UNIVERSIDAD DE EXTREMADURA

CENTRO UNIVERSITARIO DE MÉRIDA

GRADO EN INGENIERÍA INFORMÁTICA EN TECNOLOGÍAS DE LA INFORMACIÓN

TRABAJO FIN DE GRADO

METODOLOGÍA PARA EL DESARROLLO DE SIMULADORES DE DISEÑO DE INTERIORES

Autor: Daniel Flores Martín

Mérida, Febrero de 2017

UNIVERSIDAD DE EXTREMADURA

CENTRO UNIVERSITARIO DE MÉRIDA

GRADO EN INGENIERÍA INFORMÁTICA EN TECNOLOGÍAS DE LA INFORMACIÓN

TRABAJO FIN DE GRADO

"METODOLOGÍA PARA EL DESARROLLO DE SIMULADORES DE DISEÑO DE INTERIORES"

AUTOR: **D. Daniel Flores Martín**

Fdo.:

DIRECTOR: **Dr. Francisco Chávez de la O**

Fdo.:

CO-DIRECTOR: **Dr. Rafael Marcos Luque Baena**

Fdo.:

Mérida, Febrero de 2017

Palabras Clave

Diseño de interiores, web, tecnología

Resumen

A medida que el ritmo de vida se ha ido acelerando, se ha intensificado la idea de la casa como refugio y ha crecido la importancia que su aspecto y atmósfera tienen para las personas.

Por ello, resulta cada vez más frecuente que recurramos a profesionales del interiorismo para conseguir un diseño exclusivo y bien aprovechado de nuestro espacio.

A la vista de esto, este proyecto propone una solución informática para el diseño de interiores, ya sea una casa, biblioteca, despacho, etc., por medio de tecnologías destinadas para tal fin, y permitir que cualquier persona pueda realizar un diseño personalizado de su propio espacio.

Por lo tanto, se plantea la investigación, implementación y optimización de una plataforma web que permita de una forma simple pero efectiva y a la vez atractiva, el diseño de interiores de cualquier espacio.

Key Words

Interior design, web, technology

Abstract

As the place of life has accelerated, the idea of a home as refuge has been intensified and increased the importance of its appearance and atmosphere for the people living.

Therefore, it is increasingly common looking for a professional interior designer to get an exclusive and well taken advantage of our space design.

In the light of this context, this project provides us with a software solution for interior design, whether a house, library, office, etc., through technologies designed for that purpose, and allows anyone to make a personalized design of their own space.

In conclusion with the mentioned above, this research proposes the implementation and/or optimization of a web platform that allows a simple but effective and attractive way to design the interior of any space.

Agradecimientos

En primer lugar, me gustaría agradecer a Rafa, por el esfuerzo y dedicación que ha mostrado conmigo durante la realización de este proyecto que él mismo me ofreció. Sus conocimientos, maneras de trabajar, disciplina, persistencia y paciencia, han sido fundamentales para mi a lo largo de todo el desarrollo.

A Paco, por tutorizar mi trabajo y guiarme durante todo el proceso. Sus consejos e indudable experiencia han hecho que este proyecto sea llevado a cabo de la manera más profesional posible y tenga un resultado de calidad.

Al resto de profesores, que me han formado como ingeniero durante esta etapa de mi vida y enseñado no solamente los conocimientos específicos, sino además a ser mejor persona, y por estar volcados conmigo ante cualquier situación ya sea docente o personal.

Al Centro Universitario de Mérida, por su gran calidad de planes de estudios ofertados, y que en su día me atrajeron a estudiar aquí, y que hoy en día puedo decir que no me arrepiento de haber elegido este Centro y estoy orgulloso de él.

Y por su puesto, a mi familia y amigos. Gracias por vuestro apoyo incondicional y motivación que me ha ayudado tanto en los momentos buenos, como en los menos buenos.

Gracias.

Índice general

Índice de cuadros

Índice de figuras

Listados de código

Capítulo 1

Introducción

El mundo del diseño está revolucionando cada vez más el concepto que tenemos sobre cómo decorar un espacio. En la actualidad podemos encontrar profesionales dedicados íntegramente a aconsejar a personas sobre como diseñar una casa, una habitación, etc.

A su vez, el modelado 3D nos permite hacer posible lo imposible, desde crear un mundo inexistente con apariencia 100 % real, a realizar inserciones 3D sobre imagen o vídeo real. Un modelo 3D puede "verse" de dos formas distintas. Desde un punto de vista técnico, es un grupo de fórmulas matemáticas que describen un "mundo" en tres dimensiones. Desde un punto de vista visual, valga la redundancia, un modelo en 3D es una representación esquemática visible a través de un conjunto de objetos, elementos y propiedades que, una vez procesados (renderización), se convertirán en una imagen en 3D o una animación 3D. El modelado 3D puede abaratar de una manera espectacular costes en todo tipo de producciones audiovisuales, ahorrando jornadas de grabación o fotografía y consiguiendo en un ambiente totalmente controlado las condiciones necesarias para un acabado perfecto.

Por lo general, el modelo visual suele ser el modelo 3D que los diseñadores manejan, dejando las fórmulas a procesos computacionales. Esto es así, porque lo que el modelo en 3D visual representa se acerca más a la imagen en 3D final que se mostrará al renderizarse.

Existen aplicaciones de modelado en 3D, que permiten una fácil creación y modificación de objetos en tres dimensiones. Estas herramientas suelen tener objetos básicos poligonales (esferas, triángulos, cuadrados, etc.) para ir armando el modelo. Además suelen contar con herramientas para la generación de efectos de iluminación,

texturizado, animación, transparencias, etc. Algunas aplicaciones de modelado son 3D Studio Max, Alias, Blender, Cheetah3D, Cinema 4D, Generative Components, Houdini, LightWave, Maya, MilkShape 3D, Autocad, Solid Work, etc. [21]

Por lo tanto, este proyecto tiene por objetivo desarrollar una aplicación web mediante software libre, que permita el diseño de interiores en 3D, recreando espacios físicos tales como una habitación, despacho, etc, para tener una idea de cómo sería en la realidad. La idea principal de este trabajo es que dichos diseños ayuden a tomar decisiones a la hora de hacer modificaciones en la distribución del espacio o agregar nuevos objetos en un espacio real.

1.1. Motivación y desarrollo

Comenzaremos realizando un estudio preliminar de las tecnologías que permitan trabajar en entornos en tres dimensiones, así como de aplicaciones ya existentes que desempeñen funciones similares a nuestras necesidades. Aunque existen ya en el mercado herramientas informáticas que permiten realizar diseños de interiores en 3D, la gran mayoría utiliza software propietario, y por lo tanto de pago. Por eso, la motivación principal de este proyecto es crear la aplicación utilizando íntegramente software libre u Open Source.

A continuación, realizaremos un breve estudio considerando los puntos fuertes y débiles de nuestra aplicación de cara al mercado y a posibles usuarios. No se pretende competir con aplicaciones ya desarrolladas, sino analizar a qué tipo de usuario estaría destinada nuestra herramienta.

Una vez hayamos realizados los estudios oportunos, es hora de diseñar y desarrollar la herramienta web, realizando, en primer lugar, un análisis de requisitos que debemos cubrir. El diseño y desarrollo serán la parte fundamental del proyecto. Por un lado, un diseño atractivo y moderno motivarán al usuario a utilizar la herramienta, y por otro, un desarrollo con las tecnologías apropiadas, dotarán a la aplicación de las funcionalidades necesarias para que el usuario pueda plasmar su diseño de la manera más real y sencilla posible. El mantenimiento de la plataforma también es una tarea a tener en cuenta, pues un correcto mantenimiento de la misma, asegurará su adecuado funcionamiento a lo largo del tiempo.

Finalmente, se propondrán posibles líneas futuras de investigación ligadas al proyecto. Una de las más representativas es la realidad aumentada, ya que el entorno que queremos crear es el indicado para hacer uso de tal tecnología. La realidad aumentada es el término que se usa para definir una visión a través de un dispositivo tecnológico, de un entorno físico del mundo real, cuyos elementos se combinan con elementos virtuales para la creación de una realidad mixta en tiempo real. Consiste en un conjunto de dispositivos que añaden información virtual a la información física ya existente, es decir, añadir una parte sintética virtual a lo real. Esta es la principal diferencia con la realidad virtual, puesto que no sustituye la realidad física, sino que sobreimprime los datos informáticos al mundo real. Por ello se pretende que en un futuro los espacios creados con nuestra herramienta puedan ser visualizados con herramientas de realidad virtual como pueden ser las gafas de Google Glass, Oculus Rift, Ssamsung Gear VR o HTC Vive y que darán al usuario la sensación de estar recorriendo ellos mismos el espacio 3D que han diseñado. También se pretende poder hacer uso de la realidad aumentada, que permitirá añadir información virtual a la real a través de diferentes dispositivos tecnológicos.[32]

Y para concluir, se mencionarán algunos de los problemas más importantes encontrados a lo largo del desarrollo de la aplicación web y se mostrará la bibliografía más relevante.

1.2. Objetivos y alcance del proyecto

El principal objetivo de este trabajo de fin de grado (TFG) es la creación de una herramienta que permita la representación en 3D de cualquier espacio físico para tener una visión lo más realista posible de cómo sería en la realidad. Por espacio físico nos referimos a cualquier lugar que pueda ser recreado en tres dimensiones y en el que podamos colocar objetos, por lo que según nuestro contexto podría ser una habitación, un despacho o la distribución de una casa completa. Para ello se hará uso de tecnologías como HTML5, CSS3 y Bootstrap para el diseño de la web; JavaScript, Java y JSP para la parte de programación de la plataforma web, que será alojada en un servidor Apache Tomcat; y MySQL como sistema gestor de base de datos, encargado de almacenar toda la información necesaria.

También como objetivos adicionales podemos mencionar:

- Entorno 3D. Proporcionar un entorno capaz de representar un objeto en 3D.

- Introducir objetos propios. La herramienta permitirá a través de la parte de administración, la introducción de nuevos objetos o texturas.
- Representación de cualquier espacio. Capacidad para permitir a los usuarios la creación de un diseño totalmente personalizable.
- Diseño intuitivo. Una interfaz intuitiva y atractiva permitirá tener una mejor experiencia de usuario.
- Integración con un sistema gestor de base de datos (SGBD). Una base de datos será la encargada de gestionar toda la información de los usuarios, objetos y demás elementos relevante de la herramienta.
- Escalabilidad. Posibilidad de ampliar en un futuro la herramienta introduciendo nuevas características de manera sencilla gracias a la utilización de los últimos estándares en programación y diseño web.
- Estudio y utilización de las últimas tecnologías web. Además del desarrollo de nuestra herramienta web, también es importante conocer la actualidad en cuanto al diseño y programación en 3D se refiere para crear un desarrollo profesional.

Además, en el Centro Universitario de Mérida contamos con el Grado en Ingeniería en Diseño Industrial y Desarrollo de Productos, por lo que otro enfoque que se le da al proyecto, y que le daría un valor añadido, es que los alumnos de dicha titulación puedan utilizar la plataforma para probar sus propios diseños 3D y comprobar como quedarían en un espacio en tres dimensiones creado a través de la web, para poder llevarlo a la realidad. Con esto se pretende que el proyecto no se centre solamente en la parte informática, sino que además pueda expandirse a otras ramas, y que están integradas en el Centro.

Por otro lado, la introducción de la realidad aumentada podría ser otro objetivo que podría introducirse en un futuro y que permitiría que la aplicación proporcionase una experiencia mejorada para el usuario, al añadir información virtual a la información física ya existente de los espacios físicos.

Capítulo 2

Estado del arte

2.1. Estudio preliminar

El interés de aplicaciones web está creciendo, ya sea para una simple presentación de productos, visualización de información o sencillamente para proporcionar al usuario una navegación más "enriquecida". Podemos distinguir dos vertientes claramente diferenciadas a la hora de hablar de tecnologías para gráficos 3D en la web:

Basadas en lenguajes de marcado:

- VRML: *Virtual Reality Modeling Language.*
- X3D: Evolución de VRML.
- X3DOM: X3D en HTML (X3D + DOM) .
- Desarrollado por Web 3D *Consortium.*

Basadas en script:

- WebGL: OpenGL + GLSL mediante JavaScript.
- Desarrollado por *Khronos Group.*

Dentro de las mencionadas anteriormente, nos centraremos en X3DOM, como tecnología basada en lenguaje de marcado, y en WebGL para la basada en script.

Para explicar ambas tecnologías antes debemos conocer dos importantes conceptos ligados a la representación de gráficos en la web (HTML), que son Canvas y SVG.

CANVAS	SVG
Orientado al pixel (el Canvas es básicamente un elemento de imagen con un API de dibujo)	Basado en el modelo de objetos (los elementos SVG son similares a los elementos HTML)
Elemento HTML individual, similar en su comportamiento a la etiqueta <img>	Múltiples elementos gráficos que pasan a formar parte del Modelo de Objeto de Documento (DOM)
La representación visual se crea y modifica por programa, mediante *scripting*	La representación visual se genera a partir del markup y se modifica mediante CSS o por programa, mediante *scripting*
El modelo de evento/interacción con el usuario es rudimentario exclusivamente a nivel del elemento Canvas; las interacciones se deben programar manualmente a partir de las coordenadas del ratón	El modelo de evento/interacción con el usuario se basa en el objeto a nivel de elementos gráficos primitivos líneas, rectángulos, rutas
El API no soporta la accesibilidad; deben utilizarse, además del Canvas, otras técnicas basadas en markup	El markup SVG y el modelo de objeto soportan accesibilidad de forma nativa

Cuadro 2.1: Comparativa Canvas y SVG

2.1.1. Canvas y SVG

Canvas y SVG son dos funcionalidades de gráficos tremendamente interesantes que aparecen con Internet Explorer 9 y ambas se benefician de la aceleración por hardware. Estas tecnologías nos pueden servir para resolver numerosos escenarios de uso de gráficos de las webs actuales. A la vista de la gran expectación que ha ocasionado Canvas, parece que la tendencia es ignorar SVG que, en muchos de los casos, es una opción mejor.

A continuación, se resumen a grandes rasgos lo que son SVG y Canvas para delimitar el análisis y determinar mejor cuándo conviene utilizar una tecnología de gráficos vectoriales u otra.

A SVG se le conoce como un modelo de gráficos en modo retenido que persiste dentro de un modelo en memoria. De una manera similar a HTML, SVG genera un modelo de objeto de elementos, atributos y estilos. Cuando el elemento <svg> aparece en un documento HTML5, funciona igual que un bloque "*inline*" y forma parte del árbol de documento HTML.

Canvas es un mapa de bits con una interfaz de programación de aplicaciones (API) para gráficos en modo inmediato para poder dibujar sobre él. Canvas sigue el modelo de "dispara y vete" que restituye los gráficos directamente dentro de su mapa de bits y que no tiene consciencia de las formas que se han dibujado en el lienzo. Lo único que permanece es el mapa de bits resultante.

2.1.1.1. Escenarios de uso

En las líneas siguientes se explica brevemente en qué casos conviene usar una tecnología u otra.

Figura 2.1: Casos de uso de Canvas y SVG

2.1.1.2. Documentos vectoriales complejos de alta fidelidad

Los documentos vectoriales complejos de alta fidelidad han sido y siguen siendo el campo natural de acción para SVG. Un elevado nivel de detalle en la presentación e impresión, tanto a nivel de objeto individual como embebido en una página web. La naturaleza declarativa de SVG es la base para la creación de herramientas o para la generación de formas tanto en el cliente como el servidor a partir de información almacenada en bases de datos.[8]

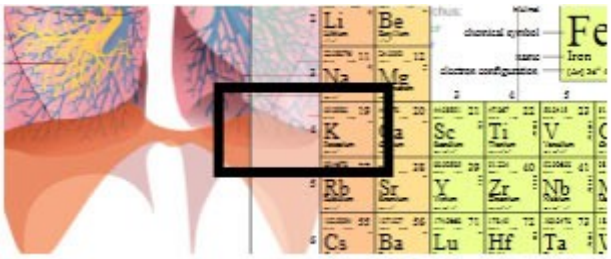

Figura 2.2: Imagen original a analizar

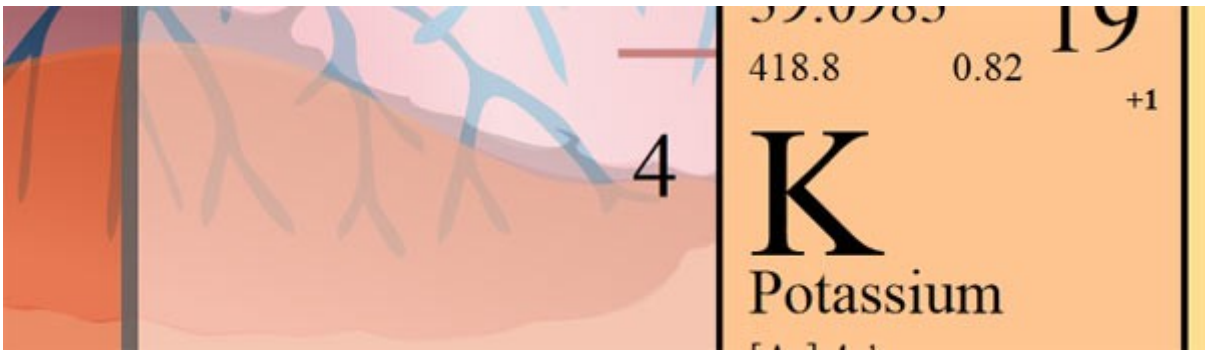

Figura 2.3: Imagen analizada (ampliación)

2.1.1.3. Manipulación de píxeles

Ya que Canvas tiene que ver fundamentalmente con el dibujo y manipulación a nivel de pixel de una superficie, existen ciertos experimentos y ejemplos de Canvas con sofisticados algoritmos que logran unos efectos gráficos realmente impresionantes, como por ejemplo simulación de rayos de luz reflejado o filtrados.

El ejemplo siguiente, llamado "*Ray-Tracing*" ha sido creado por Adam Burmister. En este experimento se genera una imagen a base de calcular el recorrido de los rayos de la luz por todo el campo de pixels en un plano de la imagen y simulando los efectos de su reflejo sobre otros objetos virtuales dentro de la escena.

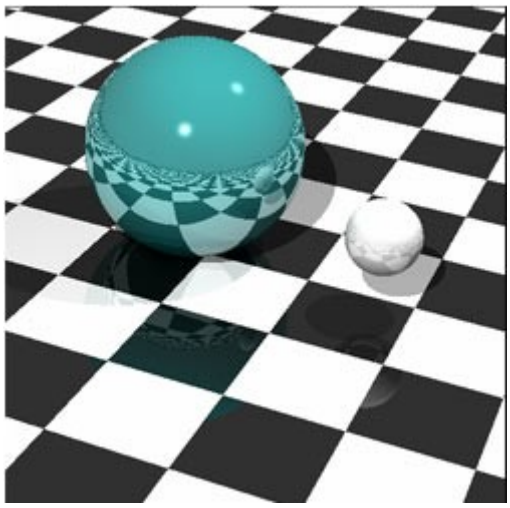

Figura 2.4: Simulación de rayos de luz

2.1.1.4. Escenarios mixtos y ambivalentes

El grupo más interesante de casos de uso no se decanta de manera clara por ninguna de las dos tecnologías. Los podemos ejemplificar con dos escenarios básicos: diagramas/gráficos/mapas y juegos en dos dimensiones.

Los gráficos y diagramas requieren gráficos vectoriales, y aquí pueden valer tanto Canvas como SVG. No obstante, SVG es a menudo una opción más conveniente dadas sus cualidades

2.1.1.5. Escenarios de uso de SVG

Algunas de las familias más conocidas de gráficos y diagramas que podemos ver en la Web son:

- Los diagramas de flujo y organigramas interactivos
- Mapas interactivos y diseño de rutas óptimas
- Planos de construcción y distribución en planta
- Esquemas de ingeniería
- Mapas de ubicación de asientos en líneas aéreas y organización de eventos
- Gráficos financieros o de cualquier otra naturaleza a partir de datos (gráficos de barras, columnas, líneas, tartas, donuts, etc.)

En todos estos casos la tecnología más adecuada es SVG:

- Estos gráficos se pueden generar fácilmente a partir de datos mediante transformaciones de XML a SVG
- Las versiones estáticas se pueden exportar mediante herramientas (por ejemplo, Inkscape, Adobe Illustrator, Microsoft Visio y algunos programas de CAD)
- Exigen precisión en la interacción con el usuario
- Los creadores de las webs pueden personalizar módulos de otros fabricantes utilizando plantillas de estilos CSS
- Cumplen con los requisitos de accesibilidad

2.1.1.6. Diagramas y gráficos con Canvas

También Canvas cuenta con propiedades que permiten el tratamiento de diagramas y gráficos. Para fijar el contexto en este caso, tenemos que comparar el rendimiento de SVG y de Canvas. Algunas veces se producen influencias externas que nos obligan a elegir una tecnología que es, o en principio debería ser, independiente de la funcionalidad. En el caso de SVG y Canvas, hay dos elementos diferenciales básicos.

El conocimiento del desarrollador, sus competencias y los elementos de apoyo que tiene a mano juegan un papel muy importante a la hora de decantarse por una u otra tecnología. Si en el caso del desarrollo de un juego los programadores suelen tener un excelente nivel de conocimientos sobre las APIs de gráficos a bajo nivel y un conocimiento limitado de las tecnologías Web, la tecnología que probablemente elegirán será Canvas. Si se trata de migrar juegos, existen herramientas que soportan la conversión desde implementaciones de terceros a Canvas.

Si el rendimiento es un factor esencial, a menudo en el rango de los milisegundos, será preciso comparar el rendimiento de ambas tecnologías. Esto no quiere decir que Canvas, que habitualmente se considera de alto rendimiento, sea la opción natural. No obstante, en el caso de las aplicaciones que involucran grandes cantidades de datos que deben representarse a nivel de pixel, el Canvas es, sin lugar a dudas, la mejor opción.

2.1.1.7. Juegos en dos dimensiones (2D)

Los juegos ocasionales son el escenario más complicado de analizar. Algunas observaciones preliminares:

- Las librerías de juegos incluyen APIs de gráficos de bajo nivel optimizadas
- Las competencias de los desarrolladores en el mundo de los juegos están muy orientadas hacia el uso de estas APIs de bajo nivel
- Muchos juegos se basan casi exclusivamente o en una gran parte, en imágenes o *sprites*
- Los fabricantes como Adobe empiezan a dar soporte a Canvas como un formato de exportación

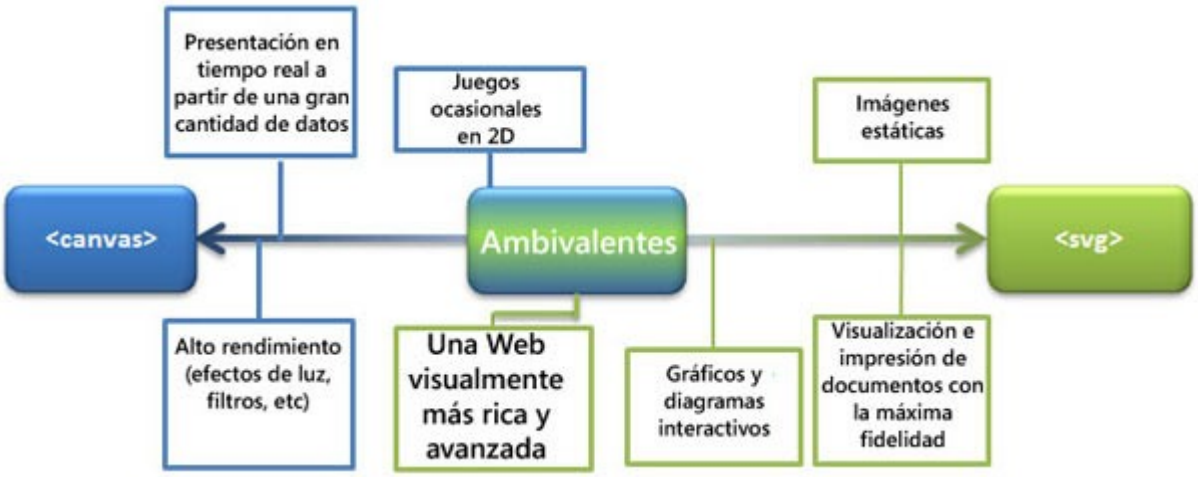

Figura 2.5: Ámbitos de utilización de Canvas y SVG

- Los juegos ocasionales no suelen requerir técnicas sofisticadas de captura de eventos de ratón
- Los juegos ocasionales no tienen normalmente un número exageradamente grande de "objetos"

En las librerías gráficas, como por ejemplo los conocidos motores de efectos de física, el modelo de gráficos es independiente y los gráficos se convierten en un detalle de la implementación. Los elementos de geometría como bordes, velocidades, tamaños y posiciones se envían a los motores y éstos responden en consecuencia con velocidades, colisiones y posiciones. Los gráficos se utilizan únicamente para representar en pantalla la escena previamente calculada.

2.1.1.8. Análisis final

El análisis de las tecnologías de gráficos vectoriales disponibles en las últimas versiones de los navegadores nos muestra cómo se pueden crear nuevos escenarios de uso utilizando las tecnologías Web estándar de forma interactiva.

Vistos los conceptos de Canvas y SVG podemos concluir lo siguiente respecto a WebGL y X3DOM:

	2D	3D
Declarativo: • Gráfico • Parte del documento HTML • Integración en el DOM • CSS/Eventos		
Imperativo: • API de gestión • Contexto de dibujo • Necesidad de JavaScript		

Cuadro 2.2: Equivalencia 2D y 3D

2.1.2. X3DOM

Como se ha indicado al inicio de este capítulo X3DOM es la combinación de X3D y DOM.

Las características más importantes de X3D son:

- Integración XML:
  - Servicios Web, multiplataforma, transferencia de datos entre aplicaciones.
- Compartimentado: o
  - Distribuciones, perfiles y extensiones.
- Visualización en múltiples dispositivos.
- Gráficos, audio y vídeo interactivos en tiempo real.
- Estándar ISO bien especificado

Asimismo, X3 presenta las siguientes funcionalidades:

- Objetos definidos por el usuario: Posibilidad de extender la funcionalidad del navegador para definir tipos de datos definidos por el usuario.
- Scripting: Capacidad de cambio dinámico de la escena utilizando lenguajes de programación y de script.

- Red: Capacidad de composición de una escena X3D a partir de recursos ubicados en una red; hiperenlace de objetos a otras escenas o recursos alojados en cualquier lugar de la WWW.
- Simulación física: Animación humanoide, conjuntos de datos geoespaciales, integración con protocolos DIS (*Distributive Interactive Simulation*). Por lo tanto, al combinar X3D con el DOM tenemos que X3DOM integra los gráficos 3D definidos de forma declarativa en HTML, y que además permite el uso de JavaScript y el acceso al DOM para contenido 3D.

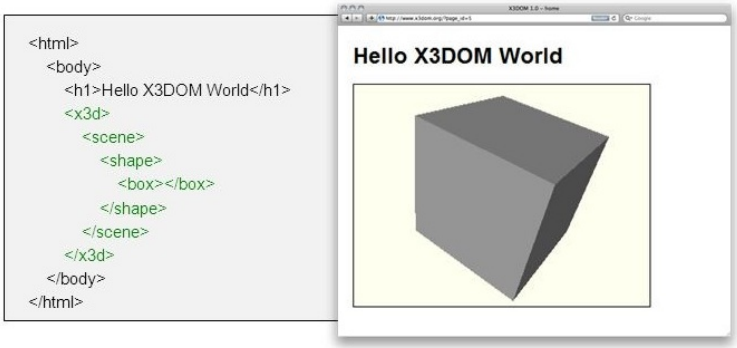

Figura 2.6: "Hello world" con X3DOM

2.1.2.1. Ventajas de X3DOM

Algunas ventajas de X3DOM son:

- Integración nativa con el navegador:
  - No necesita la instalación de plugins o aplicaciones.
  - Evita problemas de instalación, permisos y seguridad
  - Independiente del sistema operativo, especialmente en dispositivos móviles
- Facilita el desarrollo:
  - Entorno de ejecución completo en el navegador.
  - Ubicuidad de navegadores incrementa accesibilidad.
  - Carece de un API especial y avanzada.
- Integración con otras técnicas Web estándar:
  - CSS, Ajax, etc.

- Descripción de contenido declarativo y legible:
  - Encapsula detalles gráficos de bajo nivel.
  - Interacción con el usuario mediante APIs Web estándar (p.ej. JavaScript y CSS).
  - Interoperabilidad: El mismo código se ejecuta en escritorio, red y dispositivo móvil.
- Integración HTML vs Sistema cerrado:
  - Metadatos. Facilita la indexación y búsqueda.
  - Permite la combinación de contenidos de diversas fuentes (*mashups*).
  - Conexión de datos existentes (p.ej. geo-información) con contenido 3D.

2.1.3. WebGL

WebGL es una interfaz de programación de gráficos en 3D dentro de páginas web, al igual que X3DOM.

Tiene su origen en 2006, desarrollado por Mozilla y utilizando su propio Canvas3D.

Actualmente desarrollado por Khronos Group del cual forman parte tanto Mozilla, como Opera, Apple y Google.

La primera especificación WebGL (WebGL 1.0) fue liberada en febrero de 2011. (`https://www.khronos.org/registry/webgl/specs/1.0/`)

Podemos decir por lo tanto que WebGL es una herramienta multiplataforma para dibujar contenido 3D en HTML que se basa en OpenGL ES 2.0. Además, se apoya en JavaScript para acceder al DOM a bajo nivel y utiliza el elemento Canvas (<canvas>) de HTML5 para realizar los gráficos.

Es nativamente soportado por Firefox, Chrome, Opera y Safari. Asimismo existe una web donde se puede comprobar la compatibilidad del navegador:
`http://www.doesmybrowsersupportwebgl.com/`

En cuanto a los modelos utilizados, se suelen importar modelos 3D creados con programas como Blender y exportados de otros, o importar modelos ya creados.

Se puede trabajar a bajo nivel con el API de programación que WebGL proporciona o utilizar algunas librerías de alto nivel que encapsulan esta API. Algunas de estas librerías son:

- WebGLU: la primera librería WebGL disponible públicamente.
- C3DL: Canvas 3D JavaScript Library.
- Librerías alto nivel: Three.js, GLGE, SceneJS, Oak3D
- Processing.js: utiliza WebGL para renderizar un objeto 2D o 3D, programado en *Processing*, dentro de un canvas de HTML.
- GLOW: Envolvente (wrapper) de WebGL desarrollado por la BBC (*British Broadcasting Corporation*).
- SpiderGL: librería gráfica JavaScript 3D basada en WebGL.
- gwt-g3d: librería 3D para GWT (Google Web Toolkit).

```
<canvas id="webgl-canvas"></canvas>
<script type="application/javascript">
    var canvas = document.getElementById("webgl-canvas");
    var renderer = new THREE.WebGLRenderer({
        canvas: canvas,
        antialias: true
    });
    renderer.setClearColor(new THREE.Color(0x000000), 1);
    renderer.setSize(document.body.offsetWidth, document.body.offsetHeight);

    var camera =
        new THREE.PerspectiveCamera(45, canvas.width / canvas.height, 1, 200);
    camera.position.set(70, 80, 90);
    camera.lookAt(new THREE.Vector3(0, 0, 0));

    var scene = new THREE.Scene();

    var cube = new THREE.Mesh(
        new THREE.BoxGeometry(50, 50, 50),
        new THREE.MeshNormalMaterial());
    scene.add(cube);

    renderer.render(scene, camera);
</script>
```

Figura 2.7: “Hello world” con WebGL

2.1.3.1. Ventajas de WebGL

Las ventajas más importantes de WebGL son:

- WebGL se basa en un conocido y ampliamente aceptado estándar de gráficos 3D (OpenGL).

- Compatibilidad con distintos navegadores y plataformas.
- Al ser una API DOM, es compatible con todos los elementos del estándar HTML.
- Gráficos 3D acelerados por hardware (CPU y tarjeta gráfica).
- Al utilizar lenguajes de programación interpretados no es necesario compilar tu código antes de renderizarlo y mostrarlo por pantalla.

Resumiendo, el principio del funcionamiento de X3DOM nos permitiría describir la escena de una manera más sencilla, pero cuyas modificaciones o personalizaciones serían más limitadas. Por el contrario, WebGL nos permitiría la creación de las escenas a través de código JavaScript pero necesitaríamos de bastante código adicional para realizar las modificaciones más básicas, por lo que estaríamos trabajando a más bajo nivel pero tendríamos más control sobre el escenario.

La decisión de una tecnología u otra podría reducirse a la siguiente pregunta: ¿Queremos tener un enfoque más de diseño o de programador? Ninguna de las dos opciones es mala, pero en nuestro caso necesitaremos realizar la implementación de funcionalidades que cubran las necesidades de nuestro proyecto, y que veremos más adelante, por lo tanto debemos decantarnos por la opción de darle un toque más de programador sobre el de diseñador. Por lo tanto, la tecnología a utilizar para el desarrollo de nuestro trabajo será **WebGL**.

Además trabajaremos con una librería de alto nivel que nos facilitará la implementación: **Three.js**.

2.1.4. Three.js

Three.js es una biblioteca liviana escrita en JavaScript para crear y mostrar gráficos animados por ordenador en 3D en un navegador Web y puede ser utilizada en conjunción con el elemento Canvas de HTML5, SVG o WebGL. El código fuente está alojado en un repositorio en GitHub.

Se ha popularizado como una de las más importantes para la creación de las animaciones en WebGL.

Bibliotecas de alto nivel tales como Three.js o GlgE, SceneJS, PhiloGL u otras, permiten a los desarrolladores la implementación de complejas animaciones 3D que se muestran en el navegador sin la necesidad de una aplicación independiente o algún plugin.

Además, JavaScript (abreviado comúnmente JS) es un lenguaje de programación interpretado, dialecto del estándar ECMAScript, y relativamente fácil de aprender debido a que es un lenguaje de scripting. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. Se utiliza principalmente en su forma del lado del cliente (client-side), implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas. Su uso en aplicaciones externas a la web, por ejemplo en documentos PDF, aplicaciones de escritorio (mayoritariamente widgets) es también significativo. Además, no necesita prácticamente ninguna configuración en el navegador de Internet.

2.1.4.1. Contexto histórico

Esta biblioteca fue creada y liberada en GitHub por el español Ricardo Cabello [41] en abril de 2010, conocido por su seudónimo de Mr.doob. El código habría sido primeramente desarrollado en ActionScript y luego traducido al javaScript. Los dos puntos decisivos para la transferencia a JavaScript fueron no tener que compilar el código antes de cada carga y la independencia de plataforma. Las contribuciones de Cabello incluyen el diseño de la API, CanvasRenderer, SVGRenderer y ser responsable de la fusión del trabajo de los diversos colaboradores en el proyecto.

Con el advenimiento de WebGL, Paul Brunt añade el renderizador como un módulo en lugar de en el propio núcleo.

Branislav Ulicny se une en 2010, después de haber publicado una serie de demos WebGL en su propio sitio, con la intención de que las capacidades del renderizador WebGL en Three.js excedieran los CanvasRenderer y SVGRenderer. Sus principales contribuciones generalmente involucran materiales, shaders y post-procesamiento.

Poco después de la introducción de WebGL en Firefox 4 en marzo de 2011 Joshua Koo se une construyendo su primer demo de texto en 3D en septiembre de 2011. Sus

contribuciones con frecuencia se refieren a la generación de la geometría.

Actualmente cuenta con la contribución de alrededor de 90 codificadores, entre ellos incluyendo "gero3", "WestLangley", Jerome Etienne, Erik Kitson y "AddictArts" y una comunidad creciente de programadores.

Three.js así como el propio estándar de WebGL están todavía prácticamente en sus días de nacimiento por lo que aún es realmente prematuro hablar de una verdadera historia que el tiempo irá construyendo paso a paso. [45]

Además, en su página web oficial [41] podemos encontrar multitud de proyectos que hacen uso de three.js, desde simples animaciones, hasta juegos.

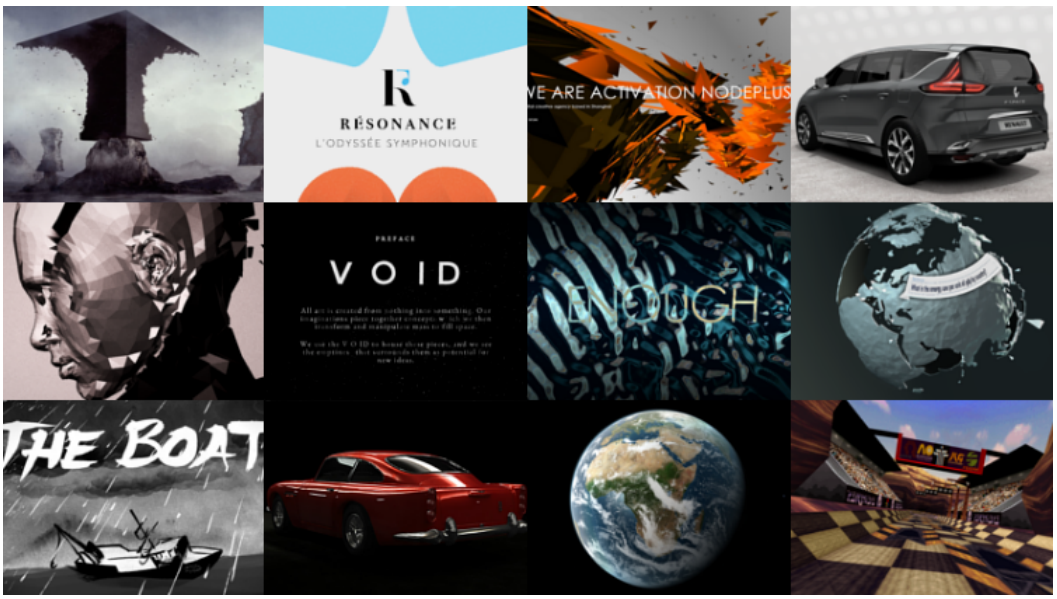

Figura 2.8: Web Three.js

2.1.4.2. Características

Las principales características de Three.js son:

- Renderizadores: Canvas, SVG y WebGL.
- Efectos: anaglifo, bizco (*cross-eyed*) y la barrera de paralaje.
- Escenas: añadir y eliminar objetos en tiempo de ejecución; niebla.
- Cámaras: perspectiva y ortográfica; controladores: trackball, FPS, trayectoria y otras.

- Animación: armaduras, cinemática directa, cinemática inversa, morphing y fotogramas clave.
- Luces: ambiente, dirección, luz de puntos y espacios, sombras: emite y recibe.
- Materiales: Lambert, Phong, sombreado suave, texturas y otras.
- Shaders: el acceso a las capacidades del OpenGL Shading Language (GLSL): reflejos en la lente, pase profundo y una extensa biblioteca de post-procesamiento.
- Objetos: mallas, partículas, sprites, líneas, cintas, huesos y otros.
- Geometría: plano, cubo, esfera, toroide, texto en 3D y otras; modificadores: torno, extrusión y tubo.
- Cargadores de datos: binario, imagen, JSON y escena.
- Utilidades: conjunto completo de funciones matemáticas en 3D, incluyendo tronco, matriz Quaternian, UVs y otras.
- Exportación e importación: utilidades para crear archivos compatibles con JSON Three.js desde: Blender, openCTM, FBX, Max, y OBJ.
- Soporte: La documentación de la API se encuentra en construcción, foro público y wiki en pleno funcionamiento.
- Ejemplos: Más de 150 archivos de códigos de ejemplo más las fuentes, modelos, texturas, sonidos y otros archivos soportados.
- Depuración: Stats.js, WebGL Inspector, Three.js Inspector.

Three.js funciona en todos los navegadores compatibles con WebGL.

Three.js está disponible bajo la licencia MIT. [33]

2.2. Tecnología a utilizar

Como hemos avanzado en la sección anterior, utilizaremos WebGL apoyado en la librería Three.js. Pero además necesitamos un software especializado en el diseño de objetos o piezas en 3D.

La elección del software "adecuado" para el modelado 3D de los objetos dependerá del tipo de proyecto que se pretende abarcar, la experiencia y habilidad con la que el diseñador o usuario cuenta, y los recursos físicos y técnicos que se tienen a disposición, entre otros factores. En nuestro caso, el software elegido para realizar los modelados en 3D será Blender. La elección de Blender como software de modelado se debe principalmente a que es software libre, pero además porque se integra muy bien con las tecnologías que vamos a utilizar y que explicaremos más adelante.

Además, Blender es una de las aplicaciones que mejor se integra con la importación y exportación de archivos con Three.js.

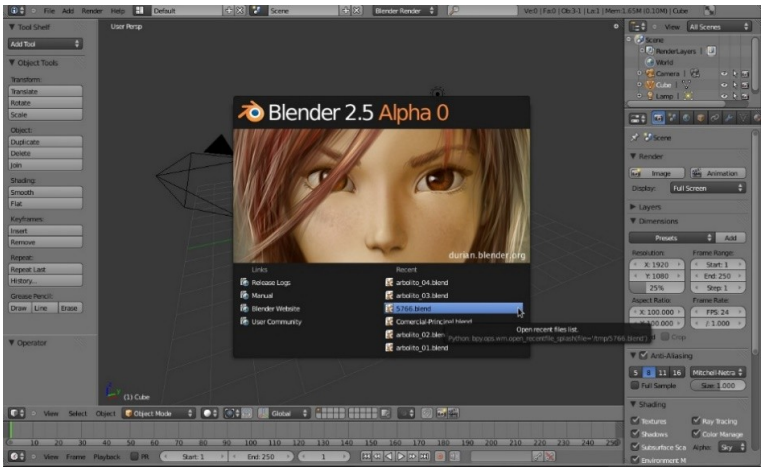

Figura 2.9: Modelado 3D con Blender

2.3. Revisión de proyectos anteriores

Una vez hemos analizado las tecnologías disponibles para desarrollar nuestro proyecto, es hora de hacer una revisión de proyectos que ya existan y que tengan como objetivo el diseño en 3D a través de la web.

2.3.1. ThreeFab

ThreeFab es una herramienta que permite a los diseñadores y desarrolladores fabricar y manipular una escena three.js rápidamente. Está actualmente una versión alfa como una prueba de concepto. Para utilizarlo solamente se necesita un navegador que soporte WebGL [20] .

Esta herramienta se puede probar a través de su web: `http://blackjk3.github.io/threefab/`

ThreeFab presenta las siguientes características:

- Controles de teclado para la escena y objetos.
- Orientación y posicionamiento de objetos.
- Transformaciones de objetos en 3D.
- Incorporación de texturas a los objetos.
- Capacidad para exportar una escena.
- Posibilidad de incorporar objetos animados (animaciones)

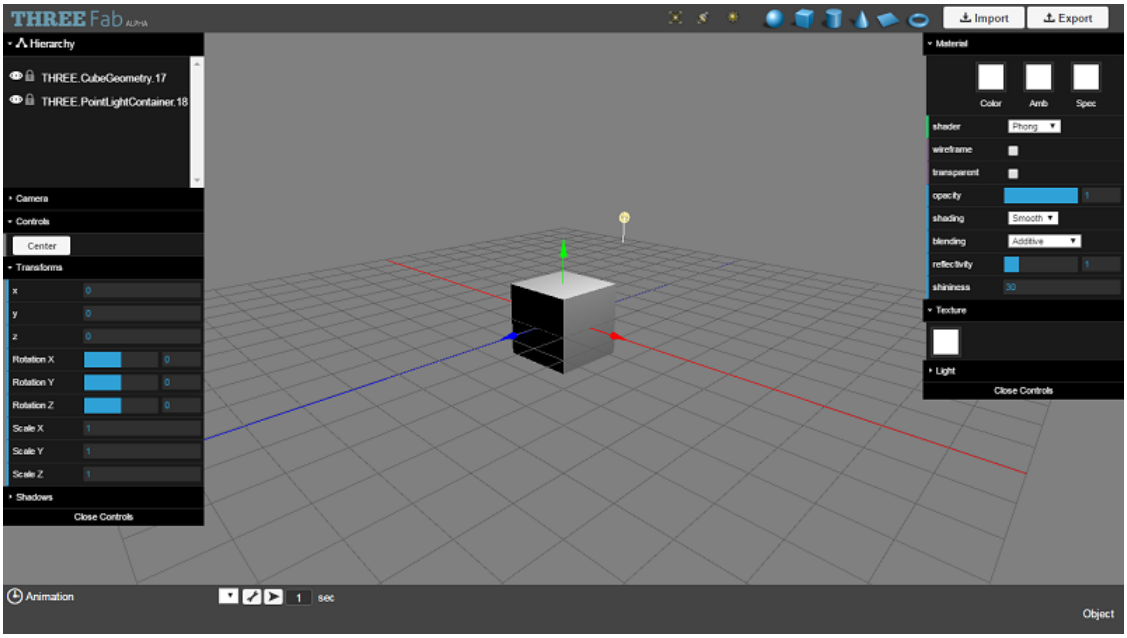

Figura 2.10: ThreeFab

Atendiendo a las características que ThreeFab tiene, puede ser un buen punto de partida para nuestro proyecto, pero debemos seguir profundizando más.

2.3.2. Blueprint3D

Se trata de una aplicación web basada en three.js que permite a los usuarios diseñar un espacio interior, como una casa o apartamento [13]. Cuenta con tres apartados principales que son:

- Editar el plano en 2D
- Añadir objetos
- Realizar el diseño

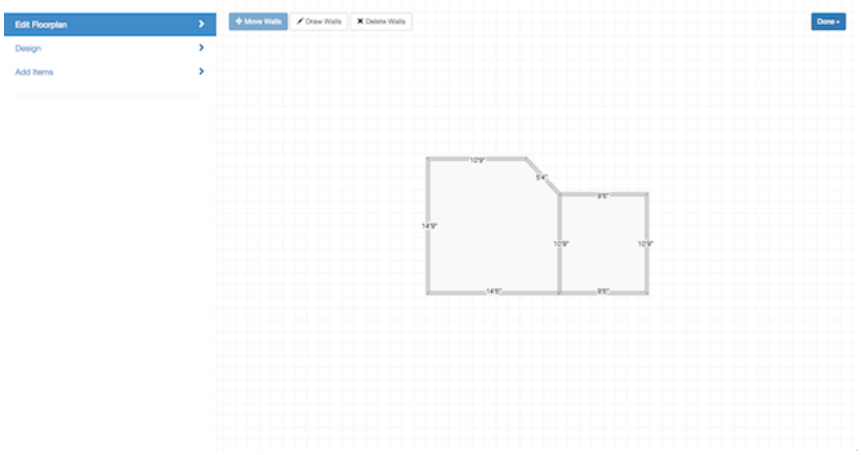

Figura 2.11: Blueprint3D: Editor de plano en 2D

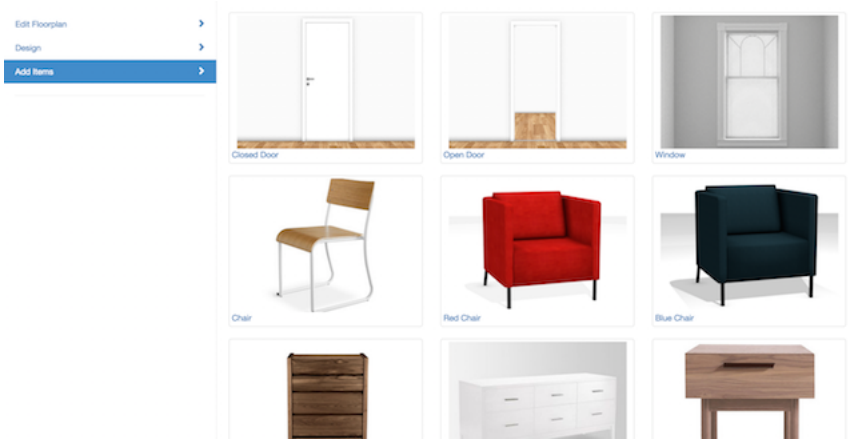

Figura 2.12: Blueprint3D: Añadir objetos al diseño

Figura 2.13: Blueprint3D: Realizar diseño en 3D

Blueprint3D es un proyecto open-source en el que cualquier persona puede contribuir si lo desea, o utilizarlo para adaptarlo a ciertas necesidades. Utiliza WebGL como motor gráfico de 3D y la librería three.js. Se puede probar a través de su enlace oficial: `http://furnishup.github.io/blueprint3d/example/`

Como podemos apreciar, Blueprint3D reúne bastante de los requisitos básicos solicitados, tales como:

- Crear un diseño dado su plano.
- Añadir objetos al diseño.
- Importar / exportar el diseño.

Además, como podemos leer en su página de GitHub [13], su instalación es muy sencilla y solamente requiere de un servidor HTTP.

Podemos afirmar con total seguridad, que este puede ser un buen punto de partida para nuestro proyecto. Combinando las funcionalidades que Blueprint3D nos ofrece, desarrollando el resto de requerimientos solicitados por el cliente y dotando a la web de un sistema gestor de base de datos que gestione toda la información, podremos desarrollar un proyecto bastante atractivo.

Capítulo 3

Metodología de desarrollo

En este capítulo hablaremos sobre la metodología de diseño y desarrollo utilizada para crear nuestro software. La metodología de desarrollo de software en ingeniería de software es un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información.

Desarrollar un software significa construirlo simplemente mediante su descripción. Esta es una muy buena razón para considerar la actividad de desarrollo de software como una ingeniería. En un nivel más general, la relación existente entre un software y su entorno es clara ya que el software es introducido en el mundo de modo que provoque ciertos efectos en el mismo. Aquellas partes del mundo que afectarán al software y que serán afectadas por él será el dominio de aplicación. Es allí donde los usuarios o clientes observarán si el desarrollo del software ha cumplido su propósito. [10]

Una de las mayores deficiencias en la práctica de construcción de software es la poca atención que se presta a la discusión del problema, y por eso queremos tener el máximo número de detalles o requisitos para desarrollar la aplicación. Evidentemente, un software está sujeto a cambios durante su desarrollo, eso es algo que no se puede evitar. Siempre surgen aspectos nuevos que el cliente quiere introducir porque ha visto en algúna herramienta similar, o no se especificó con claridad en las entrevistas previas, o funcionalidades que deben ser cambiadas o eliminadas. Es por ello que la comunicación con el cliente a lo largo del proceso de desarrollo es muy importante.

A la hora de desarrollar un software, tenemos que regirnos por un principio básico (principio rector) que podemos denominar "filosofía de la metodología". La filosofía de la metodología es la norma o idea fundamental que rige el pensamiento o la conducta,

y orienta el análisis, diseño y desarrollo del software. Es el "Principio", el que ordena y estructura las herramientas que son aplicables en la metodología, así como los procedimientos con los que se aplica. Tradicionalmente, se apellida a cada metodología en función del principio que la rige:

- **Metodología estructurada**: se fundamenta en que lo más importante de un sistema de información, son las estructuras que lo componen y que, por lo tanto, el análisis se debe centrar en ellas, descomponiéndolas en nuevas subestructuras hasta tener elementos tan simples, que puedan ser resueltos en forma sencilla.
- **Metodología orientada a objetos**: indica que el principio rector es la orientación a objetos, es decir el análisis de todos los componentes del sistema como un conjunto de objetos que poseen propiedades y que, a través de mensajes, se interrelacionan entre sí.

Como para la realización de este proyecto se utilizará programación orientada a objetos (POO), la metodología de desarrollo debe seguir también este mismo principio.

3.1. Metodología orientada a objetos

3.1.1. Conceptos (Objeto, Clase, Encapsulamiento, Herencia, etc.)

La metodología orientada a objetos ha derivado de las metodologías anteriores a éste. Así como los métodos de diseño estructurado realizados guían a los desarrolladores que tratan de construir sistemas complejos utilizando algoritmos como sus bloques fundamentales de construcción, similarmente los métodos de diseño orientado a objetos han evolucionado para ayudar a los desarrolladores a explotar el poder de los lenguajes de programación basados en objetos y orientados a objetos, utilizando las clases y objetos como bloques de construcción básicos.

Actualmente el modelo de objetos ha sido influenciado por un número de factores no sólo de la Programación Orientada a Objetos, POO (*Object Oriented Programming*, OOP por sus siglas en inglés). Además, el modelo de objetos ha probado ser un concepto uniforme en las ciencias de la computación, aplicable no sólo a los lenguajes

de programación sino también al diseño de interfaces de usuario, bases de datos y arquitectura de computadoras por completo. La razón de ello es, simplemente, que una orientación a objetos nos ayuda a hacer frente a la inherente complejidad de muchos tipos de sistemas.

Se define a un objeto como "una entidad tangible que muestra alguna conducta bien definida". Un objeto "es cualquier cosa, real o abstracta, acerca de la cual almacenamos datos y los métodos que controlan dichos datos".

Los objetos tienen una cierta "integridad" la cual no deberá ser violada. En particular, un objeto puede solamente cambiar estado, conducta, ser manipulado o estar en relación con otros objetos de manera apropiada a este objeto.

Actualmente, el Análisis Orientado a Objetos (AOO) va progresando como método de análisis de requisitos por derecho propio y como complemento de otros métodos de análisis. En lugar de examinar un problema mediante el modelo clásico de entrada-proceso-salida (flujo de información) o mediante un modelo derivado exclusivamente de estructuras jerárquicas de información, el AOO introduce varios conceptos nuevos. Estos conceptos nuevos le parecen inusuales a mucha gente, pero son bastante naturales.

Una clase es una plantilla para objetos múltiples con características similares. Las clases comprenden todas esas características de un conjunto particular de objetos. Cuando se escribe un programa en lenguaje orientado a objetos, no se definen objetos verdaderos sino se definen clases de objetos.

Una instancia de una clase es otro término para un objeto real. Si la clase es la representación general de un objeto, una instancia es su representación concreta. A menudo se utiliza indistintamente la palabra objeto o instancia para referirse, precisamente, a un objeto.

En los lenguajes orientados a objetos, cada clase está compuesta de dos cualidades: atributos (estado) y métodos (comportamiento o conducta). Los atributos son las características individuales que diferencian a un objeto de otro (ambos de la misma clase) y determinan la apariencia, estado u otras cualidades de ese objeto. Los

atributos de un objeto incluyen información sobre su estado.

Los métodos de una clase determinan el comportamiento o conducta que requiere esa clase para que sus instancias puedan cambiar su estado interno o cuando dichas instancias son llamadas para realizar algo por otra clase o instancia. El comportamiento es la única manera en que las instancias pueden hacerse algo a sí mismas o tener que hacerles algo. Los atributos se encuentran en la parte interna mientras que los métodos se encuentran en la parte externa del objeto, como se representa en la siguiente figura.

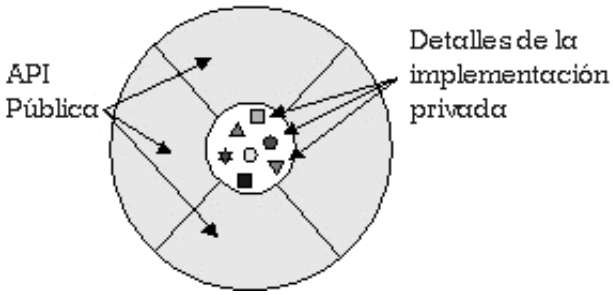

Figura 3.1: Representación visual de un objeto como componente de software

Para definir el comportamiento de un objeto, se crean métodos, los cuales tienen una apariencia y un comportamiento igual al de las funciones en otros lenguajes de programación, los lenguajes estructurados, pero se definen dentro de una clase. Los métodos no siempre afectan a un solo objeto; los objetos también se comunican entre sí mediante el uso de métodos. Una clase u objeto puede llamar métodos en otra clase u objeto para avisar sobre los cambios en el ambiente o para solicitarle a ese objeto que cambie su estado.

Cualquier cosa que un objeto no sabe, o no puede hacer, es excluida del objeto. Además, como se puede observar de los diagramas, las variables del objeto se localizan en el centro o núcleo del objeto. Los métodos rodean y esconden el núcleo del objeto de otros objetos en el programa. Al empaquetamiento de las variables de un objeto con la protección de sus métodos se le llama encapsulamiento. Típicamente, el encapsulamiento es utilizado para esconder detalles de la puesta en práctica no importantes de otros objetos. Entonces, los detalles de la puesta en práctica pueden cambiar en cualquier tiempo sin afectar otras partes del programa.

Esta imagen conceptual de un objeto —un núcleo de variables empaquetadas en una membrana protectora de métodos— es una representación ideal de un objeto y es el ideal por el que los diseñadores de sistemas orientados a objetos luchan. Sin embargo, no lo es todo: a menudo, por razones de eficiencia o la puesta en práctica, un objeto puede querer exponer algunas de sus variables o esconder algunos de sus métodos.

El encapsulamiento de variables y métodos en un componente de software ordenado es, todavía, una simple idea poderosa que provee dos principales beneficios a los desarrolladores de software:

- Modularidad, esto es, el código fuente de un objeto puede ser escrito, así como darle mantenimiento, independientemente del código fuente de otros objetos. Así mismo, un objeto puede ser transferido alrededor del sistema sin alterar su estado y conducta.

- Ocultamiento de la información, es decir, un objeto tiene una "interfaz publica" que otros objetos pueden utilizar para comunicarse con él. Pero el objeto puede mantener información y métodos privados que pueden ser cambiados en cualquier tiempo sin afectar a los otros objetos que dependan de ello. Los objetos proveen el beneficio de la modularidad y el ocultamiento de la información. Las clases proveen el beneficio de la reutilización. Los programadores de software utilizan la misma clase, y por lo tanto el mismo código, una y otra vez para crear muchos objetos.

En las implantaciones orientadas a objetos se percibe un objeto como un paquete de datos y procedimientos que se pueden llevar a cabo con estos datos. Esto encapsula los datos y los procedimientos. La realidad es diferente: los atributos se relacionan al objeto o instancia y los métodos a la clase. ¿Por qué se hace así? Los atributos son variables comunes en cada objeto de una clase y cada uno de ellos puede tener un valor asociado, para cada variable, diferente al que tienen para esa misma variable los demás objetos. Los métodos, por su parte, pertenecen a la clase y no se almacenan en cada objeto, puesto que sería un desperdicio almacenar el mismo procedimiento varias veces y ello va contra el principio de reutilización de código.

Otro concepto muy importante en la metodología orientada a objetos es el de herencia. La herencia es un mecanismo poderoso con el cual se puede definir una clase

en términos de otra clase; lo que significa que cuando se escribe una clase, sólo se tiene que especificar la diferencia de esa clase con otra, con lo cual, la herencia dará acceso automático a la información contenida en esa otra clase.

Con la herencia, todas las clases están arregladas dentro de una jerarquía estricta. Cada clase tiene una superclase (la clase superior en la jerarquía) y puede tener una o más subclases (las clases que se encuentran debajo de esa clase en la jerarquía). Se dice que las clases inferiores en la jerarquía, las clases hijas, heredan de las clases más altas, las clases padres.

Las subclases heredan todos los métodos y variables de las superclases. Es decir, en alguna clase, si la superclase define un comportamiento que la clase hija necesita, no se tendrá que redefinir o copiar ese código de la clase padre, como representa la figura a continuación.

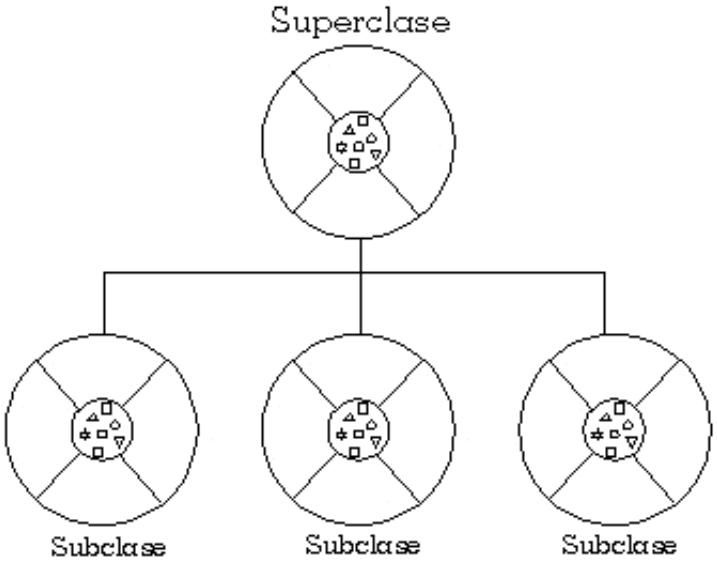

Figura 3.2: Representación de clases y superclases

De esta manera, se puede pensar en una jerarquía de clase como la definición de conceptos demasiado abstractos en lo alto de la jerarquía y esas ideas se convierten en algo más concreto conforme se desciende por la cadena de la superclase.

Sin embargo, las clases hijas no están limitadas al estado y conducta provistos por sus superclases; pueden agregar variables y métodos además de los que ya heredan

de sus clases padres. Las clases hijas pueden, también, sobrescribir los métodos que heredan por implementaciones especializadas para esos métodos. De igual manera, no hay limitación a un sólo nivel de herencia por lo que se tiene un árbol de herencia en el que se puede heredar varios niveles hacia abajo y mientras más niveles descienda una clase, más especializada será su conducta.

La herencia presenta los siguientes beneficios:

- Las subclases proveen conductas especializadas sobre la base de elementos comunes provistos por la superclase. A través del uso de herencia, los programadores pueden reutilizar el código de la superclase muchas veces.
- Los programadores pueden implementar superclases llamadas clases abstractas que definen conductas "genéricas". Las superclases abstractas definen, y pueden implementar parcialmente, la conducta pero gran parte de la clase no está definida ni implementada. Otros programadores concluirán esos detalles con subclases especializadas.

3.1.2. Ventajas de la metodología orientada a objetos

Resumiendo lo visto anteriormente, podemos decir que algunas ventajas la POO que presenta son:

- Reutilización. Las clases están diseñadas para que se reutilicen en muchos sistemas. Para maximizar la reutilización, las clases se construyen de manera que se puedan adaptar a los otros sistemas. Un objetivo fundamental de las técnicas orientadas a objetos es lograr la reutilización masiva al construir el software.
- Estabilidad. Las clases diseñadas para una reutilización repetida se vuelven estables, de la misma manera que los microprocesadores y otros chips se hacen estables.
- El diseñador piensa en términos del comportamiento de objetos y no en detalles de bajo nivel. El encapsulamiento oculta los detalles y hace que las clases complejas sean fáciles de utilizar.
- Se construyen clases cada vez más complejas. Se construyen clases a partir de otras clases, las cuales a su vez se integran mediante clases. Esto permite construir componentes de software complejos, que a su vez se convierten en bloques de construcción de software más complejo.

- Calidad. Los diseños suelen tener mayor calidad, puesto que se integran a partir de componentes probados, que han sido verificados y pulidos varias veces.
- Un diseño más rápido. Las aplicaciones se crean a partir de componentes ya existentes. Muchos de los componentes están construidos de modo que se pueden adaptar para un diseño particular.
- Integridad. Las estructuras de datos (los objetos) sólo se pueden utilizar con métodos específicos. Esto tiene particular importancia en los sistemas cliente-servidor y los sistemas distribuidos, en los que usuarios desconocidos podrían intentar el acceso al sistema.
- Mantenimiento más sencillo. El programador encargado del mantenimiento cambia un método de clase a la vez. Cada clase efectúa sus funciones independientemente de las demás.
- Una interfaz de pantalla sugestiva para el usuario. Hay que utilizar una interfaz de usuario gráfica de modo que el usuario apunte a iconos o elementos de un menú desplegado, relacionados con los objetos. En determinadas ocasiones, el usuario puede ver un objeto en la pantalla. Ver y apuntar es más fácil que recordar y escribir.
- Independencia del diseño. Las clases están diseñadas para ser independientes del ambiente de plataformas, hardware y software. Utilizan solicitudes y respuestas con formato estándar. Esto les permite ser utilizadas en múltiples sistemas operativos, controladores de bases de datos, controladores de red, interfaces de usuario gráficas, etc. El creador del software no tiene que preocuparse por el ambiente o esperar a que éste se especifique.
- Interacción. El software de varios proveedores puede funcionar como conjunto. Un proveedor utiliza clases de otros. Existe una forma estándar de localizar clases e interactuar con ellas. El software desarrollado de manera independiente en lugares ajenos debe poder funcionar en forma conjunta y aparecer como una sola unidad ante el usuario.
- Computación Cliente-Servidor. En los sistemas cliente-servidor, las clases en el software cliente deben enviar solicitudes a las clases en el software servidor y recibir respuestas. Una clase servidor puede ser utilizada por clientes diferentes. Estos clientes sólo pueden tener acceso a los datos del servidor a través de los métodos de la clase. Por lo tanto los datos están protegidos contra su corrupción.

- Computación de distribución masiva. Las redes a nivel mundial utilizarán directorios de software de objetos accesibles. El diseño orientado a objetos es la clave para la computación de distribución masiva. Las clases de una máquina interactúan con las de algún otro lugar sin saber donde residen tales clases. Ellas reciben y envían mensajes orientados a objetos en formato estándar.

- Mayor nivel de automatización de las bases de datos. Las estructuras de datos (los objetos) en las bases de datos orientadas a objetos están ligadas a métodos que llevan a cabo acciones automáticas. Una base de datos OO tiene integrada una inteligencia, en forma de métodos, en tanto que una base de datos relacional básica carece de ello.

- Migración. Las aplicaciones ya existentes, sean orientadas a objetos o no, pueden preservarse si se ajustan a un contenedor orientado a objetos, de modo que la comunicación con ella sea a través de mensajes estándar orientados a objetos.

- Mejores herramientas CASE. Las herramientas CASE (*Computer Aided Software Engineering*, Ingeniería de Software Asistida por Computadora) utilizarán las técnicas gráficas para el diseño de las clases y de la interacción entre ellas, para el uso de los objetos existentes adaptados a nuevas aplicaciones. Las herramientas deben facilitar el modelado en términos de eventos, formas de activación, estados de objetos, etc. Las herramientas OO del CASE deben generar un código tan pronto se definan las clases y permitir al diseñador utilizar y probar los métodos recién creados. Las herramientas se deben diseñar de manera que apoyen el máximo de creatividad y una continua afinación del diseño durante la construcción.[37]

3.2. Ciclo de vida orientado a objetos

Los tipos de ciclos de vida tradicionales (metodología estructurada) son relativos al análisis y diseño estructurados, pero los objetos tienen una particularidad, y es que están basados en componentes que se relacionan entre ellos a través de interfaces, o lo que es lo mismo, son mas modulares y por lo tanto el trabajo se puede dividir en un conjunto de miniproyectos.

Además, hoy en día la tendencia es a reducir los riesgos, y en este sentido, el ciclo de vida en cascada no proporciona muchas facilidades. Debido a todo esto, el ciclo de

vida típico en una metodología de diseño orientado a objetos es iterativo e incremental.

Un proyecto se divide en las fases:

1. **Planificación del negocio**: Se establecen los elementos básicos para la realización del proyecto. Es una fase de análisis y estudio de las necesidades del software (análisis de requisitos).

2. **Construcción**: El centro de la metodología. Es la más importante y se divide a su vez en otras cinco actividades

   a) **Planificación**: Se evalúan los requerimientos y con ellos un análisis inicial del software a desarrollar.

   b) **Investigación**: Sobre los elementos establecidos en la actividad anterior se realiza una investigación de las tecnologías vinculadas al diseño y construcción.

   c) **Especificación**. Se realiza el detalle del diseño de los elementos que serán implementados para el software.

   d) **Implementación**. Se efectúa la construcción del software.

   e) **Revisión**. Todo proceso debe ser verificado y el código para garantizar su calidad debe ser sometido a una serie de pruebas.

3. **Entrega**. Esta fase implica un proceso complejo, puesto que el software no finaliza con la escritura de la última línea de código, si no que hay que garantizar que funciona en los equipos de usuario (evaluación y pruebas).

La primera y la tercera fase son independientes de la metodología de desarrollo orientado a objetos. Además de las tres fases, existen dos periodos:

1. **Crecimiento**: Es el tiempo durante el cual se construye el sistema

2. **Madurez**: Es el periodo de mantenimiento del producto. Cada mejora se planifica igual que el periodo anterior, es decir, con las fases de Planificación del negocio, Construcción y Entrega.

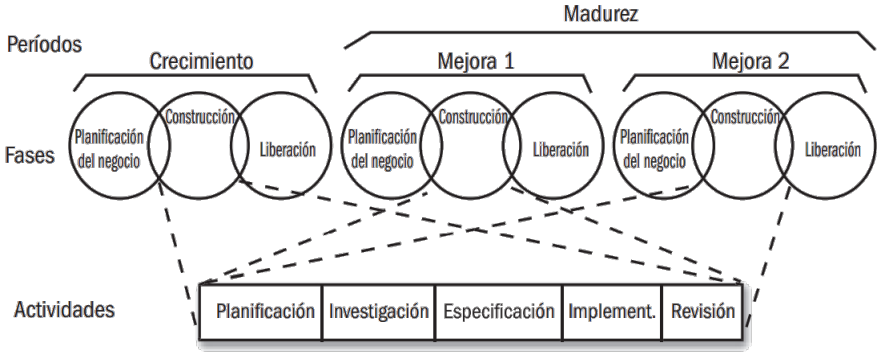

Figura 3.3: Ciclo de vida orientado a objetos

En este texto sólo veremos un tipo de ciclo de vida orientado a objetos, que es además el más representativo y el que será utilizado para nuestro proyecto: el modelo fuente.

3.2.1. El "Modelo Fuente"

El modelo fuente fue creado por Henderson-Sellers y Edwards en 1990. Es un tipo de ciclo de vida pensado para la orientación a objetos y posiblemente el más seguido.[16]

Presenta un alto grado de solapamiento/iteración entre fases. Cada clase/agrupamiento tiene un ciclo de vida propio. La "piscina software" (repositorio de clases) refleja reutilización: el ciclo de desarrollo "brota" de la piscina sw.

La ventaja de este modelo es que permite un desarrollo solapado e iterativo. En la figura siguiente se muestra un esquema de este tipo de ciclo de vida. [7]

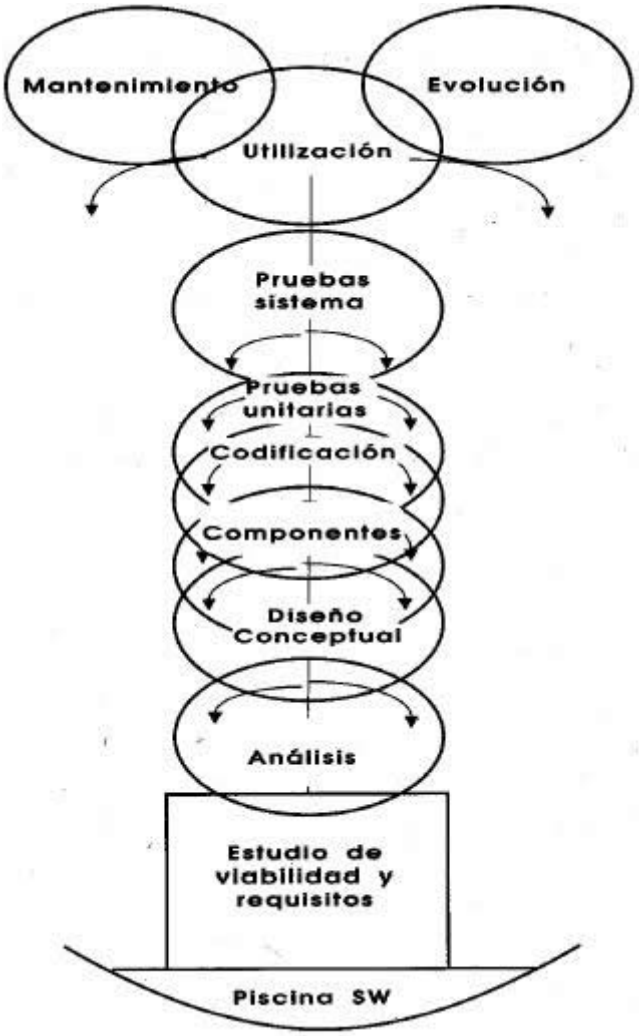

Figura 3.4: Modelo fuente

El modelo fuente nos ayudará a la hora de realizar un estudio preliminar de los requisitos, así como a la hora de realizar la implementación orientada a objetos que pretendemos hacer.

Además, como se ha mencionado en las características de la metodología orientada a objeto, esta técnica es la ideal para aplicaciones basadas en cliente-servidor, como va a ser la nuestra.

En relación a esto, haremos uso de una arquitectura de software denominada modelo-vista-controlador (MVC) y que nos permitirá separar los datos de la lógica de negocio de la aplicación. Esto se verá más detenidamente en el capítulo 6 "Diseño".

Capítulo 4

Estudio de oportunidad y viabilidad

4.1. Especificación de requisitos

El análisis de requisitos comprende todas las tareas relacionadas con la determinación de las necesidades o de las condiciones a satisfacer para un software nuevo o modificado, tomando en cuenta los diversos requisitos de las partes interesadas, que pueden entrar en conflicto entre ellos.

Llegados a a este punto de nuestro trabajo, las actividades que debemos desarrollar son de cinco clase:

- Obtener requisitos: a través de entrevistas o comunicación con clientes o futuros usuarios, para saber cuáles son sus expectativas.
- Analizar requisitos: detectar y corregir las carencias o falencias comunicativas, transformando los requisitos obtenidos de entrevistas y requisitos, en condiciones apropiadas para ser tratados en el diseño.
- Documentar requisitos: igual que todas las etapas, los requisitos deben estar debidamente documentados.
- Verificar los requisitos: consiste en comprobar la implementación de los requisitos.
- Validar los requisitos: comprobar que los requisitos implementados sean funcionales para lo que inicialmente se construyó el producto.

Por lo tanto abordaremos este capítulo mostrando los requisitos fundamentales de nuestra aplicación web, que han sido obtenidos mediante entrevistas con el cliente,

el Centro Universitario de Mérida, y serán documentados para su posterior análisis, validación, diseño e implementación.

4.1.1. Descripción textual

El Centro Universitario de Mérida tiene como objetivo desarrollar una plataforma que permita el diseño de interiores en 3D a través de la web. Los usuarios deben poder crear sus diseños personalizados partiendo de un plano de la habitación, cuyas paredes puedan ser editadas a voluntad para que se asemeje lo más posible a la realidad. Una vez esté definido el plano, la textura del suelo como de las paredes pueden ser cambiadas para obtener un diseño más atractivo y real. Ya tenemos nuestra habitación, pero está vacía, por lo tanto, el siguiente paso es añadir los objetos o piezas deseados. Para ello se contará con un banco de piezas que los usuarios puedan utilizar y modificar para ajustarlas a su diseño.

Con estas herramientas iniciales se puede crear un diseño personalizado. Los requisitos del cliente siempre están sujetos a cambios, por lo que es probable que a lo largo de este proyecto algunas de estas funcionalidades vayan cambiando.

Además es necesario contar con un sistema que permita la gestión de usuarios así como de todos los aspectos relacionados con la misma, como pueden ser los objetos que puedan ser añadidos a los diseños de los usuarios, las texturas que puedan ser utilizadas para los objetos, categorías y subcategorías para realizar una correcta gestión de los objetos, grupos de permisos para los usuarios que permita la utilización de ciertos objetos y de una gestión mínima de la web como es la eliminación de ficheros temporales o visualización de estadísticas.

Contaremos con dos zonas claramente diferenciadas, y que toda web posee: el *frontend*, que es la parte visible al usuario, en otras palabras, la parte donde el usuario creará su diseño y el *backend*, la parte de administración del sitio.

En diseño de software el front-end es la parte del software que interactúa con el o los usuarios y el back-end es la parte que procesa la entrada desde el front-end. La separación del sistema en front-ends y back-ends es un tipo de abstracción que ayuda a mantener las diferentes partes del sistema separadas. La idea general es que el front-end sea el responsable de recolectar los datos de entrada del usuario, que pueden ser

de muchas y variadas formas, y los transforma ajustándolos a las especificaciones que demanda el back-end para poder procesarlos, devolviendo generalmente una respuesta que el front-end recibe y expone al usuario de una forma entendible para éste.[46]

4.1.2. Requisitos del proyecto

A continuación pasamos a detallar los requisitos propuestos para nuestro proyecto:

1. **Registro y autenticación de usuarios:** creación y autenticación de usuarios.
   a) El registro de usuarios deberá hacerse mediante un formulario, introduciendo un correo electrónico y una contraseña. Una vez se haya rellenado y enviado dicho formulario se enviará automáticamente un email al usuario para que proceda a activar su cuenta. Si no se activa la cuenta, el usuario no podrá acceder a la plataforma.
      1) Alternativamente, un usuario podrá registrarse utilizando la red social Facebook.
   b) Además se permitirá restaurar una contraseña que haya sido olvidada.
2. **Diseño de interiores:** creación, modificación y eliminación de diseños.
   a) **Interfaz de diseño del plano:** ventana que permitirá la creación del plano para el diseño.
      1) Se puede editar el plano de la habitación como el usuario lo desee, creando las distribuciones necesarias para empezar el diseño. Para ello se permitirá dibujar muros, así como poder moverlos o eliminarlos. Se mostrará además su tamaño.
      2) Se permitirá tomar una captura del plano actual y poder descargarla como una imagen.
   b) **Interfaz de diseño:** ventana principal donde el usuario podrá crear su diseño.
      1) Las opciones principales de esta ventana serán:
         a′ Nuevo diseño: que permitirá la elaboración de un diseño nuevo.
         b′ Guardar diseño: que guardará el diseño actual en la base de datos.
         c′ Nombre del diseño: para poder darle un nombre al diseño.
         d′ Importar/Exportar: como su propio nombre indica, para poder importar y exportar diseños.

e' Menú de usuario: un pequeño menú de usuario que contendrá:

- Los diseños actuales del usuario que podrán ser cargados o eliminados del sistema.
- Las opciones por defecto del usuario.
- Posibilidad de cambiar la contraseña.
- Salir de la plataforma.

2) Se contará con una barra de herramienta que permita al usuario:

   a' Tomar una fotografía del diseño.

   b' Medir la distancia entre dos puntos.

   c' Activar/desactivar las colisiones entre objetos.

   d' Activar/desactivar la magnetización de objetos.

   e' Activar/desactivar la rotación del diseño (escena).

3) Se proporcionará una pantalla de ayuda al usuario indicando las funcionalidades principales de la interfaz.

4) Posibilidad de crear un presupuesto:

   a' Dicho presupuesto será creado a partir del diseño actual del usuario.

   b' Contendrá información sobre el usuario que lo crea (email, nombre, etc).

   c' Además se permitirá editar campos tales como el nombre de los objetos, descripción, cantidad o precio, para ajustarlo lo más posible a un presupuesto real.

   d' Posibilidad de exportarlo a PDF.

5) Botones de control:

   a' Para facilitar el desplazamiento por el diseño, se contará con diversos botones que hagan las funciones de mover o aumentar/disminuir el zoom de la escena.

   b' Además, se propondrán varios atajos del teclado con el fin de realizar las operaciones de una manera más rápida.

6) Propiedades de las piezas del diseño:

   a' Cada pieza tendrás sus propias propiedades tales como ancho, tamaño o profundidad, que podrán ser editadas según las necesidades del usuario.

b′ También se mostrará la posición dentro del plano, pudiendo igualmente ser editada.

c′ Se permitirá bloquear la pieza dentro del diseño para evitar que pueda ser movida por error.

d′ Se mostrará la distancia de la pieza con las paredes superior, inferior, izquierda y derecha, respectivamente de la pieza, pudiéndose, igualmente, editadas.

e′ Se permitirá elegir una textura para la pieza seleccionada.

f′ Además se contará con tres opciones principales que permitan eliminar, duplicar o mover la pieza.

7) Paredes y suelo:

a′ Posibilidad de elegir el color o textura del suelo y paredes.

c) **Agregación de piezas:** Ventana que permitirá al usuario agregar nuevas piezas al diseño.

1) Se permitirá agregar una pieza realizando una búsqueda por su nombre, descripción o identificador teniendo en cuenta que un usuario podrá utilizar varios grupos de pieza.

a′ Por grupo de pieza nos referimos a aquellos grupos a los que el usuario tendrá acceso. Este aspecto puede ser modificado a través del panel de administración.

2) Además se permitirá filtrar las piezas por categorías.

3. **Panel de administración:**

a) Posibilidad de gestionar:

1) **Piezas**: alta, baja y modificación de piezas. Para ello se hará uso de un formulario que permita:

a′ Especificar datos específicos de la pieza (nombre, descripción, dimensiones, precio, etc).

b′ Seleccionar su categoría.

c′ Seleccionar o agregar texturas que puede utilizar.

d′ Seleccionar grupos de usuario que pueden utilizarla.

e′ Reconocerla a través de su imagen de miniatura.

2) **Categorías**: alta, baja y modificación de categorías a través de su formulario correspondiente.

3) **Grupos**: alta, baja y modificación de grupos a través de su formulario correspondiente.
4) **Texturas**: alta, baja y modificación de texturas a través de su formulario correspondiente.
5) **Usuarios**: modificación de usuarios. Se permite modificar de un usuario:
   a′ La dirección de correo electrónico.
   b′ Su rol (si es usuario normal o administrador).
   c′ Los grupos de pieza a los que tendrá acceso.
   d′ Habilitarlo o deshabilitarlo temporalmente.

b) Mantenimiento del sitio, que incluye:

1) Ver la información del sitio, como el numero de usuarios registrados, la cantidad de piezas creadas en la base de datos, numero de categorías, etc.
2) Visualizar el registro (log) de operaciones realizadas.
3) Eliminar los ficheros temporales.
4) Habilitar o deshabilitar la plataforma, en el caso de que haya que realizar tareas de mantenimiento.
5) Habilitar o deshabilitar el registro de nuevos usuarios.
6) Responder a sugerencias o comentarios de los usuarios.

4. **Otros requisitos:**

a) Se proporcionará también un formulario para poder contactar con el administrador, en caso de querer enviar alguna sugerencia o pregunta. Dichas sugerencias podrán ser contestadas directamente desde el panel de administración.

b) Diseño responsive (adaptativo) para adaptarse a cualquier tipo de dispositivo.

c) Desarrollo de una interfaz intuitiva y agradable para el usuario.

4.2. Planificación y estimación de costes

4.2.1. Planificación temporal

La distribución temporal de los siete paquetes de trabajo puede verse en el diagrama de Gantt de la Figura 4.1. La planificación de las tareas se ha realizado en función a las etapas que podemos encontrar en modelo fuente y que como se ha comentado en capítulos anteriores, es el modelo a seguir.

A continuación se procederá a enumerar los paquetes junto con las tareas que los componen.

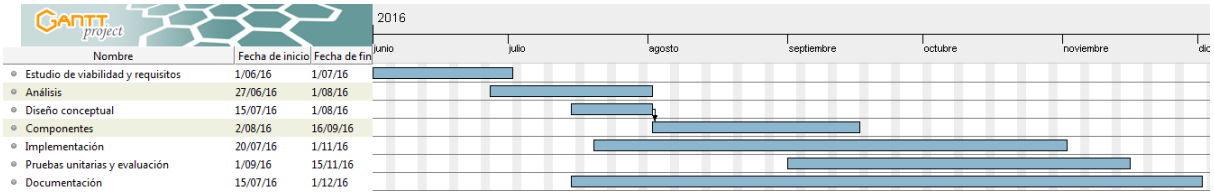

Figura 4.1: Diagrama de Gantt, con la planificación temporal del proyecto[35].

4.2.1.1. Tarea 1: Estudio de viabilidad y requisitos

El estudio de viabilidad es una tarea fundamental a la hora de realizar un proyecto. A grandes rasgos sabemos lo que queremos hacer, sin entrar en detalles de lo que quiere el cliente, queremos desarrollar una web para realizar diseños de interiores en 3D. Para ello el primer paso es conocer las tecnologías disponibles para desarrollar tal proyecto y además hacer una revisión de trabajos anteriores en los que pueda basarse el nuestro. Esto estaría comprendido dentro del capítulo "Estado del arte", donde se analizan las tecnologías 3D y se concreta la utilización de WebGL con la ayuda de Three.js, partiendo del proyecto Open-Source de Blueprint3D.

También entraremos a especificar lo más detalladamente posible cada uno de los requisitos que el cliente necesita. Para ello realizaremos varias entrevistas con nuestro cliente en las que intentaremos concretar todos o casi todos los requerimientos del proyecto. Sabemos que tener todos los requisitos al principio es una tarea muy complicada, por no decir imposible, pero cuantos más detalles tengamos, más rápidamente se podrá avanzar en el proyecto y menos cosas habrá que modificar ante futuros cambios.

4.2.1.2. Tarea 2: Análisis

En la tarea de análisis se realizarán los primeros diagramas donde se mostrarán las operaciones básicas a realizar por los usuarios de la web o plataforma, con el fin de representar visualmente las características principales.

4.2.1.3. Tarea 3: Diseño conceptual

Es hora de ponernos a diseñar. Ya tenemos todos (o la gran mayoría) de los requisitos, por lo que podemos diseñar nuestra aplicación. Para ello se realizarán los diagramas oportunos de manera que tengamos sobre el papel todos los detalles para que después, a la hora de implementar, no se nos escape nada. Además, estos diagramas ayudarán a visualizar de una forma más gráfica aquellos pequeños detalles que no se vean claramente a simple vista.

También comenzaremos a realizar el diseño de la página web, todo lo relacionado con el diseño gráfico como logos, imágenes, distribución de contenido, etc.

4.2.1.4. Tarea 4: Componentes

Los componentes necesarios para la realización del proyecto también necesitamos conocerlos. En nuestro caso, a la hora de realizar la planificación, se realizará una estimación de recursos, tanto del hardware como del software que será utilizado durante este proyecto.

Además de los componentes materiales, también necesitamos contar con un componente humano que serán los directores del proyecto y el desarrollador.

4.2.1.5. Tarea 5: Implementación

Se procede a la implementación de la aplicación. Es la tarea principal de nuestro proyecto, la que más tiempo nos llevará, y que debemos combinar con las anteriores para obtener un producto final de calidad. Se utilizarán los lenguajes y herramientas de desarrollo acordadas en tareas anteriores para dotar a nuestra plataforma de todas las funcionalidades solicitadas por el cliente.

4.2.1.6. Tarea 6: Pruebas unitarias y evaluación

A medida que se va implementando la herramienta plataforma web, se van realizando pruebas y evaluaciones de la misma para comprobar que su funcionamiento es

el correcto. Asimismo, una vez acabada la fase de implementación, se procederá a la comprobación y puesta a punto, así como la evaluación de la solución, contrastando su funcionamiento y resultados, y si procede, con otras soluciones vistas durante la revisión de proyectos anteriores.

4.2.1.7. Tarea 7: Documentación

La realización de una memoria de documentación que recoja los detalles de este proyecto se irá realizando de forma paralela a cada una de las tareas anteriores.

4.2.2. Recursos

4.2.2.1. Humanos

- Dr. Francisco Chávez de la O, profesor del departamento de Ingeniería de Sistemas Informáticos y Telemáticos del Centro Universitario de Mérida, en calidad de director del proyecto.
- Dr. Rafael Marcos Luque Baena, profesor del departamento de Ingeniería de Sistemas Informáticos y Telemáticos del Centro Universitario de Mérida, en calidad de co-director del proyecto.
- D. Daniel Flores Martín, alumno del Centro Universitario de Mérida, en calidad de autor del proyecto.

4.2.2.2. Hardware

- Ordenador personal, para el desarrollo de la aplicación.
- Raspberry Pi 2+, para hacer las funciones de servidor y gestión de base de datos.

4.2.2.3. Software y tecnologías

- Sistemas operativos:
  - Microsoft® [24] Windows 7, Windows 10
  - Raspbian 7 (*Wheezy*) [36].
- Desarrollo de la aplicación:
  - Programación del servidor:

    - Java 8 [30].
  - Diseño y programación web:
    - HTML5 [44]
    - CSS3 [43]
    - Bootstrap [6]
    - JavaScript [19].
    - Java Server Pages (JSP) [31].
  - Entorno de desarollo:
    - Eclipse IDE. Versión Mars y Neon [9].
  - Gestión del servidor y base de datos:
    - Apache Server [11].
    - Tomcat 8.0 [12].
    - MySQL 5.5.44-0+deb7u1 [29].
    - PHPMyAdmin 3.4.11.1deb2+deb7u1 [34].
  - Diseño en 3D:
    - WebGL [15].
    - Three.js [41].
  - Otro software utilizado:
    - Sublime Text[40].
    - Notepad++[17].
- Navegadores de Internet:
  - Mozilla Firefox[26]
  - Google Chrome[14]
- Modelado de piezas en 3D:
  - Blender v2.65 [5]
- Ofimática:
  - LyX 2.0.4 y MiKTeX 2.9 [22] .

Fase	Duración (días)
Revisión de proyectos anteriores	23
Investigación de tecnologías 3D en la web	26
Análisis de requisitos	12
Diseño	34
Implementación	75
Evaluación y Pruebas	54
TOTAL	224
DÍAS CON TAREAS SOLAPADAS	40
DURACIÓN FINAL DEL PROYECTO	184

Cuadro 4.1: Coste temporal del proyecto por tareas

4.2.3. Estimación de costes

4.2.3.1. Recursos humanos

En el Cuadro 4.1 puede verse el coste temporal de los diferentes paquetes de trabajo. En total se computa en días, teniendo en cuenta la jornada de 2-3 horas diarias, sin contar sábados, domingos y festivos. La documentación no aparece por ser una tarea totalmente paralela a las demás y de necesaria realización para cualquier proyecto que se proponga, por lo tanto, no es específica del nuestro.

4.2.3.2. Herramientas

Se han empleado para el diseño y desarrollo del proyecto herramientas gratuitas que garanticen el principal objetivo de nuestro proyecto, desarrollar la aplicación con software libre. El único software que no es libre sería Microsoft Windows, utilizado en el equipo principal, pero al venir este de fábrica en la fecha de su adquisición, no se considera un coste.

4.2.4. Presupuesto

En principio, los costes de realización del proyecto son nulos al ser llevado a cabo íntegramente con software libre.

Suponiendo que tuviéramos que llevar nuestro sistema a una empresa, tendríamos que tener en cuenta otro tipo de costes, como el precio de venta, el mantenimiento o la explotación. Por lo tanto en esta sección se pretende confeccionar un presupuesto para un hipotético caso en el que tuviéramos que montar la aplicación en una empresa.

4.2.4.1. Tipo de empresa

Tenemos que tener en cuenta si la aplicación va ir destinada a una empresa privada, o a otro tipo de entidad como una Universidad o centro de formación. Para ello se podría contar con varios tipos de licencia o plan, en función del destino de la aplicación.

- Para empresas privadas: adquisición de licencia por un valor de 1850€, y cuya duración sería indefinida.
- Para centros de formación: licencia por valor de 616,67€ y de duración indefinida.

El coste de ambas soluciones se ha calculado atendiendo a los siguientes factores:

- Total de horas de realización el proyecto (NHP): 370 horas, teniendo en cuenta los 184 días que se ha tardado en implementar el proyecto trabajando unas 2 horas diarias (detallado en la siguiente sección).
- Precio base (PB): 15€/hora para empresas privadas y 5€/hora para centros de formación.
- Suponemos además, que la aplicación va a ser adquirida por 3 entidades distintas, por lo que dividiremos el coste final entre 3, y así poder establecer un precio más competitivo. Este número podrá ir aumentando si la aplicación es comprada por muchas empresas, lo que abarataría su precio final.

Por lo tanto el precio de venta final (PF) sería:

$$\mathbf{PF = (NHP * PB) / 3}$$

- Empresas: PF = (370*15)/3 = 1850€.
- Centros de formación: PF = (370*5)/3 = 616,67€.

4.2.4.2. Servidor

Necesitaremos un servidor donde alojar la aplicación. Dicho servidor debe contar con los servicios Apache, MySQL y Tomcat, como mínimo. En este punto hay dos alternativas:

- Que la empresa cuente ya con algún servidor, que sería lo normal, y haya que instalar los servicios necesarios. Hoy en día, prácticamente cualquier empresa por pequeña que sea, suele contar con un servidor que gestione usuarios o proporcione servicios como correo electrónico o página web.
- Que la empresa prefiera contratar algún tipo de servicio en la nube (*cloud coumputing*) para alojar la aplicación porque no quiera sobrecargar sus servidores, o simplemente porque no los tienen.

En el primero de los casos, bastaría con instalar los servicios necesarios. En el segundo tendríamos que buscar proveedores que permitan desplegar aplicaciones web que necesiten de un servidor y conexión con base de datos. Algunos de los proveedores web más utilizados son:

- Google Cloud Computing.
- Amazon.
- IBM Bluemix.
- Open Shift.

Cada uno de ellos tiene sus ventajas e inconvenientes y cuentan con tarifas muy dispares, en función del plan contratado, y del tráfico de datos. Sería cuestión de llegar a un acuerdo con la empresa sobre cual sería el idóneo en relación calidad-precio, en relación a nuestras necesidades.

También habría que tener en cuenta que si la instalación y configuración, tanto de los servicios o de la plataforma, la hacemos nosotros, hay que facturar el tiempo empleado.

Siguiendo la política de precios bases, se tarificará con 15€/hora para empresas privadas, y con 5€/hora para centros de formación.

4.2.4.3. Nombre de dominio

Si la empresa cuenta ya con servicio de página web, solamente tendríamos que asociar la carpeta donde esté el proyecto a dicha página web. En otras palabras, partiendo de la empresa cuyo nombre de dominio es "http://www.miempresa.es", se ha desplegado la aplicación en la carpeta "InteriorDesign", por lo que bastaría con asociar dicha carpeta al servicio web (teniendo en cuenta la estructura de carpetas que tiene un servidor web), para que el resultado final fuera "http://www.miempresa.es/InteriorDesign". De esta manera ya tendríamos configurado un nombre de dominio a través del cuál pudieran acceder los usuarios. Esto tendría como ventaja el no tener que contratar un nombre de dominio nuevo.

En caso contrario de que la empresa no cuente con un nombre de dominio asociado, habría que contratar uno. En Internet podemos encontrar infinidad de proveedores que nos pueden ofrecer un dominio a precios muy competitivos. Estos precios variarán dependiendo si queremos un ".com", ".es", ".org", etc.

Según el proveedor Wesped.com (`http://www.wesped.com`), utilizado para la contratación de alojamiento y nombres de dominio en proyectos anteriores, contamos con la siguiente tabla de precios para la contratación de dominios de manera anual:

Tipo de dominio	Registro	Traslado	Renovación
.es	5.95€	GRATIS	6.95€
.com	7.99€	7.99€	7.99€
.org	8.99€	8.99€	8.99€
.net	7.99€	7.99€	7.99€

Cuadro 4.2: Tarifas de dominios

4.2.4.4. Otros gastos

A lo visto anteriormente, tendríamos que sumar los gastos de mantenimiento (PM) y explotación (PE). Una vez instalada la aplicación, hay que mantenerla y hacer las oportunas tareas de marketing que sean necesarias para publicitarla.

Para el mantenimiento vamos a suponer que cada mes habrá que llevar a cabo tareas que requieran de 10 horas en total (NHM). Por lo que el coste de mantenimiento mensual quedaría:

$$\mathbf{PM = (NHM * PB) / 3}$$

- Empresas: PM = (10*15)/3 = 50€/mes
- Centros de formación: PM = (10*5)/3 = 16.67€/mes

En cuanto a la explotación, queda a cargo de la empresa realizar el estudio de mercado correspondiente para publicitar la aplicación, ya que cada empresa tendrá sus propias estrategias y mecanismos para acercar el producto a los clientes.

4.2.4.5. Presupuesto final

Como resultado final después de estudiar cada uno de los apartados, podemos resumir:

	Empresa privada	Centro de formación
Precio base de compra	1850€	616.67€
Precio de mantenimiento mensual	50€	16.67
Precio de configuración/instalación	15€/hora	5€/hora
Alojamiento en la nube	Consultar diferentes soluciones	
Nombre de dominio	5.95€ ~ 8.99€	

Cuadro 4.3: Resumen de precios orientativo

Nuevamente se recuerda que el presente presupuesto ha sido calculado teniendo en cuenta que la aplicación va a ser comprada por 3 entidades diferentes como mínimo, lo que hace que su precio final sea más bajo y por lo tanto más asequible. Asimismo, el coste total del desarrollo de nuestra aplicación es mayor si no tenemos en cuenta dicho principio.

4.3. Oportunidad de mercado

Nuestro proyecto tiene por objetivo el desarrollo de una herramienta que permita a cualquier usuario crear de manera sencilla, una representación en 3D para el diseño de interiores. Además, otro de los objetivos es que el desarrollo de dicha herramienta sea realizado íntegramente con software libre (Open Source) y que sirva como plataforma para satisfacer las necesidades de nuestros usuarios.

En la actualidad existen ya distintas soluciones web que permitan realizar un diseño en 3D de cualquier cosa, ya sea desde un simple modelo de cualquier objeto,

hasta la representación de una casa o habitación, como es nuestro caso. Muchos de estos sitios utilizan software propietario, y aunque en un principio ofertan un plan gratuito para los usuarios con las herramientas más básicas para realizar los diseños, finalmente acaban sacando planes de pago que permiten la explotación completa de todas las herramientas del sitio. Al fin y al cabo, es un negocio, y como buen negocio debe ser rentable.

Por eso creemos que nuestra plataforma tiene algo diferente. No somos un negocio, y ni mucho menos pretendemos competir con grandes empresas o sitios webs dedicadas desde hace años a este mundo. Simplemente, queremos dar a conocer el potencial que el 3D puede llegar a tener en la web, y la gran cantidad de posibilidades que ofrece, y todo, sin coste alguno para los usuarios. Aquí es donde puede estar nuestra gran oportunidad.

Mirando de cerca muchos de esos sitios ya desarrollados, pretendemos dotar a nuestra plataforma de todas las funcionalidades posibles, y que proporcionen al usuario gran cantidad de herramientas disponibles a la hora de realizar un diseño. Por lo tanto, podemos concretar que el alcance de nuestro proyecto se resume en ofrecer una herramienta lo más completa posible para desarrollar el diseño de interiores en 3D de forma gratuita.

Es un proyecto ambicioso. El diseño en 3D es un tema complicado de desarrollar, pero a la vez cuenta con muchísima documentación en la que podemos apoyarnos, y, sobre todo, si te gusta como es nuestro caso, se hace divertido. Y no nos quedamos aquí. De nada vale un buen desarrollo en 3D sin una web que puedan visitar los usuarios. Por lo tanto, también se va a crear un sitio web que aparte de permitir el diseño en 3D de interiores, sea capaz de realizar un mantenimiento de usuarios, objetos del diseño, texturas, categorías de piezas, etc. todo mediante un sistema gestor de base de datos y un "backend" que facilite la administración de todos los componentes. Esto se explicará más detalladamente en posteriores capítulos.

Creemos que es un proyecto viable gracias a la gran documentación que podemos encontrar en Internet. Si bien, la parte de gestión de la web no será demasiado complicada, y la parte más compleja del proyecto irá dedicado a las tecnologías utilizadas para poder crear los diseños, su comprensión y utilización, que como ya se ha indicado, serán WebGL y Three.js.

Capítulo 5

Análisis

Para entender mejor el funcionamiento de la aplicación, haremos uso de distintos tipos de diagramas que nos ayudarán a comprender mejor cada uno de los aspectos más significativos.

5.1. Diagramas de casos de uso

Un diagrama de casos de uso actúa como foco en la descripción de los requisitos del usuario. En él se describen las relaciones entre los requisitos, los usuarios y los componentes principales. Atendiendo a los requisitos anteriores, el diagrama de casos de uso que representa las posibilidades iniciales de la aplicación es el siguiente:

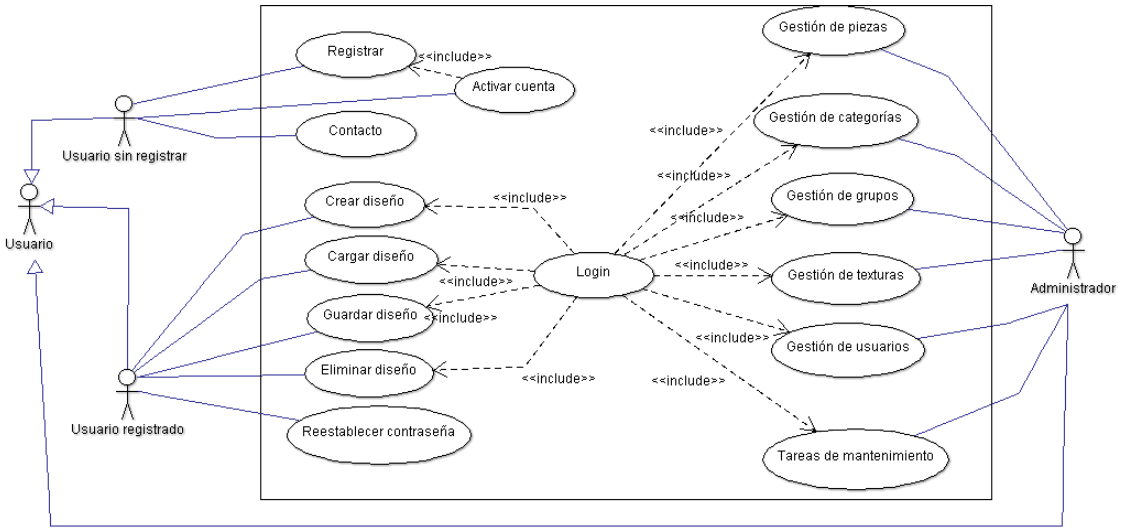

Figura 5.1: Diagrama de casos de uso

Partimos de que hay tres figuras (actores) claramente diferenciados que son:

- Usuario no registrado
- Usuario registrado
- Administrador.

5.1.1. Usuario no registrado

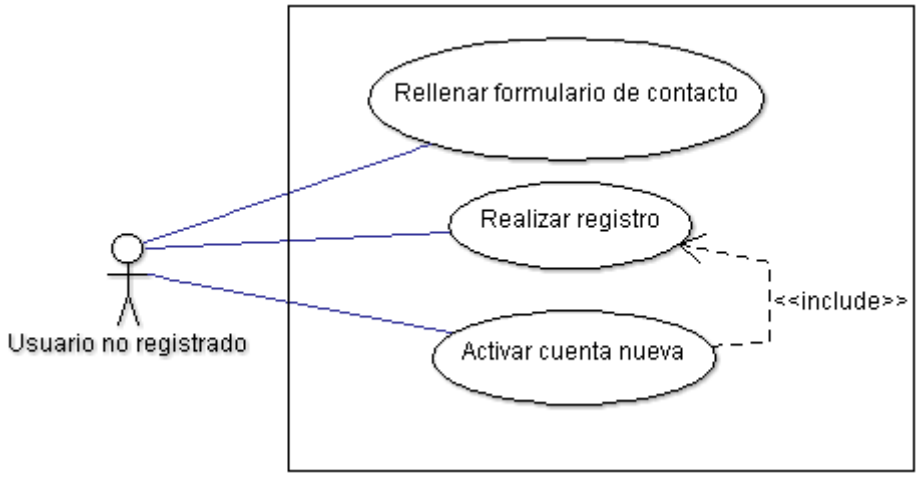

Figura 5.2: Casos de uso del usuario no registrado

5.1.2. Usuario registrado

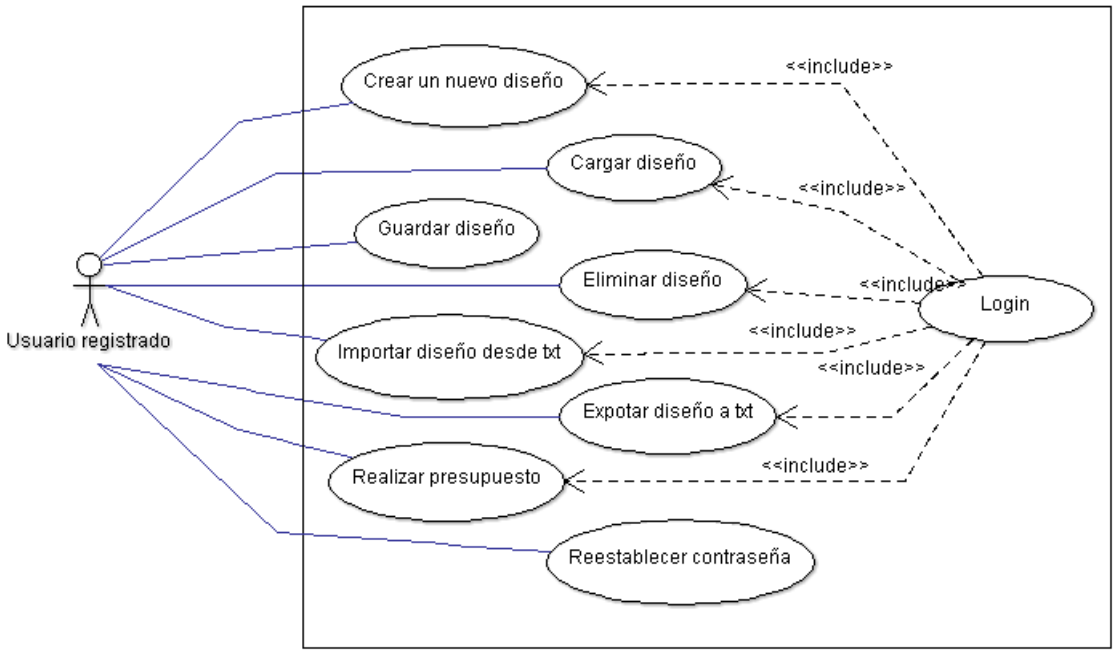

Figura 5.3: Casos de uso del usuario registrado

5.1.3. Administrador

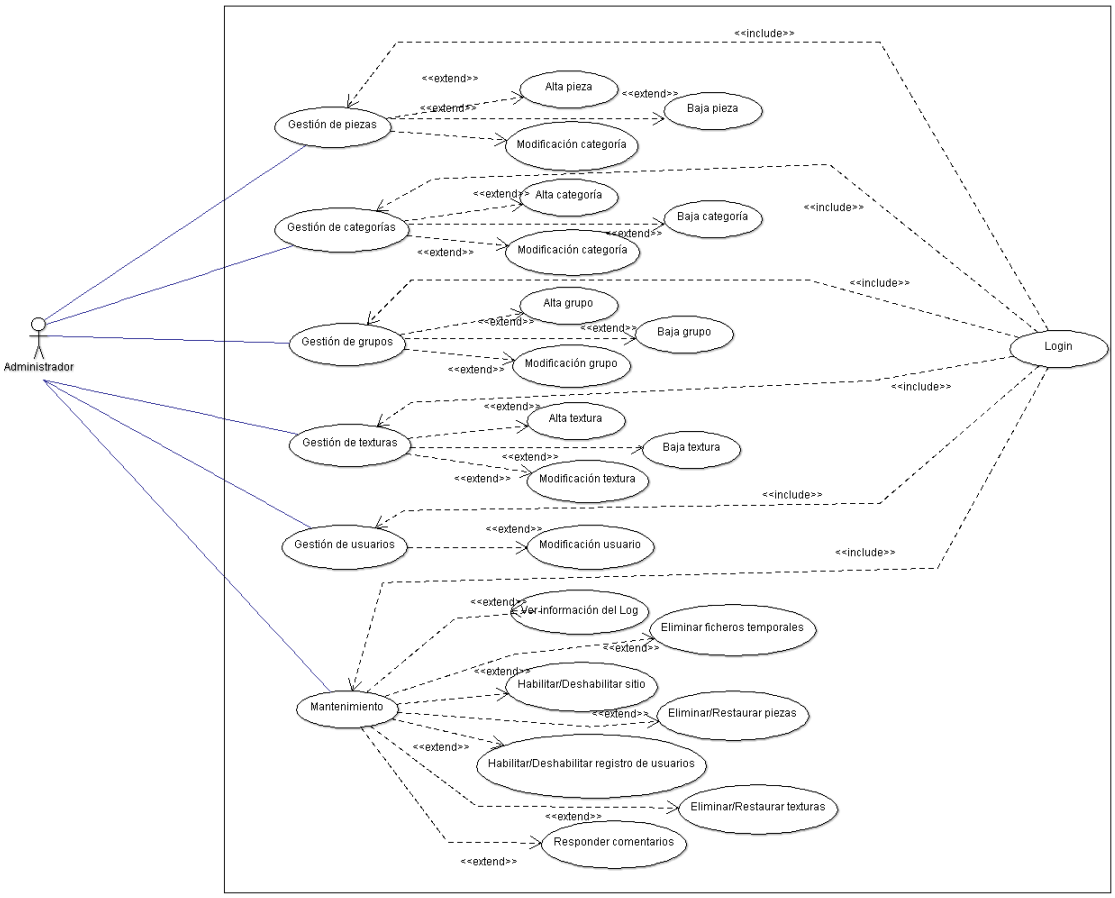

Figura 5.4: Casos de uso del administrador

5.2. Diagramas de caso de uso detallados

Los diagramas de caso de uso detallados nos ayudan identificar con más nivel de detalle las operaciones a realizar en la aplicación.

5.2.1. Registro de usuario

REGISTRO DE USUARIO	
Actores:	Usuario no registrado
Descripción:	Registro de un nuevo usuario en la plataforma. Un usuario registrado puede crear nuevos diseños y modificar los ya existente creados por él
Precondiciones:	1. El registro de usuarios debe estar habilitado 2. La web debe estar habilitada
Poscondiciones:	1. Activar la cuenta a través del correo electrónico si se ha registrado a través del formulario y no con Facebook

Flujo normal	Flujo alternativo
1. Rellenar formulario de registro con los siguientes datos: *a*) Correo electrónico *b*) Nombre *c*) Contraseña (por duplicado) 2. Pulsar botón ¡Dar de alta! 3. Grabar usuario en la base de datos 4. Mensaje de información	1. Realizar registro con Facebook *a*) Pulsar botón "Entrar con Facebook" 2. Dar los permisos solicitados a la aplicación a través de Facebok 3. Grabar usuario en la base de datos 4. Mensaje de información

Descripción	
Excepciones:	1. El usuario ya existe 2. Las contraseñas no coinciden
Prioridad:	Alta
Frecuencia de uso:	Una vez por usuario

Cuadro 5.1: Registro de usuario

5.2.2. Activar cuenta

ACTIVAR CUENTA		
Actores:	Usuario registrado	
Descripción:	Activación de una nueva cuenta	
Precondiciones:	1. Haber rellenado el formulario de registro correctamente *a*) Caso de uso "Registro de usuario"	
Poscondiciones:		
Flujo normal		**Flujo alternativo**
1. Entrar al correo electrónico personal con el que se realizó el registro 2. Clicar en el enlace proporcionado automáticamente por la plataforma para activar la cuenta 3. Mensaje de información		
Descripción		
Excepciones:	1. El mensaje enviado puede encontrarse en la carpeta span del gestor de correo electrónico	
Prioridad:	Alta	
Frecuencia de uso:	Una vez por usuario	

Cuadro 5.2: Activar cuenta

5.2.3. Login

LOGIN	
Actores:	Usuario registrado
Descripción:	Entrar a la plataforma web con un nombre de usuario y contraseña o perfil de Facebook
Precondiciones:	1. El sitio web debe estar habilitado 2. Se debe haber dado de alta una cuenta de usuario normal o de Facebook *a*) Caso de uso "Registro de usuario"
Poscondiciones:	

Flujo normal	Flujo alternativo
1. Rellenar formulario de login con los siguientes datos: *a*) Correo electrónico *b*) Contraseña 2. Pulsar sobre el botón de ¡Entrar! 3. Si los datos son correctos ▪ Redirección a web de diseño Si no ▪ Mensaje de error	1. Pulsar botón "Entra con Facebook" si el perfil des de Facebook 2. Redirección a web de diseño

Descripción	
Excepciones:	1. El usuario no existe 2. Los datos introducidos no son válidos
Prioridad:	Muy alta
Frecuencia de uso:	Muy alta

Cuadro 5.3: Login

5.2.4. Contactar con el administrador

CONTACTAR CON EL ADMINISTRADOR		
Actores:	Usuario no registrado	
Descripción:	Contactar con el administrador de la web a través del formulario para realizar alguna pregunta o sugerencia	
Precondiciones:	1. El sitio web debe estar habilitado	
Poscondiciones:		
Flujo normal		**Flujo alternativo**
1. Rellenar formulario de registro con los siguientes datos: *a*) Asunto *b*) Email *c*) Mensaje 2. Pulsar botón Enviar 3. Mensaje de información		
Descripción		
Excepciones:		
Prioridad:	Media	
Frecuencia de uso:	Media	

Cuadro 5.4: Contactar con el administrador

5.2.5. Crear diseño

CREAR DISEÑO	
Actores:	Usuario registrado
Descripción:	Crea un nuevo diseño personalizado en la web
Precondiciones:	1. El sitio web debe estar habilitado 2. El usuario debe estar registrado y con la cuenta activa *a*) Casos de uso "Registro de usuario y "Activar cuenta" *b*) Caso de uso Login
Poscondiciones:	

Flujo normal	Flujo alternativo
1. Empezar a crear el diseño *a*) Dar nombre al diseño *b*) Diseñar plano *c*) Agregar nuevos objetos *d*) Cambiar texturas 2. Guardar diseño 3. Mensajde de información	

Descripción	
Excepciones:	
Prioridad:	Muy alta
Frecuencia de uso:	Muy alta

Cuadro 5.5: Crear diseño

5.2.6. Modificar diseño

MODIFICAR DISEÑO	
Actores:	Usuario registrado
Descripción:	Modifica un diseño creado previamente
Precondiciones:	1. El sitio web debe estar habilitado 2. El usuario debe haber creado un diseño previamente y haberlo guardado *a*) Caso de uso "Crear diseño"
Poscondiciones:	

Flujo normal	Flujo alternativo
1. Pulsar sobre el menu de usuario "Bienvenido,@Usuario" *a*) Seleccionar "Mis diseños" 2. Seleccionar un diseño existente y pulsar "Cargar" 3. Empezar a modificar el diseño *a*) Dar nombre al diseño *b*) Diseñar plano *c*) Agregar nuevas piezas *d*) Cambiar texturas *e*) Etc... 4. Guardar diseño 5. Mensaje de información	

Descripción	
Excepciones:	1. El usuario no tiene diseños que cargar
Prioridad:	Muy alta
Frecuencia de uso:	Muy alta

Cuadro 5.6: Modificar diseño

5.2.7. Gestión de piezas

GESTIÓN DE PIEZAS	
Actores:	Administrador
Descripción:	Realiza la gestión (alta, baja, modificación y consulta) de las piezas (objetos)
Precondiciones:	1. Caso de uso "Login" a) El usuario debe ser de tipo "admin" (administrador)
Poscondiciones:	1. Esperar caso de uso "Alta pieza", "Baja pieza" o "Modificación pieza"

Flujo normal	Flujo alternativo
1. Seleccionar en el menú superior del panel de administración "Piezas"	

Descripción	
Excepciones:	
Prioridad:	Alta
Frecuencia de uso:	Alta

Cuadro 5.7: Gestión de piezas

5.2.8. Alta pieza

ALTA PIEZA	
Actores:	Administrador
Descripción:	Añade una pieza nueva a la plataforma para que los usuarios puedan usarla en sus diseños
Precondiciones:	Estar logueado como administrador y haber entrado a la sección de "Piezas" 1. Caso de uso 'Gestión de piezas'
Poscondiciones:	

Flujo normal	Flujo alternativo
1. Completar el formulario *a*) Destino de la pieza: tipo de pieza (suelo, pared, etc) *b*) Fichero.js: fichero de la pieza en formato JS *c*) Textura principal: indica la textura principal *d*) Id: indica el id de la pieza. *e*) Nombre: nombre de la pieza *f*) Descripción: descripción de la pieza *g*) Proporciones: ancho, alto y profundidad *h*) Categoría: categoría de la pieza *i*) Precio: precio de la pieza (€) *j*) Selección de grupos: grupos asociados *k*) Selección de texturas: texturas asociadas *l*) Miniatura: miniatura para su previsualización 2. Pulsar botón "Dar de alta"	

Descripción	
Excepciones:	1. Los datos introducidos no son válidos
Prioridad:	Muy alta
Frecuencia de uso:	Alta

Cuadro 5.8: Alta de piezas

5.2.9. Modificar pieza

MODIFICAR PIEZA		
Actores:	Administrador	
Descripción:	Modifica una pieza ya existente	
Precondiciones:	Estar logueado como administrador y que se haya creado una pieza 1. Caso de uso 'Alta pieza'	
Poscondiciones:		
Flujo normal		**Flujo alternativo**
1. Seleccionar pieza existente 2. Seleccionar pieza a editar 3. Modificar datos del formulario *a*) Destino de la pieza: tipo de pieza (suelo, pared, etc) *b*) Fichero.js: fichero de la pieza en formato JS *c*) Textura principal: indica la textura principal *d*) Id: indica el id de la pieza. *e*) Nombre: nombre de la pieza *f*) Descripción: descripción de la pieza *g*) Proporciones: ancho, alto y profundidad *h*) Categoría: categoría de la pieza *i*) Precio: precio de la pieza (€) *j*) Selección de grupos: grupos asociados *k*) Selección de texturas: texturas asociadas *l*) Miniatura: miniatura para su previsualización 4. Pulsar botón "Guardar"		
Descripción		
Excepciones:	1. Los datos introducidos no son válidos	
Prioridad:	Muy alta	
Frecuencia de uso:	Alta	

Cuadro 5.9: Modificación de piezas

5.2.10. Eliminar pieza

ELIMINAR PIEZA	
Actores:	Administrador
Descripción:	Elimina una pieza existente
Precondiciones:	1. Estar logueado como administrador y que se haya creado una pieza a) Caso de uso "Alta pieza"
Poscondiciones:	

Flujo normal	Flujo alternativo
1. Seleccionar pieza existente a) Menu "Piezas" -> "Editar/Eliminar" 2. Seleccionar pieza a eliminar 3. Pulsar botón "Eliminar"	

Descripción	
Excepciones:	1. La pieza no se ha encontrado en la base de datos
Prioridad:	Alta
Frecuencia de uso:	Media

Cuadro 5.10: Eliminar pieza

5.2.11. Eliminar ficheros temporales

ELIMINAR FICHEROS TEMPORALES	
Actores:	Administrador
Descripción:	Elimina los ficheros temporales utilizados por las piezas y texturas a la hora de realizar altas o bajas de piezas
Precondiciones:	1. Caso de uso "Login" a) El usuario debe ser de tipo "admin"
Poscondiciones:	

Flujo normal	Flujo alternativo
1. Seleccionar en el menú superior "Mantenimiento" 2. Seleccionar opción "Eliminar ficheros temporales"	

Descripción	
Excepciones:	1. Si no hay ficheros temporales, no hay nada que eliminar
Prioridad:	Media
Frecuencia de uso:	Baja

Cuadro 5.11: Eliminar ficheros temporales

5.2.12. Purgar base de datos

PURGAR BASE DE DATOS	
Actores:	Administrador
Descripción:	Realiza la eliminación completa de la base de datos de una pieza o textura que haya sido eliminada (deshabilitada) previamente
Precondiciones:	1. Caso de uso "Login" a) El usuario debe ser de tipo "admin"
Poscondiciones:	

Flujo normal	Flujo alternativo
1. Seleccionar en el menú superior "Mantenimiento" 2. Seleccionar las piezas o texturas a eliminar definitivamente	

Descripción	
Excepciones:	1. Si no hay piezas o texturas deshabilitadas, no hay nada que eliminar
Prioridad:	Media
Frecuencia de uso:	Baja

Cuadro 5.12: Purgar base de datos

5.3. Diagramas de actividades

Un diagrama de actividades muestra un proceso de negocio o un proceso de software como un flujo de trabajo a través de una serie de acciones. Las personas, los componentes de software o los equipos pueden realizar estas acciones.

Es importante recalcar que aunque un diagrama de actividades es muy similar en definición a un diagrama de flujo (típicamente asociado en el diseño de Software), estos no son lo mismo. Un diagrama de actividades es utilizado en conjunción de un diagrama de caso de uso para auxiliar a los miembros del equipo de desarrollo a entender como es utilizado el sistema y como reacciona en determinados eventos, mientras que un diagrama de flujo ayuda a un programador a desarrollar el código a través de una descripción lógica de un proceso.

Podemos considerar entonces que un diagrama de actividades describe el problema, mientras un diagrama de flujo describe la solución.

A continuación mostramos los diagramas de actividades por cada tipo de usuario que hemos analizado anteriormente en los casos de uso:

5.3.1. Usuario no registrado

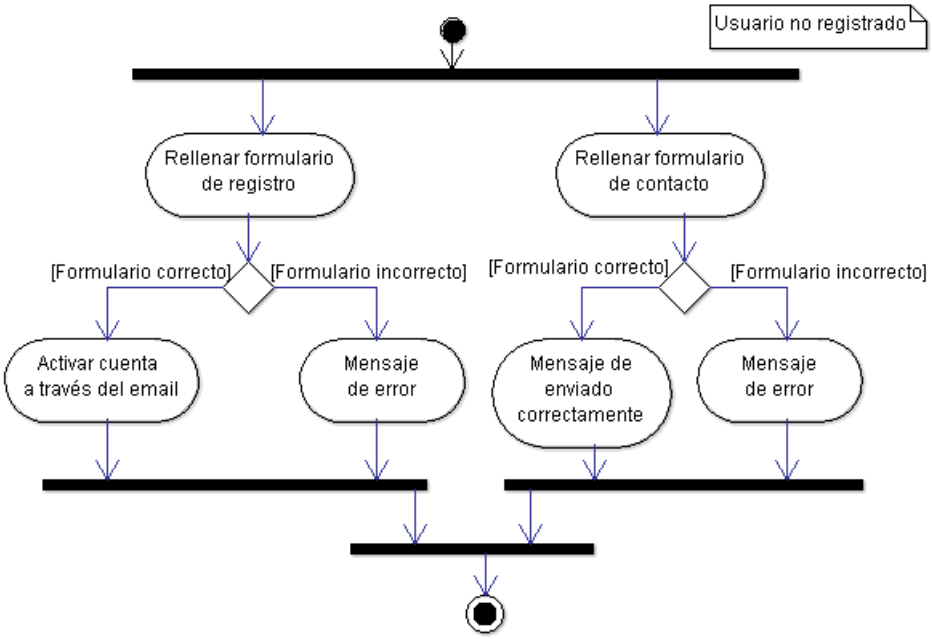

Figura 5.5: Actividades del usuario no registrado

5.3.2. Usuario registrado

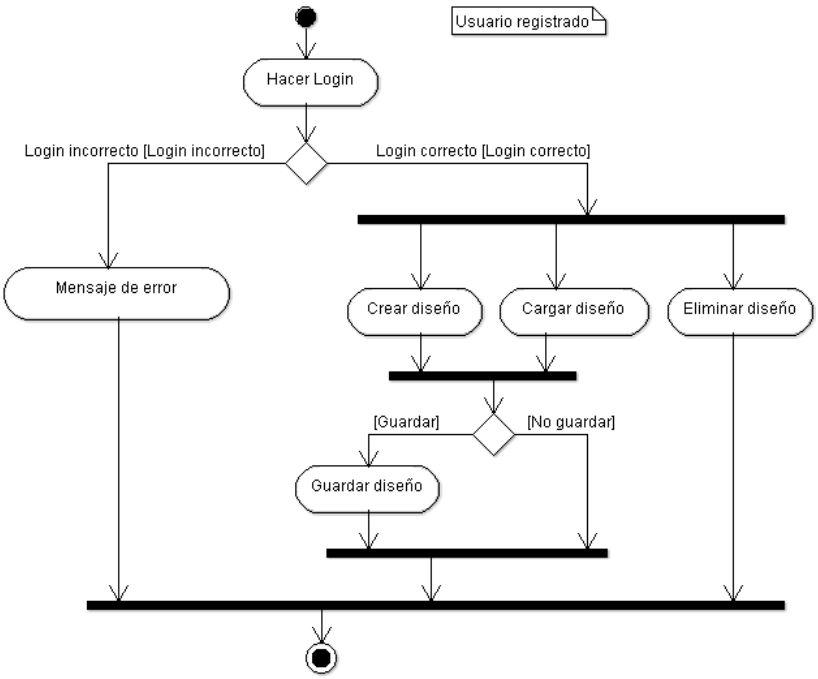

Figura 5.6: Actividades del usuario registrado

5.3.3. Administrador

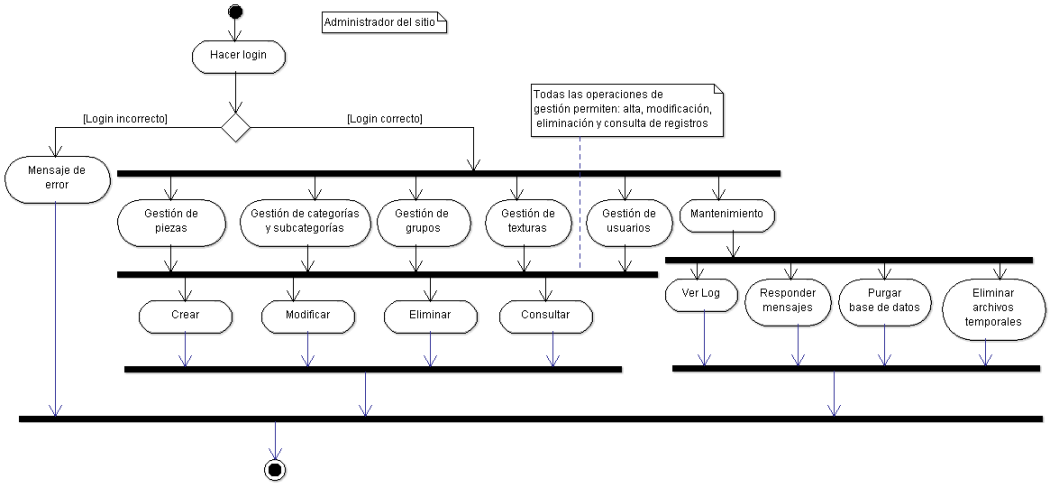

Figura 5.7: Diagrama de actividades del administrador

5.4. Diagrama entidad-relación

Un diagrama o modelo entidad-relación es una herramienta utilizada para el modelado de datos que permite representar las entidades relevantes de un sistema de información así como sus interrelaciones y propiedades.

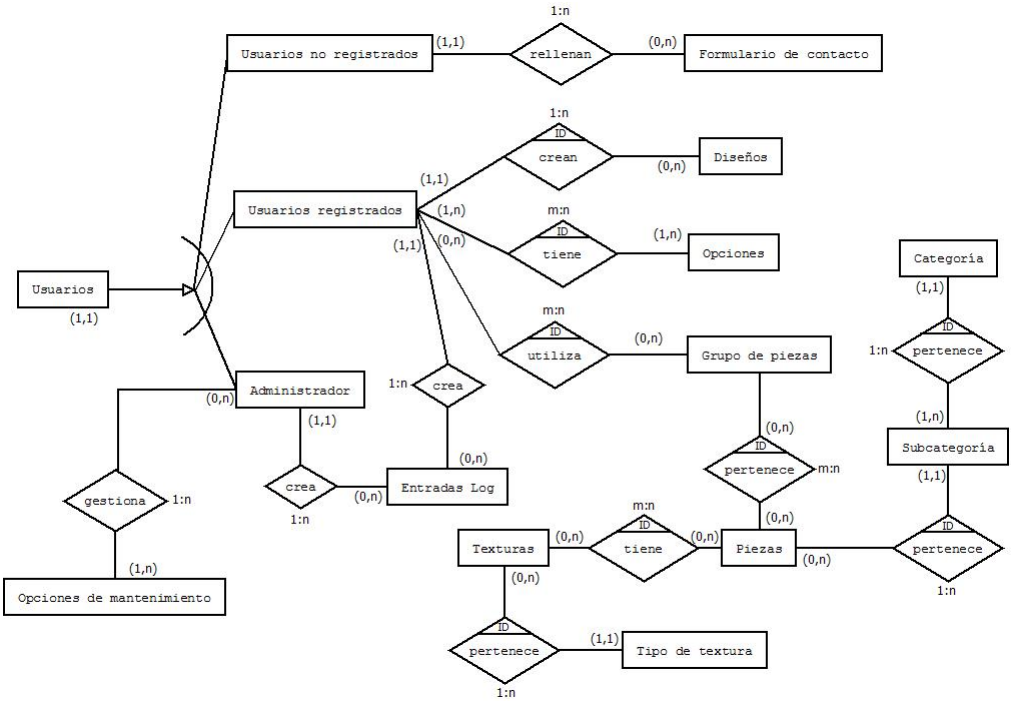

Figura 5.8: Diagrama entidad-relación (E-R)

Capítulo 6

Diseño

El diseño de la plataforma se dividirá en dos pasos: en primer lugar y con un mayor nivel de abstracción, se procederá al diseño de la arquitectura de forma que quede definido el comportamiento global del sistema; en segundo lugar, se profundizará en su estructura a través de los diagramas de clases en el apartado de diseño de los datos.

6.1. Diseño de la arquitectura

Seguiremos el tipo de arquitectura cliente-servidor. Podemos definir la arquitectura cliente-servidor como [38]:

- Desde un punto de vista conceptual:
  - «Es un modelo para construir sistemas de información, que se sustenta en la idea de repartir el tratamiento de la información y los datos por todo el sistema informático, permitiendo mejorar el rendimiento del sistema global de información»
- En términos de arquitectura:
  - «Los distintos aspectos que caracterizan a una aplicación (proceso, almacenamiento, control y operaciones de entrada y salida de datos) en el sentido más amplio, están situados en más de un computador, los cuales se encuentran interconectados mediante una red de comunicaciones».
- Cliente/Servidor
  - «Es la tecnología que proporciona al usuario final el acceso transparente a las aplicaciones, datos, servicios de cómputo o cualquier otro recurso

del grupo de trabajo y/o, a través de la organización, en múltiples plataformas. El modelo soporta un medio ambiente distribuido en el cual los requerimientos de servicio hechos por estaciones de trabajo inteligentes o "clientes", resultan en un trabajo realizado por otros computadores llamados servidores".

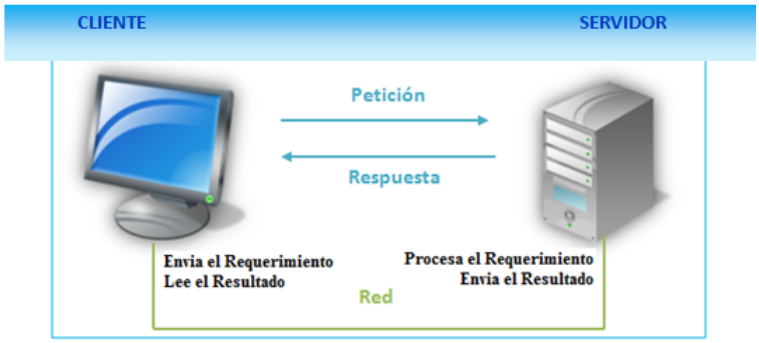

Figura 6.1: Arquitectura Cliente-Servidor

6.1.1. Servidor

El servidor es cualquier recurso de cómputo dedicado a responder a los requerimientos del cliente. Los servidores pueden estar conectados a los clientes a través de redes LANs o WANs, para proveer de múltiples servicios a los clientes y ciudadanos tales como impresión, acceso a bases de datos, fax, procesamiento de imágenes, etc.

En nuestro caso el servidor será el encargado de proporcionar los servicios web, como realizar login y registro de usuario, administrar los diseños o toda la parte de administración, entre otros.

Nuestra aplicación estará basada en el patrón Modelo-Vista-Controlador. A continuación pasaremos a detallar qué supone utilizar esta tecnología.

6.1.1.1. Modelo-Vista-Controlador (MVC)

El patrón de arquitectura MVC (Modelo-Vista-Controlador) es un patrón que define la organización independiente del Modelo (Objetos de Negocio), la Vista (interfaz con el usuario u otro sistema) y el Controlador (controlador del workflow de la aplicación).[42]

De esta forma, dividimos el sistema en tres capas donde tenemos la encapsulación de los datos, la interfaz o vista por otro y por último la lógica interna o controlador.

El patrón de arquitectura "modelo vista controlador", es una filosofía de diseño de aplicaciones, compuesta por:

- **Modelo**:
  - Contiene el núcleo de la funcionalidad (dominio) de la aplicación.
  - Encapsula el estado de la aplicación.
  - No sabe nada / independiente del Controlador y la Vista.
- **Vista**:
  - Es la presentación del Modelo.
  - Puede acceder al Modelo pero nunca cambiar su estado.
  - Puede ser notificada cuando hay un cambio de estado en el Modelo.
- **Controlador**:
  - Reacciona a la petición del Cliente, ejecutando la acción adecuada y creando el modelo pertinente

La siguiente figura muestra un esquema básico del MVC.

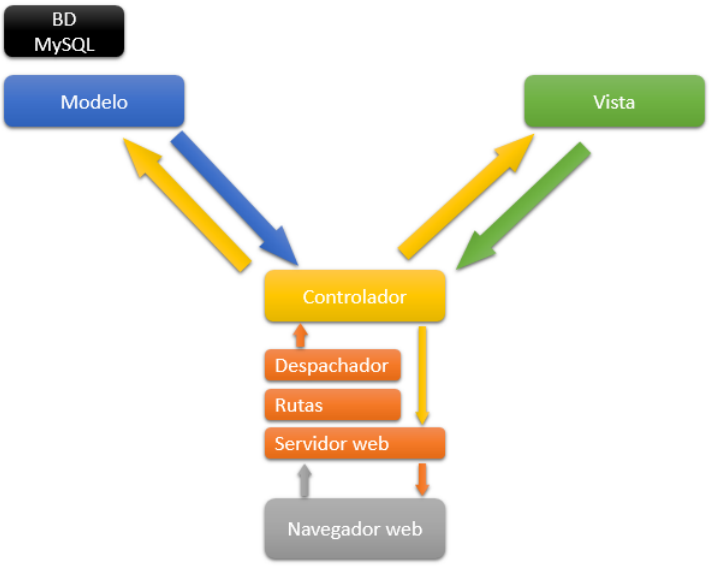

Figura 6.2: Modelo-Vista-Controlador (MVC)

Para entender cómo funciona nuestro patrón Modelo vista controlador, se debe entender la división a través del conjunto de estos tres elementos y como estos componentes se comunican unos con los otros y con otras vistas y controladores externos a el modelo principal. Para ello, es importante saber que el controlador interpreta las entradas del usuario (tanto teclado como el ratón), enviado el mensaje de acción al modelo y a la vista para que se proceda con los cambios que se consideren adecuados. Veamos paso a paso cómo sería el flujo de trabajo característico en un esquema MVC.

1. El usuario realiza una solicitud a nuestro sitio web. Generalmente estará desencadenada por acceder a una página de nuestro sitio. Esa solicitud le llega al controlador.

2. El controlador comunica tanto con modelos como con vistas. A los modelos les solicita datos o les manda realizar actualizaciones de los datos. A las vistas les solicita la salida correspondiente, una vez se hayan realizado las operaciones pertinentes según la lógica del negocio.

3. Para producir la salida, en ocasiones las vistas pueden solicitar más información a los modelos. En ocasiones, el controlador será el responsable de solicitar todos los datos a los modelos y de enviarlos a las vistas, haciendo de puente entre unos y otros. Sería corriente tanto una cosa como la otra, todo depende de nuestra implementación; por eso esa flecha la hemos coloreado de otro color.

4. Las vistas envían al usuario la salida. Aunque en ocasiones esa salida puede ir de vuelta al controlador y sería éste el que hace el envío al cliente, por eso he puesto la flecha en otro color.

Comunicación

El modelo, la vista y el controlador deben comunicarse de una manera estable los unos con los otros, de manera que sea coherente con las iteraciones que el usuario realizara. Como es lógico la comunicación entre la vista y el controlador es bastante básica pues están diseñados para operar juntos, pero los modelos se comunican de una manera diferente, un poco más sutil

Modelo pasivo

No es necesario para el modelo hacer ninguna tener alguna disposición a él, simplemente basta con tener en cuenta su existencia. El modelo no tiene ninguna responsabilidad para comunicar los cambios a la vista porque ocurren solo por orden del

usuario, por lo que esta función la llevara a cabo el controlador porque será el que interprete las ordenes de este usuario debido a que solo debe comunicar que algo ha cambiado. Por esto, el modelo es se encuentra en modo inconsciente y su participación en este caso es irrisoria.

Unión del modelo con la vista y el controlador

Como no todos los modelos pueden ser pasivos, necesitamos algo que comunique al controlador y a la vista, por lo que en este caso, si que necesitamos el modelo, ya que solo este puede llevar a cabo los cambios necesarios al estado actual en el que estos se encuentran.

Al contrario que el modelo, que puede ser asociado a múltiples asociaciones con otras vistas y controladores, cada vista solo puede ser asociada a un único controlador, por lo que han de tener una variable de tipo controler que notificara a la vista cual es su controlador o modelo asignado. De igual manera, el controlador tiene una variable llamada View que apunta a la vista. De esta manera, pueden enviarse mensajes directos el uno al otro y al mismo tiempo, a su modelo.

Al final, la vista es quien lleva la responsabilidad de establecer la comunicación entre los elementos de nuestro patrón MVC. Cuando la vista recibe un mensaje que concierne al modelo o al controlador, lo deja registrado como el modelo con el cual se comunicara y apunta con la variable controller al controlador asignado, enviándole al mismo su identificación para que el controlador establezca en su variable view el identificador de la vista y así puedan operar conjuntamente. El responsable de deshacer estas conexiones, seguirá siendo la vista, quitándose a si misma como dependiente del modelo y liberando al controlador.

6.1.1.2. Estructura

Nuestro servidor de aplicaciones estará montado en una Raspberry Pi 2+, con el sistema operativo Raspbian 7 (Wheezy). Además contaremos con la ayuda de Apache Tomcat 8.0, para desplegar la aplicación, y MySQL 5.5.44 para gestionar la base de datos.

6.1.2. Cliente

El cleinte es el que inicia un requerimiento de servicio. El requerimiento inicial puede convertirse en múltiples requerimientos de trabajo a través de redes LAN o WAN. La ubicación de los datos o de las aplicaciones es totalmente transparente para el cliente.

En nuestro caso el cliente será cualquier usuario que acceda a la plataforma desde un navegador web (Chrome, Firefox, Safari, etc.) y realice las peticiones que crea oportunas para que el servidor las resuleva.

6.2. Diseño de los datos

Dividiremos la aplicación por paquetes, siguiendo la lógica MVC y añadiremos paquetes que englobarán funcionalidades adicionales como cifrado de contraseñas, conexión con Facebook o reportes:

- **Controller**: albergará los servlets encargado de recibir las peticiones de los usuarios y dar una respuesta.
- **Model**: contendrá las clases base que serán utilizadas para realizar las operaciones.
- **Service**: reunirá las clases encargadas de realizar las operaciones con la base de datos.
- **Report**: clases destinadas a la generación de reportes, en nuestro caso para realizar los presupuestos.
- **Util**: clases que proporcionen funcionalidades adicionales estarán englobadas en este paquete.

En la figura siguiente podemos ver la dependencia entre los paquetes detallados anteriormente:

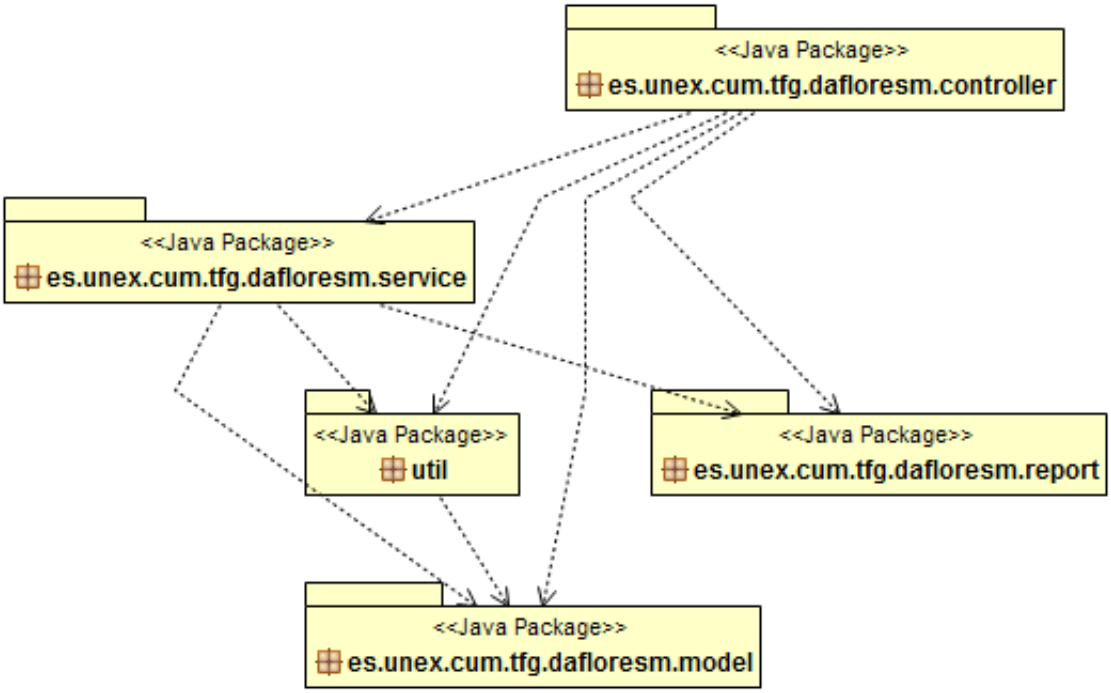

Figura 6.3: Dependencia de paquetes

A continuación pasaremos a analizar las clases que irán dentro de cada paquete junto con su correspondiente diagrama.

6.2.1. Controller

6.2.1.1. Action

Servlet principal que recogerá la petición del usuario y redirigirá al servlet oportuno para realizar las operaciones.

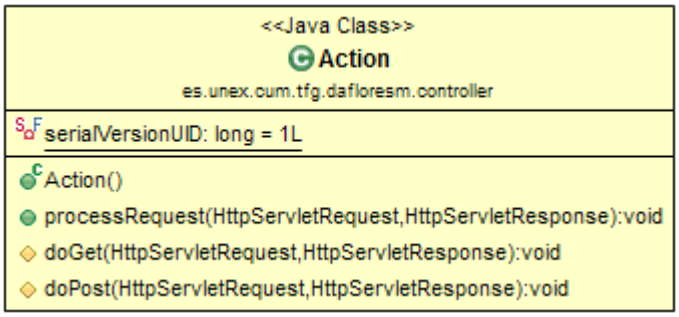

Figura 6.4: Action

6.2.1.2. CategoriaController

Servlet dedicado a la gestión de todo lo que tenga que ver con las categorías de las piezas, ya sea alta, modificación, bajas.

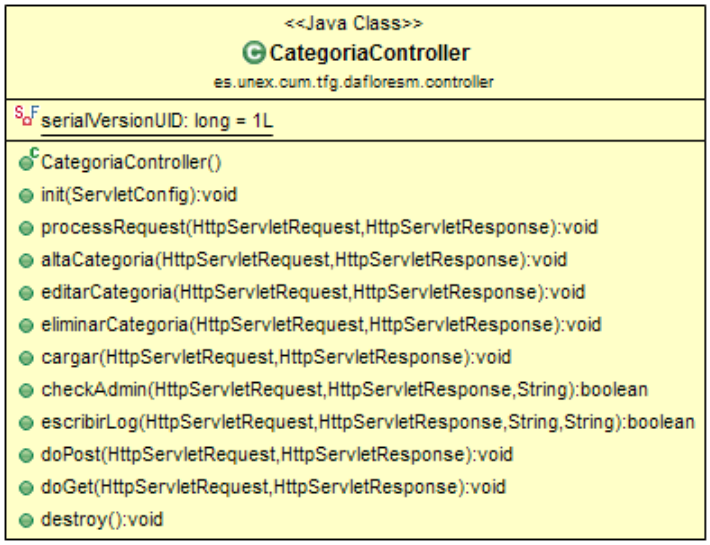

Figura 6.5: CategoriaController

6.2.1.3. ComentarioController

Todo lo relacionado con la gestión de comentarios que envíen los usuario a través del formulario, o las operaciones de administración de los mismos, serán gestionado por este servlet.

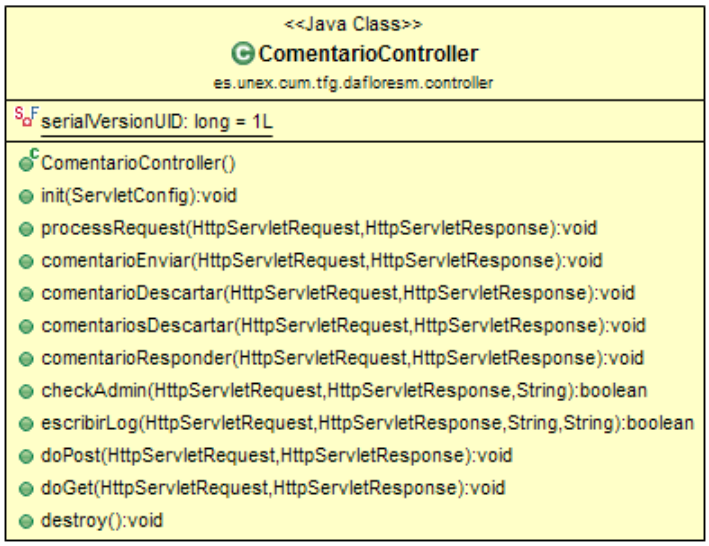

Figura 6.6: ComentarioController

6.2.1.4. DisenioController

El servlet destinado a la gestión de los diseños, será el encargado de cargar los diseños de los usuarios. Las operaciones para guardar o eliminar los diseños se harán dinámicamente con la ayuda de JSP y AJAX en la propia web del diseño.

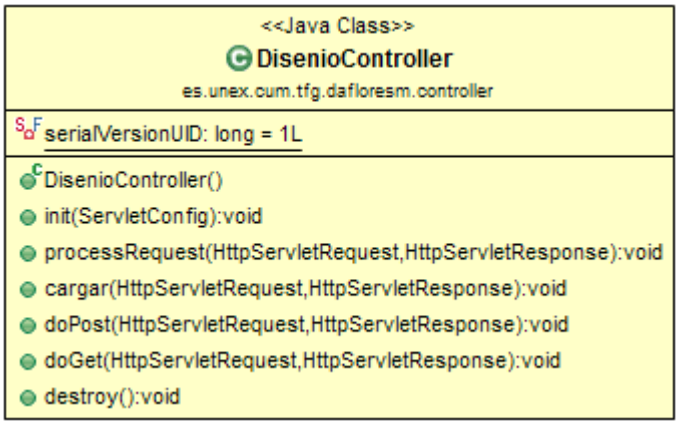

Figura 6.7: DisenioController

6.2.1.5. FacebookController

Configuración de los datos relativos a la aplicación de Facebook que permitirá registrarse y acceder a la aplicación utilizando la red social.

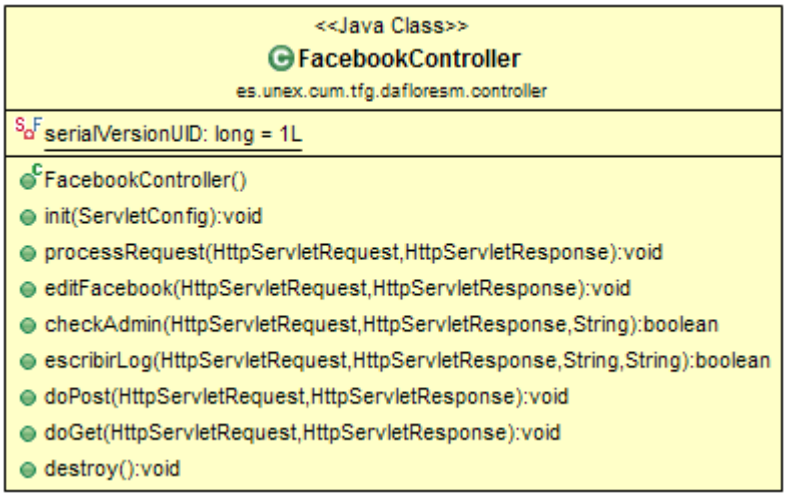

Figura 6.8: FacebookController

6.2.1.6. GrupoController

Administración de los grupos a los que puede pertenecer un usuario. Todo lo relacionado con la gestión de grupos, creación, eliminación o modificación, estará

recogido en este servlet.

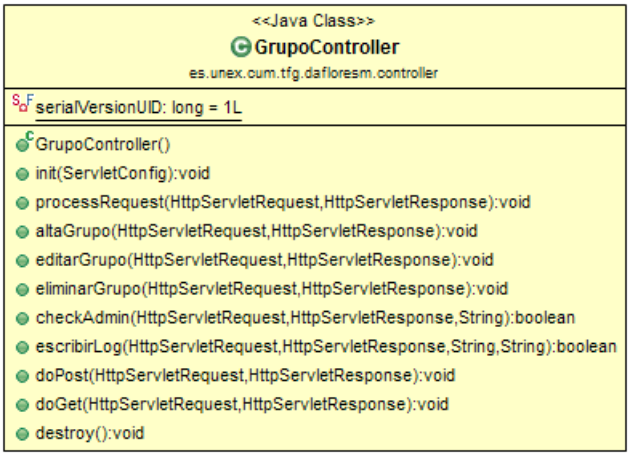

Figura 6.9: GrupoController

6.2.1.7. MantenimientoController

Las tareas administrativas también tienen un servlet dedicado para tal función. Aquí se podrán realizar operaciones tales como limpiar el log, cambiar el estado de la plataforma por si hay que realizar tareas de mantenimiento o limpiar los ficheros temporales.

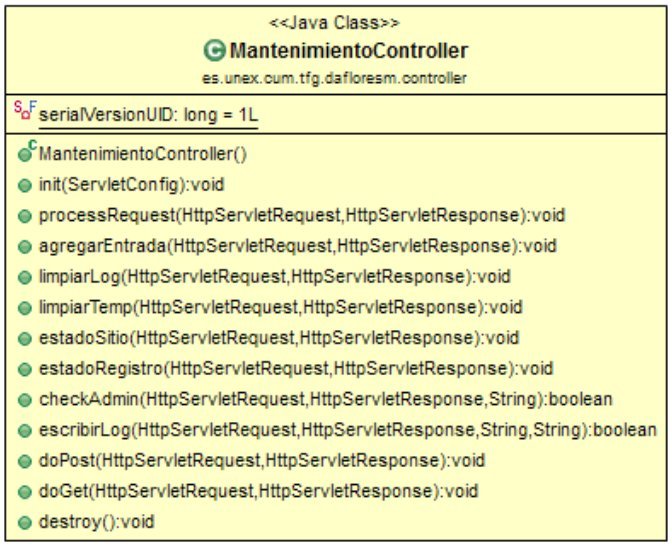

Figura 6.10: MantenimientoController

6.2.1.8. PiezaController

La administración de las piezas será llevada a cabo por este servlet. Permitirá las operaciones básicas que estamos viendo en los anteriores, como la creación, baja y modificación de registros, además de servir de ayuda en el panel de administración a la hora de seleccionar piezas para su gestión.

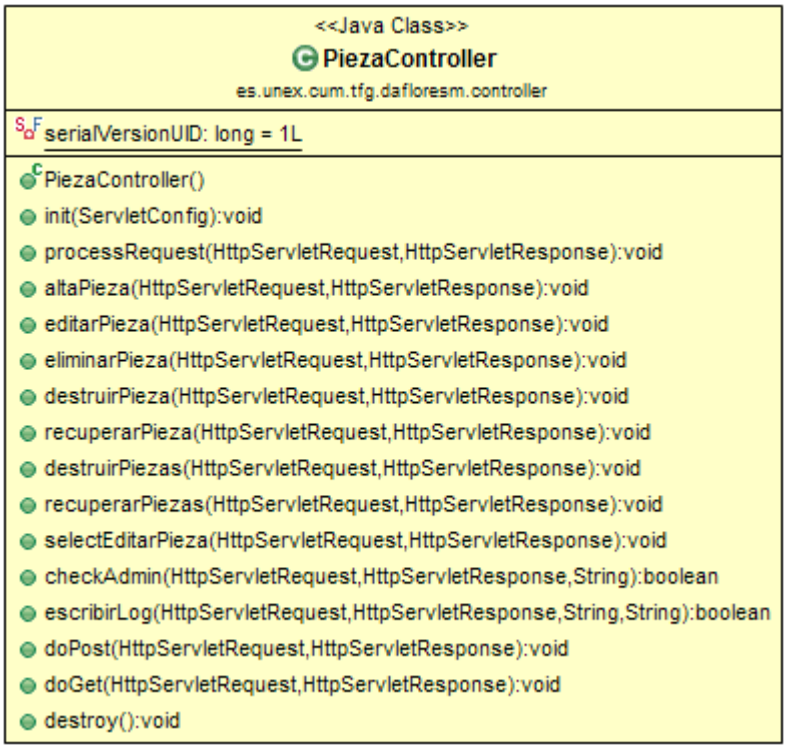

Figura 6.11: PiezaController

6.2.1.9. RegistroController

Todo lo que tenga que ver con el registro, login, cambiar o recordar contraseña, será gestionado por este servlet.

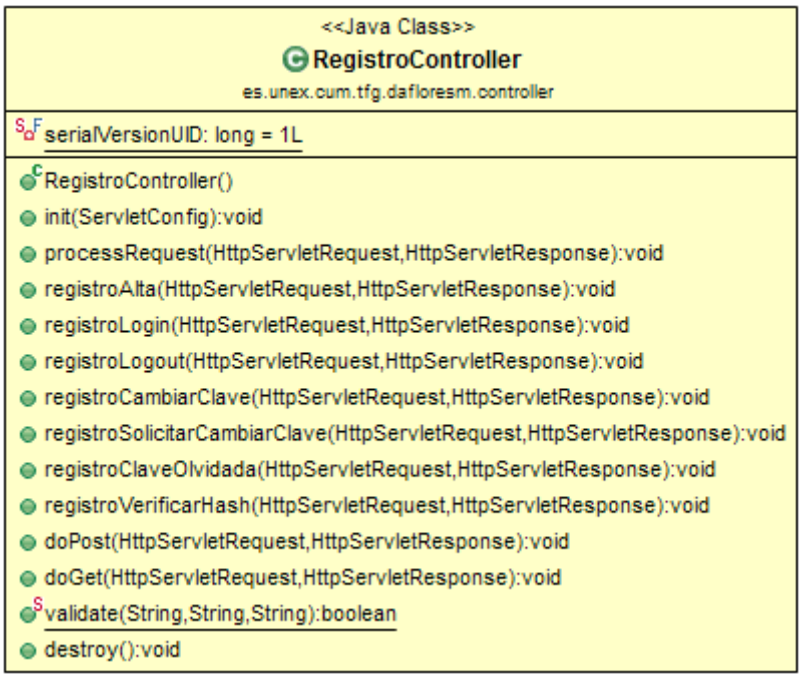

Figura 6.12: RegistroController

6.2.1.10. ReporteController

Los presupuestos también tendrán un servlet dedicado que será el encargado de proporcionar al usuario un resumen económico de su diseño.

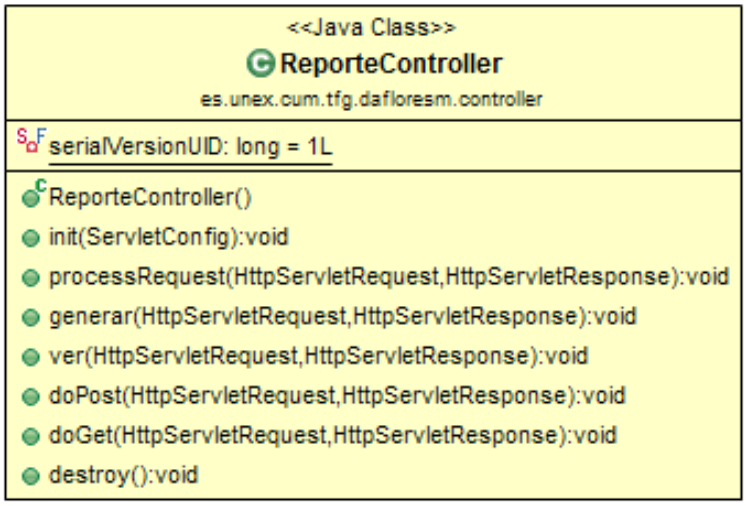

Figura 6.13: ReporteController

6.2.1.11. SMTPController

Este controlador será el encargado de gestionar los datos de configuración relativos al servidor de correo saliente SMTP.

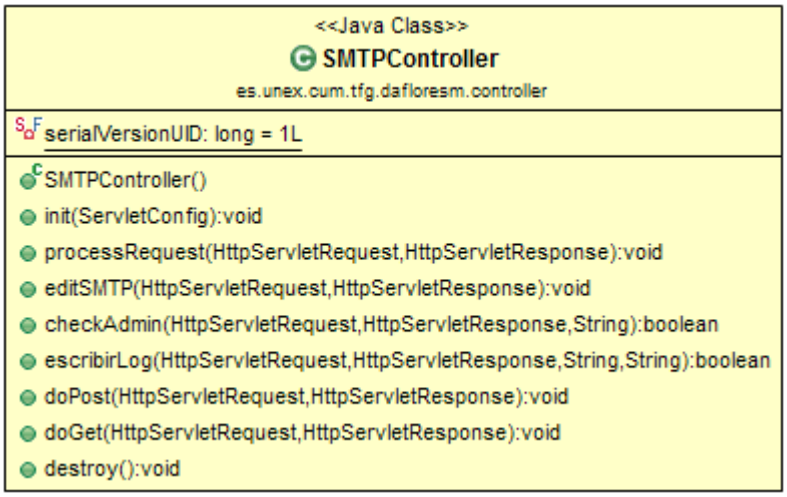

Figura 6.14: SMTPController

6.2.1.12. SubcategoriaController

Además de las categorías, contamos con subcategorías que serán a las que pertenecerán las piezas. Toda la gestión de alta, baja o modificación será llevada a cabo por este servlet.

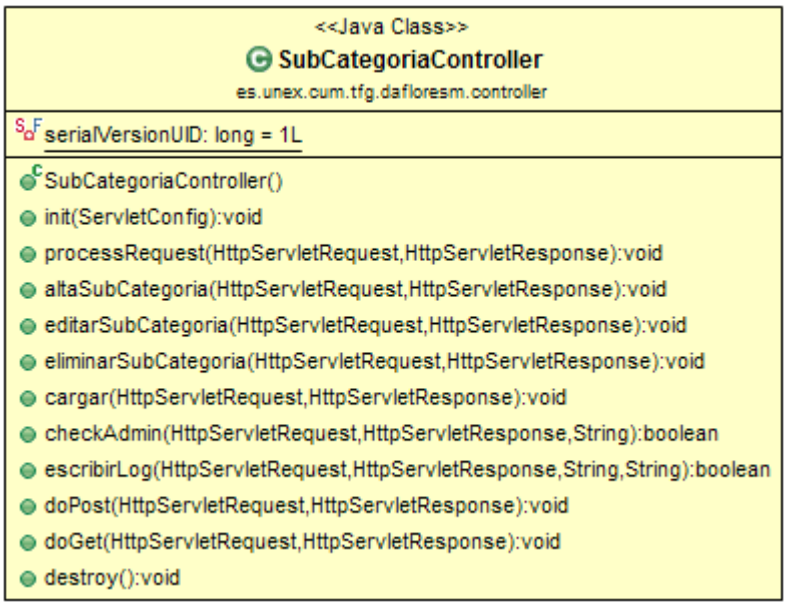

Figura 6.15: SubcategoriaController

6.2.1.13. TexturaController

Las texturas que podrán usar las piezas también tienen su propio servlet que proporcionarán las operaciones básicas de creación, modificación y eliminación de registros.

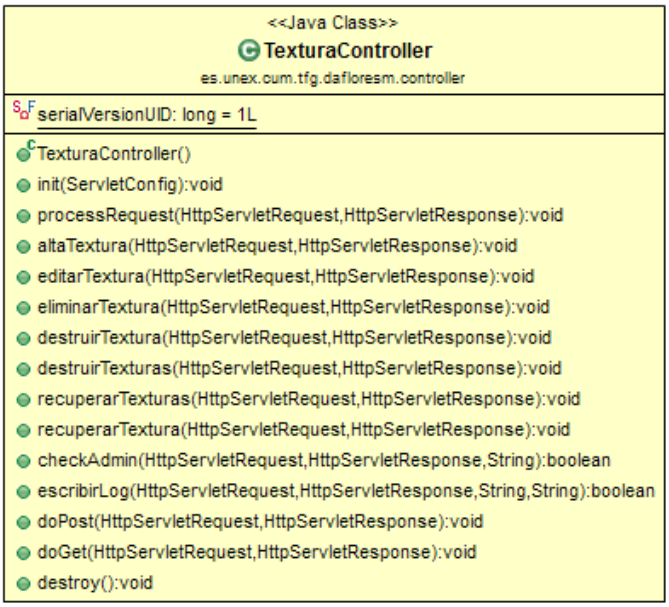

Figura 6.16: TexturaController

6.2.1.14. UsuarioController

La gestión de usuario del panel de administración será llevada a cabo por este servlet.

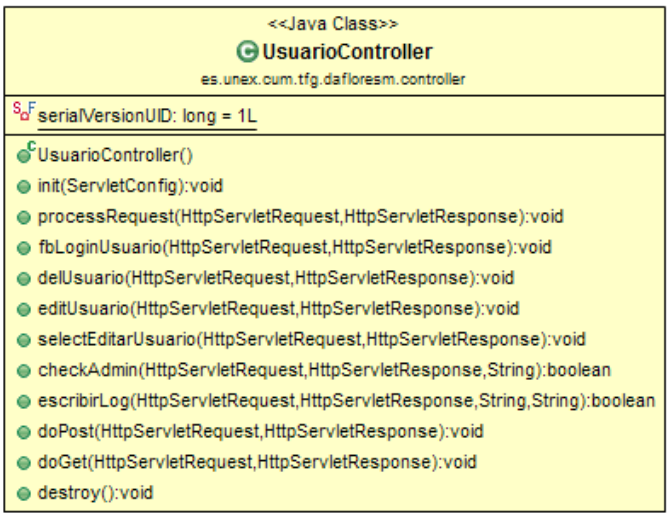

Figura 6.17: UsuarioController

6.2.2. Model

El paquete "model" está formado por las clases base que serán instanciadas para almacenar los datos. Por datos entendemos todos los datos de un usuario, de una pieza, textura, grupo, etc. Son clases base que no tienen ninguna función especial, solamente los atributos y los métodos de acceso a los mismos (getters y setters).

La figura siguiente representa todas las clases con sus atributos, así como las relaciones entre ellas (los métodos de acceso han sido omitidos en el diagrama).

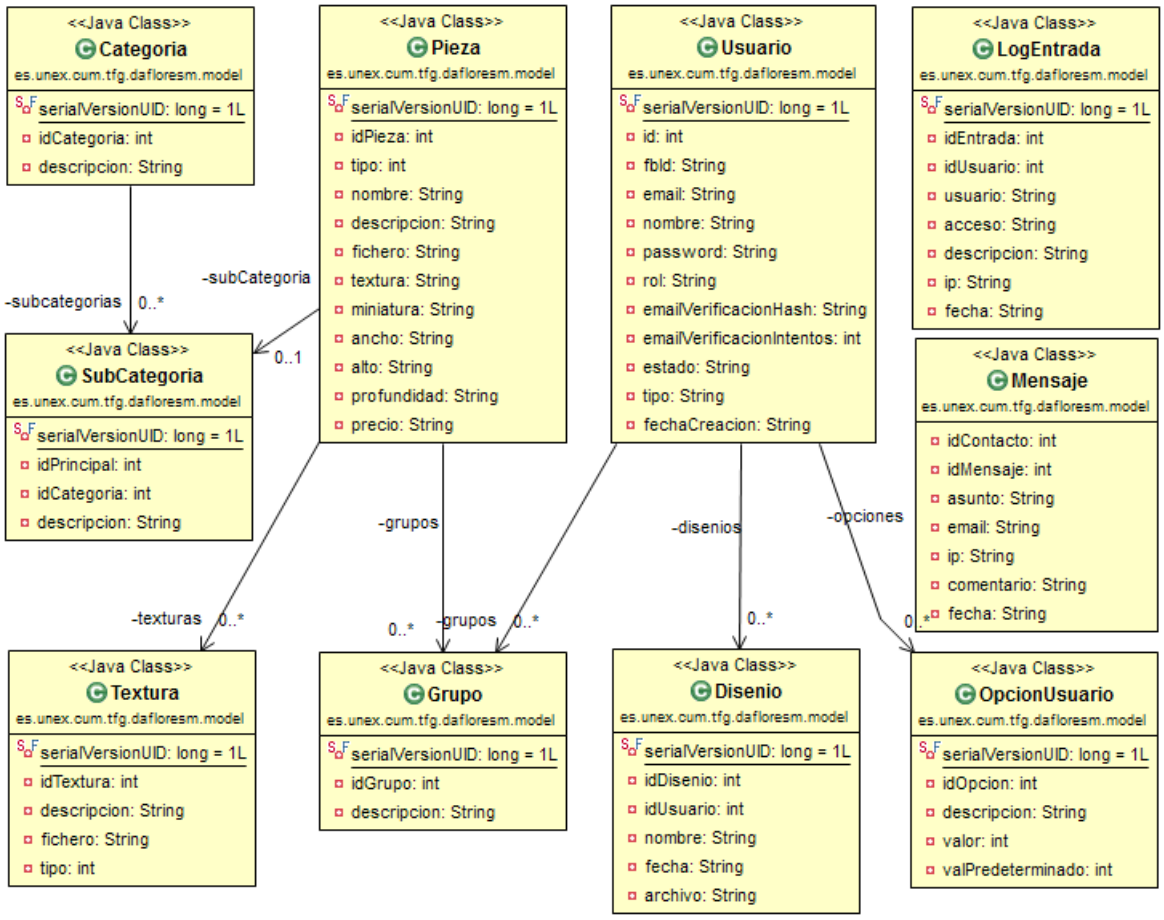

Figura 6.18: Clases base del paquete "Model"

6.2.3. Service

Las operaciones que se llevarán a cabo en la base de datos están recogidas en este paquete. Cuando un servlet del controlador requiera hacer alguna operación que necesite de la base de datos, invocará a su servlet correspondiente en esta capa. Está basada en clases que implementan interfaces.

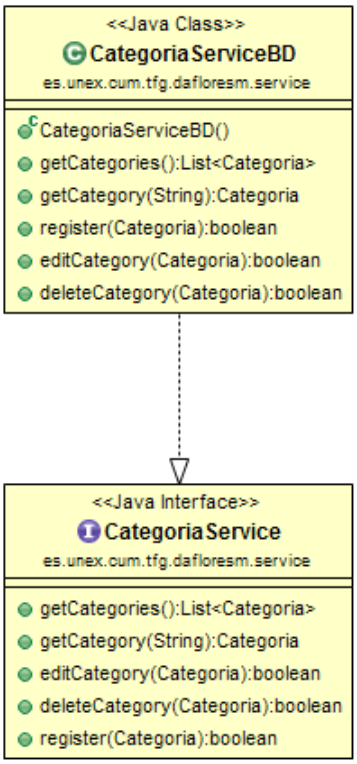

Figura 6.19: CategoriaService

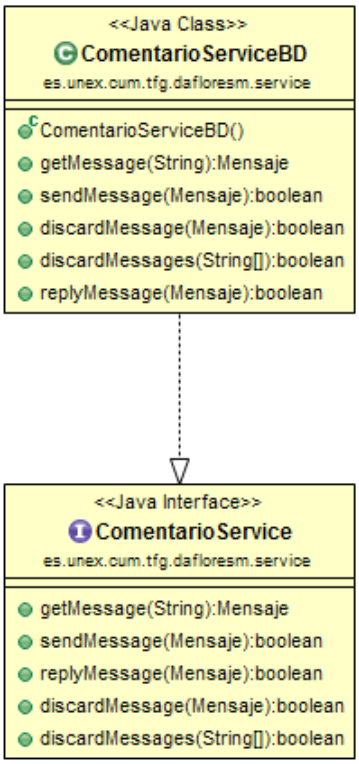

Figura 6.20: ComentarioService

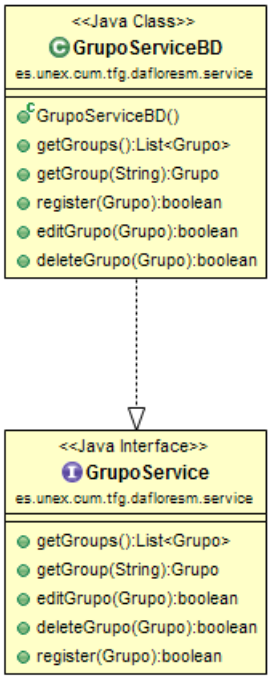

Figura 6.21: GrupoService

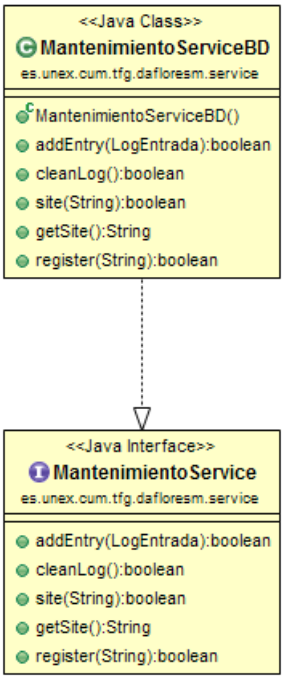

Figura 6.22: MantenimientoService

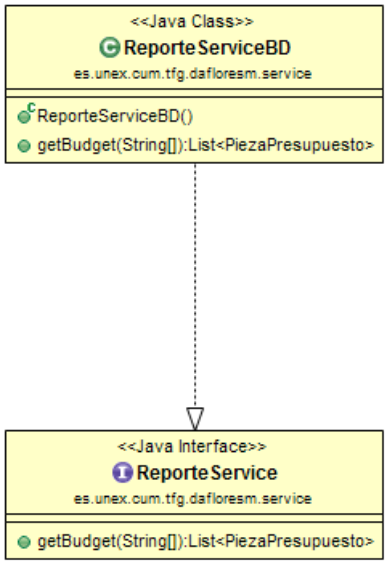

Figura 6.23: ReporteService

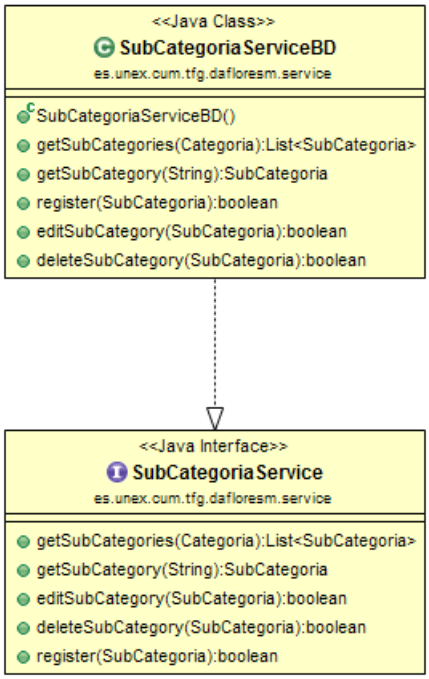

Figura 6.24: MantenimientoService

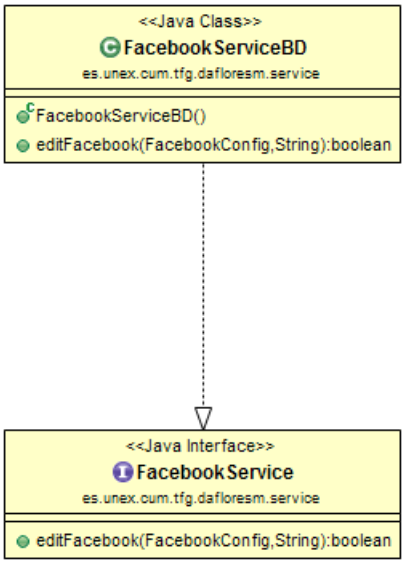

Figura 6.25: FacebookService

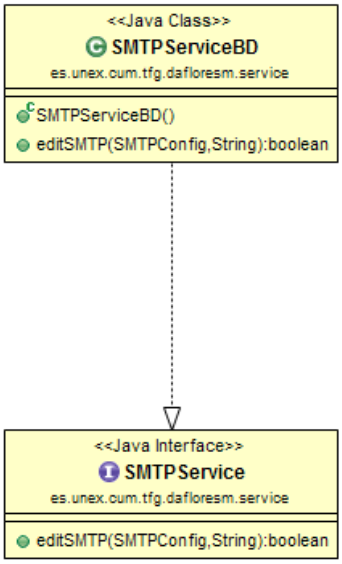

Figura 6.26: SMTPService

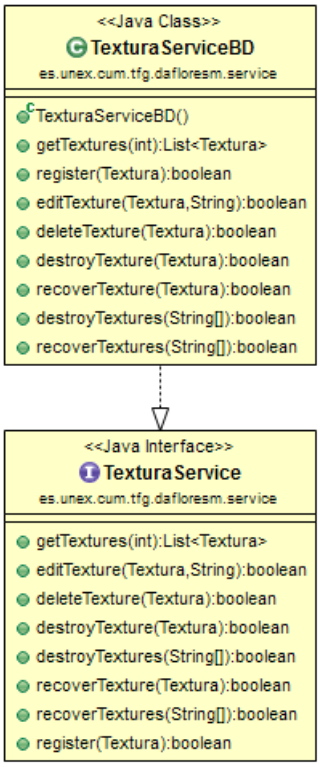

Figura 6.27: TexturaService

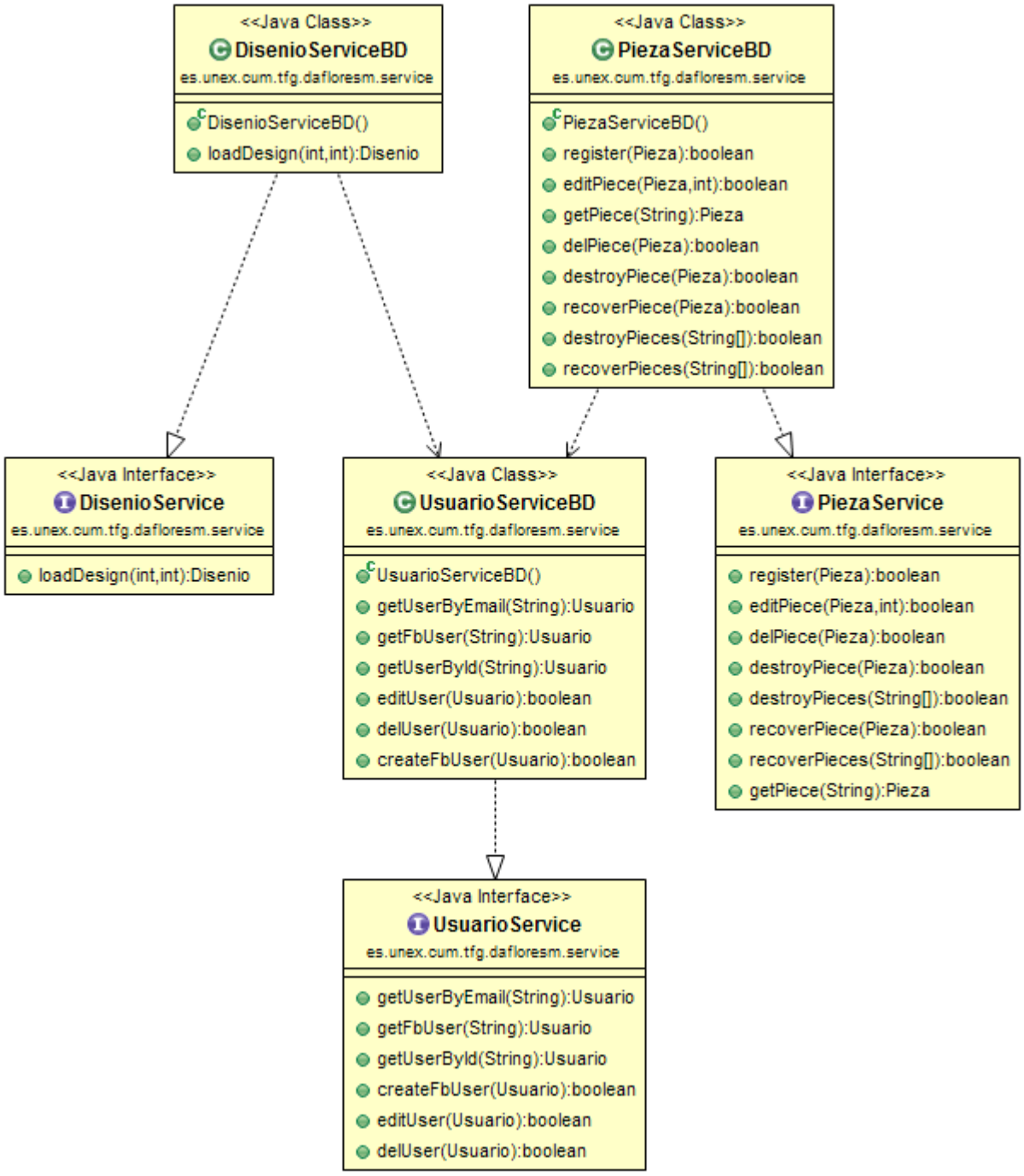

Figura 6.28: Disenio-Pieza-UsuarioService

6.2.4. Util

El paquete util recoge aquellas clases que sirven para alguna cuestión específica como cifrar contraseñas, realizar login con Facebook, variables globales, utilidades del email, etc.

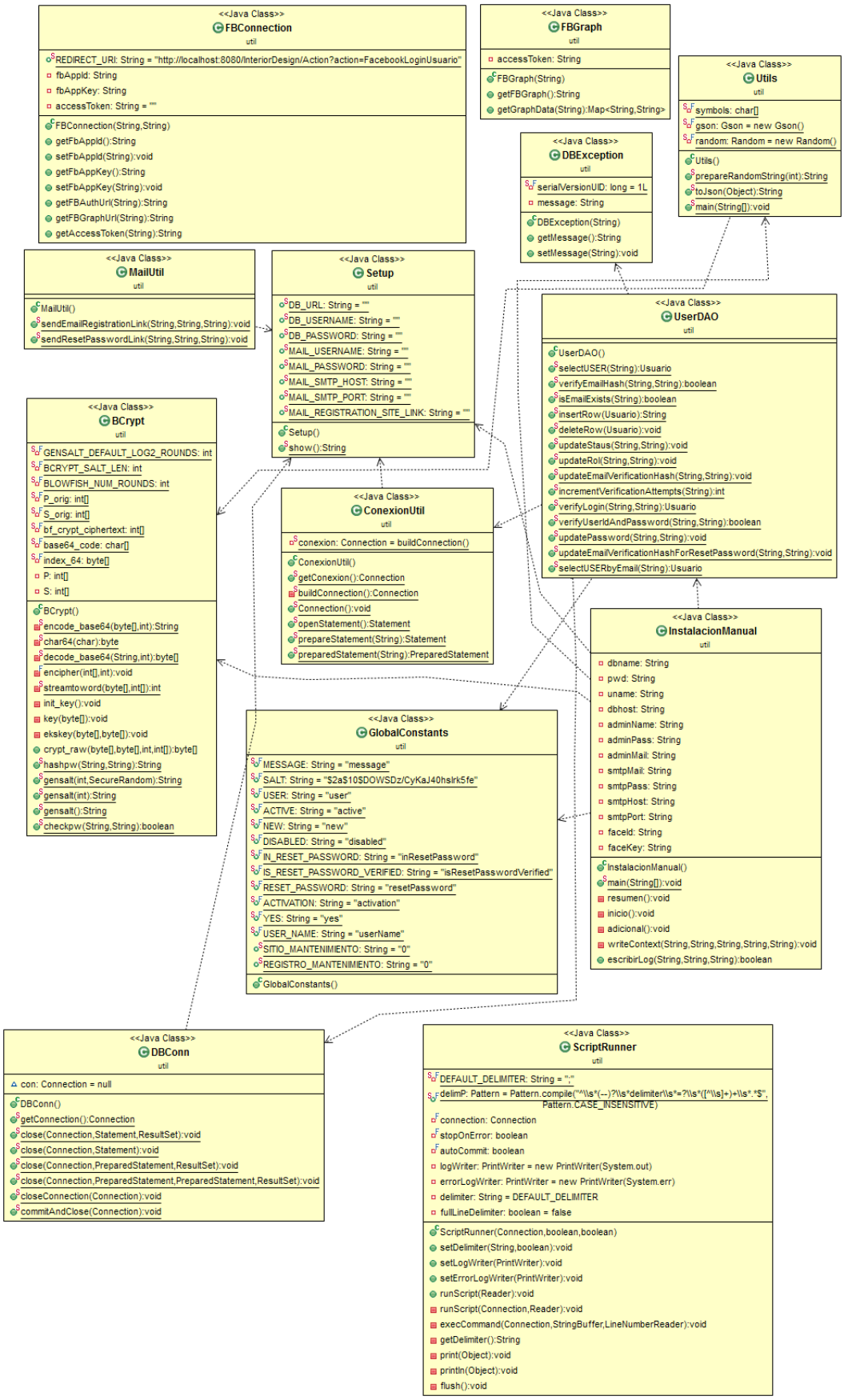

Figura 6.29: Util

6.3. Base de datos

Siguiendo con el modelo entidad relación (E-R) visto en el capítulo 5, "Análisis", podemos diseñar la base de datos. Además podemos acompañarla de los diagramas relacional y entidad atributo para que nos ayude a visualizarla más detalladamente.

6.3.1. Modelo relacional

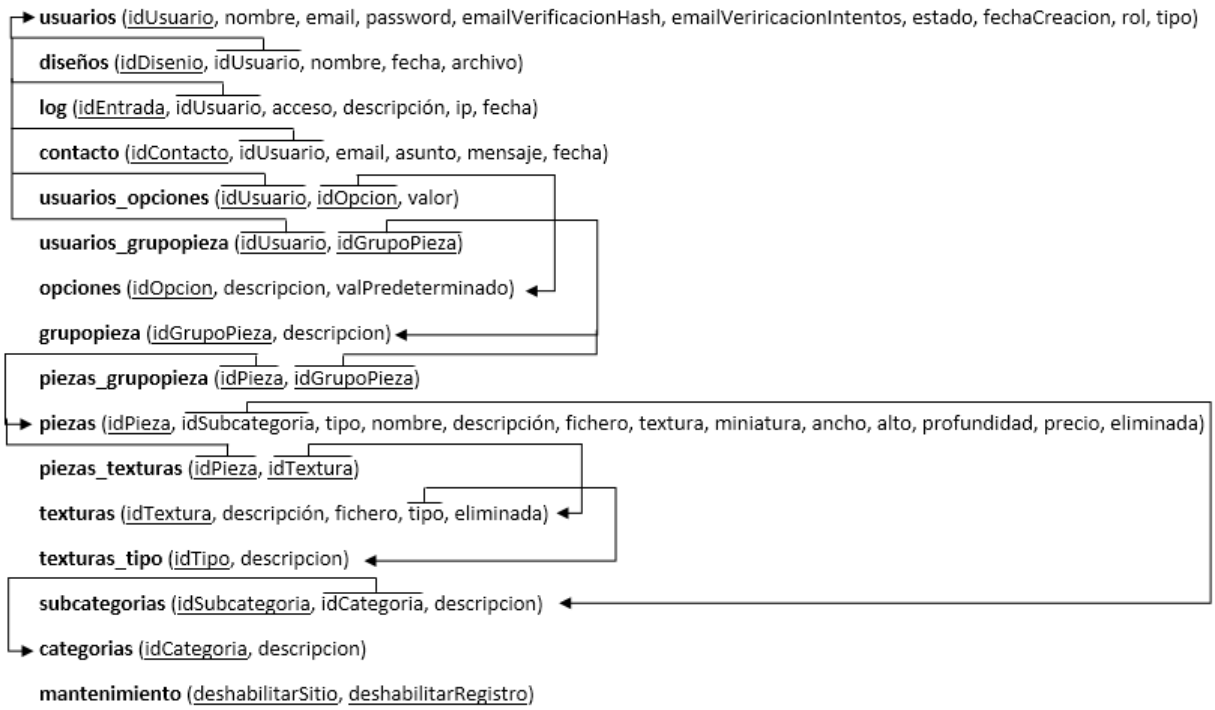

Figura 6.30: Modelo relacional

6.3.2. Modelo entidad atributo

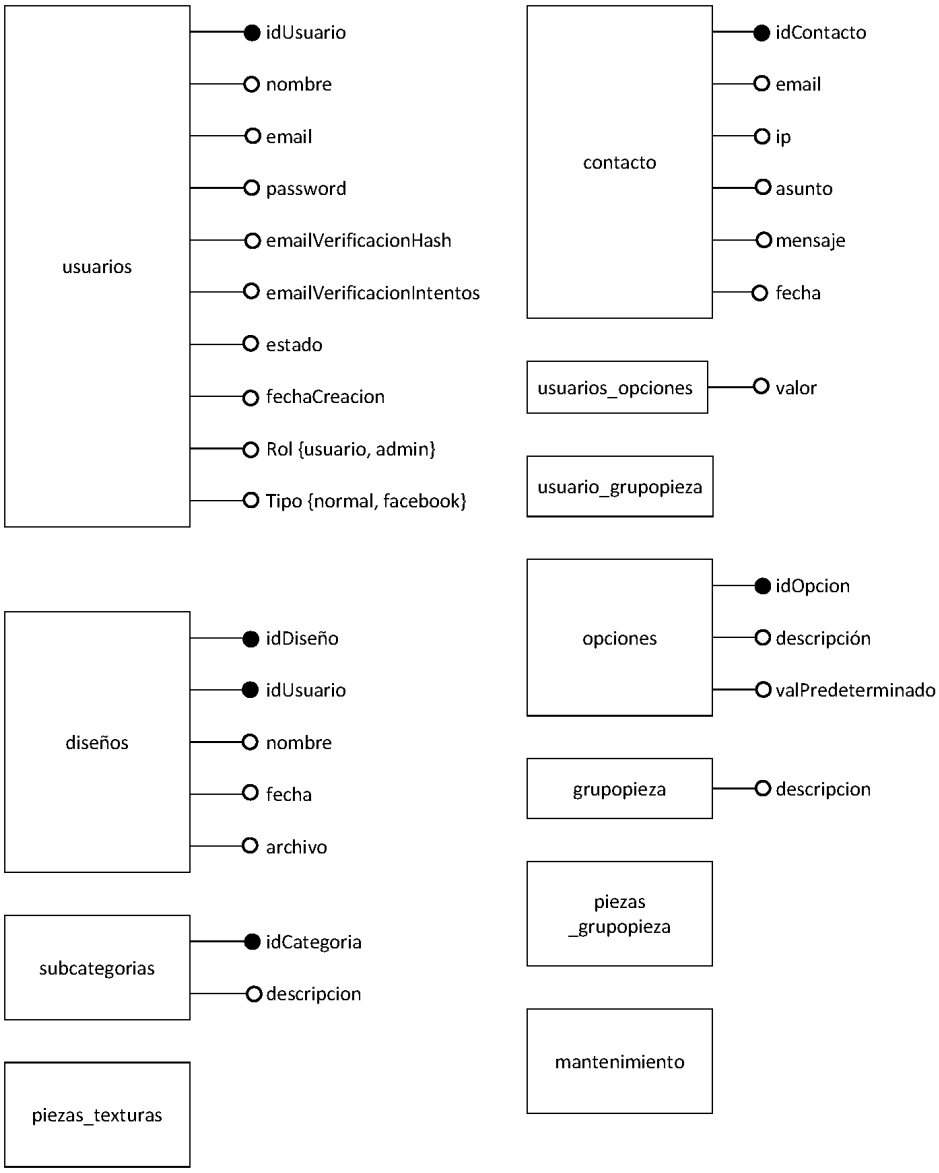

Figura 6.31: Modelo Entidad-Atributo (1/2)

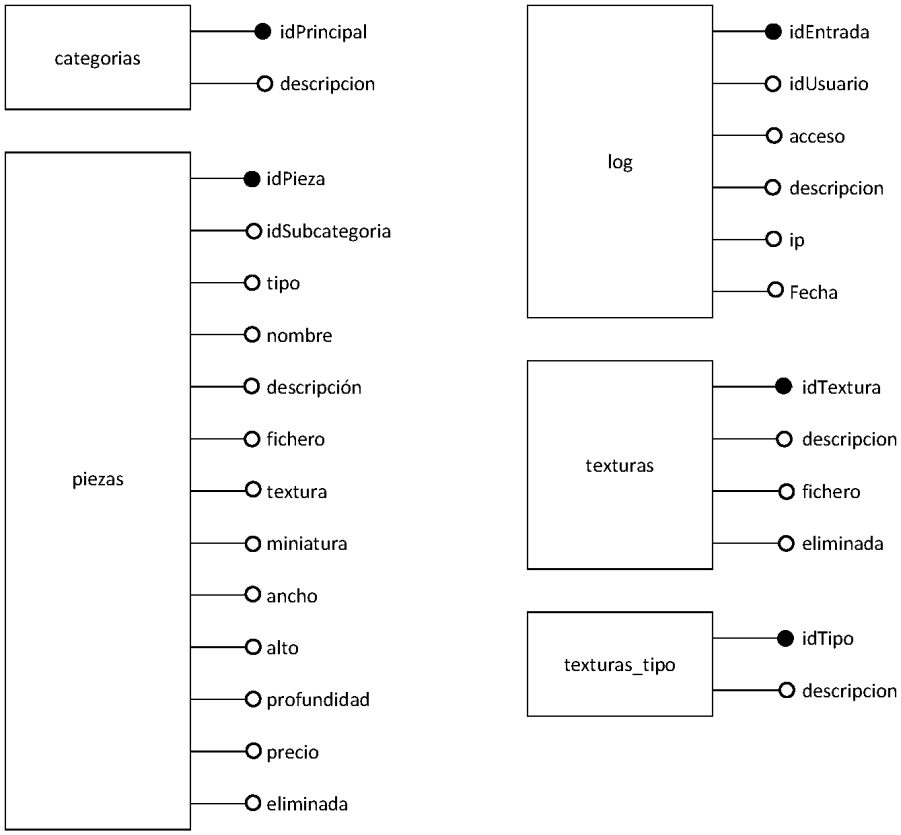

Figura 6.32: Modelo Entidad-Atributo (2/2)

6.3.3. Diseño de la base de datos

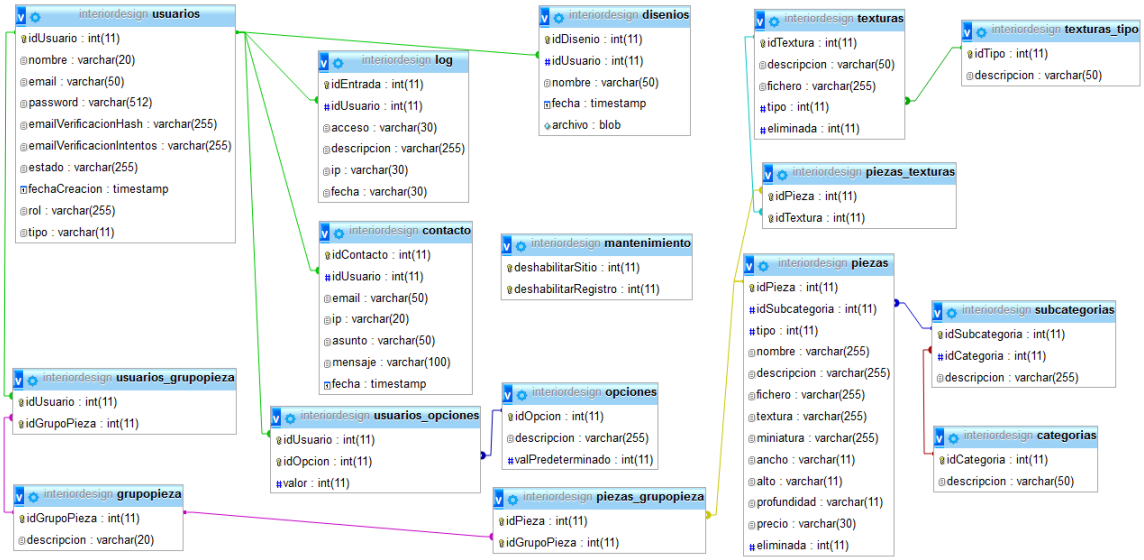

Figura 6.33: Diseño de la base de datos

Capítulo 7

Configuración inicial

Antes de nada debemos instalar los requisitos que nuestra aplicación va a necesitar, que son Apache, Tomcat y MySQL. Podemos instalar tanto Apache, Tomcat y MySQL de manera independiente y manual, pero hay herramientas que facilitan este proceso, y que serán utilizadas para la elaboración de este proyecto. Es el caso de XAMPP.

En este manual se explicará como configurar XAMPP para tener nuestro Tomcat y MySQL configurados y funcionando correctamente en Windows [23].

7.1. XAMPP: X, Apache, MySQL, PHP, Perl

7.1.1. Introducción

XAMPP es el entorno más popular de desarrollo con PHP. El nombre proviene del acrónimo de X (para cualquiera de los diferentes sistemas operativos), Apache, MySQL, PHP, Perl. XAMPP es una distribución de Apache completamente gratuita y fácil de instalar que contiene MariaDB (derivado de MySQL), PHP y Perl. El paquete de instalación de XAMPP ha sido diseñado para ser increíblemente fácil de instalar y usar.

7.1.2. Descarga

Basta con ir a la página principal (`https://www.apachefriends.org/es/index.html`) y seleccionar el sistema operativo para comenzar la descarga [4].

Figura 7.1: Descarga de XAMPP

7.1.3. Instalación

Nota: Antes de instalar un servidor de páginas web es conveniente comprobar si no hay ya uno instalado. Para ello, es suficiente con abrir el navegador y escribir la dirección *http://localhost*. Si no se obtiene un mensaje de error es que hay algún servidor de páginas web instalado.

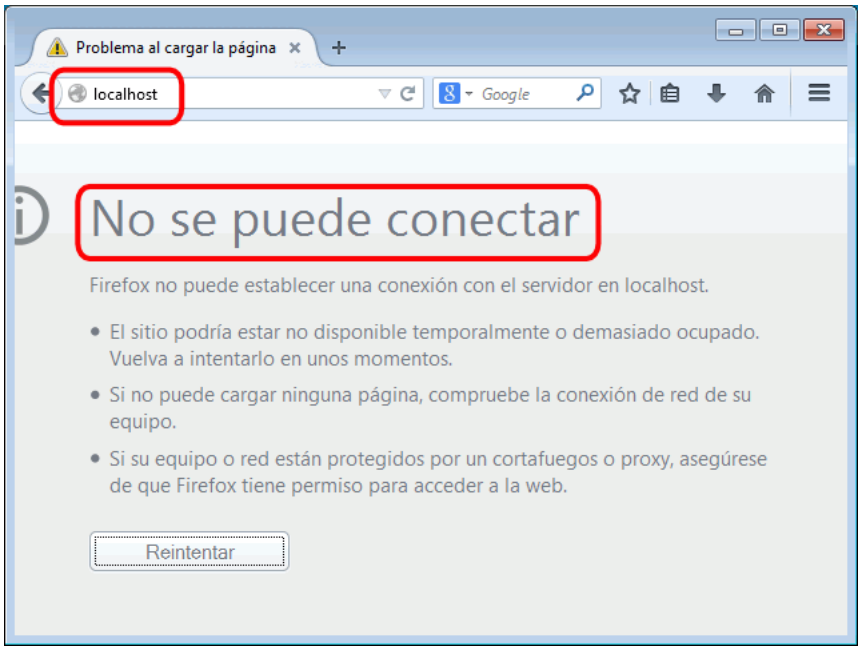

Figura 7.2: Comprobación Apache

Ejecutando el archivo de instalación Al poner en marcha el instalador XAMPP nos muestra dos avisos:

- El primero aparece si en el ordenador hay instalado un antivirus:

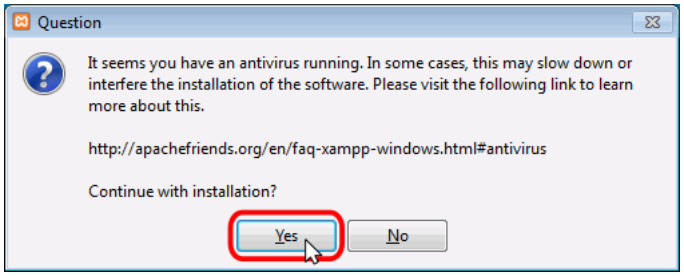

Figura 7.3: XAMPP - Aviso antivirus

- El segundo aparece si está activado el Control de Cuentas de Usuario (UAC) y recuerda que algunos directorios tienen permisos restringidos:

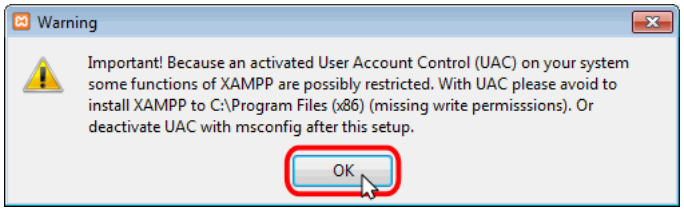

Figura 7.4: XAMPP - Control de Cuentas de Usuario (UAC)

Comenzando la instalación Una vez aceptados los dos avisos anteriores podemos proceder a la instalación.

A continuación se inicia el asistente de instalación. Para continuar, hay que hacer clic en el botón "Next".

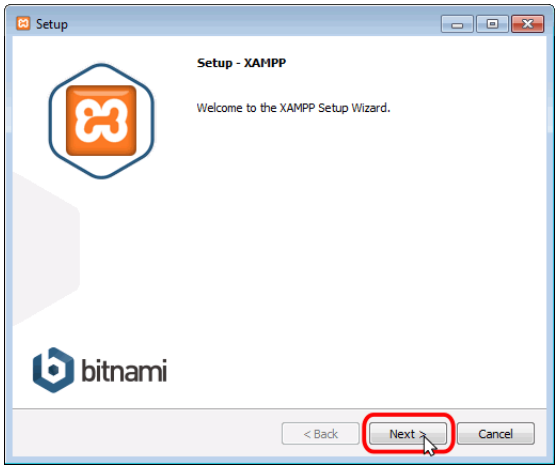

Figura 7.5: XAMPP - Bienvenida

Los componentes mínimos que instala XAMPP son el servidor Apache y el lenguaje PHP, pero XAMPP también instala otros elementos. En la pantalla de selección de componentes puede elegirse la instalación o no de estos componentes. Para nuestra aplicación se necesita al menos instalar Tomcat, MySQL y phpMyAdmin. *phpMyAdmin* es un gestor que nos permitirá administrar la base de datos de manera gráfica. No es un requisito indispensable de la aplicación, pues las tareas con la base de dato se pueden hacer a través de la consola de MySQL, pero nos ayudará bastante a la hora de realizar las operaciones, por lo que se recomienda instalar.

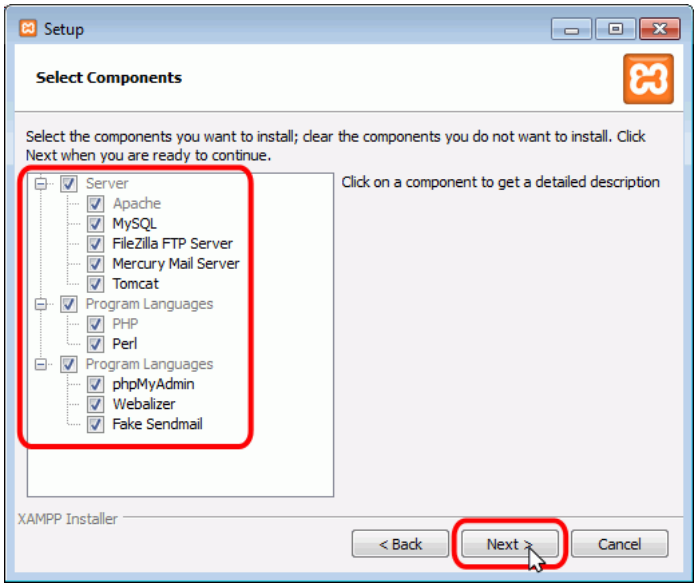

Figura 7.6: XAMPP - Selección de componentes

En la siguiente pantalla se puede elegir la carpeta de instalación de XAMPP. La carpeta de instalación predeterminada es **C:\xampp**. Si se quiere cambiar, hay que hacer clic en el icono de carpeta y seleccionar la carpeta donde se quiere instalar XAMPP. Para continuar la configuración de la instalación, hay que hacer clic en el botón "Next".

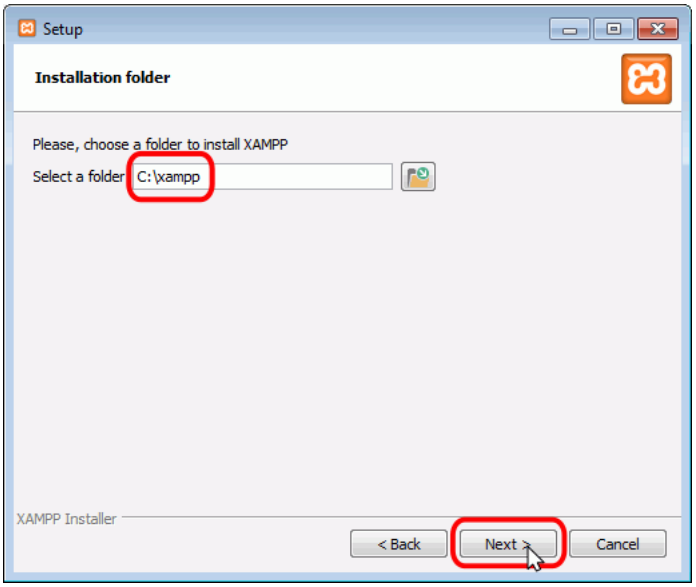

Figura 7.7: XAMPP - Directorio de instalación

La siguiente pantalla nos ofrece información sobre los instaladores de aplicaciones para XAMPP creados por Bitnami. Para que no se abra la página web de Bitnami, habría que desmarcar la casilla correspondiente.

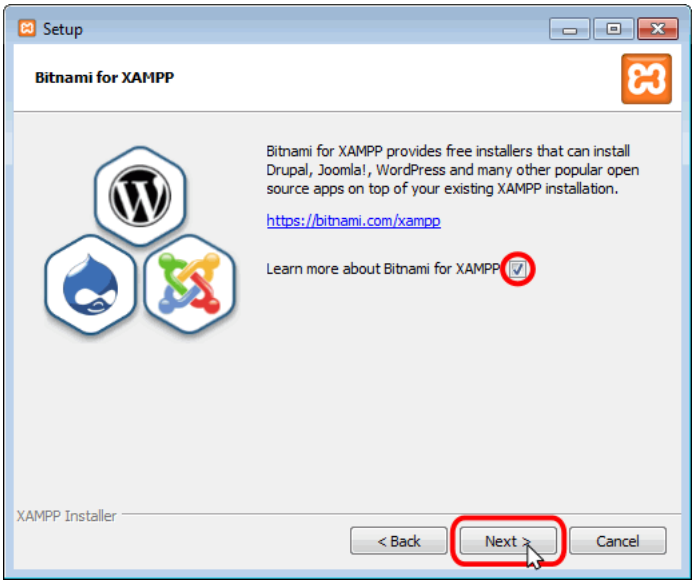

Figura 7.8: XAMPP - Información adicional

Para empezar la instalación de XAMPP, hay que hacer clic en en el botón "Next" en la pantalla siguiente.

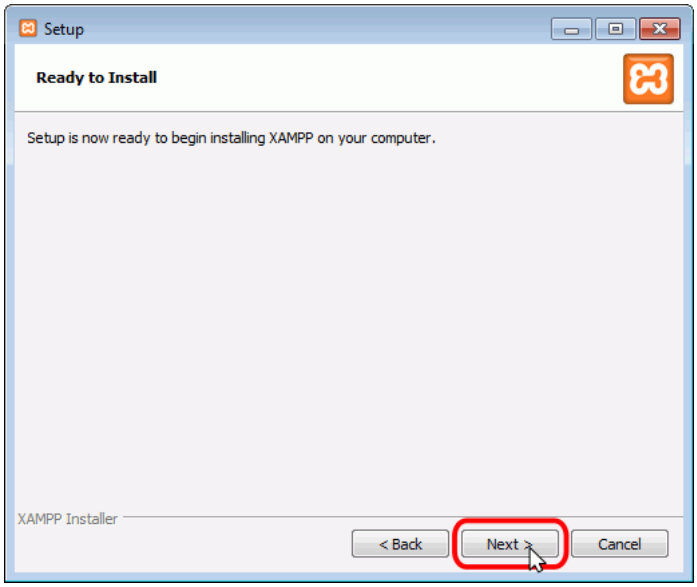

Figura 7.9: XAMPP - Preparado para instalar

A continuación, se inicia el proceso de copia de archivos, que puede durar unos minutos.

Figura 7.10: XAMPP - Copiando archivos

Durante la instalación, si en el ordenador no se había instalado Apache anteriormente, se mostrará un aviso del cortafuegos de Windows para autorizar a Apache para comunicarse en las redes domésticas o de trabajo, lo que debemos permitir haciendo clic en el botón "Permitir acceso".

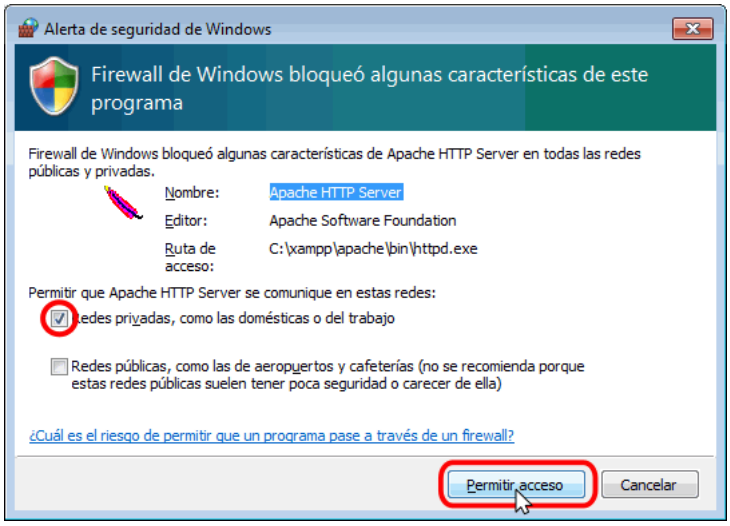

Figura 7.11: XAMPP - Permisos Firewall Windows

Una vez terminada la copia de archivos, se muestra la pantalla que confirma que XAMPP ha sido instalado. Hay que hacer clic en el botón "Finish". Para no abrir a continuación el panel de control de XAMPP habría que desmarcar la casilla correspondiente.

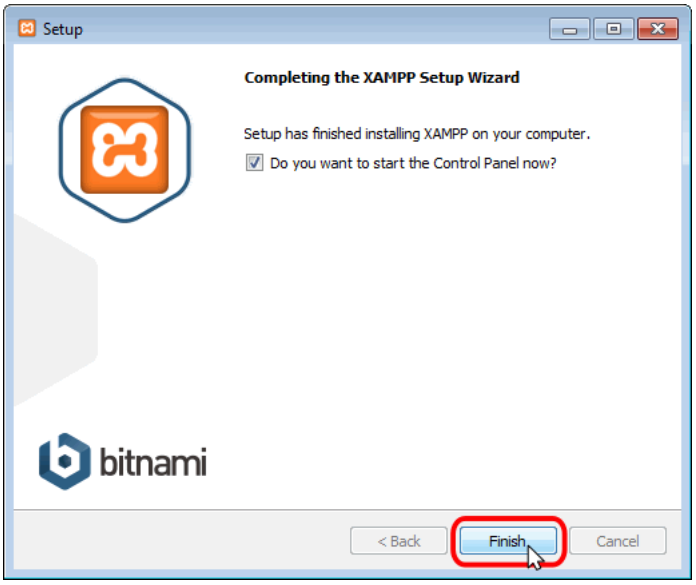

Figura 7.12: XAMPP - Instalación finalizada

7.1.4. El panel de control de XAMPP

7.1.4.1. Abrir y cerrar el panel de control

Al panel de control de XAMPP se puede acceder mediante el menú de inicio "Todos los programas > XAMPP > XAMPP Control Panel" o, si ya está iniciado, mediante el icono del área de notificación.

La primera vez que se abre el panel de control de XAMPP, se muestra una ventana de selección de idioma que permite elegir entre inglés y alemán.

Figura 7.13: XAMPP - Selección de idioma

El panel de control de XAMPP se divide en tres zonas:

1. La zona de módulos, que indica para cada uno de los módulos de XAMPP: si está instalado como servicio, su nombre, el identificador de proceso, el puerto utilizado e incluye unos botones para iniciar y detener los procesos, administrarlos, editar los archivos de configuración y abrir los archivos de registro de actividad.

2. La zona de notificación, en la que XAMPP informa del éxito o fracaso de las acciones realizadas la zona de utilidades, para acceder rápidamente .

3. La zona de utilidades, para acceder rápidamente.

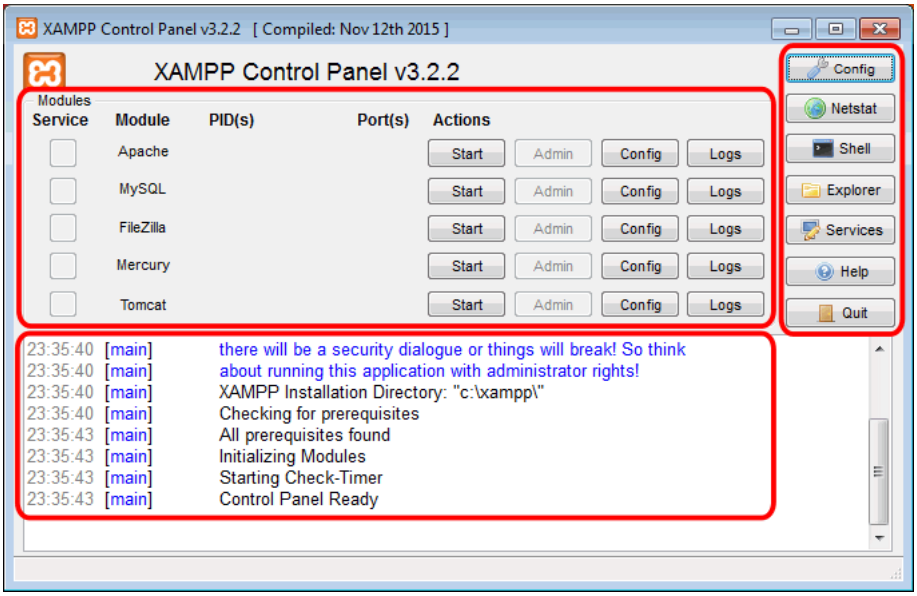

Figura 7.14: XAMPP - Panel de control

Para cerrar el panel de control de XAMPP hay que hacer clic en el botón Quit (al cerrar el panel de control no se detienen los servidores):

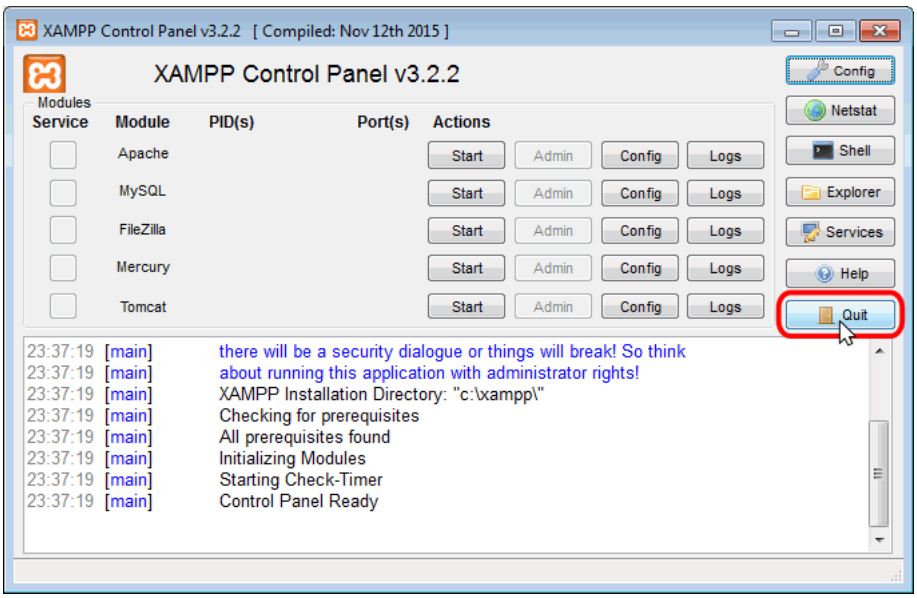

Figura 7.15: XAMPP - Salir del panel de control

El botón Cerrar en forma de aspa no cierra realmente el panel de control, sólo lo minimiza:

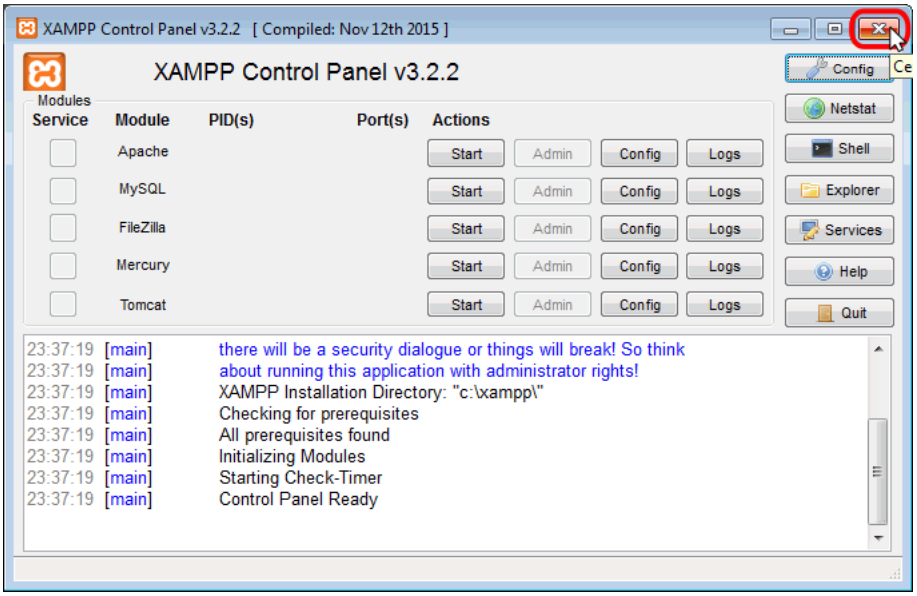

Figura 7.16: XAMPP - Minimizar panel de control

Si se ha minimizado el panel de control de XAMPP, se puede volver a mostrar haciendo doble clic en el icono de XAMPP del área de notificación.

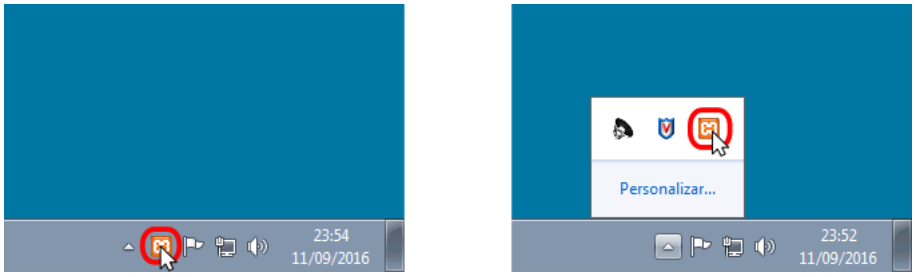

Figura 7.17: XAMPP - Área de notificación

Haciendo clic derecho en el icono de XAMPP del área de notificación se muestra un menú que permite mostrar u ocultar el panel de control, arrancar o detener servidores o cerrar el panel de control.

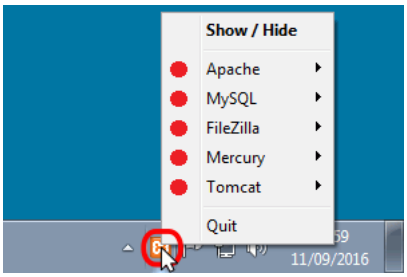

Figura 7.18: XAMPP - Icono del área de notificación

Se pueden abrir varios paneles de control simultáneamente y cualquiera de ellos puede iniciar o detener los servidores, pero no es aconsejable hacerlo ya que puede dar lugar a confusiones (por ejemplo, al detener un servidor desde un panel de control los otros paneles de control interpretan la detención como un fallo inesperado y muestran un mensaje de error).

7.1.4.2. El cortafuegos de Windows

Cuando se pone en marcha por primera vez cualquiera de los servidores que instala XAMPP, el cortafuegos de Windows pide al usuario confirmación de la autorización.

Por ejemplo, la primera vez que se pone en marcha Apache mediante el botón Start correspondiente...

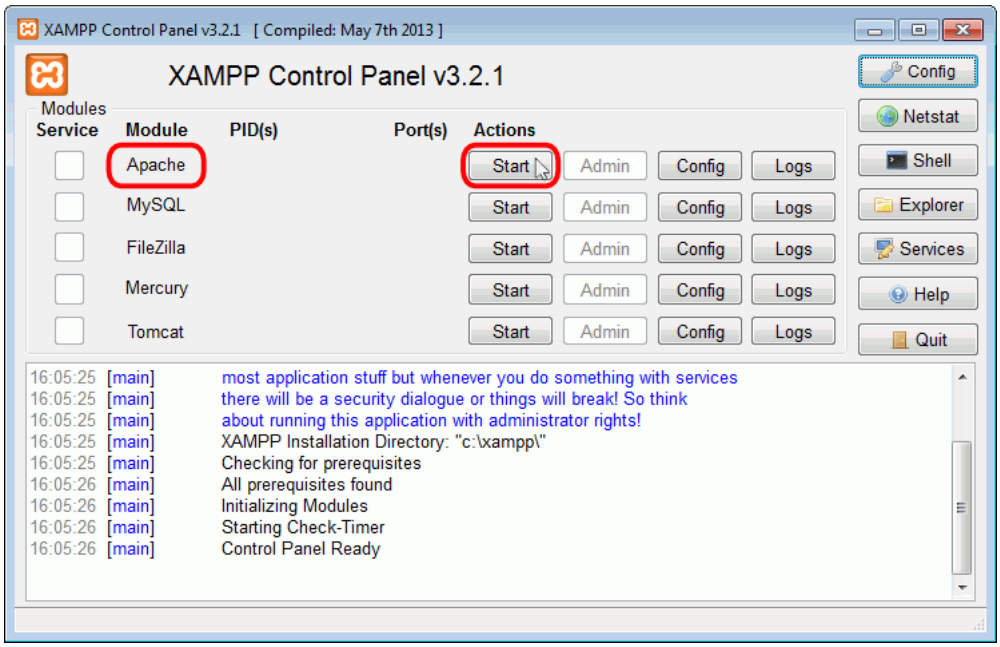

Figura 7.19: XAMPP - Iniciando Apache por primera vez

...como Apache abre puertos en el ordenador (por primera vez), el cortafuegos de Windows pide al usuario confirmación. Para poder utilizarlo hace falta al menos autorizar el acceso en redes privadas:

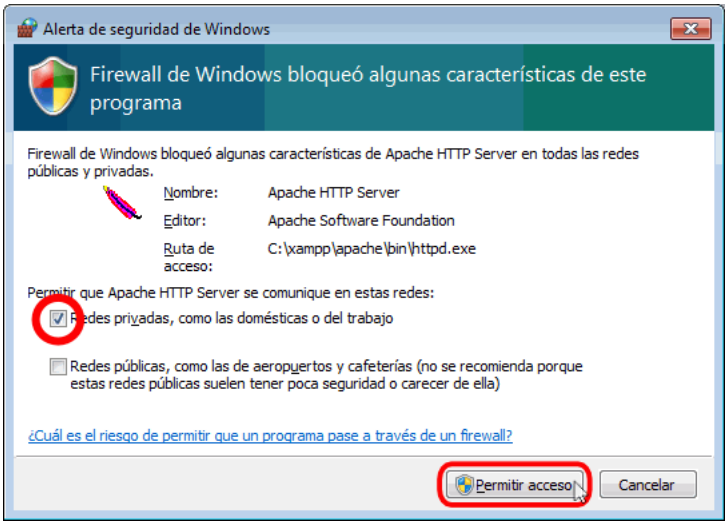

Figura 7.20: XAMPP - Concediendo permisos a Apache

Si el arranque de Apache tiene éxito, el panel de control mostrará el nombre del módulo con fondo verde, su identificador de proceso, los puertos abiertos (http y https), el botón "Start" se convertirá en el botón "Stop" y en la zona de notificación se verá el resultado de las operaciones realizadas.

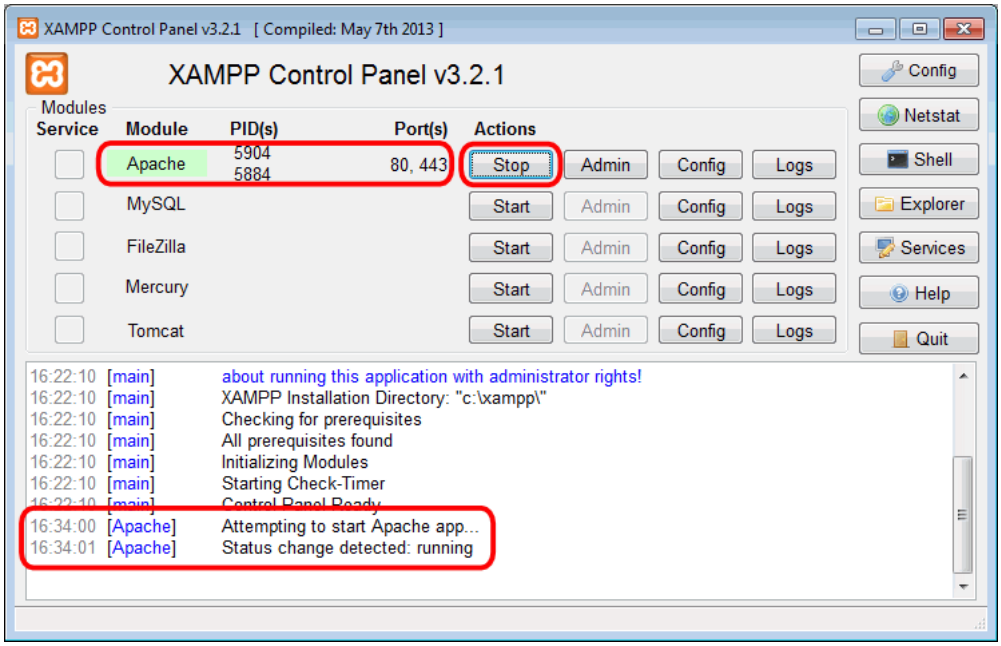

Figura 7.21: XAMPP - Identificadores y puertos

Si se abre el programa "Firewall de Windows con seguridad avanzada", en el apartado de Reglas de entrada pueden verse las nuevas reglas añadidas.

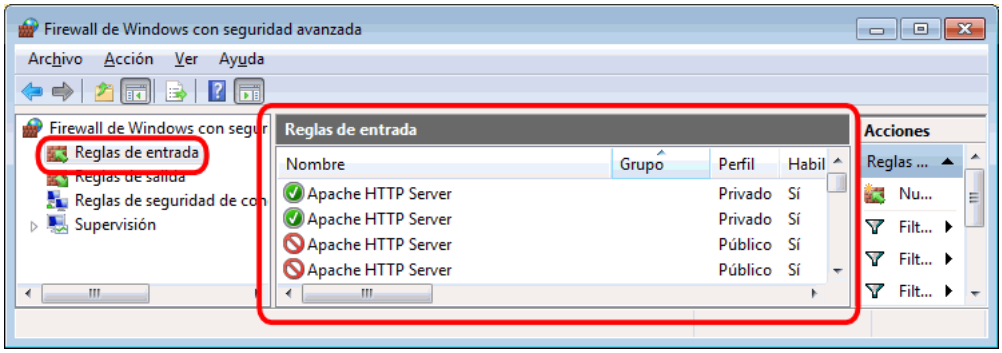

Figura 7.22: XAMPP - Seguridad avanzada del Firewall de Windows

7.1.4.3. Iniciar, detener y reiniciar servidores

A veces es necesario detener y reiniciar los servidores. Por ejemplo, los archivos de configuración de Apache se cargan al iniciar Apache. Si se modifica un archivo de configuración de Apache (httpd.conf, php.ini u otro) mientras Apache está en marcha, para recargar los archivos de configuración es necesario detener y reiniciar el servidor Apache.

Nota: Si al modificar el archivo de configuración hemos introducido errores, el servidor no será capaz de iniciarse. Si no sabemos encontrar el origen del problema, se recomienda restaurar los archivos de configuración originales, de los que se aconseja tener una copia de seguridad.

Ya hemos visto como arrancar Apache por primera vez. Para detenerlo hay que hacer clic en el botón "Stop" correspondiente a Apache. Si la parada de Apache tiene éxito, el panel de control mostrará el nombre del módulo con fondo gris, sin identificador de proceso ni puertos abiertos (http y https), el botón "Stop" se convertirá en un botón "Start" y en la zona de notificación se verá el resultado de las operaciones realizadas.

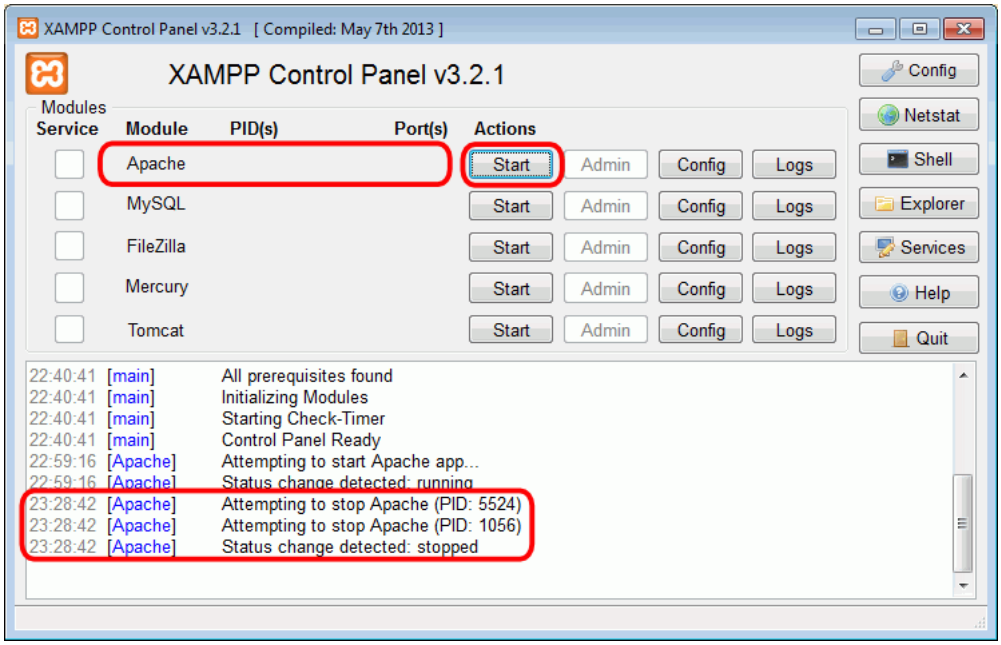

Figura 7.23: XAMPP - Deteniendo Apache

7.1.4.4. Ejecutar el panel de control como administrador

En algunas situaciones es necesario ejecutar el panel de control como administrador, por ejemplo, para configurar los servidores como servicios o deshabilitarlos.

Para ejecutar el panel de control como administrador, hay que hacer clic derecho sobre el icono de acceso directo (Inicio > Todos los programas > XAMPP > XAMPP Control Panel) y elegir la opción "Ejecutar como administrador".

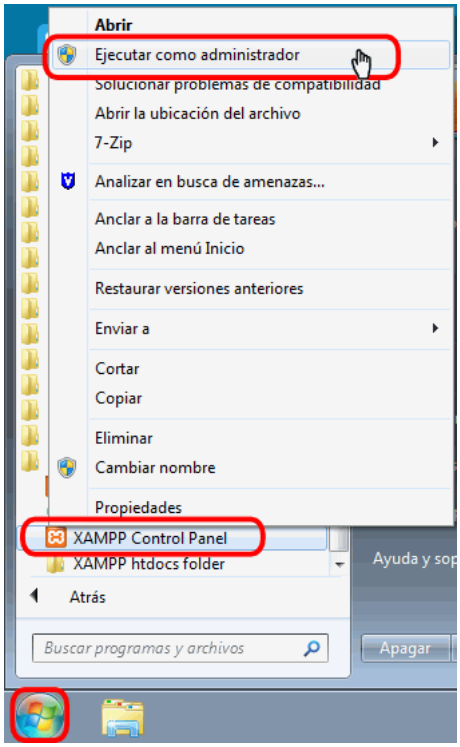

Figura 7.24: XAMPP - Ejecutar como Administrador

7.1.5. El panel de administración web de XAMPP

Si se ha iniciado el servidor Apache, para comprobar que todo funciona correctamente, hay que escribir en el navegador la dirección *http://localhost*. XAMPP abrirá el nuevo panel de administración web (dashboard):

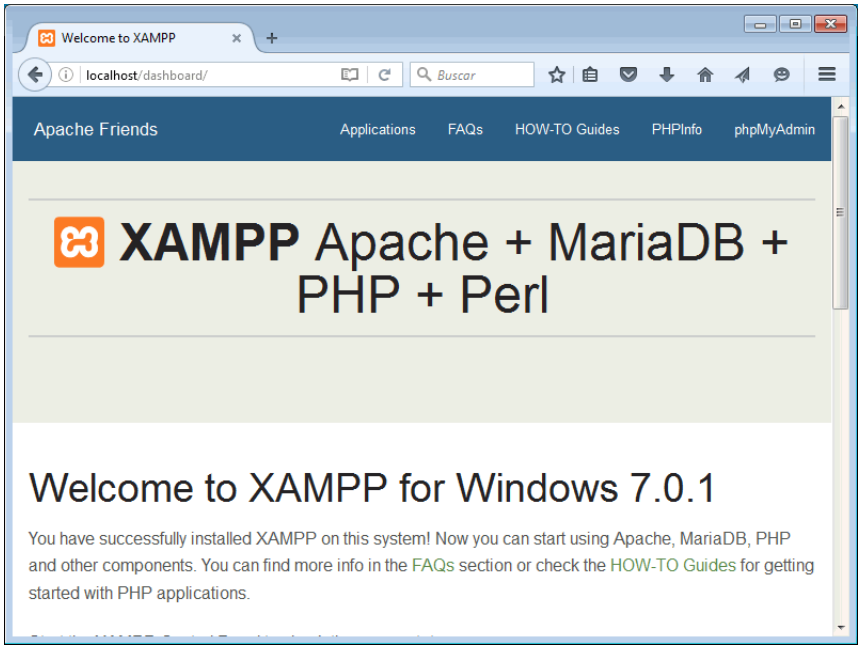

Figura 7.25: XAMPP - Dashboard

Además, haciendo clic en los enlaces superiores podremos acceder a phpMyAdmin.

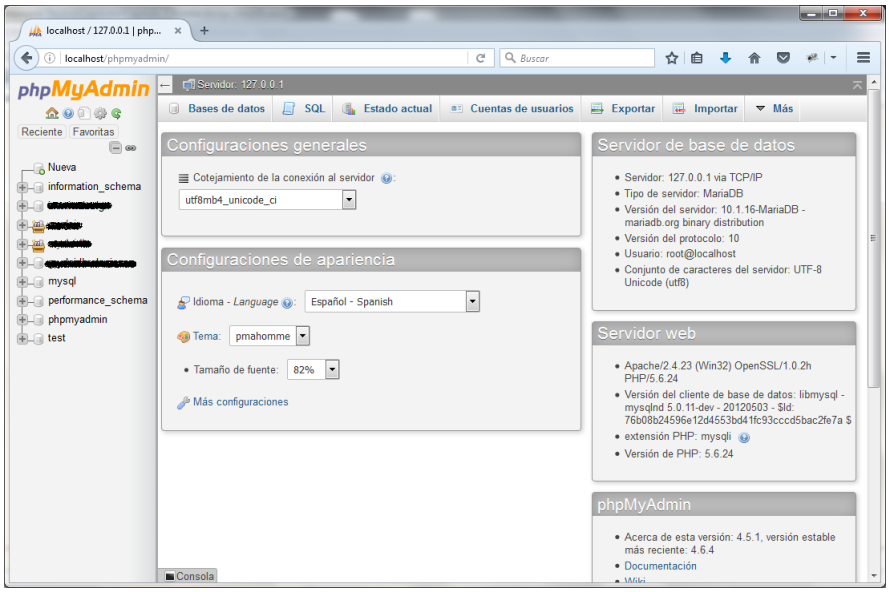

Figura 7.26: XAMPP - phpMyAdmin

Una vez hemos preparado los requisitos de nuestra aplicación, podemos comenzar a implementarla.

7.1.6. Configuración de Tomcat

Para configurar Tomcat, debemos modificar el archivo “tomcat-users.xml” para crear los usuarios y darles los privilegios necesarios para desplegar aplicaciones. Para ello, desde el propio panel de control de XAMPP podemos editar el archivo haciendo clic en “Config -> tomcat-users.xml”, y automáticamente se abrirá el archivo con un editor de texto. También podemos ir a la ruta donde tengamos instalado XAMPP y abrirlo manualmente (C:\...\xampp\tomcat\conf\tomcat-users.xml).

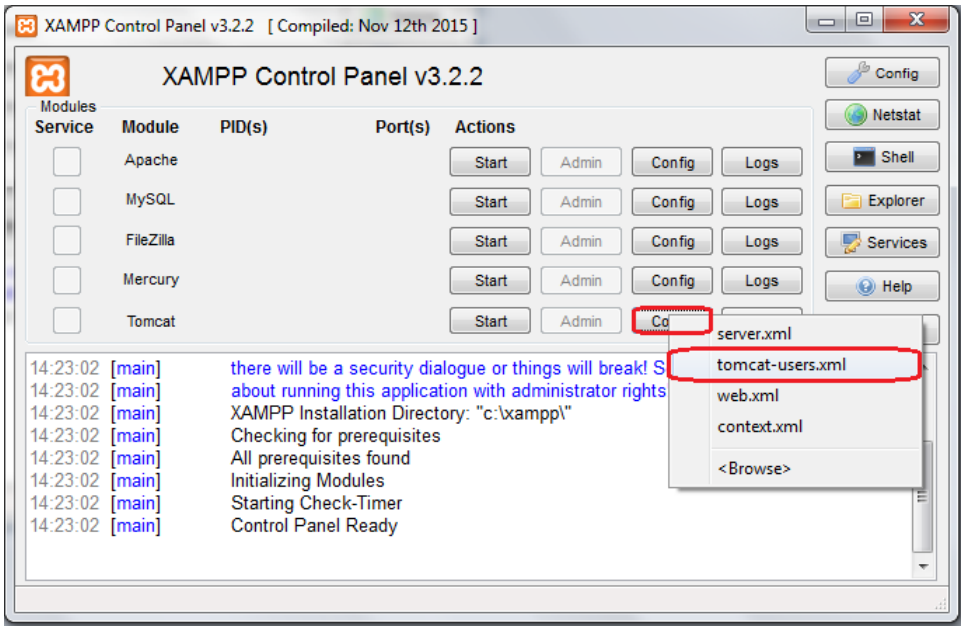

Figura 7.27: XAMPP - Configurando Tomcat

Tendremos que crear un rol que permita realizar cambios en las aplicaciones, y asignar nuestro usuario. Para ello deberemos agregar la siguiente configuración:

```
...
<role rolename="manager-gui"/>
<user username="tomcat" password="tomcat" roles="manager-gui"/>
...
```

Listado de código 7.1: XAMPP - Tomcat - Usuarios

El rol “manager-gui” hace referencia al tipo de rol que tiene los permisos necesarios para desplegar y eliminar aplicaciones, en otras palabras, para tener acceso al apartado “Manager App” de Tomcat.
Con la configuración anterior, tendríamos acceso al gestor de aplicaciones a través del usuario “tomcat” y la contraseña “tomcat”.
A la hora de editar el fichero, hay que tener en cuenta que esta parte normalmente

la encontraremos al final del mismo contenida entre comentarios, por lo que cuando realicemos la modificaciones oportunas, tendremos que eliminar dichos comentarios para que la configuración surta efecto.

7.2. Eclipse IDE

Para la implementación de la plataforma utilizaremos el entorno de desarrollo (IDE) Ecplipse IDE, en su versión JEE (Java Enterprise Edition), que es la que nos permitirá crear aplicaciones web.

7.2.1. Descarga

La descarga de Eclipse es tan sencilla como ir a su página web oficial y hacer clic sobre el botón "Download", para después seleccionar la arquitectura correspondiente, en nuestro caso, de 64bits. Se descargará el instalador oficial de Eclipse, donde posteriormente podremos seleccionar la versión que nos interese (JEE)

Figura 7.28: Eclipse - Botón de descarga

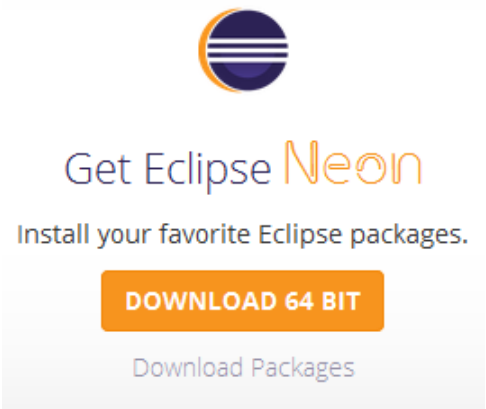

Figura 7.29: Eclipse - Descarga versión 64bits

Las versiones empleadas para el desarrollo de este proyecto han sido Eclipse Mars2 y Eclipse Neon.

7.2.2. Instalación

Una vez descargado procedemos a su instalación. Para ello ejecutamos el archivo descargado y seguimos los pasos.
Lo primero que debemos hacer es seleccionar la versión, que como ya se ha especificado es la JEE.

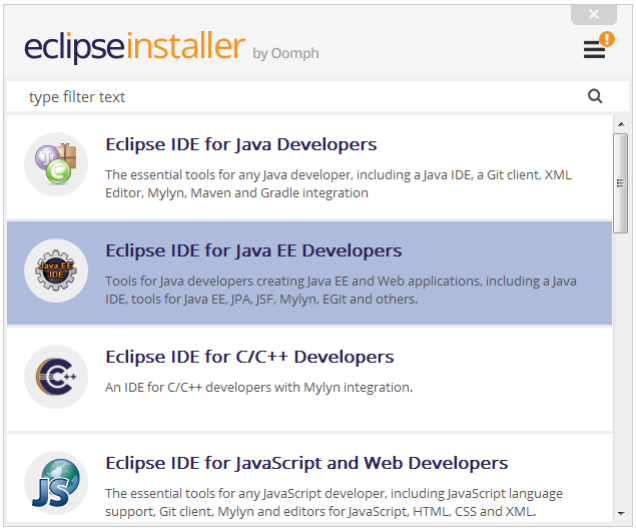

Figura 7.30: Eclipse - Selección de versión (JEE)

El siguiente paso será seleccionar el directorio de instalación.

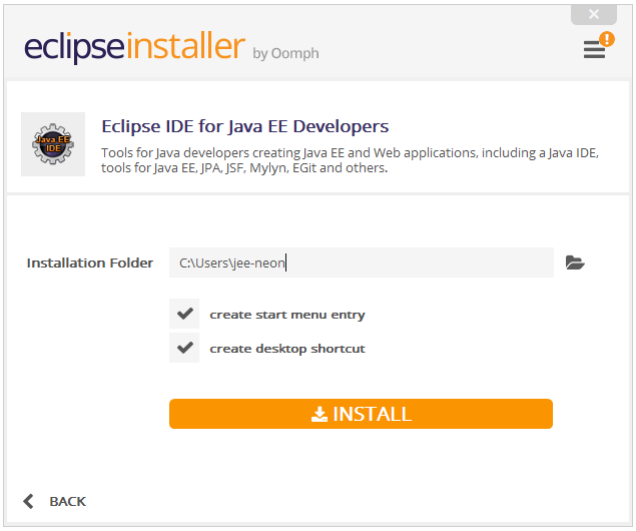

Figura 7.31: Eclipse - Directorio de instalación

Hacemos clic en instalar para comenzar la instalación....
Con esto tendremos ya Eclipse instalado y completamente operativo.

Capítulo 8

El proyecto

Ya tenemos todo preparado para comenzar nuestra aplicación. El primer paso es crear el proyecto. Se entiende que se tienen nociones básicas sobre Java y la utilización de Eclipse.

8.1. Creando el proyecto

El tipo de proyecto que utilizaremos para nuestra aplicación será de tipo "Maven". Maven es un proyecto de Apache (como Tomcat) que permite la gestión de proyectos de software. Basado en el concepto de un modelo de objeto de proyecto (POM), Maven puede gestionar la compilación y las dependencias de un proyecto a partir de una pieza central de la información, conocido como repositorio. Este repositorio será el que proveerá a nuestro proyecto de todas las librerías (dependencias) necesarias para su correcto funcionamiento [3].

Para crear el proyecto hacemos clic en File > New > Other y seleccionamos "Maven Project".

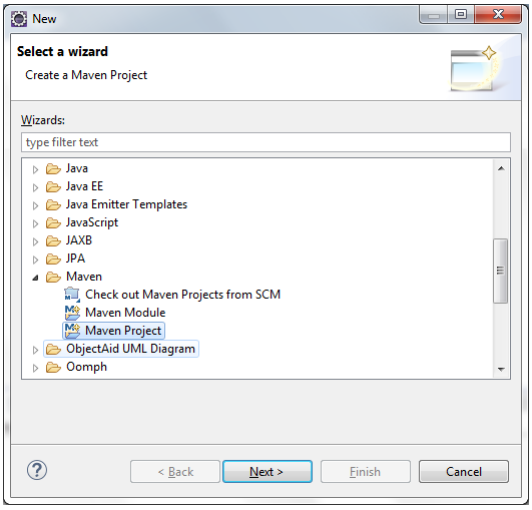

Figura 8.1: Creando el proyecto Maven

Seleccionamos el workspace deseado (normalmente en el que ya estamos) y hacemos clic en "Next".
A continuación seleccionamos el tipo de proyecto Maven, que será de tipo "webapp", porque es una aplicación web.

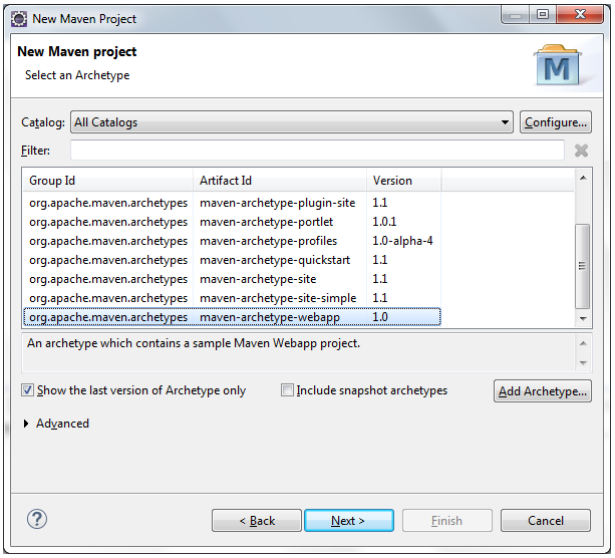

Figura 8.2: Maven "webapp"

El siguiente paso es darle un nombre. Para ello tendremos que especificar el "GroupId", que es el grupo de trabajo, y el "ArtifactId", que es el nombre del proyecto.

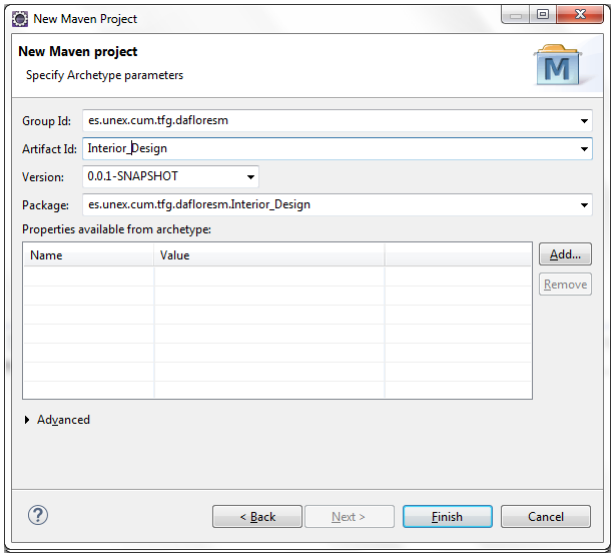

Figura 8.3: Especificando el nombre

Por último hacemos clic en "Finish" y nuestro proyecto será creado.

8.1.1. Corrección del error inicial

Es probable, dependiendo de la versión, que nada más crear el proyecto aparezca como que contiene errores. Esto es porque es un proyecto web al que todavía no le hemos dicho que servidor web va a utilizar. La solución de este problema la veremos cuando veamos la configuración del fichero pom.xml en el capítulo siguiente, donde agregaremos la dependencia de *javax.servlet*.

8.2. Estructura

La estructura del proyecto es la siguiente:

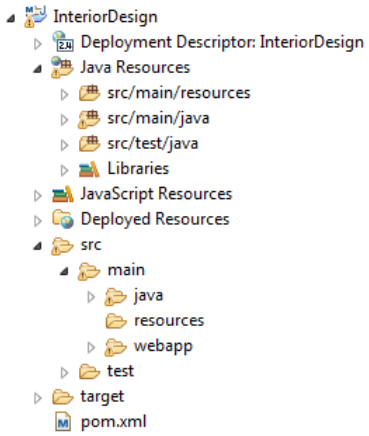

Figura 8.4: Estrcutrura del proyecto

Donde tenemos:

- **Java Resources**
  - **src/main/resources**: donde colocaremos los recursos externos que sean necesarios.
  - **src/main/java**: donde irá todo el código java.
  - **src/test/java**: donde irán las pruebas unitarias.
  - **Libraries**: donde encontraremos las librerías que descargue Maven al ser incluidas en el pom.xml.
- **src/main/webapp**: donde irá todo el contenido relacionado con los diseños web HTML, JSP, etc, así como CSS u otras cosas necesarias.
- **pom.xml**: archivo de configuración del proyecto donde especificaremos las dependencias.

8.3. Dependencias

Comenzaremos por agregar todas las dependencias necesarias. Como ya sabemos de antemano los requisitos del proyecto y hemos elaborado un diseño previo, podemos agregar inicialmente las dependencias necesarias. También podemos ir agregándolas a medida que las vayamos necesitando. En nuestro caso lo haremos de la primera forma y así tendremos este apartado cubierto.

Para nuestro proyecto tenemos tres tipos de dependencias principales: Servlet, reportes y conexión con base de datos. Para cada uno de los tipos debemos agregar las librerías que estimemos oportunas buscándolas en el repositorio oficial de maven e incluyéndolas en el fichero pom.xml. Además, Maven se descargará automáticamente otras dependencias que sean necesarias para la librerías que nosotros necesitamos, con lo que nos facilitará mucho el trabajo.

Estas dependencias pueden buscarse en el repositorio oficial. Solamente tendremos que introducir el nombre de la librería y seleccionar la versión correspondiente.

La web es: `https://mvnrepository.com/`

A continuación se muestra un cuadro con las dependencias utilizadas en el proyecto.

8.3.1. Agregando una dependencia al pom

Para agregar la dependencia únicamente tendremos que abrir el archivo pom.xml y copar la dependencia que habremos buscado en el repositorio de Maven. A continuación se muestra el ejemplo de cómo añadir la dependencia para el servlet.
Primero la buscamos en el repositorio:

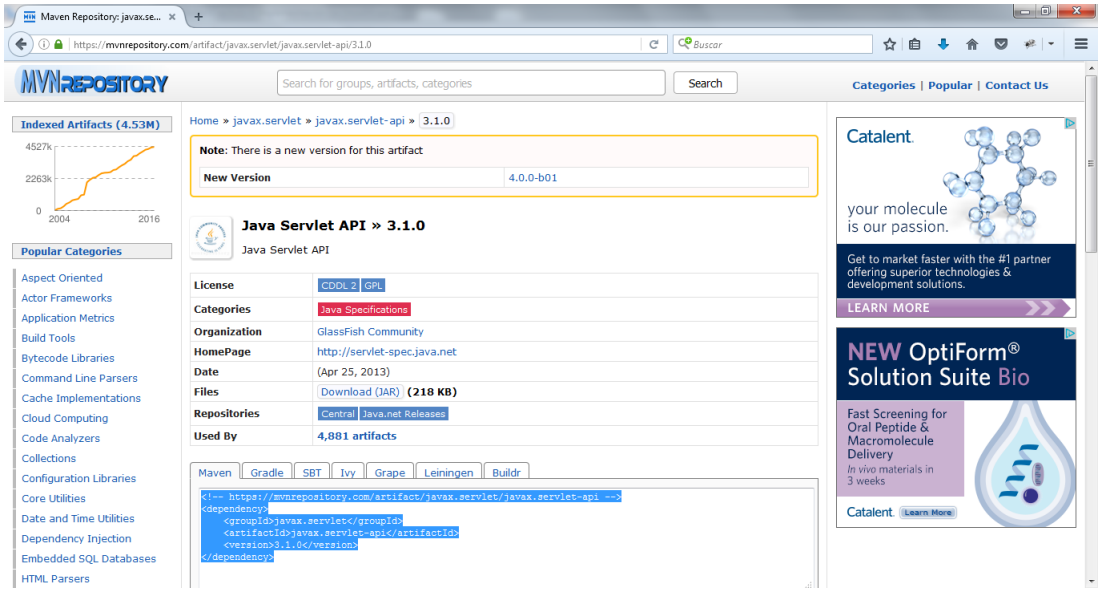

Figura 8.5: Dependencia javax.servlet

Tipo	GroupId	ArtifactId	Versión
Servlet	javax.servlet	javax.servlet-api	3.1.0
Reportes	org.jdesktop	beansbinding	1.2.1
	commons-beanutils	commons-beanutils	1.8.2
	commons-collections	commons-collections	3.2.1
	commons-digester	commons-digester	2.1
	commons-fileupload	commons-fileupload	1.3
	commons-io	commons-io	2.4
	com.google.code.maven-play-plugin.org.apache.commons	commons-javaflow	1590792
	commons-logging	commons-logging	1.1
	org.codehaus.groovy	groovy-all	2.0.1
	com.google.code.gson	gson	2.2.3
	com.lowagie	itext	2.1.7
	net.sf.jasperreports	jasperreports	5.5.0
	net.sf.jasperreports	jasperreports-fonts	5.6.1
	org.json	json	20140107
	org.jfree	jcommon	1.0.17
	org.jfree	jfreechart	1.0.14
	com.keypoint	png-encoder	1.5
	org.apache.poi	poi	3.7
Base de datos	mysql	mysql-connector-java	5.1.6
Otras	javax.mail	mail	1.5.0-b01
	junit	junit	3.8.1
	org.jasypt	jasypt	1.9.0

Cuadro 8.1: Dependencias del proyecto

Copiamos el texto que aparece en la parte inferior y lo pegamos en nuestro pom.xml:

```
InteriorDesign/pom.xml
4     <groupId>es.unex.cum.tfg.dafloresm</groupId>
5     <artifactId>InteriorDesign</artifactId>
6     <packaging>war</packaging>
7     <version>0.0.1-SNAPSHOT</version>
8     <name>InteriorDesign</name>
9     <url>http://maven.apache.org</url>
10    <dependencies>
11        <dependency>
12
13            <groupId>junit</groupId>
14            <artifactId>junit</artifactId>
15            <version>3.8.1</version>
16            <scope>test</scope>
17        </dependency>
18
19
20        <!-- https://mvnrepository.com/artifact/javax.servlet/javax.servlet-api -->
21        <dependency>
22            <groupId>javax.servlet</groupId>
23            <artifactId>javax.servlet-api</artifactId>
24            <version>3.1.0</version>
25        </dependency>
26
Overview | Dependencies | Dependency Hierarchy | Effective POM | pom.xml
```

Figura 8.6: Añadiendo javax.servlet al pom

Una vez guardemos el archivo, Maven comenzará a descargarse las dependencias.

Podemos comprobar que estén descargadas mirando dentro de Java Resources -> Libraries -> Maven Dependencies.

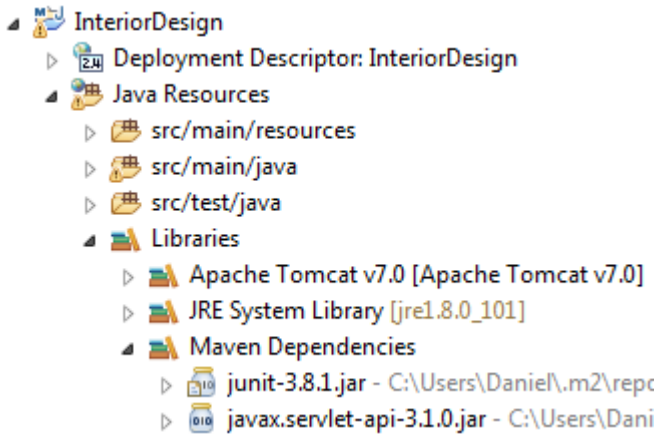

Figura 8.7: Comprobación de dependencias

Capítulo 9

Implementación

En este capítulo trataremos de explicar las implementaciones de los algoritmos más representativos de la aplicación, así como los ficheros de configuración. La carpeta referencia para nosotros será "src/main/webapp", que es donde se encuentran todos los ficheros de la web. Podemos diferenciar ficheros de varios tipos atendiendo a sus funciones:

- Programación del servidor
  - .java
  - Algunas funciones del servidor también estarán implementadas directamente en los ficheros ".jsp"
- Ficheros de configuración
  - .properties
  - .xml
- Páginas webs
  - .html
  - .jsp
- Diseño web
  - .css
- Programación web
  - .js

9.1. Configuración

9.1.1. Fichero de configuración: config.properties

Este es el fichero donde se guardará toda la configuración de la web. Dicha configuración consta de los datos de acceso a la base de datos, los datos del servidor SMTP y los de la aplicación de Facebook.

Podemos encontrar este fichero en la carpeta "WEB-INF" (siempre partiendo de "webapp").

```
faceKey=G9gwJSJbAy6f0+tLS0EQ34RSZQbba9K1IhVX73X95yfVPDBI8C5xJm5mHi//5aee
dbhost=localhost
faceId=1013189332112047
dbname=interiordesign
dbpassword=c97mqXf5dv/zooALoqPlRA\=\=
smtpPass=Ac55DbEv12DNZtBmABMQwpLKHeLsAAFo
smtpHost=smtp.proveedor.es
dbuser=root
smtpPort=puerto
smtpMail=tu@correo.es
```

Listado de código 9.1: Fichero config.properties

Como se puede observar en el código anterior, las contraseñas se encuentran cifradas. Esto se ha realizado utilizando la clase "StandardPBEStringEncryptor" de la librería "jasypt": `http://www.jasypt.org/api/jasypt/1.8/org/jasypt/encryption/pbe/StandardPBEStringEncryptor.html`

Para cifrar las contraseñas únicamente tendremos que crear una instancia de la clase anterior y añadirle una contraseña maestra para posteriormente cifrar nuestras contraseñas utilizando el método "encrypt(String arg0)":

```
StandardPBEStringEncryptor encryptor = new StandardPBEStringEncryptor();
encryptor.setPassword("ides17ucum");
pwd = encryptor.encrypt("contraseniaparaencriptar");
```

Listado de código 9.2: Cifrando contraseñas

Igualmente para descifrarlas utilizaremos el método "decrypt(String arg0)" de la misma clase. Con esto mantendremos la privacidad de las contraseñas en el fichero.

9.1.2. Context.xml

El fichero "context.xml", localizado en la carpeta "META-INF", dotará a la aplicación de un pool de conexiones para poder acceder a la base de datos.

```
<?xml version="1.0" encoding="UTF-8"?>
<Context reloadable="true"> <Resource name="jdbc/testdb" auth="Container" type="javax
    .sql.DataSource" maxTotal="10" maxIdle="5" maxWaitMillis="-1" username="root"
    password="" driverClassName="com.mysql.jdbc.Driver" url="jdbc:mysql://localhost
    :3306/interiordesign?autoReconnect=true" />
</Context>
```

Listado de código 9.3: context.xml

Esta configuración será escrita automáticamente al instalar la plataforma con los datos introducidos por el usuario, ya sea de manera automática o manual.

Para ello contamos con un método llamado "writeContext" en el archivo "InstalacionController", si la instalación se hace de manera automática, o en "InstalaciónManual", si se hace de forma manual.

```
public void writeContext(String fileName, String dbhost, String dbname, String uname,
     String pwd) {
    String context = "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n<Context reloadable
        =\"true\">\n\t<Resource name=\"jdbc/testdb\" auth=\"Container\" type=\"javax.
        sql.DataSource\" maxTotal=\"10\" maxIdle=\"5\" maxWaitMillis=\"-1\" username
        =\"" + uname + "\" password=\"" + pwd + "\" driverClassName=\"com.mysql.jdbc.
        Driver\" url=\"jdbc:mysql://" + dbhost + ":3306/" + dbname + "?autoReconnect=
        true\" />\n</Context>";
    BufferedWriter bufferedWriter = null;
    try {
        FileWriter fileWriter = new FileWriter(fileName);
        bufferedWriter = new BufferedWriter(fileWriter);
        bufferedWriter.write(context);
    } catch (Exception ex) {
        System.out.println("Error writing to file '" + fileName + "'");
        ex.printStackTrace();
    } finally {
        try {
            bufferedWriter.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Listado de código 9.4: Escritura del context.xml

9.2. Blueprint3d.js

El fichero "blueprint3d.js", contenido dentro de la carpeta "js", es el archivo más importante de nuestro proyecto. Este archivo contiene a la librería three.js, en su versión r71, además de todas las funciones necesarias para conseguir los efectos en 3D de nuestros diseños.

Es una librería de más de 50000 líneas código. La mayoría de ellas pertenecen exclusivamente a Three.js. El hecho de que sea una librería tan grande, nos lleva a que

sea una tarea bastante compleja el modificar cualquiera de los comportamientos que ofrece, o agregar uno nuevo. Una de las ideas futuras de este proyecto sería separar esta librería en dos, y dejar por un lado a Three.js y por otro el diseño 3D. De hecho, en una revisión posterior a la utilizada en este proyecto de "Blueprint3D" utilizada como base, ya se está realizando.

Examinemos a continuación algunas de sus características más interesantes así como algunas funcionalidades que han sido añadidas.

9.2.1. Scene

"Scene", o escena, es el nombre técnico que le damos al diseño. Para cada diseño dispondremos de un objeto de tipo "Scene" que será el objeto principal del diseño y que contendrá todos los planos y objetos.

```
var Scene = function(model, textureDir) {
    var scope = this;
    var model = model;
    var textureDir = textureDir;
    // activar o desactivar colisiones entre objetos
    this.colisiones = true;
    // activar o desactivar magnetizacion
    this.magnet;
    this.magnet_snap;
    // medir distancia
    this.medir = false;
    this.sprCount = 0;
    // mover a un punto
    this.mover = false;
    var scene = new THREE.Scene();
    var items = [];
    this.needsUpdate = false;
    // init item loader
    var loader = new THREE.JSONLoader();
    loader.crossOrigin = "";
    var item_types =
    {
        1: FloorItem,
        2: WallItem,
        3: InWallItem,
        7: InWallFloorItem,
        8: OnFloorItem,
        9: WallFloorItem
    };
    ...
}
```

Listado de código 9.5: Escena

En el fragmento de código anterior podemos ver algunas variables que son utilizadas para cambiar el comportamiento de la escena como las colisiones, la rotación o la magnetización. Además, podemos ver que tenemos 6 tipos de objetos posibles para poder incorporar a la escena, cada uno identificado por un número:

- FloorItem: objetos de tipo suelo. Objeto más utilizado (sillas, mesas, camas, etc)
- WallItem: objetos de tipo pared (cuadros)
- InWallItem: objetos dentro de la pared -flotante- (ventanas)
- InWallFloorItem: objetos dentro de la pared -pegado al suelo- (puertas)
- OnFloorItem: pegado al suelo (alfombras)
- WallFloorItem: (pared o suelo)

La escena cuenta además con métodos propios (getters y setters) y específicos para el control de los objetos. Uno de los métodos a destacar es el de agregar un objeto nuevo: "addItem".

9.2.1.1. Agregando un objeto

El método "addItem" nos permite agregar un objeto a la escena. Veamos cómo se comporta este método.

```
this.addItem = function(itemType, fileName, metadata, textureFill, position, rotation
    , scale, fixed) {
    itemType = itemType || 1;
    var loaderCallback = function(geometry, materials) {
        var item = new item_types[itemType](
            model,
            metadata,
            geometry,
            new THREE.MeshFaceMaterial(materials),
            position,
            rotation,
            scale
        );
        item.fixed = fixed || false;
        item.textureFill = textureFill;
        items.push(item);
        scope.add(item);
        item.initObject();
        var ruta = fileName;

        // Relleno el array de materiales que tienen textura
        item.materialsFill = new Array();
        var materials = item.material.materials;
        for (var i = 0; i < materials.length; i++){
            var mat = materials[i];
            var map = mat["map"];
            if (map != null){
                item.materialsFill.push(i);
            }
        }

        // Para cargar la textura
        if (metadata.model_texture!=null && metadata.model_texture!=""){
            var materials = item.material.materials;
            if (item.textureFill){
```

```
                for (var i = 0; i < materials.length; i++){
                    var mat = materials[i];
                    mat.map = THREE.ImageUtils.loadTexture( metadata.model_texture )
                    mat.map.minFilter = THREE.LinearFilter;
                    mat.needsUpdate = true;
                }
            } else {
                for (var i = 0; i < materials.length; i++){
                    var mat = materials[i];
                    var map = mat["map"];
                    if (map != null){
                        mat.map = THREE.ImageUtils.loadTexture( metadata.
                            model_texture )
                        mat.map.minFilter = THREE.LinearFilter;
                        mat.needsUpdate = true;
                    }
                }
            }
            item.scene.needsUpdate = true;
        }
        scope.itemLoadedCallbacks.fire(item);
    }
    scope.itemLoadingCallbacks.fire();
    loader.load(fileName, loaderCallback, textureDir);
}
```

Listado de código 9.6: Agregando un objeto al diseño

Una vez seleccionado un objeto de la lista, se ejecutará este método. Lo primero que se hace es crear un objeto (item) del tipo especificado, y que hemos visto en la sección anterior con la escena. A continuación se comprueban datos como la posición, la rotación o la escala.
El siguiente paso ha sido añadido como funcionalidad nueva. Se recorren todos los materiales de la pieza (una pieza puede tener más de un material), y se comprueba a cuál de ellos se le puede aplicar una textura (no todos los materiales pueden llevar una textura asociada). Esto nos permitirá en un futuro aplicar la misma textura sobre todos los materiales admitan o no una textura. A continuación se carga la textura por defecto que tiene esa pieza sobre cada uno de los materiales y finalmente se carga en la escena.
Con esto conseguimos cargar un objeto seleccionado y ponerle su textura asociada.
El método asociado a duplicar un objeto "duplicateItem" es muy similar al anterior en el que se sustituye el objeto seleccionado del buscador para añadirlo al diseño por un objeto ya existente.

9.2.2. Model

La escena está compuesta principalmente por un objeto de tipo "Model" que es el que contiene el plano principal del diseño.

```
var Model = function(textureDir) {
    var scope = this;
```

```
    this.floorplan = new Floorplan();
    this.scene = new Scene(scope, textureDir);
    ...
}
```

Listado de código 9.7: Model

Tiene además varios métodos interesantes que permiten importar y exportar el diseño completo en forma de texto plano serializado.

9.2.2.1. Importando el diseño

El siguiente fragmento de código nos permitirá importar un diseño desde un archivo de texto, que hayamos exportado previamente.

```
this.loadSerialized = function(data_json) {
    this.roomLoadingCallbacks.fire();
    data = JSON.parse(data_json)
    scope.newRoom(
        data.floorplan,
        data.items
    );
    scope.roomLoadedCallbacks.fire();
}
```

Listado de código 9.8: Importar diseño desde texto

9.2.2.2. Exportando el diseño

Para exportar un diseño ya creado utilizamos el siguiente método.

```
this.exportSerialized = function() {
    var items_arr = [];
    var objects = scope.scene.getItems();
    for ( var i = 0; i < objects.length; i++ ) {
        var object = objects[i];
        var src=null;
        for (var j = 0; j < object.material.materials.length; j++){
            var mat = object.material.materials[j];
            var map = mat["map"];
            if (map != null){
                src = map["sourceFile"];
                break;
            }
        }
        var textureMap="";
        if (src!=null){
            var res = src.split("/");
            textureMap="models/textures/"+res[res.length-1];
        }
        items_arr[i] = {
            itemId : object.metadata.itemId,
            item_name: object.metadata.itemName,
            item_type: object.metadata.itemType,
            model_url: object.metadata.modelUrl,
            // Siguiente linea aniadida para guardar la textura
            model_texture: textureMap,
            // Siguiente linea aniadida para aplicar la textura a toda la pieza o no
            textureFill: object.textureFill,
            xpos: object.position.x,
            ypos: object.position.y,
```

```
            zpos: object.position.z,
            rotation: object.rotation.y,
            scale_x: object.scale.x,
            scale_y: object.scale.y,
            scale_z: object.scale.z,
            fixed: object.fixed
        };
    }
    var room = {
        floorplan: (scope.floorplan.saveFloorplan()),
        items: items_arr
    };
    return JSON.stringify(room);
}
```

Listado de código 9.9: Exportando el diseño a un fichero de texto

La exportación del diseño es bien sencilla. Basta con iterar sobre todos los objetos de nuestro diseño e ir almacenando todas sus propiedades como la posición, la escala, si está bloqueado o no, la textura, etc. Finalmente se guardarán las habitaciones que componen el diseño, con sus respectivos planos.
Un ejemplo de fichero de texto con un diseño exportado sería el siguiente:


```
{"floorplan":{"corners":{"f90da5e3-9e0e-eba7-173d-eb0b071e838e":{"x"
    :-104.0130000000003,"y":331.7239999999996},"da026c08-d76a-a944-8e7b-096b752da9ed"
    :{"x":406.0189999999998,"y":331.7239999999996},"4e3d65cb-54c0-0681-28bf-
    bddcc7bdb571":{"x":406.0189999999998,"y":-178.308},"71d4f128-ae80-3d58-9bd2-711
    c6ce6cdf2":{"x":-104.0130000000003,"y":-178.308}},"walls":[{"corner1":"71d4f128-
    ae80-3d58-9bd2-711c6ce6cdf2","corner2":"f90da5e3-9e0e-eba7-173d-eb0b071e838e","
    frontTexture":{"url":"rooms/textures/wallmap.png","stretch":true,"scale":0},"
    backTexture":{"url":"rooms/textures/wallmap.png","stretch":true,"scale":0}},{"
    corner1":"f90da5e3-9e0e-eba7-173d-eb0b071e838e","corner2":"da026c08-d76a-a944-8
    e7b-096b752da9ed","frontTexture":{"url":"rooms/textures/wallmap.png","stretch":
    true,"scale":0},"backTexture":{"url":"rooms/textures/wallmap.png","stretch":true,
    "scale":0}},{"corner1":"da026c08-d76a-a944-8e7b-096b752da9ed","corner2":"4e3d65cb
    -54c0-0681-28bf-bddcc7bdb571","frontTexture":{"url":"rooms/textures/wallmap.png",
    "stretch":true,"scale":0},"backTexture":{"url":"rooms/textures/wallmap.png","
    stretch":true,"scale":0}},{"corner1":"4e3d65cb-54c0-0681-28bf-bddcc7bdb571","
    corner2":"71d4f128-ae80-3d58-9bd2-711c6ce6cdf2","frontTexture":{"url":"rooms/
    textures/wallmap.png","stretch":true,"scale":0},"backTexture":{"url":"rooms/
    textures/wallmap.png","stretch":true,"scale":0}}],"wallTextures":[],"
    floorTextures":{},"newFloorTextures":{}},"items":[{"itemId":"10","item_name":"
    Puerta cerrada","item_type":7,"model_url":"models/js/closed-door28x80_baked.js","
    model_texture":"models/textures/closed-door28x80_baked.png","textureFill":false,"
    xpos":-22.31710493729699,"ypos":110.80000022010701,"zpos":-177.80799865722656,"
    rotation":0,"scale_x":1,"scale_y":1,"scale_z":1,"fixed":false},{"itemId":"2","
    item_name":"Mesa central","item_type":1,"model_url":"models/js/
    BlakeAvenuejoshuatreecheftable.js","model_texture":"models/textures/
    pine_wood_coloured.jpg.jpg","textureFill":false,"xpos":188.52824495728404,"ypos"
    :39.5,"zpos":66.05361045156104,"rotation":-1.5707963267948966,"scale_x":1.0017,"
    scale_y":0.9938981411469876,"scale_z":1.8022561937375934,"fixed":false},{"itemId"
    :"7","item_name":"Sillón 3 plazas","item_type":1,"model_url":"models/js/cb-
    rochelle-gray_baked.js","model_texture":"models/textures/cb-rochelle-gray_baked.
    png","textureFill":false,"xpos":187.9573312162814,"ypos":42.54509923821,"zpos"
    :-126.04946289725086,"rotation":0,"scale_x":1,"scale_y":1,"scale_z":1,"fixed":
    false},{"itemId":"7","item_name":"Sillón 3 plazas","item_type":1,"model_url":"
    models/js/cb-rochelle-gray_baked.js","model_texture":"models/textures/cb-rochelle
    -gray_baked.png","textureFill":false,"xpos":352.8104288144992,"ypos"
    :42.54509923821,"zpos":75.42328654988358,"rotation":-1.5707963267948966,"scale_x"
    :1,"scale_y":1,"scale_z":1,"fixed":false},{"itemId":"5","item_name":"Mueble
    televisor","item_type":1,"model_url":"models/js/cb-clapboard_baked.js","
    model_texture":"models/textures/cb-clapboard_baked.png","textureFill":false,"xpos
    ":-72.79494114362804,"ypos":67.88999754395999,"zpos":69.26805862120449,"rotation"
    :1.5707963267948966,"scale_x":1,"scale_y":1,"scale_z":1,"fixed":false},{"itemId":
```

```
"21","item_name":"Tierra","item_type":1,"model_url":"models/js/tierra.js","
model_texture":"models/textures/tierra.jpg","textureFill":false,"xpos"
:350.41314209202653,"ypos":50,"zpos":276.42753818560794,"rotation":0,"scale_x":1,
"scale_y":1,"scale_z":1,"fixed":false},{"itemId":"18","item_name":"Póster NY","
item_type":2,"model_url":"models/js/nyc-poster2.js","model_texture":"models/
textures/nyc2.jpg","textureFill":false,"xpos":181.04634200146648,"ypos"
:155.41330445994436,"zpos":-172.05424833972592,"rotation":0,"scale_x":1,"scale_y"
:1,"scale_z":1,"fixed":false},{"itemId":"20","item_name":"Lámpara","item_type":1,
"model_url":"models/js/ore-3legged-white_baked.js","model_texture":"models/
textures/ore-3legged-white_baked.png","textureFill":false,"xpos"
:378.68796666335794,"ypos":72.163997943445,"zpos":-143.83573939982017,"rotation"
:0,"scale_x":1,"scale_y":1,"scale_z":1,"fixed":false}]}
```


Listado de código 9.10: Diseño exportado (diseño binario)

Si a continuación importásemos el fichero de texto anterior conseguiríamos el siguiente diseño:

Figura 9.1: Diseño importado

Se debe tener en cuenta que al importar un diseño se guardan todos sus objetos. Si un objeto es modificado o eliminado por el administrador de la plataforma, el diseño no se importará correctamente.

9.2.3. ThreeController

El "ThreeController" es el objeto que controla todo lo relacionado con la librería Three.js. Este objeto es el encargado de inicializar las cámaras, controlar las vistas, la escena, el modelo o controlar los estados de los objetos, entre otros.

```
var ThreeController = function(three, model, camera, element, controls, hud) {
    var scope = this;
    this.enabled = true;
    //medir distancia
    var sprite1 = null;
    var sprite2 = null;
    var sprCount = 0;

    var three = three;
    var model = model;
    var scene = model.scene;
    var element = element;
    var camera = camera;
    var controls = controls;
    var hud = hud;
    var plane;

    // ground plane used for intersection testing
    var mouse;
    var intersectedObject;
    var mouseoverObject;
    var selectedObject;
    var mouseDown = false;
    var mouseMoved = false;
    // has mouse moved since down click
    var rotateMouseOver = false;
    var states = {
        UNSELECTED: 0, // no object selected
        SELECTED: 1, // selected but inactive
        DRAGGING: 2, // performing an action while mouse depressed
        ROTATING: 3,  // rotating with mouse down
        ROTATING_FREE: 4, // rotating with mouse up
        PANNING: 5
    };
    var state = states.UNSELECTED;
    ...
}
```

Listado de código 9.11: ThreeController

9.2.4. Room

El objeto de tipo "Room" representa las habitaciones del diseño. Cada habitación está compuesta por su plano. Una habitación es creada cuando se tienen al menos 3 paredes cerradas entre sí y que den lugar a un suelo. Con ayuda de las paredes, se pueden sacar los puntos que delimitan la habitación (interiorCorners).

```
var Room = function(floorplan, corners) {
    var scope = this;
    // ordered CCW
    var floorplan = floorplan;
    this.corners = corners;
    this.interiorCorners = [];
    this.edgePointer = null;
```

```
    // floor plane for intersection testing
    this.floorPlane = null;
    this.customTexture = false;
    var defaultTexture = {
        url: "rooms/textures/hardwood.png",
        scale: 400
    }
    ...
}
```

Listado de código 9.12: Room

9.2.5. Floorpanel

El "floorpanel" o vista del plano, es la parte donde se realiza el plano de las habitaciones. Para crear estos planos se hace uso de las paredes, que se crearán, moverán, eliminarán o cambiarán según el usuario lo necesite. Para ello contamos con una función denominada "Floorplanner", que contendrá todos los atributos y métodos asociados a todas las operaciones que necesitemos hacer con el plano.

```
var Floorplanner = function(canvas, floorplan) {
    var scope = this;
    var floorplan = floorplan;
    this.modes = {
        MOVE: 0,
        DRAW: 1,
        DELETE: 2
    };
    this.mode = 0;
    var mouseDown = false;
    var mouseMoved = false;
    this.activeWall = null;
    this.activeCorner = null;
    this.originX = 0;
    this.originY = 0;
    // how much will we move a corner to make a wall axis aligned (cm)
    var snapTolerance = 25;
    ...
}
```

Listado de código 9.13: Floorpanel

Además, la vista del plano se apoya en la función "FloorplannerView", que es la que permitirá la previsualización en tiempo real del plano en nuestro Canvas.

```
var FloorplannerView = function(floorplan, viewmodel, canvas) {
    var scope = this;
    var floorplan = floorplan;
    var viewmodel = viewmodel;
    var canvas = canvas;
    var canvasElement = document.getElementById(canvas);
    var context = canvasElement.getContext('2d');

    // grid parameters
    var gridSpacing = 20;

    // pixels
    var gridWidth = 1;
    var gridColor = "#f1f1f1";
```

```
    // room config
    var roomColor = "#f9f9f9";

    // wall config
    var wallWidth = 5;
    var wallWidthHover = 7;
    var wallColor = "#dddddd"
    var wallColorHover = "#008cba"
    var edgeColor = "#888888"
    var edgeColorHover = "#008cba"
    var edgeWidth = 1
    ...
}
```

Listado de código 9.14: FloorplannerView

Una de las modificaciones que se han realizado en este apartado ha sido cambiar la unidad de medida de pies a metros.

```
function cmToFeet(cm) {
    /* // Codigo antiguo indicando los pies
    * var realFeet = ((cm*0.393700) / 12);
    * var feet = Math.floor(realFeet); var
    * inches = Math.round((realFeet - feet) * 12);
    * return feet + "'" + inches +   * '"';
    */
    var meters = Math.floor(cm) / 100;  return meters + " m";
}
```

Listado de código 9.15: Pies a metros

9.2.6. FloorItem

Los objetos de tipo "FloorItem" son los que corresponden con los objetos o piezas de los diseños que están pegados al suelo, como sillas, mesas, alfombras etc.

```
var FloorItem = function(three, metadata, geometry, material, position, rotation,
    scale) {
    Item.call(this, three, metadata, geometry, material, position, rotation, scale);
};
```

Listado de código 9.16: FloorItem

Además, asociados a esta variable (FloorItem), se van creando dinámicamente las funciones que este tipo de objetos puede utilizar. Algunas de las funciones más representativas son:

- FloorItem.prototype.placeInRoom = function() {...}: posiciona el objeto en la habitación
- FloorItem.prototype.moveToPosition = function(vec3, intersection, keys) {...}: mueve el objeto a una posición determinada
- FloorItem.prototype.isValidPosition = function(vec3) {...}: comprueba si la posición del objeto es válida respecto de las paredes y a otros objetos (colisiones)

- ...

9.2.7. Item

El objeto de tipo "Item" es el objeto genérico que será asociado a cualquier objeto de la escena, sea de suelo, pared, etc.

```
var Item = function(model, metadata, geometry, material, position, rotation, scale) {
    this.model = model;
    this.scene = model.scene;
    this.errorGlow = new THREE.Mesh();
    this.hover = false;
    this.selected = false;
    this.highlighted = false;
    this.error = false;
    this.emissiveColor = 0x444444;
    this.errorColor = 0xff0000;

    //Distancia con las paredes de los cuatro planos del objeto
    this.northWall;
    this.southWall;
    this.westWall;
    this.eastWall;

    //Array para guardar las lineas
    this.north = new Array();
    this.east = new Array();
    this.south = new Array();
    this.west = new Array();

    //Mostar u ocultar lineas auxiliares
    this.lineWalls = false;

    //Aplicar textura a toda la pieza
    this.textureFill = false;

    //Array para guardar las posiciones de los materiales iniciales que contienen
        textura
    this.materialsFill;
    this.metadata = metadata;
    this.resizable = metadata.resizable;
    THREE.Mesh.call(this, geometry, material);
    this.castShadow = true;
    this.receiveShadow = false;

    // does this object affect other floor items
    this.obstructFloorMoves = true;
    if (position) {
        this.position.copy(position);
        this.position_set = true;
    } else {
        this.position_set = false;
    }
    // show rotate option in context menu
    this.allowRotate = true;
    this.fixed = false;
    // dragging     this.dragOffset = new THREE.Vector3();
    // center in its boundingbox
    this.geometry.computeBoundingBox();
    this.geometry.applyMatrix( new THREE.Matrix4().makeTranslation(
        - 0.5 * ( this.geometry.boundingBox.max.x + this.geometry.boundingBox.min.x )
            ,
        - 0.5 * ( this.geometry.boundingBox.max.y + this.geometry.boundingBox.min.y )
            ,
        - 0.5 * ( this.geometry.boundingBox.max.z + this.geometry.boundingBox.min.z )
```

```
    ) );
    this.geometry.computeBoundingBox();
    this.halfSize = this.objectHalfSize();
    if (rotation) {
        this.rotation.y = rotation;
    }
    if (scale != null) {
        this.setScale(scale.x, scale.y, scale.z);
    }
};
```

Listado de código 9.17: Item

Cuenta además con funciones asociadas que permiten cambiar el comportamiento de estos objetos:

- Item.prototype.moveKeyUp = function (intersection){...}: al pulsar la flecha arriba del teclado, el objeto seleccionado se moverá hacia arriba en la escena. Esta función será replicada para el resto de direcciones (abajo, derecha e izquierda)
- Item.prototype.remove = function() {...}: elimina el objeto seleccionado de la escena
- Item.prototype.setTexture = function(texture, callback){...}: aplica, sobre el objeto seleccionado, la textura sobre la que hayamos hecho clic. Esta función será detallada más adelante
- Item.prototype.resize = function(height, width, depth) {...}: cambia el tamaño del objeto
- ...

9.2.8. Wall

El objeto de tipo "Wall" será el que haga la función de las paredes. Cuando creamos un plano, este a su vez está compuesto por objetos de tipo "Wall".

```
var Wall = function(start, end) {
    this.id = getUuid();
    var scope = this;
    var start = start;
    var end = end;
    this.thickness = 10;
    this.height = 250;

    // front is the plane from start to end
    // these are of type HalfEdge
    this.frontEdge = null;
    this.backEdge = null;
    this.orphan = false;
```

```
    // items attached to this wall
    this.items = [];
    this.onItems = [];
    var moved_callbacks = JQUERY.Callbacks();
    var deleted_callbacks = JQUERY.Callbacks();
    var action_callbacks = JQUERY.Callbacks();
    var defaultTexture =
    {
        url: "rooms/textures/wallmap.png",
        stretch: true,
        scale: 0
    }
    this.frontTexture = defaultTexture;
    this.backTexture = defaultTexture;
    ...
}
```

Listado de código 9.18: Wall

9.2.9. WallItem

Estos objetos corresponden a los objetos que están pegados en las paredes, como por ejemplo, los cuadros.

```
var WallItem = function(model, metadata, geometry, material, position, rotation,
    scale) {
    Item.call(this, model, metadata, geometry, material, position, rotation, scale);

    this.allowRotate = false;

    // TODO:
    // This caused a huge headache.
    // HalfEdges get destroyed/created every time floorplan is edited.
    // This item should store a reference to a wall and front/back,
    // and grab its edge reference dynamically whenever it needs it.
    this.currentWallEdge = null;

    // used for finding rotations
    this.refVec = new THREE.Vector2(0, 1.0);
    this.wallOffsetScalar = 0;
    this.sizeX = 0;
    this.sizeY = 0;

    this.addToWall = false;
    this.boundToFloor = false;

    this.frontVisible = false;
    this.backVisible = false;
};
```

Listado de código 9.19: WallItem

Al igual que sucede con el “FloorItem”, los objetos de tipo “WallItem” también cuentan con funciones asociadas que determinan su comportamiento. Algunas de estas funciones son:

- WallItem.prototype.closestWallEdge = function() {...}: determina la arista más cercana

- WallItem.prototype.updateSize = function() {...}: actualiza el tamaño del objeto
- WallItem.prototype.changeWallEdge = function(wallEdge) {...}: cambia el objeto de arista
- ...

9.2.10. Otras funciones

Hemos visto algunos de los tipos de objetos que se crean cuando realizamos algún diseño. Dentro de Blueprint3D hay muchísimas más funciones que nos permiten modificar completamente el comportamiento de la escena, el diseño, los objetos, etc.
En este apartado se enumerarán algunas de las funciones que han sido añadidas a Three.js, como requisito del cliente, y que proporcionan funcionalidades añadidas a las que ya teníamos.

9.2.10.1. Medir la distancia entre dos puntos

Dentro de la función que controla el clic del ratón (mouseDownEvent (event)), se ha implementado esta funcionalidad.
Su funcionamiento es sencillo. Cuando se hace clic sobre el botón de "Medir distancia entre dos puntos", se dibujan dos partículas invisibles dentro de la escena que permitirán calcular dicha distancia, tendiendo en cuenta la posición de la cámara, y que estamos en un entorno 3D. Los pasos a seguir son:

1. Clic en el botón "Medir distancia entre dos puntos"
2. Clic en cualquier parte de la escena que determinará el punto de origen
3. Clic en cualquier parte de la escena que determinará el punto final
4. Se calcula la distancia y se muestra

Para desactivar esta función haremos clic sobre el mismo botón.

```
function mouseDownEvent( event ) {
    ...
    // medir distancia entre dos puntos
    if (scene.medir && scene.sprCount > 0){
        var particleMaterial;
        particleMaterial = new THREE.SpriteMaterial(
        {
            color: 0x000000,
            program: function ( context ) {
                context.beginPath();
                context.arc( 0, 0, 0.5, 0, PI2, true );
```

```
                context.fill();
            }
        } );

        var raycaster;
        var mouse;

        raycaster = new THREE.Raycaster();
        mouse = new THREE.Vector2();
        var viewerInnerWidth = $("#viewer").innerWidth();
        var viewerInnerHeight = $("#viewer").innerHeight();
        var topHeight = $("#viewer-header").height();
        var leftWidth = $("#side-menu").width();
        leftWidth += 33.33333;

        mouse.x = ( (event.clientX - leftWidth) / viewerInnerWidth ) * 2 - 1;
        mouse.y = - ( (event.clientY - topHeight) / viewerInnerHeight ) * 2 + 1;

        raycaster.setFromCamera( mouse, camera );
        var intersects = raycaster.intersectObjects( scene.getScene().children );
        if ( intersects.length > 0 ) {
            if (scene.sprCount == 1){
                sprite1 = new THREE.Sprite( particleMaterial );
                sprite1.position.copy( intersects[ 0 ].point );
                sprite1.scale.x =sprite1.scale.y = 1;
                scene.add( sprite1 );
                scene.sprCount+=1;

                $("#measureInfo").val("Distancia = '?'");
            }else if (scene.sprCount == 2){
                sprite2 = new THREE.Sprite( particleMaterial );
                sprite2.position.copy( intersects[ 0 ].point );
                sprite2.scale.x = sprite2.scale.y = 1;
                scene.add( sprite2  );

                var dx = sprite1.position.x - sprite2.position.x;
                var dy = sprite1.position.y - sprite2.position.y;
                var dz = sprite1.position.z - sprite2.position.z;
                var dis = Math.sqrt( dx * dx + dy * dy + dz * dz );
                dis = Math.round(dis * 100.00/100.00)
                if (dis > 100){
                    dis = dis / 100;
                    $("#measureInfo").val("Distancia = " + dis + " m");
                } else{
                    $("#measureInfo").val("Distancia = " + dis + " cm");
                }
                scene.remove(sprite1);
                scene.remove(sprite2);
                scene.sprCount = 1;
            } else {
                scene.sprCount+=1;
            }
        }
    }
    ...
}
```

Listado de código 9.20: Calcular distancia entre dos puntos

9.2.10.2. Mover objeto a un punto

También se ha implementado un método que permite, dado un objeto seleccionado, moverlo a otro punto con un simple clic. Este método, al igual que el anterior,

también se ha implementado dentro del manejador del clic de ratón (mouseDownEvent (event)).
El principio del funcionamiento de este método es el mismo que el anterior. Cuando seleccionamos un objeto, en el menú lateral izquierdo encontramos el botón correspondiente para moverlo a un punto de la escena. Si pulsamos este botón, se activa una variable que nos permitirá entrar en la función. Los pasos a seguir son:

1. Seleccionar objeto
2. Pulsar botón para mover el objeto ("Mover pieza")
3. Seleccionar un punto de la escena para moverlo

```
function mouseDownEvent( event ) {
    ...
    // mover objeto a un punto
    if (scene.mover){
        var raycaster;
        var mouse;
        raycaster = new THREE.Raycaster();
        mouse = new THREE.Vector2();
        var viewerInnerWidth = $("#viewer").innerWidth();
        var viewerInnerHeight = $("#viewer").innerHeight();
        var topHeight = $("#viewer-header").height();
        var leftWidth = $("#side-menu").width();
        leftWidth += 33.33333;
        mouse.x = ( (event.clientX - leftWidth) / viewerInnerWidth ) * 2 - 1;
        mouse.y = - ( (event.clientY - topHeight) / viewerInnerHeight ) * 2 + 1;
        raycaster.setFromCamera( mouse, camera );
        var intersects = raycaster.intersectObjects( scene.getScene().children );
        if ( intersects.length > 0 ) {
            return ( intersects[ 0 ].point );
        }   else {
            return null;
        }
    }
    ...
}
```

Listado de código 9.21: Mover objeto a un punto de la escena

Debemos tener en cuenta que al mover un objeto de esta manera, no se tendrán en cuenta las colisiones ni si es una posición válida, por lo que un objeto puede ser movido fuera de la habitación, o quedar en medio de una pared.

9.2.10.3. Cambiar la textura de un objeto

Otro de los requisitos del cliente, es que a una pieza se le pudiera aplicar cualquier textura. Para ello se ha implementado la siguiente función.
Dada una pieza seleccionada, al hacer clic sobre alguna de sus texturas asociadas se llama a esta función. Su funcionamiento es el siguiente:

1. Se comprueba si está seleccionada la opción para aplicar la textura a toda la pieza, independientemente de si los materiales aceptan textura o no (propiedad "map")

   a) Si la opción está activada, se recorren todos los materiales de la pieza y se aplica la textura

   b) Si no, solamente se aplican las texturas en los materiales originales, que son los que admiten texturas

```
Item.prototype.setTexture = function(texture, callback){
    ...
    var materials = this.material.materials;
    if (this.textureFill){
        //Textura para toda la pieza
        for (var i = 0; i < materials.length; i++){
            var mat = materials[i];
            mat.map = THREE.ImageUtils.loadTexture( texture.url )
            mat.map.minFilter = THREE.LinearFilter;
            mat.needsUpdate = true;
        }
    } else {
        //Textura para los materiales que contengan la propiedad 'map'
        // Elimino todas las texturas
        for (var i = 0; i < materials.length; i++){
            var mat = materials[i];
            mat.map = null;
            mat.needsUpdate = true;
        }
        // Pongo la textura en los materiales originales
        for (var i = 0; i < this.materialsFill.length; i++){
            var mat = materials[this.materialsFill[i]];
            mat.map = THREE.ImageUtils.loadTexture( texture.url )
            mat.map.minFilter = THREE.LinearFilter;
            mat.needsUpdate = true;
        }
    }
    this.scene.needsUpdate = true;
}
```

Listado de código 9.22: Cambiar textura a un objeto

9.2.10.4. Calcular la distancia entre los muros

Este requisito fue uno de los últimos que se implementó. Con él se pretende, dado un objeto seleccionado, calcular la distancia hacia los muros norte, sur, este y oeste. Para entender esto, partiremos de que todos los objetos constan de 4 planos, independientemente de su forma. Identificaremos estos planos por la posición en la que se encuentran en la pieza: norte, sur, este y oeste.
Mostraremos el proceso mediante imágenes para comprenderlo mejor

- Para cada plano se describe una recta perpendicular desde su punto inicial, y otra desde el punto final.

- Norte: verde
- Sur: rosa
- Este: azul
- Oeste: amarillo

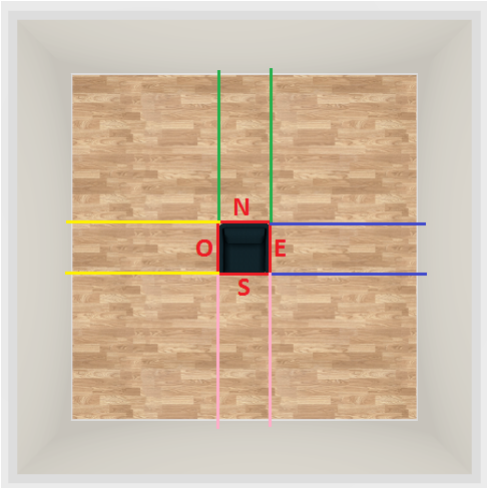

Figura 9.2: Distancia entre muros. Definiendo las líneas auxiliares

- Para cada plano, se calcula con que muro corta cada una de sus dos líneas.
  - Puede ocurrir que cada línea corte un muro y la otra con otro si tenemos la pieza girada. En este ejemplo, las dos cortarán el mismo muro
  - También puede ocurrir que dos planos corten el mismo muro. En el ejemplo cada plano corta con un muro para simplificar la explicación
- Una vez se ha calculado la distancia con la que corta cada línea, nos quedamos con la más corta para cada plano. De esta manera tendremos la mínima distancia hasta el muro

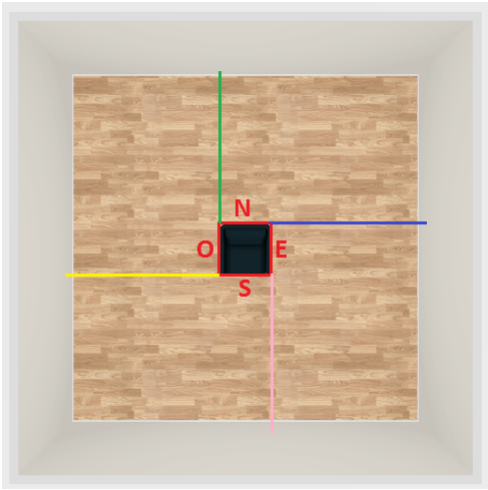

Figura 9.3: Distancia entre muros. Calculando mínima distancia

- Finalmente obtenemos la longitud de cada una de las líneas, y esa será la distancia que haya hacia los muros. Esta distancia será mostrada en los cuadros de texto correspondientes a la pieza en el menú lateral izquierdo

Esta funcionalidad nos permite también dejar una determinada distancia entre la pieza y un muro. Para ello bastará con editar la distancia del cuadro de texto correspondiente.
Adicionalmente a las líneas anteriores, se definen líneas desde el centro de la pieza, hasta cada una de las líneas auxiliares

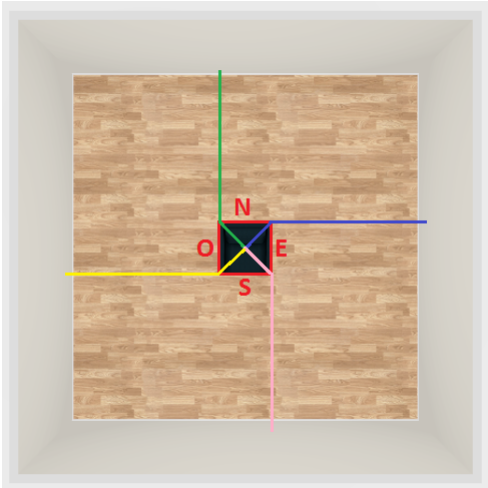

Figura 9.4: Distancia entre muros. Líneas interiores

Estas líneas interiores nos permitirán conservar la referencia de rotación y posición de la pieza respecto a las paredes, a la hora de moverlas.

Supongamos que queremos dejar una distancia de 2 metros entre la pieza y el muro superior (norte). El proceso que se sigue es el siguiente.

- Se al seleccionar la pieza, se definen sus líneas auxiliares
- Cambiamos la distancia del campo de texto "Norte" a 200 (200 centímetros = 2 metros), para moverla
- Desde el centro de la pieza se pinta otra línea (blanca), no visible para el usuario, paralela a la correspondiente a la línea auxiliar del plano referencia, en nuestro caso el norte (verde). Para ello se calcula la pendiente de dicha línea para definir la nueva

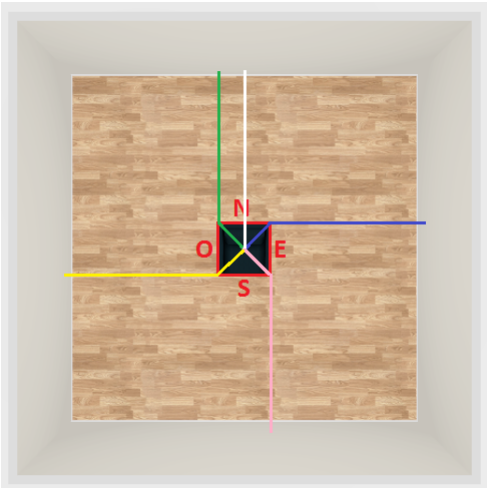

Figura 9.5: Distancia entre pieza y muro. Línea auxiliar desde el centro

- A continuación se calcula la nueva distancia, dada por el usuario, y se pinta otra línea (negra), paralela a la primera que pintamos desde el centro y que cortará con la anterior. Al igual que en el paso anterior, se hará uso de la pendiente para calcular la línea

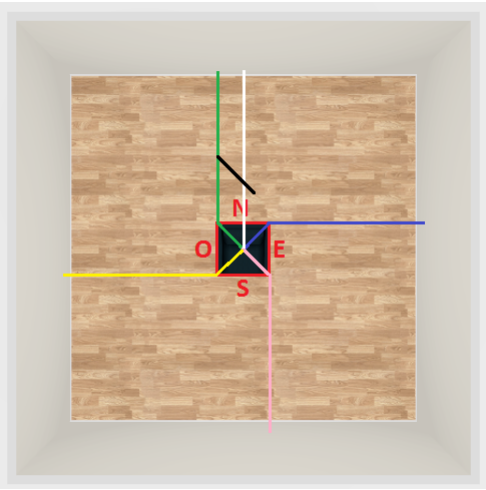

Figura 9.6: Distancia entre pieza y muro. Línea con la nueva distancia dada

- Calculando el punto en el que se cortan las dos líneas anteriores (negra y blanca), tendremos el nuevo centro de la pieza (naranja y verde), que estará a la distancia indicada por el usuario

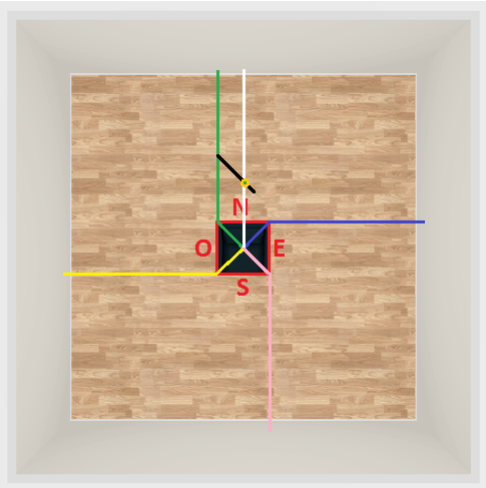

Figura 9.7: Distancia entre pieza y muro. Nuevo centro de la pieza

- El último paso será mover la pieza al punto calculado en el punto anterior

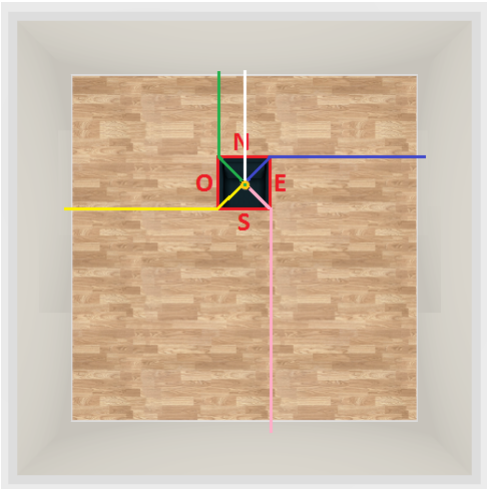

Figura 9.8: Distancia entre pieza y muro. Pieza en la nueva posición

Este proceso será el que se siga para mover la pieza en relación a los demás muros: sur, este y oeste.

Estos son algunos de los algoritmos más representativos de nuestra aplicación, aunque hay muchísimos más a tener en cuenta a la hora de realizar futuras modificaciones. Si se desea conocer más detalles sobre la implementación se recomienda que se revise el fichero "blueprint3d.js", que como se ha dicho al principio de este capítulo, es el que contiene la parte más importante de la implementación.

Capítulo 10

Instalación

La instalación de la plataforma Interior Design puede ser realizada de dos formas:

- Automáticamente a través del instalador
- Manualmente

Se recomienda realizar a través del instalador para facilitar los pasos. En el caso de que este método de algún problema, también puede realizarse manualmente. Adicionalmente, se necesita que se haya instalado previamente:

- XAMPP (Apache, MySQL y Tomcat)

10.1. Antes de empezar

Antes de comenzar la instalación tenemos que tener a mano los datos de acceso a:

- **Tomcat**: cuando instalamos Tomcat (o XAMPP), en el fichero "tomcat-users.xml" definimos el usuario y la contraseña del usuario de Tomcat que tiene acceso al "manager_gui" (Consultar 7.1.6).
- **MySQL**: usuario que tenga privilegios de creación y modificación de bases de datos en nuestro MySQL. Por defecto, y si no decimos lo contrario durante la instalación, suele ser el usuario "root" y sin contraseña.

También deberemos iniciar los servidores (Consultar 7.1.4.3):

- **Apache**
- **MySQL**
- **Tomcat**

Una vez tenemos las credenciales necesarias, y los servidores iniciados, podemos comenzar la instalación.

10.2. Automática

Se ha implementado además un sencillo instalador que facilita la instalación de la plataforma. Está basado en el instalador del conocido gestor de contenidos Wordpress.

10.2.1. Desplegar la aplicación

El primer paso es desplegar la aplicación en nuestro Tomcat, para lo que es necesario iniciar tanto el servicio de Apache como el del propio Tomcat en XAMPP.
Para acceder a Tomcat abriremos el navegador de Internet e introduciremos la siguiente url:

- `http://localhost:8080/`

Accediendo a "localhost" accedemos al servidor Apache, si además especificamos el puerto 8080, accederemos a Tomcat.

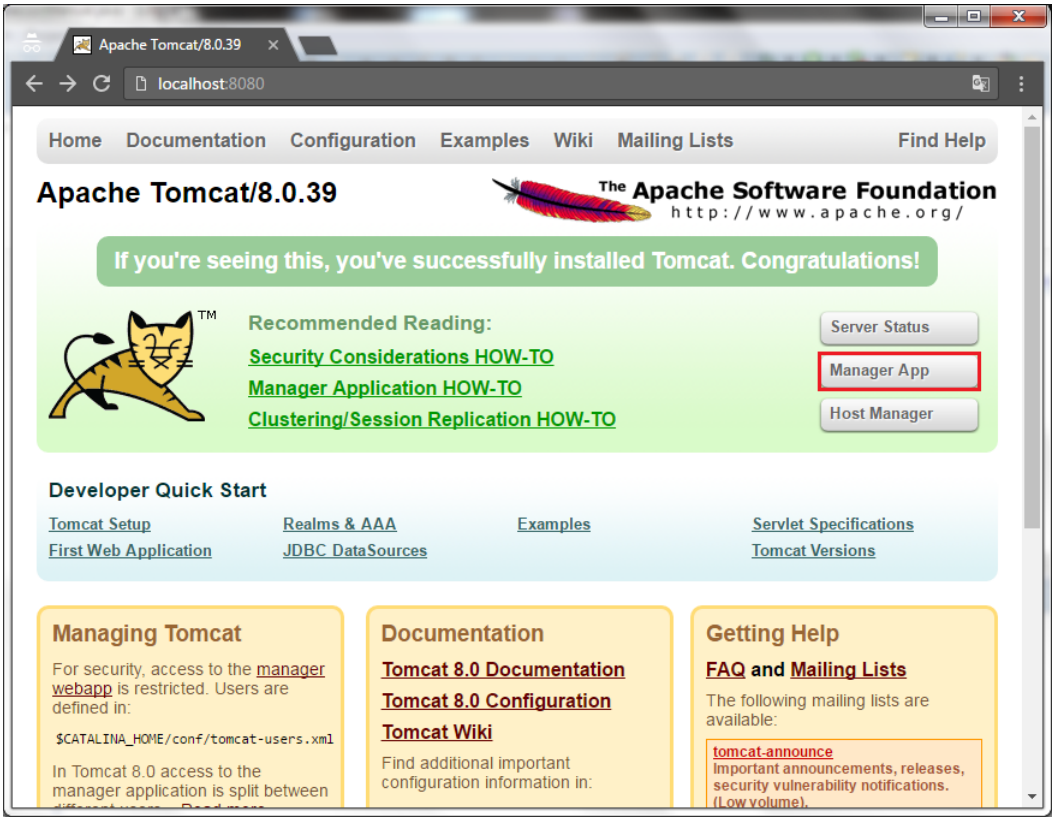

Figura 10.1: Accediendo a Tomcat

A continuación accedemos a la opción "Manager App", que nos permitirá gestionar todas las aplicaciones que tengamos desplegadas. Para entrar en esta configuración, tendremos que introducir el usuario y contraseña que habremos tenido que especificar en el fichero "tomcat-users.xml" de Tomcat cuando instalamos XAMPP. Si no estás usando XAMPP, deberás ir a la carpeta donde esté instalado Tomcat y modificar dicho archivo.

Una vez estemos en el panel de gestión, seleccionaremos nuestro archivo "InteriorDesign.war" y lo desplegaremos.

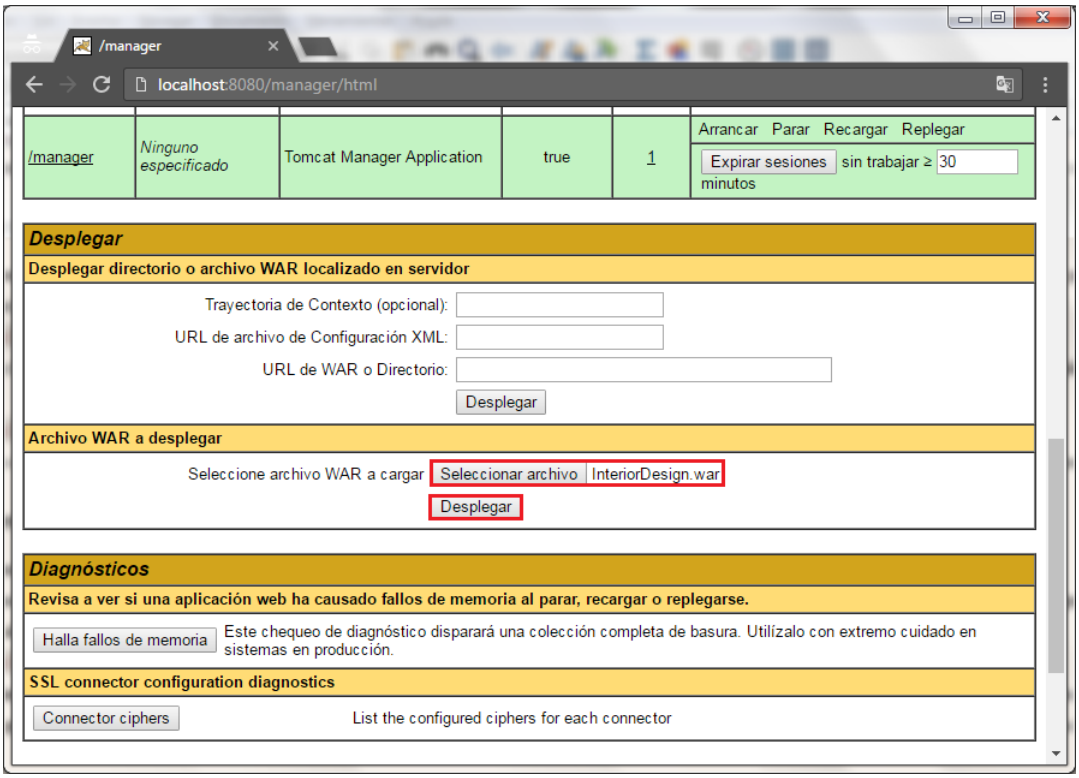

Figura 10.2: Desplegando la aplicación

10.2.2. Comenzar instalación

Para empezar la instalación basta con ir a la url de nuestro servidor (Tomcat) donde hayamos desplegado la aplicación, seguido de la carpeta "/install". Para este ejemplo se utilizará el servidor localhost. Por lo tanto, para comenzar la instalación tendríamos que ir a la siguiente url:

- `http://localhost:8080/InteriorDesign/install/`

En la página aparecerá un mensaje de bienvenida junto con los primeros datos que debemos introducir, que son:

- Nombre de la base de datos
- Usuario de la base de datos
- Contraseña
- Servidor de la base de datos
- Indicar si queremos instalar datos de ejemplo

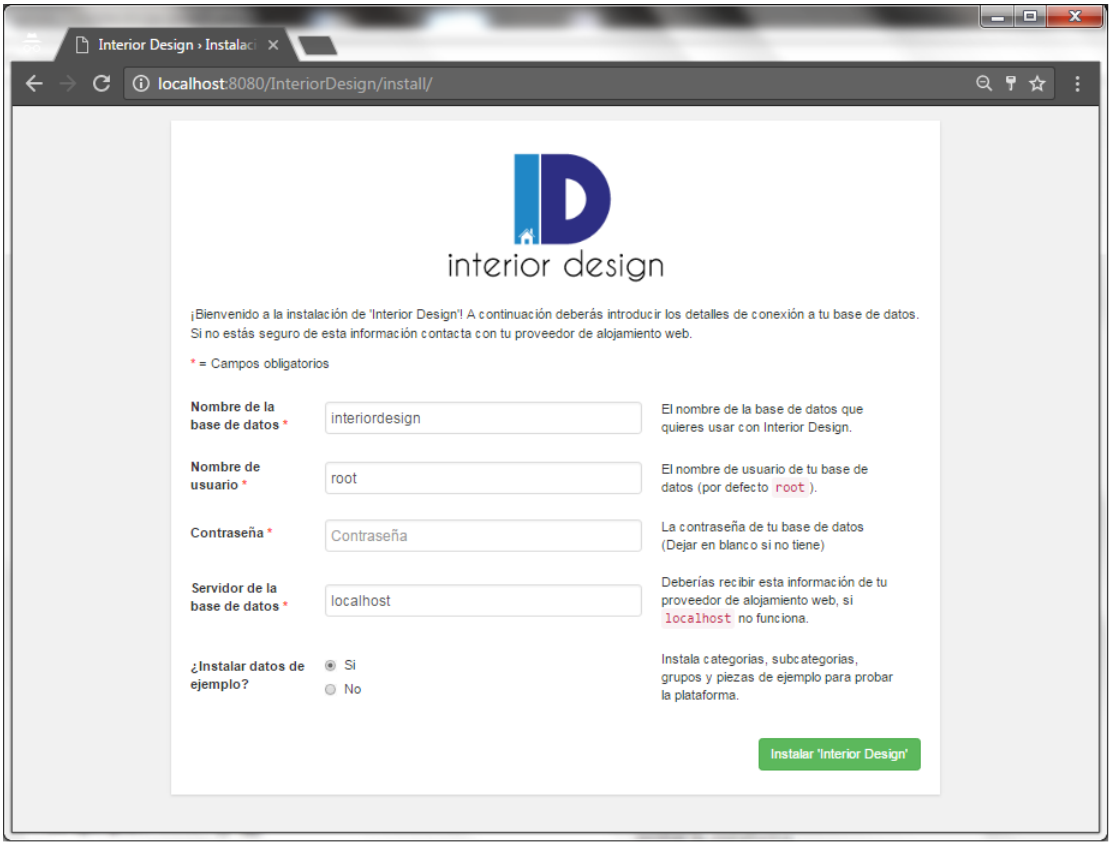

Figura 10.3: Instalación automática - Bienvenida

Una vez introducidos los datos solicitados, serán enviados al servidor y se comprobarán que son correctos. Si los datos son correctos se procede a la instalación de la plataforma y se redirigirá a la siguiente parte de la instalación, en caso contrario se redirige a otra página indicando el error provocado.

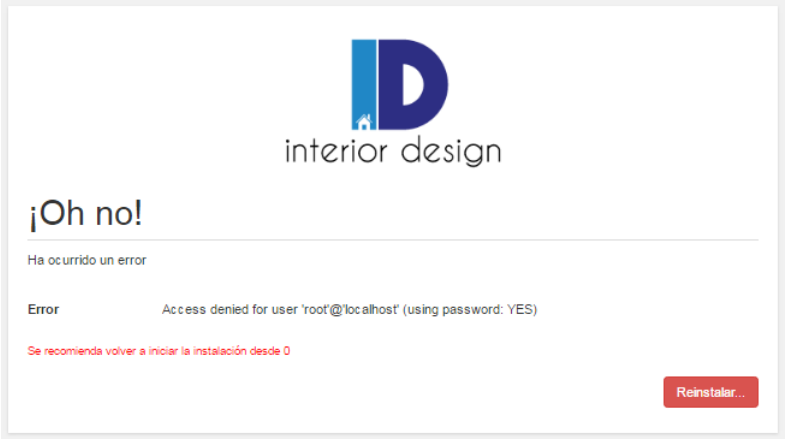

Figura 10.4: Instalación automática - Página de error

10.2.2.1. Errores conocidos

Durante esta parte de la instalación pueden aparecer varios tipos de errores:

- Estado HTTP 500.
- Estado HTTP 503.
- Estado HTTP 404.

Estos errores están provocados por una incompatibilidad entre el conector JDBC de Java con la base de datos MySQL y Tomcat, ya que todavía no existe un contexto propio de la aplicación. En cualquiera de los tres casos se recomienda reiniciar la instalación de nuevo. Normalmente después de varios intentos el proceso acaba realizándose correctamente. Se pretende corregir este problema en futuros trabajos.

Si después de varios intentos el problema persiste, se recomienda realizar la instalación de manera manual.

Este problema solamente afectará durante el proceso de instalación, por lo que una vez instalada la plataforma no volverá a ocurrir.

10.2.3. Datos adicionales

Una vez la instalación base se ha completado, debemos facilitar unos datos adicionales que son:

- Usuario administrador

- Datos del servidor SMTP
- Login con Facebook

10.2.3.1. Usuario administrador

La plataforma debe tener un usuario administrador que sea el encargado de gestionarla. En este apartado es donde se crea este usuario, indicando un nombre de usuario, un correo electrónico y una contraseña.

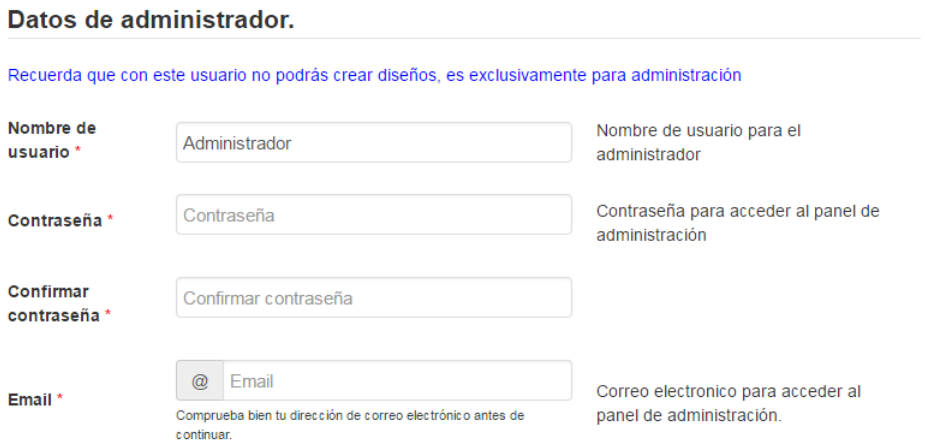

Figura 10.5: Instalación automática - Datos adicionales - Administrador

Hay que tener en cuenta que este usuario es exclusivamente para acceder al panel de administración y que no podrá crear o modificar diseños. Además se recomienda que sea una cuenta de correo electrónico que se utilice únicamente para la gestión de la plataforma, y que no sea una cuenta personal. Para este ejemplo se ha creado la cuenta “management.interiordesign@gmail.com”.

10.2.3.2. Servidor de correo saliente SMTP

Para poder enviar correos a los usuarios que se acaban de registrar y que puedan activar su cuenta, o para responder comentarios, es necesario realizar la configuración del servidor SMTP. Para realizar esta configuración hay que facilitar tanto la dirección de correo electrónico como la contraseña de acceso a dicho correo, el host, y el puerto de acceso. Cada proveedor de correo electrónico (Hotmail, outlook, yahoo...) tiene su propio host y puerto. Por ejemplo, los de gmail son “smtp.gmail.com” y “587”. Es necesario consultar la configuración SMTP del proveedor de correo electrónico que vayamos a utilizar. Adicionalmente, en el caso de gmail, hay que configurar la cuenta

como “No segura” para que permita enviar correos electrónicos desde el servidor. Posiblemente en otros proveedores se deba realizar alguna configuración similar.

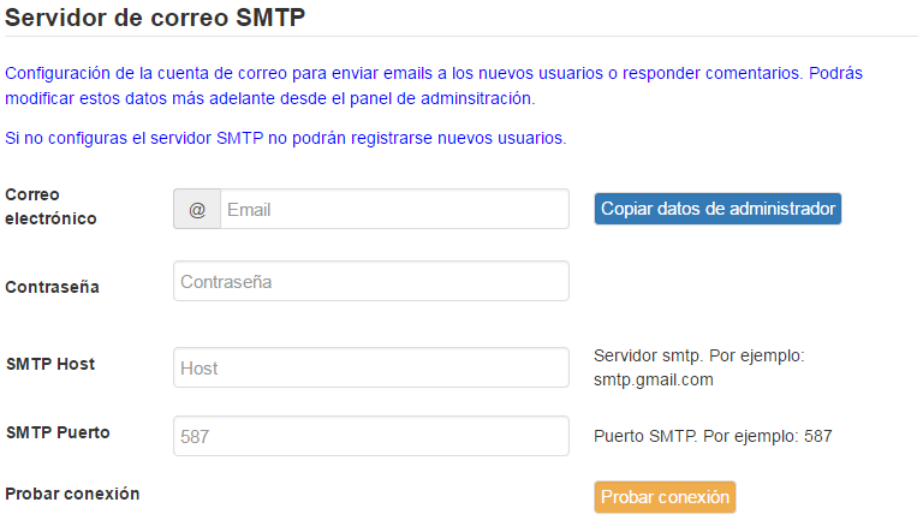

Figura 10.6: Instalación automática - Datos adicionales - Servidor SMTP

Además una vez introducidos los datos podemos probar si hay conectividad utilizando el botón “Probar conexión” que nos devolverá un mensaje de éxito en el caso de que la conexión haya sido satisfactoria o el correspondiente mensaje de error en caso contrario.

10.2.3.3. Login con Facebook

Se puede habilitar también el login o registro mediante Facebook. Para activar esta funcionalidad es necesario crear previamente la APP en Facebook Developers o utilizar la que se proporciona por defecto.

Figura 10.7: Instalación automática - Datos adicionales - Login con Facebook

Si se desea utilizar la APP por defecto, hay que comunicar al administrador de dicha APP el dominio bajo el que se instalará la plataforma para que acepte las peticiones de login y registro.
Una vez completados todos los campos, pulsaremos el botón "Continuar" para seguir con la instalación.

10.2.4. Instalación completada

Si la instalación se ha realizado correctamente, se redirigirá a la página final donde se indicará tal información, donde tendremos un botón "Acceder" para ir a la página principal de la plataforma. En este momento podremos hacer login con el usuario Administrador del que hemos facilitado los datos previamente.

10.2.4.1. Carpeta "install"

Una vez finalizada la instalación, y comprobado que los datos son correctos, se recomienda borrar la carpeta "install" del servidor para evitar que se pueda sobrescribir la instalación actual.

10.3. Manual

Para realizar la instalación manualmente deberemos tener instalado Eclipse IDE en su versión JEE para que nos permita importar el proyecto correctamente. Para esta parte se requieren unos conocimientos mínimos de Java y de soltura utilizando el entorno de desarrollo Eclipse.

10.3.1. Importando la base de datos

Lo primero que debemos hacer es importar la base de datos en MySQL. En nuestro caso, con XAMPP, podemos hacerlo a través de la dirección "http://localhost/phpmyadmin". Crearemos una base de datos con el nombre deseado, y a continuación en la pestaña "import" seleccionaremos el fichero ".sql". Recordemos que contamos con dos ficheros.

- interiordesign.sql: configuración predeterminada
- interiordesign_example.sql: configuración con datos de ejemplo para probar la plataforma

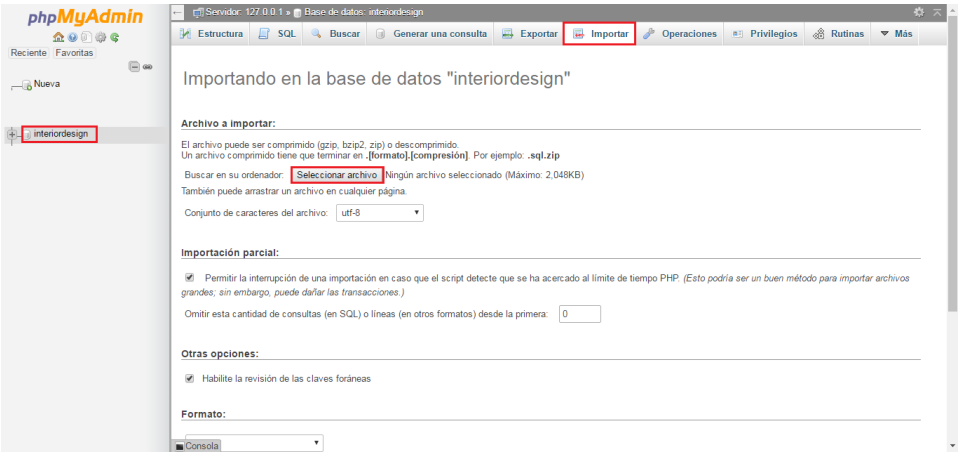

Figura 10.8: Importando la base de datos

Debemos importar el que más se adecue a nuestras necesidades.

10.3.2. Importando el proyecto

Lo primero que debemos hacer es importar el proyecto a través del menú superior "File -> Import". Seleccionaremos la opción "General" y dentro de esta "Existing Projects into Workspace".

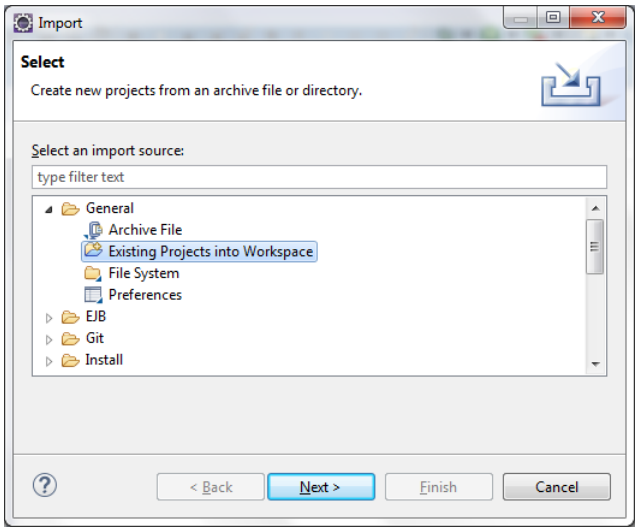

Figura 10.9: Importando el proyecto en Eclipse

A continuación seleccionamos el proyecto de donde lo tengamos guardado y lo importamos. Automáticamente Eclipse compilará el proyecto, y si todo va correctamente, se importará sin errores.

10.3.3. Realizando la configuración

Ahora tenemos que facilitar la configuración necesaria para que la plataforma funcione correctamente. Para ello ejecutaremos el fichero contenido en el paquete "util" llamado "InstalaciónManual.java", dentro de "JavaResources - src/main/java", como una aplicación Java (Java Application).

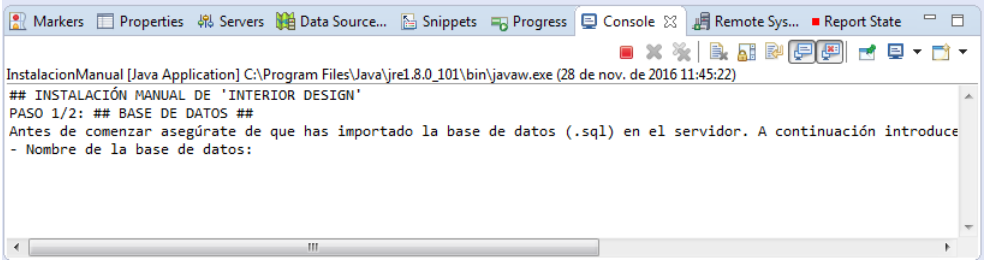

Figura 10.10: Configuración manual

Una vez facilitemos todos los parámetros de configuración, nos saldrá un resumen de todos los datos. Si los datos son correctos podremos desplegar la aplicación. En caso contrario, deberíamos ejecutar de nuevo el script.

```
## INSTALACIÓN FINALIZADA ##
-- SI LOS SIGUIENTES PARÁMETROS NO SON
CORRECTOS VUELVA A INICIAR ESTE SCRIPT --
## RESUMEN ##
## 1. BASE DE DATOS ##
 - Nombre de la base de datos: interiordesign
 - Usuario de la base de datos: root
 - Pass de la base de datos:
 - Host: localhost

## 2. DATOS ADICIONALES ##

## 2a) USUARIO ADMINISTRADOR ##
 - Nombre de usuario: Administrador
 - Contraseña: tucontraseña
 - Correo electrónico: tu@correo.es

## 2b) CONFIGURACIÓN SMTP ##
 - Correo electrónico: tu@correo.es
 - Contraseña: tucontraseña
 - Host: smtp.proveedor.es
 - Puerto: puerto

## 2c) CONFIGURACIÓN FACEBOOK ##
 - ID: 1013189332112047
 - KEY: ffdd537f78e7e822f690c4603303803a

## ¡DISFRUTA INTERIOR DESIGN! ##
```

Figura 10.11: Resumen de configuración manual

En caso de producirse algún error como por ejemplo que no haya conexión con la base de datos, o los datos de conexión sean incorrectos, podrá tracearse a través de la interfaz de Eclipse.

10.3.4. Desplegando el proyecto

Una vez hemos realizado la configuración podemos desplegar la aplicación. Para ellos hacemos doble clic sobre el proyecto "Export -> WAR File".
El último paso es desplegarlo en nuestro servidor de aplicaciones Tomcat. Para acceder a Tomcat abriremos el navegador de Internet e introduciremos la siguiente url:

- `http://localhost:8080/`

Accediendo a "localhost" accedemos al servidor Apache, si además especificamos el puerto 8080, accederemos a Tomcat.

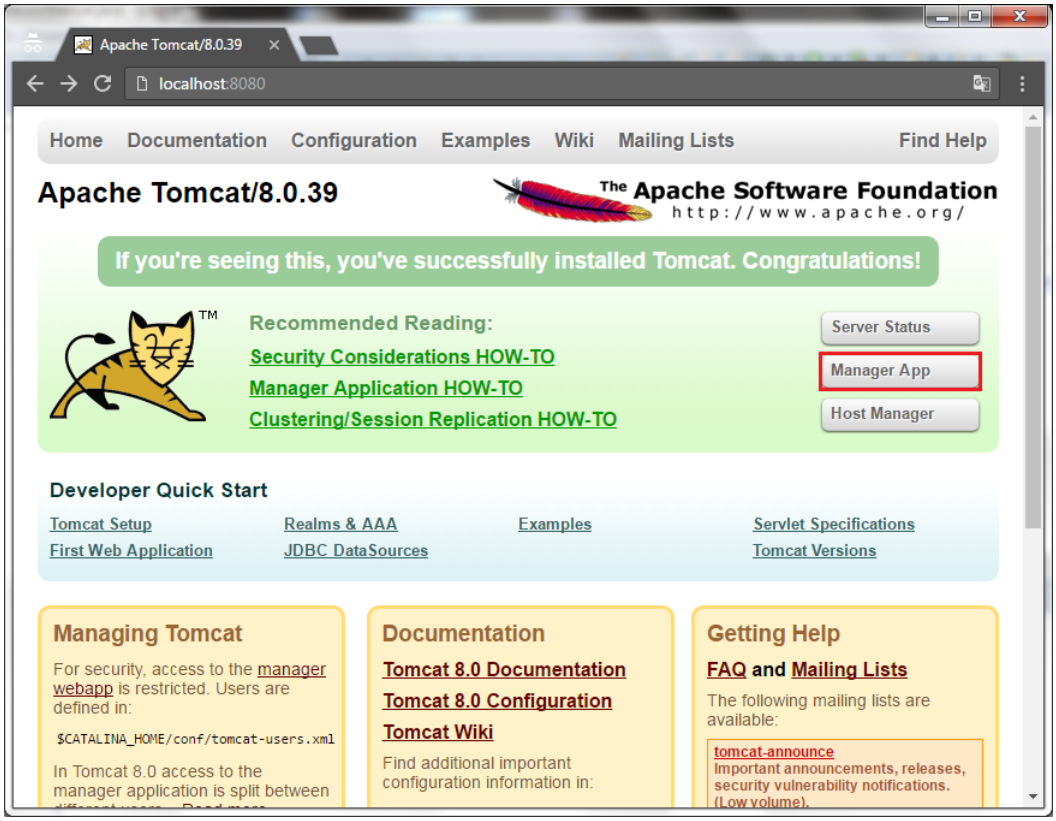

Figura 10.12: Accediendo a Tomcat

A continuación accedemos a la opción "Manager App", que nos permitirá gestionar todas las aplicaciones que tengamos desplegadas. Para entrar en esta configuración, tendremos que introducir el usuario y contraseña que habremos tenido que especificar

en el fichero “tomcat-users.xml” de Tomcat cuando instalamos XAMPP. Si no estás usando XAMPP, deberás ir a la carpeta donde esté instalado Tomcat y modificar dicho archivo.
Una vez estemos en el panel de gestión, seleccionaremos nuestro archivo “InteriorDesign.war” y lo desplegaremos.

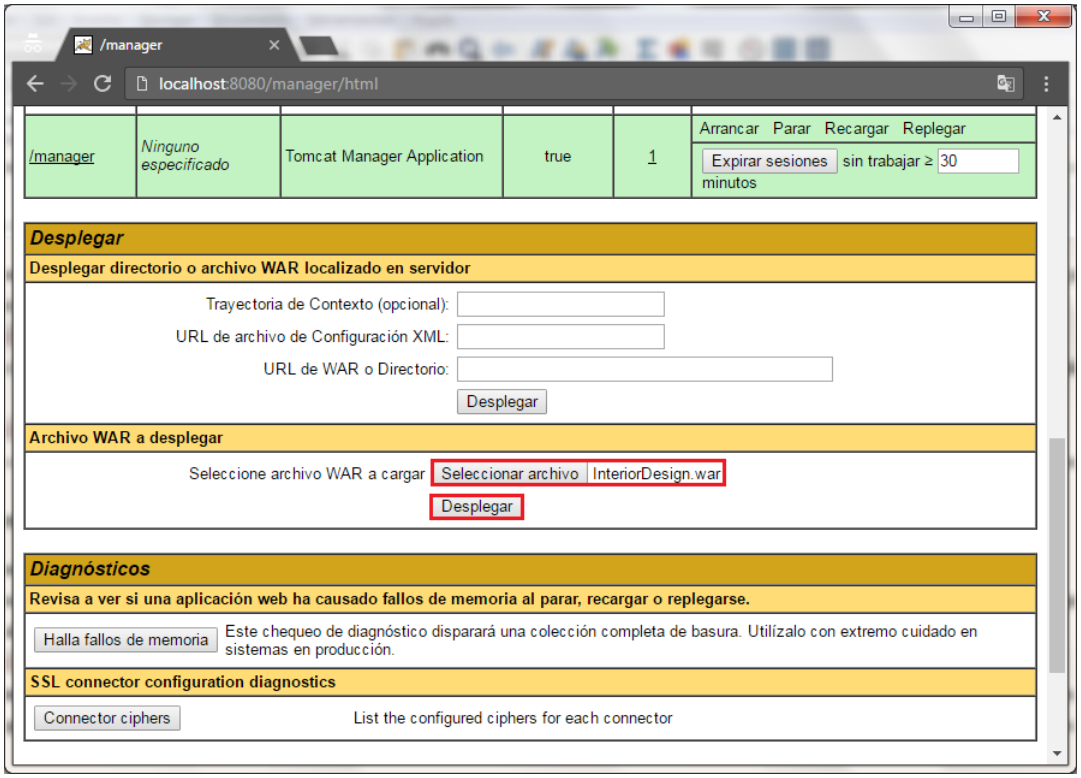

Figura 10.13: Desplegando la aplicación

Si todo va bien, podremos acceder a la aplicación a través del enlace:

- `http://localhost:8080/InteriorDesign`

Capítulo 11

Funcionamiento

Tal y como se ha indicado a lo largo de este texto, nuestro diseño será completamente responsive (adaptativo). Para ello haremos uso del framework Bootstrap [6]. En combinación con CSS y JavaScript, nuestras páginas webs tendrán un diseño intuitivo y atractivo para el usuario.

Toda petición de usuario será enviada al servlet principal denominado "Action", que a su vez derivará la petición al servlet correspondiente, en función de la petición (piezas, categorías, diseños, registro, etc). Estos servlets serán los encargados de realizar las operaciones oportunas y de contactar con la base de datos para obtener un resultado, que posteriormente, será devuelto al usuario. La estructura de estos servlets ha sido detallada en el capítulo de "Diseño".

Esta será la mecánica de todas las operaciones que se puedan realizar en la web. Excepcionalmente, aquellas operaciones que requieran mantenerse en la misma página aún cuando se le envíe la respuesta, no serán enviadas al servlet, si no que utilizando Ajax se procesará la petición y se devolverá el resultado (por ejemplo, al buscar una pieza, los resultados de la búsqueda se muestran en la misma página, no se envía a una página nueva).

11.1. Página principal

La página principal de la plataforma que será cargada al acceder a ella cuenta con un menú superior que permite:

- Login: acceso a la parte de diseño por parte del usuario o a la parte de administración por parte del administrador.

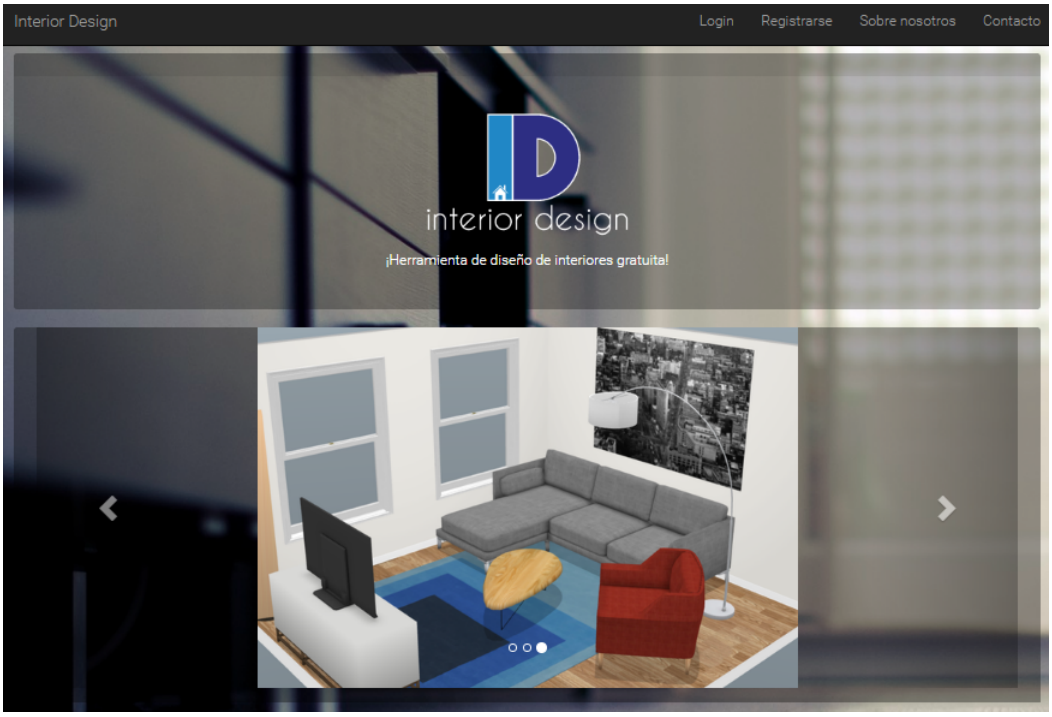

Figura 11.1: Página principal (1/4) - Menú de usuario

- Registrarse: creación de un nuevo usuario.
- Sobre nosotros: información sobre las tecnologías más relevantes utilizadas en la plataforma.
- Contacto: mapa de situación del CUM y formulario de contacto.

Además, cuando se accede a la web, se comprueban varios aspectos:

- Si el sitio está deshabilitado para realizar tareas de mantenimiento.
    - En este caso, el usuario "administrador" deberá entrar al panel de administración a través de la dirección "/admin.jsp".
- Si el registro de usuarios está deshabilitado.
- Si el registro de usuarios mediante Facebook está deshabilitado.

Estas opciones pueden habilitarse o deshabilitarse según las necesidades del administrador a través del panel de administración.

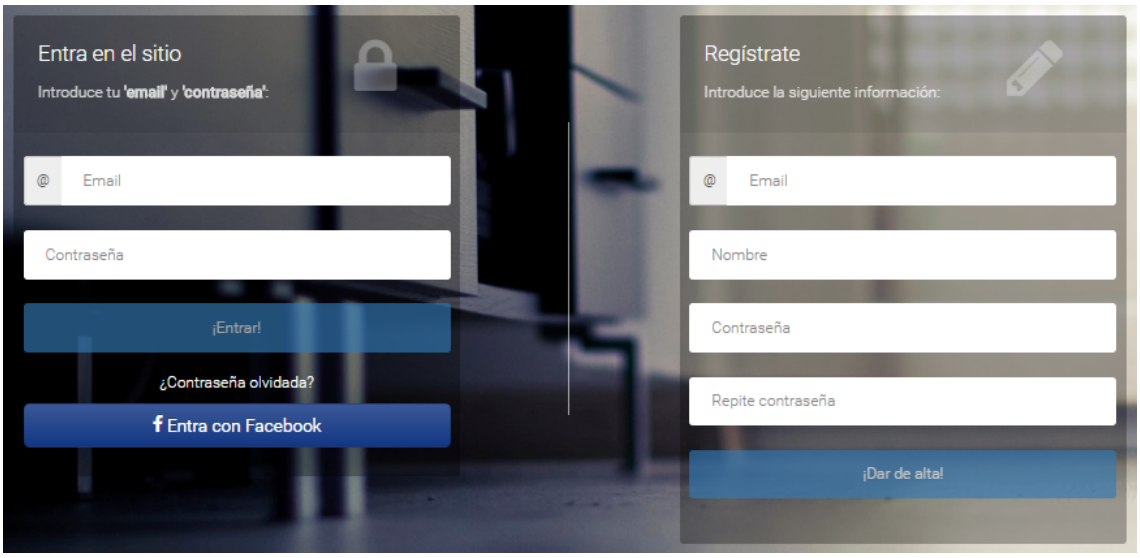

Figura 11.2: Página principal (2/4) - Registro y login

Figura 11.3: Página principal (3/4) - Sobre nosotros

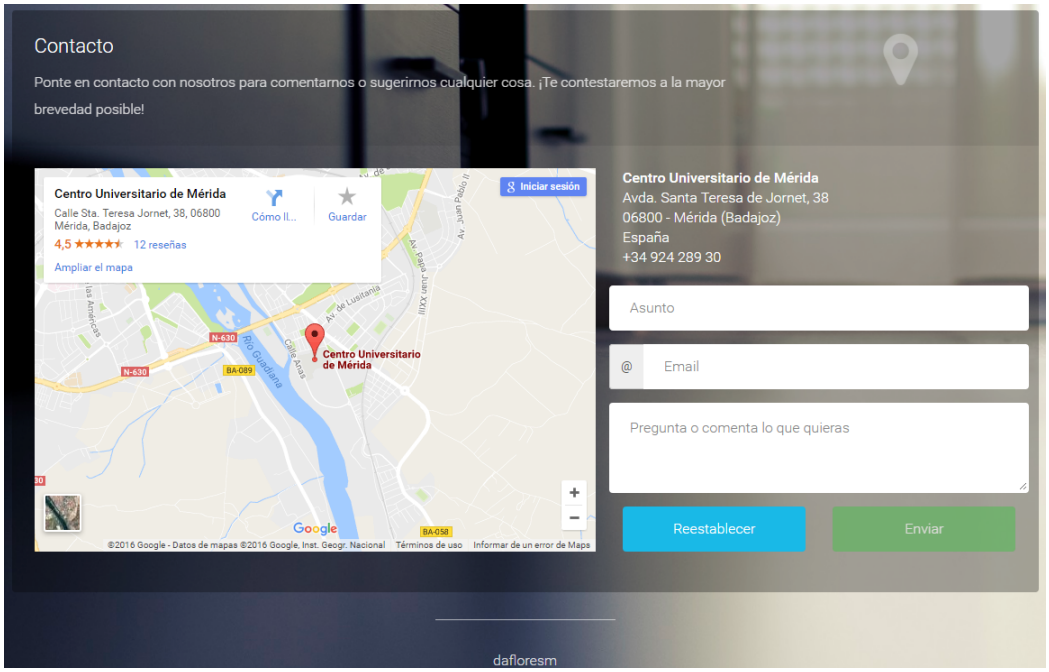

Figura 11.4: Página principal (4/4) - Contacto

11.1.1. Registro y login de usuarios

Para el registro y login de usuarios se utilizará una plantilla (template) específica para tal fin (11.2), creada con Bootstrap [2].

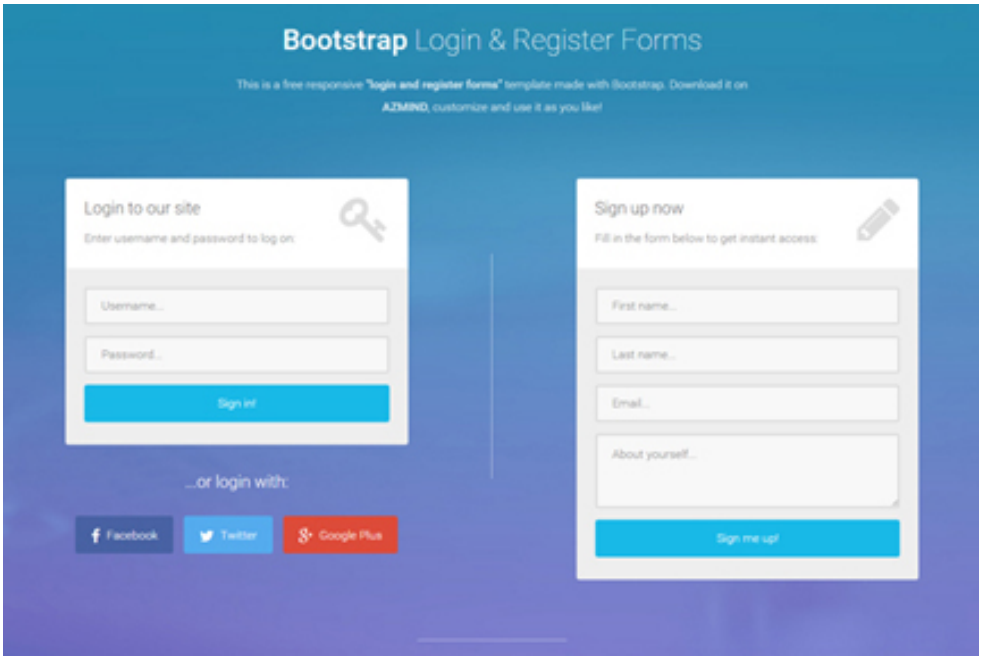

Figura 11.5: Plantilla login y registro de usuarios

Esta plantilla será modificada para adaptarla a nuestras necesidades.
Además se ha implementado un registro y login de usuarios [39] bastante completo que permite:

- Registro de usuarios
  - A través del formulario
    - Se envía un correo de confirmación para activar la cuenta (La configuración de correo saliente SMTP debe estar correctamente configurada)
  - Conectando con Facebook
    - Se debe dar permisos a la aplicación
- Login de usuarios
- Recuperación de contraseña olvidada
- Modificación de contraseña

Todas las contraseñas y datos de recuperación de contraseñas (tokens) están debidamente cifrados para mantener la privacidad de los usuarios.

11.1.1.1. Registro de nuevo usuario

Formulario de registro El registro de un usuario nuevo es tan sencillo como rellenar el formulario con los datos solicitados, que son:

- Email
- Nombre
- Contraseña (por duplicado)

Una vez rellenado los datos pulsaríamos el botón "¡Dar de alta!". Si el registro es realizado correctamente, el sistema enviará automáticamente un email de confirmación al usuario para activar la cuenta. Si no se activa, no podrá utilizarse.

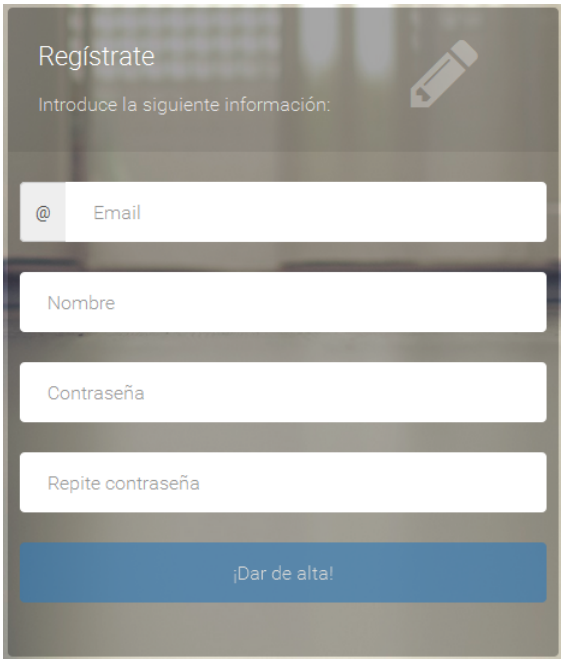

Figura 11.6: Formulario de registro de usuario

Como en todos los formularios de este trabajo, la validación de los datos se lleva a cabo mediante una librería JavaScript desarrollada para tal fin: *validator.js* [1] en su versión para Bootstrap3.

- No se puede registrar un email que ya esté registrado previamente
- La contraseña no puede estar vacía. Debe ser mayor de 6 y menor de 20 caracteres y usar caracteres A-z, 0-9, @ # $ % ! ^ & *
- Debe introducirse la contraseña por duplicado
- Todos los campos son obligatorios

Registrar con Facebook Un usuario también puede registrarse en la plataforma y acceder a ella utilizando Facebook. Para ello basta con pulsar el botón "Entra con Facebook" y dar permisos a la aplicación. Los datos relativos a la aplicación de Facebook (ID y Key) a la que se hará referencia, serán configurados a la hora de instalar la plataforma o a través del panel de administración.

También se da la posibilidad de utilizar la aplicación creada por defecto para el proyecto. Si se desea utilizar la app por defecto, se debe comunicar al administración de dicha aplicación el dominio que será utilizado para acceder a la plataforma, ya que

habría que añadirlo a la lista de direcciones válidas que permiten realizar peticiones a Facebook.

Esta implementación se ha realizado gracias a un tutorial que describe los pasos a seguir para conseguir la autenticación en un sitio web mediante Facebook [18].

Opciones por defecto del usuario Cuando un usuario se registra, se le asigna automáticamente las opciones por defecto, que son:

- Colisiones entre objetos: desactivada por defecto
- Rotación de escena: desactivada por defecto
- Magnetización de objetos: activada por defecto
- Tolerancia de magnetización: establecida a un valor de 15 centímetros

11.1.1.2. Login de usuario

Para entrar a la plataforma basta con introducir el email y la contraseña con la que el usuario se haya registrado previamente, y una vez haya activado la cuenta. En su defecto, se puede acceder a ella a través de Facebook.

11.1.1.3. Contraseña olvidada

También se da la posibilidad de restablecer la contraseña si ha sido olvidada. Para ello se hará click sobre el la opción "¿Contraseña olvidada?" que automáticamente redirigirá a otra página en la que tendremos que introducir el correo electrónico de la cuenta cuya contraseña queramos restablecer.

Una vez introducido el email pulsaremos el botón de "Enviar email de verificación" para poder restablecerla. Automáticamente será enviado un email a dicha dirección con los pasos a seguir para cambiar la contraseña.

11.1.2. Contacto

Un mapa interactivo de Google (11.4) indica la posición y dirección del CUM y además también se permite contactar con el administrador de la plataforma mediante un formulario. Para enviar un mensaje solo hay que completar los campos "asunto", "email" y el "mensaje" que se desee enviar. Automáticamente se almacenará el mensaje en la base de datos y el administrador podrá responderlo (o descartarlo) desde la sección de mantenimiento en el panel de administración.

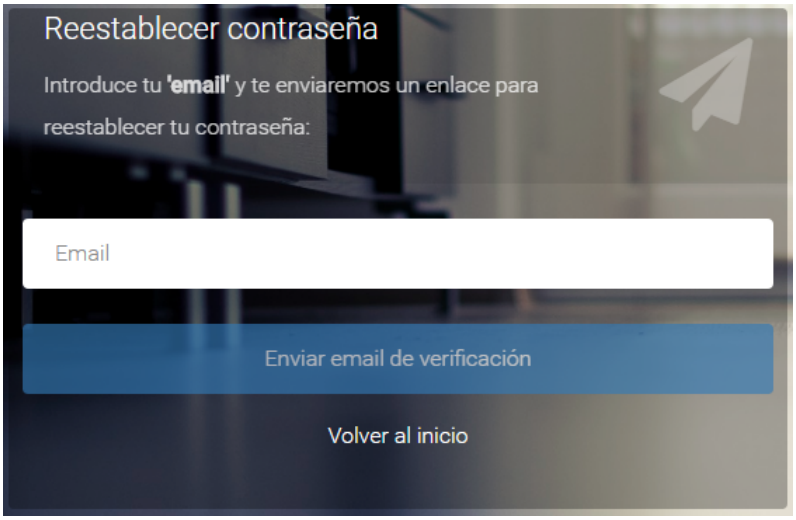

Figura 11.7: Restablecer contraseña

11.2. Página de diseño

Cuando un usuario se loguea en la plataforma, automáticamente accede a la página donde podrá crear o modificar diseños. Esta es la página más importante de la web, pues es la que cuenta con las principales funcionalidades que el cliente nos ha solicitado, y es donde el usuario podrá realizar sus diseños personalizados.

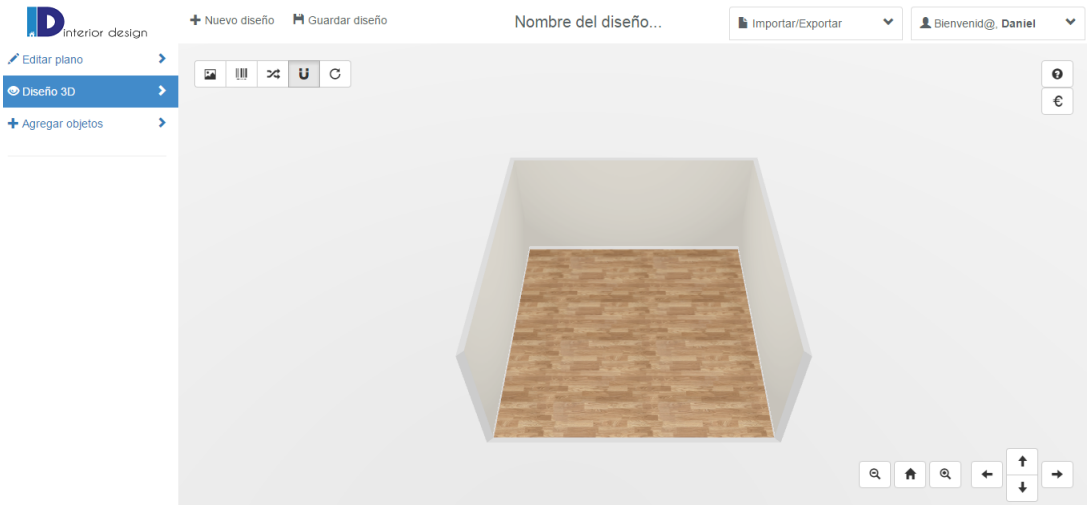

Figura 11.8: Página de diseño

En esta página está dividida en tres partes fundamentales, accesibles a través del menú lateral izquierdo, que son:

- Editar plano

- Diseño 3D
- Agregar objetos

11.2.1. Editar plano

En esta sección podremos definir el plano de nuestro diseño.

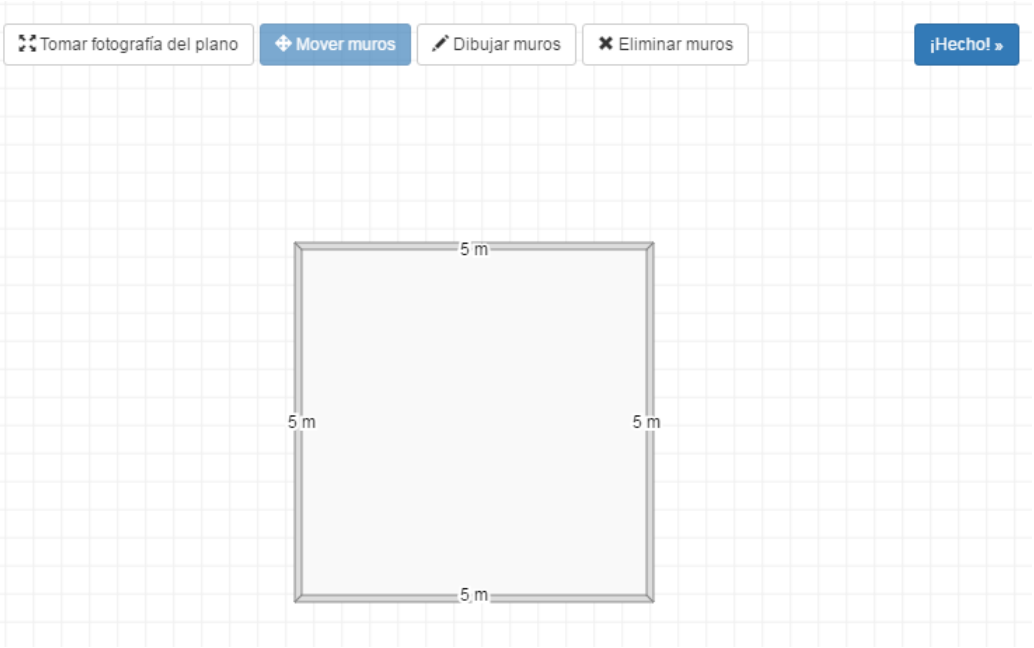

Figura 11.9: Editar plano

Para ello podremos:

- Tomar fotografía del plano: descargar una imagen del plano actual. Gracias a Three.js podremos descargar una imagen del plano. El nombre de la imagen por defecto es el título del diseño seguido de la fecha actual
- Mover muros: cambiar la disposición de los muros ya creados
- Dibujar muros: crear nuevos muros para añadir distribuciones al plano de la habitación
- Eliminar muros: elimina un muro existente

Además se indica la longitud en metros para cada uno de los muros creados, para ayudar al usuario a crear el plano con la mayor exactitud posible.

11.2.2. Diseño 3D

Sección donde se crea y previsualiza el diseño (escena). Es la parte más importante de la plataforma.

11.2.2.1. Funciones principales

Figura 11.10: Detalle de la pantalla de diseño

Las funciones principales son:

1. Creación y guardado de diseños: estas opciones permiten crear un diseño nuevo y guardar el diseño actual en la base de datos

2. Nombre del diseño: se permite personalizar el nombre del diseño

3. Importar/Exportar diseño: importa el diseño actual a un archivo de texto plano (txt) para poder conservarlo en el equipo del usuario. También podrá ser exportado posteriormente dentro de la plataforma.

4. Menú de usuario: menú del usuario que cuenta con las opciones:

   a) Mis diseños: se visualizan los diseños actuales que pueden ser cargados o eliminados

   b) Opciones: se permite cambiar las opciones por defecto del usuario en cuando al comportamiento de la escena

   c) Cambiar contraseña: permite al usuario cambiar su contraseña. Esta opción no estará disponible si la cuenta es de Facebook

   d) Salir: cerrar sesión y salir de la plataforma.

5. Menú de opciones: permite las siguientes funcionalidades:

   a) Tomar una captura del diseño: al igual que en el plano, se permite descargar una imagen de la situación actual del diseño

   b) Medir la distancia entre dos puntos: permite al usuario calcular la distancia entre dos puntos de la escena

   c) Activar/desactivar colisiones: activa o desactiva las colisiones entre los objetos

   d) Activar/desactivar magnetización: activa o desactiva la magnetización de objetos

   e) Activar/desactivar rotación: activa o desactiva la rotación de la escena

6. Menú secundario que permite:

   a) Visualizar la ayuda: muestra un dialogo de ayuda explicando a los usuarios las funciones principales del diseño

   b) Crear un presupuesto: confecciona un presupuesto para el diseño actual. Además, una vez creado el presupuesto, se pueden editar los detalles de los objetos (cantidad, precio, etc) y además se puede exportar a PDF

7. Botones de navegación: además de poder utilizar el ratón para moverse, se proporcionan botones para navegar por la escena

Además, para cada pieza se cuenta con un menú contextual donde se pueden personalizar sus propiedades.

11.2.2.2. Menú contextual de objetos

Para cada objeto o pieza que se añada al diseño, se cuenta con un menú contextual que permite visualizar y/o cambiar sus propiedades principales.

Figura 11.11: Menú contextual de objetos

Este menú, que aparecerá cuando se seleccione la pieza, se divide en varias partes:

- Botones superiores. Permiten:
  - Borrar pieza
  - Duplicar pieza
  - Mover pieza
- Tamaño: se indican las dimensiones de la pieza en centímetros
- Posición: se indica la posición dentro del plano
  - Bloquear: bloquea la pieza para evitar movimientos no deseados
- Distancia con paredes: muestra la distancia de la pieza con las paredes superior, inferior, derecha e izquierda respecto de sus ejes
  - Mostrar lineas auxiliares: se muestran las lineas utilizadas para realizar el cálculo de las distancias y que puedan servir al usuario de guía
- Texturas del objeto: un objeto puede tener asociadas más de una texturas que puedan ser aplicadas, por lo que en este apartado se puede seleccionar la deseada
  - Aplicar a la pieza completa: aplicar la textura a la pieza completa sin tener en cuenta las partes diferenciadas dentro de su archivo de configuración. Este apartado se detallará cuando se hable sobre la creación de piezas con Blender

11.2.2.3. Texturas de paredes y suelo

Además de poder aplicar texturas a las piezas, también se le pueden aplicar al suelo y las paredes. Cada pared puede tener su propia textura, al igual que cada suelo. Hablamos de un suelo cuando hay una composición de paredes cerradas, por ejemplo, un cuadrado. Si tenemos un plano con tres paredes, no tenemos un suelo al que poder aplicar textura alguna.

Figura 11.12: Texuras de paredes y suelo y menú contextual

Para cambiar estas texturas basta con seleccionar la pared o suelo deseado y elegir la textura deseada a través del menú contextual. Estas texturas pueden ser añadidas o eliminadas a través del panel de administración.

11.2.2.4. Atajos de teclado

También se han incluido varios atajos de teclados que faciliten algunas de las operaciones más comunes.

	Atajo	Función
Escena	Flechas de dirección	Mover escena
Objeto seleccionado	Flechas de dirección	Mover objeto
	Supr.	Eliminar objeto de la escena
	Ctrl. + Flecha derecha	Rotar objeto hacia la derecha
	Ctrl. + Flecha izquierda	Rotar objeto hacia la izquierda

Cuadro 11.1: Atajos de teclado

Estos atajos pueden no funcionar correctamente en algunos navegadores de Internet.

11.2.3. Agregar objetos

En esta sección los usuarios podrán agregar las piezas que deseen a sus diseños.

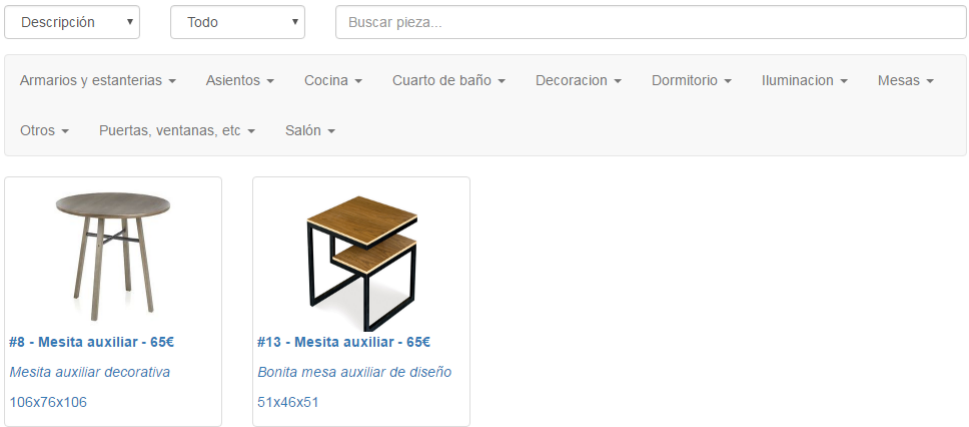

Figura 11.13: Selección de piezas

Para agregar una pieza al diseño primero tendremos que localizarla. Podemos localizar una pieza en la base de datos de dos formas:

- Seleccionando por categoría: muestra todas las piezas pertenecientes a una categoría seleccionada
- A través del buscador: se puede buscar una pieza a través del buscador por su descripción, nombre o id. Además se puede filtrar también por categoría

Cualquiera de las dos formas anteriores son válidas para buscar las piezas. Una vez localizadas aparecerán en la pantalla las piezas que concuerden con la búsqueda y podrán agregarse al diseño haciendo clic sobre la deseada.

11.3. Página de administración

El usuario administrador es el encargado de administrar y mantener la plataforma web operativa. Para conseguir esto, se ha desarrollado un panel de administración o "backend" que permite al administrador configurar todos los aspectos necesarios de la plataforma. El acceso a este panel se consigue realizando el login desde la página principal con un usuario de tipo administrador.

Figura 11.14: Panel de administración

Este panel permite administrar:

- Piezas: creación, modificación y eliminación de piezas
- Categorías: creación, modificación y eliminación de categorías
- Subcategorías: creación, modificación y eliminación de subcategorías
- Grupos: creación, modificación y eliminación de grupos
- Texturas: creación, modificación y eliminación de texturas
- Usuarios: modificación y eliminación de usuarios
- Mantenimiento: visualizar información de la web, habilitar o deshabilitar el sitio web o registro de usuarios, purgar piezas y texturas, configuración del servidor de correo saliente SMTP, configuración de la APP para el login con Facebook y responder comentarios

11.3.1. Piezas

El mantenimiento de piezas permite tanto crearlas como modificarlas. Esta gestión se lleva a cabo a través de un formulario que contiene todos los campos necesarios para configurar la pieza.

Al añadir el fichero “js”, automáticamente se lee su textura asociada, en caso de que la tenga. Además se permite previsualizar la pieza para ver como quedaría en un diseño.

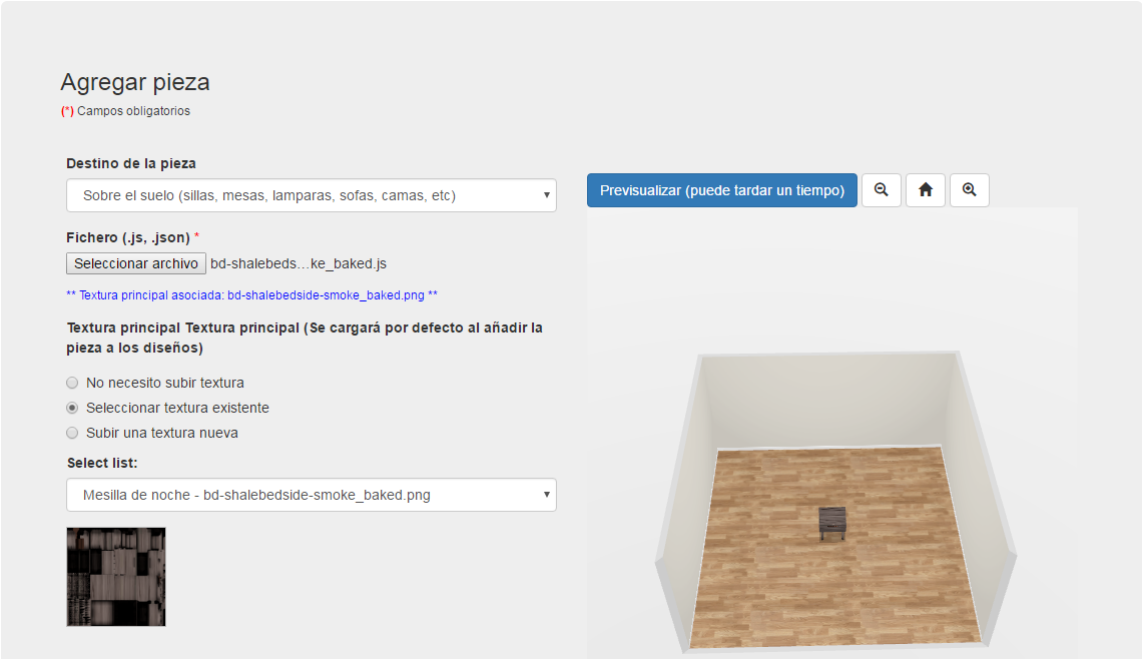

Figura 11.15: Gestión de piezas (1/3)

Figura 11.16: Gestión de piezas (2/3)

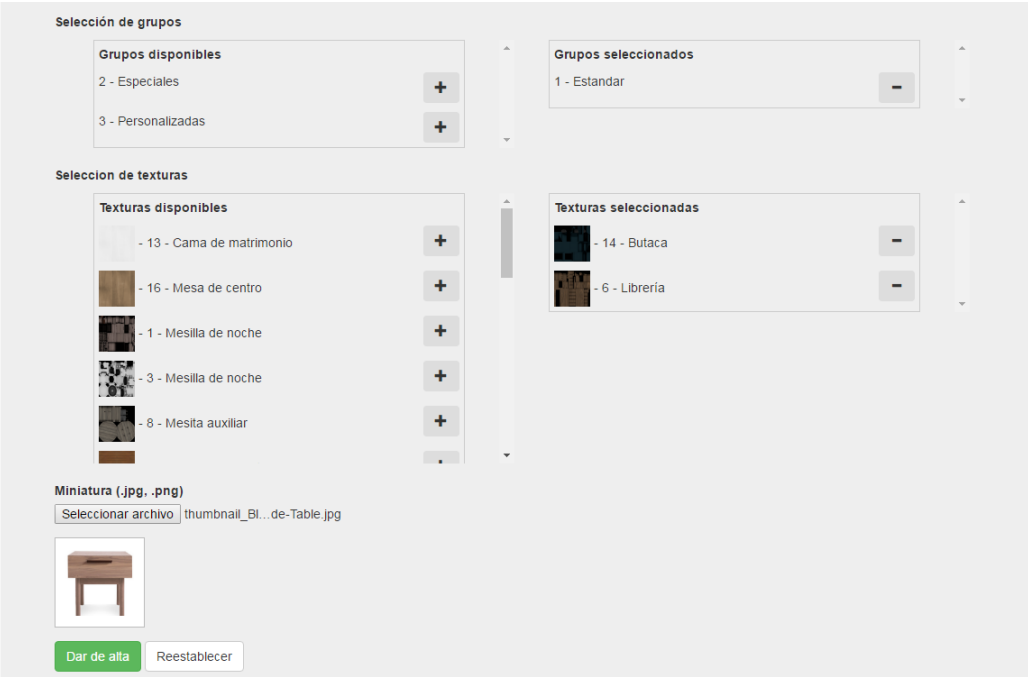

Figura 11.17: Gestión de piezas (3/3)

Una pieza puede pertenecer a varios grupos de piezas, o a ninguno si no queremos que esa pieza pueda ser utilizada en los diseños. También puede haber varias texturas que se le puedan aplicar a cada pieza.

Supongamos que tenemos la pieza “Butaca” y pertenece al grupo “Estándar” y “Especial”. Todos los usuarios que pertenezcan a esos grupos podrán utilizarla. Además, la pieza tiene tres texturas asociadas: “Cuero”, “Sintético” y “Tela premium”. A nuestra butaca se le podrá aplicar cualquiera de las tres texturas en los diseños de los usuarios. Esto facilita que para un mismo objeto podamos tener diseños completamente distintos en función de la textura aplicada, y que no sea necesario tener un objeto por cada tipo de textura.

La eliminación de las piezas únicamente las deshabilita para que no puedan ser utilizadas en futuros diseños, pero si que pueden ser cargadas por diseños ya existentes. Para eliminar una pieza definitivamente se debe primero deshabilitar desde esta sección (opción eliminar), y a continuación desde el panel de mantenimiento eliminar completamente. Hay que tener en cuenta que si se elimina una textura que se esté utilizando en diseño almacenados en la base de datos, cuando los usuarios quieran cargar sus diseños no se mostrarán correctamente.

11.3.2. Categorías

La gestión de categorías permite crear, modificar o eliminar categorías. Para ello se hace uso de tres sencillos formularios destinados a realizar cada una de las operaciones.

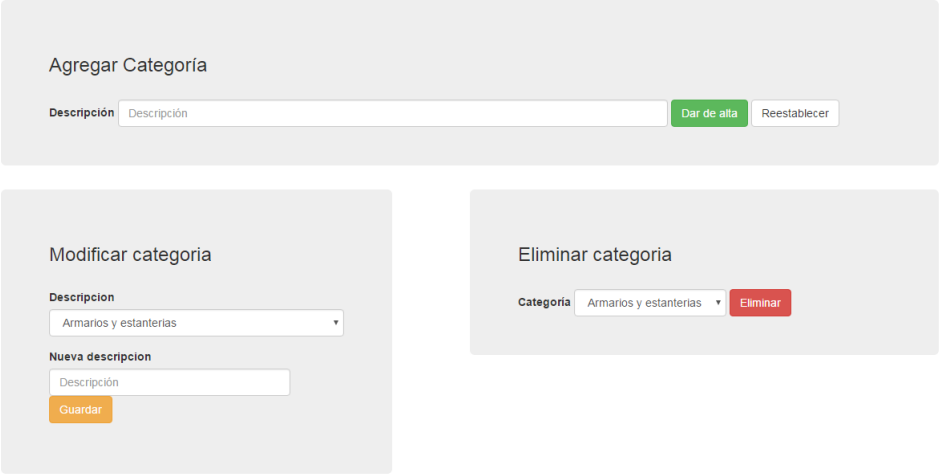

Figura 11.18: Gestión de categorías

11.3.3. Subcategorías

Como en la gestión de categorías, la gestión de subcategorías permite crear, modificar o eliminar subcategorías a través de sus formularios correspondientes. Una subcategoría debe pertenecer a una categoría.

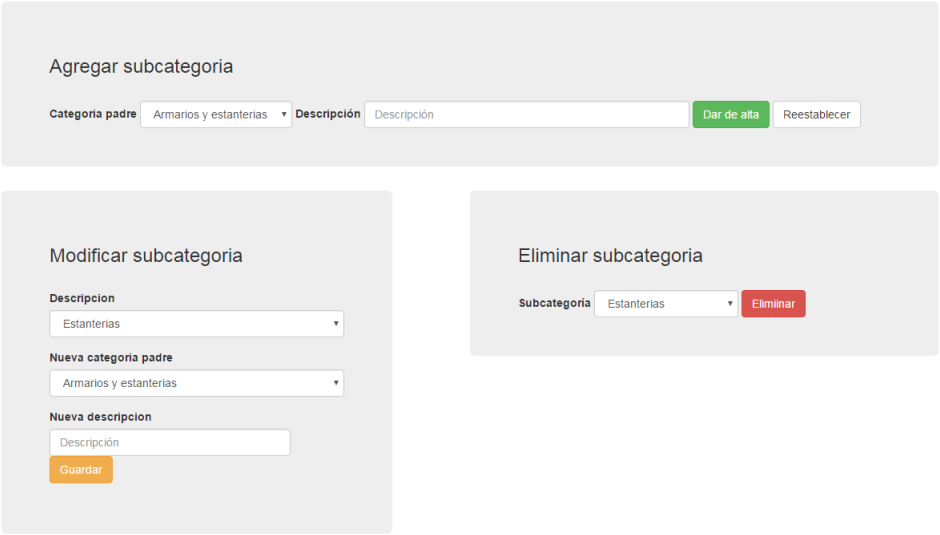

Figura 11.19: Gestión de subcategorías

11.3.4. Grupos

Los grupos contendrán las piezas que podrán ser utilizadas por los usuarios que pertenezcan a dichos grupos. También se permite realizar operaciones de creación, modificación y eliminación de grupos. El grupo "Estándar" no puede se eliminado porque es el grupo que es asignado por defecto a los usuarios cuando se dan de alta en la plataforma.

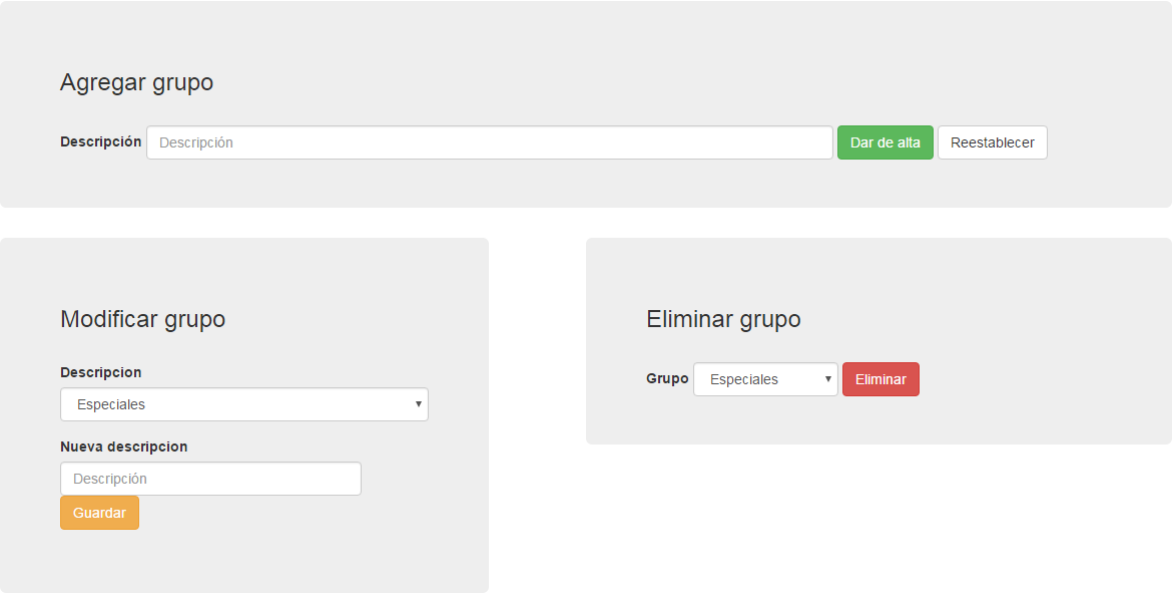

Figura 11.20: Gestión de grupos

11.3.5. Texturas

Como ya se ha hablado anteriormente, una pieza puede tener varias texturas asociadas. En esta sección se pueden tanto crear, como modificar o eliminar todas las texturas que deseemos. Se pueden crear tres tipos de textura:

- Mobiliario: texturas destinadas a los objetos de los diseños como sillas, mesas, cuadros, lámparas, etc.
- Pared: textura destinada a personalizar las paredes de los diseños. Cada pared puede tener una textura distinta.
- Suelo: textura destinada a personalizar los suelos de los diseños. Al igual que las paredes, cada suelo puede tener una textura distinta

Además la gestión de textura permite la previsualización de la imagen (textura). Al igual que con las piezas, la eliminación de las texturas únicamente las deshabilita

para que no puedan ser utilizadas en futuros diseños, pero si que pueden ser cargadas por diseños ya existentes. Para eliminar una textura definitivamente se debe primero deshabilitar desde esta sección (opción eliminar), y a continuación desde el panel de mantenimiento eliminar completamente. Hay que tener en cuenta que si se elimina una textura que se esté utilizando en diseño almacenados en la base de datos, cuando los usuarios quieran cargar sus diseños no se mostrarán correctamente.

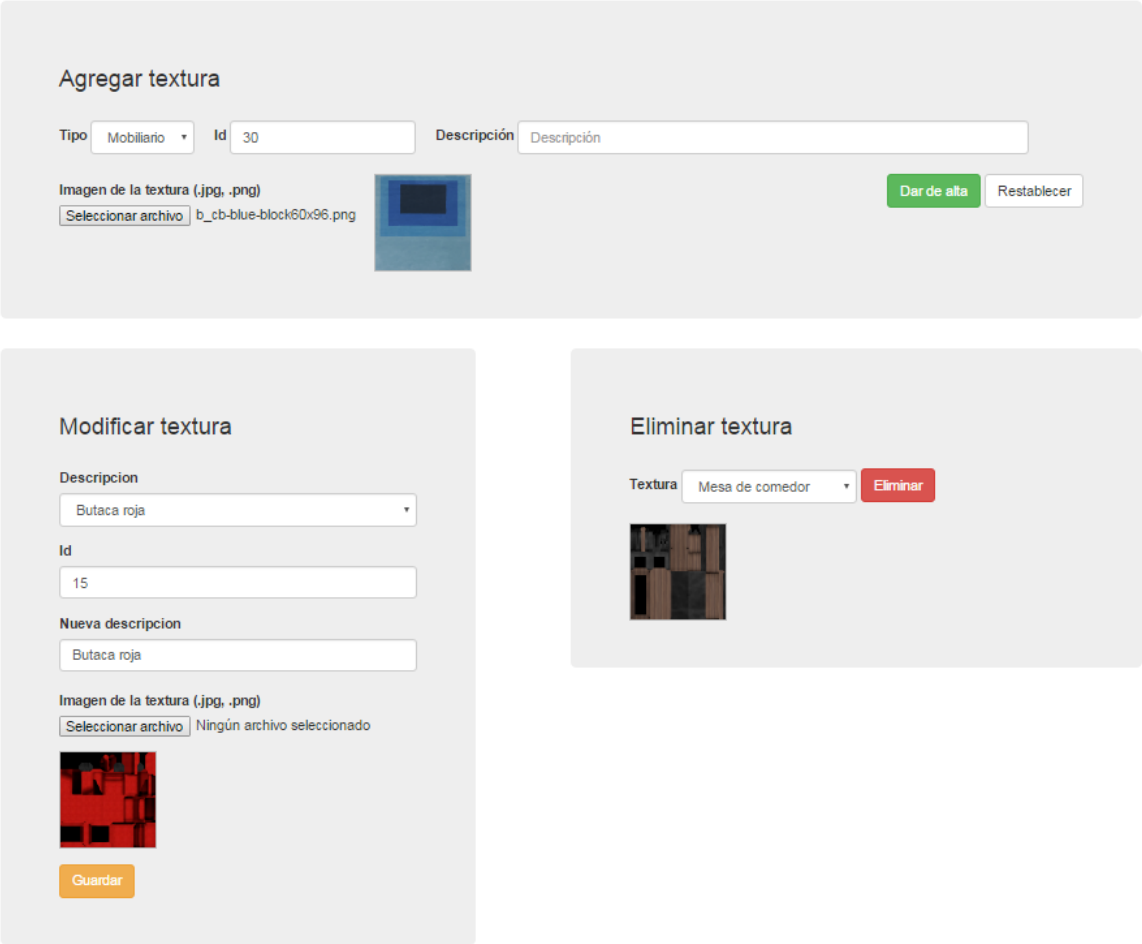

Figura 11.21: Gestión de texturas

11.3.6. Usuarios

La gestión de usuarios está dividida en dos subapartados. El primero de ellos permite la selección del usuario a editar o eliminar, mientras que el segundo permite realizar las modificaciones oportunas en el usuario seleccionado.

11.3.6.1. Selección

Para seleccionar el usuario basta con desplegar la lista y seleccionar el deseado. Además se puede ver el estado en el que está el usuario a través de una intuitiva tabla de colores.

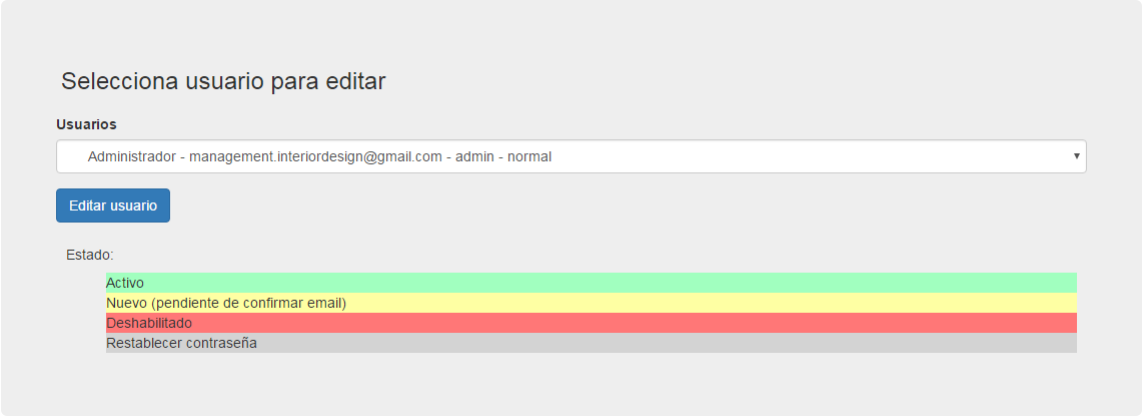

Figura 11.22: Gestión de usuarios - Selección

11.3.6.2. Edición

Una vez seleccionado el usuario podemos realizar los cambios oportunos, como cambiar sus datos personales, cambiar la contraseña (si no es usuario de Facebook), convertirlo a usuario administrador, cambiar su estado o incluirlo en grupos de piezas.

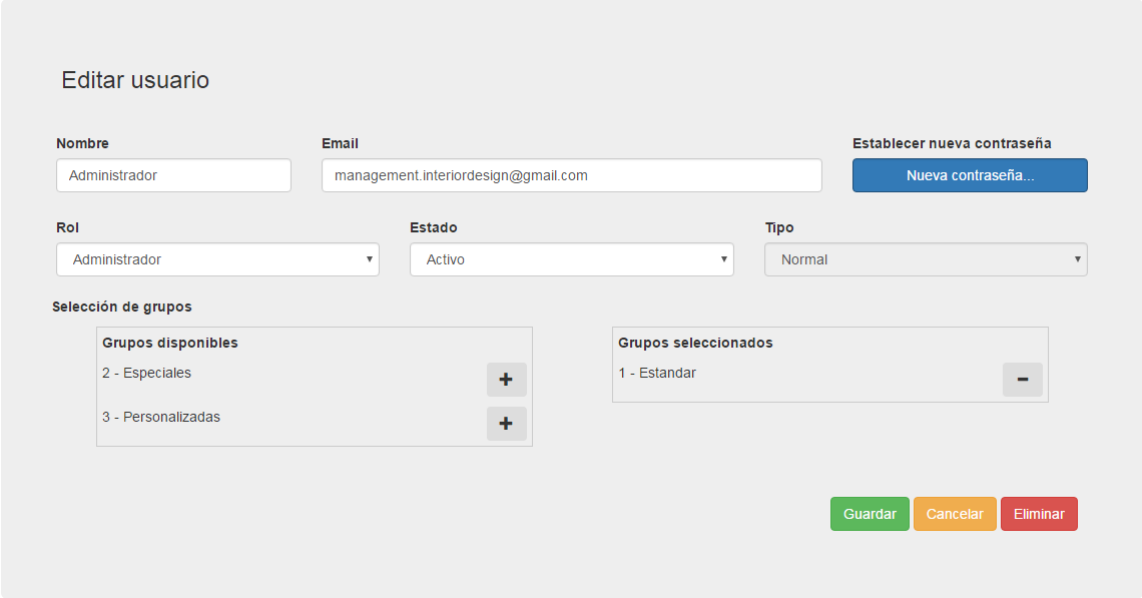

Figura 11.23: Gestión de usuarios - Edición

11.3.7. Mantenimiento

El mantenimiento de la plataforma es una de las secciones más importantes del panel de administración. En esta sección se permite ver la información sobre la plataforma, consultar el log, realizar tareas de mantenimiento, purgar piezas y texturas, configurar el servidor SMTP, configurar la APP de Facebook y responder comentarios.

11.3.7.1. Ver información y consultar el log

Ver información sobre la plataforma: nombre de la base de datos, número de usuarios registrados, piezas dadas de alta, texturas, grupos, etc

Figura 11.24: Mantenimiento - Información

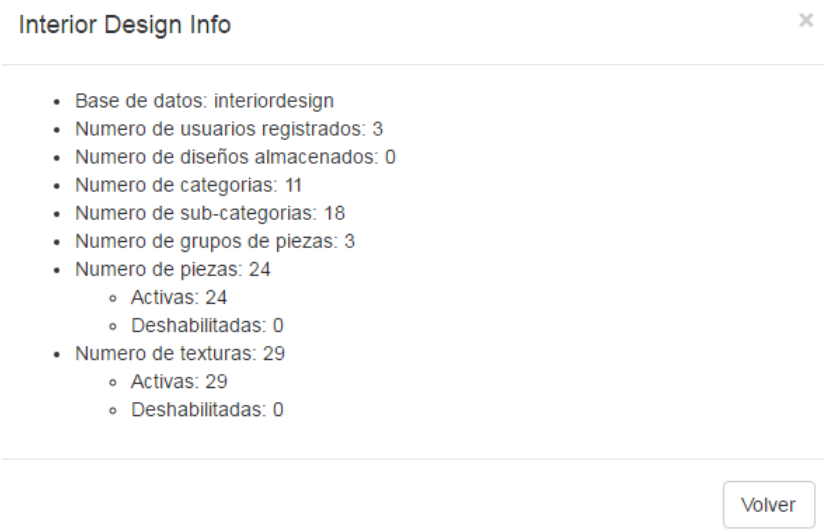

Figura 11.25: Mantenimiento - Información - Ver info

También se permite ver el log sobre las operaciones realizadas desde el panel de administración.

Interior Design Log

Fecha	Usuario	ID	Acceso	Descripción	IP
2016/11/24 12:12:22	#1 - Administrador	1	OK	Instalación de 'Interior Design'	0:0:0:0:0:0:0:1
2016/11/24 12:12:39	#1 - Administrador	2	OK	Editar pieza: 15-Cómoda butaca	0:0:0:0:0:0:0:1
2016/11/24 12:12:44	#1 - Administrador	3	OK	Deshabilitar sitio	0:0:0:0:0:0:0:1
2016/11/24 12:12:47	#1 - Administrador	4	OK	Habilitar sitio	0:0:0:0:0:0:0:1
2016/11/24 12:13:11	#1 - Administrador	5	OK	Crear categoria: Miscelanea	0:0:0:0:0:0:0:1

Volver Limpiar log

Figura 11.26: Mantenimiento - Información - Ver log

11.3.7.2. Tareas de mantenimiento

Se pueden realizar varias tareas:

- Deshabilitar el sitio: deshabilita el sitio en el caso de que se necesiten realizar tareas de mantenimiento. Si se deshabilita, el usuario administrador deberá acceder al panel de administración a través de la url "/admin.jsp"
- Deshabilitar registro: impide que se puedan registrar nuevos usuarios desde el formulario principal. Si podrían registrarse mediante el botón de Facebook
- Eliminar ficheros temporales: cuando se previsualiza una pieza a la hora de crearla o editarla, se crean archivos temporales para que la previsualización sea correcta. Desde esta opción se permite eliminar esos archivos que ya no son necesarios y que están ocupando espacio en el servidor

Figura 11.27: Mantenimiento - Tareas

11.3.7.3. Purgar piezas

Purgar piezas elimina completamente las piezas que hayan sido deshabilitadas o permite habilitarlas de nuevo para poder utilizarlas

Figura 11.28: Mantenimiento - Purgar piezas

11.3.7.4. Purgar texturas

Purgar texturas elimina completamente las texturas que hayan sido deshabilitadas o permite habilitarlas de nuevo para poder utilizarlas

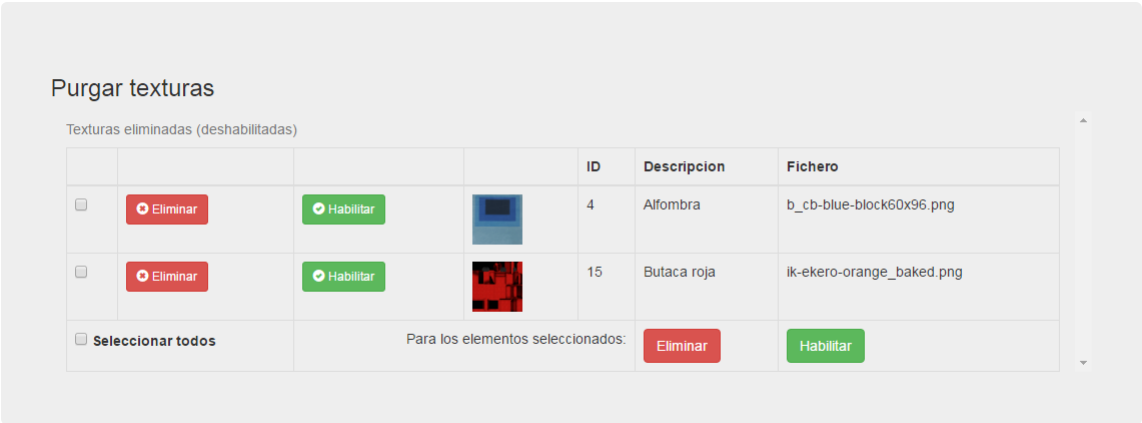

Figura 11.29: Mantenimiento - Purgar texturas

11.3.7.5. Configuración del SMTP

- Configuración de correo saliente SMTP: permite realizar la configuración del servidor SMTP para enviar desde la plataforma a los usuarios
    - Se cuenta además con una opción para comprobar si dicha configuración es correcta

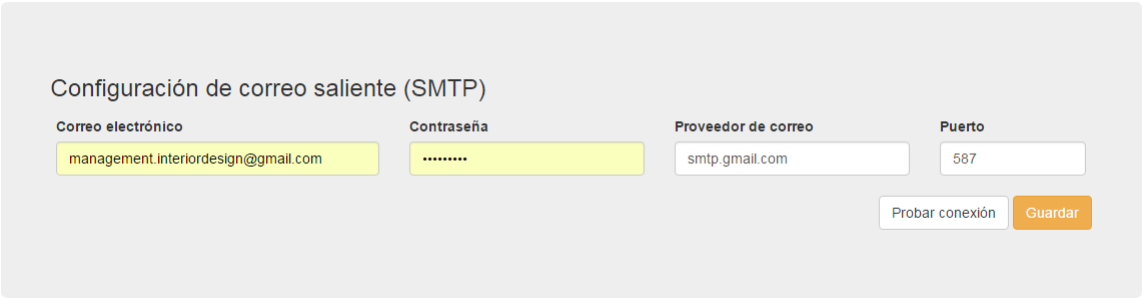

Figura 11.30: Mantenimiento - Configuración del correo saliente SMTP

11.3.7.6. Configuración de la APP de Facebook

La configuración de login con Facebook sirve por si se desea habilitar el login/registro mediante Facebook. Para ello hay que introducir los datos (ID y KEY) de nuestra APP de Facebook. Para crear una APP hay que ir a la url `https://developers.facebook.com/` y seguir todos los pasos para obtener dichos datos. También se da la opción de utilizar la APP creada como ejemplo para este proyecto. En el caso de utilizar la APP por defecto, habría que comunicar al administrador de

dicha APP el dominio bajo el que se instalará la plataforma para que sea incluido en la lista de dominios que pueden recibir peticiones de la APP.

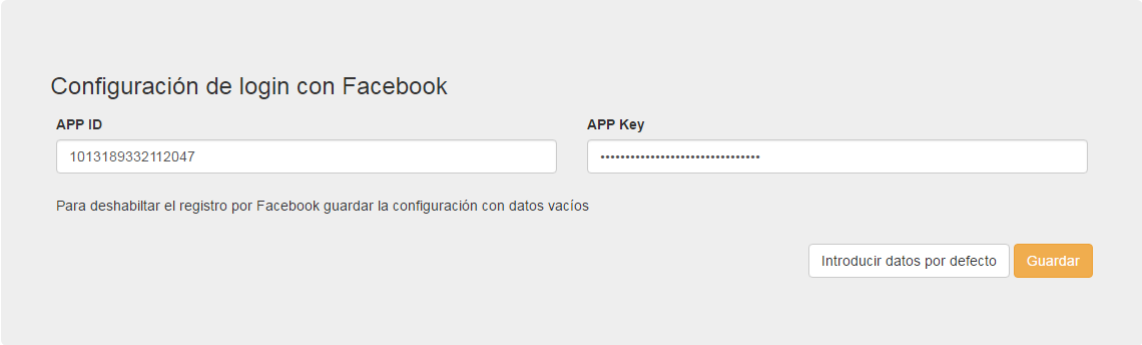

Figura 11.31: Mantenimiento - Configuración APP Facebook

11.3.7.7. Responder comentarios

Los comentarios que se envíen desde el formulario de la página principal en la sección de "Contacto", podrán responderse automáticamente desde este apartado. Hay que tener en cuenta que si el servidor SMTP no está correctamente configurado esta opción no funcionará.

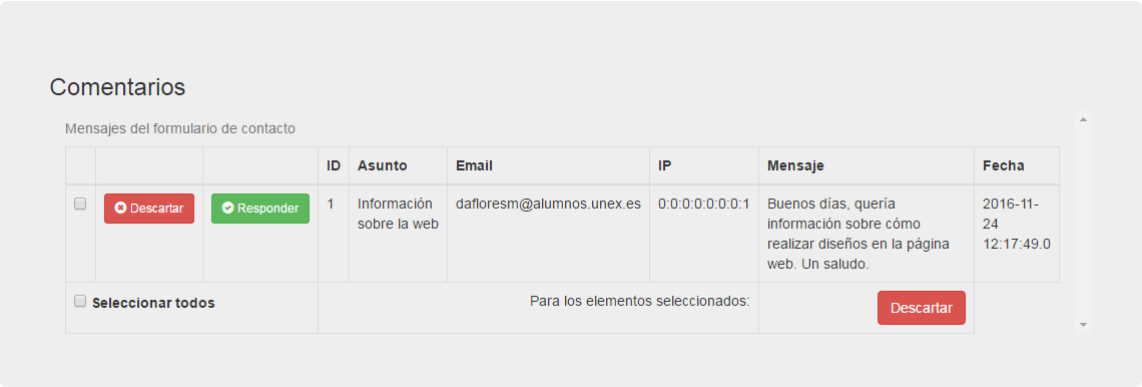

Figura 11.32: Mantenimiento - Responder comentarios

Capítulo 12

Creando los modelos: Blender

El editor de modelado en 3D utilizado para crear los objetos será Blender. Además, tenemos que incorporar la librería propia de Three.js para poder exportar la pieza creada de manera correcta.

12.1. Descarga

La versión de Blender utilizada para crear los objetos ha sido la 2.65, cuyo enlace de descarga puede encontrarse en su página oficial.

- `https://download.blender.org/release/Blender2.65/`

12.2. Instalación

Su instalación es sencilla, como la de cualquier otro programa. Basta con seguir el asistente de instalación y esperar que los archivos sean copiados en nuestro equipo.

12.3. Librería Three.js

La librería Three.js para Blender nos permitirá exportar los objetos que diseñemos en formato .js o .json, dependiendo de la versión. Este es el formato que nuestra plataforma, Interior Design, es capaz de interpretar para poder utilizar y manipular las piezas en 3D. Los pasos a seguir para su instalación serán descritos a continuación [25].

12.3.1. Descarga e instalación del módulo Three.js para Blender

Podemos seguir las instrucciones de la página de Three JS GitHub[28] para instalar el exportador de Blender, pero aquí explicaremos una forma más sencilla:

1. Descargamos ThreeJS [27] para obtener los archivos JavaScript, exportadores y otros archivos útiles. Podemos clonar este repositorio de Git y realizar un "git pull" para obtener el último y mejor código, o simplemente podemos descargar el archivo zip. Se recomienda descargar la siguiente versión del add-on, que es la que ha sido utilizada durante el proyecto:

   - `http://s2.amazonaws.com/blueprint3d/io_mesh_threejs.zip`

2. En la carpeta ThreeJS que acabamos de descargar, abrimos la carpeta “utils / exporters / blender / addons”. Creamos un archivo zip de la carpeta "io_three" haciendo clic con el botón derecho en la carpeta y comprimiendo el archivo. Esto creará un archivo zip en la misma carpeta.

3. A continuación abrimos Blender y vamos a “File -> User Preferences”. En Preferencias del usuario, haga clic en la pestaña "Add-ons". A continuación, pulse el botón "Install from File..." y navegamos hasta el archivo zip que creamos en el paso anterior y lo seleccionamos.

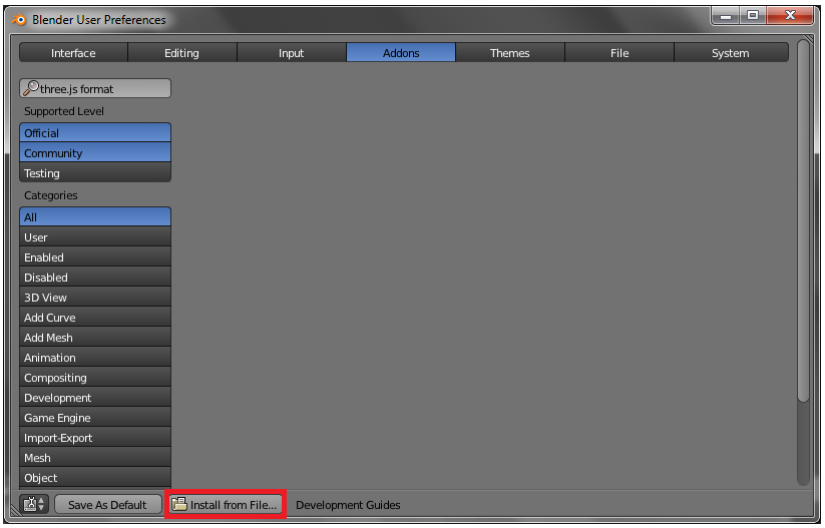

Figura 12.1: Three.js exporter. Agregando a Blender

4. Por último marcamos la opción para habilitar el add-on

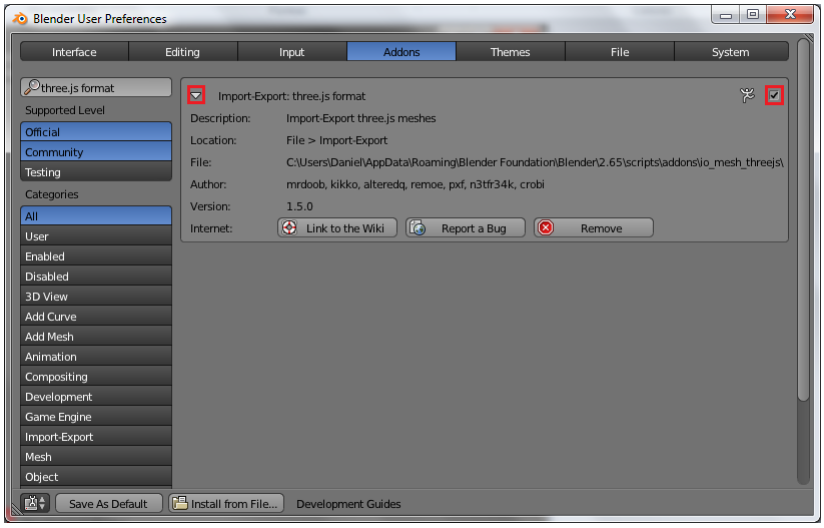

Figura 12.2: Three.js exporter. Agregando a Blender

12.4. Preparando el modelo

Hay varias cuestiones que debemos tener en cuenta a la hora de realizar nuestra pieza.

La primera de ella son las medidas. Debemos configurar Blender para que muestre las medidas según el sistema métrico internacional. Para ello seleccionaremos la opción de “Scene” de la barra de herramientas y haremos clic sobre “Metric” dentro de la sección “Units”.

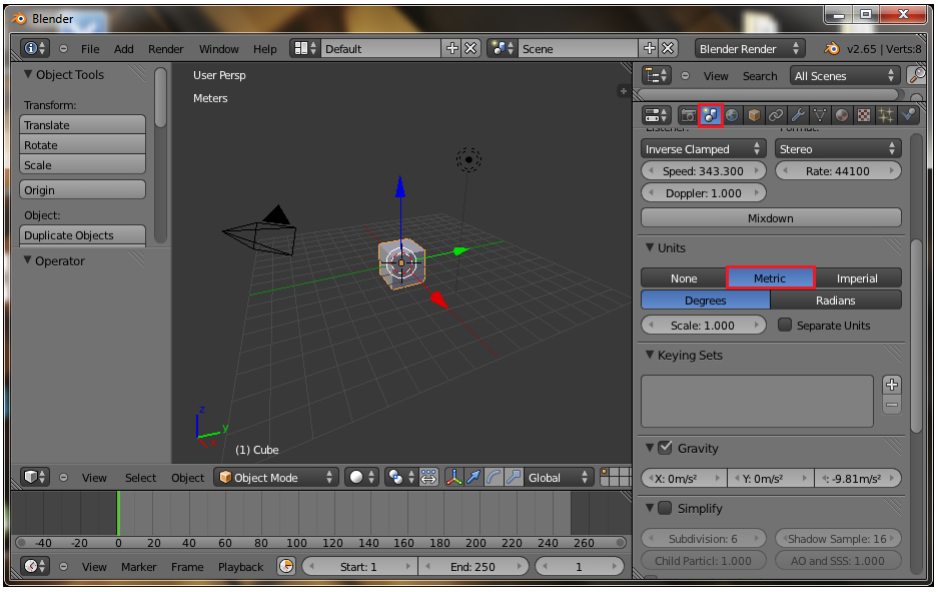

Figura 12.3: Blender - Ajustando las unidades

El siguiente aspecto a tener en cuenta son las dimensiones. Para que la plataforma interprete de manera correcta las dimensiones de las piezas, éstas deben estar multiplicadas por un factor de 100.
Supongamos que queremos diseñar un cubo de 2x2x2 (2 metros de alto, ancho y profundo), como el de la figura anterior. Si seleccionamos la pieza, y accedemos al menú contextual de transformación pulsando la letra "n", podremos especificar las dimensiones.

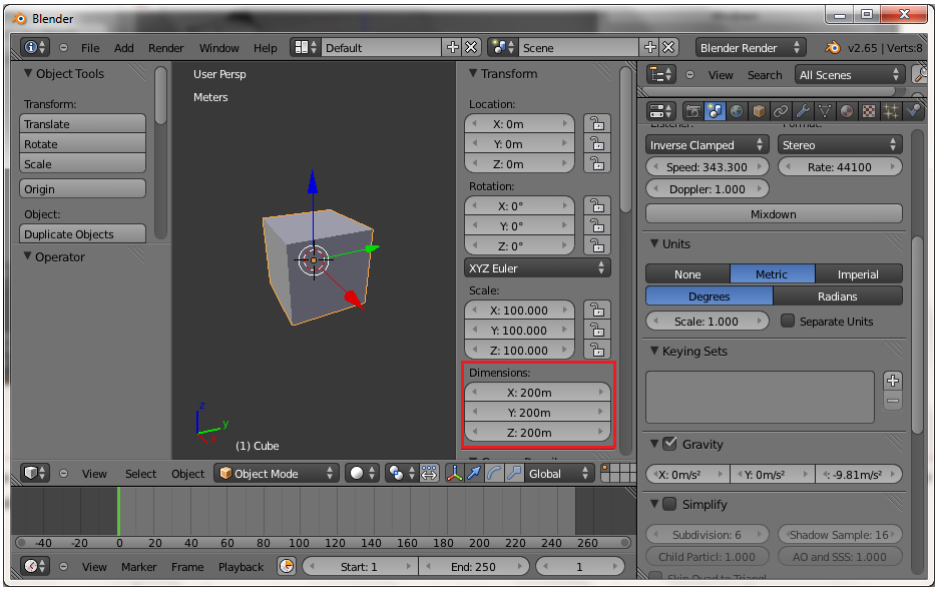

Figura 12.4: Blender - Cambiando las dimensiones

Dentro de la sección de "Dimensions" podemos especificar las dimensiones que deseamos. Introduciremos el valor 200 en los 3 campos. Aunque el valor va acompañado de una letra "m" haciendo referencia a metros, es necesario multiplicar el valor por 100 para que se vea correctamente en la plataforma. Una vez agregada a la plataforma y utilizada en los diseños, su tamaño será realmente de 2 metros, tal y como nosotros queríamos.

12.5. Exportando el modelo

Una vez hemos diseñado nuestra pieza, es hora de exportarla. Para ello utilizaremos el complemento que hemos instalado en el paso anterior, de manera que la plataforma sea capaz de interpretarla.
El exportador exportará un objeto a la vez, por lo que al seleccionar el modelo que desea exportar, creará un archivo JS, que representa los datos para reconstruir su

modelo.
Seleccione cada objeto que desee exportar e vaya a “File -> Export -> Three.js (.js). Esta nomenclatura puede variar en función de la versión de Blender y Three.js que se esté utilizando y en vez de aparecer “js” aparecerá “json”, pero el resultado es el mismo. Aún así se recomienda encarecidamente utilizar las versiones arriba descritas tanto para Blender como para el complemento exportador de Three.js.

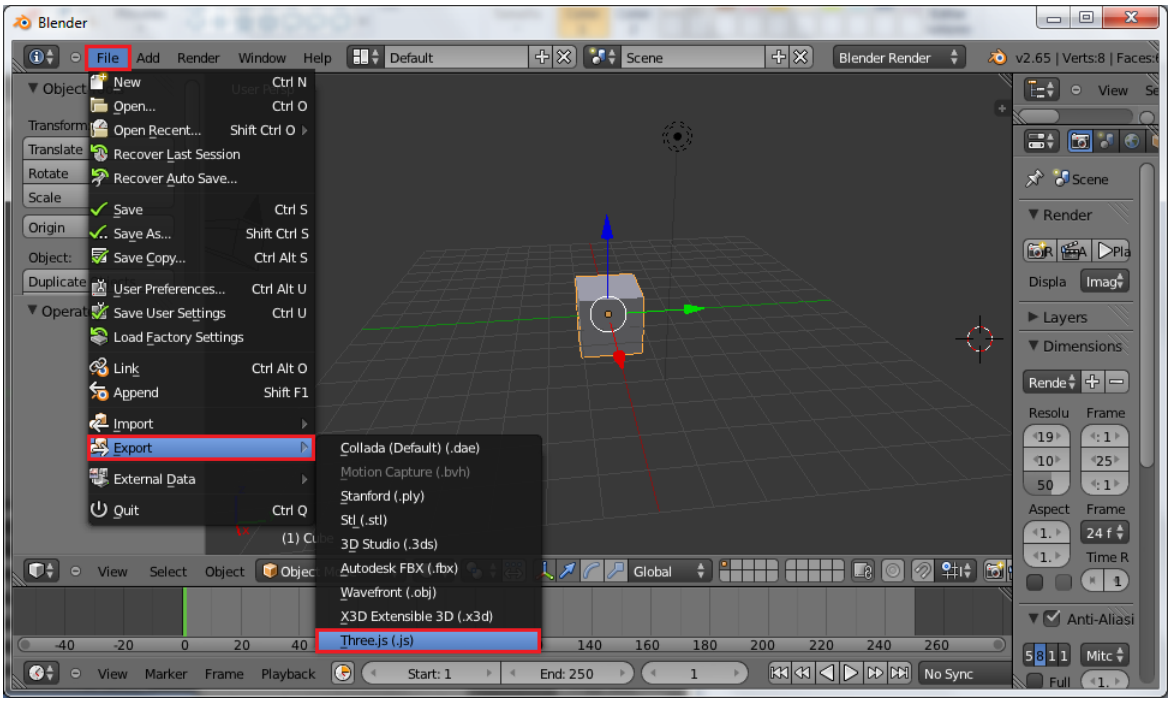

Figura 12.5: Blender - Exportando el modelo (1/2)

A continuación nos aseguraremos de marcar las siguientes opciones:

- Geometry:
  - Vertices
  - Faces
  - Normals
- Materials:
  - UVs
  - Colors
  - Materials

- Settings:
  - Flip YZ
  - Scale: 1.00
- Scene:
  - Embed meshes
- Settings (experimental)
  - All meshes

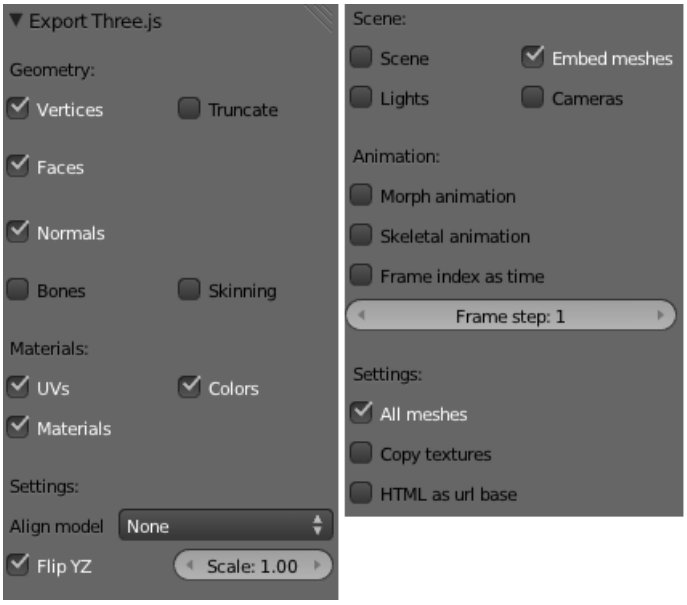

Figura 12.6: Blender - Exportando el modelo (2/2)

Finalmente hacemos clic sobre el botón "Export Three.js" y se creará un archivo en la localización indicada y con el nombre deseado.
Nuevamente se hace hincapié en que estas opciones pueden variar dependiendo de las versiones que se estén utilizando, pero básicamente esta es la forma de exportar cualquier modelo en cualquier versión.

12.6. Añadiendo la pieza a la plataforma

Para añadir la pieza recién creada seguiremos los pasos descritos en secciones anteriores donde se detalla minuciosamente los pasos a seguir.

12.7. Utilizando la pieza en nuestros diseños

Para comprobar que la pieza se ha creado correctamente y se ha introducido en la plataforma, crearemos un diseño de ejemplo y la agregaremos.

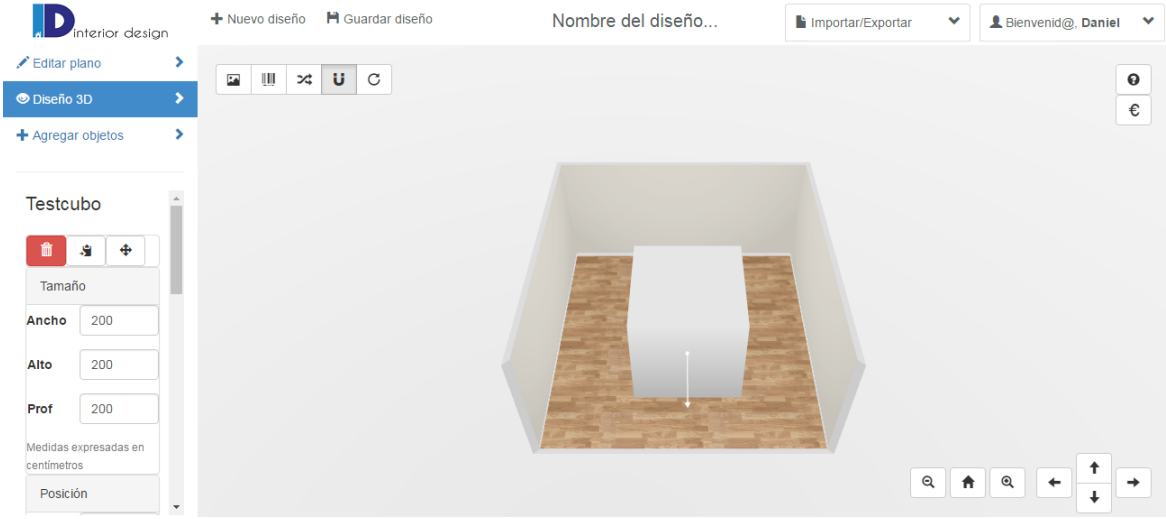

Figura 12.7: Blender - Utilizando la pieza en los diseños

Si nos fijamos en el tamaño de la pieza de la figura anterior, las tres dimensiones, alto, ancho y profundidad son de 200 centímetros, que se corresponden con los 2 metros que queríamos darle a nuestra pieza al inicio de su diseño.
En caso de que la pieza no se muestre correctamente deberemos rediseñarla, cambiando su tamaño, texturas, etc, hasta conseguir el efecto deseado.

12.8. Descargando modelos de la red

También podemos encontrar algunos modelos ya hechos en Internet. Algunas de las páginas consultadas y de los que se han descargado y probado algunos modelos dentro de la plataforma son:

- `http://www.blendswap.com`
- `https://www.blender3darchitect.com/category/furniture-models/`
- `http://blender-archi.tuxfamily.org`

Asimismo, no todas las piezas serán adaptadas por nuestra plataforma porque pueden contener propiedades especiales como animaciones propias o varios tipos de texturas,

por lo que habría que analizar el fichero .js descargado. Aún así, se recomienda incorporar a la plataforma la pieza descargada de alguno de estos sitios e intentar agregarla a un diseño para ver su comportamiento. Otra posibilidad es editarla con Blender para intentar mejorar la compatibilidad.

12.9. A tener en cuenta

Hay varios aspectos que debemos tener en cuenta a la hora de realizar nuestros modelos:

- Las versiones utilizadas, tanto de Blender como del módulo de exportación Three.js pueden influir en el comportamiento de la pieza a la hora de ser introducida en la plataforma.
- A la hora de exportar una pieza, tenemos que asegurarnos de seleccionar todas las capas que las que esté compuesta.
- Si aplicamos una textura a una pieza, deberemos añadirla también a la plataforma para que se comporte correctamente en los diseños.

Capítulo 13

Mantenimiento y explotación

13.1. Pruebas unitarias

Durante la elaboración de este proyecto se han ido realizando un conjunto de pruebas para todos y cada uno de los apartados para comprobar su correcto funcionamiento. Al ser una aplicación web, las pruebas unitarias no se han implementado utilizando JUnit, como sería lo habitual, si no que se ha ido testeando dinámicamente cada funcionalidad manualmente a través de la web y comprobando su comportamiento en el servidor.

Por lo tanto, podemos decir que no existen las pruebas unitarias como tal, si no que a medida que se iba desarrollando una nueva funcionalidad, ésta se iba comprobando para ver que su comportamiento era el adecuado.

Algunas de las pruebas realizadas han sido:

- Registro y login de usuarios: comprobar que un usuario se registra y accede correctamente a su cuenta así como la modificación de los datos existentes.
- Creación, modificación y eliminación de registros: para todos los apartados que requieran las operaciones básicas de creación, modificación y eliminación de registros (piezas, categorías, usuarios, etc.) se ha comprobado que la aplicación trabaja de manera satisfactoria en todos los casos posibles.
- Correo electrónico: el servidor de correo saliente con SMTP es uno de los aspectos en los que más se ha trabado para que la plataforma pueda comunicarse de manera automática con los usuarios a la hora de registrarse o para responder comentarios por parte del administrador.

- Facebook: la aplicación de Facebook que posibilita el registro y acceso de usuarios también a contado con su juego de pruebas para comprobar su correcto funcionamiento.

- Entorno 3D: esta ha sido sin duda la parte más compleja de probar, pero también en la que más hemos hecho hincapié. Era fundamental que los diseños y las piezas de las que están compuestos trabajasen de manera adecuada, por eso se ha probado cuidadosamente que las piezas tengan un comportamiento correcto y que puedan ser manipuladas como nosotros deseamos. Además, se ha trabajado en el formato que deben tener los diseños para que puedan ser almacenados y recuperados tanto de la base de datos como de archivos de texto.

- Base de datos: los datos son lo más importante para nosotros. De nada vale una gran infraestructura debidamente implementada si los datos con los que trabaja no son almacenados y recuperados de manera adecuada. Por eso se ha realizado el diseño de la base de datos en base a las necesidades de nuestro cliente, y cubriendo todas y cada una de las necesidades, para que en conjunto con la web formen una gran herramienta de trabajo tanto para los administradores, como para los usuarios.

- ...

13.2. Mantenimiento

El mantenimiento de la plataforma es sencillo. Gracias al panel de administración, se pueden gestionar fácilmente todos los apartados de los que la web está compuesta. De esta manera podemos realizar el mantenimiento de piezas, texturas, grupos, usuarios, etc. de una manera sencilla y eficiente.

Además el Log de operaciones nos ayudará a recordar todas y cada una de las operaciones que los administradores han ido realizando.

Aún así, se recomienda prestar especial atención a la sección de “Mantenimiento” del panel de administración, en la que podremos eliminar los ficheros temporales que ya no se necesiten, o eliminar piezas y texturas definitivamente de la plataforma para liberar espacio.

13.3. Explotación

En cuanto a la explotación, este proyecto está pensado para que pueda desplegarse en cualquier entorno compatible de una manera sencilla y rápida. Para eso se ha implementado un instalador, que ayudará al administrador del sitio a configurar la aplicación para su utilización rápidamente.

Además pensamos que la utilización por parte de los usuarios va a resultar bastante sencilla e intuitiva, lo que los motivará a que sigan utilizándola cada vez más. También se cuenta con esta documentación que trata de explicar todos los pasos seguidos a lo largo del desarrollo del proyecto, tanto para el administrador o programador que desee hacer modificaciones, como para los usuarios que necesiten ayuda para realizar alguna función.

Esta aplicación va dirigida a usuarios que necesiten realizar un diseño de interiores de una manera sencilla y rápida, por eso creemos que con una acertada difusión de la misma podríamos llegar a bastantes personas para ayudarles a cubrir su necesidad.

Capítulo 14

Problemas encontrados

Durante la realización de este proyecto, han sido varios los problemas que han ido surgiendo.

El primero y más importante, es el desconocimiento de las tecnologías 3D que pueden usarse en un entorno web, así como su utilización. Esto ha supuesto tener que investigar y aprender cómo funcionan las tecnologías más representativas y cuales son sus características más importantes. Gracias a la gran cantidad de documentación y ejemplos prácticos que están en Internet, se consiguió superar este problema satisfactoriamente y elegir la tecnología que más se adaptase a nuestras necesidades.

Una vez se decidió la tecnología a utilizar, había que aprender todo lo posible sobre ella. Esto llevó su tiempo, mucho tiempo. WebGL en combinación con Three.js es una tecnología complicada de utilizar si no se conoce y nunca se ha utilizado. Nuevamente, gracias a la gran comunidad de desarrolladores que podemos encontrar en la red, fuimos aprendiendo todo lo necesario para desarrollar nuestra aplicación. A destacar en este apartado, está el hecho de trabajar con las distancias entre dos puntos en un entorno 3D. Medir una distancia entre dos puntos es sencillo, pero cuando estamos en un entorno en tres dimensiones la cosa cambia. Y si además le añadimos que el tipo de cámara utilizada o la posición de la misma puede influir en el resultado de la medición, la cosa se complica. Este ha sido el aspecto más complicado de resolver, el de medir la distancia entre dos puntos, o entre una pieza y los muros. Three.js es una librería relativamente nueva y sin terminar. Está en constante cambio, lo que provoca que lo que se implementa hoy, posiblemente en la siguiente versión sufra cambios. De ahí que una de las futuras mejoras sea la de actualizar Three.js a la última versión estable, una vez exista dicha versión.

La investigación de proyectos existentes fue bien. Tan bien que se encontraron proyectos que implementaban algunas de las funcionalidades que nosotros requeríamos, que nuestro cliente nos había pedido. Pero el problema es que no es nada fácil interpretar código escrito por otra persona. Dedicándole tiempo, se fue descubriendo como funcionaban estos proyectos, y si podían ser reutilizados o modificados para adaptarlos a nuestras necesidades.

En general el proyecto se ha ido desarrollando por fases, tal y como se ha descrito a lo largo de este texto. La programación del servidor no ha tenido especial problema, aunque si ha sido muy laboriosa. Y decimos laboriosa porque hemos intentado cuidar hasta el último detalle a la hora de escribir el código. Un código limpio y claro a la vista de cualquier desarrollador siempre se entiende mejor y queda más elegante. También hemos intentado prestar especial atención al rendimiento de la aplicación, realizando únicamente las tareas necesarias en la base de datos o en el servidor cuando se realicen peticiones por parte de los usuarios, sin realizar tareas adicionales. Lo mismo sucede en la parte del cliente, donde el uso de Bootstrap para diseñar las páginas web cuida la interfaz de usuario y la hace agradable e intuitiva para los usuarios. Algunos de los problemas surgidos en esta parte ha sido a la hora de realizar peticiones Ajax para actualizar la información de la página en cuestión, sin necesidad de refrescarla completamente.

Otro de los apartados en los que nos hemos encontrado con algunos problema ha sido a la hora de exportar modelos (objetos) realizados con Blender. El proyecto base sobre el que hemos trabajado, Blueprint3D, nos facilitaba por defecto bastantes objetos para añadir a los diseño. Pero uno de las necesidades era la de poder diseñar objetos personalizados. El problema viene debido a la diversidad de versiones, tanto de Blender como del plugin de exportación para Three.js que podemos encontrar en la red. Finalmente, después de probar varias combinaciones de versiones, y varias opciones de exportación, se ha dado con una solución óptima que permite a los usuarios exportar sus propios modelos para que puedan ser incluidos en la plataforma y utilizados en los diseños. Como siempre, hay varias soluciones posibles en cuanto a la exportación se refiere, y esta variará en función a la cantidad de capas del modelo, si está dividido en varias partes, si tiene textura, etc. Nosotros hemos probado con modelos simples, y añadiendo una textura, el resultado ha sido bueno. Aún así, hay que trabajar más en este aspecto, y como se propone en las futuras líneas de investigación, uno de los objetivos futuros sería el poder exportar las piezas desde otras

aplicaciones como Autocad o SolidWorks, que aunque son soluciones software que necesitan licencia de pago, son de las más utilizadas en el mundo del diseño 3D.

Uno de los problemas más importantes, y que queda abierto a una futura solución está en el servidor, concretamente en el servicio MySQL. MySQL por defecto cierra las conexiones después de un periodo de inactividad, por lo que si no se accede a la plataforma en un largo periodo de tiempo, se muestra un error cuyo mensaje nos dice que no se puede conectar con la base de datos. Curiosamente, al intentar acceder a la web de nuevo, se restablece la conexión. Por lo tanto este problema debería resolverse en el futuro. Actualmente, se ha tomado una solución alternativa, y es la de reiniciar el servicio MySQL en el servidor automáticamente cada cierto tiempo, lo que impide que el error aparezca, pero naturalmente, no es la mejor de las soluciones.

Capítulo 15

Conclusiones

El diseño de interiores es una profesión polifacética en la cual la creatividad y las soluciones técnicas se aplican dentro de una estructura para construir un ambiente interior (espacio físico). Estas soluciones son funcionales, mejoran la calidad de vida y cultura de los ocupantes y son estéticamente atractivas. Si esto lo combinamos con las posibilidades que las tecnologías 3D nos ofrecen hoy en día, conseguimos una potente herramienta capaz de recrear cualquier diseño que se nos ocurra a través de un ordenador. El auge que este tipo de soluciones de diseño de interiores en 3D tiene en la actualidad, proporciona un sinfín de posibilidades en cuanto a la personalización de espacios se refiere, ya que nos permite recrear de manera sencilla cualquier idea que tengamos de un espacio físico. Además, la aparición de nuevas tecnologías como la realidad virtual o realidad aumentada, nos proporcionan nuevas funcionalidades con las que podemos mejorar todavía más la experiencia de usuario.

En este proyecto, Metodología para el Desarrollo de Simuladores de Diseño de Interiores, se ha diseñado e implementado una aplicación que permite el diseño de interiores en 3D a través de un entorno web haciendo uso exclusivamente de software Open Source. El desarrollo de este proyecto ha conseguido cubrir todos y cada uno de los requisitos de nuestro cliente, el Centro Universitario de Mérida.

Tras la realización del proyecto, concluimos que:

1. La utilización del motor gráfico en 3D WebGL con ayuda de la librería Three.js para desarrollar la funcionalidad principal, permite a los usuarios crear sus diseños lo más reales posibles acordes con la realidad. Esto permite al usuario tener una idea de cómo puede quedar un futuro cambio en un diseño de una casa, una habitación, un despacho, etc.

2. Se ha elaborado una interfaz de usuario adaptativa por la que cada usuario podrá desplazarse de una manera intuitiva para realizar sus diseños. Una interfaz clara e intuitiva ayuda al usuario a la hora de realizar los diseños. Además, la tecnología responsive permite que la web pueda ser visualizada desde dispositivos móviles. Aunque WebGL no es una tecnología propia diseñada para dispositivos móviles, la implementación de un sitio web *responsive* se ha llevado a cabo sobre todo para el apartado de administración, donde los administradores de la plataforma podrán hacer cambios desde dispositivos móviles como smartphones o tablets.

3. Se ha integrado la plataforma con un sistema gestor de base de datos. MySQL es la base de datos encargada de almacenar y gestionar todos los datos de la aplicación tales como usuarios, piezas, texturas, etc.

4. Además, se ha tenido especial cuidado en todos los detalles, desde el diseño de la web, hasta el tratamiento de los datos. La seguridad es un apartado importante, por eso el cifrado de datos sensibles como las contraseñas de usuarios en un aspecto que se ha tenido muy en cuenta.

5. También hemos pensado en los administradores que quieran instalar la plataforma para que sus clientes puedan utilizarla. El desarrollo de un asistente de instalación, propio de gestores de contenido como Wordpress o Joomla, guiarán al administrador de la plataforma en el proceso de instalación a través de la web. Además, se plantea una instalación manual ideal para usuarios avanzados o aquellos que deseen realizar cambios en el código base.

15.1. Líneas de trabajo futuras

Quedan abiertas las siguientes líneas de trabajo:

- Optimización de la librería principal “blueprint3d.js” que permita separar la funcionalidad de la plataforma de la de “Three.js”.

- Actualización de Three.js a la última revisión, a día de hoy, 28 de diciembre de 2016, la *release* r78.

- Introducir piezas animadas. Dar a las piezas la capacidad de contener animaciones propias a través de Blender y que puedan ser reproducidas dentro de los diseños.

- Introducir iluminación en determinados puntos. Aunque los diseños cuentan con una iluminación por defecto, Three.js permite crear puntos de luz específicos en ciertas zonas de la escena.
- Sistema de realidad aumentada que proporcionan las nuevas versiones de Three.js y que darán más realismo a la hora de "pasear" por los diseños.
- Adaptación de la plataforma para que acepte otros tipos de formatos de piezas importados de Autocad o Solidworks.

Nomenclatura

CSS3 Hoja de estilo en cascada o CSS (siglas en inglés de cascading style sheets) es un lenguaje usado para definir y crear la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML). El World Wide Web Consortium (W3C) es el encargado de formular la especificación de las hojas de estilo que servirán de estándar para los agentes de usuario o navegadores.

HTML5 Sigla en inglés de HyperText Markup Language (lenguaje de marcas de hipertexto), hace referencia al lenguaje de marcado para la elaboración de páginas web en su versión 5

HTTP Hypertext Transfer Protocol o HTTP (en español protocolo de transferencia de hipertexto) es el protocolo de comunicación que permite las transferencias de información en la World Wide Web.

JSP JavaServer Pages (JSP) es una tecnología que ayuda a los desarrolladores de software a crear páginas web dinámicas basadas en HTML, XML, entre otros tipos de documentos. JSP es similar a PHP, pero usa el lenguaje de programación Java.

MVC El patrón de arquitectura MVC (Modelo Vista Controlador) es un patrón que define la organización independiente del Modelo (Objetos de Negocio), la Vista (interfaz con el usuario u otro sistema) y el Controlador (controlador del workflow de la aplicación). De esta forma, dividimos el sistema en tres capas donde tenemos la encapsulación de los datos, la interfaz o vista por otro y por último la lógica interna o controlador.

MySQL MySQL es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual GPL/Licencia comercial por Oracle Corporation y está considerada como la base datos open source más popular del mundo, y una de las

más populares en general junto a Oracle y Microsoft SQL Server, sobre todo para entornos de desarrollo web.

SGBD Un sistema gestor de base de datos (SGBD) es un conjunto de programas que permiten el almacenamiento, modificación y extracción de la información en una base de datos, además de proporcionar herramientas para añadir, borrar, modificar y analizar los datos. Los usuarios pueden acceder a la información usando herramientas específicas de interrogación y de generación de informes, o bien mediante aplicaciones al efecto.

Referencias

[1] 1000hz. A simple and user-friendly form validator plugin for bootstrap 3. *https://1000hz.github.io/bootstrap-validator/*, 2016.

[2] Anli. Bootstrap login and register forms in one. *http://azmind.com/2015/11/06/bootstrap-login-register-forms-templates/*, 2015.

[3] Apache. Apache maven. *https://maven.apache.org/*, 2016.

[4] Apache. Apache xampp. *https://www.apachefriends.org/es/index.html*, 2016.

[5] Blender. Blender: Creative freedom starts her. *https://www.blender.org/*, 2015.

[6] Twitter Bootstrap. Bootstrap. *http://getbootstrap.com/*, 2015.

[7] Juan Miguel Muriel Cespedes. Ciclo de vida orientado a objetos. *http://juanmurielc.blogspot.com.es/2012/05/ciclos-de-vida-orientados-objetos.html*, 2012.

[8] Patrick Dengler. Como elegir canvas o svg para tus sitios web. *http://www.desarrolloweb.com/articulos/elegir-canvas-svg.html*, 2012.

[9] Eclipse. Eclipse ide. mars and neon. *https://eclipse.org/*, 2016.

[10] Educared. Desarrollo de software. *https://www.ecured.cu/Desarrollo_de_software*, 2012.

[11] The Apache Software Foundation. Apache. *https://www.apache.org/*, 2015.

[12] The Apache Software Foundation. Apache tomcat. *http://tomcat.apache.org/*, 2015.

[13] furnishup. Blueprint3d git. *https://github.com/furnishup/blueprint3d*, 2015.

[14] Google. Google chrome. *https://www.google.com/chrome*, 2015.

[15] Khronos Group. Opengl es 2.0 for the web. *https://www.khronos.org/webgl/*, 2014.

[16] María Angélica Guerrero. Ciclo de vida orientado a objetos. *https://ciclodevidasoftware.wikispaces.com/CICLO DE VIDA ORIENTADO A OBJETOS*, 2009.

[17] Don Ho. Notepad++. *https://notepad-plus-plus.org/*, 2016.

[18] Joe Javapapers. Java facebook login with oauth authentication. *http://javapapers.com/java/java-facebook-login-with-oauth-authentication/*, 2014.

[19] JavaScript. Ready to try javascript? *https://www.javascript.com/*, 2015.

[20] Jason (blackjk3) Kadrmas. Fabrication tool for three.js. *https://github.com/blackjk3/threefab*, 2011.

[21] Lupe. La importancia del modelado 3d. *http://materialdibujoindustrial-lupe.blogspot.com.es/*, 2012.

[22] LyX. Lyx. the document processor. *https://www.lyx.org/*, 2016.

[23] Bartolomé Sintes Marco. Instalación y uso de xampp en windows. *http://www.mclibre.org/consultar/php/otros/in_php_instalacion.html*, 2015.

[24] Microsoft. Microsoft. *http://www.microsoft.com*, 2016.

[25] Mike. Webgl and threejs using blender models. *http://www.electronicarmory.com/articles/exporting-blender-models-to-threejs-webgl/*, 2016.

[26] Mozilla. Mozilla firefox. *https://www.mozilla.org/es-ES/firefox/*, 2015.

[27] Mr.doob. three.js. *https://github.com/mrdoob/three.js*, 2016.

[28] Mr.doob. Three.js exexport para blender. (.js, .json file). *https://github.com/mrdoob/three.js/tree/master/utils/exporters/blender*, 2016.

[29] MySQL. Mysql database. *https://www.mysql.com/*, 2015.

[30] Oracle. Java jre and java jdk solutions. *http://www.oracle.com/technetwork/es/java/index.html*, 2015.

[31] Oracle. Java server pages technology: General information. *http://www.oracle.com/technetwork/java/javaee/jsp/index.html*, 2016.

[32] Javier Pastor. La guerra de la realidad virtual: 2016. *http://www.xataka.com/realidad-virtual-aumentada/la-guerra-de-la-realidad-virtual-2016-ya-esta-aqui-comparativa-a-fondo-de-todas-las-opciones*, 2016.

[33] Paulmasson. Threarquitectura. the mit license. *https://github.com/mrdoob/three.js/blob/master/LICENSE*, 2015.

[34] phpMyAdmin. Phpmyadmin. bringing mysql to the web. *https://www.phpmyadmin.net/*, 2015.

[35] The Gantt Project. Herramienta de desarrollo de diagramas de gantt. *https://www.ganttproject.biz/*, 2016.

[36] RaspberryPi. Raspbian. *https://www.raspberrypi.org/downloads/raspbian/*, 2015.

[37] Carlos Alberto Roman Zamitiz. Análisis y diseño orientados a objetos con uml. *http://profesores.fi-b.unam.mx/carlos/aydoo/conceptos_ oo.html*, 2013.

[38] sadanary. Modelo cliente-servidor desde el punto de vista de la arquitectura. *http://docente.ucol.mx/sadanary/public_ html/bd/cs.htm*, 2012.

[39] Dasari Srinivas. User email registration using java, jquery and mysql. *http://blog.sodhanalibrary.com/2015/11/user-email-registration-using-java.html*, 2015.

[40] Sublime Text. Sublime text: The text editor you'll fall in love with. *https://www.sublimetext.com/*, 2016.

[41] Three.js. three.js - javascript 3d library. *https://threejs.org/*, 2015.

[42] uc3m. Patrón de arquitectura modelo vista controlador (mvc). *http://www.lab.inf.uc3m.es/ a0080802/RAI/mvc.html*, 2012.

[43] W3C. Css3. the web style. *https://www.w3.org/Style/CSS/current-work*, 2012.

[44] W3C. Html5. the standard web. *https://www.w3.org/TR/html5/*, 2013.

[45] Wikipedia. Three.js. *https://es.wikipedia.org/wiki/Three.js*, 2013.

[46] Wikipedia. Backend y frontend. *https://es.wikipedia.org/wiki/Front-end_y_back-end/*, 2015.