



**Servicio Web de información geoposicionada de l’Escola Tècnica Superior d’Enginyeria**

# Trabajo Final de Grado

Escola Tècnica Superior d’Enginyeria

Autores:

Bustos, Adrián

Valencia, Abril 2016

Índice de Contenidos

[Trabajo Final de Grado i](#_Toc453021982)

[1.1 Introducción 1](#_Toc453021983)

[1.2 Estado del Arte 2](#_Toc453021984)

[1.2.1 Lenguajes de programación orientados a objetos 2](#_Toc453021985)

[1.2.2 Lenguajes para la generación de páginas dinámicas 6](#_Toc453021986)

[1.2.3 Servidores de aplicaciones Java 7](#_Toc453021987)

[1.2.4 Base de Datos 11](#_Toc453021988)

[1.2.5 Comunicación de la base de datos con la aplicación 17](#_Toc453021989)

[1.2.6 Gráficos vectoriales 18](#_Toc453021990)

[1.3 Planificación y Metodología 21](#_Toc453021991)

[1.3.1 Planificación y gestión del proyecto 21](#_Toc453021992)

[1.3.2 Diagrama de hitos 21](#_Toc453021993)

[1.3.3 Diagrama de Gantt 21](#_Toc453021994)

[1.3.4 Estimación de costes y estudio de viabilidad 21](#_Toc453021995)

[1.3.5 Estimación de costes de personal 23](#_Toc453021996)

[1.3.6 Estimación de recursos Software/Hardware 23](#_Toc453021997)

[1.3.7 Estudio de viabilidad económica y legal 23](#_Toc453021998)

[1.4 Análisis del Sistema 23](#_Toc453021999)

[1.4.1 Especificación de requisitos 24](#_Toc453022000)

[1.4.2 Requisitos del usuario 24](#_Toc453022001)

[1.4.3 Requisitos del sistema 25](#_Toc453022002)

[1.4.4 Análisis del modelo de datos 25](#_Toc453022003)

[1.4.5 Datos 26](#_Toc453022004)

[1.4.6 Análisis del modelo de procesos 29](#_Toc453022005)

[1.5 Diseño del Sistema 30](#_Toc453022006)

[1.5.1 Diseño de datos 31](#_Toc453022007)

[1.6 Implementación del sistema 36](#_Toc453022008)

[1.6.1 Implementación de la base de datos 36](#_Toc453022009)

[1.6.2 Implementación de las Entidades en el servidor 38](#_Toc453022010)

[1.6.3 Organización del sitio web 39](#_Toc453022011)

[1.7 Resultados 40](#_Toc453022012)

[1.7.1 Correcto funcionamiento del sistema 40](#_Toc453022013)

## 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.

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.

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.

## 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.

### 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: facilitan 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.

#### HTML

HTML, sigla en inglés de HyperText Markup Language (lenguaje de marcas de hipertexto), hace referencia al [lenguaje de marcado](https://es.wikipedia.org/wiki/Lenguaje_de_marcado) para la elaboración de [páginas web](https://es.wikipedia.org/wiki/P%C3%A1gina_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](https://es.wikipedia.org/wiki/World_Wide_Web_Consortium)([W3C](https://es.wikipedia.org/wiki/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](https://es.wikipedia.org/wiki/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.

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).

#### 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](https://es.wikipedia.org/wiki/HTML) o [XML](https://es.wikipedia.org/wiki/Extensible_Markup_Language) (y por extensión en [XHTML](https://es.wikipedia.org/wiki/XHTML)). El [World Wide Web Consortium](https://es.wikipedia.org/wiki/World_Wide_Web_Consortium) (W3C) es el encargado de formular la especificación de las [hojas de estilo](https://es.wikipedia.org/wiki/Hoja_de_estilo) que servirán de estándar para los [agentes de usuario](https://es.wikipedia.org/wiki/Agente_de_usuario) o [navegadores](https://es.wikipedia.org/wiki/Navegador_web).

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”.

### 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:

#### 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 está 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 forma que se puede utilizar en casi cualquier sistema operativo, se integra con 8 y tiene accesos a 20 bases de datos con funciones específicas que hace que sea bastante eficiente. Además permite cualquier acceso a base de datos por medio de ODBC.

Otra de las ventajas de 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.

(Imagen arquitectura PHP)

#### 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, fue lanzada al mercado mediante una estrategia de mercado denominada .NET.

Se desarrollado para resolver las limitantes que brindaba tu antecesor ASP. Para el desarrollo de ASP.NET se puede utilizar C#, VB.NET o J#. Los archivos cuentan con la extensión (aspx). Para su 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.

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

(imagen Peticion ASP)

#### 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 contendido de los ficheros.

Es un lenguaje libre de uso, eso quiere decir que es gratuito. Antes estaba muy asociado a la plataforma Uníx, 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](http://www.desarrolloweb.com/articulos/25.php?manual=15) o [ASP](http://www.desarrolloweb.com/articulos/393.php?manual=15). 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 Uníx), pero está enfocado a ser más práctico y fácil que estos últimos. Es por ello que un programador que haya trabajado con el lenguaje C y los otros 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.

(imagen PERL)

#### 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.).

#### 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 que a éste. 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 fines 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, mismo.

La máquina virtual de Java o JRE, que permite la portabilidad en ejecución.

El API Java, un 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. La API Java 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 se 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 uso de la JVM clásica, inclusión de un pequeño y rápido recolector de basura y otras diferencias que ya iremos viendo más adelante.

#### 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. Por ello Java se diseñó siendo similar a C++ para facilitar su aprendizaje y rápida asimilación de la sintaxis.

**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.

### 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 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:

#### 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](https://es.wikipedia.org/wiki/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](https://es.wikipedia.org/wiki/HTTP), [SMTP](https://es.wikipedia.org/wiki/SMTP), [TCP](https://es.wikipedia.org/wiki/Transmission_Control_Protocol) o [JMS](https://es.wikipedia.org/wiki/JMS)).
* Independencia (SOAP permite cualquier modelo de programación).

#### REST

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

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

* Un protocolo [sin estado](https://es.wikipedia.org/wiki/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](https://es.wikipedia.org/wiki/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](https://es.wikipedia.org/wiki/Uniform_Resource_Identifier).
* El uso de hipermedios, 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](https://es.wikipedia.org/wiki/HTML) o [XML](https://es.wikipedia.org/wiki/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.

#### 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.

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.

#### 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.

1

REST focuses on resource-based (or data-based) operations and inherits its operations (GET, PUT, POST, DELETE) from HTTP. This makes it easy for both developers and web-browsers to consume it, which is beneficial for public APIs where you don’t have control over what’s going on with the consumer. Simplicity is one of the strongest reasons that major companies like Amazon and Google are moving their APIs from SOAP to REST.

2

APIs used by apps that require a lot of back-and-forth messaging should always use REST. For example, mobile applications. If a user attempts to upload something to a mobile app (say, an image to Instagram) and loses reception, REST allows the process to be retried without major interruption, once the user regains cell service.

However, with SOAP, the same type of service would require more initialization and state code. Because REST is stateless, the client context is not stored on the server between requests, giving REST services the ability to be retried independently of one another.

3

REST allows easy, quick calls to a URL for fast return responses. The difference between SOAP and REST, in this case, is complexity—-SOAP services require maintaining an open stateful connection with a complex client. REST, in contrast, enables requests that are completely independent of each other. The result is that testing with REST is much simpler.

Helpfully, REST services are now well-supported by tooling. The available tools and browser extensions make testing REST services continually easier and faster.

4

[Stormpath](https://stormpath.com/) is an REST+JSON API-based authentication and user management system for your  
web and mobile services and APIs. We <3 REST+JSON.

If you want to learn more about how to build, design, and secure REST+JSON APIs, here are some developer tutorials and educational blogposts on REST+JSON API Development:

[Beautiful REST+JSON API Design](https://www.youtube.com/watch?v=hdSrT4yjS1g) – this 90-minute best practices presentation for developers dives deep into REST+JSON API Design. Three-time winner of the JavaOne RockStar Award, and viewed over 250,000 times.

[Handy Slides for Beautiful REST+JSON API Design](http://www.slideshare.net/stormpath/rest-jsonapis) – no need to take notes!

[Secure Your REST API The Right Way](https://stormpath.com/blog/secure-your-rest-api-right-way/) – learn the ins and outs of API Security protocols from OAuth to HTTP Basic Authentication.

[Use Stormpath to Manage your API Authentication](http://docs.stormpath.com/guides/api-key-management/) – instead of building out API Access Control and Authentication yourself, Stormpath can save your development team a lot of time. Our API Authentication service handles OAuth 2.0 Access Tokens, Bearer Tokens, Authentication to your API, Access Control, Authorization and Token and Key Management and Revocation.

### 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 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 formularios no están rellenos. 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:

#### 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 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.

#### VBScript

El lenguaje de scripts Visual Basic (VBScript) es un lenguaje propietario 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.

#### 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.

### 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 rede 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

#### 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 concentramos 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.

#### 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 función 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.

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.

#### 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 deja Red Hat.

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

* Implementa 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. Hibernate 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-bussines. 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.

#### 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 es llamada 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.

#### 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.

### 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, distribuidas, 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.

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.

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:

#### 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.

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 Sistema Operativos de Microsoft.

#### 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 y 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.

#### 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 relaciones, 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. MySQL usa el 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.

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.

#### 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.

### 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 OJDBC.

Oracle Java Database Connectivity, 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.

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

### 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(x1, y1), su punto final (x2, y2), 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 pude 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 desee. 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

#### 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 mejor 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.

#### 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.

#### 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.

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

### 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
* 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)

## Planificación y Metodología

### 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.

Para estimar los plazos a cumplir en la realización del proyecto utilizaremos un diagrama de hitos y un diagrama de Gantt, ya que de esta manera podremos visualizar la planificación.

### 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 hitos)

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.

### 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.

(Diagrama de Gantt)

### 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.

Estimación en función de las líneas de código

La métrica del software es un factor realmente importante en el análisis de un proyecto. Las métricas orientadas al tamaño proporcionan medidas directas del software y del proceso por el cual se desarrolla. Se basan en la medición del número de Líneas De Código, o LDC, que contiene el desarrollo, entendiendo por línea de código una sentencia del lenguaje de programación sin incluir comentarios ni líneas en blanco.

Una forma de clasificarlos es atendiendo al número de líneas de código:

(Figura del proyecto según líneas)

Nuestro proyecto ha generado más de X líneas de código, con lo que nos enfrentamos a un proyecto software con una clasificación de complejidad X.

#### El método COCOMO

Una metodología que se encarga de medir proyectos software es COCOMO. La metodología COCOMO (Constructive Cost Model) se debe a Barry Boehm, y está orientada a líneas de código.

Hay una jerarquía de modelos COCOMO: básico, intermedio y avanzado, la cual se aplica a tres tipos diferentes de software:

Orgánico: proyectos relativamente sencillos, menores de 50.000 líneas de código. Se tiene experiencia en proyectos similares y se encuentra en un entorno estable.

Semiacoplado: proyectos intermedios en complejidad y tamaño. La experiencia en este tipo de proyectos es variable, y las restricciones intermedias.

Empotrado: proyectos bastante complejos, en los que apenas se tiene experiencia y en un entorno de gran innovación técnica. Se trabaja con unos requisitos muy restrictivos y de gran volatilidad.

Dado que sólo se va a emplear una variable para la estimación, la línea de código, se empleará COCOMO básico, ya que es un modelo univariable estático, con lo que se obtiene una valoración objetiva del esfuerzo realizado. Este proyecto será considerado como software orgánico, ya que posee menos de 50.000 líneas de código.

La ecuación del esfuerzo de COCOMO básico tiene la siguiente expresión:

Donde KLDC es el número de líneas de código, distribuidas en millares, para el proyecto.

La ecuación del tiempo de desarrollo es:

Por su parte los coeficientes a, b, c y d se obtienen empíricamente del estudio de una serie de proyectos, y sus valores son:

(Diagrama COCOMO)

….

### 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.

### 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.

(Estimación de costes)

### 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.

#### 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.

#### 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.

(Presupuesto)

## 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.

### Especificación de requisitos

Nuestra Aplicación Software consiste en un sistema para la búsqueda y presentación de información de personal docente y espacios en un edificio. En nuestro caso hemos optado por realizarlo sobre l’Escola Tècnica Superior d’Enginyeria de la Universitat de València.

Se intenta crea una aplicación que nos devuelva la mayor cantidad posible de información sobre el recurso que buscamos ya sea personal docente, asignaturas o espacios dentro del edificio, nos muestre también un mapa interactivo en el que podamos localizar cualquiera de estos espacios y, si se dispone de imágenes esféricas de 360 grados, se muestren.

Para realizar todo esto nos apoyaremos en una base de datos creada exclusivamente para este desarrollo, gestionado por Oracle Xpress, un lenguaje potente como lo es Java junto a uno de los mejores servidores de aplicaciones como es GlassFish y el uso de librerías JavaScript para mostrar los mapas y las fotografías esféricas.

### 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:

#### 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 más 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.

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.

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:

(Diagrama de casos de uso) (Visual Paradime)

#### 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.

(Caso de uso)

#### Listado de edificios

Los usuarios tendrán la posibilidad mediante un interfaz Web de seleccionar y mostrar el emplazamiento de cualquier edificio de un listado de los distintos edificios que conforman la Universitat de València.

(Caso de uso)

#### 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.

(Caso de uso)

#### 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 no

(Caso de uso)

#### Interacción Mapa

Los usuarios podrán obtener información básica sobre un espacio concreto catalogado al presionar sobre dicho lugar.

Se mostrara un texto emergente sobre el punto presionado con la información básica del espacio como puede ser la denominación, código, si dispone de personas adscritas al espacio y si se disponen de imágenes esféricas del lugar.

Si la consulta arrojara la existencia de personas adscritas al espacio, este será un botón que realice la consulta correspondiente que muestre mayor información sobre la persona.

Si la consulta arrojara la existencia de fotografías esféricas, se mostrara un botón que muestre las fotografías en una ventana modal.

(Caso de uso)

### Requisitos del sistema

### Análisis del 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 pues un diagrama del modelo conceptual, que muestra 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.

### Datos

#### 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.

**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.

**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.

#### Objetos de datos del Sistema Servidor

En la parte del servidor crearemos los siguientes objetos:

**Objeto Coordenadas**

Tendrá los siguientes componentes:

* IdCoordenada, 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, IdCoordenada.
* 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, IdCoordenada, 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.

**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.

#### Objetos de datos del Sistema Cliente

En la parte del cliente crearemos los siguientes Objetos:

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 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 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

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

El objeto buscador/\*\*/\*/\*/

### Análisis del 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.

#### 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.

(Imagen visual paradigm)

#### 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.

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.

(Imagen Caso de uso DFD)

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.

(Imagen Caso de uso DFD)

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

(Imagen Caso de uso DFD)

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:

(Imagen Caso de uso DFD)

## 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.

### 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.

#### Diseño de datos en la Base de datos

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

Tabla Profesores

(tabla)

Contiene toda la información referente a los profesores:

Consta de los siguientes atributos:

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

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

**Apellidos**: varchar de 40 caracteres como máximo con el apellido del profesor.

**Correo**: varchar de 40 caracteres como máximo con el apellido del profesor.

**Ficha**: varchar de 40 caracteres como máximo con el apellido del profesor.

**Visibilidad**: varchar 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**: varchar de 40 caracteres como máximo con el departamento al que pertenece el profesor.

Tabla Asignaturas

(Tabla)

Dispone los siguientes atributos:

**IdAsignatura**: clave primaria [PK] formada por un varchar2 de 20 caracteres como máximo con el código localizador de asignatura.

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

**Nombre**\_**ES**: varchar de 40 caracteres como máximo con el nombre de la asignatura en castellano.

**Nombre\_EN**: varchar de 40 caracteres como máximo con el nombre de la asignatura en inglés.

Tabla ProfesorAsignatura

(tabla)

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

**IdAsignatura**: clave primaria [PK] formada por un varchar2 de 20 caracteres como máximo con el código localizador de asignatura.

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

Tabla Espacio

(tabla)

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

**Nombre**: varchar2 de 20 caracteres como máximo con el nombre del espacio.

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

**Tipo**: varcha2r 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.

**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.

Tabla Edificio

(tabla)

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

**Nombre**: varchar2 de 20 caracteres como máximo con el nombre del edificio.

**Dirección**: varchar2 de 90 caracteres como máximo con la dirección del edificio.

**Telefono**: varcha2r de 20 caracteres como máximo con el teléfono del edificio.

**Enlace**: varchar2 de 20 caracteres como máximo con el enlace web del edificio.

**Chano**: varchar2 de 20 caracteres como máximo con el enlace al motivo “chano” correspondiente al edificio.

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

Tabla Panoramas

(tabla)

**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 20 caracteres como máximo con el id único de panorama.

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

#### Diseño de datos en el sistema

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 será la siguiente:

/\*71-77\*/

#### 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 /\*78\*/

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

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.

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.

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.

Dispondremos de la opción Facultades donde aparecerá un listado de todas las facultades presentes que nos permitirán 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.

/\*78\*/

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.

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 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 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.

## 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.

### 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"

( "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"

( "IDESPACIO" 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 ("IDESPACIO")

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"

("IDESPACIO" VARCHAR2 (20 BYTE) NOT NULL ENABLE,

"IDPANORAMA" VARCHAR2 (2 BYTE) NOT NULL ENABLE,

"PANORAMA" VARCHAR2 (20 BYTE),

CONSTRAINT "PANORAMAS\_PK" PRIMARY KEY ("IDESPACIO", "IDPANORAMA")

TABLESPACE "SYSTEM” ENABLE,

CONSTRAINT "PANORAMAS\_IDESPACIO\_FK" FOREIGN KEY ("IDESPACIO")

REFERENCES "ABUSPAZ"."ESPACIOS" ("IDESPACIO") ENABLE

)

Tabla Profesores

CREATE TABLE "PROFESORES"

( "IDPROFESOR" VARCHAR2 (6 BYTE) NOT NULL ENABLE,

"NOMBRE" VARCHAR2 (40 BYTE),

"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\_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 …. 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 “” into table profesores.

Load data infile “” into table asignaturas.

Load data infile “” into table profesorasignatura.

Load data infile “” into table edificios.

Load data infile “” into table coordenadas.

Load data infile “” into table panoramas.

Load data infile “” into table espacios.

### Implementación de las Entidades en el servidor

El Sistema que hemos desarrollado está compuesto por una serie de clases escritas en java que forman la estructura interna de la aplicación. A continuación, haremos un repaso general de cada una de las clases que componen el sistema.

Implementación Entidad Profesores

/\*82\*/

Implementación Entidad Asignaturas

/\*82\*/

Implementación Entidad ProfesorAsignatura

/\*82\*/

Implementación Entidad Panorama

Implementación Entidad Edificio

Implementación Entidad Espacio

Implementación Entidad Coordenadas

Implementación de los Servicios en el Servidor

Implementación Servicio Profesores

Implementación Servicio Asignaturas

Implementación Servicio ProfesorAsignatura

Implementación Servicio Panorama

Implementación Servicio Edificio

Implementación Servicio Espacio

Implementación Servicio Coordenadas

Implementación Servicio CORS

Implementación Servicio Abstract

/\*82\*/

### 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.

#### 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 que el usuario necesite consultar.

#### Organización de directorios

En la parte del servidor GlassFish distribuiremos la organización de directorios de la siguiente manera:

/siguv/

/siguv/css

/siguv/js

/siguv/web

/\*\*/

En la parte del Servidor la distribución será:

/SiguvServer/Entidad

/SiguvServer/Servicios

/\*\*/

## Resultados

Una vez terminada la fase de implementación debemos realizar una serie de 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.

### 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.

Página principal

Cuando el usuario accede a nuestra web, 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.

(Imagen página inicial)

A continuación, veremos qué ocurre al acceder a las distintas opciones.

Localizar 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 permitirán buscar por Profesor, Asignatura, Espacio o por todo lo anterior en una búsqueda conjunta.

En cualquiera de los apartados, el sistema proporciona sugerencias de posibles recursos que coincidan con el texto introducido por el usuario.

Una vez confirmada la búsqueda deseada y siendo esta válida, pasaremos a la situación de mostrar resultados que explicaremos más adelante.

En el caso de las asignaturas, en lugar de pasar directamente a mostrar resultados de búsqueda, se nos presentará una lista de profesores que impartan la materia independientemente de su situación de titular o de laboratorio a fin de poder escoger el profesor que estamos buscando.

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.

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