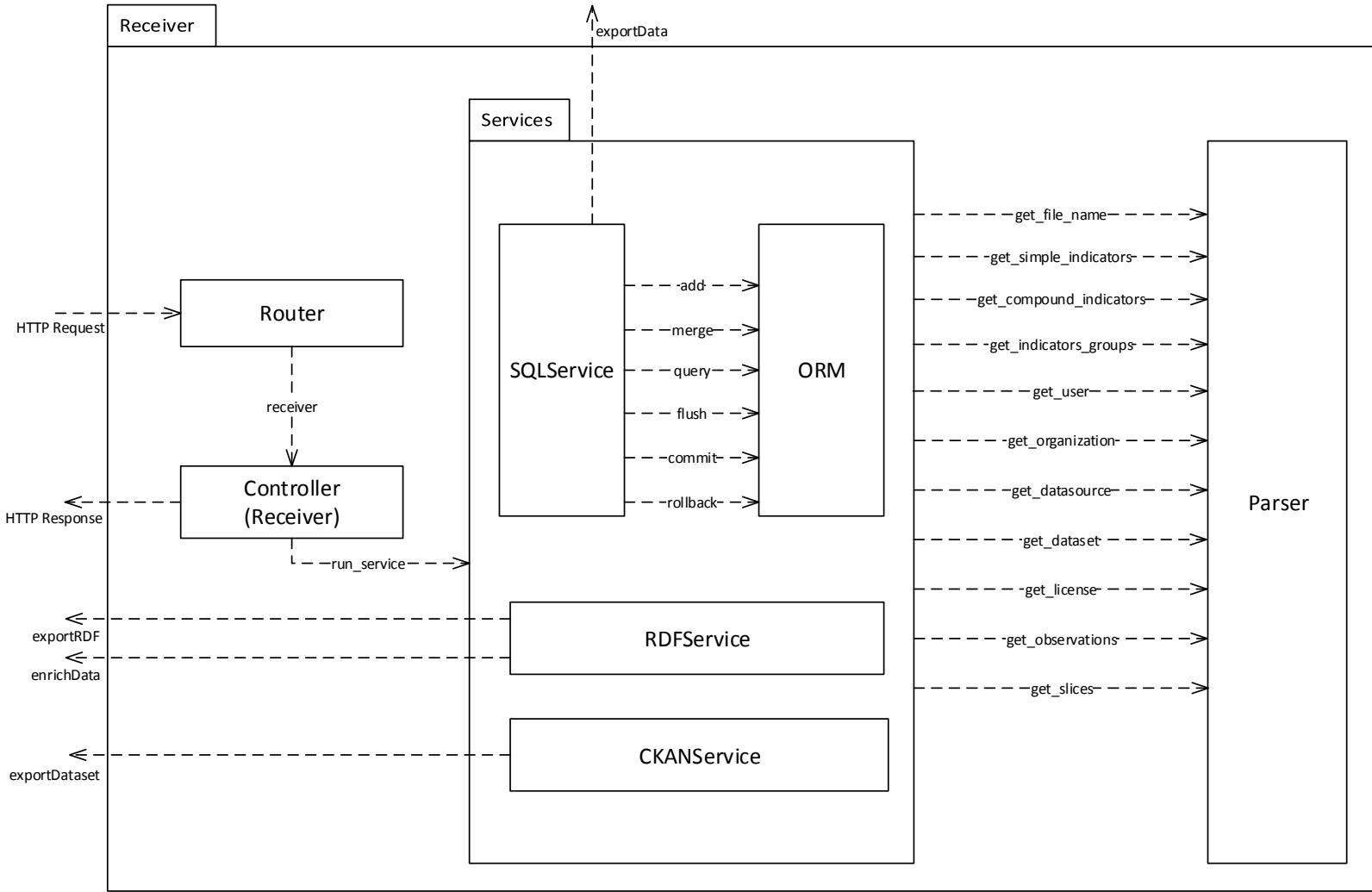


Figura 5.2: Diagrama de componentes del punto de entrada de datos

### 5.1.2.2. Descripción de los componentes

Tras haber mostrado el diagrama de los componentes que forman parte del punto de entrada de datos es el momento de explicar el papel que juega cada uno de ellos.

**Router** El enrutador se encarga de dirigir las peticiones HTTP que provienen del exterior hacia el controlador. También es el encargado de responder a las peticiones que no se correspondan con ningún controlador, o que no utilicen un verbo adecuado.

**Controller (Receiver)** El controlador se encarga de las peticiones de inserción de datos. Su misión es comprobar si la petición que recibe tiene una forma adecuada y llamar a los distintos servicios para que procesen los datos.

**Parser** El parser es el encargado de transformar los datos provenientes de las peticiones externas a un modelo de datos que será utilizado por los servicios para realizar los diferentes procesados. El modelo de datos será visto con más detalle en la sección “Diseño del modelo de datos” de este mismo capítulo.

**SQLService** El servicio de SQL se encarga de procesar los datos que llegan al punto de entrada y generar las consultas necesarias para almacenarlos en una base de datos relacional. Este servicio delega la generación de consultas SQL al ORM.

**ORM** El Mapeador Objeto-Relacional actúa como una capa de abstracción sobre la base de datos y permite al desarrollador trabajar con objetos sin necesidad de ocuparse de su transformación al modelo relacional.

**RDFService** El servicio de RDF se encarga de procesar los datos que llegan al punto de entrada y generar los grafos RDF que serán almacenados en el servidor semántico.

**CKANService** El servicio de CKAN se encarga de procesar los datos que llegan al punto de entrada y generar los metadatos que se almacenan en el catálogo de datos.

### 5.1.2.3. Relacion entre los componentes

Las peticiones provenientes de los importadores llegan al enrutador. El enrutador redirige las peticiones que utilicen un verbo adecuado (POST) hacia el controlador, quien comprueba si la petición cuenta con todos los parámetros necesarios.

Si la petición cuenta con los parámetros adecuados, el controlador llama a cada uno de los servicios para que realicen su trabajo. Los distintos servicios utilizarán a su vez el parser para convertir la información que proviene de la petición a un modelo de datos con el que ellos pueden trabajar.

### 5.1.2.4. Interfaces y puertos

A continuación se detallarán las interfaces y puertos de los componentes que forman parte del punto de entrada de datos.

#### 5.1.2.4.1. Receiver

La tabla 5.1 muestra el detalle de las interfaces del *Receiver*.

Cuadro 5.1: Vista del punto de entrada de datos - interfaces del *Receiver*.

Interfaz	Tipo	Tecnología	Propiedades
<b>HTTP Request</b>	Proveída	Servicio web REST	Recibe las peticiones procedentes del exterior
<b>HTTP Response</b>	Proveída	Servicio web REST	Responde a las peticiones HTTP recibidas
<b>exportRDF</b>	Requerida	API del servidor semántico	Exporta en formato RDF los datos recibidos de la petición
<b>exportData</b>	Requerida	Consultas a base de datos	Exporta en formato SQL los datos recibidos de la petición
<b>exportDataset</b>	Requerida	API del catálogo de datos	Exporta el catálogo de datos recibido de la petición junto con sus metadatos
<b>enrichData</b>	Requerida	Consultas SPARQL	Enriquece los datos mediante consultas SPARQL a servidores semánticos externos

#### 5.1.2.4.2. Router

La tabla 5.2 muestra el detalle de las interfaces del *Router*.

Cuadro 5.2: Vista del punto de entrada de datos - interfaces del *Router*.

Interfaz	Tipo	Tecnología	Propiedades
<b>HTTP Request</b>	Puerto de entrada	Servicio web REST	Recibe las peticiones procedentes del exterior
<b>receiver</b>	Puerto de salida	Llamada a método	Pasa la petición al controlador correspondiente tras haber comprobado que utiliza el verbo adecuado

#### 5.1.2.4.3. Controller

La tabla 5.3 muestra el detalle de las interfaces del *Controller*.

Cuadro 5.3: Vista del punto de entrada de datos - interfaces del *textitController*.

Interfaz	Tipo	Tecnología	Propiedades
<b>receiver</b>	Puerto de entrada	Llamada a método	Recibe la petición procedente el enrutador y comprueba si incluye todos los parámetros necesarios
<b>run service</b>	Puerto de salida	Llamada a método	Ejecuta la funcionalidad de un servicio
<b>response</b>	Puerto de salida	Servicio web REST	Envía una respuesta con un código de error HTTP adecuado en función del éxito o no en el procesamiento de la petición.

#### 5.1.2.4.4. Interfaces comunes a todos los servicios

La tabla 5.4 muestra el detalle de las interfaces comunes a todos los servicios.

Cuadro 5.4: Vista del punto de entrada de datos - interfaces comunes a todos los servicios.

Interfaz	Tipo	Tecnología	Propiedades
<b>run service</b>	Puerto de entrada	Llamada a método	Comienza la ejecución del servicio

Continuación de la tabla 5.4			
Interfaz	Tipo	Tecnología	Propiedades
<b>get file name</b>	Puerto de salida	Llamada a método	Obtiene el nombre del fichero que contiene los datos en bruto
<b>get simple indicators</b>	Puerto de salida	Llamada a método	Obtiene una lista de los indicadores simples obtenidos del fichero de datos
<b>get compound indicators</b>	Puerto de salida	Llamada a método	Obtiene una lista de los indicadores compuestos obtenidos del fichero de datos
<b>get user</b>	Puerto de salida	Llamada a método	Obtiene los del usuario que realiza la petición de inserción de datos
<b>get organization</b>	Puerto de salida	Llamada a método	Obtiene la información sobre la organización que proporciona el fichero de datos
<b>get datasource</b>	Puerto de salida	Llamada a método	Obtiene la información sobre la fuente de datos de la que procede el fichero de datos
<b>get dataset</b>	Puerto de salida	Llamada a método	Obtiene la información sobre el propio fichero de datos
<b>get license</b>	Puerto de salida	Llamada a método	Obtiene la información sobre la licencia bajo la que se publica el fichero de datos
<b>get observations</b>	Puerto de salida	Llamada a método	Obtiene la lista de observaciones procedentes del fichero de datos
<b>get slices</b>	Puerto de salida	Llamada a método	Obtiene la lista con las <i>slices</i> <sup>3</sup> procedentes del fichero de datos.

#### 5.1.2.4.5. Interfaces del *Parser*

La tabla 5.5 muestra el detalle de las interfaces pertenecientes al *Parser*.

Cuadro 5.5: Vista del punto de entrada de datos - interfaces pertenecientes al *Parser*.

Interfaz	Tipo	Tecnología	Propiedades
<b>get file name</b>	Puerto de entrada	Llamada a método	Devuelve el nombre del fichero que contiene los datos en bruto
<b>get simple indicators</b>	Puerto de entrada	Llamada a método	Devuelve una lista de los indicadores simples obtenidos del fichero de datos
<b>get compound indicators</b>	Puerto de entrada	Llamada a método	Devuelve una lista de los indicadores compuestos obtenidos del fichero de datos
<b>get user</b>	Puerto de salida	Llamada a entrada	Devuelve los del usuario que realiza la petición de inserción de datos
<b>get organization</b>	Puerto de entrada	Llamada a método	Devuelve la información sobre la organización que proporciona el fichero de datos
<b>get datasource</b>	Puerto de entrada	Llamada a método	Devuelve la información sobre la fuente de datos de la que procede el fichero de datos
<b>get dataset</b>	Puerto de entrada	Llamada a método	Devuelve la información sobre el propio fichero de datos
<b>get license</b>	Puerto de entrada	Llamada a método	Devuelve la información sobre la licencia bajo la que se publica el fichero de datos
<b>get observations</b>	Puerto de entrada	Llamada a método	Devuelve la lista de observaciones procedentes del fichero de datos

<sup>3</sup>Para más información sobre las *slices* véase la sección “RDF Data Cube” perteneciente al capítulo 3

Continuación de la tabla 5.5			
Interfaz	Tipo	Tecnología	Propiedades
<b>get slices</b>	Puerto de entrada	Llamada a método	Devuelve la lista con las <i>slices</i> procedentes del fichero de datos.

#### 5.1.2.4.6. SQLService

La tabla 5.6 muestra el detalle de las interfaces del *servicio de SQL* que no han sido incluidas anteriormente en la tabla 5.4.

Cuadro 5.6: Vista del punto de entrada de datos - interfaces del *servicio de SQL*.

Interfaz	Tipo	Tecnología	Propiedades
<b>exportData</b>	Requerida	Consultas a base de datos	Exporta en formato SQL los datos recibidos de la petición
<b>add</b>	Puerto de salida	Llamada a método	Almacena los datos de un objeto del modelo en la base de datos
<b>merge</b>	Puerto de salida	Llamada a método	Actualiza los datos de un objeto del modelo en la base de datos
<b>query</b>	Puerto de salida	Llamada a método	Realiza una consulta a la base de datos y devuelve los resultados
<b>flush</b>	Puerto de salida	Llamada a método	Vuelca los cambios realizados en memoria a la base de datos
<b>commit</b>	Puerto de salida	Llamada a método	Cierra una transacción con la base de datos
<b>rollback</b>	Puerto de salida	Llamada a método	Deshace los cambios de la transacción en curso

#### 5.1.2.4.7. ORM

La tabla 5.7 muestra el detalle de las interfaces del *ORM*.

Cuadro 5.7: Vista del punto de entrada de datos - interfaces del *ORM*.

Interfaz	Tipo	Tecnología	Propiedades
<b>add</b>	Puerto de entrada	Llamada a método	Almacena los datos de un objeto del modelo en la base de datos
<b>merge</b>	Puerto de entrada	Llamada a método	Actualiza los datos de un objeto del modelo en la base de datos
<b>query</b>	Puerto de entrada	Llamada a método	Realiza una consulta a la base de datos y devuelve los resultados
<b>flush</b>	Puerto de entrada	Llamada a método	Vuelca los cambios realizados en memoria a la base de datos
<b>commit</b>	Puerto de entrada	Llamada a método	Cierra una transacción con la base de datos
<b>rollback</b>	Puerto de entrada	Llamada a método	Deshace los cambios de la transacción en curso

#### 5.1.2.4.8. RDFService

La tabla 5.8 muestra el detalle de las interfaces del *servicio de RDF* que no han sido incluidas anteriormente en la tabla 5.4.

Cuadro 5.8: Vista del punto de entrada de datos - interfaces del *servicio de RDF*.

Interfaz	Tipo	Tecnología	Propiedades
<b>exportRDF</b>	Requerida	API del servidor semántico	Exporta en formato RDF los datos recibidos de la petición
<b>enrichData</b>	Requerida	Consultas SPARQL	Enriquece los datos mediante consultas SPARQL a servidores semánticos externos

#### 5.1.2.4.9. CKANService

La tabla 5.9 muestra el detalle de las interfaces del *servicio de CKAN* que no han sido incluidas anteriormente en la tabla 5.4.

Cuadro 5.9: Vista del punto de entrada de datos - interfaces del *servicio de CKAN*.

Interfaz	Tipo	Tecnología	Propiedades
<b>exportDataset</b>	Requerida	API del catálogo de datos	Exporta el catálogo de datos recibido de la petición junto con sus metadatos

### 5.1.3. Vista de los módulos del gestor de contenidos

A continuación se detallará la estructura completa del gestor de contenidos. Como se ha explicado en la sección “Alternativas elegidas” perteneciente al capítulo 2, el gestor de contenidos seleccionado ha sido Drupal.

#### 5.1.3.1. Diagrama de componentes

La figura 5.3 muestra el diagrama de los componentes que forman parte del gestor de contenidos.

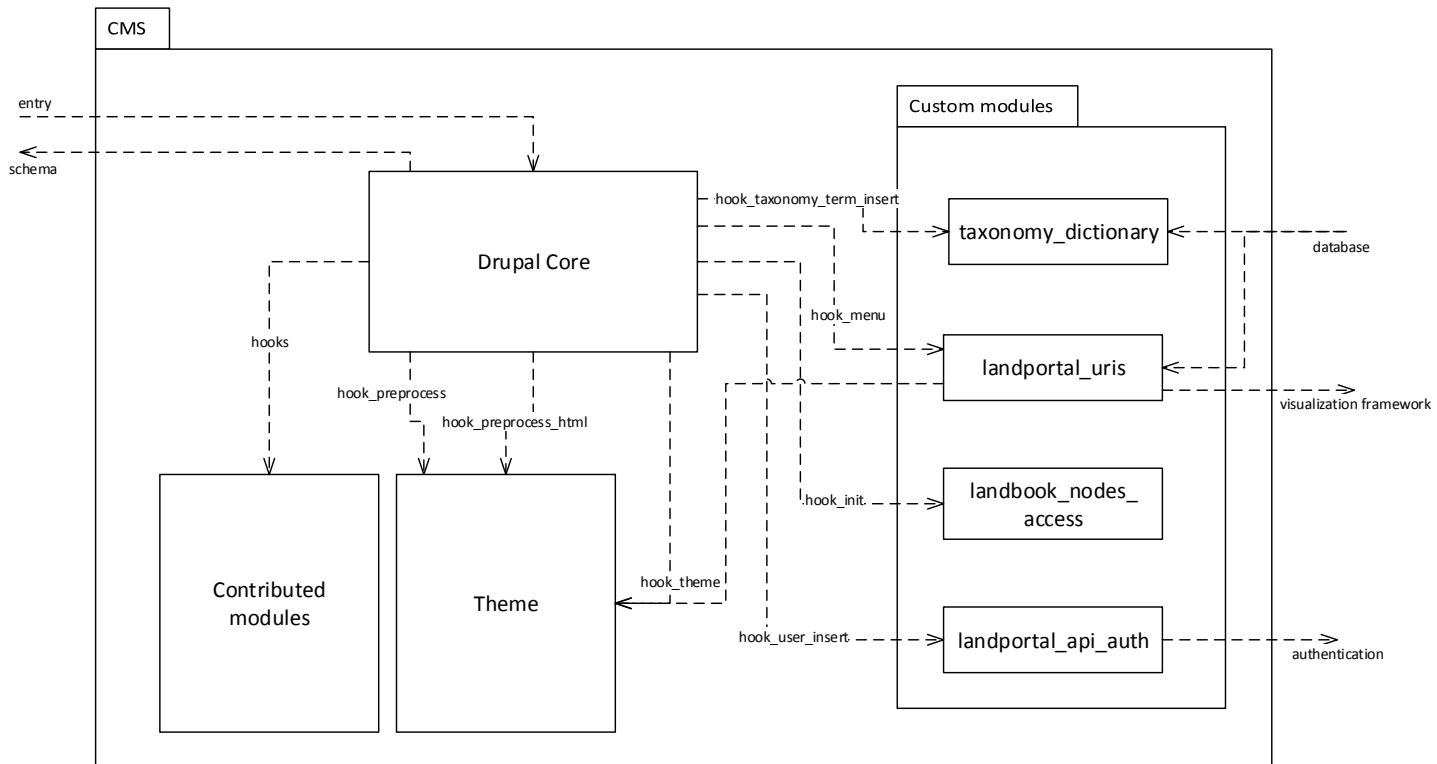


Figura 5.3: Diagrama de componentes del gestor de contenidos

### 5.1.3.2. Descripción de los componentes

Se describirán ahora los distintos componentes que forman parte del gestor de contenidos. La organización de dichos componentes puede verse en la figura 5.3

**Drupal Core** El núcleo de Drupal implementa la funcionalidad principal del gestor de contenidos. Es el encargado de procesar y responder a las peticiones, además de coordinar el funcionamiento de los diferentes módulos y temas visuales.

**Contributed modules** Los módulos contribuidos son módulos realizados por terceros y que han sido públicamente incluidos en el repositorio de módulos de Drupal. Estos módulos amplían la funcionalidad existente en el núcleo de Drupal<sup>4</sup>.

**Custom modules** Los módulos propios han sido especialmente creados para este proyecto y realizan alguna funcionalidad que no forma parte del núcleo de Drupal ni de ningún módulo contribuido. Se han desarrollado cuatro módulos propios:

- *taxonomy\_dictionary* Permite relacionar los datos existentes en la base de datos con los términos de las taxonomías existentes en Drupal.
- *landportal\_uris* Implementa un sistema similar al MVC para facilitar la creación de vistas. También incluye el framework que provee de datos a las visualizaciones. Este módulo se verá con más detalle posteriormente en la sección “Vista del módulo *landportal\_uris*” de este mismo capítulo.
- *landbook\_nodes\_access* Permite redirigir a las vistas adecuadas cuando se visite alguno de los nodos de la sección de datos.
- *landportal\_api\_auth* Genera y almacena las claves de autenticación del el API para los nuevos usuarios que se creen en el sistema.

**Theme** El tema implementa la interfaz de las diferentes vistas que forman parte del gestor de contenidos.

### 5.1.3.3. Relación entre los componentes

Las peticiones entrantes son procesadas por el núcleo de Drupal. En cada petición el núcleo de Drupal invoca los módulos que estén suscritos a los *hooks* adecuados para que realicen su funcionalidad<sup>5</sup>:

- Cuando se crea un nuevo término en la taxonomía se invoca al módulo *taxonomy\_dictionary* para que lo enlace con el término correspondiente en la base de datos.
- Cuando se realiza una petición que no va dirigida a mostrar la vista de administración se invoca al módulo *landportal\_uris* para que provea los datos con los que se renderizarán posteriormente las vistas del tema.
- Cuando se accede a alguno de los nodos pertenecientes a la zona de datos se invoca al módulo *landbook\_nodes\_access* para que realice la redirección hacia la vista adecuada.
- Cuando se crea un nuevo usuario en el portal se invoca al módulo *landportal\_api\_auth* para que cree y almacene sus claves de acceso al API.

<sup>4</sup>Actualmente existen unos 15.000 módulos que forman parte del repositorio. Dichos módulos pueden verse en [https://www.drupal.org/project/project\\_module](https://www.drupal.org/project/project_module)

<sup>5</sup>El módulo “*landportal\_uris*” será detallado posteriormente en la sección “Vista del módulo *landportal\_uris*” de este mismo capítulo. El resto de módulos propios cuentan simplemente con un único fichero que implementa el *hook* correspondiente por lo que, dada su sencillez, sus vistas se omitirán

Por último, el núcleo de Drupal invoca al tema para que renderice las vistas que serán mostradas al usuario que realiza la petición.

#### 5.1.3.4. Interfaces y puertos

A continuación se detallarán las interfaces y puertos de los componentes que forman parte del gestor de contenidos.

##### 5.1.3.4.1. CMS

La tabla 5.10 muestra el detalle de las interfaces del *CMS*.

Cuadro 5.10: Vista del gestor de contenidos - interfaces del *CMS*.

Interfaz	Tipo	Tecnología	Propiedades
<b>entry</b>	Proveída	Petición HTTP	Recibe las peticiones procedentes del exterior
<b>schema</b>	Requerida	API del motor de búsqueda	Envía los datos del gestor de contenidos al motor de búsqueda para su indexación <sup>6</sup>
<b>database</b>	Requerida	Conexión a base de datos	Accede a la base de datos donde se almacena la información perteneciente al subsistema de datos
<b>visualization framework</b>	Proveída	API Web	Provee datos con los que posteriormente crear las visualizaciones <sup>7</sup>
<b>authentication</b>	Proveída	API REST	Genera claves de autenticación en el API para los usuarios del portal

##### 5.1.3.4.2. Drupal Core

La tabla 5.11 muestra el detalle de las interfaces del *núcleo de Drupal*.

Cuadro 5.11: Vista del gestor de contenidos - interfaces del *núcleo de Drupal*.

Interfaz	Tipo	Tecnología	Propiedades
<b>entry</b>	Puerto de entrada	Petición HTTP	Recibe las peticiones procedentes del exterior
<b>schema</b>	Puerto de salida	API del motor de búsqueda	Envía los datos del gestor de contenidos al motor de búsqueda para su indexación
<b>hooks</b>	Puerto de salida	Llamada a método	Invoca a los módulos que estén suscritos al <i>hook</i> correspondiente
<b>hook preprocess</b>	Puerto de salida	Llamada a método	Indica al tema que preprocese las variables que sean necesarias para las vistas
<b>hook preprocess html</b>	Puerto de salida	Llamada a método	Indica al tema que preprocese las variables que sean necesarias para la plantilla HTML
<b>hook theme</b>	Puerto de salida	Llamada a método	Indica al tema que renderice una determinada plantilla para una vista
<b>hook taxonomy term insert</b>	Puerto de salida	Llamada a método	Indica a un módulo que se ha creado un nuevo término en la taxonomía para que este realice su funcionalidad

<sup>6</sup>Como se ha explicado en la sección “Alternativas elegidas” perteneciente al capítulo 2, el motor de búsqueda seleccionado ha sido *Apache Solr*

<sup>7</sup>Los elementos de este framework podrán ser vistos con mayor detalle posteriormente en la sección “Vista del módulo *landportal uris*” perteneciente a este mismo capítulo

Continuación de la tabla 5.11			
Interfaz	Tipo	Tecnología	Propiedades
<b>hook menu</b>	Puerto de salida	Llamada a método	Permite a un módulo registrar rutas para definir cómo son atendidas las peticiones HTTP
<b>hook init</b>	Puerto de salida	Llamada a método	Indica a un módulo que acaba de tener lugar una petición por parte de un usuario
<b>hook user insert</b>	Puerto de salida	Llamada a método	Indica a un módulo que acaba de producirse el registro de un nuevo usuario

#### 5.1.3.4.3. Contributed modules

La tabla 5.12 muestra el detalle de las interfaces de los *módulos contribuidos*.

Cuadro 5.12: Vista del gestor de contenidos - interfaces de los *módulos contribuidos*.

Interfaz	Tipo	Tecnología	Propiedades
<b>hooks</b>	Puerto de entrada	Llamada a método	Invocados por el núcleo de Drupal cuando tienen lugar ciertos eventos en el sistema

#### 5.1.3.4.4. Theme

La tabla 5.13 muestra el detalle de las interfaces del *tema*.

Cuadro 5.13: Vista del gestor de contenidos - interfaces del *tema*.

Interfaz	Tipo	Tecnología	Propiedades
<b>hook preprocess</b>	Puerto de entrada	Llamada a método	Preprocesa las variables que sean necesarias para la posterior renderización de las vistas
<b>hook preprocess html</b>	Puerto de entrada	Llamada a método	Preprocesa las variables necesarias para renderizar la plantilla HTML
<b>hook theme</b>	Puerto de entrada	Llamada a método	Renderiza la plantilla de una determinada vista

#### 5.1.3.4.5. Taxonomy dictionary

La tabla 5.14 muestra el detalle de las interfaces del módulo “*Taxonomy dictionary*”.

Cuadro 5.14: Vista del gestor de contenidos - interfaces del módulo *Taxonomy dictionary*.

Interfaz	Tipo	Tecnología	Propiedades
<b>hook taxonomy term insert</b>	Puerto de entrada	Llamada a método	Recibe la información de un nuevo término que se ha creado en la taxonomía
<b>database</b>	Puerto de salida	Conexión a base de datos	Almacena la información del término insertado en la taxonomía en la entrada correspondiente de la base de datos

#### 5.1.3.4.6. LandPortal URIs

La tabla 5.15 muestra el detalle de las interfaces del módulo “*LandPortal URIs*”.

Cuadro 5.15: Vista del gestor de contenidos - interfaces del módulo *LandPortal URIs*.

Interfaz	Tipo	Tecnología	Propiedades
<b>hook menu</b>	Puerto de entrada	Llamada a método	El gestor de contenidos está definiendo las rutas que tendrán las vistas del portal
<b>hook theme</b>	Puerto de salida	Llamada a método	Indica al tema que renderice una determinada plantilla para una vista
<b>visualization framework</b>	Puerto de salida	API Web	Provee datos con los que posteriormente crear las visualizaciones
<b>database</b>	Puerto de salida	Conexión a base de datos	Obtiene información de la base de datos

#### 5.1.3.4.7. *LandbBook nodes access*

La tabla 5.16 muestra el detalle de las interfaces del módulo “*LandbBook nodes access*”.

Cuadro 5.16: Vista del gestor de contenidos - interfaces del módulo *LandbBook nodes access*.

Interfaz	Tipo	Tecnología	Propiedades
<b>hook init</b>	Puerto de entrada	Llamada a método	Invocada por el núcleo del gestor de contenidos cuando acaba de tener lugar una petición por parte de un usuario

#### 5.1.3.4.8. *Landportal API auth*

La tabla 5.17 muestra el detalle de las interfaces del módulo “*LandPortal API auth*”.

Cuadro 5.17: Vista del gestor de contenidos - interfaces del módulo *LandPortal API auth*.

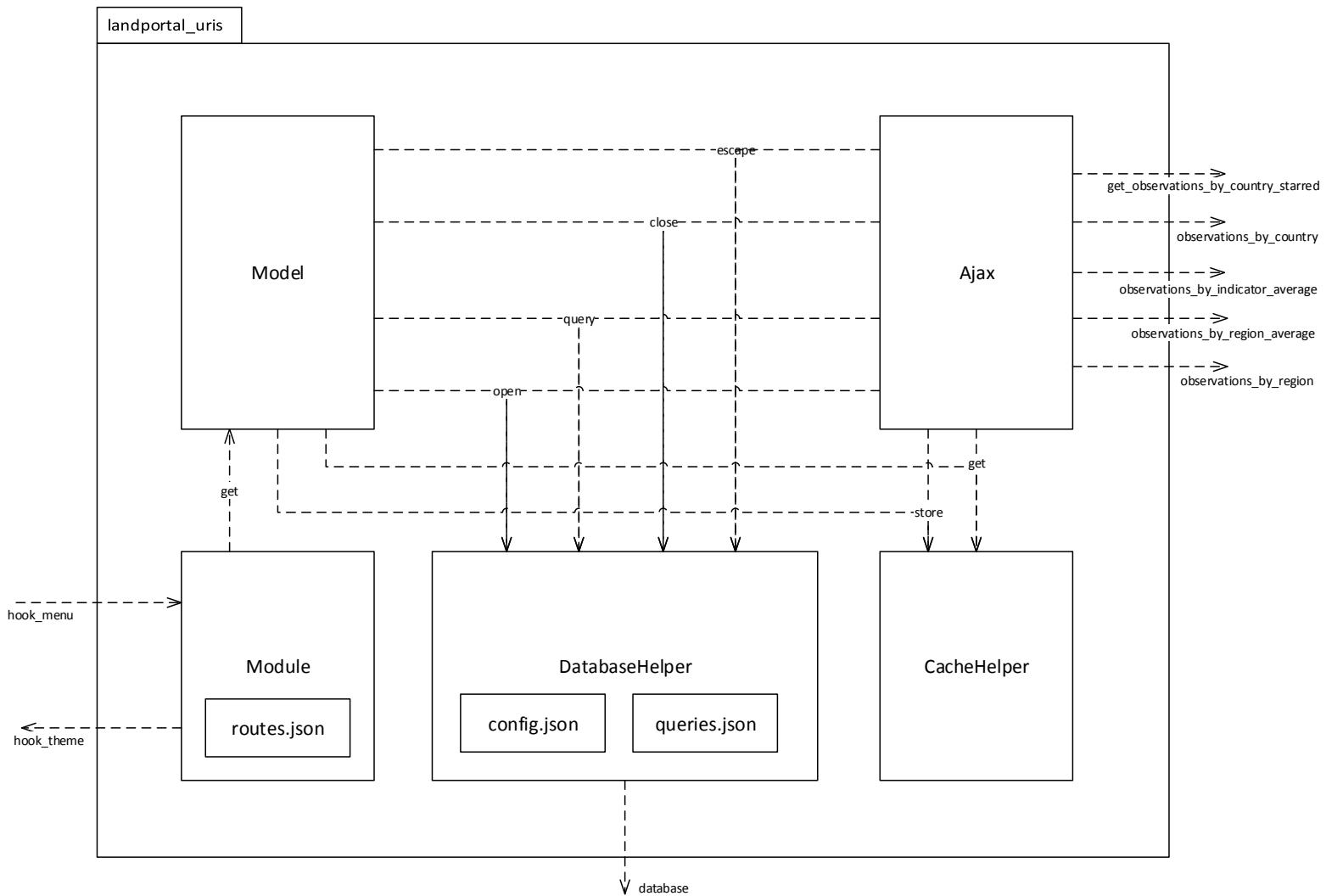
Interfaz	Tipo	Tecnología	Propiedades
<b>hook user insert</b>	Puerto de entrada	Llamada a método	Invocado por el núcleo del gestor de contenidos cuando se realiza el registro de un nuevo usuario
<b>authentication</b>	Puerto de salida	API REST	Genera claves de autenticación en el API para el usuario recién registrado en el portal

### 5.1.4. Vista del módulo *landportal uris*

A continuación se detallará la vista completa del módulo *landportal\_uris*. Tal y como se ha explicado anteriormente en la vista del gestor de contenidos, se omitirá la presentación de las vistas del resto de módulos debido su baja complejidad, ya que estarán formados por un único fichero donde se implementa el *hook* correspondiente.

#### 5.1.4.1. Diagrama de componentes

La figura 5.4 muestra el diagrama de los componentes que forman parte del módulo “*landportal\_uris*”.

Figura 5.4: Diagrama de componentes del módulo “`landportal_uris`”

#### 5.1.4.2. Descripción de los componentes

Se describirán ahora los distintos componentes que forman parte del módulo “*landportal\_uris*”. La organización de dichos componentes puede verse en la figura 5.4

**Module** Contiene la implementación del *hook* que el gestor de contenidos utilizará para llamar a este módulo. Es el principal punto de entrada al módulo. Incluye un fichero donde se declararán las rutas y las vistas que estarán disponibles.

**DatabaseHelper** Funciona como una capa de abstracción sobre la base de datos. Ofrece una serie de métodos que serán utilizados por el resto de componentes que necesiten acceder a la base de datos. Incluye un fichero que configura la información de conexión a la base de datos y otro que incluye las consultas disponibles.

**CacheHelper** Permite el cacheo de datos con el fin de agilizar la consulta y procesado de grandes cantidades de datos. Ofrece una serie de métodos que serán utilizados por el resto de componentes que necesiten acceder al caché.

**Ajax** Implementa el framework que da soporte a la creación de visualizaciones. Expone varias interfaces que serán utilizadas por las visualizaciones para obtener los datos que posteriormente presentarán a los usuarios.

**Model** Devuelve los datos necesarios para presentar las diferentes vistas del sistema. Los modelos se invocarán utilizando un convenio de nombres en función de la vista a la que pertenezcan.

#### 5.1.4.3. Relación entre los componentes

El módulo lee la configuración de rutas desde el fichero “*routes.json*”. Con cada petición recibida desde el núcleo del gestor de contenidos se buscará el modelo correspondiente a la vista mediante un convenio de nombres y se invocará su funcionalidad. Posteriormente se invocará al tema para que renderice la vista adecuada.

Los modelos cargarán los datos necesarios para la renderización de la vista a través de diferentes consultas a la base de datos. El framework de soporte a visualizaciones funcionará de forma similar a los modelos, realizando consultas a la base de datos para obtener los datos necesarios.

En ambos casos, para agilizar la consulta y el procesamiento de los datos, los resultados se cachearán antes de ser devueltos, de forma que las posteriores peticiones accedan directamente a la información del caché.

El anexo 12.6 contiene una guía que explica en detalle el proceso de creación de plantillas personalizadas.

#### 5.1.4.4. Interfaces y puertos

A continuación se detallarán las interfaces y puertos de los componentes que forman parte del módulo “*landportal\_uris*”.

##### 5.1.4.4.1. LandPortal URIs

La tabla 5.18 muestra el detalle de las interfaces del módulo “*LandPortal URIs*”.

Cuadro 5.18: Vista del módulo *LandPortal URIs*- interfaces del módulo *LandPortal URIs*.

Interfaz	Tipo	Tecnología	Propiedades
<b>hook menu</b>	Proveída	Llamada a método	Invocado por el gestor de contenidos durante la definición de las rutas que tendrán las vistas del portal
<b>hook theme</b>	Requerida	Llamada a método	Pide al tema que renderice una determinada plantilla para una vista
<b>get observations by country starred</b>	Proveída	API Web	Devuelve todas las observaciones referentes a un determinado país para los indicadores favoritos
<b>observations by country</b>	Proveída	API Web	Devuelve todas las observaciones referentes a un determinado país sin tener en cuenta el indicador
<b>get observations by indicator average</b>	Proveída	API Web	Devuelve el valor medio de todas las observaciones para un determinado indicador
<b>get observations by region average</b>	Proveída	API Web	Devuelve el valor medio de todas las observaciones referente a una determinada región sin tener en cuenta el indicador
<b>observations by region</b>	Proveída	API Web	Devuelve todas las observaciones referentes a una determinada región sin tener en cuenta el indicador
<b>database</b>	Requerida	Conexión a base de datos	Obtiene información de la base de datos

#### 5.1.4.4.2. Module

La tabla 5.19 muestra el detalle de las interfaces del componente “*module*”.

Cuadro 5.19: Vista del módulo *LandPortal URIs*- interfaces del componente “*module*”.

Interfaz	Tipo	Tecnología	Propiedades
<b>hook menu</b>	Puerto de entrada	Llamada a método	Invocado por el gestor de contenidos durante la definición de las rutas que tendrán las vistas del portal
<b>hook theme</b>	Requerida	Llamada a método	Pide al tema que renderice una determinada plantilla para una vista
<b>get</b>	Puerto de salida	Llamada a método	Pide a un modelo los datos necesarios para que posteriormente el tema renderice la vista

#### 5.1.4.4.3. DatabaseHelper

La tabla 5.20 muestra el detalle de las interfaces del componente “*DatabaseHelper*”.

Cuadro 5.20: Vista del módulo *LandPortal URIs*- interfaces del componente “*DatabaseHelper*”.

Interfaz	Tipo	Tecnología	Propiedades
<b>open</b>	Puerto de entrada	Llamada a método	Abre una conexión con la base de datos

Continuación de la tabla 5.20			
Interfaz	Tipo	Tecnología	Propiedades
<b>query</b>	Puerto de entrada	Llamada a método	Realiza una consulta a la base de datos y devuelve su resultado
<b>close</b>	Puerto de entrada	Llamada a método	Cierra una conexión con la base de datos
<b>escape</b>	Puerto de entrada	Llamada a método	Escapa los parámetros de una consulta antes de enviarla a la base de datos <sup>8</sup>
<b>database</b>	Puerto de salida	Conexión a base de datos	Obtiene información de la base de datos

#### 5.1.4.4.4. CacheHelper

La tabla 5.21 muestra el detalle de las interfaces del componente “CacheHelper”.

Cuadro 5.21: Vista del módulo *LandPortal URIs*- interfaces del componente “CacheHelper”.

Interfaz	Tipo	Tecnología	Propiedades
<b>get</b>	Puerto de entrada	Llamada a método	Obtiene el elemento almacenado en el caché bajo una determinada clave
<b>store</b>	Puerto de entrada	Llamada a método	Almacena un elemento en el caché utilizando una determinada clave que posteriormente podrá ser utilizada para recuperarlo

#### 5.1.4.4.5. Model

La tabla 5.22 muestra el detalle de las interfaces comunes a todos los modelos.

Cuadro 5.22: Vista del módulo *LandPortal URIs*- interfaces del componente “Model”.

Interfaz	Tipo	Tecnología	Propiedades
<b>open</b>	Puerto de salida	Llamada a método	Abre una conexión con la base de datos
<b>query</b>	Puerto de salida	Llamada a método	Realiza una consulta a la base de datos y devuelve su resultado
<b>close</b>	Puerto de salida	Llamada a método	Cierra una conexión con la base de datos
<b>escape</b>	Puerto de salida	Llamada a método	Escapa los parámetros de una consulta antes de enviarla a la base de datos
<b>get</b>	Puerto de entrada	Llamada a método	Devuelve la información necesaria para que el tema renderice la vista correspondiente
<b>get (CacheHelper)</b>	Puerto de salida	Llamada a método	Obtiene el elemento almacenado en el caché bajo una determinada clave
<b>store</b>	Puerto de salida	Llamada a método	Almacena un elemento en el caché utilizando una determinada clave que posteriormente podrá ser utilizada para recuperarlo

<sup>8</sup>El escapado de argumentos es un procedimiento necesario para evitar ataques del tipo *SQLInjection* cuando no se utilizan *PreparedStatement*s. Para más información al respecto véase <http://www.php.net/manual/en/mysqli-real-escape-string.php>

#### 5.1.4.4.6. Ajax

La tabla 5.23 muestra el detalle de las interfaces pertenecientes al framework de soporte a las visualizaciones.

Cuadro 5.23: Vista del módulo *LandPortal URIs*- interfaces del componente “Ajax”.

Interfaz	Tipo	Tecnología	Propiedades
<b>open</b>	Puerto de salida	Llamada a método	Abre una conexión con la base de datos
<b>query</b>	Puerto de salida	Llamada a método	Realiza una consulta a la base de datos y devuelve su resultado
<b>close</b>	Puerto de salida	Llamada a método	Cierra una conexión con la base de datos
<b>escape</b>	Puerto de salida	Llamada a método	Escapa los parámetros de una consulta antes de enviarla a la base de datos
<b>get (CacheHelper)</b>	Puerto de salida	Llamada a método	Obtiene el elemento almacenado en el caché bajo una determinada clave
<b>store</b>	Puerto de salida	Llamada a método	Almacena un elemento en el caché utilizando una determinada clave que posteriormente podrá ser utilizada para recuperarlo
<b>get observations by country starred</b>	Puerto de salida	API Web	Devuelve todas las observaciones referentes a un determinado país para los indicadores favoritos
<b>observations by country</b>	Puerto de salida	API Web	Devuelve todas las observaciones referentes a un determinado país sin tener en cuenta el indicador
<b>get observations by indicator average</b>	Puerto de salida	API Web	Devuelve el valor medio de todas las observaciones para un determinado indicador
<b>get observations by region average</b>	Puerto de salida	API Web	Devuelve el valor medio de todas las observaciones referente a una determinada región sin tener en cuenta el indicador
<b>observations by region</b>	Puerto de salida	API Web	Devuelve todas las observaciones referentes a una determinada región sin tener en cuenta el indicador

## 5.2. Comportamiento del sistema

En esta sección se explicarán con detalle algunos aspectos del comportamiento del sistema. Concretamente se explicará cómo funciona el proceso de importación de datos, el framework de soporte a las visualizaciones y la llamada a las plantillas de la sección de datos.

### 5.2.1. Proceso de importación de datos

A continuación se detallarán los pasos del proceso de importación de datos al portal. Tal y como se ha explicado en la “Vista del sistema” este proceso es realizado por el Punto de Entrada de Datos o *Receiver*, perteneciente al subsistema de datos. La arquitectura de dicho componente puede verse con detalle en la sección “Vista del punto de entrada de datos” del presente capítulo.

La figura 5.5 muestra el diagrama de actividad del proceso de importación de datos. Los pasos de dicho proceso son los siguientes:

1. El Punto de Entrada de Datos o *Receiver* recibe una petición de inserción de datos por parte de algún importador.
2. En primer lugar el Punto de Entrada de Datos comprueba si el verbo utilizado por la petición es un verbo correcto. Concretamente el único verbo HTTP aceptado es POST. En caso de que la petición utilizara un verbo distinto de POST, el Punto de Entrada de Datos envía una respuesta con código de error HTTP 405<sup>9</sup>.
3. Posteriormente el Punto de Entrada de Datos comprueba si la dirección IP de la que procede la petición está en la lista blanca de direcciones admitidas para la importación de datos. En caso de que la dirección IP de origen no se encontrara en la lista blanca, el Punto de Entrada de Datos envía una respuesta con un código de error HTTP 403<sup>10</sup>.
4. Tras comprobar que la petición utiliza un verbo correcto y procede de una fuente confiable, el Punto de Entrada de Datos comprueba que también incluye todos los parámetros necesarios. En caso de que la petición no incluyera todos los parámetros, el Punto de Entrada de Datos responde con un código de error HTTP 400<sup>11</sup>.
5. Una vez que todas las validaciones han sido realizadas con éxito, se procede a la ejecución de los servicios de persistencia de datos.
  - a) En primer lugar se ejecuta el servicio de persistencia SQL, el cual almacena los datos provenientes de la petición en la base de datos MySQL.
  - b) Posteriormente se ejecuta el servicio de persistencia RDF, que almacena los datos en el servidor semántico (como se ha explicado en la sección “Alternativas elegidas” del capítulo 2 el servidor semántico seleccionado ha sido Virtuoso).
  - c) Por último se ejecuta el servicio de persistencia de CKAN, que almacena la información en el catálogo de datos (tal y como se ha explicado anteriormente en la sección “Alternativas elegidas” del capítulo 2 el catálogo de datos seleccionado ha sido CKAN.).
6. En último lugar y tras haber ejecutado todos los servicios de persistencia, el Punto de Entrada de Datos finaliza el proceso enviando una respuesta HTTP 200 e indicando que todo ha funcionado correctamente.

### 5.2.2. Funcionamiento del framework de soporte a visualizaciones

A continuación se explicará el funcionamiento del framework que provee los datos necesarios para crear las visualizaciones. Anteriormente, en la “Vista del sistema” se explicó que este componente se sitúa dentro del CMS y forma parte del subsistema de datos y se encarga de ofrecer los datos que se utilizarán por las visualizaciones para presentarlos de forma gráfica al usuario. La arquitectura de este componente pude verse en detalle en la “Vista del módulo *landportal uris*” del presente capítulo.

La figura 5.6 muestra el diagrama de actividad de este componente. Los siguientes pasos

<sup>9</sup>El código de error HTTP 405 tiene el significado “Method Not Allowed”. Para más información a cerca de los códigos de estado HTTP véase la especificación del W3C al respecto, [W3C]

<sup>10</sup>El código de error 405 tiene el significado “Forbidden”. Para más información a cerca de los códigos de estado HTTP véase la especificación del W3C al respecto, [W3C]

<sup>11</sup>El código de error 4005 tiene el significado “Bad Request”. Para más información a cerca de los códigos de estado HTTP véase la especificación del W3C al respecto, [W3C]

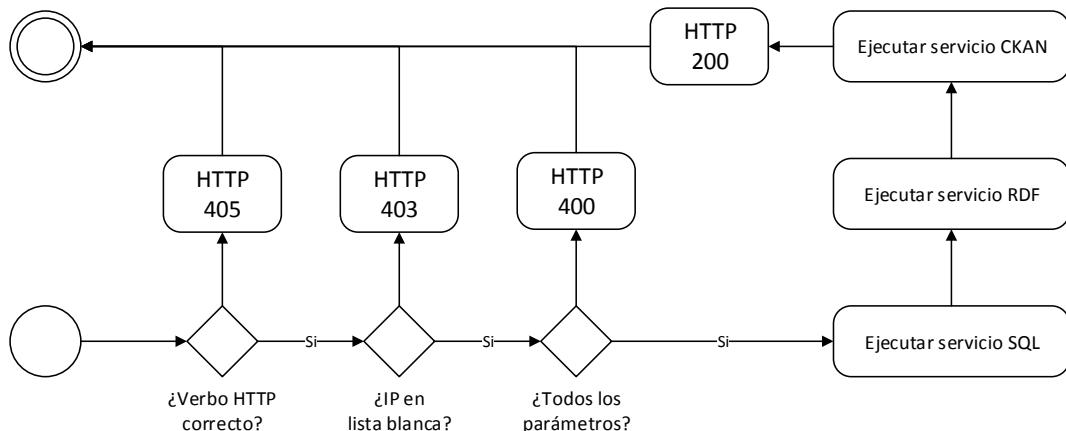


Figura 5.5: Diagrama de actividad del proceso de importación de datos

tienen lugar durante su actividad.

1. En primer lugar, el framework recibe una petición de datos por parte de una visualización para mostrarlos de forma gráfica a un usuario.
2. El primer paso que tiene lugar es la extracción de parámetros de la petición. Por ejemplo: en caso de necesitar datos sobre una determinada región, uno de los parámetros será el identificador de dicha región.
3. El segundo paso consiste en comprobar si el sistema ha recibido alguna petición similar anteriormente y, por tanto se encuentra almacenada en el caché. En caso de que una petición anterior con los mismos parámetros se encontrara cacheada, el sistema devuelve los datos del caché y finaliza la ejecución.
4. Si no hay cacheada ninguna petición similar anterior será necesario extraer los datos de la base de datos y construir una respuesta.
  - a) Primeramente se abre una nueva conexión con la base de datos. La conexión se realiza a través del componente *DatabaseHelper*, como se puede comprobar en la “Vista del módulo *landportal uris*”.
  - b) Posteriormente se procede al escapado de los argumentos que llegaron originalmente en la petición. Este proceso es necesario para evitar posibles ataques del tipo *SQL Injection*<sup>12</sup>.
  - c) Tras garantizar que los parámetros recibidos son seguros, el sistema procede a realizar las consultas necesarias a la base de datos y recibir la información que devolverá posteriormente.
  - d) Una vez se haya terminado de trabajar con la base de datos, se cerrará la conexión puesto que no se volverá a utilizar posteriormente.
5. Cuando ya se han obtenido todos los datos necesarios de la base de datos y se haya cerrado la conexión, el sistema procesará los datos para transformarlos al formato JSON en el que se enviará la respuesta.
6. Una vez que los datos ya se encuentran en formato JSON y listos para ser de-

<sup>12</sup>Una inyección de código SQL o *SQL Injection* es un mecanismo de ataque que permite alterar las consultas SQL para obtener información oculta, modificar la información almacenada en la base de datos o ejecutar instrucciones del sistema de gestión de base de datos. Para obtener más información sobre se recomienda ver la entrada del manual de PHP a cerca de este tema en <http://php.net/manual/en/security.database.sql-injection.php>

vueltos, el sistema los almacenará en el caché para que, como ya se ha explicado anteriormente, las posteriores peticiones similares no tengan que volver a consultar la base de datos ni volver a procesar los resultados.

7. Por último, se devolverán los datos ya procesados en formato JSON para que sean utilizados por las visualizaciones.

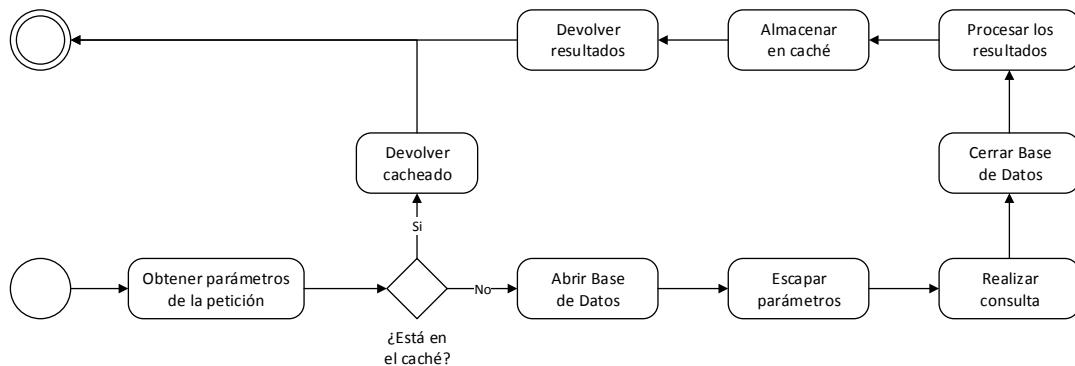


Figura 5.6: Diagrama de actividad del framework de soporte a visualizaciones

El anexo 12.6 contiene una guía que explica en detalle el proceso de creación de plantillas personalizadas.

### 5.2.3. Llamada a las plantillas de la sección de datos

A continuación se explicará el funcionamiento del módulo “*landportal-uris*” en conjunción con el tema del CMS. En la “Vista de los módulos del gestor de contenidos” puede verse con detalle la organización de ambos componentes como parte del gestor de contenidos.

La figura 5.7 muestra el diagrama de actividad de las llamadas a las plantillas del tema por parte del módulo “*landportal-uris*”. Cabe destacar que este diagrama sólo es aplicable a las plantillas pertenecientes a la sección de datos, puesto que las plantillas de la sección social serán llamadas directamente por el núcleo de Drupal<sup>13</sup>. La actividad está formada por los siguientes pasos:

1. En primer lugar, cuando el módulo recibe una petición invoca a un *callback* que se encarga de seleccionar tanto el modelo como la plantilla adecuados.
  - Como se verá posteriormente, el modelo es el encargado de obtener los datos con los que se rellenarán las vistas. El modelo adecuado para una petición se obtiene mediante un convenio de nombres, por ejemplo una petición a la ruta *country* instanciará el modelo *Country*, de este modo la creación de nuevas rutas no requiere ningún tipo de configuración adicional.
  - La plantilla contiene el código HTML que se mostrará al usuario. La plantilla adecuada se obtiene utilizando un convenio de nombres de forma similar a como sucede con el modelo.
2. Una vez que se ha obtenido el modelo se procederá a pedirle sus datos. En este punto es necesario mencionar que la forma en la que los modelos obtienen sus datos es similar a la forma en la que lo hacen las clases del framework de soporte

<sup>13</sup>El mecanismo utilizado por el núcleo de Drupal para seleccionar las plantillas recibe el nombre de “*template suggestions*”. Para más información se recomienda ver el contenido de la documentación de Drupal al respecto en <https://www.drupal.org/node/1089656>

a las visualizaciones. Todo lo explicado en la sección anterior (“Funcionamiento del framework de soporte a visualizaciones”) es también aplicable aquí.

3. En caso de que el modelo no devolviera ningún dato indicaría que los parámetros de la petición son incorrectos y, por tanto, la plantilla se cambiará para mostrar un error 404.
4. Tanto si el modelo ha obtenido datos o no, se realizará un preprocesado de la plantilla. El preprocesado permite incluir algunas variables comunes que son utilizadas por todas las plantillas.
5. Por último se renderizará la plantilla para incluir todos los datos obtenidos del modelo y se devolverá el código HTML resultante al usuario que realizó la petición.

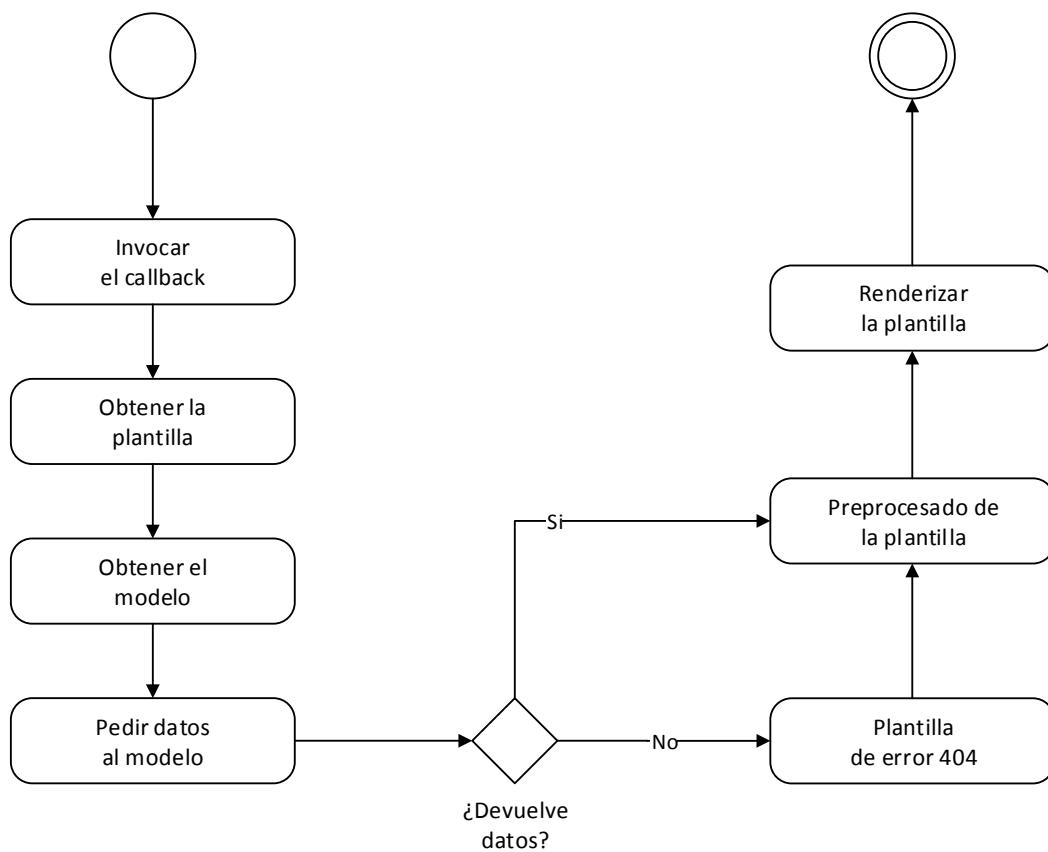


Figura 5.7: Diagrama de actividad de la llamada a las plantillas en la sección de datos

### 5.3. Diseño del modelo de datos

En esta sección se explicará el diseño final del modelo de datos que se utilizará en el sistema. Para facilitar la lectura se dividirá en dos partes: una parte mostrará el modelo relativo a la zona de datos (subsistema de datos) y la otra mostrará el modelo utilizado en la zona social (subsistemas de gestión de usuarios, debates, noticias, eventos, etc.).

### 5.3.1. Modelo de la zona de datos

#### 5.3.1.1. Diagrama del modelo de datos

La figura 5.8 muestra el diagrama con el diseño final del modelo del subsistema de datos.

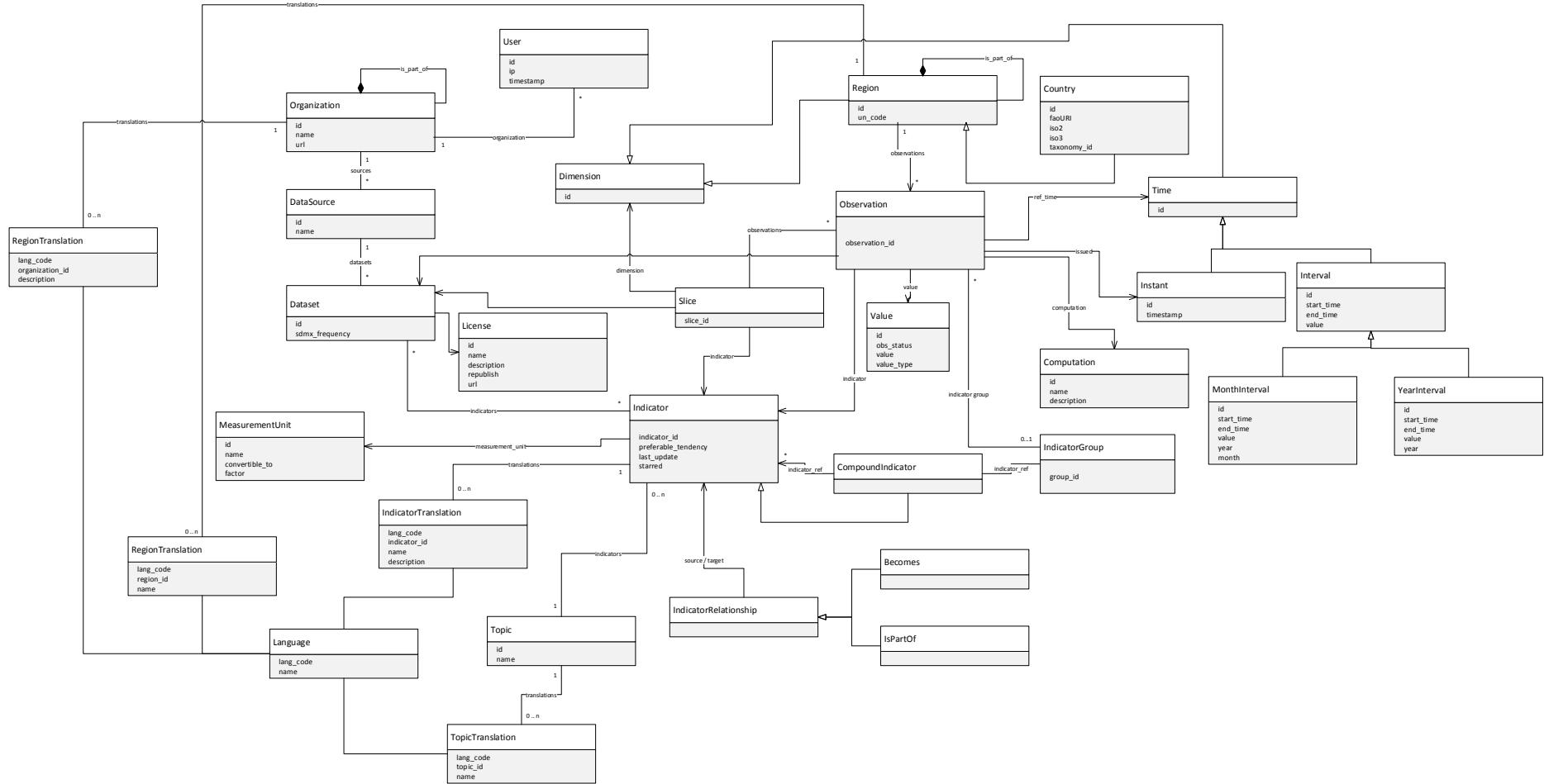


Figura 5.8: Diseño final del modelo de datos

Como se puede comprobar, existe una variación importante respecto a lo originalmente diseñado en la figura 4.11 perteneciente a la sección “Análisis preliminar de clases de la zona de datos” del capítulo 4. A continuación se expondrán las razones para esta divergencia:

#### **5.3.1.1.1. Especificación más cercana al RDF Data Cube Vocabulary**

Como se explicó anteriormente en el capítulo 3, el RDF Data Cube Vocabulary organiza las observaciones en torno a un grupo de *dimensiones*, *medidas* y *atributos*.

Para dar soporte a esto se han incluido la clase “*Dimension*”, de la que heredarán todas las dimensiones, las clases “*MeasurementUnit*” y “*Value*” pretenden dar entidad completa a las medidas y, por último, las clases “*Computation*” y “*Time*” pretenden dar entidad a los atributos de las observaciones.

También se ha incluido soporte a slices con la clase “*Slice*”.

#### **5.3.1.1.2. Inclusión de soporte para la internacionalización de datos**

Tal y como se definió en el requisito *RLB 7* en la sección “Requisitos de la sección de datos” del capítulo 4, es necesario soportar la internacionalización en el nivel de datos.

Este soporte ha sido incluido con las clases “*Language*”, “*OrganizationTranslation*”, “*RegionTranslation*”, “*TopicTranslation*” e “*IndicatorTranslation*”.

#### **5.3.1.1.3. Soporte a distintos intervalos temporales**

Puesto que las observaciones pueden hacer referencia a diferentes intervalos temporales, se han incluido las clases “*Interval*”, “*YearInterval*” y “*MonthInterval*”.

#### **5.3.1.2. Elementos del modelo de datos**

A continuación se explicará el cometido de cada una de las clases del modelo que se han mostrado en la figura 5.8.

**Language** Representa un idioma en el que se traducirán los datos. Sus atributos son los siguientes:

- **Lang code** Código de dos letras del idioma
- **Name** Nombre del idioma

**RegionTranslation** Traduce el nombre de una región. Sus atributos son los siguientes:

- **Lang code**: Código del idioma en el que se almacena la traducción
- **Region id**: Identificador único de la región para la cual se almacena la traducción
- **Name**: Nombre traducido de la región en el idioma correspondiente

**User** Representa a un usuario que incluye información en el sistema (un importador). Sus atributos son los siguientes:

- **Id**: Identifica de forma única a cada usuario
- **IP**: Dirección IP desde la que se realiza la petición de importación de datos
- **Timestamp**: Momento temporal en el que se recibe la petición de importación

de datos

- **Organization id:** Identificador de la organización a la que pertenece el importador

**Organization** Representa a una organización de la cual se incluyen datos en el sistema.

Sus atributos son los siguientes:

- **Id:** Identifica de forma única a cada organización
- **Name:** Es el nombre de la organización
- **URL:** Dirección del sitio web de la organización
- **Is part of id:** Identificador de la organización a la que pertenece (en el caso de que pertenezca a alguna)

**DataSource** Representa una fuente de datos que aporta información al sistema. Sus atributos son los siguientes:

- **Id:** Identifica de forma única a cada fuente de datos
- **Name:** Es el nombre de la fuente de datos
- **Organization id:** Identificador de la organización a la que pertenece la fuente de datos

**DataSet** Representa un conjunto de datos que se ha incluido en el sistema. Sus atributos son los siguientes:

- **Id:** Identifica de forma única a cada fuente de datos.
- **Sdmx frequency:** Frecuencia con la cual la fuente actualiza sus datos. Los valores para este campo son tomados de la ontología SDMX<sup>14</sup>
- **Datasource id:** Identificador de la fuente de datos a la que pertenece este conjunto de datos
- **License id:** Identificador de la licencia bajo la que se publica este conjunto de datos
- **Indicators:** Indicadores con los que cuenta la fuente de datos.

**OrganizationTranslation** Almacena los datos de una organización en diferentes idiomas. Sus atributos son los siguientes:

- **Lang code:** Código del idioma en el que se almacena la traducción
- **Organization id:** Identificador único de la organización para la cual se almacena la traducción
- **Description:** Descripción traducida de la organización en el idioma correspondiente

**Slice** Representa una slice<sup>15</sup>. Sus atributos son los siguientes:

- **Id:** Identificador único de la slice
- **Indicator id:** Identificador único del indicador con el cual se relaciona el slice

<sup>14</sup>Para más información véase <http://publishing-statistical-data.googlecode.com/svn/trunk/specs/src/main/vocab/sdmx-code.ttl>

<sup>15</sup>El concepto de slice fue explicado en la sección “RDF Data Cube” del capítulo 3

- **Dimension:** Identificador único de la otra dimensión con la que se relaciona el slice (o bien una región o bien un momento temporal)
- **Dataset id:** Identificador único del conjunto de datos del que procede la slice

**Observation** Representa una observación para un indicador y una región en un determinado momento temporal. Sus atributos son los siguientes:

- **Id:** Identificador único de la observación
- **Ref time id:** Identificador único del momento temporal al que la observación hace referencia
- **Issued id:** Identificador único del momento temporal en el que la observación fue incluida en el sistema
- **Computation id:** Identificador único de la computación con la que se relaciona la observación
- **Indicator group id:** Identificador único del grupo de indicadores con el que se relaciona la observación (si se relaciona con alguno)
- **Value id:** Identificador único del valor de la observación
- **Indicator id:** Identificador único del indicador al que pertenece la observación
- **Dataset id:** Identificador único del conjunto de datos al que pertenece la observación
- **Region id:** Identificador único de la región a la que hace referencia la observación
- **Slice id:** Identificador único del slice al que pertenece la observación (en caso de pertencer a alguno)

**Indicator** Representa un indicador sobre el cual se realizan mediciones. Sus atributos son los siguientes:

- **Id:** Identificador único del indicador
- **Preferable tendency:** Representa la tendencia preferible del indicador a lo largo del tiempo
- **Measurement unit id:** Identificador único de la unidad de medida para los valores de las observaciones del indicador
- **Compound indicator id:** Identificador del indicador compuesto con el que se relaciona el indicador (en caso de relacionarse con alguno)
- **Last update:** Fecha de la ultima actualización del indicador o alguna de sus observaciones en el sistema
- **Starred:** Indica si el indicador ha sido marcado como favorito o no
- **Topic id:** Identificador único del tópico al que pertenece el indicador

**IndicatorTranslation** Almacena los datos de un indicador en diferentes idiomas. Sus atributos son los siguientes:

- **Lang code:** Código del idioma en el que se almacena la traducción
- **Indicator id:** Identificador único del indicador para el cual se almacena la

traducción

- **Name:** Nombre del indicador traducido en el idioma correspondiente
- **Description:** Descripción del indicador traducida en el idioma correspondiente

**Topic** Representa un tópico sobre el cual se guardan indicadores. Su único atributo es un código que lo identifica.

**TopicTranslation** Almacena los datos de un tópico en diferentes idiomas. Sus atributos son los siguientes:

- **Lang code:** Código del idioma en el que se almacena la traducción
- **Indicator id:** Identificador único del tópico para el cual se almacena la traducción
- **Name:** Nombre del tópico traducido en el idioma correspondiente

**IndicatorGroup** Representa una agrupación de indicadores compuestos. Su único atributo es un código que lo identifica.

**MeasurementUnit** Representa a una unidad de medida. Sus atributos son los siguientes:

- **Id:** Identificador único de la unidad de medida
- **Name:** Nombre de la unidad de medida
- **Convertible to:** Unidad estándar a la que es convertible<sup>16</sup>
- **Factor:** Factor de conversión a aplicar al valor para convertirlo a la unidad estándar

**License** Representa una licencia bajo la cual se publica algún conjunto de datos almacenado en el sistema. Sus atributos son los siguientes:

- **Id:** Identificador único de la licencia
- **Name:** Nombre de la licencia
- **Description:** Descripción larga de la licencia
- **Republish:** Indica si la licencia autoriza o no a republicar los datos
- **URL:** Sitio web de la licencia

**Computation** Representa una computación que se realiza sobre los valores de una observación. Sus atributos son:

- **Id:** Identificador único de la computación
- **Name:** URL de la computación<sup>17</sup>
- **Description:** Descripción de la computación

**Value** Representa un valor de una observación. Sus atributos son:

---

<sup>16</sup>Por ejemplo: si la unidad de medida fueran centímetros, la unidad estándar a la que es convertible serían metros

<sup>17</sup>Las URL de las computaciones son tomadas de la ontología WESO-Computex, para más información véase <https://raw.githubusercontent.com/weso/computex/master/ontology/computex.rdf>

- **Id:** Identificador único del valor
- **Obs status:** Estado de la observación<sup>18</sup>
- **Value:** Almacena el valor concreto
- **Value type:** Indica el tipo del valor almacenado

**IndicatorRelationship** Representa una relación entre varios indicadores. Su único atributo es un código que identifica la relación. Existen dos tipos de relaciones: **Becomes**, que representa un indicador que se transforma en otro a lo largo del tiempo y *IsPartOf*, que representa un indicador que forma parte de otro

**Dimension:** Representa una dimensión tal y como se define en el RDF Data Cube. Su único atributo es un código que la identifica

**Time** Representa una unidad de tiempo. Hereda los atributos de **Dimension**

**Instant** Representa un momento concreto en el tiempo. Además de los atributos que hereda de **Time** también incluye:

- **Timestamp:** Momento concreto en el tiempo al que hace referencia

**Interval** Representa un intervalo arbitrario de tiempo. Además de los atributos que hereda de **Time** también incluye:

- **Start time:** Momento en el que comienza el intervalo de tiempo
- **End time:** Momento en el que finaliza el intervalo de tiempo
- **Value:** Valor del intervalo en forma de texto para su posterior presentación en las vistas

**YearInterval** Representa un intervalo de un año. Además de los atributos que hereda de **Interval** también incluye:

- **Year:** Año concreto al que hace referencia el intervalo

**MonthInterval** Representa un intervalo de un mes. Además de los atributos que hereda de **Interval** también incluye:

- **Year:** Año concreto al que hace referencia el intervalo
- **Month:** Mes concreto al que hace referencia el intervalo

**Region** Representa una región sobre la que se almacenan observaciones. Además de los atributos que hereda de **Dimension** también tiene:

- **Un code:** Será el código numérico asignado por la División Estadística de las Naciones Unidas<sup>19</sup> para la región
- **Is part of id:** Identificador único de la región de la que forma parte (en caso de formar parte de alguna)

**Country** Representa un país sobre el que se almacenan observaciones. Además de los atributos que hereda de **Region** también tiene:

- **Fao URI:** URI única del país en la Ontología Geopolítica de la Organización

---

<sup>18</sup>Este campo permite distinguir entre aquellas observaciones nulas y aquellas observaciones que no cuentan con un valor. Esta distinción es necesaria para asegurar la calidad estadística de los datos.

<sup>19</sup>Este código recibe el nombre formal de “ISO 3166-1 numeric”. En [Div91] y [Sta] puede encontrarse más información sobre estos códigos.

para la Alimentación y la Agricultura de las Naciones Unidas<sup>20</sup> “Análisis preliminar de clases de la zona de datos” perteneciente al capítulo 4

- **ISO2:** Código alfabético de dos letras asignado por la división estadística de las Naciones Unidas<sup>21</sup>
- **ISO3:** Código alfabético de tres letras asignado por la división estadística de las Naciones Unidas<sup>22</sup>
- **Taxonomy id:** Identificador del país en la taxonomía creada por el gestor de contenidos

**CompoundIndicator** Representa un indicador compuesto de varios indicadores simples. Además de los atributos que hereda de **Indicator** también incluye:

- **Indicator ref group id:** Identificador único del grupo de indicadores al que pertenece este indicador compuesto

### 5.3.2. Modelo de la zona social

#### 5.3.2.1. Diagrama del modelo de datos

La figura 5.9 muestra el diagrama con el diseño final del modelo de datos de la zona social.

---

<sup>20</sup>La información sobre esta ontología se encuentra disponible en [FU]

<sup>21</sup>Este código recibe el nombre formal de “ISO 3166-1 alpha-2”

<sup>22</sup>Este código recibe el nombre formal de “ISO 3166-1 alpha-3”

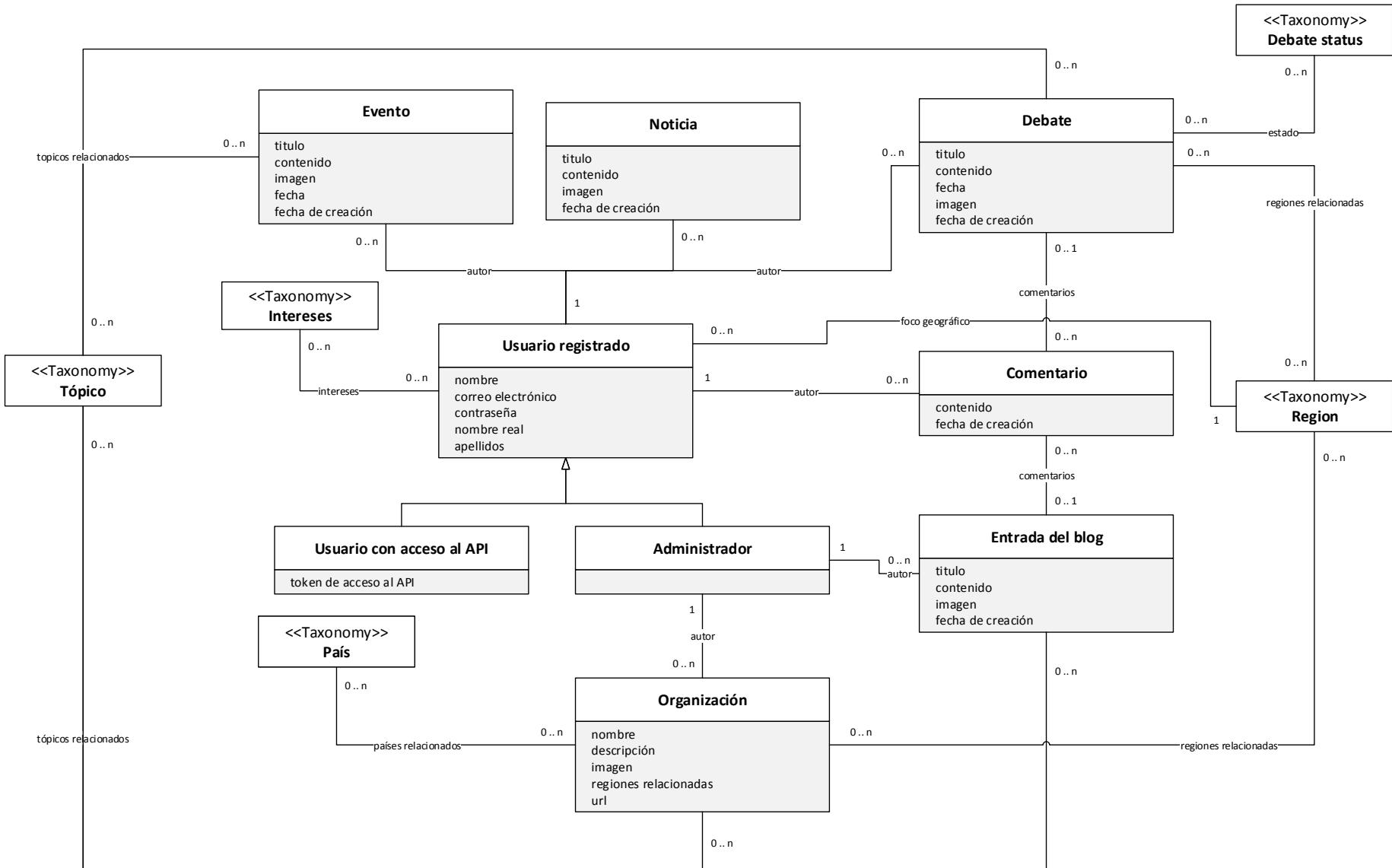


Figura 5.9: Diseño final del modelo de datos de la zona social

La variación respecto al diseño preliminar de este modelo que se puede observar en la sección “Análisis preliminar de clases de la zona social” perteneciente al capítulo 4 es mínima y se explicará a continuación:

- Los **tópicos relacionados** de los eventos, debates, organizaciones y entradas del blog han sido extraídos a una taxonomía externa.
- Los **países relacionados** de las organizaciones también han sido extraídos a una taxonomía externa.
- Al igual que los anteriores, las **regiones relacionadas** de las organizaciones y los debates también han sido extraídos a una taxonomía externa.
- El **estado** de los debates también pertenece a una taxonomía externa.
- En respuesta a lo reportado por los clientes en las pruebas de aceptación (véase la sección 7.3) a los usuarios se les ha añadido un campo **intereses** donde podrán especificar las temáticas que les resultan interesantes. También se han unificado los campos *región* y *países relacionados* en un único campo llamado **foco geográfico**. Tanto el campo *intereses* como el campo *foco geográfico* pertenecen a una taxonomía externa.

La razón principal del uso de taxonomías es la capacidad de enlazar los diferentes contenidos del portal. Por ejemplo, ver un elemento de la taxonomía *Tópico* permitirá acceder a todos los eventos, debates, organizaciones y entradas del blog relacionados con dicho tópico.

En cuanto a los debates, la existencia de una taxonomía que represente su estado permite añadir o eliminar nuevos estados de forma sencilla y sin tener que realizar ninguna modificación en los componentes del modelo.

### 5.3.2.2. Elementos del modelo de datos

A pesar de la inclusión de taxonomías para representar algunos elementos del modelo de datos de la zona social, este no ha sufrido ninguna variación respecto a lo ya expuesto en la fase de análisis del sistema. Todas las descripciones de campos y elementos realizadas en la sección “Análisis preliminar de clases de la zona social” del capítulo 4 siguen siendo válidas.

### 5.3.3. Sistema de gestión de base de datos

Puesto que, tal y como se explicó en las secciones “Vista del módulo *landportal uris*” y “Vista del punto de entrada de datos” pertenecientes a este mismo capítulo, el subsistema de datos está formado por dos componentes diferentes (uno de ellos incluido dentro del gestor de contenidos), se utilizarán dos aproximaciones distintas para acceder a la base de datos.

En primer lugar, el punto de entrada de datos utilizará un ORM (*Mapeador Objeto-Relacional*) que actúe como intermediario con la base de datos. De esta forma durante la inserción de datos no se tendrá que recurrir a realizar consultas SQL manualmente y el ORM será el encargado de persistir los nuevos objetos y actualizar el estado de los ya existentes.

Por otra parte, el framework que da soporte a la creación de visualizaciones y obtiene los datos con los que rellenar las vistas del CMS utilizará consultas manuales en lenguaje SQL. En este caso se ha optado por la realización de consultas directamente en lenguaje SQL principalmente por la necesidad de reducir tanto el número de consultas como el número de datos innecesarios que se piden a la base de datos. La necesidad de

optimización viene dada por los grandes volúmenes de datos con los que se trabajará, teniendo lugar además estas transacciones en un momento crítico debido a que el usuario estará esperando activamente a que el sistema le muestre los datos y, por tanto, el tiempo de respuesta deberá ser el mínimo posible<sup>23</sup>.

En cuanto al sistema de gestión de bases de datos, se ha utilizado una base de datos MySQL debido a su probada estabilidad y rendimiento, además de ser la base de datos utilizada nativamente por Drupal.

## 5.4. Diseño de la interfaz de usuario

En la sección “Análisis de interfaces de usuario” perteneciente al capítulo 4 se ha mostrado una primera aproximación al posible diseño de las interfaces del sistema. A continuación se mostrará el diseño final de las interfaces y se explicarán las variaciones que puedan existir entre ambos diseños.

En la sección “Definición del sistema” perteneciente al capítulo 4 también se explicó que la interfaz de zona de datos (o *LandBook*) queda fuera del alcance de este proyecto, y la interfaz de administración será directamente proveída por el gestor de contenidos. Debido a esto, únicamente se mostrarán las interfaces de la zona social (o *LandDebate*).

### 5.4.1. Consideraciones generales

Puesto que, como ya se ha indicado anteriormente en la sección “Requisitos del sistema” perteneciente al capítulo 4, el sistema soportará internacionalización tanto de interfaces como de contenidos, se incluirán capturas en diferentes idiomas para mostrar esta capacidad.

En general, las vistas pertenecientes a la zona social cuentan con una cabecera similar. Se ha decidido utilizar esta cabecera para dar cohesión a todas las partes del portal. Otra razón para incluir esta cabecera es situar de forma accesible las opciones más comunes para los usuarios, dichas opciones permiten iniciar una sesión en el portal y acceder a las diferentes secciones que lo componen.

### 5.4.2. Vista del blog del Land Portal

La figura 5.10 muestra la interfaz final del blog del Land Portal, el mockup original de dicha vista puede observarse en la imagen 4.13. Como se puede observar no hay muchas variaciones respecto al diseño original.

La descripción original de los elementos de esta vista puede consultarse en la sección 4.7.3.1 perteneciente al capítulo 4.

### 5.4.3. Vista de detalle de una entrada del blog

La figura 5.11 muestra la interfaz final de una entrada del blog. Como se puede comprobar no hay muchas variaciones respecto al diseño original, el cual puede observarse en la imagen 4.14.

<sup>23</sup>Para intentar reducir aún más el tiempo de respuesta, también se utiliza un sistema de caché para evitar pedir en múltiples ocasiones los mismos datos. El funcionamiento del caché puede consultarse en la sección “Vista del punto de entrada de datos” del presente capítulo

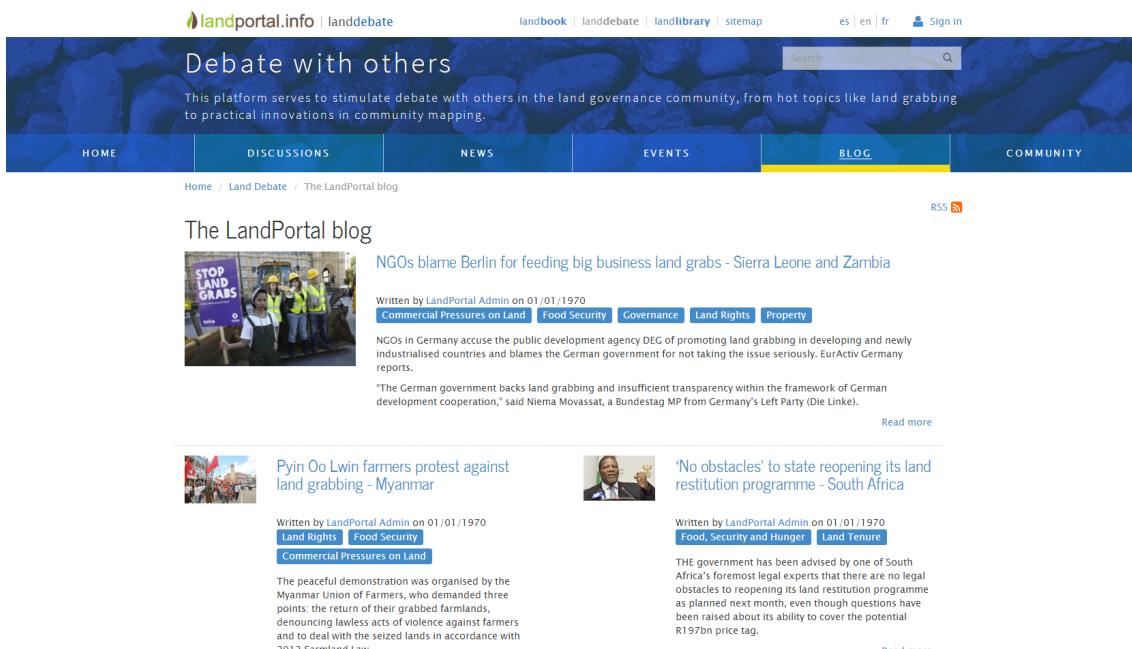


Figura 5.10: Captura de la vista del blog del Land Portal (inglés)

La descripción original de los elementos de esta vista puede consultarse en la sección 4.7.3.2 perteneciente al capítulo 4.

#### 5.4.4. Vista de eventos

La figura 5.12 muestra la interfaz final de la vista de eventos. Como se puede comprobar no hay muchas variaciones respecto al diseño original, el cual puede observarse en la imagen 4.15.

La descripción original de los elementos de esta vista puede consultarse en la sección 4.7.3.3 perteneciente al capítulo 4.

#### 5.4.5. Vista de noticias

La interfaz final de la vista de noticias puede verse en la figura 5.13, el diseño original para esta vista puede observarse en la imagen 4.16. La descripción original de los elementos de esta vista fue realizada anteriormente en la sección 4.7.3.4 perteneciente al capítulo 4.

#### 5.4.6. Vista de debates

La figura 5.14 muestra el diseño final de la vista de debates. Al igual que en el mockup original (imagen 4.17), los debates más recientes se muestran de forma destacada y los últimos comentarios se muestran ordenados de más a menos recientes.

Las etiquetas en las que se especifica el estado de cada debate se han coloreado de forma diferente para cada estado. Este cambio ha sido motivado por los reportes recibidos de los usuarios durante las pruebas de aceptación (los resultados de éstas pruebas pueden verse con detalle en la sección 7.3).

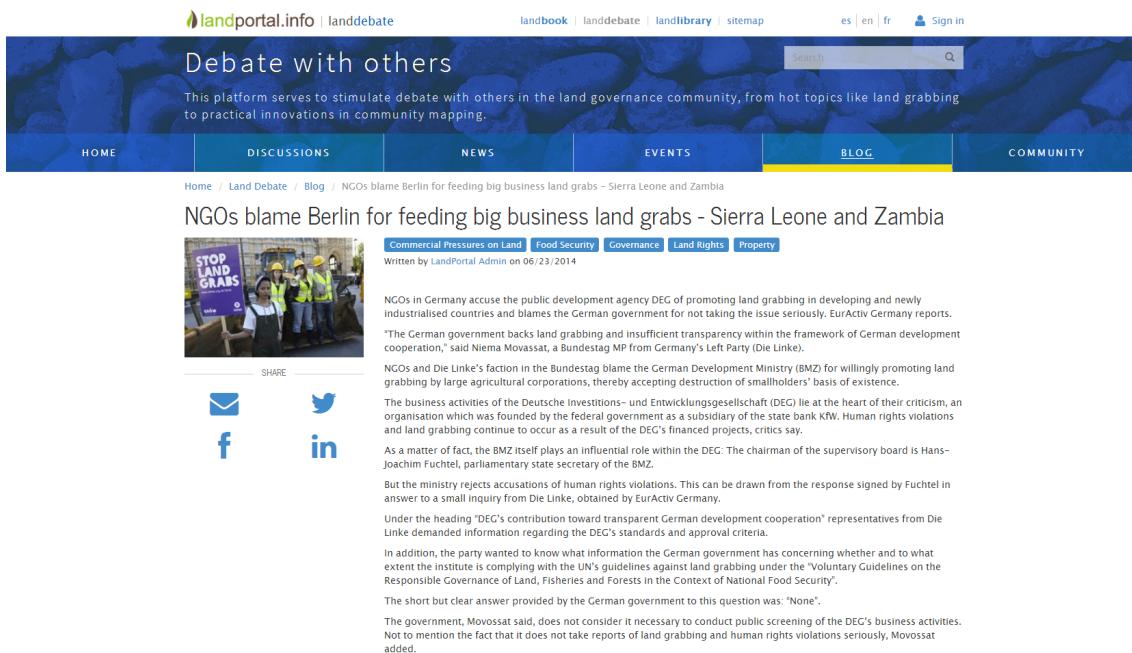


Figura 5.11: Captura de la vista de detalle de una entrada del blog (inglés)

Como se puede comprobar, el autor de cada debate se muestra junto a su imagen de perfil.

#### 5.4.7. Vista de detalle de un debate

La figura 5.15 muestra el diseño final de la vista de detalle de un debate, como se puede observar en la figura 4.18 a penas ha habido variaciones respecto al mockup original. En la imagen puede verse que el debate está abierto y permite nuevos comentarios de los usuarios así como respuestas a comentarios ya existentes.

#### 5.4.8. Vista de detalle de una noticia y evento

En la sección 4.7.3.7 perteneciente al capítulo 4 se explicó que la vista de detalle de una noticia y un evento sería similar a la vista de una entrada del blog (figura 5.11). La única diferencia sería que las vistas de una noticia o un evento no tendrían sección de comentarios.

Las figuras 5.16 y 5.17 muestran respectivamente las vistas de detalle de una noticia y un evento. Como se puede ver guardan una gran similitud entre sí y respecto al mockup original.

#### 5.4.9. Vista de organizaciones

La figura 5.18 muestra el diseño final de la vista de organizaciones. Al igual que el mockup original (ver la figura 4.19), esta vista incluye unos controles que permiten la búsqueda de organizaciones por nombre, tópicos relacionados, regiones de operación o países en los que trabajan. También se incluye un widget de la red social Facebook, para permitir a los usuarios entrar a formar parte de la comunidad de LandPortal en dicha red social.

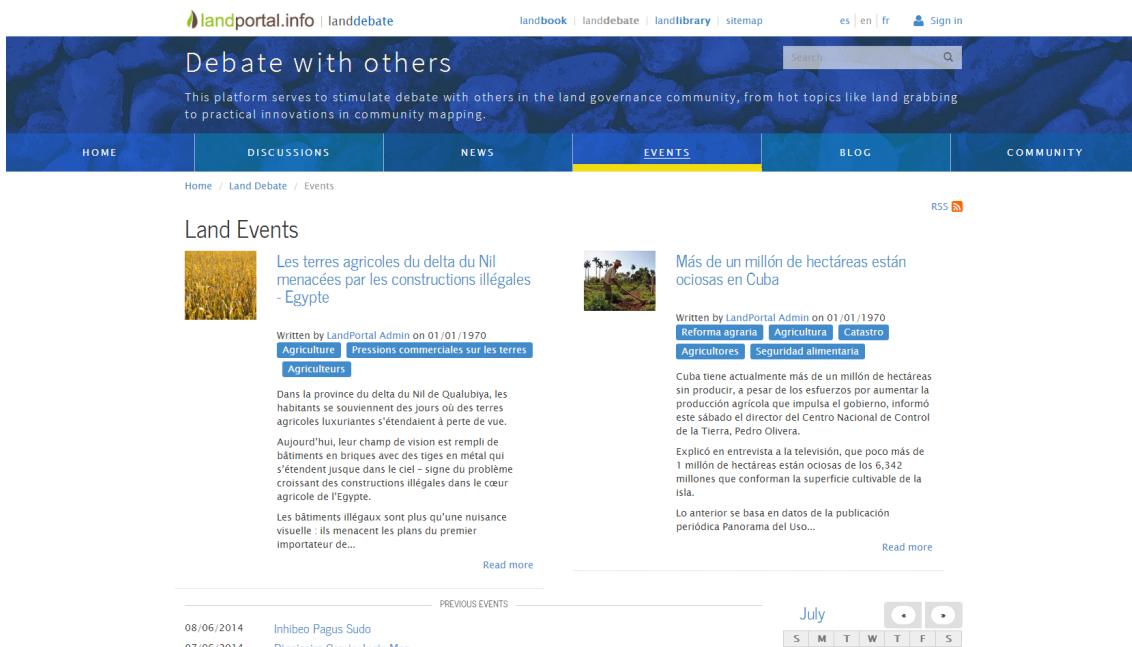


Figura 5.12: Captura de la vista de eventos (inglés)

#### 5.4.10. Vista de detalle de una organización

La figura 5.19 muestra el diseño final de la vista de organizaciones. Esta vista no ha sufrido variaciones respecto a su mockup original, dicho mockup puede verse en la figura 4.20.

#### 5.4.11. Vista de login

La interfaz final de la vista de login puede verse en la figura 5.20, el mockup original de esta vista puede observarse en la figura 4.21.

El diseño final sigue las líneas generales del mockup, pero se han añadido unos *placeholders*<sup>24</sup> a los campos de texto para facilitar la comprensión del usuario.

#### 5.4.12. Vista de registro

La figura 5.21 muestra el diseño final de la vista de registro en el sistema, como se puede observar, esta imagen se ha incluido en idioma español. El mockup original de esta vista puede observarse en la figura 4.22.

Para aumentar la usabilidad de esta vista, se han incluido unas pequeñas descripciones explicando las reglas para los campos de nombre de usuario y correo electrónico. Los campos para introducir las regiones y países relacionados cuentan también con un sistema de autocompletado utilizando llamadas AJAX. Por último, los campos requeridos se han marcado con un asterisco de color rojo con objetivo de hacer más fácil su identificación.

<sup>24</sup>Un *placeholder* es una pequeña guía que describe lo que se espera recibir en un campo de entrada y desaparece cuando el usuario empieza a escribir en él.

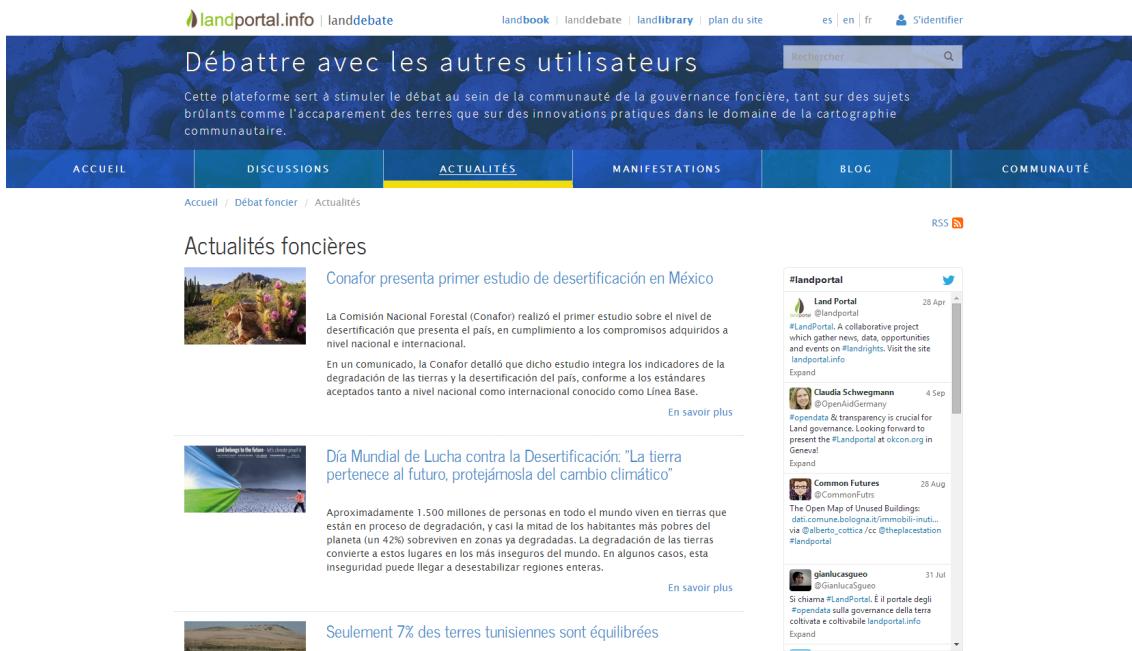


Figura 5.13: Captura de la vista de noticias (francés)

#### 5.4.13. Vista del perfil de un usuario

La figura 5.22 muestra la vista del perfil de un usuario. Puesto que esta vista no figuraba entre los mockups de la sección “Análisis de interfaces de usuario” se describirán aquí sus características:

- El nombre de usuario se muestra de forma destacada como título de la vista
- El resto de información del usuario se muestra a continuación, con un tamaño más pequeño. Entre los campos de usuario merece la pena destacar el correo electrónico y la clave de acceso al API.
  - Para facilitar el proceso de contactar con el usuario el correo electrónico se incluye como un enlace
  - La clave de acceso al API sólo se mostrará en aquellos usuarios que tengan permisos de acceso al API. En cualquier caso, dicha clave no se mostrará al resto de usuarios que visiten el perfil
- Los países y regiones en los que el usuario esté interesado también se mostrarán como enlaces, permitiendo cada uno de ellos acceder al resto de elementos del portal etiquetados con el mismo término
- Cuando un usuario visite su propio perfil, se incluye un botón que permite editar la información del perfil. Este botón no se mostrará al resto de usuarios.
- Se incluirán también los contenidos creados por el usuario cuyo perfil se está visitando. Cada contenido se mostrará con su título (enlazado al contenido completo), su URL, un pequeño resumen del contenido y una pequeña etiqueta que identifica el tipo del contenido (si es un debate, una noticia, etc).

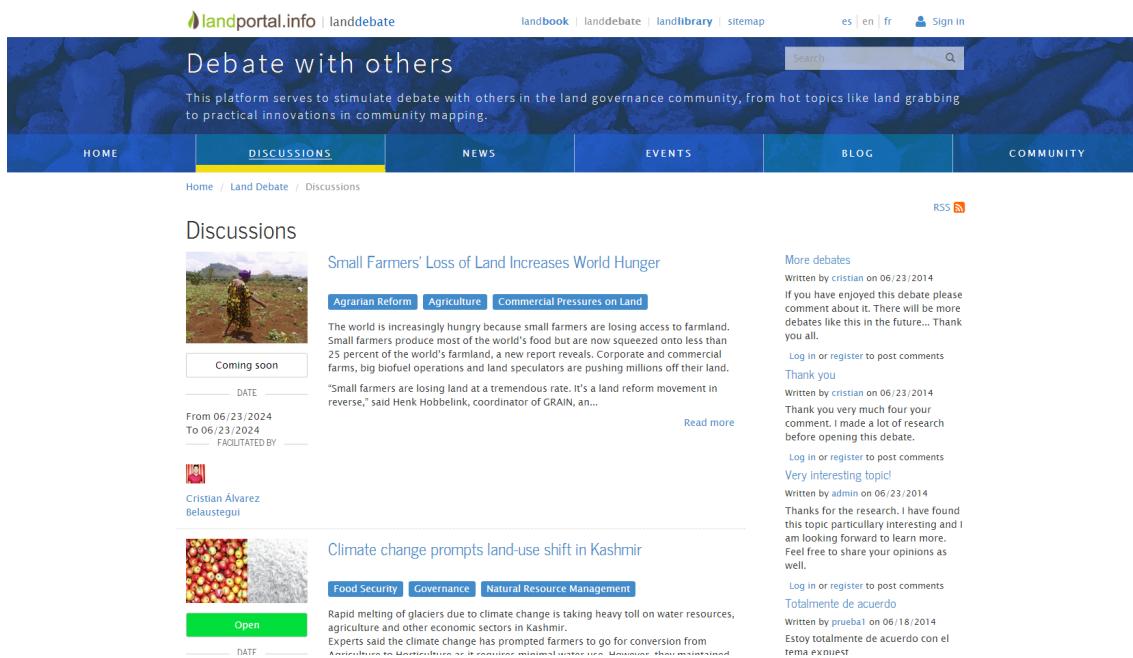


Figura 5.14: Captura de la vista de debates (inglés)

#### 5.4.14. Vista de búsqueda

La figura 5.23 muestra el diseño final de la vista de búsqueda. El mockup original de esta vista puede observarse en la figura 4.23.

Un pequeño detalle que se ha incluido en esta vista han sido las etiquetas de colores para facilitar a los usuarios la distinción de los tipos de contenido de los resultados.

#### 5.4.15. Diseño adaptable

La interfaz del nuevo Land Portal cuenta con un diseño adaptable o *responsive* con el fin de que pueda mostrarse correctamente aprovechando el espacio ofrecido por los distintos dispositivos como *smartphones* o *tablets*.

La figura 5.24 muestra la interfaz de la vista del blog en un tamaño de pantalla pequeño y la interfaz de la vista de los debates en un tamaño de pantalla intermedio.

Figura 5.15: Captura de la vista de detalle de un debate (inglés)

Figura 5.16: Captura de la vista de detalle de una noticia (francés)

The screenshot shows a web page from landportal.info. At the top, there's a navigation bar with links for 'landbook', 'landdebate', 'landlibrary', 'sitemap', language options ('es | en | fr'), and a 'Sign in' button. Below the header, a banner reads 'Debate with others' and describes the platform as a space for stimulating debate in the land governance community. The main content area features a large image of a field. Below it, the title of the event is displayed: 'Les terres agricoles du delta du Nil menacées par les constructions illégales - Egypte'. The article is categorized under 'Agriculture', 'Pressions commerciales sur les terres', and 'Agriculteurs'. It was written by LandPortal Admin on 06/23/2014. The text discusses agricultural land loss due to illegal construction in the Nile Delta, mentioning the loss of 12,140 hectares annually. It also touches on the impact of urban sprawl and the role of the Ministry of Agriculture. Social sharing icons for email, Twitter, Facebook, and LinkedIn are present.

Figura 5.17: Captura de la vista de detalle de un evento (inglés)

This screenshot shows the 'Comunidad' (Community) section of the landportal.info website. The top navigation bar includes links for 'landbook', 'landdebate', 'landlibrary', 'mapa web', language options ('es | en | fr'), and a 'Iniciar sesión' (Sign in) button. The main content area is titled 'Debate con usuarios' and describes the platform as a space for organizing debates in the soil management community. Below this, there's a section titled 'Comunidad' featuring logos of several organizations: WWF (World Wide Fund for Nature), IFPRI (International Food Policy Research Institute), OECD (The Organisation for Economic Co-operation and Development), World Bank, World Health Organization, and UNDP (United Nations Development Programme). To the right, there are search fields for 'Tópicos relacionados', 'Países relacionados', 'Regiones relacionadas', and a 'Nombre' (Name) input field. A 'Buscar' (Search) button is also present. A sidebar on the right shows a Facebook plugin with a 'Land Portal' page and a 'Me gusta' (Like) count of 546.

Figura 5.18: Captura de la vista de organizaciones (español)

The Organisation for Economic Co-operation and Development (OECD) is a forum where governments work together to share experiences and seek solutions to common problems. We work with governments to understand what drives economic, social and environmental change. We measure productivity and global flows of trade and investment. We analyse and compare data to predict future trends. We set international standards on a wide range of things, from agriculture and tax to the safety of chemicals. We look, too, at issues that directly affect the lives of ordinary people, like how much they pay in taxes and social security, and how much leisure time they can take. We compare how different countries' school systems are readying their young people for modern life, and how different countries' pension systems will look after their citizens in old age. Drawing on facts and real-life experience, we recommend policies designed to make the lives of ordinary people better. We work with business, through the Business and Industry Advisory Committee to the OECD, and with labour, through the Trade Union Advisory Committee. We have active contacts as well with other civil society organisations. The common thread of our work is a shared commitment to market economies backed by democratic institutions and focused on the wellbeing of all citizens.

Along the way, we also set out to make life harder for the terrorists, tax dodgers, crooked businessmen and others whose actions undermine a fair and open society.

[Visita su página](#)

Figura 5.19: Captura de la vista de detalle de una organización (español)

Figura 5.20: Captura de la vista de login (inglés)

The screenshot shows the registration page for landportal.info. At the top, there is a logo with three green leaves and the text "landportal.info". To the right of the logo are links for "es | en | fr". Below the header, a green banner with white text reads "Únete a nosotros" (Join us) and "¡Bienvenido al Land Portal! Rellena el formulario para crear una cuenta. Una vez enviado recibiremos una notificación para aprobar tu solicitud." (Welcome to the Land Portal! Fill out the form to create an account. Once sent, we will receive a notification to approve your request.). Below the banner, there are two buttons: "INICIO DE SESIÓN" (Login) on the left and "REGISTRAR" (Register) on the right, with "REGISTRAR" being highlighted in yellow. The main form area contains several input fields with validation messages:

- Nombre de usuario \***: An input field with a placeholder message: "Los espacios están permitidos. Los signos de puntuación no están permitidos (excepto puntos, guiones, apóstrofes y guiones bajos)".
- Correo \***: An input field with a placeholder message: "Una dirección de correo válida. Todos los correos del sistema serán enviados a ésta dirección. La dirección de correo electrónico no se hará pública y sólo se utilizará para recibir una nueva contraseña y algunas notificaciones del sistema."
- Nombre \***: An input field.
- Apellidos \***: An input field.
- Foco geográfico**: A dropdown menu with one visible option.
- Intereses**: A dropdown menu with one visible option.

A large green button at the bottom center of the form says "Registrarse" (Register).

Figura 5.21: Captura de la vista de registro (español)

The screenshot shows the user profile for 'cristian'. At the top, there's a banner with the text 'Debate con usuarios' and a subtext about organizing debates in soil management. Below the banner, a navigation bar includes links for INICIO, DIÁLOGOS, NOTICIAS, EVENTOS, BLOG, and COMUNIDAD. The main content area displays the user's profile information: Nombre (cristian), Apellidos (Álvarez Belaustegui), Correo (cristian.alvb@gmail.com), and Clave de acceso al API (846f266fd51fb2b855b5ad174fbbe75d667d7cd284f669dd57fce65acde55). It also shows the user joined on 01/01/1970. A section titled 'Contenido creado' lists a post about small farmers losing land to world hunger. On the right, there's a profile picture of a man, geographical focus (Global, España, Europa), interests (Karate, Informática, Tecnología, Deportes), and buttons for 'Editar' and 'Cerrar sesión'.

Figura 5.22: Captura de la vista del perfil de un usuario (español)

The screenshot shows search results for 'Spain'. The top navigation bar has links for Accueil and Rechercher. A search bar contains the word 'spain'. The results list includes 'Otros debate de prueba /node/1895' (Wednesday, June 18, 2014 Coming soon Español ...), 'Debate de prueba /node/1895' (Wednesday, June 18, 2014 Open English ...), and 'Refovéo /node/122' (República Centroafricana Sierra Leona Samoa américaines Barbade Spain Afrique http://localhost:1100/ ...). There are also sections for 'DÉBAT' and 'PAYS'.

Figura 5.23: Captura de la vista de búsqueda (francés)

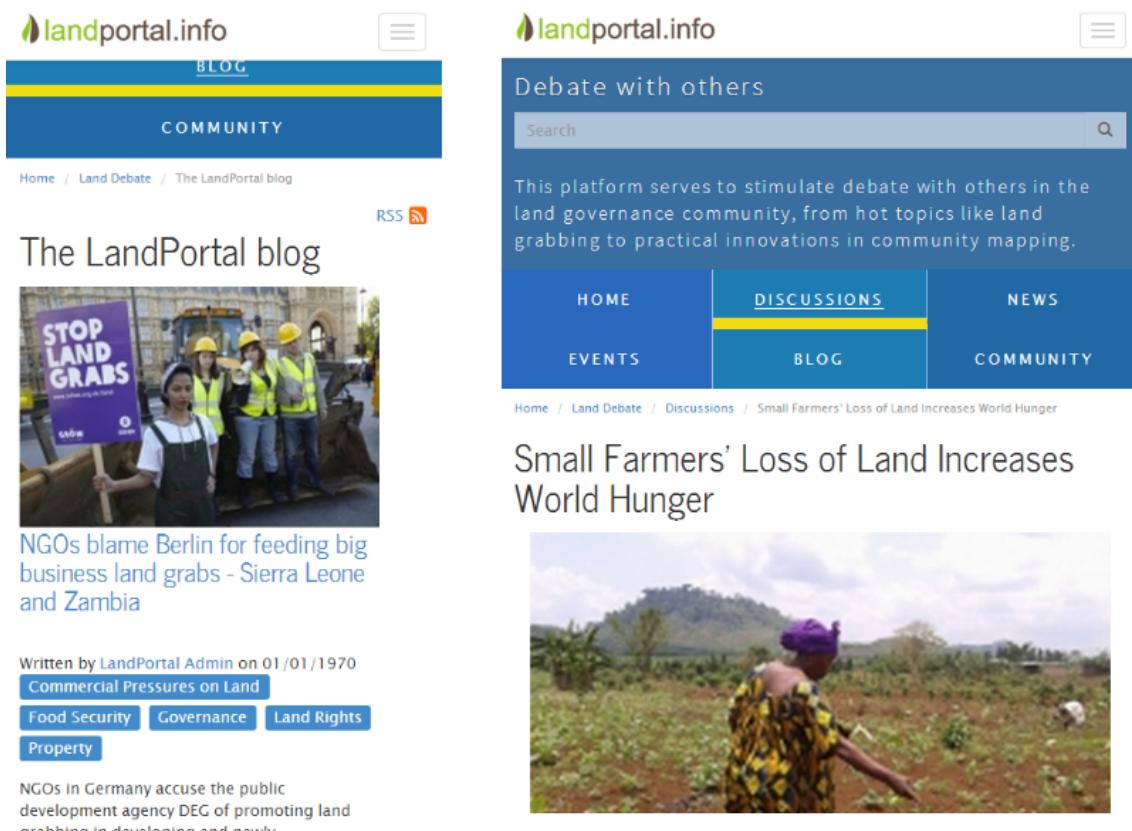


Figura 5.24: Captura del diseño adaptable (inglés)

# Capítulo 6

## Implementación del sistema

En este capítulo se detallarán los aspectos más importantes de la implementación del sistema, desde los lenguajes de programación y herramientas utilizadas hasta los principales problemas surgidos durante el desarrollo y sus soluciones.

### 6.1. Lenguajes de programación utilizados

A continuación se describirán los lenguajes de programación utilizados y sus respectivas versiones.

**PHP** Para el desarrollo de los módulos pertenecientes al gestor de contenidos, así como las plantillas del tema visual, se ha utilizado el lenguaje PHP (*PHP: Hypertext Preprocessor*). La versión del lenguaje PHP utilizada es la 5.5<sup>1</sup>. La única extensión de PHP utilizada ha sido APC (*Alternative PHP Cache*), que permite cachear y optimizar el código intermedio de PHP con el objetivo de aumentar el rendimiento siempre que sea posible, el uso de este caché puede verse descrito en la sección “Funcionamiento del framework de soporte a visualizaciones” perteneciente al capítulo 5.

**Python** Para el desarrollo del Punto de Entrada de Datos (o *Receiver*) se ha utilizado el lenguaje Python en su versión 2.7. Se ha decidido utilizar la versión 2.7 de Python debido a que las versiones 3.x contienen varias incompatibilidades con las anteriores versiones, estas incompatibilidades provocan que la existencia de librerías externas sea más reducida. En el PEP (*Python Enhancement Proposal*) 373<sup>2</sup> se establece que la versión 2.7 de Python estará soportada hasta el año 2020. En la posterior sección “Herramientas utilizadas” se explicarán las herramientas que se han utilizado para aislar el entorno y gestionar las dependencias externas.

**SQL** Para interactuar con la base de datos se ha utilizado el lenguaje SQL, Cabe destacar que todas las consultas realizadas se adhieren a la especificación de SQL estándar y no utilizan características propias de ningún sistema de gestión de bases de datos.

**Otros** A pesar de no ser lenguajes de programación como tal, también se incluirán algunos lenguajes que se han utilizado en varias partes del sistema.

- **XML** Para el intercambio de datos entre los Importadores y el Punto de

<sup>1</sup>La versión 5.5 del lenguaje PHP incluye, entre otras características, soporte a generadores y capacidad de iteración por cualquier tipo de clave (no sólo numérica) en los bucles *foreach*. Para una mayor información al respecto se recomienda leer el anuncio oficial en [http://php.net/releases/5\\_5\\_0.php](http://php.net/releases/5_5_0.php)

<sup>2</sup>El PEP 373 está disponible para su consulta en <http://legacy.python.org/dev/peps/pep-0373/>

Entrada de Datos se ha utilizado el lenguaje de marcas XML (*Extensible Markup Language*). A pesar de la existencia de otros lenguajes con un objetivo similar (JSON es uno de ellos y se verá a continuación) se ha decidido utilizar XML para esta tarea por su capacidad para validar que la estructura del documento recibido satisface un esquema dado.

- **JSON** Para el envío de datos a las vistas y visualizaciones se ha utilizado el formato JSON (*JavaScript Object Notation*). Se ha escogido JSON por ser un formato mucho más ligero que XML. Además, puesto que en estos casos los datos provienen desde dentro del sistema su integridad está garantizada.
  - El formato JSON también se ha utilizado para albergar varios ficheros de configuración, como se puede ver en la “Vista del módulo *landportal uris*”. A pesar de la existencia de otros formatos como YAML (*YAML Ain't Markup Language*) se ha decidido utilizar JSON por consistencia con el resto de partes del sistema.
- **HTML y CSS** Toda la interfaz web del portal se ha desarrollado utilizando el lenguaje HTML 5 para crear el contenido de las diferentes páginas, así como el lenguaje CSS 3 para dar estilo visual a las mismas.
- **Mustache** Para facilitar la creación de plantillas HTML se ha utilizado Mustache<sup>3</sup>. Mustache define sus plantillas como *logic-less* debido a que no cuenta con ningún tipo de estructura de control (*if*, *for*, *while*, etc) características de los lenguajes de programación imperativos, sino que construye las plantillas HTML expandiendo las etiquetas especificadas con una serie de valores que se envíen a través de un diccionario u objeto.

## 6.2. Herramientas utilizadas

A continuación se describirán las herramientas utilizadas durante el desarrollo del sistema.

**PyCharm** Para el desarrollo del Punto de Entrada de Datos se ha utilizado el entorno de desarrollo integrado PyCharm en su versión 3.3 y 3.4. PyCharm es un IDE para el desarrollo en lenguaje Python desarrollado por JetBrains<sup>4</sup>. PyCharm se ha utilizado en su versión comercial debido a su soporte para diversos frameworks de desarrollo web, en concreto para el framework *Flask* que se ha utilizado en el proyecto.

**Editor de texto** Para el desarrollo del código PHP que forma parte de los módulos de Drupal se ha utilizado en primer lugar el editor de texto Sublime Text 2<sup>5</sup> y posteriormente el editor Atom<sup>6</sup>. Se ha decidido cambiar desde Sublime Text a Atom debido a ser este último un producto de código abierto, gratuito y con un ritmo de desarrollo muy activo.

**VirtualEnv** Se ha utilizado la herramienta VirtualEnv<sup>7</sup> para crear entornos de Python aislados. Utilizar entornos aislados permite instalar librerías y paquetes sin que colisionen entre los diferentes entornos existentes, también permite mantener múltiples intérpretes de Python funcionando de forma simultánea.

**PIP** En conjunción con la herramienta VirtualEnv mencionada anteriormente, también

---

<sup>3</sup><http://mustache.github.io/>

<sup>4</sup><http://www.jetbrains.com/pycharm/>

<sup>5</sup><http://www.sublimetext.com/2>

<sup>6</sup><https://atom.io/>

<sup>7</sup><http://virtualenv.readthedocs.org/en/latest/virtualenv.html>

se ha utilizado la herramienta PIP<sup>8</sup> para gestionar los distintos paquetes y dependencias. PIP permite descargar e instalar de forma automática paquetes externos junto a sus dependencias, también permite especificar todas las dependencias en un único fichero de texto.

**Git y GitHub** Durante todo el desarrollo se ha utilizado Git<sup>9</sup> como sistema de control de versiones. En conjunción con Git, se ha utilizado GitHub<sup>10</sup> como *hosting* para los distintos repositorios que forman parte del sistema.

**Flask** Para el desarrollo del Punto de Entrada de Datos se ha utilizado Flask<sup>11</sup>. Flask es un framework para el desarrollo de aplicaciones web escrito en el lenguaje Python. Flask tiene una licencia BSD. Se ha decidido utilizar Flask en lugar de otros frameworks como Django<sup>12</sup> debido a que Flask se define a sí mismo como un *microframework*, lo que provoca que su núcleo sea muy simple y cuente con una gran capacidad de extensión.

## 6.3. Problemas encontrados

Se describirán a continuación los problemas técnicos encontrados durante el desarrollo del sistema y las soluciones que se han dado a cada uno de ellos.

### 6.3.1. Internacionalización de datos

Como ya se explicó en la sección “Requisitos de la sección de datos”, no sólo era necesario internacionalizar las vistas del sistema si no también los propios datos. La internacionalización de los datos supuso un gran reto durante la construcción del sistema, puesto que requirió ser soportada por todas las partes que lo componen.

La solución para la internacionalización de los datos consistió en (como se puede ver en la sección “Diseño del modelo de datos” perteneciente al capítulo 5) la creación de varias clases en el modelo de datos destinadas al almacenamiento de las traducciones en diferentes idiomas. Como contrapunto a esta solución, el modelo de datos adquirió una gran complejidad, con 30 clases diferentes. Los problemas derivados de esta complejidad se detallarán en el punto siguiente.

### 6.3.2. Complejidad del modelo de datos

Tal y como se mencionó en el punto anterior y se puede comprobar en la sección “Diseño del modelo de datos” el modelo de datos adquirió una gran complejidad, alcanzando las 30 clases diferentes.

Esta complejidad proviene principalmente de la necesidad de cumplir con la especificación del vocabulario RDF Data Cube (este concepto fue explicado en detalle en la sección 3.4 perteneciente al capítulo “Conceptos teóricos”) y de la necesidad de internacionalizar los datos del sistema.

La solución a este problema consistió en asumir dicha complejidad y tenerla siempre presente de cara a optimizar la inserción y la extracción de datos en la medida de lo posible.

---

<sup>8</sup><https://pypi.python.org/pypi/pip>

<sup>9</sup><http://www.git-scm.com/>

<sup>10</sup><https://github.com/>

<sup>11</sup><http://flask.pocoo.org/>

<sup>12</sup><https://www.djangoproject.com/>

### 6.3.3. Importación de datos

La implementación de la importación de datos en el sistema fue problemática debido a la necesidad de soportar la entrada de grandes conjuntos de datos. Algunas fuentes de datos incluyen catálogos de datos de unos 40mb en formato de texto plano sin espacios ni saltos de líneas.

Debido a esta cantidad de información fue necesario tomar algunas medidas para hacer más efectivo el proceso de importación. A continuación se explicará que soluciones se tomaron al respecto.

- Uso de un nuevo módulo para el parseo de los ficheros entrantes. Python cuenta con dos implementaciones del parser XML de su librería estándar<sup>13</sup>: una de ellas en lenguaje Python y la otra en lenguaje C. Puesto que ambas implementaciones tienen la misma interfaz, pueden intercambiarse de forma transparente. Usar la implementación en lenguaje C del parser permitió aumentar la velocidad y reducir el consumo de memoria del proceso de importación de datos.
- Uso de generadores. El uso de listas hace necesario mantener grandes cantidades de objetos de forma simultánea en la memoria. La utilización de generadores permite que los objetos sean devueltos bajo demanda y, por tanto, los objetos no utilizados pueden ser eliminados por el recolector de basura con la consiguiente reducción en el uso de memoria.
- Inserciones parciales. Con grandes cantidades de datos las consultas de inserción en la base de datos producen que el SGBD<sup>14</sup> falle al no poder gestionar tanta información. La realización de varias inserciones parciales en las que sólo se inserta un determinado número de objetos, en lugar de una única gran inserción permitió eliminar este problema.

### 6.3.4. Adaptación al CMS

El desarrollo de los módulos de Drupal que se pueden ver en la sección “Vista de los módulos del gestor de contenidos” perteneciente al capítulo 5 requirió adaptar el desarrollo a las técnicas y mecanismos utilizados por el gestor de contenidos y, en muchas ocasiones, luchar contra el propio CMS para realizar algunas tareas.

La solución a este problema consistió en utilizar extensivamente la documentación oficial de Drupal<sup>15</sup> intentar comprender al máximo posible su funcionamiento para, de esta forma, intentar reducir al mínimo dichas situaciones.

---

<sup>13</sup>El parser recibe el nombre de *ElementTree*. Para más información véase la entrada en la documentación oficial de Python al respecto en <https://docs.python.org/2/library/xml.etree.elementtree.html>

<sup>14</sup>Como se detalló en la sección “Alternativas elegidas” perteneciente al capítulo 2, el Sistema de Gestión de Bases de Datos utilizado es MySQL.

<sup>15</sup>La documentación oficial de Drupal puede consultarse en <https://www.drupal.org/documentation>

# Capítulo 7

## Desarrollo de las pruebas

En este capítulo se detallarán las pruebas que se han realizado durante el desarrollo del proyecto así como los resultados de las mismas y su influencia.

### 7.1. Pruebas de integración

#### 7.1.1. Pruebas de la zona de datos

Como se ha explicado anteriormente en la sección “Especificación del plan de pruebas” perteneciente al capítulo 4, el subsistema de datos se ha desarrollado utilizando una metodología de Desarrollo Dirigido por Pruebas acompañada de un proceso de integración continua. A continuación se explicará la forma en la que se han aplicado las pruebas y los beneficios que han aportado al proceso de desarrollo.

##### 7.1.1.1. Desarrollo Dirigido por Pruebas

El Desarrollo Dirigido por Pruebas (o *Test Driven Development*) consiste en repetir los siguientes pasos continuamente durante el proceso de desarrollo del sistema:

1. Antes de añadir una nueva funcionalidad escribir una o varias pruebas para ella. Las pruebas fallarán hasta que la funcionalidad en cuestión no esté implementada.
2. Implementar la funcionalidad hasta que pase correctamente las pruebas diseñadas anteriormente.
3. Refactorizar el código cuando sea necesario. Puesto que las pruebas ya están escritas, se puede comprobar rápidamente si la refactorización ha introducido nuevos errores en el código.

Las ventajas que ha aportado el uso de esta metodología durante el desarrollo se detallarán a continuación:

- En muchas ocasiones las pruebas se realizan únicamente al final del desarrollo del software, lo que provoca que estas sean ineficientes e incompletas. La obligación de escribir las pruebas antes que la propia funcionalidad revierte esta situación y asegura la existencia de las pruebas (la corrección y completitud de las mismas queda siempre en manos del desarrollador).
- Pensar en las pruebas previamente a la implementación hace más fácil encontrar

posibles situaciones extrañas que puedan provocar problemas en el sistema.

- Dado que las pruebas actúan como los primeros clientes del código, existe una cierta obligación a hacer que la implementación real tenga una interfaz más definida y consistente.
- La existencia de las pruebas ayuda a aumentar la confianza durante las refactorizaciones de código. Esto ha sido especialmente útil dado que el lenguaje utilizado para el desarrollo del subsistema de datos ha sido Python, cuya dinamicidad aporta en muchas ocasiones menos seguridad que otros más estáticos.

#### 7.1.1.2. Integración continua

Con el objetivo de automatizar el proceso de pruebas y notificar al equipo de desarrollo de posibles fallos en los casos de prueba que hayan podido pasar desapercibidos, las pruebas descritas en el punto anterior se acompañan de la ayuda de un servidor de integración continua.

Una posible definición de integración continua, extraída de [Fowa] es la siguiente:

*“Continuous Integration is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily - leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible.”*

Por su utilidad y su capacidad de integración con GitHub (plataforma de control de versiones con la que se desarrolló este proyecto) se ha utilizado el servidor de integración continua Travis-CI<sup>1</sup>.

La siguiente cita de Martin Fowler [Fowa] puede ayudar al lector a comprender la funcionalidad de este tipo de servidores:

*“A continuous integration server acts as a monitor to the repository. Every time a commit against the repository finishes the server automatically checks out the sources onto the integration machine, initiates a build, and notifies the committer of the result of the build. The committer isn’t done until she gets the notification - usually an email.”*

Las principales ventajas que ha aportado el uso de un servidor de integración continua como Travis-CI al proceso de desarrollo han sido varias, entre las que destacan:

- El servidor de integración continua asegura que las pruebas se ejecutarán ante cualquier cambio del código que se produzca en el repositorio.
- En caso de que surga algún fallo durante la ejecución de las pruebas, el servidor de integración continua notifica al autor de los cambios que lo producen. Esta capacidad de notificación prácticamente inmediata ha ayudado a detectar y solucionar varios problemas de forma rápida antes de que se propaguen a otras partes del sistema.

#### 7.1.2. Pruebas de la zona social

A continuación se presentarán los diferentes casos de prueba realizados sobre los subsistemas pertenecientes a la zona zona social (subsistema de gestión de usuarios, noticias, debates, eventos, blog, organizaciones, comentarios y búsqueda). Cada caso de prueba cuenta con una descripción, el resultado esperado en la ejecución y el resultado obtenido.

Los fallos encontrados durante esta fase han sido corregidos antes de entregar la versión final

---

<sup>1</sup><https://travis-ci.org/>

del sistema.

#### 7.1.2.1. Subsistema de gestión de usuarios

La tabla 7.1 muestra los casos de prueba del subsistema de gestión de usuarios.

Cuadro 7.1: Casos de prueba del subsistema de gestión de usuarios

Casos de prueba del subsistema de gestión de usuarios			
Descripción	Proceso de prueba	Resultado esperado	Resultado obtenido
Registrar un nuevo usuario con un nombre y email no existentes en el sistema	Acceder al formulario de registro e introducir los como nombre de usuario “prueba” y como email “prueba@example.com”. El resto de campos pueden tener cualquier valor siempre que se llenen los campos obligatorios	La nueva cuenta de usuario se crea correctamente y con estado bloqueado	Resultado esperado
Registrar un nuevo usuario con un nombre ya existente en el sistema	Acceder al formulario de registro e introducir como nombre de usuario “prueba” y como email “prueba2@example.com”. El resto de campos pueden tener cualquier valor siempre que se llenen los campos obligatorios	El sistema no crea la cuenta de usuario y notifica de que el nombre no puede utilizarse	Resultado esperado
Registrar un nuevo usuario con un email ya existente en el sistema	Acceder al formulario de registro e introducir como nombre de usuario “prueba2” y como email “prueba@example.com”. El resto de campos pueden tener cualquier valor siempre que se llenen los campos obligatorios	El sistema no crea la cuenta de usuario y notifica de que el email no puede utilizarse	Resultado esperado
Registrar un nuevo usuario con un nombre inválido	Acceder al formulario de registro e introducir como nombre de usuario “prueba,3” y como email “prueba3@example.com”. El resto de campos pueden tener cualquier valor siempre que se llenen los campos obligatorios	El sistema no crea la cuenta de usuario y notifica de que el nombre contiene caracteres incorrectos	Resultado esperado

Continuación de la tabla 7.1

Descripción	Proceso de prueba	Resultado esperado	Resultado obtenido
Registrar un nuevo usuario con un email inválido	Acceder al formulario de registro e introducir como nombre de usuario “prueba3” y como email “correofalse”. El resto de campos pueden tener cualquier valor siempre que se rellenen los campos obligatorios	El sistema no crea la cuenta de usuario y notifica de que el email tiene un formato inválido	Resultado esperado
Registrar un nuevo usuario sin llenar todos los campos obligatorios	Acceder al formulario de registro e introducir como email “prueba3@example.com”. El resto de campos puede tener cualquier valor salvo el nombre, que se dejará en blanco	El sistema no crea la cuenta de usuario y notifica de que el nombre debe ser introducido	Resultado esperado
Iniciar sesión en el sistema con un usuario bloqueado	Acceder al formulario de login e intentar iniciar sesión con el usuario “prueba”	El sistema no permite el inicio de sesión puesto que el usuario está bloqueado	Resultado esperado.
Activar un usuario bloqueado	Acceder a la vista de administración y activar al usuario “prueba”, que se encuentra bloqueado	El sistema cambiará el estado del usuario y enviará un correo indicándole que establezca su contraseña para iniciar sesión en el sistema	Falla. Se produce un error al enviar el correo electrónico, aunque el estado del usuario se cambia a activo correctamente
Iniciar sesión en el sistema con un usuario activado	Acceder al formulario de login e introducir como nombre “prueba” y como contraseña la que haya establecido el usuario	El sistema permite iniciar sesión correctamente al usuario	Resultado esperado
Iniciar sesión en el sistema con campos vacíos	Acceder al formulario de login e introducir como nombre “prueba”, dejar el campo contraseña vacío e iniciar sesión	El sistema no permitirá iniciar sesión por haber dejado campos vacíos en el formulario	Resultado esperado

Continuación de la tabla 7.1

Descripción	Proceso de prueba	Resultado esperado	Resultado obtenido
Iniciar sesión en el sistema con una contraseña incorrecta	Acceder al formulario de login e introducir como nombre “prueba” y como contraseña un diferente a la escogida por el usuario	El sistema no permitirá iniciar sesión porque la contraseña es incorrecta	Resultado esperado
Un usuario anónimo accede al perfil de un usuario registrado	Sin haber iniciado sesión en el sistema acceder a la vista de perfil de un usuario registrado	El sistema no permite que los usuarios anónimos vean los perfiles de los usuarios registrados y mostrará la página de error	Resultado esperado
Un usuario registrado accede al perfil de otro usuario registrado	Habiendo iniciado sesión en el sistema acceder a la vista de perfil de un usuario registrado	El sistema mostrará los datos del perfil del usuario	Resultado esperado
Un usuario registrado accede al perfil de otro usuario con capacidad de acceso al API	Habiendo iniciado sesión en el sistema acceder a la vista de perfil de un usuario con capacidad de acceso al API	El sistema mostrará los datos del perfil del usuario excepto la clave de acceso al API	Resultado esperado
Un usuario registrado accede a su perfil	Habiendo iniciado sesión en el sistema acceder a la vista del propio perfil	El sistema mostrará los datos del usuario e incluirá un botón para editar la información de su perfil	Resultado esperado
Un usuario con capacidad de acceso al API accede a su perfil	Habiendo iniciado sesión en el sistema con una cuenta de usuario que tenga capacidad de acceso al API acceder a la vista del propio perfil	El sistema mostrará los datos del usuario incluyendo la clave de acceso al API y un botón para editar la información del perfil	Resultado esperado
Un usuario cierra sesión en el sistema	Habiendo iniciado sesión en el sistema pulsar el botón para cerrar sesión	El sistema cerrará la sesión del usuario y redirigirá a la página de inicio del portal	Resultado esperado
Dar permisos de acceso al API a un usuario registrado	Habiendo iniciado sesión como administrador acceder a la vista de usuarios del sistema y modificar el rol de un usuario registrado para darle permisos de acceso al API	Cuando el usuario modificado inicie sesión y acceda a la vista de su perfil podrá ver su clave de acceso al API junto con el resto de sus datos	Resultado esperado

Continuación de la tabla 7.1

Descripción	Proceso de prueba	Resultado esperado	Resultado obtenido
Dar permisos de administración a un usuario registrado	Habiendo iniciado sesión como administrador acceder a la vista de usuarios del sistema y modificar el rol de un usuario registrado para darle permisos de administración	Cuando el usuario modificado inicie sesión podrá ver la barra superior de administración y realizar las tareas reservadas para el rol de administrador del sistema	Resultado esperado

### 7.1.2.2. Subsistema de gestión de entradas del blog

La tabla 7.2 muestra los casos de prueba del subsistema de gestión de entradas del blog.

Cuadro 7.2: Casos de prueba del subsistema de gestión de entradas del blog

Casos de prueba del subsistema de gestión de entradas del blog			
Descripción	Proceso de prueba	Resultado esperado	Resultado obtenido
Crear una nueva entrada en el blog como administrador	Habiendo iniciado sesión en el portal como administrador acceder a la vista del blog y pulsar el botón para crear una nueva entrada. Rellenar el formulario de creación de una entrada sin dejar ningún campo obligatorio en blanco y guardar los cambios	La nueva entrada aparece como la más reciente del blog y los usuarios pueden acceder a ella	Resultado esperado
Crear una nueva entrada en el blog sin ser administrador	Habiendo iniciado sesión en el portal como un usuario sin privilegios de administración acceder a la vista del blog e intentar crear una nueva entrada	El botón para crear una nueva entrada no se mostrará y el usuario no podrá crear la entrada	Resultado esperado
Ver una entrada del blog como usuario anónimo	Como usuario anónimo acceder a la vista del blog y pulsar sobre una entrada cualquiera	El sistema mostrará la entrada en detalle y los comentarios si existen	Resultado esperado
Ver una entrada del blog como usuario registrado	Habiendo iniciado sesión en el portal como un usuario sin privilegios de administración acceder a la vista del blog y pulsar sobre una entrada cualquiera	El sistema mostrará la entrada en detalle a los comentarios también permitirá al usuario crear nuevos comentarios	Resultado esperado
Ver una entrada del blog como administrador	Habiendo iniciado sesión en el portal como administrador acceder a la vista del blog y pulsar sobre una entrada cualquiera	El sistema mostrará la entrada en detalle y los comentarios, permitirá agregar nuevos comentarios y también mostrará los botones para editar y eliminar la entrada	Resultado esperado

Continuación de la tabla 7.2

Descripción	Proceso de prueba	Resultado esperado	Resultado obtenido
Editar una entrada del blog como administrador	Habiendo iniciado sesión en el portal como administrador acceder a la vista del blog y pulsar sobre una entrada cualquiera. Una vez en la vista de detalle de la entrada pulsar sobre el botón correspondiente para editar su contenido. Rellenar el formulario de modificación de la entrada sin dejar ningún campo obligatorio en blanco y guardar los cambios	El sistema actualiza la entrada con los nuevos datos	Resultado esperado
Editar una entrada del blog como administrador	Habiendo iniciado sesión en el portal como administrador acceder a la vista del blog y pulsar sobre una entrada cualquiera. Una vez en la vista de detalle de la entrada pulsar sobre el botón correspondiente para editar su contenido. Rellenar el formulario de modificación de la entrada dejando algún campo obligatorio en blanco y guardar los cambios	El sistema no modifica la entrada y avisa al usuario de que algún campo requerido se ha dejado en blanco	Resultado esperado
Editar una entrada del blog sin ser administrador	Habiendo iniciado sesión en el portal como un usuario sin privilegios de administración acceder a la vista del blog y pulsar sobre una entrada cualquiera. Una vez en la vista de la entrada intentar modificar su contenido	El sistema no muestra el botón de modificación de la entrada	Resultado esperado

Continuación de la tabla 7.2

Descripción	Proceso de prueba	Resultado esperado	Resultado obtenido
Eliminar una entrada del blog como administrador	Habiendo iniciado sesión en el portal como administrador acceder a la vista del blog y pulsar sobre una entrada cualquiera. Una vez en la vista de la entrada pulsar sobre el botón correspondiente para su eliminación	El sistema elimina la entrada del blog	Resultado esperado
Eliminar una entrada del blog sin ser administrador	Habiendo iniciado sesión en el portal como un usuario sin privilegios de administración acceder a la vista del blog y pulsar sobre una entrada cualquiera. Una vez en la vista de la entrada intentar eliminarla	El sistema no muestra el botón de eliminación de la entrada	Resultado esperado

### 7.1.2.3. Subsistema de gestión de debates

La tabla 7.3 muestra los casos de prueba del subsistema de gestión de debates.

Cuadro 7.3: Casos de prueba del subsistema de gestión de debates

Casos de prueba del subsistema de gestión de debates			
Descripción	Proceso de prueba	Resultado esperado	Resultado obtenido
Crear un nuevo debate como usuario anónimo	Sin iniciar sesión en el portal acceder a la vista de debates e intentar crear un nuevo debate	El sistema no mostrará el botón de creación de un nuevo debate	Resultado esperado
Crear un nuevo debate como usuario registrado	Habiendo iniciado sesión en el portal acceder a la vista de debates y pulsar en el botón correspondiente para crear un nuevo debate. En el formulario llenar todos los campos requeridos y pulsar el botón de guardar	El sistema crea el debate con los comentarios cerrados y estando “próximamente”	Resultado esperado
Crear un nuevo debate como usuario registrado con datos insuficientes	Habiendo iniciado sesión en el portal acceder a la vista de debates y pulsar en el botón correspondiente para crear un nuevo debate. En el formulario dejar algún campo obligatorio en blanco y pulsar el botón de guardar	El sistema no crea el debate y notifica al usuario de que algunos campos se han dejado en blanco	Resultado esperado
Acceder a un debate como usuario registrado	Habiendo iniciado sesión en el portal como un usuario sin privilegios de administración acceder a la vista de debates y pulsar sobre un debate creado por otro usuario	El sistema muestra la vista de detalle del debate pero no incluye el botón de eliminar ni modificar el debate	Resultado esperado

Continuación de la tabla 7.3

Descripción	Proceso de prueba	Resultado esperado	Resultado obtenido
Acceder a un debate como autor	Habiendo iniciado sesión en el portal como un usuario sin privilegios de administración acceder a la vista de debates y pulsar sobre un debate creado por el propio usuario	El sistema muestra la vista de detalle del debate incluyendo un botón para eliminar el debate	Resultado esperado
Acceder a un debate como administrador	Habiendo iniciado sesión en el portal como administrador acceder a la vista de debates y pulsar sobre un debate creado por otro usuario	El sistema muestra la vista de detalle del debate incluyendo un botón para eliminar el debate y otro para modificar su contenido	Resultado esperado
Editar un debate	Habiendo iniciado sesión en el portal como administrador acceder a la vista de debates y pulsar sobre un debate. Una vez en la vista del debate pulsar sobre el botón correspondiente para editar su contenido. En el formulario llenar todos los campos obligatorios y guardar los cambios	El sistema modifica la información del debate	Resultado esperado
Editar un debate con datos insuficientes	Habiendo iniciado sesión en el portal como administrador acceder a la vista de debates y pulsar sobre un debate. Una vez en la vista del debate pulsar sobre el botón correspondiente para editar su contenido. En el formulario dejar en blanco algún campo obligatorio y guardar los cambios	El sistema no modifica la información del debate y notifica al usuario de que algunos campos se han dejado en blanco	Resultado esperado

Continuación de la tabla 7.3

Descripción	Proceso de prueba	Resultado esperado	Resultado obtenido
Abrir un debate	Habiendo iniciado sesión como administrador acceder a la vista de debates y pulsar sobre un debate con estado “próximamente”. Una vez en la vista del debate pulsar en el botón correspondiente para su modificación, cambiar su estado a “abierto” y abrir los comentarios	El sistema cambia el estado del debate y permite que los usuarios creen nuevos comentarios	Resultado esperado
Cerrar un debate	Habiendo iniciado sesión como administrador acceder a la vista de debates y pulsar sobre un debate con estado “abierto”. Una vez en la vista del debate pulsar en el botón correspondiente para su modificación, cambiar su estado a “cerrado” y cerrar los comentarios	El sistema cambia el estado del debate y no permite que los usuarios creen nuevos comentarios	Resultado esperado

#### 7.1.2.4. Subsistema de gestión de eventos

La tabla 7.4 muestra los casos de prueba del subsistema de gestión de eventos.

Cuadro 7.4: Casos de prueba del subsistema de gestión de eventos

Casos de prueba del subsistema de gestión de eventos			
Descripción	Proceso de prueba	Resultado esperado	Resultado obtenido
Acceder a un evento como usuario anónimo	Sin iniciar sesión en el portal acceder a la vista de eventos y pulsar sobre un evento cualquiera	El sistema muestra la información del evento	Resultado esperado
Acceder a un evento como autor	Habiendo iniciado sesión como un usuario sin privilegios de administración acceder a la vista de eventos y pulsar sobre un evento creado por el propio usuario	El sistema muestra la información del evento junto con un botón para editar su contenido	Resultado esperado
Acceder a un evento como administrador	Habiendo iniciado sesión como administrador acceder a la vista de eventos y pulsar sobre un evento cualquiera	El sistema muestra la información del evento junto con un botón para modificar su contenido y otro para eliminar el evento	Resultado esperado
Crear un evento	Habiendo iniciado sesión como un usuario sin privilegios de administración acceder a la vista de eventos y pulsar sobre el botón para crear un nuevo evento. En el formulario llenar todos los campos obligatorios y guardar los cambios	El sistema crea el evento y lo hace visible para el resto de usuarios	Resultado esperado

Continuación de la tabla 7.4

Descripción	Proceso de prueba	Resultado esperado	Resultado obtenido
Crear un evento con datos insuficientes	Habiendo iniciado sesión como un usuario sin privilegios de administración acceder a la vista de eventos y pulsar sobre el botón para crear un nuevo evento. En el formulario dejar algún campo obligatorio en blanco y guardar los cambios	El sistema no crea el evento y notifica al usuario de que algún campo obligatorio está vacío	Resultado esperado
Modificar un evento	Habiendo iniciado sesión como un usuario sin privilegios de administración acceder a la vista de eventos y pulsar sobre un evento creado por el propio usuario. Una vez en el evento pulsar sobre el botón para editar su información. En el formulario llenar todos los campos obligatorios y guardar los cambios	El sistema modifica la información del evento	Resultado esperado
Modificar un evento con datos insuficientes	Habiendo iniciado sesión como administrador acceder a la vista de eventos y entrar en un evento cualquiera. En el evento pulsar sobre el botón para editar su información. En el formulario dejar algún campo obligatorio en blanco y guardar los cambios	El sistema no modifica la información del evento y notifica al usuario de que algún campo se ha dejado en blanco	Resultado esperado
Eliminar un evento	Habiendo iniciado sesión como administrador acceder a la vista de eventos y entrar en un evento cualquiera, una vez en el evento pulsar sobre el botón para eliminarlo	El sistema elimina toda la información del evento	Resultado esperado

### 7.1.2.5. Subsistema de gestión de noticias

La tabla 7.5 muestra los casos de prueba del subsistema de gestión de noticias.

Cuadro 7.5: Casos de prueba del subsistema de gestión de noticias

Casos de prueba del subsistema de gestión de noticias			
Descripción	Proceso de prueba	Resultado esperado	Resultado obtenido
Acceder a una noticia como usuario anónimo	Sin iniciar sesión en el portal acceder a la vista de noticias y pulsar sobre una noticia cualquiera	El sistema muestra la información de la noticia	Resultado esperado
Acceder a una noticia como autor	Habiendo iniciado sesión como un usuario sin privilegios de administración acceder a la vista de noticias y pulsar sobre una noticia creada por el propio usuario	El sistema muestra la información de la noticia junto con un botón para editar su contenido	Resultado esperado
Acceder a una noticia como administrador	Habiendo iniciado sesión como administrador acceder a la vista de noticias y pulsar sobre una noticia cualquiera	El sistema muestra la información de la noticia junto con un botón para modificar su contenido y otro para eliminarla	Resultado esperado
Crear una noticia	Habiendo iniciado sesión como un usuario sin privilegios de administración acceder a la vista de noticias y pulsar sobre el botón para crear una nueva noticia. En el formulario llenar todos los campos obligatorios y guardar los cambios	El sistema crea la noticia y la hace visible para el resto de usuarios	Resultado esperado

Continuación de la tabla 7.5

Descripción	Proceso de prueba	Resultado esperado	Resultado obtenido
Crear una noticia con datos insuficientes	Habiendo iniciado sesión como un usuario sin privilegios de administración acceder a la vista de noticias y pulsar sobre el botón para crear una nueva noticia. En el formulario dejar algún campo obligatorio en blanco y guardar los cambios	El sistema no crea la noticia y notifica al usuario de que algún campo obligatorio está vacío	Resultado esperado
Modificar una noticia	Habiendo iniciado sesión como un usuario sin privilegios de administración acceder a la vista de noticias y pulsar sobre una noticia creada por el propio usuario. Una vez en la noticia pulsar sobre el botón para editar su información. En el formulario llenar todos los campos obligatorios y guardar los cambios	El sistema modifica la información de la noticia	Resultado esperado
Modificar una noticia con datos insuficientes	Habiendo iniciado sesión como administrador acceder a la vista de noticias y entrar en una noticia cualquiera. En la noticia pulsar sobre el botón para editar su información. En el formulario dejar algún campo obligatorio en blanco y guardar los cambios	El sistema no modifica la información de la noticia y notifica al usuario de que algún campo se ha dejado en blanco	Resultado esperado
Eliminar una noticia	Habiendo iniciado sesión como administrador acceder a la vista de noticias y entrar en una noticia cualquiera, una vez en la noticia pulsar sobre el botón para eliminarla	El sistema elimina toda la información de la noticia	Resultado esperado

### 7.1.2.6. Subsistema de gestión de organizaciones

La tabla 7.6 muestra los casos de prueba del subsistema de gestión de organizaciones.

Cuadro 7.6: Casos de prueba del subsistema de gestión de organizaciones

Casos de prueba del subsistema de gestión de organizaciones			
Descripción	Proceso de prueba	Resultado esperado	Resultado obtenido
Acceder a una organización como usuario anónimo	Sin iniciar sesión en el portal acceder a la vista de organizaciones y pulsar sobre una organización cualquiera	El sistema muestra la información de la organización	Resultado esperado
Acceder a una organización como usuario registrado	Habiendo iniciado sesión como un usuario sin privilegios de administración acceder a la vista de organización y pulsar sobre una organización cualquiera	El sistema muestra la información de la organización	Resultado esperado
Acceder a una organización como administrador	Habiendo iniciado sesión como administrador acceder a la vista de organizaciones y pulsar sobre una organización cualquiera	El sistema muestra la información de la organización junto con un botón para modificar su contenido y otro para eliminarla	Resultado esperado
Crear una organización	Habiendo iniciado sesión como administrador acceder a la vista de organizaciones y pulsar sobre el botón para crear una nueva organización. En el formulario llenar todos los campos obligatorios y guardar los cambios	El sistema crea la organización y la hace visible para el resto de usuarios	Resultado esperado
Crear una noticia con datos insuficientes	Habiendo iniciado sesión como administrador acceder a la vista de organizaciones y pulsar sobre el botón para crear una nueva organización. En el formulario dejar algún campo obligatorio en blanco y guardar los cambios	El sistema no crea la organización y notifica al usuario de que algún campo obligatorio está vacío	Resultado esperado

Continuación de la tabla 7.6

Descripción	Proceso de prueba	Resultado esperado	Resultado obtenido
Modificar una organización	Habiendo iniciado sesión como administrador acceder a la vista de organizaciones y pulsar sobre una organización cualquiera. Una vez en la organización pulsar sobre el botón para editar su información. En el formulario llenar todos los campos obligatorios y guardar los cambios	El sistema modifica la información de la organización	Resultado esperado
Modificar una organización con datos insuficientes	Habiendo iniciado sesión como administrador acceder a la vista de organizaciones y entrar en una organización cualquiera. En la organización pulsar sobre el botón para editar su información. En el formulario dejar algún campo obligatorio en blanco y guardar los cambios	El sistema no modifica la información de la organización y notifica al usuario de que algún campo se ha dejado en blanco	Resultado esperado
Eliminar una organización	Habiendo iniciado sesión como administrador acceder a la vista de organizaciones y entrar en una organización cualquiera, una vez en la organización pulsar sobre el botón para eliminarla	El sistema elimina toda la información de la organización	Resultado esperado

### 7.1.2.7. Subsistema de gestión de comentarios

La tabla 7.7 muestra los casos de prueba del subsistema de gestión de comentarios.

Cuadro 7.7: Casos de prueba del subsistema de gestión de comentarios

Casos de prueba del subsistema de gestión de comentarios			
Descripción	Proceso de prueba	Resultado esperado	Resultado obtenido
Comentar en una entrada del blog	Habiendo iniciado sesión como un usuario sin privilegios de administración acceder al blog y pulsar sobre una entrada cualquiera. Una vez en la entrada introducir un nuevo comentario y guardar los cambios	El sistema guarda el comentario como parte de la entrada y lo muestra a todos los usuarios	Resultado esperado
Comentar en un debate cerrado	Habiendo iniciado sesión como un usuario sin privilegios de administración acceder a los debates y pulsar sobre un debate con estado “cerrado”	El sistema muestra los comentarios existentes y no permite que el usuario introduzca un nuevo comentario	Resultado esperado
Comentar un debate abierto	Habiendo iniciado sesión como un usuario sin privilegios de administración acceder a los debates y pulsar sobre un debate con estado “abierto”	El sistema muestra los comentarios existentes y permite que el usuario introduzca un nuevo comentario	Resultado esperado
Modificar un comentario	Habiendo iniciado sesión como administrador acceder al blog y pulsar sobre una entrada con comentarios. En uno de los comentarios pulsar el botón correspondiente para editarla e introducir el nuevo mensaje	El sistema actualiza el comentario con los cambios introducidos por el administrador	Resultado esperado

Continuación de la tabla 7.7

Descripción	Proceso de prueba	Resultado esperado	Resultado obtenido
Eliminar un comentario	Habiendo iniciado sesión como administrador acceder al blog y pulsar sobre una entrada con comentarios. En uno de los comentarios pulsar el botón correspondiente para eliminarlo	El sistema elimina el comentario y deja de mostrarlo a los usuarios	Resultado esperado

#### 7.1.2.8. Subsistema de búsqueda

La tabla 7.8 muestra los casos de prueba del subsistema de búsqueda.

Cuadro 7.8: Casos de prueba del subsistema de búsqueda

Casos de prueba del subsistema de gestión de búsqueda			
Descripción	Proceso de prueba	Resultado esperado	Resultado obtenido
Crear índice de contenidos	Habiendo iniciado sesión como administrador acceder a las opciones de búsqueda en panel de administración. Pulsar el botón destinado a la actualización del índice de contenidos	El sistema envía los datos al motor de búsqueda para su indexación	Resultado esperado
Realizar búsqueda	Habiendo iniciado sesión como un usuario sin privilegios de administración acceder a la vista de búsqueda. En el campo de búsqueda introducir el término “España” y pulsar el botón para buscar	El sistema mostrará al menos un resultado correspondiente al país España junto a su bandera. Al pulsar en el resultado el sistema cargará la vista de la sección de datos para el país España	Resultado esperado

## 7.2. Pruebas de rendimiento

Anteriormente, en la sección “Especificación del plan de pruebas” perteneciente al capítulo 4 se explicó que tendrían lugar una serie de pruebas de rendimiento sobre el Punto de Entrada de Datos del sistema. En esta sección se detallará la metodología seguida para la realización de dichas pruebas así como los resultados obtenidos sobre las mismas.

### 7.2.1. Proceso y ámbito de las mediciones

En la sección “Definición del sistema”, también del capítulo 4 se indicó que sólo forma parte del alcance de este proyecto la creación de un servicio de generación de SQL como parte del Punto de Entrada de Datos. Debido a esto y con el fin de aislar las mediciones a un ámbito relevante para esta documentación, estas han sido realizadas con los servicios de generación de RDF y CKAN desactivados.

Las mediciones tendrán lugar mediante el envío de un fichero XML al Punto de Entrada de Datos para su procesado e inserción en base de datos usando el parser común y el servicio de generación de SQL. Para hacer las mediciones lo más exactas posible se repetirán un total de tres veces y posteriormente se calculará la mediana. Con el fin de evitar interferencias externas, cada medición se realizará sobre una base de datos MySQL<sup>2</sup> vacía y un servicio Apache<sup>3</sup> recién iniciado.

Las herramientas utilizadas para medir realizar las mediciones son el módulo *memory profiler*<sup>4</sup> para la medición del rendimiento espacial, y el comando *time*<sup>5</sup> para la medición del rendimiento temporal. Las mediciones de espacio y tiempo se realizarán en ejecuciones diferentes para que no interfieran entre sí.

### 7.2.2. Resultado de las mediciones

A continuación se expondrán los resultados obtenidos de las mediciones.

Las figuras 7.1, 7.2 y 7.3 muestran respectivamente el número de observaciones del catálogo de datos entrante frente al tiempo de procesado del mismo, el número de observaciones procesadas por segundo y el consumo de memoria del Punto de Entrada de Datos durante el procesado. Los datos utilizados para la creación de dichas figuras pueden consultarse con mayor detalle en el anexo 12.2.

### 7.2.3. Análisis de los resultados

A continuación se expondrán las conclusiones extraídas de los resultados obtenidos durante las mediciones. Los resultados han sido expuestos en la tabla 12.1 y las figuras 7.1, 7.2 y 7.3.

- El número de observaciones del catálogo de datos entrante es el valor dominante en el consumo de tiempo y memoria durante el procesado de los datos. Conforme aumenta el número de observaciones presentes en el catálogo de datos, el resto de factores (indicadores, usuario, etc) son despreciables.
- El número de observaciones entrantes es directamente proporcional tanto al tiempo como a la memoria utilizados durante el procesado de datos.

<sup>2</sup>La versión de MySQL utilizada durante las mediciones es la 5.5.37

<sup>3</sup>La versión de Apache utilizada durante las mediciones es la 2.4.7

<sup>4</sup>El módulo *memory profiler* puede descargarse en [https://pypi.python.org/pypi/memory\\_profiler](https://pypi.python.org/pypi/memory_profiler)

<sup>5</sup>El comando *time* forma parte de los sistemas tipo UNIX y permite medir la duración de la ejecución de un determinado programa

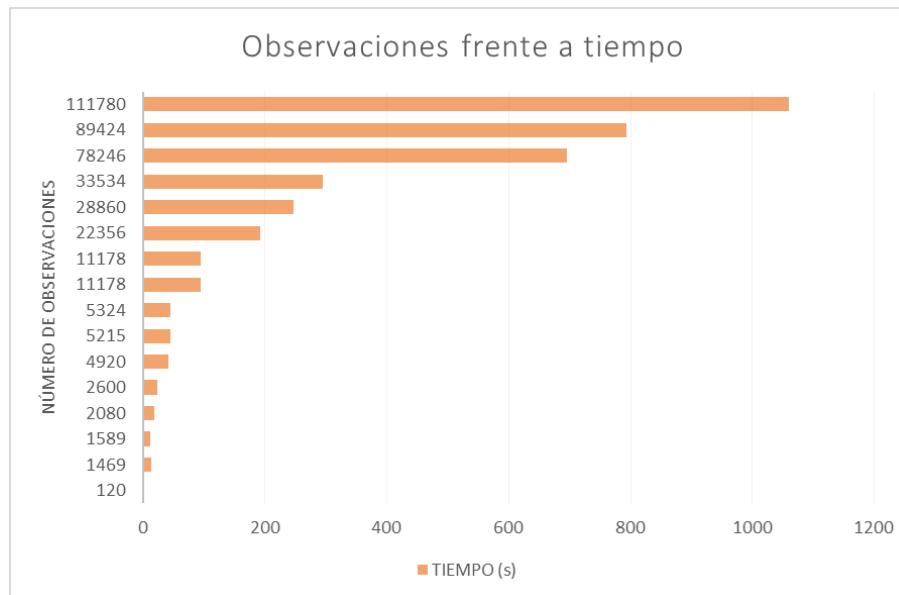


Figura 7.1: Número de observaciones entrantes frente a tiempo de procesado

- La velocidad de procesado de datos tiende a estabilizarse en torno a un valor concreto. En el caso de la máquina en la que se realizaron las mediciones este valor fue de 105 observaciones por segundo.
- El *profiling* de memoria hace notablemente más lento el procesado de datos, por lo que la decisión de medir por separado el consumo de memoria y de tiempo ha sido positiva para evitar interferencias entre ambas mediciones.
- La complejidad del modelo de datos, así como el uso de un Mapeador Objeto-Relacional (ORM) para gestionar la persistencia obligan a crear multitud de objetos relacionados entre sí durante el parseo y la persistencia del catálogo de datos. Esto provoca el alto consumo de memoria mostrado en las mediciones, que en varias ocasiones llega a multiplicar por un factor de 70 el tamaño del catálogo de datos entrante.

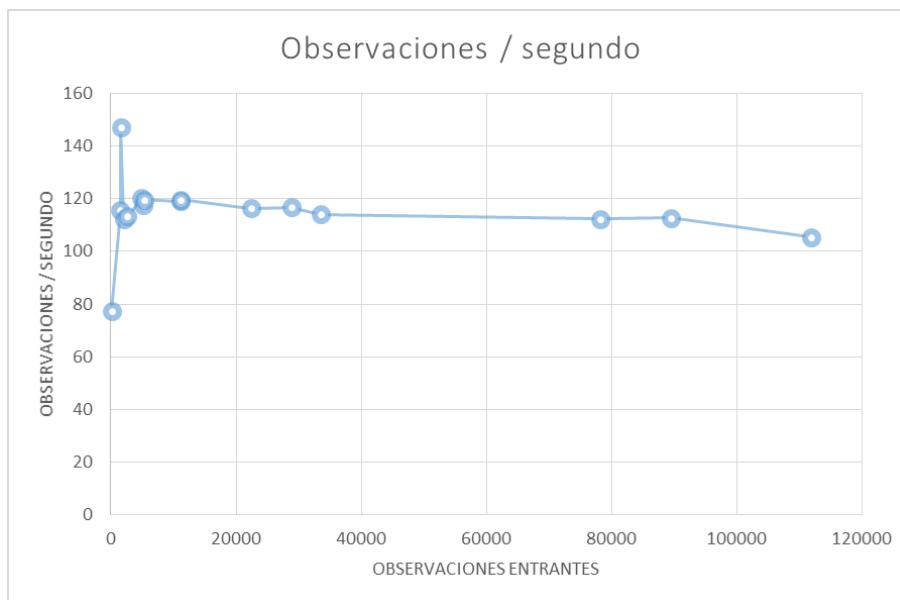


Figura 7.2: Número de observaciones por segundo procesadas

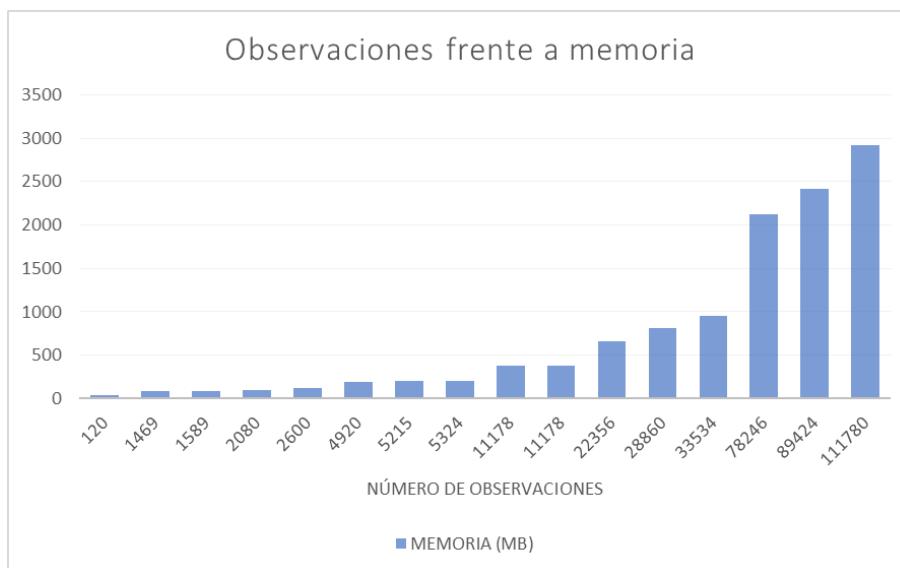


Figura 7.3: Número de observaciones frente a memoria durante el procesado

### 7.3. Pruebas de aceptación

A continuación se muestran los reportes obtenidos del cliente durante las pruebas de aceptación<sup>6</sup>. La categorización filtrado de los reportes ha sido realizada por el jefe de proyecto.

Todos los reportes que se listan a continuación han podido reproducirse correctamente y se encuentran arreglados en la versión final del sistema.

- En los debates y las entradas del blog sería conveniente mostrar el nombre real del usuario en lugar de su *nickname*.
- Cuando un registro se completa adecuadamente el sistema no muestra ningún mensaje al usuario tras hacer la redirección a la página de inicio.
- En el menú de la sección social se debe modificar la entrada de los debates por “Discussions — Diálogos — Discussions”.
- Cuando un usuario solicita una nueva contraseña el sistema debería mostrar un mensaje indicando que la contraseña se ha restaurado correctamente antes de hacer la redirección a la página de inicio.
- El nombre de los usuarios que se muestra en las entradas de la sección social debería estar enlazado al perfil del usuario.
- Cuando se pulsa en una etiqueta de la zona social en ocasiones se muestra una vista errónea y similar a la de un nodo.
- Los enlaces para compartir contenidos en redes sociales como Twitter o Facebook contienen muchos espacios en el contenido.
- En los debates y las entradas del blog se muestra el texto “*What the user say*”. Este texto es incorrecto y debería modificarse por “*What the users say*”.
- En la vista de eventos el nombre del calendario se muestra incorrectamente y en ocasiones se tapa por los botones de navegación.
- Las etiquetas “*Coming soon*”, “*Open*” y “*Closed*” de los debates no se muestran internacionalizadas.
- No existe ningún botón para que un usuario cierre sesión en el portal.
- Los *combobox* de países muestran los países desordenados y con algunos espacios en blanco. Han de mostrarse ordenados alfabéticamente según su nombre.
- Los *combobox* de indicadores muestran los indicadores desordenados. Han de mostrarse ordenados alfabéticamente según su nombre.
- Los indicadores que se muestran en la vista de detalles de un país deberían incluir la fuente de datos de la que provienen.
- El modelo de país no devuelve los indicadores del gráfico *spider* si no tienen observaciones. Esto provoca que en el gráfico las etiquetas se muestren vacías.
- En la zona social los debates deberían figurar como primera pestaña para darles una mayor visibilidad, puesto que son el punto central de participación de los usuarios.
- Las etiquetas que contienen el estado de los debates deberían estar coloreadas de forma diferente según su estado. Ésto permite a los usuarios distinguir más fácilmente el estado en

<sup>6</sup>Únicamente se incluyen los reportes que afectan a las partes desarrolladas en este TFG. La lista con todos los reportes obtenidos puede consultarse en <http://goo.gl/GhBDRw>

que se encuentra un debate.

- En el perfil de usuario se debería incluir una sección en la que se muestren todos los contenidos creados por el usuario.
- Los usuarios deberían incluir un campo en el que especificar sus intereses.
- En los formularios de la zona social se debería incluir una pequeña ayuda sobre las etiquetas HTML válidas, el comportamiento de los enlaces, etc.

## Capítulo 8

# Planificación del proyecto

En este capítulo se detallarán los trabajos que son necesarios para desarrollar el proyecto y la planificación de los mismos a lo largo del tiempo.

### 8.1. WBS

Con el objetivo de conocer todas las tareas necesarias para desarrollar el proyecto para posteriormente estimar su duración y planificarlas correctamente es necesario realizar una Estructura de Desglose del Trabajo (o WBS<sup>1</sup>).

A continuación se mostrarán los diversos diagramas de la Estructura de Desglose del Trabajo. Por su tamaño los diagramas se han subdividido para facilitar su lectura. El orden en el que se relacionan los diagramas es el siguiente:

1. La figura 8.1 muestra el diagrama de la fase de análisis del sistema, que produce como resultado la especificación formal del sistema a desarrollar.
2. Posteriormente, la figura 8.2 muestra el diagrama de la fase del diseño de sistema, que produce como resultado el diseño de la arquitectura del sistema, de sus componentes y el modelo de datos final.
3. Una vez obtenido el diseño, la figura 8.3 muestra el diagrama de las tareas pertenecientes al desarrollo del gestor de contenidos y sus módulos. Por su tamaño, las tareas pertenecientes al desarrollo de la zona social se han dividido en un diagrama diferente (figura 8.4).
4. En paralelo al desarrollo del gestor de contenidos y sus módulos podría tener lugar el desarrollo del Punto de Entrada de Datos al portal, cuyo diagrama de tareas se muestra en la figura 8.5.
5. Una vez desarrollados los componentes del sistema tendrían lugar las fases de integración y pruebas de aceptación. El diagrama de tareas de esta fase se muestra en la figura 8.6.

---

<sup>1</sup>El concepto de *Work Breakdown Structure* fue introducido durante el desarrollo de la Técnica de Revisión y Evaluación de Programas (PERT) por el Departamento de Defensa de los Estados Unidos. El WBS permite conocer todas las tareas necesarias para desarrollar un proyecto. Para más información se recomienda consultar el capítulo 5 del PMBOK [PMI09]

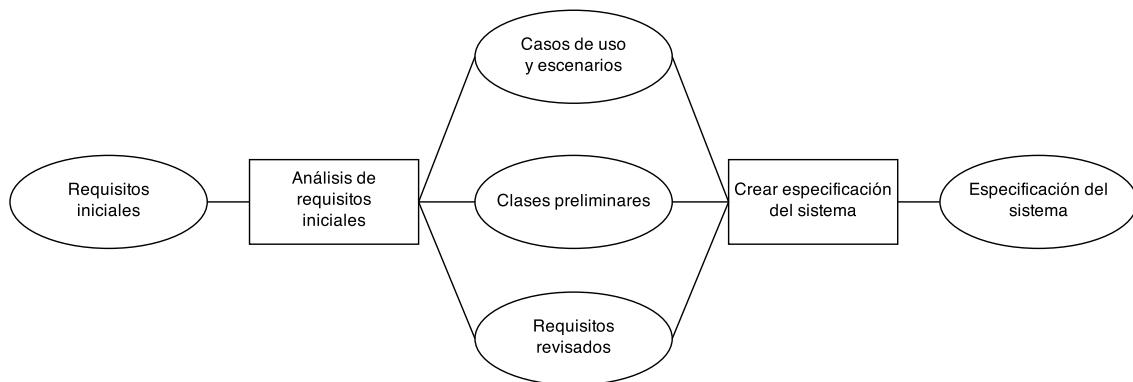


Figura 8.1: Estructura de desglose del trabajo de la fase de análisis del sistema

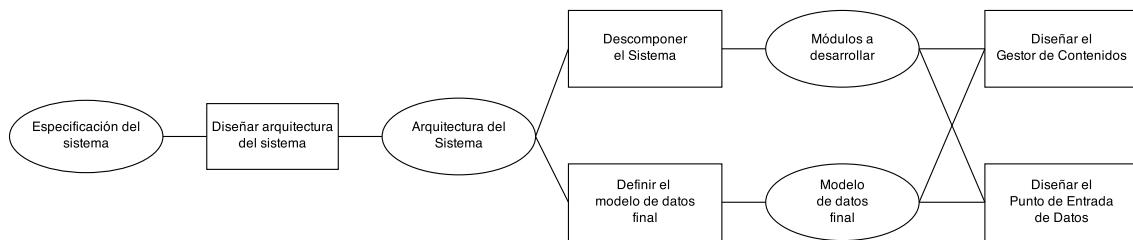


Figura 8.2: Estructura de desglose del trabajo de la fase de diseño del sistema

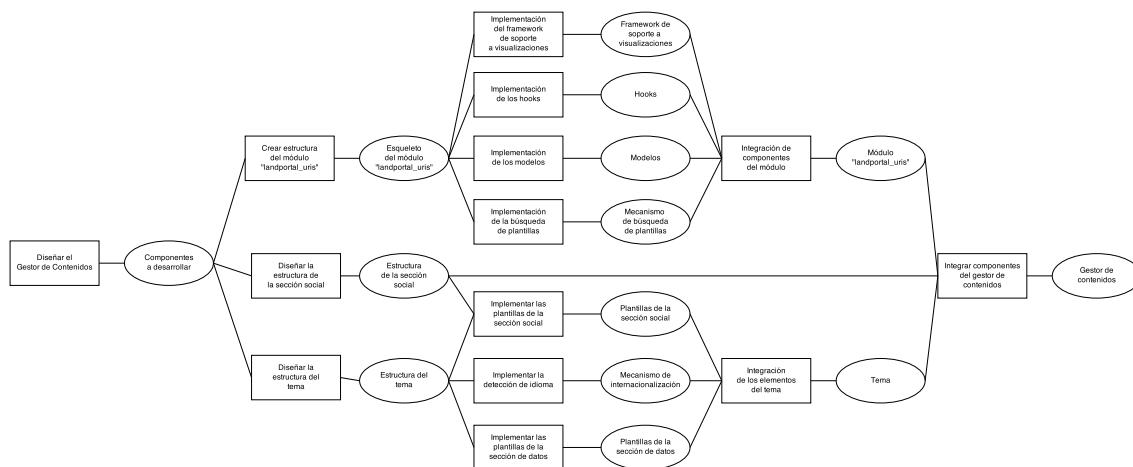


Figura 8.3: Estructura de desglose del trabajo del gestor de contenidos

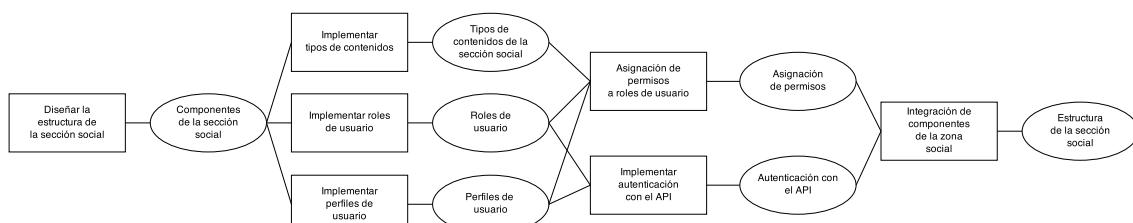


Figura 8.4: Estructura de desglose del trabajo de la fase desarrollo de la zona social

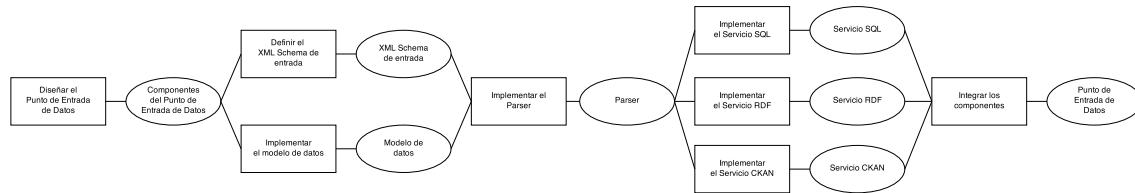


Figura 8.5: Estructura de desglose del trabajo de la fase desarrollo del Punto de Entrada de Datos

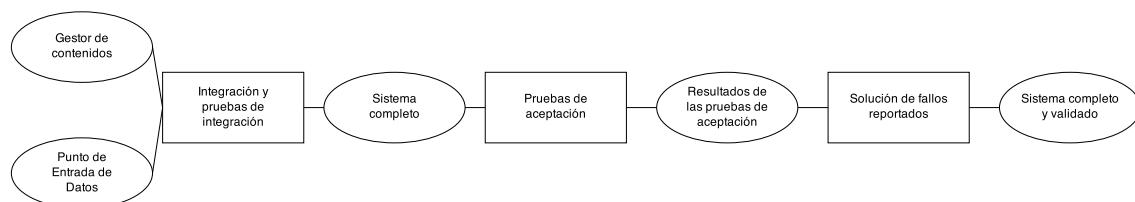


Figura 8.6: Estructura de desglose del trabajo de la fase final del sistema

## 8.2. Planificación inicial

La figura 8.7 muestra el diagrama de Gantt con la planificación inicial del proyecto. La planificación inicial se ha realizado basándose en la Estructura de Desglose del Trabajo detallada en el apartado anterior. A la Estructura de Desglose de Trabajo se han añadido las tareas de gestión del proyecto y los hitos necesarios para ir desarrollando el sistema.

La planificación inicial contaba con unas 820 horas de trabajo.

A pesar de no figurar en el diagrama Gantt, cabe destacar que una vez por semana se realiza una reunión de seguimiento del proyecto por parte del equipo de desarrollo. El objetivo de estas reuniones es informar a todo el equipo del avance del proyecto y tratar los posibles problemas y cambios que vayan surgiendo durante el desarrollo.

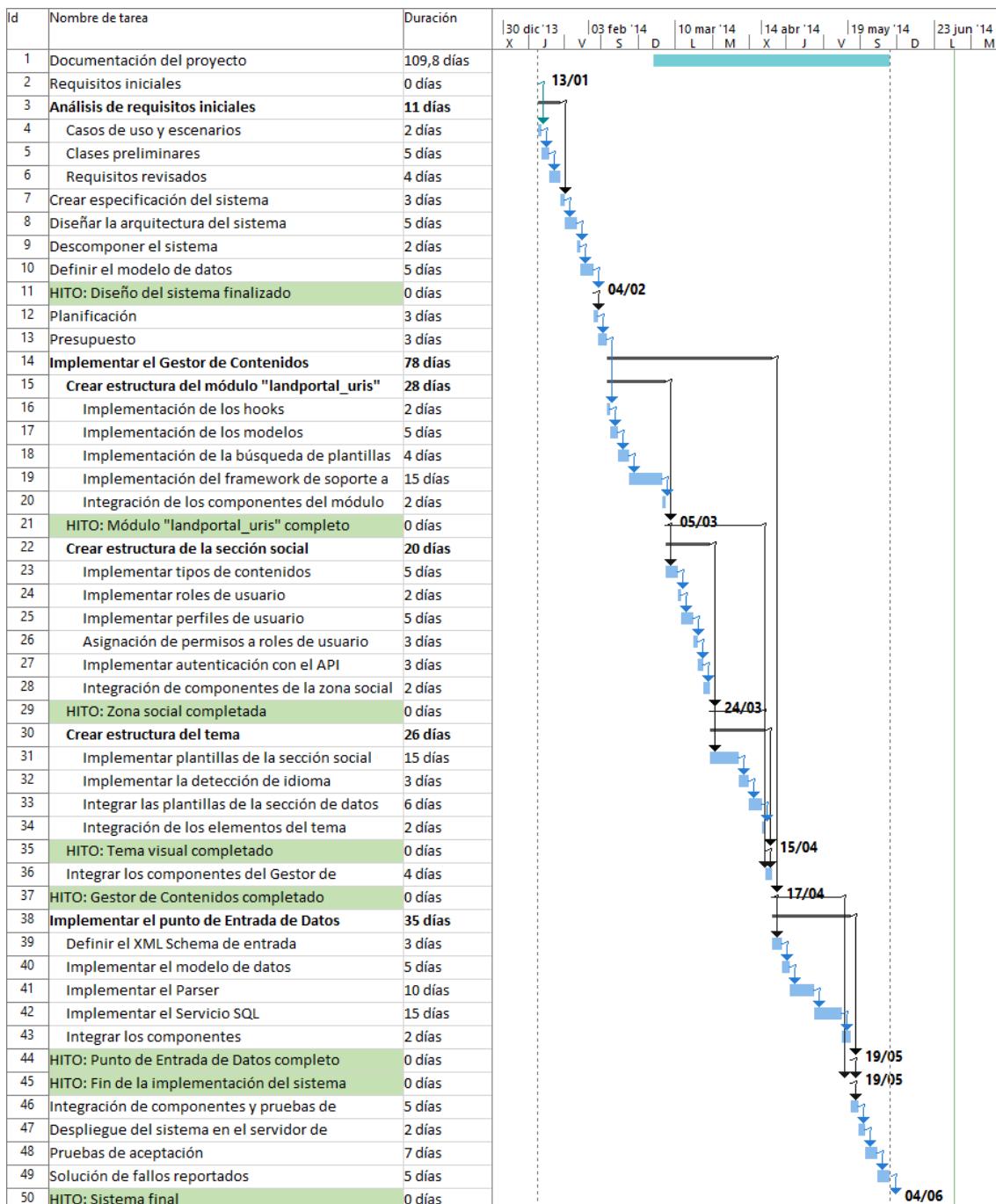


Figura 8.7: Diagrama de Gantt con la planificación inicial del proyecto

### 8.3. Planificación real

La figura 8.8 muestra el diagrama de Gantt con la planificación real del proyecto. La planificación real responde al ritmo real con el que se ha desarrollado el proyecto. Al igual que sucedió con la planificación inicial, cabe destacar que una vez por semana se ha realizado una reunión de seguimiento por parte del equipo de desarrollo.

La planificación real ha resultado en unas 970 horas de trabajo.

Las diferencias entre la planificación inicial y real son varias y se explicarán con detalle en la siguiente sección de este capítulo.

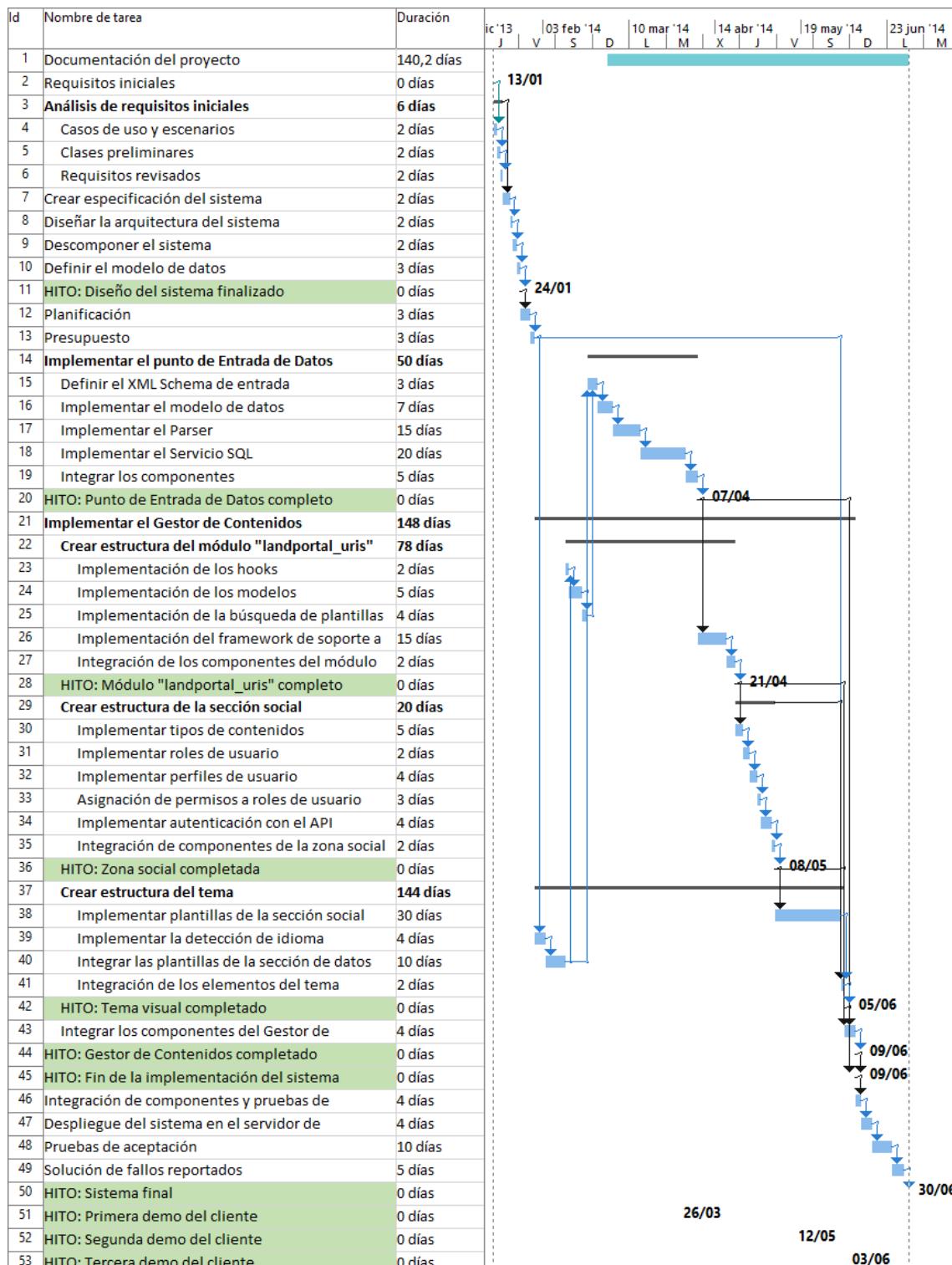


Figura 8.8: Diagrama de Gantt con la planificación real del proyecto

## 8.4. Valoración

Como se ha podido comprobar existen varias diferencias entre la planificación inicial (sección 8.2) y la planificación real (sección 8.3). A pesar de que la planificación inicial nunca es perfecta y es normal que la planificación real no se ajuste completamente, en esta sección se realizará una valoración de ambas explicando las razones para dichas divergencias.

- Durante el desarrollo del proyecto ha habido algunos cambios en los requisitos iniciales, lo que ha obligado a reestructurar algunos componentes del sistema que ya estaban implementados o a reorganizar tareas futuras.
- Como se puede comprobar en la planificación real, se han añadido tres hitos para demostraciones por parte del cliente. El cliente del proyecto ha necesitado encontrar financiación para seguir adelante con futuros proyectos, uno de los mecanismos para conseguir inversores ha sido realizar varias demostraciones del portal que se ha desarrollado en este proyecto. La necesidad de realizar estas demostraciones ha obligado a reestructurar algunas tareas para finalizar partes del sistema y que se pudieran ir enseñando.
- El Punto de Entrada de Datos ha tenido varios problemas de rendimiento a lo largo de su desarrollo. La realización de pruebas de rendimiento y su posterior solución ha alargado el desarrollo de este componente, lo que ha repercutido en el resto de componentes del sistema. El resultado de estas pruebas puede verse con detalle en la sección “Pruebas de rendimiento” perteneciente al capítulo 7.
- El desarrollo como parte de un equipo hace que unas partes del sistema sean dependientes de otras que, en muchas ocasiones, son realizadas por otras personas. La coordinación con el resto de miembros del equipo ha producido variaciones en las fechas de inicio y fin de algunos componentes e hitos.

# Capítulo 9

## Presupuesto

En este capítulo se presentarán los presupuestos de costes y de cliente, en los cuales se desglosan los diferentes costes del proyecto.

### 9.1. Presupuesto de costes

La figura 9.1 muestra de forma detallada el presupuesto de costes con todos los costes del proyecto. Los valores utilizados para la realización del presupuesto son los siguientes:

- Las horas de trabajo totales serán las obtenidas en la “Planificación inicial”. En total 820h.
- El número de horas de trabajo diarias será de 8h con 20 jornadas de trabajo mensuales.
- El porcentaje del salario del desarrollador destinado a la seguridad social será del 35 %.

Concepto	Recurso	Unidades	Precio/Unidad	Precio/Hora	Precio/mes	Total Recurso	Total Concepto
Materiales	Servidor Amazon IP elástica	1	- €	0,10 €	- €	84,46 €	150,81 €
Trabajo	Salario del programador	1	- €	20,00 €	3.200,00 €	16.400,00 €	19.000,00 €
	Desarrollo de documentación y manuales	1	1.000,00 €	- €	- €	1.000,00 €	
	Preparación de la oferta inicial	1	1.000,00 €	- €	- €	1.000,00 €	
	Instalación y configuración del sistema	1	600,00 €	- €	- €	600,00 €	
Equipamiento e instalaciones	Equipamiento del desarrollador	1	- €	- €	41,70 €	213,71 €	6.388,96 €
	Licencia PyCharm	1	179,00 €	- €	- €	179,00 €	
	Gastos de oficina	1	- €	- €	800,00 €	4.100,00 €	
	Otros (luz, agua, etc)	1	- €	- €	300,00 €	1.537,50 €	
	Internet	1	- €	- €	70,00 €	358,75 €	
Otros gastos	Imprevistos y reservas	1	4.000,00 €	- €	- €	4.000,00 €	10.252,50 €
	Gastos de gestión	1	- €	- €	100,00 €	512,50 €	
	Seguridad social	1	- €	7,00 €	- €	5.740,00 €	
					Total	35.792,27 €	

Figura 9.1: Presupuesto de costes del proyecto

El presupuesto de costes está dividido en 4 categorías que se describirán a continuación

#### 9.1.1. Materiales

Los materiales se refieren al hardware y software con los que se quedará el cliente después de finalizar el desarrollo, pero que tendrán un coste durante el mismo.

- El **servidor** utilizado durante el desarrollo será un servidor *Amazon EC2 m3.large*. El coste de este servidor será de 0,14 \$ por cada hora de funcionamiento (sin IVA)<sup>1</sup>. Este servidor correrá a cargo del equipo de desarrollo durante el proyecto, una vez finalizado el proyecto correrá a cargo del cliente.
- Una dirección **elastic IP** permite asignar una misma dirección IP a diferentes instancias y conservarla sin que se modifique al apagar o encender la máquina. El coste de este servicio es de 0,11 \$ por hora (sin IVA).

### 9.1.2. Trabajo personal

El trabajo personal se refiere al trabajo realizado por los desarrolladores del proyecto.

El salario del desarrollador será de 20€ por cada hora de trabajo, lo que equivale a 3200€ mensuales trabajando 8 horas diarias durante 20 días al mes. En total resulta en 16400€ para todo el proyecto.

También se incluye el coste del desarrollo de la documentación y los manuales que se entregarán al cliente, así como la instalación y configuración del sistema y la preparación de la oferta inicial, conjuntamente tienen un coste de 2600€.

### 9.1.3. Equipamiento e instalaciones

Se incluyen aquí todos los costes relativos al equipamiento utilizado por los desarrolladores así como las instalaciones en las que se efectuará el trabajo.

El lugar de trabajo será una oficina especialmente dedicada, con un coste de 800€ cada mes, lo que produce un coste de 4100€ para el proyecto. Al coste de la oficina se añadirán los gastos de agua, luz e Internet, conjuntamente tienen un coste de 1896,25€ en el proyecto.

Además también se incluirá en esta partida el gasto necesario para amortizar el equipamiento del desarrollador. El equipamiento ha tenido un coste inicial de 1500€ y se espera amortizar en 3 años (36 meses). El coste de amortización de este equipamiento es de 41,70€ mensuales, lo que produce un coste de 213,71€ a lo largo del proyecto.

Por último, se incluye también el precio del entorno de desarrollo integrado utilizado por el desarrollador, este tiene un coste de 179€<sup>2</sup>.

### 9.1.4. Otros gastos

En esta partida se incluyen el resto de gastos que afectan al proyecto y no forman parte de las partidas anteriores. Se incluye aquí un fondo para posibles imprevistos que surjan a lo largo del desarrollo, este fondo está valorado en 4000€.

También se incluyen los gastos de gestoría con un coste de 100€ mensuales, lo que resulta en 512,50€ para el proyecto completo. Los costes de la seguridad social del desarrollador se calculan tomando el 35 % de su salario, lo que resulta en 7€ a la hora y 5740€ a lo largo del proyecto.

---

<sup>1</sup>La información detallada de los precios de Amazon EC2 puede consultarse en <http://aws.amazon.com/es/ec2/pricing/>

<sup>2</sup>La información detallada sobre el precio de este componente puede consultarse en <http://www.jetbrains.com/pycharm/buy/>

### 9.1.5. Total

Tras sumar los costes de todas las partidas se obtiene un coste total del proyecto de 35792,27 €. Puesto que este es el presupuesto de costes, no se incluirán los impuestos. Los impuestos serán incluidos en el presupuesto del cliente tras aplicar el beneficio deseado.

## 9.2. Presupuesto del cliente

La figura 9.2 muestra el presupuesto del cliente. En este presupuesto se han eliminado los detalles sobre los costes internos.

Para facilitar la comprensión del presupuesto por parte del cliente los costes internos se han dividido según los módulos con los que cuenta el sistema. El peso de cada módulo se ha calculado dividiendo su tiempo de desarrollo entre el tiempo de desarrollo total del proyecto (exceptuando el tiempo destinado a tareas de pruebas, diseño y análisis).

Para la realización de este presupuesto se han utilizado dos nuevas variables:

- El beneficio que se espera obtener del proyecto es de un 15 % sobre su coste total.
- Los impuestos aplicados corresponden a un IVA del 21 %

El coste total del proyecto incluyendo un beneficio del 15 % es de 41161,11 €, tras aplicar los impuestos el coste final del proyecto resulta en 49804,94 €.

Concepto	Recurso	Unidades	Total Recurso	Total Concepto
Materiales	Servidor	1	150,81 €	150,81 €
Trabajo	Desarrollo de documentación y manuales	1	1.000,00 €	35.641,46 €
	Instalación y configuración del sistema	1	600,00 €	
	Desarrollo del punto de entrada de datos	1	10.930,74 €	
	Desarrollo de la sección social	1	6.246,14 €	
	Desarrollo del tema visual	1	8.119,98 €	
	Desarrollo del módulo LandPortal URIs	1	8.744,60 €	
	<b>Total</b>			41.161,11 €
	<b>Total con Impuestos</b>			49.804,94 €

Figura 9.2: Presupuesto del cliente

### 9.2.1. Condiciones del presupuesto

A continuación se detallarán las condiciones a tener en cuenta por el cliente para aceptar este presupuesto:

- El servidor utilizado será una instancia de *Amazon EC2 m3.large* y se entregará totalmente configurado y con el sistema en marcha. Una vez entregado el sistema será el cliente el encargado de pagar el coste de funcionamiento de dicho servidor. Junto con el sistema también se entregarán unos *scripts* que ejecuten la instalación automática del sistema. Dichos *scripts* requieren que el sistema resultante sea configurado manualmente. En caso de que el cliente decidiera utilizar otro tipo de *hosting* diferente al entregado la configuración no entrará en la garantía y tendría lugar como un proyecto nuevo.
- El código fuente del sistema estará disponible públicamente para su consulta en GitHub<sup>3</sup>. En caso de que el cliente decida modificar alguna parte del mismo, deberá crear previamente un

<sup>3</sup>Todos los repositorios que contienen el código del sistema están públicamente accesibles en <https://github.com/weso>

*fork* del repositorio que almacena el código antes de realizar alguna modificación. El encargado de desplegar las modificaciones resultantes de *forks* del sistema será en todo caso el propio cliente.

- Tras la entrega del sistema, el cliente contará con una garantía de un año. Durante dicho periodo de garantía el cliente contará con actualizaciones para fallos de seguridad sin un coste añadido. En cualquier caso, los desarrolladores no se harán responsables de los componentes que hayan sido modificados por el cliente tras su entrega.
- Los posibles bugs que no conlleven problemas de seguridad sólo se arreglarán de forma gratuita durante los tres meses posteriores a la entrega del sistema: Puesto que se ha efectuado un periodo de pruebas de aceptación en las cuales ha participado el propio cliente, este tipo de fallos se considerarán un fallo en dicho periodo de prueba del que será responsable el cliente.

# **Capítulo 10**

## **Manuales del sistema**

En este capítulo se encontrarán los distintos manuales del sistema, incluyendo los manuales de instalación, configuración y uso.

### **10.1. Manual de instalación**

En esta sección se explicarán los pasos necesarios para instalar el sistema y todas las herramientas requeridas para su correcto funcionamiento.

El manual de instalación ha sido originalmente creado para el cliente y se encuentra disponible en el anexo “Installation manual” (escrito en inglés), perteneciente al capítulo 12.

### **10.2. Manual de configuración**

Tras la instalación de todos los componentes del portal, será necesario realizar una configuración manual en la parte del gestor de contenidos.

Al igual que el manual de instalación, el manual de configuración ha sido originalmente creado para el cliente y se encuentra disponible en el anexo “Configuration manual” (escrito en inglés) perteneciente al capítulo 12.

### **10.3. Manual de usuario**

Dado que este proyecto consiste en la creación de un portal de datos, y no cuenta con una estructura compleja de cara al usuario como podría ser una doble interfaz de comunicación entre una parte web y una aplicación móvil, se ha optado por orientar el manual de usuario hacia una serie de preguntas y respuestas. Estas preguntas y respuestas tratarán los temas que se han considerado más interesantes para los usuarios.

Al igual que el resto de manuales, esta serie de preguntas y respuestas han sido originalmente creados para el cliente y se encuentran disponibles (escritas en inglés) en el anexo “User manual” perteneciente al capítulo 12.

# Capítulo 11

## Conclusiones y ampliaciones

En este capítulo se incluirán las conclusiones extraídas a lo largo del proyecto así como las posibles ampliaciones que se podrían llevar a cabo para aumentar el valor del sistema.

### 11.1. Conclusiones

Una vez finalizado el sistema es el momento de realizar una retrospectiva y extraer conclusiones, tanto técnicas como personales.

En primer lugar, se han adquirido multitud de conocimientos en nuevos lenguajes y frameworks que nunca había utilizado antes de este proyecto. En concreto, el conocimiento de un nuevo lenguaje de programación no se limita únicamente al conocimiento de su sintaxis, si no que también incluye el conocimiento de sus conceptos (por ejemplo: *duck typing* en Python frente a tipado fuerte en Java) y abstracciones (por ejemplo: *list comprehensions* y *generator expressions* en Python frente a bucles *for* y *foreach* en PHP), y la forma de combinarlas para conseguir un resultado adecuado.

También se ha comprendido la necesidad de evitar caer en la optimización prematura durante el desarrollo. La optimización prematura causa en multitud de ocasiones un aumento en la complejidad del código y, por tanto, en el riesgo de introducir nuevos errores. En relación con esto, también se ha aprendido a utilizar un *profiler* para obtener distintas métricas objetivas sobre el funcionamiento del programa y analizar así qué puntos requieren una optimización y qué puntos funcionan adecuadamente.

El tamaño del sistema a desarrollar ha puesto de manifiesto los beneficios del uso de una buena arquitectura que permita desacoplar los diferentes componentes. Este desacoplamiento entre los diferentes componentes ha facilitado su desarrollo y hace también posible el remplazo de un componente por otro con mínimos cambios en el resto del sistema.

Siendo este el primer proyecto en el que participo como parte de un equipo de desarrollo y con un cliente real al que se destina el proyecto, ha sido muy importante aprender a trabajar en equipo y a tomar decisiones sobre el sistema de forma conjunta. En relación con esto, también ha sido importante tomar responsabilidad sobre diversas partes del sistema y comprender que de su correcto funcionamiento depende el trabajo del resto de miembros del equipo.

Dado que, como se ha dicho antes, este es mi primer proyecto con un cliente real, ha sido también necesario aprender a trabajar manteniendo hitos inamovibles (o *deadlines*) y adoptando requisitos cambiantes.

En varias ocasiones durante el desarrollo se han tenido que tomar decisiones para poder alcanzar un determinado hito, esto me ha introducido al concepto de *deuda técnica*<sup>1</sup>, sus beneficios e

---

<sup>1</sup>El concepto de *deuda técnica* fue creado por Ward Cunningham para explicar el coste producido en el sistema

inconvenientes y la forma de utilizarlo en las situaciones que sea necesario.

En resumen, este proyecto ha supuesto multitud de cambios para mí: desde cambios en las herramientas y metodologías de desarrollo hasta cambios en la organización y coordinación del trabajo. Formar parte de un equipo de trabajo en el que todos los compañeros superan mis conocimientos ha sido una gran lección tanto técnica como personal, y aprender a aprovechar esa situación para crecer y mejorar en ambos sentidos ha sido, sin ningún tipo de duda, un acierto.

## 11.2. Ampliaciones

A continuación se enumerarán las posibles ampliaciones que se podrían realizar para aumentar el valor y la funcionalidad del sistema.

### 11.2.1. Incluir soporte a nuevas visualizaciones en el subsistema de datos

El framework de soporte a visualizaciones podría ser ampliado para facilitar la creación de nuevos tipos de visualizaciones con las que enriquecer la sección del portal dedicada a los datos.

### 11.2.2. Incluir soporte a nuevas redes sociales para iniciar sesión en el sistema

Actualmente el sistema soporta el inicio de sesión utilizando las redes sociales Twitter y Facebook. Permitir el inicio de sesión en el portal utilizando nuevas redes sociales como Google+, LinkedIn o GitHub facilitaría el acceso a los usuarios.

### 11.2.3. Facilitar el mecanismo de obtención de claves de acceso al API

Actualmente los usuarios que quieran obtener una clave de acceso al API deben enviar un correo electrónico a la administración del portal, para que sea el administrador quien modifique el rol del usuario y le permita obtener su clave de acceso al API.

Una ampliación interesante para los usuarios consistiría en permitirles solicitar una nueva clave de acceso al API de forma automática desde la vista de su perfil de usuario.

### 11.2.4. Gestión automática de los debates

Actualmente es la administración del portal quien se encarga de abrir y cerrar los debates cuando se alcance su periodo de participación.

Una ampliación destinada a gestionar automáticamente la apertura y el cierre de los debates facilitaría la tarea de los administradores, sobre todo cuando el número de debates existentes es muy alto.

---

por aquellas situaciones en las que es necesario adelantar trabajo (aunque no sea de la forma más correcta) y arreglar lo ya hecho (pagar la deuda técnica) posteriormente. Para más información sobre este concepto se recomienda el artículo de Martin Fowler al respecto [Fowb]

### 11.2.5. Migración del contenido desde el viejo LandPortal

El viejo LandPortal cuenta con una gran cantidad de información creada por sus usuarios a lo largo del tiempo. Una posible ampliación consistiría en migrar dicha información desde el viejo al nuevo portal.

La tesis de Alan Chavoshe, titulada “Linked Data for the Land Portal” [Cha13], contiene una explicación detallada de la estructura con la que se almacenan los datos en el viejo Land Portal. Atendiendo a lo expuesto en dicha tesis, el modelo de datos del viejo Land Portal difiere bastante del modelo utilizado en este proyecto.

El primer paso para realizar esta migración requeriría crear un nuevo módulo que extienda del módulo *migrate*<sup>2</sup>.

El mayor problema provendría de los tipos de contenido que no existen en el nuevo portal como: *OrganisationType*, *GPL Drivers* o *Expertise*. Para migrar este tipo de contenidos sería necesario modificar la estructura de datos del nuevo Land Portal y modificar también las vistas correspondientes para presentarlos a los usuarios.

Una primera estimación sobre el tiempo necesario para realizar esta ampliación sería entre 100 o 150 horas, en las que modificar las estructuras del modelo de datos y las vistas necesarias.

### 11.2.6. Futuros proyectos con el IFAD

Además de las posibles ampliaciones detalladas anteriormente, también existen una serie de futuros proyectos que se realizarán con el mismo cliente.

- Un *hackatón*<sup>3</sup> para la creación de nuevos componentes del portal. Concretamente el *hackatón* consistiría en crear nuevos importadores para incluir catálogos de datos en el portal y nuevas visualizaciones con las que presentar los datos ya existentes. Originalmente el hackatón ha sido planteado para tener lugar durante el verano de 2014 en Alemania.
- Creación de la biblioteca o LandLibrary. El LandLibrary es la tercera sección del portal y tiene como objetivo ser un repositorio de documentos, publicaciones, estudios, mapas, etc. El LandLibrary tendrá también capacidad de búsqueda y contará con un enriquecimiento semántico de los metadatos de las publicaciones que albergue.

La información almacenada en el LandLibrary se hará pública en forma de datos enlazados abiertos para ser utilizados por servicios de terceros y para ser incluidos en diferentes partes del portal, concretamente en el LandBook.

Por su gran tamaño, esta ampliación tendrá lugar como un proyecto con entidad propia. Este proyecto está planificado para comenzar en otoño del 2014.

---

<sup>2</sup>El módulo *migrate* es un módulo de Drupal destinado a facilitar la migración de contenidos entre distintos portales. El módulo *migrate* está disponible en <https://www.drupal.org/project/migrate>

<sup>3</sup>Un *hackatón* o *hackathon* es un encuentro de programadores destinado a realizar un desarrollo colaborativo de software

# Capítulo 12

## Anexos

### 12.1. Material entregado

En esta sección se describirá el material entregado junto a la presente documentación. El material entregado se encuentra en un fichero comprimido y tiene el siguiente contenido:

- **Mediciones receiver** Este fichero contiene los resultados obtenidos de las mediciones realizadas durante las pruebas de rendimiento del Punto de Entrada de Datos. El análisis de los resultados puede consultarse con detalle en la sección 7.2. Para una mayor comodidad del lector, los resultados de las mediciones también se encuentran en forma tabular en el anexo 12.2.
- **Planificación inicial** Este fichero contiene la planificación inicial del proyecto. La planificación fue explicada con detalle en la sección 8.2.
- **Planificación final** Este fichero contiene la planificación real del proyecto. La planificación fue explicada con detalle en la sección 8.3.
- **Presupuestos** Este fichero contiene los presupuestos de cliente y de costes para el proyecto. Dichos presupuestos pueden consultarse en las secciones 9.1 y 9.1.
- **Database** Este fichero alberga los contenidos de la base de datos del LandPortal y puede ser utilizado para restaurar una copia de la base de datos sin necesidad de ejecutar los importadores<sup>1</sup>. El fichero se encuentra comprimido debido a que, por el gran volumen de datos que se maneja, el mismo fichero sin compresión ocupa 200MB.
- **Fuentes** Este directorio contiene los ficheros de código fuente del sistema construido.
  - **Installation scripts** Este directorio contiene los *scripts* de instalación del sistema. Estos *scripts* se han utilizado con éxito para desplegar el sistema en producción
  - **Landportal model**<sup>2</sup> Este directorio contiene el código del modelo de datos del nuevo Land Portal. Este modelo de datos puede consultarse con detalle en la sección 5.3.1.
  - **Landportal drupal**<sup>3</sup> Este directorio contiene el tema y los módulos de Drupal desarrollados durante este proyecto. El diseño de estos componentes puede consultarse en las secciones 5.1.3 y 5.1.4.

<sup>1</sup>Los importadores se conectan a multitud de APIs y fuentes de datos externas para obtener los catálogos de datos que procesarán y enviarán al Punto de Entrada. Debido al número de fuentes externas a consultar y al volumen de datos, este proceso puede durar varias horas.

<sup>2</sup><https://github.com/weso/landportal-drupal>

<sup>3</sup>[https://github.com/weso/landportal\\_model](https://github.com/weso/landportal_model)

- **Landportal receiver<sup>4</sup>** Este directorio contiene el Punto de Entrada de Datos del sistema. El diseño de este componente fue explicado anteriormente en la sección 5.1.2.
- **Otros** Este directorio contiene varios ficheros relacionados con el proyecto:
  - **IFAD 2013 016 RFQ Technical Specifications** Este fichero contiene las especificaciones técnicas del proyecto *Rebuilding IFAD's LandPortal*, del cual forma parte el presente Trabajo Fin de Grado.
  - **SBC4D Technical Offer** Este fichero contiene la oferta técnica propuesta por la consultora SBC4D y bajo al cual se ha realizado este proyecto.

## 12.2. Resultados de las mediciones

La tabla 12.1 muestra los resultados obtenidos durante las mediciones y que se han utilizado para la realización de las figuras 7.1, 7.2 y 7.3.

Como ya se ha mencionado anteriormente en la sección “Proceso y ámbito de las mediciones” perteneciente al capítulo 7, para obtener estos resultados se han utilizado dos herramientas:

- El módulo *memory profiler* para medir el consumo de memoria durante el procesado de datos
- El comando *time* para medir el tiempo de ejecución del programa ante cada entrada

Los valores mostrados en la tabla 12.1 son el resultado de realizar tres mediciones y posteriormente calcular la mediana entre los datos obtenidos.

---

<sup>4</sup><https://github.com/weso/landportal-receiver>

Cuadro 12.1: Mediciones de rendimiento del Punto de Entrada de Datos

Mediciones de rendimiento del Punto de Entrada de Datos				
Número de observaciones	Tamaño del fichero (MB)	Uso de memoria (MB)	Tiempo (s)	Observaciones / Tiempo (Obs / s)
120	0,0506	39,5	1,552	77,31958763
1469	0,6355	86,5	12,705	115,6237702
1589	0,679	89,6	10,794	147,2114137
2080	0,8897	102,5	18,53	112,2504047
2600	1,1	120,7	22,938	113,3490278
4920	1,8	193,1	40,927	120,2140396
5215	1,9	201,6	44,27	117,7998645
5324	1,9	204,5	44,481	119,6915537
11178	4,2	374	93,964	118,9604529
11178	4,2	373,8	93,461	119,6006891
22356	8,8	657,8	192,125	116,3617437
28860	10,8	813,8	247,301	116,699892
33534	13	957,9	293,974	114,0713124
78246	29,7	2125,5	696,019	112,4193449
89424	35,5	2422,6	794,02	112,6218483
111780	42,3	2918,9	1059,911	105,461685

## 12.3. Installation manual

For a successful system installation the following elements are required:

- **Installation scripts.** To facilitate the installation task, and since this system has multiple internal and third-party components, the developers have created a series of scripts that automate the system installation.
- **GNU/Linux distribution.** The installation scripts have been only tested under Ubuntu 14.04 LTS, so it is highly recommended to use this distribution for the system deployment.

The first step consists of logging into the system with a non-root account. The non-root account is important because the file paths in the scripts are relative to the user's home directory, the root home directory is `/root` while the home directory for regular users is `/home/USERNAME`.

Once you have logged into the system with a non-root account, the following step consists of copying the content of the scripts folder into the user's home directory. Such as in the image 12.1, the files `install.sh` and `settings.php` and the folders `scripts` and `[solr]` must be located into the root of the user's home directory.

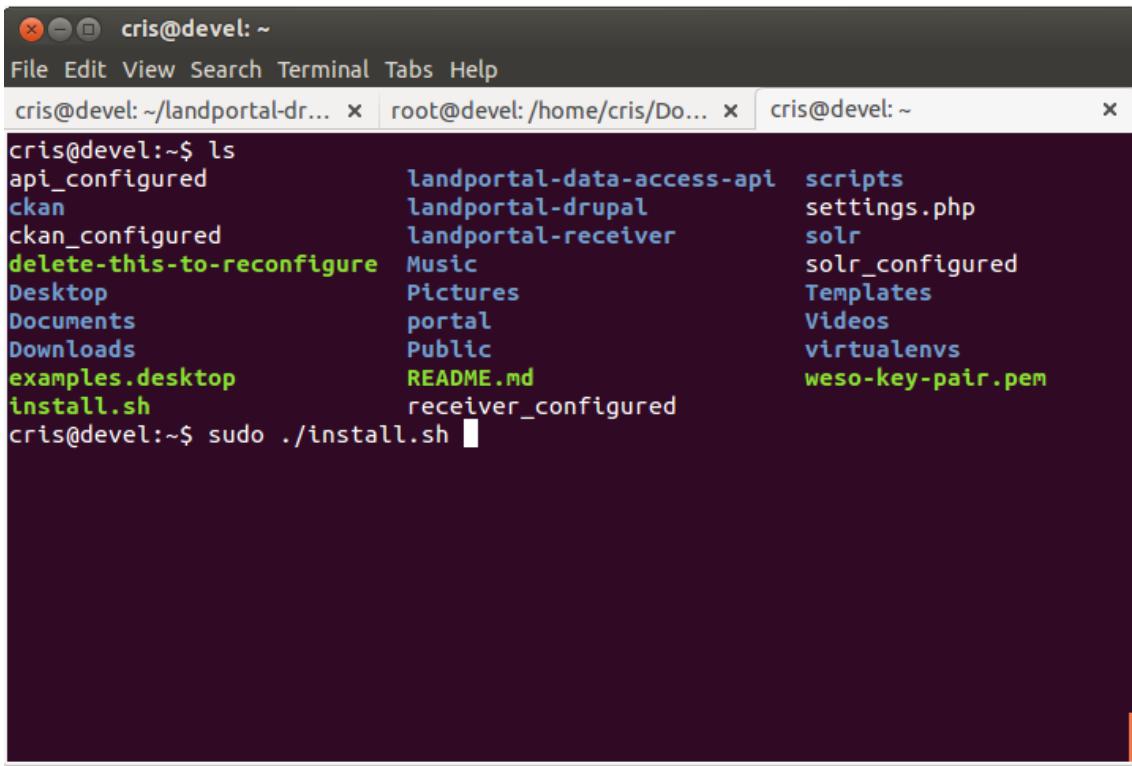
```
cris@devel: ~
File Edit View Search Terminal Tabs Help
cris@devel: ~/landportal-dr... x root@devel: /home/cris/Do... x cris@devel: ~ x
cris@devel:~$ ls
api_configured          landportal-data-access-api  scripts
ckan                   landportal-drupal           settings.php
ckan_configured         landportal-receiver      solr
delete-this-to-reconfigure Music                  solr_configured
Desktop                Pictures                 Templates
Documents              portal                  Videos
Downloads              Public                  virtualenvs
examples.desktop        README.md               weso-key-pair.pem
install.sh              receiver_configured
cris@devel:~$
```

Figura 12.1: User's home folder with the script files (underlined in red)

After copying the installation scripts into the home directory, the system installation can be easily triggered with the command `sudo ./install.sh`, which means to run the script `install.sh`, which is located into the current directory, with superuser privileges. The superuser privileges are required to install some packages and configurations. The image 12.2 shows the command before starting the installation.

When you hit the *enter* key, the installation will begin. The installation process is completely automated and requires no user interaction, but since it is such a big system, the installation can last a long time.

After the installation ends, the system is completely functional, but it still needs some configuration. Please, take a look into the “Configuration manual”.



The screenshot shows a terminal window with three tabs. The active tab shows the command `cris@devel:~$ ls` followed by a listing of files and directories including `api_configured`, `ckan`, `ckan_configured`, `delete-this-to-reconfigure`, `Desktop`, `Documents`, `Downloads`, `examples.desktop`, `install.sh`, `landportal-data-access-api`, `landportal-drupal`, `landportal-receiver`, `Music`, `Pictures`, `portal`, `Public`, `README.md`, `receiver_configured`, `scripts`, `settings.php`, `solr`, `solr_configured`, `Templates`, `Videos`, `virtualenvs`, and `weso-key-pair.pem`. Below this, the command `cris@devel:~$ sudo ./install.sh` is entered.

Figura 12.2: Launch system installation command

## 12.4. Configuration manual

This configuration manual is also published online in the LandPortal’s Drupal repository at GitHub<sup>5</sup>.

Here are the instructions for configuring the new *LandPortal*. All of the following steps can be easily followed using the Drupal’s administration interface. To use the administration interface you must log into the *LandPortal* using an account with administrator privileges.

**Important notice:** sometimes in this manual, there will appear some paths. Those paths are used to easily navigate for the Drupal’s administration interface. Each path is relative to the current server host. This means that, if the current server host is *landportal.weso.es*, the path *admin/appearance* really means: *landportal.weso.es/admin/appearance*.

### 12.4.1. Enable the *LandPortal* theme

The new *LandPortal* appearance is provided by a custom theme created specially for the occasion. The theme receives the name of “book”. To enable the “book” theme go to the tab *Appearance* in the top bar of the administration interface.

Once in the *Appearance* tab, scroll down to the bottom of the page, and in the section *disabled themes* click “Enable and set default” in the *Book theme for LandPortal*. Now, the “book” theme is used by default in all pages<sup>6</sup>

<sup>5</sup><https://github.com/weso/landportal-drupal>

<sup>6</sup>For security reasons the administration interface will still use the default Drupal theme.

### 12.4.1.1. Configuring the *favicon*

The *favicon* is a little icon that shows in the browser's tabs and in the browser's bookmark section, representing the entire site. The new *LandPortal* has a nice *favicon* which can be easily enabled:

1. Go to the path *admin/appearance/settings*
2. Go to the section *Shortcut icon settings* and uncheck the option *Use the default shortcut icon*
3. In the field *Path to custom icon* write **sites/all/themes/book/favicon.png**

The image 12.3 shows how those options should look like.

**SHORTCUT ICON SETTINGS**

Your shortcut icon, or 'favicon', is displayed in the address bar and bookmarks of most browsers.

Use the default shortcut icon.  
Check here if you want the theme to use the default shortcut icon.

**Path to custom icon**

`sites/all/themes/book/favicon.png`

The path to the image file you would like to use as your custom shortcut icon.

**Upload icon image**

No file chosen

If you don't have direct file access to the server, use this field to upload your shortcut icon.

Figura 12.3: *LandPortal*'s favicon configuration

### 12.4.1.2. Configuring *home* and *error* pages

The new *LandPortal* has a nice home and error pages. The home page is called *hub*, and it has been designed to be the entry point to the new *LandPortal*. The error page shows a nice compass to entertain the users when an error happens. To enable those views take the following steps:

1. Go to the path *admin/config/system/site-information*
2. In the field *Default front page* write **home**
3. In the *ERROR PAGES* fields write **e404**

The image 12.4 shows how those fields should look like.

### 12.4.1.3. Configuring the *login* redirection

The default behaviour of Drupal consists of redirecting a user to its profile page after he has logged into the system. As stated in the previous section, the *hub* page has been designed as the entry point for the new *LandPortal*. We want to override the Drupal's default behaviour to load

<b>Default front page</b>	<input type="text" value="http://localhost:1100/ home"/>
Optionally, specify a relative URL to display as the front page. Leave blank to display the default content feed.	
<b>ERROR PAGES</b>	
<b>Default 403 (access denied) page</b>	<input type="text" value="http://localhost:1100/ e404"/>
This page is displayed when the requested document is denied to the current user. Leave blank to display a generic "access denied" page.	
<b>Default 404 (not found) page</b>	<input type="text" value="http://localhost:1100/ e404"/>
This page is displayed when no other content matches the requested document. Leave blank to display a generic "page not found" page.	

Figura 12.4: *LandPortal*'s home and error pages

our *hub* page after a user logs into the system.

Two steps are required to achieve this:

1. Go to the path *admin/config/system/actions*. In the bottom page you will see the page creation form. Choose the option *redirect to URL* in the dropdown menu and click the button *Create*. A new page will load in which you can set the action label (we suggest something readable, for example: “*LandPortal redirect on login*”). In the field *url* write **home** (the *hub* path). The image 12.5 shows how those options should look.
2. Go to the path *admin/structure/trigger/user* and in the option *After an user has logged in* select the action created in the previous step. Then click the button *assign*.

An advanced action offers additional configuration options which may be filled out below. Changing the *I Trigger module when assigning actions to system events*, so it is best if it is as descriptive as possible (for

#### **Label**

A unique label for this advanced action. This label will be displayed in the interface of modules that integ

#### **URL \***

The URL to which the user should be redirected. This can be an internal URL like node/1234 or an extern

**Save**

Figura 12.5: User redirection configuration

#### 12.4.2. Import the *taxonomy terms*

The *LandDebate* uses 5 different *taxonomies* or *vocabulary* to classify the different contents.

- Continents

- Countries
- Regions
- Debate status
- Topics

Each of those *taxonomies* is populated by *terms*. Unfortunately, those *terms* must be imported in a manual way (the import process only needs to be done once). The following steps explain how to import the *taxonomy terms*:

1. Go to the path *admin/structure/taxonomy*
2. Click the button *CSV IMPORT* in the right upper side
3. Choose *Translation* as the type of import. In the field *list of languages* write **und, es, fr**
4. Paste the taxonomy terms<sup>7</sup> into the text box called *Terms to import*
5. in the field *vocabulary choice* select the taxonomy **Continents**
6. Repeat the above steps with the **countries**<sup>8</sup>, **topics**<sup>9</sup>, **regions**<sup>10</sup> and **debate status**<sup>11</sup> taxonomies. Don't forget to select the corresponding destiny taxonomy in each case.

The images 12.7 and 12.6 illustrate some of the above steps.

1. What do you want to import?

2. Where are items to import?

3. How is your source formatted?

4. Which vocabulary do you want to import into?

5. When a term exists, what to do with it?

6. Informations on process and advanced options

**Vocabulary choice**

Topics

Automatically delete all terms of the selected vocabulary before import  
Warning: You won't be warned before deletion.

Automatically check vocabulary hierarchy  
Warning: to calculate true hierarchy of vocabulary is memory intensive. Choose to check automatically only if your vocabulary is little.

Import    Reset to defaults

Figura 12.6: Taxonomy import destiny vocabulary selection

### 12.4.3. Configure the content types

A new Drupal installation creates two content types by default, those content types are called *Article* and *Basic page* and will not be used into the new *LandPortal*, so they can be omitted or deleted without problem. The content types can be accessed in the path *admin/structure/types*.

<sup>7</sup>[https://github.com/weso/landportal-drupal/blob/develop/taxonomy\\_terms/continents.csv](https://github.com/weso/landportal-drupal/blob/develop/taxonomy_terms/continents.csv)

<sup>8</sup>[https://github.com/weso/landportal-drupal/blob/develop/taxonomy\\_terms/countries.csv](https://github.com/weso/landportal-drupal/blob/develop/taxonomy_terms/countries.csv)

<sup>9</sup>[https://github.com/weso/landportal-drupal/blob/develop/taxonomy\\_terms/topics.csv](https://github.com/weso/landportal-drupal/blob/develop/taxonomy_terms/topics.csv)

<sup>10</sup>[https://github.com/weso/landportal-drupal/blob/develop/taxonomy\\_terms/regions.csv](https://github.com/weso/landportal-drupal/blob/develop/taxonomy_terms/regions.csv)

<sup>11</sup>[https://github.com/weso/landportal-drupal/blob/develop/taxonomy\\_terms/debate\\_status.csv](https://github.com/weso/landportal-drupal/blob/develop/taxonomy_terms/debate_status.csv)

<b>1. What do you want to import?</b>	Choose the type of import. Help for each type is displayed below when format is selected.
<b>2. Where are items to import?</b>	<b>Important:</b> If you have a tree structure, you should import it before fields or translations, and for each Translate mode.
<b>3. How is your source formatted?</b>	<input type="radio"/> Structure <input type="radio"/> Fields <input checked="" type="radio"/> Translation
<b>4. Which vocabulary do you want to import into?</b>	Allow to import name and descriptions and their translation.
<b>5. When a term exists, what to do with it?</b>	<b>Line format</b> Vocabulary with Translate mode: term name/id, first translated term name name... Vocabulary with Localize mode: term name/id, first translation of term n translation of description...
<b>6. Informations on process and advanced options</b>	<b>Examples</b> <ul style="list-style-type: none"> <li>· foo, bar</li> <li>· "United Kingdom", "Royaume-Uni", "Vereinigte Königreich"</li> <li>· "Germany", "Allemagne", "A European country", "Un pays européen" [voca only]</li> </ul> <p>The term is in the first column followed by its translations. If the i18n mode is Localize, then description and Note:</p> <ul style="list-style-type: none"> <li>· With a vocabulary in Translate mode, a term with an undefined language cannot be translated, so do not fit import original terms.</li> <li>· With a vocabulary in Localize mode, only terms with a undefined language can be translated, so do not set terms.</li> </ul> <p><b>Type of source</b></p> <p><input checked="" type="radio"/> First item is the source term name  <input type="radio"/> First item is the source term id</p> <p>Choose how to identify the source term. If the source term doesn't exist, it will be created.</p> <ul style="list-style-type: none"> <li>· To use name is simpler, but cannot be used if vocabulary has duplicate names.</li> <li>· To use term id is quicker, but you need to export your vocabulary to get tids.</li> </ul> <p><b>List of languages</b></p> <p>und, es, fr</p> <p>Set the list of languages of terms, for example "en, fr, de".</p> <p><b>Notes:</b></p> <ul style="list-style-type: none"> <li>· All languages should have been enabled in <a href="#">Regional and language settings</a> before import.</li> </ul>

Figura 12.7: Taxonomy import type and languages configuration

#### 12.4.3.1. Configure the *Blog posts*

The *Blog posts* content type can be configured in the path *admin/structure/types/manage/blog-post*

- Change the option *preview before submitting* to *disabled*
- In the section *comment settings* uncheck the options *Threading* and *Allow comment title*
- In the section *comment settings* set the option *Preview comment* to **Disabled**

#### 12.4.3.2. Configure the *Debates*

The *Debates* content type can be configured in the path *admin/structure/types/manage/debate*

- Change the option *preview before submitting* to *disabled*
- In the section *comment settings* set the option *Default comment status for new content* to **Closed**
- In the section *comment settings* set the option *Preview comment* to **Disabled**

#### 12.4.3.3. Configure the *Events*

The *Events* content type can be configured in the path `admin/structure/types/manage/event`

- Change the option *preview before submitting* to *disabled*
- In the section *comment settings* uncheck the options *Threading* and *Allow comment title*
- In the section *comment settings* set the option *Preview comment* to **Disabled**
- In the section *comment settings* set the option *Default comment status for new content* to **Hidden**

#### 12.4.3.4. Configure the *News*

The *News* content type can be configured in the path `admin/structure/types/manage/news`. The configuration for this content type is the same as the configuration for the *Events*.

#### 12.4.3.5. Configure the *Organizations*

The *Organizations* content type can be configured in the path `admin/structure/types/manage/organization`. The configuration for this content type is the same as the configuration for the *Events*.

### 12.4.4. Configure the search

#### 12.4.4.1. Connect to the *Apache Solr* service

The new *LandPortal* uses *Apache Solr* to provide a high quality search service. The following steps are required to configure *Apache Solr*:

- Go to the *Configuration* tab in the top bar of the administration interface
- Go to the option *Apache Solr search* in the administration panel
- Go to the tab *Settings* and click the link named *edit*
- In the field *Solr server URL* write `http://localhost:8983/solr/drupal` and click the button *Save*

The image 12.8 shows how the *Apache Solr* search server should look after its configuration. The green colour means that Drupal has successfully contacted Solr.

NAME	URL
localhost server (Default)	http://localhost:8983/solr/drupal

**ADVANCED CONFIGURATION**

**Save configuration**

Figura 12.8: Solr connection configuration result

#### 12.4.4.2. Set *Apache Solr* as the default search provider

Drupal supports multiple search providers. To use *Apache Solr* as the default search provider in the new *LandPortal* take the following steps:

1. Go to the path `admin/config/search/settings`
2. In the section *Default search module* choose the option *Apache Solr search*

The image 12.9 shows how those options should look like.

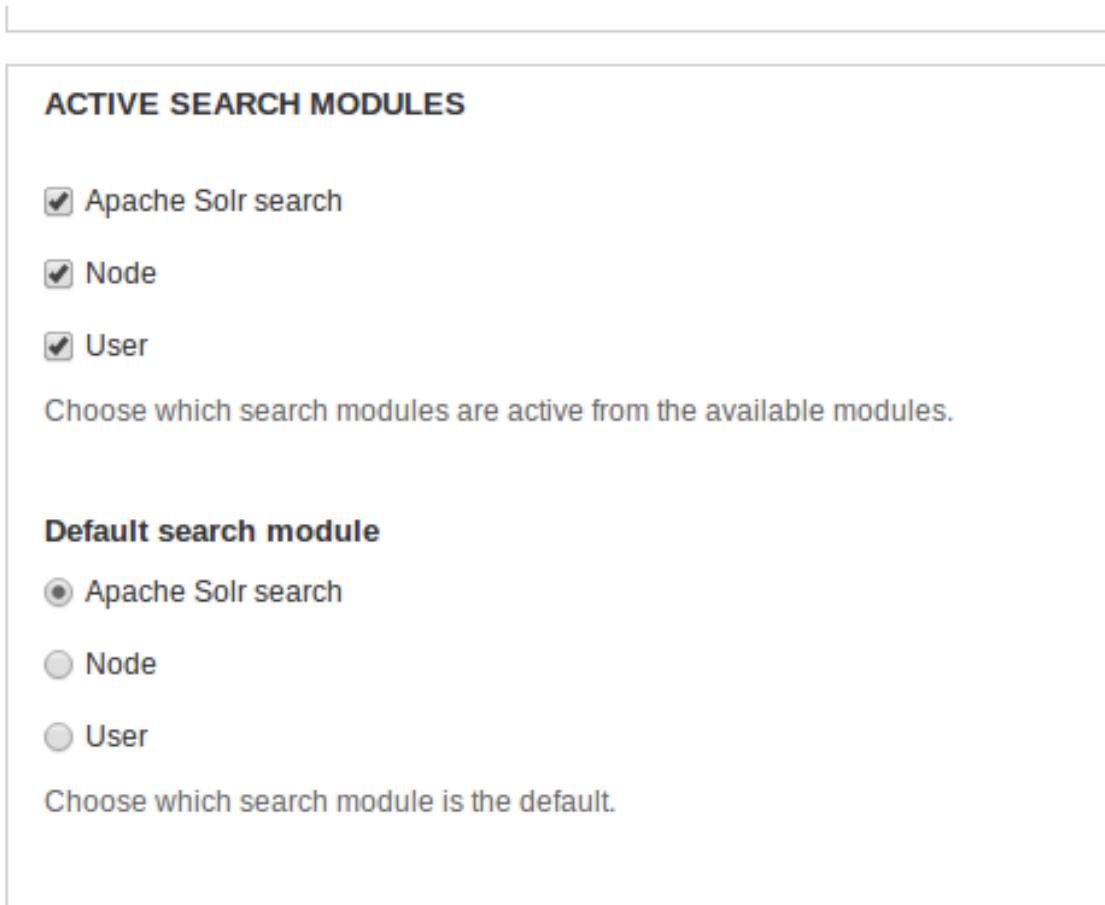


Figura 12.9: Solr as the default search server configuration

#### 12.4.5. Configure the *WYSIWYG* editor

The *WYSIWYG* module allows Drupal to show a nice text editor component in which the users can easily format the text and insert images. To enable this module take the following steps:

1. Go to the path `admin/config/content/wysiwyg`
2. For each profile choose the editor *markItUp 1.1.14*
3. After selecting the editor you can change its options and choose which buttons to show. We suggest enabling all the buttons for the best user experience.

The image 12.10 shows how to enable those buttons.

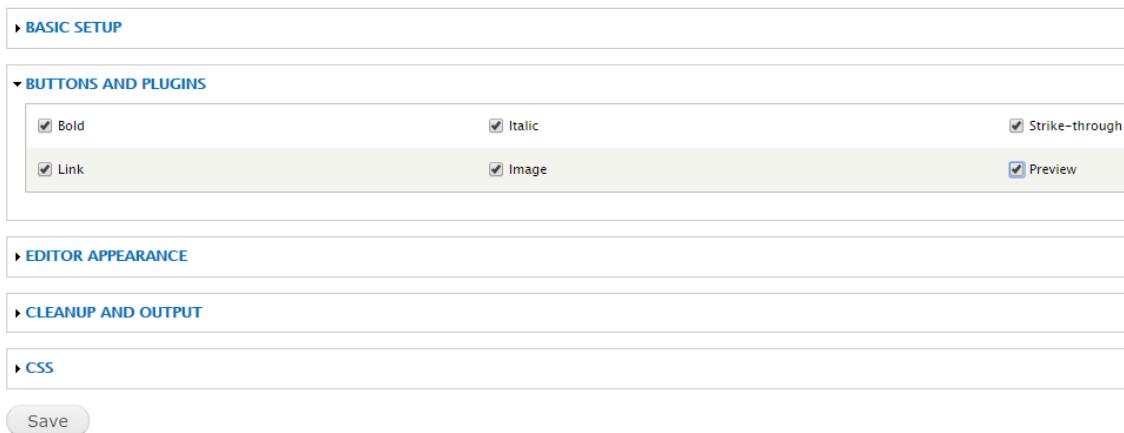


Figura 12.10: WYSIWYG text editor configuration

## 12.5. User manual

This is the user manual for the new *LandPortal*, it has been structured as a set of questions and answers.

### 12.5.1. How do I start a new debate?

To start a new debate, you must load the debates view in the path `/debate/debates`. If you are logged into the new *LandPortal* you will see a blue button to create a new debate in the right side.

After clicking that button the form to create a new debate will load. You must fill the form (the fields marked with an asterisk are required and can not be left empty) and click the *save* button. After clicking the *save* button (and if the form was correctly filled) the system will create the new debate.

By default all new debates are created with a “*coming soon*” status, which means that the comments are closed until the debate’s period starts. When an administrator changes the debate status to “*open*” and opens the comments, the users will be able to post new comments and participate into the debate.

### 12.5.2. How do I close a debate?

This action is restricted to users with administration privileges.

To close an open debate you must load the detailed view of the debate that you want to close. In the bottom left you will see a blue button to edit the debate.

After clicking that button, the system will load a form (similar to the debate creation form) in which you can choose the correct settings for the debate. In the form, change the status of the debate to “*closed*” and close the comments. Click the *save* button to save the changes.

After a debate has been closed, the users can see the existing comments, but can not create new ones.

### 12.5.3. How can I register into the portal?

To create a new account into the new *LandPortal* you must load the home view and click the “*sign in*” button in the upper right corner. Once in the *sign in* view you can choose between a tab to log into the system with an existing account, or to create a new account.

Choose the tab to create a new account and you will see the register form. To create your account fill the form (the required fields are marked with an asterisk and can not be left empty) and click the button “*register*”.

**WARNING** Your new account will not be usable right after the registration, For security reasons an administrator has to activate your account before you can log into the system.

### 12.5.4. How can I moderate new registrations?

This action is restricted to users with administration privileges.

By default, the new user registrations are in a “*disabled*” state, which means that the user can not log into the system. To enable the new user accounts, the admin must load the “*people*” tab in the administration view. The *people* view shows a list of all user accounts (disabled or not) existing into the system. To activate the user accounts, the administrator has to select the users and click the *update* button.

### 12.5.5. How can I moderate articles?

This action is restricted to users with administration privileges.

The administrator can edit or delete any *article*<sup>12</sup>. To delete a certain *article* you must load its detailed view and click the button *edit* or *delete*.

The administrator can also edit or delete any user comments. To do this, you must click the button *edit* or *delete* that appears under every comment.

## 12.6. Tutorial: creating a custom view

As Larry Garfield explains in [Gar06] Drupal uses the architectural pattern PAC<sup>13</sup> for organising its structure and functionality. The PAC pattern is less known and used than the architectural pattern MVC<sup>14</sup>. The new Land Portal tries to transform the PAC pattern to work in a way more similar to the MVC pattern.

In this tutorial we will explain how to create a custom view and include it in the new Land Portal. Those three steps are required and will be explained in detail.

1. Create the entry in the routes configuration file
2. Create the model that returns the required data
3. Create the view to show the data returned by the model

Before taking the following steps it is recommended to take a look at the structure of the *Land-Portal uris* module (section 5.1.4, chapter 5) and the activity of the custom templates rendering

---

<sup>12</sup>In this context, *articles* refers to any content in the social section of the LandPortal, such as news, events, debates, etc.

<sup>13</sup>Presentation Abstraction Control - [http://en.wikipedia.org/wiki/Presentation\\_abstraction-control](http://en.wikipedia.org/wiki/Presentation_abstraction-control)

<sup>14</sup>Model View Controller - <http://martinfowler.com/eaaDev/uiArchs.html>

(section 5.2.3, chapter 5).

### 12.6.1. Creating the route

The first step to create a custom view is to create the route under the users will access the view. The routes are declared in the *routes.json* file. This file is stored in the path *DRUPAL\_HOME/sites/all/modules/custom/modules/landportal\_uris/routes.json*.

The image 12.11 shows some entries of the *routes.json* file. Each entry has the following fields:

- **Path** The path specifies the URL of the view. The path is always relative to the main Drupal URL. To specify a *wildcard* use a % symbol. For example, in the image 12.11 the path *book/regions/%* will match the URLs *book/regions/1*, *book/regions/150*, etc
- **Name** The name is used to retrieve the corresponding model and view. The name *regions* will look for the model *Regions* and the view *regions.mustache*.
- **Title** The title is printed in the browser tab in which the user loads the template.
- **Navigation** The navigation sets the tab that will be selected in the header bar when te user loads the template.
- **Params** The params specify which parameters will be taken from the URL and passed to the model. The parameter corresponds to the position of the *wildcard* in the *path*, the position starts in 0. For example, in the image 12.11 the path *book/regions/%* has the *wildcard* in the position 2, being the position 0 *book* and the position 1 *regions*. If a path does not receive any arguments leave the parameters empty.
- **Redirect** Apart from the previous fields, an entry in the routes file may have a *redirect* field. This field contains a path that will be loaded automatically. For example, in the image 12.11 the path *book* will redirect to the path *book/regions*. When the *redirect* field is present, the fields *navigation* and *params* can be omitted.

```

37     },
38     {
39         "path": "book/regions",
40         "name": "regions",
41         "title": "Region list",
42         "navigation": "regions",
43         "params": []
44     },
45     {
46         "path": "book",
47         "name": "book",
48         "title": "Land Book",
49         "redirect": "book/regions"
50     },
51     {
52         "path": "book/regions/%",
53         "name": "regions",
54         "title": "Region",
55         "navigation": "regions",
56         "params": [2]
57     }

```

Figura 12.11: Snippet of the *routes.json* file content

The image 12.12 shows a custom created route for this tutorial. The route will be accessible in the path *helloworld* and will load the model *helloworld.php* and the template *helloworld.mustache*. In the following sections we will see how to create the model and the template.

```

113▼      {
114      "path": "helloworld",
115      "name": "helloworld",
116      "title": "Hello World!",
117      "navigation": "",
118      "params": []
119

```

Figura 12.12: Example of new entry in the *routes.json* file

### 12.6.2. Creating the model

The second step to create a custom view is to create a model. The models return the data that is shown to the user by the template. The models are created in the directory *DRUPAL\_HOME/sites/all/modules/custom/modules/landportal\_uris/models/*. The model will be automatically loaded by its name.

In the image 12.12 we created a new path to make a custom template. The path had the name *helloworld*, so our model will have the name *helloworld.php* in order to be automatically loaded by Drupal.

The image 12.13 shows the content of our new model. Every model must have two basic characteristics in order to be correctly loaded by Drupal:

- The class name of the model must be the same as the file name, but with the first letter uppercase. Our model file was called *helloworld.php* so the model's class must be called *Helloworld*.
- Every model must have a method *get* that returns an array with the contents to populate the template. In this case the model returns a single key called *name*, which will be rendered in the template.

```

1  <?php
2
3  class Helloworld {
4
5      public function get() {
6          return array(
7              "name" => "Land Portal"
8          );
9      }
10 }

```

Figura 12.13: Example of the model class in the *helloworld.php* file

### 12.6.3. Creating the template

The final step to create a custom view is to create the template that will be rendered to the user. The templates are created in the directory *DRUPAL\_HOME/sites/all/themes/book/views/*. The template will be automatically load by its name.

In the image 12.12 we created a new path to make a custom template. The path had the name *helloworld*, so our template will have the name *helloworld.mustache* in order to be automatically loaded by Drupal.

The templates are created using *mustache*<sup>15</sup>. The arguments between the mustaches ({{ and }}) will be automatically replaced by its value before the template is rendered to the user.

The image 12.14 shows the contents of our custom template in the file *helloworld.mustache*.

<sup>15</sup>It is not the object of this tutorial to explain *mustache* and its syntax. For more details about how *mustache* works, take a look at the official documentation in <http://mustache.github.io/>

The model that we created in the image 12.13 returned a key named *name* that we will use in the template.

The image 12.15 shows the result of the view that we have created in this tutorial. As you can see, the path corresponds to the one specified in the image 12.12, and the mustaches in the image 12.14 have been substituted by the data returned from the model (image 12.13).

```
1 <h1>
2   Hello {{name}}!
3 </h1>
```

Figura 12.14: Example of the template in the *helloworld.mustache* file

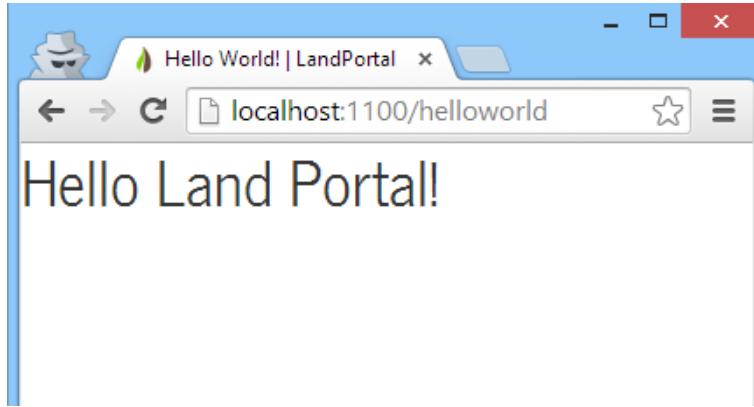


Figura 12.15: Result of the custom view

# Bibliografía

- [Bec02] Kent Beck. *Test Driven Development: By Example*. 2002. ISBN: 0321146530.
- [Ber06] Tim Berners-Lee. «Linked Data – Design Issues». En: (2006). URL: <http://www.w3.org/DesignIssues/LinkedData.html>.
- [Biz] Chris Bizer. «DBpedia Architecture». En: (). URL: <http://wiki.dbpedia.org/Architecture>.
- [Cha13] Alan Chavoshe. *Linked Data for the Land Portal*. 2013.
- [Dav13] Tim Davies. «Land Portal Strategy 2013 – 2016». En: (2013).
- [Div91] United Nations Statistics Division. «United Nations Standard Country Codes». En: (1991). URL: <http://www.iarc.fr/en/publications/pdfs-online/epi/sp95/sp95-app1.pdf>.
- [Fou] Open Knowledge Foundation. «Open Definition». En: (). URL: <http://opendefinition.org/>.
- [Fowa] Martin Fowler. «Continuous Integration». En: (). URL: <http://martinfowler.com/articles/continuousIntegration.html>.
- [Fowb] Martin Fowler. «Technical Debt». En: (). URL: <http://martinfowler.com/bliki/TechnicalDebt.html>.
- [Fowc] Martin Fowler. «UnitTest». En: (). URL: <http://martinfowler.com/bliki/UnitTest.html>.
- [Fowd] Martin Fowler. «XUnit». En: (). URL: <http://martinfowler.com/bliki/Xunit.html>.
- [FU] Food y Agriculture Organization of the United Nations. «FAO Geopolitical ontology (RDF)». En: (). URL: <http://www.fao.org/countryprofiles/geoinfo/geopolitical/resource/>.
- [Gar06] Larry Garfield. «MVC vs. PAC». En: (2006). URL: <http://www.garfieldtech.com/blog/mvc-vs-pac>.
- [Knu74] Donald E. Knuth. «Structured programming with go to statements». En: *Computing Surveys* 6 (1974), págs. 261-301.
- [Lut13] Mark Lutz. *Learning Python*. 5.<sup>a</sup> ed. O'Reilly, jun. de 2013. ISBN: 9781449355739.
- [PMI09] Project Management Institute (PMI). *A Guide to the Project Management Body of Knowledge*. 2009. ISBN: 1933890517.
- [Sta] International Organization for Standardization. «Country Codes - ISO 3166». En: (). URL: [http://www.iso.org/iso/country\\_codes.htm](http://www.iso.org/iso/country_codes.htm).
- [W3C] W3C. «Status Code Definitions». En: (). URL: <http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html>.
- [W3C05] W3C. «Large Triple Stores». En: (2005). URL: <http://www.w3.org/wiki/LargeTripleStores>.
- [W3C14] W3C. «The RDF Data Cube Vocabulary». En: (2014). URL: <http://www.w3.org/TR/2014/REC-vocab-data-cube-20140116/>.