

GRAU EN ENGINYERIA MULTIMÈDIA



**VNIVERSITAT
DE VALÈNCIA**

TREBALL DE FI DE GRAU

**BUSCADOR WEB DE ESPACIOS Y EDIFICIOS EN LA
ESCOLA TÈCNICA SUPERIOR D'ENGINYERIA**

AUTOR:

ADRIÁN BUSTOS PAZ

TUTORA:

INMACULADA COMA TATAY

SEPTIEMBRE, 2016



VNIVERSITAT
DE VALÈNCIA



Escola Tècnica Superior
d'Enginyeria ETSE-UV

GRAU EN ENGINYERIA MULTIMÈDIA

TREBALL DE FI DE GRAU

BUSCADOR WEB DE ESPACIOS Y EDIFICIOS EN LA ESCOLA TÈCNICA SUPERIOR D'ENGINYERIA

AUTOR:

ADRIÁN BUSTOS PAZ

TUTORA:

INMACULADA COMA TATAY

TRIBUNAL:

PRESIDENT:

VOCAL 1:

VOCAL 2:

DATA DE DEFENSA:

QUALIFICACIÓ:

**A mis abuelos, Fernando e Isabel, por dármelo todo siempre.
A mis padres, por todo su apoyo y por inculcarme la cultura del esfuerzo.
A mi pareja Cristina, porque nada de esto habría sido posible sin ti.**

Índice de Contenidos

1	Introducción	1
2	Motivaciones y Objetivos	3
3	Estado del Arte	5
3.1	Tecnologías de la Web	5
3.1.1	HTML	5
3.1.2	CSS	6
3.2	Lenguajes del lado del Servidor	6
3.2.1	PHP	7
3.2.2	ASP	7
3.2.3	PERL	8
3.2.4	Visual Basic.NET	9
3.2.5	Java	9
3.2.6	Elección de la tecnología a utilizar en el sistema	11
3.3	Servicios Web	11
3.3.1	SOAP	12
3.3.2	REST	13
3.3.3	WDSL	13
3.3.4	Elección de la tecnología a utilizar en el sistema	14
3.4	Lenguajes en el lado del cliente	14
3.4.1	JavaScript	15
3.4.2	VBScript	15
3.4.3	Elección de la tecnología a utilizar en el sistema	16
3.5	Servidores de aplicaciones Java	18
3.5.1	Oracle WebLogic	19
3.5.2	Apache Tomcat	19
3.5.3	WildFly	20
3.5.4	GlassFish	21
3.5.5	Elección de la tecnología a utilizar en el sistema	21

3.6	Base de Datos	22
3.6.1	Microsoft SQL Server	24
3.6.2	Oracle	25
3.6.3	MySQL	26
3.6.4	Elección de la tecnología a utilizar en el sistema	27
3.7	Comunicación de la base de datos con la aplicación	28
3.8	Gráficos vectoriales	29
3.8.1	SVG	30
3.8.2	SWF (Flash)	31
3.8.3	Elección de la tecnología a utilizar en el sistema	31
3.9	Resumen elección de tecnologías	33
3.10	Aplicaciones similares	33
3.10.1	Universidad de Valencia	34
3.10.2	Universidad Carlos III de Madrid	35
3.10.3	University of Washington	36
3.10.4	University of Ottawa	37
4	Planificación y Metodología	39
4.1	Planificación y gestión del proyecto	39
4.2	Análisis de requisitos	39
4.2.1	Funcionales	39
4.2.2	No funcionales	41
4.3	Diagrama de hitos	42
4.4	Diagrama de Gantt	43
4.5	Estimación de costes y estudio de viabilidad	43
4.5.1	El método Juicio de expertos	43
4.5.2	Estimación de costes de personal	51
4.5.3	Estimación de recursos Software/Hardware	52
4.5.4	Estudio de viabilidad económica y legal	53
5	Desarrollo	55
5.1	Análisis del Sistema	55
5.1.1	Actores del sistema	55

5.1.2	Requisitos del usuario	55
5.1.3	Diagrama de secuencia	60
5.1.4	Modelo de datos	66
5.1.5	Modelo de procesos	71
5.2	Diseño del Sistema	75
5.2.1	Diseño de datos	75
5.2.2	Diseño de servicios en el sistema Servidor	91
5.2.3	Diseño del interfaz	93
5.3	Implementación del sistema	99
5.3.1	Fotografías 360 grados	99
5.3.2	Creación de planos de l'Escola Tècnica Superior d'Enginyeria*	100
5.3.3	Implementación de la base de datos	102
5.3.4	Implementación del sistema servidor	105
5.3.5	Implementación del sistema cliente	111
5.3.6	Organización del sitio web	124
6	Pruebas	129
6.1	Correcto funcionamiento del sistema	129
6.1.1	Página principal	129
6.1.2	Buscar un recurso	130
6.1.3	Configurar capas del mapa	135
6.1.4	Mostrar GPS	136
6.1.5	Mostrar Leyenda	136
6.1.6	Mostrar Ayuda	137
6.1.7	Listado de Edificios	138
6.1.8	Resultado de Recurso Profesor	138
6.1.9	Resultado de Recurso Espacio	140
6.1.10	Compartir Recurso	141
6.1.11	Visualizar Panoramas de 360 grados	143
6.2	Rendimiento del sistema	144
6.2.1	Pruebas Usuarios	144

6.2.2	Pruebas Cliente _____	151
6.2.3	Pruebas Servidor _____	157
7	Conclusiones _____	163
8	Trabajos futuros _____	165
9	Bibliografía _____	167

Índice de Figuras

<i>Figura 3.2.1-1: Arquitectura PHP</i>	7
<i>Figura 3.2.2-1: Petición ASP</i>	8
<i>Figura 3.3.1-1: Arquitectura SOAP</i>	12
<i>Figura 3.3.2-1: Petición REST</i>	13
<i>Figura 3.8.1-1: Comparación entre escalado matricial y vectorial</i>	31
<i>Figura 3.10.1-1: Buscador Universitat de València</i>	34
<i>Figura 3.10.2-1: Visor fotografías 360 grados Universidad Carlos III de Madrid</i>	35
<i>Figura 3.10.3-1: Mapas University of Washington</i>	36
<i>Figura 3.10.4-1: Mapas University of Ottawa</i>	37
<i>Figura 4.2.2-1: Diagrama de hitos</i>	42
<i>Figura 4.2.2-1: Diagrama de Gantt</i>	43
<i>Figura 4.5.1-1: Estimación Inmaculada Coma Tatay</i>	45
<i>Figura 4.5.1-2: Estimación Francisco Grimaldo Moreno</i>	46
<i>Figura 4.5.1-3: Estimación José Javier Samper Zapater</i>	47
<i>Figura 4.5.1-4: Estimación Christian Sánchez León</i>	48
<i>Figura 4.5.1-5: Estimación Oscar Abella Abella</i>	49
<i>Figura 4.5.1-6: Estimación resultado</i>	51
<i>Figura 4.5.3-1: Estimación de costes de herramientas software utilizados</i>	52
<i>Figura 4.5.3-2: Estimación de costes de herramientas hardware utilizados</i>	52
<i>Figura 4.5.4-1: Estimación económica del sistema</i>	53
<i>Figura 5.1.2-1: Caso de uso acceso al localizador</i>	56
<i>Figura 5.1.2-2: Caso de uso configurar mapa</i>	56
<i>Figura 5.1.2-3: Caso de uso listar edificios</i>	57
<i>Figura 5.1.2-4: Caso de uso posición GPS</i>	57
<i>Figura 5.1.2-5: Caso de uso acceso a la ayuda</i>	57
<i>Figura 5.1.2-6: Caso de uso interacción mapa</i>	58
<i>Figura 5.1.2-7: Caso de uso mostrar resultados</i>	59
<i>Figura 5.1.2-8: Caso de uso visor imágenes 360 grados</i>	59
<i>Figura 5.1.3-1: Diagrama de secuencia del buscador de profesores</i>	60
<i>Figura 5.1.3-2: Diagrama de secuencia del buscador de profesores por asignaturas</i>	61
<i>Figura 5.1.3-3: Diagrama de secuencia de buscador de espacios</i>	62
<i>Figura 5.1.3-4: Diagrama secuencia visualizar panorámica de 360 grados</i>	63
<i>Figura 5.1.3-5: Diagrama de secuencia de compartir en redes sociales</i>	64
<i>Figura 5.1.3-6: Diagrama secuencia configurar capas del mapa</i>	65
<i>Figura 5.1.3-7: Diagrama de secuencia buscar edificio</i>	66
<i>Figura 5.1.5-1: Diagrama de flujo de datos entrada y salida del sistema</i>	71
<i>Figura 5.1.5-2: Diagrama de flujo de datos carga de página</i>	72
<i>Figura 5.1.5-3: Diagrama de flujo de datos usuario-formulario</i>	72
<i>Figura 5.1.5-4: Diagrama de flujo de datos búsqueda y muestra de resultados</i>	73
<i>Figura 5.1.5-5: Diagrama de flujo de datos de acceso a otros recursos</i>	73
<i>Figura 5.2.1-1: Diagrama Relacional</i>	75
<i>Figura 5.2.1-2: Tabla Coordenadas</i>	75
<i>Figura 5.2.1-3: Tabla Profesores</i>	76
<i>Figura 5.2.1-4: Tabla Asignaturas</i>	77
<i>Figura 5.2.1-5: Tabla Profesorasignatura</i>	77
<i>Figura 5.2.1-6: Tabla Espacios</i>	78
<i>Figura 5.2.1-7: Patrón de IdEspacio</i>	78
<i>Figura 5.2.1-8: Tabla Edificios</i>	79
<i>Figura 5.2.1-9: Tabla Panorama</i>	80
<i>Figura 5.2.1-10: Diagrama clases del Sistema Servidor</i>	80
<i>Figura 5.2.1-11: Clase Coordenadas</i>	81
<i>Figura 5.2.1-12:Clase Edificios</i>	82
<i>Figura 5.2.1-13:Clase Espacios</i>	83
<i>Figura 5.2.1-14: Clase Panoramás</i>	84
<i>Figura 5.2.1-15: Clase Profesores</i>	84

<i>Figura 5.2.1-16: Clase Asignaturas</i>	85
<i>Figura 5.2.1-17: Clase Profesoraasignatura</i>	85
<i>Figura 5.2.1-18: Diagrama de datos del sistema Cliente</i>	86
<i>Figura 5.2.3-1: Wireframe página principal</i>	94
<i>Figura 5.2.3-2: Wireframe buscador</i>	95
<i>Figura 5.2.3-3: Wireframe de configuración</i>	95
<i>Figura 5.3.1-1: Esquema de panoramas</i>	100
<i>Figura 5.3.2-1: Plano Básico</i>	101
<i>Figura 5.3.2-2: Plano categorizado</i>	101
<i>Figura 5.3.2-3: Leyenda Categorías</i>	101
<i>Figura 5.3.5-1: Pegado de textura panorama 360 a objeto 3D esférico</i>	121
<i>Figura 6.1.1-1: Página inicial de la aplicación web</i>	129
<i>Figura 6.1.2-1: Módulo buscador</i>	130
<i>Figura 6.1.2-2: Sugerencia de búsqueda para Espacios y Profesores</i>	131
<i>Figura 6.1.2-3: Sugerencia de búsqueda para asignaturas</i>	132
<i>Figura 6.1.2-4: Sugerencia de recurso inaccesible</i>	132
<i>Figura 6.1.2-5: Sugerencia de recurso espacio</i>	133
<i>Figura 6.1.2-6: Sugerencia de búsqueda para Todo</i>	134
<i>Figura 6.1.3-1: Configurador de capas</i>	135
<i>Figura 6.1.4-1: Posición GPS</i>	136
<i>Figura 6.1.5-1: Leyenda</i>	137
<i>Figura 6.1.6-1: Ayuda</i>	137
<i>Figura 6.1.7-1: Listado de edificios</i>	138
<i>Figura 6.1.8-1: Información Resultado de Profesor</i>	139
<i>Figura 6.1.9-1: Información Resultado de Espacio</i>	140
<i>Figura 6.1.10-1: Compartir en Twitter</i>	141
<i>Figura 6.1.10-2: Compartir en Facebook</i>	142
<i>Figura 6.1.10-3: Compartir Enlace</i>	142
<i>Figura 6.1.11-1: Visor de Panoramas</i>	143
<i>Figura 6.2.1-1: Gráfico dificultad Tarea 1</i>	145
<i>Figura 6.2.1-2: Gráfico dificultad Tarea 2</i>	146
<i>Figura 6.2.1-3: Gráfico dificultad Tarea 3</i>	146
<i>Figura 6.2.1-4: Gráfico dificultad Tarea 4</i>	147
<i>Figura 6.2.1-5: Gráfico dificultad Tarea 5</i>	148
<i>Figura 6.2.1-6: Gráfico dificultad Tarea 6</i>	149
<i>Figura 6.2.1-7: Gráfico dificultad Tarea 7</i>	150
<i>Figura 6.2.1-8: Gráfico dificultad Tarea 8</i>	151
<i>Figura 6.2.2-1: Resultado primer test del sistema cliente con YSlow</i>	152
<i>Figura 6.2.2-2: Resultado primer test del sistema cliente con Webpagetest.org</i>	152
<i>Figura 6.2.2-3: Resultado primer test detalle del sistema cliente con Webpagetest.org</i>	153
<i>Figura 6.2.2-4: Resultado segundo test del sistema cliente con YSlow</i>	155
<i>Figura 6.2.2-5: Resultado segundo test del sistema cliente con Webpagetest.org</i>	156
<i>Figura 6.2.2-6: Resultado segundo test detalle del sistema cliente con Webpagetest.org</i>	156
<i>Figura 6.2.3-1: Tiempo de espera del usuario con 10 usuarios en el Sistema</i>	157
<i>Figura 6.2.3-2: Gráfico Rendimiento Memoria RAM y Carga CPU con 10 usuarios durante 1 minuto</i>	158
<i>Figura 6.2.3-3: Tiempo de espera del usuario con 25 usuarios en el Sistema</i>	158
<i>Figura 6.2.3-4: Gráfico Rendimiento Memoria RAM y Carga CPU con 25 usuarios durante 1 minuto</i>	159
<i>Figura 6.2.3-5: Tiempo de espera del usuario con 50 usuarios en el Sistema</i>	160
<i>Figura 6.2.3-6: Gráfico Rendimiento Memoria RAM y Carga CPU con 50 usuarios durante 1 minuto</i>	160
<i>Figura 6.2.3-7: Tiempo de espera del usuario con 75 usuarios en el Sistema</i>	161
<i>Figura 6.2.3-8: Gráfico Rendimiento Memoria RAM y Carga CPU con 75 usuarios durante 1 minuto</i>	162

Resumen

El objetivo de este trabajo fin de grado está centrado en el desarrollo de un servicio web que permita la búsqueda de un recurso, ya sea profesorado, despacho o aula, haciendo uso de un formulario dado y que, tras la búsqueda, proporcione información relacionada con el recurso, así como geoposicionar el espacio dentro de l'Escola Tècnica Superior d'Enginyeria en un mapa personalizado, con la posibilidad de visualizar, de forma interactiva, fotografías de 360 grados del mismo. Para poder lograr dicho objetivo, hemos llevado a cabo una recopilación y evaluación de las tecnologías que se encuentran disponibles en la actualidad, priorizando aquellas que son Open Source, y realizando una planificación de las diversas etapas del desarrollo del proyecto. El marco teórico se encuentra estructurado en varios apartados, dónde se abordan las siguientes temáticas: Análisis del Sistema, Diseño del Sistema, Implementación del sistema y Pruebas. La conclusión de este trabajo se basa en un conjunto de reflexiones finales, así como una serie de propuestas de desarrollos futuros a raíz del proyecto presentado.

Palabras Clave: Servicio Web, Geo posicionamiento, Mapa, Fotografías de 360 grados.

Abstract

The aim of this final project is focused on the development of a web service that allows finding a resource, whether teachers, office or classroom, using a form. After the search, provide information relating to the resource and geoposition space within the *Escola Tècnica Superior d'Enginyeria* on a personalized map, with the ability to view, interactively, 360-degree photographs of it. To achieve this objective, we have prepared a compilation and evaluation of technologies that are available today, giving priority to Open Source, planning and performing the various stages of project development. The theoretical framework is structured into several sections, where the following topics are: system analysis, system design, system implementation and testing. The conclusion of this work is based on a set of final thoughts as well as a number of proposals for future developments from this project.

Keywords: Web Service, Geoposition, Map, 360-degree photography.



Introducción

1 Introducción

En la actualidad existen numerosas Aplicaciones Web en la red global. Las hay de diversos tipos y enfocadas a distintas finalidades y cada vez son más.

Para llevar a cabo este proyecto de una forma ordenada estableceremos un proceso de planificación. Antes de comenzar, se deberá tener una idea clara de las tareas a desarrollar y establecer una serie de pautas a seguir. La planificación es un punto clave a fin de lograr éxito a la hora de cubrir plazos de realización, presupuesto y entrega ante el cliente.

Lo primero que haremos será recopilar la información necesaria referente al personal de la ETSE, asignaturas impartidas, planos correspondientes a las diversas plantas del edificio, las estancias disponibles del edificio y las coordenadas geoposicionadas de las mismas, para posteriormente incluirlos en una base de datos y acceder a los recursos desde la aplicación creada.

Una vez recopilada toda la información, pasaremos a analizar las diferentes opciones que existen en el mercado para construir nuestra aplicación Web, valorando sus ventajas e inconvenientes, así como las opciones más óptimas a lo que deseamos hacer.

Concretadas las tecnologías que se vayan a emplear y antes de comenzar la construcción del proyecto, deberemos analizar si es viable desde el punto de vista temporal y presupuestal sin perder en ningún momento el aspecto legal. Para ello realizaremos una planificación realista que tenga en cuenta los aspectos de nuestra aplicación.

Llegados a este punto, pasaremos a analizar cómo va a ser nuestro buscador y qué elementos va a necesitar desde un punto de vista conceptual.

Establecida la forma que tendrá nuestro sistema pasaremos a hacer un diseño preliminar del sistema sin llegar a implementar las rutinas o funciones.

Cuando tengamos claro cómo se va a realizar el sistema pasaremos a implementarlo. Por una parte, introduciremos la información obtenida al principio en una base de datos y, por otra parte, crearemos la aplicación que accederá a la base de datos y mostrará su información al usuario que la solicita.

Por último, realizaremos una serie de pruebas a nuestra Aplicación para verificar su correcto funcionamiento y evaluaremos cuales han sido los resultados.



Motivaciones y Objetivos

2 Motivaciones y Objetivos

Desde que se comienza la universidad, incluso tiempo antes de formalizar la matrícula del primer curso de grado, muchas son las dudas que asaltan al futuro estudiante. Los horarios, las asignaturas o los laboratorios ocupan gran parte de las preocupaciones de los alumnos. Al superar el primer cuatrimestre, viene una nueva etapa que conlleva nuevos horarios, asignaturas y profesores. Conforme se van superando las distintas etapas, muchas son las actividades, tutorías, laboratorios y eventos que el estudiante realizará durante su periodo académico. Todos estos elementos tendrán un factor en común, conocer y visitar multitud de espacios en su vida universitaria.

Tras varios años dedicados a la universidad compaginando estudios académicos y actividades de participación (IEEE rama estudiantil de la Universitat de València, diversos cargos en la jerarquía gubernamental de la escuela y programa 'entreiguals'), no han sido pocos los que he podido observar albergar las mismas dudas una y otra vez. ¿Dónde está el Laboratorio de Física? ¿Dónde está el aula 1.2.20? ¿Dónde está el despacho del profesor de Informática?

De igual manera, muchas asignaturas son impartidas por profesores no pertenecientes a esas misma facultades por lo que deben desplazarse de uno a otro edificio, incluso cambiando de campus, para llegar a una aula o laboratorio del cual no siempre conocen su localización.

Ante este torrente incesante de estas y otras preguntas similares planteé la idea de que la Universitat de València podría estar falta de una herramienta de consulta que pudiera dar solución a cualquier colectivo tanto dentro como fuera de la Universidad de Valencia.

En este caso vamos a tratar de desarrollar un buscador que sea capaz de acometer principalmente dos objetivos. Por una parte, mostrar la máxima información posible sobre el personal que trabaja en un edificio, y por otra, que sea capaz de ubicarlo mediante un mapa dentro del mismo. Debido a que la Universitat de València carece de una solución integral de estas características y viendo lo útil que puede resultar a alumnos a encontrar aulas, a profesores de otros edificios a encontrar su docencia o a nuevos alumnos a conocer sus futuras instalaciones cuando accedan a la universidad, hemos decidido realizar dicha aplicación sobre la Escola Tècnica Superior d'Enginyeria (ETSE).

La ETSE está ubicado en el Campus de Burjassot-Paterna en su zona oeste, a medio camino entre los edificios de ciencias (Física, Farmacia, Matemáticas, etc.) y los edificios de ciencias aplicadas ubicados en Robótica (IRTIC).

La idea principal es que mediante un formulario lo más sencillo posible, el usuario, conociendo algún dato sobre el profesor, PDI o PAS, pueda obtener la mayor información posible sobre él. De igual manera, existirán formularios alternativos donde el usuario pueda obtener la información solicitada por otras vías como puede ser las asignaturas impartidas por el docente o el despacho de tutorías que emplee el mismo.



3 Estado del Arte

Para la aplicación que vamos a diseñar existen varios aspectos que se deben estudiar con detalle antes de tomar una decisión adecuada sobre la configuración final de los componentes del sistema que se van a utilizar. A continuación veremos las diferentes alternativas disponibles para realizar nuestro sistema y discutiremos porque hemos elegido unas tecnologías y no otras.

3.1 Tecnologías de la Web

Las tecnologías Web sirven para acceder a los recursos de conocimiento disponibles en Internet o en las intranets utilizando un navegador. Están muy extendidas por muchas razones como: facilitar el desarrollo de sistemas de Gestión del Conocimiento (en lo adelante GC), su flexibilidad en términos de escalabilidad, es decir, a la hora de expandir el sistema; su sencillez de uso y que imitan la forma de relacionarse de las personas al poner a disposición de todos el conocimiento de los demás, por encima de jerarquías, barreras formales u otras cuestiones. Estas tecnologías pueden llegar a proporcionar recursos estratégicos, pero, evidentemente, no por la tecnología en sí misma, que está disponible ampliamente, sino por lo fácil que es personalizarla y construir con ella sistemas de GC propietarios de la empresa.

Internet, Intranet o extranet permiten, a los usuarios, el acceso a una gran cantidad de información: leer publicaciones periódicas, buscar referencias en bibliotecas, realizar paseos virtuales por museos, compras electrónicas y otras muchas funciones. Gracias a la forma en que está organizada la World Wide Web (WWW), los usuarios pueden saltar de un recurso a otro con facilidad.

3.1.1 HTML

HTML, sigla en inglés de HyperText Markup Language (lenguaje de marcas de hipertexto), hace referencia al lenguaje de marcado para la elaboración de páginas web. Es un estándar que sirve de referencia del software que conecta con la elaboración de páginas web en sus diferentes versiones, define una estructura básica y un código, denominado código HTML, para la definición de contenido de una página web, como texto, imágenes, videos, juegos, entre otros.

Es un estándar a cargo del World Wide Web Consortium (W3C) o Consorcio WWW, organización dedicada a la estandarización de casi todas las tecnologías ligadas a la web, sobre todo en lo referente a su escritura e interpretación. Se considera el lenguaje web más importante siendo su invención crucial en la aparición, desarrollo y expansión de la World Wide Web (WWW). Es el estándar que se ha impuesto en la visualización de páginas web y es el que todos los navegadores actuales han adoptado.



Servicio Web de Información geoposicionada

Actualmente se encuentra en su versión 5 que introduce nuevas características:

- Etiquetas con significado semántico para su empleo en la Web 3.0
- Etiquetas con capacidad de gráficos 2D, 3D, audio y video, con códec para contenido multimedia.
- Nuevas APIs que amplían funcionalidades (Geolocalización, Websockets, Drag and Drop, etc).

3.1.2 CSS

Hoja de estilo en cascada o CSS (siglas en inglés de cascading style sheets) es un lenguaje usado para definir y crear la presentación de un documento estructurado escrito en HTML o XML y, por extensión, en XHTML. El World Wide Web Consortium (W3C) es el encargado de formular la especificación de las hojas de estilo que servirán de estándar para los agentes de usuario o navegadores.

La idea que se encuentra detrás del desarrollo de CSS es separar la estructura de un documento de su presentación, ya sea mediante la definición de los estilos en el propio documento HTML o con la inclusión de los mismos en documentos separados.

Actualmente se encuentra en su versión 2.1, aunque no todos sus módulos se encuentran en fase de recomendación por la W3C, únicamente los “selectores”, “espacios de nombres” y “color”.

3.2 Lenguajes del lado del Servidor

La Programación del lado del servidor es una tecnología que consiste en el procesamiento de una petición de un usuario mediante la interpretación de un script en el servidor web para generar páginas HTML dinámicamente como respuesta o cuando, por razones de seguridad, los cálculos no se pueden realizar en la computadora del usuario. Todo lo que suceda dentro del servidor es llamado procesamiento del lado del servidor.

Cuando la aplicación cliente necesita interactuar con el servidor, ésta ejecuta una aplicación en el procesador local: el denominado programa cliente. Este programa cliente se encarga de ponerse en contacto con el procesador remoto para solicitar el servicio deseado. El procesador remoto por su parte responderá a lo solicitado mediante un programa que está ejecutando. Este último se denomina programa servidor.

La utilización de las diferentes aplicaciones o servicios de Internet se lleva a cabo respondiendo al llamado modelo cliente-servidor.

A continuación expondremos brevemente una descripción de algunos de estos lenguajes orientados a objetos:

Estado del Arte

3.2.1 PHP

PHP es un lenguaje sencillo de scripts en servidor con sintaxis muy similar a C, Java y Perl, con solamente algunas características específicas. Es un lenguaje open-source cuya meta es permitir escribir a los creadores de páginas Web páginas dinámicas de manera rápida y fácil. Desde sus inicios fue un lenguaje con mucha aceptación para el desarrollo de aplicaciones Web de medianas dimensiones y, actualmente, la gran parte de los sitios Web en el mundo están escritos en este lenguaje con servidor Apache como servidor Web. Entre sus características, destacamos el control transparente de sesiones de usuario, soporte con XML, Java y modelos DOM COM. La interfaz con el servidor Web ha sido completamente abstraída, permitiendo compatibilidad con otros servidores Web diferentes de Apache. Es un lenguaje multiplataforma, de manera que se puede utilizar en casi cualquier sistema operativo, se integra con 8 y tiene accesos a 20 bases de datos con funciones específicas haciéndolo bastante eficiente. Además permite cualquier acceso a base de datos por medio de ODBC.

Otra de las ventajas que ofrece es la gran cantidad de software y código fuente de ejemplos que, de acuerdo con la filosofía open-source, existe para los desarrolladores. Uno de los aspectos negativos que ofrece este lenguaje de programación es que exige mayor conocimiento de programación que otros entornos visuales.

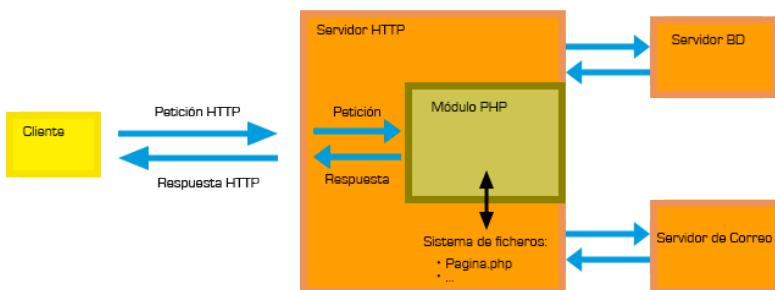


Figura 3.2.1-1: Arquitectura PHP

3.2.2 ASP

Lenguaje comercializado por Microsoft y usado por programadores para desarrollar, entre otras funciones, sitios web. ASP.NET es el sucesor de la tecnología ASP que fue lanzada al mercado mediante una estrategia denominada .NET.

Fue desarrollado para resolver las limitaciones que brindaba su antecesor, ASP. Para el desarrollo de ASP.NET se puede utilizar C#, VB.NET o J#. Los archivos cuentan con la extensión (aspx). Para el funcionamiento de las páginas se necesita tener instalado IIS con el Framework .Net. Microsoft Windows 2003 incluye este framework, solo se necesitará instalarlo en versiones anteriores.

El lenguaje ASP consiste en una serie de clases .NET utilizadas para crear aplicaciones Web, tanto del lado cliente (Web Form) como del lado servidor (Web Service). La integración de nativa .NET Framework con el sistema operativo Windows Server 2003 hace que su ejecución sea más estable y rápida que otros lenguajes de programación. Dispone Actualizaciones Dinámicas, Soporte de servicios web XML y Conexiones del tipo DSN, o sin utilización de DSN, para acceder a fuentes de datos ODBC.



Servicio Web de Información geoposicionada

Las páginas creadas con la tecnología ASP.NET funcionan en todo tipo de navegadores, incluyendo Netscape, Safari e Internet Explorer.

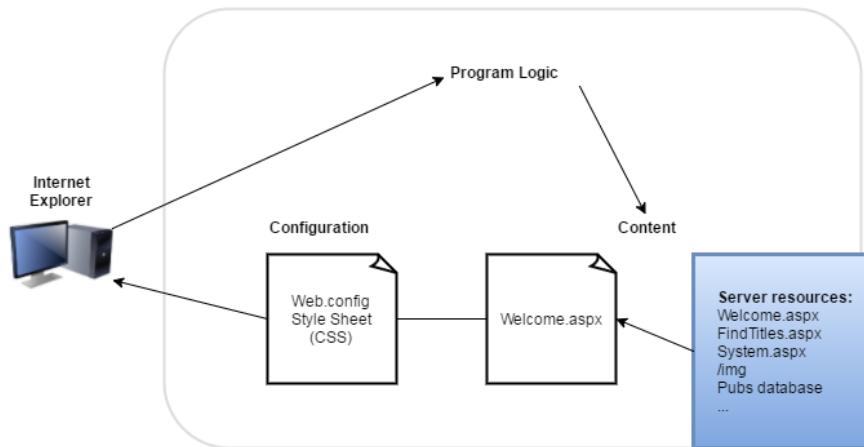


Figura 3.2.2-1: Petición ASP

3.2.3 PERL

Es un lenguaje de programación muy utilizado para construir aplicaciones CGI para el web. Perl es un acrónimo de “Practical Extracting and Reporting Languaje”, que viene a indicar que se trata de un lenguaje de programación muy práctico para extraer información de archivos de texto y generar informes a partir del contenido de los ficheros.

Es un lenguaje libre de uso, es decir, es gratuito. Antes estaba muy asociado a la plataforma Unix, pero, en la actualidad, está disponible en otros sistemas operativos, como Windows.

Perl es un lenguaje de programación interpretado, al igual que muchos otros lenguajes de Internet como JavaScript o ASP. Esto quiere decir que el código de los scripts en Perl no se compila sino que, cada vez que se quiere ejecutar, se lee el código y se pone en marcha interpretando lo que hay escrito. Además, es extensible a partir de otros lenguajes, ya que desde Perl podremos hacer llamadas a subprogramas escritos en otros lenguajes. También desde otros lenguajes podremos ejecutar código Perl.

Perl está inspirado a partir de lenguajes como C, sh, awk y sed (algunos provenientes de los sistemas Unix), pero está enfocado a ser más práctico y fácil que estos últimos. Es por ello que un programador que haya trabajado con los lenguajes anteriormente nombrados tendrá menos problemas en entenderlo y utilizarlo rápidamente. Una diferencia fundamental de Perl con respecto a los otros lenguajes es que no limita el tamaño de los datos con los que trabaja, el límite lo pone la memoria que en ese momento se encuentre disponible.

Si queremos trabajar con Perl será necesario tener instalado el intérprete del lenguaje. A partir de ese momento podemos ejecutar CGIs en nuestros servidores web. El proceso para conseguirlo puede variar de unos servidores a otros, pero se suelen colocar en un directorio especial del servidor llamado cgi-bin donde hemos colocado los correspondientes permisos CGI. Además, los archivos con el código también deberán tener permiso de ejecución.



Estado del Arte

3.2.4 Visual Basic.NET

Es una versión de Visual Basic enfocada al desarrollo de aplicaciones .NET. El lenguaje de programación es Visual Basic, que apareció el año 1991 como una evolución del QuickBasic que fabricaba Microsoft.

Es un lenguaje de programación orientado a objetos y, como novedades más importantes en la versión .Net, podemos citar la posibilidad de definir ámbitos de tipo, clases que pueden derivarse de otras mediante herencia, sobrecarga de métodos, nuevo control estructurado de excepciones o la creación de aplicaciones con múltiples hilos de ejecución.

Otras de sus características más importantes son:

- Diseño de controles de usuario para aplicaciones Windows y Web.
- Programación de bibliotecas de clase.
- Envío de datos vía documentos XML.
- Generación de reportes basados en Crystal Reports a partir de información obtenida de orígenes de datos (archivos de texto, bases, etc.).

3.2.5 Java

Es una plataforma de software desarrollada por Sun Microsystems, de tal manera que los programas creados en ella puedan ejecutarse sin cambios en diferentes tipos de arquitecturas y dispositivos computacionales.

El lenguaje mismo se inspira en la sintaxis de C++, pero su funcionamiento es más similar al de Smalltalk. Incorpora sincronización y manejo de tareas en el lenguaje mismo, similar a Ada, e incorpora interfaces como un mecanismo alternativo a la herencia múltiple de C++.

A finales del siglo XX, Java llegó a ser el lenguaje de mayor acogida para programas de servidor. Utilizando una tecnología llamada JSP, similar a otras tecnologías del lado del servidor como ASP de Microsoft o PHP; se hizo muy fácil escribir páginas dinámicas para sitios de Internet. Sumado a esto, la tecnología de JavaBeans, al incorporarse con JSP, permitía adaptar al mundo Web el patrón MVC (modelo-vista-controlador) que ya se había aplicado con éxito a interfaces gráficas.

Los programas en Java generalmente son compilados a un lenguaje intermedio llamado bytecode, que luego son interpretados por una máquina virtual (JVM). Esta última sirve como una plataforma de abstracción entre la máquina y el lenguaje permitiendo que se pueda “escribir el programa una vez, y correrlo en cualquier lado”. También existen compiladores nativos de Java, tanto software libre como no libre. El compilador GCC de GNU compila Java a código de máquina con algunas limitaciones al año 2002.

La plataforma Java consta de las siguientes partes:

- El lenguaje de programación.
- La máquina virtual de Java o JRE, que permite la portabilidad en ejecución.



Servicio Web de Información geoposicionada

- El API Java, una interfaz de programación de aplicaciones que da a los programadores un ambiente de desarrollo completo, así como una infraestructura. Como el lenguaje Java es un lenguaje orientado a objetos, la API de Java provee de un conjunto de clases utilitarias para efectuar toda clase de tareas necesarias dentro de un programa. Esta API está organizada en paquetes, donde cada paquete contiene un conjunto de clases relacionadas semánticamente.
- Sun, dispuesto a proporcionar las herramientas necesarias para cubrir las necesidades de todos los usuarios, creó distintas versiones de Java de acuerdo a las necesidades de cada uno. Según esto nos encontramos con que el paquete Java 2 lo podemos dividir en 3 ediciones distintas. J2SE (Java Standard Edition), orientada al desarrollo de aplicaciones independientes de la plataforma; J2EE (Java Enterprise Edition), orientada al entorno empresarial; y J2ME (Java Micro Edition), orientada a dispositivos con capacidades restringidas. Veamos cuáles son las características de cada una de las versiones:

Java 2 Platform, Standard Edition (J2SE). Esta edición de Java es la que, en cierta forma, recoge la iniciativa original del lenguaje Java. Tiene las siguientes características:

- Inspirado inicialmente en C++, pero con componentes de alto nivel como soporte nativo de strings y recolector de basura.
- Código independiente de la plataforma, precompilado a bytecodes intermedio y ejecutado en el cliente por una JVM (Java Virtual Machine).
- Modelo de seguridad tipo sandbox proporcionado por la JVM.
- Abstracción del sistema operativo subyacente mediante un juego completo de APIs de programación.

Esta versión de Java contiene el conjunto básico de herramientas usadas para desarrollar Java Applets, así como las APIs orientadas a la programación de aplicaciones de usuario final: Interfaz gráfica de usuario, multimedia, redes de comunicación, etc.

Java 2 Platform, Enterprise Edition (J2EE). Esta versión está orientada al entorno empresarial. El software empresarial tiene unas características propias marcadas: está pensado no para ser ejecutado en un equipo, sino para ejecutarse sobre una red de ordenadores de manera distribuida y remota mediante EJB (Enterprise Java Beans). De hecho, el sistema se monta sobre varias unidades o aplicaciones. En muchos casos, además, el software empresarial requiere que sea capaz de integrar datos provenientes de entornos heterogéneos. Esta edición está orientada, especialmente, al desarrollo de servicios web, servicios de nombres, persistencia de objetos, XML, autenticación, APIs para la gestión de transacciones, etc. El cometido de esta especificación es ampliar la J2SE para dar soporte a los requisitos de las aplicaciones de empresa.

Java 2 Platform, Micro Edition (J2ME). Esta versión de Java está enfocada a la aplicación de la tecnología Java en dispositivos electrónicos con capacidades computacionales y gráficas muy reducidas, tales como teléfonos móviles, PDAs o electrodomésticos inteligentes. Esta edición tiene unos componentes básicos que la diferencian de las otras versiones, como el uso de una máquina virtual denominada KVM (Kilo Virtual Machine), debido a que requiere sólo unos pocos Kilobytes de memoria para funcionar, en vez del



Estado del Arte

uso de la JVM clásica; inclusión de un pequeño y rápido recolector de basura y otras diferencias que ya iremos viendo más adelante.

3.2.6 Elección de la tecnología a utilizar en el sistema

La elección del lenguaje orientado a objetos que vamos a emplear en nuestro sistema por parte del servidor es Java.

Para esta elección hemos tenido en cuenta las siguientes características sobre el lenguaje Java:

Simplicidad: Java ofrece la funcionalidad de un lenguaje potente sin las características menos usadas o más confusas de estos. C++ es un lenguaje que, aunque potente y flexible, adolece de falta de seguridad.

Es orientado a objetos: Java implementa la funcionalidad básica de C++ con varias mejoras y eliminando elementos para mantener la filosofía de simplicidad del lenguaje. Java trabaja con sus datos como objetos y con interfaces a los mismos. Soporta las tres características propias del paradigma de la orientación a objetos: encapsulación, herencia y polimorfismo.

Es distribuido: Java se ha construido con extensas capacidades de interconexión TCP/IP. Existen librerías de rutinas para acceder e interactuar con protocolos como http y ftp. Esto permite a los programadores acceder a la información a través de la red con tanta facilidad como a los ficheros locales. Esta es una de las características más importantes relacionadas con nuestro sistema, ya que se basa en gran medida en la programación en red.

Es robusto: Java realiza verificaciones en busca de problemas, tanto en tiempo de compilación como en tiempo de ejecución. La comprobación de tipos en Java ayuda a detectar errores lo antes posible en el ciclo de desarrollo. Java obliga a la declaración explícita de métodos, reduciendo así las posibilidades de error. Maneja la memoria para eliminar las preocupaciones por parte del programador de la liberación o corrupción de memoria.

Como hemos comentado anteriormente, Java posee diferentes versiones dependiendo de las necesidades del usuario. Dado que nuestras necesidades están enfocadas a servicios Web con funciones empresariales en nuestro servidor Web, utilizaremos la última versión del J2EE, que es la 8.0.

3.3 Servicios Web

Un Servicio Web es un componente al que podemos acceder mediante protocolos Web estándar, utilizando lenguajes XML, JSON u otros para el intercambio de información.

Cuando hablamos de servicios web, hablamos de una colección de métodos a los que podemos llamar desde cualquier lugar de Internet o intranet, siendo estos procedimientos de invocación totalmente independientes de la plataforma que



Servicio Web de Información geoposicionada

utilicemos y del lenguaje de programación en el que se haya implementado internamente el servicio.

Cuando conectamos con un servidor web desde nuestro navegador, el servidor nos devuelve la página web solicitada, que es un documento que se mostrará en el navegador para que lo visualice el usuario, pero es difícilmente entendible por una máquina. Podemos ver esto como una web para humanos. En contraposición, los Servicios Web ofrecen información con un formato estándar que puede ser entendido fácilmente por una aplicación. En este caso estaríamos ante una web para máquinas.

Los servicios Web son componentes de aplicaciones distribuidas que están disponibles de forma externa. Se pueden utilizar para integrar aplicaciones escritas en diferentes lenguajes y que se ejecutan en plataformas diferentes. Los servicios Web son independientes de lenguaje y de la plataforma gracias a que los vendedores han admitido estándares comunes de Servicios Web.

Como servicios web disponemos de las siguientes tecnologías más habituales:

3.3.1 SOAP

Es un tipo de arquitectura web que usa un paradigma de mensajería de una dirección sin estado, que puede ser utilizado para formar protocolos más complejos y completos según las necesidades de las aplicaciones que lo implementan. Puede formar y construir la capa base de una "pila de protocolos de web service", ofreciendo un framework de mensajería básica en el cual los web services se pueden construir. Este protocolo está basado en XML y se conforma de tres partes:

- Sobre (envelope): el cual define qué hay en el mensaje y cómo procesarlo.
- Conjunto de reglas de codificación para expresar instancias de tipos de datos.
- La Convención para representar llamadas a procedimientos y respuestas.

El protocolo SOAP tiene tres características principales:

- Extensibilidad (seguridad y WS-routing son extensiones aplicadas en el desarrollo).
- Neutralidad (SOAP puede ser utilizado sobre cualquier protocolo de transporte como HTTP, SMTP, TCP o JMS).
- Independencia (SOAP permite cualquier modelo de programación).

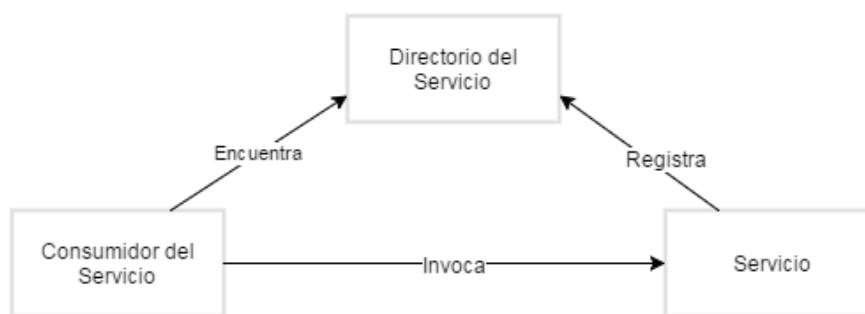


Figura 3.3.1-1: Arquitectura SOAP

Estado del Arte

3.3.2 REST¹

Es un tipo de arquitectura web que sirve para describir cualquier interfaz entre sistemas que utilice directamente HTTP para obtener datos o indicar la ejecución de operaciones sobre los datos en cualquier formato (XML, JSON, etc), sin las abstracciones adicionales de los protocolos basados en patrones de intercambio de mensajes, como por ejemplo SOAP.

Los sistemas que siguen los principios REST se llaman con frecuencia RESTful y cuentan con las siguientes características:

- Un protocolo sin estado: cada mensaje HTTP contiene toda la información necesaria para comprender la petición. Como resultado, ni el cliente ni el servidor necesitan recordar ningún estado de las comunicaciones entre mensajes. Sin embargo, en la práctica, muchas aplicaciones basadas en HTTP utilizan cookies y otros mecanismos para mantener el estado de la sesión (algunas de estas prácticas, como la reescritura de URLs, no son permitidas por REST)
- Un conjunto de operaciones bien definidas que se aplican a todos los recursos de información: HTTP en sí define un conjunto pequeño de operaciones, las más importantes son POST, GET, PUT y DELETE. Con frecuencia estas operaciones se asemejan a las operaciones CRUD en bases de datos que se requieren para la persistencia de datos.
- Una sintaxis universal para identificar los recursos. En un sistema REST, cada recurso es direccionable únicamente a través de su URI.
- El uso de hipermédios, tanto para la información de la aplicación como para las transiciones de estado de la aplicación. La representación de este estado en un sistema REST son típicamente HTML o XML. Como resultado de esto, es posible navegar de un recurso REST a muchos otros, simplemente siguiendo enlaces sin requerir el uso de registros u otra infraestructura adicional.

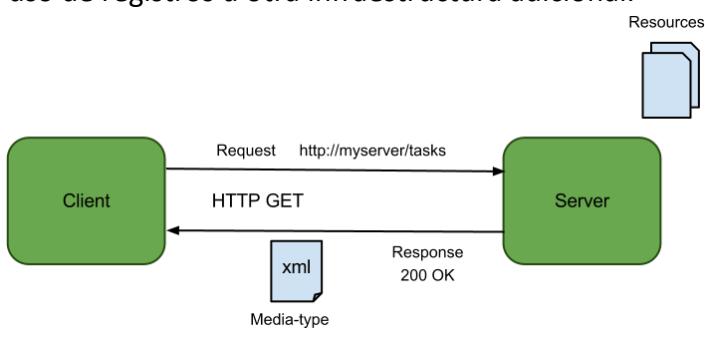


Figura 3.3.2-1: Petición REST

3.3.3 WDSL

Describe la interfaz pública a los servicios Web. Está basado en XML y describe la forma de comunicación, es decir, los requisitos del protocolo y los formatos de los mensajes necesarios para interactuar con los servicios listados en su catálogo. Las operaciones y mensajes que soporta se describen en abstracto y se ligan después al protocolo concreto de red y al formato del mensaje.

¹ Véase *Servicios web RESTful con HTTP. Parte I: Introducción y bases teóricas* en la Bibliografía



Servicio Web de Información geoposicionada

Un programa cliente que se conecta a un servicio web puede leer el WSDL para determinar qué funciones están disponibles en el servidor. Los tipos de datos especiales se incluyen en el archivo WSDL en forma de XML Schema. El cliente puede usar SOAP para hacer la llamada a una de las funciones listadas en el WSDL.

3.3.4 Elección de la tecnología a utilizar en el sistema

La tecnología escogida para la realización del servicio web es REST debido a las siguientes razones:

Posibilidad de utilización de elementos JSON en lugar de XML, debido a la posibilidad de compartir datos en forma de vectores en lugar de en árboles, lo que facilita la integración con otros lenguajes sin necesidad de conversiones.

Se centra en las operaciones basadas en recursos naturales y hereda sus operaciones (GET, PUT, POST, DELETE) de HTTP. Esto hace que sea fácil para los desarrolladores y los navegadores web para consumir los recursos que se sirven.

REST permite fáciles y rápidas llamadas a una dirección URL para las respuestas rápidas de retorno. La diferencia entre SOAP y REST, en este caso, es la complejidad. Los servicios SOAP necesitan mantener una conexión abierta con un cliente complejo mientras que REST realiza peticiones de recursos de forma independiente. Esta característica permite hacer pruebas test mucho más simples y rápidas.

3.4 Lenguajes en el lado del cliente

La utilización de páginas HTML estáticas tiene ciertos inconvenientes:

- Todo acceso a un documento da el mismo resultado.
- No satisface las necesidades de interactividad.
- No adaptan los documentos a clientes individuales.
- Hay problemas para actualizar los documentos, especialmente si parte de los datos están replicados en varios documentos.
- Son imposibles las aplicaciones Web.
- No proporcionan una apropiada gestión de contenidos.

Para solventar estas limitaciones se crearon determinados lenguajes que dotaban de dinamismo a las páginas. Se definieron lenguajes dinámicos en el lado del cliente, se ejecutaban cuando un usuario se descargaba la página, y los lenguajes dinámicos en el lado del servidor, que se generaban en el momento de la petición.

Se produce por la incrustación de determinado código en las páginas HTML con la que el servidor Web responde como petición de una url. El responsable en el lado del cliente de procesar estos programas es el propio navegador. Las ventajas de hacer uso de estos programas son los siguientes:

Se produce menor sobrecarga en la parte del servidor. En muchas ocasiones, nos evitamos múltiples llamadas al servidor para verificar que ciertos campos de los



Estado del Arte

formularios no están llenos. Además, podemos evitarnos ciertos cálculos que se pueden realizar en la parte del cliente.

Como desventajas citaremos:

Es necesario que el navegador tenga un motor de scripts y que soporte el lenguaje en concreto que se está utilizando. También podemos tener problemas con las versiones y navegadores de forma que ciertas instrucciones funcionen correctamente en una versión de un navegador y no función en otros.

Como lenguajes de scripts en el lado del cliente tenemos principalmente:

3.4.1 JavaScript

Es un lenguaje de scripts basados en objetos. Netscape fue el primer navegador que incorporó JavaScript aunque, con posterioridad, también fue incorporado en Internet Explorer.

Los programas realizados en JavaScript se incrustan en documentos HTML tal cual, como si del mismo código se tratase. Como el lenguaje de script no es necesario compilarlo y generar el fichero binario correspondiente, el código es interpretado directamente por el navegador.

Mucha gente conoce JavaScript como una extensión de HTML, ya que permite realizar ciertas acciones como detectar y controlar eventos, por ejemplo: los clics del ratón sobre una determinada zona de la pantalla puede validar formularios, botones con iluminación e incluso permite desarrollar interfaces Web completas. En definitiva, permite realizar múltiples tareas que no serían posibles con etiquetas HTML.

Además de todo esto, otra cosa que caracteriza a JavaScript es su independencia con el servidor Web utilizado. De hecho, ni siquiera es necesario que exista servidor Web para que JavaScript se pueda ejecutar. Basta con crear una página Web con el código HTML correspondiente y visualizar la página en el navegador sin tener que conectarse a Internet.

Sin duda, después de todo lo expuesto, lo mejor de JavaScript es su universalidad, debido a que los navegadores más utilizados lo soportan en sus últimas versiones.

3.4.2 VBScript

El lenguaje de scripts Visual Basic (VBScript) es un lenguaje propio de Microsoft con una sintaxis muy parecida a Visual Basic y Visual Basic for Applications. Básicamente se podría considerar como un subconjunto de las instrucciones que ofrece Visual Basic.

Las páginas HTML que contengan código VBScript sólo podrán visualizarse correctamente en navegadores de Microsoft. Este aspecto limita la universalidad de una aplicación Web. Posiblemente, si tiene alguna ventaja el uso de este lenguaje es su curva de aprendizaje, especialmente si se ha trabajado previamente con VBScript.



3.4.3 Elección de la tecnología a utilizar en el sistema

En este caso la elección no es muy difícil, nos decantaremos por utilizar JavaScript, principalmente por su independencia del servidor Web empleado, su universalidad de soporte en los navegadores más utilizados y por el gran catálogo de librerías disponibles que facilitan enormemente las capacidades de las páginas.

a. Librerías JavaScript

Este lenguaje nos permite realizar multitud de tareas que tengan como objetivo manipular el DOM de nuestra página web. Sin embargo, muchas funcionalidades rutinarias, sencillas como la selección de elementos del DOM o más complejas como la gestión de eventos, requieren de muchas líneas de código para realizar funciones concretas y de amplio uso en desarrollos web. Este modo de programación es conocido como “Vanilla” JavaScript, cuya traducción más aproximada sea JavaScript “a pelo”. Por esta razón, conforme se hizo necesaria la mejora de funcionalidades y esta programación se hacía muy pesada, muchos desarrolladores han creado librerías JavaScript para acometer principalmente 2 objetivos: simplificar la programación de tareas recurrentes o muy extendidas y dotar a nuestros proyectos de características concretas.

Seguidamente pasaremos a enumerar y explicar las bibliotecas que hemos escogido:

– *JQuery*

Librería JavaScript que permite la manipulación de documentos HTML en el DOM y los estilos CSS, la gestión de eventos, efectos y animaciones, AJAX y compatible con todos los navegadores de última generación. Actualmente en versión 3.1.0.

– *Bootstrap*

Librería JavaScript perteneciente al framework *Bootstrap* que nos permitirá manejar los distintos eventos que se produzcan en la página relacionados con responsive y/o representación de los elementos mostrados. Actualmente en versión 3.

– *Mapbox*²³

Librería JavaScript perteneciente a *Mapbox.com* que nos permitirá acceder mediante su API a los estilos y personalización de mapas provenientes de *OpenStreetMap*. Actualmente en versión 2.4.0.

– *Raphael*

Librería JavaScript creada por *Dmitry Baranovskiy* que simplifica el tratamiento y generación de gráficos vectoriales en la web. Actualmente en versión 2.2.1.

– *API GoogleMaps*

Librería JavaScript perteneciente a *Google* que nos proporciona, mediante su API correspondiente, los mapas de su servicio *GoogleMaps*. Actualmente en versión 3.0.

² Véase *Mapbox* en Bibliografía

³ Véase *Survey of the Best Online Mapping Tools* en Bibliografía



Estado del Arte

– *Leaflet*⁴

Librería JavaScript de código abierto empleado en la representación de mapas interactivos y con capacidad responsive. Actualmente en versión 0.7.7.

- [Leaflet.google.js](#): Plugin JavaScript que, en combinación con la librería principal *Leaflet.js* permite la utilización de los mapas de *GoogleMap* en el entorno Leaflet.
- [Leaflet.control.sidebar.js](#): Plugin JavaScript que, en combinación con la librería principal *Leaflet.js* permite la creación y uso de barras laterales en el entorno Leaflet.
- [Leaflet.awesom-markers.js](#): Plugin JavaScript que, en combinación con la librería principal *Leaflet.js* permite el uso de iconos vectoriales *Font awesome* en el entorno Leaflet.
- [Leaflet.easybutton.js](#): Plugin JavaScript que, en combinación con la librería principal *Leaflet.js* permite la creación y uso de botones multipropósito en el entorno Leaflet
- [Leaflet.rlayer.js](#): Plugin JavaScript que, en combinación con la librería principal *Leaflet.js* permite la gestión y uso de la librería *Raphael.js* entorno Leaflet.

– *Handlebars*

Librería de JavaScript basado en *Mustache Templates*, que nos sirve para generar plantillas HTML a partir de objetos con datos en formato JSON.

– *Clipboard*

Sencilla Librería de JavaScript que permite la copia de texto al portapapeles del sistema operativo con independencia del dispositivo.

– *Typeahead*

Librería de JavaScript desarrollada por *Twitter* que proporciona la capacidad de sugerir resultados posibles a una entrada dada por el usuario.

– *Three*⁵

Librería de JavaScript para crear y mostrar gráficos animados por ordenador en 3D en un navegador Web utilizando elementos canvas de HTML5, SVG o WebGL.

– *Three-FullScreen.js*

Plugin JavaScript que, en combinación con la librería principal *Three.js* permite la gestión y uso del evento pantalla completa en el entorno Threejs.

⁴ Véase *Leaflet* en la Bibliografía

⁵ Véase *Threejs* en la Bibliografía



3.5 Servidores de aplicaciones Java

Los servidores de aplicaciones Java ofrecen una manera de integrar y ofrecer las funcionalidades requeridas por la gran mayoría de sistemas empresariales, una de las razones por las cuales el mercado ha sido inundado con estos “Application Servers” es que están diseñados alrededor de JEE, que es sólo un grupo de especificaciones definidas por Oracle.

Dependiendo de la empresa que desarrolle el “Application Server”, éste puede contener inclusive hasta un “Servidor de Páginas” o algún otro desarrollo propietario, sin embargo, los dos elementos primordiales (aunque no sean comercializados como tal) son el “Servlet Engine” (Web-Container) y “Enterprise Bean Engine” (Bean-Container).

El Servlet Engine (Web-Container) en un “Application Server” realiza las mismas funcionalidades que fueron mencionadas anteriormente, ofrecer un ambiente para JSP y Servlets.

El “Enterprise Bean Engine” (Bean-Container) ofrece un “ambiente” donde residen EJBs (“Enterprise Java Beans”). Es mediante estos que se ejecuta la lógica de negocios sobre la información que reside en los sistemas empresariales (“EIS”). En el “Bean Container”, al igual que en el “Web Container”, se contemplan varias funcionalidades: “Pooling” hacia bases de Datos (JDBC), control de transacciones (JTA-JTS), conectividad con ERP (Connectors), aplicaciones legacy (CoRBA), entre otras cosas.

La mayor ventaja de este tipo de arquitectura se debe a la separación de funcionalidades y uso de protocolos de redes, como RMI/CORBA. Esto facilita que puedan existir 4 o 5 Hosts en diferentes regiones geográficas, cada uno empleando cualquiera de los componentes antes mencionados. Por último, existen diversos “Application Servers” que son denominados “Fully JEE Compilant”, esto indica que cumplen con todas las especificaciones JEE indicadas por Oracle.

Algunos servidores de aplicaciones son:

- Oracle WebLogic
- Apache Tomcat
- Glassfish
- WildFly

Estado del Arte

3.5.1 Oracle WebLogic

Como una plataforma líder en la industria de comercio electrónico, WebLogic Server nos permite desarrollar y desplegar rápidamente aplicaciones fiables, seguras, escalables y manejables. Maneja los detalles a nivel del sistema para que podamos concentrarnos en la lógica de negocio y la presentación.

WebLogic Server utiliza tecnologías de la plataforma Java Enterprise Edition. JEE es la plataforma estándar para desarrollar aplicaciones multi-capa basadas en el lenguaje de programación Java.

Las aplicaciones JEE están basadas en componentes estandarizados y modulares. WebLogic Server proporciona un conjunto completo de servicios para esos componentes y maneja automáticamente muchos detalles del comportamiento de la aplicación, sin requerir programación.

WebLogic Server proporciona características esenciales para desarrollar y desplegar aplicaciones críticas de comercio electrónico a través de entornos de computación distribuidos y heterogéneos.

3.5.2 Apache Tomcat

Funciona como un contenedor de servlets desarrollado bajo el proyecto Jakarta en la Apache Software Foundation. Tomcat implementa las especificaciones de los servlets y de JavaServer Pages (JSP) de Oracle. Se le considera un servidor de aplicaciones.

Tomcat empezó siendo una implementación de la especificación de los servlets comenzada por James Duncan Davidson, que trabajaba como arquitecto de software en Sun y que, posteriormente, ayudó a hacer los proyectos open source y en su donación a la Apache Software Foundation.

Duncan Davidson inicialmente esperaba que el proyecto se convirtiese en open source y, dado que la mayoría de los proyectos open source tienen libros de O'Reilly asociados con un animal en la portada, quiso ponerle al proyecto nombre de animal. Eligió Tomcat (gato) pretendiendo representar la capacidad de cuidarse por sí mismo, de ser independiente.

Tomcat funciona con cualquier servidor web con soporte para servlets y JSPs. Tomcat incluye el compilador Jasper, que compila JSPs convirtiéndolas en servlets. El motor de servlets del Tomcat a menudo se presenta en combinación con el servidor web Apache. Tomcat puede, asimismo, funcionar como servidor web por sí mismo. Opera de tal manera en entornos de desarrollo poco exigentes en términos de velocidad y de manejo de transacciones.

Dado que Tomcat fue escrito en Java, funciona en cualquier sistema operativo que disponga de la máquina virtual.



Servicio Web de Información geoposicionada

Tomcat lo desarrollan y lo mantienen miembros de la Apache Software Foundation y voluntarios independientes. Los usuarios disponen de libre acceso a su código fuente y a su forma binaria en los términos establecidos en la Apache Software Licence. Las primeras distribuciones de Tomcat fueron las versiones 3.0.x. Las versiones estables más recientes son las 8.x, que implementan las especificaciones de Servlet 3.0 y de JSP 2.3.

3.5.3 WildFly

WildFly, anteriormente conocido como JBoss, es un servidor de aplicaciones Java EE de código abierto implementado en Java puro. Al estar basado en Java, JBoss puede ser utilizado en cualquier sistema operativo para el que esté disponible la máquina virtual de Java. JBoss Inc., empresa fundada por Marc Fleury y que desarrolló inicialmente JBoss, fue adquirida por Red Hat en abril del 2006. En febrero de 2007, Marc Fleury dejó Red Hat.

WildFly es software libre y de código abierto, sujeto a los requisitos de la GNU Lesser General Public License (LGPL), versión 2.1.

El proyecto se nutre de una red mundial de colaboradores. Los ingresos de la empresa están basados en un modelo de negocio de servicios.

WildFly es el primer servidor de aplicaciones de código abierto, preparado para la producción y certificado J2EE 1.4, disponible en el mercado, ofreciendo una plataforma de alto rendimiento para aplicaciones de e-business. Combinando una arquitectura orientada a servicios SOA, con una licencia GNU de código abierto, JBoss puede ser descargado, utilizado, incrustado y distribuido sin restricciones por la licencia.

Las características destacadas de JBoss incluyen:

- Implementación de la especificación inicial de EJB 3.0.
- JBoss AOP está orientado a trabajar con Programación Orientada a Aspectos. Esto permitirá añadir fácilmente servicios empresariales (transacciones, seguridad, persistencia) a clases Java simples.
- Hibernate es un servicio de persistencia objeto/relaciones y consultas para Java. Facilita a los desarrolladores crear las clases de persistencia utilizando el lenguaje Java incluyendo la asociación, herencia, polimorfismo y composición y el entorno de colecciones Java.
- JBoss Cache es un producto diseñado para almacenar en caché los objetos Java más frecuentemente accedidos de manera que aumente de forma notable el rendimiento de aplicaciones e-business. Eliminando accesos innecesarios a la base de datos. JBoss Cache reduce el tráfico de red e incrementa la escalabilidad de las aplicaciones.
- Cumple los estándares.
- Confiable a nivel de empresa.
- Incrustable, orientado a arquitectura de servicios.
- Flexibilidad consistente.
- Servicios del middleware para cualquier objeto de Java.



Estado del Arte

3.5.4 GlassFish

GlassFish es un servidor de aplicaciones de código abierto creado por Sun Microsystems para la plataforma Java EE y, ahora, patrocinado por Oracle. La versión soportada se llama Oracle GlassFish Server.

GlassFish es de software libre y bajo dos tipos de licencias: Common Development and Distribution License (CDDL) y la GNU General Public License (GPL).

Es la implementación de referencia para Java EE y como soporte para JavaBeans Empresariales, JPA, JavaServers, Faces, JMS, RMI, JSP, Servlets, etc. Esto permite a desarrolladores crear aplicaciones empresariales que son portables y escalables, además de ser capaces de integrarse con tecnologías menos actuales así como instalar servicios adicionales que amplíen sus capacidades de forma opcional.

Construido sobre el núcleo modular de OSGi, GlassFish se ejecuta directamente en la parte superior de la implementación Apache Felix.

GlassFish se basa en el código fuente liberado por Sun y Oracle. Este usa un derivado de Apache Tomcat como contenedor de Servlet para servir contenido web, con el añadido llamado Grizzly que usa Java New I/O para la escalabilidad y velocidad.

3.5.5 Elección de la tecnología a utilizar en el sistema

En este caso, la elección es complicada ya que todos los servidores actuales implementan nuestras necesidades de desarrollo en mayor o menor medida.

Como nos interesa un servidor de aplicaciones capaz de contener servicios web basados en restful JAX-RS 2.0 con Jersey para nuestra aplicación servidor, y Servlets y JSP para nuestra aplicación cliente, que sea potente, sencillo y de software libre, nos hemos decantado por el uso del servidor de aplicaciones GlassFish.

Aunque la configuración de GlassFish pueda resultar algo más complicada que otras opciones más sencillas, hemos creído que vale la pena emplear el mismo servidor de aplicaciones para desplegar la aplicación restful y la aplicación cliente soportando la versión Java EE 7.

Utilizaremos la última versión estable de GlassFish, la 4.0 que es compatible con las especificaciones a utilizar, Servlet 3.0, JSP 2.3, JAX-RS y Java EE7.



3.6 Base de Datos

Una base o banco de datos es un conjunto de datos que pertenecen al mismo contexto almacenados sistemáticamente para su posterior uso. Existen multitud de formas de ordenar los datos de manera que después podamos acceder a ellos de una forma rápida y sencilla, es lo que se conoce como modelos de datos.

Un modelo de datos es, básicamente, una “descripción” de algo conocido como contenedor de datos (algo en donde se guarda la información), así como de los métodos para almacenar y recuperar información de esos contenedores. Los modelos de datos no son cosas físicas: son abstracciones que permiten la implementación de un sistema eficiente de base de datos; por lo general se refieren a algoritmos y conceptos matemáticos. Existen multitud de modelos de datos: bases de datos jerárquicas, de red, orientadas a objetos, relacionales, documentales, deductivas, etc.

Para nuestra base de datos elegiremos el modelo relacional que es uno de los modelos más extendidos. Una base de datos relacional archiva datos en tablas separadas en vez de colocar todos los datos en un gran archivo. Las tablas están conectadas por relaciones definidas que hacen posible combinar datos de diferentes tablas. Esto permite velocidad y flexibilidad que es precisamente lo que necesitamos para que desde nuestra base de datos que maneja bastante información (tabla de profesores, asignaturas, mapas, etc.) obtengamos un rápido resultado a nuestras peticiones. Además es fácil de utilizar (peticiones mediante consultas) y de entender (cada relación es como si fuese un tabla con filas y columnas).

Como lenguaje utilizado para programar nuestra base de datos relacional utilizaremos SQL, que es el lenguaje más extendido para este tipo de bases de datos.

Sistemas de gestión de base de datos relacionales (SGBD)

Son muchas las aplicaciones que requieren acceder a datos. Estos datos se deben almacenar en algún soporte permanente, y las aplicaciones deben disponer de un medio para acceder a ellos. Normalmente, la forma en que un programa accede a un fichero es a través del Sistema operativo. Este SO ayuda a proveer de funciones como abrir archivo, leer información del archivo, guardar información, etc., pero no indica que el SGBD sea parte del SO, solamente es una aplicación. No obstante, este procedimiento de acceso a ficheros es altamente ineficaz cuando se trata con un volumen elevado de información. Es aquí donde aparecen los Sistemas Gestores de Bases de Datos: proporcionan un interfaz entre aplicaciones y sistema operativo, consiguiendo, entre otras cosas, que el acceso a los datos se realice de una forma más eficiente, más fácil de implementar y, sobre todo, más segura.



Estado del Arte

Existen distintos objetivos que deben cumplir los SGBD:

Abstracción de la información. Los usuarios de los SGBD ahorran a los usuarios detalles acerca del almacenamiento físico de los datos. Da lo mismo si una base de datos ocupa uno o cientos de archivos, este hecho se hace transparente al usuario. Así, se definen varios niveles de abstracción.

Independencia. La independencia de los datos consiste en la capacidad de modificar el esquema (físico o lógico) de una base de datos sin tener que realizar cambios en las aplicaciones que se sirven de ella.

Redundancia mínima. Un buen diseño de una base de datos logrará evitar la aparición de información repetida o redundante. De entrada, lo ideal es lograr una redundancia nula; no obstante, en algunos casos la complejidad de los cálculos hace necesaria la aparición de redundancias.

Consistencia. En aquellos casos en los que no se ha logrado esta redundancia nula, será necesario vigilar que aquella información que aparece repetida se actualice de forma coherente, es decir, que todos los datos repetidos se actualicen de forma simultánea.

Seguridad. La información almacenada en una base de datos puede llegar a tener un gran valor. Los SGBD deben garantizar que esta información se encuentra asegurada frente a usuarios malintencionados, que intenten leer información privilegiada; frente a ataques que deseen manipular o destruir la información; o, simplemente, ante las torpezas de algún usuario autorizado pero despistado. Normalmente, los SGBD disponen de un complejo sistema de permisos a usuarios y grupos de usuarios, que permiten otorgar diversas categorías de permisos.

Integridad. Se trata de adoptar las medidas necesarias para garantizar la validez de los datos almacenados. Es decir, se trata de proteger los datos ante fallos de hardware, datos introducidos por usuarios descuidados, o cualquier otra circunstancia capaz de corromper la información almacenada.

Respaldo y recuperación. Los SGBD deben proporcionar una forma eficiente de realizar copias de seguridad de la información almacenada en ellos, y de restaurar a partir de estas copias los datos que se hayan podido perder.

Control de la concurrencia. En la mayoría de entornos (excepto quizás el doméstico), lo más habitual es que sean muchas las personas que acceden a una base de datos, bien para recuperar información, bien para almacenarla. Y es también frecuente que dichos accesos se realicen de forma simultánea. Así pues, un SGBD debe controlar este acceso concurrente a la información, que podría derivar en inconsistencias.

Tiempo de respuesta. Lógicamente, es deseable minimizar el tiempo que el SGBD tarda en darnos la información solicitada y en almacenar los cambios realizados.



Servicio Web de Información geoposicionada

Ventajas:

- Facilidad de manejo de grandes volúmenes de información.
- Gran velocidad en muy poco tiempo.
- Independencia del tratamiento de información.
- Seguridad de la información (acceso a usuarios autorizados), protección de información, de modificaciones, inclusiones, consulta.
- No hay duplicidad de información, comprobación de información en el momento de introducir la misma.
- Integridad referencial al terminar los registros.

Desventajas

- El costo de actualización del hardware y software son muy elevados.
- Costo (salario) del administrador de la base de datos es costoso.
- El mal diseño de esta puede originar problemas en el futuro.
- Un mal adiestramiento a los usuarios puede originar problemas a futuro.
- Si no se encuentra un manual del sistema no se podrán hacer relaciones con facilidad.
- Generan campos vacíos en exceso.
- El mal diseño de seguridad genera problemas en esta.

Seguidamente pasaremos a enumerar los diferentes tipos de Sistemas más importantes que existen:

3.6.1 Microsoft SQL Server

Es un sistema de gestión de bases de datos relacionales basada en el lenguaje SQL, capaz de poner a disposición de muchos usuarios grandes cantidades de datos de manera simultánea.

Entre sus características figuran:

- Soporte de transacciones.
- Gran estabilidad.
- Gran seguridad.
- Escalabilidad.
- Soporta procedimientos almacenados.

Incluye también un potente entorno gráfico de administración, que permite el uso de comandos DDL (Lenguaje de Definición de Datos) y DML (Lenguaje de Manipulación de Datos) gráficamente.

Permite trabajar en modo cliente-servidor donde la información y datos se alojan en el servidor y las terminales o clientes de la red sólo acceden a la información.

Además permite administrar información de otros servidores de datos.

Este sistema incluye una versión reducida, llamada MSDE con el mismo motor de base de datos pero orientado a proyectos más pequeños.

Microsoft SQL Server constituye la alternativa de Microsoft a otros potentes sistemas gestores de bases de datos como son Oracle o MySQL.



Estado del Arte

Es común desarrollar completos proyectos complementando Microsoft SQL Server y Microsoft Access a través de los llamados ADP (Access Data Project).

De esta forma se completa una potente base de datos (Microsoft SQL Server) con un entorno de desarrollo cómodo y de alto rendimiento (VBA Access) a través de la implantación de aplicaciones de dos capas mediante el uso de formularios Windows.

Microsoft SQL Server, al contrario de su más cercana competencia, no es multiplataforma, ya que sólo está disponible en Sistemas Operativos de Microsoft.

3.6.2 Oracle

Es un sistema de administración de bases de datos desarrollado por Oracle Corporation. Se considera a Oracle como uno de los sistemas de bases de datos más completos, destacando:

- Soporte de transacciones.
- Estabilidad.
- Escalabilidad.
- Es multiplataforma.

Su mayor defecto es su enorme precio, que es de varios miles de euros (según versiones y licencias). Otro aspecto que ha sido criticado por algunos especialistas es la seguridad de la plataforma y las políticas de suministro de parches de seguridad, modificadas a comienzos de 2005, que incrementan el nivel de exposición de los usuarios. En los parches de actualización provistos durante el primer semestre de 2005 fueron corregidas 22 vulnerabilidades públicamente conocidas, algunas de ellas con una antigüedad de más de 2 años.

Aunque el dominio en el mercado de servidores empresariales ha sido casi total hasta hace poco, recientemente sufre la competencia del Microsoft SQL Server de Microsoft y de la oferta de otros sistemas de gestión de bases de datos relacionales con licencia libre como PostgreSQL o MySQL.

Oracle surge a finales de los 70 bajo el nombre de Relational Software a partir de un estudio sobre SGDB de George Koch. Computer World definió este estudio como uno de los más completos jamás escritos sobre bases de datos. Este artículo incluía una comparativa de productos que erigía a Relational Software como el más completo desde el punto de vista técnico. Esto se debía a que usaba la filosofía de las bases de datos relacionales, algo que por aquella época era todavía desconocido.



3.6.3 MySQL

Es un sistema de gestión de base de datos multihilo y multiusuario con más de seis millones de instalaciones como software libre bajo licencia GNU, pero, mediante licencia dual, también lo proporcionan bajo la licencia tradicional de software propietario para los casos en los que su uso sea incompatible con la licencia GPL.

Al contrario de proyectos donde el software es desarrollado por una comunidad pública y el copyright del código está en poder del autor individual, MySQL está poseído y patrocinado por una empresa privada que posee el copyright de la mayor parte del código. La compañía desarrolla y mantiene la aplicación vendiendo soporte y servicios, como lo hacen las empresas con software propietario, y contratan trabajadores alrededor del mundo que colaboran vía Internet.

Es muy utilizado en aplicaciones web. Se trata de una base de datos muy rápida en la lectura cuando utiliza el motor no transaccional MyISAM, pero puede provocar problemas de integridad en entornos de alta concurrencia en la modificación. En aplicaciones web hay baja concurrencia en la modificación de datos y en cambio el entorno es intensivo en lectura de datos, lo que hace a MySQL ideal para este tipo de aplicaciones. Algunas de estas aplicaciones que implementan MySQL son Facebook, Twitter, Wikipedia o Youtube, entre otras.

Inicialmente, carecía de elementos considerados esenciales en las bases de datos relacionales, tales como integridad referencial y transacciones. A pesar de ello, atrajo a los desarrolladores de páginas web con contenido dinámico justamente por su simplicidad; aquellos elementos faltantes fueron llenados por la vía de las aplicaciones que la utilizan.

Poco a poco los elementos faltantes están siendo incorporados tanto por desarrollos internos como por desarrolladores de software libre. Entre las características disponibles en las últimas versiones se puede destacar.

- Funciona sobre múltiples plataformas.
- Amplio subconjunto del lenguaje SQL. Algunas extensiones son incluidas igualmente.
- Disponibilidad en gran cantidad de plataformas y sistemas.
- Diferentes opciones de almacenamiento según si se desea velocidad en las operaciones o el mayor número de operaciones disponibles.
- Transacciones y claves foráneas.
- Conectividad segura.
- Replicación
- Búsqueda e indexación de campos de texto.

MySQL es software de fuente abierta. Hace uso de GPL (GNU General Public License) para definir qué puede hacer y que no puede hacer con el software en diferentes situaciones. Si las necesidades de nuestra base de datos no se ajustan al GLP o requiere introducir código MySQL en aplicaciones comerciales, se puede comprar una versión comercial licenciada.



Estado del Arte

Hay tres tipos de compilación del servidor MySQL:

- Estándar: Los binarios estándares de MySQL son los recomendados para la mayoría de los usuarios e incluyen el motor de almacenamiento InnoDB.
- Max (No se trata de MaxDB, que es una cooperación con SAP): Los binarios incluyen características adicionales que no han sido lo bastante probadas o que normalmente no son necesarias.
- MySQL-Debug: Son binarios que han sido compilados con información de depuración extra. No debe ser usada en sistemas en producción porque el código de depuración puede reducir el rendimiento.

3.6.4 Elección de la tecnología a utilizar en el sistema

En este caso la elección es bastante sencilla dado que la Universitat de València trabaja actualmente con Oracle en materia de gestión de datos. Este es un factor importante que hará que el presupuesto en este aspecto esté amortizado a la vez que se cuenta con un sistema potente. De igual manera, el emplear un sistema ya adquirido y usado por la Universitat facilita la migración de datos o reutilización de los mismos sin complicaciones.

Por estas razones hemos escogido emplear Oracle como sistema de bases de datos ya que, en su visión global, es la que mejor nos conviene emplear.

Utilizaremos la última versión, Oracle Database Xpress Edition 11g.



3.7 Comunicación de la base de datos con la aplicación

Hemos analizado cada una de las partes de nuestro sistema por separado. Pero aún no sabemos cómo es capaz de comunicarse la aplicación con la base de datos. Está claro que, si conocemos un poco Java y sabemos de su potencial, llegaremos a pensar que Java dispondrá de alguna librería que nos facilite la comunicación con la base de datos y así es, se trata de JDBC.

Oracle Java Database Connectivity⁶, más conocida por sus siglas OJDBC, es una API modificada del conector estándar JDBC por Oracle que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java, independientemente del sistema operativo donde se ejecute o de la base de datos a la cual se accede, utilizando el dialecto SQL del modelo de base de datos que se utilice.

El API JDBC se presenta como una colección de interfaces Java y métodos de gestión de manejadores de conexión hacia cada modelo específico de base de datos. Un manejador de conexiones hacia un modelo de base de datos en particular es un conjunto de clases que implementan las interfaces Java y que utilizan los métodos de registro para declarar los tipos de localizadores a base de datos (URL) que pueden manejar. Para utilizar una base de datos particular, el usuario ejecuta su programa junto con la biblioteca de conexión apropiada al modelo de su base de datos y accede a ella estableciendo una conexión, para ello provee el localizador a la base de datos y los parámetros de conexión específicos. A partir de allí puede realizar cualquier tipo de tarea con la base de datos a la que tenga permiso: consulta, actualización, creación, modificación y borrado de tablas, ejecución de procedimientos almacenados en la base de datos, etc.

Está compuesto por dos capas:

API JDBC que se encarga de comunicar con la API del administrador de controladores JDBC enviándole las sentencias SQL.

Un administrador de controladores JDBC que se comunica de forma transparente para el programador con los distintos controladores disponibles para la base de datos.

Los controladores JDBC están clasificados como:

- Tipo 1: Traducen JDBC a ODBC y se delega en ODBC para la comunicación con la base de datos.
- Tipo 2: está escrito parcialmente en Java y en código nativo.
- Tipo 3: es una biblioteca cliente escrita completamente en Java que utiliza un protocolo independiente de la BD para comunicar las peticiones a un servidor que las traduce a un protocolo específico de la BD.
- Tipo 4: es una biblioteca escrita completamente en Java que traduce las peticiones a un protocolo específico de la Base de Datos.

⁶ Véase *Como conectar a Oracle con Java* en Bibliografía



Estado del Arte

En nuestro sistema utilizaremos un controlador de tipo 4, por ello deberemos disponer de las librerías correspondientes, en nuestro caso usaremos ojdbc7.jar.

3.8 Gráficos vectoriales

Los gráficos vectoriales son los que se representan en los gráficos por ordenador por medio de “trazos”, es decir, por primitivas geométricas como puntos, líneas, curvas o polígonos. En contraste, se encuentran los gráficos formados por una retícula de píxeles como los bitmap, también llamados gráficos rasterizados.

En los gráficos vectoriales la imagen se genera como descripción de trazos. Por ejemplo, para crear un segmento de línea recta se indica: su punto inicial(x_1, y_1), su punto final (x_2, y_2), su grosor, color, etc. En cambio, en una imagen bitmap, esa misma línea estaría formada por un número determinado de puntos (píxeles) de color contiguos.

Al contrario que un bitmap, una imagen vectorial puede ser escalada, rotada o deformada sin que ello perjudique en su calidad. Normalmente un conjunto de trazos se puede agrupar formando objetos y crear formas más complejas que permiten el uso de curvas de Bézier, degradados de color, etc.

Ventajas

En muchas ocasiones requieren menor espacio en disco que un bitmap, aunque depende mucho de la imagen y de la calidad que se deseé. Las imágenes formadas por colores planos o degradados sencillos son más factibles de ser vectorizadas. A menor información para crear la imagen, menor será el tamaño del fichero. Dos imágenes con dimensiones de presentación distintas pero con la misma información vectorial, ocuparán el mismo espacio en disco.

Algunos formatos permiten animación. Está será realizada de forma sencilla mediante operaciones básicas como traslación o rotación y no requiere un gran acopio de datos.

No pierden calidad al ser escalados, rotados o deformados. Ciertamente se puede hacer zoom sobre una imagen vectorial de forma ilimitada.

Desventaja

No son aptos para mostrar fotografías o imágenes complejas, aunque algunos formatos admiten una composición mixta (vector + bitmap).

Deben ser procesados, es decir, el ordenador debe computar todos los datos para formar la imagen final. Si hay demasiados datos se puede ralentizar la presentación de la imagen, incluso en imágenes pequeñas.

Formatos vectoriales:

- PostScript
- SVG
- SWF
- WMF (Windows Metafiles)



- .AI (Adobe Illustrator)
- ...

Formatos vectoriales más importantes

3.8.1 SVG

Scalable Vector Graphics (SVG) es un lenguaje para describir gráficos vectoriales bidimensionales, tanto estáticos como animados (estos últimos con ayuda de SMIL), en XML.

SVG se convirtió en una recomendación del W3C en septiembre de 2001, por lo que se encuentra incluido en los navegadores actuales de forma nativa.

El SVG permite tres tipos de objetos gráficos:

- Formas gráficas de vectores
- Imágenes de mapa de bits/digitales
- Texto

Los objetos gráficos pueden ser agrupados, transformados y compuestos en objetos previamente renderizados, y pueden recibir un estilo común. El texto puede estar en cualquier espacio de nombres XML admitido por la aplicación, lo que mejora la posibilidad de búsqueda y la accesibilidad de los gráficos SVG. El juego de características incluye las transformaciones anidadas, los clipping paths, las máscaras alfa, los filtros de efectos, las plantillas de objetos y la extensibilidad.

El dibujado de los SVG puede ser dinámico e interactivo. El Document Object Model (DOM) para SVG, que incluye el DOM XML completo, permite animaciones de gráficos vectoriales sencillos y eficientes mediante ECMAScript o SMIL. Un juego amplio de manejadores de eventos, como “onMouseOver” y “onClick”, pueden ser asignados a cualquier objeto SVG. Debido a su compatibilidad y relación con otras normas Web, características como el scripting pueden ser aplicadas a elementos SVG y a otros elementos XML desde distintos espacios de nombre XML simultáneamente dentro de la misma página web.

Si el espacio de almacenamiento es un problema, las imágenes SVG pueden salvarse comprimidas con gzip, en cuyo caso pasan a ser imágenes SVGZ. Debido a la verbosidad del XML, este tiende a comprimirse muy bien y estos ficheros pueden ser mucho más pequeños. Aun así, a menudo el fichero vectorizado original (SVG) es más pequeño que la versión de mapa de bits.

La amplia adopción de clientes SVG, particularmente aquellos integrados nativamente en los navegadores, puede traer un significativo cambio de imagen en la web.

Estado del Arte

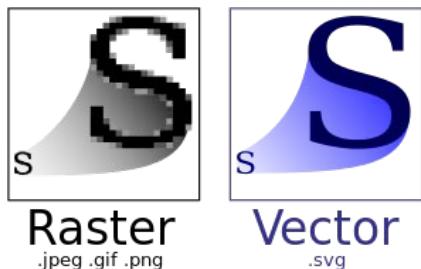


Figura 3.8.1-1: Comparación entre escalado matricial y vectorial

3.8.2 SWF (Flash)

SWF es la extensión de los archivos creados por Macromedia Flash (actualmente Adobe Flash), y significa ShockWave Flash. Los archivos SWF pueden protegerse para que no sean editables, y son una compilación y compresión del archivo de autor editable desde Flash.

Los ficheros SWF están construidos principalmente por dos elementos: objetos basados en vectores e imágenes. Las versiones más modernas también incorporan audio, vídeo (en formato Flash Video-FLV) y multitud de formas diferentes de interacción con el usuario.

El formato SWF fue desarrollado con un objetivo principal: crear ficheros de reducido tamaño, pero de gran calidad que permita interactividad. La idea fue disponer de un formato que pudiese funcionar sobre cualquier sistema y sobre un reducido ancho de banda.

El formato es bastante simple, si bien es cierto que está en formato binario y por lo tanto no es de lectura accesible como su rival SVG basado en XML. SWF ha utilizado la compresión Zlib desde el 2002 y, en general, el objetivo del formato es almacenar todos los datos usando el menor número de bits, minimizando la redundancia.

Aunque la especificación completa del formato está disponible, no es un formato abierto ya que la licencia de la especificación no permite crear software que reproduzca el formato. Por otro lado, la creación de software que cree archivos SWF sí está permitida con la condición que el archivo resultante pueda ser renderizado sin problemas por la última versión pública del reproductor Flash de Adobe.

3.8.3 Elección de la tecnología a utilizar en el sistema

La decisión para esta tecnología es bastante sencilla. Aunque los dos formatos son muy similares y se ajustan a nuestras necesidades. Flash es un formato propietario, por lo que está sujeto a cambios a voluntad de Adobe mientras que SVG es libre y fue desarrollado explícitamente para la visualización de contenido dentro de la web.

Además, hay que tener en cuenta que actualmente la reproducción de formatos Flash en web se encuentra en decadencia llegando a ser bloqueado o en proceso de bloqueo en futuras actualizaciones de los navegadores más populares.



Servicio Web de Información geoposicionada

Por estas razones pensamos que SVG es la tecnología adecuada para nuestras necesidades y utilizaremos la versión 1.1.



Estado del Arte

3.9 Resumen elección de tecnologías

Una vez repasadas las posibilidades más habituales en la actualidad para desarrollo de servicios web, pasaremos a realizar una recopilación de las tecnologías que se emplearán en el proyecto:

- Tecnologías de la Web
 - HTML versión 5.0 (HTML5)
 - CSS versión 3.0 (CSS3)
- Lenguajes del lado del servidor
 - Java Enterprise Edition (J2EE) versión 8.0
- Servicios Web
 - REST
- Lenguajes del lado del cliente
 - JavaScript
- Servidores de aplicaciones
 - GlassFish versión 4.0
- Base de Datos
 - Oracle Database Xpress Edition versión 11g
- Comunicación de la base de datos con la aplicación
 - Controlador OJDBC versión 7.
- Gráficos vectoriales
 - Scalable Vector Graphics (SVG)

3.10 Aplicaciones similares

Como ocurre con todo nuevo desarrollo, este es fruto de anteriores proyectos que, previamente, dan o daban solución a algunas necesidades o servicios que sirven como inspiración para la creación de nuevas aplicaciones que mejoren lo ya existente y nuestro desarrollo no es diferente. Para ello emplearemos la técnica de Benchmarking, que se basa en la idea de tomar referencias de lo ya existente y aquellas que mejores resultados estén dando serán adaptadas e integradas en nuestro desarrollo.

Nosotros aplicaremos esta técnica desde dos puntos de vista, el Benchmarking interno que busca identificar y evaluar los aspectos propios a fin de potenciarlos, y los externos, que procuran encontrar los aspectos que den mejores resultados de nuestros competidores, sean directos o no.

Seguidamente pasamos a comentar algunos de los servicios que tomaremos como referencia para nuestra aplicación a desarrollar:



Servicio Web de Información geoposicionada

3.10.1 Universidad de Valencia

Investigando en la aplicación propia de la universidad de valencia, cuyo desempeño trata de servir información de personal adscrito a la Universidad, se puede encontrar esta información mediante dos modalidades de búsqueda. Una de ellas es totalmente pública y permite buscar a cualquier persona que no sea estudiante en cualquiera de sus niveles. Y otra modalidad, que tiene capacidad para buscar incluso a alumnos, permite la búsqueda una vez sea haya identificado dentro del sistema como parte de la universidad mediante la introducción del binomio usuario y contraseña.

Aquest formulari permet cercar ràpidament persones en la UV, cercant per **aproximació cognoms, nom, usuari, alias i lloc/titulació i múltiples paraules.**

PAS/PDI Alumne

TROBAR! netejar Reicerca avanzada

© 2014 Servei d'Informàtica de la Universitat de València. Tel (+34) 963 54 43 10

Valencià | English

Figura 3.10.1-1: Buscador Universitat de València

Puntos a favor

La capacidad de encontrar a cualquier persona dentro la universidad que se encuentre en activo, ya sea como PDI, PAS o estudiante, es un aspecto que nos interesa preservar a fin de lograr servir una información lo más actualizada posible.

Con motivo de preservar el sentido de proteger la publicación de información que el personal no deseé ser localizado, no permitiremos que se acceda a información que un PDI o PAS no dé su aprobación.

Puntos a mejorar

La información que muestra tras una búsqueda de una persona es relativamente escaso, dejando en muchos casos a la persona que desea información igual a como empezó el proceso. Por esta razón se debe potenciar este aspecto ofreciendo una información más completa al usuario que le permita conocer más de su solicitud, sin que esta demanda derive en una pérdida de derechos del PDI o PAS que podrá habilitar o no su presencia en el servicio.

Estado del Arte

3.10.2 Universidad Carlos III de Madrid

Observando el portal de la Universidad Carlos III de Madrid, la aplicación nos ofrece un visor de fotografías panorámicas de 360 grados, lo que nos permite obtener una idea muy aproximada de los elementos que encontraremos si caminamos por esas zonas, a fin de poder orientarnos de cara a visitar algún espacio concreto que deseemos.



Figura 3.10.2-1: Visor fotografías 360 grados Universidad Carlos III de Madrid

Puntos a favor

La posibilidad de visualizar los espacios mediante fotografías de los mismos es una característica que viene integrándose progresivamente y que está totalmente extendida en aplicaciones comerciales del sector servicios como son el alojamiento y la restauración. Este aspecto es muy importante para estos negocios, pues tiende a tranquilizar al usuario sobre los elementos que va a encontrar o la localización que va a visitar sin ni siquiera moverse del sitio, además de proporcionar información adicional en forma de marcadores dentro de la imagen.

La implementación de esta capacidad a nuestro proyecto sería de gran importancia puesto que, independientemente de la finalidad del usuario (alumno, futuro alumno, profesor o visitante esporádico), este podría fácilmente formarse una idea cercana del lugar en el que estuviera interesado.

Puntos en contra

Aunque la capacidad de visualizar fotografías de 360 grados es un punto importante dentro de nuestra aplicación, este no puede eclipsar el resto de la aplicación como eje principal puesto que ese rol debe pertenecer al motor de búsquedas, siendo los panoramas un complemento de valor añadido a la información que se presenta al usuario.

Esta aplicación, por tanto, carece de la posibilidad de realizar búsquedas concretas, limitándose a la capacidad del usuario para ir seleccionando las estancias que muestra el visor.



Servicio Web de Información geoposicionada

3.10.3 University of Washington

Desde que se accede a la aplicación de la Universidad de Washington, EEUU, nos damos cuenta que presenta un diseño diáfano con una interfaz minimalista. Dispone de un buscador de edificios mediante sugerencias y muestra mapas personalizados con los nombres de los edificios en ellos.

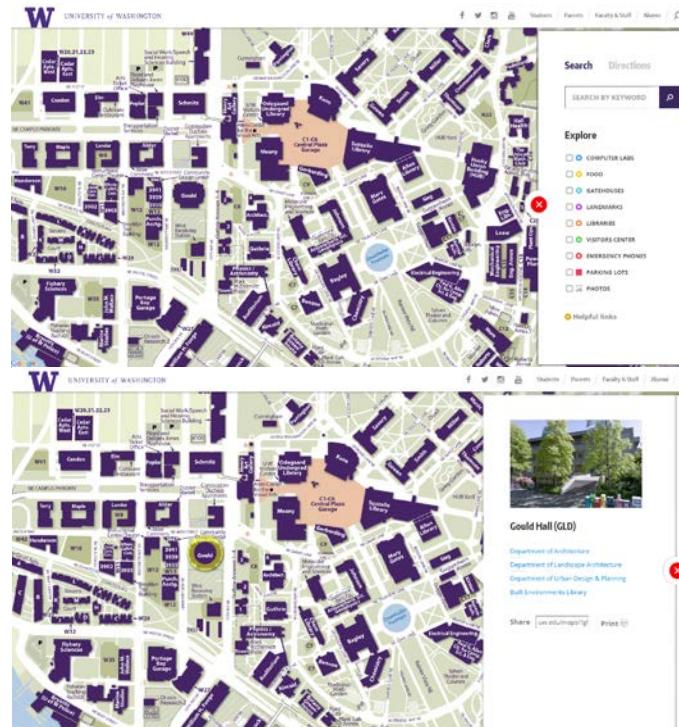


Figura 3.10.3-1: Mapas University of Washington

Puntos a favor

Existen varios elementos a tener en cuenta que pueden ser una buena incorporación para nuestra aplicación

El uso de diseños diáfanos con interfaces sencillas confirma la tendencia actual sobre los desarrollos web mediante estructuras amplias y con planteamientos minimalista que ayuden al usuario en lugar de saturarlo con diversos estímulos simultáneos.

La capacidad de generar enlaces directos al recurso con el fin de poder compartirlo a otros usuarios que estén interesados en acceder al mismo es un punto a estudiar su implementación dado el potencial de difusión del que dispondría un espacio a través de correo electrónico o redes sociales en la organización de eventos, llamamientos de profesores a espacios concretos para sus clases o cualquier otra finalidad dentro del marco universitario.

Disponer un buscador que permita proporcionar distintas sugerencias del recurso en base a los caracteres introducidos es una característica necesaria de incorporar debido a que los usuarios no siempre conocen la totalidad del recurso que buscan y esto puede suponer una ayuda considerable.

Estado del Arte

Puntos en contra

Debido a que deseamos mantener un balance equitativo entre información y privacidad debemos prescindir de mostrar información global de forma muy evidente sin que el usuario aporte ningún dato concreto de su búsqueda. Con esta acción podemos prevenir la exposición sin control de cualquier persona, ya sea PDI o PAS, sin que exista un requerimiento expreso por parte del usuario dentro de la aplicación.

3.10.4 University of Ottawa

La Aplicación de mapas de la Universidad de Ottawa es un claro ejemplo de como la sencillez y la categorización pueden dar buenos resultados de usabilidad. Se nos plantea una interfaz en dos partes donde a la izquierda disponemos de un mapa personalizado bajo el uso del proveedor de mapas Google que nos permite visualizar los distintos edificios con la visualización plano o satélite, y a la derecha poseeremos una interfaz de opciones para activar o desactivar distintas categorías presentes para su visualización en el mapa, seleccionar un edificio de una lista o proporcionar indicaciones para llegar a un edificio dado un origen.

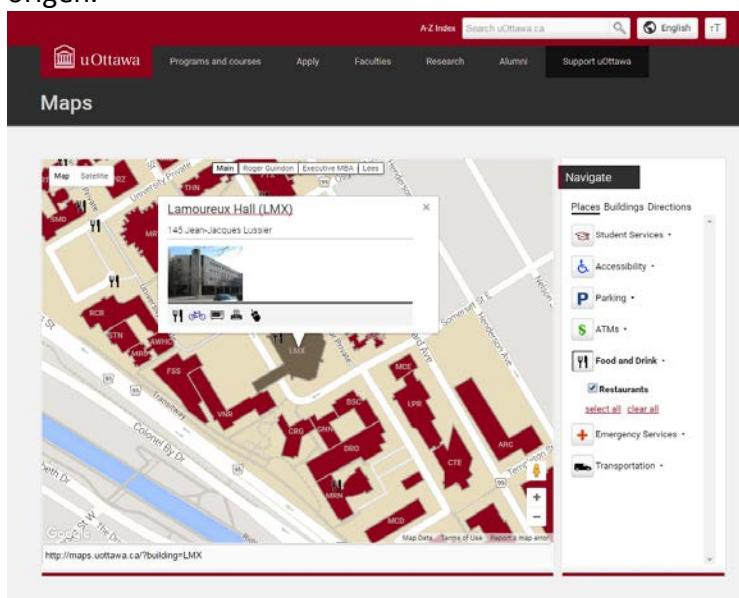


Figura 3.10.4-1: Mapas University of Ottawa

Puntos a favor

La idea de categorizar los recursos es una opción a tener en cuenta a fin de proporcionar información de localizaciones clasificada por bloques temáticos dependiendo del interés del usuario. Esta característica es aplicable a nuestro proyecto realizando una subcategorización de espacios que nos permita ajustar la búsqueda de los mismos.

Un punto a considerar es la inclusión de iconos informativos sobre elementos o servicios disponibles en el espacio buscado. Esto supondría proporcionar una información más completa de la localización deseada.

La capacidad de obtener indicaciones a un edificio dada una posición de origen es una característica a tener en cuenta puesto que facilita mucho la navegación a un lugar deseado, sin que el éxito o fracaso de la localización de un espacio corresponda



Servicio Web de Información geoposicionada

exclusivamente del usuario. Evidentemente, esta funcionalidad debería ser más concreta y localizar a nivel de espacios mediante navegación en interiores, tecnología todavía no estandarizada y con soluciones a medida.

Puntos en contra

Aunque se utiliza un diseño sencillo y minimalista, el espacio destinado al mapa es limitado, dejando mucho del espacio lateral sin aprovechar, causando una sensación de asfixia en la composición visual de su estructura.

La interfaz carece de buscador y se limita su uso a la capacidad del usuario para seleccionar el edificio que desea visualizar. Esto obliga al usuario a recorrer una lista considerable buscando el lugar indicado, con la consecuente pérdida de tiempo que esto supone si el usuario no está familiarizado con la ubicación del elemento que quiere localizar.



4 Planificación y Metodología

4.1 Planificación y gestión del proyecto

La planificación de un proyecto debe afrontarse de manera adecuada para que, al final del mismo, se pueda hablar de éxito. No se trata de una etapa independiente abordable en un momento concreto del ciclo del proyecto, es decir, no se puede hablar de un antes y un después al proceso de planificación, puesto que, según avance el proyecto, será necesario modificar tareas, reasignar recursos, etc. Se debe tener claro que, si bien sí podemos hablar de una “etapa de planificación”, llamada así porque aglutina la mayor parte de los esfuerzos para planificar todas las variables que se darán cita, cada vez que se intenta prever un comportamiento futuro y se toman las medidas necesarias se está planificando.

4.2 Análisis de requisitos

Para poder estimar posteriormente los plazos a cumplir en la realización del proyecto deberemos reflexión sobre los objetivos que pretendemos alcanzar con el mismo. Extrayendo y recopilando las características principales, agrupándolos en funcionales y no funcionales, podremos visualizar los aspectos necesarios a tener en cuenta en la planificación.

4.2.1 Funcionales

Los requerimientos funcionales son declaraciones de las funcionalidades que proveerá el sistema. Generalmente hacer referencia al comportamiento y funciones que el sistema debe cumplir y que cristalizan posteriormente en los casos de uso. En algunos casos, los requerimientos funcionales de los sistemas también declaran explícitamente lo que el sistema no debe hacer.

Seguidamente, pasaremos a enumerar los requisitos funcionales y su justificación:

4.2.1.1 El sistema debe permitir localizar en el mapa el despacho de un profesor

Es una necesidad troncal de la aplicación y una de las razones de su existencia. Debe encontrarse el profesor deseado por el usuario y el sistema marcarnos su localización en el mapa y delimitarnos en el plano su ubicación espacial.

4.2.1.2 El sistema permitirá introducir el nombre de profesor

Para facilitar la tarea o acelerar el proceso de búsqueda, el usuario puede escribir en un formulario el nombre del docente directamente en lugar de buscarlo en una tabla o lista.

4.2.1.3 El sistema debe permitir localizar en el mapa el despacho de profesores por asignatura impartida:

Es una necesidad no prioritaria de la aplicación como variación a la búsqueda de profesor ante el desconocimiento del nombre del profesor. Debe encontrarse el profesor deseado que imparte la asignatura seleccionada por el usuario y el sistema marcarnos su localización en el mapa y delimitarnos en el plano su ubicación espacial.



4.2.1.4 El sistema permitirá introducir el nombre de asignaturas:

Para facilitar la tarea o acelerar el proceso de búsqueda, el usuario puede escribir en un formulario el nombre de la asignatura directamente en lugar de buscarlo en una tabla o lista.

4.2.1.5 El sistema debe permitir localizar en el mapa espacios

Es una necesidad troncal de la aplicación y una de las razones de su existencia. Debe encontrarse el espacio deseado por el usuario y el sistema marcarlos su localización en el mapa y delimitarnos en el plano su ubicación espacial.

4.2.1.6 El sistema permitirá introducir el nombre del espacio

Para facilitar la tarea o acelerar el proceso de búsqueda, el usuario puede escribir en un formulario el nombre del espacio directamente en lugar de buscarlo en una tabla o lista.

4.2.1.7 El sistema debe permitir cambiar las características del mapa.

Debido a que los usuarios se ubican en los mapas empleando la disposición de las calles en un plano como la vista satélite para posicionarse o buscar referencias evidentes en el terreno, se debe proporcionar la posibilidad de que los usuarios puedan variar el tipo de representación en cualquier momento de uso del mapa.

4.2.1.8 El sistema debe permitir cambiar las características del plano.

Debido a que existen varias maneras de designar a un mismo espacio (por su código de situación o por su descripción), tenemos varias maneras de categorizar los espacios y debemos representar varios pisos que componen un edificio, con la superposición bidimensional que esto supone en un plano. El usuario deberá disponer de la posibilidad de elegir la representación de planos que más se ajuste a sus necesidades.

4.2.1.9 El sistema debe evitar acceder a recursos no permitidos.

El acceso a recursos que no se puedan mostrar por no disponer de la preceptiva autorización de dirección o del propio profesor dará como resultado la imposibilidad de acceder a los datos mostrando un mensaje de advertencia informando del suceso.

4.2.1.10 El sistema permitirá localizar en el mapa la posición GPS del usuario.

Al acceder a la localización de un recurso de nuestro sistema, a modo de complemento, se muestra la posición actual del usuario mediante su localizador GPS y se representa en el mapa mediante un marcador que lo identifique.

4.2.1.11 El sistema debe permitir interactuar con el mapa.

Para aumentar la interactividad del usuario y la libertad de observar el mapa de forma dinámica y accesible, se debe permitir que el usuario realice acciones de desplazamiento, aproximación y alejamiento del mismo a fin de que el usuario disponga el mapa a su gusto o necesidad.

4.2.1.12 El sistema debe permitir visualizar fotografías de 360 grados.

Con tal de enriquecer la información que se da de un espacio accesible, se proporcionan imágenes panorámicas de 360 grado que permitan visualizar el espacio requerido que faciliten la compresión, localización y orientación del usuario.



Planificación y Metodología

4.2.1.13 El sistema debe permitirnos acceder directamente a un recurso web.

Para aumentar la capacidad de difusión de un recurso sea cual sea la circunstancia, como por ejemplo la de informar de un seminario, tutoría, evento universitario, etc. Disponiendo de un enlace directo que pueda ser enviado por medios electrónicos como el correo electrónico donde el receptor, con solo acceder al enlace, pueda visualizar el recurso de forma directa y sin recurrir a reproducir el proceso de búsqueda inicial.

4.2.1.14 El sistema debe permitir generar un enlace directo a un recurso web.

Con motivo de facilitar la difusión y el acceso directo a un recurso. El sistema debe ser capaz de generar el enlace correspondiente que envíe de una manera directa e inequívoca a la búsqueda efectuada por el usuario.

4.2.1.15 El sistema debe permitir compartir el recurso web en redes sociales

Debido a la proliferación y el continuo avance de las redes sociales como medio de difusión masivo de información, eventos y noticias, se deben tener en cuenta estos “medios de comunicación” que proporcionen un grado superior de extensión de un recurso con motivo de eventos institucionales, seminarios en salones de actos o actos de graduación entre otros. El sistema es capaz de realizar publicaciones en las redes sociales más extendidas como Facebook y Twitter y, por tanto, se automatiza la publicación de contenidos basados en geolocalización.

4.2.2 No funcionales

Los requerimientos no funcionales representan características generales y restricciones de la aplicación que se esté desarrollando. Generalmente hacen referencia a atributos relacionados con la eficiencia, seguridad o usabilidad del sistema, entre otros.

4.2.2.1 El sistema debe respetar la legislación relacionada con el derecho a la imagen

Debido a la existencia de fotografías 360 grados que pueden captar individuos presentes en el espacio capturado. Debemos ser conscientes de los deberes y obligaciones a los que estamos sujetos.

4.2.2.2 El sistema debe respetar la legislación relacionada con el secreto industrial

Debido al tratamiento de información, recursos y material cedidos por la Universitat de València y que pueden suponer un foco de filtración a terceros. Debemos ser conscientes de los deberes y obligaciones a los que estamos sujetos sobre la no divulgación de material sensible o comprometido.

4.2.2.3 El sistema debe respetar la legislación en materia de protección de datos

Debido al tratamiento de datos que se hará de información relativa a profesores, asignaturas y/o espacios durante el desarrollo del proyecto, debemos ser conscientes de los deberes y obligaciones a los que estamos sujetos



4.2.2.4 El sistema deberá estar disponible en los navegadores más conocidos

Debido a que existe una gran cantidad de navegadores extendidos entre los usuarios, necesitamos confeccionar nuestro sistema haciéndolo compatible con los navegadores más empleados como Chrome, Firefox, Safari e IE, a fin de dar cobertura al mayor número de usuarios posible.

4.2.2.5 El sistema dispondrá de una interfaz abierta, diáfana e intuitiva

Deberemos tener en cuenta las últimas tendencias que se están aplicando en el diseño web, donde predominan los sitios diáfanos con controles bien localizados y que permitan una visión global de la información sin realizar excesivos clics, a la vez que realiza una curva de aprendizaje bastante rápida que hace innecesaria la formación en el uso de la aplicación.

4.2.2.6 El sistema deberá permitir la concurrencia de al menos 30 personas simultáneas

Nuestro sistema debe dar cobertura a múltiples usuarios que puedan necesitar consultar algún espacio en un momento determinado. Por esta razón, debemos contemplar la posibilidad de que varios usuarios accedan a nuestro sistema desde diferentes máquinas solicitando el servicio de algún recurso. Como nuestra aplicación no necesita de una exigencia especial de uso constante, sino que es una utilidad puntual y durante un corto periodo de tiempo, podemos considerar que 30 usuarios simultáneos deben poder utilizar el sistema sin producirse problemas de caída del servicio.

4.2.2.7 El sistema deberá mantener el servicio 24/7

Nuestro sistema, debido a que cualquier usuario puede acceder en cualquier momento para realizar una consulta, necesita mantenerse activo 24 horas los 7 días de forma ininterrumpida.

4.2.2.8 Ante un fallo de sistema, no se tardará más de 5 minutos en restaurar el sistema

Puesto que ningún sistema es infalible ante fallos, sobrecargas o pérdidas de conexión, en el caso de que nuestro sistema tenga una caída debido a problemas leves, deberá rearmanzarse en un máximo de 5 min para re establecer el servicio nuevamente.

4.3 Diagrama de hitos

En toda planificación de un proyecto se definen unos puntos temporales en los que se finaliza cada tarea a realizar. Estos puntos en el tiempo se definen como hitos que se van alcanzando durante la vida del proyecto. Para su representación haremos uso de un diagrama de hitos.

Diagrama de Hitos	
Tareas	Finalización
Análisis del sistema	22/01
Diseño del sistema	11/02
Desarrollo del sistema	21/04
Ánalisis de resultados	09/05
Redacción de la documentación	14/07

Figura 4.2.2-1: Diagrama de hitos



Planificación y Metodología

Este diagrama es poco visual ya que no se aprecian claramente cómo se desarrollan las distintas etapas de los hitos, pero sí se observa su finalización. Para una mejor clasificación de la estructura temporal del proyecto haremos uso de un diagrama de Gantt.

4.4 Diagrama de Gantt

Existen múltiples opciones en las técnicas de planificación, la herramienta clásica y más utilizada para la programación de proyectos es la gráfica de Gantt.

Una gráfica de Gantt es un diagrama bidimensional en el que se representan las diferentes actividades o tareas del proyecto frente a los tiempos necesarios para llevar a cabo las mismas.

Cada una de las actividades que tiene lugar en el proyecto se representa en la gráfica de Gantt mediante una barra horizontal, cuyo extremo izquierdo representa la fecha de inicio de la actividad, viniendo expresada su duración por la longitud de la misma a lo largo del calendario.



Figura 4.2.2-1: Diagrama de Gantt

4.5 Estimación de costes y estudio de viabilidad

Existen una serie de métricas propuestas por la Ingeniería del Software para determinar el esfuerzo de un proyecto, el alcance del mismo y la productividad de sus programadores. Vamos a aplicar algunas de las mismas a este desarrollo, para calibrar su dificultad y rendimiento obtenido.

4.5.1 El método Juicio de expertos

Una metodología muy utilizada en estimación de costos de proyectos software es el conocido como Juicio por Expertos, que se basa en el conjunto de opciones y valoraciones cimentadas en la experiencia y el conocimiento por parte de una élite experta en desarrollos a largo plazo.



Servicio Web de Información geoposicionada

Las ventajas de esta técnica se centran, fundamentalmente, en el conocimiento profundo que tienen expertos sobre un tema que es objeto de evaluación. Esto permite un ahorro de tiempo considerable al adquirir información de personas que han experimentado anteriormente desarrollos similares, obteniendo credibilidad ante las estimaciones y reduciendo costes derivados de un estudio completo mediante otros métodos.

Sin embargo, cometer el error de confiarse u olvidar factores importantes del nuevo desarrollo, que primen la opinión de determinados expertos a costa del resto o que se tienda a exceder el campo de competencia, son algunas de las posibles amenazas que pueden producir un fuerte impacto en nuestra estimación. Por ello debemos prevenir estos problemas mediante la elaboración de una definición clara de nuestro sistema describiendo los aspectos que se desean evaluar, la especialización requerida para cada experto y en el medio que proporcionarán la información.

Para crear este comité de expertos, hemos contactado con personas con una amplia experiencia en el campo a tratar, proporcionándoles una descripción del proyecto con los objetivos a desarrollar e indicarles que debían estimar la planificación del tiempo en días basándose en una media jornada de 5 horas diarias.

Los expertos elegidos son los siguientes:

Inmaculada Coma Tatay: Profesora Titular, coordinadora del grado en Ingeniería Multimedia por la Universidad de València y profesora de programación web y entornos de usuario.

Francisco Grimaldo Moreno: Profesor Contratado Doctor. Subdirector de la Escuela Tècnica Superior de Ingeniería por de la Universitat de València y profesor de programación multimedia.

José Javier Samper Zapater: Contratado Doctor de la Escuela Tècnica Superior de Ingeniería por de la Universitat de València, profesor de programación hipermedia y programación web avanzada.

Óscar Abella: Analista programador en la Unitat Web i Màrqueting de la Universitat de València.

Christian Sánchez León: Programador Web Freelance y con Master en Ingeniería del Software por la Universidad de Dongbeidaxue, República Popular China.

Cada uno de estos expertos ha llenado una tabla con las tareas y tres categorías diferentes: Tiempo optimista, tiempo pesimista y el tiempo más probable de realización de cada tarea. El primer caso, tiempo optimista, hace referencia al tiempo más corto para la realización de una tarea determinada. El segundo caso, tiempo pesimista, al tiempo más largo para la realización de esa misma tarea. Y el tiempo probable, se trata del tiempo más probable de duración de la tarea.

A partir de estos tiempos, podemos calcular el tiempo estimado para cada tarea.

Para ello, utilizamos la siguiente función:



Planificación y Metodología

$$t_{experto} = \frac{t_{optimista} + 4 \cdot t_{probable} + t_{pesimista}}{6}$$

Estas tablas se quedan de la siguiente manera, indicando los tiempos en días:

- Experto nº1: Inmaculada Coma Tatay

Nombre de Tarea	Tiempo pesimista	Tiempo más probable	Tiempo optimista	Tiempo Experto
Estado del Arte	32	24	18	24
Especificación del sistema	16	8	6	9
Planificación	20	12	8	13
Costes	16	8	6	9
Análisis				
Definición del sistema	16	8	6	9
Establecimiento de los requisitos del cliente	16	8	6	9
Identificar casos de uso	8	4	3	5
Prototipo interfaz del usuario	16	8	6	9
Análisis de clases	16	8	6	9
Diseño				
Definición de la arquitectura	16	8	6	9
Diseño de las clases	16	8	6	9
Diseño de los datos	16	8	6	9
Diseño Web	16	8	6	9
Desarrollo				
Creación Base de datos	32	24	18	24
Toma de panorámicas 360 grados	24	16	10	16
Creación planos ETSE	48	32	24	33
Aplicación Servidor (REST)	48	32	24	33
Aplicación Cliente				
Codificación de las páginas	48	32	24	33
Estilos CSS	48	32	24	33
Codificación de los scripts	48	32	24	33
Pruebas				
Pruebas sistema	16	8	6	9
Pruebas de estrés	16	8	6	9
Pruebas de usuarios	16	8	6	9
Redacción de documentación	200	120	100	130

Figura 4.5.1-1: Estimación Inmaculada Coma Tatay



Servicio Web de Información geoposicionada

- Experto nº2: Francisco Grimaldo Moreno

Nombre de Tarea	Tiempo pesimista	Tiempo más probable	Tiempo optimista	Tiempo Experto
Estado del Arte	35	25	15	25
Especificación del sistema	16	10	5	11
Planificación	5	5	5	5
Costes	5	5	5	5
Análisis				10
Definición del sistema	15	10	5	10
Establecimiento de los requisitos del cliente	25	15	10	16
Identificar casos de uso	25	15	10	16
Prototipo interfaz del usuario	35	25	15	25
Análisis de clases	20	15	10	15
Diseño				
Definición de la arquitectura	10	5	5	6
Diseño de las clases	15	10	5	10
Diseño de los datos	15	10	5	10
Diseño Web	75	50	25	50
Desarrollo				
Creación Base de datos	5	5	5	5
Toma de panorámicas 360 grados	25	15	10	16
Creación planos ETSE	35	25	15	25
Aplicación Servidor (REST)	75	50	35	52
Aplicación Cliente				
Codificación de las páginas	75	50	35	52
Estilos CSS	20	10	5	11
Codificación de los scripts	50	35	25	36
Pruebas				
Pruebas sistema	25	20	10	19
Pruebas de estrés	20	15	10	15
Pruebas de usuarios	35	25	10	24
Redacción de documentación	150	120	100	122

Figura 4.5.1-2: Estimación Francisco Grimaldo Moreno



Planificación y Metodología

- Experto Nº3: José Javier Samper Zapater

Nombre de Tarea	Tiempo pesimista	Tiempo más probable	Tiempo optimista	Tiempo Experto
Estado del Arte	40	30	25	31
Especificación del sistema	15	10	5	10
Planificación	10	5	5	6
Costes	5	5	5	5
Análisis				
Definición del sistema	25	20	15	20
Establecimiento de los requisitos del cliente	20	15	5	15
Identificar casos de uso	10	5	5	6
Prototipo interfaz del usuario	10	5	5	6
Ánálisis de clases	10	5	5	6
Diseño				
Definición de la arquitectura	20	15	10	15
Diseño de las clases	15	10	5	10
Diseño de los datos	15	10	5	10
Diseño Web	15	10	5	10
Desarrollo				
Creación Base de datos	50	40	30	40
Toma de panorámicas 360 grados	30	20	10	20
Creación planos ETSE	30	20	10	20
Aplicación Servidor (REST)	70	60	50	60
Aplicación Cliente				
Codificación de las páginas	25	20	15	20
Estilos CSS	25	20	15	20
Codificación de los scripts	20	15	10	15
Pruebas				
Pruebas sistema	25	20	10	19
Pruebas de estrés	15	10	5	10
Pruebas de usuarios	15	10	5	10
Redacción de documentación	160	140	120	140

Figura 4.5.1-3: Estimación José Javier Samper Zapater



Servicio Web de Información geoposicionada

- Experto nº4: Christian Sánchez León

Nombre de Tarea	Tiempo pesimista	Tiempo más probable	Tiempo optimista	Tiempo Experto
Estado del Arte	40	30	13	29
Especificación del sistema	18	13	7	13
Planificación	12	7	6	8
Costes	10	6	5	7
Análisis				
Definición del sistema	15	9	7	10
Establecimiento de los requisitos del cliente	24	13	11	15
Identificar casos de uso	22	15	10	15
Prototipo interfaz del usuario	33	27	12	26
Análisis de clases	22	18	11	18
Diseño				
Definición de la arquitectura	12	7	5	8
Diseño de las clases	15	11	7	11
Diseño de los datos	14	12	6	11
Diseño Web	73	50	23	49
Desarrollo				
Creación Base de datos	9	7	6	7
Toma de panorámicas 360 grados	15	10	6	10
Creación planos ETSE	29	20	10	20
Aplicación Servidor (REST)	80	55	38	56
Aplicación Cliente				
Codificación de las páginas	90	53	42	57
Estilos CSS	23	12	7	13
Codificación de los scripts	60	37	28	39
Pruebas				
Pruebas sistema	29	19	7	19
Pruebas de estrés	23	15	12	16
Pruebas de usuarios	27	22	10	21
Redacción de documentación	180	150	120	150

Figura 4.5.1-4: Estimación Christian Sánchez León



Planificación y Metodología

Experto nº5: Oscar Abella Abella

Nombre de Tarea	Tiempo pesimista	Tiempo más probable	Tiempo optimista	Tiempo Experto
Estado del Arte	60	34	16	35
Especificación del sistema	19	12	6	12
Planificación	10	6	5	7
Costes	15	10	8	11
Análisis				
Definición del sistema	13	8	7	9
Establecimiento de los requisitos del cliente	22	11	8	12
Identificar casos de uso	20	12	8	13
Prototipo interfaz del usuario	30	24	12	23
Ánálisis de clases	20	15	9	15
Diseño				
Definición de la arquitectura	10	8	6	8
Diseño de las clases	14	10	8	10
Diseño de los datos	16	13	7	13
Diseño Web	65	55	25	52
Desarrollo				
Creación Base de datos	12	6	4	7
Toma de panorámicas 360 grados	30	15	10	17
Creación planos ETSE	35	27	20	27
Aplicación Servidor (REST)	60	48	28	47
Aplicación Cliente				
Codificación de las páginas	65	50	40	51
Estilos CSS	20	16	8	15
Codificación de los scripts	55	35	25	37
Pruebas				
Pruebas sistema	20	15	10	15
Pruebas de estrés	28	20	15	21
Pruebas de usuarios	35	25	10	24
Redacción de documentación	170	130	110	137

Figura 4.5.1-5: Estimación Oscar Abella Abella



4.5.1.1 Distribución de pesos

Aunque todos son considerados expertos en el campo del desarrollo web y tienen experiencia acreditada en este tipo de proyectos, existen diversas valoraciones a tener en cuenta que hacen que debamos considerar unas opiniones por encima de otras a la hora de planificar. Por esta razón realizaremos una ponderación justa en la medida que las opiniones contribuyen a nuestra estimación en relación al grado de experiencia invertida en proyectos de similar envergadura.

a. Profesores de Universidad

Puesto que el personal docente es un elemento con acreditada experiencia en investigación y desarrollo, tiene un grado de entendimiento del proyecto muy sólido que le hace estar muy próximo a la estimación adecuada de tiempo puesto que evalúan y tutorizan trabajos similares continuamente.

b. Analista Programador

Este perfil es de gran consideración puesto que desarrolla su actividad profesional de una manera continuada y en perpetuo contacto con la tecnología web. Tiene experiencia demostrada en desarrollo y, aunque no realiza tareas de investigación o tutorización, se vale de su experiencia previa de desarrollos similares para estimar tiempos acordes al nuevo proyecto.

c. Programador Freelance

Perfil que se debe tener en cuenta el estado del mercado en cuanto a funcionalidad y diseño a fin de desarrollar su actividad profesional de la manera más actual posible integrando los usos y tendencias más demandadas en sus proyectos venideros. Es personal que, sin una actividad investigadora o analista, conocen el estado del mercado y cuál es su evolución natural.

Explicados los tres perfiles que disponemos, trataremos de ponderarlos a fin de acomodarlos en una proporción que demuestre su peso en el proyecto. La ecuación matemática que nos dará la contribución de cada perfil es la siguiente:

$$\begin{aligned} \text{Estimación} = & \text{ Personal docente} \cdot 75\% + \text{Analista} \cdot 15\% \\ & + \text{Programador Freelance} \cdot 10\% \end{aligned}$$

Personal docente - 75%

- Inmaculada coma (25%)
- Francisco Grimaldo (25%)
- José Javier Samper Zapater (25%)

Analista - 15%

- Oscar Abella (15%)

Programador Freelance - 10%

- Christian Sánchez León (10%)



Planificación y Metodología

Realizados todos los cálculos anteriormente descritos incluyendo los pesos específicos de cada experto, obtenemos el siguiente cuadro de tiempos estimados.

Nombre de Tarea	Tiempo estimado
Estado del Arte	29
Especificación del sistema	11
Planificación	8
Costes	8
Análisis	
Definición del sistema	13
Establecimiento de los requisitos del cliente	14
Identificar casos de uso	11
Prototipo interfaz del usuario	17
Análisis de clases	12
Diseño	
Definición de la arquitectura	10
Diseño de las clases	10
Diseño de los datos	11
Diseño Web	30
Desarrollo	
Creación Base de datos	19
Toma de panorámicas 360 grados	17
Creación planos ETSE	26
Aplicación Servidor (REST)	49
Aplicación Cliente	
Codificación de las páginas	40
Estilos CSS	20
Codificación de los scripts	31
Pruebas	
Pruebas sistema	16
Pruebas de estrés	14
Pruebas de usuarios	17
Redacción de documentación	134

Figura 4.5.1-6: Estimación resultado

4.5.2 Estimación de costes de personal

En esta parte de la planificación se definen las personas que intervienen en el proyecto. Para ellos hay que especificar la posición dentro de la organización y la especialidad. Para proyectos relativamente pequeños, una persona-año o menos puede llevar a cabo todos los pasos de la ingeniería del software.



Servicio Web de Información geoposicionada

En nuestro caso estimaremos los costes en función de la media del sueldo mensual de los ingenieros⁷. Si ponemos un sueldo mensual de 1300€, una duración de 7 meses y aplicando el método Juicio por expertos de estimación de costes, nos ha dado que el coste de personal del proyecto nos da un sueldo total de 9100 euros.

4.5.3 Estimación de recursos Software/Hardware

A continuación, mostramos los distintos costes que hacen referencia al Software y al Hardware utilizado durante el proceso de desarrollo.

Recursos Software	Precio
Microsoft Windows 7 Home 64bits	150€
Adobe Photoshop CC	144€
AutoCAD 2016	250€
Maptiler Start	29€
Netbeans 8.1	0€
Oracle Database 11g Express Edition	0€
Glassfish 4.0	0€
TOTAL	573 €

Figura 4.5.3-1: Estimación de costes de herramientas software utilizados

Recursos Hardware	Precio
PC Intel Core i7 950 3.07GHz 64bits, 4Gb RAM, 500Gb disco duro, Monitor TFT 22'	750€
Ricoh Theta S	372€
TOTAL	1122 €

Figura 4.5.3-2: Estimación de costes de herramientas hardware utilizados

⁷ Tablas salariales para ingeniería extraídas de CC. OO publicadas en 2011.



Planificación y Metodología

4.5.4 Estudio de viabilidad económica y legal

La viabilidad de un proyecto es la fase que culmina con la decisión de entrar en el desarrollo del mismo o bien abandonar. Para dicho estudio tendremos en cuenta la viabilidad legal y económica.

4.5.4.1 Viabilidad legal

Dentro del marco de un proyecto es importante estudiar la viabilidad legal, ya que podríamos estar fuera de la ley y, por tanto, nuestro proyecto no podría seguir adelante.

En este caso estamos en posesión de información privada que pertenece a l'Escola Tècnica Superior d'Enginyeria de la Universitat de València, desde datos sensibles del personal docente hasta planos técnicos detallados de las distintas plantas del edificio. Puesto que se trata de una aplicación web creada para la Universitat de València, basta con tener una autorización de tratamiento de datos y que, como es lógico, poseemos.

Por ello, debido a que no existe ningún requisito legal que haga peligrar nuestro proyecto, debemos comprobar si el factor económico no hace inviable nuestro desarrollo.

4.5.4.2 Estudio Económico

Llegado a este punto podría ocurrir que, aunque el coste económico no fuese muy elevado, no tuviésemos suficiente presupuesto para abordar el proyecto, por lo que deberíamos abandonar el proyecto.

Elementos económicos del proyecto	Precio
Coste de personal	9.100€
Recursos Software	573€
Recursos Hardware	1.122€
TOTAL	10.795 €

Figura 4.5.4-1: Estimación económica del sistema



Desarrollo

5 Desarrollo

5.1 Análisis del Sistema

El análisis del sistema determina los requisitos del mismo y debe especificar perfectamente las funcionalidades del sistema. Se encuentra en un nivel de abstracción lejano de la implementación. Plasma las necesidades planteadas por el cliente de forma clara para todas las partes implicadas en el proyecto.

5.1.1 Actores del sistema

Para focalizar las demandas de funcionalidades debemos definir los actores que compondrán los marcos de actuación de las mismas. Estos actores son las entidades externas al sistema que guardan algún tipo de relación con la funcionalidad del sistema.

En el caso de los seres humanos, se pueden ver a los actores como definiciones de rol, por lo que un mismo individuo puede corresponder a uno o más actores.

Debido a que es una aplicación de sólo lectura de información, es decir, el usuario final no va agregar, borrar o modificar recursos de nuestra aplicación web y no existen restricciones de autenticación para accederlos, nuestro usuario sólo dispondrá de la posibilidad de realizar consultas a nuestros recursos y navegar en base a esta idea. Por esta razón sólo dispondremos de un único actor en nuestro desarrollo inicial.

En futuras versiones más avanzadas se puede plantear la inclusión de dos roles con mayores permisos. Uno con la capacidad de consultar recursos con el valor de *inaccesible*, previa autenticación en el sistema. Y otro con la capacidad de alterar los registros de la base de datos que se consultan a fin de lograr un mantenimiento y actualización de la información contenida en los mismos.

5.1.2 Requisitos del usuario

Para ver mejor la forma en que el usuario se relaciona con el sistema, debemos describir caso por caso todas las opciones de interacción que se presentan. Utilizaremos un diagrama de casos de usos, que representa la forma en como un Cliente (Actor) opera con el sistema, para obtener una visión más gráfica de lo explicado:

5.1.2.1 Acceso al localizador

Los usuarios tendrán la posibilidad, mediante un interfaz Web, de seleccionar el recurso específico para la búsqueda.

Las tres opciones de búsqueda (Personal, Asignaturas, Espacios) son independientes. Además, existe una cuarta opción que engloba las tres anteriores y sólo se podrá realizar una consulta a la vez, es decir, no podemos buscar un profesor y asignatura simultáneamente.

El sistema podrá mostrar sugerencias de búsqueda “al vuelo” que se aproximen al texto introducido por el usuario.



Servicio Web de Información geoposicionada

Los usuarios, tras validar la búsqueda, podrán recibir un mensaje informativo si los datos no son correctos. Si son correctos se realizará la consulta para que devuelva los correspondientes resultados.

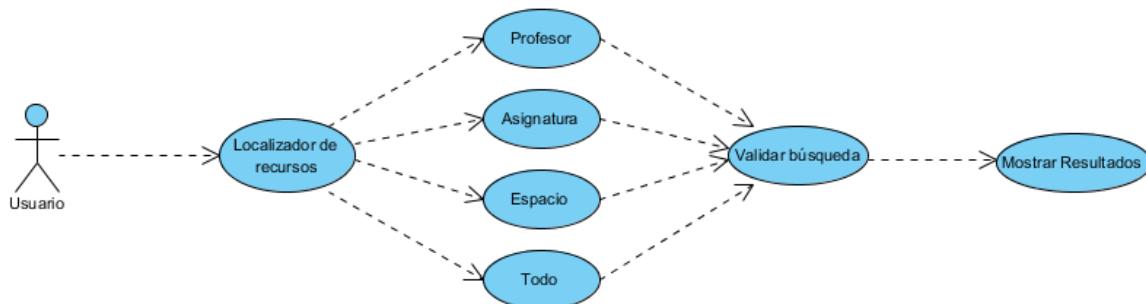


Figura 5.1.2-1: Caso de uso acceso al localizador

5.1.2.2 Configurador de capas

Los usuarios tendrán la posibilidad, mediante un interfaz Web, de seleccionar las distintas representaciones de planos (denominación/código) o color (color/básico), proveedores de mapas (Mapbox/Google Maps) y la visualización de iconos.

Cada uno de los grupos de opciones, explicadas anteriormente, es excluyente. Es decir, la activación de una opción dentro de una sección implica la cancelación de la otra representación.

Como valores por defecto al cargar la imagen se configurará la representación mediante plano básico con descripción y con la visualización de iconos activada. Desde ese momento el Usuario podrá alterarlo a su gusto.

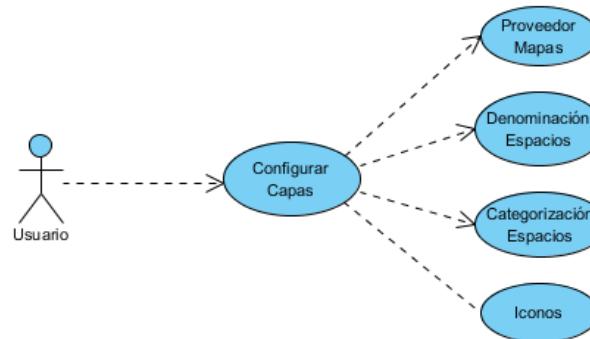


Figura 5.1.2-2: Caso de uso configurar mapa

Desarrollo

5.1.2.3 Buscar edificios

Los usuarios tendrán la posibilidad, mediante un interfaz Web, de seleccionar y mostrar el emplazamiento de cualquier edificio de un listado de las distintas ubicaciones que conforman la Universitat de València.

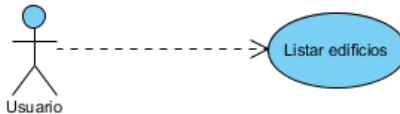


Figura 5.1.2-3: Caso de uso listar edificios

5.1.2.4 GPS

Los usuarios tendrán la posibilidad de mostrar la localización GPS actual que proporcione información sobre la posición del mismo sobre el mapa.

El sistema pondrá una marca en el mapa que represente la posición actual GPS del usuario cuando esta opción sea activada y removida del mapa cuando sea desactivada.

El sistema ajustará la visión del mapa para mostrar la posición actual y la de GPS dentro del mismo campo de visión.

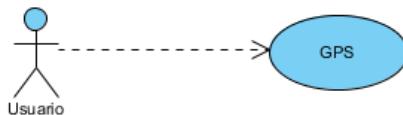


Figura 5.1.2-4: Caso de uso posición GPS

5.1.2.5 Acceso a la ayuda

Los usuarios podrán obtener ayuda acerca del manejo de la aplicación y sus opciones a fin de que el usuario pueda resolver cualquier duda en cuanto al uso del sistema.

El sistema mostrará la información de ayuda mediante una ventana modal que permita no perder la ubicación durante la navegación.

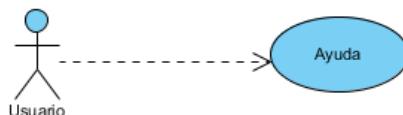


Figura 5.1.2-5: Caso de uso acceso a la ayuda

5.1.2.6 Interacción Mapa

Se mostrarán iconos sobre el mapa pertenecientes a las distintas facultades y campus de la Universitat de València.

Al pulsar sobre cualquiera de los iconos se mostrará información adicional sobre la zona marcada. En el caso de las facultades, se proporcionará nombre, teléfono, correo y enlace web de la facultad marcada. En el caso de los campus, se mostrará el nombre del campus y todos aquellos edificios docentes que pertenezcan a él.



Servicio Web de Información geoposicionada

El Sistema permitirá cambiar los distintos niveles de mapas mediante la pulsación del botón correspondiente al piso por parte del usuario. Aquel piso que se encuentre activo mostrará un aspecto pulsado y aquel piso que albergue un espacio activo como resultado de una búsqueda presentará un aspecto distintivo de su estado para informar al usuario.

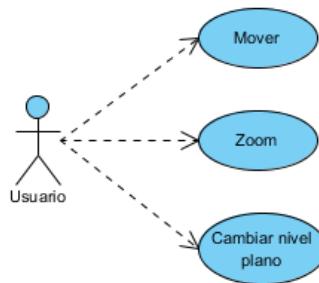


Figura 5.1.2-6: Caso de uso interacción mapa

5.1.2.7 Mostrar Resultados de Búsqueda

Se mostrarán los resultados de la búsqueda de profesores o espacios mediante una rejilla dividida por secciones.

El sistema permitirá crear un enlace que se pueda copiar al portapapeles, de tal manera que se pueda acceder de forma directa al recurso.

El sistema permitirá compartir en redes sociales Facebook⁸ y Twitter⁹ el enlace al recurso, de tal manera que se pueda acceder de forma directa a la búsqueda.

El sistema permitirá enviar directamente un correo al profesor mediante el gestor predeterminado que tenga configurado el usuario.

El sistema recuperará la información relativa a panoramas de 360 grados colocando un ícono interactivo representativo en la ficha.

Se mostrará un botón marcador en la ficha de resultados que permita ocultar y recuperar la misma sin perder la búsqueda efectuada.

El sistema centrará el mapa en la ubicación recuperada por la búsqueda y volverá a la misma posición cada vez que se pulse el botón de ubicación interactivo sobre la ficha.

El sistema creará un perímetro de color rojo translúcido que delimitará la zona o espacio resultado de la búsqueda del usuario el cual se mantendrá activo sólo en el piso del que se obtuvo la información.

⁸ Véase *Facebook for developers* en la Bibliografía

⁹ Véase *Twitter Documentation* en la Bibliografía

Desarrollo

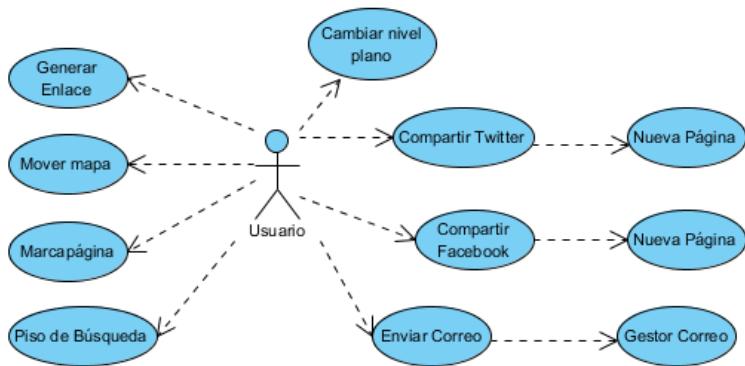


Figura 5.1.2-7: Caso de uso mostrar resultados

5.1.2.8 Visualización de panoramas de 360 grados

Se mostrará un visor donde se presentarán las imágenes panorámicas de 360 grados que pertenezcan al espacio resultado de la búsqueda efectuada por el usuario.

El sistema mostrará pestañas adicionales en base al número de panorámicas que se hayan recuperado del espacio para que el usuario pueda cambiar entre las distintas imágenes de la misma ubicación. Por defecto, el sistema mostrará en el visor la primera imagen recuperada por los resultados de búsqueda.

El sistema permitirá activar el modo pantalla completa ajustando el visor al tamaño del dispositivo y ocultando el resto de elementos. Este proceso será reversible volviendo sobre los pasos antes descritos.

El sistema ajustará el tamaño del visor a las medidas del dispositivo guardando la relación de aspecto de las imágenes panorámicas sin sufrir deformación.

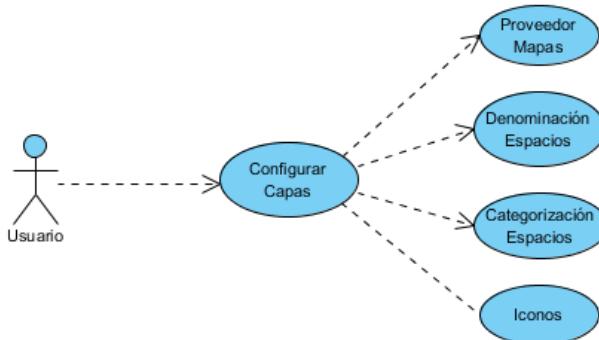


Figura 5.1.2-8: Caso de uso visor imágenes 360 grados



Servicio Web de Información geoposicionada

5.1.3 Diagrama de secuencia

5.1.3.1 Buscador de profesores

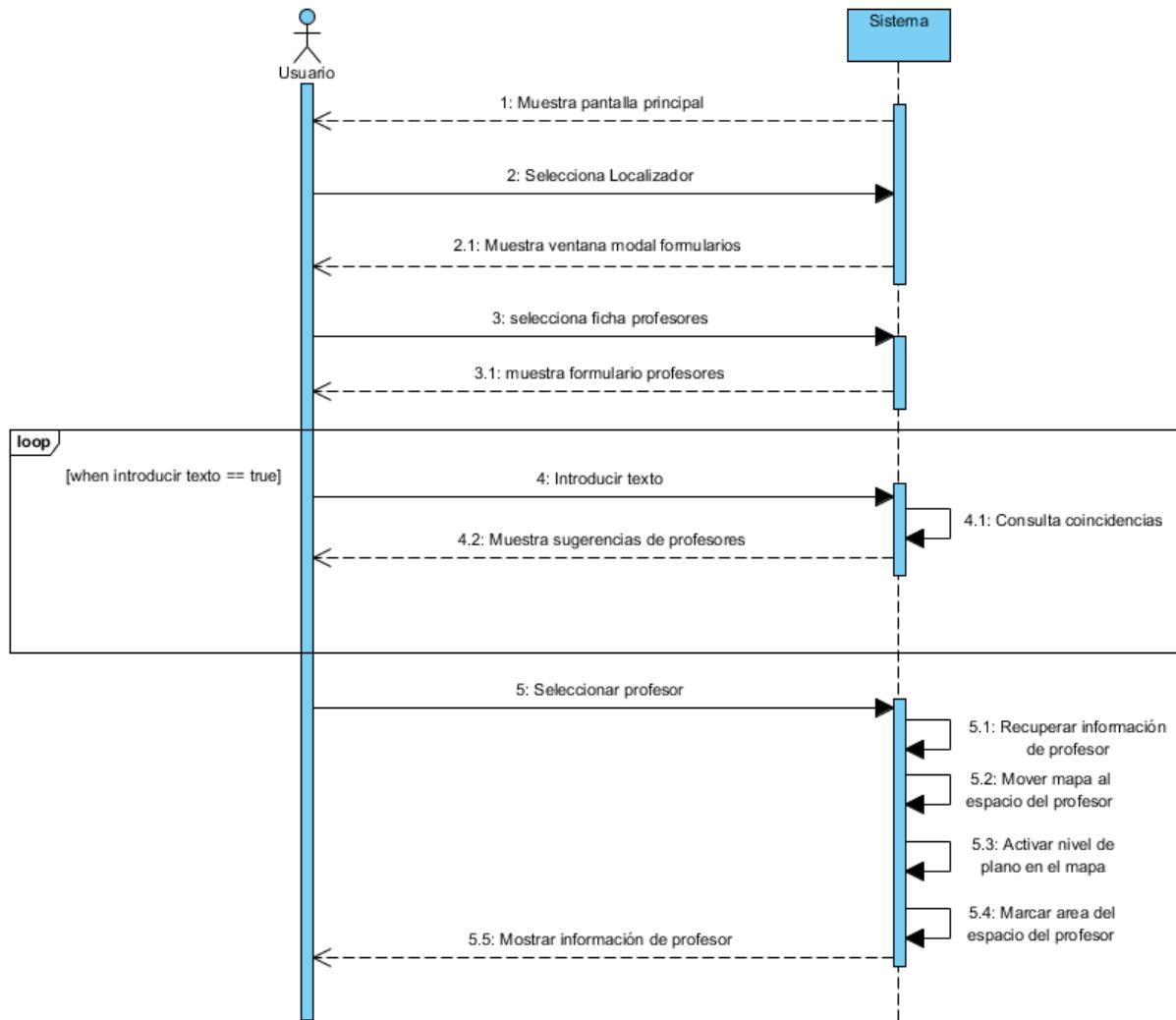


Figura 5.1.3-1: Diagrama de secuencia del buscador de profesores

Desarrollo

5.1.3.2 Buscador de profesores por asignaturas

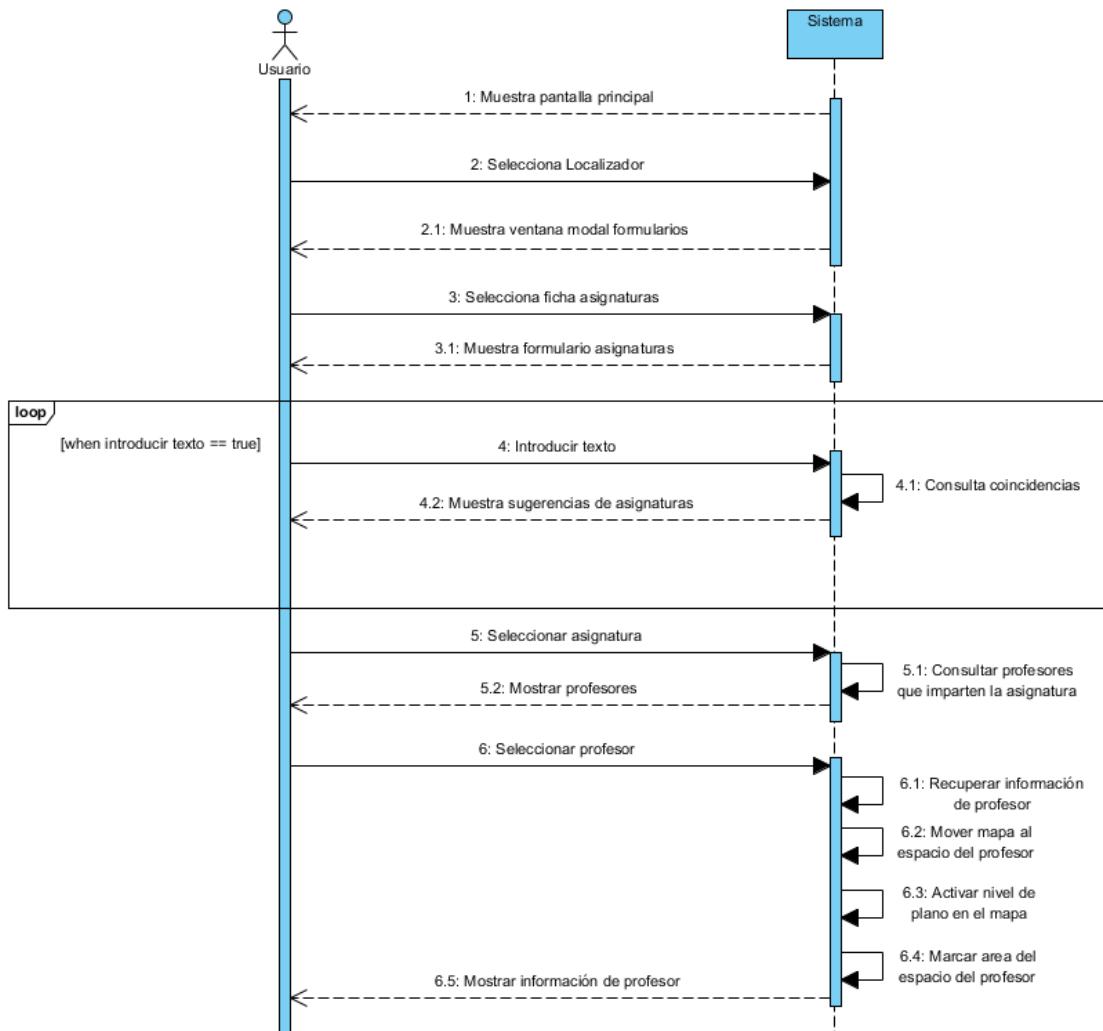


Figura 5.1.3-2: Diagrama de secuencia del buscador de profesores por asignaturas



Servicio Web de Información geoposicionada

5.1.3.3 Buscador de espacios

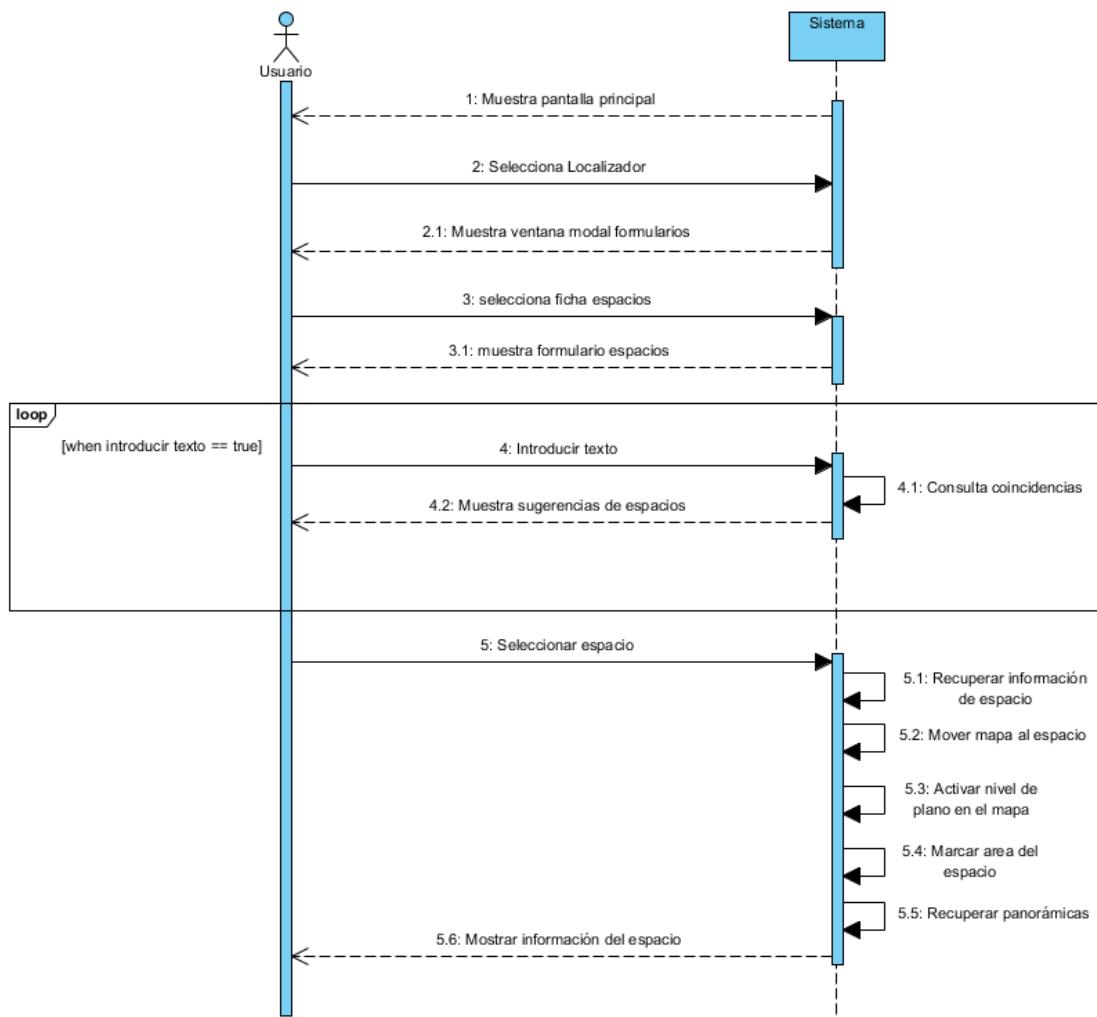


Figura 5.1.3-3: Diagrama de secuencia de buscador de espacios

Desarrollo

5.1.3.4 Visualizar panorámicas de 360 grados

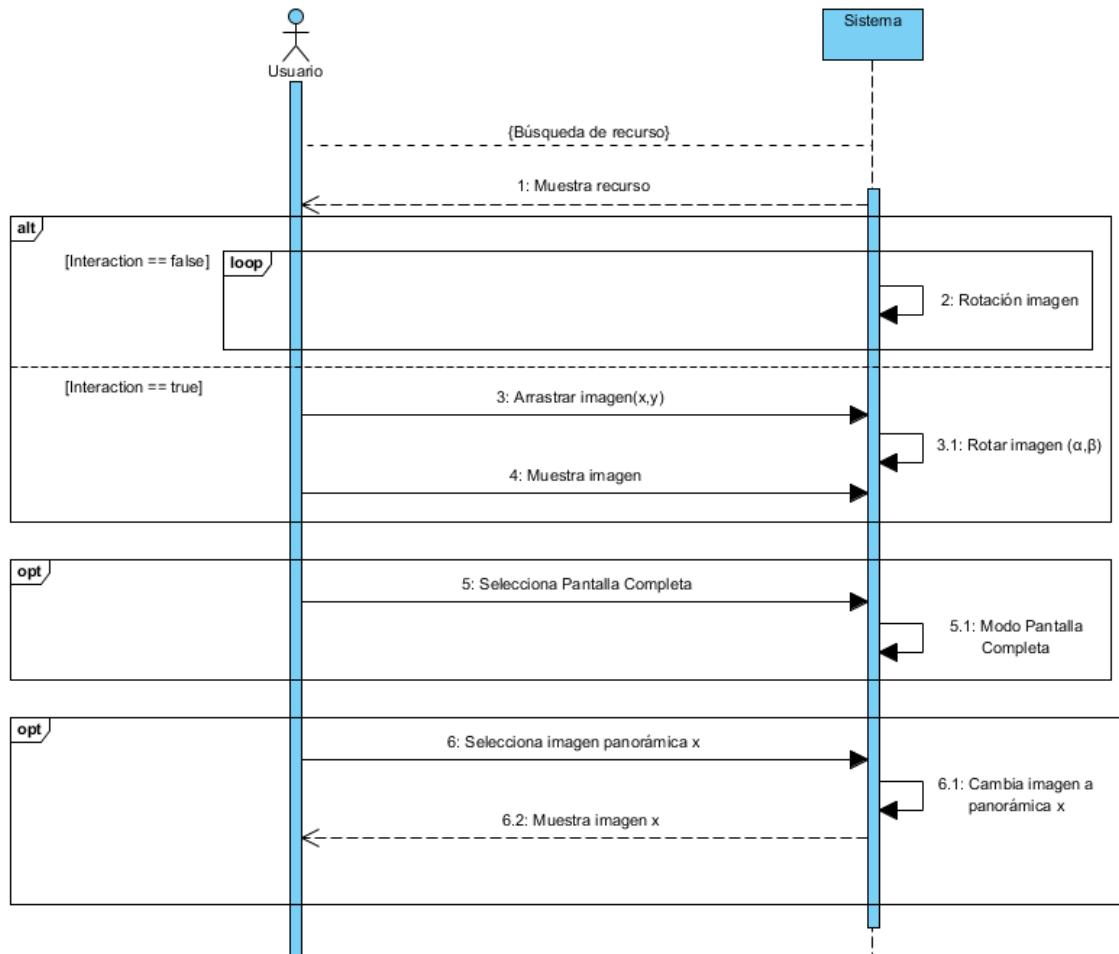


Figura 5.1.3-4: Diagrama secuencia visualizar panorámica de 360 grados



5.1.3.5 Compartir en redes sociales

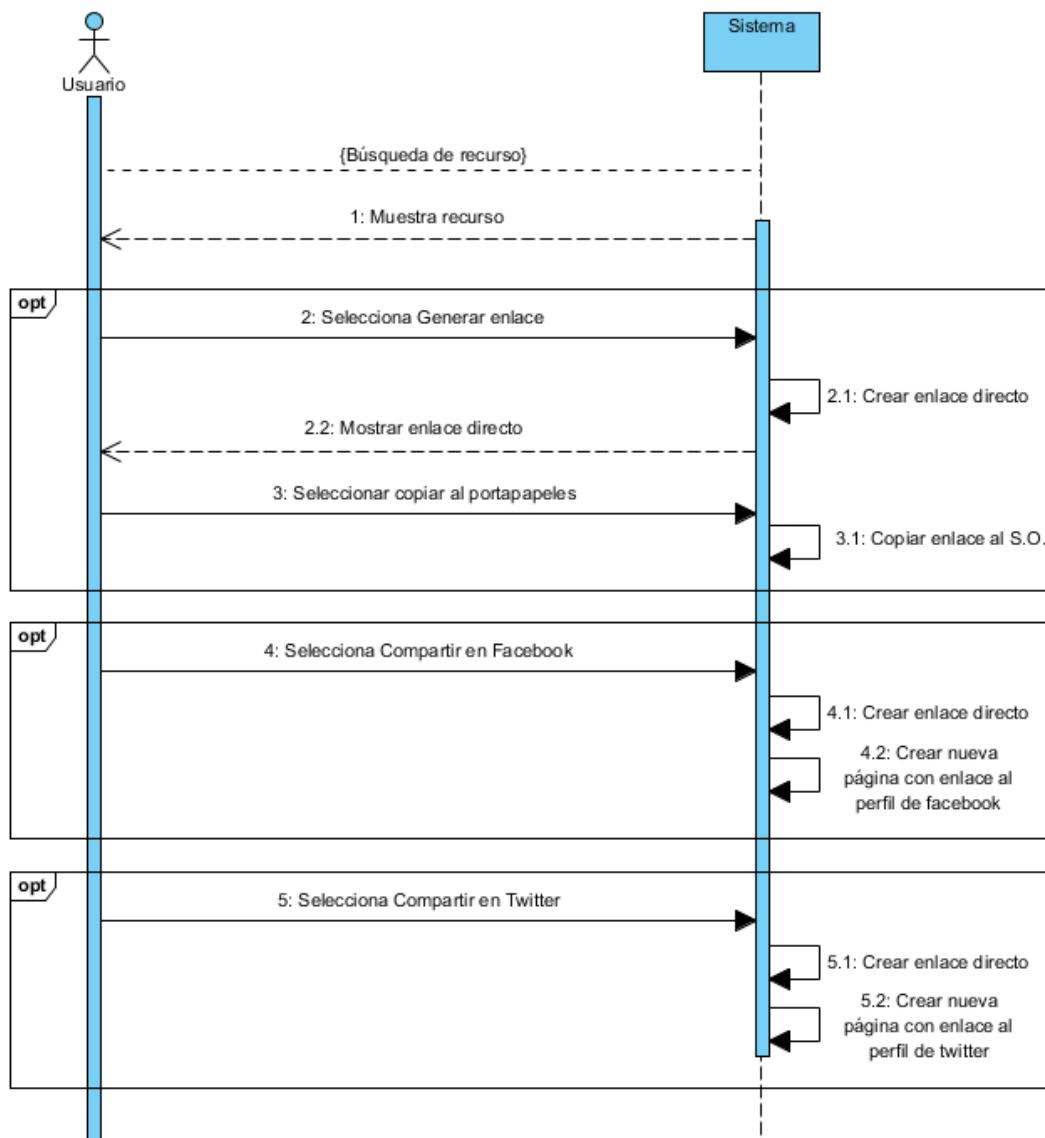


Figura 5.1.3-5: Diagrama de secuencia de compartir en redes sociales

Desarrollo

5.1.3.6 Configurar capas del mapa

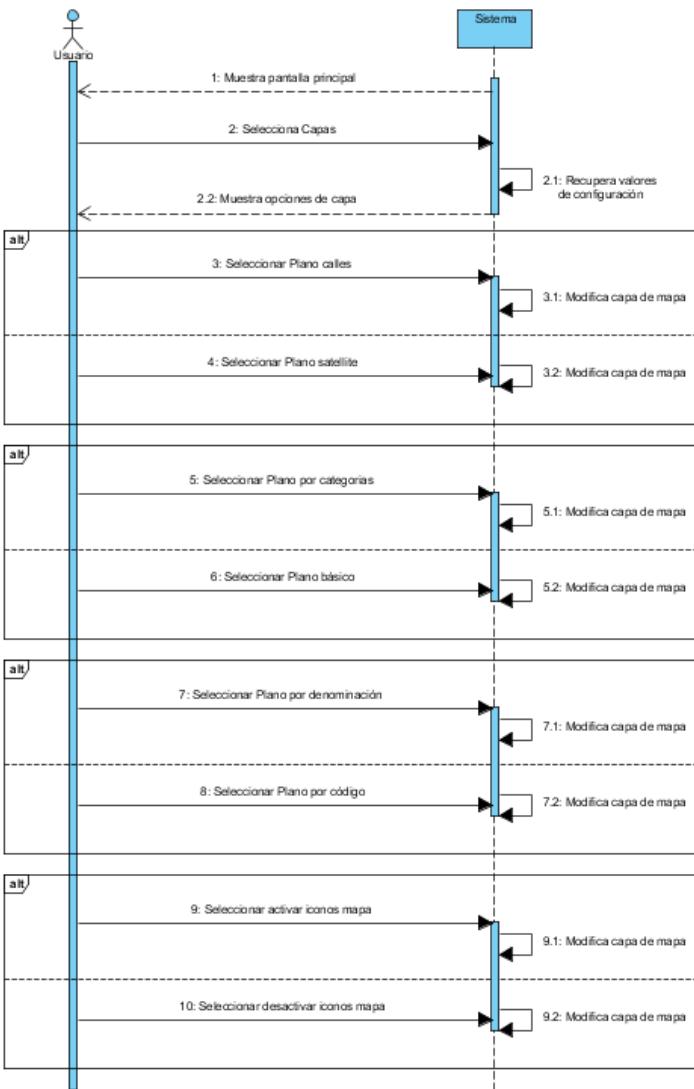


Figura 5.1.3-6: Diagrama secuencia configurar capas del mapa



Servicio Web de Información geoposicionada

5.1.3.7 Buscar Edificio

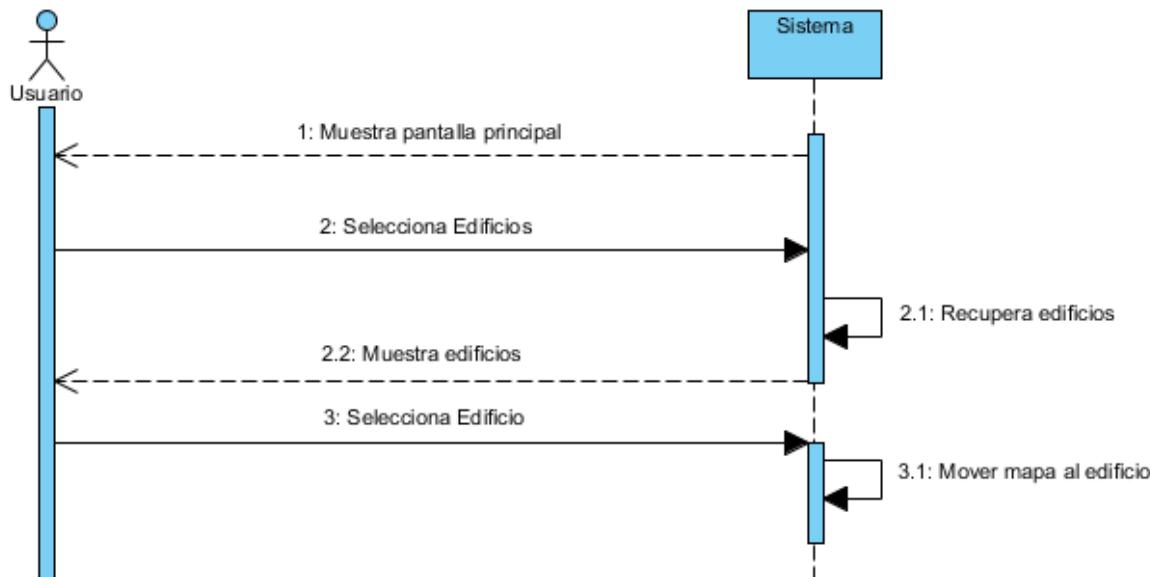


Figura 5.1.3-7: Diagrama de secuencia buscar edificio

5.1.4 Modelo de datos

El modelo de datos estudia los datos que intervienen en el sistema independientemente de los procesos que los gestionan.

Un objeto de datos es una representación de cualquier composición de información compuesta que debe comprender el software. Por composición de información entendemos todo aquello que tiene un número de propiedades o campos diferentes.

Necesitaremos un diagrama del modelo conceptual que muestre la relación de los objetos que intervienen en el sistema.

Antes de representar el diagrama vamos a describir los objetos de datos que forman parte del sistema en función del análisis de requisitos.

5.1.4.1 Objetos de datos en la Base de datos

La base de datos llamada *SigUV* tendrá estructuras de datos llamadas tablas, las cuales pasamos a describir cada una de ellas:

Tablas coordenadas

En ella estarán las coordenadas de los distintos espacios.

- Identificador único de coordenada.
- Descripción de la coordenada.
- Latitud de la coordenada.
- Longitud de la coordenada.



Desarrollo

Tabla edificios

Poseerá los edificios pertenecientes al Campus Burjassot-Paterna de la Universitat de València.

- Identificador único de edificio.
 - Nombre en Castellano.
 - Nombre en Valenciano.
 - Nombre en Inglés.
 - Dirección.
 - Teléfono.
 - Enlace web.
 - Enlace chano UV.
 - Identificador de coordenada.

Tabla espacios

Será la tabla que albergará toda la información referente a los espacios de la ETSE.

- Identificador único de espacio
 - Nombre.
 - Descripción.
 - Descripción en Valenciano.
 - Descripción en Inglés.
 - Tipo.
 - Bloque.
 - Piso.
 - Visibilidad en búsquedas.
 - Identificador de edificio.
 - Identificador de coordenada.
 - Bounding box del área del espacio.

Tabla panoramas

Dispondrá todos los elementos de panorámicas o fotografías esféricas 360 de los espacios.

- Identificador de espacio.
 - Identificador de panorama.
 - Enlace panorámica.

Tabla profesores

Contendrá toda la información referente a los profesores:

- Identificador único de profesor.
 - Nombre y Apellidos.
 - Área de conocimiento.
 - Web personal.
 - Correo Electrónico.
 - Horario de tutorías.
 - Visibilidad en búsquedas.
 - Identificador de espacio.



Servicio Web de Información geoposicionada

Tabla asignaturas

En ella encontraremos las asignaturas que se imparten en la ETSE. Estará estructurada de la siguiente manera:

- Código identificador de asignatura.
- Nombre en Castellano.
- Nombre en Valenciano.
- Nombre en Inglés.

Tabla profesorasignaturas

Será la tabla que albergará las relaciones entre profesores y asignaturas:

- Identificador de profesor.
- Identificador de asignatura.
- Relación del profesor con la asignatura.

5.1.4.2 *Objetos de datos del Sistema Servidor*

En la parte del servidor crearemos los siguientes objetos:

Objeto Coordenadas

Tendrá los siguientes componentes:

- IdCoordinada, Descripción, Latitud, Longitud.
- NamedQueries
- Getters y Setters

Objeto Edificios

Contará con los siguientes componentes:

- IdEdificio, Nombre, Nombre_ca, Nombre_en, Dirección, Teléfono, Enlace web, enlace chano UV, IdCoordinada.
- NamedQueries
- Getters y Setters

Objeto Espacios

Constará de los siguientes elementos:

- IdEspacio, Nombre, Descripción, Descripcion_ca, Descripcion_en, Tipo, Bloque, Piso, Visibilidad en búsquedas, IdEdificio, IdCoordinada, Bounding box.
- NamedQueries.
- Getters y Setters.

Objeto Panoramas

Dispone de los siguientes componentes:

- IdEspacio, IdPanorama, Enlace panorámica.
- NamedQueries.
- Getters y Setters.

Objeto Profesores

Estará compuesto por los siguientes elementos:

- IdProfesor, Nombre, Área de conocimiento, Web personal, Correo Electrónico, Tutorías, Visibilidad, IdEspacio.
- NamedQueries.
- Getters y Setters.



Desarrollo

Objeto Asignaturas

Dispondrá de los siguientes componentes:

- IdAsignatura, Nombre, Nombre_ca, Nombre_en.
- NamedQueries.
- Getters y Setters.

Objeto ProfesorAsignatura

Tendrá los siguientes elementos:

- IdProfesor, IdAsignatura, Relación.
- NamedQueries.
- Getters y Setters.

5.1.4.3 *Objetos de datos del Sistema Cliente*

En la parte del cliente crearemos los siguientes Objetos:

Objeto Webservice

Dispone de los siguientes componentes:

- Parámetros recogidos por la selección de usuario: Edificio, Panorama, Profesor, Asignatura, Coordenada o Espacio.
- Url donde se encuentra el recurso

Objeto Buscador

Tendrá los siguientes componentes:

- Parámetros introducidos por el usuario: Profesor, Espacio o Asignatura.
- Lista de Profesores, Espacios y Asignaturas
- Objeto Webservice que usará para reclamar los datos del API Restful.

Objeto Panorama

Contará con los siguientes componentes:

- Parámetro recogido por la selección de usuario: Lista de Panoramas.
- Objeto Webservice para consultar con la API Restful la petición.

Objeto Polígono

Constará de los siguientes elementos:

- Parámetro recogido por la selección de usuario: Coordenadas
- Objeto Webservice para consultar en la API Restful la petición.

Objeto Mapa

Estará compuesto de los siguientes elementos:

- Parámetros recogidos por la selección del usuario: Coordenada
- Nivel, Zoom, Fondo, Tema, Topónimo, Área, Token de Acceso Mapbox, Token de Acceso Google Maps, Url donde se encuentran los mapas.



Servicio Web de Información geoposicionada

Una vez descritos los objetos del Sistema y los de la Base de Datos comentaremos como se interrelacionan.

Desde el momento en el que la página del navegador carga por completo la aplicación, se realizan dos acciones:

Por un lado, el objeto *mapa* verifica si ha sido invocado por defecto o si hay una petición de recurso directo. Si nos encontramos con una invocación por defecto, el objeto se instanciará con valores precargados en la aplicación. Por el contrario, si estamos delante de una solicitud de recurso, se llamará al objeto *webservice* para que efectúe la petición del profesor o el espacio concreto que instancia los valores del mapa. Por otro lado, el objeto *webservice* realiza una petición al servicio *edificios* y carga los mismos en la barra lateral oculta.

A partir de esta situación, dependiendo de la actuación del usuario pueden ocurrir los siguientes eventos:

- Si el usuario ha desplegado la barra lateral y selecciona uno de los edificios, el objeto *mapa* cargará las coordenadas asociadas al mismo
- Si el usuario selecciona un piso, el objeto *mapa* cargará la ruta asociada al plano elegido.
- Si el usuario selecciona una representación de mapa, el objeto *mapa* cargará la ruta asociada a la representación escogida.
- Si se selecciona el buscador, el objeto *buscador* llamará al objeto *webservice* que realizará tres peticiones de servicios. El servicio *profesores* que cargará los nombres de los profesores presentes en la base de datos. El servicio *asignaturas* que cargará las asignaturas y finalmente el servicio *espacios* que contendrá todas las referencias de espacios almacenados.

Tras cargar el buscador y estar operativo para realizar las consultas y sugerencias se realizarán las siguientes acciones según la elección del usuario:

- Si el usuario selecciona un profesor, el objeto *buscador* llamará al objeto *webservice* que realizará la petición del profesor al servicio *profesor* y devolverá su información. Posteriormente, se invocará al objeto *mapa* pasándole los valores de coordenadas sobre su ubicación.
- Si el usuario selecciona una asignatura, el objeto *buscador* llamará al objeto *webservice* que realizará la petición al servicio *asignaturas*, devolviendo la relación de profesores sujetos a la materia. Las siguientes interacciones se producen de igual forma a cuando se selecciona un profesor.
- Si el usuario selecciona un espacio, el objeto *buscador* llamará al objeto *webservice* que realizará la petición del espacio al servicio *espacios* y devolverá su información. Posteriormente se invocará al objeto *mapa* pasándole los valores de coordenadas sobre su ubicación.

Desarrollo

Una vez se accede a la información concreta de un profesor o espacio se realizarán las siguientes acciones según la elección del usuario:

- Si el usuario selecciona *compartir URL*, el objeto *webservice* realizará una petición de obtener la dirección de enlace.
- Si el usuario selecciona *compartir en Facebook*, el objeto *webservice* realizará una petición de obtener la dirección de enlace y abrirá una nueva ventana que cargará la web de Facebook.
- Si el usuario selecciona *compartir en Twitter*, el objeto *webservice* realizará una petición de obtener la dirección de enlace y abrirá una nueva ventana que cargará la web de Twitter.
- Si el usuario selecciona *panorámicas 360 grados*, el objeto *webservice* realizará una petición al servicio *Panoramitas* que nos devolverá *las imágenes panorámicas relacionadas con el espacio*.

5.1.5 Modelo de procesos

En este punto del análisis tenemos que proporcionar una indicación de cómo se transforman los datos a medida que avanza el sistema y representar las funciones que transforman el flujo de datos.

A medida que la información se mueve a través del software, es modificada por una serie de transformaciones. El diagrama de flujo de datos es una técnica que representa del flujo de la información y las transformaciones que se aplican a los datos al moverse desde la entrada hasta la salida.

5.1.5.1 Sistema Global

En el primer nivel de proceso nos encontramos con el flujo de datos de nivel 0 que representa el sistema como un todo. Tenemos únicamente una entidad externa que representa al usuario que interactúa con la aplicación. Es el único productor o consumidor de información que reside fuera de los límites de la aplicación a ser modelado.

El usuario accede al sistema interactuando con un formulario para configurar la búsqueda y el sistema devuelve los resultados de la misma.

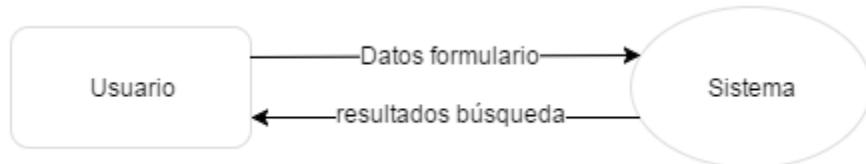


Figura 5.1.5-1: Diagrama de flujo de datos entrada y salida del sistema



Servicio Web de Información geoposicionada

5.1.5.2 Definición de niveles del sistema

Para jerarquizar de alguna manera el flujo de datos que pasa desde que se carga la página hasta que el usuario recibe los resultados de su búsqueda.

a. Nivel 1, Carga de la página:

En este nivel el sistema accede a la API Restful para obtener todos los edificios, asignaturas, profesores y espacios. Una vez realizada la acción, se mostrará el mapa en el navegador.

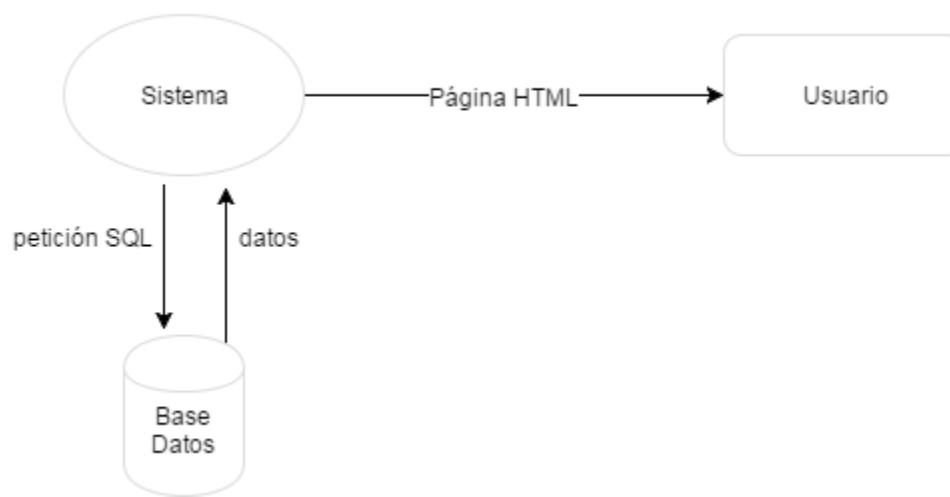


Figura 5.1.5-2: Diagrama de flujo de datos carga de página

b. Nivel 2, Interacción del usuario con el formulario:

En este nivel el usuario, mediante un interfaz, accede al tipo de información que necesita buscar: profesor, asignatura o espacio. Una vez decidido qué tipo de búsqueda va a realizar y confirmada la búsqueda, se validan los datos y, si son correctos, se inicia el proceso de búsqueda.

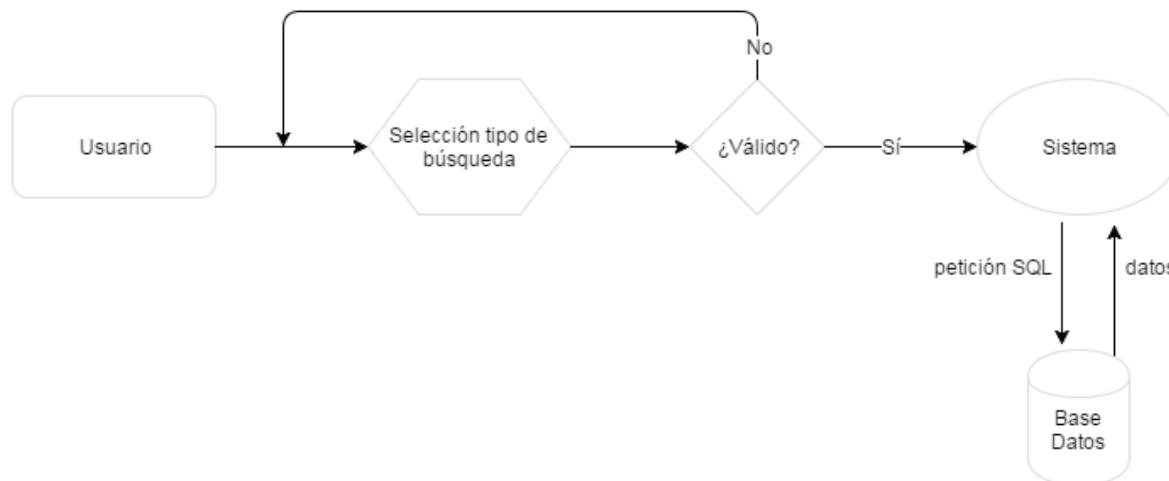


Figura 5.1.5-3: Diagrama de flujo de datos usuario-formulario

Desarrollo

c. Nivel 3, Proceso de búsqueda y muestra de resultados:
 En este nivel el sistema analiza la petición del usuario, envía la consulta hacia la base de datos y acaba mostrando la información.



Figura 5.1.5-4: Diagrama de flujo de datos búsqueda y muestra de resultados

d. Nivel 4, Acceso a otros recursos:
 En este nivel el usuario tendrá la posibilidad de acceder a otros recursos que le ofrece la página de resultados:



Figura 5.1.5-5: Diagrama de flujo de datos de acceso a otros recursos

Desarrollo

5.2 Diseño del Sistema

En esta fase vamos a concretar lo que hace cada uno de los procesos que tienen lugar en el sistema. Se tomarán decisiones acerca de las estructuras de datos, de control y de interfaz pertinentes a nuestra aplicación.

Para el desarrollo de esta fase haremos uso del diseño de datos y de interfaz.

5.2.1 Diseño de datos

En este apartado del diseño de la aplicación se van a transformar el modelo del dominio de la información, creado durante la fase de análisis, en las estructuras de datos necesarias para implementar el software. A continuación, pasaremos a diseñar cada uno de los objetos de datos.

5.2.1.1 Diagrama relacional de la Base de datos

Después de exponer los distintos campos y restricciones de nuestra base de datos, vamos a mostrar un diagrama general que nos permita visualizar las distintas relaciones entre las tablas. Este diagrama se muestra en la siguiente figura:

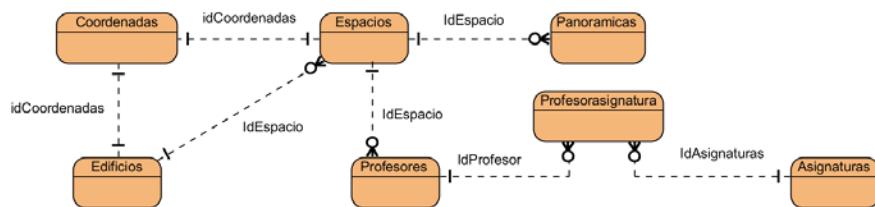


Figura 5.2.1-1: Diagrama Relacional

5.2.1.2 Diseño de datos en la Base de datos

A continuación describiremos todas las tablas que forman la base de datos *SigUV*.

a. Tabla Coordenadas

ABUSPAZ.COORDENADAS	
P	* IDCOORDENADA VARCHAR2 (20 BYTE)
	DESCRIPCION VARCHAR2 (40 BYTE)
*	LATITUD VARCHAR2 (20 BYTE)
*	LONGITUD VARCHAR2 (20 BYTE)
	COORDENADAS_PK (IDCOORDENADA)
	COORDENADAS_PK (IDCOORDENADA)

Figura 5.2.1-2: Tabla Coordenadas

Contiene toda la información referente a las coordenadas de posicionamiento:

Alberga los siguientes atributos:

IdCoordenadas: clave primaria [PK] formada por un varchar2 de 20 caracteres como máximo con el id único de coordenada.

Descripción: varchar2 de 40 caracteres como máximo con el nombre de la coordenada.



Servicio Web de Información geoposicionada

Latitud: varchar2 de 20 caracteres como máximo con el valor de latitud geográfico.

Longitud: varchar2 de 20 caracteres como máximo con el valor de longitud geográfico.

b. Tabla Profesores

ABUSPAZ.PROFESORES		
P	* IDPROFESOR	VARCHAR2 (6 BYTE)
	NOMBRE	VARCHAR2 (40 BYTE)
U	CORREO	VARCHAR2 (40 BYTE)
	FICHA	VARCHAR2 (120 BYTE)
	VISIBILIDAD	VARCHAR2 (1 BYTE)
F	IDESPACEO	VARCHAR2 (20 BYTE)
	TUTORIAS	CLOB
	DEPARTAMENTO	VARCHAR2 (120 BYTE)
	PROFESORES_CORREO_UNIQUE (CORREO)	
	PROFESORES_PK (IDPROFESOR)	
	PROFESORES_IDESPACIO_FK (IDESPACEO)	
	PROFESORES_CORREO_UNIQUE (CORREO)	
	PROFESORES_PK (IDPROFESOR)	

Figura 5.2.1-3: Tabla Profesores

Consta de los siguientes atributos:

IdProfesor: clave primaria [PK] formada por un varchar2 de 6 caracteres como máximo con el id único de profesor.

Nombre: varchar2 de 40 caracteres como máximo con el nombre del profesor.

Correo: varchar2 de 40 caracteres como máximo con el apellido del profesor y será único.

Ficha: varchar2 de 120 caracteres como máximo con el apellido del profesor.

Visibilidad: varchar2 de 1 carácter con la disponibilidad del profesor.

IdEspacio: clave foránea [FK] formada por un varchar2 de 20 caracteres como máximo con el id único de espacio.

Tutorías: clob (long text) con la información de las tutorías del profesor.

Departamento: varchar2 de 120 caracteres como máximo con el departamento al que pertenece el profesor.



Desarrollo

c. Tabla Asignaturas

ABUSPAZ.ASIGNATURAS		
P *	IDASIGNATURA	VARCHAR2 (6 BYTE)
	NOMBRE	VARCHAR2 (40 BYTE)
	NOMBRE_ES	VARCHAR2 (40 BYTE)
	NOMBRE_EN	VARCHAR2 (40 BYTE)
	ASIGNATURAS_PK (IDASIGNATURA)	
	ASIGNATURAS_PK (IDASIGNATURA)	

Figura 5.2.1-4: Tabla Asignaturas

Dispone los siguientes atributos:

IdAsignatura: clave primaria [PK] formada por un varchar2 de 6 caracteres como máximo con el código localizador de asignatura.

Nombre: varchar2 de 40 caracteres como máximo con el nombre de la asignatura.

Nombre_ES: varchar2 de 40 caracteres como máximo con el nombre de la asignatura en castellano.

Nombre_EN: varchar2 de 40 caracteres como máximo con el nombre de la asignatura en inglés.

d. Tabla ProfesorAsignatura

ABUSPAZ.PROFESORASIGNATURA		
P *	IDPROFESOR	VARCHAR2 (6 BYTE)
PF *	IDASIGNATURA	VARCHAR2 (6 BYTE)
	SITUACION	VARCHAR2 (20 BYTE)
	PROFESORASIGNATURA_PK (IDASIGNATURA, IDPROFESOR)	
	PROFE_ASIG_IDASIGNATURA_FK (IDASIGNATURA)	
	PROFESORASIGNATURA_PK (IDASIGNATURA, IDPROFESOR)	

Figura 5.2.1-5: Tabla Profesorasignatura

IdProfesor: clave primaria [PK] formada por un varchar2 de 6 caracteres como máximo con el id único de profesor.

IdAsignatura: clave primaria [PK] formada por un varchar2 de 6 caracteres como máximo con el código localizador de asignatura.

Situación: varchar2 de 20 caracteres como máximo con la relación entre profesor y asignatura.



Servicio Web de Información geoposicionada

e. Tabla Espacio

ABUSPAZ.ESPACIOS		
P	* IDESPACIO	VARCHAR2 (20 BYTE)
	NOMBRE	VARCHAR2 (90 BYTE)
	DESCRIPCION	VARCHAR2 (90 BYTE)
	TIPO	VARCHAR2 (20 BYTE)
	BLOQUE	VARCHAR2 (20 BYTE)
	VISIBILIDAD	VARCHAR2 (1 BYTE)
F	IDCOORDENADA	VARCHAR2 (20 BYTE)
F	IDEDIFICIO	VARCHAR2 (20 BYTE)
	PISO	VARCHAR2 (1 BYTE)
	BOUNDINGBOX	CLOB
	ESPACIOS_PK (IDESPACIO)	
	ESPACIOS_IDCOORDENADA_FK (IDCOORDENADA)	
	ESPACIOS_IDEDIFICIO_FK (IDEDIFICIO)	
	ESPACIOS_PK (IDESPACIO)	

Figura 5.2.1-6: Tabla Espacios

IdEspacio: clave primaria [PK] formada por un varchar2 de 20 caracteres como máximo con el id único de profesor. El formato convenido para la introducción de espacios mantendrá una estructura que explicaremos con el siguiente ejemplo para el espacio 1.0.2A perteneciente al bloque 1 y al piso 0 dentro de la ETSE:

- Nombre de Facultad.
- Bloque.
- Piso.
- Espacio

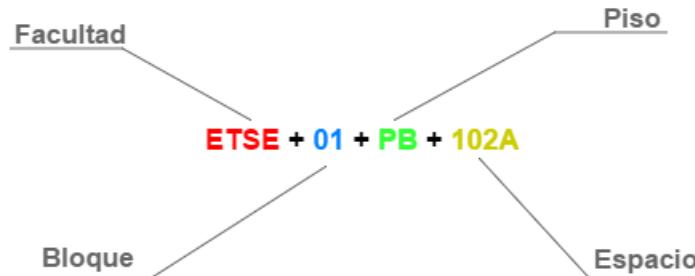


Figura 5.2.1-7: Patrón de IdEspacio

Nombre: varchar2 de 90 caracteres como máximo con el nombre del espacio.

Descripcion: varchar2 de 90 caracteres como máximo con la descripción del espacio.

Tipo: varchar2 de 20 caracteres como máximo con la finalidad del espacio.

Bloque: varchar2 de 20 caracteres como máximo con el bloque del espacio.

Piso: varchar2 de 20 caracteres como máximo con el piso al que pertenece el espacio.

Visibilidad: varchar2 de 1 carácter con la disponibilidad del espacio.



Desarrollo

IdCoordenada: clave foránea [FK] formada por un varchar2 de 20 caracteres como máximo con el id único de la coordenada.

IdEdificio: clave foránea [FK] formada por un varchar2 de 20 caracteres como máximo con el id único de edificio.

Boundingbox: clob (long text) con la información de las de los puntos que conforman el área del espacio.

f. Tabla Edificio

ABUSPAZ.EDIFICIOS		
P	* IDEDIFICIO	VARCHAR2 (30 BYTE)
	NOMBRE	VARCHAR2 (40 BYTE)
	DIRECCION	VARCHAR2 (80 BYTE)
	TELEFONO	VARCHAR2 (30 BYTE)
	ENLACE	VARCHAR2 (120 BYTE)
	CHANO	VARCHAR2 (50 BYTE)
F	* IDCORORDENADA	VARCHAR2 (20 BYTE)
	EDIFICIOS_PK (IDEDEDIFICO)	
	EDIFICIOS_IDCOORDENADAS_FK (IDCOORDENADA)	
	EDIFICIOS_PK (IDEDEDIFICO)	

Figura 5.2.1-8: Tabla Edificios

IdEdificio: clave primaria [PK] formada por un varchar2 de 30 caracteres como máximo con el id único de edificio.

Nombre: varchar2 de 40 caracteres como máximo con el nombre del edificio.

Dirección: varchar2 de 80 caracteres como máximo con la dirección del edificio.

Telefono: varchar2 de 30 caracteres como máximo con el teléfono del edificio.

Enlace: varchar2 de 120 caracteres como máximo con el enlace web del edificio.

Chanó: varchar2 de 50 caracteres como máximo con el enlace al motivo “chanó” correspondiente al edificio.

IdCoordenada: clave foránea [FK] formada por un varchar2 de 20 caracteres como máximo con el id único de la coordenada.



Servicio Web de Información geoposicionada

g. Tabla Panoramas

ABUSPAZ.PANORAMAS		
PF *	IDESPACIO	VARCHAR2 (20 BYTE)
P *	IDPANORAMA	VARCHAR2 (2 BYTE)
	PANORAMA	VARCHAR2 (20 BYTE)
	PANORAMAS_PK (IDESPACIO, IDPANORAMA)	
	PANORAMAS_IDESPACIO_FK (IDESPACIO)	
	PANORAMAS_PK (IDESPACIO, IDPANORAMA)	

Figura 5.2.1-9: Tabla Panorama

IdEspacio: clave primaria [PK] formada por un varchar2 de 20 caracteres como máximo con el id único de espacio.

IdPanorama: clave primaria [PK] formada por un varchar2 de 2 caracteres como máximo con el id único de panorama.

Panorama: varchar2 de 20 caracteres como máximo con la latitud de la coordenada.

5.2.1.3 *Diagrama de datos del sistema Servidor*

Una vez descritas las principales características de las clases de nuestro sistema servidor, ofreceremos una visión global del conjunto por medio del diagrama de clases. Un diagrama de clases es una representación del conjunto de clases con sus relaciones y herencia. Este diagrama se muestra en la siguiente figura:

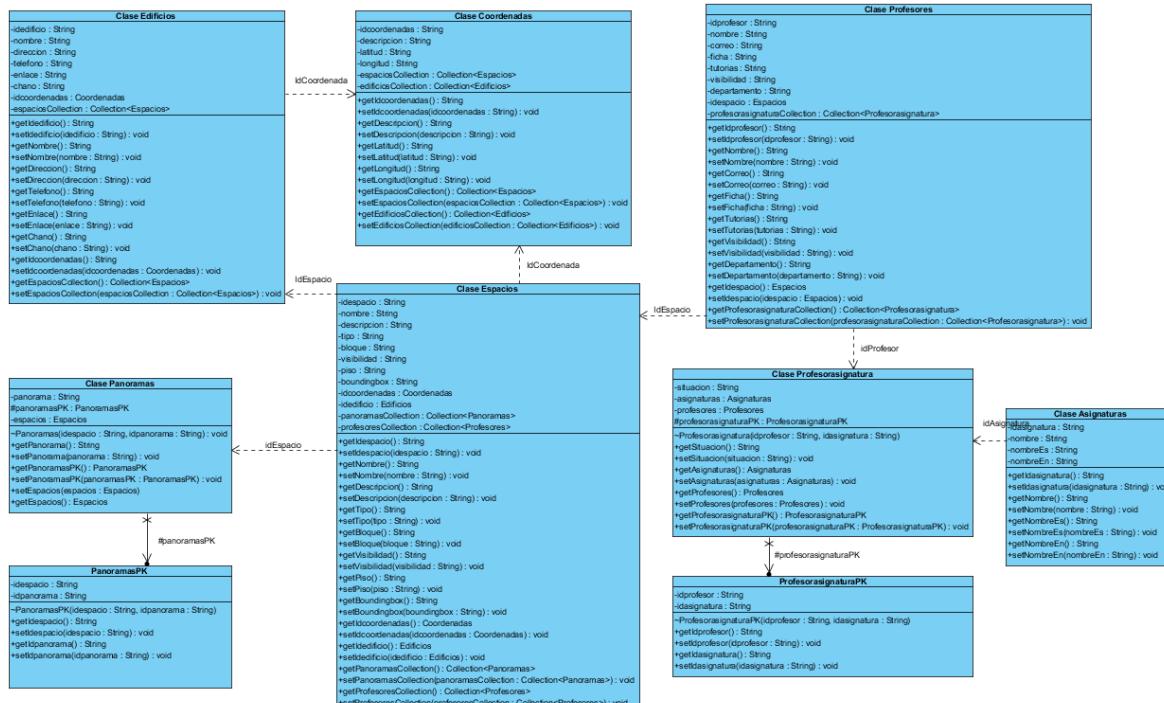


Figura 5.2.1-10: Diagrama clases del Sistema Servidor



Desarrollo

5.2.1.4 Diseño de datos en el sistema Servidor

A continuación, describiremos todas y cada una de las clases que forman nuestro sistema, veremos el origen de los datos utilizados, así como las funciones que utilizan estas clases.

Para la representación de esta información haremos uso de un diagrama de clases. Como representar toda la información en conjunto resultaría complejo, comentaremos cada módulo por separado para llegar al diagrama final simplificado.

La nomenclatura utilizada será la siguiente para los métodos:

- Private -
- Public +
- Protected #
- Constructor ~
- NameQueries @

Las variables serán descritas de la siguiente manera:

- Nombre:tipo

Las funciones serán descritas de la forma siguiente:

- Nombre(argumento:tipo):retorno.

Dicho esto ya podemos empezar a comentar las clases que forman nuestro sistema servidor.

a. Clase Coordenadas

Clase Coordenadas	
-idcoordenadas : String	
-descripcion : String	
-latitud : String	
-longitud : String	
-espaciosCollection : Collection<Espacios>	
-edificiosCollection : Collection<Edificios>	
+getIdcoordenadas() : String	
+setIdcoordenadas(idcoordenadas : String) : void	
+getDescripcion() : String	
+setDescripcion(descripcion : String) : void	
+getLatitud() : String	
+setLatitud(latitud : String) : void	
+getLongitud() : String	
+setLongitud(longitud : String) : void	
+getEspaciosCollection() : Collection<Espacios>	
+setEspaciosCollection(espaciosCollection : Collection<Espacios>) : void	
+getEdificiosCollection() : Collection<Edificios>	
+setEdificiosCollection(edificiosCollection : Collection<Edificios>) : void	

Figura 5.2.1-11: Clase Coordenadas

Posee los siguientes parámetros y métodos:

-idcoordenadas, String con el identificador de coordenada.

-descripción, String con la información de la coordenada.

-latitud, String con la latitud geográfica.



Servicio Web de Información geoposicionada

- longitud, String con la longitud geográfica.
- espaciosCollection, Colección de espacios necesario para cruces de datos entre tablas
- edificiosCollection, Colección de edificios necesario para cruces de datos entre tablas
- +Getters y Setters de los parámetros descritos.

b. Clase Edificios

Clase Edificios
<pre>-idedificio : String -nombre : String -direccion : String -telefono : String -enlace : String -chano : String -idcoordenadas : Coordenadas -espaciosCollection : Collection<Espacios> +getIdEdificio() : String +setIdEdificio(idEdificio : String) : void +getNombre() : String +setNombre(nombre : String) : void +getDireccion() : String +setDireccion(direccion : String) : void +getTelefono() : String +setTelefono(telefono : String) : void +getEnlace() : String +setEnlace(enlace : String) : void +getChano() : String +setChano(chano : String) : void +getIdCoordenadas() : String +setIdCoordenadas(idcoordenadas : Coordenadas) : void +getEspaciosCollection() : Collection<Espacios> +setEspaciosCollection(espaciosCollection : Collection<Espacios>) : void</pre>

Figura 5.2.1-12:Clase Edificios

Posee las siguientes características y métodos:

- idedificio, String con el identificador de edificio.
- nombre, String con el nombre del edificio.
- direccion, String con la dirección del edificio.
- enlace, String con el enlace a la web del edificio.
- chano, String con la ruta a la imagen de chano correspondiente al edificio.
- idcoordenadas, objeto Coordenadas resultado de la propagación de la clave primaria en la cardinalidad.
- espaciosCollection, Colección de Espacios, necesario para los cruces de datos entre tablas.
- +Getters y Setters de los parámetros descritos.



Desarrollo

c. Clase Espacios

Clase Espacios	
-idespacio : String	
-nombre : String	
-descripcion : String	
-tipo : String	
-bloque : String	
-visibilidad : String	
-piso : String	
-boundingbox : String	
-idcoordenadas : Coordenadas	
-idedificio : Edificios	
-panoramasCollection : Collection<Panoramas>	
-profesoresCollection : Collection<Profesores>	
+getIDESpacio() : String	
+setIDESpacio(idespacio : String) : void	
+getNombre() : String	
+setNombre(nombre : String) : void	
+getDescripcion() : String	
+setDescripcion(descripcion : String) : void	
+getTipo() : String	
+setTipo(tipo : String) : void	
+getBloque() : String	
+setBloque(bloque : String) : void	
+getVisibilidad() : String	
+setVisibilidad(visibilidad : String) : void	
+getPiso() : String	
+setPiso(piso : String) : void	
+getBoundingbox() : String	
+setBoundingbox(boundingbox : String) : void	
+getIdcoordenadas() : Coordenadas	
+setIdcoordenadas(idcoordenadas : Coordenadas) : void	
+getIdedificio() : Edificios	
+setIdedificio(idedificio : Edificios) : void	
+getPanoramasCollection() : Collection<Panoramas>	
+setPanoramasCollection(panoramasCollection : Collection<Panoramas>) : void	
+getProfesoresCollection() : Collection<Profesores>	
+setProfesoresCollection(profesoresCollection : Collection<Profesores>) : void	

Figura 5.2.1-13:Clase Espacios

Dispone los atributos y métodos siguientes:

-idespacio, String con el identificador de espacio.

-nombre, String con el nombre del espacio.

-descripción, String con la descripción del espacio.

-bloque, String con el bloque del espacio.

-boundingbox, String que alberga los puntos que delimitan el área del espacio.

-idcoordenadas, objeto Coordenadas resultado de la propagación de la clave primaria en la cardinalidad.

-panoramaCollection, Colección de Panoramas necesario para los cruces de datos entre tablas.

-profesoresCollection, Colección de Profesores necesario para los cruces de datos entre tablas.

+Getters y Setters de los atributos descritos.

@findByPiso(piso:String), consulta que recupera los espacios pertenecientes al piso pasado en el argumento.



Servicio Web de Información geoposicionada

d. Clase Panoramas

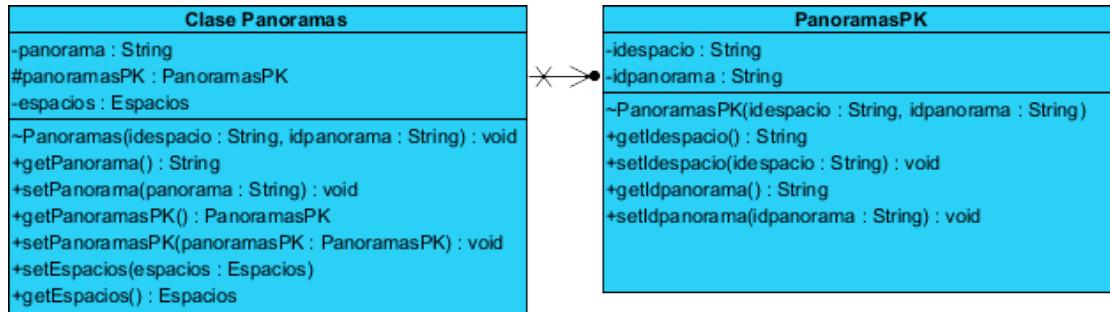


Figura 5.2.1-14: Clase Panoramas

Se confecciona con los siguientes parámetros y métodos:

-panorama, String que posee el identificador del panorama.

-panoramaPK, objeto PanoramaPK que alberga la clave compuesta para panoramas.

-espacios, objeto Espacios resultado de la propagación de la clave primaria en la cardinalidad.

+Getters y Setters de los atributos descritos.

@findPanoramasByldespacio(idespacio:String), consulta que recupera los panoramas pertenecientes al espacio pasado en el argumento.

e. Clase Profesores

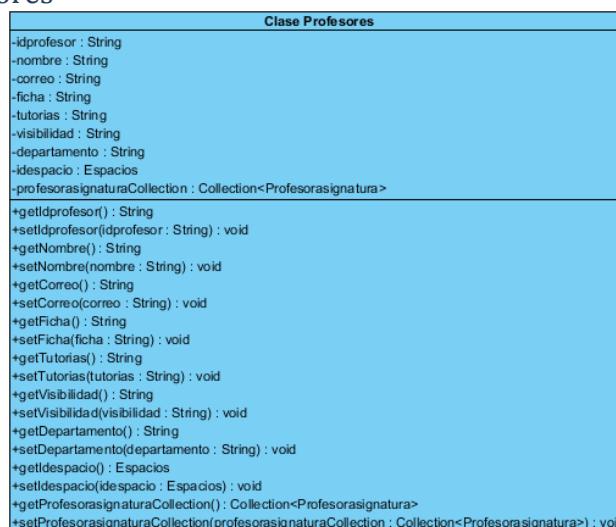


Figura 5.2.1-15: Clase Profesores

Se compone de los siguientes parámetros y métodos:

@getAsignaturasById(idprofesor:String), consulta que recupera las asignaturas que imparte el profesor pasado en el argumento.

Desarrollo

f. Clase Asignaturas

Clase Asignaturas	
-idasignatura : String	
-nombre : String	
-nombreEs : String	
-nombreEn : String	
+getIdasignatura() : String	
+setIdasignatura(idasignatura : String) : void	
+getNombre() : String	
+setNombre(nombre : String) : void	
+getNombreEs() : String	
+setNombreEs(nombreEs : String) : void	
+getNombreEn() : String	
+setNombreEn(nombreEn : String) : void	

Figura 5.2.1-16: Clase Asignaturas

Se confecciona con los siguientes parámetros y métodos:

- idasignatura, String que posee el identificador de la asignatura.
- nombre, String que posee el nombre en valenciano de la asignatura.
- nombreEs, String que posee el nombre en castellano de la asignatura.
- nombreEn, String que posee el nombre en inglés de la asignatura.
- +Getters y Setters de los atributos descritos.

@getProfesoresById

g. Clase ProfesorAsignatura

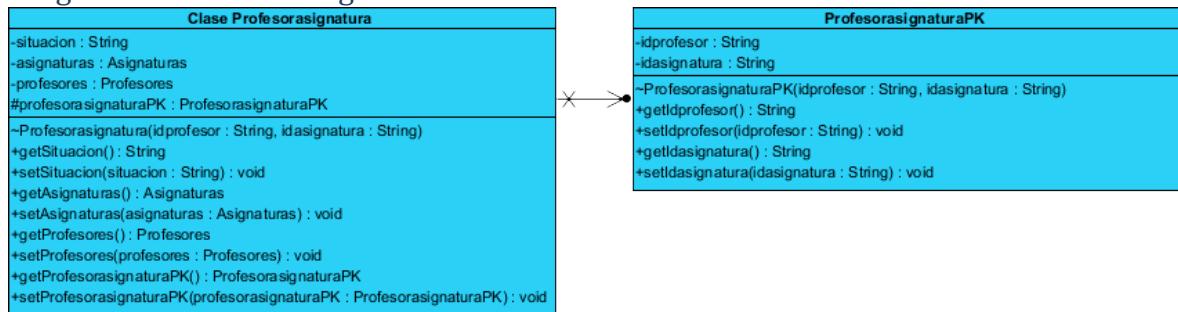


Figura 5.2.1-17: Clase ProfesorAsignatura

Se crea con los siguientes atributos y métodos:

- asignatura, objeto Asignaturas resultado de la propagación de la clave primaria en la cardinalidad.



Servicio Web de Información geoposicionada

-profesor, objeto Profesores resultado de la propagación de la clave primaria en la cardinalidad.

-profesorasignaturaPK, objeto ProfesorasignaturaPK que alberga la clave compuesta para la relación de cardinalidad muchos a muchos (n : m).

-situacion, String con la relación profesor-asignatura, generalmente titular o laboratorio.

+Getters y Setters de los atributos descritos.

5.2.1.5 Diagrama de datos del sistema Cliente* cierre:boolean

Tras haber explicado los atributos y métodos más relevantes de nuestro sistema Cliente, pasaremos a representar el diagrama de datos que componen la aplicación. De esta manera podemos observar, de una manera más sencilla, cómo interactúan los diversos objetos presentes en la solución cliente. Este diagrama se muestra en la siguiente figura:

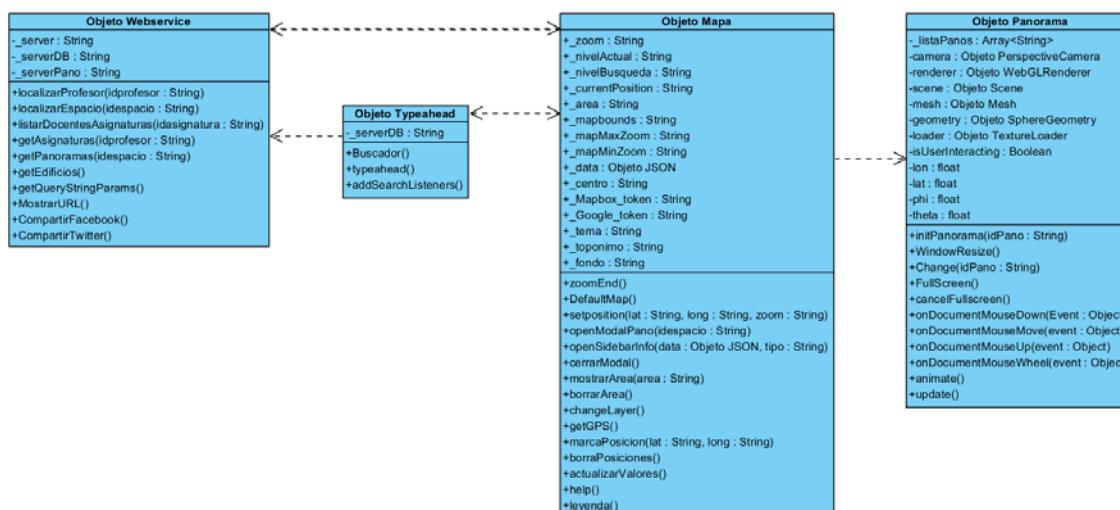


Figura 5.2.1-18: Diagrama de datos del sistema Cliente

5.2.1.6 Diseño de datos en el sistema Cliente

Tras la descripción en el lado del servidor, pasaremos a detallar el diseño de datos correspondiente a la parte del cliente siguiendo la misma estructura que en el apartado anterior.

a. Objeto Mapa

Se crea con los siguientes atributos y métodos:

+_zoom, String que almacena el nivel de zoom que presenta el mapa.

+_nivelActual, String con el nivel de piso actual en el que se encuentra el usuario.

+_nivelBusqueda, String con el nivel en el que se encuentra el resultado de la búsqueda.

+_currentposition, String con la posición actual del usuario en el mapa.

+_area, String con los puntos geográficos que delimitan el área del espacio tras su búsqueda.



Desarrollo

+_mapbounds, String con los puntos geográficos que delimitan el área de visión del plano en el mapa.

+_mapMaxzoom, String con el valor máximo de zoom que puede mostrarse en el mapa.

+_mapMinzoom, String con el valor mínimo de zoom que puede mostrarse en el mapa.

+_data, Objeto JSON resultado de la búsqueda para mostrarla en la barra lateral.

+_centro, String con las coordenadas donde se carga o desplaza el mapa.

+_mapbox_token, String con el identificador único de usuario del proveedor de planos Mapbox.

+_google_token, String con el identificador único de usuario del proveedor de mapas Google.

+_tema, String que determina si el mapa carga el plano básico o temático.

+_toponimo, String que determina si el mapa carga el plano con descripción o código.

+_fondo, String que determina si el mapa carga el proveedor de mapas Google o Mapbox-Leaflet.

+zoomEnd(), actualiza la vista de iconos según el nivel de zoom en el que se encuentra el usuario.

+DefaultMap(), se inicializan las variables a valores por defecto para mostrar el mapa.

+setPosition(lat:String ,long:String ,zoom:String,cierre:Boolean), se realiza desplazamiento del mapa a la latitud y longitud además de ajustar el zoom. Además informaremos si debe cerrar la barra lateral o no al ejecutarse.

+openModalPano(idEspacio:String), abre un visor de panorámicas del espacio en una ventana modal sobre el mapa.

+openSidebarInfo(data:Objeto JSON, tipo:String), abre la barra lateral con la información relativa al profesor o el espacio en formato JSON según el tipo de recurso.

+cerrarModal(), cierra la ventana modal creada.

+mostrarArea(area:String), muestra la zona delimitada por coordenadas geográficas del espacio resultado de la búsqueda.

+borrarArea(), elimina la zona delimitada del espacio resultado de la búsqueda.

+changeLayer(), cambia el modelo de representación del mapa en base a los valores de los atributos.

+getGPS(), obtiene y muestra la ubicación aproximada del usuario en el mapa mediante GPS y lo muestra en el mapa.

+marcaPosicion(lat:String,long:String), coloca un marcador en las coordenadas establecidas en el argumento.



Servicio Web de Información geoposicionada

- +borrarPosiciones(), elimina todos los marcadores creados por el usuario.
- +actualizarValores(), realiza una actualización de los valores recogidos o que hayan sufrido variaciones.
- +help (), muestra una ventana modal con la información sobre el uso de la aplicación.
- +leyenda (), muestra una ventana modal con la información sobre los elementos visuales.

b. Objeto Panorama

Se crea con los siguientes atributos y métodos:

- +_listaPanos, Array de Strings con los identificadores de panorámicas
- +camera, Objeto THREE.PerspectiveCamera con los valores de cámara.
- +renderer, Objeto THREE.WebGLRenderer con la referencia del render.
- +scene, Objeto THREE.Scene con la referencia de escena.
- +mesh, Objeto THREE.Mesh que contiene geometría y textura.
- +geometry, Objeto THREE.SphereGeometry que contiene la geometría
- +loader, Objeto THREE.TextureLoader que contiene la textura.
- +isUserInteracting, Boolean que indica si el usuario está interactuando con la panorámica.
- +lon, Float con la posición x relativa al visor de panorámicas.
- +lat, Float con la posición y relativa al visor de panorámicas.
- +phi, Float con el valor de longitud en radianes.
- +theta, Float con el valor de latitud en radianes.
- +initPanorama(idPano:String), se inicializan las variables del visor de panoramas y se carga la panorámica pasada por el argumento.
- +WindowResize(), realiza un reescalado del visor de panoramas ajustándolo proporcionalmente al dispositivo.
- +Change(idPano:String), cambia el panorama actual por el panorama pasado por el argumento.
- +Fullscreen(), activa el modo Pantalla completa ajustando el visor de panoramas al tamaño del dispositivo.
- +cancelFullscreen(event), desactiva el modo Pantalla completa y devuelve el visor de panoramas al tamaño por defecto.
- +onDocumentMouseDown(event:Object), captura la posición inicial (x,y) del visor y activa el flag de *isUserInteracting*.



Desarrollo

+onDocumentMouseMove(event:Object), captura la posición final (x,y) continuamente mientras *isUserInteracting* está activado.

+onDocumentMouseUp(event:Object), desactiva el flag de *isUserInteracting*.

+onDocumentMouseWheel(event:Object), actualiza el atributo *camera* con los valores de la rueda del ratón existentes en el evento.

+animate(), inicia la secuencia de animación.

+update(), actualiza los atributos *phi*, *theta*, *camera* y *renderer* en cada fotograma. Si no hay interacción, *phi* aumenta de forma constante.

c. Objeto Webservice

Se crea con los siguientes atributos y métodos:

+_server, String con la ruta base en el que se encuentran los recursos gráficos y librerías JavaScript.

+_serverDB, String con la dirección url base donde se encuentran los recursos de la base de datos.

+_serverPano, String con la ruta base en el que se encuentran las panorámicas de 360 grados.

+localizarProfesor(idprofesor:String), realiza una consulta a la base de datos recuperando la información del profesor presente en el argumento.

+localizarEspacio(idespacio:String), realiza una consulta a la base de datos recuperando la información del espacio presente en el argumento.

+listarDocentesAsignaturas(idasignatura:String), realiza una consulta a la base de datos recuperando la relación de profesores que imparten la asignatura presente en el argumento.

+getAsignaturas(idprofesor:String), realiza una consulta a la base de datos recuperando la relación de asignaturas adscritas al profesor presente en el argumento.

+getPanoramas(idespacio:String), realiza una consulta a la base de datos recuperando la relación de panoramas pertenecientes al espacio presente en el argumento.

+getEdificios(), realiza una consulta a la base de datos recuperando todos los edificios.

+getQueryStringParams(), realiza una recuperación de los parámetros de consulta en la dirección del navegador.

+MostrarURL(), muestra la dirección web completa de acceso al recurso.

+CompartirFacebook(), genera una nueva ventana que accede a la red social *Facebook* para compartir el enlace directo al recurso.



Servicio Web de Información geoposicionada

+CompartirTwitter(), genera una nueva ventana que accede a la red social *Twitter* para compartir el enlace directo al recurso.

d. Objeto buscador

Se crea con los siguientes atributos y métodos:

+_serverDB, String con la dirección url base donde se encuentran los recursos de la base de datos.

+Buscador(), Muestra ventana modal con los distintos modelos de búsqueda.

+typeahead(), crea los dataset de espacios, asignaturas y profesores para las sugerencias de resultados.

+addSearchListeners(), activa los distintos oyentes de eventos sobre las acciones efectuadas en las búsquedas.



Desarrollo

5.2.2 Diseño de servicios en el sistema Servidor

Tal y como comentamos en el apartado REST¹⁰, esta tecnología basa uno de sus principios en conformar todos los recursos y acciones a modo de URI. Por esta razón, pasaremos a describir los servicios que dan lugar a las distintas acciones que debe realizar nuestro servidor cuando nuestro cliente solicite una dirección concreta. Posteriormente, comentaremos la clase *CORS*, que permitirá agregar las cabeceras necesarias en nuestras peticiones HTTP mediante el servicio de intercambio de recursos para dominios cruzados (**Cross Origins Resource Service**).

5.2.2.1 Servicio Espacios

/espacios/

Servicio tipo GET que devuelve una lista con todos los espacios presentes en la base de datos que estén disponibles.

/espacios/{id}

Servicio tipo GET que devuelve un objeto JSON con la información del espacio con identificador *id*. Si el espacio está vetado, se devolverá un objeto predefinido de espacio no disponible.

/espacios/{id}/panoramas

Servicio tipo GET que devuelve una lista con los panoramas que pertenezcan al espacio con el identificador *id*.

/espacio/piso/{id}

Servicio tipo GET que devuelve una lista de espacios que estén asociados al piso con el valor *id* y que estén disponibles.

5.2.2.2 Servicio Asignaturas

/asignaturas/

Servicio tipo GET que devuelve una lista con todas las asignaturas presentes en la base de datos.

/asignaturas/{id}/

Servicio tipo GET que devuelve un objeto JSON con la información de la asignatura con identificador *id*.

/asignaturas/{id}/profesores

Servicio tipo GET que devuelve una lista con los profesores que imparten la asignatura con el identificador *id*.

5.2.2.3 Servicio Panorama

/panoramas/

¹⁰ Véase *RESTful Java web services : design scalable and robust RESTful web services with JAX-RS and Jersey extension APIs* en Bibliografía.



Servicio Web de Información geoposicionada

Servicio tipo GET que devuelve una lista con todos los panoramas presentes en la base de datos.

/panoramas/{id}/

Servicio tipo GET que devuelve un objeto JSON con la información del panorama con identificador *id*.

5.2.2.4 *Servicio Edificio*

/edificios/

Servicio tipo GET que devuelve una lista con todos los edificios presentes en la base de datos.

Implementación Servicio Coordenadas

/coordenadas/

Servicio tipo GET que devuelve una lista con todas las coordenadas presentes en la base de datos.

5.2.2.5 *Servicio Profesores*

/profesores/

Servicio tipo GET que devuelve una lista con todos los profesores presentes en la base de datos.

/profesores/{id}/

Servicio tipo GET que devuelve un objeto JSON con la información del profesor con identificador *id*.

/profesores/{id}/asignaturas

Servicio tipo GET que devuelve una lista con las asignaturas que imparte el profesor con identificador *id*.

5.2.2.6 *Servicio CORS*

Puesto que nuestro sistema es dual en una arquitectura cliente-servidor, puede darse la situación de que ambas aplicaciones sean alojadas en distintas máquinas. Esta posibilidad provocará que la comunicación entre ambas soluciones no se produzca a consecuencia de la política de seguridad del mismo origen, la cual previene que un recurso origen pueda ser accedido desde un origen distinto.

Para salvar esta situación debemos otorgar permisos de acceso añadiendo las cabeceras oportunas que permitan los métodos HTTP que designemos.

Dado que nuestra aplicación será únicamente de consumo para el usuario final, daremos permisos de acceso a métodos GET, POST y OPTION a cualquier máquina agregando las líneas:

- *Access-Control-Allow-Headers*", "*Content-Type*".
- *Access-Control-Allow-Methods*", "*OPTIONS, GET, POST*".



Desarrollo

- *Access-Control-Allow-Origin*", "*".

En el caso de que quisiéramos disponer de la capacidad de modificar o eliminar recursos a través del uso de sentencias HTTP, podemos otorgar permisos específicos de edición para máquinas o redes específicas con las siguientes líneas en las cabeceras HTTP:

- *Access-Control-Allow-Headers*", "Content-Type".
- *Access-Control-Allow-Methods*", "PUT, DELETE".
- *Access-Control-Allow-Origin*", "147.156.XX.XX".

5.2.3 Diseño del interfaz

En el diseño del interfaz entran en juego modelos diferentes que hay que tener en cuenta.

- **Modelo de usuario:** Muestra el perfil de los usuarios finales del sistema. Para construir un interfaz de usuario eficaz deberemos comenzar con el conocimiento de los usuarios a quién va dirigido y su relación con el mundo de la informática. En lo que respecta a perfiles de usuario, en general, serán profesores, estudiantes o futuros estudiantes. Además, debemos tener en cuenta que la aplicación que se va a desarrollar está basada en la Web, un entorno en el que la gran mayoría de la población se encuentra familiarizado.
- **Percepción del sistema:** Esta es la imagen del sistema que el usuario final lleva en la cabeza. Como los usuarios ya estarán bastante acostumbrados al funcionamiento de un buscador tendrán una amplia percepción sobre nuestro sistema. Y, aunque no posean mucha relación con el funcionamiento de un buscador, el interfaz es lo suficientemente simple e intuitivo como para prever el mecanismo de utilización de este.
- **La imagen del sistema:** Este aspecto combina la manifestación exterior del sistema con toda la información de soporte que describen sintaxis y semántica del sistema. Cuando coinciden la imagen del sistema y la percepción del sistema, los usuarios se sienten a gusto con el software y hacen uso de él eficazmente.

Una vez evaluados los modelos de interfaz debemos tener en cuenta que las aplicaciones Web tienen características distintas a las aplicaciones de escritorio. Con HTML no se puede controlar la presentación de una manera fiable al 100% y hay que aceptar ciertos compromisos de visualización.

Nuestra aplicación Web puede ser visualizada desde una gran variedad de navegadores y dispositivos de pantalla.

A la hora de controlar cómo se mueve un usuario por una aplicación Web resulta complicado, ya que puede hacerlo introduciendo manualmente URLs o bien por medio de enlaces. Nuestro sistema dispone de un método que determina si el usuario accede directamente a un recurso determinado, profesor o espacio, o si por el contrario accede de manera por defecto.

A continuación, describiremos la estructura que seguirá nuestra aplicación Web:



Servicio Web de Información geoposicionada

En la página principal mostraremos una barra de navegación con varias funcionalidades (Localizar, Capas, Facultades y Miscelánea), seguido de un mapa personalizado donde se muestren marcadores en facultades y/o campus en función del zoom actual.

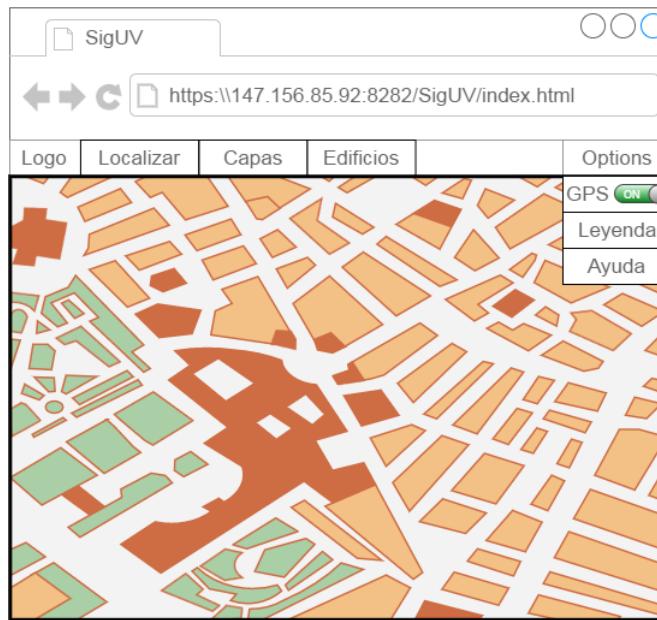


Figura 5.2.3-1: Wireframe página principal

Dispondremos de la opción de Localizar para buscar en función de tres apartados dispuestos en fichas:

- Profesores, Asignaturas y Espacios, que estarán formados por sendos campos de texto donde introduciremos el nombre del profesor, asignatura y/o espacio a buscar y un botón Buscar donde validaremos la búsqueda. Durante la introducción de texto, el sistema proporcionará sugerencias de elementos a fin de proporcionar una ayuda al usuario.
- Estos tres buscadores facilitarán al usuario llegar al recurso que busca por tres vías diferentes, permitiéndole la elección del que considere más oportuno para llegar a dicho recurso.

Desarrollo

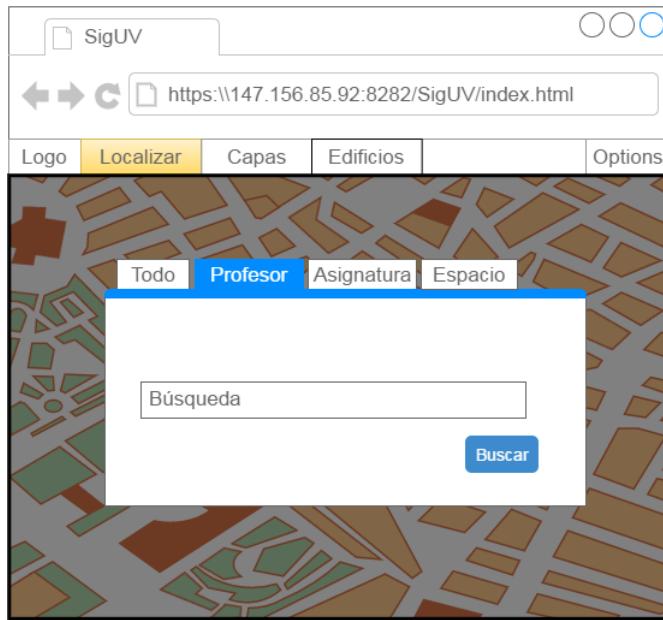


Figura 5.2.3-2: Wireframe buscador

- Dispondremos de la opción Capas para configurar el modo de visualización del mapa:
 - Fondo, con ello podremos cambiar el mapa de fondo empleado, a elegir entre plano y Google Maps.
 - Tema, con ello podremos escoger entre un plano básico de B/N o uno categorizado por colores según la finalidad del espacio.
 - Topónimo, con ello podremos seleccionar entre un plano con el nombre del espacio o uno con la codificación del mismo.
 - Iconos, con ello podremos activar o desactivar los marcadores del mapa.

Todos los grupos de configuración podremos combinarlos a nuestro gusto para obtener la visualización del mapa que más nos interese.

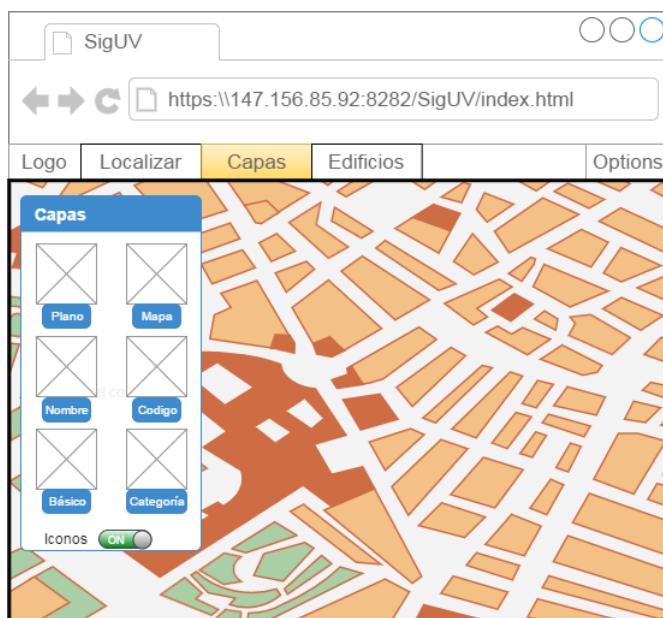


Figura 5.2.3-3: Wireframe de configuración



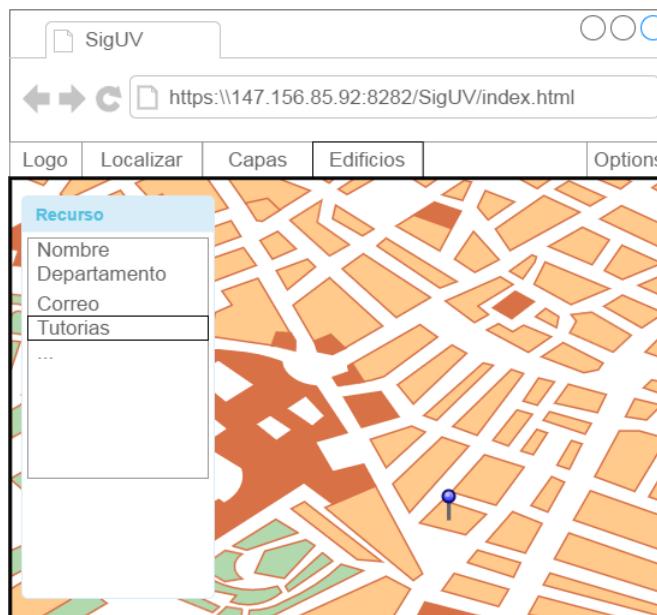
Servicio Web de Información geoposicionada

Dispondremos de la opción Facultades, donde aparecerá un listado de todas las facultades presentes, que nos permitirá desplazarnos inmediatamente a su ubicación en el mapa.

Por último, dispondremos en la barra de navegación de una opción adicional como miscelánea:

- GPS, con ello activaremos el geo posicionamiento del dispositivo y nos situará en el mapa.
- Leyenda, con ello mostraremos una ventana con información relativa al código de colores empleado y qué significan.
- Ayuda, donde se dispondrá una ventana con asistencia y guías sencilla sobre el funcionamiento de la aplicación web.

Cuando sea escogido un profesor, asignatura o espacio válido, el mapa mostrará su localización mediante un marcador y un área delimitadora del espacio. Además, se mostrará una ventana con la información relativa al profesor o espacio buscado.



En el caso de que el espacio dispusiera de panoramas de 360 grados, se mostrará un botón que permita visualizarlas mediante una ventana modal e ir cambiando entre las distintas imágenes disponibles del mismo. De igual manera, se podrá acceder al modo pantalla completa mediante un botón dispuesto para tal fin en el visor de panoramas.

Otras características de nuestra Aplicación:

- El usuario no realiza pasos innecesarios. Hemos comentado que el sistema proporciona ayuda mediante sugerencias conforme introduce texto, por lo que ayuda al usuario a encontrar el recurso deseado. Si el usuario, en cambio, introdujera una búsqueda de un recurso inexistente, el sistema mostraría un mensaje informando de que el recurso solicitado no existe mediante una ventana modal.
- Los elementos de la interfaz son estándar de HTML. Con lo cual, el usuario no tiene que aprender el significado de nuevas imágenes e iconos. De esta manera se

Desarrollo

unifica el diseño en botones en los que se describe en texto el significado de hacer clics sobre él.

- Se hace uso de una plantilla de estilos. El uso de una plantilla de estilos hace que la aplicación tenga una apariencia homogénea en fuentes, tamaños y colores haciendo que el usuario se sienta siempre dentro del contexto de la aplicación.
 - Se puede aplicar la información que nos devuelve la página de resultados utilizando enlaces a las fichas personales del profesor o las webs de cada una de las facultades.
 - Visualización responsive de todos los elementos de la aplicación de tal manera que se ajuste la presentación del contenido a todos los tamaños de dispositivo más habituales (PC, tablets, móviles).

Posibilidad de volver al comienzo de la página con sólo cerrar el buscador de recursos o la información del profesor o espacio.



Desarrollo

5.3 Implementación del sistema

La implementación del sistema llevará a la práctica todo lo analizado y diseñado en los pasos anteriores. En este apartado no se toman decisiones importantes en cuanto al funcionamiento sino que sólo se evalúan aspectos de la implementación.

5.3.1 Fotografías 360 grados

Para la captura de panoramas hemos realizado varias solicitudes a Dirección a fin de obtener los permisos necesarios para tomar fotografías dentro de los espacios de la facultad y que el personal de conserjería sea informado de darnos acceso a los mismos.

Con el fin de salvaguardar toda situación legal que involucre los derechos a la intimidad, imagen u honor, tomaremos las siguientes acciones en cuenta:

- Realizaremos fotografías con el menor número posible de personas.
- De no ser posible evitar la captura de personas reconocibles, se aplicarán efectos de difusión en la imagen mediante técnicas de edición fotográfica con software adicional.
- Si hubiera la necesidad de capturar personas reconocibles, se les informará que deben llenar el documento correspondiente al consentimiento de cesión de imagen para publicación.
- No se tomarán fotografías de zonas privadas, despachos personales y toda aquella estructura considerada sensible o designada por la división de seguridad de la ETSE.

Hemos realizado capturas de panorámicas de 360 grados empleando la cámara Ricoh Theta S que nos proporciona imágenes con una resolución de 5.376px de ancho por 2.688px de alto, una relación 2:1. Posteriormente hemos aplicado la técnica de alto rango dinámico o HDR (High Dynamic Range), donde conseguimos unas imágenes más similares a las del ojo humano ya que realizamos una compensación de luces y sombras no posible con fotografías captadas de forma estándar. Para conseguir este efecto debemos realizar al menos 3 disparos de la misma escena y ajustando el diafragma de la cámara a distintos valores de fotómetro para cada uno, generalmente en el punto 0, un punto por encima y otro por debajo.

Para la toma de fotografías hemos empleado un trípode fotográfico ajustado a una altura aproximada de 1.65m para lograr una perspectiva visual aproximada a la de una persona promedio. Procuraremos disponer el trípode en su posición horizontal lo más nivelado posible para evitar capturas no equilibradas que dificulten su visionado posterior.

Realizaremos al menos 1 captura de todos los espacios disponibles, aumentando su número en función del tamaño, finalidad o necesidad del mismo, de esta manera cubriremos toda el área del espacio si existen varias zonas de interés para ser mostradas.

Una vez completado el proceso de captura de imágenes entre los espacios disponibles de la ETSE, debemos cargar las fotografías en el servidor para que nuestro sistema las localice y muestre cuando el usuario seleccione un espacio deseado. Para facilitar la



Servicio Web de Información geoposicionada

identificación de las panorámicas, estas se cargarán siguiendo un formato de etiquetado muy similar al utilizado para los espacios:

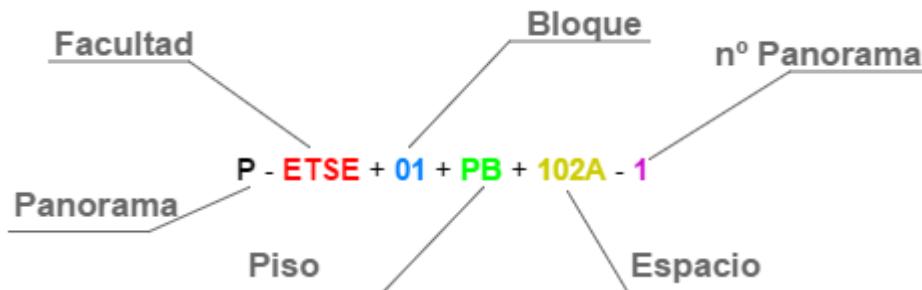


Figura 5.3.1-1: Esquema de panoramas

5.3.2 Creación de planos de l'Escola Tècnica Superior d'Enginyeria*

Para la creación de los planos de la Escuela hemos solicitado los preceptivos permisos a los órganos de la ETSE para obtener los planos técnicos del edificio en formato CAD.

Posteriormente haremos uso de la herramienta de diseño AutoCAD a fin de realizar una limpieza de las distintas capas que son prescindibles para nuestro proyecto como pueden ser las acometidas de luz y agua, mobiliario o cristalería, entre muchas otras. De igual manera a como nos ocurrió con la toma de fotografías, ocultaremos en el mapa todas aquellas zonas irrelevantes o estructuralmente sensibles que nos designaron desde Dirección y Seguridad.

Para ello, editaremos el espacio con la herramienta software de tal manera que ocultemos los espacios vetados dando la apariencia de ser una ubicación inexistente o inaccesible.

Una vez limpios los planos, exportaremos los mismos a formatos de imagen con una alta resolución para poder ser procesados con software de edición fotográfica, en nuestro caso emplearemos Adobe Photoshop.

Después de cargar las imágenes en nuestro nuevo software, editaremos imperfecciones y daremos un aspecto contorneado y agregaremos las etiquetas de texto correspondientes a cada habitación. Este será nuestro plano *básico* cuando el usuario desee una representación donde todos los espacios mantendrán una misma apariencia.

Desarrollo

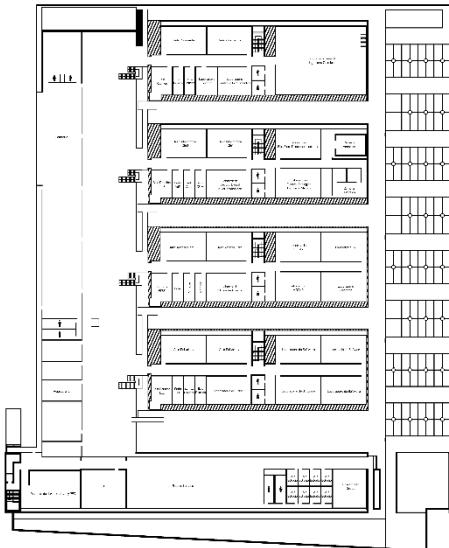


Figura 5.3.2-1: Plano Básico

Posteriormente, clonaremos el plano y lo editaremos asignando categorías por finalidad del espacio y esta categoría estará asociada a un color determinado dentro una paleta creada por nosotros, este plano será nuestro plano *categorizado*.



Figura 5.3.2-3: Leyenda Categorías

Figura 5.3.2-2: Plano categorizado

Una vez obtenidos los planos *básico* y por *categorías*, deberemos clonar nuevamente sendos planos para cambiar las etiquetas de texto por el código definido para cada espacio. De esta manera obtendremos dos planos más con la denominación codificada.

Estando en posesión de todos los planos en sus distintos pisos, debemos transformar nuestras imágenes a un conjunto de cuadrantes que puedan ser cargados en forma de celdas y que responda su resolución de forma adecuada al nivel de zoom que se emplee.



Servicio Web de Información geoposicionada

Para ello usaremos la aplicación software Maptiler¹¹ la cual nos permitirá generar estas celdas multinivel. Comenzaremos por cargar el plano en la aplicación, le indicaremos el sistema de referencia global que vamos a emplear para posicionar, en nuestro caso usaremos EPSG:3857, también conocido como WGS84 Web Mercator, dado que es el más empleado en situaciones de cartografía, navegación marítima/aérea y, por extensión, en aplicaciones de webs de mapas como Google, Bing u OpenStreetMap, asociaremos puntos característicos de nuestro plano con puntos geográficos en un mapa, seleccionaremos el rango de niveles de zoom que deseamos que en nuestro caso será entre 10 y 25, marcaremos que deseamos exportar la imágenes en formato PNG con transparencia de 256px por 256px y comenzaremos el proceso de exportación.

Una vez realizado el mismo proceso para cada uno de los modelos de plano y sus distintos niveles, simplemente debemos alojar estos conjuntos de imágenes en un espacio de disco que sea accesible a través de la red. De esta manera, estaremos en posesión de todos lo necesario para integrar nuestro plano ETSE en nuestro sistema de información geoposicionada.

5.3.3 Implementación de la base de datos

Para la gestión y creación de la base de datos, como ya se acordó en el apartado de Estado del Arte, utilizaremos Oracle Express. Concretamente utilizaremos la última versión disponible hasta la fecha, Oracle XE 11g.

A continuación mostraremos como hemos creado la base de datos y las diferentes tablas:

-Creación de la base de datos siguv:

```
CREATE DATABASE SIGUV;
```

-Seleccionamos esta base de datos después de crearla para posteriormente crear las tablas.

```
USE SIGUV;
```

-Creación de las tablas

Tabla Coordenadas

```
CREATE TABLE "COORDENADAS"
  ("IDCOORDENADA" VARCHAR2 (20 BYTE) NOT NULL ENABLE,
   "DESCRIPCION" VARCHAR2 (40 BYTE),
   "LATITUD" VARCHAR2 (20 BYTE) NOT NULL ENABLE,
   "LONGITUD" VARCHAR2 (20 BYTE) NOT NULL ENABLE,
   CONSTRAINT "COORDENADAS_PK" PRIMARY KEY ("IDCOORDENADA")
  )
```

Tabla Edificios

```
CREATE TABLE "EDIFICIOS"
```

¹¹ Véase *Create an overlay of Google Maps with Photoshop* en Bibliografía



Desarrollo

```
( "IDEDIFICIO" VARCHAR2 (30 BYTE) NOT NULL ENABLE,
  "NOMBRE" VARCHAR2 (40 BYTE),
  "DIRECCION" VARCHAR2 (80 BYTE),
  "TELEFONO" VARCHAR2 (30 BYTE),
  "ENLACE" VARCHAR2 (120 BYTE),
  "CHANO" VARCHAR2 (50 BYTE),
  "IDCOORDENADA" VARCHAR2 (20 BYTE) NOT NULL ENABLE,
  CONSTRAINT "EDIFICIOS_PK" PRIMARY KEY ("IDEDIFICIO")
TABLESPACE "SYSTEM" ENABLE,
  CONSTRAINT      "EDIFICIOS_IDCOORDENADAS_FK"      FOREIGN      KEY
("IDCOORDENADA")
  REFERENCES "ABUSPAZ"."COORDENADAS" ("IDCOORDENADA") ENABLE
)
)
```

Tabla Espacios

```
CREATE TABLE "ESPACIOS"
( "IDESPACE" VARCHAR2 (20 BYTE) NOT NULL ENABLE,
  "NOMBRE" VARCHAR2 (20 BYTE),
  "DESCRIPCION" VARCHAR2 (90 BYTE),
  "TIPO" VARCHAR2 (20 BYTE),
  "BLOQUE" VARCHAR2 (20 BYTE),
  "VISIBILIDAD" VARCHAR2 (1 BYTE),
  "IDCOORDENADA" VARCHAR2 (20 BYTE),
  "IDEDIFICIO" VARCHAR2 (20 BYTE),
  "PISO" VARCHAR2 (1 BYTE),
  "BOUNDINGBOX" CLOB,
  CONSTRAINT "ESPACIOS_PK" PRIMARY KEY ("IDESPACE")
TABLESPACE "SYSTEM" ENABLE,
  CONSTRAINT "ESPACIOS_IDCOORDENADA_FK" FOREIGN KEY ("IDCOORDENADA")
  REFERENCES "ABUSPAZ"."COORDENADAS" ("IDCOORDENADA") ENABLE,
  CONSTRAINT "ESPACIOS_IDEDIFICIO_FK" FOREIGN KEY ("IDEDIFICIO")
  REFERENCES "ABUSPAZ"."EDIFICIOS" ("IDEDIFICIO") ENABLE
)
```

Tabla Panoramas

```
CREATE TABLE "PANORAMAS"
("IDESPACE" VARCHAR2 (20 BYTE) NOT NULL ENABLE,
  "IDPANORAMA" VARCHAR2 (2 BYTE) NOT NULL ENABLE,
  "PANORAMA" VARCHAR2 (20 BYTE),
  CONSTRAINT "PANORAMAS_PK" PRIMARY KEY ("IDESPACE", "IDPANORAMA")
TABLESPACE "SYSTEM" ENABLE,
  CONSTRAINT "PANORAMAS_IDESPACE_FK" FOREIGN KEY ("IDESPACE")
  REFERENCES "ABUSPAZ"."ESPACIOS" ("IDESPACE") ENABLE
)
```

Tabla Profesores

```
CREATE TABLE "PROFESORES"
( "IDPROFESOR" VARCHAR2 (6 BYTE) NOT NULL ENABLE,
  "NOMBRE" VARCHAR2 (40 BYTE),
```



Servicio Web de Información geoposicionada

```
"CORREO" VARCHAR2 (40 BYTE),
"FICHA" VARCHAR2 (120 BYTE),
"VISIBILIDAD" VARCHAR2 (1 BYTE),
"IDESPACIO" VARCHAR2 (20 BYTE),
"TUTORIAS" CLOB,
"DEPARTAMENTO" VARCHAR2 (120 BYTE),
CONSTRAINT "PROFESORES_PK" PRIMARY KEY ("IDPROFESOR")
TABLESPACE "SYSTEM" ENABLE,
CONSTRAINT "PROFESORES_CORREO_UNIQUE" UNIQUE ("CORREO"),
CONSTRAINT "PROFESORES_IDESPACIO_FK" FOREIGN KEY ("IDESPACIO")
REFERENCES "ABUSPAZ"."ESPACIOS" ("IDESPACIO") ENABLE
)
```

Tabla Asignaturas

```
CREATE TABLE "ASIGNATURAS"
(   "IDASIGNATURA" VARCHAR2 (6 BYTE),
    "NOMBRE" VARCHAR2 (40 BYTE),
    "NOMBRE_ES" VARCHAR2 (40 BYTE),
    "NOMBRE_EN" VARCHAR2 (40 BYTE),
    CONSTRAINT "ASIGNATURAS_PK" PRIMARY KEY ("IDASIGNATURA")
)
```

Tabla Profesorasignatura

```
CREATE TABLE "PROFESORASIGNATURA"
("IDPROFESOR" VARCHAR2 (6 BYTE) NOT NULL ENABLE,
 "IDASIGNATURA" VARCHAR2 (6 BYTE) NOT NULL ENABLE,
 "SITUACION" VARCHAR2 (20 BYTE),
 CONSTRAINT "PROFESORASIGNATURA_PK" PRIMARY KEY ("IDASIGNATURA",
"IDPROFESOR")
TABLESPACE "SYSTEM" ENABLE,
CONSTRAINT "PROFE_ASIG_IDPROFESOR_FK" FOREIGN KEY ("IDPROFESOR")
REFERENCES "ABUSPAZ"."PROFESORES" ("IDPROFESOR") ENABLE,
CONSTRAINT "PROFE_ASIG_IDASIGNATURA_FK" FOREIGN KEY ("IDASIGNATURA")
REFERENCES "ABUSPAZ"."ASIGNATURAS" ("IDASIGNATURA") ENABLE
)
```

-Introducción de la información: En este caso hemos optado por crear un archivo en formato texto plano. Para la posterior lectura de este archivo desde la base de datos hemos seguido las siguientes instrucciones para cada una de las tablas enumeradas anteriormente:

Load data infile “coordenadas.csv” into table coordenadas.

Load data infile “edificios.csv” into table edificios.

Load data infile “espacios.csv” into table espacios.

Load data infile “panoramas.csv” into table panoramas.

Load data infile “profesores.csv” into table profesores.



Desarrollo

Load data infile “asignaturas.csv” into table asignaturas.

Load data infile “profesorasignaturas.csv” into table profesorasignaturas.

5.3.4 Implementación del sistema servidor

El sistema que hemos desarrollado está compuesto por una serie de clases y servicios escritos en java que forman la estructura interna¹² de la aplicación servidora.

La implementación¹³ de las entidades y clases es muy sencilla¹⁴, simplemente debemos definir los atributos necesarios que queramos recuperar de la base de datos y proporcionarle sus respectivos getters y setters, funciones que permitan devolver y modificar los atributos de una manera pública en el objeto. La parte importante del servidor REST se encontrará en la configuración de sus servicios, los cuales debemos definir sus acciones de acuerdo al recurso solicitado por la dirección de acceso que solicitemos, así como los usuarios o máquinas que tienen permitido realizar consultas. La ruta base que disponemos cuando accedemos al sistema servidor es el siguiente:

Dirección del servidor + /ServerRest/

A partir de esta ruta se confecciona toda la estructura que permite el acceso a los distintos recursos y que debemos elaborar siguiendo las directrices de la tecnología REST ya comentada en apartados anteriores.

A continuación, haremos un repaso general de cada uno de los aspectos que componen el mismo.

5.3.4.1 *Coordenadas*

Este servicio dará cuenta de aquellas consultas que soliciten datos referentes a las coordenadas geoposicionadas de algún elemento al que se asocie, ya sea edificio o espacio.

Tal y como comentamos en la fase de diseño, crearemos la dirección a este tipo de recursos mediante la aplicación de una estructura definida en la ruta. En nuestro caso definiremos la ruta a partir de la base como *coordenadas*, obteniendo lo siguiente:

Dirección del servidor + /ServerRest/coordenadas

De esta manera, toda dirección que empiece por la anterior ruta hará referencia a la clase *Coordenadas*, ya que implementaremos sus métodos en este servicio.

Puesto que nuestro interés en este servicio no va más allá de servir las coordenadas recuperadas de la base de datos, sólo construiremos una opción en esta ruta que será la consulta mediante un identificador pasado en la ruta.

Dirección del servidor + /ServerRest/coordenadas/{id}

¹² Véase *Developing RESTful web services with Jersey 2.0* en Bibliografía.

¹³ Véase *Creando servicios restful con netbeans 8* en Bibliografía.

¹⁴ Véase *Jersey 2.23.1 User Guide* en Bibliografía.



Servicio Web de Información geoposicionada

Mediante esta dirección y pasando un identificador de coordenadas valido y presente en la base de datos, el sistema buscará en la clase *Coordenadas* el identificador y devolverá su información asociada.



Desarrollo

5.3.4.2 Espacios

Este servicio proporcionará aquellas consultas que soliciten datos referentes a los espacios asociados a un edificio.

Tal y como comentamos en la fase de diseño, crearemos la dirección a este tipo de recursos mediante la aplicación de una estructura definida en la ruta. En nuestro caso definiremos la ruta a partir de la base como espacios, obteniendo lo siguiente:

Dirección del servidor + /ServerRest/espacios

De esta manera, toda dirección que empiece por la anterior ruta hará referencia a la clase *Espacios* ya que implementaremos sus métodos en este servicio.

Dispondremos dos métodos codificados en la ruta base de espacios los cuales atenderán las necesidades de nuestro sistema cliente. Por ello comentaremos las funciones de consultar por identificador y la consulta de panoramas asociadas al espacio.

Dirección del servidor + /ServerRest/espacios/{id}

Para el primero, la consulta por identificador, no podemos repetir el mismo método que utilizamos en el servicio de coordenadas debido a que necesitamos implementar medidas de verificación sobre la disponibilidad del recurso y si este puede ser servido o no. Por ello, una vez recuperado el recurso solicitado, debemos, previamente a su entrega en el programa cliente, comprobar si el atributo *disponibilidad* es true o false. Si es el primer caso, el programa asistirá el recurso sin mayor impedimento. Si por el contrario es el segundo caso, se construirá un objeto Espacio genérico con información limitada del recurso solicitado que preserve su deseo de no ser localizado.

También implementaremos un método que nos permita solicitar las panorámicas asociadas al espacio actual mediante una dirección unívoca del recurso:

Dirección del servidor + /ServerRest/espacios/{id}/panoramas

Para este servicio emplearemos las NameQueries de la clase *Panoramias* que nos permitirá filtrar por espacio todas aquellas panorámicas que dispongan del mismo identificador presente en la ruta. La NameQuery se traduce en la siguiente consulta:

SELECT p FROM Panoramias p WHERE p.panoramias.idespacio =: idespacio



Servicio Web de Información geoposicionada

5.3.4.3 *Edificios*

Este servicio proporcionará aquellas consultas que soliciten datos referentes a los edificios presentes en la base de datos.

Tal y como comentamos en la fase de diseño, crearemos la dirección a este tipo de recursos mediante la aplicación de una estructura definida en la ruta. En nuestro caso definiremos la ruta a partir de la base como espacios, obteniendo lo siguiente:

Dirección del servidor + /ServerRest/edificios

De esta manera, toda dirección que empiece por la anterior ruta hará referencia a la clase *Edificios*, ya que implementaremos sus métodos en este servicio.

Puesto que nuestro interés en este servicio no va más allá de servir los edificios recuperados de la base de datos, sólo construiremos una opción en esta ruta que será la consulta mediante un identificador pasado en la ruta.

Dirección del servidor + /ServerRest/edificios/{id}

Mediante esta dirección y pasando un identificador de edificios valido y presente en la base de datos, el sistema buscará en la clase *Edificios* el identificador y devolverá su información asociada.

5.3.4.4 *Profesores*

Este servicio proporcionará aquellas consultas que soliciten datos referentes a los profesores presentes en la base de datos.

Tal y como comentamos en la fase de diseño, crearemos la dirección a este tipo de recursos mediante la aplicación de una estructura definida en la ruta. En nuestro caso definiremos la ruta a partir de la base como profesores, obteniendo lo siguiente:

Dirección del servidor + /ServerRest/profesores

De esta manera, toda dirección que empiece por la anterior ruta hará referencia a la clase *Profesores*, ya que implementaremos sus métodos en este servicio.

Dispondremos dos métodos codificados en la ruta base de profesores los cuales atenderán las necesidades de nuestro sistema cliente. Por ello comentaremos las funciones de consultar por identificador y la consulta de asignaturas asociadas al profesor.

Dirección del servidor + /ServerRest/profesores/{id}

Para el primero, la consulta por identificador, emplearemos la misma técnica utilizada en el caso del servicio espacios, verificación sobre la disponibilidad del recurso y si este puede ser servido o no. Por ello, una vez recuperado el recurso solicitado, debemos, previamente a su entrega en el programa cliente, comprobar si el atributo *disponibilidad* es true o false. Si es el primer caso, el programa asistirá el recurso sin mayor impedimento. Si por el contrario es el segundo caso, se construirá un objeto Profesor genérico con información limitada del recurso solicitado que preserve su deseo de no ser localizado.



Desarrollo

También implementaremos un método que nos permita solicitar las asignaturas asociadas al profesor bajo su identificador mediante una dirección univoca del recurso:

Dirección del servidor + /ServerRest/profesores/{id}/asignaturas

Para este servicio deberemos confeccionar una NameQuery que contendrá una consulta SQL que implique el cruce de varias tablas que intervienen en nuestro servicio de tal manera que podamos obtener qué asignaturas de la tabla *Asignaturas* están asociadas a un profesor de la tabla *Profesores* y atravesando la tabla unión de ambas por su cardinalidad muchos a muchos llamada *Profesorasignatura*. La consulta resultante es la siguiente:

```
select ASIG FROM Profesores pr JOIN PR.profesorasignatura PA  
JOIN PA.asignaturas ASIG WHERE PR.idprofesor =:idprofesor"
```

5.3.4.5 Asignaturas

Este servicio proporcionará aquellas consultas que soliciten datos referentes a las asignaturas presentes en la base de datos.

Tal y como comentamos en la fase de diseño, crearemos la dirección a este tipo de recursos mediante la aplicación de una estructura definida en la ruta. En nuestro caso definiremos la ruta a partir de la base como asignaturas, obteniendo lo siguiente:

Dirección del servidor + /ServerRest/asignaturas

De esta manera, toda dirección que empiece por la anterior ruta hará referencia a la clase *Asignaturas* ya que implementaremos sus métodos en este servicio.

Dispondremos dos métodos codificados en la ruta base de asignaturas los cuales atenderán las necesidades de nuestro sistema cliente. Por ello comentaremos las funciones de consultar por identificador y la consulta de profesores asociadas a la asignatura.

Dirección del servidor + /ServerRest/asignaturas/{id}

Para el primero, la consulta por identificador, pasando un identificador de coordenadas valido y presente en la base de datos, el sistema buscará en la clase *Coordenadas* el identificador y devolverá su información asociada.

También implementaremos un método que nos permita solicitar los profesores que imparten la asignatura bajo su identificador mediante una dirección univoca del recurso:

Dirección del servidor + /ServerRest/asignaturas/{id}/profesores

Para este servicio deberemos confeccionar una NameQuery, que contendrá una consulta SQL que implique el cruce de varias tablas que intervienen en nuestro servicio, de tal manera que podamos obtener qué profesores de la tabla *Profesores* *imparten* la



Servicio Web de Información geoposicionada

asignatura de la tabla *Asignaturas* y atravesando la tabla unión de ambas por su cardinalidad muchos a muchos llamada *Profesorasignatura*.

La consulta resultante es la siguiente:

```
select PR FROM Asignaturas ASIG JOIN ASIG.profesorasignatura PA  
JOIN PA.profesores PR WHERE ASIG.idasignatura =:idasignatura
```

5.3.4.6 Panoramas

Este servicio dará cuenta de aquellas consultas que soliciten datos referentes a los panoramas presentes en la base de datos.

Tal y como comentamos en la fase de diseño, crearemos la dirección a este tipo de recursos mediante la aplicación de una estructura definida en la ruta. En nuestro caso definiremos la ruta a partir de la base como *panoramas*, obteniendo lo siguiente:

Dirección del servidor + /ServerRest/panoramas

De esta manera, toda dirección que empiece por la anterior ruta hará referencia a la clase *Panoramas*, ya que implementaremos sus métodos en este servicio.

Puesto que nuestro interés en este servicio no va más allá de servir las panorámicas recuperadas de la base de datos, sólo construiremos una opción en esta ruta que será la consulta mediante un identificador pasado en la ruta.

Dirección del servidor + /ServerRest/panoramas/{id}

Mediante esta dirección y pasando un identificador de panoramas válido y presente en la base de datos, el sistema buscará en la clase *Panoramas* el identificador y devolverá su información asociada.



Desarrollo

5.3.5 Implementación del sistema cliente

Dispondremos de una única página HTML para nuestra aplicación, la cual irá variando su información según los eventos que se vayan produciendo gracias a las distintas librerías JavaScript que emplearemos y que realizarán las correspondientes peticiones¹⁵ al sistema Servidor.

5.3.5.1 *Index.html*

Desde que accedemos a la página principal en ".../SigUV/index.html", el sistema reenvía también a esa página con ".../SigUV/", o realizando una consulta queryString directa, como por ejemplo ".../SigUV/index.html? idprofesor=4", nuestro sistema cliente cargará todos los recursos CSS y JavaScript que vamos a emplear en nuestro proyecto, los cuales son:

- a. CSS

Estilos.css : Hoja de estilos que crearemos a fin de personalizar la representación de nuestra página en base al contenido que mostremos.

Font-awesome.css : Hoja de estilos creada por *Dave Gandy* que nos proporciona iconos vectoriales personalizables en sus distintos atributos de tamaño, color, etc.

Fons.googleapis/icon : Hoja de estilos de la compañía *Google* que nos proporciona iconos vectoriales personalizables en sus distintos atributos de tamaño, color, etc.

Ionicons.css : Hoja de estilos gratuita y de código abierto licenciada por el MIT que nos proporciona iconos vectoriales personalizables en sus distintos atributos de tamaño, color, etc.

Bootstrap.css : Hoja de estilos perteneciente al framework *Bootstrap* que nos permitirá la representación responsive de los elementos de la web mediante clases.

Leaflet.css: Hoja de estilos perteneciente a la librería *Leaflet.js* para mapas interactivos.

Leaflet.awesome-markers.css : Hoja de estilos que da soporte a la personalización de iconos vectoriales pertenecientes a *Font awesome* en la librería *Leaflet.js* para mapas interactivos.

Leaflet.Control.sidebar.css : Hoja de estilos que da soporte a la personalización de barras laterales que proporciona la librería *L.Control.sidebar.js* para *Leaflet.js*

Leaflet.modal.css : Hoja de estilos que da soporte a la personalización de las ventanas modales que proporciona la librería *Leaflet.js*.

Leaflet.easybutton.css : Hoja de estilos que da soporte a la personalización de los botones multipropósito que proporciona la librería *Leaflet-easybutton.js* para *Leaflet.js*.

¹⁵ Véase en *Create a REST client* Bibliografía



Servicio Web de Información geoposicionada

b. Librerías JavaScript

Poi.js: Archivo JavaScript que contiene los datos referentes a Facultades y Campus para los distintos marcadores que se mostrarán en el mapa.

Panoramas.js : Librería creada para gestionar la visualización de panoramas de 360 grados basada en *Three.js* con su extensión *Threex.Fullscreen.js*.

Mapas.js : Librería creada para gestionar la visualización de mapas interactivos basada en la librería *Leaflet.js* y sus librerías asociadas.

Webservice.js : Librería creada para gestionar las solicitudes de recursos que reclamen las distintas librerías, basándose y empleando la librería *JQuery.js*.

Typeahead.js : Librería creada para gestionar las sugerencias provenientes de las búsquedas y sus distintos eventos asociados basándose en la librería *typeahead.bundle.js*.

Jquery.js : Librería JavaScript que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.

Bootstrap.js : Librería JavaScript perteneciente al framework *Bootstrap* que nos permitirá manejar los distintos eventos que se produzcan en la página relacionados con responsive y o representación de los elementos mostrados.

Mapbox.js : Librería JavaScript perteneciente a *Mapbox.com* que nos permitirá acceder mediante su API a los estilos y personalización de mapas provenientes de *OpenStreetMap*.

Raphael.js: Librería JavaScript creada por *Dmitry Baranovskiy* que simplifica el tratamiento y generación de gráficos vectoriales en la web.

Maps.googleapis/maps: Librería JavaScript perteneciente a *Google* que nos proporciona, mediante su API correspondiente, los mapas de sus servicio *GoogleMaps*.

Leaflet.js: Librería JavaScript de código abierto empleado en la representación de mapas interactivos y con capacidad responsive.

Leaflet.google.js: Plugin JavaScript que, en combinación con la librería principal *Leaflet.js* permite la utilización de los mapas de *GoogleMap* en el entorno Leaflet.

Leaflet.control.sidebar.js: Plugin JavaScript que, en combinación con la librería principal *Leaflet.js* permite la creación y uso de barras laterales en el entorno Leaflet.

Leaflet.awesom-markers.js: Plugin JavaScript que, en combinación con la librería principal *Leaflet.js* permite el uso de iconos vectoriales *Font awesome* en el entorno Leaflet.

Leaflet.easybutton.js: Plugin JavaScript que, en combinación con la librería principal *Leaflet.js* permite la creación y uso de botones multipropósito en el entorno Leaflet



Desarrollo

Leaflet.rlayer.js: Plugin JavaScript que, en combinación con la librería principal *Leaflet.js* permite la gestión y uso de la librería *Raphael.js* entorno Leaflet.

Handlebars.js: Librería de JavaScript basado en *Mustache Templates*, que nos sirve para generar plantillas HTML a partir de objetos con datos en formato JSON.

Clipboard.js: Sencilla Librería de JavaScript que permite la copia de texto al portapapeles del sistema operativo con independencia del dispositivo.

Typeahead.bundle.js: Librería de JavaScript desarrollada por *Twitter* que proporciona la capacidad de sugerir resultados posibles a una entrada dada por el usuario.

Three.js: Librería de JavaScript para crear y mostrar gráficos animados por ordenador en 3D en un navegador Web utilizando elementos canvas de HTML5, SVG o WebGL.

Three-FullScreen.js: Plugin JavaScript que, en combinación con la librería principal *Three.js* permite la gestión y uso del evento pantalla completa en el entorno Threejs.

c. Html

Utilizando el Sistema de etiquetas HTML en sinergia con el framework Bootstrap, elaboraremos la estructura principal que consistirá en dos partes: una barra de navegación en la parte superior, que albergará las distintas opciones con las que el usuario interactuará durante la ejecución, y el espacio restante, que contendrá la visualización del mapa generado por las librerías JavaScript. De igual manera, dejaremos creadas las barras laterales que contendrán la información y se mostrarán cuando las preceptivas funciones hagan referencia a ellas.

5.3.5.2 *Mapas.js*

Una vez nuestra página se carga completamente, se llama al inicializador del objeto *mapas* que comenzará con una consulta al objeto *webservice* para que recoja las *queryString* que puedan estar presentes en la URL, devolviéndonos la pareja clave-valor si existe query y un error en caso de carecer de la misma.

De existir consulta, guardaremos los valores de posicionamiento en las variables del mapa para su inicialización, en caso contrario, realizaremos una instancia del mapa con valores predefinidos que sitúan el mapa sobre la ETSE mostrando el nivel 0 del plano.

Después de instanciar el mapa, comenzaremos a agregarle todas las funcionalidades y capas necesarias para nuestro sistema como pueden ser los fondos de Mapbox, Google y los planos ETSE propios, dejando por defecto el fondo Mapbox. Instanciaremos 3 barras laterales que añadiremos a nuestro objeto mapa y tendrán la siguiente finalidad:

- sidebarLayers: mostrará el cuadro de opciones que permita la alteración, a gusto del usuario, de las distintas posibilidades de representación del plano y/o mapa.
- sidebarFacultades: mostrará un listado de las distintas facultades presentes en la base de datos.
- sidebarInfo: mostrará toda la información relativa a profesores y espacios que se recupere de sus sendas consultas.



Servicio Web de Información geoposicionada

Seguidamente, crearemos conjuntos de capas que nos ayuden a agrupar los distintos marcadores que mostraremos en el mapa a fin de poder activar y desactivarlos todos los pertenecientes de forma sencilla. Serán los siguientes:

- layerGroupFac: Agruparemos los marcadores destinados a las facultades.
- layerGroupCam: Reuniremos los marcadores destinados a los campus.
- layerGroupGPS: Comprenderá los marcadores de señal GPS.
- layerGroupSearch: Definirá los marcadores de búsqueda de espacios.

Creados los conjuntos de marcadores, debemos instanciar los distintos marcadores que formarán parte de las facultades y campus respectivamente. Para ello, leeremos del archivo poi.js los Array Facultades y Campus que poseen la información relativa a los distintos elementos. Instanciaremos un marcador por cada entrada del Array y le asignaremos la información contenida en la misma la cual deberá ser revelada al pulsar sobre el marcador. En el proceso de creación, indicaremos mediante awesome-icons cuál debe ser el aspecto para el marcador, en nuestro caso, rojo con sobrero de graduación para las facultades y verde con edificio institucional para los campus.

Posteriormente, debemos instanciar el botón multipropósito *marcapáginas* de easybutton.js y añadirlo al objeto mapa indicando que su tarea es mostrar u ocultar, según el estado que se encuentre, la barra lateral de información *sidebarInfo* que previamente hemos comentado. Igualmente añadiremos el oyente de zoom, el cual nos ejecutará la función de retornar el valor de zoom actual en el mapa para, en el caso de ser mayor al valor 17, el mapa nos revelará sólo los marcadores de campus y si, por el contrario es menor a dicho valor, sólo se mostrarán aquellos marcadores de facultad.

Finalmente, realizaremos la gestión de la queryString que anteriormente hemos recuperado y tratado. Para ello discriminaremos si es una búsqueda con clave-valor y, de ser así, comprobar si estamos ante una búsqueda de espacio o de profesor, o si carece de query. Se realizarán las siguientes acciones a consecuencia de lo anteriormente citado:

- Espacio: Obtenidos los valores para realizar la búsqueda, se realizará la llamada Ajax mediante método GET y tipo JSON a la dirección

Dirección del servidor + /webresources/espacios/ + id del recurso

La llamada a dicho servicio, nos devolverá un objeto JSON con la información relativa al espacio contenido en el identificador. Esta información modificará los valores de posición del mapa para llevarlo al indicado, abrirá la barra lateral de información pasándole en el argumento del objeto recuperado y hará visible el botón *marcapáginas* para poder ocultar y mostrar dicha barra lateral.

- Profesor: Obtenidos los valores para realizar la búsqueda, se realizará la llamada Ajax mediante método GET y tipo JSON a la dirección

Dirección del servidor + /webresources/profesores/ + id del recurso

La llamada a dicho servicio, nos devolverá un objeto JSON con la información relativa al profesor contenido en el identificador. Esta información modificará los valores de posición del mapa para llevarlo al indicado, abrirá la barra lateral de información



Desarrollo

pasándole en el argumento del objeto recuperado y hará visible el botón *marcapáginas* para poder ocultar y mostrar dicha barra lateral.

Si estuviéramos en el caso de que no exista query, llamaremos a la función DefaultMap() que asignará los valores predefinidos del sistema que nos muestra el plano del nivel 0 de la ETSE que ya hemos comentado.

Seguidamente pasaremos a comentar de una forma más concreta la ejecución de las funciones más importantes de las que hacemos uso:

a. OpenSidebarInfo

Mediante esta función conseguiremos que un espacio o profesor sea representado en la barra lateral correspondiente a la información de recursos.

Comienza discriminando el tipo de recurso recibido en el argumento. Si el recurso es un espacio, el tratamiento es realmente sencillo, se llama a la función getPanoramas perteneciente a *webservice.js* que nos devolverá la lista de panoramas asociados al espacio, se pinta la información en los apartados creados a tal efecto en la ficha.

Sin embargo, si el recurso es un profesor, el tratamiento es algo más elaborado. Se llaman a sendos métodos de *webservice.js* que nos devolverá las asignaturas asociadas al profesor y los panoramas asociados al espacio que ocupa el profesor, se rellenan los preceptivos apartados de las fichas, teniendo especial tacto en colocar la información referente a tutorías según si tiene un horario fijo anual, si es semestral y si el profesor participa en el programa de tutorías electrónicas. También se mostrarán las asignaturas con las que el profesor tenga algún tipo de relación, ya sea profesor titular de la asignatura o profesor asistente de laboratorio.

Finalmente, sea cual fuere el recurso, se creará un marcador mediante *AddMarker()* para mostrar el marcador correspondiente al espacio buscado y haciendo visible nuestro botón *marcapáginas* que permita mostrar u ocultar la barra de información.

b. ChangeMapLayer

Esta función se ejecuta cada vez que el usuario altera alguno de los siguientes atributos:

- Fondo del mapa.
- Topónimo del mapa.
- Tema del mapa.
- Nivel del plano.

La finalidad será actualizar la representación del plano y mapa a los valores globales que estén activos en ese momento. Para ello redefiniremos la capa del plano indicándole los valores globales en la siguiente url:

servidor + /mapas/ + _tema + _toponimo + _nivelActual + /{z}/{x}/{y}.png

Finalmente le indicaremos nuevamente cuales son los atributos de zoom máximo y mínimo y lo asociaremos al objeto mapa.

Existe una condición adicional que nos ocultará el área delimitadora de espacio cuando el piso seleccionado no coincida con el piso en el que se encuentre el recurso buscado.



Servicio Web de Información geoposicionada

c. OpenModalPano

Esta función realiza la apertura del visor de panoramas cargando las imágenes asociadas al espacio buscado.

Este visor presentará una estructura sencilla con un espacio donde se observen las panorámicas de 360 grados, un pequeño panel de control que nos permita acceder al modo pantalla completa y un sencillo paginador que nos informe de las imágenes que estén disponibles para el espacio.

Una vez creada la estructura de nuestro visor, se llamará a las funciones *initPanorama()* y *animate()* de *panoramas.js* que iniciarán el proceso de visualización de las imágenes pasadas en el argumento.

5.3.5.3 *webservice.js*

En el momento que nuestra página HTML se carga por completo, el sistema debe recopilar la información referente a los edificios que estén presentes en la base de datos, para ello, realizará una consulta Ajax mediante el acceso a la siguiente dirección del sistema servidor:

Dirección del servidor + /webresources/edificios/

Recuperadas las distintas facultades en forma de objeto JSON, construiremos la estructura que asociando a cada una de las entradas del mismo el evento onclick que invocará la función *setPosition(lat,long,zoom,cierre)* de *Mapas.js* con los valores propios de geoposicionamiento de cada edificio. Tras confeccionar toda la información, añadiremos los elementos en la barra lateral nombrada como *sidebarFacultades*.

A continuación pasaremos a describir las funciones más importantes de las que hacemos uso:

a. LocalizarProfesor

Esta función realiza una conexión Ajax con nuestro sistema servidor consultando la información referente al profesor pasado en el argumento dando como resultado un objeto JSON con dicha información. Para ello accederemos a la siguiente dirección del sistema servidor:

Dirección del servidor + /webresources/profesores/ + id profesor

Una vez obtenida la información, evaluaremos la visibilidad del recurso profesor, si es un objeto considerado disponible, realizaremos las siguientes acciones:

- Llamar a la función *setPosition* de *Mapas.js* pasando los valores de geoposición del despacho del profesor para que el mapa se desplace y centre en dicha localización.
- Invocar al método *AddMarker* de *Mapas.js* con los argumento del emplazamiento del despacho del profesor donde el sistema nos creará un marcador que defina dicha ubicación.
- Convocar al método *OpenSidebarInfo* de *Mapas.js* con los argumentos correspondientes a la información relativa al profesor.



Desarrollo

- Modificar los valores globales de `_nivelActual`, `_nivelBusqueda` de `Mapas.js` y llamaremos a las funciones `setOptionLayer` y `ChangeMapLayer` de `Mapas.js` para actualizar los valores presentes en nuestro objeto mapa y modificar la representación del mapa si fuera necesario.
- Invocar a la función `MostrarArea16` de `Mapa.js` que nos creará un polígono cerrado de color rojo translúcido cuyos vértices corresponderán a las distintas posiciones geográficas almacenadas como Array en el objeto JSON.

Si, por el contrario, la evaluación de visibilidad determina que es un recurso vetado a ser mostrado, el sistema lanzará una ventana modal informando que el recurso al que se intenta acceder está restringido y no se mostrará ningún dato al respecto.

De forma adicional, se realizarán otras tareas menores en importancia que consistirán en realizar un restablecimiento de los valores del buscador o de limpiar la información residual de cualquier otra búsqueda anterior.

b. LocalizarEspacio

Este método es homólogo al antes citado en todos los aspectos a excepción de la dirección de consulta con el sistema servidor la cual será:

Dirección del servidor + /webresources/espacios/ + id espacio

Una vez recibida la información contenida en un objeto JSON, se desencadenarán las mismas acciones del apartado *LocalizarProfesor* con la salvedad de que la información que se representará esta vez en la barra lateral *sidebarInfo* serán espacios lo que supone que deberá crearse con su estructura concreta para este tipo de recurso.

c. ListarDocentesAsignaturas

Esta función realiza una conexión Ajax con el sistema servidor a fin de que nos entregue información en formato JSON de los profesores que imparten la asignatura presente en el argumento con el atributo identificador. Para ello accederemos a la siguiente dirección:

*Dirección del servidor + /webresources/asignaturas/+idAsignatura
+/profesores/*

Una vez realizada la consulta a la dirección anterior, de existir información al respecto, el objeto JSON recibido será enviado al programa principal para que continúe con la ejecución habitual.

d. getAsignaturas

Este método realiza una conexión Ajax con el sistema servidor a fin de que nos entregue información en formato JSON de las asignaturas que imparte el profesor que se encuentre en el argumento con el atributo identificador. Deberemos acudir a la siguiente dirección:

*Dirección del servidor + webresources/profesores/+idProfesor
+asignaturas*

¹⁶ Véase *Raphael* en Bibliografía.



Servicio Web de Información geoposicionada

Realizada la consulta de la dirección, si existen elementos que recuperar de la base de datos, el sistema servidor no entregara un objeto JSON que será enviado al programa principal para continuar con la ejecución habitual.

e. getPanoramas

Este método realiza una conexión Ajax con el sistema servidor a fin de que nos entregue información en formato JSON de los panoramas presentes en el espacio proporcionado en el argumento con el atributo identificador. Para ello, accederemos a la siguiente dirección:

Dirección del servidor + /webresources/espacios/+idEspacio +/panoramas

Realizada la consulta de la dirección, si existen elementos que recuperar de la base de datos, el sistema servidor no entregará un objeto JSON que será enviado al programa principal para continuar con la ejecución habitual.

f. getQueryStringParams¹⁷

Mediante esta función obtendremos el binomio clave-valor presente en la dirección de la aplicación de modo que podamos conocer aquella consulta directa al que el usuario desea acceder a un recurso de forma directa.

Primeramente recogeremos aquella información de la dirección de nuestra aplicación cliente que forme parte de la queryString, es decir, toda cadena de texto presente a la derecha del carácter “?”.

De no existir cadena de texto, supondremos que estamos accediendo de la manera por defecto y devolveremos la ejecución al programa principal para continuar la ejecución. Sin embargo, si existe texto, separaremos el mismo en distintas subcadenas de clave-valor tomando como elemento diferenciador el carácter “&”.

En este punto deberemos recorrer las subcadenas para evaluar si las claves presentes son válidas y/o aceptadas por nuestro sistema las cuales son “idespacio” e “idprofesor”.

En cada iteración necesitaremos separar en cada subcadena el nombre de la clave y el contenido del valor, por ello realizaremos una nueva distinción, esta vez tomando el carácter “=”. Comprobaremos la validez de las claves comparándolas con las antes enunciadas y, de ser correcto, enviará finalmente el binomio al programa principal para continuar con la ejecución habitual.

Se introduce una medida de seguridad ante la presencia de varias claves válidas presentes en la dirección por la cual, de encontrar más de una referencia de espacio o profesor, fuera cual fuere su disposición, sólo sería atendido y enviado al programa principal la primera ocurrencia encontrada, descartando el resto de subcadenas.

¹⁷ Véase *How to Write Valid URL Query String Parameters* en Bibliografía.



Desarrollo

g. MostrarURL

Este método proporcionará una ventana modal que contenga la dirección completa que permita un acceso directo al presente recurso, ya sea profesor o espacio, además de facilitar la tarea de copiar la misma al portapapeles de nuestro sistema operativo.

Simplemente debemos discriminar qué recurso estamos tratando, ya sea profesor o espacio, para conocer cuál es la construcción de dirección que debemos efectuar. Si disponemos de un recurso perteneciente a profesor formaremos la dirección mediante la siguiente estructura:

Dominio base + /SigUV/index.html?idprofesor = + id recurso profesor

Si por el contrario es de un recurso espacio, formaremos la misma con una estructura parecida:

Dominio base + /SigUV/index.html?idespacio = + id recurso espacio

Tras construir la dirección colocaremos la información en la ventana modal que mostraremos y la cual dispondrá de un botón que permita copiar la dirección en el portapapeles del sistema operativo.

5.3.5.4 Typeahead.js

Con esta librería trataremos de gestionar los procesos que intervienen durante la acción de búsqueda de algún recurso que proporcionen sugerencias, elementos o cualquier otra información que sirva de apoyo al usuario en el curso de su búsqueda. Para ello nos apoyaremos en la librería JavaScript *typeahead.bundle.js* que nos proporcionará el motor de búsqueda necesario y en la librería JavaScript Handlebars.js que nos permitirá dar un aspecto uniforme mediante plantillas en la acción de mostrar nuestras sugerencias al usuario que las solicita.

Esta librería entra en funcionamiento tras la ejecución del método Buscador() de *typeahead.js* cuyo proceso y demás métodos serán explicados a continuación.

a. Buscador()

Esta función nos mostrará una ventana modal que contendrá un sencillo formulario organizado en 4 pestañas que corresponden a los distintos recursos a buscar, lo cuales son, espacios, profesores, asignaturas y una pestaña extra con los tres anteriores juntos. Además, dispondrá de un cuadro de texto donde introducir la correspondiente búsqueda del usuario y un botón con el que confirmar nuestra elección.

Finalmente, invocaremos a la función *typeahead()* de *typeahead.js* que proporcionará la información relativa a la estructura antes mencionada.



Servicio Web de Información geoposicionada

b. Typeahead()¹⁸

Este método actuará como inicializador de nuestro motor de sugerencias que presentará a nuestro usuario las opciones más cercanas a su información introducida en el cuadro de texto del formulario presente en la ventana modal.

Debemos comenzar creando sendos objetos por cada aspecto que deseamos agregar la capacidad de sugerir opciones. Por lo tanto, deberemos reclamar a nuestro sistema servidor los elementos que vamos a integrar en la búsqueda mediante las siguientes direcciones:

- *Dirección del servidor + /webresources/profesores/*
- *Dirección del servidor + /webresources/espacios/*
- *Dirección del servidor + /webresources/asignaturas/*

Creados los distintos objetos¹⁹, mediante las instrucciones de tokenizer²⁰ filtraremos cada uno de los registros por el nombre presente que es lo que nosotros queremos sugerir en las búsquedas. Tras esto, integraremos nuestro objeto JSON en el motor de sugerencias con la instrucción *Bloodhound*²¹, elemento perteneciente a la librería *typeahead.bundle.js*.

Logrado este objetivo, nos dispondremos a asociar cada uno de nuestros motores en sus respectivas pestañas de la estructura de la ventana modal explicada en el punto anterior. En su definición, impondremos una serie de opciones²² que deben realizarse durante su ejecución y son las siguientes:

- Favoreceremos la sugerencia si permitimos que los caracteres coincidentes entre el texto introducido por el usuario y los nombres presentes en el motor se destaque para una mejor visualización.
- Permitiremos que nuestro motor de búsqueda comience a mostrarnos posibilidades desde el primer carácter que nuestro usuario escriba.
- Las sugerencias dispondrán de una representación que codificaremos en forma de plantilla gracias a la librería Handlebars.js. De esta manera lograremos ofrecer una información más concreta y estructurada a nuestro usuario que podría encontrar problemas ante dos recursos que en principio se llamen igual, pero pertenezcan a departamentos distintos.

Finalmente, agregaremos distintos oyentes²³ que permitirán recoger y tratar la introducción de texto por parte del usuario de forma que renueve las sugerencias de recursos con cada escritura y/o borrado de texto en el formulario correspondiente.

En el caso de las asignaturas, el proceso de sugerencias listará una tabla con los profesores que imparten la materia seleccionada y al departamento al cual pertenecen. Si el docente tuviera la condición de recurso inaccesible, el registro mostraría un color rojo

¹⁸ Véase *A simple jQuery Autocomplete Search Tutorial* en Bibliografía.

¹⁹ Véase *How to use Json objects with twitter bootstrap typehead* en Bibliografía.

²⁰ Véase *Autocomplete Suggestion + Bloodhount* en Bibliografía.

²¹ Véase *Bloodhound* por Harding, J. en Bibliografía.

²² Véase *jQuery Typehead Search: Demo* en Bibliografía.

²³ Véase *jQuery#typehead* por Harding, J. en Bibliografía.

Desarrollo

que, de ser seleccionado, nos informaría que dicho recurso no está disponible para su observación.

5.3.5.5 Panoramas.js

Esta librería nos permite el visionado de las distintas panorámicas de 360 grados pertenecientes a la ETSE y que dicho visionado sea manipulado por el usuario de tal manera que pueda desplazarse de forma radial en toda su extensión. Para ello nos apoyaremos de la librería Three.js y su accesoria Threex.FullScreen que nos permitirá aumentar las capacidades de la librería principal.

a. initPanorama()

Este método nos permite generar toda la representación de los panoramas de 360 grados dentro de nuestro visor de imágenes y permitir que se realicen las distintas interacciones en tiempo real de nuestro usuario. Estas interacciones comprenden las acciones de rotación libre en el eje X e Y además de acercar y alejar la imagen mostrada.

- Comenzamos creando los distintos elementos necesarios para introducir nuestra imagen en el sistema mediante las siguientes acciones:
- Creamos el objeto cámara donde indicaremos, entre otros valores, la relación de aspecto que deseamos emplear. En nuestro caso, obtendremos este valor mediante las dimensiones horizontales y verticales de nuestro dispositivo.
- Instanciaremos el objeto escena que será la encargada de albergar los distintos elementos esenciales para nuestro visor.
- Crearemos los objetos geometría, textura y malla. Mediante la geometría dispondremos de una esfera hueca en el que situaremos al usuario en el centro de la misma. La textura la obtendremos reclamando a nuestro sistema servidor la información relativa a la panorámica concreta que hemos recogido con el argumento de la función, logrando la siguiente dirección:

Dirección del servidor + webresources/panoramas/+idPanorama

Recuperaremos la ruta en la que se encuentra nuestra imagen y la asociaremos a la textura. Finalmente lograremos la malla al combinar la geometría esférica colocando en su interior la textura.

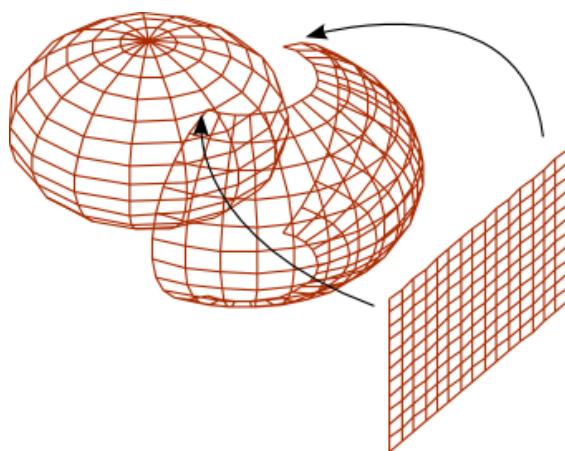


Figura 5.3.5-1: Pegado de textura panorama 360 a objeto 3D esférico



Servicio Web de Información geoposicionada

- Situaremos la cámara en el eje de coordenadas, justo donde se encuentra nuestra esfera a fin de lograr que la primera se encuentre dentro.

Una vez dispuesta nuestra estructura y asociarla pertinente a nuestro visor de panoramas, agregaremos los distintos oyentes que nos permitirán interactuar con la representación de la imagen dependiendo de las acciones de nuestro usuario entre los que se encuentran los siguientes:

- Pulsación de tecla izquierda del ratón.
- Liberación de tecla izquierda del ratón.
- Movimiento de ratón.
- Rueda del ratón.
- Pantalla completa.

Finalizaremos llamando a la función *WindowsResize()* de *panoramas.js* quién adecuará la representación del visor al tamaño de nuestro dispositivo.

b. Change()

Este método es, en parte, similar al comentado en el apartado anterior con la salvedad de que sólo trata el cambio de textura que sufre nuestra geometría esférica, con lo que podemos alterar, a voluntad nuestra, la representación de imágenes en el visor consultando al sistema servidor la dirección del identificador de panorama que se encuentra en el argumento de la función.

c. WindowsResize()

Mediante esta función tratamos que nuestro visor de panoramas siempre sea reescalado, manteniendo la relación de aspecto, a las dimensiones de nuestro dispositivo o navegador. El valor de reescalado siempre será a la mitad del valor original de las dimensiones del dispositivo de tal manera que no ocupe toda la visión y se mantenga el propósito de la funcionalidad e interactividad durante la experiencia del usuario.

d. Serie onDocument...()

Esta serie de funciones actúan como un sistema conjunto que trata de ofrecer interactividad al usuario mediante su ratón en el visionado de panoramas de tal manera que sea libre de realizar las acciones más habituales como son clicar y arrastrar o utilizar la rueda de zoom.

Esta serie de funcionalidades tratan, mediante una lógica muy sencilla, evaluar cuando se encuentra el usuario arrastrando la imagen y cuando no mediante distintas banderas a tal efecto o cuando se está haciendo uso de la rueda del ratón.

Todos estos oyentes actuarán en los valores de nuestra cámara, la cual deberemos actualizar sus valores de sus atributos como pueden ser la dirección de visión o el campo de visión.

e. Update()

Este método pretende actualizar los valores internos de la funcionalidad del visor de panoramas. Puesto que el usuario puede interactuar con el visor en cualquier momento de la ejecución del visor, ejecutaremos esta función con cada fotograma de la escena



Desarrollo

creada a fin de siempre disponer con los valores actuales en la representación de la imagen de 360 grados.

Principalmente, se refrescarán los valores propios de la cámara, concretamente el vector dirección de nuestra cámara que será la dirección en la que apuntará y el campo de visión que será cuánto de esa dirección se debe ver.

Para mantener el sentido de espacio y evitar confusión con una imagen estática impondremos que, si no se está interactuando con la imagen, se aplicará un valor base de rotación en el eje vertical que nos permita visualizar la imagen en su extensión horizontal.



5.3.6 Organización del sitio web

La decisión más básica, y sin embargo a menudo más descuidada, que hay que tomar cuando se empieza a desarrollar una aplicación, es el diseño de directorios a utilizar.

5.3.6.1 Organización de las páginas web

El usuario del sistema realizará toda su navegación a través de una única página llamada index.html y serán las distintas librerías JavaScript las que realizarán toda la generación de contenidos dinámicos que el usuario necesite consultar.

5.3.6.2 Organización de directorios

En la parte del servidor GlassFish distribuiremos la organización de directorios de la siguiente manera:

- a. Sistema Servidor

/siguv/

- Favicon.ico
- Index.html

/siguv/css

- L.Control Sidebar.css
- easyButton.css
- estilos.css
- leaflet.awesome-markers.css
- leaflet.modal.min.css
- leaflet.css

/siguv/css/favicon

- android-icon-144x144.png
- android-icon-192x192.png
- android-icon-36x36.png
- android-icon-48x48.png
- android-icon-72x72.png
- android-icon-96x96.png
- apple-icon-114x114.png
- apple-icon-120x120.png
- apple-icon-144x144.png
- apple-icon-152x152.png
- apple-icon-180x180.png
- apple-icon-57x57.png
- apple-icon-60x60.png
- apple-icon-72x72.png
- apple-icon-76x76.png
- apple-icon-precomposed.png
- apple-icon.png
- browserconfig.xml



Desarrollo

- favicon-16x16.png
- favicon-32x32.png
- favicon-96x96.png
- favicon.ico
- manifest.json
- ms-icon-144x144.png
- ms-icon-150x150.png
- ms-icon-310x310.png
- ms-icon-70x70.png

/siguv/css/images

- Markers-matte.png
- Markers-matte@2x.png
- Markers-plain.png
- Markers-shadow.png
- Markers-shadow@2x.png
- Markers-soft.png
- Markers-soft@2x.png

/siguv/images

- EscudoUV.svg
- bc.png
- bd.png
- cc.png
- cd.png
- google.png
- layers-icon.png
- location.svg
- logoUV.png
- mapbox.png
- signs-1.svg
- signs.svg

/siguv/images/social

- 360-activo.png
- 360-inactivo.png
- copy-activo.svg
- copy-inactivo.svg
- fb-activo.svg
- fb-inactivo.svg
- link-activo.svg
- link-inactivo.svg
- location-inactivo.svg
- tw-activo.svg



Servicio Web de Información geoposicionada

- tw-inactivo.svg

/siguv/js

- Leaflet.js
- Google.js
- L.Control.Sidebar.js
- Clipboard.min.js
- easyButton.js
- handlebars-v4.0.5.js
- mapas.js
- panorama.js
- poi.js
- raphael-min.js
- rlayer.js
- three-FullScreen.js
- three.min.js
- typeahead.bundle.min.js
- typeahead.js
- webservice.js

b. Sistema Cliente

En la parte del Servidor la distribución será:

/ServerRest/Entidad

- Asignaturas.java
- Coordenadas.java
- Edificios.java
- Espacios.java
- Panoramas.java
- PanoramasPK.java
- Profesorasignatura.java
- ProfesorasignaturaPK.java
- Profesores.java

/ServerRest/Servicios

- AbstractFacade.java
- ApplicationConfig.java
- AsignaturasFacadeREST.java
- CORS.java
- CoordenadasFacadeRest.java
- EdificiosFacadeREST.java
- EspaciosFacadeREST.java
- PanoramasFacadeREST.java
- ProfesorasignaturaFacadeREST.java
- ProfesoresFacadeREST.java



Desarrollo

/ServerRest/WEB-INF

- Glassfish-resources.xml

/ServerRest/configuration files

- Manifest.mf
- Glassfish-resources.xml
- Persistence.xml

Pruebas

6 Pruebas

Una vez terminada la fase de implementación debemos realizar diversas pruebas que nos proporcionarán una serie de resultados con los que podremos evaluar cualitativamente el sistema. Las pruebas del sistema deberán contemplar todos los aspectos que se especificaron en el análisis de requisitos de nuestro sistema.

6.1 Correcto funcionamiento del sistema

Para comprobar el correcto funcionamiento del sistema realizaremos diferentes pruebas para cada tipo de búsqueda, mostraremos cada uno de los resultados y comprobaremos la existencia de errores. Haremos algunas capturas de pantalla para que se pueda apreciar mejor lo que percibe el usuario en cada momento de la navegación. Algunas de estas capturas están ampliadas o reducidas en resolución con el fin de que se puedan apreciar mejor ciertos detalles.

6.1.1 Página principal

Cuando el usuario accede a nuestra web de la manera por defecto, se muestra una página inicial en la que el usuario observa una barra de navegación en la parte superior con las opciones de Localizador, Capas, Facultades y Miscelánea, seguido de un mapa que ocupa el resto de la página.

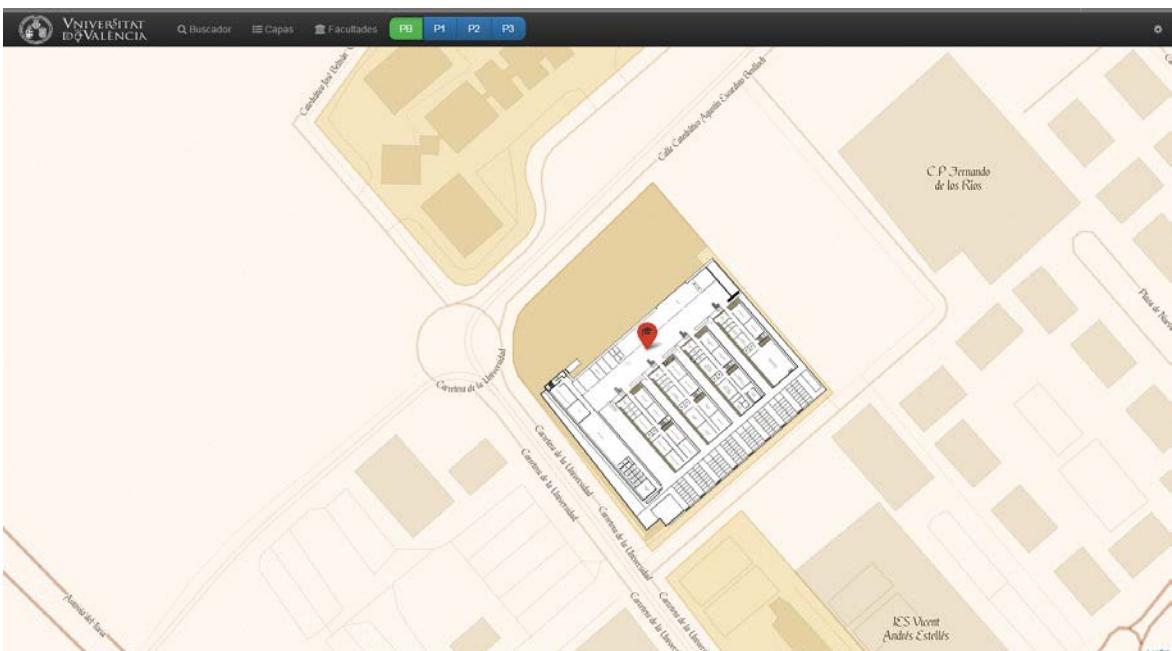


Figura 6.1.1-1: Página inicial de la aplicación web

A continuación, veremos qué ocurre al acceder a las distintas opciones.



Servicio Web de Información geoposicionada

6.1.2 Buscar un recurso

Cuando queramos buscar un recurso dentro de nuestro sistema se nos presentará una ventana modal con 4 apartados dispuestos en fichas. Cada uno de estos apartados permitirá buscar por Profesor, Asignatura, Espacio o por todo lo anterior en una búsqueda conjunta mediante el texto introducido en el formulario dispuesto a tal efecto.

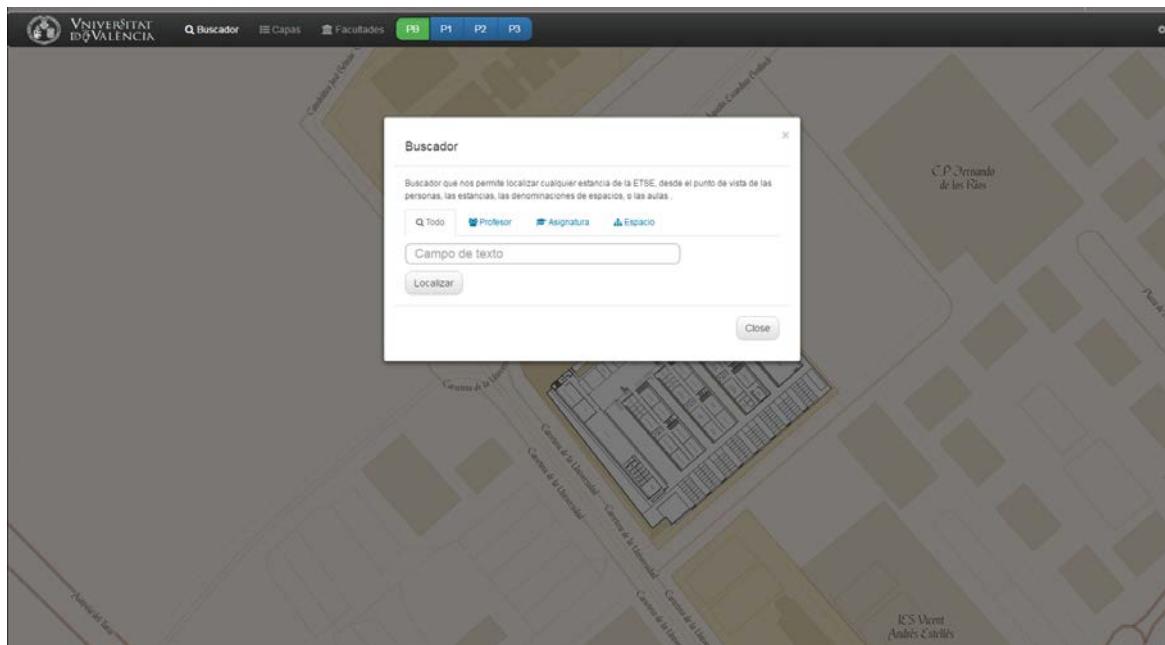


Figura 6.1.2-1: Módulo buscador

En cualquiera de los apartados, el sistema proporciona sugerencias de posibles recursos que coincidan con el texto introducido por el usuario.

De igual manera, si pulsamos en la zona exterior al formulario, saldremos del buscador y retornaremos al mapa por defecto en el punto que lo dejamos si no hemos realizado ningún movimiento previo.

6.1.2.1 Búsqueda por Profesor

Seleccionando la ficha profesor, se dispondrá una barra de introducción de texto que permitirá escribir el nombre del profesor, ya sea nombre o apellidos. Durante la introducción de texto, el sistema nos muestra coincidencias en los nombres presentes en la base de datos.

Pruebas

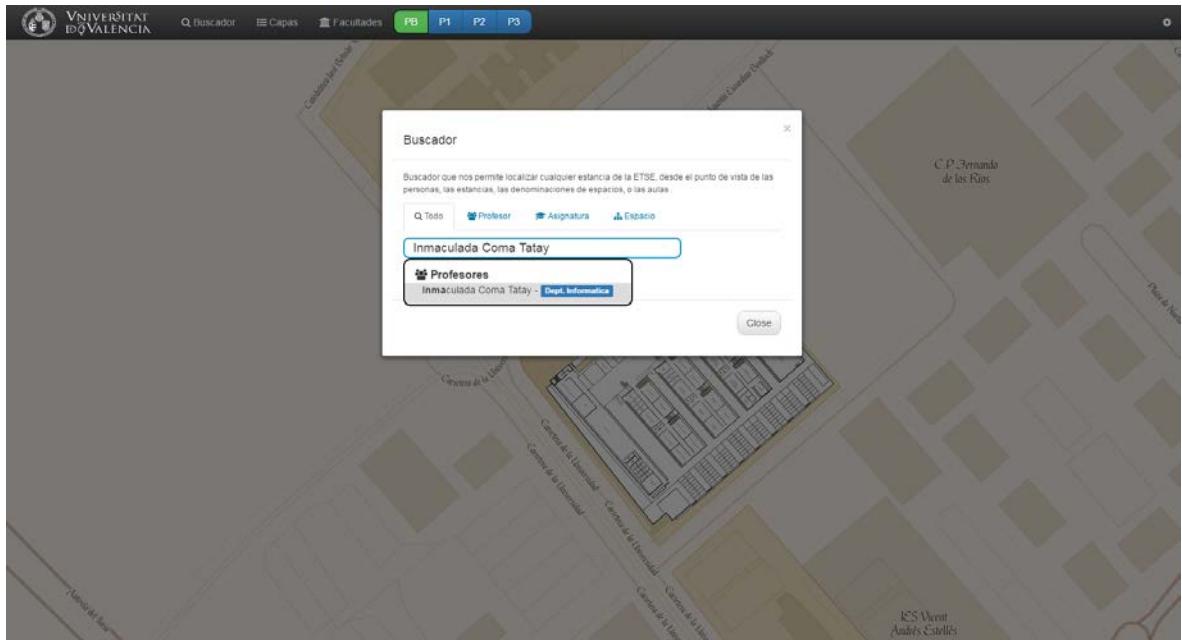


Figura 6.1.2-2: Sugerencia de búsqueda para Espacios y Profesores

Durante la representación de resultados, se visualizan las coincidencias encontradas entre el texto introducido por el usuario y las distintas sugerencias que se ajustan a esos parámetros. De igual manera, se dispone a la derecha del profesor una etiqueta que indique a qué departamento pertenece el docente a fin de evitar posibles confusiones.

Una vez confirmada la búsqueda deseada y siendo esta válida, pasaremos a la situación de mostrar resultados que explicaremos en su apartado correspondiente en “Mostrar resultado de profesor”.

6.1.2.2 Búsqueda por Asignatura

Seleccionando la ficha asignatura, se dispondrá una barra de introducción de texto que permitirá escribir el nombre de la asignatura. Durante la introducción de texto, el sistema nos muestra coincidencias en los nombres presentes en la base de datos.

Durante la representación de resultados, se visualizan las coincidencias encontradas entre el texto introducido por el usuario y las distintas sugerencias que se ajustan a esos parámetros.

Una vez confirmada la búsqueda deseada y siendo esta válida, se nos presentará una lista de profesores que imparten la materia independientemente de su condición de titular o de laboratorio a fin de poder escoger el profesor que estamos buscando.



Servicio Web de Información geoposicionada

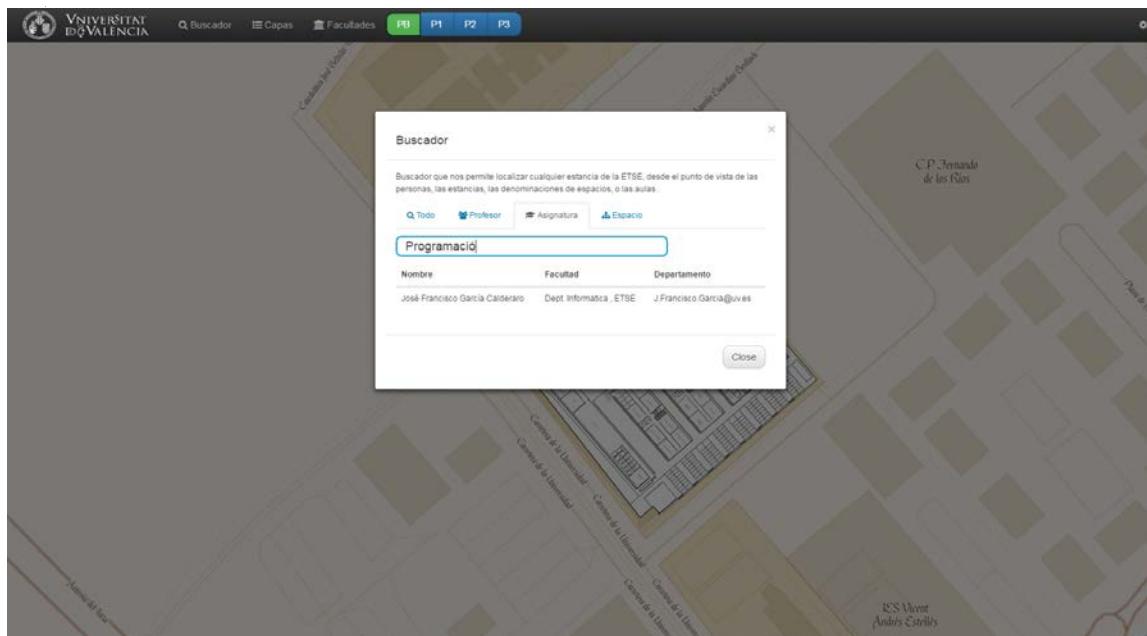


Figura 6.1.2-3: Sugerencia de búsqueda para asignaturas

Si durante la búsqueda de profesores que imparten una asignatura concreta no están disponibles, por la razón que fuera, su registro se mostrará en rojo indicando que es un recurso inaccesible y que no se mostrará información al respecto sobre el mismo.

Una vez confirmada la búsqueda deseada y siendo esta válida, pasaremos a la situación de mostrar resultados que explicaremos en su apartado correspondiente en “Mostrar resultado de profesor”.

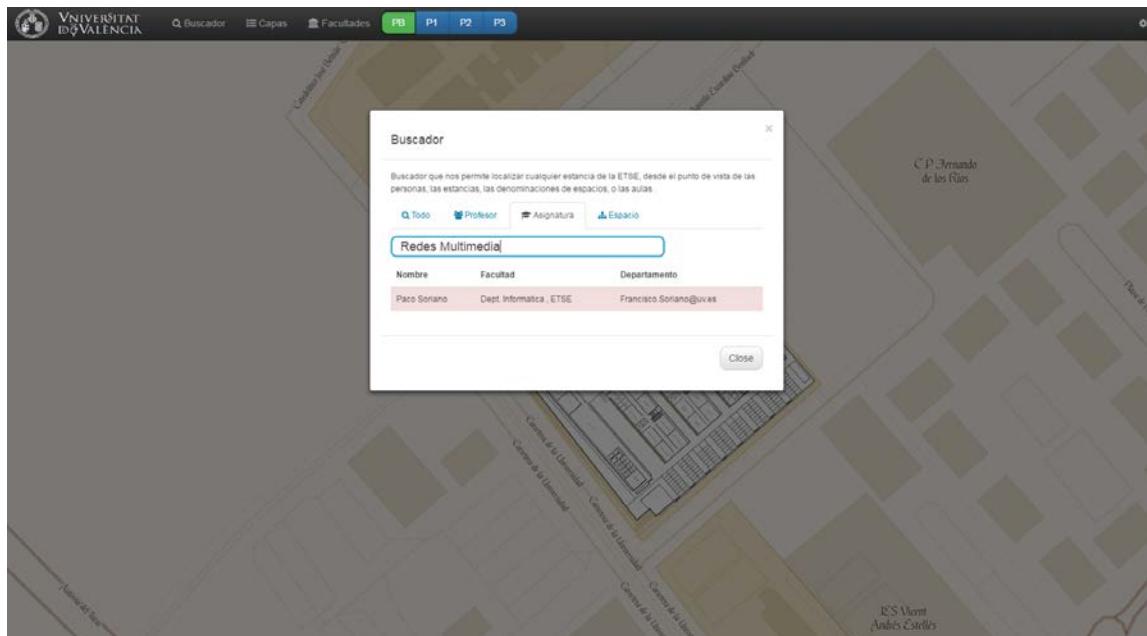


Figura 6.1.2-4: Sugerencia de recurso inaccesible

Pruebas

En cualquier caso, si pulsamos en la zona exterior al formulario, saldremos del buscador y retornaremos al mapa por defecto en el punto que lo dejamos si no hemos realizado ningún movimiento previo.

6.1.2.3 *Búsqueda por Espacio*

Seleccionando la ficha espacio, se dispondrá una barra de introducción de texto que permitirá escribir el nombre del espacio, ya sea por descripción o por su codificación de estancia. Durante la introducción de texto, el sistema nos muestra coincidencias en los nombres presentes en la base de datos.

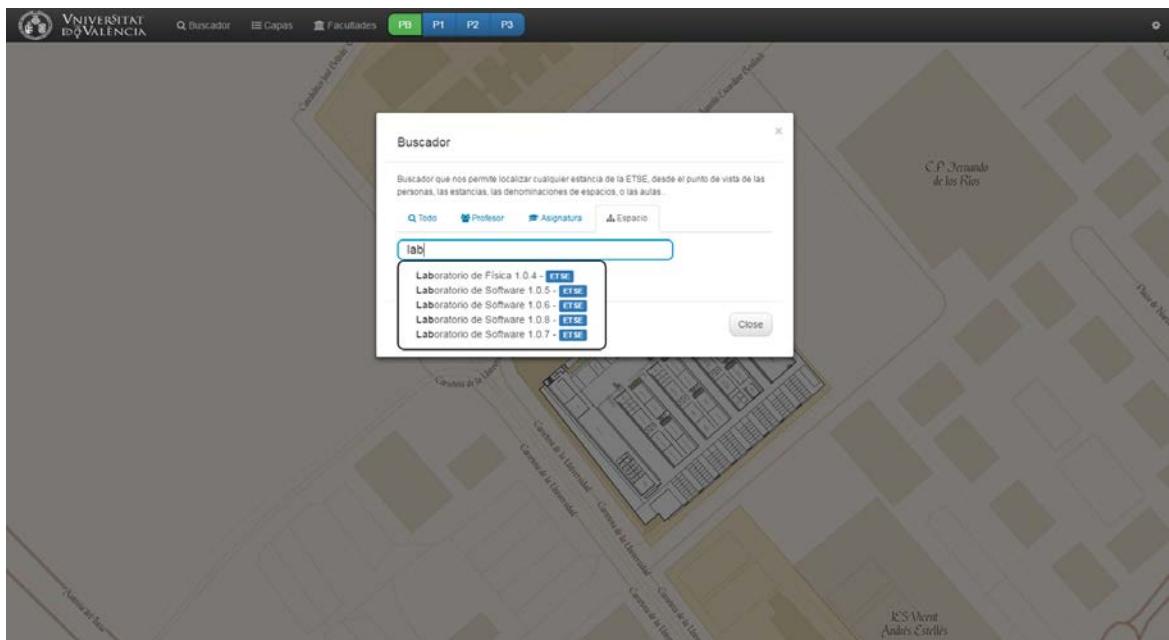


Figura 6.1.2-5: Sugerencia de recurso espacio

Durante la representación de resultados, se visualizan las coincidencias encontradas entre el texto introducido por el usuario y las distintas sugerencias que se ajustan a esos parámetros.

Una vez confirmada la búsqueda deseada y siendo esta válida, pasaremos a la situación de mostrar resultados que explicaremos en su apartado correspondiente en “Mostrar resultado de espacio”.



6.1.2.4 Búsqueda por Todo

Seleccionando la ficha ‘todo’, se dispondrá una barra de introducción de texto que permitirá escribir el nombre de profesor, asignatura o espacio. Durante la introducción de texto, el sistema nos muestra coincidencias en los nombres presentes en la base de datos en sendas secciones dispuestas para cada grupo de objetos.

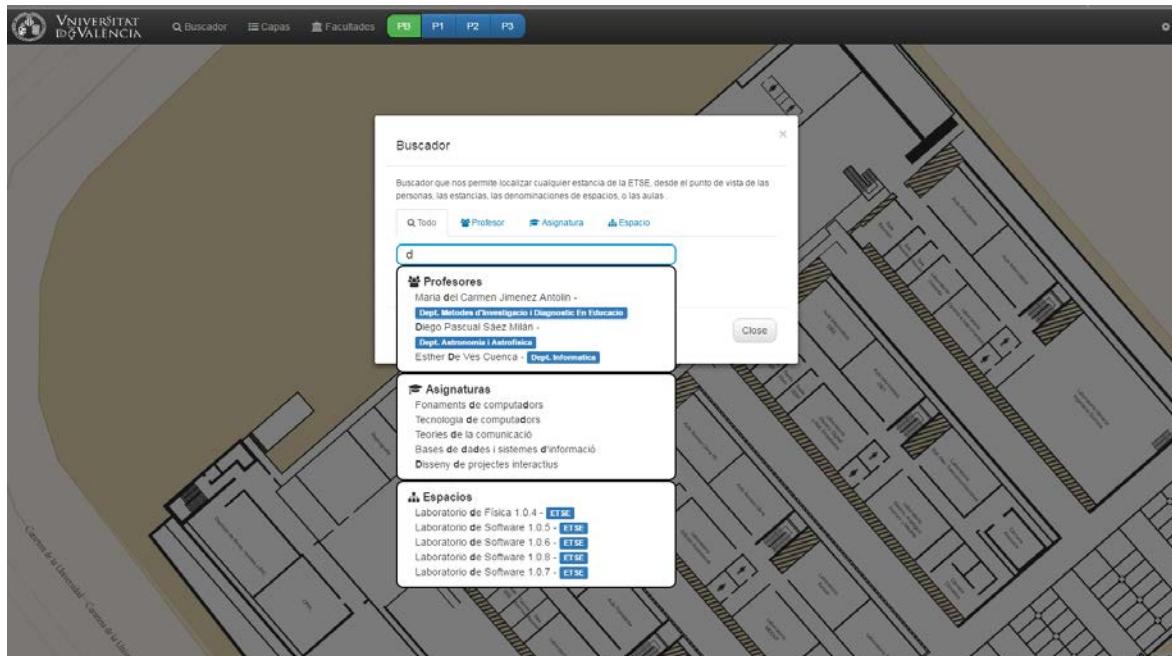


Figura 6.1.2-6: Sugerencia de búsqueda para Todo

Durante la representación de resultados en sus respectivos grupos, se visualizan las coincidencias encontradas entre el texto introducido por el usuario y las distintas sugerencias que se ajustan a esos parámetros de cada grupo.

Una vez confirmada la búsqueda deseada y siendo esta válida, se nos presentará una lista de profesores que imparten la materia, en el caso de seleccionar una asignatura, independientemente de su condición de titular o de laboratorio, a fin de poder escoger el profesor que estamos buscando. En el resto de casos, pasaremos a la situación de mostrar resultados que explicaremos en su apartado correspondiente en “Mostrar resultado de espacio”.

Pruebas

6.1.3 Configurar capas del mapa

En esta opción se nos presentará una ventana lateral a la izquierda del mapa que nos ofrecerá distintas posibilidades de configuración del modelo de vista de nuestro mapa a fin de ajustarlo a nuestras necesidades de búsqueda.

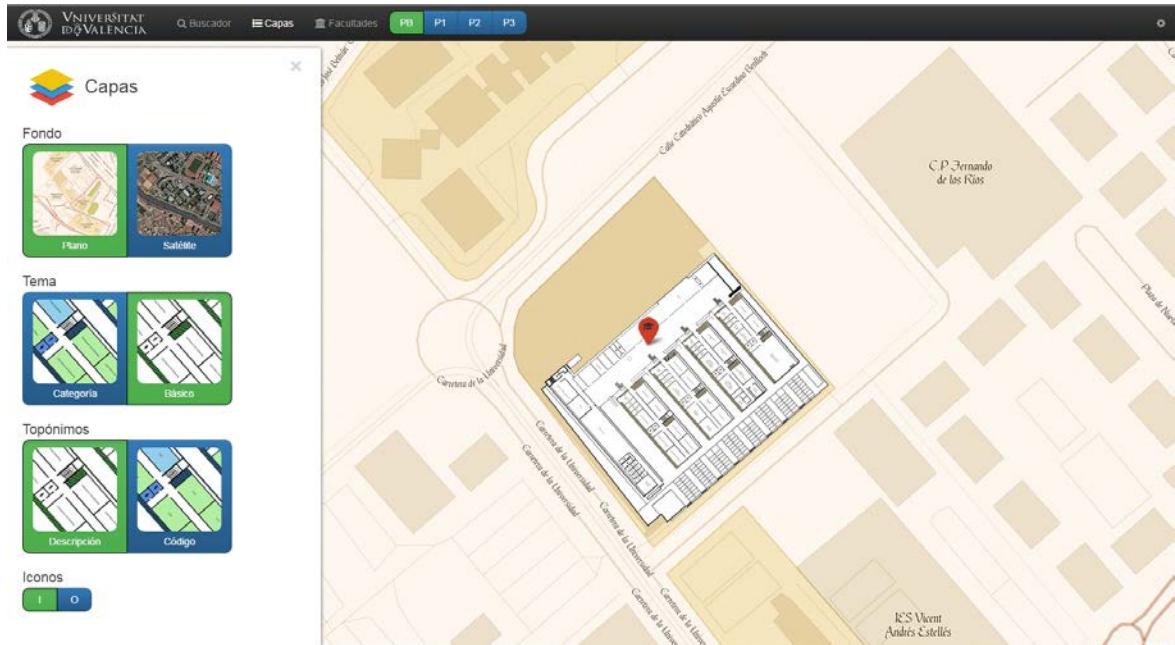


Figura 6.1.3-1: Configurador de capas

Las posibilidades disponibles atienden a las siguientes características:

Fondo: podremos elegir entre un mapa personalizado mediante cartoCSS de los mapas de dominio público OpenStreetMap y el servicio de mapas más popular hasta la fecha, Google Maps.

Tema: Atiende a la posibilidad de mostrar los planos de los edificios en formato B/N o de mostrar los planos con un color codificado según la finalidad del espacio.

Topónimo: Posibilidad de mostrar los espacios por la descripción de su finalidad o mediante su codificación determinada por el edificio.

La selección efectuada es compatible de forma transversal a los tres grupos pero es excluyente dentro del mismo además de que, mientras tengamos control del mapa y podamos manipularlo, seremos capaces de editar su configuración de visualización.

Además, podemos mostrar u ocultar los marcadores de Facultad y Campus bajo las mismas condiciones que el resto de capas.



Servicio Web de Información geoposicionada

6.1.4 Mostrar GPS

Cuando deseamos conocer cuál es la posición de nuestro dispositivo, esta opción nos mostrará un marcador de color azul claro que indicará en la posición aproximada en la que se encuentra el usuario, acomodando la vista del mapa incluyendo nuestra posición.

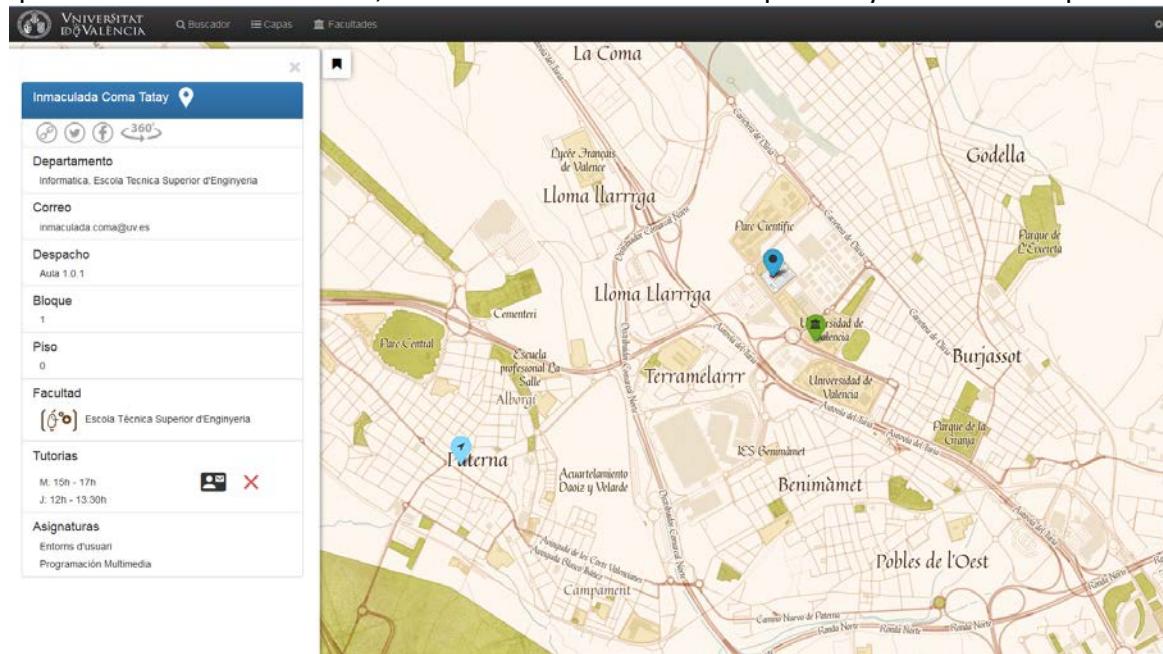


Figura 6.1.4-1: Posición GPS

Si volvemos a seleccionar la misma opción, que en este momento veremos que se presenta como una opción activada, procederemos a desactivar la funcionalidad eliminando el marcador del mapa.

6.1.5 Mostrar Leyenda

Cuando deseamos conocer cuáles son las normas, imágenes e iconos que rigen la información de nuestra aplicación, esta opción nos mostrará una ventana modal que nos enseñará el significado de marcadores, iconos y colores que ayuden a entender el uso dado en la aplicación.

Pruebas

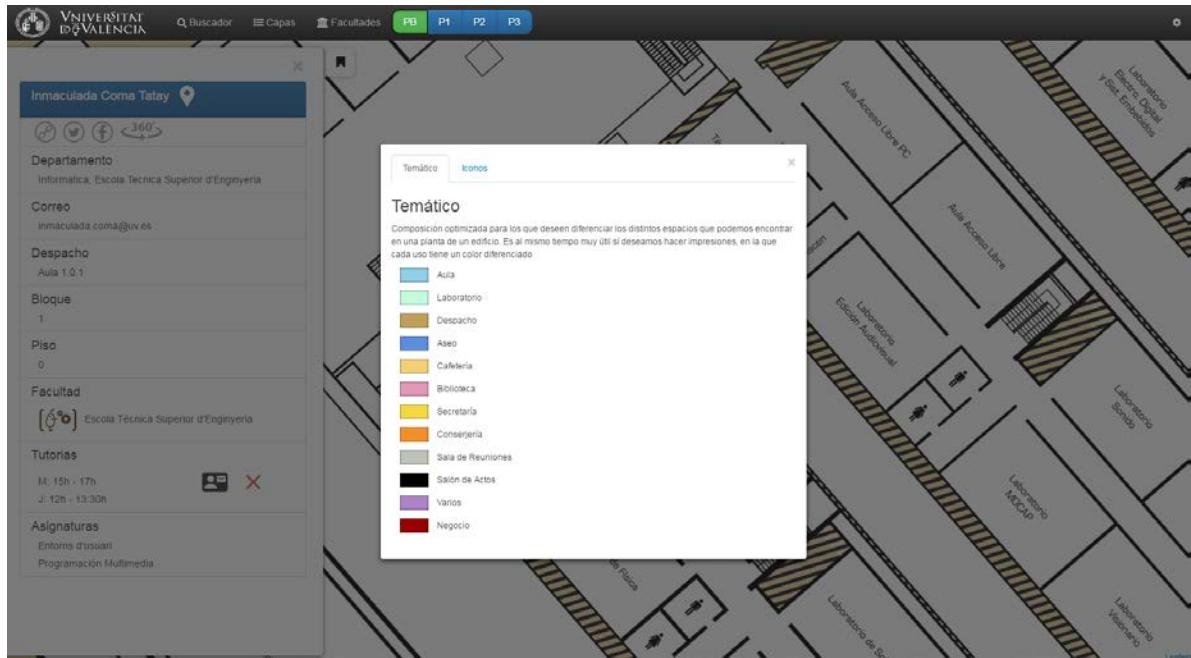


Figura 6.1.5-1: Leyenda

Pulsando en cualquier zona exterior a la ventana modal, cerrará esta ventana devolviendo la visión al estado anterior al que se encontraba antes de acceder a esta opción. De esta manera no se pierde la noción de situación del usuario cuando necesita de conocer algún color o símbolo en el sistema.

6.1.6 Mostrar Ayuda

Cuando deseamos conocer cómo funciona nuestra aplicación y cuáles son sus posibilidades, esta opción nos mostrará una ventana modal que nos enseñará los usos que dispone la aplicación.

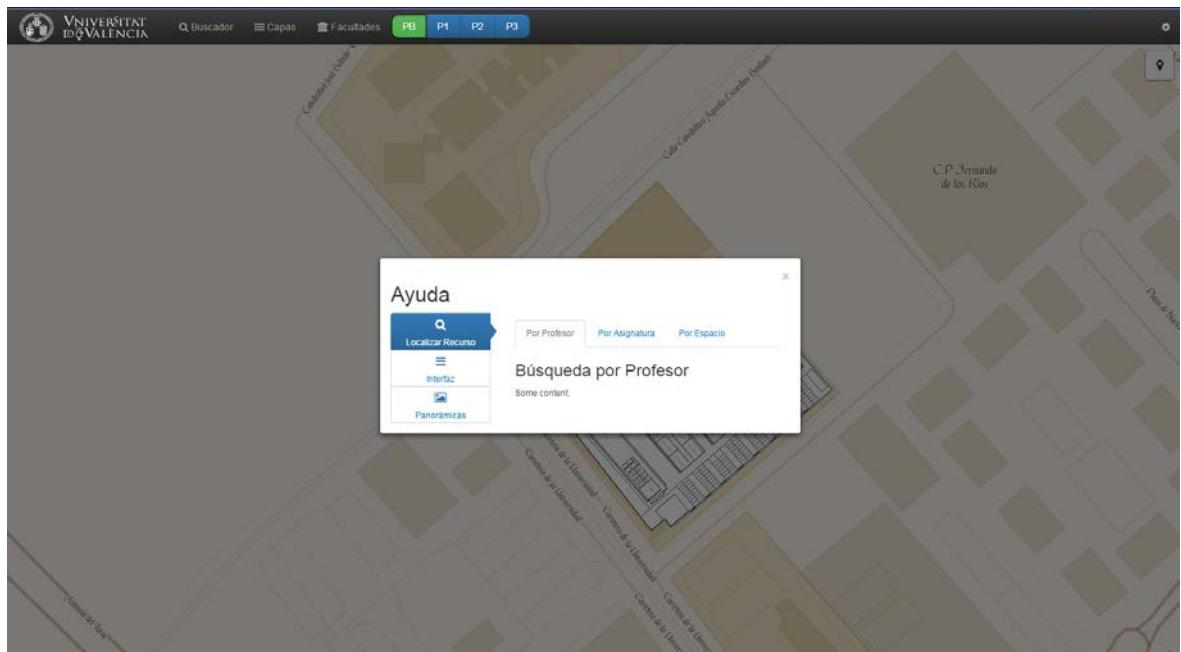


Figura 6.1.6-1: Ayuda



Servicio Web de Información geoposicionada

Pulsando en cualquier zona exterior a la ventana modal, cerrará esta ventana devolviendo la visión al estado anterior al que se encontraba antes de acceder a esta opción. De esta manera no se pierde la noción de situación del usuario cuando necesitó de conocer algún color o símbolo en el sistema.

6.1.7 Listado de Edificios

Cuando queramos obtener un acceso rápido a los edificios presentes en el sistema, al acceder a la opción de *Facultades*, mostraremos en una barra lateral a la izquierda un listado de todos aquellos edificios o emplazamientos introducidos en el sistema.

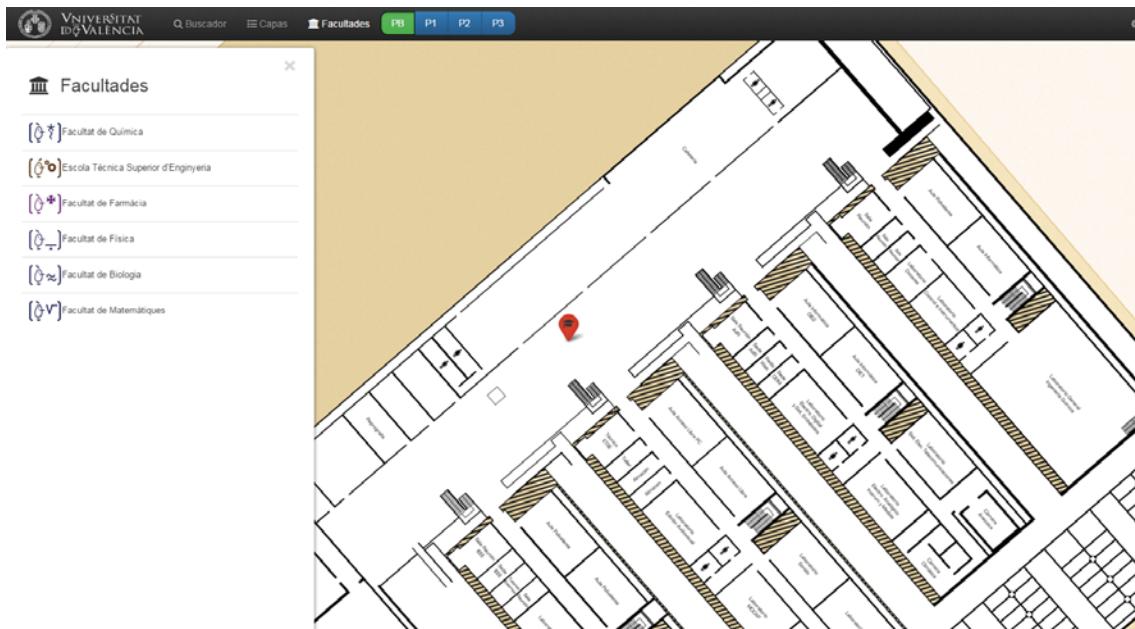


Figura 6.1.7-1: Listado de edificios

Dentro del listado, mostraremos el nombre del edificio y este tendrá adjuntado una imagen correspondiente al chano asociado al mismo. Pulsando en cualquiera de los elementos del listado, seremos desplazados sobre el mapa a la situación geográfica en la que se encuentre la construcción. Una vez desplazado a la posición final dentro del mapa, la barra lateral se ocultará a fin de no obstaculizar la visión.

6.1.8 Resultado de Recurso Profesor

Cuando hayamos seleccionado un recurso profesor y se realice la acción de mostrar la información, se abrirá una barra lateral a la izquierda que presentará los datos referentes al mismo.

Dispone de un botón en la esquina superior derecha de la barra lateral que nos permite ocultar la información para tener amplia visión del mapa y poder recuperarla posteriormente sin tener que volver a realizar la búsqueda.

Esta opción es especialmente útil en dispositivos pequeños ya que permite no obstaculizar la visión del mapa por la oclusión efectuada por la información de la barra lateral. De esta manera podemos ocultar la información para poder observar el espacio marcado en el mapa y, posteriormente, recuperar los datos concernientes al mismo.

Pruebas

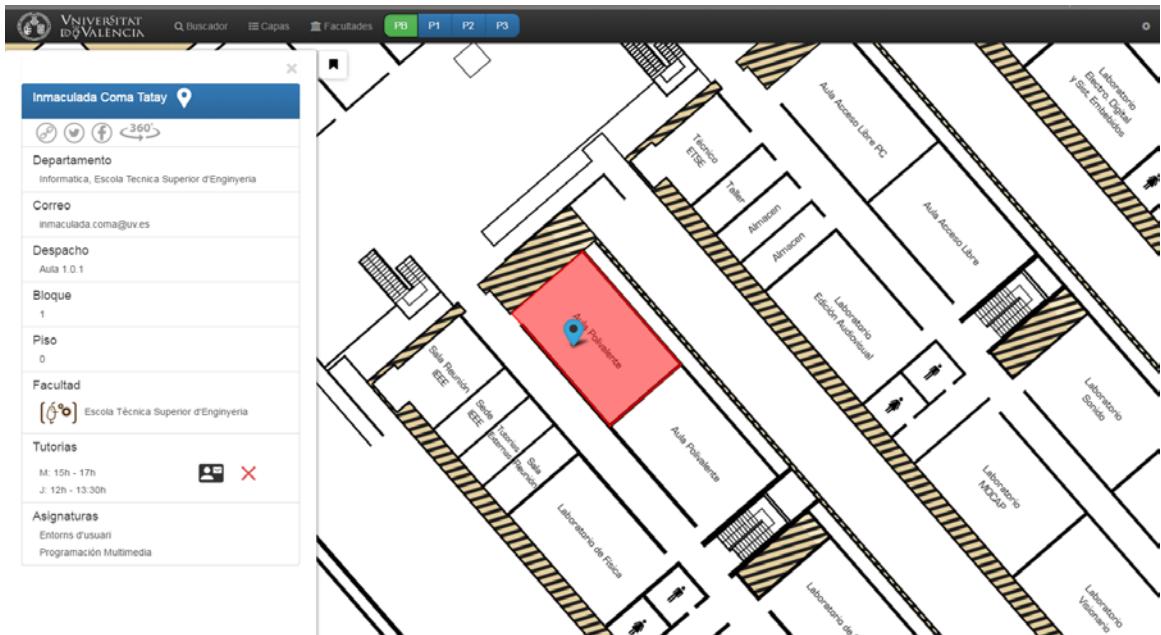


Figura 6.1.8-1: Información Resultado de Profesor

Entre los datos que se mostrarán estarán:

- El departamento al que pertenece.
- El correo del profesor con el que tendremos la posibilidad de enviar un correo desde el propio navegador si tenemos configurado un gestor de correo y no hemos bloqueado este tipo de eventos.
- El Despacho habitual donde se encuentra el profesor en horario de tutorías.
- El Bloque al que pertenece el despacho.
- El Piso en el que se encuentra el despacho.
- La Facultad a la que pertenece el despacho y el chanjo asociado a dicha facultad.
- El Horario de tutorías que el profesor pone a disposición de los alumnos para consultar dudas. En este punto, se muestran los horarios, anuales o semestrales, y se dispone la información en consecuencia a este hecho. Además, se informará al usuario si el profesor pertenece o no al programa de tutorías electrónicas.
- Las asignaturas en las que el profesor imparte docencia, ya sea como titular o como profesor de laboratorio.

De igual manera, se informa al usuario de las capacidades de compartir el recurso en las redes sociales más conocidas, Facebook y Twitter, o copiar la dirección de acceso directo a la búsqueda del profesor para emplear en otras formas de comunicación como pudiera ser correo electrónico. Además, si el espacio relacionado dispone de panorámicas de 360 grados, mostrará un icono identificativo de que podemos visualizar las imágenes correspondientes.



Servicio Web de Información geoposicionada

6.1.9 Resultado de Recurso Espacio

Cuando hayamos seleccionado un recurso espacio y se realice la acción de mostrar la información, se abrirá una barra lateral a la izquierda que presentará los datos referentes al mismo.

Dispone de un botón en la esquina superior derecha de la barra lateral que nos permite ocultar la información para tener amplia visión del mapa y poder recuperarla posteriormente sin tener que volver a realizar la búsqueda.

Esta opción es especialmente útil en dispositivos pequeños ya que permite no obstaculizar la visión del mapa por la occlusión efectuada por la información de la barra lateral. De esta manera podemos ocultar la información para poder observar el espacio marcado en el mapa y, posteriormente, recuperar los datos concernientes al mismo.

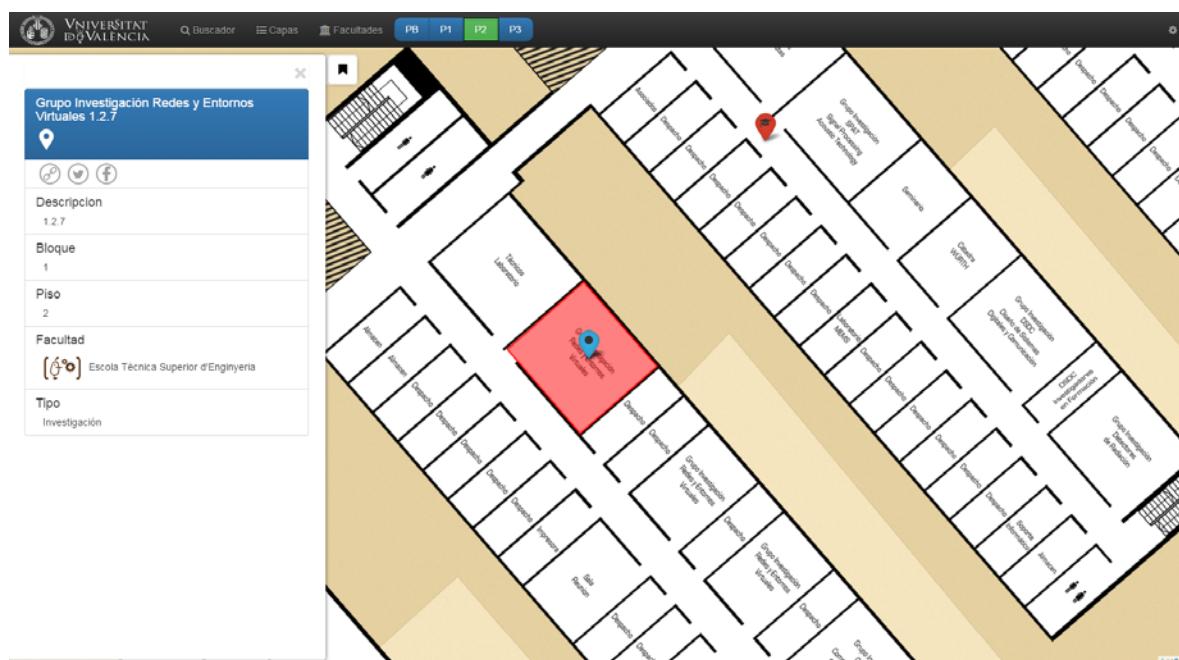


Figura 6.1.9-1: Información Resultado de Espacio

Entre los datos que se mostrarán estarán:

- La descripción o código al que pertenece el espacio.
 - El Bloque al que pertenece el espacio.
 - El Piso en el que se encuentra el espacio.
 - La Facultad a la que pertenece el espacio y el chanoy asociado a dicha facultad.
 - El Tipo de finalidad que tiene el espacio.

De igual manera, se informa al usuario las capacidades de compartir el recurso en las redes sociales más conocidas, Facebook y Twitter, o copiar la dirección de acceso directo a la búsqueda del espacio para emplear en otras formas de comunicación como pudiera ser correo electrónico. Además, si el espacio relacionado dispone de panorámicas de 360 grados, mostrará un ícono identificativo de que podemos visualizar las imágenes correspondientes.

Pruebas

6.1.10 Compartir Recurso

Cuando dispongamos de un recurso, ya sea un profesor o un espacio, y queramos compartirlo a una comunidad mayor sin que deban realizar cada uno de ellos la búsqueda individual. Podemos recurrir a tres opciones que nos permitan dar a conocer el recurso para obtener notoriedad, alcance y difusión de lo que necesitemos comunicar como, por ejemplo, informar a un sector de alumnos donde se encuentra un seminario o informar a un tercero el despacho de tutorías de un profesor dado.

Cuando compartamos con Twitter, tras pulsar el botón correspondiente al símbolo de la conocida red social, se abrirá una ventana nueva que nos precargará la dirección completa al recurso y la posibilidad de extender más información en la publicación bajo nuestro usuario personal de Twitter. Si no disponemos de cuenta o no estamos identificados, la nueva ventana nos impondrá la identificación para poder compartir la dirección. De otro modo, no será posible.

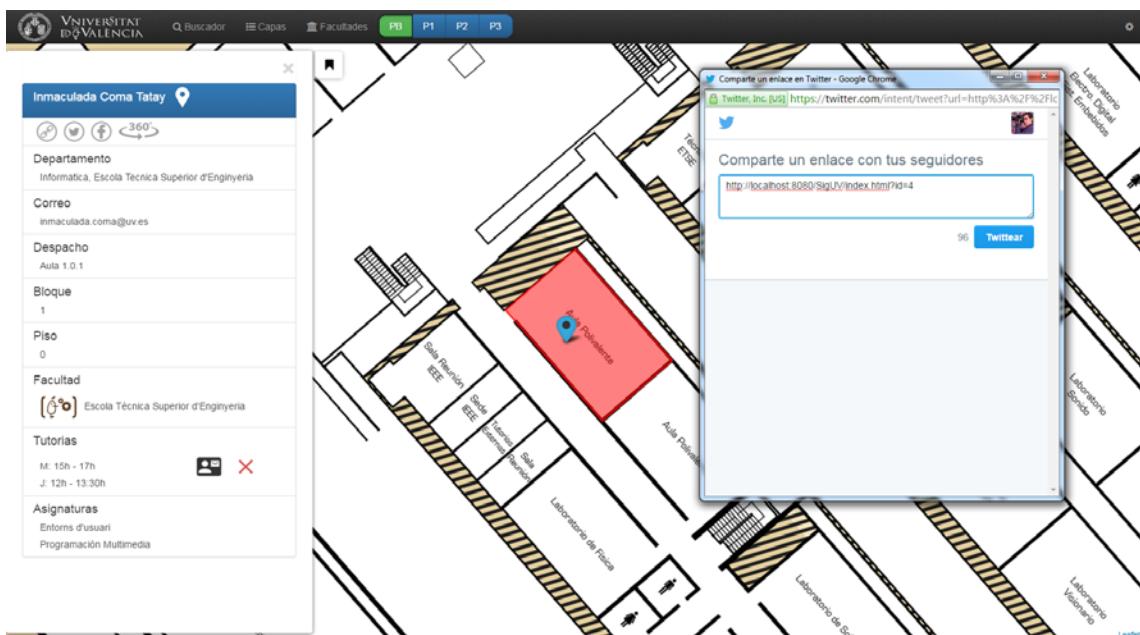


Figura 6.1.10-1: Compartir en Twitter

Cuando compartamos con Facebook, tras pulsar el botón correspondiente al símbolo de la conocida red social, se abrirá una ventana nueva que nos precargará la dirección completa al recurso y la posibilidad de extender más información en la publicación bajo nuestro usuario personal de Facebook. Si no disponemos de cuenta o no estamos identificados, la nueva ventana nos impondrá la identificación para poder compartir la dirección. De otro modo, no será posible.



Servicio Web de Información geoposicionada

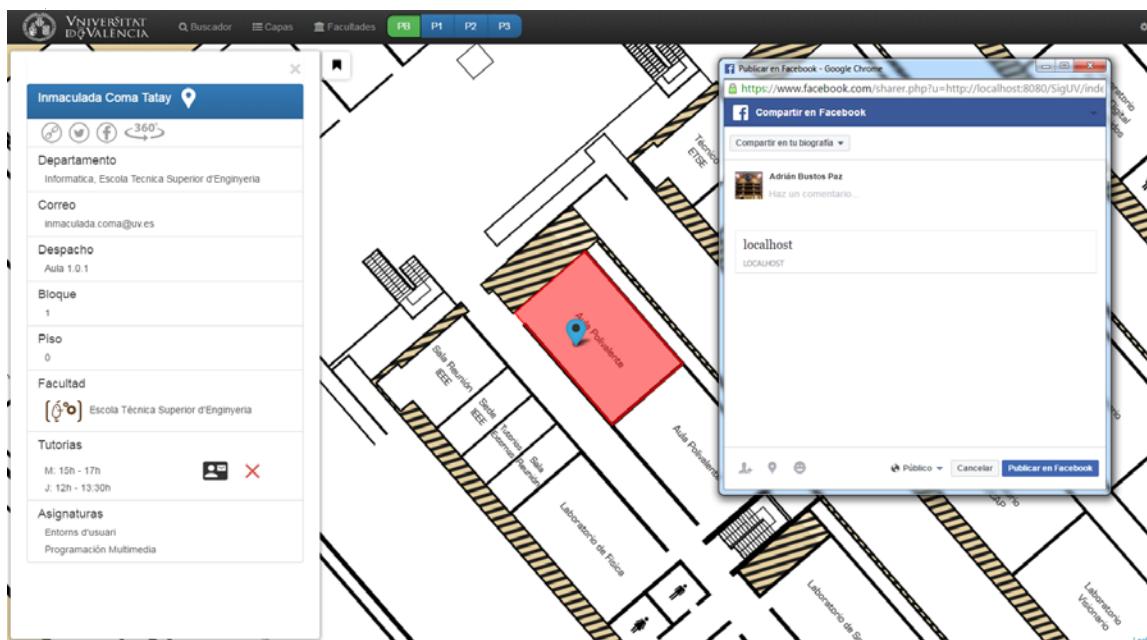


Figura 6.1.10-2: Compartir en Facebook

Cuando deseemos dar un fin distinto al de difusión por las redes sociales más conocidas, por ejemplo, correo electrónico para enviar a uno o varios destinatarios, tras pulsar el botón correspondiente al símbolo de hipervínculo, se abrirá una ventana modal que nos mostrará la dirección completa al recurso y la posibilidad de copiarla al portapapeles mediante un botón a la derecha del texto. De esta manera ya podemos compartir o difundir por los medios que deseemos la dirección al recurso.

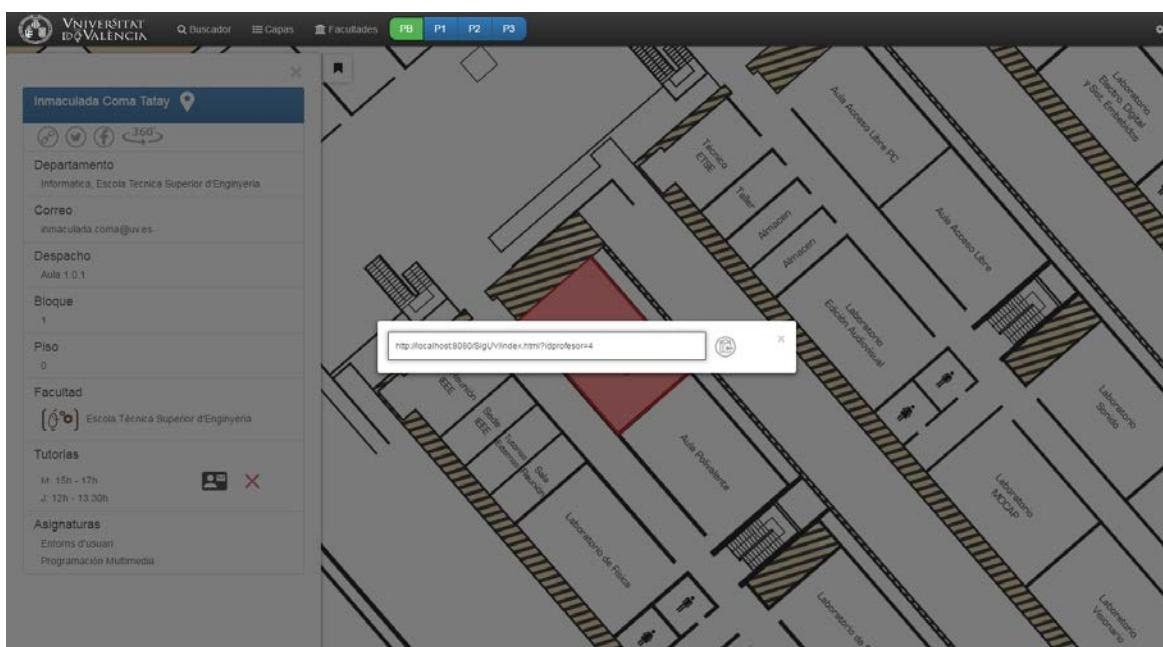


Figura 6.1.10-3: Compartir Enlace

Pruebas

6.1.11 Visualizar Panoramas de 360 grados

Cuando queramos visualizar un espacio mediante imágenes de 360 grados y el mismo nos lo permite porque dispone de ellas, pulsaremos sobre el icono correspondiente a tal efecto. Se nos mostrará una ventana modal en el que se encontrará el visor de imágenes, en el que podremos interactuar arrastrando sobre la imagen para efectuar los cambios de vista rotando desde la posición del observador. También podremos realizar zoom de aproximación y de alejamiento mediante la rueda de ratón. Si no efectuamos ninguna interacción en el visor, este mantendrá una velocidad constante de rotación que nos mostrará de forma horizontal toda la estancia de forma continua.

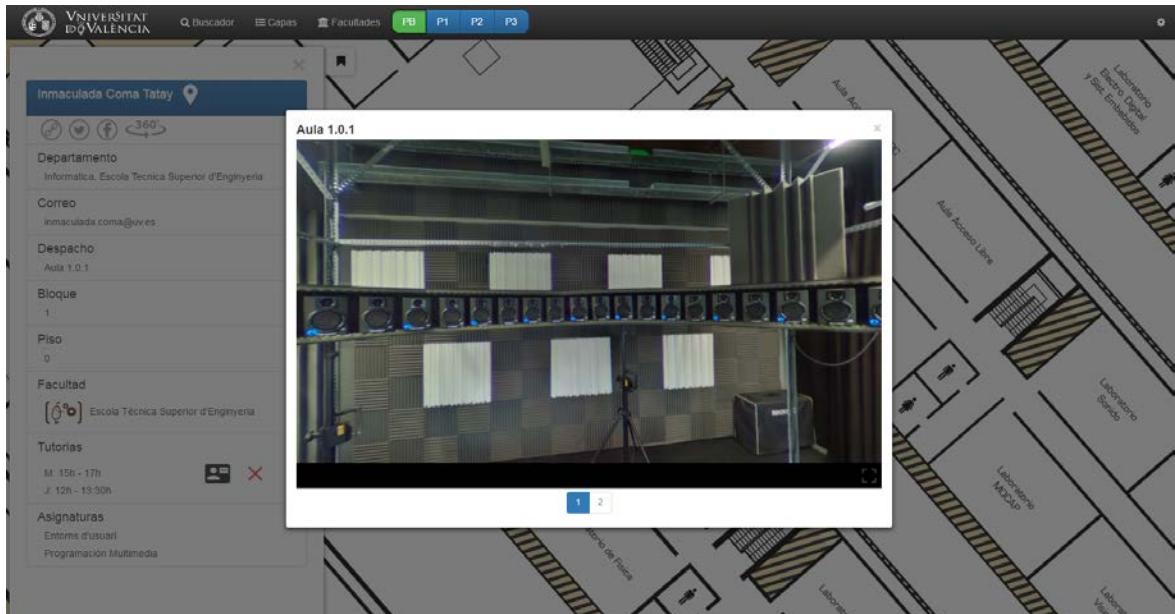


Figura 6.1.11-1: Visor de Panoramas

Si el espacio dispone de más de una imagen asociada, se dispondrán todas ellas en una lista de páginas que nos informará de cuantas hay disponibles y con la posibilidad de pulsarla para proceder a cargarla en el visor de imágenes.

También se dispondrá de un botón dispuesto en la parte derecha inferior del visor que nos permite pasar al modo Pantalla Completa y visualizar la imagen al tamaño que el dispositivo posea.

Pulsando en cualquier zona exterior a la ventana modal, cerrará esta ventana devolviendo la visión al estado anterior al que se encontraba antes de acceder a esta opción.



6.2 Rendimiento del sistema

Es muy importante, aparte de que el sistema funcione correctamente, analizar cómo se comportaría el sistema ante una conexión masiva de usuarios. La aglomeración de usuarios accediendo a la vez a los mismos recursos, hará que el sistema tarde más tiempo en resolver dichos recursos y el usuario tenga que esperar mucho tiempo si el sistema no está bien dimensionado.

6.2.1 Pruebas Usuarios

Para analizar el grado de usabilidad que dispone nuestra aplicación cliente debemos conocer cómo interactúan los usuarios con ella en un entorno controlado donde podamos evaluar la usabilidad haciendo uso de unas tareas propuestas que detallaremos posteriormente.

Se pretende observar a un grupo de usuarios con las tareas típicas para detectar potenciales problemas de usabilidad y dar pistas sobre su corrección.

Es un análisis cuantitativo que se realizará con un grupo de x usuarios y que puede servir de base para evaluar futuras versiones. Su principal objetivo es realizar un informe sobre aspectos a mejorar dirigido a los diseñadores que deban efectuar los cambios apropiados.

6.2.1.1 Selección de Usuarios

Debemos realizar una selección de usuarios que encajen en nuestro perfil de “usuario tipo”. Puesto que nuestra aplicación busca convertirse en una utilidad que ayude al mayor número de personas posible, la selección de participantes seguirá los siguientes criterios de inclusión:

- Personal perteneciente a la Universitat de València o en proceso de pertenencia (pre-universitarios).
- Vinculación, de cualquier índole, con la Universidad de València. (Ponentes, profesores invitados, etc).
- Mayor de 17 años.
- Entender castellano.
- Uso de internet de nivel estándar.
- No tener experiencia previa con el desarrollo SigUV.

Y los siguientes criterios de exclusión:

- Tener una discapacidad que impida el uso de las opciones por defecto de la aplicación web (visión, motricidad).

Pruebas

6.2.1.2 *Tareas a realizar*

Con el fin de evaluar por igual todas las interacciones que un usuario tipo realizará durante el test, debemos elaborar una serie de tareas que reproduzcan de forma natural las funcionalidades que un usuario final emplearía durante el uso de nuestra aplicación.

a. Tarea 1: Buscar Profesor en SigUV

Esta tarea consiste en buscar un profesor/a que imparta clase en el grado en Ingeniería Multimedia de la Universitat de València. Si no se conociera a ninguno se propone el nombre de Inmaculada Coma Tatay como ejemplo.

Los resultados obtenidos son los siguientes:

DIFICULTAD TAREA 1

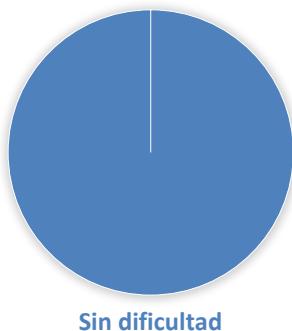


Figura 6.2.1-1: Gráfico dificultad Tarea 1

Podemos observar que la totalidad de los usuarios han logrado sin esfuerzo localizar el profesor y extraer la información correspondiente al mismo. De igual manera, ninguno de los encuestados ha proporcionado ningún dato adicional sobre la realización de la tarea por lo que suponemos que no existe elemento alguno que deba ser replanteado o modificado en su diseño.

b. Tarea 2: Buscar Profesor por Asignatura en SigUV

Esta tarea consiste en buscar un profesor/a que imparta clase en alguna asignatura perteneciente al plan de estudios del grado en Ingeniería Multimedia de la Universitat de València. Si no se conociera ninguna materia se propone el nombre de Entornos de Usuario como ejemplo.

Los resultados obtenidos son los siguientes:



DIFICULTAD TAREA 2

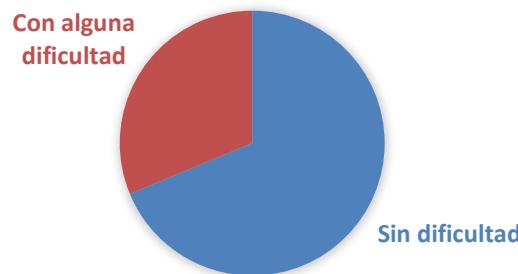


Figura 6.2.1-2: Gráfico dificultad Tarea 2

Obtenidos los resultados observamos que la práctica totalidad de nuestros encuestados ha realizado sin esfuerzo la tarea encomendada. En cuanto a los usuarios que mostraron algun esfuerzo leve a la hora de acometer las instrucciones podemos atribuirlo inicialmente a la falta de conocimiento sobre la aplicación. No obstante, este tipo de problemas se solventan, en gran medida, accediendo a la ayuda presente en la aplicación. Por esta razón consideramos que la realización de la tarea propuesta es adecuada.

c. Tarea 3: Buscar Espacio en SigUV

Esta tarea consiste en buscar un espacio, independiente de su finalidad, que pertenezca al bloque 1 de *l'Escola Tècnica Superior d'Enginyeria* de la Universitat de València. Si no se conociera ningún espacio se propone el despacho de los Técnicos de Laboratorio como ejemplo.

Los resultados obtenidos son los siguientes:

DIFICULTAD TAREA 3

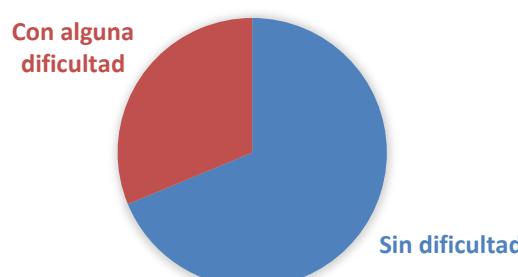


Figura 6.2.1-3: Gráfico dificultad Tarea 3

Obtenidos los resultados observamos que la totalidad de nuestros encuestados ha realizado sin esfuerzo la tarea propuesta. Sin embargo, algunos de ellos, aun cuando han logrado el objetivo sin problemas sugieren que el uso de acentos en una búsqueda puede hacer compleja una consulta sencilla.

Pruebas

El hecho de desabilitar la distinción entre caracteres con y sin acento puede verse como una acción que va en contra de la ortografía presente en nuestro idioma. No obstante, nuestro objetivo no es la de ser gramáticamente correctos sino lograr un buscador accesible y fácil de manejar. Ante esta dicotomía podemos determinar que el objetivo de hacer más sencilla una búsqueda tiene más peso que la corrección del lenguaje que no impide su comprensión en ningún caso.

Por esta razón consideramos importante el cambio de los criterios de las búsquedas actuales a otro donde no se empleen los acentos, mejorando la experiencia del usuario.

d. Tarea 4: Localizar por GPS en SigUV

Esta tarea consiste en localizar la posición del usuario en el mapa.

Los resultados obtenidos son los siguientes:



Figura 6.2.1-4: Gráfico dificultad Tarea 4

Obtenidos los resultados observamos que pocos encuestados lograron realizar la tarea sin esfuerzo alguno. De igual manera, se aprecia un claro predominio de usuario que han acometido este objetivo con relativa dificultad afirmando en sus opiniones que les ha supuesto un tiempo encontrar la funcionalidad que les permitiera alcanzar la meta.

Este hecho es necesario tenerlo en cuenta pues, asumir que un número importante de usuarios les dificulta acceder o localizar una funcionalidad, puede indicarnos que debemos replantear la distribución o disposición de la característica y, en este caso, evaluar la prioridad en su modificación.

Puesto que estamos hablando de un cambio de ubicación dentro de nuestra página y no supone una alteración en la propia funcionalidad en la adopción de la medida, podemos introducir este cambio en la versión actual del cliente.



Servicio Web de Información geoposicionada

e. Tarea 5: Modificar Capas en SigUV

Esta tarea consiste en cambiar la representación del mapa visualizado solicitando que el usuario muestre la visión satélite de Google en combinación con la representación temática y por código del plano de l'*Escola Tècnica Superior d'Enginyeria* de la Universitat de València.

Los resultados obtenidos son los siguientes:

DIFICULTAD TAREA 5



Figura 6.2.1-5: Gráfico dificultad Tarea 5

Obtenidos los resultados observamos que la totalidad de nuestros encuestados ha realizado sin esfuerzo la tarea propuesta siendo capaces de alterar la configuración de la representación en varias acciones o en una sola. De igual manera, ninguno de los encuestados ha proporcionado ningún dato adicional sobre la realización de la tarea por lo que suponemos que no existe elemento alguno que deba ser replanteado o modificado en su diseño.

Pruebas

f. Tarea 6: Ver fotografías panorámicas 360 de un espacio en SigUV

Esta tarea consiste en ver las fotografías panorámicas de 360 grados de un espacio perteneciente al Bloque 1 de *l'Escola Tècnica Superior d'Enginyeria* de la Universitat de València e interactuar con ellas cambiando entre las fotografías de un mismo espacio, activando y desactivando el modo pantalla completa. Si no se conoce un espacio, se propone la Sala de Reunión de la Asociación IEEE, código de espacio 1.0.2A, como ejemplo.

Los resultados obtenidos son los siguientes:

DIFICULTAD TAREA 6

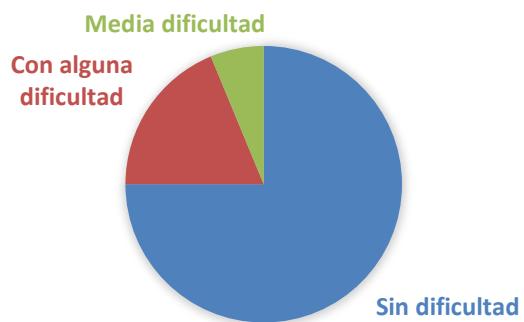


Figura 6.2.1-6: Gráfico dificultad Tarea 6

Obtenidos los resultados observamos que la práctica totalidad de nuestros encuestados ha realizado sin esfuerzo la tarea encomendada. En cuanto a los usuarios que mostraron algún esfuerzo leve a la hora de acometer los objetivos propuestos, nos transmiten la incomodidad de no poder detener el movimiento continuo de la imagen a voluntad para poder observar alguna zona con mayor tranquilidad.

La inclusión de esta capacidad no estaba contemplada en un principio al asumir que un usuario no desearía ver un espacio concreto de una imagen si no toda en su conjunto. Sin embargo, la revisión de este punto y su consecuente desarrollo en la aplicación cliente es un elemento asumible a realizar en la misma versión. De no ser posible en la versión final, se debería plantear en una versión posterior de la aplicación.



Servicio Web de Información geoposicionada

g. Tarea 7: Compartir con redes sociales en SigUV

Esta tarea consiste en buscar el espacio "Laboratorio de Física" o al profesor "José Francisco García Calderaro", uno a escoger, y compartir su búsqueda en las redes sociales principales, Facebook y Twitter.

Los resultados obtenidos son los siguientes:



Figura 6.2.1-7: Gráfico dificultad Tarea 7

Obtenidos los resultados observamos que la totalidad de nuestros encuestados ha realizado sin esfuerzo la tarea propuesta siendo capaces de alterar la configuración de la representación en varias acciones o en una sola. De igual manera, ninguno de los encuestados ha proporcionado ningún dato adicional sobre la realización de la tarea por lo que suponemos que no existe elemento alguno que deba ser replanteado o modificado en su diseño.

Pruebas

h. Tarea 8: Generar enlace directo de búsqueda en SigUV
 Esta tarea consiste en buscar el espacio "Laboratorio de Física" o al profesor "José Francisco García Calderaro", uno a escoger, generar el enlace directo que permita compartir, mediante medios digitales, el recurso y copiarlo al portapapeles.

Los resultados obtenidos son los siguientes:



Figura 6.2.1-8: Gráfico dificultad Tarea 8

Obtenidos los resultados observamos que la totalidad de nuestros encuestados ha realizado sin esfuerzo la tarea propuesta siendo capaces de alterar la configuración de la representación en varias acciones o en una sola. De igual manera, ninguno de los encuestados ha proporcionado ningún dato adicional sobre la realización de la tarea por lo que suponemos que no existe elemento alguno que deba ser replanteado o modificado en su diseño.

6.2.2 Pruebas Cliente

Para analizar el rendimiento en el sistema cliente, nos ayudaremos de las utilidades web [Webpagetest.org²⁴](#) y [YSlow²⁵](#). Esta utilidad realiza diversas pruebas a nuestro cliente sobre los tiempos de descarga de ficheros, representando gráficos de rendimiento y emitiendo informes sobre optimización de nuestro sistema. De esta manera seremos capaces de observar si existen cuellos de botella en nuestra aplicación, donde se encuentran y sugerencia de cómo mejorar la calificación.

Hemos comenzado con la realización de una primera evaluación de nuestro sistema cliente haciendo uso de ambas utilidades para conocer su calificación y saber qué elementos nos perjudican en el proceso de carga o si existe la posibilidad de optimizar más el sistema²⁶.

Una vez terminados sendos test, obtenemos los siguientes informes:

²⁴ Véase [WebPageTest](#) en Bibliografía

²⁵ Véase [YSlow](#) en Bibliografía

²⁶ Véase [Optimizar la codificación y el tamaño de transferencia de los recursos basados en texto](#) en la Bibliografía



Servicio Web de Información geoposicionada

Grade **D** Overall performance score 61 Ruleset applied: YSlow(V2) URL: http://adserverpc.uv.es:8282/SigUV/

ALL(23) FILTER BY: **CONTENT(6)** | **COOKIE(2)** | **CSS(6)** | **IMAGES(2)** | **JAVASCRIPT(4)** | **SERVER(6)**

F Make fewer HTTP requests

F Use a Content Delivery Network (CDN)

A Avoid empty src or href

F Add Expires headers

F Compress components with gzip

A Put CSS at top

F Put JavaScript at bottom

A Avoid CSS expressions

n/a Make JavaScript and CSS external

B Reduce DNS lookups

F Minify JavaScript and CSS

A Avoid URL redirects

A Remove duplicate JavaScript and CSS

A Configure entity tags (ETags)

A Make AJAX cacheable

A Use GET for AJAX requests

A Reduce the number of DOM elements

D Avoid HTTP 404 (Not Found) errors

A Reduce cookie size

F Use cookie-free domains

A Avoid AlphaImageLoader filter

D Do not scale images in HTML

A Make favicon small and cacheable

Grade F on Make fewer HTTP requests

This page has 26 external Javascript scripts. Try combining them into one. This page has 111 external stylesheets. Try combining them into one.

Decreasing the number of components on a page reduces the number of HTTP requests required to render the page, resulting in faster page loads. Some ways to reduce the number of components include: combine files, combine multiple scripts into one script, combine multiple CSS files into one style sheet, and use CSS Sprites and image maps.

[Read More](#)

Copyright © 2016 Yahoo Inc. All rights reserved.

[Twitter](#) [Share](#)

Figura 6.2.2-1: Resultado primer test del sistema cliente con YSlow

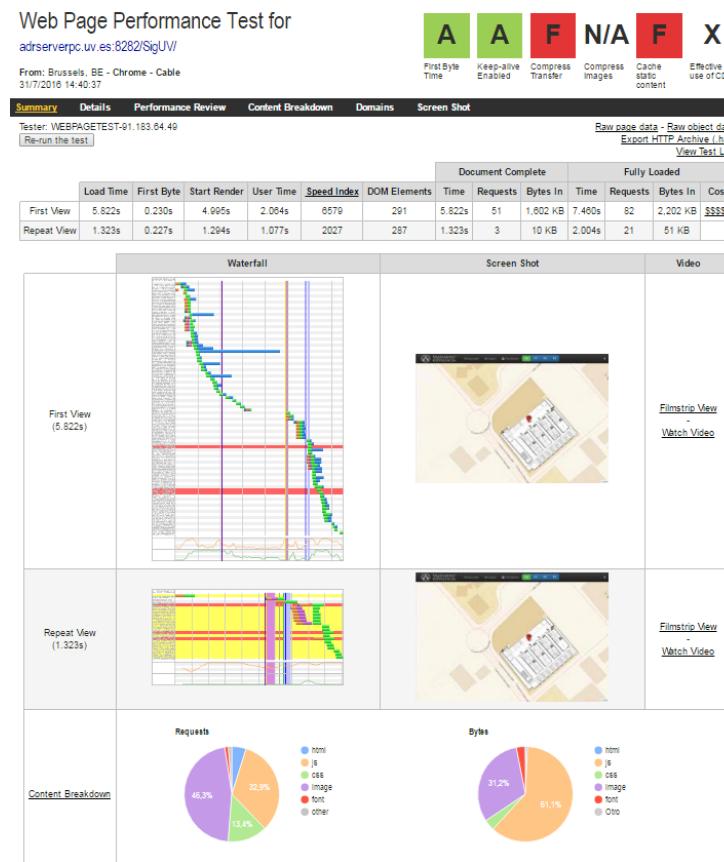


Figura 6.2.2-2: Resultado primer test del sistema cliente con Webpagetest.org

Pruebas

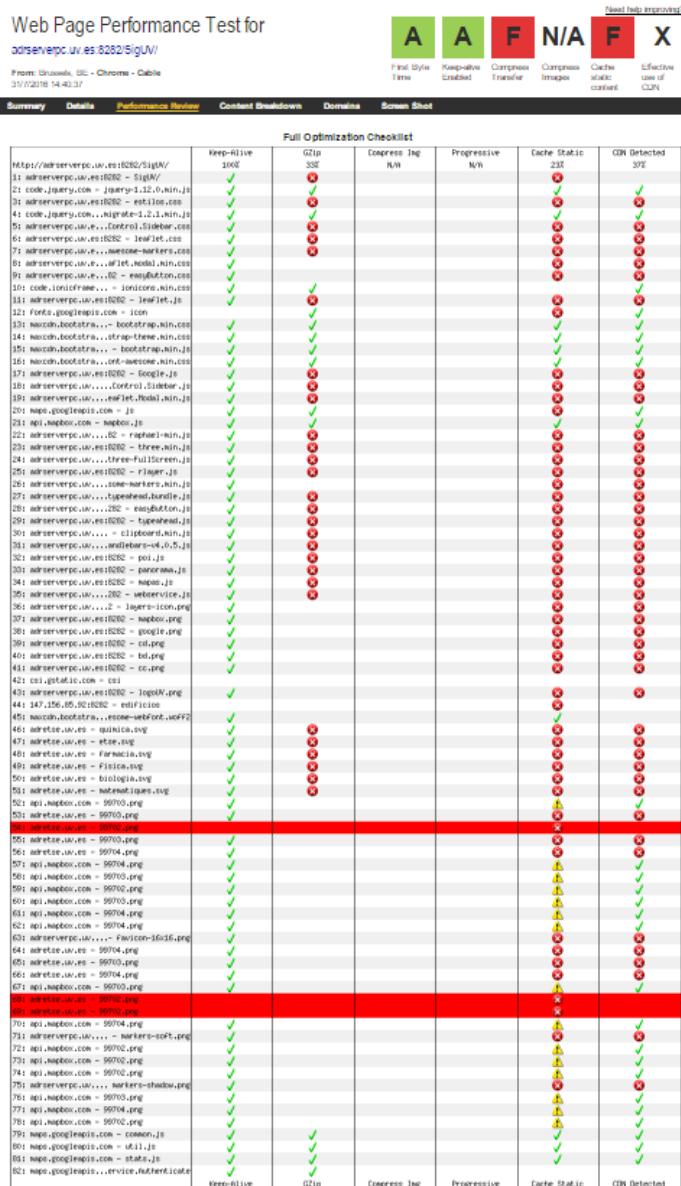


Figura 6.2.2-3: Resultado primer test detalle del sistema cliente con Webpagetest.org

En ellos se pueden apreciar, desde el primer momento, que existen problemas de optimización de nuestro sistema cliente que provocan un gasto superior de ancho de banda y, por lo tanto, un mayor tiempo de espera en la descarga de ficheros. Esto supone, en este momento, un peso de 2.200Kb en 6 segundos en el momento de la carga completa de la página.

Observando los informes de resultados de ambas utilidades, podemos ver que nuestro sistema presenta deficiencias que hacen nuestra aplicación cliente más pesada y lenta en su descarga de lo que podría ser. La situación se puede ver empeorada al introducir el factor de velocidad de conexión del usuario, la cual influye en los procesos de carga de nuestra página siendo inversamente proporcional al tiempo de carga, por tanto, si la velocidad del usuario es baja, los tiempos de espera aumentarán reduciendo la experiencia de la aplicación. Esto no es conveniente puesto que todo desarrollo web intenta ser lo más liviano y rápido posible a fin de que la carga de la página se produzca en el menor tiempo posible de acuerdo a su conexión.



Servicio Web de Información geoposicionada

Entre las deficiencias de nuestra aplicación en lo que a tamaño y tiempos de espera se refiere y en la que la utilidad nos ha dado una baja calificación representada con una ‘F’, se encuentran los siguientes puntos clasificados por orden de prioridad, puesto que habrá puntos que podremos, con poco esfuerzo y dedicación, lograr mejoras notables en la aplicación y otros que, por ser elementos más complejos, necesitarán de una revisión más elaborada por nuestra parte:

Compresión²⁷: la utilidad nos indica que en nuestra aplicación no existe la compresión de contenidos, esto produce una emisión de datos en formato de texto plano que provoca un aumento innecesario de nuestro ancho de banda necesario para la transmisión de datos. Será conveniente emplear sistemas de compresión empleados en las conexiones HTTP como son gzip y/o deflate, cuyo uso en combinación con archivos minificados puede lograr ratios de entre 60% y 88% de reducción de tamaño.

Minificación de archivos CSS y JS: la utilidad nos advierte que los archivos que hemos solicitado para integrar en nuestra aplicación no se encuentran en su forma reducida. Minificar significa eliminar todos los espacios en blanco y reducir a su mínima expresión el texto que contenga el archivo y, por consiguiente, reducir el peso del mismo.

Archivos JavaScript al final: Las peticiones de scripts de JavaScript bloquean las descargas paralelas. Es decir, cuando un script está en proceso de descarga, el navegador no inicia ninguna otra descarga, perjudicando el resto de peticiones y, por lo tanto, aumentando los tiempos de espera. Para ayudar a la carga de la página más rápido, dispondremos estos scripts en la parte baja de nuestro HTML para que sean las últimas peticiones de nuestro proyecto.

Excesivas llamadas HTTP: la utilidad nos avisa que existe una cantidad excesiva de peticiones HTTP que contribuye a una dilatación de los tiempos de carga de la página. La solución pasa por la combinación de nuestros archivos css y js a fin de reducir la cantidad de peticiones al mínimo y así mejorar la velocidad de carga de nuestro cliente.

Uso de CDN (Content Delivery Network): La utilidad nos indica que no se está empleando una red de entrega de contenidos. Una CDN es una red superpuesta de computadoras que contienen copias de datos que, colocados en varios puntos de una red, logramos maximizar el ancho de banda para el acceso a los datos de clientes por la red. Deberemos, por tanto, procurar alojar nuestros ficheros en alojamientos con capacidad CDN para reducir los tiempos de descarga.

Uso de cabeceras expire: La utilidad nos muestra que las peticiones a nuestros archivos carecen de cabecera expire. El acceso por primera vez a una página puede requerir varias peticiones HTTP para cargar todos los componentes. Mediante el uso de cabeceras Expire estos componentes se convierten en cacheables, lo que evita las peticiones HTTP innecesarias en posteriores visitas. Por lo tanto, deberemos introducir el uso de este tipo de cabecera a fin de aliviar la carga en posteriores accesos a nuestro cliente.

Tras analizar los primeros resultados arrojados por ambas utilidades, hemos trazado una serie de actuaciones que nos permite mejorar los resultados acometiendo, en primera

²⁷ Véase Compresión en Bibliografía



Pruebas

instancia aquellos puntos de fácil solución con poco esfuerzo y planificando para futuras revisiones aquellos puntos que requieran de una elaboración más compleja.

Tal y como hemos comentado en los puntos de evaluación en los que nuestra aplicación cliente ha obtenido baja calificación, hemos realizado las siguientes operaciones en orden secuencial de actuación:

1. Compresión: Añadiremos las cabeceras de compresión en gzip y deflate en nuestro servidor Glassfish a fin de que reduzcamos el ancho de banda empleado en la trasmisión de datos.
2. Minificar: Mediante utilización de utilidades de minificación, eliminaremos contenido prescindible de nuestros ficheros JavaScript y CSS como pueden ser los espacios en blanco y con ello reduciremos el espacio de los mismos.
3. Reducir el número de llamadas HTTP: Combinaremos, en la medida de lo posible, los archivos JavaScript y CSS a fin de lograr un menor número de ficheros. De esta manera reduciremos las solicitudes HTTP que debe realizar nuestro sistema.
4. Evitar el bloqueo de descargas: Dispondremos las librerías JavaScript al final del documento para evitar que la paralelización de descargas se bloquee cuando se solicitan ficheros JS.
5. Uso de CDN: Alojaremos nuestros ficheros JavaScript, CSS y similares en servidores que dispongan de red de entrega de contenido para acelerar la descarga de nuestros archivos.
6. Añadir cabeceras expire: Añadiremos las cabeceras expire en nuestro servidor Glassfish a fin de que nuestros archivos sean cacheables y no se descarguen en visitas posteriores.

Tras la realización de estas acciones hemos vuelto a emplear las utilidades de evaluación a fin de comprobar si nuestras medidas han mejorado el estado de nuestra aplicación cliente. El segundo test nos arroja los siguientes informes:

Grade B Overall performance score 90 Ruleset applied: YSlow(V2) URL: http://localhost:8080/SigUV/index.html#
All (23) FILTER BY: Content (6) | Cookies (2) | CSS (6) | Images (2) | JavaScript (4) | Server (16)

B. Make fewer HTTP requests

A Use a Content Delivery Network (CDN)
A Avoid empty src or href
F Add Expires headers
A Compress components with gzip
A Put CSS at top
A Put JavaScript at bottom
A Avoid CSS expressions
N/A Make JavaScript and CSS external
A Reduce DNS lookups
A Minify JavaScript and CSS
A Avoid URL redirects
A Remove duplicate JavaScript and CSS
A Configure entity tags (ETags)
A Make AJAX cacheable
A Use GET for AJAX requests
A Reduce the number of DOM elements
A Avoid HTTP 404 (Not Found) error
A Reduce cookie size
A Use cookie-free domains
A Avoid AlphaImageLoader filter
C Do not scale images in HTML
A Make favicon small and cacheable

Grade B on Make fewer HTTP requests
This page has 6 external Javascript scripts. Try combining them into one.
Decreasing the number of components on a page reduces the number of HTTP requests required to render the page, resulting in faster page loads. Some ways to reduce the number of components include: combine files, combine multiple scripts into one script, combine multiple CSS files into one style sheet, and use CSS Sprites and image maps.
[Read More](#)

Copyright © 2016 Yahoo Inc. All rights reserved.

Figura 6.2.2-4: Resultado segundo test del sistema cliente con YSlow



Servicio Web de Información geoposicionada

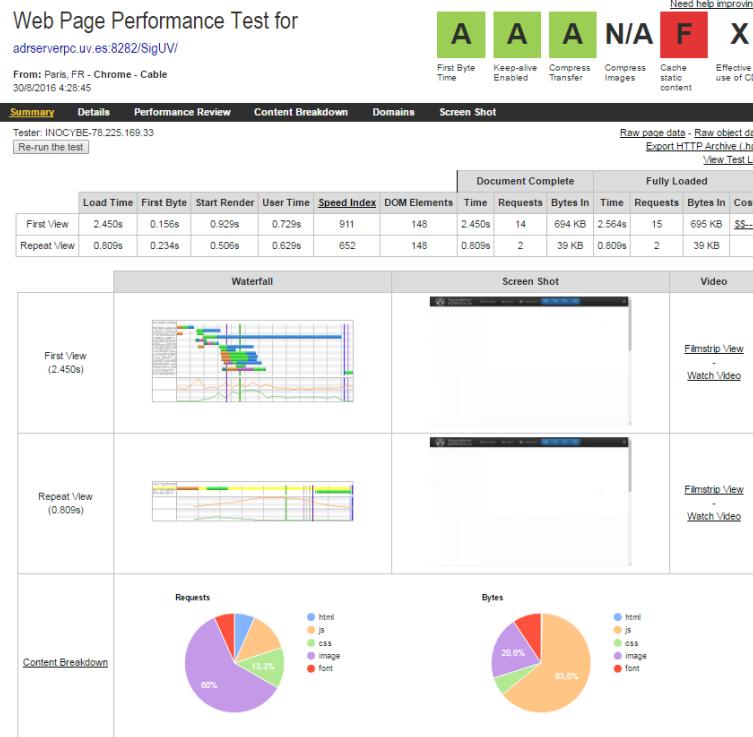


Figura 6.2.2-5: Resultado segundo test del sistema cliente con Webpagetest.org

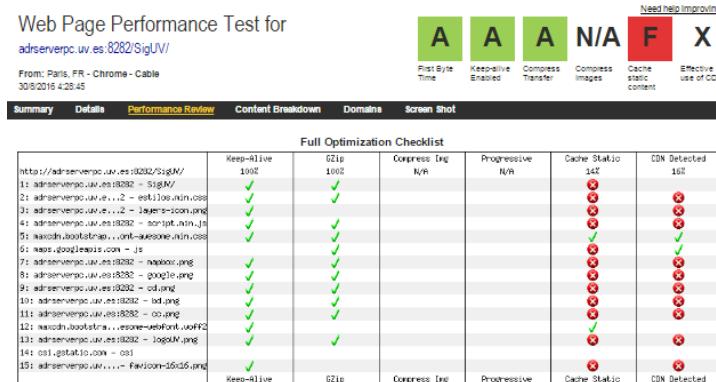


Figura 6.2.2-6: Resultado segundo test detalle del sistema cliente con Webpagetest.org

Como podemos observar se pueden apreciar una evidente mejora de rendimiento en el uso del ancho de banda y los tiempos de descarga de archivos donde la mejora con respecto al primer test es aproximadamente de 60%. Estos datos nos muestran que hemos logrado una optimización de nuestro sistema respecto de nuestra primera versión especialmente positiva al aplicar la mayor parte de las técnicas descritas.

Pruebas

6.2.3 Pruebas Servidor

Para analizar el rendimiento en el sistema, nos ayudaremos del programa Web Stress Tool 8. Este software somete al Servidor a unas condiciones extremas de funcionamiento para medir y comprobar que el sistema es capaz de soportar las peores condiciones de funcionamiento.

En las siguientes gráficas mediremos el tiempo de espera del usuario desde que selecciona un recurso hasta que accede a él y la capacidad que tiene el Servidor cuando tiene conectados 10, 25, 50 y 75 usuarios.

6.2.3.1 *Test 1: Rendimiento con 10 usuarios*

En primer lugar, realizaremos una prueba con 10 usuarios que acceden a recursos de nuestro sistema varias veces durante 1 minuto efectuando peticiones al sistema varias veces con un tiempo aleatorio entre peticiones, pero no mayor a 5 segundos.

Como se puede apreciar en esta primera gráfica, cuando el porcentaje de usuarios es del 90%, el tiempo de espera del usuario desde que realiza la petición hasta que le llega la respuesta sería, en la mayoría de los casos, inferior al segundo y en casos aislados no superan los 2 segundos.

Por estos datos podemos concluir que nuestro sistema para 10 usuarios tiene unos tiempos de servicio más que aceptables y una carga de servidor pequeña.

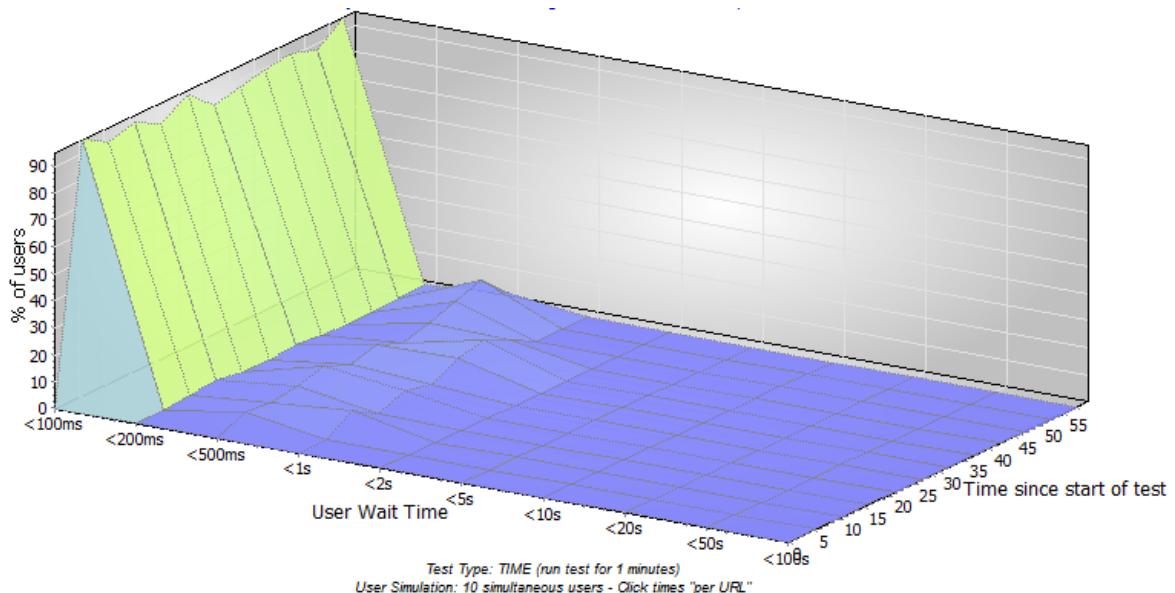


Figura 6.2.3-1: Tiempo de espera del usuario con 10 usuarios en el Sistema

En esta otra gráfica podemos apreciar que el rendimiento de la CPU para 10 usuarios es del 23% un porcentaje aceptable. Por otra parte el uso de la memoria es de 755 MB, lo que supone poco más de un cuarto de la memoria total disponible en el sistema.



Servicio Web de Información geoposicionada

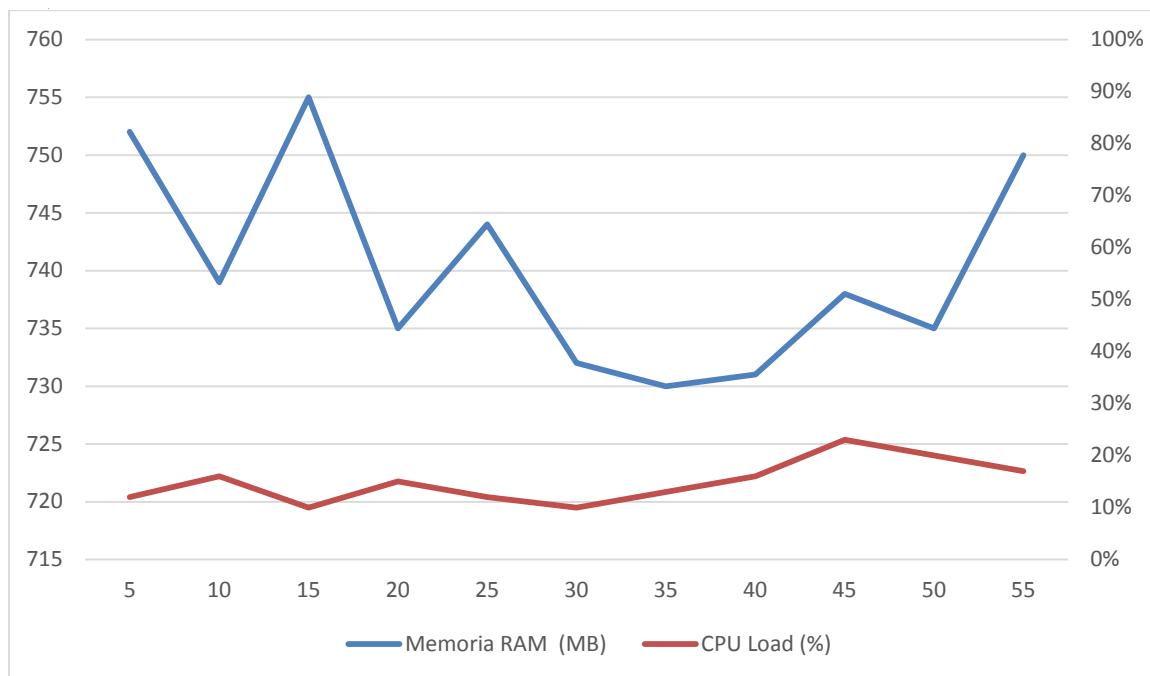


Figura 6.2.3-2: Gráfico Rendimiento Memoria RAM y Carga CPU con 10 usuarios durante 1 minuto

Con estos datos podemos decir que nuestro sistema para 10 usuarios dispone de unos tiempos de servicio aceptables con una carga del servidor pequeña.

6.2.3.2 Test 2: Rendimiento con 25 usuarios

A continuación realizaremos una segunda prueba, en las mismas condiciones anteriores pero variando en esta ocasión el número de participantes a 25 usuarios.

En la siguiente gráfica observamos que igual que en el caso anterior, con un 90% de usuarios utilizando el sistema, el tiempo de espera como máximo sería de 1 segundo en la mayor parte de los casos

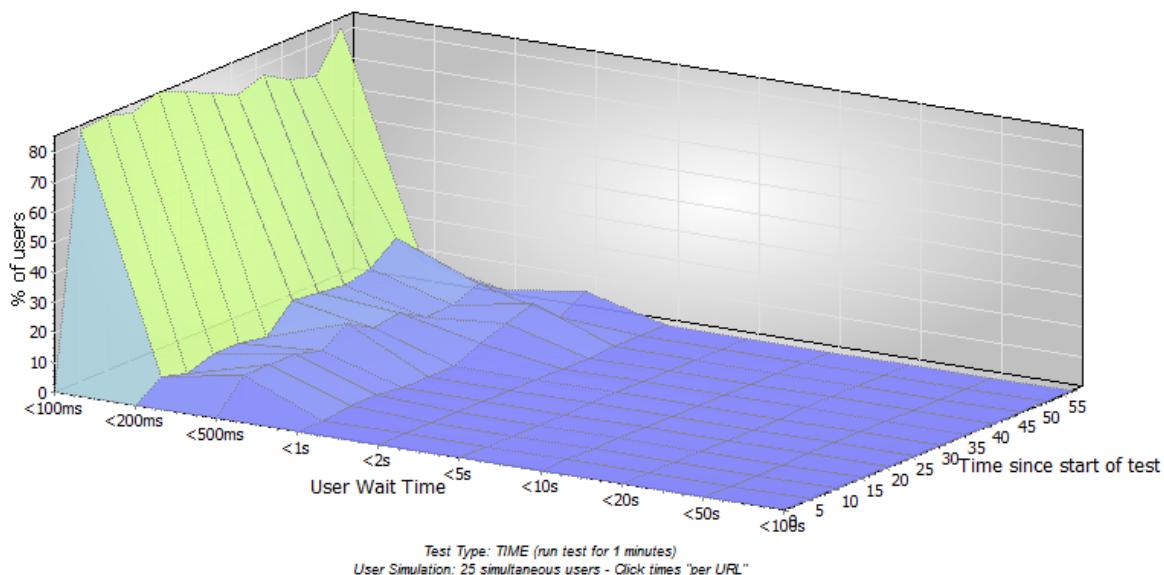


Figura 6.2.3-3: Tiempo de espera del usuario con 25 usuarios en el Sistema

Pruebas

En la siguiente gráfica podemos apreciar que el rendimiento de la CPU para 25 usuarios es de un 43%, un porcentaje todavía aceptable. Por otra parte, el uso de la memoria es de 792 MB, casi un tercio de la memoria disponible en el sistema.



Figura 6.2.3-4: Gráfico Rendimiento Memoria RAM y Carga CPU con 25 usuarios durante 1 minuto

Con estos datos podemos afirmar que nuestro sistema se mantiene dentro de unos valores aceptables de rendimiento y carga de servidor con una afluencia de 25 usuarios.

6.2.3.3 Test 3: Rendimiento con 50 usuarios

A continuación, realizaremos una segunda prueba, en las mismas condiciones anteriores, pero variando en esta ocasión el número de participantes a 50 usuarios.

En la siguiente gráfica observamos que igual que el sistema empieza a encontrarse bastante ocupado arrojando tiempos de espera que en ningún caso sería inferior a los 2 segundos.



Servicio Web de Información geoposicionada

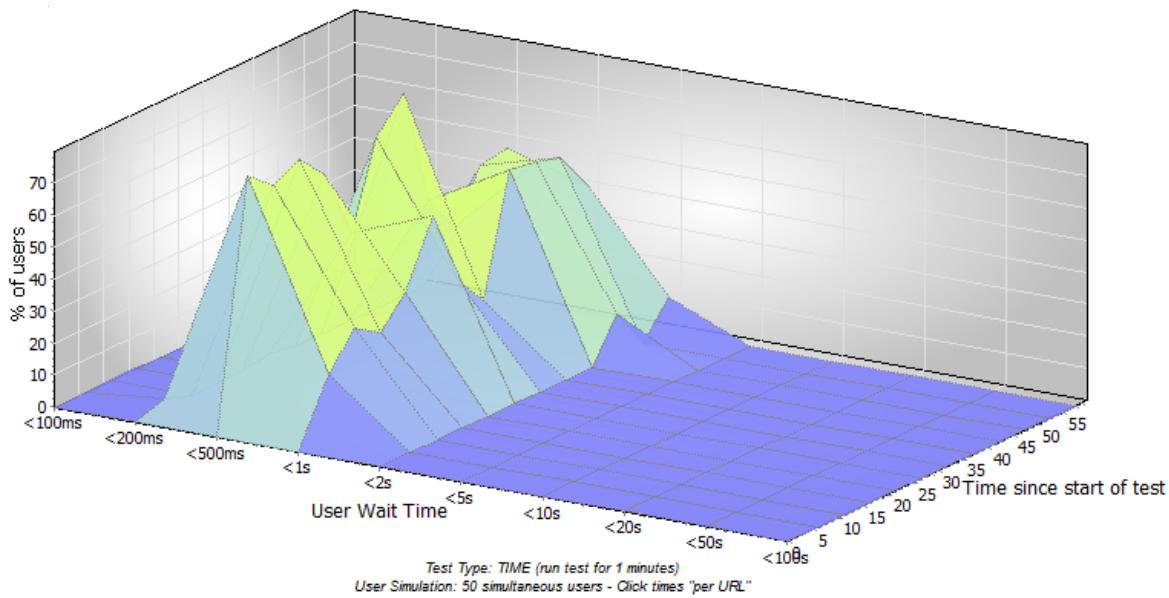


Figura 6.2.3-5: Tiempo de espera del usuario con 50 usuarios en el Sistema

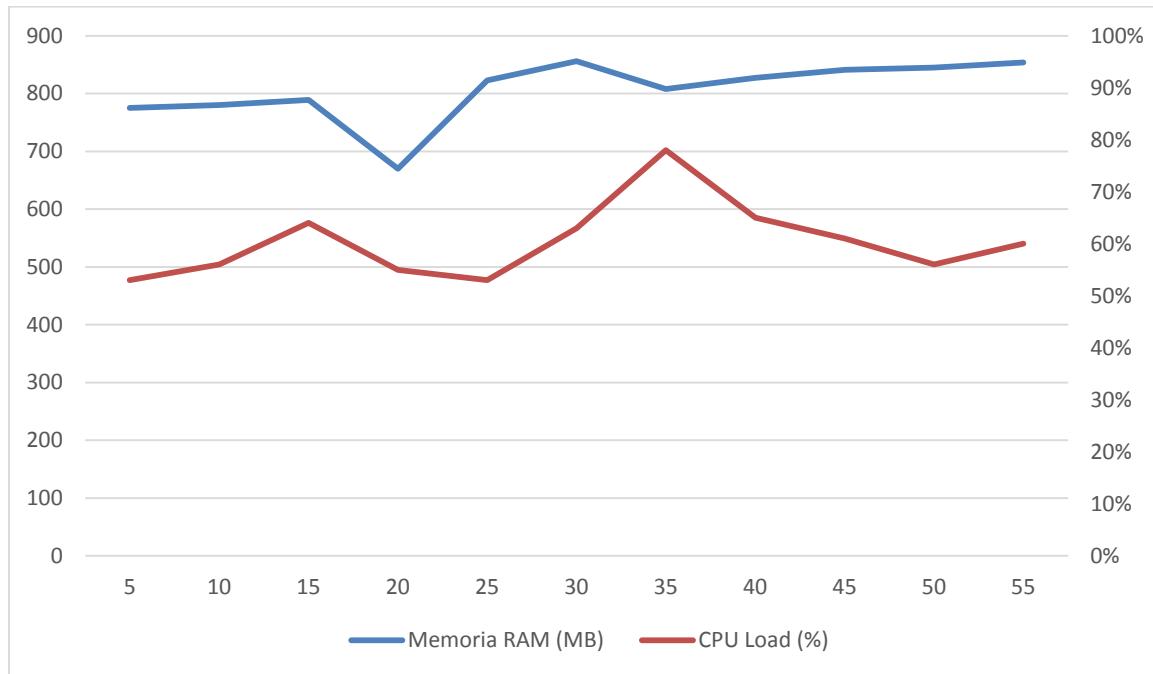


Figura 6.2.3-6: Gráfico Rendimiento Memoria RAM y Carga CPU con 50 usuarios durante 1 minuto

Aquí el porcentaje máximo de uso de CPU se elevaría al 78% y la ocupación de la memoria con respecto al caso anterior sería similar al anterior test.

En este caso se puede decir que aunque el servidor se encuentre bastante ocupado, los tiempos de servicio de los recursos es muy similar a cuando teníamos 25 usuarios. Se puede decir que con 50 usuarios funcionará bien dentro de unos márgenes aceptables de tiempo de respuesta.

Pruebas

6.2.3.4 Test 4: Rendimiento con 75 usuarios

Por último, pondremos intentaremos poner a nuestro servidor en su límite de trabajo realizando un test en las mismas condiciones anteriores, pero aumentando los usuarios a 75.

En este caso el servidor se encontraría desbordado, dando en algunos casos tiempos de espera de los usuarios por encima de los 10 segundos, tiempo excesivo para cargar una página web.

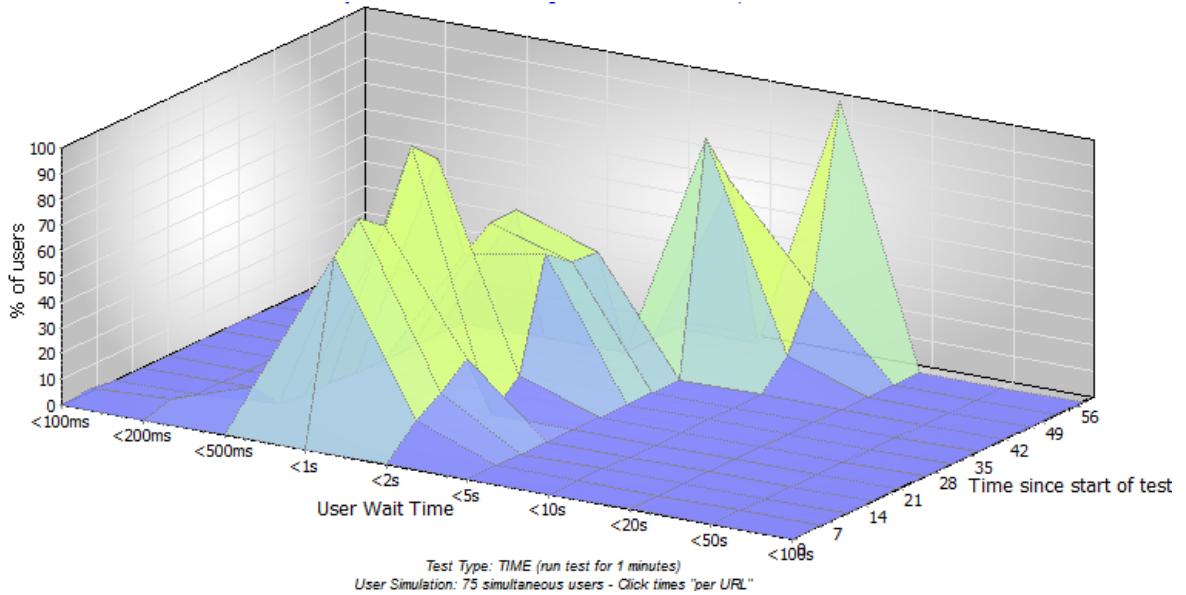


Figura 6.2.3-7: Tiempo de espera del usuario con 75 usuarios en el Sistema

Para finalizar veremos la gráfica de carga del servidor. Aquí se puede apreciar que el servidor se pasa gran parte del tiempo al 100%, lo que supone que el tiempo de espera del usuario aumente considerablemente, tal y como veíamos en la gráfica anterior. Mientras, la ocupación de memoria es similar.



Servicio Web de Información geoposicionada

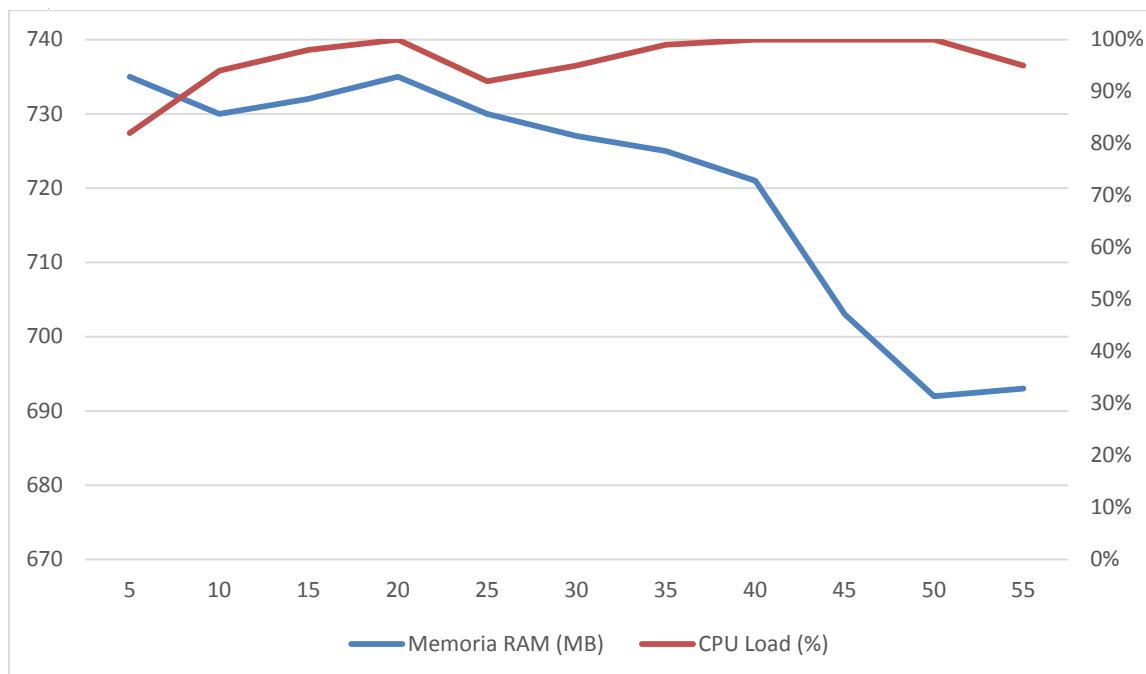


Figura 6.2.3-8: Gráfico Rendimiento Memoria RAM y Carga CPU con 75 usuarios durante 1 minuto

Aquí el sistema se encontraría desbordado, estaría trabajando prácticamente siempre al 100% lo que supondría a los usuarios esperas por encima de 10 segundos en la mayoría de casos. En principio parece poco tiempo, pero durante un periodo prolongado podría llegar a ser molesto de cara al usuario y provocar que los usuarios abandonen el sistema.

Finalizadas las distintas pruebas de servidor, y tras las correspondientes optimizaciones realizadas sobre el ancho de banda que ocupa nuestra aplicación, podemos afirmar que cumplimos de forma apropiada nuestra meta de permitir, al menos, 30 usuarios concurrentes empleando nuestra aplicación. Dado que nuestro servicio no pretende ser un lugar de acceso continuo o que necesite de unas capacidades de concurrencia especialmente elevadas para un sistema de consulta puntual que proporcione información por un periodo corto de tiempo, lograr más del doble de la capacidad de concurrencia inicialmente propuesta en los requisitos es un resultado muy significativo.

Si en un futuro se viera incrementada la demanda al servicio deberíamos realizar un aumento de las capacidades de concurrencia. Para ello habría que optar a disponer de un servidor que se dedique a balancear la carga que realicen las peticiones al servicio. El hecho de poner más servidores no afecta a la aplicación ya que, al tratarse de tecnología J2EE, ésta es replicable en todas las máquinas, siendo su carga balanceada mediante el servidor.



Conclusiones

7 Conclusiones

Tras haber realizado las pruebas en el capítulo anterior ya estamos en disposición de hacer una evaluación final del proyecto y extraer conclusiones.

A lo largo del proyecto hemos realizado un Sistema capaz de buscar información sobre profesores y espacios pertenecientes a l'Escola Tècnica Superior d'Enginyeria y ubicando su posición en el mismo. Para el desarrollo de todo el proyecto tuvimos que evaluar el mercado actual para ver cuáles eran las tecnologías web que mejor se adaptaban a nuestras necesidades de implementación.

A continuación, tuvimos que confeccionar una base de datos bastante laboriosa, ya que la cantidad de registros de datos a introducir fueron muchos, lo que nos obligó a acotar la demostración al departamento de Informática y bloque 1. No sólo fue laboriosa la implementación sino también la recopilación de información, toma de imágenes 360, configuración de la máquina servidor, obtención de planos del edificio y sus correspondientes autorizaciones. A continuación, implementamos el Sistema sometiéndolo a varias revisiones hasta llegar a una que, aparentemente, no contenía errores.

Después, analizamos los resultados y fueron bastante satisfactorios. Fuimos capaces de construir un sistema con tiempos de respuesta breves, siempre y cuando el número de usuarios concurrentes a la aplicación fuese menor a 50 usuarios. También hemos quedado bastante satisfechos con el diseño de la interfaz, el cual, tras un diseño inicial propuesto por nosotros, ha sufrido variaciones en sus revisiones gracias a las pruebas de usuarios que nos proporcionaron información sobre la distribución de elementos a fin de hacerlo más intuitivo.

En cuanto a los problemas surgidos a lo largo de la realización del proyecto, inicialmente fueron el desconocimiento de servicios web basados en REST, que obligó a disponer de un tiempo aproximado de un mes a fin de comprender, asimilar y aplicar esta tecnología en el proyecto. Estos hechos provocaron un ligero reajuste de la planificación que retrasó la fecha de finalización del desarrollo inicialmente estimado.

La principal complicación fue la integración de las diversas librerías en el sistema cliente, principalmente typeahead.js y three.js, puesto que provocaban conflictos entre algunos de sus métodos, por lo que hubo que realizar una configuración exhaustiva de las librerías y en el orden de sus llamadas a fin de no interferir entre sus funciones.

En pasos muy iniciales del proyecto, pensamos en crear mapas de forma estática y mostrarlo al usuario de esta manera. Sin embargo, esto resultaba poco atractivo y poco funcional en el momento de la interacción con el mapa, además de que limitaba la experiencia del usuario, esto nos hizo replantearnos las opciones y escoger una tecnología más actual que permitiera la interacción directa del usuario, además de facilitarnos la modificación de datos representados al tenerlos almacenados en la base de datos de la cual se extraen los parámetros.



Trabajos futuros

8 Trabajos futuros

Realizados todos los desarrollos planificados, corregidos los problemas derivados de los mismos y acabados los trabajos de optimización a consecuencia de las diversas pruebas, propondremos una serie de ampliaciones que puedan dotar a nuestra aplicación de nuevas características o servicios:

Realizar un estudio más detallado de toda la información disponible a fin de poder ofrecer otros servicios asociados a la geolocalización de espacios en la universidad, como pudiera ser la integración de los enlaces directos a espacios a los distintos servicios de agenda o eventos que ofrece la web de la universidad.

Crear roles de autenticación mediante la cesión de permisos en la aplicación. Uno que permita acceder a información oculta no disponible a usuarios públicos estándar y otro con permisos de administrador que ofrezca la posibilidad de crear, actualizar o eliminar recursos de nuestra aplicación en función de las necesidades de cada ejercicio académico.

Desarrollar un gestor web que permita, a un usuario con permisos de administrador, realizar tareas CRUD para modificar los registros de la base de datos de una manera accesible para personal de administración o gestión de recursos.

Incluir servicio de navegación en los mapas de tal manera que pueda indicarnos el camino a tomar para llegar desde nuestra posición actual al destino deseado.

Respecto a los mapas, se podría realizar una nueva capa que permita visualizar en 3D el espacio a visitar, empleando la misma librería three.js, creando mapas en x3dom o similares tecnologías.

Aunque nuestra aplicación es responsive y se adapta la visualización a los dispositivos portátiles tablets y smartphones, se puede integrar el desarrollo con la aplicación móvil que ofrece la propia Universitat de València a fin de tener todos los servicios centralizados en una única aplicación.



Bibliografía

9 Bibliografía

Agafonkin, V. (11 de Julio de 2016). Obtenido de Leaflet: <http://leafletjs.com/reference.html>

Página principal de la librería JavaScript Leafletjs desde donde se puede descargar tanto la propia librería, así como extensiones desarrolladas por terceros, además de contar con una extensa documentación con ejemplos varios.

Bacinger, T. (11 de Julio de 2016). *Survey of the Best Online Mapping Tools for Web Developers: The Roadmap to Roadmaps*. Obtenido de Toptal:
<https://www.toptal.com/web/the-roadmap-to-roadmaps-a-survey-of-the-best-online-mapping-tools>

Artículo comparativo elaborado por Tomislav Bacinger sobre las opciones que se encuentran actualmente disponibles en materia de herramientas de mapas online.

CC.OO. (26 de Julio de 2016). *Tablas salariales para Ingenierías*. Obtenido de CC.OO:
<http://www.ccoo-servicios.es/archivos/tablas2011ingenierias.pdf>

Documento extraído de la fuente CC.OO donde se publican las tablas salariales.

Facebook. (11 de Julio de 2016). Obtenido de Facebook for developers:
<https://developers.facebook.com/docs/>

Una de las redes sociales más populares en la actualidad que cuenta con una sección dedicada a desarrolladores independientes, donde se proporciona una extensa documentación, ejemplos prácticos de desarrollo y herramientas de testeo o sandbox para probar nuestros desarrollos propios antes de publicarlos.

Fernández, A. (11 de Julio de 2016). *Servicios web RESTful con HTTP. Parte I: Introducción y bases teóricas*. Obtenido de Asociación Desarrolladores web de España:
<http://www.adwe.es/general/colaboraciones/servicios-web-restful-con-http-parte-i-introduccion-y-bases-teoricas>

Artículo elaborado por Alberto Fernández sobre las bases de la tecnología REST, incidiendo en sus fundamentos y funcionamiento sobre HTTP.

Grigorik, I. (29 de Julio de 2016). *Optimizar la codificación y el tamaño de transferencia de los recursos basados en texto*. Obtenido de Google Developers:
<https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/optimize-encoding-and-transfer>

Documentación proporcionada por Google sobre la compresión y minificación de archivos, con el objetivo de reducir el peso y el consiguiente ancho de banda requerido en su transmisión.

Gudelli, A. (11 de Julio de 2016). *A simple jQuery Autocomplete Search Tutorial*. Obtenido de Arungudelli: <http://www.arungudelli.com/jquery/simple-jquery-autocomplete-search-tutorial/>

Artículo de Arunkumar Gudelli, donde se explican las capacidades de configuración y personalización de la librería Typeahead.js.



Servicio Web de Información geoposicionada

Gulabani, S. (2013). *Developing RESTful web services with Jersey 2.0*. Birmingham: Packt Publishing.

Libro sobre desarrollo de servicios web RESTful mediante Jersey 2.0, empleado en el aprendizaje, comprensión y posterior desarrollo de esta tecnología en nuestro desarrollo. Publicación muy completa y extensa sobre la tecnología REST que me ayudó en poco tiempo asimilar los conocimientos necesarios para comenzar a desarrollar.

Harding, J. (11 de Julio de 2016). *Bloodhound*. Obtenido de Github:
<https://github.com/twitter/typeahead.js/blob/master/doc/bloodhound.md>

Documentación perteneciente a Typeaheadjs que explica el funcionamiento y configuración del motor de búsqueda Bloodhunt. Empleado para la localización y muestra de los recursos que se ajusten la búsqueda deseada.

Harding, J. (11 de Julio de 2016). *jQuery#typehead*. Obtenido de Github:
https://github.com/twitter/typeahead.js/blob/master/doc/jquery_typeahead.md

Documentación perteneciente a Typeaheadjs que explica la gestión de eventos en conjunción con la librería JavaScript JQuery. Empleado para la personalización del comportamiento de las búsquedas en base a los datos de entrada y salida.

I run my website. (11 de Julio de 2016). *Raphael* . Obtenido de I run my website:
<http://www.irunmywebsite.com/raphael/additionalhelp.php?q=bearbones#pagetop>

Espacio web donde se muestran y explican multitud de ejemplos prácticos en el empleo de la librería Raphaeljs. Empleado para mostrar el área delimitadora de los espacios buscados.

Klokang Technologies. (11 de Julio de 2016). *Create an overlay of Google Maps with Photoshop*. Obtenido de Maptiler: <http://www.maptiler.org/photoshop-google-maps-overlay-tiles/>

Utilidad proporcionada por Maptiler que facilita la tarea de creación de mapas personalizados, añadiéndolos a los ya existentes que proporcionan los distintos proveedores de estos servicios. Empleado para la creación y ajuste a su localización física de los planos ETSE en el mapa del proveedor Google Maps.

Mapbox. (11 de Julio de 2016). *Help*. Obtenido de Mapbox: <https://www.mapbox.com/help/>

Servicio web de cartografía digital que emplea tecnología Open Source. Permite su uso en conjunción con el proveedor de mapas OpenStreetMap y las librerías, como Leafletjs, además de ser capaz de personalizar los mismos mediante hojas de estilo dedicadas a la representación visual de los planos.

Mora, R. C. (11 de Julio de 2016). *Creando servicios restful con netbeans 8*. Obtenido de Adictos Al Trabajo: <https://www.adictosaltrabajo.com/tutoriales/creando-servicios-restful-con-netbeans-8/>

Documentación basada en la creación de servicios RESTful mediante la aplicación de desarrollo Netbeans 8.



Bibliografía

My Codde. (11 de Julio de 2016). *Typeahead.js Autocomplete Suggestion + Bloodhound Remote Data - Tutorial and Demo*. Obtenido de My Codde: <http://mycodde.blogspot.com.es/2014/12/typeaheadjs-autocomplete-suggestion.html>

Documentación perteneciente a Typeaheadjs que explica el funcionamiento y configuración del motor de sugerencias. Empleado para la localización y muestra de sugerencias de recursos que se ajusten a la búsqueda deseada.

Oracle Corporation. (11 de Julio de 2016). *Jersey 2.23.1 User Guide*. Obtenido de Jersey Java: <https://jersey.java.net/documentation/latest/index.html>

Documentación empleada en la asimilación y puesta en práctica de los servicios RESTful empleando la variante Jersey, que nos proporciona facilidades en la creación de los mismos.

Purushothaman, J. (2015). *RESTful Java web services : design scalable and robust RESTful web services with JAX-RS and Jersey extension APIs*. Birmingham: Packt Publishing.

Libro sobre diseño de servicios web RESTful mediante Jersey 2.0, empleado en el aprendizaje, comprensión y posterior desarrollo de esta tecnología en nuestro desarrollo. Publicación muy completa y extensa sobre la tecnología REST que me ayudó en poco tiempo asimilar los conocimientos necesarios para comenzar a desarrollar

RunningCoder. (11 de Julio de 2016). *jQuery Typehead Search: Demo*. Obtenido de RunningCoder: http://www.runningcoder.org/jquerytypeahead/demo/?game_v1%5Bquery%5D=ac

Sitio web que presenta multitud de ejemplos prácticos y extensamente explicados sobre la utilización de la librería JavaScript Typeaheadjs, que ayuda a comprender y asimilar la utilización de distintos apartados del código y sus métodos.

Sánchez, P. H. (11 de Julio de 2016). *Como conectar a Oracle con Java*. Obtenido de Devjoker: <http://www.devjoker.com/contenidos/catss/132/Como-conectar-a-ORACLE-con-Java-.aspx>

Artículo web donde se explica de forma detallada y paso a paso la vinculación de bases de datos Oracle con el entorno de desarrollo Java a fin de poder efectuar las consultas correspondientes.

Starr, J. (11 de Julio de 2016). *How to Write Valid URL Query String Parameters*. Obtenido de Perishable Press: <https://perishablepress.com/how-to-write-valid-url-query-string-parameters/>

Artículo web donde se informa, explica y ejemplifica el uso de QueryStrings a fin de lograr acceder a un recurso concreto mediante un hipervínculo o enlace directo al mismo. Empleado para el acceso a un profesor o espacio de forma directa mediante un enlace dado.

Tatiyants, A. (11 de Julio de 2016). *How to use Json objects with twitter bootstrap typehead*. Obtenido de Tatiyants: <http://tatiyants.com/how-to-use-json-objects-with-twitter-bootstrap-typeahead/>



Servicio Web de Información geoposicionada

Artículo web elaborado por Alex Tatiyants, donde explica la utilización de objetos JSON en connivencia con la librería Typeaheadjs. Empleado en la utilización de información en formato JSON proveniente de nuestra base de datos para ser empleada en el motor de búsqueda Typeaheadjs.

Threejs. (11 de Julio de 2016). Obtenido de three.js: <http://threejs.org/>

Librería JavaScript para crear y mostrar gráficos animados por ordenador en 3D en un navegador Web y puede ser utilizada en conjunción con el elemento canvas de HTML5, SVG ó WebGL. Empleado para mostrar panorámicas 360 grados en los distintos espacios.

Twitter Inc. (11 de Julio de 2016). *Documentation*. Obtenido de Twitter :
<https://dev.twitter.com/overview/documentation>

Una de las redes sociales más populares en la actualidad, que cuenta con una sección dedicada a desarrolladores independientes donde se proporciona una extensa documentación y ejemplos prácticos de desarrollo.

Vogella. (11 de Julio de 2016). *Create a REST client*. Obtenido de Vogella:
<http://www.vogella.com/tutorials/REST/article.html#firstclient>

Documentación que proporciona información sobre la creación de aplicaciones clientes que sean capaces de consumir recursos basados en un servicio RESTful.

WebPagetest. (28 de Julio de 2016). *Test a website's performance*. Obtenido de WebPagetest:
<https://www.webpagetest.org/>

Aplicación web capaz de realizar una completa revisión técnica de nuestro sitio web y, posteriormente, elaborarnos un informe detallado de cada aspecto presente en el mismo. Algunos de los datos más significativos son los tiempos de carga por primera vez y sucesivas, ficheros minificados y/o comprimidos, número de peticiones efectuadas en el sitio, entre otras muchas.

Yslow. (29 de Julio de 2016). *Optimización web con Yslow*. Obtenido de Yslow:
<http://yslow.es/>

Extensión de navegador web capaz de realizar un informe detallado de cada aspecto relacionado con el sitio web, otorgando una puntuación a cada uno y entregando una valoración general de nuestra aplicación. Algunos de los datos más significativos que nos puede proporcionar son la cantidad de peticiones efectuadas en la carga, los archivos comprimidos y/o minificados, archivos bajo CDN, entre otras características.