Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Sistema para la monitorización remota de vehículos aéreos no tripulados

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Carlos Javier López Casero

Tutor: Carlos Miguel Tavares de Araujo Cesariny Calafate

2019/2020

Resumen

El presente proyecto ilustra el proceso necesario para la realización de un sistema de monitorización remota para vehículos aéreos no tripulados. Con el fin de obtener este resultado, un dron debe conectarse por medio de tecnología celular y enviar los datos obtenidos por un módulo GPS a un servidor intermediario, el cual, una vez recibida la información, la inserta en una base de datos para permitir un rastreo continuo y en tiempo real de dicho dron.

Adicionalmente, se desarrolla una aplicación web dinámica basada en el paquete de tecnologías conocido como MEAN Stack (MongoDB, Express, Angular y NodeJS). Dentro de sus funcionalidades, se destaca la posibilidad de supervisar y localizar en tiempo real uno o varios drones, así como revisar anteriores trayectos que se hayan realizado en una fecha concreta.

Finalmente, se comprobará el funcionamiento del sistema con una herramienta software capaz de simular el comportamiento de un dron real (ArduSim) mediante una pequeña modificación de dicho simulador.

Palabras clave: sistema de monitorización, vehículos aéreos no tripulados, GPS, aplicación web, MEAN Stack, tiempo real, Ardusim.

Abstract

This project illustrates the necessary process for the development of a remote monitoring system for unmanned aerial vehicles. With the purpose of getting this result, a drone must have cellular connectivity and transmit data from a GPS module to an intermediary server, which, upon receiving the data, inserts it into a database; this allows a continuous real-time tracking of the drone.

Furthermore, a dynamic web application is developed with the packet of technologies known as MEAN Stack (MongoDB, Express, Angular and NodeJS). Some of its functionalities are the possibility of supervising and locating in real time one or more drones. Moreover, it is possible to check previous journeys that a drone has done on a specific date.

Finally, the functionality of the system will be validated using a software tool able to simulate a realistic behaviour of a drone (ArduSim), which was made possible thanks to an improvement to its software.

Keywords: Monitoring system, unmanned aerial vehicle, GSM, GPS, intermediary server, application web, MEAN Stack, real time, Ardusim.

Índice de contenidos

Índice de ilustraciones

Índice de tablas

1. Introducción

En la actualidad, gracias al avance de las nuevas tecnologías y a las necesidades surgidas en la sociedad, resulta inevitable no echar un vistazo a las aplicaciones y capacidades que pueden y podrán proporcionar los drones en nuestro día a día. Desde ámbitos más personales como el entretenimiento, hasta más profesionales como la agricultura o el reparto a domicilio, el abanico de opciones se sigue incrementando día a día.

Como nueva necesidad, nace un interés por parte de emprendedores e investigadores para acercar nuevos servicios y productos a la población. Como consecuencia, numerosos administradores, operadores e investigadores requieren de sistemas de supervisión con fin de realizar un seguimiento de los drones que desean controlar.

En un sistema de monitorización resulta interesante la posibilidad de geolocalizar en un único mapa los vehículos no tripulados en tiempo real [1], así como realizar un seguimiento individual de cada uno de ellos. Adicionalmente, un historial de trayectos para cada dron aporta al usuario la posibilidad de comprobar viajes en cualquier momento.

En este proyecto se va a crear un sistema de monitorización capaz de recolectar e integrar los datos enviados por uno o varios drones, y agruparlos en una base de datos para su posterior procesamiento [2]. Para la exposición de los parámetros al usuario se va a crear una página web, la cual permitirá localizar los drones en un mapa, así como observar parámetros como coordenadas, altura, o incluso velocidad.

1.1. Motivación

La razón principal del desarrollo de este proyecto es ofrecer un sistema capaz de registrar el trayecto de un dron en tiempo real y visualizarlo en un mapa, así como su posterior visualización mediante análisis de trazas.

Resulta interesante para un administrador, operador o investigador que va a trabajar con varios drones a la vez, disponer de un sitio web para poder visualizar cómo se comportan unos drones ante determinadas rutas, además de proporcionarle un medio accesible e intuitivo para poder comprobar trayectos o incluso tomar estadísticas.

Con la realización del proyecto se van a reforzar y obtener conocimientos para lograr que un dron se conecte y envíe datos a un servidor. Dichos conocimientos se van a juntar con los relativos al diseño web mediante un novedoso paquete de tecnologías destinado a la programación web conocido como *MEAN Stack*.

1.2. Objetivos

El objetivo principal del proyecto consiste en la elaboración de un sistema de monitorización compuesto por un servidor, encargado de recibir información de uno o varios drones, e insertarla en una base de datos. Adicionalmente, deberá desplegar una aplicación web capaz de cargar dinámicamente los parámetros enviados por un dron. Con el fin de obtener pruebas concluyentes que determinen el correcto funcionamiento del sistema se va a desarrollar una modificación del código de un software encargado de simular el comportamiento de un dron. En resumidas cuentas, el sistema debe ser capaz de realizar las siguientes acciones:

- Establecer una conexión, con la mayor fiabilidad posible, con los drones de manera que pueda recibir todos los datos enviados y, posteriormente, insertarlos en una base de datos.
- Implementar una página web dinámica y usable capaz de conectarse a dicha base de datos para disponer de los parámetros solicitados por el usuario. Concretamente, se deberán presentar las siguientes posibilidades:
    - Visualizar en tiempo real uno o varios drones simultáneos en un mapa, junto con una lista de parámetros relacionados con ellos.
    - Permitir al usuario, de forma adicional, contemplar los movimientos que ha realizado un dron con fecha establecida.
- Emplear la herramienta modificada 'ArduSim' para simular el vuelo de los drones.

1.3. Impacto Esperado

El empleo del sistema de monitorización para drones puede abarcar desde usuarios aficionados a los drones hasta pequeñas empresas que requieran de este tipo de sistemas. Sin embargo, el proyecto ha sido diseñado y adaptado fundamentalmente a investigadores que requieran de un medio para supervisar el vuelo de vehículos no tripulados.

Previo a un vuelo con drones reales, un investigador puede emplear el sistema de monitorización junto con un simulador de vuelos de drones. De esta manera, se puede comprobar la eficacia de los vuelos mediante la contemplación del mapa y sus parámetros.

En el momento de probar con vuelos reales, el usuario podrá visualizar y controlar en tiempo real la posición y el trayecto que está realizando el conjunto de drones para poder detectar rápidamente cualquier problema que surja. La aplicación web permite la visualización simultanea de varios drones , proporcionando una vista aérea general con carácter de monitorización.

Nótese que el servidor almacena toda la información en una base de datos, por lo que el usuario puede acceder a la funcionalidad de revisar el trayecto de anteriores vuelos, junto con sus correspondientes parámetros.

1.4. Metodología

La metodología empleada durante el proceso de desarrollo del software es una de las metodologías agiles más empleadas hoy en día en el ámbito informático, la llamada *Scrum*.

Suele ser empleada en proyectos basados en el trabajo en equipo y con entornos complejos donde la innovación, flexibilidad y productividad son esenciales [3]. Se trata de un proceso iterativo que aplica un conjunto de buenas prácticas para obtener el mejor resultado posible en un proyecto.

Su objetivo es priorizar el beneficio que se aporta al receptor del proyecto. Para ello, se divide el trabajo en tareas en función de los requisitos funcionales que se establecen, y se realizan entregas parciales en ciclos temporales cortos y de duración fija (desde 1 a 4 semanas), conocidos como *sprints*. Adicionalmente, si se diese el caso, diariamente se realizarían breves reuniones en las que cada miembro del equipo indica la función que realizó el día anterior.

***Ilustración 1**. Funcionamiento de la metodología Scrum.*

Fuente: https://www.sinnaps.com/blog-gestion-proyectos/metodologia-scrum

1.5. Estructura

- **Capítulo 1: Introducción**. Corresponde al actual capítulo, y en él se presenta una breve introducción al proyecto, sus objetivos, motivaciones y metodología empleada para su desarrollo.
- **Capítulo 2: Estado del arte**. Capítulo donde se estudia el contexto tecnológico del ámbito que abarca este proyecto. Para ello se comparará con otras alternativas, y se establecerá una propuesta que las mejore.
- **Capítulo 3: Análisis del problema**. En este capítulo se va a concretar el producto a desarrollar en el proyecto, los posibles problemas que surjan, así como su alcance y funcionalidad

- **Capítulo 4: Diseño de la solución**. En este capítulo se expondrá la estructura del sistema, indicando la relación que mantienen los diferentes elementos del sistema.
- **Capítulo 5: Desarrollo de la solución.** El desarrollo y solución del proyecto se aportará en este capítulo.
- **Capítulo 6: Implantación**. Capítulo donde se presenta el desarrollo realizado para la explotación e implantación de la solución para el proyecto.
- **Capítulo 7: Pruebas.** Capítulo donde se muestra el correcto funcionamiento del sistema.
- **Capítulo 8: Conclusiones**. Finalmente, se expondrán los resultados y se llegará a unas conclusiones a la hora de llevar a cabo el software realizado, así como una pequeña reflexión de posibles mejoras para el futuro.

Al final de la memoria se encontrarán las referencias empleadas tanto para los contenidos como para las imágenes. Adicionalmente, se incorporará un anexo donde se encontrará parte del código desarrollado.

2. Estado del arte

Con el drástico descenso del precio de un vehículo aéreo no tripulado, y su gran aplicabilidad a diferentes sectores profesionales, se está considerando como uno de los recursos y medios más valorados para actividades que, hasta día de hoy, eran impensables. A pesar de que el origen del dron está relacionado con el poder militar, cada día los drones son más empleados en ámbitos recreativos y profesionales [4] como la fotografía, agricultura, ingeniería civil, control de medio ambiente, minería o incluso seguridad.

En todos estos campos, mejora considerablemente la experiencia de uso llevar un seguimiento de la posición y los movimientos que realiza o ha realizado dicho vehículo. Consecuentemente, muchos drones están comenzando a ser monitorizados de diferentes maneras.

Muchas empresas y desarrolladores se han lanzado al mercado para presentar un producto destinado al usuario con el fin de disponer de una interfaz usable para la supervisión de un dron.

En este capítulo se van a abarcar las tecnologías de comunicación posibles para conectar un dron a un servidor, las alternativas software para el visualizado de la monitorización, así como los productos software competentes que se han creado para la monitorización de un dron.

2.1. Tecnologías de comunicación

Existen diversas maneras de conectar un dron a un servidor para que este lo monitorice. A continuación, se van a mostrar aquellas tecnologías interesantes para llevar a cabo la comunicación.

2.1.1. Comunicación por radio

Actualmente existen ciertos sitios webs como *Flightradar24* capaces de proporcionar información a un usuario como la posición en un mapa, coordenadas, altura y velocidad de una gran cantidad de vuelos de aviones. Esto es posible gracias a la inmensa red mundial de radios

terrestres que envían constantemente señales *ping*, las cuales serán respondidas por los transpondedores de los aviones [5].

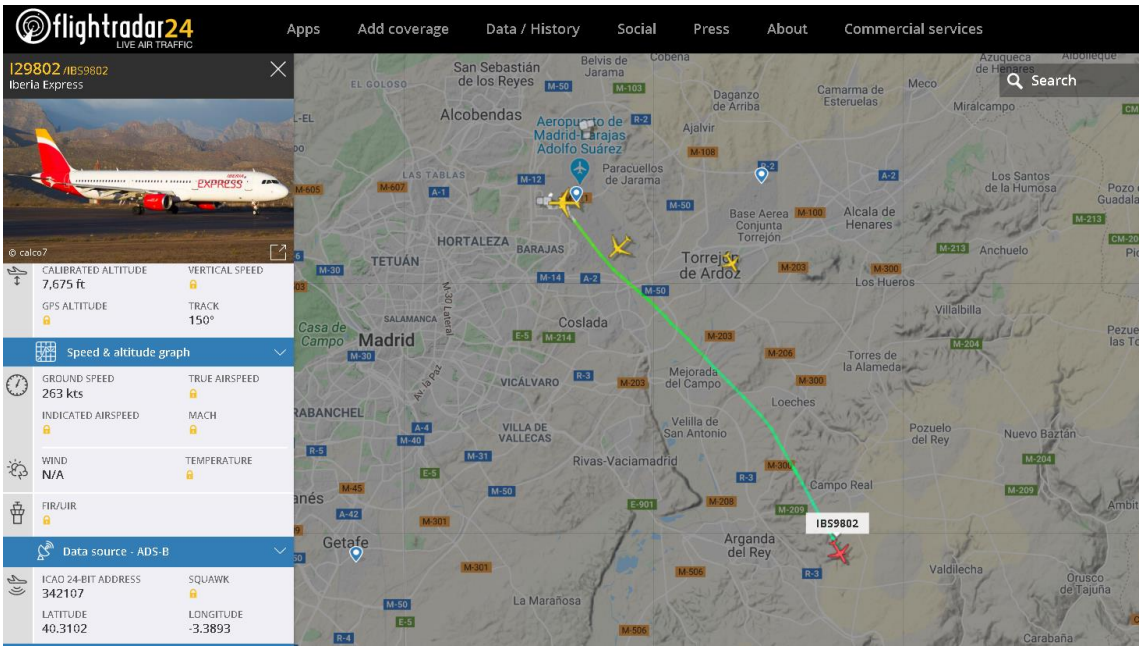


***Ilustración 2**. Ejemplo de monitorización de un avión.*

La incorporación de esta tecnología en los drones resulta sencilla, pues únicamente se debe añadir un sistema capaz de responder a estas señales *ping*, aunque solo se puede usar en drones de gran capacidad de carga debido a su tamaño y peso. Dicho sistema se conoce como **ADS-B** [6] (*Automatic Dependent Surveillance Broadcast*) y se caracteriza por transmitir regularmente (una vez por segundo) datos de interés en el canal 1090 MHz [7] e incluye los siguientes beneficios:

- Este tipo de sistema ofrece una **mejora de seguridad y eficacia** de vuelo a los pilotos y controladores aéreos.
- **Permite controlar el tráfico aéreo**, de forma que se podría obtener un mapa con todos los drones circulantes, mejorando entre ellos la seguridad aérea.
- No solo recibe y responde a las señales *pings*, sino que **puede recibir información externa** como indicaciones meteorológicas, así como **enviar mensajes de emergencia**.
- **Es una tecnología barata** en comparación a otros tipos de tecnologías basadas en radares.

***Ilustración 3**. Esquema ADS-B.*

Fuente: http://www.aviacionglobal.com/articulos-tecnicos-de-aviacion/como-funciona-el-ads-b-la-tecnologia-que-viene-ya-esta-aqui/

Estados Unidos ha obligado que esta tecnología sea incorporada en cualquier aeronave a partir del 1 de enero de 2020 [8], y los drones de pequeña dimensión deberán tener receptores ADS-B para poder esquivar las aeronaves. El objetivo de esta implantación persigue una unificación del control aéreo de cualquier vehículo volador, con el fin de mejorar y controlar la seguridad en el cielo. De esta manera, se permitirá visualizar en un único mapa todo el tráfico aéreo con los respectivos parámetros de cada vuelo.

2.1.2. Comunicación por RPS y GSM

Dada la necesidad de incluir la monitorización en muchos ámbitos del empleo de un dron, muchas compañías telefónicas han dado el paso de ofrecer una conexión entre un dron y un servidor.

El primer caso fue propuesto por la multinacional Vodafone en el año 2017, junto a CATEC (Centro Avanzado de Tecnologías Aeroespaciales) donde adaptaron el hardware y el software de un dron para conectarlo con la red móvil 4G de Vodafone. De esta manera, se logró por primera vez controlar y posicionar un dron conectado con una tarjeta SIM con la red de telefonía móvil.

La tecnología empleada para lograr este avance se basa en el RPS (*Radio Positioning System)* y se caracteriza por [9]:

- Mediante las estaciones radio de la compañía **se localiza al dron con envíos de señales**; una vez localizado, se toma como referencia las ondas enviadas y sus respectivas intersecciones. Con empleo de complejos algoritmos obtienen resultados un margen de error de aproximadamente 60 metros [10].
- **Alta robustez y protección** ante posibles intentos de falsificación o inhibición de la señal GPS.
- **Sistema autónomo**, de manera que no necesita la colaboración del dron para conocer su posición geográfica. Nótese que no será necesario el empleo de la tecnología GPS.

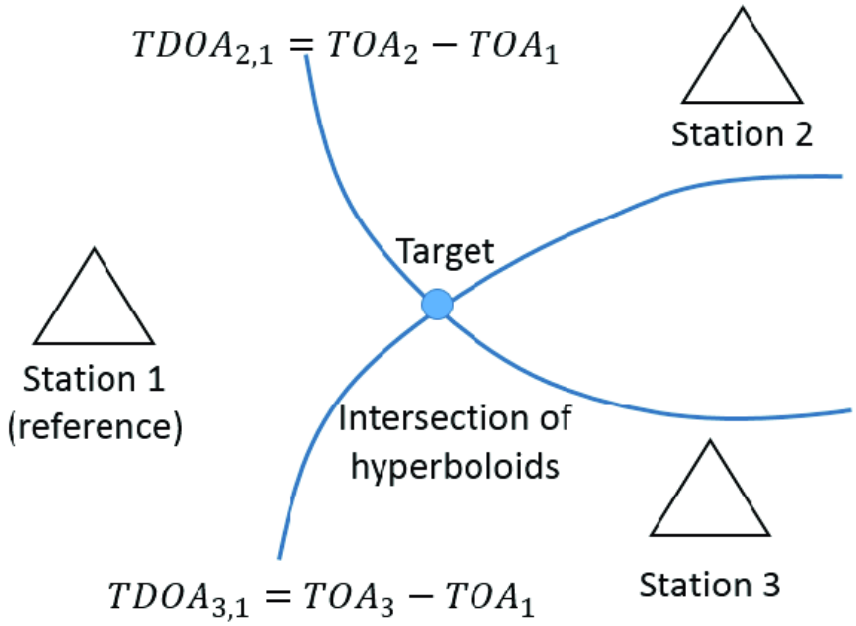


***Ilustración 4**. Esquema RPS.*

Fuente: https://www.researchgate.net/figure/An-illustrated-2D-view-of-TDOA-Localization_fig2_327647859

Esta tecnología propone una solución para gestionar y controlar eficientemente el espacio aéreo de los novedosos vehículos voladores no tripulados dedicados al ámbito profesional. Un proyecto que gusta especialmente en la comunidad aérea del transporte europeo, que busca una solución masiva para la seguridad del tráfico aéreo.

Tres años después, con el gran interés por la tecnología RPS y la llegada al mercado de la red 5G, Vodafone adaptó su proyecto con un receptor 5G. De esta manera mejoró considerablemente la comunicación con los drones, afirmando que los conflictos de proximidad entre drones en el espacio aéreo se solventarían con mayor seguridad [11].

Consecuentemente, el programa SESAR (*Single European Sky ATM Research*) encargado de desarrollar e implementar un sistema común de gestión de tráfico aéreo de Europa, está participando en dicha tecnología [12]. Dicho programa estima que podrían haber más de 400.000 drones volando por el espacio aéreo de Europa de forma controlada y segura en 2035. Se espera lograr el empleo de drones profesionalmente en situaciones delicadas, como el vuelo seguro en bajas alturas en zonas con alta densidad de población.

2.1.3. Comunicación por GPS y red celular

A día de hoy, gran parte del mundo está conectada entre sí mediante redes 4G proporcionadas por diferentes proveedores. Por eso, en lugares con cobertura resulta posible conectar un dron dotado de un receptor de señal 4G para que envíe datos a un servidor.

Esta tecnología se basa en incorporar en el dron un módulo GPS y la tecnología GSM (*Global System for Mobile*) con el objetivo de recibir datos geológicos vía GPS y enviar los correspondientes datos a un servidor a través de tecnología de telefonía móvil digital.

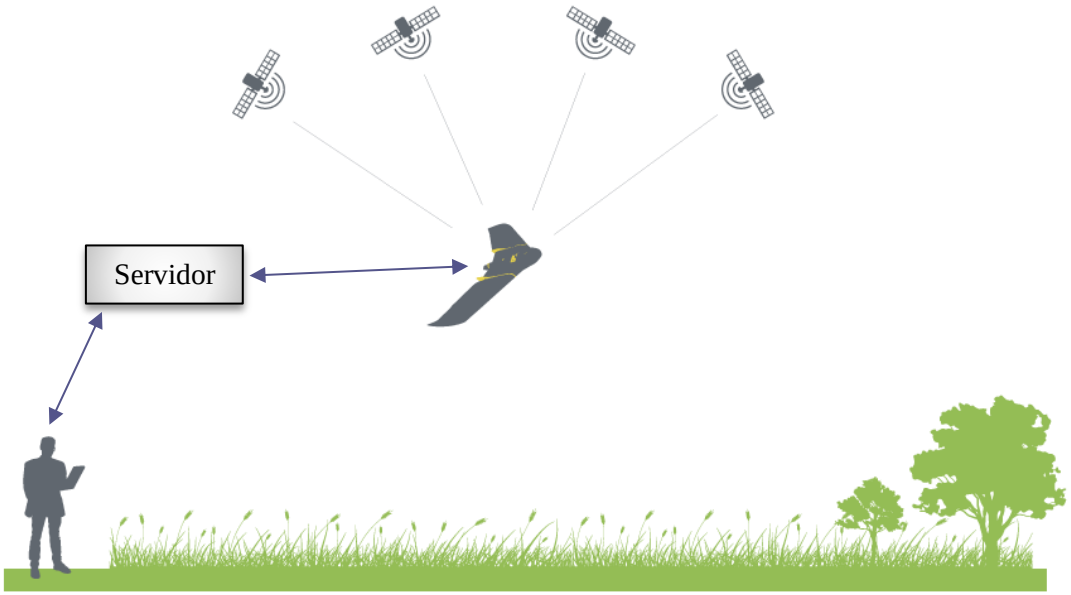


***Ilustración 5**. Esquema monitorización GPS y GSM.*

Fuente: https://www.pix4d.com/es/blog/los-drones-rtk-ppk-le-dan-mejores-resultados-que-el-uso-de-gcp

2.2. Tecnologías para el software

Diversas empresas y emprendedores dedicadas al software se han lanzado al mercado para cubrir la necesidad de la monitorización de los drones. Para ello, han implementado, mayoritariamente, aplicaciones móviles capaces de conectarse a un dron y obtener una supervisión de este.

Hoy en día, la mayoría de las páginas web que se crean son con fines informativos, de manera que se emplean principalmente para dar conocer un producto o una herramienta. En cambio, con el avance tecnológico resulta factible emplear un sitio web para crear aplicaciones con diversas funcionalidades.

Obviamente, si se tratase de desarrollar un sitio web capaz de monitorizar en tiempo real la posición de un dron, el método tradicional de elaboración de páginas web estáticas por Hipertexto resultaría mucho más laborioso y complicado ya que estas están pensadas para mostrar simples textos e imágenes con poco movimiento.

Es aquí donde entran las páginas web dinámicas, también conocidas como ***Single Page Application***. Su principal característica se basa en su similitud respecto al comportamiento y funcionalidades que podría presentar una aplicación móvil o de escritorio.

Su dinamismo es posible gracias al empleo de ejecutables o scripts basados en el lenguaje de programación JavaScript. De esta manera, se pueden eliminar, modificar o añadir elementos a la interfaz sin necesidad de refresco en la página web. Esto proporciona al usuario una sensación parecida a una aplicación móvil.

Como la mayoría de los lenguajes de programación, disponen de varios *frameworks* que facilitan y mejoran el desarrollo de un producto. Permitiendo al desarrollador centrarse en lo verdaderamente importante a la hora de diseñar una web, evitando que este vuelva a reinventar la rueda.

Para la realización de este tipo de aplicaciones web existen diversos paquetes de software encargados de proporcionar una estructura y estabilidad. Uno de los más conocidos se trata de MEAN Stack que, posteriormente, se estudiará en el apartado de tecnologías empleadas.

2.3. Propuestas Actuales

En este apartado se van a contemplar las diferentes alternativas actuales disponibles en el mercado para la monitorización de un vehículo no tripulado. Se van a abarcar diferentes productos, desde únicamente aplicaciones móviles hasta sitios web con diferentes niveles de posibilidades.

2.3.1. AirMap

La plataforma AirMap proporciona una manera segura, confiable y precisa de viajar en el espacio aéreo. Está dotado de avanzados servicios de automatización software capaces de permitir la aviación no tripulada de manera segura [13]. Dentro de sus funcionalidades se pueden destacar:

- **Automatización del espacio aéreo.** Es capaz de mostrar toda la información a considerar antes de realizar un vuelo para la zona planeada, inluyendo aviones que van a pasar por la misma zona, fuertes vientos, o incluso la presencia de otro dron, evitando una posible colisión.

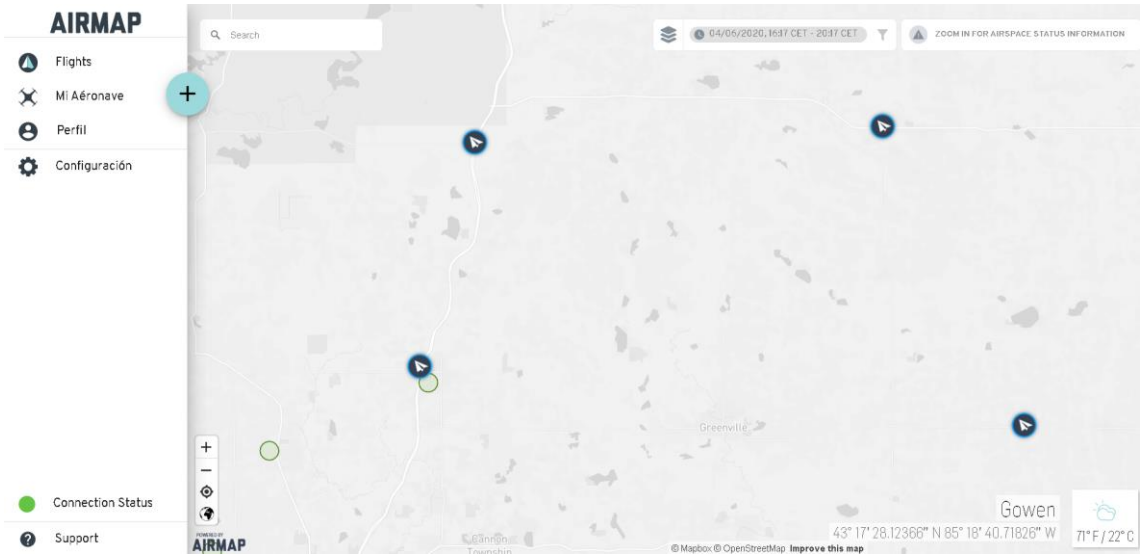


***Ilustración 6**. Captura del mapa en tiempo real de varios drones.*

- **Automatización del vuelo**. Permite al usuario automatizar el vuelo de un dron. Por ejemplo, puede planificar un trayecto, aterrizar, o incluso volver al punto de partida.

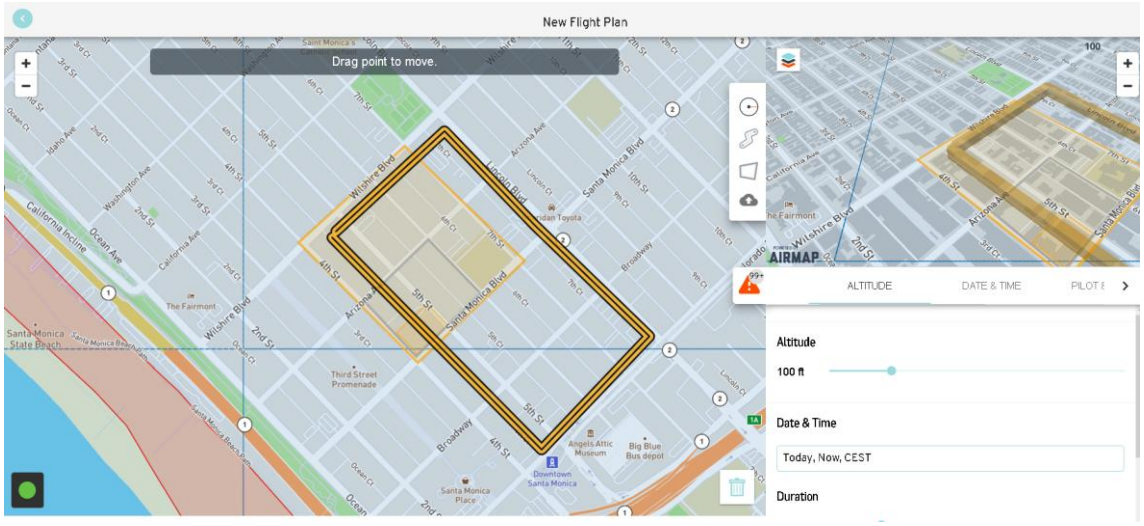


***Ilustración 7**. Captura de la planificación del viaje.*

Su servicio es accesible desde una aplicación móvil o desde una aplicación web de una sola página que se actualiza dinámicamente.

2.3.2. DroneDeploy

DroneDeploy no es únicamente una aplicación móvil para poder monitorizar un dron, sino una de las mayores organizaciones centradas en la explotación de drones en el ámbito profesional. En su página web se puede apreciar una gran cantidad de información relacionada con el mundo del vehículo no tripulado, desde consejos, guías, ámbitos profesionales, tutoriales y ayuda [14].

Es un servicio dedicado a la monitorización de drones más centrado en empresas que se dedican profesionalmente al uso del dron, que a un usuario común. Destaca la cantidad de empresas que trabajan y colaboran con ellos, alrededor de 5000 empresas por todo el mundo.

Dentro de su aplicación móvil se pueden destacar las siguientes funcionalidades:

- **Captura en tiempo real** la posición en un mapa del dron.
- **Permite la planificación de un trayecto**.
- **Muestra en pantalla de información y anotaciones**. Aquellas que pueda publicar el usuario e información meteorológica.
- **Compatible con drones con cámara**. Permite al usuario visualizar en tiempo real lo que captura la cámara.

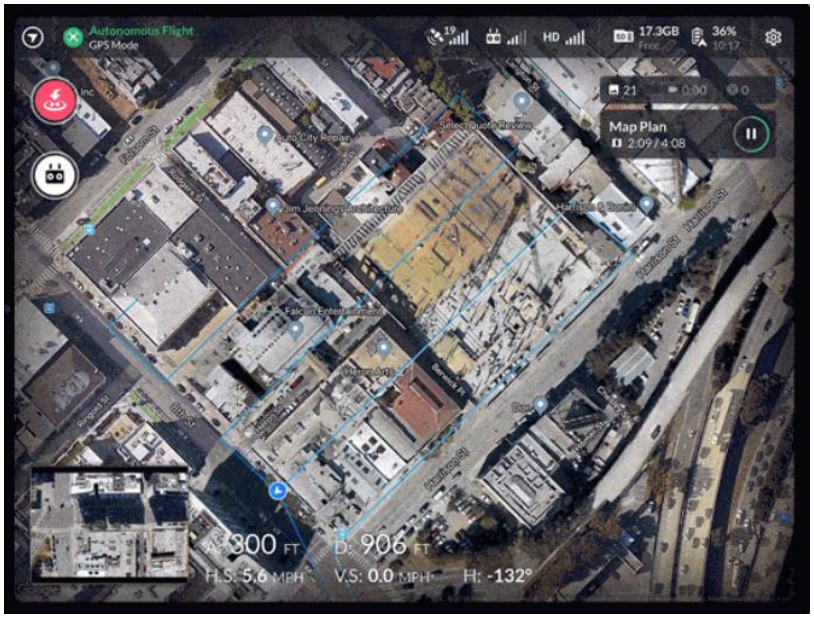


***Ilustración 8**. Captura realizada para planificar trayecto.*

En este caso, DroneDeploy únicamente despliega su servicio de monitorización de un dron mediante una aplicación móvil.

2.4. Crítica al estado del arte

Es innegable la atractiva apariencia y funcionalidad que presentan estos dos servicios, como el control remoto y la planificación de trayectos. Sin embargo, se han observado ciertas carencias en su diseño que pueden dar lugar a replantear una nueva implementación.

Es obvio que un investigador u operador que requiera de una monitorización de un dron pueda emplear ambos productos para realizar un seguimiento de éstos. En cambio, con ambas aplicaciones resulta **imposible la monitorización simultanea de varios drones a la vez**. Esto puede resultar un problema para un operador con varios drones operativos.

En cuanto a sus enfoques, la aplicación AirMap está destinada principalmente a usuarios comunes que desean volar su dron de la forma más precavida y segura posible. En cambio, DroneDeploy, se centra más en un uso profesional enfocado a las empresas, y puede resultar la opción óptima para un operador.

Sin embargo, junto con la imposibilidad de supervisar varios drones a la vez, se añade el inconveniente de no disponer de sitio web para dotar de multiplataforma al producto. Adicionalmente, DroneDeploy consta de un servicio de pago mensual con un elevado coste de 99 dólares para una versión no completa y 299 dólares para la versión completa [15] (precios rebajados por contratación anual). Como es obvio, al poner precio a un producto, su código será cerrado.

Como punto a favor de la plataforma de AirMap, cuenta con la disponibilidad de aplicación web y aplicación móvil de forma totalmente gratuita y con acceso abierto a su código y API (*Application Programming Interface*) [16]. En cambio, tras un breve periodo de prueba, se aprecia un mal rendimiento y falta de optimización en su aplicación web.

En ambos servicios se echa en falta una funcionalidad que se considera imprescindible para este proyecto, y se trata de **la posibilidad de disponer de un historial de trayectos** que ha realizado un dron.

2.5. Tecnologías empleadas

Para el correcto desarrollo de este sistema, es necesario elegir cuales van a ser las tecnologías empleadas, tanto para la comunicación entre el servidor y el dron, así como el software empleado para la interfaz del usuario.

Por un lado, resulta necesario e incluso imprescindible el empleo de una aplicación web (se detallará en los requisitos del sistema) de actualización dinámica de tipo *página única* para la monitorización en tiempo real de un dron.

Por otro lado, se han presentado tres tecnologías distintas con el fin de monitorizar un dron, siendo que cada camino presenta unas ventajas y unas desventajas claras que se van a exponer con el motivo de seleccionar la tecnología más adecuada.

En primer lugar, se ha demostrado que **el método tradicional basado en radios, en el cual** se envían señales a las aeronaves y la espera de una respuesta, resulta eficaz a la hora de llevar un seguimiento. Este método muestra la ventaja de disponer de mucha infraestructura ya establecida y dirigida al funcionamiento seguro de las aeronaves tripuladas, por lo que la mayoría del coste material para llevar a cabo la monitorización se centraría en incorporar el sistema ADS-B en los drones (coste realmente bajo en comparación a otras).

Adicionalmente, incorporar este método de seguimiento dispone del atractivo de unificar todo el control aéreo (tanto de aeronaves tripuladas como no tripuladas) en un mismo bloque.

En cambio, esta tecnología dispone de una mala seguridad ya que el canal no está cifrado y cualquiera puede interceptar la señal. Además, lo interesante de este método consta de disponer de la mayor cantidad de estaciones terrestres capaces de recibir y enviar señales, de forma que recopile la mayor cantidad de información posible. Siendo así, el empleo de este método supone depender al completo de estas organizaciones, como es el caso de *Flightradar24*. Otro aspecto negativo a la hora de emplear este sistema se manifiesta en la necesidad de que todos los drones dispongan de esta característica adicional para que el sistema funcione de forma óptima.

En segundo lugar, con el empleo de la **monitorización por comunicación con RPS y GSM,** se puede apreciar cierta similitud respecto al sistema tradicional con envío de señales *ping*, aunque se diferencian principalmente en la ausencia del GPS, obteniendo una protección extra ante posibles intentos de inhibición o falsificación.

Adicionalmente, con la integración de una tarjeta SIM en el dron resulta sencilla la comunicación, por lo que los drones podrían ser controlados remotamente. Por ejemplo, en el caso de la detección de un par de drones con pronóstico de choque en el espacio aéreo, se podría enviar una señal para ordenar al dron modificar su altura, y de esta forma no colisionarían.

A pesar de parecer un sistema ventajoso, también carece de ciertos aspectos como la cobertura de la que dispone una teleoperadora. Tomando el ejemplo de Vodafone, solo 59 naciones del planeta disponen de cobertura 4G, por lo que en el momento en que un dron no se encuentre en territorio cubierto dejaría de disponer de las funcionalidades del sistema. Además, ocurre lo mismo que en el caso anterior: este tipo de sistemas depende completamente de la entidad organizadora, la cual con mucha probabilidad venda el servicio.

***Ilustración 9**. Cobertura 4G de Vodafone.*

Fuente: https://www.zonamovilidad.es/noticia/10577/noticias-tecnologia-/vodafone-espana-ofrece-cobertura-4g-en-59-destinos-del-mundo.html

Como es obvio, las dos alternativas anteriores tienen una proyección de futuro brillantes en comparación con la que se va a describir dada su enorme cobertura, planes de escalabilidad y unificación de datos. Sin embargo, actualmente no están implementadas, y prácticamente ningún dron ha sido diseñado para estos sistemas.

Finalmente, se presenta el sistema basado en la **comunicación mediante GPS y red celular**. Esta tecnología es en las que están basadas cientos de aplicaciones móviles y servicios web dado el inmenso alcance que dispone la red móvil de Internet. Con esta tecnología no se depende de ninguna entidad organizadora que disponga de miles de estaciones terrestres capaces de localizar un dron, pues un dron con conectividad a Internet se conectará a un servidor que puede estar en cualquier parte del mundo.

El canal de comunicación que va a emplear este sistema está basado en el protocolo de IP. Consecuentemente, se podrán aplicar los cifrados que se consideren oportunos (En función de lo que se quiera proteger). Por ejemplo, no es lo mismo disponer de una comunicación cifrada débil para un dron destinado al control agrícola, que crear un cifrado potente para un dron dedicado a la seguridad de un banco.

Adicionalmente, un gran porcentaje de drones destinados al ámbito profesional disponen de módulos GPS y puertos USB para la posible conexión de un módem 3G/4G/5G. Por lo que, no es necesario disponer de un dron especializado y concreto para disponer de los beneficios del sistema.

2.6. Propuesta

En este proyecto se plantea crear un sistema de monitorización de vehículos no tripulados enfocado a administradores, operadores, e incluso investigadores dedicados al vuelo de los drones.

Se dispone de un dron dotado de conectividad GPS y posibilidad de incorporar un módem de cualquier operador de telefonía celular para otorgar conexión a Internet. El mismo se conectará a un servidor y comenzará a transmitir parámetros obtenidos del módulo GPS cada segundo mediante UDP/IP.

Se va a crear una aplicación web de una única página mediante el *framework* MEAN (MongoDB, Express, Angular y NodeJS) de forma a actualizar dinámicamente una lista de parámetros relacionados con un dron y un mapa en tiempo real con su posición. El sitio web será capaz de monitorizar varios drones al mismo tiempo, de manera que va a poder visualizar todos los trayectos en el mapa, así como permitir centrarse en los parámetros individuales de cada uno.

Adicionalmente, se considera necesaria la funcionalidad de poder revisar y comprobar anteriores viajes de cualquier dron. De esta manera, se podrá comprobar con fecha exacta la posición de un dron, así como su recorrido y los correspondientes parámetros.

3. Análisis del problema

En este capítulo se va a centrar el foco de trabajo en el producto a desarrollar del proyecto, así como su alcance y funcionalidad. De forma que se comenzará estableciendo el cómo se va a abordar el problema. Posteriormente, se detallarán los requisitos tanto funcionales como no funcionales. Finalmente se emplearán los diagramas UML con el fin de representar un sistema, y como los actores principales participan en él [17]. Se van a desarrollar diagramas de clases, diagramas de casos de uso, y los diagramas de actividad del sistema basados en las plantillas actuales de UML [18].

3.1. Especificación de requisitos

En este apartado se va a realizar una descripción completa del comportamiento que va a realizar el sistema. Para ello se van a presentar requisitos funcionales que deberá realizar el sistema cuando un usuario lo solicita, y se van a explicar los requisitos no funcionales, tales como restricciones, niveles de rendimiento, etc.

3.2.1 Requisitos funcionales

En este apartado se van a listar las diferentes funcionalidades más relevantes del sistema. Cada una de ellas se dividirá en función del actor que lo lleve a cabo, y se marcará con un identificador para posteriormente realizar una explicación y establecer una relación entre cada uno de ellos.

- **Usuario** (U):
  - **Localizar en tiempo real los drones activos en vuelo (U-01).**
    El sistema comienza a monitorizar la posibilidad de conexión de algún dron.

  - **Seguir un dron en tiempo real (U-02).**
    El sistema ha detectado el vuelo de uno o varios drones. De forma que, el usuario pueda elegir la supervisión de uno de ellos

  - **Consultar anteriores trayectos (U-03).**
    El sistema lista en el sitio web, con algunos parámetros los diversos trayectos que han realizado todos los drones que han sido monitorizados.

  - **Revisar un trayecto (U-04).**
    El sistema dibuja en el mapa mediante una línea el trayecto que ha realizado un dron en una fecha determinada. Conforme la línea avanza, muestra al usuario datos tales como coordenadas, altura y velocidad.

- **Pausar/continuar el visualizado de la línea (U-05).**
  El sistema permite al usuario parar la animación de la línea que representa el trayecto de un dron. Asimismo, el usuario puede contemplar parámetros en el momento que desee. Igualmente puede detener la animación, y el sistema permite continuar el visualizado de la misma sin necesidad de volver a mostrar todo el trayecto.

- **Cambiar estilo del mapa (U-06).**
  El sistema permite al usuario cambiar el estilo del mapa en cualquier momento.

- **Consultar información (U-07).**
  En el inicio de la página web se encuentra una presentación del servicio que el usuario puede consultar en cualquier momento.

- **Dron** (D):

  - **Enviar datos al servidor (D-01).**
    En el momento en el que el dron se active, este debe enviar los datos obtenidos por el módulo GPS al servidor.

- **Sistema** (S):

  - **Obtener todos los parámetros geográficos (S-01).**
    El sistema podrá obtener mediante petición GET dirigida al API Rest todas las coordenadas, velocidades y alturas almacenadas en la base de datos, de manera que pueda disponer de ellas para cualquier operación.

  - **Obtener los drones que han volado recientemente (S-02).**
    El sistema, con los datos geográficos de los últimos 10 segundos, será capaz de obtener los drones que han sido activados.

  - **Obtener los últimos parámetros geográficos (S-03).**
    El sistema obtiene, mediante una petición GET dirigida al API Rest, los datos geográficos recientes (últimos 10 segundos).

  - **Obtener parámetros geográficos de un dron (S-04).**
    El sistema será capaz de ir recopilando datos recibidos de un dron e ir mostrándolos en pantalla al usuario.

  - **Obtener trayectos (S-05).**
    El sistema será capaz de reconocer y agrupar en un mismo bloque aquellas coordenadas pertenecientes al mismo rango temporal.

3.2.2 Requisitos no funcionales

- ✓ **Efectividad**. Se desea el equilibrio entre la eficacia y la eficiencia de manera que el sistema deberá ser capaz de proporcionar al usuario un servicio web totalmente optimizado y eficaz, permitiendo la monitorización simultanea de todos los drones necesarios.
- ✓ **Escalabilidad.** El sistema parte recopilando ciertos tipos de datos. Sin embargo, con vistas al futuro, el sistema es capaz de incorporar nuevos datos en función de las nuevas necesidades, así como permitir la monitorización de más unidades de drones
- ✓ **Usabilidad.** El sistema resulta realmente sencillo e intuitivo. Gracias a su diseño minimalista, aparecerán muy pocos elementos en pantalla, de manera que el usuario sabrá en cualquier momento lo que está ocurriendo.
- ✓ **Disponibilidad.** El sistema deberá estar funcionando en todo momento; esto es posible dado el bajo consumo de recursos del sistema. Si fuera crucial disponer en cualquier momento de la información ante posibles caídas del servidor, sería posible establecer un servidor secundario que actúe en caso de fallo.
- ✓ **Seguridad.** El sistema puede resultar lo más seguro posible en función del ámbito que se vaya a destinar. El proyecto está destinado a investigadores u operadores de drones para un uso normal. En caso de ser necesario un sistema robusto, sería tan sencillo como fortalecer la integridad de datos de la base de datos, así como mejorar los sistemas de cifrado en la comunicación.
- ✓ **Mantenibilidad.** El software del sistema se implementará empleando un mecanismo de control de versiones. Uno de los más populares con una inmensa comunidad es *Git*, el cual permite facilitar el trabajo en equipo, disponer de un mantenimiento en el tiempo, y disponer de una versión estable.
- ✓ **Portabilidad.** Uno de los puntos fuertes de este proyecto se basa en su fácil accesibilidad. Únicamente será necesario disponer de un dispositivo con acceso a Internet y un navegador web.

3.2. Actores

En los diagramas UML, un actor implica un rol jugado por un usuario o cualquier otro sistema que interactúa con el sujeto [19]. Un actor puede estar representado por un ser humano, hardware externo, incluso un rol de alguna actividad. A continuación, se van a presentar los actores encargados de cumplir con los requisitos anteriormente nombrados.

- **Usuario**. Se trata del administrador, operador o investigador encargado de llevar a cabo la supervisión de los drones. Desde la página web podrán monitorizar parámetros y la posición real de un dron, así como un historial de trayectos.
- **Dron**. Consiste en el conjunto de uno o varios vehículos voladores no tripulados encargado de transmitir datos al servidor, ya sean reales o simulados.
- **Sistema**. Será el encargado de recopilar y mostrar a disposición del usuario, los datos correspondientes a un dron.

3.3. Diagramas de clases

Dentro de los diagramas UML, el diagrama de clases es uno de los más útiles, debido a que en ellos se establece claramente la estructura de un sistema en el momento de implantar sus clases, atributos, operaciones y relaciones entre objetos [20]. Este diagrama ofrece un esquema cuyo fin es mejorar la visión general de la funcionalidad del sistema.

Resultan de especial utilidad para expresar de una manera visual e intuitiva las necesidades específicas de un sistema, de manera que cualquier empleado de la empresa pueda entenderlo. Además, resulta ser una de las técnicas de análisis más recomendables para el inicio de un proyecto.

Previo a la explicación del diagrama expuesto, se va a proceder a una breve explicación de los componentes generales de un diagrama de clases general.

- **Clase**. Representa un objeto o conjunto de objetos que comparten una estructura y unos comportamientos comunes. Se representa con un rectángulo, y en su contenido se puede apreciar el nombre de la clase, una serie de atributos y en algunos casos operaciones.
- **Interacciones**. Para relacionar clases entre sí, generalmente se emplean las **asociaciones** (aunque, de manera más específica, existan otras maneras de conectar clases, como la herencia, composición, etc.). En cada asociación se puede asignar un valor a cada extremo que conecta, de forma que se indica la cantidad de objetos que se asocia con otro. En la explicación del diagrama se entrará en detalle.
- **Claves primarias y foráneas**. Cada uno de los atributos de una clase pertenecen a un tipo de datos como número, cadena, etc. En cambio, existe el tipo de datos conocido como clave primaria que sirve para diferenciar un objeto concreto de otro; en otras palabras, sirve como identificador. Asimismo, existe otro tipo de datos conocido como clave foránea, y se emplea para referirse al atributo primario de otra clase.

  Por ejemplo, una clase hotel tendrá un identificador de tipo clave primaria, que lo diferenciará de los demás. También, existe la clase habitación, y dentro de la misma existe un atributo hotel. Por eso, para relacionar el atributo con clave primaria de la clase hotel, será necesario crear un atributo de tipo clave foránea en la clase habitación.

Con el fin de obtener una visión general del esquema del sitio web, se presenta el siguiente diagrama de clases. En él se pueden contemplar tres clases distintas.

[1] **Dron**. Representa al dispositivo que se va a conectar al servidor y enviar datos. Como atributo, únicamente se va a considerar su identificativo de clave primaria.
[2] **Coordenadas**. Consiste en los datos que un dron va a enviar a el servidor. Se pueden destacar los de latitud, longitud, altura, _id como clave primaria, velocidad y dron de tipo clave foránea, para conectar el identificador del dron con su respectiva coordenada.
[3] **Trayecto**. Agrupación de coordenadas que dan significado a un viaje de un dron en un espacio limitado de tiempo. Cada trayecto incluye un identificador de clave primaria, una fecha de inicio, una fecha de fin, unas coordenadas inicio, unas coordenadas fin, y un dron de tipo clave foránea.

En el diagrama se pueden apreciar varias asociaciones, las cuales se van a detallar. Todo comienza con un dron, cada uno de ellos puede disponer desde 0 a infinitas coordenadas (dependerá si ha salido a volar o no). De la misma manera, un dron puede disponer de un trayecto o no, en función si ha volado o no.

En el caso de un trayecto, únicamente puede pertenecer a un dron y dispone desde 1 hasta infinitas coordenadas. Finalmente, para las coordenadas únicamente puede pertenecer a un trayecto y a un único dron.

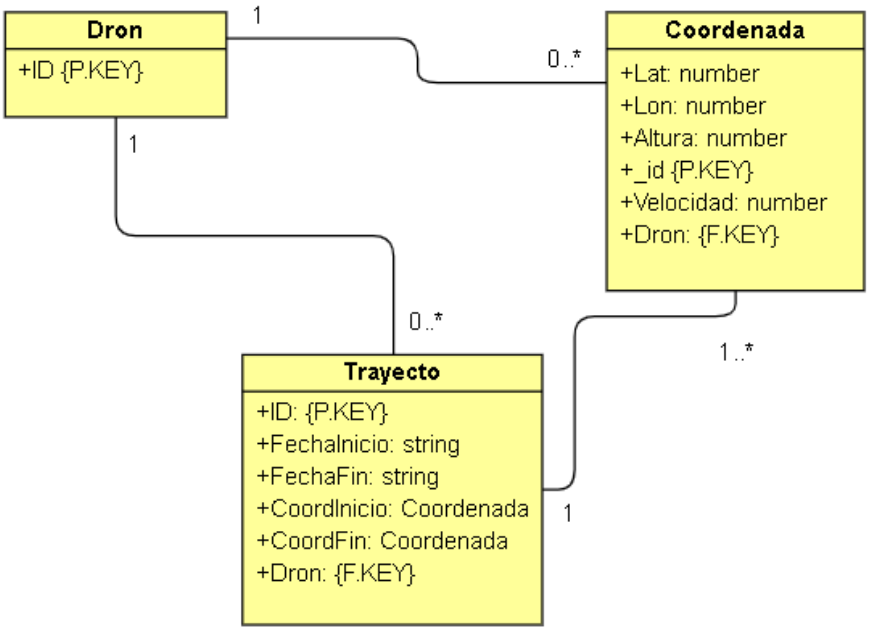


***Ilustración 10**. Diagrama de clases de la aplicación web.*

3.4. Diagrama de casos de uso

Otro de los tipos de diagramas empleados a la hora de mostrar el comportamiento de un sistema, se denomina diagrama de casos de uso, y su funcionalidad se basa en relacionar un actor con una acción posible del sistema [21]. De esta manera se obtiene una descripción de las acciones desde el punto de vista del usuario. Un diagrama de casos de uso se caracteriza por el uso de los siguientes elementos:

- **Actores**. Cumplen la misma función que en los diagramas anteriores. En resumen, se trata de un usuario, hardware o cualquier objeto que cumpla un rol en el sistema.
- **Casos de uso**. Se representa con un círculo ovalado, e implica la realización de una operación o tarea específica llevaba a cabo por un actor.
- **Relaciones**. Sirven para relacionar un caso de uso con su correspondiente actor, así como relacionar casos de uso entre sí.
    - Relación de actor con caso de uso. Se representan con una línea simple uniendo un caso de uso con el correspondiente actor que lo realiza.
    - Relación de un caso de uso con otro caso de uso. Existen dos tipos, la relación **incluir** indica que un caso de uso es necesitado por otro para poder cumplir su tarea, y la relación **extender** que implica opciones para un cierto caso de uso.

Posteriormente, se va a exponer el diagrama de casos de uso para el sistema de monitorización de vehículos no tripulados y, al final de este, se mostrará una explicación del mismo junto con un detallado listado de los casos de uso.

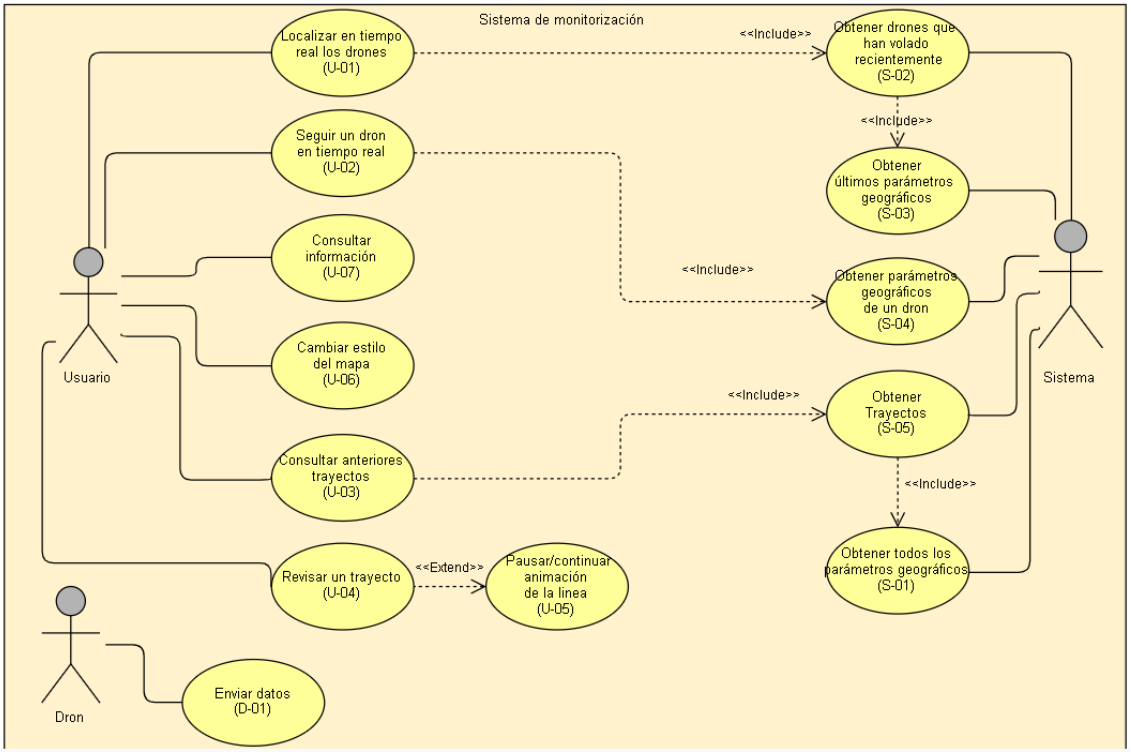


***Ilustración 11**. Diagrama de casos de uso.*

Localizar los drones en tiempo real (U-01)	
Actores	Usuario
Descripción	El usuario desea comenzar a localizar los drones disponibles para la monitorización en tiempo real.
Entradas	- Pulsación en el apartado tiempo real de la barra de navegación - Pulsación en el botón ¡de comienza ya! Localizado en la parte de información
Procesos	El sitio web muestra el componente dedicado a la monitorización en tiempo real.
Salidas	Se muestra al usuario un mapa y un espacio destinado a la información enviada por el dron.
Extiende	
Incluye	El caso de uso de "Obtener coordenadas recientes" **(S-02**) llevado a cabo por el Sistema.

Tabla 1**. Caso de uso **Localizar los drones en tiempo real (U-01).

Seguir un dron en tiempo real (U-02)	
Actores	Usuario
Descripción	Permite al usuario seguir en un mapa a un dron y conocer sus parámetros.
Entradas	Cuando se detecta el movimiento de un dron, aparece un botón a vista del usuario para el seguimiento de este. Por lo que, se accionará pulsando clic en él.
Procesos	Mediante la actualización dinámica se actualiza la posición del mapa y se actualizan datos.
Salidas	Se centra el mapa en unas posiciones geográficas y se muestra un marcador con la posición real de un dron dejando rastro de una línea indicando su trayectoria. Adicionalmente, se muestra un listado actualizado de parámetros del dron.
Extiende	
Incluye	El caso de uso de "Obtener parámetros geográficos de un dron" **(S-04**) llevado a cabo por el Sistema.

Tabla 2**. Caso de uso **Seguir en tiempo real los drones (U-02).

Consultar anteriores trayectos (U-03)	
Actores	Usuario
Descripción	El usuario dispone de un listado histórico de los trayectos de todos los drones registrados en el sistema.
Entradas	- Pulsación en el apartado trayectos de la barra de navegación - Pulsación en el botón ¡visualiza los trayectos! Localizado en la parte de información
Procesos	El sitio web muestra el componente dedicado a los anteriores trayectos.
Salidas	El usuario visualiza en pantalla un mapa no centrado y un historial de trayectos de los drones (con algunos datos como fechas, coordenadas, etc.).
Extiende	

Incluye	El caso de uso de "Obtener trayectos" **(S-05**) llevado a cabo por el Sistema.

Tabla 3**. Caso de uso **Consultar anteriores trayectos (U-03).

Revisar un trayecto (U-04)	
Actores	Usuario
Descripción	Proporciona al usuario la posibilidad de visualizar un anterior trayecto, con sus respectivos parámetros.
Entradas	- Pulsación en el botón *Go* para uno de los trayectos.
Procesos	Comienza la animación del trayecto de un dron, y la actualización de datos.
Salidas	En la parte del mapa, se visualiza el animado trazado simulando el trayecto que realizó el dron. A su lado, el usuario puede ver datos relacionados con el dron conforme pasa el tiempo.
Extiende	El caso de uso de "Obtener coordenadas recientes **(U-05**) llevado a cabo por el Sistema.
Incluye	

Tabla 4**. Caso de uso **Revisar un trayecto (U-04).

Pausar/continuar animación de la línea (U-05)	
Actores	Usuario
Descripción	Permite al usuario detener y continuar reproduciendo el trayecto de un dron.
Entradas	- Pulsación en el botón *Pause* o *Continue* de un trayecto.
Procesos	El componente para en el tiempo el trayecto de un dron.
Salidas	El usuario puede observar que la animación del trayecto para y los datos en ese momento del tiempo. Por lo que, el usuario puede parar en cualquier momento el recorrido del trayecto y visualizar parámetros.
Extiende	
Incluye	

Tabla 5**. Caso de uso **Pausar/continuar animación de la línea (U-05).

Cambiar estilo del mapa (U-06)	
Actores	Usuario
Descripción	Capacita al usuario de poder cambiar el estilo que muestra el mapa.
Entradas	- Pulsación en la selección de algún tema.
Procesos	Se modifica el parámetro *style* de la instancia del mapa.
Salidas	Cambia el estilo del mapa.
Extiende	
Incluye	

Tabla 6**. Caso de uso **Cambiar estilo del mapa (U-06).

Consultar información (U-07)	
Actores	Usuario
Descripción	Permite al usuario saber, de forma genérica, el ámbito del sistema.
Entradas	- Pulsación en el logo de la web - En el momento de acceder a la página web (al tratarse de la página inicial)
Procesos	Muestra al usuario el componente de información
Salidas	Aparece información estructurada en bloques de la funcionalidad del sistema.
Extiende	
Incluye	

Tabla 7**. Caso de uso **Consultar información (U-07).

Enviar datos (D-01)	
Actores	Dron
Descripción	Un dron con acceso a internet transmite los datos del módulo GPS al servidor.
Entradas	Siempre que esté activo, el dron mandará datos.
Procesos	La interfaz de red del dron se conectará al *socket* servidor del sistema.
Salidas	Una vez recibidos por parte del servidor, los almacena en la base de datos.
Extiende	
Incluye	

Tabla 8**. Caso de uso **Enviar datos (D-01).

Obtener todos parámetros geográficos (S-01)	
Actores	Sistema
Descripción	El sistema obtiene todos los datos enviados por los drones a lo largo de la historia del sistema de monitorización (o restablecimiento de la base de datos)
Entradas	
Procesos	Enviar una solicitud GET al API Rest del servidor *backend* para obtener todo su contenido.
Salidas	Listado completo de datos concernientes al historial de drones.
Extiende	
Incluye	

Tabla 9**. Caso de uso **Obtener todos parámetros geográficos (S-01).

Obtener los drones que han volado recientemente (S-02)	
Actores	Sistema
Descripción	El sistema, mediante la observación de las últimas coordenadas, puede determinar los drones activos.
Entradas	Listado de parámetros geográficos durante los últimos 10 segundos.
Procesos	Se determina la identidad del dron de unos parámetros geográficos.
Salidas	Listado de los drones registrados durante los 10 segundos anteriores a su llamada.
Extiende	
Incluye	El caso de uso de "Obtener los últimos parámetros geográficos" **(S-03**) llevado a cabo por el Sistema.

Tabla 10**. Caso de uso **Obtener los drones que han volado recientemente (S-02).

Obtener los últimos parámetros geográficos (S-03)	
Actores	Sistema
Descripción	El sistema obtiene un conjunto de datos relacionados con el vuelo de todos los drones en los últimos 10 segundos.
Entradas	
Procesos	El sistema realiza una petición GET al API Rest para obtener los últimos parámetros captados.
Salidas	Listado de parámetros captados en los últimos 10 segundos por la base de datos.
Extiende	
Incluye	

Tabla 11**. Caso de uso **Obtener los últimos parámetros geográficos (S-03).

Obtener parámetros geográficos de un dron (S-04)	
Actores	Sistema
Descripción	El sistema recopila los datos relacionados con un dron, para su seguimiento en tiempo real.
Entradas	
Procesos	El sistema almacena las coordenadas que va realizando los drones activos.
Salidas	Listado de datos geográficos relacionados con un dron.
Extiende	
Incluye	

Tabla 12**. Caso de uso **Obtener parámetros geográficos de un dron (S-04).

Obtener trayectos (S-05)	
Actores	Sistema
Descripción	El sistema agrupa en bloques la información recibida por los drones, para dotarles de un espacio temporal y asignarles la propiedad de trayecto.
Entradas	Listado completo del historial de datos geográficos.
Procesos	Agrupar por fecha los datos.
Salidas	Agrupación de datos geográficos en un espacio temporal próximo
Extiende	
Incluye	El caso de uso de "Obtener todos los parámetros geográficos" **(S-01**) llevado a cabo por el Sistema.

Tabla 13**. Caso de uso **Obtener trayectos (S-05).

3.5. Diagrama de actividad

El diagrama de actividad es considerado como un esquema de comportamiento de un producto, dado que describen lo que debería suceder en el sistema que se está modelando. De manera que permite describir los pasos realizados en un caso de uso, e ilustrar un flujo de trabajo entre los actores y el sistema. Algunos de los elementos más comunes en un diagrama de flujo son:

- **Acción**. Representa un paso en el que los actores realizan una tarea.
- **Nodo de decisión**. Sirve para indicar la bifurcación en diferentes ramas, en función de la decisión tomada. Siempre disponen de una única entrada y dos o más salidas.
- **Flujos de contro**l. Sirven para establecer el flujo entre los diferentes pasos.
- **Nodo inicial y final**. Se emplean para indicar el inicio y final del flujo.

A continuación, se va a mostrar el flujo correspondiente a la hora de monitorizar un dron en tiempo real y el flujo relacionado con el historial de trayectos.

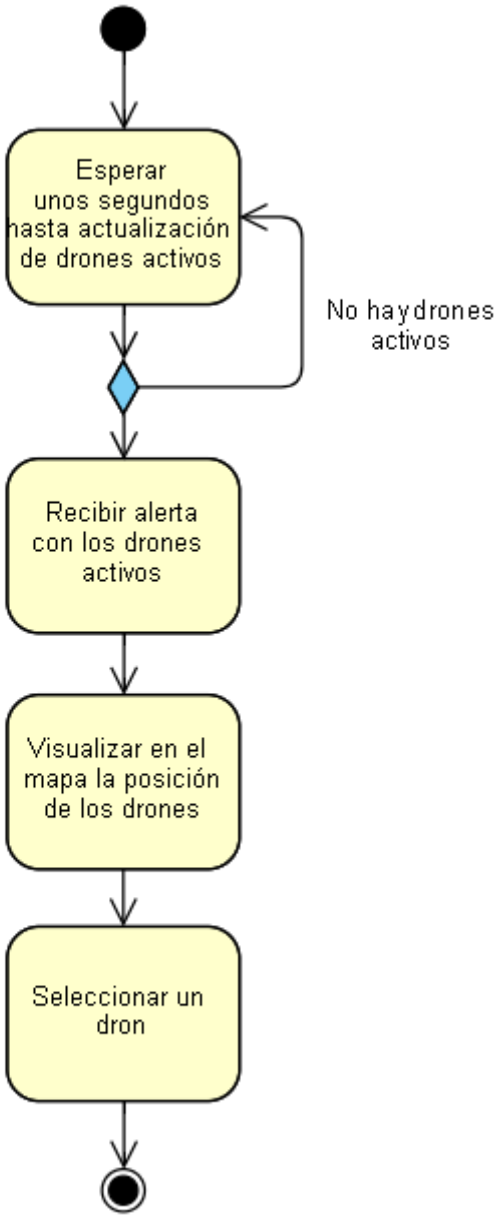


***Ilustración 12**. Diagrama de actividad de monitorizar un dron en tiempo real.*

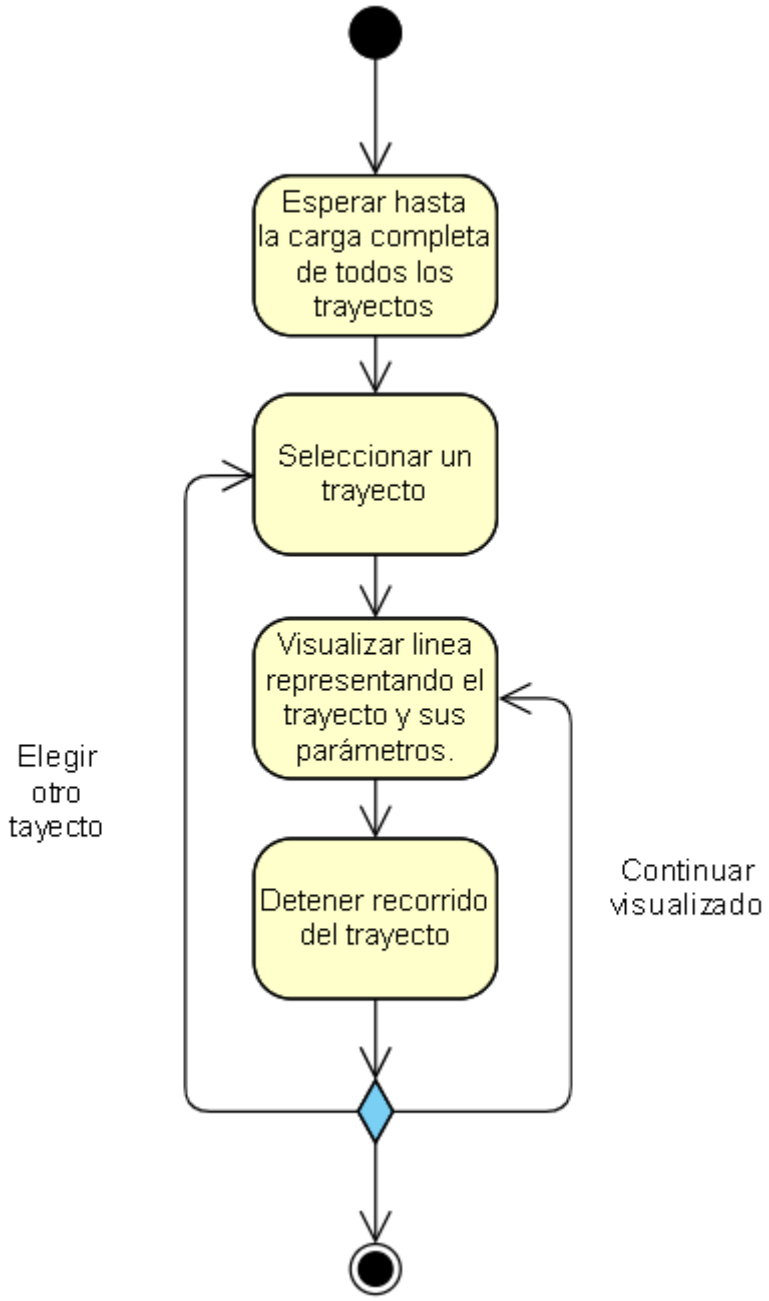


***Ilustración 13**. Diagrama de actividad de revisar un trayecto.*

3.6. Identificación y análisis de soluciones posibles

Actualmente, gracias al avance informático, resulta increíble el gran abanico de posibilidades que se abren a la hora del desarrollo de un sistema de este calibre.

En este tipo de proyectos, basados en la monitorización en tiempo real de un objeto volador, resulta de especial interés la posibilidad de disponer de los beneficios del sistema en cualquier lugar. De forma que un operador que desea supervisar el vuelo del dron pueda acceder a la información en cualquier lugar y con la mayor cantidad de dispositivos posibles.

Con una aplicación móvil, se podría acceder en cualquier lugar (con acceso a Internet) con un dispositivo móvil. En cambio, hay ocasiones donde los vuelos del dron se desarrollan y se planifican en una oficina, por lo que resulta útil la supervisión de estos desde un ordenador. En resumidas cuentas, la implementación de una aplicación web resulta más portable que una aplicación móvil.

Una de las opciones más tradicionales y con mayor capacidad de personalización es la creación del sitio web con las tecnologías HTML, JavaScript y CSS plano. Esto quiere decir que, la totalidad de la aplicación podría llevarse a cabo sin la necesidad de un *framework*.

Sin embargo, realizar un proyecto web sin *framework* supone dificultar la posibilidad de cumplir ciertos requisitos funcionales que, previamente, se han considerado cruciales. Emplear un *framework* sirve principalmente para facilitar el trabajo en equipo, y dotar una estructuración a la página web, permitiendo al programador centrarse puramente en el desarrollo (mejorando su efectividad). Además supone, a medio/largo plazo, la posibilidad de un mantenimiento de código ágil, ampliaciones de todo tipo, organización intuitiva, etc.

3.7. Propuesta

En primer lugar, se va a desarrollar un servidor basado en *Sockets*, el cual se encargará de recibir los datos enviados por un dron y agruparlos en bloques para insertarlos en una base de datos. El lenguaje de programación empleado es Java dada su simplicidad y facilidad para conectarse a la base de datos.

La colección de datos se almacenará y se controlará desde la misma máquina del servidor, y se diseñará con la tecnología de MongoDB debido a su facilidad de operabilidad con el lenguaje de programación con el que estará diseñada la página web, JavaScript.

Con el estudio del contexto tecnológico se ha observado que la mayoría de las aplicaciones están centradas en las móviles, por lo que se ha considerado necesario la elaboración exclusiva de un sitio web para que la monitorización pueda ser visualizada en cualquier dispositivo.

El servicio web se desarrollará con el novedoso *framework* **MEAN Stack** debido a su gran comunidad, uso de código abierto y gratuito, y su facilidad para la implementación de sitios web dinámicos. Dicho *framework* se caracteriza por el empleo de cuatro tecnologías distintas que se intercomunican de manera intuitiva, obteniendo, además, los siguientes beneficios [22].

- **MongoDB**. Es un sistema de bases de datos no SQL basado en la notación JSON (notación simple de objeto tipo JavaScript). Al ser una base de datos orientada a documentos permite una rápida manipulación, transferencia de datos y excelente escalabilidad [23].

- **ExpressJS**. Es un *framework* basado en las aplicaciones web para NodeJS. Es rápido, de código poco verboso y facil de usar junto a un servidor [24]. Sirve para poder manejar peticiones o solicitudes realizadas mediante el protocolo HTTP (GET, POST, etc.)

- **AngularJS**. Es un *framework* de código abierto mantenido por Google basado en JavaScript que se centrará en la parte *frontend* de la aplicación web, por lo que será la parte visible al usuario. Ofrece muchas ventajas como la organización del código por capas y la facilidad de escalabilidad [25].

- **NodeJS.** Es un *runtime* elaborado con código JavaScript y es el encargado del funcionamiento del servidor.

4. Diseño de la solución

Una vez realizado el análisis del problema, el siguiente paso consiste en focalizar los esfuerzos en el esquema del proyecto de forma que se justifique cómo van a interactuar los distintos componentes entre sí, y la tecnología empleada en cada uno de ellos.

4.1. Arquitectura del sistema

En este apartado se van a presentar los distintitos componentes y subsistemas pertenecientes al proyecto, y detallar cómo se relacionan entre sí. De manera simple, el funcionamiento del sistema se resume a la transmisión de información geográfica (obtenida por un módulo GPS) llevada a cabo por un dron con destino a un servidor, el cual servirá de intermediario para insertar los datos en la base de datos. Adicionalmente, se desplegará un servicio web al que se le conectaran los usuarios para visualizar la monitorización.

Posteriormente, se va a dividir este esquema con el fin de explicar la conexión que establece un usuario con el servidor web, y la conexión que establece un dron con el servidor intermedio.

En ambas arquitecturas, se emplea el modelo **cliente/servidor**, el cual se caracteriza por la existencia de una máquina, comúnmente, encargada de almacenar y procesar datos (servidor), y una estación de trabajo que solicita servicios a dicho servidor (cliente) [26]. En cada uno de los apartados se va a poder observar el papel del cliente y del servidor.

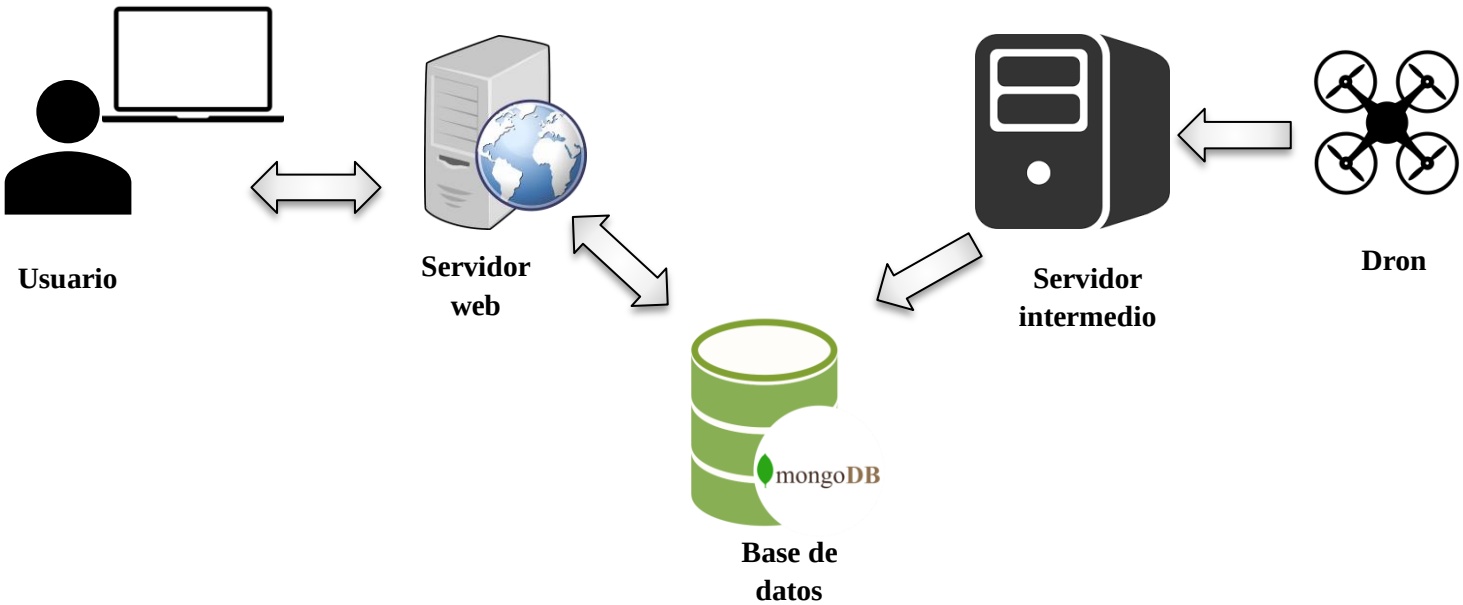


***Ilustración 14**. Esquema general proyecto.*

4.1.1. Arquitectura de la conexión dron-servidor intermedio

Con el fin de que un usuario disponga en tiempo real de información de un dron, es necesario establecer la conexión entre el mismo y un servidor. Para ello, el servidor despliega un servicio basado en *sockets* en su dirección IP y puerto de 5555, al que un dron se conecta. Cada segundo que transcurre, el dron transmite, mediante el protocolo UDP, los datos obtenidos por el módulo GPS, y el servidor los almacena temporalmente. Posteriormente el servidor insertará dicha información en la base de datos para su posterior procesamiento por parte del servidor web.

En el apartado de Tecnología empleada se abarcará más acerca de cómo se transmiten los datos, y cómo los almacena el servidor antes de insertarlos en la base de datos.

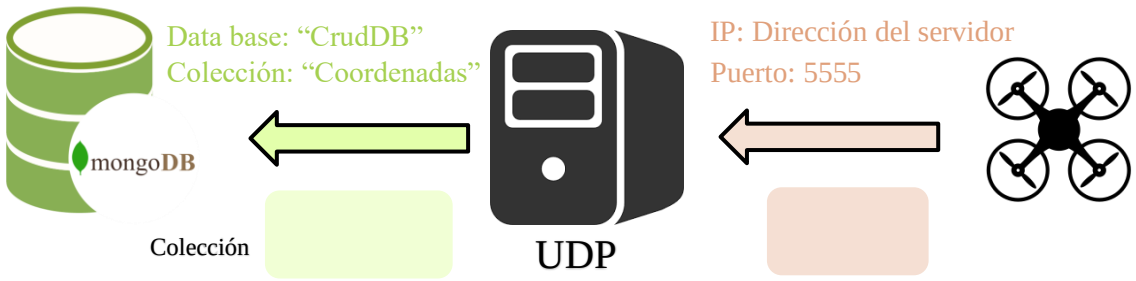


***Ilustración 15**. Esquema conexión dron-servidor.*

4.1.2. Arquitectura de la conexión usuario-servidor web

Una vez los datos están correctamente almacenados en la base de datos, el usuario podrá visualizarlos en la página web. En este caso, la arquitectura cliente/servidor se basará en el modelo del diseño del software de programación por capas, y su funcionamiento es diferente y algo más complejo que la conexión del dron con el servidor, pues en este caso el usuario y el servidor tomarán otro papel en la arquitectura cliente/servidor al tratarse de un servicio web [27].

- **Cliente**. Además de realizar la consulta al servidor, será el encargado de ejecutar parte del código necesario para el correcto funcionamiento del sistema. Esta parte se corresponde al código *frontend*, y fue diseñada para liberar parte de carga al servidor, y es la encargada de realizar pequeñas operaciones, principalmente operaciones de comunicaciones con el servidor.
- **Servidor**. Se encargará de responder a las peticiones del cliente empleando una base de datos si fuera necesario. Su código se denomina *backend* y se caracteriza por ser el encargado de la ejecución de la mayor parte del código del sistema; existe la posibilidad de ejecución en varias máquinas.

Se denomina **programación por capas** a la arquitectura cliente/servidor cuyo principal objetivo es separar las capas de presentación, negocio y datos. Su mayor ventaja reside en el desarrollo a diferentes niveles de un sistema de forma que, si se produce un error en una capa, no afectará a la otra. A continuación se explicará la función de cada una de las capas [28].

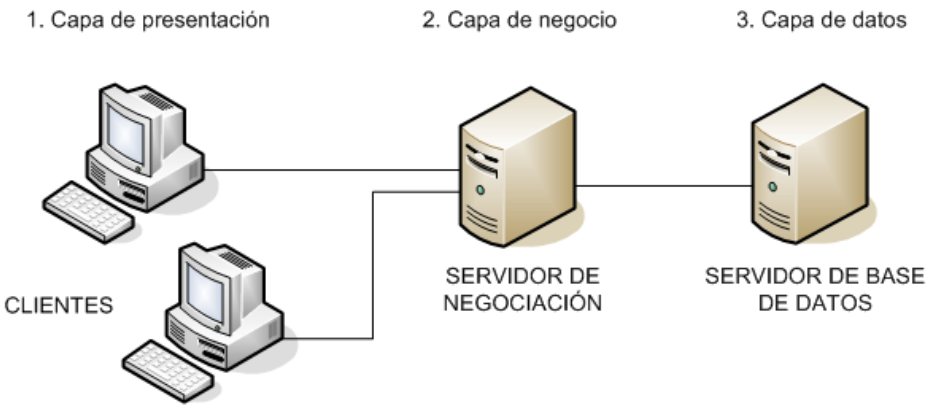


***Ilustración 16**. Responsable de cada capa.*

Dentro de esta arquitectura se puede considerar el patrón **MVC** (Modelo Vista Controlador), el cual permite el desarrollo mejorado de sitios web [29]. Este patrón se caracteriza principalmente por su facilidad a la hora de incorporar o modificar algún aspecto a la aplicación. Concretamente, permite dividir los elementos de la aplicación web en función de la responsabilidad y funcionalidad que estos tengan. Esto implica que, cuando se realice un cambio en alguna parte del código, éste no afecte a los demás.

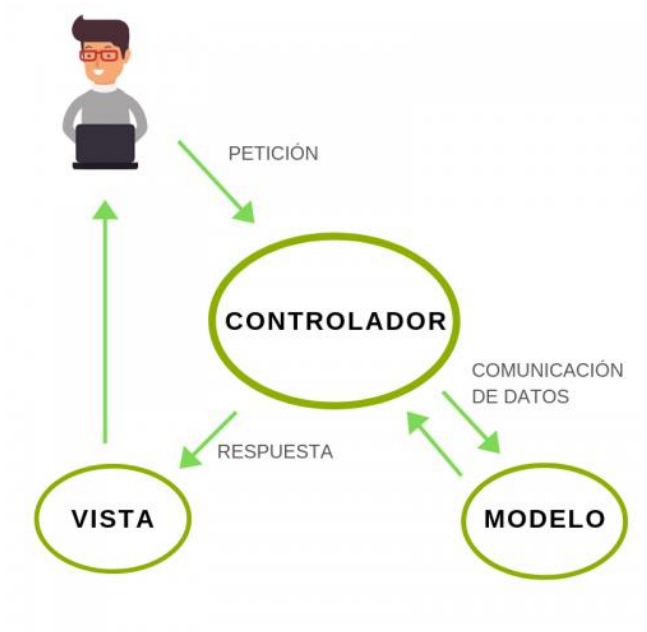


***Ilustración 17**. Esquema arquitectura MVC.*

Capa de presentación (vista)

También conocida como la capa de usuario, su función consiste en mostrar al usuario el sistema, comunicarle información y, en algunos casos, capturar información.

Principalmente se centra en la comunicación entre el cliente y el servidor de manera que en ella se traten aspectos tales como la sintaxis y la semántica de los datos transmitidos, ya que diferentes navegadores pueden tener distintas maneras de manejarlas [30].

Capa de negocio (controlador)

Es la capa encargada de ejecutar la mayor parte del código, pues se dedica a recibir las peticiones de un usuario, procesarla y enviar la respuesta. Consiste en la lógica que realiza las funciones principales de una aplicación como procesamiento de datos, cumplimiento de reglas, etc. [31].

Se trata de la capa intermedia de las tres, pues necesita conectarse con la capa de presentación para recibir y responder a las peticiones, y con la capa de datos para consultar información.

Capa de datos (modelo)

Es la capa dedicada al almacenaje y acceso a los datos de un sistema. Pueden estar formados por uno o varios gestores de bases de datos que realizan operaciones sobre la colección de datos. En resumen, es la encargada de realizar las consultas a la base de datos solicitadas por la capa de negocio.

4.2. Interfaz web

En esta sección se va a realizar el estudio del diseño de la interfaz web del proyecto. Concretamente, se mostrarán las diferentes ventanas, y se detallará cómo interactúan entre sí. Para ello se va a emplear una herramienta online llamada *mockflow* (https://mockflow.com/app/#Wireframe), la cual se basa en la creación de *mockups* para todo tipo de plataformas.

Al inicio de la aplicación web se mostrará al usuario una ventana de **información general** con información relacionada con el proyecto. En ella se reproducirá un video demo de fondo con fotogramas del vuelo de un dron junto a su monitorización. Además, aparecerán unos cuadros de información con las funcionalidades del sistema.

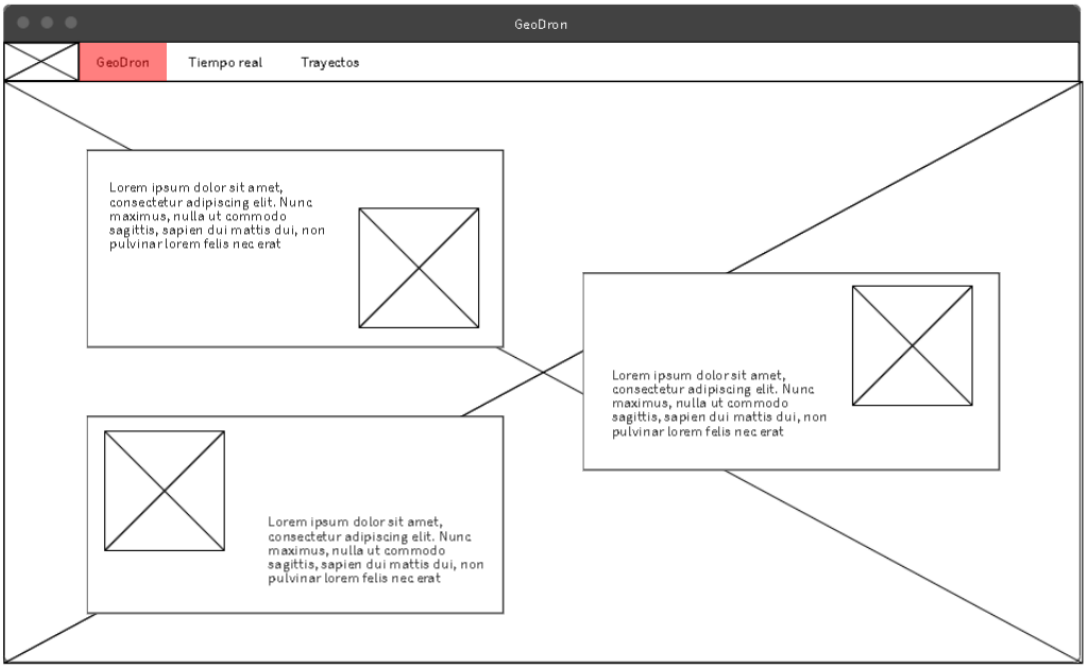


***Ilustración 18**. Ventana de información.*

Posteriormente, el usuario puede acceder a dos de las principales funcionalidades del sistema de monitorización.

En primer lugar, la ventana de **tiempo real** se emplea para la monitorización de uno o varios drones en el mismo instante de tiempo. Nada más acceder, el sistema comenzará a captar los drones con vuelo activo, mostrando en la parte web un elemento indicando el progreso.

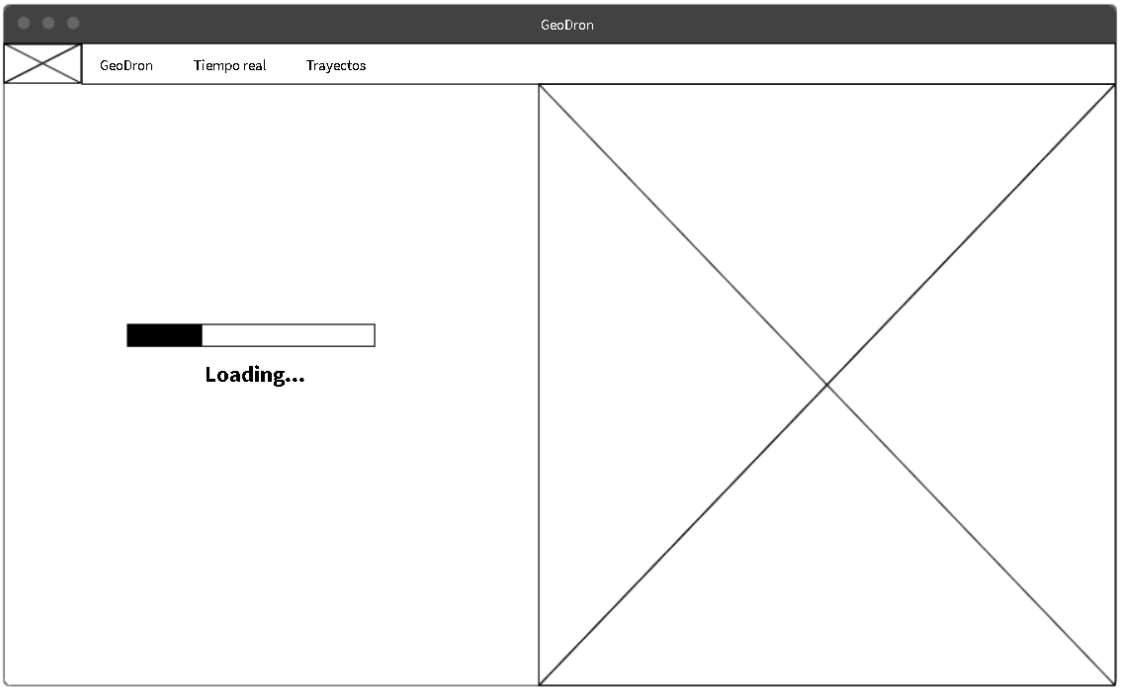


***Ilustración 19**. Ventana de tiempo real (cargando).*

Una vez cargados los datos, se mostrará al usuario una alerta para cada dron que se ha detectado en el sistema de monitorización. En la parte izquierda aparecerá un botón por cada dron disponible. En la parte derecha el mapa se centrará en un punto intermedio entre las posiciones de los drones, de esta forma se comenzará a ver a las posiciones y trayectos que están realizando.

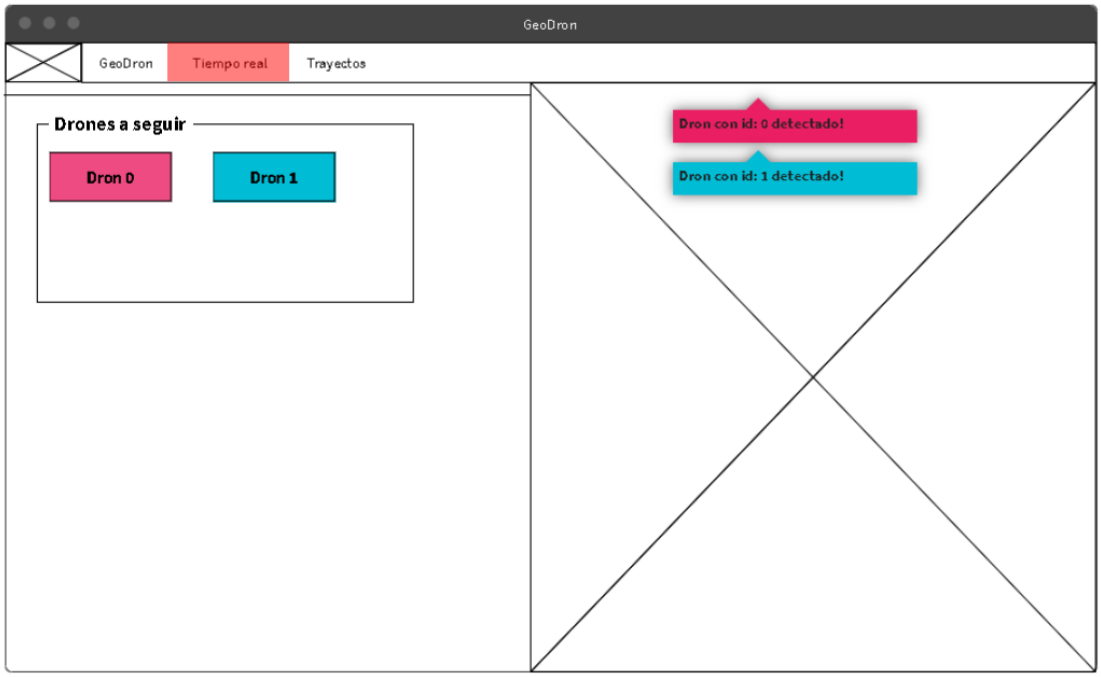


***Ilustración 20**. Ventana de tiempo real (detección de drones).*

Cuando se pulsa el botón de seguimiento de un dron, justo debajo aparecerá una ventana con los parámetros enviados por el mismo, por ejemplo, latitud, longitud, velocidad, etc. Adicionalmente, el mapa se centrará, para cada actualización de datos, en el punto geográfico donde se encuentra el dron. Además, conforme este avanza, irá dejando tras su paso una línea representando el trayecto que ha ido realizando.

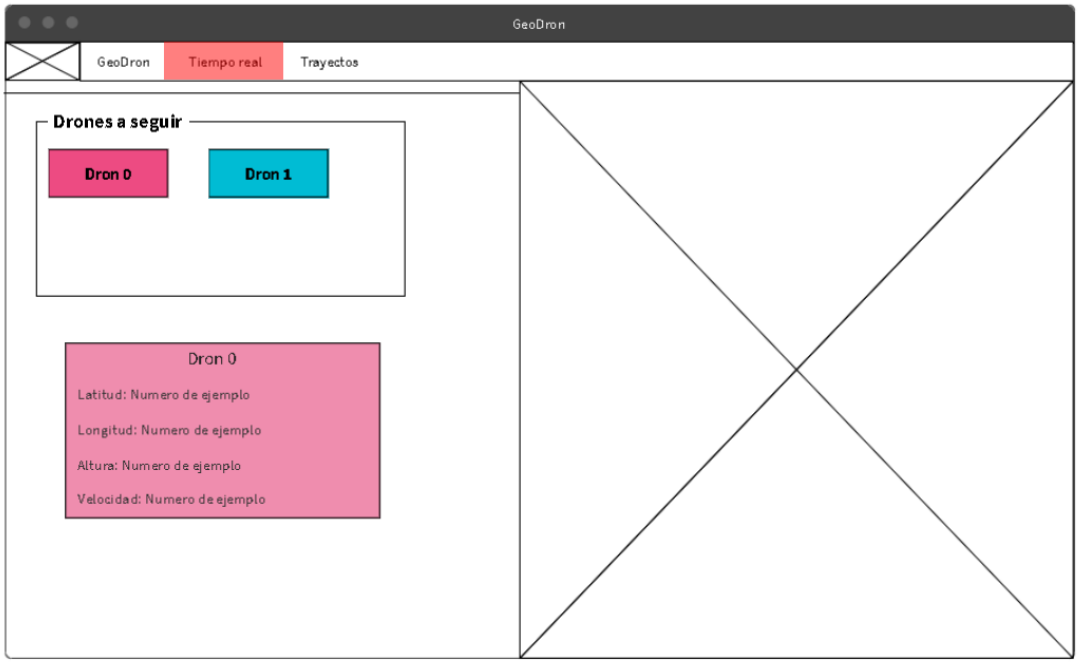


***Ilustración 21**. Ventana de tiempo real (seguimiento de un dron).*

Finalmente, cuando el sistema deja de detectar el movimiento de drones, aparecerá una alerta como las anteriores, indicando los drones que se han desconectado. Además, desaparecerán los elementos informativos de los mismos.

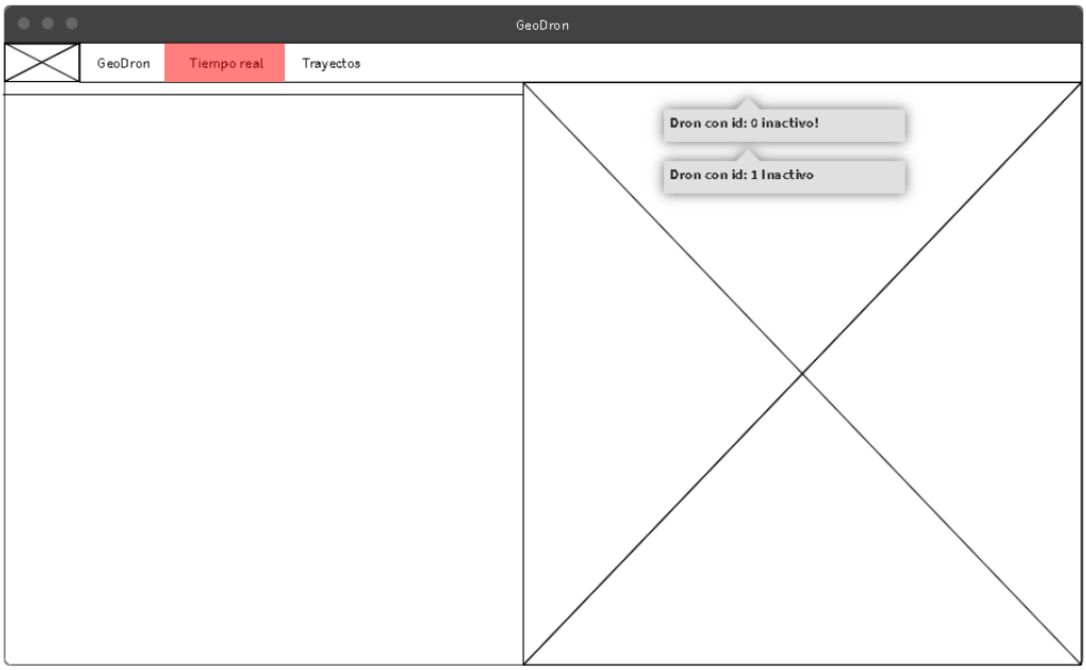


***Ilustración 22**. Ventana de tiempo real (desconexión de drones).*

Otra de las funcionalidades principales del sistema es la **visualización de trayectos anteriores**. Por eso, cuando el usuario pulsa en trayectos, el sistema necesitará de unos segundos para recibir la información, y se mostrará al usuario la ventana mostrada en la ilustración 19.

Cuando el sistema está listo, aparece la ventana de **trayectos**. En la parte izquierda aparece un listado de los trayectos previamente realizados por un dron. En cada elemento se presenta al usuario diferentes atributos del vuelo, como fechas de inicio y fin, y coordenadas de inicio y fin.

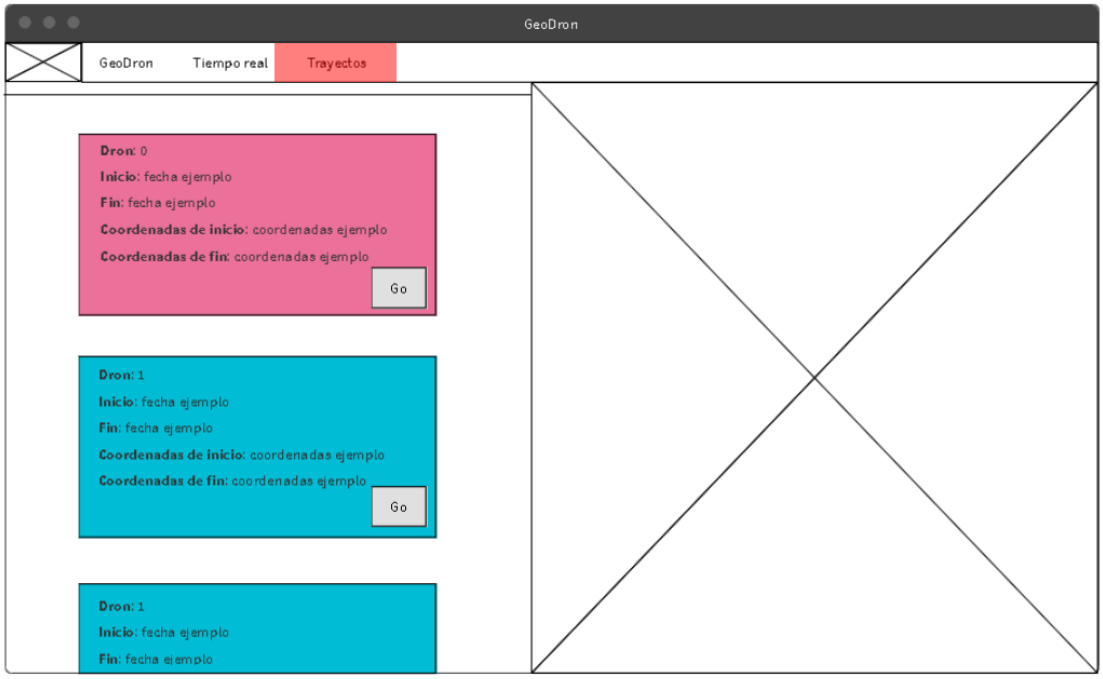


***Ilustración 23**. Ventana de trayectos (listado de trayectos).*

Además, en cada trayecto, aparece un botón que permite visualizar la representación de ese vuelo. Después el mapa comenzará a animarse, dibujando una línea con el trayecto que realizó el dron. Adicionalmente, en la parte superior izquierda, aparece una ventana con datos que se van actualizando conforme pasa el tiempo.

En el análisis de requisitos se indicó que el sistema debe disponer de la funcionalidad de parar y continuar la visualización del trayecto sin necesidad de volver a empezar, por ello se incluyen dichas funcionalidades para cada trayecto.

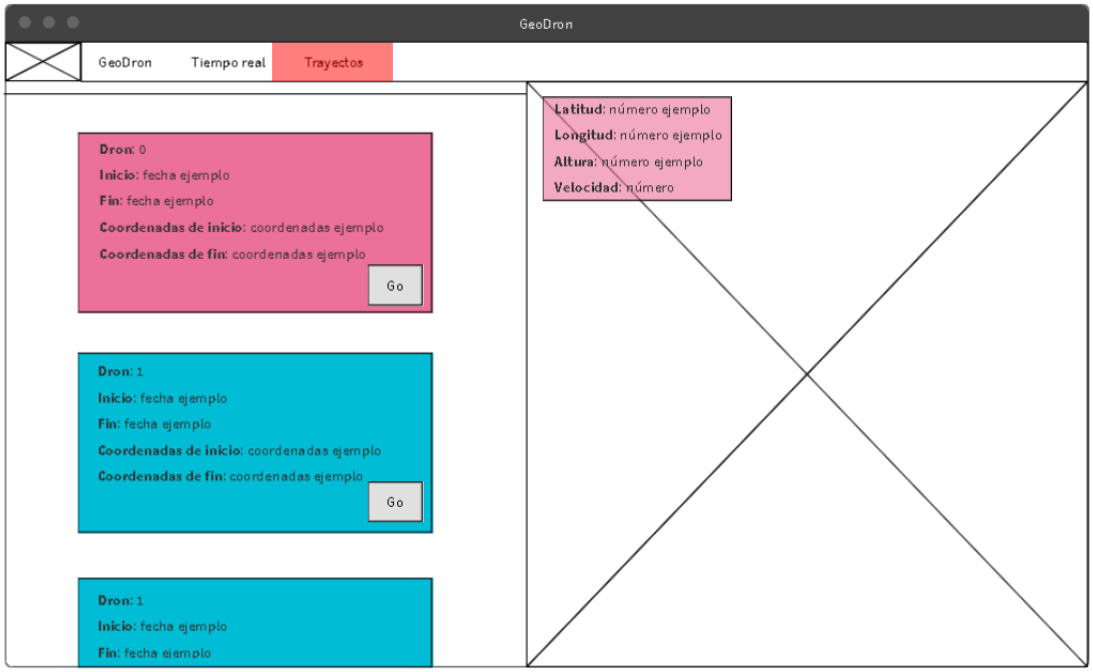


***Ilustración 24**. Ventana de trayectos (visualizado de trayecto)*

4.3.Tecnología empleada

En este apartado se va a debatir y a explicar las tecnologías empleadas, así como las posibles alternativas que surgen.

4.3.1. MEAN Stack

El lenguaje de programación JavaScript se está convirtiendo en uno de los preferidos a la hora del desarrollo del software. Es ampliamente utilizado con el fin de diseñar aplicaciones web dinámicas e interactivas. Uno de sus mayores beneficios es que permite a un desarrollador emplear un único lenguaje de programación para la realización del código tanto para el lado cliente como para el servidor [32].

Se conoce con el acrónimo de MEAN (MongoDB, Express, AngujarJS y NodeJS) a la combinación de estas tecnologías para el desarrollo de aplicaciones web, empleando como único lenguaje de programación el de JavaScript [25].

El empleo de un único lenguaje para el desarrollo de todo un sistema repercute especialmente en la mentalidad de los programadores. Pues, para un proyecto a gran escala con un gran equipo, evita que el personal se fatigue con el empleo de diferentes lenguajes de programación. Además, permite a un desarrollador, con únicamente conocimientos de JavaScript, desplegar una aplicación web completa y funcional.

***Ilustración 25**. MEAN Stack.*

A continuación se va a presentar y explicar la funcionalidad de cada uno de los subsistemas que incorpora el paquete *MEAN Stack* [33].

MongoDB

Es un motor de bases de datos de tipo documental de código abierto que ofrece un gran rendimiento y escalabilidad. MongoDB es una base de datos de tipo no SQL que prescinde de estructuras y esquemas de las bases de datos relacionales, por lo que no emplea tablas, filas o campos [34].

Este tipo de bases de datos presenta una serie de ventajas e inconvenientes respecto a las tradicionales colecciones de datos basadas en SQL [35]. Se va a comparar la tecnología MongoDB con las bases de datos SQL.

Escalabilidad y carácter descentralizado	Uso más adaptado y facilitado
Bases de datos más abiertas y flexibles	Mayor soporte y mejores herramientas
Permiten cambios en los esquemas sin necesidad de reconfigurar las colecciones	Atomicidad en sus operaciones
Bajo consumo	Los datos deben cumplir los requisitos de integridad
Consultas optimizadas para solicitudes con alta carga	

***Tabla 14**. Ventajas de MongoDB y SQL*

Peor integridad de datos	La atomicidad perjudica gravemente en el rendimiento
Falta de estandarización	Dispone de una escalabilidad mucho más pobre que en las bases de datos no relacionales
Herramientas de administración centradas en usuarios entendidos	

***Tabla 15**. Comparativa de desventajas de MongoDB y SQL.*

Para este proyecto, resulta especialmente interesante el empleo de una base de datos no relacional como MongoDB, debido su facilidad de escalado, consultas optimizadas para solicitudes de una gran carga, bajo consumo, y su fácil compatibilidad con JavaScript.

Un punto a favor para la base de datos MongoDB es que se basa en el empleo de un tipo de datos llamado *ObjectID* [36]. Se utiliza para diferenciar cualquier objeto de la base de datos, y su estructura consta del siguiente significado:

- 4 bytes representando la fecha de creación del objeto con respecto a la marca de tiempo Unix.
- 5 bytes totalmente aleatorios.
- 4 bytes que representan un contador de orden de creación de cada objeto.

El empleo de este atributo será clave en la implementación del sistema, dado que ahorra la creación de atributos como fecha, id, etc. La estructura de cada objeto de la base de datos será como la siguiente imagen.

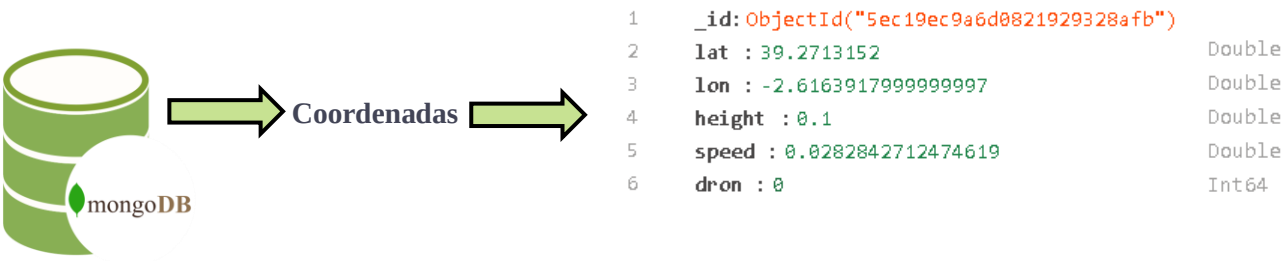


***Ilustración 26**. Esquema de la base de datos.*

Express

Emplear la tecnología node.js para elaborar la lógica de una aplicación, resultaría viable. En cambio, es complejo, costoso y obliga al desarrollador a trabajar a bajo nivel. Para ayudar a crear sitios web con mayor facilidad nació *Express*.

Se trata de un *framework* para aplicaciones basadas en node.js, el cual contiene gran cantidad de utilidades, métodos y *middleware*, con el fin de obtener un desarrollo más rápido y estable.

Será en esta parte del desarrollo del software donde se elaborará la API Rest encargada de realizar las peticiones HTTP a la base de datos.

Su principal objetivo reside en evitar que el desarrollador reinvente la rueda cada vez que se desee crear el código de la parte servidor. A la hora de realizar el desarrollo de la capa lógica de una aplicación web hay varias alternativas a *Express*. Sin embargo, son bastante menos conocidas, y con menor comunidad que el *framework Express*. Tanto es así que node.js ha considerado su *framework* como un estándar *de facto* para su desarrollo.

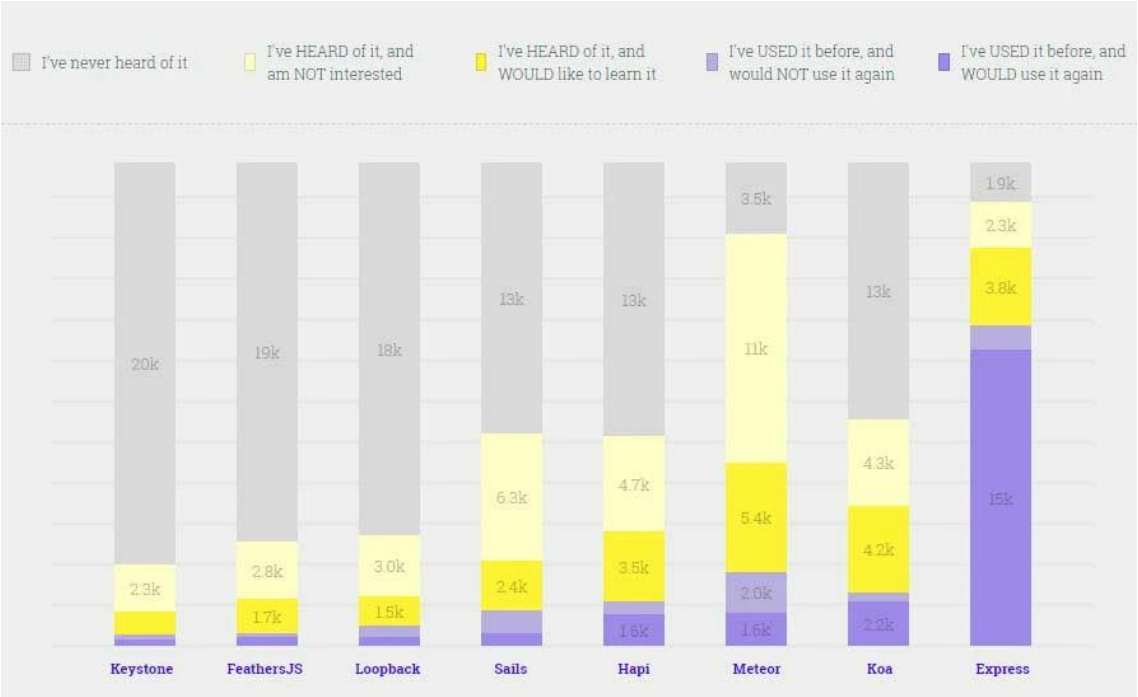


***Ilustración 27**. Gráfico de empleo de frameworks basados en nodejs (Backend).*

Fuente: https://openwebinars.net/blog/los-5-frameworks-de-javascript-para-backend-mas-usados-en-2018/

A pesar de que, JavaScript es el líder indiscutible para la creación de aplicaciones web, existen otros *backend* basados en otros lenguajes de programación como Java con el *framework* Spring, o incluso el aclamado lenguaje de programación Python. A continuación se van a mostrar las ventajas que presenta el empleo de Express en comparación a su principal competidor en el mercado, *Spring boot* [37].

node express	spring
Empleo de JavaScript, lenguaje con alto crecimiento e importante comunidad.	Empleo de Java, con una comunidad amplia y madura.
Más ligero y rápido para configurar	Java comprueba su código en tiempo de compilación
Empleo de un único hilo, debido a su ejecución asíncrona emplea bajo consumo de memoria	Mejor soporte y mantenibilidad a largo plazo
Código muy compacto	Soporte multi hilo
Mejor integración con bases de datos no relacionales	Facilidad para el empleo de dependencias

***Tabla 16**. Ventajas de Express con nodejs y Spring boot.*

No soporta multi hilos	Alta utilización de memoria dado el empleo de la tecnología JVM (Java Virtual Machine)
Problemas en tiempo de ejecución por ausencia de comprobación de los tipos.	Código muy verboso

***Tabla 17**. Desventajas de Express con nodejs y Spring boot.*

AngularJS

Ocurre de forma similar al *framework* Express, pues una página web puede ser elaborada con código HTML, CSS y JavaScript puro. En cambio, el empleo de un *framework* que aporte funcionalidades y bibliotecas al desarrollo permite al desarrollador elaborar la página de una manera más fácil.

Angular es un *framework* dedicado a la parte cliente (*frontend*) que permite el desarrollo de aplicaciones web basadas en la modalidad *Single-Page Aplicactions* compatible en la mayoría de los navegadores. Este tipo de páginas web se caracterizan por estar constituidas por una sola página que se carga al entrar al sitio web, de forma que se van actualizando elementos y componentes de forma dinámica [38].

Es un proyecto actualmente mantenido por Google, en pleno auge y de código abierto, que permite al desarrollador organizar el código por capas, evitando de este modo enormes líneas de código para una sola página web.

NodeJs

Node.js es un *framework* basado en JavaScript que se emplea como plataforma dedicada al funcionamiento y mantenimiento de la parte del servidor de una aplicación web (*backend*). Node.js emplea a bajo nivel el motor de JavaScript de Google, llamado V8, y se caracteriza por proporcionar una arquitectura orientada a eventos, permitir un conjunto de APIs asíncronas y disponer de una programación modular, proporcionando características de rendimiento y escalabilidad excepcionales [39].

Debe parcialmente su popularidad a su gestor de dependencias *Npm*, el cual es ampliamente utilizado por una gran mayoría de desarrolladores gracias a su uso realmente sencillo.

A pesar de ser creado con el fin de ser utilizado en cualquier parte de una aplicación web, se emplea principalmente en el lado del servidor web. Algunas páginas webs altamente conocidas como *Paypal*, *Microsoft* y *Groupon* lo emplean para la capa lógica de sus aplicaciones web.

4.3.2. Java

El servidor intermedio que sirve de middleware entre un dron y la base de datos, está escrito en el lenguaje de programación Java. Se trata de una plataforma informática y a la vez un lenguaje de programación orientado a objetos salido a la luz en el año 1995 [40]. Su principal propósito es el desarrollo del código de una sola vez, y sea posible su ejecución en cualquier máquina gracias a su tecnología **JVM** (Java Virtual Machine).

Las máquinas virtuales de Java se caracterizan por tomar el código en bytes una vez se ha compilado, facilitando la creación de cualquier programa basado en Java y que luego éste se pueda ejecutar en cualquier sistema operativo con soporte a esta tecnología (prácticamente todos).

La comunicación entre un dron y el servidor se basará en las librerías *DatagramSockets* y *DatargamPackets*, tecnologías que posteriormente se indagarán en el apartado del desarrollo del código. En este epígrafe, solo interesa el cómo se envía la información del dron al servidor. Para ello, el dron emplea el canal creado por el *DatagramSocket* con el servidor y, para cada envío, se crea una estructura de datos con los atributos a enviar (siempre en el mismo orden): el ID del dron, latitud, longitud, altura y velocidad. La estructura del *DatagramPacket* tendrá la siguiente estructura [41].

Dirección IP: IP del servidor {}
Puerto: Puerto del servidor = 5555 {numero}
Paquete de datos: id, latitud, longitud, altura, velocidad {byte []}

***Tabla 18**. Estructura de un DatagramPacket.*

La comunicación entre el servidor y la base de datos resulta realmente sencilla en este sistema. Únicamente se va a trabajar con un conjunto de datos que se puede agrupar en un único bloque, por lo que resulta realmente interesante trabajar con los mensajes de tipo **JSON**.

JavaScript Object Notation es un formato únicamente de texto destinado a guardar e intercambiar información de una forma legible. Dentro de un objeto JSON hay dos elementos centrales [42]:

- **Llaves**. Deben ser cadenas de caracteres, y representan el atributo de un objeto.
- **Valores**. Deben ser de un tipo de datos JSON válido (numero, cadena, arreglo, etc.) e indica el valor de un atributo.

Consecuentemente, cuando el servidor intermedio reciba la información del dron, desglosará el contenido del DatagramPacket para clasificar cada atributo. De esta forma se crea un objeto JSON con la adecuada relación entre las llaves y sus valores correspondientes, y se lo envía a la base de datos.

Posteriormente, en el capítulo del desarrollo, se indagará en las librerías empleadas para permitir la comunicación mediante *sockets* y la inserción de datos en la base de datos MongoDB.

4.3.3. ArduSim

Conforme se va desarrollando el sistema, resulta necesario comprobar si resultan efectivas las funcionalidades de este, y por temas operativos, resulta tedioso realizar las pruebas con drones reales.

A pesar de que el sistema de monitorización está diseñado y desarrollado para drones reales, cabe resaltar que también resulta funcional para un simulador de vehículos aéreos no tripulados, en el caso de ser capaz de transmitir datos al servidor.

Para este proyecto se va a emplear el software de *ArduSim* desarrollado por el Grupo de Redes de Computadores de la Universidad Politécnica de Valencia. La funcionalidad de la herramienta es la de simular el comportamiento real de un vehículo aéreo no tripulado, de manera que se puedan realizar misiones y planificar trayectos a partir de una ruta.

En cambio, el software no dispone de la posibilidad de transmitir datos por Internet a un servidor. Consecuentemente, con el permiso y colaboración del principal encargado de la aplicación *ArduSim*, Francisco Fabra, se ha desarrollado una modificación a dicho software para que los drones simulados tengan la posibilidad de enviar datos a un servidor.

5. Desarrollo de la solución

En este capítulo se van a presentar, explicar y solventar los problemas ocurridos a la hora de desarrollar el sistema. En primer lugar, se va a llevar a cabo la configuración del servidor para que reciba datos del dron y los inserte en la base de datos. En segundo lugar, se creará el servicio web con la tecnología MEAN Stack junto con algunas librerías que se comentarán.

5.1. Servidor intermedio

5.1.1. librerías empleadas

- **MongoClient**. Encargada de establecer la conexión con un servidor de MongoDB. Su constructor por defecto se conecta a la dirección IP localhost y al puerto 27017.
- **MongoDatabase**. Encargada de obtener la base de datos, con nombre correspondiente al introducido en el parámetro del constructor.
- **MongoCollection**. Funcionalidad similar a la anterior, en este caso con la colección de la base de datos.
- **DatagramSocket**. Esta clase representa a un *socket* para un posible envío o recepción de DatagramPackets [43].
- **DatagramPacket**. Esta clase representa a un datagrama que sirve para establecer conectividad entre dos dispositivos conectados a internet [44].
- **ByteBuffer**. Se emplea para la realización de diversas operaciones sobre un arreglo de bytes. Tales como, transformar de bytes a un tipo de datos o viceversa.
- **Document**. Perteneciente al API de MongoDB y se emplea para la construcción de un objeto de tipo documento admisible en las bases de datos.

5.1.2. Desarrollo del código

Como ya se ha mencionado, en el sistema será necesario la utilidad de un servidor que esté constantemente escuchando las posibles transmisiones que realice un dron. Para ello se va a desarrollar un ejecutable Java con las siguientes características.

En primer lugar, se va a realizar una conexión de tipo cliente con la base de datos MongoDB. Para ello, se emplean librerías capaces de conectarse a una base de datos y posteriormente a la colección que se desee.

```
MongoClient mongoClient = new MongoClient();
MongoDatabase db = mongoClient.getDatabase("CrudDB");
MongoCollection<Document> coleccion = db.getCollection("coordenadas");
```

***Ilustración 28**. Conexión a la base de datos y a la colección.*

Posteriormente, se va a crear el servidor basado en *sockets* capaz de recibir paquetes en su dirección IP localhost y su puerto 5555. Con el fin de recibir correctamente los datos enviados por un dron, se va a crear una variable *buffer* de tipo arreglo de bytes con tamaño de 1500 para y una variable de tipo ByteBuffer de tamaño 1500.

```
DatagramSocket s = new DatagramSocket(5555);
byte[] buffer = new byte[1500];
ByteBuffer buf = ByteBuffer.allocate(1500);
```

***Ilustración 29**. Creación del socket servidor y variables para almacenar las recepciones.*

A partir de ahora, el siguiente código se ejecuta por cada paquete que envía un dron. Inicialmente, se crear un objeto de tipo *DatagramPacket* que será el encargado de almacenar en la variable *buffer* la información transmitida por un dron.

Nótese que, cuando un dron transmite datos, lo que hace es enviar un arreglo de bytes donde se incluyen todos los parámetros. El servidor será el encargado de desglosar correctamente este conjunto de bytes y, para ello, se empleará la librería *ByteBuffer*. Para ello, cada vez que se recibe un nuevo datagrama, se vaciará el objeto ByteBuffer y se copiará el contenido de *buffer* a la variable *buf*. Lo que se pretende es emplear de su API los métodos '*getLong()*' y '*getDouble()*', de manera que se recuperen los datos en orden: ID del dron, latitud, longitud, altura y velocidad.

```
DatagramPacket req = new DatagramPacket(buffer, buffer.length);
s.receive(req);
buf.clear();
buf.put(req.getData());
buf.clear();

long id = buf.getLong();

double lat = buf.getDouble();
double lon = buf.getDouble();
double height = buf.getDouble();
double speed = buf.getDouble();
```

***Ilustración 30**. Recepción del arreglo de bytes, transformación e inserción en la base de datos.*

Con todos los datos preparados para su inserción en la base de datos, se creará un objeto de tipo *Documet* al que se le añadirá los diferentes conjuntos de llave y valor. Finalmente, con el método '*insertOne(Document)*' del objeto colección, se insertará el documento en la base de datos.

```
Document doc = new Document("lat", lat);
doc.append("lon", lon);
doc.append("height", height);
doc.append("speed", speed);
doc.append("dron", id);


coleccion.insertOne(doc);

System.out.println("Insertado correctamente");
```

***Ilustración 31**. Creación de Documento e inserción en la base de datos.*

5.2. Servicio web

El funcionamiento de la aplicación web va a ser tal y como se muestra en la siguiente imagen. En primer lugar el cliente, a través de la interfaz usable, va a realizar una operación que desembocará en una petición HTTP que realizará al *backend*. Dicha solicitud será una de las pertenecientes del API Rest del *framework Express* funcionando en el servidor NodeJS. En función de la solicitud que envíe el cliente, se realizará una solicitud a la base de datos y se devolverá el resultado al usuario.

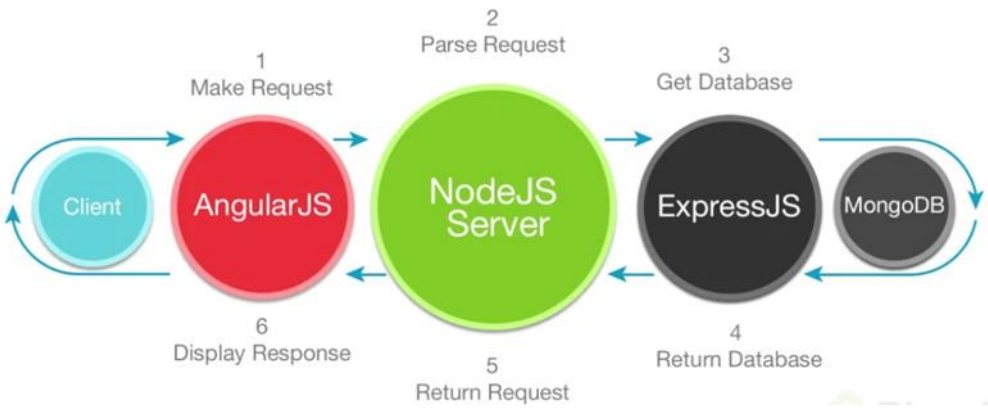


***Ilustración 32**. Funcionamiento MEAN Stack.*

5.2.1. Control de versiones

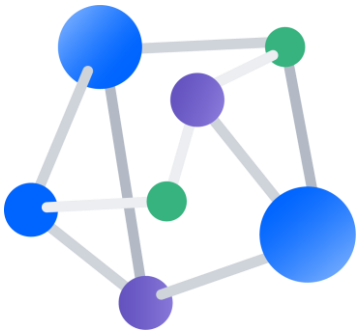
Para este tipo de proyectos, donde se va a desarrollar un software con constantes cambios en su código fuente, resulta impredecible el uso de alguna herramienta capaz de gestionar y monitorizar cambios en el código o estructura de un sistema. Dicho software se conoce como sistema de **control de versiones** o VCS (en inglés), y algunas opciones más populares las lleva Git, Mercurial y SVN [45].

El empleo de este tipo de herramienta permite un desarrollo fiable en un ambiente colectivo o individual, al proporcionar las siguientes funcionalidades:

- **Resolución de conflictos**. Una característica principalmente empleada en trabajos de equipo que permite la monitorización y control en los cambios realizados en un mismo archivo por dos o más desarrolladores.
- **Restaurar y deshacer cambios en el código**. Permite al desarrollador retroceder en el tiempo y recuperar el contenido del sistema para una fecha dada.
- **Copias de seguridad externas**. Permite realizar copias de seguridad para compartir cambios entre compañeros y evitar pérdida de código en caso de accidente.

Para este proyecto se va a emplear *Git*, considerado como el sistema de control de versiones más empleado en el mundo hoy día. Se trata de una herramienta totalmente gratuita y de código abierto, creado en 2005 por Linux Torvalds [46], y caracterizada por los siguientes aspectos:

- **Arquitectura distribuida**. A diferencia de las arquitecturas centralizadas, Git permite a cada desarrollador disponer de su propio repositorio sin tener que depender de la rama principal del sistema.
- **Herramienta muy potente**. Esto es debido a su gran sistema de ramas, las cuales permiten generar proyectos divergentes al proyecto principal, con el fin de realizar pruebas o nuevas funcionalidades sin afectar a la rama principal.
- **Gran posibilidad de operaciones**. Git permite las funcionalidades de crear, modificar o fusionar ramas y modificar historiales del repositorio



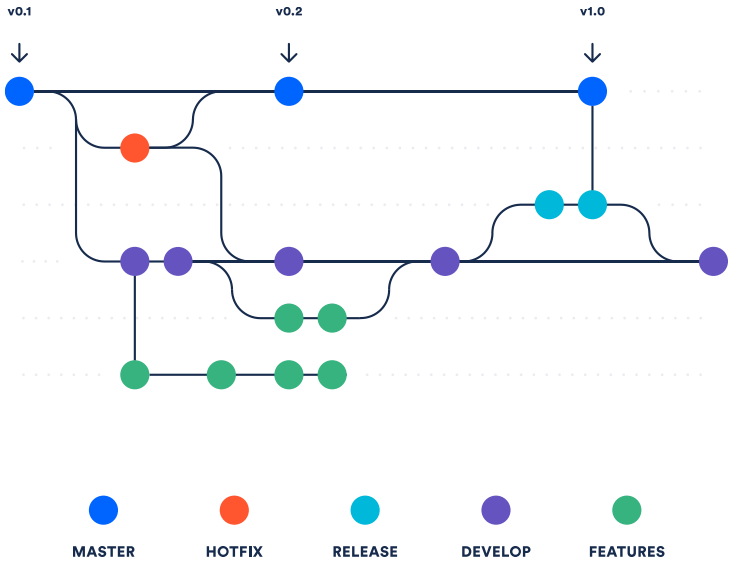


***Ilustración 33**. Arquitectura distribuida y un ejemplo de control de versiones*

Fuente: https://bitbucket.org/product/es/version-control-software

5.2.2. API Mapbox

Para el desarrollo y la animación del mapa necesitado en la monitorización de un dron, se ha empleado la librería *Mapbox GL JS*. Se trata de una librería basada en JavaScript que emplea la herramienta *WebGL* (para la carga de figuras 3D en un entorno web) para cargar y controlar mapas interactivos [47].

Se trata de una herramienta de código abierto y totalmente gratuita que contiene una amplia API con una sección de ejemplos que facilita la correcta funcionalidad y diversa operabilidad del mapa.

La funcionalidad clave que ha determinado el empleo de esta herramienta es la facilidad de inserción de estilos en el mapa. A partir de tipos de datos JSON, permite situar en un mapa, para unas coordenadas dadas, el elemento que se desee, desde una línea hasta un marcador o imagen.

***Ilustración 34**. Mapa ejemplo creado con Mapbox.*

5.2.3. Backend

En este epígrafe, se va a crear y desarrollar el código del lado del servidor (*backend*), encargado de dar respuesta a las solicitudes del usuario. Para ello, se va a partir de la siguiente estructuración de ficheros, a cada cual se explicará a continuación.

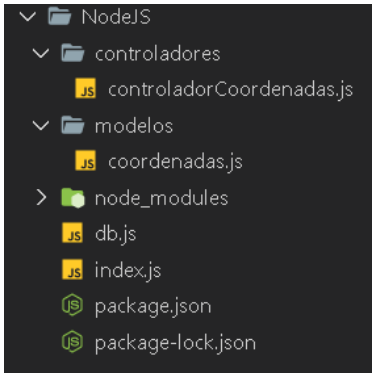


***Ilustración 35**. Esquema ficheros código backend.*

Como ya se sabe, NodeJS emplea la herramienta Npm para la gestión de sus dependencias, por lo que el primer paso consta de inicializar la herramienta e instalar las librerías necesarias: *Mongoose, Express, Body-parser y Cors*.

- **Mongoose**. Es una librería basada en JavaScript que permite crear esquemas con datos fuertemente tipados y realizar operaciones sobre una base de datos MongoDB [48].
- **Express**. Librería que referencia al ya explicado *framework* Express.
- **Body-parser**. Permite al cuerpo de una petición POST o PUT convertirse en el tipo de datos JSON.
- **Cors**. Por defecto, el servidor NodeJS rechaza peticiones desde otra maquina con IP o puerto distinto, por lo que, se emplea la librería Cors para permitir al código *frontend* realizar la petición.

A continuación, se va a crear el código JavaScript '*db.js*' encargado de realizar la conexión con la base de datos donde se están almacenando los parámetros enviados por los drones. Es aquí donde entra la librería *mongoose*, que permite conectarse a una URL correspondiente a la dirección IP y puerto del servidor de la base de datos. Finalmente, se exportará el módulo para su posterior uso en el código *index.js* del servidor.

```
1   const mongoose = require('mongoose');
2
3   mongoose.connect('mongodb://localhost:27017/CrudDB', (err) => {
4       if(!err){
5           console.log('Conexion correcta a MongoDB');
6       }else{
7           console.log('Conexion fallida a MongoDB: '+ JSON.stringify(err, undefined,2));
8       }
9   });
10
11  module.exports = mongoose;
```

***Ilustración 36**. Código db.js.*

Con el fin de poder insertar y recibir datos de la base de datos, será necesario crear un modelo basado en la librería *mongoose* con los atributos que se están empleando para formar un objeto en la base de datos. Dicho archivo se llama *Coordenadas.js*

```
const mongoose = require('mongoose');

var Coordenadas = mongoose.model('Coordenadas', {
    lat: {type: Number},
    lon: {type: Number},
    height: {type: Number},
    speed: {type: Number},
    dron: {type: Number}

});

module.exports = {Coordenadas};
```

***Ilustración 37**. Código Coordenadas.js*

Otro elemento fundamental del lado servidor es el archivo *controladorCoordenadas.js*, y se trata del controlador de rutas empleado en el API Rest, donde se crearán las distintas operaciones a las posibles solicitudes HTTP que realice el código *frontend* (GET, POST, PUT, etc.). Por ejemplo, si se desea visualizar el historial de trayectos, será necesario que se realice una petición GET para recibir todos los objetos de la base de datos. En el desarrollo del código *frontend* se explicará cómo realizaría la petición.

```
const express = require('express');
var router = express.Router();

var {Coordenadas} = require('../modelos/coordenadas');

router.get('/', (req, res) =>{
    Coordenadas.find((err, docs) => {
        if(!err){res.send(docs);}
        else{ console.log('Error al recibir las coordenadas: ' + JSON.stringify(err, undefined, 2)); }
    });
});

router.get('/getLast', (req, res) =>{
    Coordenadas.find((err, docs) => {
        if(!err){res.send(docs);}
        else{ console.log('Error al recibir las coordenadas: ' + JSON.stringify(err, undefined, 2)); }
    }).sort({_id:-1}).limit(1);
});

router.post('/', (req,res) =>{
    var cor = new Coordenadas({
        lat: req.body.lat,
        lon: req.body.lon
    });
    cor.save((err, doc) =>{
        if(!err) res.send(doc);
        else { console.log('Error al insertas las coordenadas: ' + JSON.stringify(err, undefined, 2)); }
    });
})

module.exports = router;
```

***Ilustración 38**. Código de controladorCoordenadas.js*

Finalmente, se elaborará el archivo encargado de hacer funcionar al servidor *backend* con nombre de *index.js*. En este se importan los anteriores archivos creados: *db.js* y *controladorCoordenadas.js*.

En primer lugar, se inicializa el servidor con el método *'express()'* y se indica que emplee el lenguaje JSON para las comunicaciones. Con el fin de permitir la conexión entre el servidor del lado cliente y el lado servidor, será necesario permitir al servidor conexiones desde otro puerto, y para ello se empleará la librería Cors.

Posteriormente, se deberá indicar al servidor Express que emplee el direccionamiento de rutas creado en el archivo controladorCoordenadas.js. Finalmente, se pondrá a escuchar en el puerto 3000.

```
1  const express = require('express');
2  const bodyParser = require('body-parser');
3  const cors = require('cors');
4
5  const { mongoose } = require('./db.js');
6  var controladorCoordenadas = require('./controladores/controladorCoordenadas.js');
7
8  var app = express();
9  app.use(bodyParser.json());
10 app.use(cors({ origin: 'http://localhost:4200'}));
11
12 app.listen(3000, ()=> console.log('Servidor inicializado al puerto 3000'));
13
14 app.use('/coordenadas', controladorCoordenadas);
```

***Ilustración 39**. Código main del servidor. index.js*

De esta forma ya se tendrá disponible el API Rest para realizar consultas e inserciones en la base de datos.

5.2.4. Frontend

La parte del cliente es desarrollada con el *framework* de Angular, el cual sigue una estructuración dividida en componentes. Cada uno de ellos representa una parte de la página web y se puede dividir en cuatro elementos:

- **HTML**, documento que permite al navegador mostrar al usuario los elementos con los que puede interactuar.
- **CSS**, archivos de hojas de estilo que personalizan los elementos HTML.
- **TypeScript**, se encarga de darle lógica y dinamismo a los elementos HTML.
- **Spec.ts**, es un archivo de testeo para la lógica del componente.

Adicionalmente, Angular permite la creación de un componente de tipo servicio, el cual implica la única existencia de un fichero basado en *Typescript* y se caracteriza por no mantener una relación directa con la interacción del usuario.

En la estructuración mostrada en la figura de abajo, se puede observar que se han creado tres componentes y un servicio junto con un componente creados por defecto por el *framework* llamado appComponent. A continuación, se procede a la explicación de cada uno de ellos.

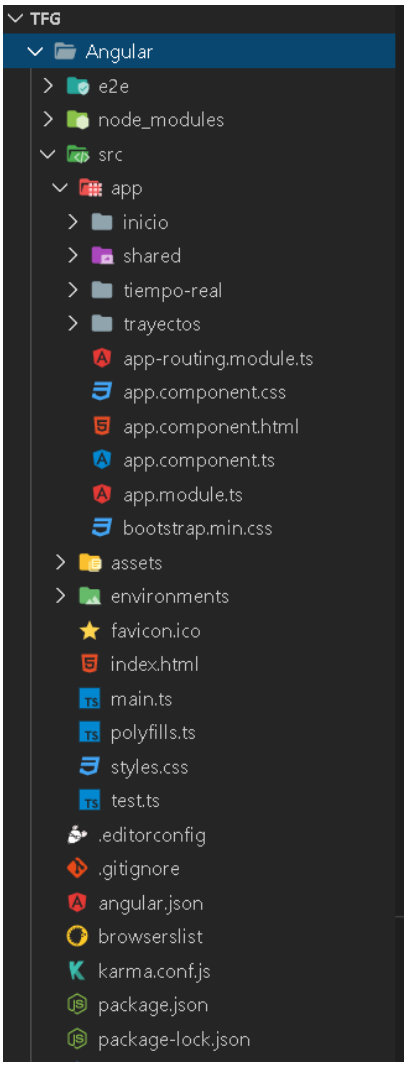


***Ilustración 40**. Estructura ficheros frontend.*

Coordenadas.service.ts

Se trata de un componente de tipo servicio, y es el encargado de realizar las peticiones HTTP al servidor *backend* en el momento que el usuario lo requiere. Para su correcto funcionamiento, es necesario la importación del modelo de tipo de datos Coordenadas y un par de librerías de Angular: *Injectable*, que permite la inserción del servicio en los posteriores componentes, y *HttpClient*, que permite realizar las consultas HTTP.

Se creará una clase exportable *CoordenadasService*, con una instancia '*cord'* que será usada para indicar la última coordenada registrada, un arreglo de coordenadas de nombre '*cords*' donde se almacenarán los datos de la base de datos solicitados, y una cadena de caracteres '*baseURL'* con la URL del servidor *backend*. Adicionalmente, la clase dispondrá de un constructor que creará una instancia de tipo *HttpClient* para realizar las consultas.

Las posibles solicitudes que puede realizar el sistema son realmente escasas y únicamente de tipo GET, dada la funcionalidad que se desea para el proyecto. De entre ellas, se encuentra la solicitud GET por defecto que recupera todas las coordenadas de la base de datos y la solicitud GET con *BaseURL* + "*/getLast*" que recupera la última coordenada registrada en la colección.

```
import { Injectable } from '@angular/core';
import {HttpClient} from '@angular/common/http';


import {Coordenadas} from './coordenadas.model';

@Injectable()
export class CoordenadasService {
  cord: Coordenadas;

  cords: Coordenadas[];

  readonly baseURL = "http://localhost:3000/coordenadas";

  constructor(private http: HttpClient) {}


  getUltimaCoordenada(){

    return this.http.get(this.baseURL + '/getLast');
  }

  getCoordenadas(){
    return this.http.get(this.baseURL);
  }
}
```

***Ilustración 41**. Código coordenadas.service.ts.*

Dicha clase, será importada en los siguientes componentes para disponer de los datos con el fin de mostrar información al usuario. En el siguiente código ejemplo se recibe todas las coordenadas insertadas en la base de datos y se almacenan en la variable *cords* del servicio. De esta manera, el código *frontend* puede acceder a todas las coordenadas mediante dicha variable.

```
getCoordenadas(){
  this.cordServ.getCoordenadas().subscribe((res)=>{
    this.cordServ.cords = res as Coordenadas[];
  });
}
```

***Ilustración 42**. Código ejemplo para recibir las coordenadas desde la API Rest.*

InicioComponent y appComponent

En cuanto al *appComponent*, se trata de un componente creado por defecto al inicializar un proyecto Angular, y se emplea para relacionar los demás componentes creados. En cuanto a su código HTML, únicamente representa la barra de navegación situada en la parte superior que permite el flujo entre componentes.

En cuanto al componente de inicio, es el que muestra al usuario cuando se conecta a la página web. En él se reproduce un video de fondo mientras se muestra al usuario unos bloques informativos del sistema de monitorización.

TiempoRealComponent

Se trata del componente encargado de mostrar al usuario la ventana de monitorización en tiempo real de los drones disponibles. Para el funcionamiento del mismo, cuando el usuario invoca al componente de tiempo real, automáticamente se realiza una petición HTTP al servidor *backend* para obtener un listado de las coordenadas almacenadas en la base de datos.

Su funcionamiento depende principalmente de tres métodos, uno encargado de detectar los drones activos, otro de agrupar las coordenadas en función de su dron y el ultimo se encarga de actualizar los parámetros y el mapa.

```
checkDronesActivos()
```

Esta función se encarga de mantener actualizada una lista con los drones que actualmente están en vuelo. Para ello, el componente realiza periódicamente una petición HTTP con el fin de obtener las coordenadas registradas en la base de datos durante los 10 segundos anteriores a la fecha de invocación de la función.

En el momento en el que el sistema detecta la incorporación o la desconexión de un dron, este método se encarga de alertar al usuario mediante la interfaz web de lo ocurrido. Adicionalmente, cuando un dron aparece o desaparece, este método se encarga de crear o hacer desaparecer el marcador animado, así como la línea que representaba su recorrido.

```
rellenarTrayectos()
```

Esta función se encarga de mantener un objeto de tipo Trayectos para cada dron que se ha detectado. En primer lugar, obtiene mediante la solicitud HTTP las coordenadas de la base de datos. Posteriormente, recorre la lista de coordenadas registradas con fecha posterior a la fecha de despliegue de la aplicación web. Para cada instancia de coordenadas, las añade al trayecto al que corresponda en función del dron al que pertenezca. Dichos objetos trayectos serán empleados para la actualización geográfica del mapa.

```
actualizarLineaMarcador()
```

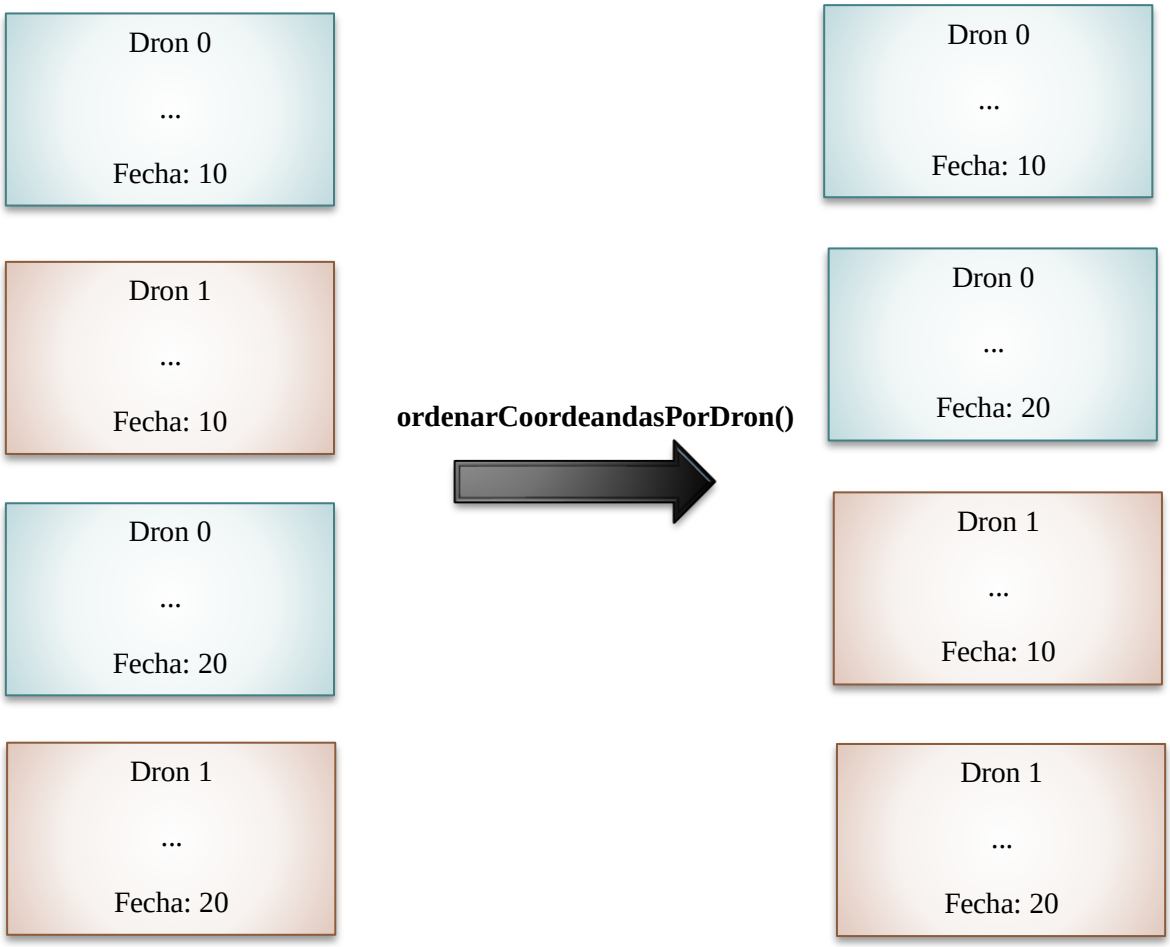
Para cada trayecto que se disponga en el sistema, se debe actualizar una serie de elementos en el mapa. De entre ellos, un marcador animado y una línea que representa el trayecto del dron. Para ello, este método se encarga de recorrer el listado de coordenadas de un trayecto y transforma sus valores al tipo de datos GeoJSON empleado en la API de *Mapbox* para la animación de los elementos.

TrayectosComponent

Este componente será el encargado de mostrar la ventana de trayectos explicada en el capítulo de diseño, y proporcionar al usuario la posibilidad de visualizar trayectos previamente realizados. Para ello, se realizará una única petición HTTP al API Rest con el fin de recuperar todas las coordenadas registradas en la base de datos para dicho trayecto, y se agruparán los datos en bloques para la disposición del usuario. Se destaca el uso de las siguientes funciones.

```
ordenarCoordenadasPorDron(): Coordenadas[]
```

Evidentemente, en la transmisión de datos por parte de los drones, existe un orden cronológico pues la primera coordenada insertada resulta la más antigua. En cambio, cuando se encuentra más de un dron transmitiendo datos, las coordenadas se encontrarán mezcladas en la base de datos. Como solución a este problema surge esta función, cuyo principal cometido es ordenar las coordenadas por su dron encargado. El siguiente esquema resume su función.



```
coordenadasToTrayectos(ordenado)
```

Una vez ordenada la lista de coordenadas por su id de dron y orden cronológico, resulta realmente sencillo realizar la agrupación de coordenadas por trayectos. En esta función se va a recorrer dicha lista, y para aquellas con fechas muy cercanas entre ellas, van a formar un objeto de tipo trayecto.

Con el funcionamiento de estos dos métodos, se le muestra al usuario el listado de los trayectos con algunos parámetros descriptivos tales como inicio, fin, coordenadas inicio, etc. Para cada trayecto, se dispone de la posibilidad de realizar una animación representado el trayecto que realizó el dron, cuya acción se realiza en la siguiente función.

```
mostrarLinea(tra: Trayectos, boton)
```

Será el método que se ejecute cuando el usuario desee visualizar o parar la animación de un trayecto concreto. En el momento en el que el usuario pulse en el botón se le pasará al servidor cliente el trayecto que se ha seleccionado y el elemento HTML correspondiente al botón. El código realizará una acción u otra en función del texto que tenga el botón.

- ***Go*.** Es el texto por defecto que aparece en todos los trayectos, e implica el inicio de la animación del trayecto. Cuando se pulsa se muestra al usuario una ventana con datos del vuelo del dron, junto con un mapa animado simulando el vuelo que realizó un trayecto para una fecha concreta. Dicha actualización es llevada a cabo por un método '*setInterval()*'
- ***Pause***. Es el texto que se visualiza después de pulsar el botón *Go* e implica la parada de la animación del trayecto. Cuando este se pulsa, se detiene el *'setInterval()'* y se almacenan los datos por donde se ha quedado el trayecto para su posterior continuación.
- ***Continue***. Es el texto que se observa después de pulsar el botón *Pause,* e implica la continuación de la animación del trayecto. Cuando se pulsa, se recuperan los datos por donde se quedó la animación y se vuelve a ejecutar el '*setInterval()*'

5.3. ArduSim

Como se ha comentado, surge la necesidad de emplear un simulador de vuelo de vehículos aéreos no tripulados con el fin de realizar pruebas del correcto funcionamiento del sistema. Para que el software realice la transmisión de datos hacia un servidor se van a realizar los siguientes cambios.

En primer lugar, será necesario modificar el cuadro de dialogo de la aplicación para permitir al usuario del simulador la opción de monitorizar los drones. Al activar la casilla de *Enable web logging* aparecerá una nueva ventana de dialogo con el usuario donde éste puede modificar parámetros como la dirección del servidor, el puerto donde escucha, la frecuencia con la que transmitirá los datos (en milisegundos), y el tamaño del buffer (en bytes).

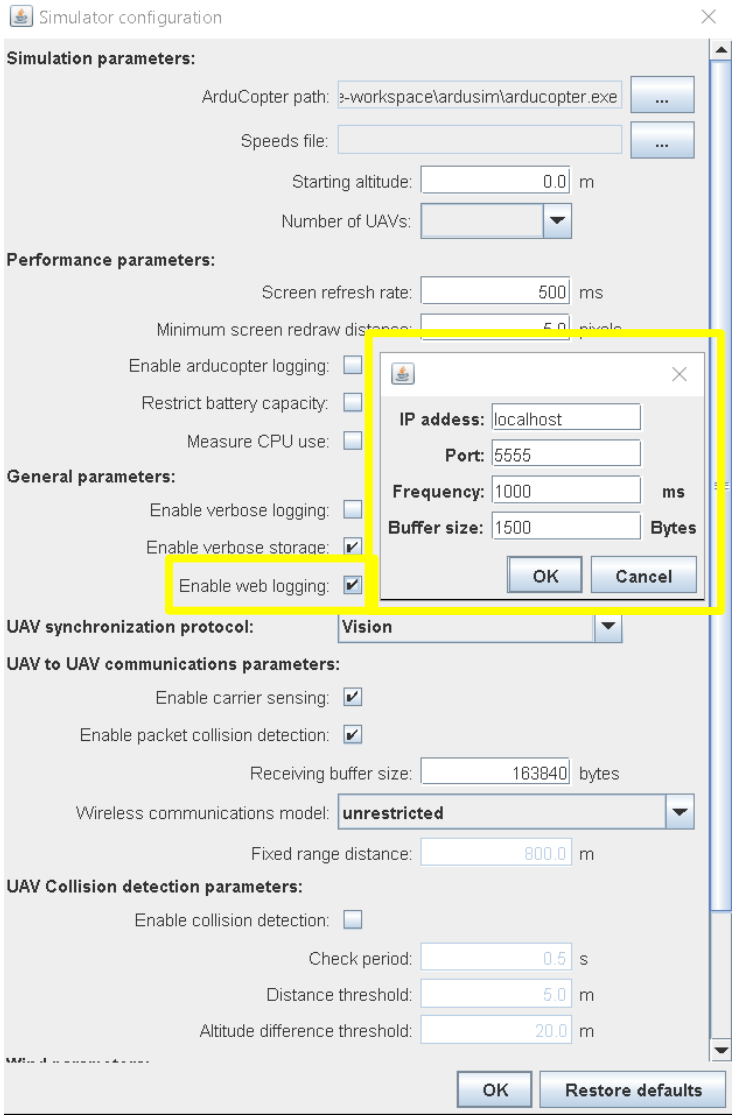


***Ilustración 43**. Cuadro de dialógo de 'ArduSim'.*

En segundo lugar, cuando se activa la casilla se pondrá en marcha una clase, que se va a crear, llamada *clientThread.java*. Sin entrar mucho en detalles poco relevantes para este proyecto, mediante la propia API del software se obtienen los objetos *Copter*, que representa a un dron, y se emplean los parámetros introducidos por el usuario en el cuadro de dialogo con el fin de establecer la conexión con el servidor, creando un *DatagramSocket*. De manera análoga al empleo del *ByteBuffer* y *DatagramPacket* utilizado en el servidor intermedio, transforma los parámetros en Bytes y los transfiere mediante un objeto DatagramPacket. Su código intermedio se puede visualizar en el anexo.

6. Implantación

Una vez desarrollada la solución del proyecto, el siguiente paso consiste en la instalación y configuración para su correcto funcionamiento.

En primer lugar, será necesaria la creación de la base de datos y la colección “coordenadas” para permitir el almacenamiento de los parámetros geográficos. Para ello, se inicia el servidor MongoDB con un archivo descargable desde su página web oficial. De esta forma, el servidor MongoDB se encuentra escuchando en la dirección: 127.0.0.1:27017.

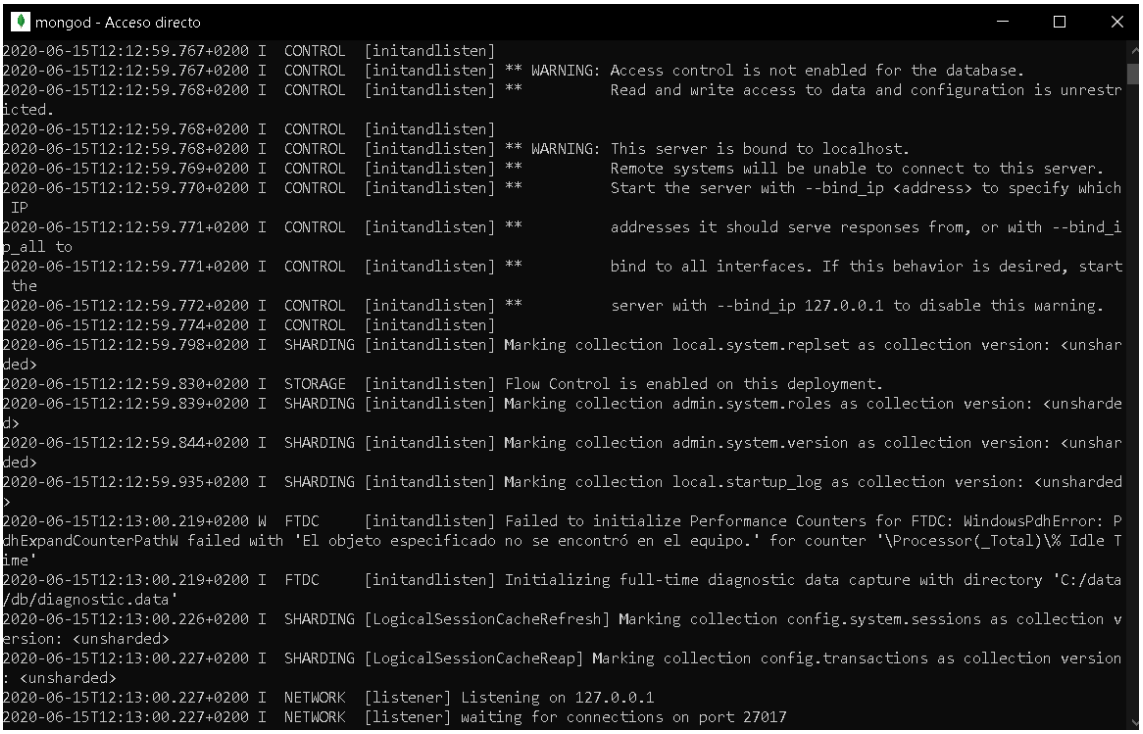


Ilustración 44*. Servidor MongoDB.*

MongoDB dispone de una herramienta llamada *MongoDB Compass* que permite al administrador de la base de datos conectarse al servidor y poder monitorizar u operar sobre ella. De manera que, se va a crear y la base de datos “CrudDB” y la colección “coordenadas” donde se van a almacenar los datos.

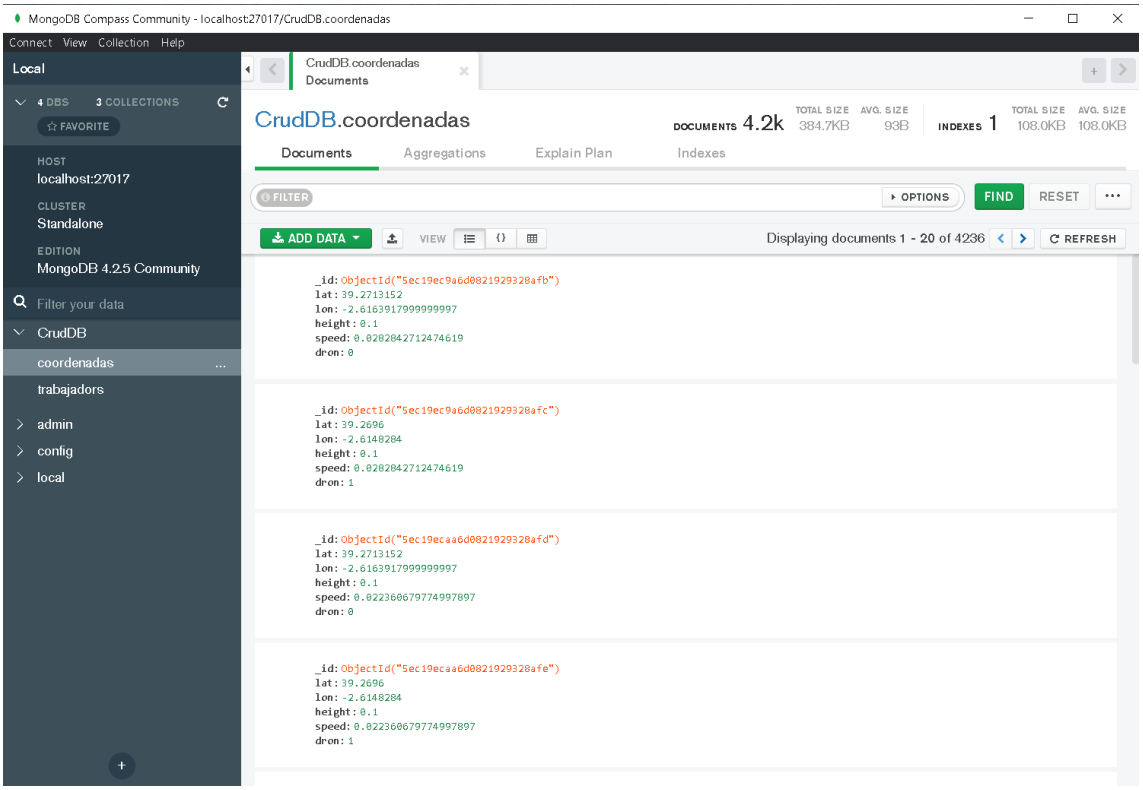


***Ilustración 45**. Ventana de administración MongoDB Compass.*

Con la base de datos preparada para recibir información de una fuente externa, se va a inicializar el servidor intermedio encargado de integrar los datos enviados por un dron e insertarlos en la base de datos. Para ello, se exporta el proyecto como un archivo JAR ejecutable y se inicia.

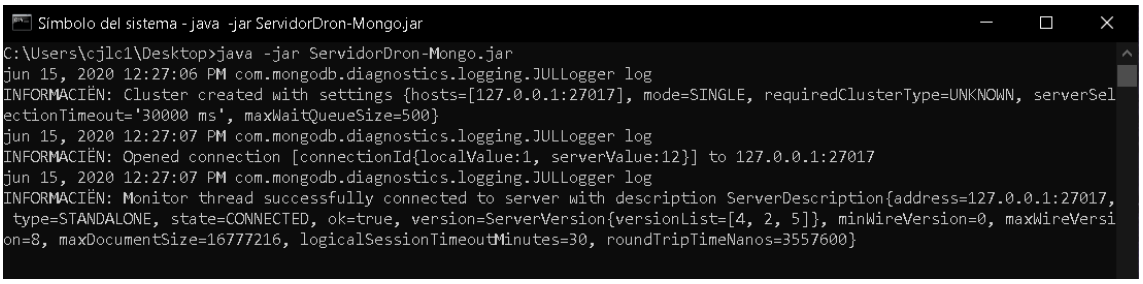


***Ilustración 46**. Inicialización del servidor intermedio.*

En este paso, la conexión del servidor con el dron está cubierta y únicamente falta inicializar el servidor web para pasar a disposición del usuario. En primer lugar, se deberá de iniciar el servidor *backend* de manera que se conecte a la base de datos y esté disponible la API Rest.

```
PS C:\Users\cjlc1\Documents\TFG\NodeJS> node .\index.js
(node:19528) DeprecationWarning: current URL string parser is deprecated,
 { useNewUrlParser: true } to MongoClient.connect.
(node:19528) DeprecationWarning: current Server Discovery and Monitoring e
w Server Discover and Monitoring engine, pass option { useUnifiedTopology:
Servidor inicializado al puerto 3000
Conexion correcta a MongoDB
```

***Ilustración 47**. Inicialización servidor backend.*

En segundo lugar, será necesario desplegar el servicio generado por el *framework* de Angular; para ello, se dispone del comando '*ng serve*' que permite a un usuario conectarse a través de un navegador al servidor web y recibir la interfaz.

```
PS C:\Users\cjlc1\Documents\TFG\Angular> ng serve

chunk {main} main.js, main.js.map (main) 649 kB [initial] [rendered]
chunk {polyfills} polyfills.js, polyfills.js.map (polyfills) 150 kB [initial] [rendered]
chunk {runtime} runtime.js, runtime.js.map (runtime) 6.15 kB [entry] [rendered]
chunk {styles} styles.js, styles.js.map (styles) 1.16 MB [initial] [rendered]
chunk {vendor} vendor.js, vendor.js.map (vendor) 5.2 MB [initial] [rendered]
Date: 2020-06-15T11:08:43.386Z - Hash: a0d9f024dc0095025610 - Time: 14144ms

WARNING in ./src/app/trayectos/trayectos.component.css
Module Warning (from ./node_modules/postcss-loader/src/index.js):
Warning

(81:1) @import must precede all other statements (besides @charset)
```

***Ilustración 48**. Inicialización frontend.*

7. Pruebas

En este capítulo, se va a comprobar el correcto funcionamiento del sistema realizando pruebas de la funcionalidad de éste monitorizando dos drones a la vez.

Una vez desplegado el servicio, un dron debe enviar datos al servidor para que éste lo inserte en la base de datos. Para ello se empleará el software modificado de *ArduSim*, y se le pasará un par de rutas para que los drones simulados la sigan.

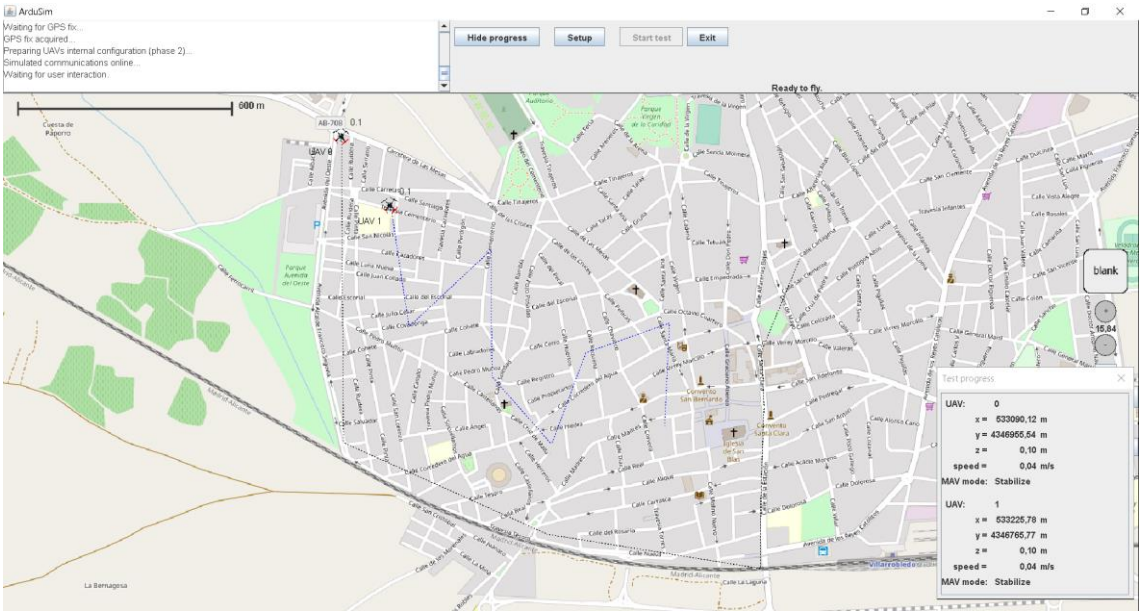


***Ilustración 49**. Ejecución de ArduSim.*

Con la herramienta de *MongoDB Compass* se puede ir comprobando que se van insertando los datos correctamente.

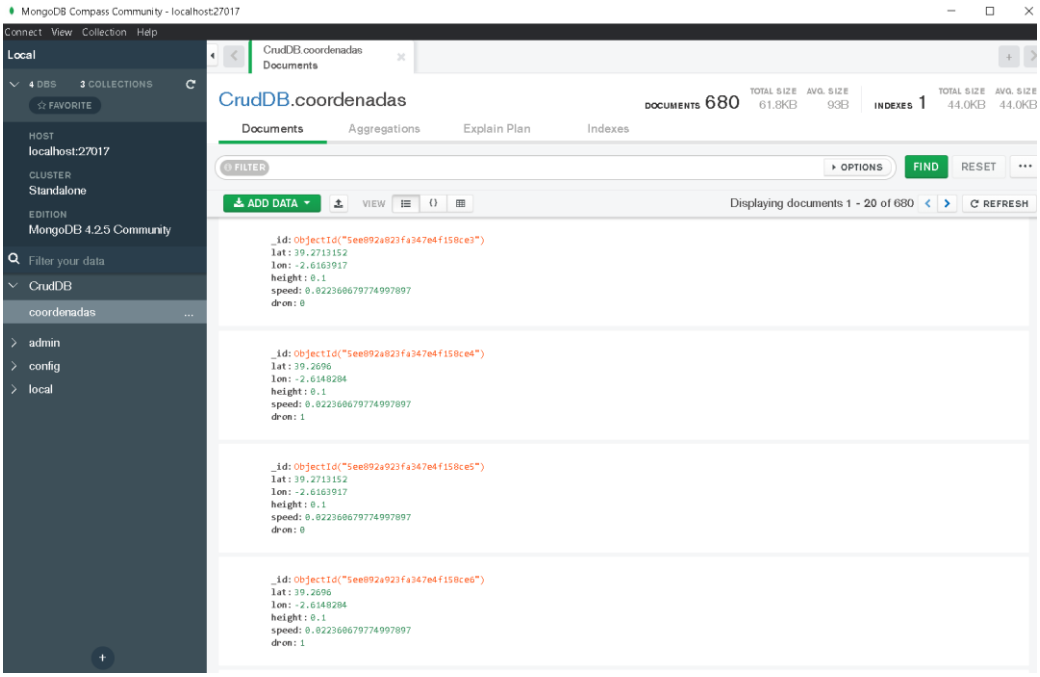


***Ilustración 50**. Datos insertados en la base de datos.*

Una vez iniciada la simulación en la herramienta *ArduSim*, si se accede a la ventana de Tiempo real se puede observar que, tras unos 5 segundos, aparecen en el mapa los marcadores indicando la posición de los drones, y se muestran en pantalla un par de alertas indicando la conexión de dos drones, con sus correspondientes Id.

***Ilustración 51**. Ventana monitorización tiempo real (supervisión simultánea de varios drones).*

***Ilustración 52**. Ventana monitorización tiempo real (conexión drones).*

Adicionalmente, aparecen dos botones con los Id de cada uno de los drones con el fin de centrar el seleccionado en el mapa, así como mostrar al usuario el listado detallado de los parámetros geográficos en los que se encuentra el dron. De entre ellos, su longitud, latitud, altura y velocidad.

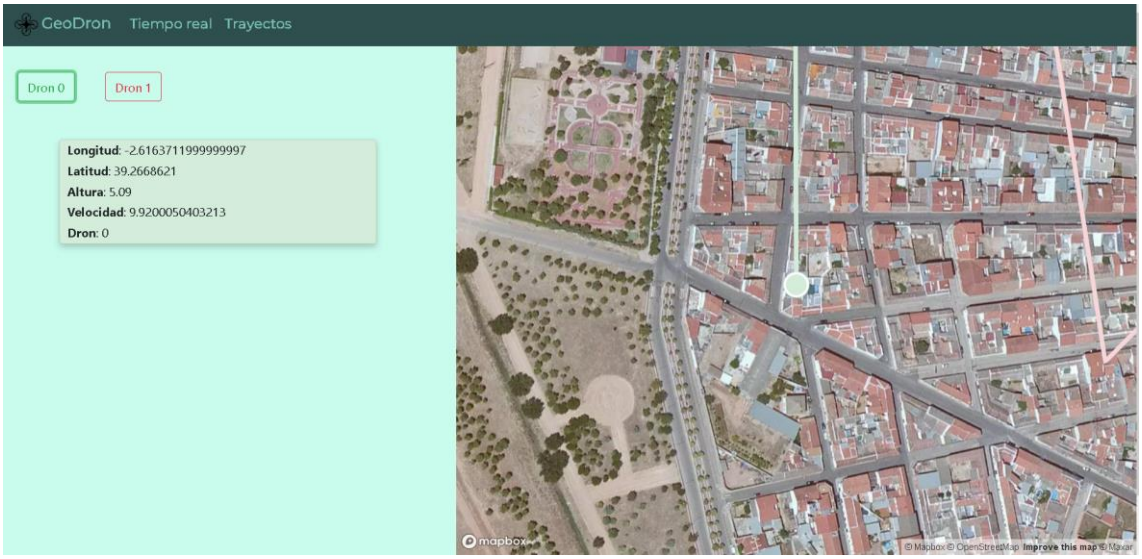


***Ilustración 53**. Ventana monitorización tiempo real (seguimiento del dron 0).*

***Ilustración 54**. Ventana monitorización tiempo real (seguimiento del dron 1).*

En el momento en el que el servidor web detecta que no hay más registros en la base de datos por ningún dron en los últimos 10 segundos, desaparecerá el marcador y el recorrido que ha realizado dicho dron, y se informa al usuario mediante una alerta la desconexión del dron junto a su Id.

***Ilustración 55**. Ventana monitorización tiempo real (desconexión drones).*

En la ventana de trayecto se requiere al servidor web unos segundos para la carga de los datos almacenados en la base de datos.

***Ilustración 56**. Ventana trayectos (carga de datos).*

Una vez los recibe y los estructura en bloques, muestra al usuario un listado de los trayectos que han realizado todos los drones monitorizados a lo largo de la vida útil del sistema junto con las coordenadas de inicio y fin, así como fecha de inicio y fin. El usuario podrá pausar y continuar la animación del trayecto.

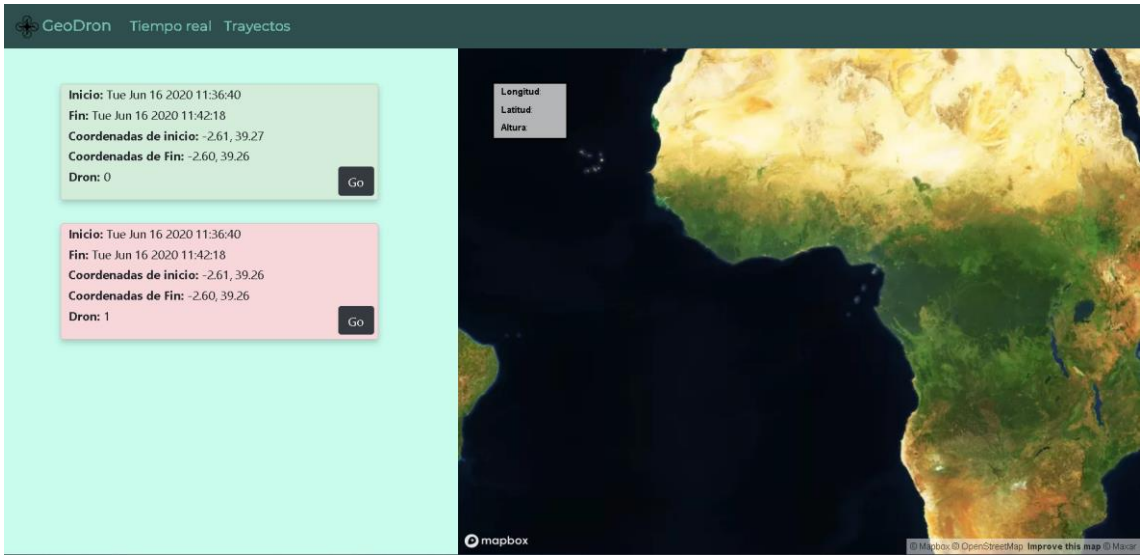


***Ilustración 57**. Ventana de trayectos (listado de trayectos).*

***Ilustración 58**. Ventana de trayectos (animación en curso).*

***Ilustración 59**. Ventana de trayectos (animación en pausa).*

8. Conclusiones

Fruto de la mezcla de conocimientos previamente asentados en el Grado y los conocimientos adquiridos en la investigación de las tecnologías empleadas, este proyecto ha desembocado en un sistema de monitorización remota de vehículos aéreos no tripulados capaz de supervisar tanto drones reales como simulados.

Aunque ha sido necesaria una amplia investigación sobre novedosas tecnologías que hasta día de hoy no había trabajado, todos los conocimientos base necesarios para este proyecto fueron previamente introducidos en el Grado, tales como redes de comunicación, programación Java, diseño web, bases de datos, e incluso integración de datos.

Con la tecnología Java y algunas librerías investigadas se ha logrado crear una comunicación entre un dron y la base de datos, mediante la intervención de un servidor intermedio capaz de realizar las operaciones necesarias para integrar los datos entre ellos.

Una completa y exhausta investigación sobre el paquete de subsistemas MEAN Stack ha supuesto una alta carga de trabajo para este proyecto. Con el empleo de estas tecnologías punteras se ha diseñado una aplicación web de una sola página capaz de cargar dinámicamente parámetros útiles para el usuario. En ella se han logrado realizar todos los requisitos funcionales que se han establecido para el software, desde la monitorización en tiempo real hasta la visualización de anteriores trayectos.

Finalmente, se realizó una breve modificación al software ‘ArduSim’ para permitir la transmisión de datos de los drones simulados al servidor intermedio. De esta forma, se ha podido validar el correcto funcionamiento del sistema.

A pesar de haber cumplido todos los objetivos propuestos al comienzo, me gustaría dejar este proyecto con un final abierto, pues aún puede mejorarse considerablemente el servicio con **futuros trabajos**. Resultaría interesante poder manejar el dron desde la aplicación web, dotándolo de funcionalidades como regresar al punto de inicio, aterrizar, o incluir un sistema preventivo capaz de detectar una posible colisión entre dos drones, modificando su ruta para evitar el choque. Igualmente, cabría mejorar la seguridad tanto de la comunicación de los drones, así como de la aplicación web, con mecanismos de cifrado y un sistema de autenticación.

El desarrollo e implantación de este proyecto me ha supuesto una agradable experiencia, pues he podido valorar el amplio abanico de conocimientos que he adquirido durante la carrera, el cual me ha servido para profundizar en diversos ámbitos de tecnologías que no había tratado nunca. Consecuentemente, este reto me ha servido para dar los primeros pasos en lo que he considerado el reflejo del mundo profesional.

Bibliografía

[1] *Gianluca Casagrande, András Sik, Gergely Szabó.* Small Flying Drones: Applications for Geographic Observation. ISBN 3-319-66577-4, publicado en 2018.

[2] *Syed Omar Faruk Towaha*. Building Smart Drones with ESP8266 and Arduino: Bui.d exciting drones by leveraging the capabilities of Arduino and ESP8266. ISBN: 1788477510, publicado en 2018

[3] Qué es SCRUM. Consultado el 03/05/2020. https://proyectosagiles.org/que-es-scrum/

[4] *Jeffer Chaparro Mendivelso, Jhoan Coronado, Miguel García, Achouak Rabia, Andrés Zárate*. El dron como herramienta tecnológica de control territorial (páginas 3-5). ISSN 1578-0007, publicado en diciembre de 2018.

[5] Los Drones podrán ser monitorizados en tiempo real como los aviones. Consultado el 03/05/2020. https://www.adslzone.net/2016/07/04/los-drones-podran-monitorizados-tiempo-real-los-aviones/

[6] Sistema de Vigilancia Dependiente Automática. Consultado el 03/05/2020 (actualización del 10 de diciembre de 2019). https://es.wikipedia.org/wiki/Sistema_de_Vigilancia_Dependiente_Automática

[7] Rastreando aviones: cómo funciona el Flightradar24. Consultado el 04/05/2020. https://www.kaspersky.es/blog/tracking-airplanes-how-flightradar24-works/5808/

[8] Entra en funcionamiento el sistema ADS-B. Consultado el 04/05/2020. https://a21.com.mx/index.php/aeronautica/2020/01/01/entra-en-funcionamiento-sistema-ads-b

[9] Vodafone usa la red móvil 4G como radar para el primer vuelo monitorizado de un dron. Consultado el 04/05/2020. https://www.networkworld.es/telecomunicaciones/vodafone-usa-la-red-movil-4g-como-radar-para-el-primer-vuelo-monitorizado-de-un-dron

[10] Vodafone demuestra cómo se puede geolocalizar y controlar un dron usando una red 4G. Consultado el 04/05/2020. https://www.xatakamovil.com/conectividad/vodafone-demuestra-como-se-puede-geolocalizar-y-controlar-un-dron-usando-una-red-4g

[11] Vodafone realiza el primer vuelo de drones en entorno urbano con control 5G. Consultado el 04/05/2020. https://cincodias.elpais.com/cincodias/2020/01/30/companias/1580393686_243834.html

[12] SESAR: Cielo Único Europeo. Consultado el 04/05/2020. https://www.ineco.com/webineco/mercados/navegacion-aerea/sesar-cielo-único-europeo

[13] The AirMap Platform. Consultado el 04/05/2020. https://www.airmap.com/platform/

[14] Drone software for industry innovators. Consultado el 04/05/2020. https://www.dronedeploy.com

[15] Precios para el servicio de DroneDeploy. Consultado el 04/05/2020. https://www.dronedeploy.com/pricing.html

[16] Desarrollo AirMap. Consultado el 04/05/2020. https://www.airmap.com/developers/

[17] Programa para hacer diagramas UML online. Consultado el 07/05/2020 https://www.lucidchart.com/pages/es/ejemplos/diagrama-uml#:~:text=Utilizado%20por%20desarrolladores%20y%20profesionales,%2C%20un%20proceso%20de%20negocio).

[18] *Gilles Vanwormhoudt, Oliver Caron, Bernard Carré*. Aspectual templates in UML. Publicado en julio de 2019. Consultado el 08/05/2020.

[19] Qué es un actor em UML. Consultado el 07/05/2020 (actualización del 5 de agosto de 2019). https://es.wikipedia.org/wiki/Actor_(UML)

[20] Qué es un diagrama de clases UML. Consultado el 08/05/2020. https://www.lucidchart.com/pages/es/tutorial-de-diagrama-de-clases-uml

[21] Qué es un diagrama de Casos de Uso. Consultado el 08/05/2020. http://stadium.unad.edu.co/ovas/10596_9839/diagramas_de_casos_de_uso.html

[22] Qué es MEAN Stack. Consultado el 09/05/2020. https://es.wikipedia.org/wiki/MEAN#Componentes

[23] Qué es MEAN: desarrollo full-stack en JavaScript. Consultado el apartado de MongoDB el 09/05/2020. https://platzi.com/blog/que-es-mean-full-stack-javascript/

[24] Qué es Express.js. Consultado el 09/05/2020. https://www.it-swarm.dev/es/node.js/que-es-express.js/1069367509/

[25] MEAN Stack, el pura sangre de los Stack. Consultado el apartado de Angular el 10/05/2020. https://www.digitaltechinstitute.com/mean-stack/

[26] Modelo cliente servidor. Consultado el 14/05/2020. https://blog.infranetworking.com/modelo-cliente-servidor/

[27] Frontend y Backend. Consultado el 14/05/2020. https://devcode.la/blog/frontend-y-backend/

[28] Programación por capas. Consultado el 14/05/2020. https://www.virtuniversidad.com/greenstone/collect/informatica/archives/HASH0195.dir/doc.pdf

[29] Explicación del patrón MVC. Consultado el 14/05/2020. https://www.guidacode.com/2017/angularjs/explicacion-del-patron-mvc-en-angularjs/

[30] Qué es la capa de presentación. Consultado el 14/05/2020 (actualización del 5 de agosto de 2019). https://es.wikipedia.org/wiki/Capa_de_presentación#cite_note-Nivel_de_presentación-2

[31] Descripción de capas lógicas. Consultado la capa de negocio el 14/05/2020) https://docs.oracle.com/cd/E19528-01/820-0888/aaubb/index.html

[32] Lenguajes de programación en 2020: Los esenciales para este año. Consultado el 15/05/2020. https://www.tokioschool.com/noticias/lenguajes-de-programacion-2020-esenciales-para-este-ano/

[33] *Brad Dayley, Brendan Dayley, Caleb Dayley*. Nodejs, MongoDN and Angular web development. ISBN-13: 978-0-13-465553-6. Consultado el 15/05/2020

[34] MongoDB: Qué es, cómo funciona y cuando podemos usarlo (o no). Consultado el 17/05/2020. https://www.genbeta.com/desarrollo/mongodb-que-es-como-funciona-y-cuando-podemos-usarlo-o-no

[35] NoSQL vs SQL; principales diferencias y cuando elegir cada una de ellas. Consultado el 17/05/2020. https://pandorafms.com/blog/es/nosql-vs-sql-diferencias-y-cuando-elegir-cada-una/

[36] Qué es ObjectId. Consultado el 17/05/2020. https://docs.mongodb.com/manual/reference/method/ObjectId/

[37] Node.js vs. Spring Boot ¿cuál elegir? Consultado el 18/05/2020. https://medium.com/@ktufernando/node-js-vs-spring-boot-cuál-elegir-5a687cd1abae

[38] Single-page Application. Consultado el 19/05/2020 (actualización del 29 de mayo de 2020). https://es.wikipedia.org/wiki/Single-page_application

[39] Qué es el Stack MEAN y cómo escoger el mejor para ti. Consultado 25/05/2020. https://www.campusmvp.es/recursos/post/Que-es-el-stack-MEAN-y-como-escoger-el-mejor-para-ti.aspx

[40] Qué es Java. Consultado el 25/05/2020. https://desarrolloweb.com/articulos/497.php

[41] Estructura de un DatagramPacket. Consultado el 25/05/2020. https://www.oreilly.com/library/view/javatm-network-programming/0201710374/0201710374_ch05lev1sec2.html

[42] Qué es JSON. Consultado el 28/05/2020. https://www.hostinger.es/tutoriales/que-es-json/

[43] Qué es un DatagramSocket. Consultado el 30/05/2020. https://docs.oracle.com/javase/7/docs/api/java/net/DatagramSocket.html

[44] Qué es un DatagramPacket. Consultado el 30/05/2020. https://docs.oracle.com/javase/7/docs/api/java/net/DatagramPacket.html

[45] Software de control de versiones para equipos profesionales. Consulado el 01/06/2020. https://bitbucket.org/product/es/version-control-software

[46] Principios básicos de Git. Consultado el 02/06/2020. https://www.atlassian.com/es/git

[47] Librería Mapbox. Consultado el 02/06/2020. https://docs.mapbox.com/mapbox-gl-js/api/

[48] Introducción a Mongoose para MongoDB y NodeJS. Consultado el 02/06/2020. https://code.tutsplus.com/es/articles/an-introduction-to-mongoose-for-mongodb-and-nodejs--cms-29527#:~:text=Mongoose%20es%20una%20biblioteca%20de,definición%20del%20esquema%20del%20modelo.

Anexo

En este capítulo sirve de complemento el apartado de código frontend y se expondrá el código de las funciones que se indican. Finalmente, se expondrá el código modificado para el software *ArduSim* explicado en su desarrollo del código.

```
checkDronesActivos(){

  //Obtiene la fecha de invocación de la función
  let fechaInvocacion: number = Date.now() - 10000;
  let length = this.cordServ.cords.length - 1;



  let dronesActivos: String[] = [];

  while(true){

    let dron = this.cordServ.cords[length].dron;

    var aux = this.cordServ.cords[length]._id
    var aux1 = mongoose.Types.ObjectId(aux);

    if(fechaInvocacion < aux1.getTimestamp().getTime()){


      dronesActivos[dron] = dron.toString();


      if(this.drones[dron] == undefined){

        let a: Alert = new Alert();

        a.type = this.tipos[dron];
        a.mensaje = "Se ha detectado un nuevo dron con id: " + dron;

        this.alerts.push(a);

        setTimeout(()=>{
          this.close(a);
        },10000);

        let lon = this.trayectos[dron].coordFin.lon;
        let lat = this.trayectos[dron].coordFin.lat;

        this.colocarMarcador(lon, lat, dron);
        this.mapa.flyTo({center: [lon, lat], zoom: 17});
      }



    }else{
      console.log("Drones Activos: " + dronesActivos);

      let length = this.drones.length;

      let i = 0;
      while(i<length){
        console.log("Entro en while")


        if(dronesActivos[this.drones[i]]==undefined){

          if(this.drones[i]!=undefined){
            console.log("Marcador y Linea eliminada para dron" + this.drones[i]);

            let a = new Alert();

            a.type = 'secondary';
            a.mensaje = "Se ha desconectado el dron con id: " + this.drones[i];

            this.alerts.push(a);
```

```
465
466                 setTimeout(()=>{
467                   this.close(a);
468                 },10000);
469
470                 this.mapa.removeLayer('trace'+this.drones[i]);
471                 this.mapa.removeSource('trace'+this.drones[i]);
472
473                 this.mapa.removeLayer('points'+this.drones[i]);
474                 this.mapa.removeSource('points'+this.drones[i]);
475
476                 this.mapa.removeImage('pulsing-dot' + this.drones[i]);
477
478                 this.drones.splice(i, 1);
479
480                 this.trayectos[this.drones[i]].coords = [];
481
482                 i--;
483               }
484             }
485             i++;
486           }
487           return;
488         }
489         length--;
490       }
491     }
492
```

***Ilustración 60**. código checkDonresActivos().*

```
157 rellenarTrayectos(){
158   this.trayectos = [];
159   let i = this.cordServ.cords.length -1;
160
161   while(true){
162     var aux = this.cordServ.cords[i]._id;
163     var aux1 = mongoose.Types.ObjectId(aux);
164
165
166     if(this.fechaActual > aux1.getTimestamp().getTime()){
167       return;
168     }
169
170     let dron = this.cordServ.cords[i].dron;
171
172     if(this.trayectos[dron] == undefined){
173       let tra = new Trayectos();
174       this.trayectos[dron] = tra;
175       this.trayectos[dron].coordInicio = this.cordServ.cords[i];
176       this.trayectos[dron].coordFin = this.cordServ.cords[this.cordServ.cords.length-1];
177     }
178
179     this.trayectos[dron].coords.push(this.cordServ.cords[i]);
180
181     console.log("Tamaño Proyecto!" + this.trayectos[dron].coords.length)
182
183     i--;
184   }
185 }
186
```

***Ilustración 61**. Código rellenarTrayecto().*

```
actualizarLineaMarcador(){
  for(let i of this.trayectos){
    console.log("Tamaño de los trayectos :" + i.coords.length);
  }

  for(let i in this.drones){


    let tra = this.trayectos[i];

    let lon = tra.coords[0].lon;
    let lat = tra.coords[0].lat;



    let coordsLocal: number[][] = [];

    for(let i of tra.coords){
      coordsLocal.unshift([i.lon, i.lat]);
    }

    this.modificarLinea(coordsLocal, i);
    this.animarMarker(lon, lat, i);
  }
}
```

***Ilustración 62**. Código actualizarLineaMarcador().*

```
ordenarCoordenadasPorDron(): Coordenadas[]{
  let ordenado = this.cordServ.cords.sort(function(a,b){
    if(a.dron > b.dron){
      return 1;
    }
    if(a.dron < b.dron){
      return -1;
    }
    return 0;
  });

  return ordenado;
}
```

***Ilustración 63**. Código ordenarCoordenadasPorDron().*

```
coordenadasToTrayectos(ordenado){

  let contadorMismoTrayecto: String[] = [];

  for(let i in ordenado){

    if(i == (ordenado.length - 1).toString()){
      //Se guarda en la variable aux las coordenadas del trayecto
      let aux: Coordenadas[] = [];
      for(let a in contadorMismoTrayecto){
        let aux1 = +contadorMismoTrayecto[a];
        aux.push(this.cordServ.cords[aux1]);
      }

      //Se crea un nuevo trayecto para introducirle las coordenadas
      let trayectoAux: Trayectos = new Trayectos();
      trayectoAux.coords = aux;
      this.trayectos.push(trayectoAux);
      contadorMismoTrayecto = [];

      return;
    }

    let dateActual = this.getTimestampDateUltimaCoord(ordenado[i]);
    let aux = +i + 1;
    let datePosterior = this.getTimestampDateUltimaCoord(ordenado[aux]);

    console.log(datePosterior - dateActual)

    if(ordenado[i].dron == ordenado[aux].dron){
      //Comprobar si la coordenada siguiente se ha guardado en la BDA tiempo después
      if(datePosterior - dateActual > 1000){

        //Se guarda en la variable aux las coordenadas del trayecto
        let aux: Coordenadas[] = [];
        for(let a in contadorMismoTrayecto){
          let aux1 = +contadorMismoTrayecto[a];
          aux.push(this.cordServ.cords[aux1]);
        }

        //Se crea un nuevo trayecto para introducirle las coordenadas
        let trayectoAux: Trayectos = new Trayectos();
        trayectoAux.coords = aux;
        this.trayectos.push(trayectoAux);
        contadorMismoTrayecto = [];

      }else{
        //Si las coordenadas son consecutivas se almacena el índice para posteriormente guardarlas en el trayecto
        contadorMismoTrayecto.push(i);
      }

    }else{
      if(contadorMismoTrayecto.length > 1){

        //Se guarda en la variable aux las coordenadas del trayecto
        let aux: Coordenadas[] = [];
        for(let a in contadorMismoTrayecto){
          let aux1 = +contadorMismoTrayecto[a];
          aux.push(this.cordServ.cords[aux1]);
        }

        //Se crea un nuevo trayecto para introducirle las coordenadas
        let trayectoAux: Trayectos = new Trayectos();
        trayectoAux.coords = aux;
        this.trayectos.push(trayectoAux);
        contadorMismoTrayecto = [];
      }
    }
  }
}
```

***Ilustración 64**. Código coordenadasToTrayecto().*

```
295  mostrarLinea(tra: Trayectos, boton){
296
297
298    if(boton.value == "Go"){
299
300      boton.value = "Pause";
301      boton.innerText = "Pause";
302
303      window.clearInterval(this.timer);
304      this.i = 0;
305
306      this.mapa.removeLayer('trace');
307      this.mapa.removeSource('trace');
308
309      this.inicializarLineaDron(tra);
310
311      this.arrayHeight = [];
312
313      this.cords = [];
314
315      let lon = tra.coordInicio.lon;
316      let lat = tra.coordInicio.lat;
317
318      this.mapa.flyTo({center: [lon, lat], zoom: 15});
319      this.mapa.setPitch(30);
320
321      for(let i of tra.coords){
322        this.cords.push([i.lon, i.lat]);
323        this.arrayHeight.push(i.height);
324      }
325
326
327
328      let coordsLocal: number[][] = [];
329
331      this.timer = setInterval(()=>{
332        if(this.i < this.cords.length){
333          console.log(this.timer)
334
335          this.height = this.arrayHeight[this.i];
336          this.lonImprimir = +this.cords[this.i][0].toFixed(5);
337          this.latImprimir = +this.cords[this.i][1].toFixed(5);
338
339          coordsLocal.push(this.cords[this.i])
340          this.modificarLinea(coordsLocal);
341          this.mapa.panTo(this.cords[this.i]);
342          this.i++;
343        }else{
344          this.cords = [];
345          window.clearInterval(this.timer);
346          this.timer = undefined;
347          boton.value = "Go"
348          boton.innerText = "Go";
349        }
350      },50);
351
352    }else if(boton.value == "Pause"){
353      boton.value = "Continue";
354      boton.innerText = "Continue";
355      window.clearInterval(this.timer);
356    }else{
357      boton.value = "Pause";
358      boton.innerText = "Pause";
359
360      let coordsLocal: number[][] = [];
361
362      let aux = 0;
363
364      while(aux < this.i ){
365        coordsLocal.push([tra.coords[aux].lon, tra.coords[aux].lat]);
366        aux++;
367      }
368
369
370      this.timer = setInterval(()=>{
371        if(this.i < this.cords.length){
372          console.log(this.timer)
373
374          this.height = this.arrayHeight[this.i];
375          this.coordenadasImprimir = this.cords[this.i];
376
377          coordsLocal.push(this.cords[this.i])
378          this.modificarLinea(coordsLocal);
379          this.mapa.panTo(this.cords[this.i]);
380          this.i++;
381        }else{
382          this.cords = [];
383          window.clearInterval(this.timer);
384          this.timer = undefined;
385          boton.value = "Go"
386          boton.innerText = "Go";
387        }
388      },50);
389    }
390  }
```

***Ilustración 65**. Código mostrarLinea().*

```java
package main;

import java.net.DatagramPacket;

public class ClientThread extends Thread {

    public void run() {
        //Clase que envia datos
        ArduSim ardusim = API.getArduSim();
        int numUAVs = ardusim.getNumUAVs();
        Copter[] copters = new Copter[numUAVs];
        InetAddress hostServidor;
        try {
            hostServidor = InetAddress.getByName(Param.dirIP);
            DatagramPacket[] datagrams = new DatagramPacket[numUAVs];
            byte[] buf = new byte[1500];
            for(int i = 0; i<numUAVs; i++)
            {
                copters[i] = API.getCopter(i);
                datagrams[i] = new DatagramPacket(buf, buf.length,  hostServidor, Param.numPort);
            }

            try{
                DatagramSocket s = new DatagramSocket();

                Location2DGeo loc;
                ByteBuffer buffer = ByteBuffer.allocate(Long.BYTES + 5 * Double.BYTES);
                while(!ardusim.isExperimentFinished())
                {
                    for(int i = 0; i<numUAVs; i++)
                    {
                        buffer.clear();
                        buffer.putLong(copters[i].getID());
                        loc = copters[i].getLocationGeo();
                        buffer.putDouble(loc.latitude);
                        buffer.putDouble(loc.longitude);
                        buffer.putDouble(copters[i].getAltitude());
                        buffer.putDouble(copters[i].getSpeed());


                         byte[] msg = buffer.array();
                         datagrams[i].setData(msg);
                         s.send(datagrams[i]);
                     }

                     API.getArduSim().sleep(Param.frequency);
                 }

                 s.close();

             }catch(SocketException s) {
                 System.out.print("SocketError");
             }catch(Exception e) {
                 System.out.println("Error");
             }
         } catch (UnknownHostException e1) {
             e1.printStackTrace();
         }

     }
}
```

***Ilustración 66**. Clase ClientThread*