UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Desarrollo de una aplicación web para el tratamiento de datos de navegación de una aeronave no tripulada.

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Autor: Julia González Valiente

Tutor: José Vicente Busquets Mataix
Pascual Pérez Blasco

Curso 2017-2018

Resum

El projecte es dividix en dos apartats. D'una banda, hi ha una connexió en temps real entre el controlador de l'aeronau no tripulada i un servidor extern. Aquest servidor realitza la funció de conectar amb la base de dades. En aquesta base de dades s'emmagatzema informació dels vols realitzats per dita aeronau. Aquesta comunicació es realitza mitjançant un traspas de dades per connexió 3G. D'altra banda, hi ha una aplicació web que mostra les dades emmagatzemades anteriorment. Aquesta aplicació permet registrar tant usuaris com aeronaus assignades a un usuari en particular, a més d'estar adaptada per a la correcta visualització en diversos dispositius (*smartphones*, *tablets*, *pc's*). Es pot visualitzar una estadística de la informació de navegació obtinguda per cada aeronau registrada. La informació que es visualitza a l'aplicació web s'obté mitjançant un mòdul GPS inclòs al dispositiu que posseïx l'aeronau i indica l'altitud mitjana,la velocitat mitjana, la duració d'un vol, la distància recorreguda, un mapa que mostra la trajectòria que realitza l'aeronau,i estadístiques de l'altitud i velocitat màxima obtingudes durant un vol.

Paraules clau: rastreig de vol, dron, aeronau, comunicació 3G, GPS, geolocalització

Resumen

El proyecto se divide en dos apartados. Por una parte, existe una conexión en tiempo real entre el controlador de la aeronave no tripulada y un servidor externo. Este servidor realiza la función de conectar con la base de datos. En esta base de datos se almacena información de los vuelos realizados por dicha aeronave. Esta comunicación se realiza mediante un traspaso de datos por conexión 3G. Por otra parte, existe una aplicación web que muestra los datos almacenados anteriormente. Esta aplicación permite registrar tanto usuarios como aeronaves asignadas a un usuario en particular, además de estar adaptada para la correcta visualización en distintos dispositivos (*smartphones*, *tablets*, *pc's*). Se puede visualizar una estadística de la información de navegación obtenida por cada aeronave registrada. La información que se visualiza en la aplicación web se obtiene mediante un módulo GPS incluido en el dispositivo que posee la aeronave e indica la altitud media, la velocidad media, la duración de un vuelo, la distancia recorrida, un mapa que muestra la trayectoria que realiza la aeronave y estadísticas de la altitud y velocidad máxima obtenidas durante un vuelo.

Palabras clave: rastreo de vuelo, dron, aeronave, comunicación 3G, GPS, geolocalización

Abstract

The project is divided into two sections. On the one hand, there is a real-time connection between the controller of the unmanned aircraft and an external server. This server performs the function of connecting to the database. In this database is stored information of the flights made by that aircraft. This communication is carried out by means of a data transfer via 3G connection. On the other hand, there is a web application that shows the previously stored data. This application can register both users and aircraft assigned to a particular user, in addition to being adapted for the correct display on different devices (smartphones, tablets, pc's). It is possible to visualize a statistic of the navigation information obtained by each registered aircraft. The information displayed in the web application is obtained through a GPS module included in the device that owns the aircraft and indicates the average altitude, the average speed, the duration of a flight, the distance traveled, a map showing the trajectory performed by the aircraft and statistics of altitude and maximum speed obtained during a flight.

Key words: flight tracking, drone, aeroplane, 3G communication, GPS, geolocation

Índice general

Índice de figuras

CAPÍTULO 1
Introducción

En este apartado se detallan los motivos principales y los objetivos que aspira tener el proyecto a implementar. También se explica la estructura que sigue la memoria, es decir, cómo se ha organizado y elaborado este proyecto.

1.1 Motivación

Hoy en día el desarrollo de aplicaciones web es uno de los campos de desarrollo más demandados en el mercado laboral.

Así que, por un lado, a la autora de este proyecto le interesa hacer un desarrollo web y obtener conocimientos de ello ya que en la universidad solo se han enseñado cosas básicas de ese campo.

Por otro lado, a la autora del proyecto siempre le ha interesado el concepto del desarrollo de dispositivos *hardware*. Por desgracia, en el Grado de Ingeniería Informática no se obtienen dichos conocimientos, así que aprenderlo en el desarrollo del Trabajo de Final de Grado era una buena idea.

De esta forma se consigue obtener unos conocimientos que hacen a una ser más competente y, por tanto, poder enfrentarse al mundo laboral con menos inconvenientes.

1.2 Objetivos

El objetivo principal de este proyecto es desarrollar una aplicación web para gestionar los datos de geolocalización almacenados a tiempo real de todos los dispositivos compatibles de los que se disponga. El proyecto estará disponible en la red dado que se establecerá en un servidor web. Los usuarios podrán registrarse de forma gratuita. Una vez estén registrados, podrán registrar sus propios dispositivos y visualizar todos los datos almacenados por dichos dispositivos. Se mostrarán estadísticas de los datos georeferenciados y almacenados para cada dispositivo. También se podrá observar a tiempo real la trayectoria realizada por uno de los dispositivos pintándola sobre una mapa.

1.3 Estructura de la memoria

En este apartado se explica cómo ha quedado estructurada la memoria respecto a las fases por las que se ha ido desarrollando el proyecto. La memoria se ha dividido en dos bloques importantes: la aplicación y la monitorización. Se ha decidido separar estos dos

apartados por el motivo de que no tienen demasiado que ver uno con el otro. Sí que existe una conexión final, pero a la hora del desarrollo son dos cosas distintas, ya que cada uno de los bloques utiliza unas tecnologías y unos entornos de trabajo completamente diferentes y sin ninguna relación unos de otros. Dicho esto, se pasa a detallar los capítulos en los cuales se ha dividido este proyecto:

- **Capítulo 1**: Introducción. En este capítulos se introduce un poco en el proyecto a desarrollar. Se divide en varios apartados donde se detalla la motivación por la cual se ha elegido este proyecto, los objetivos que se esperan al realizarlo y la estructura que tiene la memoria.
- **Capítulo 2**: Este capítulo detalla varios apartados enfocados únicamente en la realización de la aplicación web que son: una breve explicación sobre las tecnologías utilizadas en la aplicación web, un listado de todos los requisitos observados tras realizar un análisis de funcionalidad visualizados mediante casos de uso y diagramas de contexto.
- **Capítulo 3**: Este capítulo detalla varios apartados enfocados únicamente en la realización de la monitorización de la aplicación web comentada anteriormente que son: una breve explicación sobre las tecnologías utilizadas en la parte de la monitorización de la aplicación web, un listado de todos los requisitos observados tras realizar un análisis de funcionalidad visualizados mediante casos de uso y diagramas de contexto.
- **Capítulo 4**: Implantación. Se explica la forma en la cual se ha podido poner en marcha la aplicación web en la red.
- **Capítulo 5**: Pruebas. Se detallan algunas pruebas funcionales realizadas sobre la aplicación.
- **Capítulo 6**: Conclusiones. Se comentan las conclusiones obtenidas técnicas y personales sobre el proyecto. También se comentan las posibles mejoras a desarrollar en un futuro
- **Capítulo 7**: Agradecimientos. Se realiza un breve texto de agradecimiento a todas las personas implicadas.
- **Bibliografía**. Se listan todas las referencias bibliográficas o enlaces consultados para la realización del proyecto y memoria.
- **Glosario**. Se lista un conjunto de palabras o expresiones utilizadas en el proyecto para facilitar la comprensión del lector.

CAPÍTULO 2
Bloque I: La aplicación

En esta sección se explican las tecnologías utilizadas, el análisis y el desarrollo basado directamente en la creación de la aplicación web.

2.1 Tecnologías utilizadas

Este apartado describe brevemente todas las tecnologías utilizadas en la aplicación implementada.

2.1.1. ¿Qué es *MEAN Stack*?

Figura 2.1: MEAN logo

MEAN son las siglas de los cuatro sistemas *software* que lo componen: MongoDB, Express, Angular y NodeJS. Forma un conjunto de tecnologías para el desarrollo de aplicaciones, y páginas web dinámicas basadas todas ellas en el lenguaje de programación JavaScript, tanto en el servidor como en el cliente y en la base de datos.[6][23]

2.1.2. Angular 5

Angular es un *Framework* de desarrollo mantenido por Google que se utiliza para crear *Single page application (SPA)*, o aplicaciones de página única. Sirve para desarrollar aplicaciones escalables en la parte del cliente o *frontend*, ya que traslada parte de la funcionalidad que tradicionalmente se realizan en el servidor al lado del cliente (navegador web).

Figura 2.2: Angular logo

Se trata de los *frameworks* más populares actualmente [15], y como características principales, se puede decir que es dogmático (realiza tareas de configuración por defecto y de forma transparente al desarrollador), escalable, fiable y, gracias a la facilidad de modularización, facilita y promueve la reutilización del código. Además, tiene completa documentación y cuenta con una gran comunidad de desarrolladores que lo envuelve y que proporcionan herramientas y bibliotecas de código abierto.

Sigue un patrón de arquitectura *software* MVC (Modelo-Vista-Controlador), y el lenguaje principal es Typescript [10], aunque, para su ejecución, se realiza una compilación a Javascript/ECMAScript.

Una aplicación Angular está formada principalmente por componentes y servicios.[1] Los componentes son la unidad básica de Angular y están compuestos por una plantilla escrita en HTML, una clase Typescript que actúa de controlador y una hoja de estilo CSS. Los servicios se inyectan en los componentes y los proveen de funcionalidad.

Tanto los componentes, como los servicios se agrupan en unidades de funcionalidad llamadas módulos. Durante el proyecto son necesarios diferentes componentes y servicios que se describen detalladamente en el capítulo 2.3.2.

2.1.3. Node JS

Actualmente, NodeJS es una de las tecnologías más populares (según la página Stackoverflow [15]). Se trata de un entorno de ejecución multiplataforma para el lenguaje JavaScript (actualmente ya soporta ECMAScript 2015 o ES6) construido con el motor V8 de Chrome [17]. Es un entorno orientado a eventos asíncronos, es decir, operaciones de entrada y salida no bloqueantes, ésto permite que sea eficiente y liviano.

Figura 2.3: NodeJS logo

Pongamos por ejemplo que se produce una petición en la cual se tiene que realizar por un lado la gestión de la misma petición, y por otro lado, un acceso a disco. Un servidor tradicional que se gestione sincrónicamente, ejecutaría de forma secuencial las dos operaciones, quedando libre una vez finalice la operación de E/S y poder atender a nuevas peticiones. Mientras que en NodeJS, una vez gestionada la petición, se podría lanzar el acceso a disco de forma asincrónica y así ya quedaría libre para atender a nuevas peticiones. Esto es posible a la existencia de un bucle de funciones de devolución o *callbacks* [11], las cuales son llamadas después de completar los trabajos.

El entorno está formado por un conjunto de librerías (hemos comentado el motor V8 de Chrome) y un conjunto de herramientas. Una herramienta indispensable y que se utiliza a lo largo del proyecto es npm. Se trata de una combinación de tres elementos: un sitio web, una interfaz de línea de comandos y una base de datos sobre Javascript y meta-información. Nos permite gestionar módulos y configurar entornos concretos y dependencias en un proyecto Javascript.

Además, durante el desarrollo del proyecto se hace uso una pequeña herramienta (*shell script*) para facilitar la gestión de las diferentes versiones de NodeJS llamada nvm (*node version manager*).[4]

2.1.4. Express

Express, o también llamado Express.js, es un *framework* de código abierto y bajo licencia MIT (MIT License) que permite crear una infraestructura web sobre Node.js de forma rápida y sencilla. Está formado por un conjunto de módulos Javascript, que sumado a NodeJS, proporciona un conjunto de aspectos que son comunes en el desarrollo de aplicaciones web.

Entre otras cosas, podemos destacar:

- Gestión de rutas.
- Motores de plantillas.
- Integración con bases de datos.
- Mecanismos de tratamiento de peticiones.
- Mecanismos de sesiones y gestión de *cookies*.

Se instala como un módulo con npm y es necesaria la instalación previa de NodeJS. Una vez lanzada la instalación del *framework*, automáticamente la herramienta descarga e instala todos los módulos necesarios y crea una estructura de directorio inicial.

En el proyecto no se utilizan motores de plantillas ni mecanismos de sesiones, ya que la parte alojada en el servidor NodeJS consiste en una API REST y tampoco se utiliza ni sesiones ni *cookies* ya que para identificar el origen de las peticiones se utiliza JWT (Json Web Tokens) intercambiados en las cabeceras de los paquetes.

2.1.5. MongoDB

MongoDB es una base de datos de código abierto y no relacional que almacena datos como documentos en una representación binaria llamada BSON (*Binary JSON*). La información relacionada se almacena junta para tener un acceso rápido a la consulta a través del lenguaje de consulta MongoDB.

Los campos pueden variar de un documento a otro, es decir, no es preciso declarar una estructura fija de los documentos. Si es necesario añadir un nuevo campo a un documento, el campo se puede crear sin afectar a todos los demás documentos almacenados en la colección, sin tener que desconectar el sistema ni actualizar un catálogo del sistema central. [2]

Figura 2.4: MongoDB logo

El modelo de datos de documentos de MongoDB se mapea como objetos en el código de la aplicación, muy similares a objetos JSON, haciendo que sea más sencillo para los desarrolladores de aprender y usar. Los documentos ofrecen asimismo la posibilidad de representar relaciones jerárquicas para almacenar fácilmente matrices y otras estructuras de datos más complejas.

2.1.5.1. ¿Por qué base de datos no relacional?

Organizaciones de distintos tamaños están adoptando MongoDB en sus proyectos porque les permite construir aplicaciones más rápido, manejar tipos de datos muy diversos y administrar aplicaciones más eficientemente a mayor escala.

El uso de MongoDB elimina la compleja capa de mapeo objeto-relacional (ORM) que traduce los objetos en código a tablas relacionales.El modelo de datos flexible de MongoDB también significa que su esquema de base de datos puede evolucionar con los requerimientos del negocio.

La eterna lucha entre la utilización de base de datos relacional o no relacional recae actualmente sobre MongoDB y MySQL.[9]

Figura 2.5: Lucha entre mongoDB y mySQL

MongoDB se puede escalar a través de múltiples centros de datos distribuidos, por tanto, proporciona nuevos niveles de disponibilidad y escalabilidad que con MySQL (base de datos relacional) eran inalcanzables. A medida que sus implementaciones crecen en términos de volumen de datos y rendimiento, MongoDB se escala fácilmente sin tiempo de inactividad y sin cambiar su aplicación. Por el contrario, para lograr la escala con MySQL a menudo se requiere un trabajo de ingeniería significativo y personalizado.[14]

En la tabla siguiente se muestran las diferencias entre una base de datos y otra:

Mongo DB (No relacional)	SQL (relacional)
Código abierto	Código abierto sólo con MariaDB
Estructura interna de BSON	Estructura interna patentada para el motor de almacenamiento y expuesta a través de un intérprete SQL
Parcial, sólo hay soporte para estructuras básicas de datos	Modelo de datos flexibles
Tiene escalabilidad horizontal	No tiene escalabilidad horizontal
Colección	Tabla
Documento	Fila
Campo	Columna
id	Clave primaria
aggregation	Group by

Figura 2.6: Diferencias entre mongoDB y mySQL

2.1.6. Lenguajes de programación

En esta sección se detallan todos los lenguajes de programación utilizados en el desarrollo de la aplicación web.

2.1.6.1. Javascript

[22]En 1995, Brendan Eich, creó un lenguaje que después de haber sido renombrado varias veces llegó a llamarse **JavaScript**. Poco después fue estandarizado bajo el nombre ECMAScript. La última versión es ECMAScript 6 o ES6.

Javascript nació por la necesidad de ampliar las posibilidades de las páginas estáticas HTML, permite crear contenido dinámico. Es un lenguaje de programación orientado a objetos.

En la actualidad, todos los navegadores modernos interpretan el código Javascript integrado en las páginas web ya que funciona en el lado del cliente y se ejecuta desde el navegador. [5]

Figura 2.7: Javascript logo

2.1.6.2. TypeScript

TypeScript es un lenguaje de código abierto. Un superconjunto de JavaScript que añade tipos, clases y módulos opcionales a Javascript. Se utiliza este lenguaje en la parte cliente, donde angular realiza un *transpilling* o compilación para convertirlo a Javascript[24].

2.1.6.3. HTML

HTML son las siglas de Hyper Text Markup Language, lo que significa que es un lenguaje de marcas de hipertexto. El código HTML es un lenguaje fácil de interpretar. Hoy en día es el lenguaje que más predomina en el campo del desarrollo web. El lenguaje HTML sirve para definir la estructura básica de una página y organizar la forma en que se muestra su contenido.

Figura 2.8: HTML logo

2.1.6.4. CSS

CSS son las siglas en inglés *Cascading Style Sheets*, que traducido al español significa Hoja de estilos en cascada. Este lenguaje que se utiliza para determinar el diseño visual o estilo de los documentos web escritos anteriormente en HTML. Con este lenguaje se consigue una mayor precisión, ya que el tamaño y la posición de los elementos que conforman la web serán exactos. [21]

Figura 2.9: CSS logo

2.1.7. Hojas de estilo: Bootstrap

Bootstrap se trata de un *framework* desarrollado para diseñar aplicaciones web. Solo se basa en el desarrollo *frontend*. Uno de los puntos fuertes de este *framework* es su posible desarrollo *responsive*, es decir, diseños adaptables a cualquier tipo de dispositivo y tamaño de pantalla que sea capaz de navegar por la red. Hoy en día utilizando el lenguaje CSS también es posible diseñar aplicaciones *responsive*, pero es bastante más complejo de desarrollar en este lenguaje.[19][16]

Figura 2.10: Bootstrap logo

2.1.8. La seguridad: JWT

La seguridad es una parte muy importante en el desarrollo de una aplicación. El intercambio de cadenas cifradas como mecanismo de autenticación es un concepto de seguridad cada vez más utilizado en la actualidad. JWT son las siglas en inglés de *JSON Web Token*, y se utiliza para realizar un inicio de sesión único. Su forma de ejecución se basa en la generación de un token por parte del servidor, que luego es transmitido al cliente. Este cliente deberá presentarlo en cada invocación para poder ser autenticado.

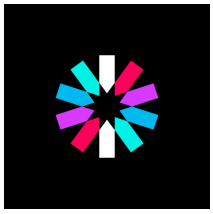

Figura 2.11: JWT logo

2.1.9. Mapa: Google maps

Uno de los mapas más conocidos y utilizados en la actualidad es Google Maps. Permite acercar a los usuarios al mundo real a través de mapas estáticos y dinámicos. Dispone de unas APIs de geolocalización que se utilizan en este proyecto.

Figura 2.12: Google maps logo

2.2 Análisis

Las actividades de análisis y diseño ayudan a transformar los requerimientos previos en un diseño implementable en *software*.En este apartado se analizan solo los requerimientos funcionales del sistema, ignorando las restricciones de la arquitectura del sistema. El objetivo del análisis es que todos los requisitos existentes en la aplicación a desarrollar queden contemplados en el diagrama de casos de uso.

2.2.1. Requisitos

El análisis de requisitos es uno de los apartados más importantes ya que debe quedar claro todo aquello que se va a implementar.[8] El listado de los requisitos debe ser lo más completo posible para evitar futuros problemas con el cliente. En este caso el cliente es el mismo desarrollador, aún así, también pueden haber problemas por no tener muy definido desde un principio todo aquello que se vaya a implementar, y por tanto, el tiempo de desarrollo podría alargarse más de lo debido.

El principal objetivo de la aplicación es gestionar los datos de vuelos que realizan varios drones. La aplicación dispone de uno o varios usuarios que pueden gestionar toda esa información. Todo el contenido de la página web se almacena en una base de datos no relacional.

A continuación, se detallan brevemente las funciones de la aplicación implementada:

Título	Crear cuenta
Propósito	Un usuario se registra en el sistema para poder acceder luego a la aplicación.
Entrada	El usuario rellena un formulario con varios datos(nombre del usuario, apellidos, email y contraseña).
Proceso	Se debe comprobar que ese usuario no existe en la base de datos y además que los datos introducidos en el formulario son todos correctos.
Salida	Si ese usuario ya existe en la base de datos, se muestra un mensaje de error en pantalla que notifica sobre la existencia de un usuario con el mismo email. Si existe algún fallo en los datos que el usuario ha rellenado en el formulario, entonces se muestra también un mensaje de error que notifica sobre la existencia de un fallo en algún campo del formulario e indica en cuál. En cambio, si todo es correcto, se añade el usuario a la base de datos con una imagen de perfil y de portada predeterminada, luego se notifica con un mensaje de éxito 'Usuario creado correctamente'.

Figura 2.13: Requisito - Crear cuenta

Título	Iniciar sesión
Propósito	Un usuario inicia sesión en el sistema con sus credenciales.
Entrada	El usuario rellena el formulario de **inicio de sesión** donde se le pide el email y la contraseña de su cuenta.
Proceso	Se debe comprobar que ese usuario existe en la base de datos y que los datos que ha rellenado corresponden con el usuario que ya está creado.
Salida	Si los datos que ha rellenado el usuario no se corresponden con ningún usuario almacenado en nuestra base de datos, entonces se muestra un mensaje de error en pantalla que notifica sobre la inexistencia de un usuario con esas credenciales. Si existe algún fallo en los datos que el usuario ha rellenado en el formulario, entonces se muestra también un mensaje de error que notifica sobre la existencia de un fallo en algún campo del formulario e indica en cuál. En cambio, si todo es correcto, el usuario accede a la aplicación y se notifica con un mensaje de éxito.

Figura 2.14: Requisito - Iniciar sesión

Título	Cerrar sesión
Propósito	El usuario cierra su sesión.
Entrada	El usuario puede cerrar su sesión cuando quiera haciendo click sobre el botón **Cerrar sesión** del menú.
Proceso	Se debe borrar la sesión que hay creada en ese momento en el navegador.
Salida	El usuario deja de tener una sesión abierta y vuelve a la página de Inicio de sesión.

Figura 2.15: Requisito - Cerrar sesión

Título	Añadir un dron
Propósito	El usuario puede añadir un dron a su cuenta.
Entrada	El usuario rellena un formulario con varios datos (nombre del dron, número de teléfono de la tarjeta sim que tiene dicho dron)
Proceso	Se debe comprobar que ese dron no existe en la base de datos y además que los datos introducidos en el formulario son todos correctos.
Salida	Si ese dron ya existe en la base de datos, se muestra un mensaje de error en pantalla que notifica sobre la existencia de un dron que tiene el mismo usuario y con el mismo nombre. Si existe algún fallo en los datos que el usuario ha rellenado en el formulario, entonces se muestra también un mensaje de error que notifica sobre la existencia de un fallo en algún campo del formulario e indica en cuál. En cambio, si todo es correcto, se añade el dron a la base de datos con una imagen de perfil y de portada predeterminada, luego se notifica con un mensaje de éxito 'Dron creado correctamente'.

Figura 2.16: Requisito - Añadir dron

Título	Conexión a tiempo real del dron
Propósito	El usuario puede conectar la aplicación con un dron registrado y que esté activado en ese momento.
Entrada	El usuario puede conectarse a cualquiera de sus drones cuando quiera haciendo click sobre el botón **Conexión** que aparece en perfil de cada dron.
Proceso	Se debe acceder al servidor y realizar una petición a ese dron, entonces conectarse si está activado.
Salida	Si ese dron no está activado, se muestra un mensaje de error en pantalla que notifica sobre algún fallo en la conexión. En cambio, si no hay fallo en la conexión, se notifica al usuario con un mensaje de éxito 'Conectado al dron' y se abre una ventana que muestra un mapa indicando la posición del dron en ese preciso momento.

Figura 2.17: Requisito - Conexión a tiempo real

Título	Editar perfil de un usuario
Propósito	El usuario puede editar sus datos de perfil.
Entrada	El usuario modifica los campos que desea de un formulario. En este formulario, además de poder editar el nombre, apellidos, email y contraseña, también puede subir y actualizar la imagen de perfil y la de portada.
Proceso	Se debe comprobar que no hay ningún error en el formulario y que todos los campos son correctos. Si el usuario modifica una de sus imágenes de perfil o portada, se deben subir al servidor esas imágenes. Después, se actualiza el usuario modificado en la base de datos. Si no hay imágenes modificadas, se actualiza el usuario directamente en la base de datos.
Salida	Si ese usuario no existe en la base de datos quiere decir que ha habido algún error en el servidor, entonces se muestra un mensaje de error de servidor en pantalla. Si existe algún fallo en los datos que el usuario ha rellenado en el formulario, entonces se muestra también un mensaje de error que notifica sobre la existencia de un fallo en algún campo del formulario e indica en cuál. En cambio, si todo es correcto, se actualiza el usuario en la base de datos, y luego se notifica con un mensaje de éxito 'Usuario modificado correctamente'.

Figura 2.18: Requisito - Editar perfil usuario

Título	Editar perfil de un dron
Propósito	El usuario puede editar los datos de perfil de un dron determinado.
Entrada	El usuario modifica los campos que desea de un formulario. En este formulario, además de poder editar el nombre del dron, también puede subir y actualizar su imagen de perfil y de portada.
Proceso	Se debe comprobar que no hay ningún error en el formulario y que todos los campos son correctos. Si el usuario modifica una de las imágenes de perfil o portada del dron, se deben subir al servidor esas imágenes. Después, se actualiza el dron modificado en la base de datos. Si no hay imágenes modificadas, se actualiza el dron directamente en la base de datos.
Salida	Si ese dron no existe en la base de datos quiere decir que ha habido algún error en el servidor, entonces se muestra un mensaje de error de servidor en pantalla. Si existe algún fallo en los datos que el usuario ha rellenado sobre ese dron en el formulario, entonces se muestra también un mensaje de error que notifica sobre la existencia de un fallo en algún campo del formulario e indica en cuál. En cambio, si todo es correcto, se actualiza el dron en la base de datos, y luego se notifica con un mensaje de éxito 'Dron modificado correctamente'.

Figura 2.19: Requisito - Editar perfil dron

Título	Visualización de los drones
Propósito	El usuario puede ver un listado de todos los drones que tiene asignados a su cuenta.
Entrada	El usuario visualiza una lista de todos sus drones cuando accede a la aplicación y muestra su página de perfil.
Proceso	Cuando el usuario accede a su página de perfil se debe obtener de la base de datos todos los drones que tenga registrados.
Salida	Se visualiza un listado de drones donde aparece la imagen de perfil de cada dron y su nombre.

Figura 2.20: Requisito - Visualizar drones

Título	Visualización de los vuelos
Propósito	El usuario puede ver un listado de todos los vuelos asignados a un dron determinado.
Entrada	El usuario visualiza una lista de todos los vuelos que tiene uno de sus drones cuando accede a la página de perfil de ese dron.
Proceso	Cuando el usuario accede a la página de perfil de un dron se debe obtener obtener de la base de datos todos los vuelos registrados por ese dron.
Salida	Se visualiza en un listado de vuelos donde aparece el nombre de cada vuelo.

Figura 2.21: Requisito - Visualizar vuelos

2.2.2. Diagrama de contexto

El diagrama de contexto sirve para definir desde un nivel general el contexto del sistema, es decir, definir las interacciones existentes entre los agentes externos y el sistema sin describir la estructura del sistema de información.[18][7]

Otra forma de referirse al diagrama de contexto es **diagrama de flujo de datos de alto nivel**.

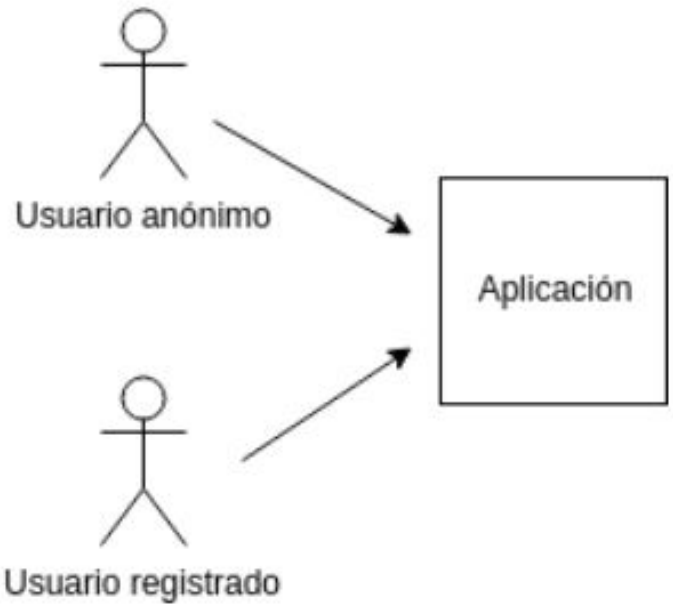

Figura 2.22: Diagrama de contexto de la aplicación

2.2.3. Casos de uso

Este apartado define un diagrama de casos de usos que define gráficamente el comportamiento UML de forma mejorada. En el diagrama de casos de uso se representa la interacción del modo en que opera el cliente con el sistema desarrollado.[20]

El diagrama de casos de uso consta de varios elementos:

- **Actor**: Es el agente externo que realiza una labor frente al sistema

Figura 2.23: Actor

- **Casos de uso**: Es una operación o tarea específica que se realiza trs una orden de algún agente externo.

Figura 2.24: Caso de uso

- **Relaciones**: Indica la invocación desde un actor o un caso de uso a otro caso de uso, es decir, a otra operación.

Figura 2.25: Relación

A continuación se muestra el diagrama de casos de uso de la aplicación que se va a implementar:

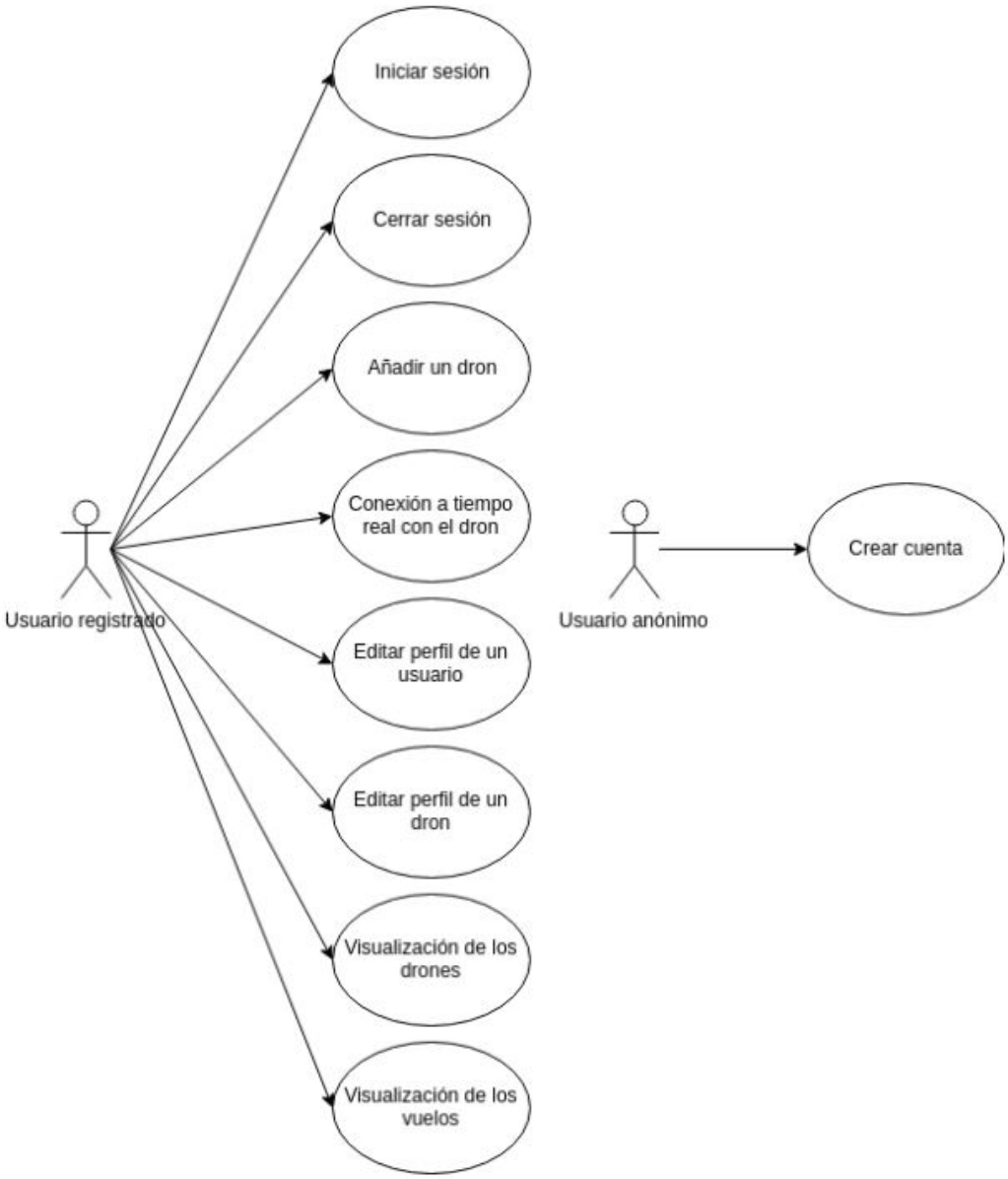

Figura 2.26: Diagrama de casos de uso

2.2.4. Modelos de la base de datos

Este apartado muestra gráficamente los modelos que existen en la base de datos y sus relaciones.

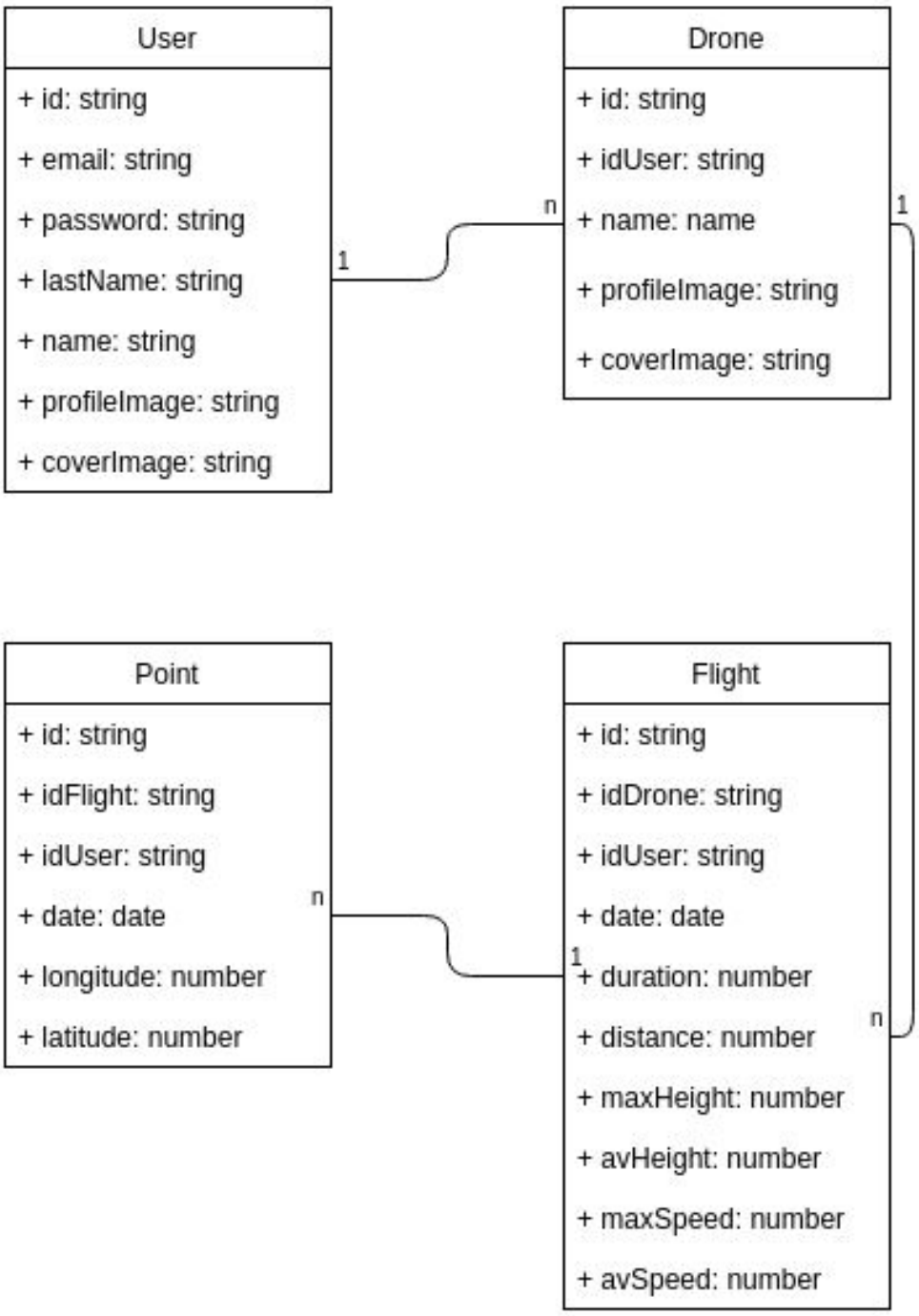

Figura 2.27: Diagrama de modelos en la base de datos

2.3 Desarrollo

Este capítulo describe los pasos que se han seguido en la instalación de las herramientas utilizadas y el desarrollo de la aplicación web. Como bien se ha comentado anteriormente, este proyecto se basa en un proyecto *MEAN stack*, que se compone de MongoDB, Express, Angular y NodeJS. La estructura de flujo de peticiones y llamadas queda de la siguiente forma:

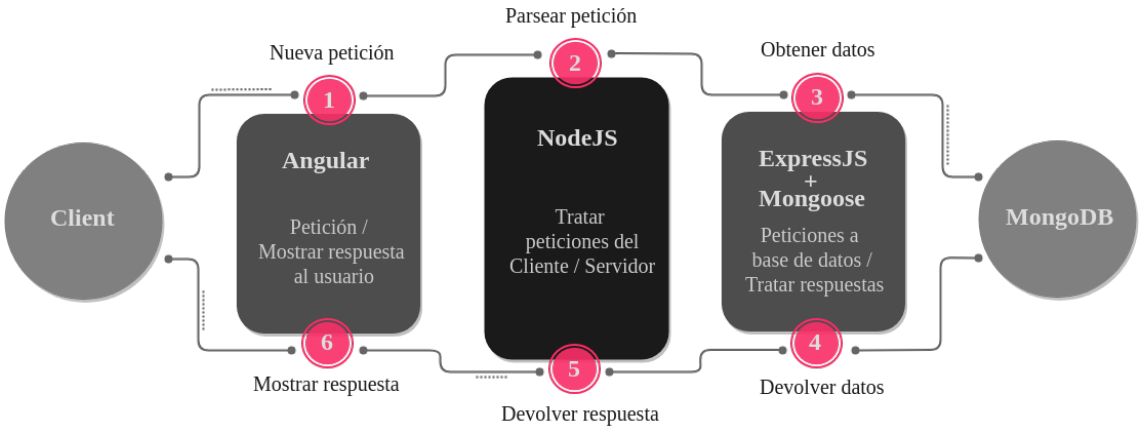

Figura 2.28: Flujo de datos de una aplicación MEAN Stack

En dicha estructura se puede observar una clara distinción entre cliente y servidor. Técnicamente se le suelen llamar a esas partes *Frontend* (cliente) y *Backend* (servidor). En la actualidad, prácticamente todas las aplicaciones web se dividen en esos dos conceptos para poder crear aplicaciones completamente independientes entre sí de una parte u otra. De esta forma, se podría cambiar de tecnología en el *frontend* utilizando el mismo tratamiento de datos de un *backend* ya implementado.

Figura 2.29: *Frontend* y *Backend*

2.3.1. *Backend*

El *backend* se dedica a gestionar todas las consultas que se hacen a la base de datos además de todas las funciones que realiza el servidor. Las tecnologías que se encuentran en esta parte son: NodeJS, Express y la base de datos que en este caso es MongoDB.

2.3.1.1. Instalación

El primer paso es instalar las tecnologías que se van a utilizar. NodeJS se instala utilizando los siguientes comandos:

```
$ sudo apt install nodejs
$ sudo aprt install npm
```

A continuación se crea un directorio llamado wild-flights-backend donde se instala Express con los siguientes comandos:

```
$ mkdir wild-flights-backend
$ cd wild-flights-backend
$ npm install express --save
```

Por último se instala MongoDB y Mongoose:

```
$ sudo apt install -y mongodb-org
$ npm install mongoose --save
```

2.3.1.2. Inicio del proyecto

Después de la instalación de las tecnologías, se crean unas carpetas y ficheros en el directorio del proyecto *backend* que se completan hasta tener una estructura de carpetas como la que sigue:

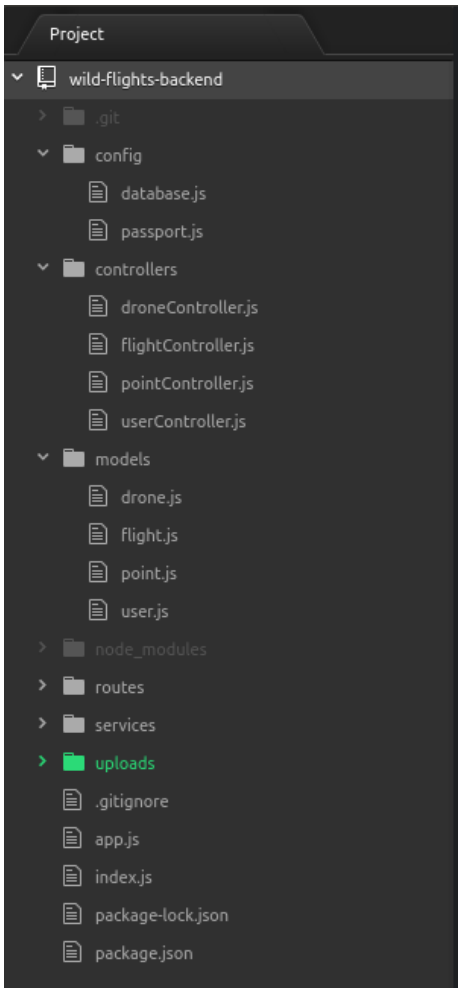

Figura 2.30: Directorio del *Backend*

2.3.1.3. Modelos de base de datos

Según lo especificado en el diagrama de clases del capítulo 2.2.4, es necesario crear un mecanismo en la aplicación para mapear cada objeto de forma correcta y controlada a la base de datos. Utilizaremos el módulo Mongoose, ya que ofrece mecanismos para facilitar la tarea.

Entre las funcionalidades que proporciona la librería, en el proyecto se ha utilizado la definición de esquemas para controlar y validar los tipos de cada elemento de la colección, y además, ganchos lógicos o mecanismos de enlace de funciones a determinados eventos y operaciones.

En la figura 2.31 se puede observar la definición del esquema para una clase o entidad Usuario. También se llama a una función después de realizar una inserción en una colección de la misma clase, que se encarga de encriptar la contraseña del usuario. Además,existe otra función que se utiliza para comprobar que la contraseña del usuario es correcta utilizada en el inicio o login de la aplicación.

```
var userSchema = new Schema({
        email : {type: String,unique: true,required: true},
        password        : String,
        lastName        : String,
        name            : String,
        profileImage    : String,
        coverImage      : String
});

// generating a hash
userSchema.pre('save', function (next) {
    var user = this;
    if (this.isModified('password') || this.isNew) {
        bcrypt.genSalt(10, function (err, salt) {
            if (err) {return next(err);}
            bcrypt.hash(user.password, salt, null, function (err, hash) {
                if (err) {return next(err);}
                user.password = hash;
                next();
            });
        });
    } else {
        return next();
    }
});

// checking if password is valid
userSchema.methods.comparePassword = function (password, callback) {
    bcrypt.compare(password, this.password, function (error, isMatch) {
        if (error) {return callback(error);}
        callback(null, isMatch);
    });
};
```

Figura 2.31: Fragmento de código del modelo de Usuario

2.3.1.4. Rutas

Para que el servidor redireccione las peticiones realizadas por el cliente a los puntos finales de la aplicación es necesario utilizar un sistema que funcione como *middleware*. Por este motivo se utiliza Express ya que dispone de una clase llamada *Router* que permite crear manejadores de rutas montables y modulares.

Según la ruta y el método de la petición HTTP, el *middleware* ejecuta la función correspondiente. Las funciones en las cuales derivan las peticiones, se agrupan en controladores, uno por cada clase o entidad. Toda petición debe devolver un estado informando sobre el éxito o fallo de la solicitud indicada.

En la figura 2.32 se muestran unos ejemplos de manejadores de rutas.

```
11 // Register
12 api.post('/signup', userCtrl.signup);
13
14 // Login
15 api.post('/login', userCtrl.login);
16
17 // Drones
18 api.route('/drones')
19   .get(passport.authenticate('jwt', {session: false}), droneCtrl.findAllDrones)
20   .post(passport.authenticate('jwt', {session: false}), droneCtrl.addDrone);
21
22 api.route('/drones/:id')
23   .get(passport.authenticate('jwt', {session: false}), droneCtrl.findById)
24   .put(passport.authenticate('jwt', {session: false}), droneCtrl.updateDrone)
25   .delete(passport.authenticate('jwt',{session: false}),droneCtrl.deleteDrone);
```

Figura 2.32: Fragmento de código de la configuración de las rutas

2.3.1.5. Autenticación

La autenticación es el mecanismo por el cual se identifica a un usuario en el sistema y se le otorga o no acceso a determinados recursos.

En el proyecto se ha optado por un mecanismo de autenticación libre de sesiones, también llamado *stateless*. Esto proporciona mayor escalabilidad e independencia. Además, a parte de evitar el establecimiento de sesiones por parte del servidor, gracias a este mecanismo se evita la utilización de *cookies* previniendo ataques CSRF (*Cross-Site Request Forgery*).

Se ha desarrollado un sistema basado en el intercambio de *tokens*, que funciona de la siguiente manera: un usuario se autentica en la aplicación mediante el par usuario/contraseña, entonces el servidor contesta con un *token* o firma cifrada que, posteriormente, el usuario almacenará y utilizará en cada petición que realice sobre el API para identificarse.

El formato utilizado del *token* es JWT. El *token* se enviará por parte del cliente en la cabecera *Authorization* de la petición HTTP y será el servidor quien lo descifre y compruebe si el *token* es correcto o no para ese cliente.

Esta comprobación se realiza solo en las rutas protegidas del *backend*. En éstas, se utiliza un *middleware* cuyo objetivo es obtener el *token*, descifrarlo, y comprobar que el usuario esté correctamente autenticado y tenga acceso a la ruta. Si el token es correcto, el *middleware* continúa la ejecución llamando a la función del controlador encargada de gestionar la petición, en caso contrario, responde con un código 403 o no autorizado.

Tanto para generar el *token* como para obtenerlo y descifrarlo, se utilizan las librerías Passport y Passport-JWT que ofrecen funcionalidades para ello.

En las siguientes figuras, se puede ver tanto un ejemplo de ruta protegida, como el fragmento de código del *middleware* que se utiliza para comprobar que el *token* es correcto. En el fragmento de la ruta, se puede ver la llamada al *middleware*, y a continuación la función que será llamada en caso de que todo se ejecute correctamente.

```
17 // Drones
18 api.route('/drones')
19   .get(passport.authenticate('jwt', {session: false}), droneCtrl.findAllDrones)
20   .post(passport.authenticate('jwt', {session: false}), droneCtrl.addDrone);
```

Figura 2.33: Autenticación en las rutas

```
1  var JwtStrategy = require('passport-jwt').Strategy,
2      ExtractJwt = require('passport-jwt').ExtractJwt;
3
4  // load up the user model
5  var User = require('../models/user');
6  var config = require('../config/database'); // get db config file
7
8  module.exports = function(passport) {
9    var opts = {};
10   opts.jwtFromRequest = ExtractJwt.fromAuthHeaderWithScheme("jwt");
11   opts.secretOrKey = config.secret;
12   passport.use(new JwtStrategy(opts, function(jwt_payload, done) {
13     User.findOne({id: jwt_payload.id}, function(err, user) {
14           if (err) {
15               return done(err, false);
16           }
17           if (user) {
18               done(null, user);
19           } else {
20               done(null, false);
21           }
22       });
23   }));
24 };
```

Figura 2.34: Configuración del *middleware* de gestión del *token*

El *token* intercambiado, no solo se utiliza para ver si determinado usuario tiene acceso a una ruta en concreto o si está correctamente autenticado, sino que también hay rutas que utilizan el *token* para identificar al usuario y responder solo con los elementos que pertenecen a éste.

Por ejemplo, cuando un usuario realiza una petición de los drones que tiene dados de alta en el sistema, el servidor comprueba en primer lugar si el *token* es correcto y no ha expirado, y en segundo lugar, obtiene el usuario y lo utiliza para obtener de la base de datos los drones que pertenecen al mismo, simulando así el establecimiento de sesiones de un servidor tradicional. Lo podemos ver en el fragmento de código de la figura 2.35.

```
//GET - Return all Drones in the DB
exports.findAllDrones = function(req, res) {
  var token = getToken(req.headers);
  var user = jwt.verify(token, 'wordsecret');
  console.log('Decoded user '+JSON.stringify(user));
  if (token) {
    Drone.find({idUser: user._id},function(err, drones) {
      if(err) res.send(500, err.message);

      console.log('GET /drones')
      res.status(200).jsonp(drones);
    });
  } else {
    return res.status(403).send({success: false, msg: 'Unauthorized.'});
  }
};
```

Figura 2.35: Obtención de los drones de un usuario

2.3.1.6. Cargas de imágenes

En este proyecto se ha implementado la carga de imágenes al servidor mediante la utilización del *middleware* Multer para NodeJS. El funcionamiento es sencillo, básicamente se crea un directorio en el servidor donde se almacenan las imágenes de perfil y portada de cada usuario y dron registrado. En la base de datos se almacena la dirección a cada una de esas imágenes.

```
api.post(
  '/upload-user-profile',
  passport.authenticate('jwt', { session: false }),
  userCtrl.uploadUserProfileImage
);
api.post(
  '/upload-user-cover',
  passport.authenticate('jwt', { session: false }),
  userCtrl.uploadUserCoverImage
);
```

Figura 2.36: Rutas de subida de imágenes del Usuario

```
135 //POST - Upload user profile image
136 exports.uploadUserProfileImage = function(req, res) {
137   console.log("POST /api/upload-user-profile");
138
139   var token = getToken(req.headers);
140   var currentUser = jwt.verify(token, 'wordsecret');
141
142   if (token) {
143     const path = './uploads/user/profile/';
144     var uploader = multer({dest: path}).single('profile');
145
146     uploader(req, res, function(err) {
147       if (err) {
148       console.log(err);
149       return res.status(500).send({ msg: 'Ha ocurrido un error al subir la
              imagen de perfil' });
150       }
151
152       console.log('Upload completed to ' + req.file.path);
153       setProfileImageUser(currentUser._id, req.file.filename, res);
154     })
155     } else {
156       return res.status(403).send({success: false, msg: 'No tienes permisos'});
157     }
158 };
```

Figura 2.37: Subida del archivo de la imagen

```
185 //Update profile image from user
186 setProfileImageUser = function(idUser, file, res) {
187   User.findById(idUser, function(err, user) {
188     user.profileImage = file;
189     user.save( function(err) {
190       if(err) {
191         console.log("Error", err);
192         return res.status(500).send({ msg: 'No se ha podido actualizar el
              Usuario' });
193       } else {
194         res.status(200).send({ msg: 'Actualización de imagen de perfil
              completada', filename: file });
195       }
196     });
197   });
198 };
```

Figura 2.38: Actualización en la base de datos de la imagen

2.3.2. *Frontend*

El *frontend* se dedica a visualizar de una forma u otra todo aquello que ha obtenido del servidor. Angular es la tecnología que se encuentra en este apartado.

2.3.2.1. Instalación

El primer paso es instalar la última versión de Angular-CLI utilizando los siguientes comandos:

```
1 $ npm install -g @angular/cli@latest
```

Si la instalación se realiza correctamente, el comando ng -v muestra las versiones que se han instalado.

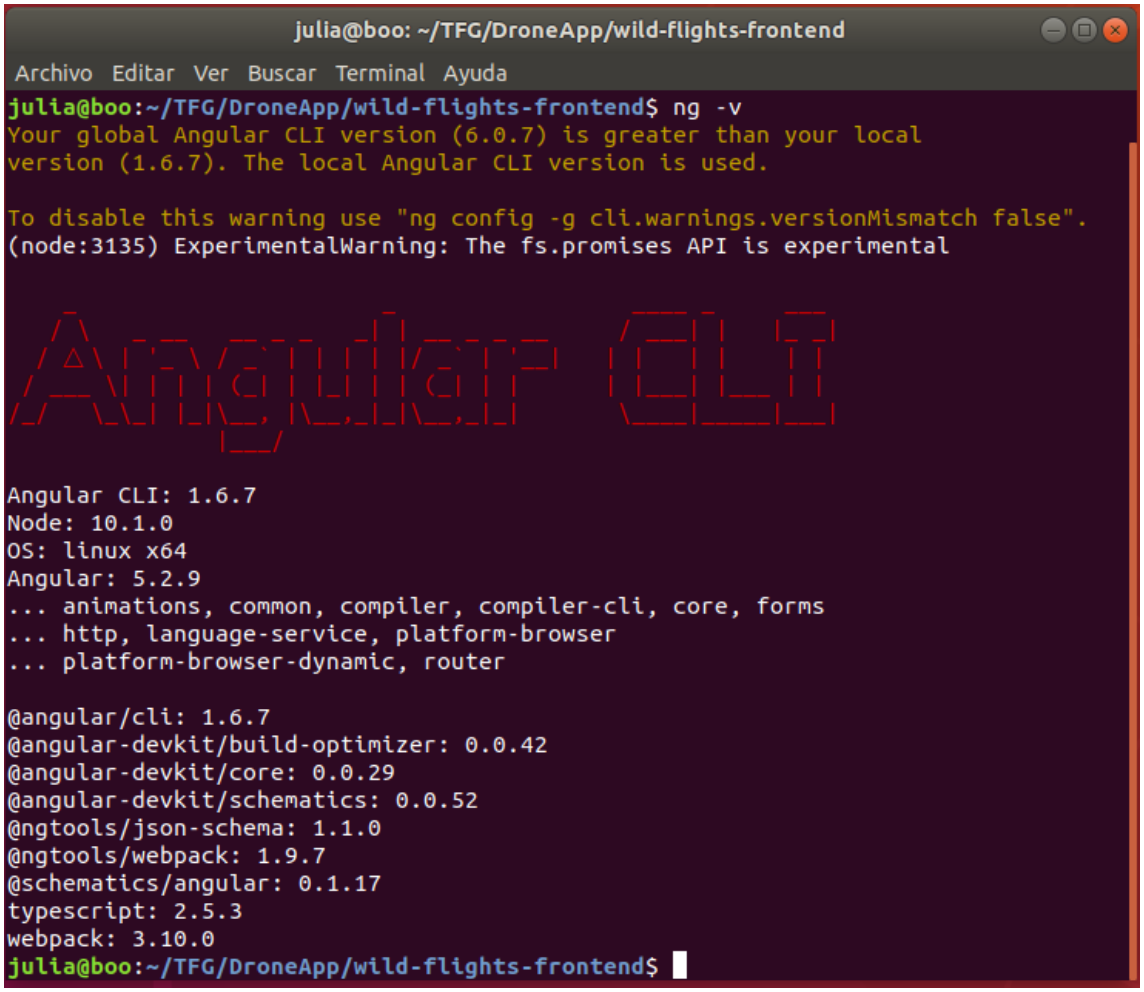

Figura 2.39: Comprobación de la versión de angular

2.3.2.2. Inicio del proyecto

El siguiente paso es generar un nuevo proyecto mediante comandos de Angular-CLI. La siguiente serie de comandos hacen que se genere y se inicialice:

```
1  $ ng new wild-flights-frontend
2  $ cd new wild-flights-frontend
3  $ npm start
```

Como se ha explicado anteriormente, Angular sigue una estructura de MVC Modelo-Vista-Controlador. Y su flujo de datos funciona de la siguiente manera:

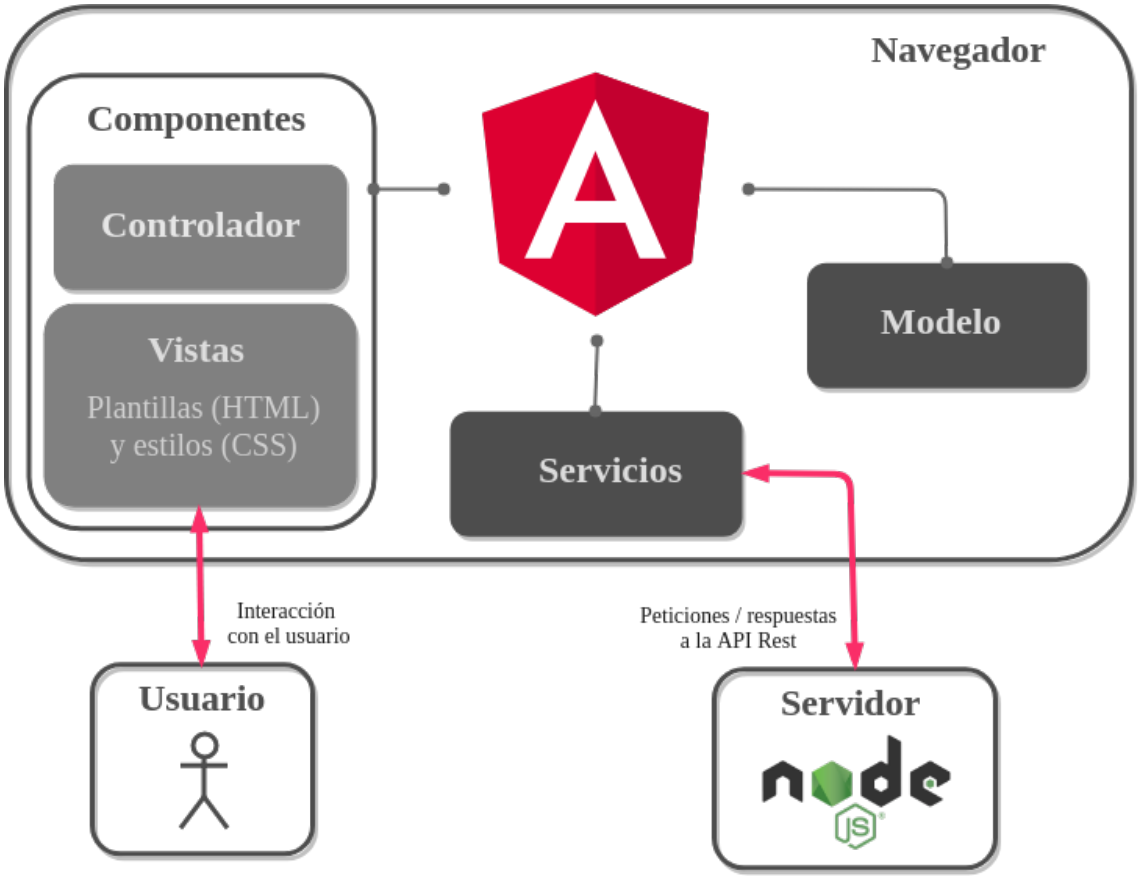

Figura 2.40: Flujo de una estructura MVC en Angular

Esta estructura se consigue con la aparición de los componentes. Estos son los bloques básicos de una aplicación Angular.

Cada componente contiene una parte visual en HTML (la Vista) y una funcional en Typescript (el Controlador).

En este proyecto se genera un componente distinto para cada una de las vistas de las que dispone la aplicación mediante la utilización del siguiente comando:

```
1  $ ng g c "Nombre de cada componente"
```

Un componente no siempre es una vista o página completa, también puede ser un trozo de una de las páginas.

2.3.2.3. Vistas

Las vistas son páginas estáticas que muestran solo la parte visual y que están desarrolladas en HTML.

A la hora de diseñar las vistas, como angular se basa en páginas *SPA*, es decir, que no se cargan las páginas enteras sino sus componentes, ha sido posible generar dos componentes generales que aparecen siempre en todas las páginas de la aplicación. Estos dos componentes son el *Header* y el *Footer*.

Además de generar dichos componentes, también se han creado el resto de componentes referentes a las vistas que se visualizan de forma dinámica según la URL de navegación.

Este proyecto cuenta con varias vistas diferentes las cuáles se muestran a continuación:

Página de inicio Esta vista aparece al iniciar la aplicación, es decir, es la primera toma que tiene un usuario al arrancar la aplicación. Se muestra una imagen de un dron de fondo y un mensaje dirigido directamente al usuario para que se registre en la aplicación.

Figura 2.41: Vista - Inicio

Inicio de sesión La vista de inicio de sesión muestra un formulario con dos campos a rellenar: el *email* del usuario registrado y su contraseña.

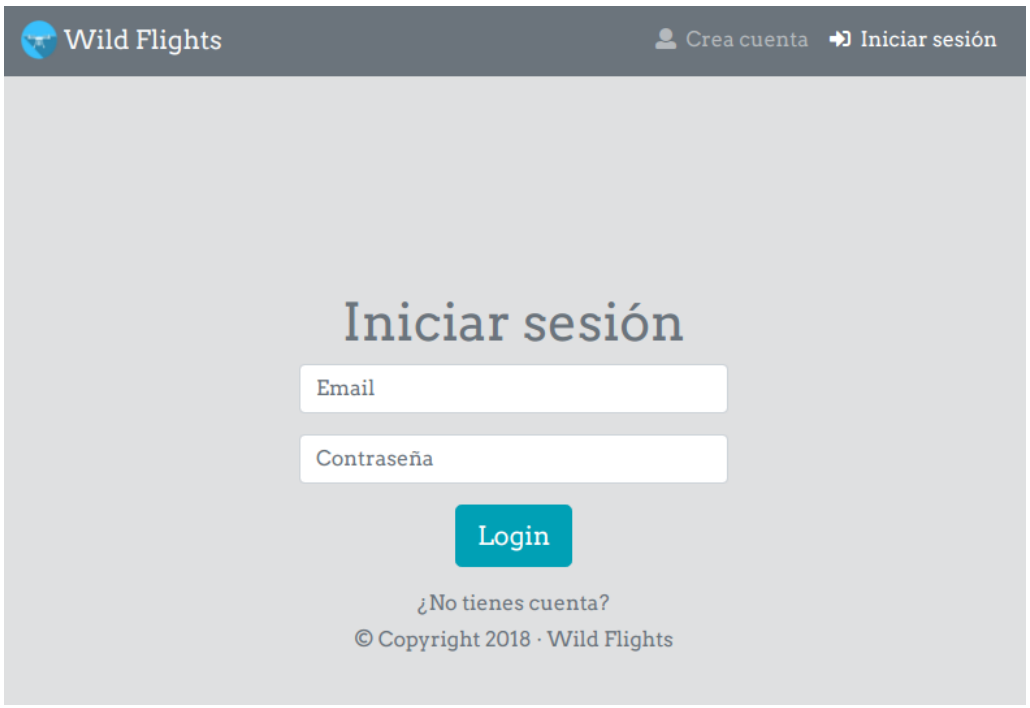

Figura 2.42: Vista - Inicio de Sesión

Registro de usuario La vista de registro de un usuario muestra un formulario con cuatro campos a rellenar: el nombre del usuario a registrar, sus apellidos, su *email* y una contraseña.

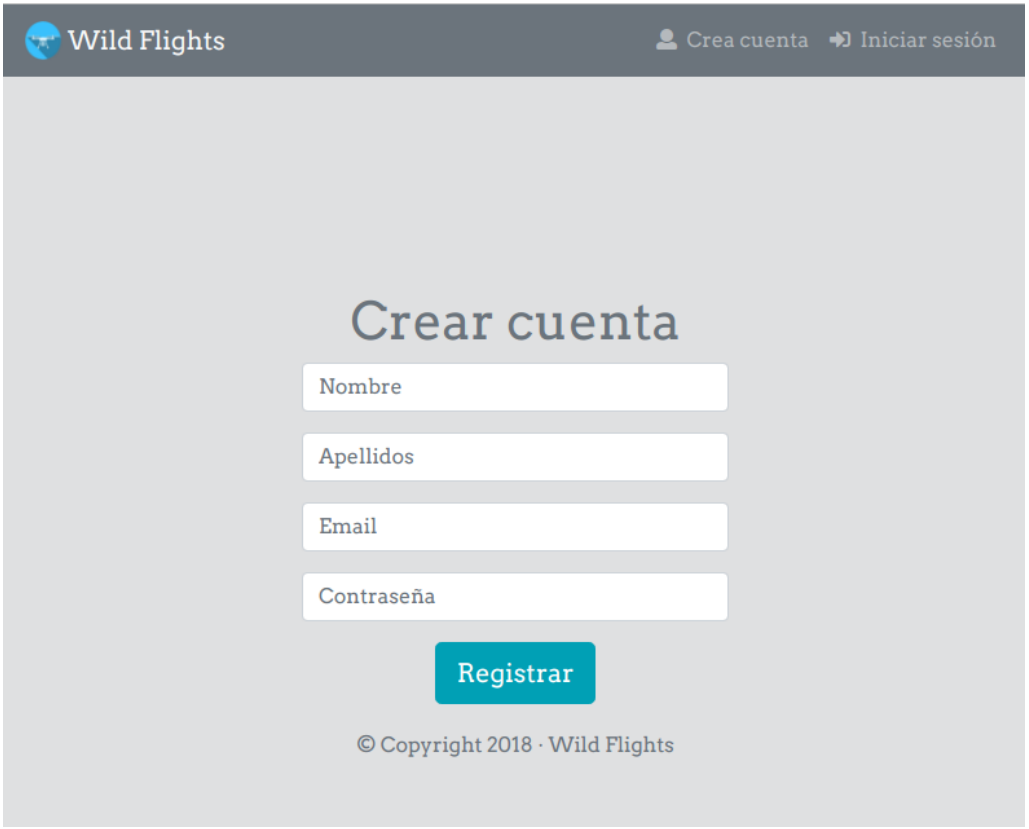

Figura 2.43: Vista - Registro de Usuario

Perfil de usuario Una vez registrado un usuario e iniciado sesión, se muestra la vista del perfil de usuario. Ésta se compone del nombre, imagen de perfil y de portada del usuario. Además, incluye la lista de los drones que ha registrado dicho usuario.

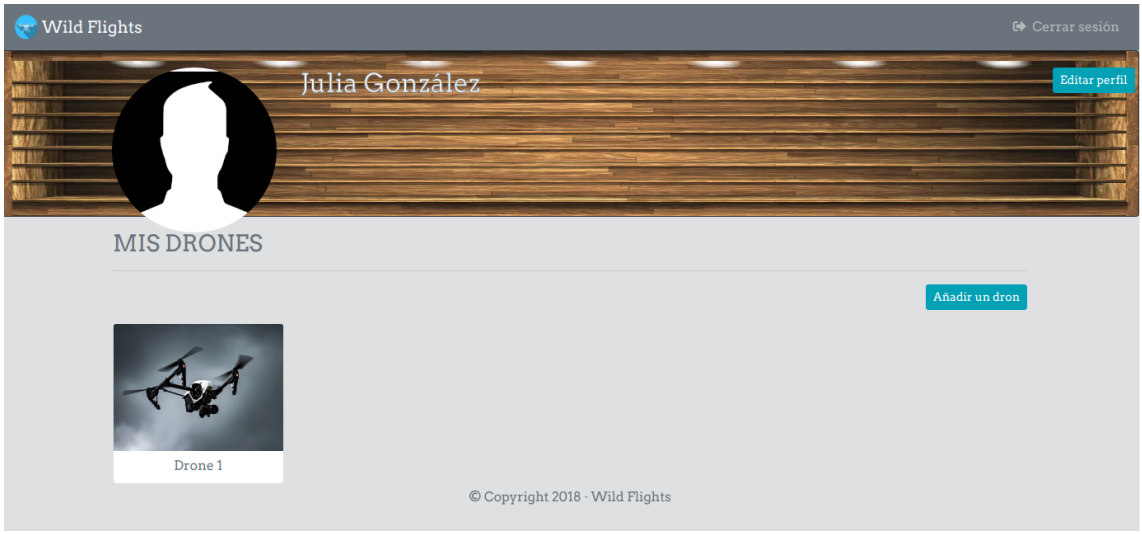

Figura 2.44: Vista - Perfil de Usuario

Perfil de dron Para acceder a la vista del perfil de un dron se debe seleccionar uno de los drones existentes en la lista del perfil del usuario que ha iniciado sesión. Esta vista se compone del nombre, imagen de perfil y de portada del dron seleccionado. También, se incluye una lista de vuelos registrados por dicho dron y un resumen de la distancia, duración, velocidad media y altitud media total registrados por el dron hasta ese preciso momento.

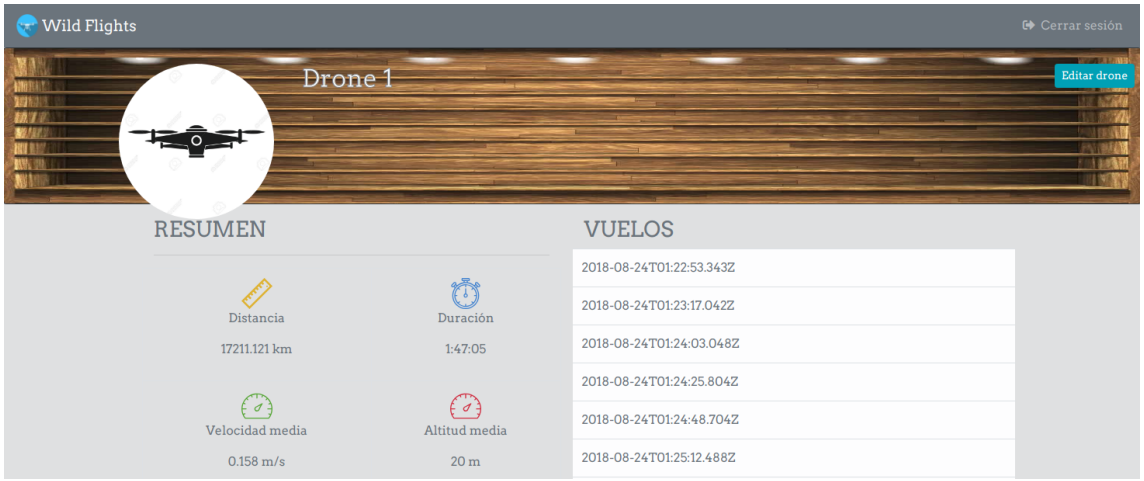

Figura 2.45: Vista - Perfil de Dron

Perfil de vuelo La forma de acceder al perfil de un vuelo es mediante la selección de uno de ellos en la lista de vuelos que aparece en el perfil de un dron. En esta vista se muestra un resumen de la distancia, duración, velocidad media y altitud media del registro del vuelo. Además, es posible visualizar la trayectoria realizada que aparece pintada sobre un mapa. También, aparecen dos gráficas indicando la velocidad y la altitud máxima en cada instante.

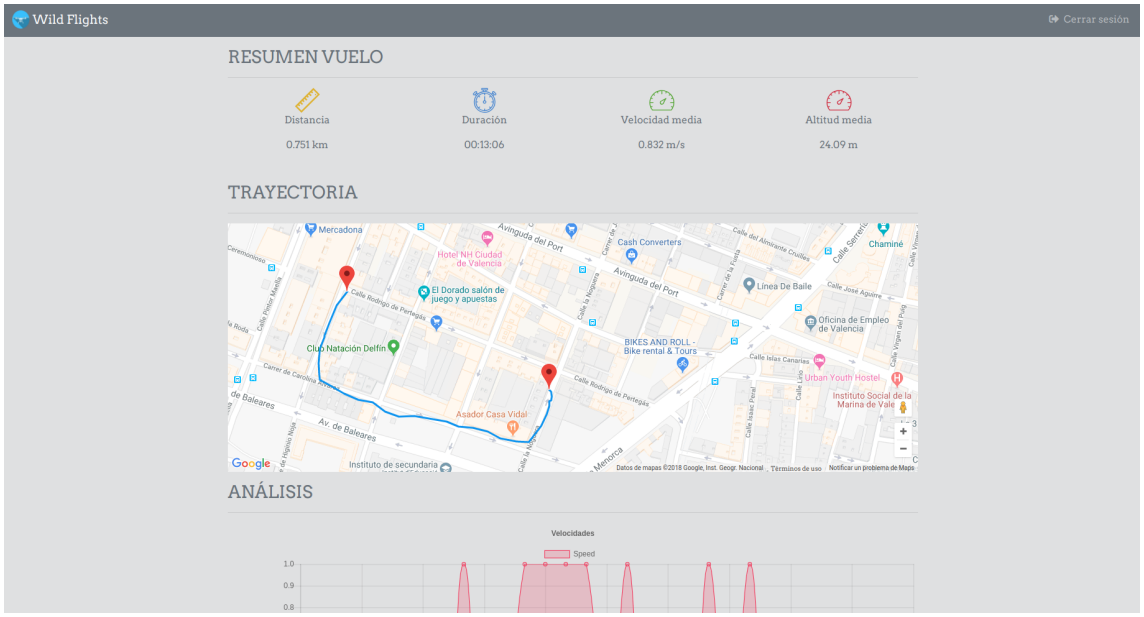

Figura 2.46: Vista - Perfil de Vuelo

Editar usuario El acceso a esta vista se realiza pulsando sobre el botón Editar Usuario que aparece en la página del perfil de usuario. Esta vista contiene el mismo formulario

que a la hora de registrar el usuario pero, en este caso, se añaden dos campos que sirven para modificar las imágenes de perfil y de portada de dicho usuario.

Figura 2.47: Vista - Editar Usuario

Editar dron El acceso a esta vista se realiza pulsando sobre el botón Editar Dron que aparece en la página del perfil de dron. Esta vista contiene el mismo formulario que a la hora de registrar un dron pero, en este caso, se añaden dos campos que sirven para modificar las imágenes de perfil y de portada de dicho dron.

Figura 2.48: Vista - Editar Dron

2.3.2.4. Controladores

Los controladores son los encargados de convertir una página estática en dinámica, es decir, se encargan de la funcionalidad en los componentes. Son los que realizan peticiones

para obtener la información de los servidores que se quiera mostrar al cliente o para enviar datos que ha introducido el cliente al servidor. Toda petición pasa antes por un servicio desarrollado en el *frontend* que se encarga de realizar la conexión directa con el servidor.

Todo controlador tiene una base inicial formada por tres grandes bloques (importaciones, decorador y una clase) como la que se muestra a continuación:

```
import { Component, OnInit } from '@angular/core';
import { DroneService } from '../services/drone.service';
import { AuthService } from '../services/auth.service';
import { UserService } from '../services/user.service';
import { Router } from '@angular/router';

@Component({
  selector: 'app-user-profile',
  templateUrl: './user-profile.component.html',
  styleUrls: ['./user-profile.component.css']
})
export class UserProfileComponent implements OnInit {
  drones: any;
  user: any;

  constructor(
    private router: Router,
    private authService: AuthService,
    private userService: UserService,
    private droneService: DroneService
  ) { }
```

Figura 2.49: Inicio del controlador básico en un componente

En los controladores se inyectan servicios para contener lógica de negocio, clases para el acceso a datos o utilidades de infraestructura. En la figura 2.49 se observa la inyección de los servicios AuthService, UserService y DroneService que utilizamos en el método ngOnInit del mismo controlador. Encontramos un ejemplo en la figura 2.50 donde se utilizan los servicios anteriormente comentados para obtener el usuario autenticado y sus drones.

```
23  ngOnInit() {
24    if (!this.authService.getToken()) {
25      this.router.navigate(['login']);
26    }
27
28    // Obtener Usuario
29    this.userService.getUser().subscribe(
30      data => {
31        console.log('GET response', data);
32        this.user = data['user'];
33        this.userService.getUrlImageUser(this.user);
34      }, err => {
35        if (err.status === 401) {
36          console.log('Fired redirect');
37          this.router.navigate(['login']);
38        }
39      }
40    );
41
42    // Obtener Drones
43    this.droneService.getAll().subscribe(data => {
44      this.drones = data;
45      console.log('Drones', this.drones);
46    }, err => {
47      if (err.status === 401) {
48        console.log('Fired redirect');
49        this.router.navigate(['login']);
50      }
51    });
52  }
```

Figura 2.50: Utilización de servicios en el controlador del perfil de usuario

Una forma de generar un servicio sería mediante la utilización del siguiente comando ofrecido por Angular-CLI:

```
1  $ ng g s "Nombre del servicio"
```

En la parte del *frontend* se utiliza un módulo llamado ng2-file-upload para subir las imágenes y almacenarlas en el servidor.

Para notificar al cliente sobre el éxito o fallo de las peticiones realizadas al servidor se ha utilizado la librería o módulo angular5-toaster que muestra unos paneles en la esquina superior derecha con la información mostrada.

CAPÍTULO 3
Bloque II: Monitorización

En esta sección se explica todo aquello relacionado con la monitorización de la aplicación web ya comentada anteriormente.

3.1 Tecnologías utilizadas

Este apartado describe brevemente todas las tecnologías utilizadas en la parte de la monitorización de la aplicación implementada.

3.1.1. Arduino

Arduino es una plataforma de electrónica de código abierto formada tanto por dispositivos *hardware* como por un *software* específico. Además, y no menos importante, existe una gran comunidad y filosofía alrededor de la plataforma para compartir y crear librerías, proyectos y mejores prácticas.

Figura 3.1: Arduino logo

Como se ha indicado, existe un *software* específico de desarrollo integrado llamado *software* Arduino (IDE) el cual se utiliza para implementar y subir los programas al *hardware*. Los programas o fragmentos de código se llaman bocetos o *sketches*, y tienen la extensión de archivo INO.

Ésta herramienta dispone de diversos componentes que nos hace más fácil la tarea de desarrollo. Entre otros, ofrece un editor de texto para escribir el código, junto a una herramienta de compilación y subida del boceto la cual muestra en una área de mensajes si existen errores de compilación, de sintaxis o conexión con la placa. También ofrece una herramienta de monitor serie, que sirve para loguear errores o información directamente desde la placa o microcontrolador. Contiene una herramienta de instalación de librerías de terceros, la cual permite instalar y añadir librerías de terceros directamente en el boceto desde un buscador.

La utilización del Arduino IDE, permite abstraer el desarrollo *software* del *hardware* final, ya que es la misma herramienta el que se encarga de compilar el código y subirlo al dispositivo seleccionado. Un mismo boceto podría servir para diferentes placas, siempre que éstas sean compatibles en funcionalidad y pines (la mayoría de veces se pueden realizar adaptaciones con pocas modificaciones).

Hay diferentes placas o diseños de *hardware* (archivos CAD) disponibles bajo licencia *open-source Creative Commons*, las cuales se pueden modificar o adaptar según las necesidades.

Se describe en concreto el modelo de placa Arduino UNO Rev3, que es la utilizada durante el proyecto, por tratarse de una placa económica y que cumple los requisitos y necesidades del mismo.

Las principales características que nos ofrece el Arduino UNO Rev3:

- Microcontrolador: ATmega328.
- Tensión operativa: 5V.
- Tensión de alimentación: 7-12V.
- Máxima VC para entradas: 40mA.
- Máxima VC para pines de 3.3V: 50mA.
- Entradas/salidas digitales: 14. Seis de ellas pueden actuar como analógicas.
- Entradas/salidas analógicas (serigrafía de a0 a a5): 6.
- Pines de alimentación: uno de 5V y otro de 3.3V.
- Pines de toma tierra: 3.
- Memoria *flash*: 32 Kb.
- SRAM: 2 Kb.
- EEPROM: 1Kb.

Además del dispositivo programable, se utilizan una serie de dispositivos que transforman magnitudes físicas o químicas en señales eléctricas que son interpretadas por el mismo. Éstos son los sensores. Durante el proyecto se utiliza un módulo GPS, del cual hablamos más adelante, el cual obtiene y transmite la geolocalización al Arduino.

Existen un gran abanico de actuadores, los cuales nos permiten interactuar con el entorno, y también, módulos especializados, los cuales nos permite recabar información y comunicar el Arduino a través de diferentes tecnologías con otros dispositivos (*ethernet*, *bluetooth*, *wifi*, *3G*...).

3.1.2. Módulo GPS

El módulo GPS está basado en el *chip* receptor NEO 6M y sirve perfectamente para controlarlo con una placa Arduino o con cualquier otro microcontrolador.

El GPS incluye una antena de Cerámica preparada para instalarse directamente en la placa base. La placa base viene provista de conectores para la alimentación y la transmisión de datos (Vcc, Tx, Rx y Gnd). Además, también dispone de un led que se ilumina de color azul cuando está conectado el módulo y emitiendo datos.

El voltaje de alimentación mínimo que necesita son 3V y no puede exceder de los 5V. Como la mayoría de módulos GPS, utiliza un protocolo NMEA (*National Marine electronics Asociation*) ya que es un protocolo estándar en la recepción de datos por GPS.

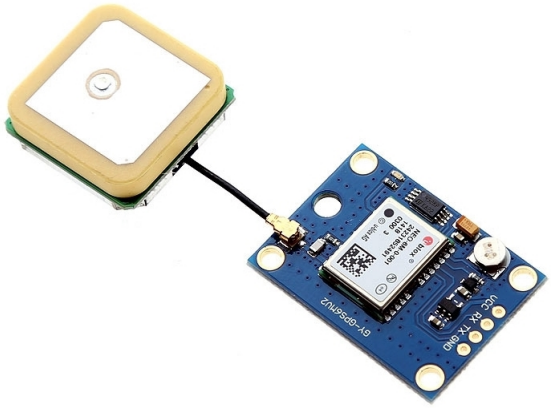

Figura 3.2: Módulo GPS NEO6M

3.1.3. Módulo SIM900

Se trata de un módulo que permite hacer y recibir llamadas, hacer y recibir mensajes de textos, y conectar a Internet y hacer peticiones HTTP como si de un teléfono móvil se tratara.

El módulo GSM/GPRS se puede decir que tiene dos componentes principales. En primer lugar está basado en un *chip* de SIMCom, el SIM900, es un módulo completo GSM/GPRS cuatribanda 2G con un solo núcleo ARM926EJ-S, de pequeñas dimensiones y precio económico. Soporta protocolo TCP/UDP y FTP/HTTP entre otros.[13]

Y por otro lado, el *chip* va montado sobre una placa que proporciona diferentes interfaces de conexión adaptadas al desarrollo con Arduino. Entre otros mecanismos que ofrece la placa, se pueden ver las siguientes características:

- Conexiones de audio y micrófono.
- Posibilidad de alimentación externa, y configuración mediante interruptor. La corriente recomendada es de 9V a 2A, pero puede soportar desde un rango de 5V a 26V a 2A. En el caso de una alimentación inferior a 9V es posible que no funcione correctamente.
- Configuración mediante *jumpers* de la comunicación con el arduino. La velocidad de comunicación por defecto en baudios es de 19200 bps.
- Antena integrada.
- Ranura para tarjeta SIM (parte inferior de la placa).
- Botón de encendido y *reset*. Además incorpora luces de control de encendido, y de correcto funcionamiento.
- Reloj y ranura para una pila para su funcionamiento.
- Conectores para los pines de Arduino UNO (También compatible con Arduino MEGA), los cuales sirven para montar la tarjeta del módulo SIM900 sobre el Arduino.

Para el desarrollo es necesario una SIM accesible, y una conexión serie con un Arduino. Éste se basa en el envío de una secuencia de comandos AT hacia el módulo con funciones concretas y éste actúa en consecuencia.[12]

Figura 3.3: Módulo SIM900

3.1.3.1. ¿Por qué 3G y no wifi?

Figura 3.4: Comparativa entre 3G y Wifi

Este proyecto está pensado sobre todo para dispositivos que se alejan considerablemente de su base sin que la conexión se interrumpa. Por tanto, si se hubiera utilizado Wifi en vez de 3G, el rango por el cual podría moverse el dispositivo sería limitado a la zona con conexión proporcionada por el punto de acceso o repetidores.

Utilizando un módulo 3G, se puede conectar con servidor web remoto sin depender de un punto de acceso, proporcionando al dispositivo de independencia y flexibilidad en el movimiento.

3.2 Análisis

Las actividades de análisis y diseño ayudan a transformar los requerimientos previos en un diseño implementable en *software*.En este apartado se analizan solo los requerimientos funcionales del sistema, ignorando las restricciones de la arquitectura del sistema. El objetivo del análisis es que todos los requisitos existentes en la aplicación a desarrollar queden contemplados en el diagrama de casos de uso.

3.2.1. Requisitos

Como bien se explica en el bloque anterior el análisis de requisitos es uno de los apartados más importantes a la hora de implementar cualquier aplicación. El principal

objetivo de la monitorización es almacenar todos los datos obtenidos por el GPS en la base de datos. Todo el contenido de la página web se almacena en una base de datos no relacional. En este apartado detallaremos brevemente los requisitos funcionales de la monitorización de la aplicación:

Título	Conexión del módulo SIM900
Propósito	Conectar la placa arduino con el módulo SIM900
Entrada	Peticiones por el monitor serial abierto en el puerto 19200 al módulo SIM900 conectado mediante los pines 7 y 8.
Proceso	Se deben realizar varias peticiones de conexión mediante comandos AT. Comprobar que existe conexión, indicar el pin de la tarjeta sim y indicar el apn que utiliza dicha tarjeta.
Salida	Si la conexión ha sido satisfactoria, cada petición realizada por la placa arduino sobre el módulo SIM900 debe de haber devuelto un string “OK” por el monitor serie.

Figura 3.5: Requisito - Conexión del módulo SIM900

Título	Obtener datos del GPS
Propósito	Obtener los datos que está capturando el módulo GPS.
Entrada	Peticiones por el monitor serial abierto en el puerto 9600 al módulo SIM900 conectado mediante los pines 3 y 4.
Proceso	Se recogen los datos obtenidos del identificador de vuelo, la fecha, longitud, latitud, altitud y velocidad.
Salida	Cuando se han obtenido todos los datos nombrados, se devuelve un objeto con dichos datos.

Figura 3.6: Requisito - Obtener datos del GPS

Título	Creación de un punto
Propósito	Almacenar un punto en la base de datos que se ha creado con los datos obtenidos por el GPS.
Entrada	Utiliza el módulo SIM900.
Proceso	Cada vez que se obtienen los datos del GPS, se realiza una petición post HTTP a la url de puntos que existe en la API/Servidor para que éste se encargue de guardar dicho punto en la base de datos.
Salida	Si se guarda el punto en la base de datos correctamente, el servidor devuelve un estado 200, sino devuelve un 404 para indicar que no se ha podido guardar el punto en la base de datos por algún error en el camino.

Figura 3.7: Requisito - Creación de un punto

Título	Creación de un vuelo
Propósito	Almacenar un vuelo en la base de datos cada vez que se enciende el dispositivo.
Entrada	Utiliza el módulo SIM900.
Proceso	Cada vez que se enciende el dispositivo SIM900, se crea un vuelo al que corresponden los siguientes puntos que se vayan a recoger hasta que se apague el dispositivo. Se realiza una petición HTTP a la url de vuelos que existe en la API/servidor para que éste luego añada el vuelo en la base de datos.
Salida	Si el vuelo se ha creado correctamente, se devuelve un estado 200. Y acto seguido se obtiene el identificador en la base de datos que tiene el vuelo recién creado y se guarda en una variable global. Si ha ocurrido algún error en el camino, se devuelve un estado 404 para indicar que no se ha podido guardar el punto en la base de datos.

Figura 3.8: Requisito - Creación de un vuelo

3.2.2. Diagrama de contexto

Como bien se ha explicado anteriormente el diagrama de contexto sirve para definir desde un nivel general el contexto del sistema.

En este caso se detalla el diagrama de la monitorización de la aplicación general:

Figura 3.9: Diagrama de contexto de la monitorización

3.2.3. Casos de uso

Este apartado define un diagrama de casos de usos que define gráficamente el comportamiento UML de forma mejorada.

A continuación se muestra el diagrama de casos de uso de la monitorización de la aplicación que se va a implementar:

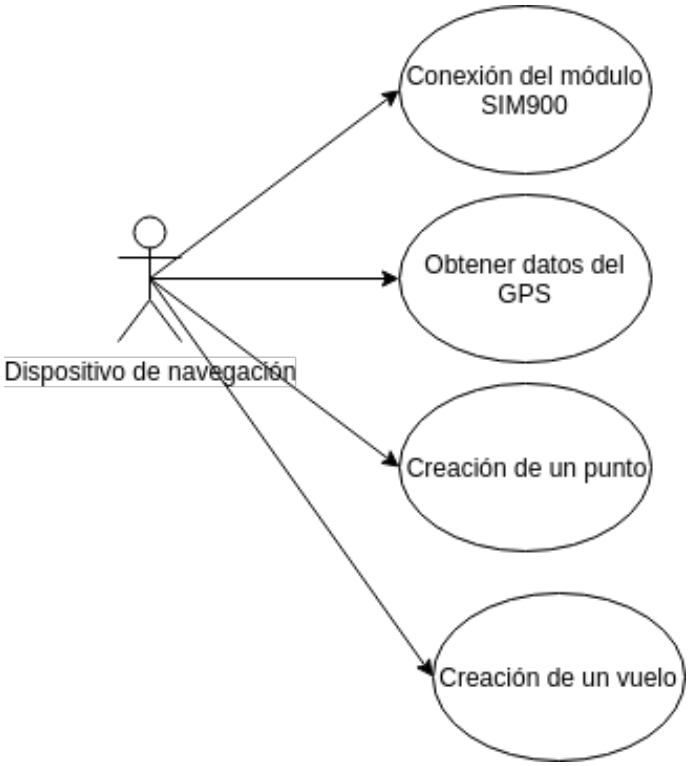

Figura 3.10: Casos de uso de la monitorización

3.3 Desarrollo

Para poder empezar a desarrollar esta parte del proyecto es necesario tener instalado el *software* Arduino o IDE de Arduino ya que permite escribir y subir programas al microcontrolador Arduino UNO. Para instalarlo se ha seguido la guía de instalación de la página oficial de Arduino.[3]

Después de instalar el IDE de Arduino el siguiente paso a seguir es estructurar y montar las conexiones entre los tres componentes *hardware*. Se puede ver la estructura resultado en el esquema de la figura siguiente:

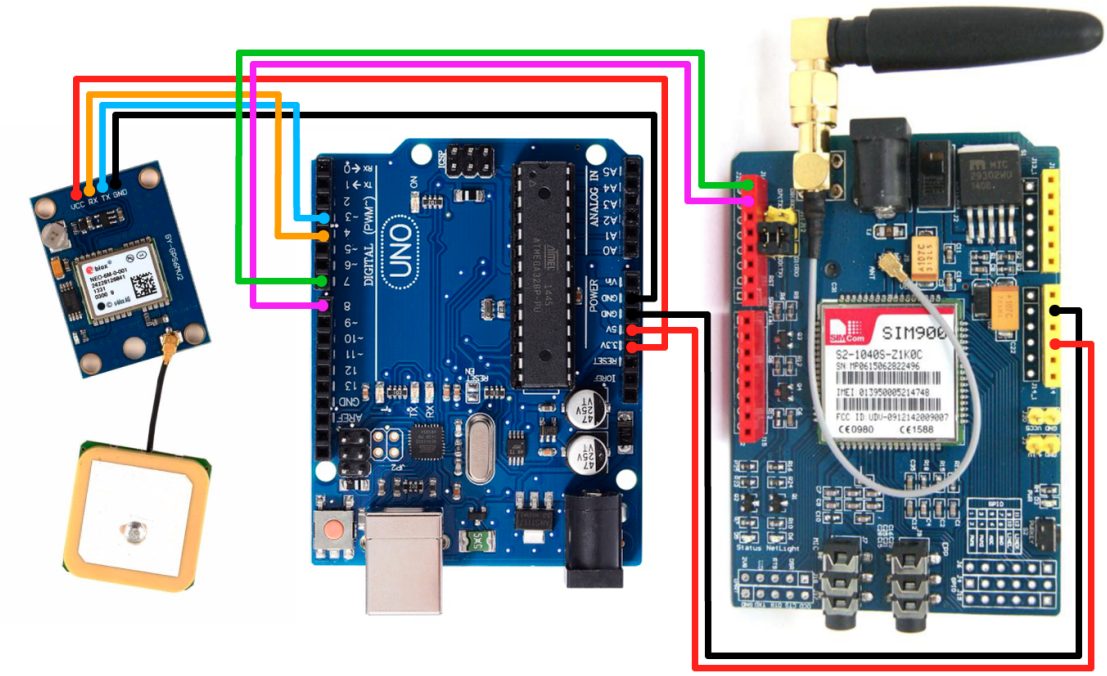

Figura 3.11: Esquema de las conexiones

El GPS se alimenta a través de la salida de 3.3V que nos proporciona el Arduino, y en el caso de el módulo SIM900 y el microcontrolador Arduino UNO, se utilizan fuentes externas, en concreto dos pilas de 9V (aceptan voltajes entre los 7 y 12 V) conectadas mediante un adaptador (Ver 3.12). Para poder alimentar el SIM900 a través de una fuente externa, hay que posicionar correctamente el interruptor hacia el interior del módulo (lo encontramos al lado de la conexión o *jack* de alimentación de fuente externa). Hay que tener en cuenta que el módulo GPS no debe alimentarse a más de 5V y el módulo SIM900 y Arduino a más de 12V.

Figura 3.12: Conector de la batería

Una vez realizado el montaje y el cableado, se pasa a programar el microcontrolador para que realice las funciones oportunas.

3.3.1. Inicialización

En primer lugar, escribimos la importación de librerías. Será necesario la creación de las dos comunicaciones entre el microcontrolador Arduino y los dos módulos, para ello, se utiliza la librería de SoftwareSerial, la cual permite crear comunicaciones serie indicando dos pines del Arduino. Se crea una conexión en los pines 7 y 8 con el módulo SIM900 y otra en los pines 3 y 4 con el GPS.

También, se utiliza una librería de Arduino llamada TinyGPS que ayuda a la ordenación y comprensión de los datos obtenidos mediante el GPS, ya que como se ha comentado anteriormente, el módulo GPS muestra los datos siguiendo el protocolo NMEA y que no es fácil de descifrar.

Además, se añaden un par de variables globales. Por un lado, una constante que se utilizará en la petición inicial para que el servidor pueda identificar correctamente al dispositivo de navegación. Y por otro lado, una variable para poder identificar correctamente el vuelo en el envío de los puntos, la cual se rellenará después de crear el vuelo.

```
1  #include<SoftwareSerial.h>
2  #include <TinyGPS.h> //incluimos TinyGPS
3
4  TinyGPS gps; //Declaramos el objeto gps
5
6  //Declaramos el pin 7 Tx y 8 Rx con el SIM900
7  SoftwareSerial client(7,8);
8
9  //Declaramos el pin 4 Tx y 3 Rx con el GPS
10 SoftwareSerial serialgps(4,3);
11
12 //Se declaran variables globales
13 static String ID_DRONE = "5b4399c9ad6996116dd93314";
14 String idFlight = "";
```

Figura 3.13: Definición de una operación SOAP para recuperar etiqueta de envío

3.3.2. Configuración

Se pasa a la parte de la configuración de las distintas conexiones. Iniciamos las comunicaciones seriales con los dos módulos y el monitor serie del PC para poder testearlo y monitorizarlo a través del Arduino IDE en caso de ser necesario.

Una vez creada la comunicación serial del SIM900, se inicia la configuración del mismo. Ésta se realiza mediante comandos AT seguidos de intros. Se indica, entre otros, el número PIN de la tarjeta SIM introducida en el módulo SIM900, el APN de la misma y el tipo de conexión que se va a utilizar.

3.3.3. Inicio de vuelo

Una vez configurado el sistema, se pasa a implementar la funcionalidad del dispositivo de navegación.

Para poder enviar los datos de navegación, se necesita identificar correctamente el vuelo del que forman parte. Por tanto, se necesita primero conectar con el servidor y crear el nuevo vuelo.

Se manda al servidor una petición HTTP de método *POST*, la cual contiene un JSON donde se indica el identificador del dispositivo de navegación. El servidor, si todo ha ido correctamente, contesta con un código 200, junto a un mensaje con el identificador del vuelo que acaba de crear. Mediante una función que parsea la respuesta, se guarda el valor del identificador en la variable global correspondiente, permitiendo así continuar la ejecución y pasar al siguiente punto, el cual consiste en enviar cada cierto intervalo de tiempo puntos con datos de geolocalización y de estado.

Como se puede observar en la siguiente figura, después del envío de cada instrucción de comandos AT hacia el dispositivo SIM900, se debe de esperar cierto margen de tiempo para que el mensaje se procese correctamente y se transmita entero.

```
155 void createFlight(){
156   client.listen();
157   client.println("AT+HTTPINIT");
158   delay(200);
159   client.println("AT+HTTPPARA=\"CID\",1");
160   delay(200);
161   client.println("AT+HTTPPARA=\"URL\",\"http://leafdevelop.ddns.
        net:3002/api/flights\"");//Public server IP address
162   delay(200);
163   client.println("AT+HTTPPARA=\"CONTENT\",\"application/json\"")
        ;
164   delay(200);
165
166   // Enviar identificador del drone
167   String data = "{\"idDrone\": \"" + ID_DRONE + "\"}";
168   client.println("AT+HTTPDATA=" + String(data.length()) + "
        ,100000");
169   delay(500);
170
171   client.println(data);
172   delay(1500);
173   client.println("AT+HTTPACTION=1");
174   delay(200);
175
176   // Leer datos recibidos
177   client.println("AT+HTTPREAD");
178   delay(600);
179   String res = "";
180   while(client.available() > 0){
181    res += (char)client.read();
182   }
183   idFlight = getValue(res, '"', 1);
184   client.listen();
185   client.println("AT+HTTPTERM");
186   delay(200);
187 }
```

Figura 3.14: Creación de un vuelo

3.3.4. Envío de puntos

Por último, pasamos a la obtención de coordenadas y datos de navegación proporcionados por el del módulo GPS y su envío al servidor.

Como podemos ver en la figura, una vez se ha obtenido el identificador de vuelo, procedemos a obtener y enviar datos de navegación.

```
27 void loop()
28 {
29   // Primero creamos el vuelo
30   if(idFlight == ""){
31     Serial.println("---createFlight");
32     createFlight();
33   }
34   // Una vez creado el vuelo, añadimos los puntos
35   else {
36     Serial.println("---createPoint");
37     Serial.print("idFlight: <");
38     Serial.print(idFlight);
39     Serial.println(">");
40     createPoint();
41   }
42 }
```

Figura 3.15: Función del bucle

La función de que se ha llamado creación de nuevo punto o *createPoint* es muy similar a la de creación de vuelo con la particularidad de que en vez de mandar los datos al servidor contenidos en una variable, éstos se obtienen del módulo GPS.

Por tanto, una vez lanzado la creación y preparada el dominio y URL destino, se llama a la función de obtención de los datos de geolocalización. Esto se realiza a través del monitor serial establecido con el módulo GPS, los cuales se mandan hacia el Arduino cada cierto intervalo de tiempo. Se almacenan, y se genera una cadena con formato JSON para poder mandarlo al servidor.

Podemos ver la función en cuestión y un ejemplo del JSON generado en las figuras siguientes.

```
/**
 * Genera un string JSON con los valores obtenidos
 * a través del módulo GPS.
 * @return String
 */
String getGPSData() {
  serialgps.listen();
  String data = "";
  while (data == "") {
    while(serialgps.available()) {
      int c = serialgps.read();
      if(gps.encode(c))
      {
        float latitude, longitude, altitude;
        unsigned long speed;
        int year;
        byte month, day, hour, minute, second, hundredths;
        unsigned long chars;
        unsigned short sentences, failed_checksum;

        gps.f_get_position(&latitude, &longitude);

        gps.crack_datetime(&year,&month,&day,&hour,&minute,&
           second,&hundredths);
        String pointDatetime = String(year) + "-" + printDigits(
           month) + "-" + printDigits(day) + "T" + printDigits(
           hour) + ":" + printDigits(minute) + ":" + printDigits
           (second) + "Z";

        speed = gps.f_speed_mps();
        altitude = gps.f_altitude();
        data = "{\"idFlight\": \"" + String(idFlight) + "\", \"
           latitude\": \"" + String(latitude, 10) + "\", \"
           longitude\": \"" + String(longitude, 10) + "\", \"
           datetime\": \"" + pointDatetime + "\", \"speed\": \""
            + String(speed,3) + "\", \"altitude\": \"" + String(
           altitude,5) + "\"}";

        gps.stats(&chars, &sentences, &failed_checksum);
      }
    }
  }
  return data;
```

Figura 3.16: Función para obtener los datos del módulo GPS

CAPÍTULO 4
Implantación

Este capítulo se basa en detallar el proceso de implantación de la aplicación en la red. Como bien se ha comentado en apartados anteriores, la aplicación cuenta con dos partes, *frontend* y *backend*, debido a ésto el proceso de despliegue es distinto en cada caso.

4.1 *Backend*

En el *backend*, se ha utilizado la tecnología NodeJS. Se ejecutará un proceso node el cual, utilizando Express, está a la espera de peticiones realizadas por el cliente o el dispositivo de navegación.

La figura 4.1 muestra la creación y configuración de dicho servidor.

```
var mongoose   = require ('mongoose'),
    configDB   = require('./config/database.js'),
    port       = process.env.PORT || 3002,
    app        = require('./app');

// Conexión a la base de datos
mongoose.connect(configDB.url, function(err) {
  if(err) throw err;
  console.log('Connected to Database');
});

mongoose.promise = Promise;

// Arranca el servidor
app.listen(port, function() {
  console.log("Node server running on http://localhost: "+port);
});
```

Figura 4.1: Creación del servidor

Al lanzar la aplicación, se inicializa un servidor que escucha en el puerto 3002, además se realiza una conexión a la base de datos.

4.2 *Frontend*

Para servir los documentos y archivos del *frontend*, antes de poder desplegarlos, se necesita compilar el códio TypeScript a código HTML, CSS y JavaScript.

Para ello, utilizando la interfaz de línea de comandos Angular-CLI, se ejecuta:

```
$ ng build --prod
```

Se genera un directorio llamado dist, el cual contiene los archivos necesarios ya compilados para desplegarlos en un servidor web, como Apache o NGINX, y que éste los sirva a los clientes.

4.3 Preparación del servidor

Para servir los recursos a la red, se utiliza una máquina que pertenece a una red local, y por tanto, no cuenta con una IP pública. Por tanto, para que ésta sea accesible desde el exterior, es necesario configurar el router de acceso a Internet y realizar una redirección de puertos.

En este caso se trata de un router de la empresa Vodafone y se ha podido acceder desde el navegador introduciendo la IP 192.168.0.1, es decir, indicando la IP identificada como Puerta de enlace predeterminada del servidor.

La figura 4.2 muestra la página inicial del *router* en cuestión.

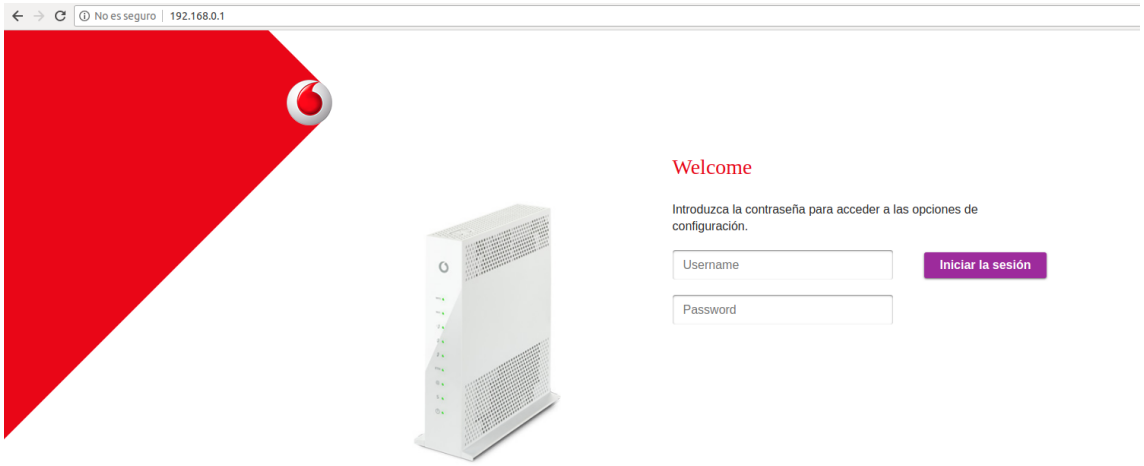

Figura 4.2: Pantalla de inicio del router Vodafone

Buscando entre la configuración del *router*, se ha llegado a la opción de redireccionamiento de puertos.

Dado que el servidor de node está lanzado desde una máquina con una IP privada determinada y escuchando en un puerto concreto, el paso a seguir ha sido añadir una redirección de puertos que provoca que todas las peticiones que llegan al *router* del exterior de la red al puerto público 3002 se redireccionen a la dirección IP del servidor al puerto 3002.

En la figura 4.3 se puede observar dicha configuración:

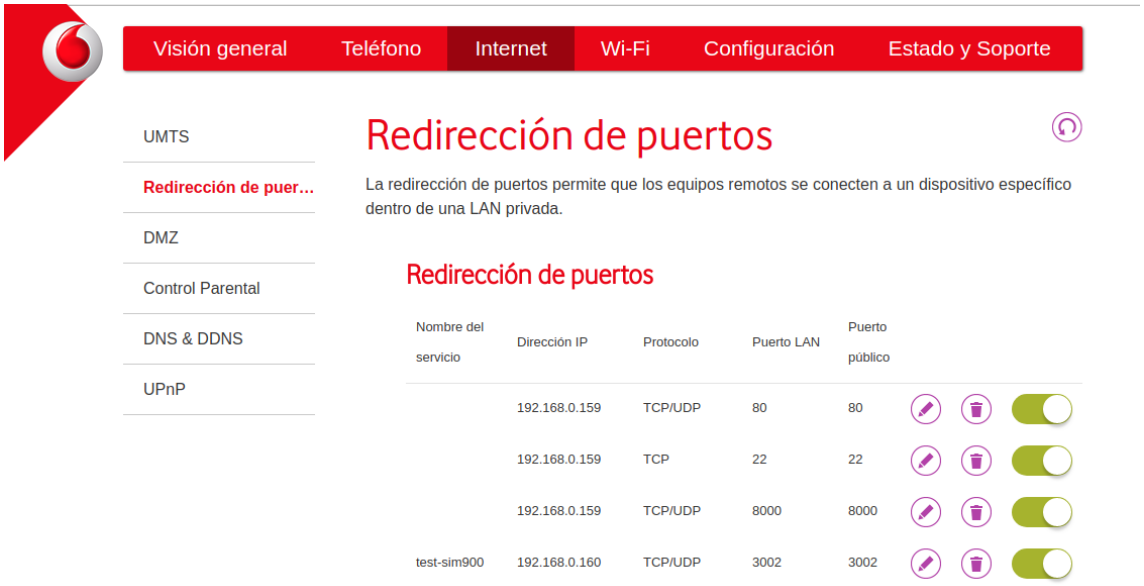

Figura 4.3: Redirección de puertos en el *router* Vodafone

Del mismo modo, es necesario una redirección de el puerto público 3003 al puerto 80 de la máquina que hace de servidor, donde se encuentra escuchando el servidor web HTTP.

CAPÍTULO 5
Pruebas

Este capítulo muestra una serie de capturas realizadas a varias pruebas funcionales de la aplicación implementada.

Las figuras siguientes muestran el proceso que realiza la edición de la imagen de perfil de un usuario:

Figura 5.1: Captura antes de editar imagen del perfil

Figura 5.2: Captura en el momento de la edición de la imagen del perfil

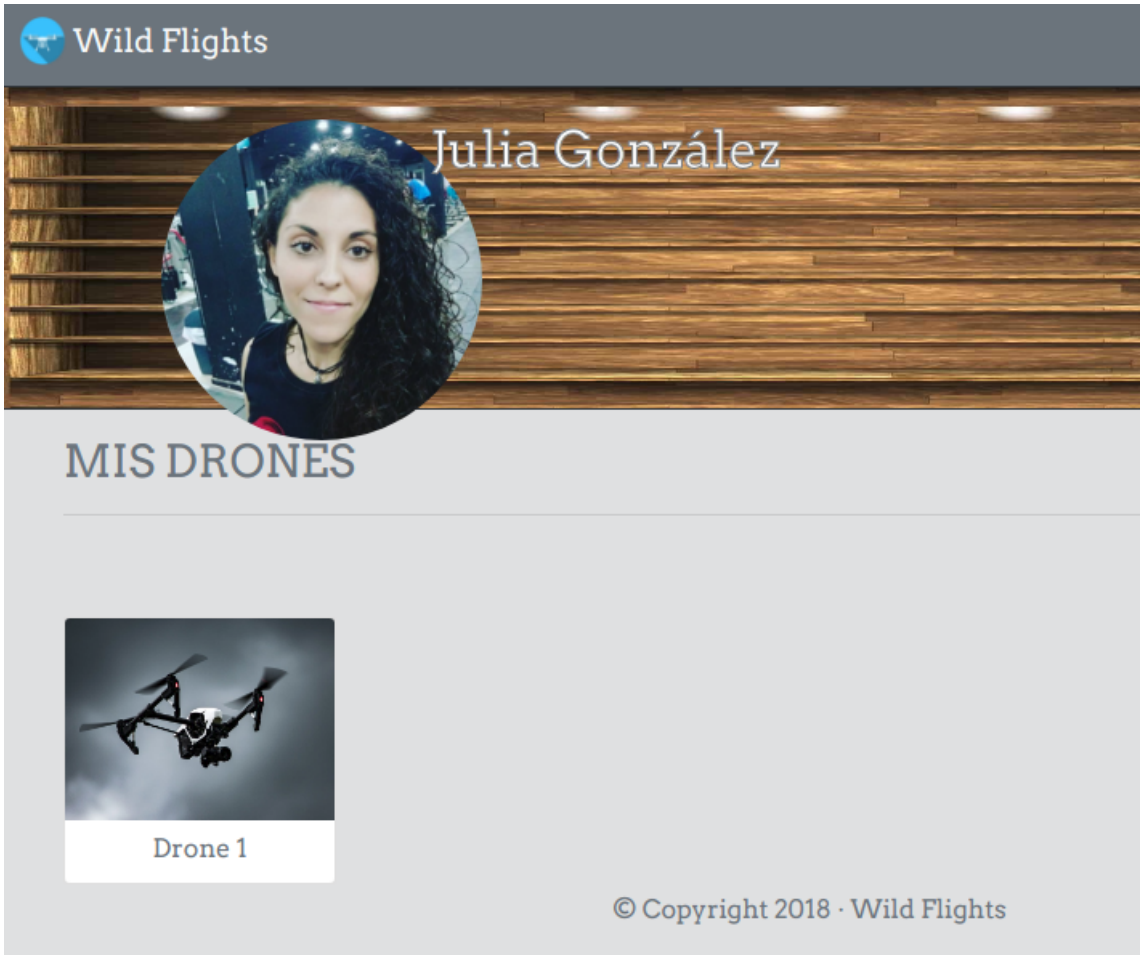

Figura 5.3: Captura después de editar imagen del perfil

En la figura 5.4 se muestra un ejemplo de notificación de error por intentar crear un usuario ya existente en la aplicación:

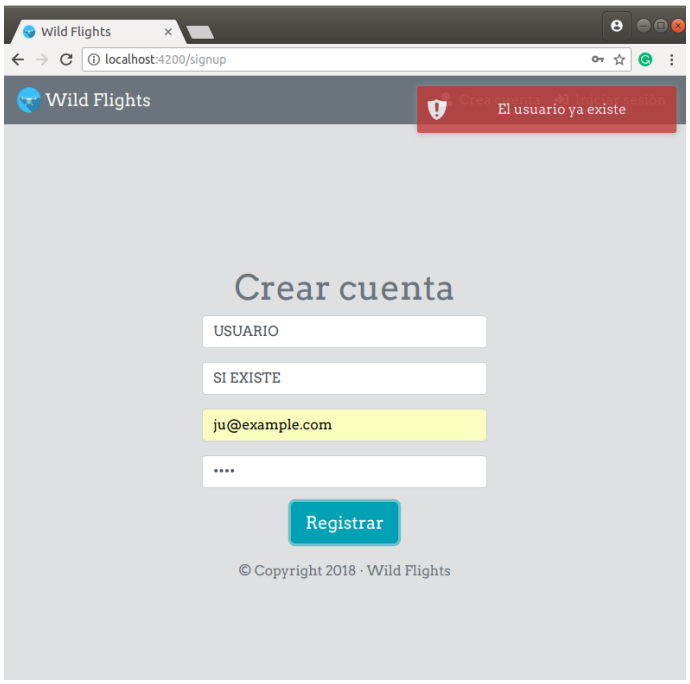

Figura 5.4: Notificación de la creación de un usuario ya existente

En la figura 5.5 se muestra un ejemplo de notificación de error por intentar crear un usuario introduciendo datos incorrectos en el formulario:

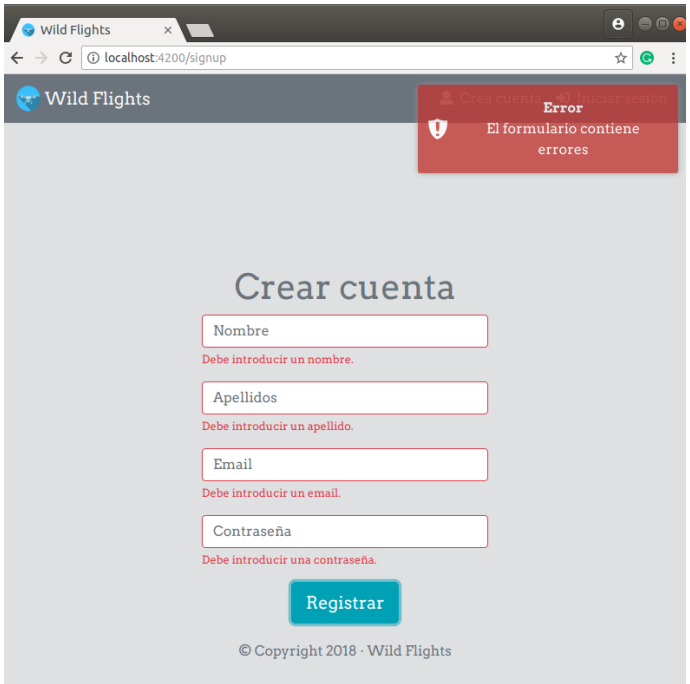

Figura 5.5: Notificación de error en el formulario de registro de usuario

En la figura 5.6 se muestra un ejemplo de notificación de error por intentar iniciar sesión introduciendo datos incorrectos:

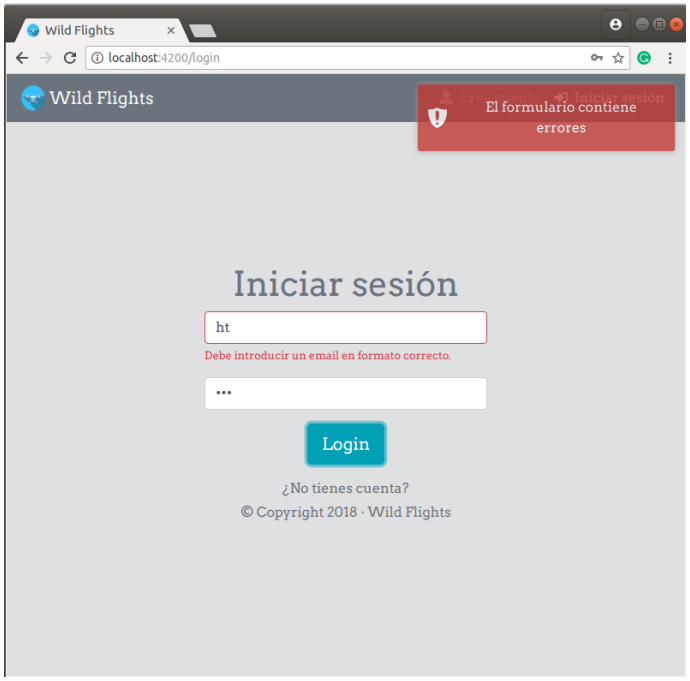

Figura 5.6: Notificación de error en el formulario de Inicio de sesión

En cambio, si el usuario ha podido iniciar sesión correctamente se muestra una ventana como la de la figura 5.7:

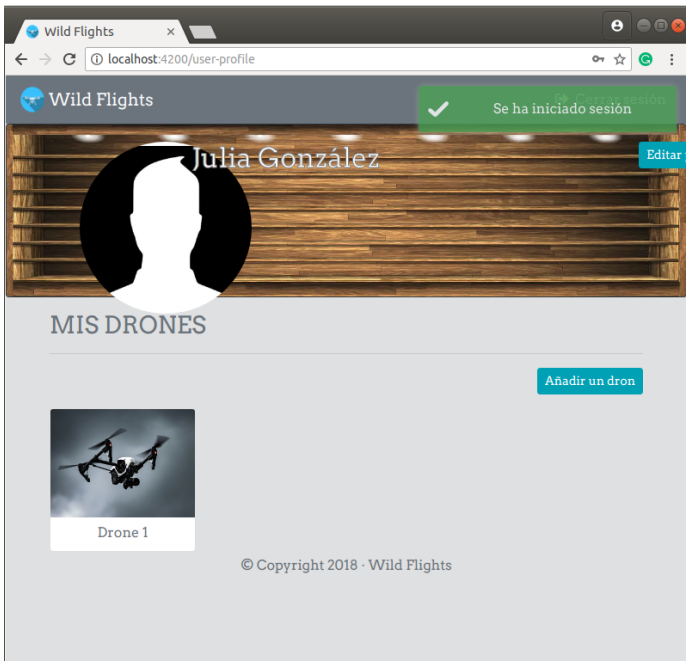

Figura 5.7: Captura después de editar imagen del perfil

CAPÍTULO 6
Conclusiones

Empezar a desarrollar este proyecto ha sido complejo por el motivo de que solo se tenía conocimientos básicos de algunas tecnologías utilizadas. En cambio, el utilizar un *framework MEAN Stack* ha facilitado mucho el desarrollo ya que solo utiliza Javascript como lenguaje y de éste ya se tenía conocimiento previo.

Este proyecto ha servido para conocer todas las caras que debe cumplir un buen desarrollador web, ya que se ha desarrollado la parte *frontend* como la parte *backend*. Se han realizado peticiones CRUD a una base de datos no relacional.

Además del desarrollo web, se ha completado el proyecto realizando el desarrollo de un dispositivo *hardware* desde cero que se encarga de enviar los datos de geolocalización. Esto ha servido para conocer aspectos de un desarrollo en la parte de *hardware*.

Durante la implementación del proyecto hubieron algunos problemas como por ejemplo:

Un problema con el disco duro del ordenador que se estaba utilizando para desarrollar la aplicación y provocó la pérdida de todo aquello que se había desarrollado hasta el momento. A partir de ahí, se utilizó un control de versiones llamado Git que sirve para realizar copias del código en un repositorio almacenado en la nube y de esta forma no perderlo.

También hubo otro problema con la compra del módulo SIM900 ya que se hizo por internet y éste llegó defectuoso. Así pues, se tuvo que devolver y pedir otro igual. Esto causó una pérdida de varios días de trabajo.

A pesar de las adversidades encontradas en el camino ha sido posible finalizar el desarrollo total del proyecto. Además, el desarrollo web es el campo de trabajo al que la autora quiere dedicarse y realizar este proyecto hace que se familiarice y abra boca con algunas tecnologías que se utilizan hoy en día en el desarrollo web. Así, podrá continuar realizando tareas sobre el mismo campo de trabajo.

6.1 Trabajos futuros

Al iniciar este proyecto se tenía una idea básica de la aplicación web que se quería implementar, pero a medida que se iba desarrollando han surgido más ideas o funcionalidades que podrían llegar a implementarse en un futuro. También se han encontrado posibles mejoras de la aplicación desarrollada.

A continuación se detallan algunas de esas ideas y mejoras encontradas:

- **Aumentar la velocidad de respuesta:** Una buena mejora sería aumentar el tiempo de respuesta de la placa arduino con el servidor mediante el módulo SIM900.
- **Cambiar de módulo *hardware*:** El módulo SIM900 tiene demasiado consumo. Una de las posibles mejorar sería utilizar otro módulo de sim que sea más pequeño y no consuma tanto. Existe un módulo que incorpora un chip SIM808 y que solo necesita entre 3.3V y 5V, no como el módulo SIM900 que necesita consumir más 9V para funcionar.
- **Cambiar de placa arduino:** En este proyecto se utiliza la placa Arduino UNO que es una de las más básica que tiene arduino. Una de las mejoras sería cambiar de placa a una más pequeña y a poder ser que llevase ya incluida la localización por GPS, de esta forma, no haría falta tener otro módulo GPS añadido a la placa después. También sería viable cambiar esta placa a una Raspberry Pi ya que estas suelen ser más potentes.
- **Aplicación válida para distintos dispositivos:** La aplicación está diseñada básicamente para aeronaves no tripuladas, pero podría realizarse uno pequeños cambios en la API y de esta forma la parte de *frontend* de la aplicación podría implementarse para distintos dispositivos utilizando el mismo servidor. El cambio es simple, basta con modificar los modelos y las rutas referidas a dron y vuelo, y modificar sus nombres a dispositivo y registro.
- **Mejorar los estilos de las páginas:** Los estilos existentes en cada vista son bastante básicos. Para mejorarlos se podría modificar los archivos css de los elementos básicos utilizados de Bootstrap. También se podría utilizar uno de los estilos predefinidos de Bootstrap que no sean básicos.
- **Convertir la aplicación en una red social:** Además de gestionar los datos de geolocalización de los dispositivos registrados por cada usuario, estaría bien poder compartir esos datos con otros usuarios registrados en la misma aplicación. De esta forma se podría crear un muro parecido a los que aparecen en cualquier red social donde es posible comentar y compartir los registros de vuelos realizados por los drones correspondientes a cada usuario.

CAPÍTULO 7
Agradecimientos

Me gustaría dar las gracias a todas las personas que han sido persuasivas conmigo para que finalice y entregue el Trabajo de Final de Grado.

En especial a toda mi familia por el apoyo que me han otorgado.Y a Jorge, mi pareja, por aconsejarme y ayudarme a pesar de mis nervios durante los últimos meses.

También quiero dar las gracias a GitHub, y a toda la comunidad de desarrolladores que lo envuelve, por el magnífico IDE que lanzaron llamado **Atom** que me ha permitido desarrollar fácilmente en los distintos lenguajes utilizados.

Bibliografía

[1] Angular, ed. *Angular*. 22 de jun. de 2018. URL: https://angular.io/guide/architecture (visitado 22-06-2018).

[2] aponxi. "SQL to MongoDB Mapping Chart". En: 25 de ago. de 2018. URL: https://gist.github.com/aponxi/4380516 (visitado 25-08-2018).

[3] Arduino, ed. *Arduino*. 28 de ago. de 2018. URL: https://www.arduino.cc/en/Guide/Linux (visitado 28-08-2018).

[4] creationix, ed. *Node Version Manager*. 20 de ago. de 2018. URL: https://github.com/creationix/nvm (visitado 18-04-2018).

[5] MDN web docs. "Fundamentos de Javascript". En: 30 de mayo de 2018. URL: https://developer.mozilla.org/es/docs/Learn/Getting_started_with_the_web/JavaScript_basics (visitado 30-05-2018).

[6] Digital Tech Institute. "THE MEAN STACK: EL PURA SANGRE DE LOS STACKS". En: 30 de mayo de 2018. URL: https://www.digitaltechinstitute.com/mean-stack/ (visitado 28-02-2017).

[7] Daniel Jimenez. "El Diagrama de Contexto, Tutoriales parala ISO 9001:2015". En: 20 de jun. de 2018. URL: https://www.pymesycalidad20.com/el-diagrama-de-contexto-tutoriales-para-la-iso-90012015.html (visitado 06-03-2015).

[8] Mimedu, ed. *Ejemplo de Requisitos*. 20 de jun. de 2018. URL: https://mimedu.es/file/2015/03/calle.jpg (visitado 20-06-2018).

[9] MongoDB, ed. *MongoDB and MySQL Compared*. 25 de ago. de 2018. URL: https://www.mongodb.com/compare/mongodb-mysql (visitado 25-08-2018).

[10] Enrique Oriol. "Angular2: ¿Aprendo ES6 o TypeScript?" En: 23 de mayo de 2018. URL: http://blog.enriqueoriol.com/2016/06/angular2-aprendo-es6-o-typescript.html (visitado 05-06-2016).

[11] Priyesh Patel. "What exactly is Node.js?" En: 20 de ago. de 2018. URL: https://medium.freecodecamp.org/what-exactly-is-node-js-ae36e97449f5 (visitado 18-04-2018).

[12] Simcom, ed. *Manual SIM900*. 1 de sep. de 2018. URL: https://www.espruino.com/datasheets/SIM900_AT.pdf (visitado 01-09-2018).

[13] Simcom, ed. *SIM900*. 1 de sep. de 2018. URL: http://simcomm2m.com/En/module/detail.aspx?id=71 (visitado 01-09-2018).

[14] Srcmake, ed. *NoSQL and MongoDB Tutorial - Quick Cheat Sheet*. 20 de ago. de 2018. URL: https://www.srcmake.com/home/nosql-and-mongodb-tutorial (visitado 17-02-2018).

[15] StackOverflow, ed. *Most Popular Technologies: Frameworks, Libraries, and Tools*. 22 de jun. de 2018. URL: https://insights.stackoverflow.com/survey/2018/#technology-frameworks-libraries-and-tools (visitado 22-06-2018).

[16] Acens Technologies. "Bootstrap, un framework para diseñar portales web". En: 28 de ago. de 2018. URL: https://www.acens.com/wp-content/images/2016/10/bootstrap-framework-acens-wp.pdf (visitado 28-08-2018).

[17] Chrome V8, ed. *Chrome V8*. 16 de ago. de 2018. URL: https://angular.io/guide/architecture (visitado 16-08-2018).

[18] WikiFoundry, ed. *Diagrama de Contexto*. 20 de jun. de 2018. URL: http://clases3gingsof.wikifoundry.com/page/Diagrama+de+Contexto (visitado 20-06-2018).

[19] Wikipedia, ed. *Bootstrap*. 27 de ago. de 2018. URL: https://es.wikipedia.org/wiki/Bootstrap_(framework) (visitado 27-08-2018).

[20] Wikipedia, ed. *Casos de uso*. 22 de jun. de 2018. URL: https://es.wikipedia.org/wiki/Diagrama_de_casos_de_usol (visitado 22-06-2018).

[21] Wikipedia, ed. *CSS*. 27 de ago. de 2018. URL: https://es.wikipedia.org/wiki/Hoja_de_estilos_en_cascada (visitado 27-08-2018).

[22] Wikipedia, ed. *Javascript*. 30 de mayo de 2018. URL: https://es.wikipedia.org/wiki/JavaScript (visitado 30-05-2018).

[23] Wikipedia, ed. *Mean Stack*. 30 de mayo de 2018. URL: https://es.wikipedia.org/wiki/MEAN (visitado 30-05-2018).

[24] Wikipedia, ed. *Typescript*. 23 de mayo de 2018. URL: https://es.wikipedia.org/wiki/TypeScript (visitado 23-05-2018).

Glosario

Backend En el diseño *software* de aplicaciones web, capa de lógica y acceso a bases de datos. Obtiene entradas del *frontend*. 18

Footer Parte diferenciada visiblemente de un documento web que se encuentra en la parte inferior, y cuyo diseño normalmente se mantiene inmutable a lo largo de una aplicación web. Suele contener enlaces o información destacable. 26

Framework Conjunto de conceptos, herramientas y mejores prácticas que forman un entorno de trabajo para abordar determinada situación o problemática. 3

Header Parte superior de una página web, que se diferencia con facilidad, y que suele contener elementos como el logo, menú de navegación y otros datos importantes. Suele mantenerse inmutable a lo largo de la aplicación. 26

Single page application (SPA) En español aplicación de página única. Es una aplicación cuyo contenido se encuentra en el mismo documento web, facilitando así la navegación por la misma. 3

frontend En el diseño *software* de aplicaciones web, es la capa de presentación o parte con la que interactúa el usuario. 3

Angular-CLI Interfaz de línea de comandos del *framework* Angular. Es una herramienta para inicializar, desarrollar y mantener aplicaciones Angular. 24

API REST Una especificación API REST (Representational State Transfer), consiste en una API que respeta los métodos HTTP para la definición de la misma, es decir, *GET* para consultar, *PUT* para modificar, *POST* para añadir y *DELETE* para eliminar. 5

APIs Procede del término en inglés *Application Programming Interfaces* (Interfaces de programación de aplicaciones) y consiste en una especificación formal sobre cómo una entidad *software* se comunica o interactúa con otra. 9

CRUD Acrónimo de Crear, Leer, Actualizar y Borrar, referido normalmente a operaciones sobre objetos. 53

licencia MIT También llamada licencia X11, consiste en una licencia de código abierto originada en el Instituto Tecnológico de Massachusetts (*MIT, Massachusetts Institute of Technology*). Es compatible con muchas de licencias *copyleft*. 5

Mongoose Es un módulo de Javascript, que contiene herramientas de modelado de objetos en MongoDB para trabajar en un entorno asíncrono. 19

sesiones Información de estado y de conexión del usuario respecto del sistema. 5