UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Aplicación web para el diseño de drones modulares

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Autor: Álvaro Casado Coscollá

Tutor: Eduardo Vendrell Vidal

Curso 2017-2018

Resum

L'objectiu d'aquest projecte és el desenvolupament d'una aplicació web per al disseny i visualització de drones a partir de models tridimensionals de components modulars. Addicionalment, es planteja la creació d'una plataforma social en la qual els usuaris comparteixen els seus dissenys i peces personalitzades amb la resta de la comunitat amb l'objectiu d'ampliar les possibilitats de l'aplicació.

Aquest projecte s'emmarca dins de Beewo, una iniciativa d'emprenedoria que té per objectiu acostar als usuaris més jóvens al món de la programació per mitjà de paquets educacionals per a construir un dron personalitzat a partir de peces impreses en 3D. Per este mateix motiu, s'ha implementat un visor de realitat augmentada perquè els usuaris puguen observar les seues pròpies creacions i les d'altres membres de la comunitat de la manera més atractiva i senzilla possible.

Paraules clau: gràfics per computador, web, Ruby on Rails, threejs, simulació, 3D, modelatge geomètric

Resumen

El objetivo de este proyecto es el desarrollo de una aplicación web para el diseño y visualización de drones a partir de modelos tridimensionales de componentes modulares. Adicionalmente, se plantea la creación de una plataforma social en la que los usuarios compartan sus diseños y piezas personalizadas con el resto de la comunidad con el objetivo de ampliar las posibilidades de la aplicación.

Este proyecto se enmarca dentro de Beewo, una iniciativa de emprendimiento que tiene por objetivo acercar a los usuarios más jóvenes al mundo de la programación por medio de paquetes educacionales para construir un dron personalizado a partir de piezas impresas en 3D. Por este mismo motivo, se ha implementado un visor de realidad aumentada para que los usuarios puedan observar sus propias creaciones y las de otros miembros de la comunidad de la forma más atractiva y sencilla posible.

Palabras clave: gráficos por computador, web, Ruby on Rails, threejs, simulación, 3D, modelado geométrico

Abstract

This project aims to develop a web application for designing and visualizing of drones, which are built with three-dimensional models of modular components. Additionally, the creation of a social platform is proposed thus users are able to share their designs and custom pieces with the rest of the community so that the possibilities of the application increase significantly.

This project is part of Beewo, an entrepreneurship initiative with the ambition to bring programming closer to the youngest users through educational packages to build a custom drone from 3D printed pieces. Likewise, an augmented reality viewer has been implemented so that users can observe their own creations, and those of other members of the community, in the most attractive and simple way possible.

Key words: computer graphics, web, Ruby on Rails, threejs, simulation, 3D, geometric modeling

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1
Introducción

La necesidad de los centros educativos por una educación en el marco *STEM* (Ciencia, Tecnología, Ingeniería y Matemáticas) durante los últimos años se ha visto muy acrecentada a nivel global. En Estados Unidos se crearon 8,6 millones de puestos de trabajo relacionados con este tipo de áreas de conocimiento tan solo en mayo de 2015 [1] y se espera un crecimiento de un 6,5 % entre 2014 y 2024 [2].

Muchas son las escuelas que han comenzado a integrar asignaturas de programación y tecnología en sus programas educativos para satisfacer los nuevos requisitos del mercado laboral. Lamentablemente, la mayoría de centros de enseñanza públicos no disponen de los recursos suficientes para acceder a algunos de los paquetes educacionales disponibles en el mercado, ya que su precio es muy elevado como en el caso de Lego® Mindstorms® EV3 o los paquetes de LittleBits®. Por lo general, estos paquetes tienen como objetivo la construcción de pequeños automóviles o robots a partir de piezas modulares.

El propósito de este proyecto es el diseño de una herramienta software a modo de aplicación web para el diseño de drones modulares para su posterior fabricación por medio de impresión 3D y su programación. La elección de construir drones en lugar de otros vehículos se basa en el auge de este tipo de dispositivos dentro del mercado de juguetes y productos orientados a niños y niñas. De este modo, se pretende poner a disposición de cualquier educador un conjunto de recursos para la enseñanza de temas relacionados con la programación y la algoritmia de una forma sencilla y económica, que atraiga la atención de los alumnos.

1.1 Motivación

La herramienta software desarrollada forma parte del conjunto de herramientas de Beewo y tiene por objetivo el desarrollo de *kits* de aprendizaje compuestos por piezas impresas en 3D y los componentes electrónicos necesarios para el montaje de un dron. Todos los diseños de las piezas, al igual que los componentes hardware, son abiertos y estarán disponibles en la página web del proyecto [3]. Esto implica que cualquier persona del mundo podrá descargar los modelos 3D para su impresión y dispondrá del listado de componentes electrónicos que deberá adquirir para construir su propio dron.

[1] Stella Fayer, Alan Lacey, and Audrey Watson - U.S. Bureau of Labor Statistics, *STEM Occupations: Past, Present, And Future*.

[2] U.S. Department of Commerce - Economics and Statistics Administration Office of the Chief Economist *STEM Jobs: 2017 Update*.

[3] Beewo `http://beewo.es`

En esta misma línea se plantea la implementación de este proyecto: una aplicación web para el diseño de drones modulares a partir de las piezas de un repositorio en línea. La elección de la web como plataforma de distribución de esta herramienta se debe a la esencia multiplataforma y abierta que ofrecen este tipo de arquitecturas, ya que permite la total disponibilidad del servicio en cualquier instante de tiempo y desde cualquier región geográfica, sin necesidad de instalar ningún tipo de software específico en el dispositivo del usuario. Esto permite que cualquier centro educativo acceda a la plataforma desde sus equipos, sin importar su sistema operativo.

1.2 Objetivos

El objetivo principal de este proyecto consiste en desarrollar un editor web de drones a partir de modelos tridimensionales en formato .STL de módulos predefinidos. Para ello, el usuario debe ser capaz de visualizar en todo momento el estado de su diseño, así como de añadir más módulos y guardarlo y descargarlo.

Adicionalmente, se propone el desarrollo de una plataforma social en la que los usuarios puedan compartir sus diseños y módulos personalizados con otros, creando así un repositorio mantenido por la propia comunidad de usuarios.

En definitiva, el objetivo de este proyecto es crear una aplicación web que resulte sencilla de utilizar y que permita a cualquier usuario crear, validar y compartir sus diseños de drones desde su navegador.

Para ello, estableceremos los siguientes subobjetivos:

- Permitir la creación y edición de modelos de drones a través de una herramienta software
- Ofrecer una interfaz de usuario centrada en la usabilidad y dirigida al público infantil.
- Crear un repositorio de diseños para que los usuarios compartan sus creaciones en la plataforma.
- Asegurar un buen rendimiento en los distintos navegadores y dispositivos.

1.3 Impacto esperado

Con la implementación de esta herramienta se pretende crear una aplicación muy sencilla de utilizar, de modo que cualquier usuario - principalmente niños - sea capaz de diseñar en tres sencillos pasos un dron desde cualquier dispositivo. Existen multitud de visualizadores que permiten colocar piezas en un espacio tridimensional, pero con el diseño de una herramienta específica se pretende simplificar todo el proceso de creación. Lo cual permite que todo el tiempo empleado por educadores y alumnos se centre en la parte didáctica referente a la programación del dron y no en la utilización de un software de terceros.

Este proyecto pretende crear una plataforma social para compartir modelos tridimensionales similar a otras existentes como Thingiverse® [4]. Los usuarios no sólo tendrán acceso a las piezas diseñadas para los paquetes educacionales, sino que también podrán

[4]Thingiverse http://thingiverse.com

descubrir y emplear módulos diseñados por otros usuarios en sus diseños. De este modo, la variedad de posibilidades aumenta en base a la comunidad de usuarios que comparten sus creaciones. Esto facilita que un usuario nuevo e inexperto pueda reutilizar componentes de la comunidad sin tener que diseñar sus propias piezas, lo cual requiere conocimientos de diseño asistido por computador (CAD) y modelado tridimensional.

1.4 Metodología

Durante el desarrollo de esta aplicación se ha seguido una metodología de trabajo tradicional o iterativa. A diferencia de las metodologías ágiles, este tipo de enfoque suele dividir todo el trabajo en varias fases.

En primer lugar, se hace el diseño de todos los procesos que conforman la aplicación o proyecto. A continuación se define el plan de trabajo y los requerimientos del mismo. Una vez todos los requisitos funcionales están bien definidos, se procede al diseño de la solución propuesta y a su desarrollo.

Por último se realiza un control de calidad del producto o servicio y se detectan los errores y mejoras aplicables al mismo, dando paso así a una nueva iteración. Este proceso iterativo se repite hasta obtener un producto final que cumple con todos los requisitos estipulados.

Metodología y convenciones en el código

Para la implementación de la aplicación se ha llevado un control de versiones con Git, una herramienta de código abierto que permite el trabajo colaborativo y la gestión de distintas versiones de todos los ficheros de un proyecto software.

Pongamos por ejemplo una aplicación en la que se han de implementar tres funcionalidades distintas. En la figura se muestra un posible árbol del proyecto `git`. Como se puede observar, la primera funcionalidad se implementa antes que el resto. En cambio, las dos últimas se realizan en paralelo haciendo uso de una rama adicional a la rama principal (`master`), por lo que es necesario realizar la fusión (`merge`) de ambas ramas.

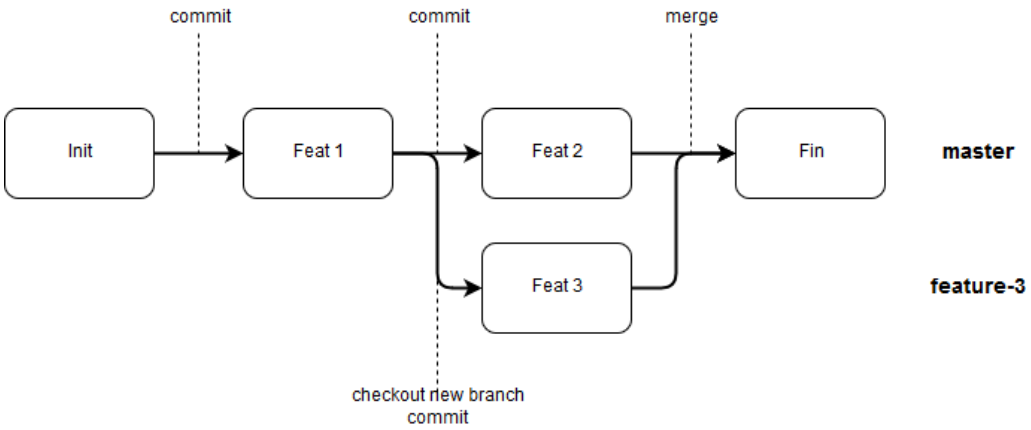


Figura 1.1: Ejemplo de flujo de trabajo con git

A la derecha, en la figura, se muestra el nombre de cada rama: `master` y `feature-3`. En la primera se han implementado las dos primeras funcionalidades y la tercera se ha implementado de forma independiente en otra rama. Para confirmar los cambios realizados se han guardado diversos `commits` o confirmaciones a lo largo del desarrollo.

De este modo es posible trabajar en varias funcionalidades de forma paralela haciendo uso de ramas. Cada rama almacena los cambios referentes a una funcionalidad realizados en el proyecto. Cuando una funcionalidad está terminada es posible fusionar su rama con la rama principal del proyecto, confirmando así esos cambios en la versión final del código.

Sin embargo, en cualquier momento es posible deshacer todos estos cambios gracias al propio control de versiones, ya que almacena todo el historial de cambios del proyecto.

1.5 Estructura

La memoria de este proyecto se estructura en nueve capítulos. En este **primer capítulo** en el que se encuentra el lector se realiza una descripción del proyecto y sus objetivos.

En el **segundo capítulo** se realiza un análisis del estado actual del campo de los gráficos tridimensionales y se compararán las soluciones de escritorio con las aplicaciones web. También se analizan soluciones similares desarrolladas por alumnos de la Escuela Técnia Superior de Ingeniería Informática (ETSINF) de la Universidad Politécnica de Valencia (UPV) y se comparan con la propuesta en este proyecto.

A continuación, en el **tercer capítulo**, se detalla el análisis de la seguridad, sostenibilidad e implicaciones legales del problema a resolver. Adicionalmente, se comentan algunas de las posibles soluciones existentes en el mercado y se describe cómo se ha diseñado esta herramienta, haciendo incapié en el plan de trabajo y el presupuesto del proyecto.

El **cuarto capítulo** comprende una descripción de la solución; desde la arquitectura del sistema a implantar hasta las tecnologías concretas empleadas para la implementación del software.

En el **quinto capítulo** se encuentra el desarrollo de la aplicación web y se describen todas las funcionalidades implementadas durante la fase de desarrollo del proyecto.

El **sexto capítulo** resume el proceso de implantación de la aplicación web en el sistema final y su configuración. En el **siguiente capítulo**, se describen todas las pruebas realizadas del sistema y los resultados obtenidos.

Por último, en el **octavo capítulo** se encuentran las conclusiones obtenidas a partir de todo el proceso de desarrollo del proyecto y las posteriores pruebas. Así mismo, en el **noveno** y último **capítulo** se indican posibles ampliaciones para el sistema que resultan de interés en este documento.

1.6 Colaboraciones

El proyecto presentado parte de la base del trabajo realizado en la asignatura de Diseño y Fabricación 3D durante el primer cuatrimestre del curso 2017-2018. Durante ese periodo de tiempo se desarrolló un sencillo visor-editor de drones modulares, que posteriormente ha sido ampliado con más funcionalidades y se ha integrado en una aplicación web completa, la cual constituye en su totalidad el proyecto que se describe en este documento.

Adicionalmente se ha contado con la colaboración de Andrea Zamora Villena para el diseño de Abee, el personaje que guía al usuario durante todo el proceso de creación.

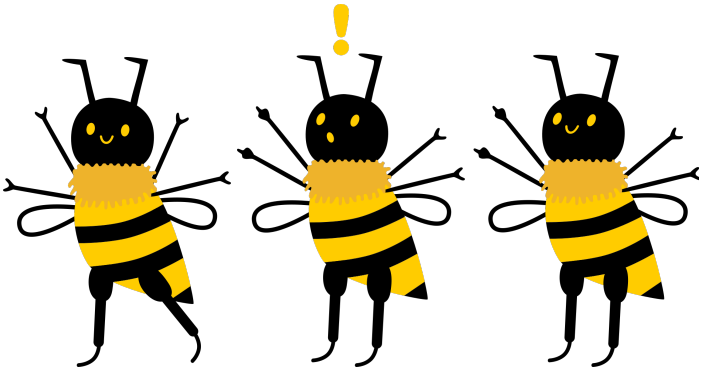


Figura 1.2: Abee

1.7 Convenciones

A lo largo de este documento el lector encontrará una serie de convenciones en la maquetación del mismo con distintas finalidades.

- Se ha tratado de evitar la inclusión de código fuente en el documento. Sin embargo, sí aparece cierta terminología y pequeños extractos que facilitan la comprensión del lector.
- El código fuente se muestra con fuente `monospace`.
- Las palabras extranjeras se remarcan en cursiva, como en el caso de la línea anterior.
- Las citas textuales externas a este documento se indican entrecomilladas.

CAPÍTULO 2
Gráficos por computador

Este capítulo no pretende ser una descripción pormenorizada de la evolución de los gráficos por computador, sino más bien un breve resumen de los hitos más relevantes durante las últimas décadas, que han llevado a la industria a la situación actual y que sirven de introducción para el análisis del estado del arte.

La historia de los gráficos por computador se remonta hasta los años 50, cuando Ben Laposky crea la primera máquina generadora de imágenes gráficas: el osciloscopio. Este dispositivo permite la visualización de las oscilaciones de las ondas a través de una pantalla. Desde entonces, los gráficos por computador han evolucionado de una manera importante, especialmente durante los últimas décadas.

En la década de los 60 proliferaron las primeras herramientas de diseño asistido por computador (CAD) como Sketchpad, una herramienta presentada por Ivan Sutherland en el M.I.T. Lincoln Laboratory en 1962. Dicha utilidad permitía al usuario realizar diseños bidimensionales y mostrarlos en una pantalla de tubo de rayos catódicos (CRT) monocromática, y sirvió de precedente para muchas otras herramientas.

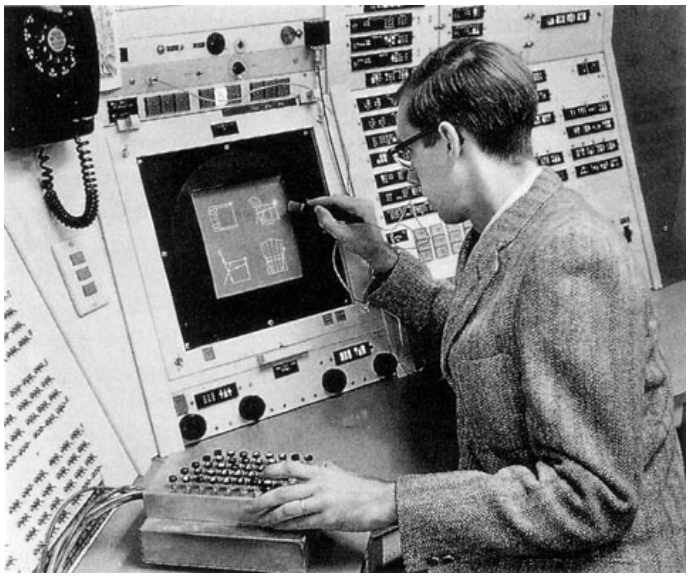


Figura 2.1: Sketchpad. Ivan Sutherland. 1962

No sería hasta los años 70 cuando los gráficos tridimensionales comenzaron a cobrar importancia. En 1976, aparece la primera animación en tres dimensiones en un largometraje, Futureworld. En la cinta se puede observar una mano y un rostro humanos en una pantalla de laboratorio. Este logro dio paso a otras producciones como Star Wars (1977), en la que aparece una reconstrucción tridimensional de la Estrella de la muerte.

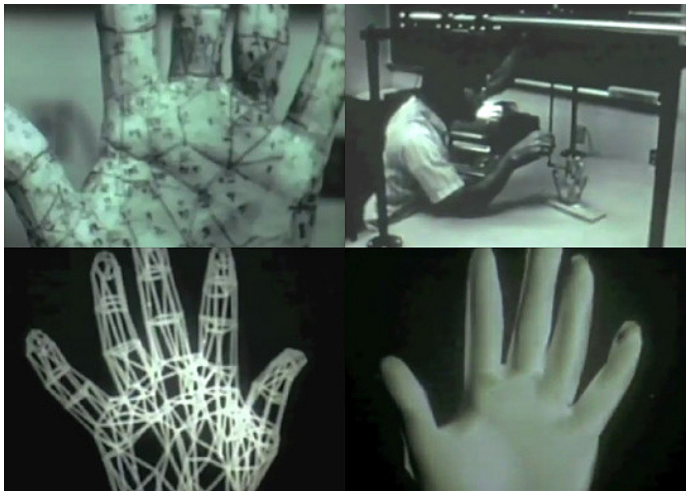


Figura 2.2: Futureworld. 1976.

Durante los finales de los años 70 y la década de los 80 se desarrollaron técnicas avanzadas de visualización de gráficos en tres dimensiones como el trazado de rayos, que permite la síntesis de imágenes tridimensionales a partir de la simulación de la trayectoria de la luz en una escena. En 1981 aparece CATIA®, uno de los primeros paquetes de software para el diseño en tres dimensiones y que se empleó 8 años después para el diseño del Boeing 777. Esta, junto con otras herramientas como AutoCAD®, sentaron las bases de las las aplicaciones industriales actuales.

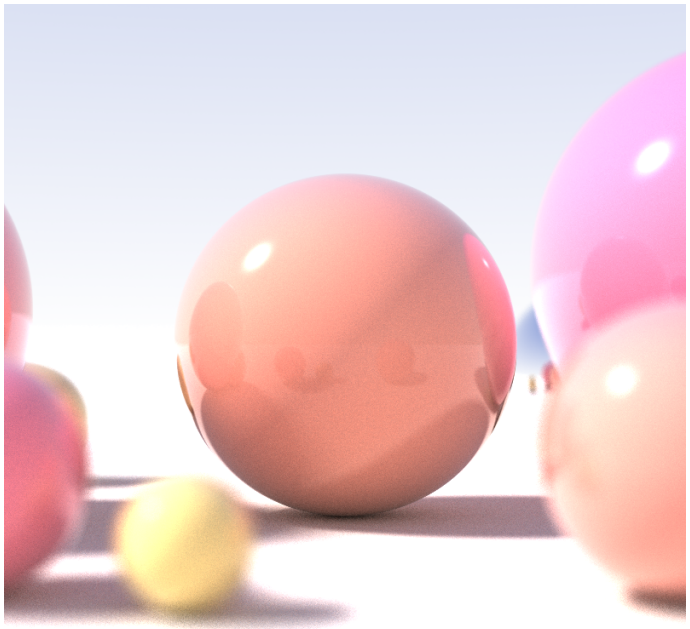


Figura 2.3: Recursive raytracing of a sphere. Tim Babb. 2008

En 1978 aparece el primer estándar para la geometría constructiva sólida o **modelado sólido**, por H. Voelcker et al. Esta técnica se emplea actualmente en numerosas herramientas CAD, como es el caso de OpenSCAD®[1], empleada en del desarrollo del presente proyecto como se podrá observar en la sección de tecnologías empleadas en el siguiente capítulo.

[1] OpenSCAD® http://www.openscad.org/

Ya en la década de los 90, los gráficos por computador pasan de ser un componente más a ser el elemento principal en el mundo del largometraje y los videojuegos. Toy Story (1995) se convierte en el primer largometraje de animación por computador.

Actualmente los gráficos por computador se emplean en una infinidad de campos más allá del diseño asistido por computador, principalmente en producciones audiovisuales, desarrollo de videojuegos y aplicaciones de visualización científica e industrial. La realidad virtual y la realidad aumentada han hecho posible una aceleración de este crecimiento por su gran atractivo entre el público.

La realidad virtual (RV o VR) permite visualizar entornos de escenas con una apariencia real, de modo que el usuario se haya inmerso en ella; mientras que la realidad aumentada (RA o AR) superpone elementos virtuales a un entorno físico existente a través de un dispositivo, como puede ser un móvil. Más recientemente se habla de realidad mixta (RM o MR), en la que se incluyen elementos virtuales a una escena real, pero teniendo en cuenta el espacio del entorno para dar la sensación de que los objetos reales y virtuales forman parte del mismo espacio.

2.1 Estado del arte

En los últimos 5 años se ha producido un rápido crecimiento en el mercado de los dispositivos móvilese que ha permitido la aparición de nuevos tipos de interacción usuario-máquina que resultan muy útiles en entornos tridimensionales a través de las pantallas táctiles. Estas nuevas posibilidades, junto con la expansión de los sistemas informáticos y la aparición de la computación en la nube, han hecho posible la aparición de numerosas aplicaciones en línea para el diseño y visualización de modelos tridimensionales. Estas herramientas permiten al usuario diseñar o visualizar objetos en tres dimensiones sin importar el hardware de su dispositivo.

Entre las tecnologías que hacen posible el renderizado en línea cabe destacar WebGL® [2] (2011), una librería implementada en JavaScript que no precisa del uso de plug-ins adicionales en cualquier plataforma compatible con OpenGL®. Actualmente, la mayoría de herramientas en línea hacen uso de esta librería estándar, en lugar de emplear software privativo como Adobe Flash® o las Java® applets.

Más recientemente se han desarrollado otras librerías como THREE.js [3], Cannon.js [4] o A-Frame [5], que permiten el desarrollo de aplicaciones más complejas, abstrayendo la complejidad del hardware gráfico al desarrollador, e incluyen motores para simulaciones de físicas y otras utilidades para crear experiencias de realidad virtual y aumentada.

A continuación se presenta una análisis de algunas herramientas desarrolladas con estas tecnologías y que resultan relevantes en el campo del diseño y prototipado rápido de modelos 3D y la simulación.

Gzweb [6] es un cliente web basado en WebGL® para Gazebo [7], un entorno libre de simulación para robótica muy popular a nivel industrial. Este cliente en línea permite interactuar con la simulación desde un navegador web sin repercutir gravemente en el rendimiento del dispositivo desde el que se emplea.

[2] WebGL® https://www.khronos.org/webgl/
[3] THREE.js https://threejs.org/
[4] Cannon.js http://www.cannonjs.org/
[5] A-Frame https://aframe.io/
[6] Gzweb http://gazebosim.org/gzweb.html
[7] Gazebo http://gazebosim.org/

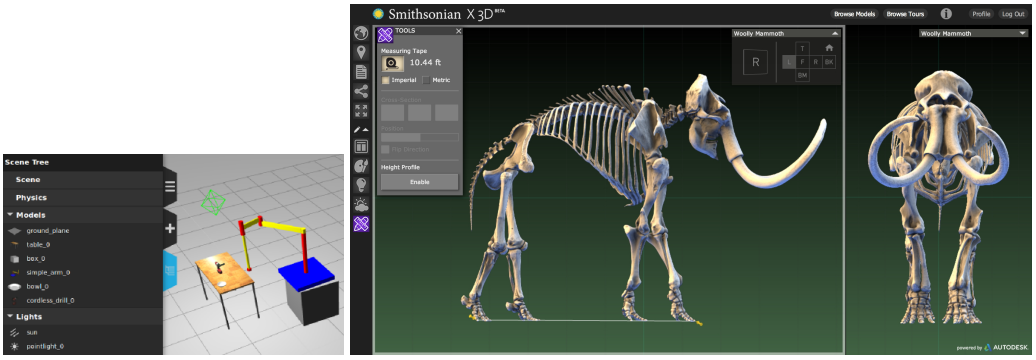


Figura 2.4: A la izquierda: Gzweb. A la derecha: Smithsonian X 3D

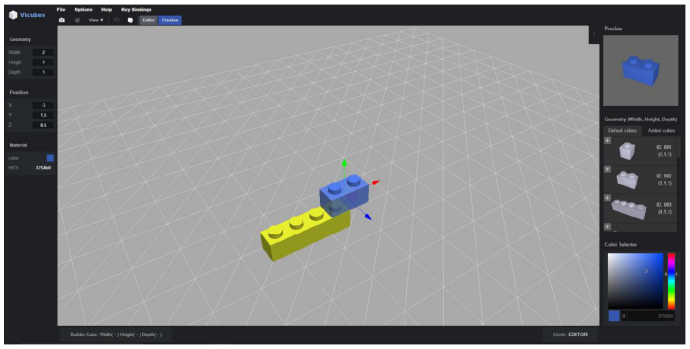


Figura 2.5: Vicubes. Desarrollado por Victor Gascó Ortiz

El Instituto Smithsonian posee un visualizador web llamado Smithsonian X 3D [8], basado en WebGL, que permite visualizar modelos generados a través de un escáner 3D almacenados en su plataforma.

Entre las herramientas CAD más populares en línea encontramos Tinkercad® [9], desarrollada por Autodesk®. Esta plataforma está enfocada a usuarios sin conocimientos previos de CAD y permite la inserción de piezas y modelos personalizados en formato STL en un visor tridimensional. También permite realizar mediciones de las piezas y combinarlas para crear nuevos objetos.

Dentro de este marco se pueden encontrar también proyectos desarrollados por miembros de la Universitat Politècnica de València, como *Videojuego de construcción mediante piezas ensamblables en navegador* - Victor Gascó Ortiz [10]. Esta aplicación permite al usuario añadir piezas de bloques a una escena para construir figuras sin solapes.

2.2 Crítica al estado del arte

Pese a que existen numerosas herramientas CAD en línea como las ya expuestas, no existe ninguna específica para diseñar drones modulares. Es por esto que nuestro propósito es hacer una aplicación lo más sencilla posible.

En relación con el último trabajo descrito, cabe destacar que pese haber detallado datos referentes al consumo de recursos de la aplicación, cabe la posibilidad de mejora

[8] Smithsonian X 3D https://3d.si.edu/

[9] Tinkercad® https://www.tinkercad.com/

[10] Videojuego de construcción mediante piezas ensamblables en navegador https://riunet.upv.es/handle/10251/64459

paralelizando la lectura de los modelos 3D, reduciendo así los tiempos de carga de la misma.

Adicionalmente, también es de interés destacar que pese a que la implementación de una red social como parte del proyecto forma parte de los objetivos del proyecto en cuestión, no parece que finalmente se realizará, ya que no se muestran pruebas que describan su implementación, ni una evaluación de su funcionamiento.

2.3 Propuesta

El propósito de este proyecto, como ya se ha comentado previamente, es desarrollar una aplicación en línea para el diseño de drones a partir de bloques. Como podemos observar, la idea guarda cierta relación con trabajos realizados anteriormente por alumnos de la Escuela Técnica Superior de Ingeniería Informática y de hecho comparte gran parte de las tecnologías que sirven de base para la implementación del mismo como en el caso del trabajo descrito anteriormente.

Sin embargo, el objetivo de nuestra aplicación es diseñar una interfaz y unos flujos de usuario que sean muy sencillos para el usuario, debido a la intención didáctica del proyecto. Por esto, durante todas las fases de diseño se da más prioridad a la usabilidad frente a la personalización, con tal de hacer una herramienta lo más sencilla posible.

Adicionalmente, y como ampliación a partir de la base del trabajo ya mencionado, la herramienta desarrollada forma parte de una aplicación web que permite que sus usuarios compartan sus diseños con otros. De este modo, la experiencia de usuario incorpora también cierto componente de comunidad con tal de reforzar el trabajo en equipo y la integración entre los alumnos.

Desde el punto de vista tecnológico, se propone la utilización de Web Workers, como hemos adelantado anteriormente, con tal de reducir los tiempos de carga de los modelos, permitiendo así la carga asíncrona y paralela de los distintos ficheros de modelos necesarios en la escena 3D.

CAPÍTULO 3
Análisis del problema

Previamente al desarrollo de este trabajo es necesario especificar el alcance del proyecto, así como una serie de requisitos para evaluar su seguridad y su sostenibilidad. A lo largo de este capítulo se describen todos los análisis y diagramas previos a la implementación de la aplicación final.

Debido al carácter educativo de esta herramienta, en las próximas secciones se hará especial hincapié en los aspectos referentes a la experiencia de usuario, ya que el usuario final de la aplicación es un público infantil, probablemente estudiantes de Educación Secundaria Obligatoria.

Todas las funcionalidades principales de la aplicación se pueden resumir en un único diagrama UML de casos de uso. En este caso, todas las funcionalidades básicas están disponibles para los alumnos.

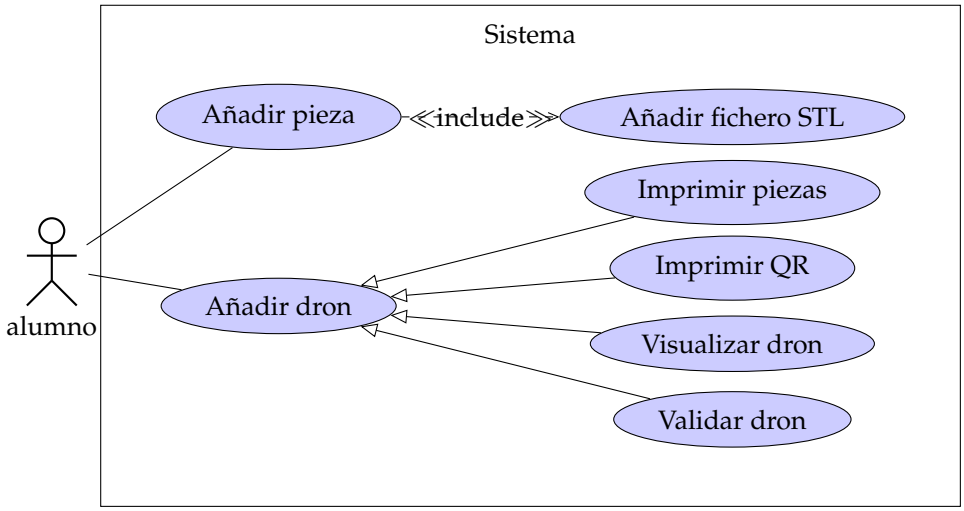


Figura 3.1: Diagrama de casos de uso del proyecto

De entre todas las funcionalidades descritas en el diagrama de casos de uso, la más importante es la creación de un dron a partir de las piezas del repositorio. Esta constituye el flujo principal de interacción del usuario con la plataforma, ya que condensa toda la experiencia de aprendizaje del alumno. Por este motivo, a continuación detallamos el diagrama de secuencia empleado para diseñar este apartado.

Cuando el usuario crea un nuevo diseño tiene la posibilidad de añadir piezas a este -siempre y cuando no comprometan la integridad estructural del mismo- hasta cierto límite definido en el sistema. Por cada pieza que añada, el sistema accede a su información

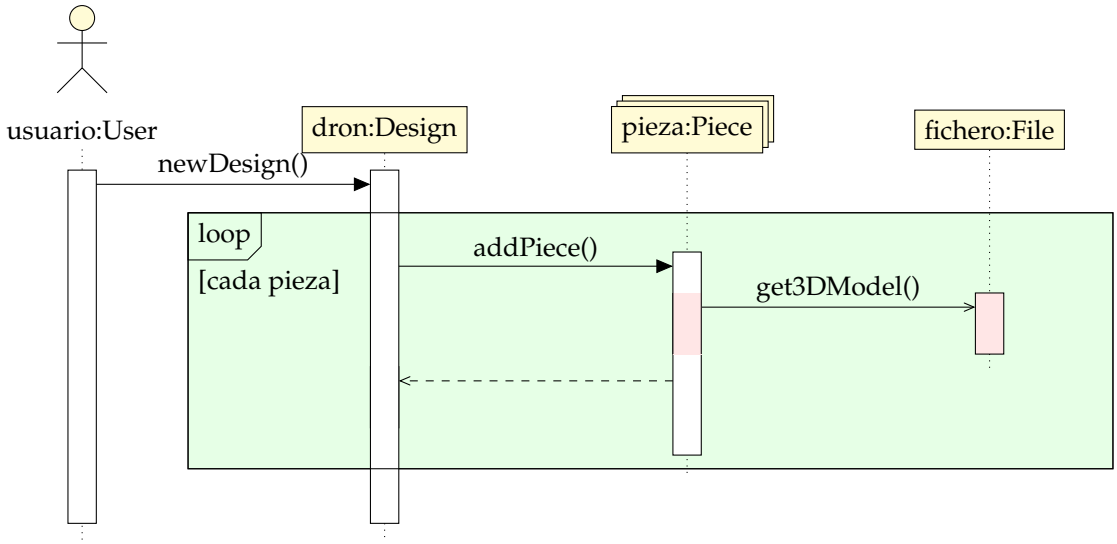


Figura 3.2: Diagrama de secuencia de la creación de un dron

almacenada en la base de datos y carga el modelo tridimensional correspondiente para su visualización en el editor.

Adicionalmente, a continuación se muestra el flujo de interacción correspondiente al proceso de visualización de un dron ya almacenado en la base de datos. Como podemos observar, cada vez que un usuario visualiza un diseño, se carga toda la información relacionada con sus componentes y los modelos 3D (en formato STL) correspondientes. La carga de estos ficheros se efectúa de manera asíncrona, como se detallará en la sección sobre diseño de la solución.

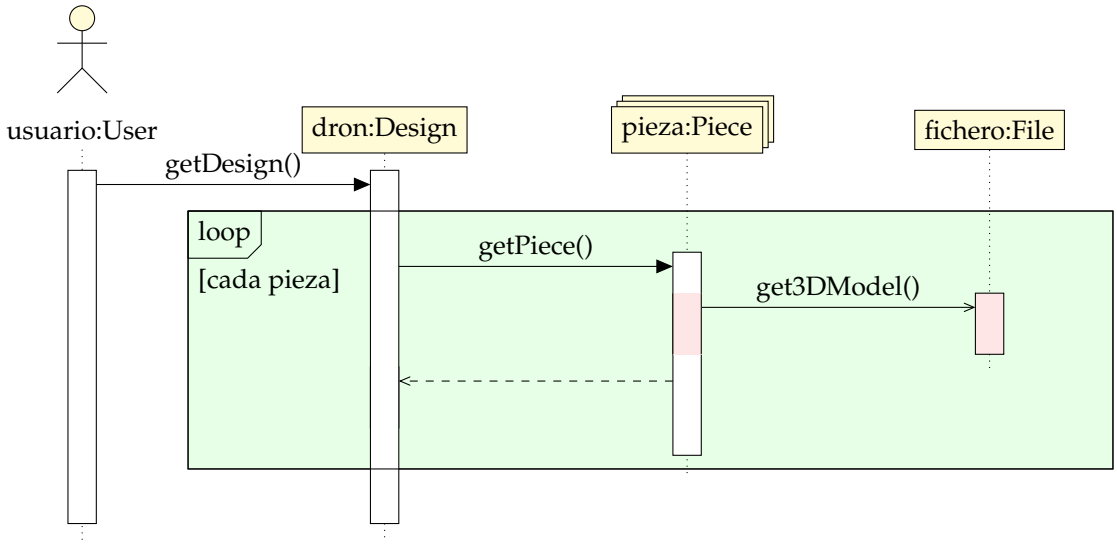


Figura 3.3: Diagrama de secuencia de la visualización de un dron

Al igual que ocurre durante el proceso de visualización, cuando un usuario accede a la información de un dron, el sistema carga toda la información relativa a este y a sus componentes para su correcta visualización en la aplicación web.

Una vez delimitado el alcance del proyecto se deben considerar otros aspectos con tal de evaluar el posible impacto de la herramienta en el entorno en el cual se implanta.

3.1 Análisis de la seguridad

La herramienta desarrollada se ha implementado empleando tecnologías web, por lo que el tratamiento de datos de los usuarios requiere de especial cuidado. Por tanto, todas las cuentas de usuario contienen la mínima cantidad de información posible, siendo toda anónima.

Por lo que respecta a la propia aplicación, es muy importante protegerla ante muchas de las vulnerabilidades más comunes, como es el caso de la inyección SQL o el *cross-site scripting* (XSS).

En el caso de la primera vulnerabilidad, el atacante es capaz de obtener información de la base de datos, a la cual no tiene acceso, por medio de agujeros de seguridad que le permiten infiltrar código malicioso a través de las entradas de alguna operación sobre la base de datos. Esto es, por ejemplo, a través de un formulario que envía datos a un servidor en el cual no se comprueba la seguridad de estos.

El *cross-site scripting* (XSS) permite a un usuario inyectar código JavaScript malicioso en una página web, de modo que puede realizar acciones como redirigir a los visitantes a una página de *phishing* o ejecutar otras acciones aún más perjudiciales en el dispositivo del cliente.

Ambas vulnerabilidades, entre otras muchas, quedan cubiertas al desarrollar la aplicación sobre un framework de código libre ya están cubiertas por las propias librerías, como Ruby on Rails [1] en nuestro caso.

Se ha escogido una librería denominada Devise [2] para la gestión de usuarios, puesto que está desarrollada por más de 500 contribuidores y su código fuente es público. Esto conlleva diversos beneficios como la reducción de la base de código de negocio de la aplicación y la robustez y fiabilidad de un proyecto auditado de forma pública.

3.2 Análisis energético

En la actualidad, uno de los retos más importantes a los que se enfrenta la sociedad actual es la correcta gestión de los recursos disponibles. Los sistemas informáticos no están aislados de esta necesidad, sino que su uso y diseño sostenible constituye uno de los pilares para asegurar el avance hacia una sociedad más limpia y ecológica.

Teniendo en cuenta este punto de vista, es posible tomar decisiones a la hora de diseñar esta herramienta y escoger el sistema en el que se implanta para reducir su impacto medioambiental. Para ello, dividiremos estas mejoras según su ámbito.

En la propia aplicación:

- Paralelizar la carga de los ficheros de los modelos tridimensionales, ya que constituyen un cuello de botella en el rendimiento de la herramienta.
- Minimizar el coste de almacenamiento de la información en la base de datos, lo que se traduce en menor coste de mantenimiento y menores tiempos de lectura y escritura.
- Emplear librerías estándares para la visualización de gráficos 3D como THREE.js o A-Frame, puesto que están optimizadas por cientos de desarrolladores.

[1] Ruby on Rails https://github.com/rails/rails
[2] Devise https://github.com/plataformatec/devise

Desde el punto de vista del sistema:

- Escoger un servidor con un buen rendimiento y bajo consumo de recursos.
- Emplear un protocolo de hibernación que suspenda el servidor en largos periodos de inactividad.

3.3 Análisis del marco legal y ético

Cualquier sistema informático se ve afectado por la legalidad vigente, por lo que es necesario conocer las obligaciones y requisitos que existen a la hora de diseñar la solución.

En el caso concreto de esta aplicación, es requisito indispensable cumplir con la legislación referente a la protección de datos. Por tanto, se debe tener en consideración:

- **General Data Protection Regulation (GDPR)**: normativa que regula la protección de datos en la Unión Europea.
- **Ley Orgánica de Protección de Datos (LOPD)**: que garantiza y protege los datos personales en España.

Con este fin, la aplicación no almacena datos personales de los usuarios referentes a su edad, género, nombre de pila, orientación sexual, ni cualquier otra información que pueda considerarse sensible.

Todos los usuarios tienen la capacidad de eliminar su perfil en cualquier momento, cumpliendo así con el **derecho al olvido**; además de la posibilidad de descargar todos sus diseños y sus ficheros asociados para cumplir con el **derecho a la portabilidad** de los datos de los usuarios.

También es necesario garantizar los derechos de **propiedad intelectual** de los autores de los diseños de la plataforma, por lo que se incluye la posibilidad de escoger una licencia al crear un diseño en la plataforma, así como indicar el autor y licencia en caso de subir un fichero de un tercero.

3.4 Soluciones posibles

Dentro del marco descrito en este capítulo es posible diseñar soluciones muy diversas que cumplan con nuestros objetivos. Por ejemplo, se podría optar por desarrollar una aplicación de escritorio en lugar de una aplicación web.

Sin embargo, la plataforma no es el único factor determinante de la estructura y el contenido de la solución. También se pueden escoger distintas tecnologías a nivel de cliente y de servidor.

A nivel de cliente se debe considerar el paradigma de programación y el rendimiento de las herramientas con las que se trabaja:

- **WebGL**: desarrollar la aplicación completamente con esta tecnología supondría un elevado coste, debido a que conlleva un desarrollo a bajo nivel, pero también reportaría un rendimiento más eficiente.
- **Java o Flash**: una aplicación implementada con estas tecnologías tendría un rendimiento mucho peor debido a las propias características de las mismas y limitaría la compatibilidad con algunos dispositivos.

- **THREE.js y A-Frame**: ambas tecnologías estan basadas en JavaScript, por lo que son compatibles con todos los navegadores actuales y proveen una capa de abstracción sobre WebGL. De este modo, es posible desarrollar una aplicación multiplataforma sobre un motor nativo con un buen rendimiento, como es WebGL.

Desde el punto de vista del servidor es posible escoger distintas plataformas para desarrollar esta herramienta. Existen multitud de *frameworks* o entornos de trabajo como Django [3], Ruby on Rails [4] o Spring Boot [5], entre otros muchos. Cada uno tiene sus ventajas y sus inconvenientes, pero son muy similares entre sí generalmente.

Adicionalmente, se debe escoger el método de almacenamiento de los datos de los modelos tridimensionales, pudiendo emplear una base de datos relacional, una base de datos documental o incluso ficheros de texto.

3.5 Solución propuesta

Para la implementación de esta herramienta, como ya se verá en el apartado diseño de la solución, se ha escogido desarrollar una aplicación web, puesto que permite desarrollar una aplicación completamente multiplataforma compatible con todos los navegadores actuales y, por tanto, con cualquier dispositivo.

A nivel interno, se ha empleado Ruby on Rails como *framework* de desarrollo para el servidor y THREE.js y A-Frame para el código del cliente. De este modo es posible desarrollar una aplicación a un nivel de abstracción suficiente para construir una base de código legible y concisa, con un buen rendimiento debido a que ambas se ejecutan sobre WebGL.

Para la persistencia de los datos se ha escogido una base de datos relacional, mientras que los ficheros de los modelos tridimensionales se almacenan en un servicio web de almacenamiento en la nube, puesto a que esto acelera el rendimiento, comparado con incrustar los ficheros en la propia base de datos.

3.6 Plan de trabajo

El desarrollo de este proyecto se ha estructurado en tres fases de trabajo: diseño y análisis de la solución, desarrollo e implantación del sistema, y validación. Tras esta fase de validación se ha sucedido una nueva iteración con las mismas fases de trabajo con tal de mejorar los errores detectados en la aplicación.

En la figura 3.4 se detallan todas las fases del trabajo desarrolladas, así como los distintos entregables que se han realizado a lo largo de todo el proceso. Como se puede observar, la primera iteración ha supuesto un coste mucho más elevado que las dos posteriores debido a la cantidad de funcionalidades a implementar en esta. El resto de iteraciones simplemente consisten en la resolución de problemas que surgen durante la implementación.

[3] Django https://www.djangoproject.com/
[4] Ruby on Rails https://rubyonrails.org/
[5] Spring Boot https://spring.io/

3.7 Presupuesto

En el desarrollo de este proyecto se considera una jornada de 5 horas laborales para todas las tareas descritas en la sección anterior. Sumando todas las horas de los cinco días laborables de cada semana durante cuatro meses se obtiene un total de 400 horas de trabajo. De este modo, considerando el salario medio de un programador junior en España (8,74€ por cada hora), se precisan de 3495€[6] para pagar los gastos asociados a las horas empleadas.

Para el desarrollo de la herramienta y su implementación se ha empleado un equipo personal, por lo que únicamente se debe computar su coste energético. Suponiendo un gasto de 60 watios, el coste de su uso por hora se sitúa en torno a 0,90 céntimos (0,009€), por lo que a lo largo de 4 meses el gasto energético es de 3,60€ , un coste despreciable dentro del cómputo global.

El servidor en el que se ha implantado el sistema, así como el servicio de almacenamiento de Amazon S3 conlleva un coste ligado al consumo de recursos asociado a la aplicación. En el caso de este proyecto el coste es nulo, ya que el plan de pago gratuito de ambos servicios es suficiente para suplir la demanda de la herramienta.

Como podemos observar, el coste del proyecto se sitúa aproximadamente en 3500€ para pagar el desarrollo de la herramienta (850€ al mes). A esta cifra se le deben añadir el coste de la contratación del dominio beewo.es (10€ anuales) y un pequeño fondo de reserva de 500€ para posibles complicaciones durante el desarrollo que conlleven la realización de horas extra. Con todo esto, el coste final de todo el proyecto asciende a 4008,60 euros.

Personal	3495 €
Coste energético	3,60 €
Dominio	10 €
Fondo de reserva	500 €
Total	4008,6 €

Tabla 3.1: Tabla resumen del presupuesto del proyecto

[6] Datos extraídos de https://www.indeed.es/salaries/Programador/a-junior-Salaries

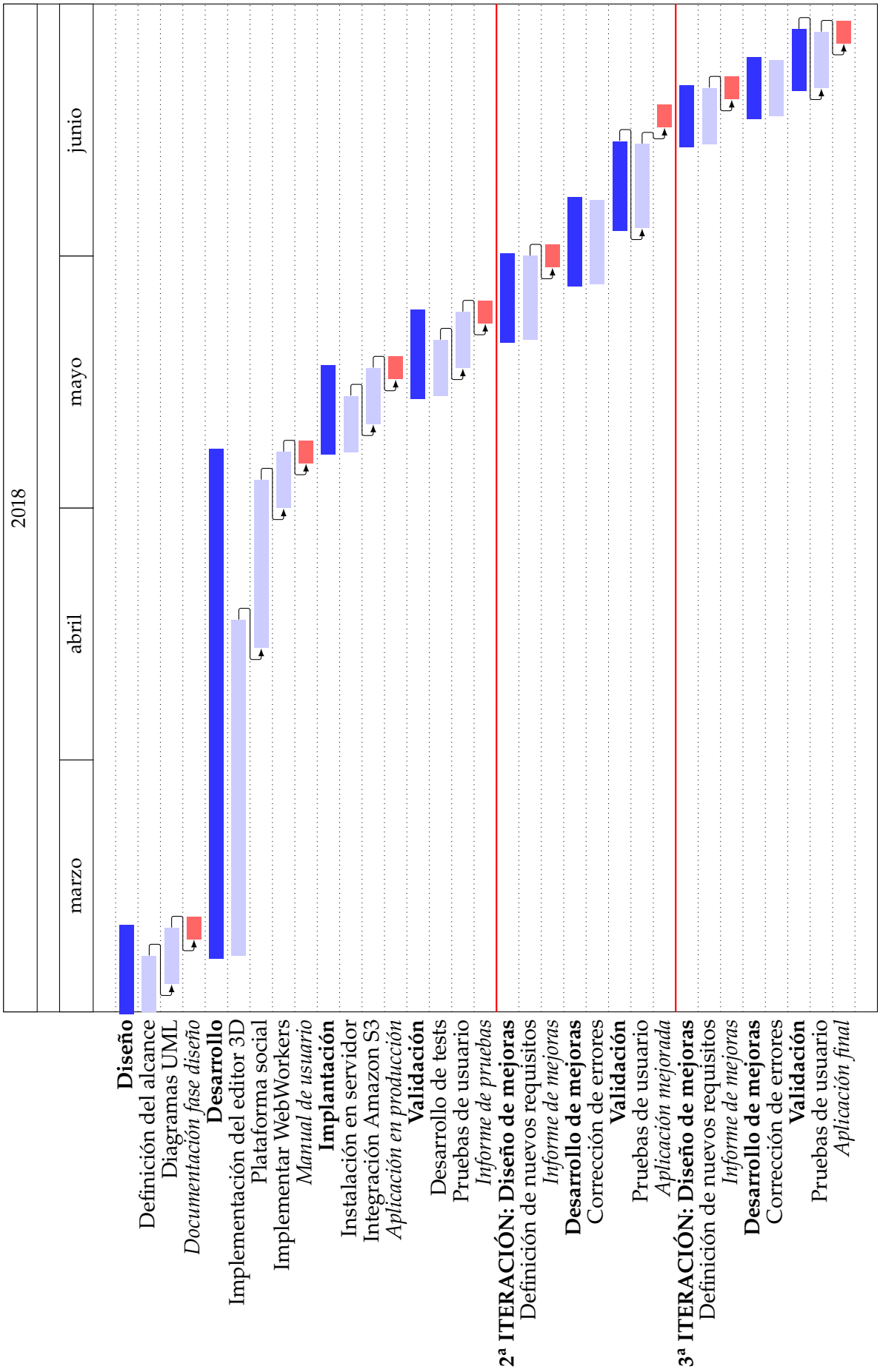


Figura 3.4: Diagrama de Gantt de la planificación del proyecto

CAPÍTULO 4
Diseño de la solución

A la hora de diseñar la herramienta debemos tener en cuenta todos los factores mencionados en el apartado de análisis, ya que estos determinan la arquitectura del sistema en el que se implanta y la propia estructura de la misma.

4.1 Arquitectura del sistema

La aplicación que se ha desarrollado requiere un sistema, preferiblemente basado en UNIX, con los siguientes requisitos:

- Tener un entorno de ejecución para aplicaciones Ruby.
- Permitir compilar programas escritos en C.
- Poseer instalados PostgreSQL, Node.js, bcrypt.
- Permitir el almacenamiento de variables de entorno en el sistema.

En este sistema, los diferentes componentes de la aplicación se estructuran como se puede observar en el siguiente diagrama.

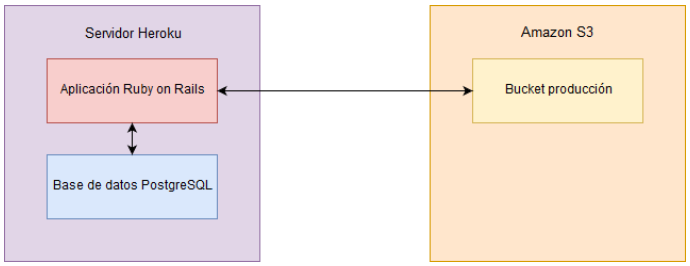


Figura 4.1: Arquitectura del sistema

4.2 Diseño detallado

Aunque la sección anterior puede servir como una base para comprender la estructura de la aplicación, todavía se puede ahondar más en detalle, ya no solo en la distribución física, sino también en los componentes funcionales de la herramienta.

Toda aplicación desarrollada en Ruby on Rails tiene una estructura por defecto, la cual sigue el patrón de diseño **modelo-vista-controlador** (MVC). Este patrón se caracteriza porque para cada recurso existe un modelo que permite representar la información,

un controlador que responde a los eventos del usuario y las vistas que consisten en las páginas que el usuario visualiza en su navegador y que muestran la información.

A su vez, el editor 3D desarrollado con THREE.js, todo su código se encuentra en varios ficheros JavaScript que se incluyen en las vistas correspondientes a las acciones de crear, visualizar y editar un dron.

Por último, el visualizador de realidad aumentada implementado sobre A-Frame también se incluye en una vista específica como en el caso anterior.

4.3 Tecnología empleada

Aplicación Ruby on Rails

Como ya se ha mencionado en la sección anterior, Ruby on Rails cumple con el patrón **modelo-vista-controlador** (MVC). El modelo obtiene la información de la base de datos y permite acceder a ella a través de objetos Ruby por medio del mapeo objeto-relacional (ORM). De este modo se minimizan las consultas a la base de datos, de forma que se desacopla el código de la capa de persistencia y previene agujeros de seguridad que permitan inyección de SQL.

Cada vez que un usuario accede a una URL de la aplicación, el *router* de Rails comprueba a qué controlador y a qué evento o acción corresponde, de modo que delega en este la petición. El controlador se encarga de ejecutar toda la lógica de negocio y es quien llama al modelo para obtener los datos de la capa de persistencia. Una vez finaliza la respuesta al evento del usuario, comunmente muestra su vista correspondiente con la información solicitada o redirige a otra acción para completar la voluntad del usuario.

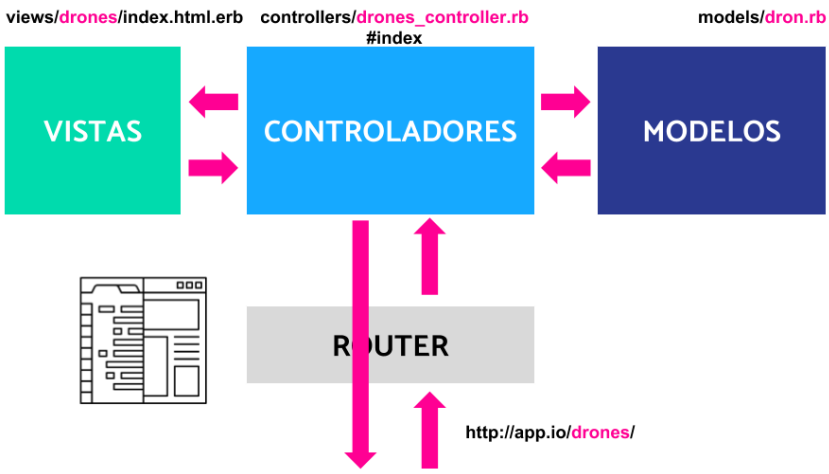
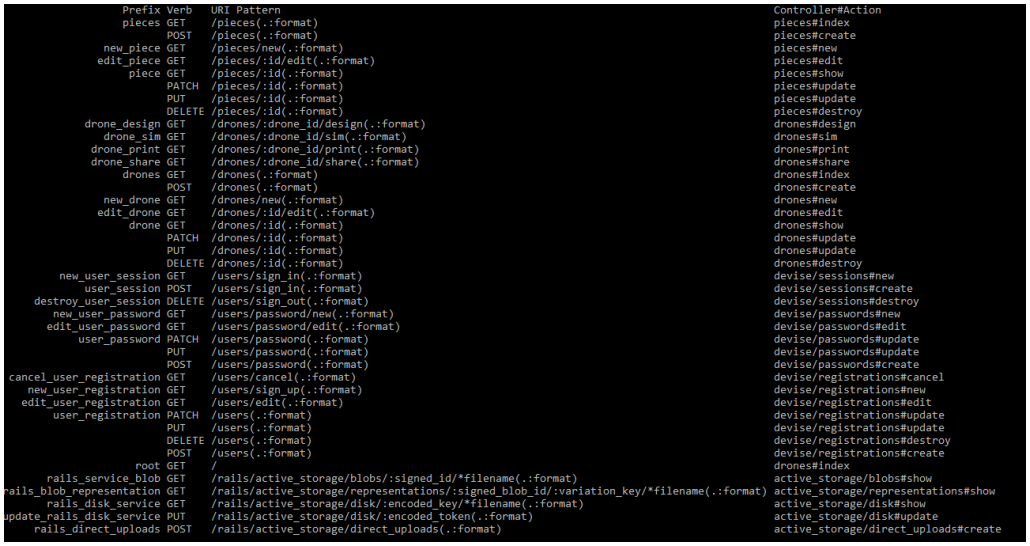


Figura 4.2: Estructura de una aplicación en Ruby on Rails. Diagrama extraído de http://www.infusionvlc.com/meetups/10

.

Todas las rutas disponibles a través de la aplicación web son accesibles a través del router y se pueden visualizar a modo de tabla como se muestra en la siguiente figura:



```
                Prefix Verb   URI Pattern                                                                                       Controller#Action
                pieces GET    /pieces(.:format)                                                                                 pieces#index
                       POST   /pieces(.:format)                                                                                 pieces#create
             new_piece GET    /pieces/new(.:format)                                                                             pieces#new
            edit_piece GET    /pieces/:id/edit(.:format)                                                                        pieces#edit
                 piece GET    /pieces/:id(.:format)                                                                             pieces#show
                       PATCH  /pieces/:id(.:format)                                                                             pieces#update
                       PUT    /pieces/:id(.:format)                                                                             pieces#update
                       DELETE /pieces/:id(.:format)                                                                             pieces#destroy
          drone_design GET    /drones/:drone_id/design(.:format)                                                                drones#design
             drone_sim GET    /drones/:drone_id/sim(.:format)                                                                   drones#sim
           drone_print GET    /drones/:drone_id/print(.:format)                                                                 drones#print
           drone_share GET    /drones/:drone_id/share(.:format)                                                                 drones#share
                drones GET    /drones(.:format)                                                                                 drones#index
                       POST   /drones(.:format)                                                                                 drones#create
            new_drone GET    /drones/new(.:format)                                                                             drones#new
           edit_drone GET    /drones/:id/edit(.:format)                                                                        drones#edit
                drone GET    /drones/:id(.:format)                                                                             drones#show
                       PATCH  /drones/:id(.:format)                                                                             drones#update
                       PUT    /drones/:id(.:format)                                                                             drones#update
                       DELETE /drones/:id(.:format)                                                                             drones#destroy
      new_user_session GET    /users/sign_in(.:format)                                                                          devise/sessions#new
          user_session POST   /users/sign_in(.:format)                                                                          devise/sessions#create
  destroy_user_session DELETE /users/sign_out(.:format)                                                                         devise/sessions#destroy
     new_user_password GET    /users/password/new(.:format)                                                                     devise/passwords#new
    edit_user_password GET    /users/password/edit(.:format)                                                                    devise/passwords#edit
         user_password PATCH  /users/password(.:format)                                                                         devise/passwords#update
                       PUT    /users/password(.:format)                                                                         devise/passwords#update
                       POST   /users/password(.:format)                                                                         devise/passwords#create
cancel_user_registration GET    /users/cancel(.:format)                                                                           devise/registrations#cancel
   new_user_registration GET    /users/sign_up(.:format)                                                                          devise/registrations#new
  edit_user_registration GET    /users/edit(.:format)                                                                             devise/registrations#edit
      user_registration PATCH  /users(.:format)                                                                                 devise/registrations#update
                       PUT    /users(.:format)                                                                                  devise/registrations#update
                       DELETE /users(.:format)                                                                                  devise/registrations#destroy
                       POST   /users(.:format)                                                                                  devise/registrations#create
                    root GET    /                                                                                                 drones#index
     rails_service_blob GET    /rails/active_storage/blobs/:signed_id/*filename(.:format)                                         active_storage/blobs#show
rails_blob_representation GET    /rails/active_storage/representations/:signed_blob_id/:variation_key/*filename(.:format) active_storage/representations#show
       rails_disk_service GET    /rails/active_storage/disk/:encoded_key/*filename(.:format)                                       active_storage/disk#show
update_rails_disk_service PUT    /rails/active_storage/disk/:encoded_token(.:format)                                               active_storage/disk#update
     rails_direct_uploads POST   /rails/active_storage/direct_uploads(.:format)                                                    active_storage/direct_uploads#create
```

Figura 4.3: Tabla de rutas de la aplicación

En la primera columna aparece el nombre con el que Ruby on Rails se dirige a cada ruta. A continuación se puede observar el tipo de petición HTTP y al patrón de la URL al que corresponde. Por último, aparece el controlador y la acción o método que gestionan esa petición. Así, por ejemplo, `new_piece_path` es la ruta que corresponde a la URL `/pieces/new` y es respondida por el controlador `pieces_controller`, concretamente en la acción `new`.

A continuación se encuentra el diagrama de clases elaborado para la aplicación en cuestión. En él se pueden ver todos los atributos que tienen, que se corresponden a su vez con los campos de la tabla en la base de datos. La tabla `User` además almacena el *hash* correspondiente a la contraseña del usuario por motivos de seguridad. Las claves ajenas de cada tabla se han obviado en el diagrama para facilitar la lectura.

Como podemos observar, un usuario puede crear piezas, que a su vez puede incluir en sus propios diseños. Además, se ha añadido una tabla para almacenar distintos tipos de licencias para que los usuarios puedan limitar el uso de sus diseños si fuera necesario.

Editor 3D con THREE.js

THREE.js es una librería escrita en JavaScript que sirve de capa de abstracción sobre WebGL. En el transcurso del desarrollo de este proyecto se ha empleado para implementar varios *scripts* para la carga asíncrona de ficheros de modelos 3D y su visualización. En el *script* principal se describe la escena, que es una representación jerárquica de todos los elementos que se desean visualizar y sus atributos o propiedades.

En esta escena, se colocan los siguientes elementos:

- **Cámara**: sitúa el punto de vista en la escena. Se define por un un punto en el espacio y un vector *up* que define la orientación de la misma, de modo que indica hacia dónde apunta y si está inclinada.
- ***Raycaster***: permite la interacción con los objetos de la escena a través de clics de ratón. Cuando el usuario realiza un clic, se traza la trayectoria que seguiría un rayo de luz desde el origen de la cámara en la dirección del vector desde este origen hacia el punto en el que el usuario hace clic situado en un plano de proyección y se selecciona el primer objeto con el que colisiona. En la siguiente figura se describe su funcionamiento.

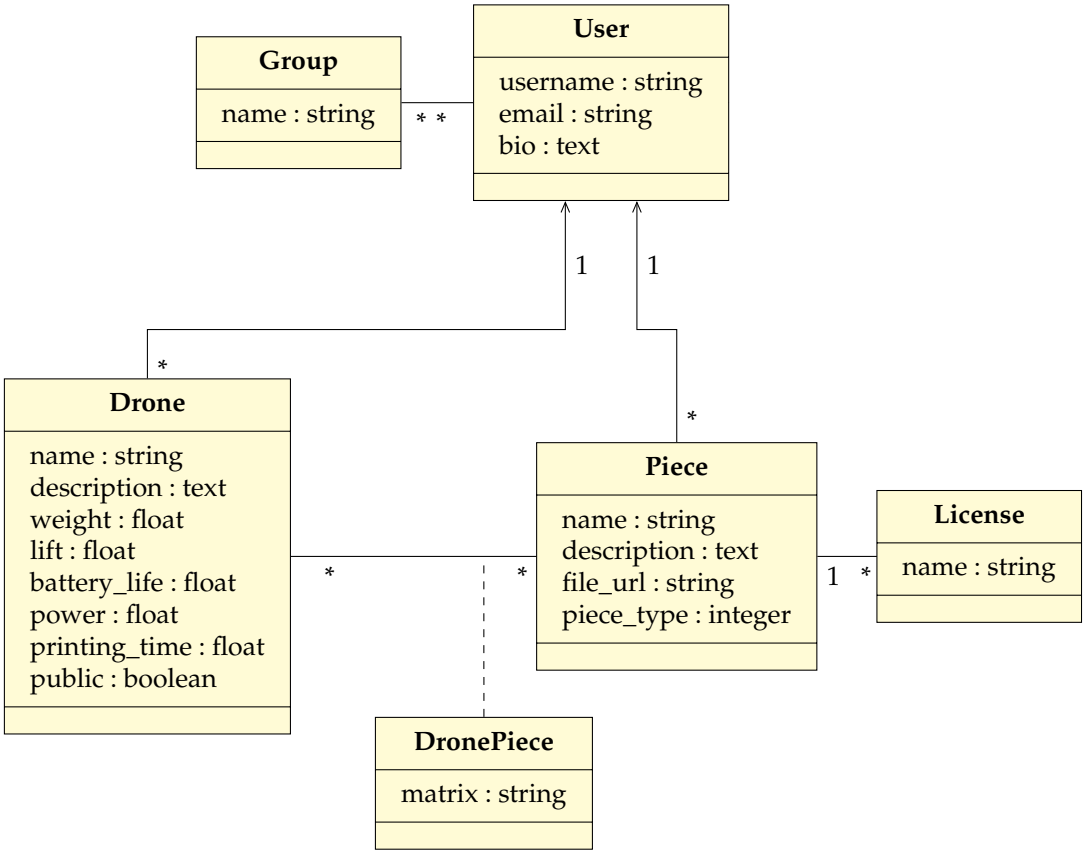


Figura 4.4: Diagrama de clases del proyecto

- **Objetos 3D**: tantos como modelos se quiera visualizar. En nuestro caso, todas las piezas del dron.

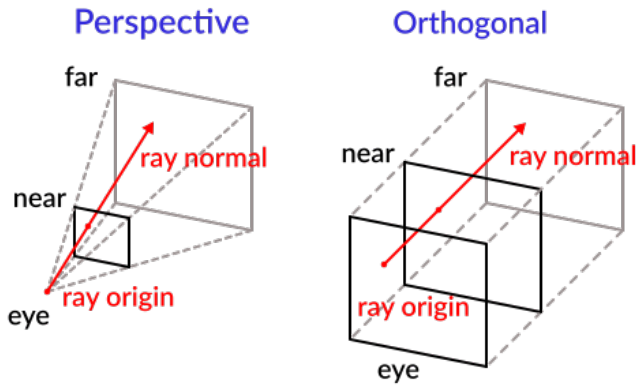


Figura 4.5: Esquema de funcionamiento del proceso de raycasting para la selección de elementos con un ratón. Imagen extraida de la documentación de Godot (CC-BY 3.0)

Cada elemento de la escena puede contener otros dentro de él, del mismo modo que se pueden anidar elementos en el modelo de objetos de un documento HTML (DOM). De este modo, cuando se aplica una transformación a un objeto, afecta a todos sus nodos hijos. Por ejemplo, supóngase que se tienen tres objetos que corresponden con un brazo, su antebrazo y la mano, donde cada objeto es el padre del siguiente. Al rotar, trasladar o escalar el brazo, también se aplican estas transformaciones al antebrazo y a la mano; pero si modificamos la mano estas transformaciones no afectarán a los nodos superiores en la jerarquía.

Transformaciones en el espacio tridimensional

A la hora de representar cualquier objeto en un espacio de tres dimensiones se debe tener en cuenta todas las transformaciones lineales que se han aplicado sobre éste. Las operaciones de rotación, translación y escalado son transformaciones de este tipo.

Toda transformación lineal es una función lineal que transforma vectores de un espacio vecorial V a otro espacio vectorial W cumpliendo siempre estas condiciones:

$$F(u+v) = F(u) + F(v) \;\; \forall u, v \in V$$

$$F(k,v) = k * F(v) \;\; \forall v \in V, \forall k \in \mathbb{R}$$

En un espacio de tres dimensiones, se pueden describir todas las operaciones de translación, escalado y rotación en una matriz de cuatro por cuatro dimensiones.

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ t_x & t_y & t_z & 1 \end{bmatrix} \qquad S = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Matriz de translación Matriz de escalado Matriz de rotación en eje x

En el caso de la matriz de translación, t_x, t_y y t_z son las distancias en las que se desplaza el objeto en cada uno de los tres ejes cartesianos.

Del mismo modo, S_x, S_y y S_z corresponden a los distintos factores de escalado en los tres ejes cartesianos a la hora de aumentar o reducir el tamaño de un objeto.

Las matrices de rotación son diferentes a las dos anteriores, ya que aplican las operaciones del seno y del coseno sobre el ángulo que se desea rotar en el eje cartesiano indicado. Según el eje sobre el que se actúa, se sitúan de distinta forma.

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R_y = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R_z = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Matriz de rotación en eje x Matriz de rotación en eje y Matriz de rotación en eje z

A la hora de aplicar múltiples transformaciones es posible representarlas como una cadena de productos entre matrices:

$$T * X = T_1 * T_2 * T_3 * X$$

Como podemos observar, T es la situación final del objeto X y es equivalente a la multiplicación de las sucesivas transformaciones (translaciones, rotaciones y/o escalados).

Por tanto, es posible describir la situación y orientación de cualquier elemento en una escena tridimensional empleando una matriz de 4x4, resultado de la combinación de todas las transformaciones que este objeto ha sufrido hasta el momento.

Estas matrices permitirán el almacenamiento de la situación de todas las piezas de un dron diseñado por un usuario. Para ello, tan solo se conserva una cadena de texto

que representa esta matriz en la base de datos, la cual se aplica a cada objeto cuando se incluye en la escena para su visualización. De este modo es posible almacenar una escena completa de una forma muy ligera, sin necesidad de guardar la posición absoluta de todos los vértices de cada objeto.

Visualizador de realidad aumentada con A-Frame

Si bien A-Frame es un *framework* basado en THREE.js, añade una nueva capa de abstracción del código siguiendo una arquitectura Entidad-Componente, muy común en el desarrollo de videojuegos. En este caso, los objetos de la escena continúan manteniendo la jerarquía ya mencionada, pero además pueden integrar componentes.

Los objetos 3D pasan a ser **entidades**, elementos que pueden contener **componentes**. Estos a su vez son módulos reutilizables de código que permiten definir la apariencia y el comportamiento de las entidades que las contienen. Por último, los **sistemas** permiten gestionar información y servicios a nivel global de toda la aplicación.

En este proyecto se ha empleado este *framework* para el desarrollo del visualizador de realidad aumentada, puesto que existe un módulo de A-Frame para integrar AR.js [1], una librería para implementar experiencias web de realidad aumentada.

En el próximo capítulo se entrará más en detalle en la implementación de cada una de las tres secciones de la aplicación y en algunas de las particularidades de todo el proceso de desarrollo.

[1] AR.js https://github.com/jeromeetienne/AR.js

CAPÍTULO 5
Desarrollo de la solución propuesta

A lo largo de esta sección se repasan las distintas fases llevadas a cabo durante la implementación del proyecto. Aunque no se pretende describir con todo detalle todo el código desarrollado, los aspectos más relevantes del mismo se encuentran descritos con pseudocódigo para la comprensión del lector.

A continuación se encuentra un esquema de la estructura de la aplicación:

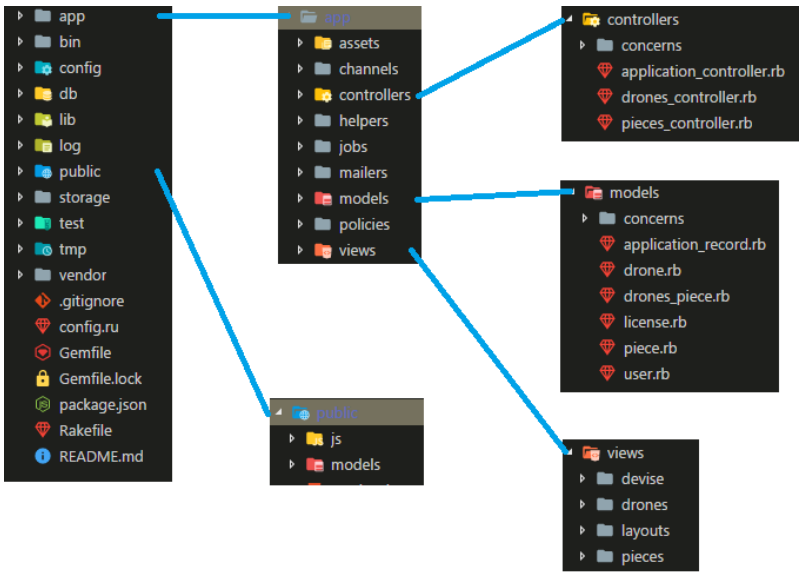


Figura 5.1: Estructura del proyecto

Como podemos ver, Ruby on Rails crea una estructura de aplicación compleja preparada para funcionalidades como la implementación de contenido en tiempo real con WebSockets o el envío masivo de emails con *mailers*.

Toda la lógica de la aplicación se encuentra en las carpetas `app/` y `public/`. En la primera, se hallan los modelos, las vistas y los controladores relativos a todas las clases de objetos del proyecto.

En la carpeta `public/js/` se encuentran los siguientes ficheros de *script*:

- `design.js`: código del editor 3D de drones.
- `show.js`: código del visor 3D de piezas.
- `showDrone.js`: código del visor 3D de drones.
- `simDrone.js`: código del visualizador en realidad aumentada.

Adicionalmente, en la carpeta `public/js/vendor/` se incluyen librerías de terceros que permiten la carga de modelos en formato STL y la interacción del usuario por medio del ratón.

Más adelante se entrará más en detalle en la implementación de cada una de las funcionalidades de la aplicación y se describirá la finalidad de cada uno de los módulos de la aplicación.

5.1 Modelado paramétrico

En primer lugar, previamente a desarrollar el visor 3D o la plataforma social, se diseñaron los distintos modelos tridimensionales de los módulos de los drones. Para ello se ha escogido OpenSCAD, una herramienta de modelado paramétrico de objetos sólidos que permite diseñar figuras en tres dimensiones por medio de operaciones matemáticas como la unión, la intersección o la diferencia entre figuras geométricas.

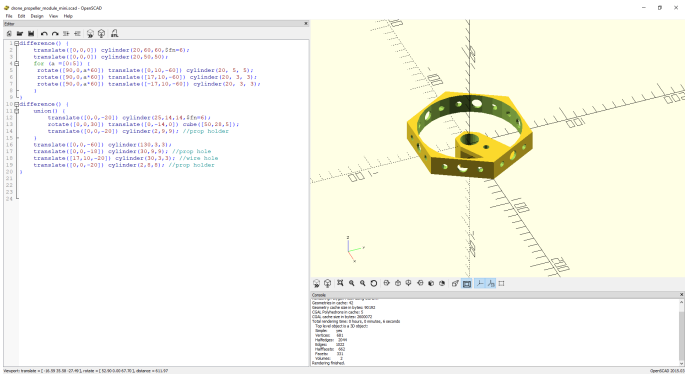


Figura 5.2: Ventana de OpenSCAD con uno de los modelos tridimensionales de los drones.

En la figura se puede observar que el modelo de la derecha se compone de un cubo y diversos cilindros sobre los cuales se han aplicado diversas transformaciones de modo que conforman la geometría descrita en el visor.

En total, existen diseñadas tres piezas por defecto en la aplicación:

- Una pieza central donde se situa la electrónica del dron.
- Soportes para propulsores.
- Módulo adicional.

Adicionalmente, cualquier usuario es capaz de añadir sus propios módulos a la plataforma y acceder a ellos a través del editor web para personalizar su propio dron.

5.2 Editor 3D

Para la implementación de este editor web se han desarrollado diversos *scripts* que se ejecutan en el dispositivo cliente. El primero, `design.js`, contiene toda la lógica de la escena 3D, así como de las respuestas a los eventos del usuario. Las funcionalidades más importantes dentro de este script son las que permiten al usuario:

- Añadir y eliminar piezas.

- Validar la estructura del dron.
- Guardar el estado del diseño.

En el fichero `parseModel.js` se encuentra el código de los WebWorkers, que permiten la carga asíncrona de todos los modelos tridimensionales de la escena, de modo que el tiempo de carga se reduce drásticamente al visualizar una escena.

Carga asíncrona de modelos

Todo el código encargado de cargar los modelos tridimensionales desde el servidor de Amazon S3 se ha encapsulado en un WebWorker. Este es el encargado de acceder a la URL del fichero y cargar el modelo en la memoria del dispositivo cliente.

Desde el *script* principal se crea una instancia de este WebWorker por cada pieza del diseño, creando así un *pool* o conjunto de trabajadores que cargan todos los ficheros de forma asíncrona. Cada vez que uno de ellos finaliza la carga del fichero, crea un `Object3D` a partir de la geometría del modelo.

Esta geometría se almacena como una instancia de `BufferGeometry`, que resulta una representación de la malla tridimensional más eficiente que las geometrías tradicionales, ya que almacena todos los atributos en *buffers* en lugar de la GPU.

Posteriormente, el WebWorkers envía ese *buffer* que contiene toda la información de la malla tridimensional al hilo principal de la aplicación y se coloca en la escena.

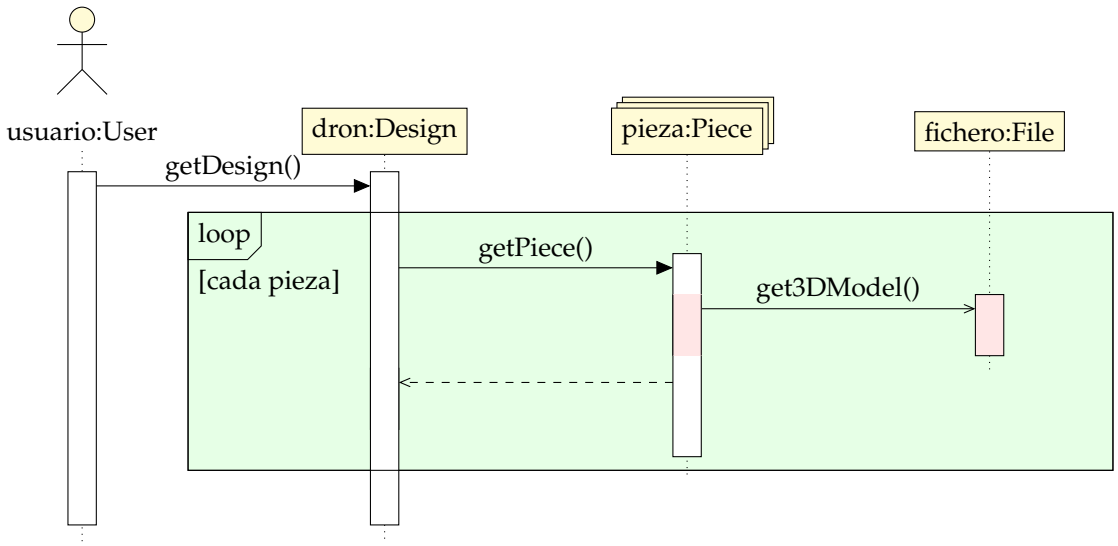


Figura 5.3: Diagrama de secuencia de la carga asíncrona de modelos 3D con WebWorkers

Adición y eliminación de piezas

En primer lugar se muestra en pantalla todas las posibles posiciones en las que se puede añadir el módulo. Para ello se le aplica un material con cierto grado de transparencia a los modelos de las piezas para que el usuario pueda identificar que no están situadas todavía. Una vez el usuario selecciona una, se le aplica el material por defecto del resto de piezas y se añade al diseño del dron.

Todas las piezas, al igual que el resto de elementos de la escena 3D, se disponen en una jerarquía. El módulo central del dron constituye el nodo principal del dron y todos

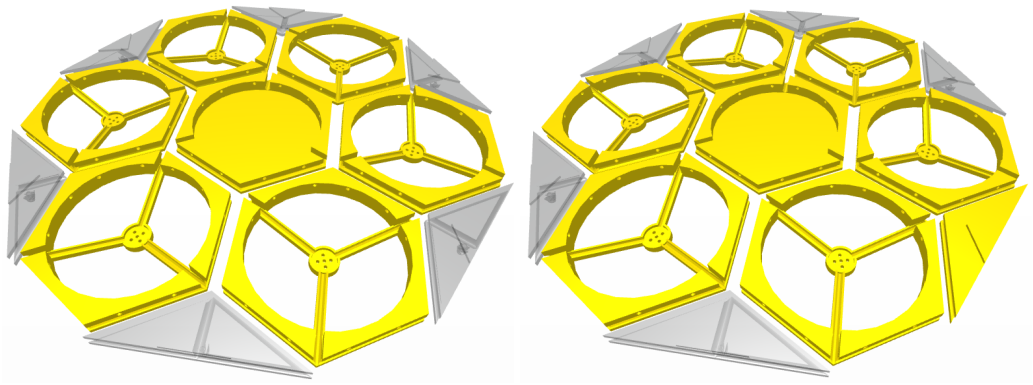


Figura 5.4: A la izquierda: visualización del modelo 3D del módulo a añadir con un material translúcido. A la derecha: aplicación del material por defecto a la pieza añadida.

los demás componentes que el usuario escoge se añaden a este. De este modo, todas las transformaciones que se apliquen a la pieza central afectarán del mismo modo al resto de componentes del dron.

```
// Supongamos un nodo para el dron y otro para el módulo
var drone = new Object3D();
var piece = new Object3D();

// Añadimos una pieza como nodo hijo
drone.add(piece);
```

Para eliminar una pieza, simplemente se debe ejecutar `drone.remove(piece)`. Sin embargo, esta acción no es posible siempre. En ocasiones, eliminar una pieza del dron puede comprometer su integridad estructural y dividirlo en dos o más partes, por lo que la aplicación previene este tipo de comportamiento.

Para ello, cuando el usuario selecciona una pieza para eliminarla, se comprueba la cantidad de piezas vecinas a esta y a las piezas colindantes. De este modo, en caso de que alguna pieza solo tenga por vecina a la pieza seleccionada, se muestra un mensaje de error al usuario.

A continuación se encuentra una descripción en pseudocódigo de esta comprobación:

```
if pieza_seleccionada then

  for all conectada in vecinas(de pieza_seleccionada) do
    conexiones = 0
    for all pieza in vecinas(conectada) do
      conexiones++
      if conexiones <2 then
        mostrarError()
      else
        eliminarPieza(pieza_seleccionada)
      end if
    end for
  end for
end if
```

Validación de la estructura del dron

Una vez el usuario ha añadido los propulsores y los módulos adicionales que desea, puede comprobar el equilibrio de su dron y calcular ciertos parámetros como la vida media de la batería.

Para el cálculo de la distribución de masa de los módulos se multiplica el peso tabulado de cada tipo de pieza empleado por la distribución espacial en los ejes X y Z de todas las instancias de la misma. De un modo simplificado, se puede entender como la suma de las masas multiplicadas por su desplazamiento desde el centro del módulo respecto al centro de masa:

$$CDM_x = \sum_{i=1}^{n} -m_i * x_i; CDM_y = \sum_{i=1}^{n} -m_i * y_i$$

Una vez efectuado el cálculo, se muestra por pantalla el vector desde el origen del dron (el punto central) hacia el lado al que se inclina, es decir, aquella dirección en la que la masa está más concentrada.

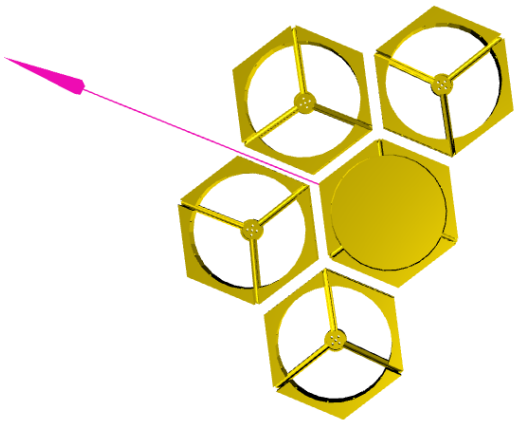


Figura 5.5: Visualización de la dirección y el sentido en el que el dron se desequilibra

Adicionalmente, también se calculan distintos parámetros como el tiempo de impresión medio de las piezas 3D o la vida estimada de una batería de 11.1 V 5500 mah a pleno rendimiento a partir de la masa del mismo.

Exportar diseños

Todos los diseños de los usuarios se conservan en el servidor de Amazon S3, los cuales se pueden descargar en cualquier momento. Para ello, cualquier usuario puede pulsar el botón *Print* en la página de visualización de un dron. En ese momento, el servidor confecciona un fichero *zip* con los modelos tridimensionales de las piezas del dron y un fichero de texto con las instrucciones para imprimirlo.

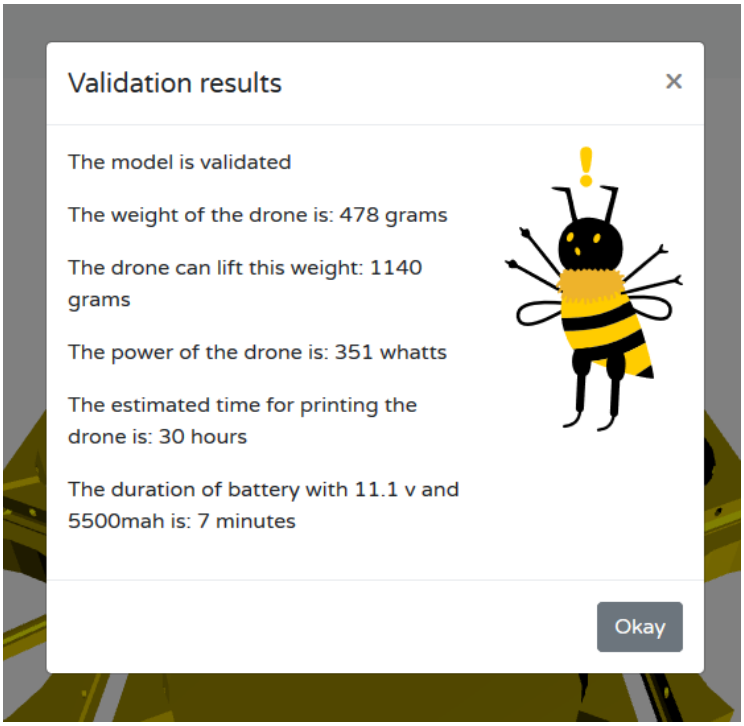


Figura 5.6: Parámetros calculados al validar un dron

5.3 Plataforma social

La aplicación web que engloba el editor 3D y el visualizador de realidad aumentada es una plataforma social. En ella, cualquier usuario puede realizar todas las acciones descritas en el diagrama de casos de uso de la sección análisis del problema.

Todas las funcionalidades de gestión (adición, edición y eliminación) de los diseños del usuario se realizan a través del editor 3D descrito en la sección anterior. Sin embargo, para la creación de módulos personalizados y la visualización del repositorio de piezas de otros usuarios es necesario implementar una herramienta.

Restricciones en los modelos

Algunos de las propiedades de los objetos de la base de datos están sujetos a ciertas restriciones. Este es el caso, por ejemplo, del nombre y la descripción de los drones y las piezas, o el nombre de perfil de un usuario, puesto que todos ellos son obligatorios para la creación de los objetos.

Por esto, cada modelo implementado en la aplicación incluye validaciones que restringen el uso del modelo. De este modo, si un usuario intenta guardar un dron o una pieza sin nombre, es redirigido por la aplicación de nuevo al formulario y se le muestra un mensaje de error. De no ser así, la aplicación incurriría en errores muy graves mostrando campos vacíos o erróneos.

Gestión de usuarios y permisos

Para la implementación de todas las funcionalidades básicas referentes a los perfiles de usuario (inicio de sesión, registro de nuevos usuarios, etc.) se ha integrado la librería Devise, como ya se adelantaba en la sección análisis de la seguridad.

Los usuarios básicos pueden realizar todas las funcionalidades especificadas en el diagrama de casos de uso anteriormente mencionado. Sin embargo, es muy importante impedir que cualquier usuario pueda modificar las piezas o drones de otro.

Para la gestión de estos permisos se ha integrado una librería denominada Pundit [1], que permite diseñar políticas de permisos del mismo modo que se implementa cualquier guarda para un bucle, por ejemplo. Cada controlador tiene un archivo de políticas asociado, en el que hay una política por cada acción del mismo.

Cuando el usuario hace una petición web a la aplicación, Pundit comprueba si existe una política asociada y en caso afirmativo asegura que se cumple. Si es así, se ejecuta la acción; en caso contrario, se muestra un mensaje de error al usuario y se le redirige a la página en la que se encontraba anteriormente.

Figura 5.7: Controladores y sus políticas asociadas

Creación de módulos personalizados

A la hora de crear nuevos módulos, el usuario rellena un sencillo formulario con la información básica del mismo y adjunta un fichero STL del modelo 3D. Este fichero se almacena en el servidor de Amazon S3 y se accede al mismo cada vez que se desea visualizar el diseño del usuario.

Para acceder al servidor de Amazon S3 se requiere aportar una clave pública y una clave privada para poder añadir ficheros, de modo que solo es posible hacerlo desde la aplicación, ya que ambas claves se almacenan como variables del sistema en el servidor. Estas claves se almacenan como variables de entorno en el sistema en el que está instalada la aplicación para prevenir posibles filtraciones.

5.4 Visor de realidad aumentada

Una vez finalizado su diseño, cualquier usuario tiene la posibilidad de imprimir un código QR [2] para compartirlo con sus amigos. Para ello, el sistema genera un fichero PNG con el QR para ser impreso. Este código QR codifica simplemente la URL correspondiente al modelo del usuario a modo de matriz de píxeles negros y blancos.

Cualquier usuario con un dispositivo móvil puede escanear el QR de modo que se abra la página del visor de realidad aumentada. El usuario observa en su pantalla la misma imagen que capta la cámara de su dispositivo y si enfoca el código QR de nuevo, puede visualizar el modelo 3D del dron correspondiente en realidad aumentada a escala real.

[1] Pundit https://github.com/varvet/pundit
[2] Código QR https://es.wikipedia.org/wiki/C%C3%B3digo_QR

Para ello se ha creado una página con A-Frame que carga el dron creado por el usuario en una escena 3D aplicando las matrices de transformación a cada componente. Esta escena permanece oculta al observador hasta que se detecta el código QR en la imagen de la cámara. En ese momento, el origen de coordenadas de la escena se coloca en el centro del código QR con la misma disposición, escala y orientación y se superpone al fondo, de modo que se crea la ilusión de que realmente el dron está presente en la escena.

Internamente, lo que ocurre es una transformación en tres dimensiones entre la posición original de la escena relativa al observador (inicialmente oculta), y la posición del código QR respecto al foco de la cámara del dispositivo. Esta transformación se expresa como una matriz de 4x4 que contiene la información relativa al desplazamiento, el escalado y la rotación de la misma. Para aplicar esta matriz simplemente se multiplica a la matriz que expresa el estado actual de la geometría del mismo modo que se describió en el apartado sobre transformaciones 3D.

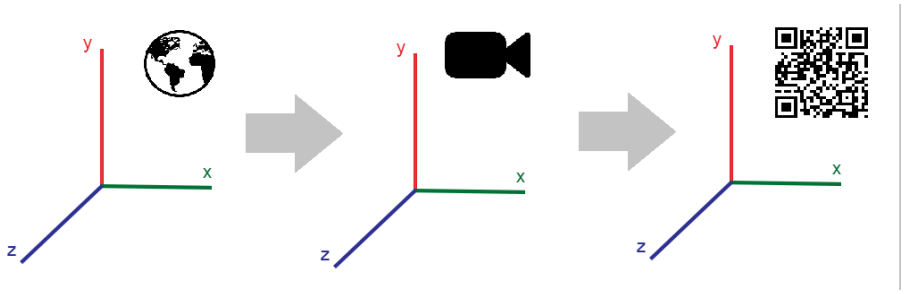


Figura 5.8: Transformación para la visualización en realidad aumentada

CAPÍTULO 6
Implantación

La aplicación web desarrollada se ha instalado en un sistema que cumple con los requisitos mencionados en la sección de arquitectura del sistema.

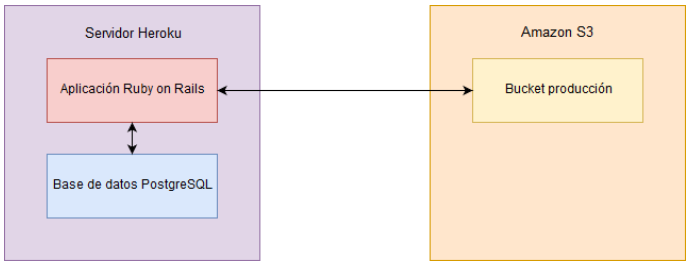


Figura ??: Arquitectura del sistema (repeated from page **??**)

Todo el código de la aplicación reside en un servidor de Heroku [1], el cual también contiene una base de datos PostgreSQL. Este servidor permite la integración continua con GitHub [2] y otros servicios, de modo que cada vez que se confirma un *commit* o se añaden cambios a la rama principal del proyecto, el código se actualiza en el servidor de forma automática sin afectar al contenido de la aplicación siempre y cuando se hayan superado todos los tests de validación.

Por otro lado, los ficheros más pesados como son los modelos tridimensionales se almacenan en Amazon S3, ya que el coste de almacenamiento es mucho menor en este servicio y asegura una mejor persistencia. Para su acceso, como ya se ha comentado en el capítulo anterior, es necesario aportar la clave pública y privada del mismo.

Todas las claves para servicios e interfaces de terceros se han almacenado en variables del sistema del servidor por motivos de seguridad, ya que al plasmarlas en el código explícitamente es muy probable que cualquier usuario con intención maliciosa robara estas credenciales.

Adicionalmente, para la puesta en marcha inicial del sistema se ha desarrollado un pequeño *script* en `db/seeds.rb`, que crea en la base de datos todas los tipos de licencias disponibles en la aplicación de forma automática.

Producto final

El desarrollo de esta herramienta forma parte de Beewo, un proyecto educativo de emprendimiento galardonado con el premio 2K16 por el Instituto IDEAS UPV en la cate-

[1] Heroku `http://heroku.com/`
[2] GitHub `https://github.com`

goría de idea avanzada. En concreto, la herramienta es accesible en http://beewomaker.herokuapp.com.

A continuación se describe el funcionamiento de la aplicación junto con la interfaz final de la misma. En primer lugar, la pantalla inicial del portal muestra los drones más recientes creados por toda la comunidad y que han sido marcados como públicos por sus creadores. Además, si el usuario ha iniciado sesión en su cuenta, también ve los suyos propios (tanto los públicos como los privados).

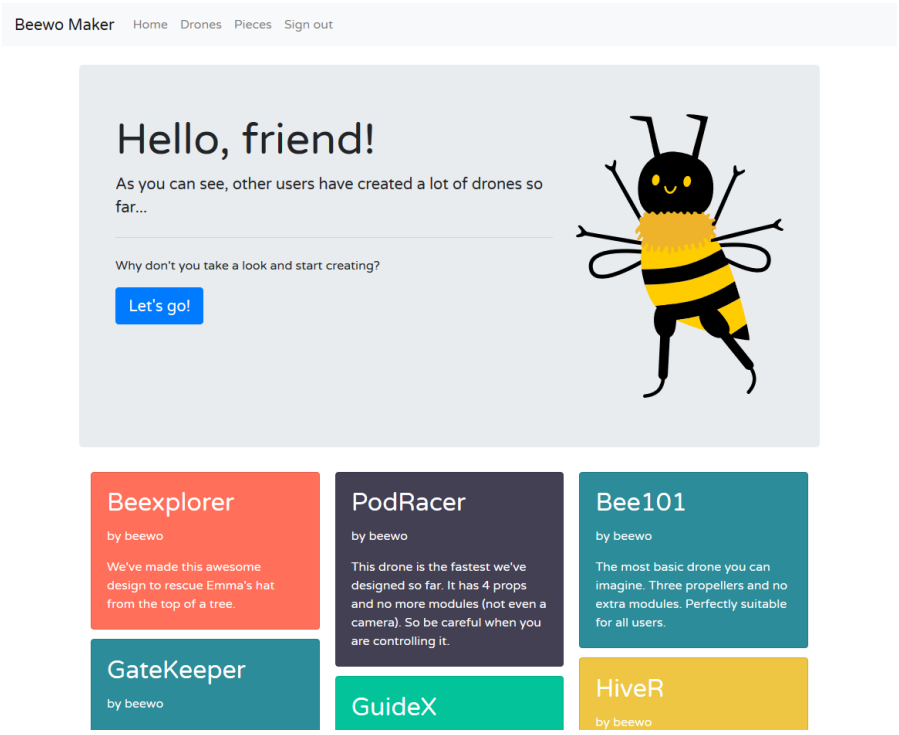


Figura 6.2: Página principal de la aplicación

Si el usuario selecciona uno de los diseños de la lista la aplicación le redirige al visualizador, donde puede ver a pantalla completa el diseño del dron y una pequeña ficha con el nombre del dron, el autor y su descripción. Adicionalmente, puede realizar una de las siguientes cuatro acciones:

- Al pulsar en el botón *Details* se despliega una tarjeta con los parámetros del dron (peso, carga máxima, etc.).
- Si el usuario activo es el autor del dron, puede pulsar en *Redesign* para editarlo de nuevo.
- El botón de *Print* permite al usuario descargar un fichero zip con los modelos de las piezas del dron y unas instrucciones para su impresión.
- El botón *Share* inicia la descarga de un fichero de imagen con el código QR del dron. Este QR, al ser escaneado por un dispositivo, redirige al visualizador de realidad aumentada para ese dron concreto (el cual va específicado como parámetro en la URL).

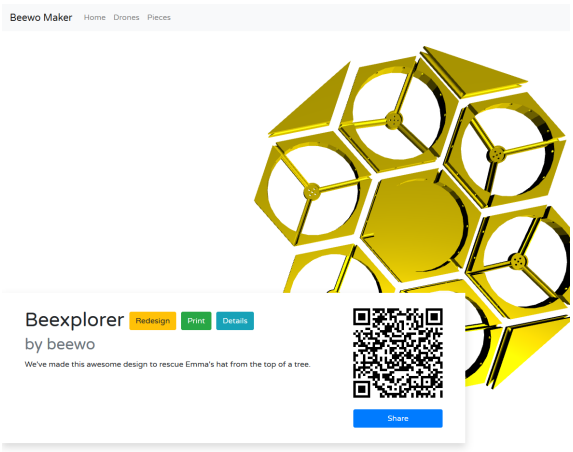


Figura 6.3: Interfaz del visualizador de drones

También existe una galería de piezas que muestra todos los modelos 3D de la comunidad que están disponibles públicamente. Los módulos personalizados se muestran en tarjetas azules para indicar que se pueden emplear en el editor.

Cuando un usuario crea una pieza, esta automáticamente pasa a formar parte del repositorio de la comunidad (siempre y cuando la haya marcado como pública). Desde ese momento, cualquier miembro de la comunidad puede emplear esa pieza en la creación de sus diseños en el editor.

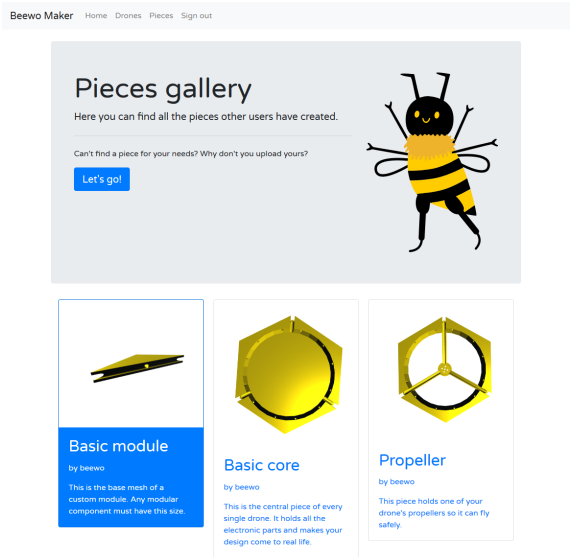


Figura 6.4: Interfaz de la galería de piezas

A la hora de crear un dron, el usuario debe rellenar en primer lugar la información básica de este en un formulario asistido por Abee, la mascota de Beewo.

Figura 6.5: Formulario para nuevos drones

Una vez el usuario confirma estos datos, la aplicación da paso al editor. Esta parte de la aplicación divide el proceso de diseño del dron en tres pasos:

1. Selección del número de propulsores.

2. Adición de módulos de la galería.

3. Validación y almacenamiento del diseño.

La aplicación permite seleccionar una de las tres disposiciones base para los propulsores del dron. De este modo, se mantiene un compromiso con la simetría del mismo y se simplifica mucho el proceso de creación gracias a la geometría hexagonal de sus piezas.

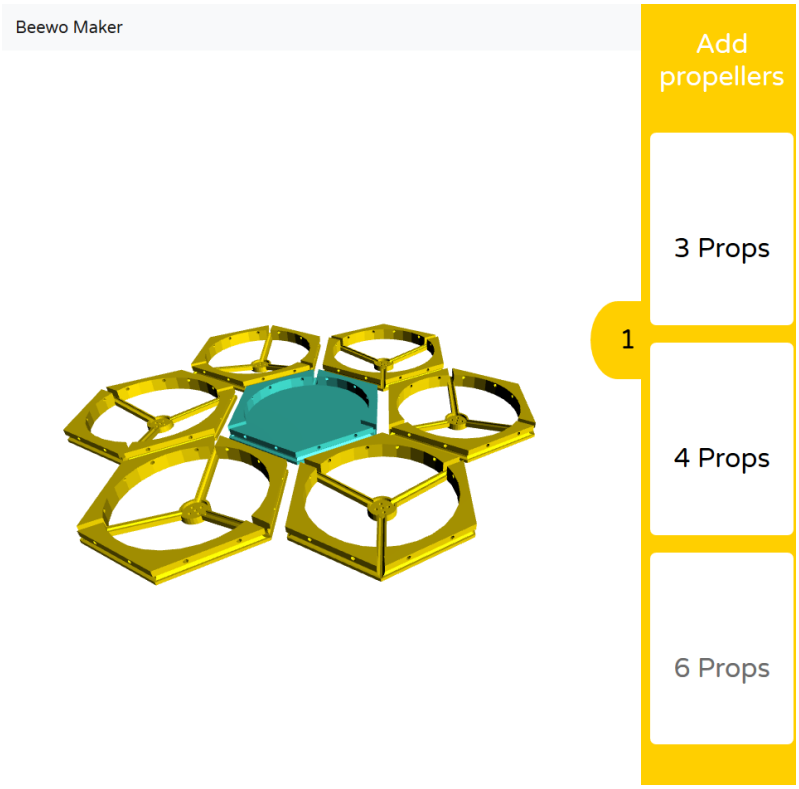


Figura 6.6: Interfaz para añadir propulsores a un dron.

A continuación, el usuario puede añadir módulos adicionales a partir de los disponibles en la barra lateral, los cuales forman parte de la galería de piezas. Para ello, la aplicación muestra las posiciones en las que puede añadir cada pieza y el usuario puede escogerla con un simple clic.

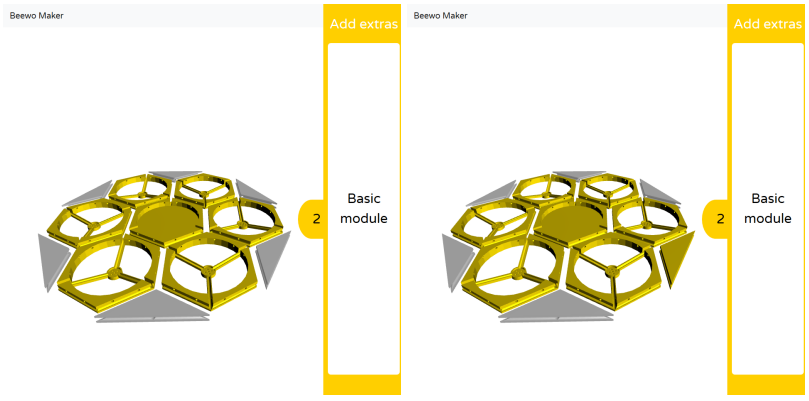


Figura 6.7: Interfaz para añadir módulos adicionales a un dron.

Por último, una vez el usuario considera que su diseño está acabado puede validarlo y guardarlo. En caso de que no quisiera validarlo, la validación ocurre en segundo plano cuando se guarda el diseño para poder efectuar la estimación de los parámetros del mismo.

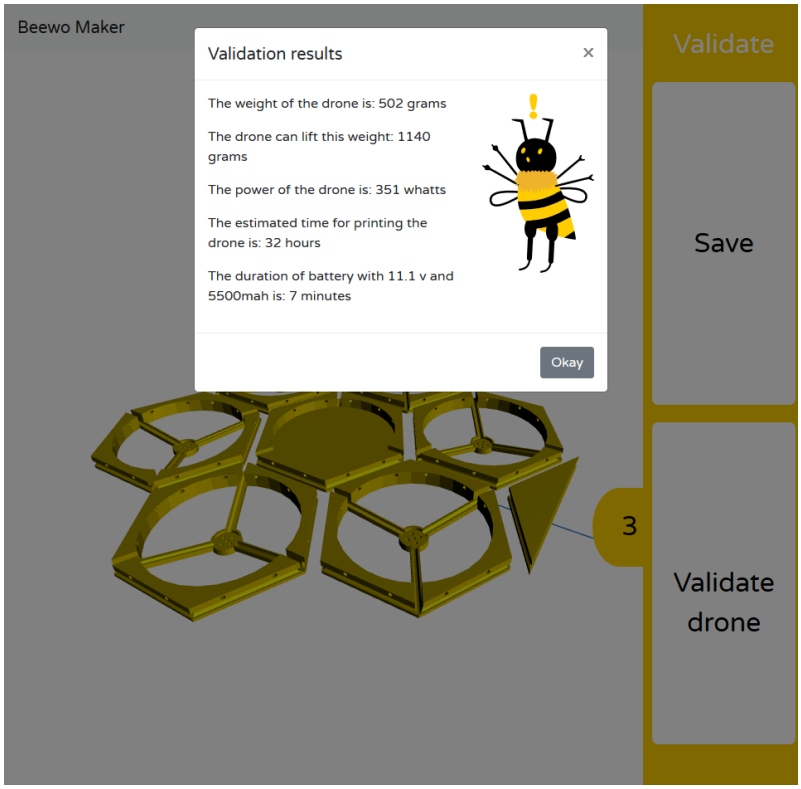


Figura 6.8: Interfaz para la validación de un dron.

En cualquier momento, el usuario puede seleccionar cualquier pieza y eliminarla. Sin embargo, si a la hora de eliminar una pieza el sistema detecta que queda una pieza inconexa muestra un mensaje de error.

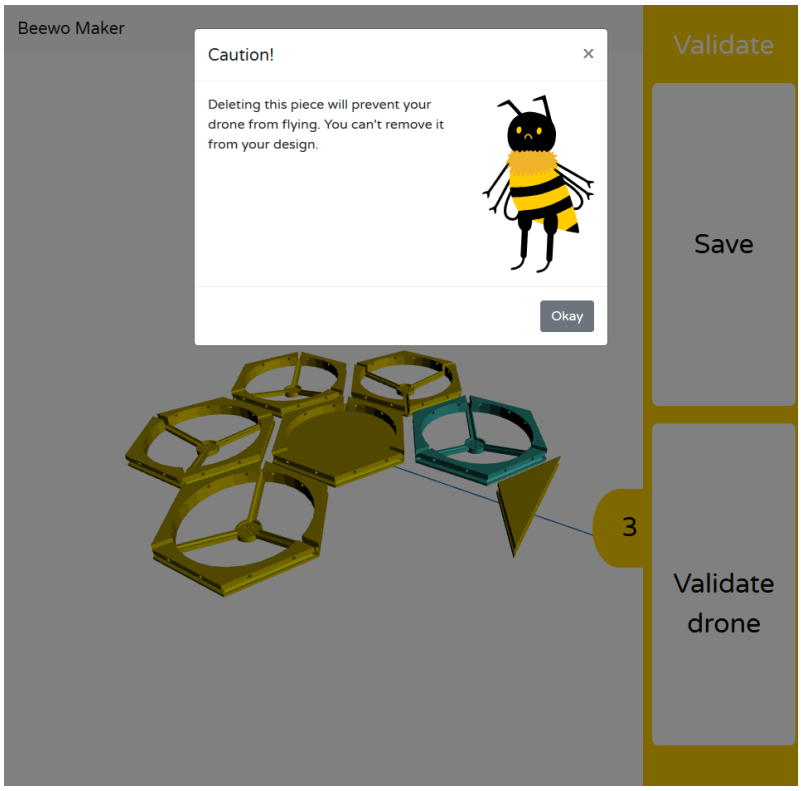


Figura 6.9: Mensaje de error tras intentar eliminar una pieza, dejando así una pieza inconexa.

CAPÍTULO 7
Pruebas

Una vez finalizado todo el desarrollo se han llevado a cabo pruebas de rendimiento de la aplicación con tal de analizar el tiempo de carga y su impacto energético. A continuación se detalla una comparativa del rendimiento de la carga de un modelo por defecto, con seis propulsores y seis módulos, en los navegadores más populares en distintos dispositivos.

Esta comparativa muestra los datos obtenidos a través de stats.js, un monitor de rendimiento para Javascript. En la primera fila se muestran en azul los resultados de fotogramas por segundo (FPS) y en verde, el tiempo de respuesta (en milisegundos) del editor de drones. La segunda fila muestra las mismas métricas, pero en el caso del visor de realidad aumentada.

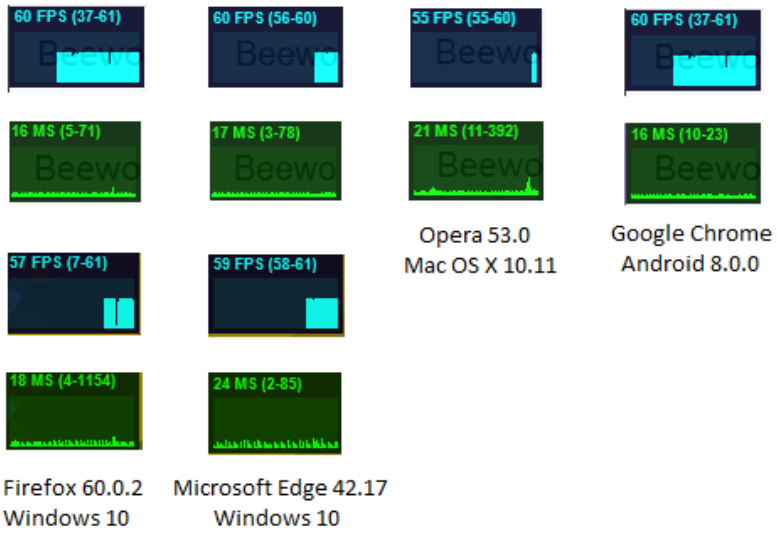


Figura 7.1: Comparativa del rendimiento de la aplicación en diferentes plataformas y navegadores.

Como se puede observar en la figura, todos los navegadores presentan un rendimiento similar a la hora de visualizar un modelo tridimensional. En el caso de Google Chrome y Opera no ha sido posible realizar un análisis del rendimiento del visualizador de realidad aumentada ya que requieren del cifrado de la conexión por SSL, lo cual conlleva un gasto económico porque Heroku no permite el uso de certificados SSL en su plan gratuito.

A partir de estas métricas se puede comprobar que el comportamiento de la aplicación es bastante estable en su totalidad, lo cual asegura su portabilidad y su compatibilidad con la gran diversidad de dispositivos y entornos que existen.

Por último, se han llevado a cabo pruebas con usuarios potenciales de la herramienta (entre 6 y 18 años) con tal de comprobar posibles fallos y mejoras en la usabilidad. En estas pruebas se le pedía al usuario crear un dron con cuatro propulsares y un par de módulos.

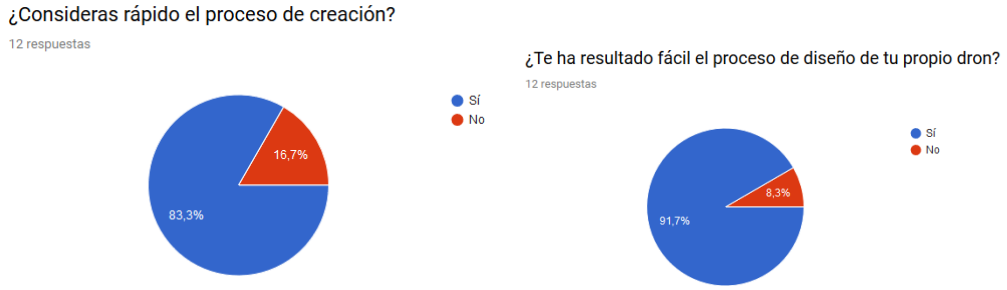


Figura 7.2: Resultados de la encuesta referentes a la agilidad y a la facilidad del proceso de creación

Entre los 12 encuestados, un 83,3 % considera que el proceso de creación es rápido y ágil, mientras que un 91,7 % valora la facilidad con la que es posible realizar dicho proceso.

Adicionalmente, también se preguntó por la velocidad de la herramienta a los encuestados pidiendo una valoración de 1 a 10. Como se puede observar en la figura, todas las respuestas se situaron entre el 8 y el 10. El 75 % de los encuestados considera que la velocidad de la herramienta es perfecta.

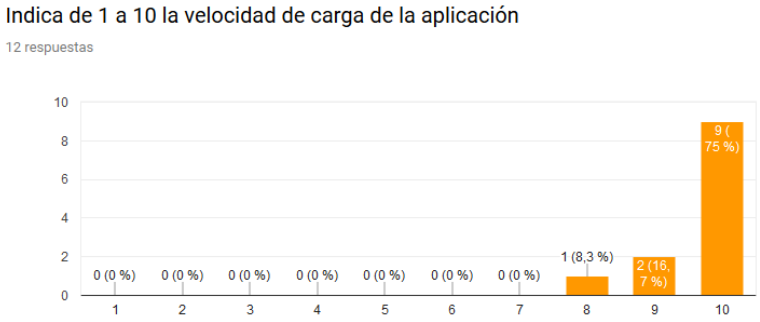


Figura 7.3: Resultados de la encuesta: referente a la velocidad de carga de la herramienta

Tras realizar estas primeras pruebas quedó validada la idea del proyecto de crear un editor sencillo y ágil, pero además se arrojaron las siguientes mejoras y sugerencias por parte de los encuestados:

- **Creación de un visualizador de realidad aumentada.** Esta parte del proyecto no consta en los objetivos de la aplicación, ya que en un inicio no estaba contemplada, pero se consideró su implementación tras la sugerencia de varios usuarios.
- **Rediseño de la interfaz.** Fueron varios los usuarios que hicieron sugerencias referentes a la estética, un punto clave en el proyecto, ya que su público objetivo son los niños y la interfaz debe ser atractiva y sencilla.

- **Integración de una mascota que asista al usuario durante el proceso de creación.** Esta última idea es posiblemente la más relevante desde el punto de vista de la usabilidad, ya que el asistente introduce a los usuarios la herramienta de una forma sencilla y con un lenguaje amigable.

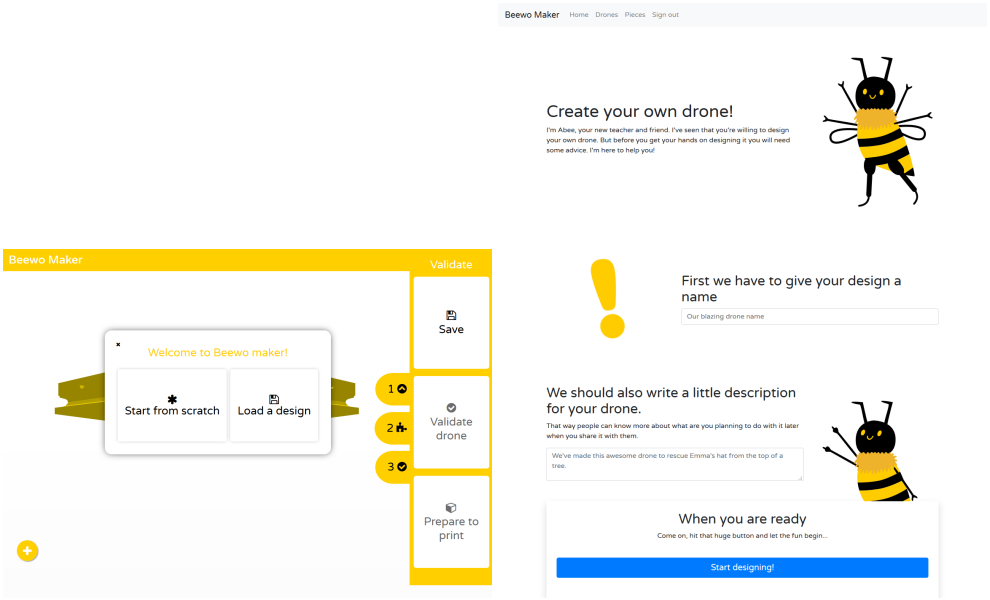


Figura 7.4: Rediseño de la interfaz de la primera pantalla editor

El visualizador de realidad aumentada se ha implementado como un módulo independiente y es accesible a través del código QR que se encuentra en el visor de drones y que se puede descargar como fichero. En él es posible visualizar el modelo diseñado por el usuario sobre un marcador para realidad virtual como se puede observar en la siguiente figura.

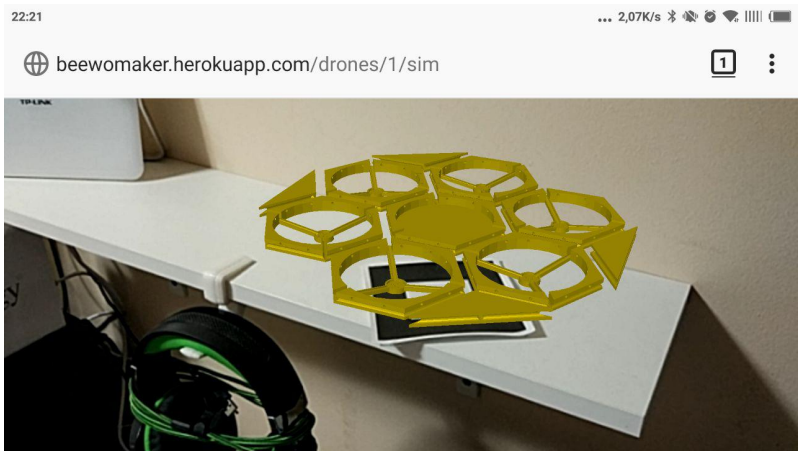


Figura 7.5: Visualizador de realidad aumentada

Una vez se desarrollaron estas nuevas funcionalidades y mejoras se realizó una nueva encuesta para validarlas. En estas nuevas preguntas, tan solo una persona de las 12 encuestadas consideraba que la inclusión de Abee, la mascota del proyecto, no resulta atractiva. Mientras que un 83,3 % de los entrevistados consideraba interesante el visor de realidad aumentada.

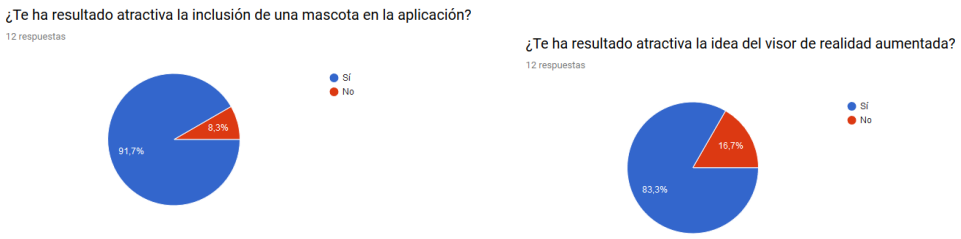


Figura 7.6: Resultados de la encuesta referentes a las ampliaciones del proyecto

Estos resultados, más basados en la subjetividad y la opinión de los usuarios, demuestran que la realidad aumentada es un medio interesante para los jóvenes y que la iconografía y el lenguaje amigable de una mascota hace que la herramienta sea más atractiva y fácil de utilizar.

Por último, en relación con ese visor de realidad aumentada, también se preguntó a los encuestados por la velocidad de carga del mismo. En este caso, una persona de las encuestadas tuvo problemas de carga con la aplicación. Esto puede ser debido a la conexión a internet o a un dispositivo móvil con *hardware* desfasado o poco potente.

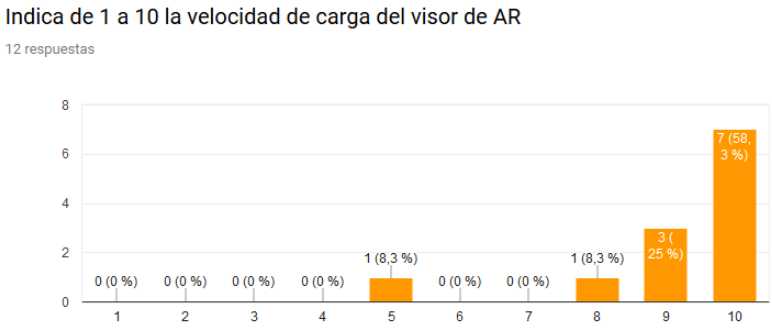


Figura 7.7: Resultados de la encuesta referentes al visualizador de realidad aumentada

Como podemos ver, las tecnologías empleadas han hecho posible el desarrollo de una herramienta ágil con poco consumo de recursos, tanto a nivel del *hardware* del dispositivo cliente como de ancho de banda.

CAPÍTULO 8
Conclusiones

La realización de este proyecto ha conllevado un aprendizaje transversal de conocimientos y tecnologías actuales como THREE.js y A-Frame, relacionados con los gráficos interactivos y la ingeniería del software gracias al cual se han desarrollado con éxito todos los objetivos plasmados al inicio de este documento.

Se ha desarrollado un editor web de drones a partir de módulos, en el que el usuario puede añadir sus propias piezas y crear diseños, que puede descargar en cualquier momento. Este editor se ha integrado en la plataforma social planteada a modo de repositorio global de drones.

Adicionalmente, se ha integrado un visor de realidad aumentada y se ha diseñado una mascota que asiste al usuario durante todo el proceso de creación, con tal de hacer la aplicación más atractiva al usuario, ya que durante el desarrollo de la misma se detectaron fallos y posibles mejoras relativas a la usabilidad.

Cabe destacar que la realización de este proyecto y todo el proceso de aprendizaje asociado han propiciado que el alumno desarrolle proyectos adicionales relacionados con la realidad virtual en la web y la simulación para reforzar los conocimientos en estas tecnologías de vanguardia [1].

8.1 Relación con los estudios cursados

Toda la implementación de la aplicación ha sido posible gracias a los conocimientos obtenidos en asignaturas como Sistemas de Gráficos Interactivos (SGI) o Diseño y Fabricación 3D (3D). Siendo de especial importancia todos los conceptos sobre transformaciones 3D y almacenamiento de ficheros en formato STL, así como el conocimiento del *hardware* dedicado a estas labores y algoritmos de visualización.

Por otro lado, durante el desarrollo de este proyecto también se han visto reforzados conceptos sobre el desarrollo de software y la planificación, introducidos en asignaturas como Ingeniería del Software (ISW) o Gestión de Proyectos (GPR), especialmente para desarrollo de todo el apartado de análisis y diseño de la solución.

Por último, tambien han sido de gran utilidad los conocimientos transversales de análisis y resolución de problemas, diseño y proyecto, creatividad y emprendimiento, aplicación y pensamiento práctico y el aprendizaje permanente, adquiridos durante el Grado y por la participación en otras actividades universitarias como son la pertenencia al ACM UPV Chapter o el desarrollo de un proyecto de emprendimiento en Start.inf.

[1] Proyectos personales https://alvarocasadoc.github.io/3December/

CAPÍTULO 9
Trabajos futuros

Durante las últimas etapas del desarrollo se han detectado posibilidades de mejora dentro del proyecto. Principalmente en las áreas relacionadas con la usabilidad. Las mejoras más importantes, como la necesidad de un rediseño o la adición de un asistente virtual, se han llevado a cabo tras las pruebas con usuarios.

Sin embargo, existen diversas vías de acción para la mejora de la herramienta. Por ejemplo, es posible añadir funcionalidades a la misma para que los educadores puedan crear tareas en la plataforma y que los alumnos puedan subir sus diseños a estas para su corrección.

De este modo, un usuario sería capaz de registrarse como usuario normal o como educador. En el caso de ser un educador, tendría a su alcance las funciones para crear grupos, añadir miembros a un grupo y reportar usuarios.

En cada grupo, el educador puede crear distintas tareas con una fecha límite de entrega. Estas tareas podrían consistir en diseñar un dron para una finalidad en concreto, como la vigilancia de un espacio abierto. En el momento que se añade una tarea a un grupo, el sistema enviaría un correo electrónico a todos sus miembros con su información.

A partir de ese momento, cada usuario miembro del grupo podría subir su solución a esta tarea y esperar a que el educador corrija su creación.

Por otro lado, se puede añadir la capacidad de simulación de un vuelo con el dron diseñado por el usuario a modo de comprobar su funcionamiento de una forma sencilla.

Para ello se crearía un nuevo visualizador 3D con capacidad de procesar las pulsaciones de teclado del usuario o los datos del giroscopio del móvil o tableta del usuario. A partir de estas entradas, el usuario sería capaz de desplazar el dron en un entorno tridimensional y simular su comportamiento.

En esta misma línea, también se podría permitir a los usuarios crear sus propios escenarios interactivos para probar funcionalidades con módulos más avanzados como podría ser un contenedor o unas pinzas para agarrar objetos.

Adicionalmente, se podrían añadir más funcionalidades sociales, como un chat para los usuarios, pero tal vez se distancien más de la idea original del proyecto, puesto que uno de los ideales del mismo es fomentar el trabajo en equipo y el trato personal entre los alumnos en el aula.

Por último, también puede ser interesante desde el punto de vista didáctico la creación de tutoriales y guías sobre el uso de la herramienta. Todas estas guías y documentos se podrían ofrecer a través de un portal de formación para jóvenes y educadores, dándoles además la posibilidad de interactuar en un foro de preguntas y respuestas para resolver sus dudas entre ellos en la comunidad.

Bibliografía

[1] Stella Fayer, Alan Lacey, and Audrey Watson - U.S. Bureau of Labor Statistics *STEM Occupations: Past, Present, And Future*.

[2] U.S. Department of Commerce - Economics and Statistics Administration Office of the Chief Economist *STEM Jobs: 2017 Update* http://www.esa.doc.gov/sites/default/files/stem-jobs-2017-update.pdf.

[3] Carnegie Mellon University School of Computer Science *History of Computer Graphics. The Computer Graphics Book of Knowledge.* http://www.cs.cmu.edu/~ph/nyit/masson/history.htm.

[4] Foley, James D. (1996) *12.7 Constructive Solid Geometry. Computer Graphics: Principles and Practice*, Addison-Wesley Professional

[5] Gascó Ortiz, V. (2015) *Videojuego de construcción mediante piezas ensamblables en navegador* https://riunet.upv.es/handle/10251/64459

[6] Scopigno, R.; Callieri, M.; Dellepiane, M.; Ponchio, F.; Potenziani, M. (2017) *Delivering and using 3D models on the web: are we ready?* https://riunet.upv.es/handle/10251/90124

[7] Salomon, David. Springer (2006) *Transformations and projections in computer graphics* https://link.springer.com/book/10.1007%2F978-1-84628-620-9

[8] Neelakantam, S. & Pant, T. (2017) *Learning Web-based Virtual Reality: Build and Deploy Web-based Virtual Reality Technology*

[9] A-Frame official documentation https://aframe.io/docs/0.8.0/introduction/

[10] Three.js official documentation https://threejs.org/docs/index.html

[11] Etienne, Jeromee (2017) *Creating Augmented Reality with AR.js and A-Frame* https://aframe.io/blog/arjs/

APÉNDICE A
Glosario

A continución se incluyen algunos términos que pueden resultar desconocidos o confusos para el lector, con tal de facilitar su familiarización con el campo de este proyecto.

- **3D**: tres dimensiones. Comunmente se emplea este término para referirse a los gráficos creados por computador.
- **AR, AV o realidad aumentada**: paradigma que permite la visualización de gráficos 3D superpuesta a un entorno real.
- **Base de datos**: sistema de almacenamiento que permite la estructuración de datos pertenecientes a un mismo campo.
- **Back-end**: dentro de un proyecto *software* es la capa que permite el acceso a los datos y su manipulación. En el caso de las aplicaciones web corresponde a todo el software que se ejecuta en el servidor.
- **Buffer**: espacio de memoria en el cual los datos se almacenan por un tiempo limitado.
- **CAD o Computer Aided Design**: el diseño asistido por computador es un campo desarrolladas para la asistencia en el diseño y en la ingeniería.
- **Cross Site Scripting**: vulnerabilidad del *software* que permite a un usuario inyectar código JavaScript malicioso en una página web.
- **Front-end**: capa de proyecto *software* con la que interactúa el usuario. En una aplicación web se constituye de las distintas interfaces y páginas que visita el usuario.
- **Gema**: librería de código para Ruby.
- **git**: herramienta de control de versiones que permite la creación de un historial con todos los cambios que sufre un proyecto.
- **Inyección SQL**: vulnerabilidad *software* por la que un usuario puede ejecutar código SQL intruso. Comunmente se debe a formularios web cuyos datos no se verifican antes de ser procesados.
- **MR, RM o realidad mixta**: paradigma que permite la visualización de gráficos 3D en el que en lugar de superponerse, se integran en el entorno de modo que crea la sensación de estar realmente en el medio.
- **MVC**: patrón de arquitectura *software* en el cual un proyecto se separa la lógica de los datos por medio de tres componentes:

  - modelos: permiten el acceso a los datos.
  - vistas: muestran los datos al usuario.
  - controladores: procesan los datos y realizan acciones.

- **Raycaster**: elemento de una escena 3D que emite rayos imaginarios de luz, cuyas colisiones se simulan para calcular su trayectoria. Esta técnica se emplea para determinar qué objeto selecciona un usuario al hacer clic en una pantalla.
- **SQL**: lenguaje que permite efectuar consultas en bases de datos y realizar operaciones sobre las mismas.
- **STL**: formato de fichero CAD que describe exclusivamente la geometría de un objeto tridimensional.
- **Test**: prueba realizada para verificar el correcto funcionamiento del código. Comunmente se diseñan tests automatizados para agilizar el proceso de validación.
- **Transformación 3D**: modificación de la geometría de un objeto en un espacio tridimensional. Típicamente, una operación de escalado, rotación o translación.
- **VR, RV o realidad virtual**: paradigma que permite la visualización de gráficos 3D en un entorno completamente virtual.
- **WebWorker**: proceso de JavaScript que se ejecuta en segundo plano de forma paralela al hilo de ejecución principal de una página web. Esta tecnología permite la carga asíncrona de ficheros pesados sin bloquear la interacción del usuario con una página web.

APÉNDICE B

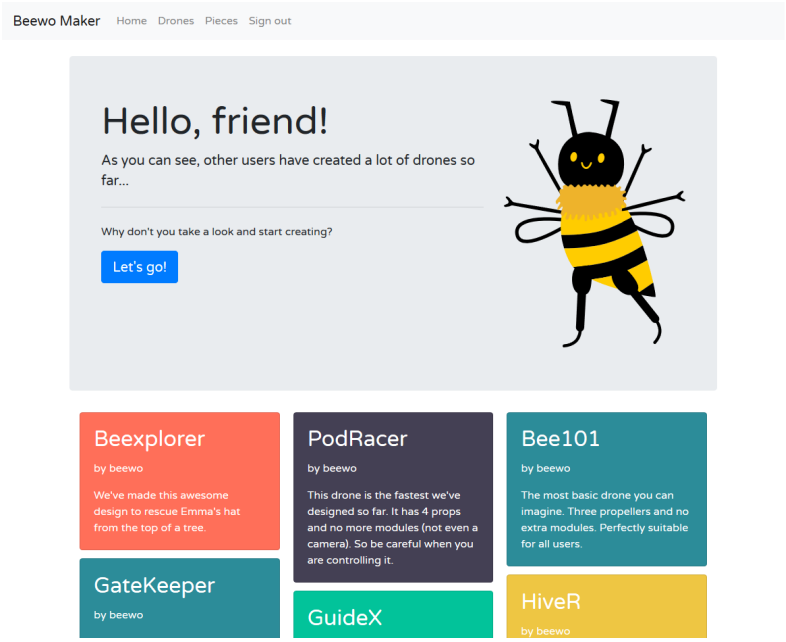
Manual de usuario de la aplicación

B.1 Creación de una cuenta de usuario e inicio de sesión

En primer lugar, visita http://beewomaker.herokuapp.com/users/sign_up y rellena el formulario para crear tu cuenta. En caso de que ya tengas una cuenta creada, puedes iniciar sesión accediendo a http://beewomaker.herokuapp.com/users/sign_in o utilizando el enlace del "Sign in"del menú superior.

B.2 Galería de diseños

La página principal de la aplicación muestra la galería de diseños de todos los usuarios. También puedes acceder desde http://beewomaker.herokuapp.com/.



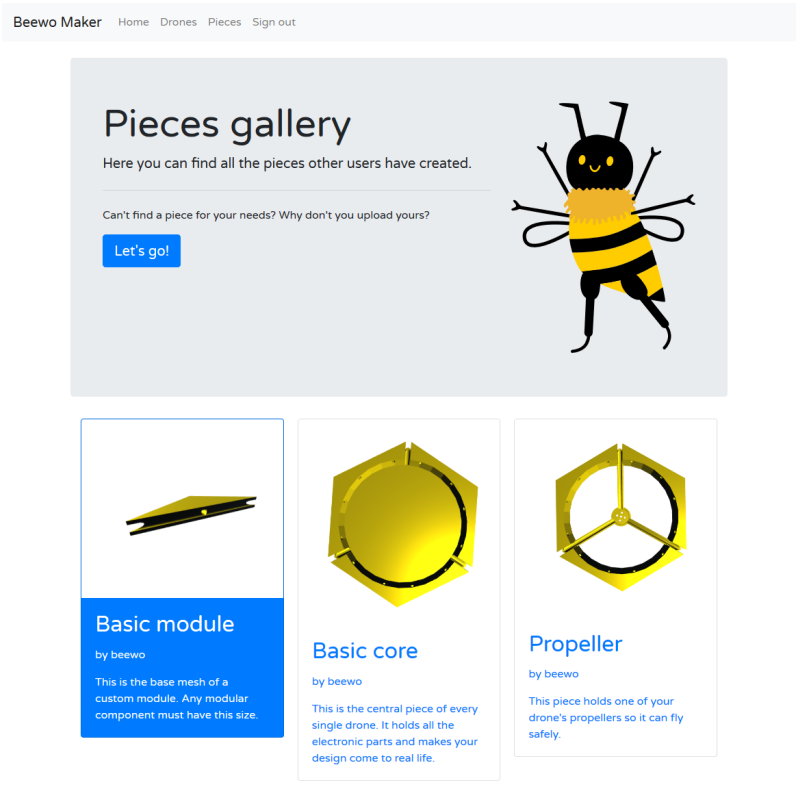
Desde esta página puedes ver la colección global de drones diseñados por otros usuarios. **Ten en cuenta que en esta galería solo aparecen los drones que otros usuarios han marcado como públicos.**

En la parte superior aparece un botón para la creación de un nuevo diseño. Ver apartado Creación de un dron para más detalles sobre el proceso.

B.3 Galería de piezas

Del mismo modo que en el apartado anterior, la galería de piezas muestra todas las piezas creadas por los usuarios y que han sido marcadas como públicas por sus autores.

Las piezas que aparecen marcadas en azul son aquellas con una licencia permisiva para reutilizarlas en tus diseños.



En la parte superior aparece un botón para la creación de nuevas piezas. Ver apartado Creación de una pieza para más detalles sobre el proceso.

B.4 Creación de un dron

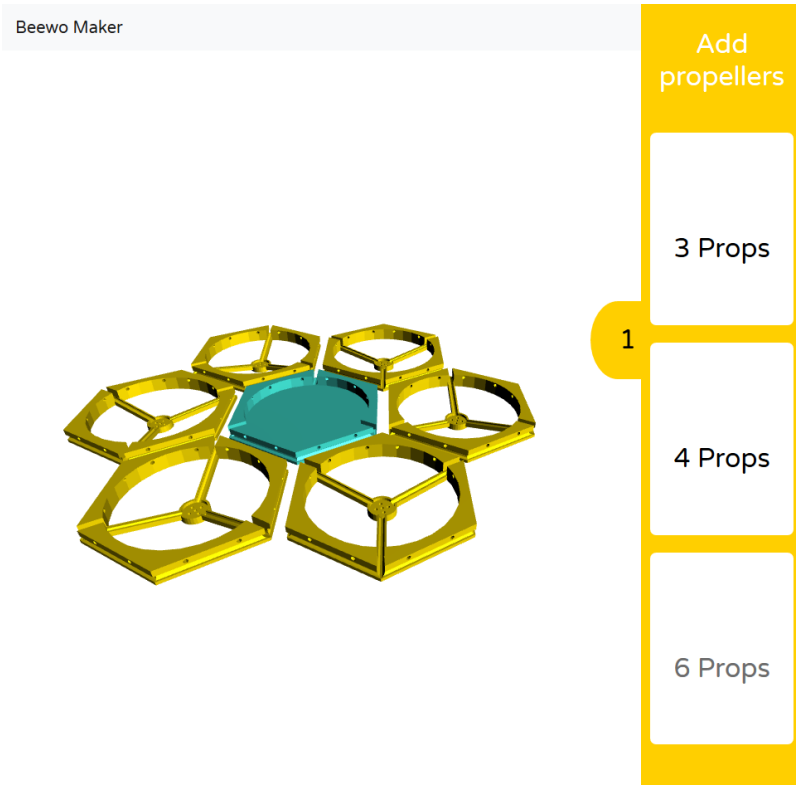
Para crear un dron, en primer en primer lugar, rellena la información básica del formulario siguiendo las indicaciones de Abee.

Una vez hayas rellenado el formulario debes tener en cuenta que el proceso de creación de un dron se compone de tres pasos:

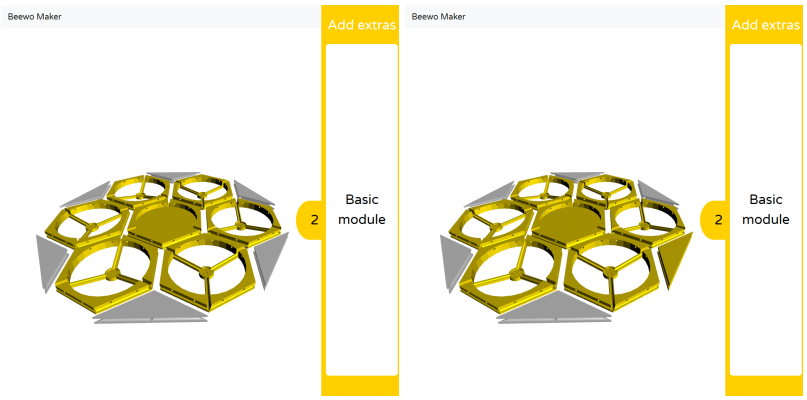
1. Selección del número de propulsores.
2. Adición de módulos de la galería.
3. Validación y almacenamiento del diseño.

Puedes girar la cámara haciendo clic y arrastrando. También puedes desplazarte haciendo clic derecho y arrastrando. La rueda del ratón permite ampliar y reducir el tamaño de todas las piezas. **En dispositivos móviles, estas acciones corresponden respectivamenta a arrastrar un dedo, arrastrar dos y pellizcar la pantalla.**

Abriendo la primera pestaña puedes seleccionar una de las tres disposiciones base para los propulsores del dron.

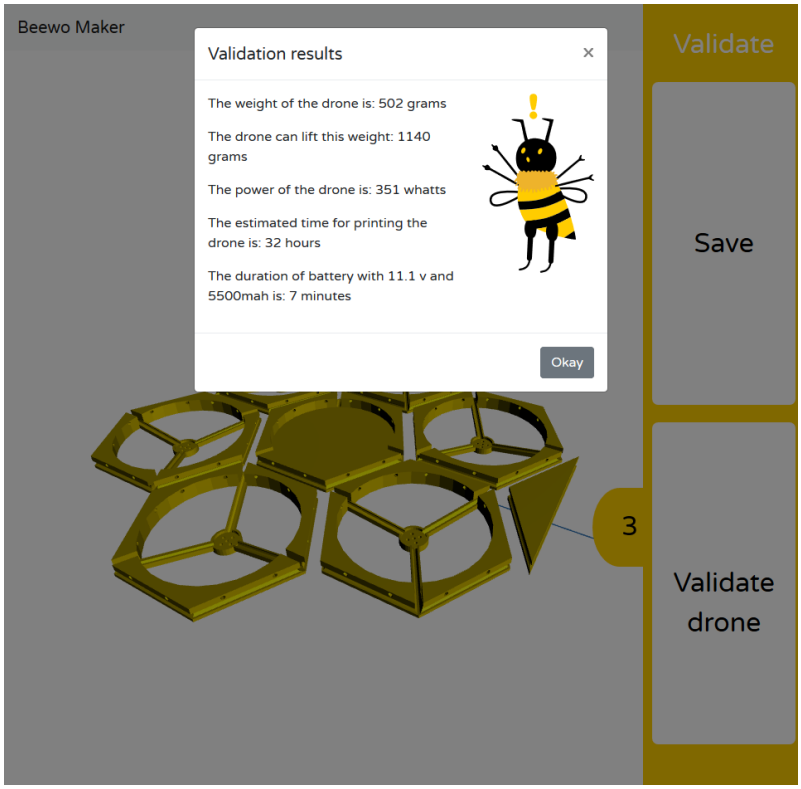


A continuación, abre la segunda pestaña para ver todos los módulos de la galería disponibles en la barra lateral. Para ello, fíjate en las posiciones grises. Puedes añadir tu pieza en cualquiera de ellas con un simple clic.

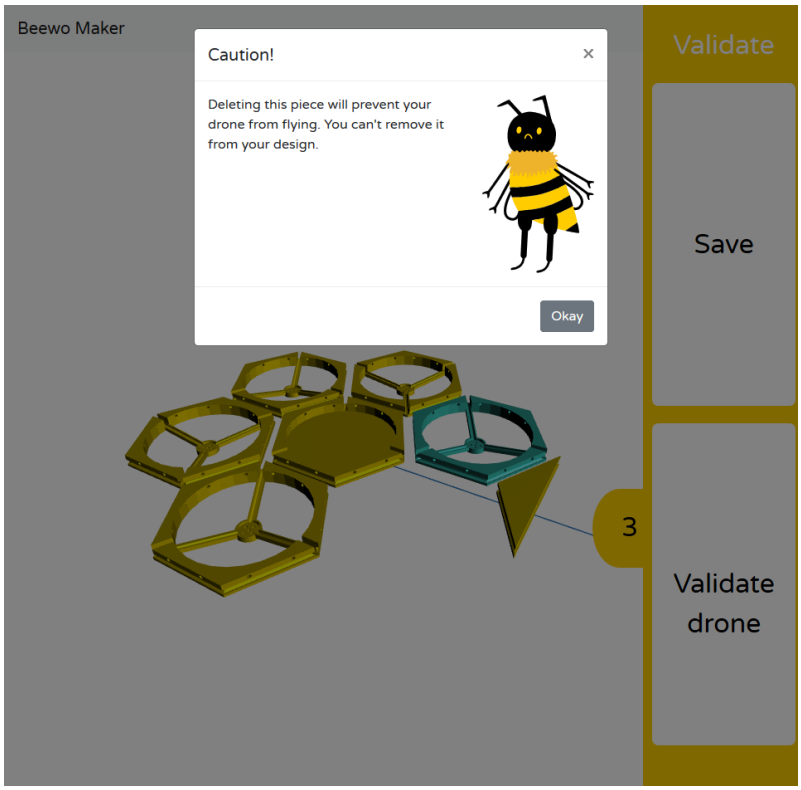


Por último, una vez hayas acabado tu diseño, abre la tercera pestaña y pulsa el botón de validar para comprobar que has construido bien tu dron. Si decides guardar directamente, la validación ocurrirá en segundo plano cuando se guarda el diseño para poder efectuar la estimación de los parámetros del mismo.

La flecha que aparece en pantalla indica hacia dónde se inclinará el dron si hay algún desequilibrio y debes añadir o quitar alguna pieza para compensar el peso. Si la flecha apunta hacia arriba es que el dron está perfecto.



En cualquier momento, puedes seleccionar cualquier pieza y eliminarla haciendo click derecho. Sin embargo, si a la hora de eliminar una pieza el sistema detecta que queda una pieza inconexa verás este mensaje de error.



B.5 Creación de una pieza

Para crear una pieza debes rellenar el formulario de http://beewomaker.herokuapp.com/pieces/new y subir un fichero en formato STL con las dimensiones correctas. Puedes descargar los modelos base desde http://beewo.es para diseñar los tuyos propios a partir de estos.

B.6 Visualización de un dron

Para visualizar un dron simplemente selecciona su tarjeta en la galería.

Puedes girar la cámara haciendo clic y arrastrando. También puedes desplazarte haciendo clic derecho y arrastrando. La rueda del ratón permite ampliar y reducir el tamaño de todas las piezas. **En dispositivos móviles, estas acciones corresponden respectivamenta a arrastrar un dedo, arrastrar dos y pellizcar la pantalla.**

Cada vez que interactúas con el visor la tarjeta de detalles del dron se aparta para que puedas hacerlo más cómodamente. **Tras 2 segundos sin interacción la tarjeta volverá a aparecer.**

Además, puedes utilizar cualquiera de los siguientes cuatro botones:

- Pulsando el botón *Details* puedes ver una tarjeta con los parámetros del dron (peso, carga máxima, etc.).
- Si eres el autor del dron que estás viendo puedes pulsar *Redesign* para editarlo de nuevo.
- El botón de *Print* te permite descargar un fichero zip con los modelos de las piezas del dron y unas instrucciones para su impresión.
- El botón *Share* inicia la descarga de un fichero de imagen con el código QR del dron. Escanea este QR con tu móvil o tablet para verlo en realidad aumentada. Solo necesitarás usar el marcador de AR adjunto. Puedes imprimirlo o mostrarlo en alguna pantalla para escanearlo con tu dispositivo.

B.7 Visualización de un dron en realidad aumentada

Escanea el QR que has descargado para abrir el visor de realidad aumentada y cuando se abra la aplicación web apunta con tu cámara al marcador de realidad aumentada adjunto.

Si tienes problemas para ver el dron realiza las siguientes acciones:

- Si tu navegador muestra un error relativo a la seguridad de la aplicación o algún mensaje relacionado con HTTPS prueba a utilizar Mozilla Firefox.
- Limpia la cámara de tu dispositivo.
- Verifica que no estás tapando el marcador.
- Mantén el marcador sobre una superficie plana.