UNIVERSIDAD INTERNACIONAL DE LA RIOJA

Universidad Internacional de La Rioja (UNIR)

Escuela de Ingeniería

Grado en Ingeniería Informática

Desarrollo de una iOS App para la generación de rutas de drones

Ubicación del código fuente:
http://github.com/josebrana/1-COMPandGO-DESARROLLO

Trabajo Fin de Grado
presentado por: Braña Pérez, José Antonio
Director/a: Sánchez Figueroa, Juan Carlos

Ciudad: Oviedo (Asturias)
Fecha: Julio 2016

A María y Alicia, por estar siempre ahí y permitirme estar tanto aquí.

Resumen

Nota: El propósito de este TFG es la consecución de una aplicación para *iPad* mediante la cual se puedan planificar rutas para vuelos de vehículos no tripulados pilotados a distancia (drones), generando un plan de vuelo operacional preliminar, imprescindible para cumplir con los requisitos exigidos por la Agencia Estatal de Seguridad Aérea (AESA). Así mismo, el sistema permitirá la captura, previsualización y reproducción de fotografías y vídeos de alta calidad.

A tal efecto, se desarrollará un producto software (en adelante ***COMPandGo***) sobre Xcode en lenguaje Objective-C, orientado al gobierno de un dron mediante su mando de control remoto y el mencionado *iPad*.

Se utilizará Scrum como metodología de trabajo a lo largo de todo el desarrollo.

Palabras Clave: Drones, Rutas, Planificación, AESA, *DJI*

Abstract

Nota: The purpose of this Final Degree Project is the achievement of an *iPad* application through which flight paths for flights of unmanned aerial vehicles controlled by remote control (drones) can be planned, generating a preliminary operational flight plan, essential to comply with the requirements of the Aerial Security State Agency (AESA). Furthermore, the system will enable you to capture, preview and playback high-qualty images and vídeos.

To this end, a software product (from now on named ***COMPandGO***) based on Xcode in Objective-C language will be developed, geared towards the control of a dron through the remote control and the aforementioned *iPad*.

Scrum will be used as work methodology along the entire process.
Keywords: Drones, Flight paths, Planning, AESA, *DJI*

AGRADECIMIENTOS

A María y Alicia, de nuevo, por convertir mi mundo un lugar mucho mejor.
A José Manuel, por su infinita paciencia y esfuerzo a lo largo de estos años.
A Juan Carlos Sánchez Figueroa, por haberme dirigido este TFG.
A mis padres y amigos, por hacer de mí quien soy.



ÍNDICE GENERAL

ÍNDICES DE IMÁGENES, TABLAS Y CÓDIGOS

1. Índice de imágenes

2. Índice de Tablas

3. Índice de Códigos

LISTADO DE ACRÓNIMOS

ACRÓNIMO	SIGNIFICADO
AESA	Agencia Estatal de Seguridad Aérea
ATS	Auxiliar de Transporte Sanitario
AUVSI	Asociación Internacional de Sistemas de Vehículos No tripulados
MVC	Modelo-Vista-Controlador
RAE	Real Academia Española
RF	Requisitos Funcionales
RNF	Requisitos No Funcionales
VPA	Velocidad en Piloto Automático

ACRONYM	MEANING
AGL	Above Ground Level
API	Application Programming Interface
BSD	Berkeley Software Distribution
BVLOS	Beyond Visual Line Of Sight
CMS	Content Management Systems
CPU	Central Processing Unit
DJI	Dà-Jiāng Innovations
DNG	Digital Negative
EVLOS	Extended Visual Line Of Sight
fps	Frames Per Second
FPV	First Person View
GCC	GNU Compiler Collection
GNU	GNU's Not Unix
GPS	Global Positioning System
HD	High Definition
ID	Identification
IDE	Integrated Development Environment
IFR	Instrument Flight Rules
IMC	Instrument Meteorological Conditions
iOS	*iPhone* Operating System
LCD	Liquid Crystal Display
LED	Light Emitting Diode
LiPo	Lithium Polymer
mAh	Milli Amp Hour
MTOM	Maximum Take-Off Mass
MVC	Model-View-Controller
NOTAM	Notice To Air Men
OS	Operating System
QR Code	Quick Response Code
SD	Secure Digital
SDK	Software Development Kit
URL	Uniform Resource Locator
UUID	Universally Unique Identifier
USB	Universal Serial Bus
UX	User Experience
VLOS	Visual Line Of Sight
VMC	Visual Meteorological Conditions

1 INTRODUCCIÓN

1.1 Contexto del TFG

A la hora de buscar un elemento de desarrollo para la elaboración de este Trabajo de Fin de Grado, se han prospectado diferentes mercados tecnológicos emergentes, en busca de áreas de futuro o nichos de negocio sobre los que poder planificar y desarrollar un software que aporte valor. En este terreno, el mercado de los vehículos no tripulados pilotados a distancia (drones) parecía un firme candidato, ya que aunaba un frenético desarrollo tecnológico (que pronosticaba su aplicación en multitud de entornos, más allá de su origen militar) junto con una popularización de su uso para la resolución de problemas en ámbitos cotidianos y domésticos.

Como recoge Jiménez, M. (2015) [10], según un informe de AUVSI (Asociación Internacional de Sistemas de Vehículos No tripulados) de abril de 2015, se estima que en 2025 existirán en Europa unos 100.000 puestos de trabajo relacionados con los drones. En las fechas de ese estudio, en España la Agencia Estatal de Seguridad Aérea (AESA) tenía registradas unas 100 empresas que reunían las características necesarias para realizar actividades profesionales. En la actualidad, tan sólo un año después, esa cifra se ha disparado hasta los casi 1.500 operadores de drones (AESA [18]).

La Comisión Europea estima que cerca del año 2025 los drones supondrán una décima parte del sector de la aviación en Europa [12]. En otras palabras, un enorme potencial de negocio. Esta nueva realidad queda reflejada en un informe de *MarketsandMarkets* del primer trimestre de este año 2016, que sostiene que el mercado de los drones moverá unos 4.800 millones de euros en 2020, lo que supondrá un crecimiento del 33% en los próximos 4 años [11].

En la actualidad en España existe un marco legal transitorio, a la espera de un real decreto que se encuentra en fase de tramitación parlamentaria y que contribuirá al desarrollo del sector, dado su carácter más permisivo. En estos momentos, los drones con un peso máximo al despegue de 25 kg y que realicen vuelos de carácter técnico-científico, sólo pueden usarse en espacios aéreos no urbanos y a una distancia superior a 8 km de un aeropuerto, no considerándose legales aquellos vuelos sin autorización explícita y que se realicen en espacios urbanos, que sobrevuelen aglomeraciones de personas o que se efectúen por la noche [17].

Presumiblemente en el nuevo marco legal todas estas restricciones dejarán de serlo, posibilitando el uso de drones para la realización legal de actividades hasta ahora no permitidas.

1.2 Problemática actual

El desarrollo a lo largo de los últimos años de nuevas tecnologías de aeronavegabilidad en el sector de los vehículos aéreos no tripulados y pilotados a distancia, ha generado un nuevo mercado emergente de servicios, que en la actualidad experimenta un vertiginoso crecimiento. Este proceso ha provocado una popularización del uso de estas novedosas tecnologías, no sólo en el ámbito profesional si no también en el doméstico.

Como consecuencia de este contexto, surge la necesidad de establecer un marco normativo de regulación legal. En España la autoridad aeronáutica civil que establece el mecanismo regulatorio es el Ministerio de Fomento a través de la Agencia Estatal de Seguridad Aérea (AESA).

En la actualidad, el uso profesional de drones, está sujeto a una serie de normas y requisitos legales que deben cumplirse tanto por parte de las empresas operadoras como de los pilotos adscritos a dichas empresas.

Uno de estos requisitos formales obliga al operador de drones a disponer, previamente a la realización de un vuelo, de un Plan de Vuelo Operacional [17]. Dicho plan de vuelo debe incluir, entre otra documentación, información relativa al espacio aéreo en el que se realiza el vuelo.

Si bien es cierto que actualmente en el mercado existen diversas soluciones tanto para planificar y ejecutar de manera autónoma rutas de vuelo, como para definir acciones concretas a realizar, existe un vacío en tanto y cuanto ninguna de ellas genera documentación específica para el cumplimiento de las normas legales establecidas dentro del territorio español. Es más, tampoco encontramos que las perspectivas de futuro sean muy halagüeñas al respecto, ya que la mayoría de las Apps que dominan el mercado son extranjeras y tendrían muy complicado optimizar el rendimiento económico del desarrollo de esa cualidad. Al ser Apps concebidas para su empleo por un usuario genérico en cualquier lugar del mundo, no les resultaría muy rentable añadir a sus Apps características que no fueran universales. La capacidad de generar la documentación legal difiere de un lugar del

mundo a otro. Por ejemplo, el marco legal que rige el espacio aéreo en España es distinto al de otros territorios. Una App dirigida a un ámbito global tendría que generar una documentación válida para cualquier país. Supondría muchísimo trabajo y altísimos gastos que serían muy difíciles de amortizar (como veremos en el estado del arte, algunas de las Apps analizadas son de pago y ya tiene un coste muy elevado respecto al precio de la App media). Consideramos que este servicio es un objetivo más abarcable para una App dirigida a un ámbito geográfico más concreto (el territorio de España en nuestro caso).

COMPandGO pretende ser una herramienta útil para la elaboración de la documentación oficial complementaria a los vuelos de carácter científico-técnico realizados por drones con un peso máximo al despegue de 25 kg. Complementando a esta característica, ***COMPandGO*** también podrá ser utilizada para la planificación de las rutas de vuelo y para la captación de imágenes de vídeo y fotografía de alta calidad.

1.3 Método y fases del trabajo

1. Introducción

Las fases en las que se va a dividir el desarrollo de este TFG son las siguientes:

1.3.1 Estudio del Problema (Magnitud y Fases)

1. Introducción | 1.3 Método y Fases de trabajo

En primer lugar se considera necesario conocer cuál es la complejidad actual, la magnitud del problema. Dicho problema plantea unas necesidades que se tratarán de abarcar con unos objetivos a cumplir.

Para ello se realizará un análisis del contexto en el que nos vamos a mover y un estudio preliminar del mismo, partiendo de un conocimiento de otras aplicaciones similares, tratando de dimensionar los obstáculos a los que ellas se han enfrentado en su día y contemplando en todo momento que a ese dimensionamiento habrá que añadir el aporte de valor de nuestra solución.

A continuación surge la necesidad de crear una planificación previa del trabajo a realizar, la cual tiene por fuerza que ajustarse a los plazos de entrega descritos en el reglamento de desarrollo del TFG. Siguiendo la metodología de trabajo recomendada por la UNIR, se ha elaborado pues un plan de trabajo tentativo previo, dividido en fases e hitos de contenido a presentar en distintos plazos de entrega escalonados, asociados a un cronograma.

1.3.2 Análisis y decisiones iniciales

Un análisis más específico que el anterior nos ayudará a especificar los requisitos que vamos a necesitar cubrir para el desarrollo de la aplicación. Se acometerá entonces la primera decisión de calado, como es la definición del entorno de desarrollo (hardware y software que se emplearán).

1.3.3 Formación previa y certificaciones

Más allá de la formación recibida a lo largo de los años de estudio de Ingeniería Informática y la correspondiente experiencia acumulada, la App desarrollada constituye la primera incursión para el autor de este TFG en el mundo de la programación con Objective-C y Xcode, por lo que la curva de aprendizaje ha sido pronunciada y ha supuesto todo un reto personal. A la considerable cantidad de tiempo que ya de por sí implica un trabajo como el que se acomete, ha habido que añadir una importante carga de formación personal para alcanzar el grado de conocimiento necesario para acometer la tarea correspondiente.

Así mismo, se ha considerado altamente relevante para el desarrollo de este TFG conseguir la certificación oficial como piloto de drones con peso máximo al despegue de 25 kg que exige la Agencia Estatal de Seguridad Aérea (AESA).

Los requisitos exigidos para obtener esa certificación fueron:

- Certificado de formación teórica de nivel avanzado, emitido por una organización de formación aprobada (ATO) por AESA, superando el correspondiente curso al efecto. Para ello se formalizó la matrícula en la escuela de pilotos comerciales *Fly School* (Madrid), donde se superó el curso de nivel avanzado.
- Presentar un certificado médico de clase 2.
- Acreditar que se disponen de los conocimientos adecuados de la aeronave que se desee pilotar y de su pilotaje, por medio de un documento que puede ser emitido por el operador, por el fabricante de la aeronave o una organización autorizada por éste, o por una organización de formación aprobada. En este caso particular, se superó la correspondiente formación y exámenes teórico-prácticos en la organización autorizada *RC Innovations* (Bilbao), pilotando un dron *DJI Phantom 4*.

A estos efectos, tras recibir la formación teórico-práctica necesaria y superar los pertinentes exámenes y reconocimientos médicos, Se logró el objetivo de la certificación oficial para realizar vuelos con drones de carácter científico-técnico dentro del marco legal vigente.

1.3.4 SCRUM como metodología de trabajo

1. Introducción | 1.3 Método y Fases de trabajo

Uno de los retos a los que nos enfrentamos en la realización de este TFG es la utilización de una metodología conveniente. Una vez concretados aquellos objetivos a conseguir, toca hacer una reflexión sobre la correcta aplicación de aquellas técnicas de la Ingeniería de Software como disciplina que deberían ser aplicadas a la consecución de los mismos. En principio, y basándonos en los conocimientos obtenidos en las asignaturas relacionadas con la Ingeniería del Software previamente estudiadas en el Grado, se entiende que la metodología SCRUM es una de las posibles soluciones adecuadas.

SCRUM se basa en una metodología ágil, ligera y flexible, características que en principio parecen adecuadas para la elaboración de este TFG, al tratarse de un proyecto desarrollado por un equipo de tan sólo dos personas: el autor y el director del proyecto. También parece adecuado la aplicación de técnicas de SCRUM vinculadas a las entregas formales que periódicamente se realizan a lo largo de los meses empleados en el desarrollo.

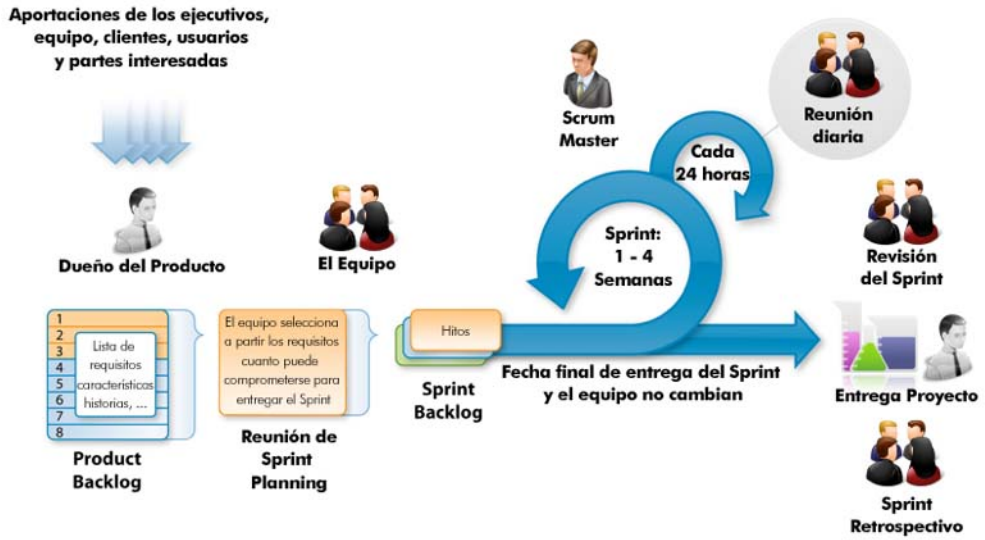


Imagen 1: SCRUM: ciclo de un sprint.

En nuestro caso, nos basamos en las técnicas de SCRUM para, a través de ingeniería concurrente, realizar nuestro proyecto a través de una estrategia de desarrollos

incrementales, solapando algunas de las fases del desarrollo en lugar de realizarlas unas tras otras de manera iterativa en ciclos secuenciales o de cascada.

El "*Manifiesto por el Desarrollo Ágil del Software*" en el que apoyan las técnicas de SCRUM recoge algunas de las ideas con las que el desarrollo de este TFG puede estructurarse. Así por ejemplo, se intenta priorizar el propio desarrollo sobre la documentación exhaustiva, dando preferencia también la experiencia de usuario (UX) sobre los procesos y las herramientas, e intentando a lo largo del proceso global tener una alta respuesta al cambio, por encima de un seguimiento profundo de un determinado plan de trabajo.

1.3.4.1 Definición de Roles y Sprints

1. Introducción | 1.3 Método y Fases de trabajo | 1.3.4 SCRUM como metodología de trabajo

En cuanto a la definición de ROLES, SCRUM define principalmente tres:

- **SCRUM MANAGER (o SCRUM MASTER):** es el responsable de asegurar que el marco de SCRUM es entendido y adaptado. En nuestro caso, este papel lo ejerce el director del TFG, D. Juan Carlos Sánchez Figueroa.
- **PRODUCT OWNER**: se trata del cliente final que demanda el proyecto y que también es responsable de definir aquellos requisitos (funcionales y no funcionales) a desarrollar. Al igual que anteriormente, este ROL lo desempeñará en este caso D. Juan Carlos Sánchez Figueroa.
- **TEAM**: se trata del equipo que va a desarrollar el proyecto y que realizará las funciones de: análisis, diseño, implementación, integración y validación experimental. Este ROL lo desempeñará íntegramente el autor del TFG.

El desarrollo iterativo incremental en SCRUM se basa en la técnica de los **SPRINTS**. Un SPRINT es un período de tiempo, normalmente definido entre dos y cuatro semanas, a lo largo del cual el equipo desarrolla un incremento de software potencialmente entregable.

El ciclo de vida de cada uno de los SPRINTS se subdivide en tres fases:

- El **SPRINT PLANNING**: se trata de una reunión del TEAM con el PRODUCT OWNER en la que se definen los requisitos del **PRODUCT BACKLOG** (listado de requisitos) que se desean completar al final del SPRINT en cuestión. A partir de ese momento, el TEAM determina su viabilidad y los recoge en el llamado **SPRINT BACKLOG** (listado de requisitos del SPRINT). En nuestro caso, el SPRINT

PLANNING se realiza en base a los requisitos exigidos en cada una de las entregas parciales previas al depósito final, que se acuerdan con el director del TFG.

- **EJECUCIÓN DEL SPRINT**: en nuestro caso tenemos cuatro entregas parciales previas al depósito del TFG que se subdividen a su vez en dos SPRINTS, definidos con el director del TFG, habitualmente mediante correo electrónico. Durante este proceso, el SCRUM MASTER (el director del TFG) supervisa la consecución de los objetivos e interactúa con el TEAM para subsanar posibles inconvenientes, garantizando así la consecución de los objetivos.
- **DEMOSTRACIÓN Y RETROSPECTIVA**: el último día del SPRINT se realiza la entrega parcial formal al director del proyecto (PRODUCT OWNER) mediante correo electrónico, a modo de nueva versión. El PRODUCT OWNER, en base a la entrega, aplica métodos correctivos y perceptivos al proyecto. Por su parte, el autor del TFG (TEAM) analiza retrospectivamente los inconvenientes surgidos previamente a la entrega, intentando evitarlos así en el futuro.

1.4 Objetivos

1. Introducción

Los objetivos de la elaboración de este TFG se subdividen en dos grandes bloques:

1.4.1 Objetivo General

1. Introducción | 1.4 Objetivos

El objetivo general de este TFG es el desarrollo de una aplicación para *iPad* dirigida al mercado de operadores y pilotos de vehículos no tripulados pilotados a distancia (drones), que recibirá el nombre de ***COMPandGO***. Como elemento diferenciador respecto al resto de aplicaciones existentes en el mercado dirigidas al mismo ámbito, ***COMPandGO*** permitirá generar un plan de vuelo operacional preliminar, imprescindible para cumplir con los requisitos exigidos por la Agencia Estatal de Seguridad Aérea (AESA) en todos los vuelos de drones con carácter científico-técnico y peso máximo al despegue de 25 kg [17]. Esta característica excepcional se unirá a otras capacidades que son comunes a las Apps más utilizadas por los usuarios (y por tanto las funcionalidades más reclamadas y empleadas), como son la planificación de rutas de vuelo y la captura, previsualización y reproducción de fotografías y vídeos de alta calidad captados por la cámara del dron.

En resumen, ***COMPandGO*** pretende ser una herramienta útil tanto para la elaboración de la documentación oficial complementaria a un vuelo de un dron (de las características

reseñadas) como en la planificación de rutas de vuelo, así como para la captación de imágenes de foto y vídeo de alta calidad.

Imagen 2: El TFG consta del código fuente y de la memoria.

El TFG consta de dos partes: el código fuente de ***COMPandGO*** (que posteriormente al desarrollo se publicará en un repositorio online con licencia abierta) y la memoria descriptiva (el presente documento).

1.4.2 Objetivos Específicos

- Creación de un sección de documentación, que permita al usuario cumplimentar los datos específicos del vuelo a realizar.
- Tener la cualidad de permitir que el usuario pueda previsualizar la documentación generada, para así comprobar que está rellenada a su gusto.
- Conseguir que esa documentación se pueda enviar por email al destinatario que le requiera, adjuntándola al correo electrónico en un archivo de formato Pdf.
- Conseguir acceso a la señal de la cámara del dron, permitiendo capturar imágenes y vídeo de alta calidad.
- Crear una galería del contenido multimedia capturado y almacenado en la tarjeta MicroSD del dron, desde donde se puedan reproducir, seleccionar, borrar o descargar al *iPad* los archivos deseados.
- Presentar una sección de Ayuda, tanto con acceso de carácter local (para aquellas dudas más comunes) como con carácter online (a través de una página web creada al efecto, que trate de solucionar las dudas específicas de cada usuario).
- Diseño de una interface agradable y práctica que permita un cómodo acceso a todas las secciones de la App, presentándolas en un orden lógico con las necesidades habituales de un operador o piloto de drones que vaya a realizar un vuelo.

2 ANÁLISIS DE CONTEXTO Y ESTUDIO PRELIMINAR

La creciente demanda en el mercado de los drones en los últimos años, así como su rápido desarrollo tecnológico, han propiciado un gran crecimiento de las empresas desarrolladoras de soluciones software en este campo.

Dado que se trata de una App iOS, es importante mencionar que el primer filtro de calidad lo establece la propia empresa *Apple* a través de los requisitos que exige para que estás aplicaciones puedan ser distribuidas globalmente desde su tienda online (*App Store*).

Para la realización de un estudio comparativo, se analizarán aquellas aplicaciones cuyas características puedan servir para extraer conclusiones sobre qué cualidades aportarían valor a nuestra aplicación.

Estas características son:

- **Soporte para :** *DJI* se ha convertido en la empresa líder en la fabricación y venta de drones en el mundo. El pasado 1 de marzo, se presentó la cuarta versión de su modelo con mayor cuota de mercado, perteneciente a la familia *Phantom*. De cara a desarrollar un software con perspectiva de futuro se ha valorado la compatibilidad de estas aplicaciones con este novedoso hardware.
- ***iPhone*:** Se valora la compatibilidad con dispositivos móviles de la familia *iPhone*.
- ***iPad*:** De manera análoga, se analiza la compatibilidad con dispositivos de la familia .
- **Gratuita:** Estudio sobre si la aplicación tiene o no un coste.
- **Captura fotográfica:** Valoración de aquellas Apps que ofrecen la posibilidad de realizar fotografía aérea digital.
- **Grabación Vídeo 4K:** Posibilidad de grabar vídeos en alta definición (calidad de hasta 4K) desde la cámara del dron.
- **Pan 360º:** Entendiendo por tal la grabación de un vídeo a la vez que el dron gira 360º alrededor de su eje vertical y en suspensión.
- **TimeLapse Mode:** Esta es una de las características que ofrecen algunos softwares para, de manera automática, generar vídeos a partir de fotografías de un mismo encuadre y que se realizan periódicamente a lo largo de un segmento de tiempo determinado.

- **Galería/Álbum:** Servicio de visionado y reproducción de las imágenes y vídeos previamente capturados por la aplicación, así como la posibilidad de borrado y descarga.
- **WayPoint (Dos puntos):** Posibilidad de definir espacialmente un punto origen y otro destino y ejecutar la misión (vuelo) entre dichos puntos de manera automática.
- **WayPoint (Múltiples puntos):** Servicio análogo al anterior en el que la misión se define por tres o más puntos.
- **Orbitar:** Se entiende por orbitar aquella posibilidad de que dron repita la misión de manera cíclica y hasta que el usuario tome el control manual de la misma.
- **Carga y Grabación de las Misiones:** Posibilidad de almacenar los WayPoints configurados para su posterior recuperación.
- **Generación de datos de Telemetría:** Las Apps que disponen de esta función recogen determina información telemétrica y la muestran por pantalla.
- **Información del espacio aéreo:** Capacidad de informar al usuario de aquellas posibilidades de ejecutar una misión de vuelo en las inmediaciones del espacio aéreo en el que se va a realizar la misión (vuelo).
- **Generación de Plan de Vuelo Provisional:** Capacidad de generar documentación legal relacionada con la misión prevista.
- **Almacenamiento de la información generada:** Posibilidad de almacenar la información telemétrica o del Plan de Vuelo Provisional.
- **Envío por email de la información generada:** Servicio que permite al usuario enviar por correo electrónico la documentación de vuelo generada.

2.1 Estado del Arte

El estudio del estado del arte se realiza en este proyecto mediante una comparativa de algunos de los casos éxito más significativos con características en común con nuestro desarrollo y un análisis posterior del valor que aporta nuestra App ***COMPandGO***.

2.1.1 Comparativas de algunos casos de éxito

De entre la amplia oferta de soluciones, se han seleccionado seis aplicaciones. Son las siguientes:

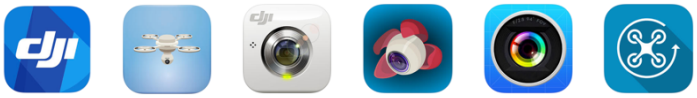
	Nombre	Descripción
	*DJI* Go	Permite conectar un *iPhone* o un con aeronaves de la familia *DJI*. Se trata de la aplicación oficial de la marca y, por tanto, la más utilizada por los operadores.
	DronView	Comprueba la disponibilidad del espacio aéreo dentro del territorio de la República Checa. Los espacios aéreos activos aparecen en pantalla destacados con un color; los inactivos sólo contorneados.
	*DJI* FC40	Aplicación de referencia en el mercado para el modelo *Phantom 1*. Proporciona control remoto de la cámara del dron, permitiendo capturar fotografías, vídeos y, posteriormente, administrarlas mediante álbumes.
	Litchi	Probablemente la aplicación más completa del sector (y una de las más caras). Aporta servicios de configuración y personalización de la mayor parte de los dispositivos y sensores del dron.
	FPV Camera for *DJI*	Otra aplicación muy completa, especialmente para la planificación de rutas y misiones. Permite integración con servicios *Goggles* y dispone de una interface de usuario *touchless*.
	DronePan	Permite la captura de Pan 360º en una sola localización mediante un proceso sencillo de menos de 3 minutos. Requiere de un software adicional para unir las fotografías capturadas en una sola imagen 360º.

Tabla 1: Apps significativas dentro del campo de los drones.

En la actualidad en el *Apple Store*, podemos encontrar referencias a no menos de cincuenta aplicaciones para el control de drones. No obstante, en la mayor parte de los casos estas aplicaciones sólo ofrecen un mínimo número de funcionalidades. En general, las soluciones de mayor aceptación son aquellas que se limitan a proporcionar una única función a cambio de un proceso más ligero y sencillo para su operación. En otros casos, sólo se trata de una adaptación del software genérico de *DJI* con variaciones en sus interfaces gráficas.

Es significativo también que, salvo contadas excepciones, las aplicaciones sólo aportan funcionalidades relacionadas con la operación de los drones, pero no para su adecuación a un determinado marco legal.

Para el análisis comparativo de todos los criterios anteriormente expuestos, se presenta la siguiente tabla:



Nombre de la App	DJI GO	DronView	DJI FC40	Litchi	FPV Camera for DJI	DronePan
Soporte para *DJI Phantom 4*	✓	✗	✗	✓	✓	✗
iPhone	✓	✓	✓	✓	✓	✓
iPad	✓	✓	✗	✓	✓	✓
Gratuita	✓	✓	✓	✗	✗	✓
Captura Fotográfica	✓	✗	✓	✓	✓	✓
Grabación Vídeo 4K	✓	✗	✗	✓	✓	✓
Pan 360º	✗	✗	✗	✓	✓	✓
TimeLapse Mode	✗	✗	✗	✗	✗	✗
Galería/Álbum	✓	✗	✓	✓	✗	✗
WayPoint (Dos puntos)	✗	✗	✗	✓	✓	✗
WayPoint (Múltiples puntos)	✗	✗	✗	✓	✓	✗
Orbitar	✗	✗	✗	✓	✓	✗
Carga y Grabación de las Misiones	✗	✗	✗	✓	✓	✗
Generación de datos de Telemetría	✗	✗	✗	✓	✓	✗
Información del espacio aéreo	✗	✓	✗	✗	✗	✗
Generación de Plan de Vuelo Operacional preliminar	✗	✗	✗	✗	✗	✗
Almacenamiento de la información generada	✓	✗	✗	✓	✓	✗
Envío por email de la información generada	✗	✗	✗	✓	✓	✗

Tabla 2: Comparativa de las Apps seleccionadas.

2.1.2 Aportación de valor de nuestra solución

El principal aporte de valor de la aplicación desarrollada en este TFG (***COMPandGO***) es que permite generar documentación útil para confeccionar el Plan de Vuelo Operacional preliminar, algo que no se ha encontrado actualmente en el mercado de Apps. Esta documentación forma parte de aquella de carácter obligatorio que debe portar el piloto de drones para cumplir con los requisitos exigidos por la Agencia Estatal de Seguridad Aérea (AESA) según el marco legal vigente [17].

De las conclusiones extraídas tras conversaciones con pilotos y operadores de drones certificados por AESA, se ha observado que existe la demanda en el mercado de un software que permitiera realizar las funciones básicas de aeronavegabilidad y fotografía aérea conjuntamente con la generación de la documentación legal vinculada con la operación de los vuelos.

En la actualidad, la mayor parte de los operadores se limita a redactar con un procesador de textos una recopilación de la información del vuelo que van a operar, antes de desplazarse a la localización concreta de trabajo, manejando en ocasiones datos facilitados por terceros y sin un conocimiento concreto de la precisión de los mismos.

Una aplicación como ***COMPandGO*** permitiría al piloto u operador trabajar *in situ*, inspeccionando la zona de vuelo, confeccionar la documentación preliminar manejando los datos exactos que recoge él mismo en la zona de vuelo, almacenarla y enviarla por correo electrónico, para a continuación ejecutar (desde la misma aplicación) la misión previamente planificada.

A modo de reflexión general, parece cada vez más necesario vincular el desarrollo de los softwares de este sector con el cumplimiento estricto de las normas legales que regulan los marcos operacionales en cada uno de los diferentes ámbitos territoriales (espacios aéreos nacionales, europeos o globales). Surge así una oportunidad para los desarrolladores a la hora adaptar los softwares genéricos de control de las aeronaves a los entornos concretos y sus condicionantes normativos.

A continuación se presentan, en relación con las Apps analizadas anteriormente, aquellos servicios que caracterizan a nuestro software:

Nombre de la App	COMPandGO	Nombre de la App	COMPandGO
Soporte para *DJI Phantom 4*	✓	WayPoint (Dos puntos)	✓
iPhone	✗	WayPoint (Múltiples puntos)	✓
iPad	✓	Orbita	✓
Gratuita	✓	Carga y Grabación de las Misiones	✗
Captura Fotográfica	✓	Generación de datos de Telemetría	✗
Grabación Vídeo 4K	✓	Información del espacio aéreo	✗
Pan 360º	✗	Generación de Plan de Vuelo Provisional	✓
TimeLapse Mode	✗	Almacenamiento de la información generada	✓
Galería/Álbum	✓	Envío por email de la información generada	✓

Tabla 3: Sometemos a ***COMPandGO*** al mismo análisis comparativo que a las 6 Apps seleccionadas.

2.2 Especificación de requisitos

2. Contexto y Estudio preliminar

Para la especificación formal de nuestra aplicación ***COMPandGO*** se identificarán y jerarquizarán aquellos requisitos funcionales y no funcionales que definen las características principales de nuestro desarrollo. Posteriormente se realizará un análisis de los casos de uso, identificando los sistemas, los actores y las acciones.

2.2.1 Requisitos funcionales

2. Contexto y Estudio preliminar | **2.2 Especificación de requisitos**

Mediante la siguiente tabla se describe un conjunto de requisitos funcionales que servirán como base de un futuro análisis para la realización de los casos de uso. Junto con el identificador único y la descripción del requisito funcional, se jerarquiza la prioridad de los comportamientos del sistema, asignándoles un valor de 0 a 10, siendo 0 la prioridad mínima de desarrollo e implementación de un requisito, y 10 la máxima.

Número	Requerimiento	Descripción	Prioridad (0-10)
RF-001	**GEOLOCALIZAR Y MOSTRAR MAPA**	Geolocalizar al usuario y mostrar un mapa de su ubicación y su entorno	10
RF-002	**INTERACTUAR GESTUALMENTE CON EL MAPA**	Capacidad de interactuar con el mapa mediante gestos táctiles sobre la pantalla	10
RF-003	**MARCAR RUTA**	Activar el proceso mediante el cual se añaden localizaciones (etapas) para definir una determinada ruta	10
RF-004	**TERMINAR RUTA**	Dar por concluido el proceso anterior	10
RF-005	**BORRAR RUTA**	Eliminar todos los localizadores que definen una determinada ruta	10
RF-006	**CONFIGURA RUTA**	Permite la configuración de todos los parámetros comunes vinculados con la ruta en su conjunto	8
RF-007	**DEFINE ALTITUD**	Establece numéricamente la altitud (en metros) a la que el dron realizará la misión	6
RF-008	**DEFINE VELOCIDAD MÁXIMA**	Establece numéricamente la velocidad máxima (en metros/segundo) a la que el dron realizará la misión	6
RF-009	**DEFINE VELOCIDAD DE PILOTO AUTOMÁTICO**	Define numéricamente la velocidad a la que se desplazará la aeronave en el modo de piloto automático	6
RF-010	**DEFINE ORIENTACIÓN: AUTOMÁTICA**	Establece de modo automático la orientación de la aeronave mientras realiza la misión de vuelo	4
RF-011	**DEFINE ORIENTACIÓN: INICIAL**	Establece la orientación que debe tener el dron en el momento del despegue camino de la primera etapa de la misión	4
RF-012	**DEFINE ORIENTACIÓN: CONTROL REMOTO**	Posibilidad de modificar remotamente la orientación del dron	4
RF-013	**DEFINE ORIENTACIÓN: USANDO ETAPAS**	Define la orientación del dron tomando como referencia la localización de cada una de las etapas de la misión	4
RF-014	**DEFINE FIN DE MISIÓN: VOLVER A BASE**	Sistema mediante el cual el dron vuelve automáticamente a la base de despegue al finalizar la misión	4
RF-015	**DEFINE FIN DE MISIÓN: VOLVER AL PRIMER PUNTO**	Sistema mediante el cual el dron vuelve a la primera estación al finalizar la misión	4
RF-016	**DEFINE FIN DE MISIÓN: ATERRIZAR**	Sistema mediante el cual el dron aterriza al finalizar la misión	4
RF-017	**DEFINE FIN DE MISIÓN: ORBITA**	Sistema mediante el cual el dron ejecuta cíclicamente la misión, hasta que reciba una nueva orden	4
RF-018	**CONFIGURA ETAPA**	Permite la configuración de todos los parámetros comunes vinculados con la etapa en su conjunto	8
RF-019	**HABILITA FOTO**	Asocia a una etapa la orden de capturar una imagen fotográfica	6
RF-020	**HABILITA VÍDEO**	Asocia a una etapa la orden de grabar vídeo	6
RF-021	**AÑADIR ETAPA**	Capacidad de añadir una nueva etapa en una misión	8
RF-022	**BORRAR ETAPAS**	Posibilidad de borrar todas las etapas de una misión	8
RF-023	**CONFIGURAR MISIÓN**	Configurar los ajustes de una misión en una única pantalla	8
RF-024	**EJECUTAR MISIÓN**	Dar inicio a una misión previamente configurada	10
RF-025	**VER GALERÍA**	Mostrar en modo álbum (miniaturas) las fotografías y vídeos capturados	6
RF-026	**SELECCIÓN DE UN CLIP**	Posibilidad de seleccionar un único clip	6
RF-027	**SELECCIONAR TODOS LOS CLIPS**	Posibilidad de seleccionar todos los clips	4
RF-028	**PREVISUALIZAR CLIP**	Capacidad de reproducir un clip en pantalla	6
RF-029	**BORRAR CLIPS**	Eliminar los clips seleccionados	6
RF-030	**DESCARGAR CLIPS**	Descargar los clips seleccionados a la memoria del dispositivo móvil	4
RF-031	**FORMULARIO DE INFORMACIÓN DE LA MISIÓN**	Capacidad de introducir a través de un formulario, información relacionada con: • Información del piloto • Información del operador • Información de la aeronave • Información del tipo de vuelo (operación normal o vuelo especial) • Información del tipo de operación • Descripción y objetivo de la operación • Localización • Restricciones de la operación • Observaciones meteorológicas • Descripción y objetivo de la operación	8
RF-032	**GENERAR DOCUMENTO DE PLAN DE VUELO PROVISIONAL PRELIMINAR**	Generar un documento a partir de los campos del formulario anterior	8
RF-033	**COMPROBACIÓN DE DOCUMENTACIÓN EN FORMATO PDF**	Posibilidad de previsualizar la documentación generada	4
RF-034	**ENVIAR INFORMACIÓN POR EMAIL**	Capacidad de enviar un correo electrónico a partir del documento generado anteriormente	6
RF-035	**VER AYUDA**	Acceso a una pantalla con instrucciones de uso de la aplicación	4

Tabla 4: Requisitos funcionales.

2.2.2 Requisitos no funcionales

La siguiente tabla complementa a la anterior, enumerando una serie de atributos de calidad que no aportan funcionalidad en sí mismos, pero que sí definen el producto final.

Número	Requerimiento	Descripción	Prioridad (0-10)
RNF-001	**COMPATIBILIDAD CON *PHANTOM 4***	Posibilidad de controlar el último de los modelos de la familia *DJI Phantom*	10
RNF-002	**GENERACIÓN DE DOCUMENTACIÓN EN FORMATO PDF**	Encapsulado de los datos del formulario en un archivo de tipo Pdf	8
RNF-003	**COMPATIBILIDAD CON DISPOSITIVOS *IPAD* DE HASTA 4ª GENERACIÓN**	La aplicación podrá ser ejecutada en dispositivos *iPad* de hasta 4ª generación	10
RNF-004	**COMPATIBILIDAD CON DISPOSITIVOS *IPAD MINI* Y *IPAD PRO***	La aplicación podrá ser ejecutada en dispositivos *iPad Mini* de hasta 4ª generación, así como en el *iPad Pro*	10
RNF-005	**SOPORTE PARA RESOLUCIONES TIPO RETINA Y NO RETINA**	Capacidad de que la interface gráfica se visualice en dispositivos que admitan altas resoluciones	10
RNF-006	**SOPORTE DE ROTACIÓN HABILITADO**	Modificación de la interface en función del giroscopio del dispositivo	8
RNF-007	**POSIBILIDAD DE VÍDEO HASTA 4 K**	Capacidad de grabar y reproducir clips de vídeo de calidad hasta 4K	6

Tabla 5: Requisitos no funcionales.

2.2.3 Diagramas de casos de uso

El SISTEMA PRINCIPAL de la App ***COMPandGO*** se compone de cinco actores (dos de ellos humanos y tres sistemas), así como de trece casos de uso que recogen los principales requisitos de la aplicación.

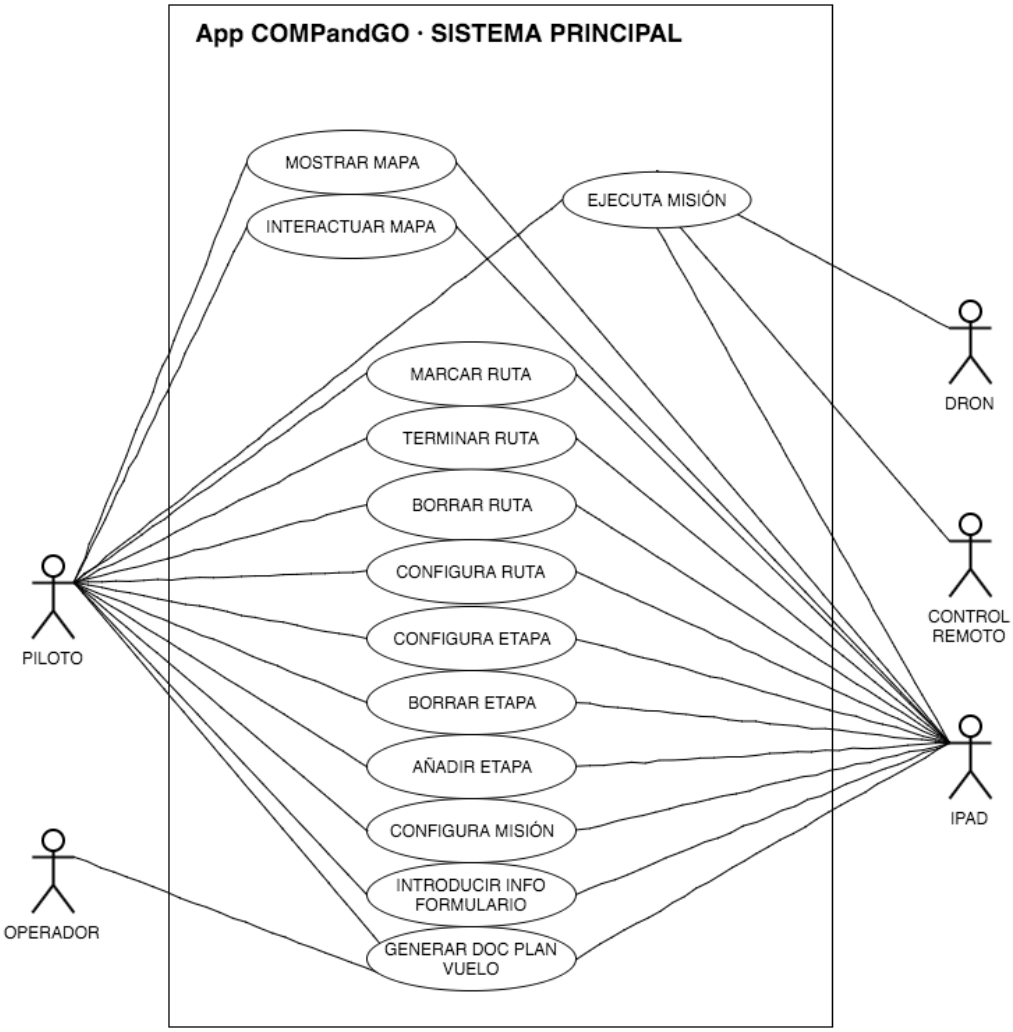


Imagen 3: Diagrama de casos de uso: Sistema principal.

Actores	Casos de Uso
Piloto: Piloto de la aeronave. **Operador:** Gestor de la empresa de drones dada de alta como tal en AESA **Control Remoto:** Dispositivo que controla por RF la aeronave ***iPad*:** Dispositivo sobre el el que se ejecuta la App	**Mostrar Mapa** (Piloto, *iPad*), **Interactuar Mapa** (Piloto, *iPad*), **Ejecutar Misión** (Piloto, *iPad*, Control Remoto, Dron), **Marcar Ruta** (Piloto, *iPad*), **Terminar Ruta** (Piloto, *iPad*), **Borrar Ruta** (Piloto, *iPad*), **Configura Ruta** (Piloto, *iPad*), **Configura Etapa** (Piloto, *iPad*), **Borrar Etapa** (Piloto, *iPad*), **Añadir Etapa** (Piloto, *iPad*), **Configurar Misión** (Piloto, *iPad*), **Introducir info Formulario** (Piloto, *iPad*), **Generar doc Plan Vuelo** (Piloto, *iPad*, Operador)

Tabla 6: Casos de uso: Sistema principal.

El SISTEMA SECUNDARIO consta de cuatro actores (dos de ellos humanos y dos sistemas), así como de diecisiete casos de uso que recogen los requisitos de prioridad secundaria de la aplicación.

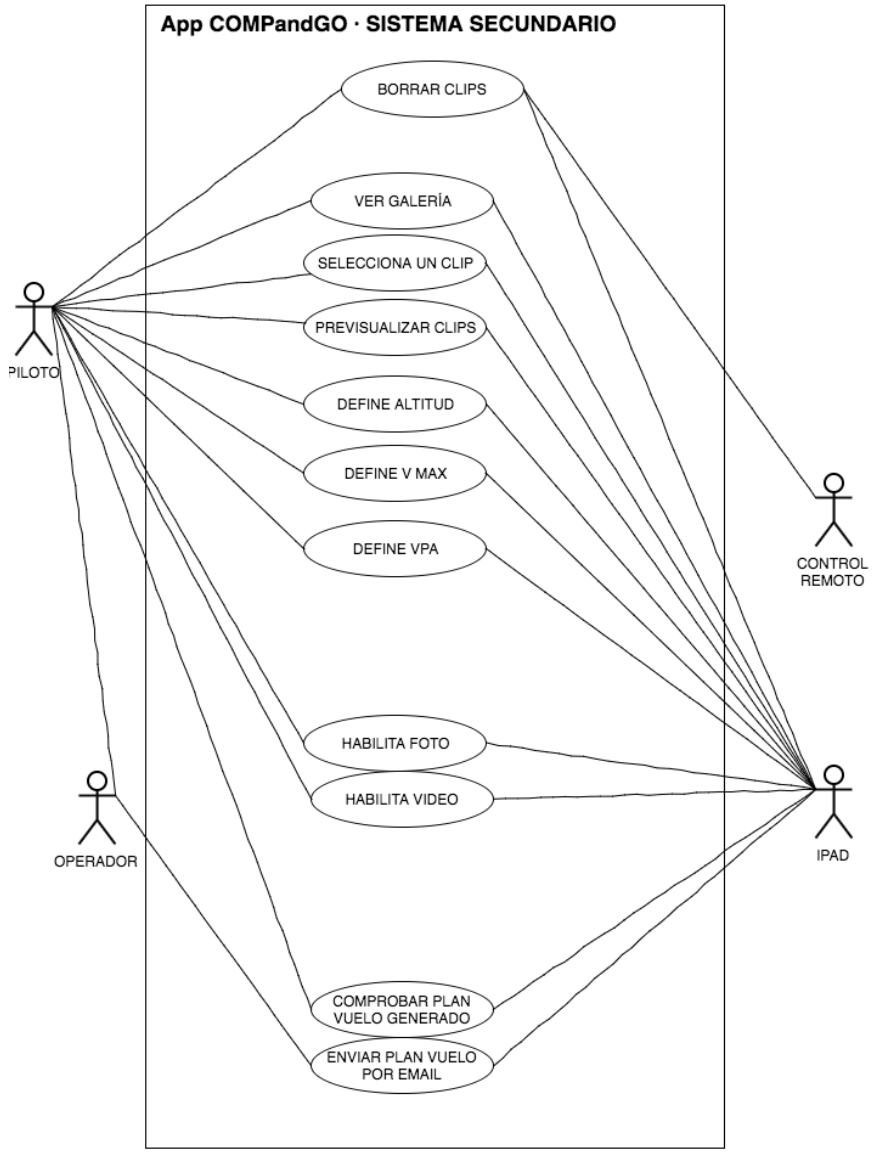


Imagen 4 Diagrama de casos de uso: Sistema secundario.

Actores	Casos de Uso
Piloto: Piloto de la aeronave. **Operador:** Gestor de la empresa de drones dada de alta como tal en AESA **Control Remoto:** Dispositivo que controla por RF la aeronave ***iPad*:** Dispositivo sobre el el que se ejecuta la App	**Borrar clips** (Piloto, *iPad, Control Remoto*), **Ver galería** (Piloto, *iPad*), **Selecciona un clip** (Piloto, *iPad*), **Previsualizar clips** (Piloto, *iPad*), **Define altitud** (Piloto, *iPad*), **Define V Max** (Piloto, *iPad*), **Define VPA** (Piloto, *iPad*), **Habilita foto** (Piloto, *iPad*), **Habilita vídeo** (Piloto, *iPad*), **Comprobar Plan Vuelo generado** (Piloto, *iPad*), **Enviar Plan Vuelo por email** (Piloto, *iPad*)

Tabla 7: Casos de uso: Sistema secundario.

El SISTEMA TERCIARIO consta de tres actores (dos de ellos humanos y un sistema), así como de doce casos de uso que recogen los requisitos de menor prioridad de la aplicación.

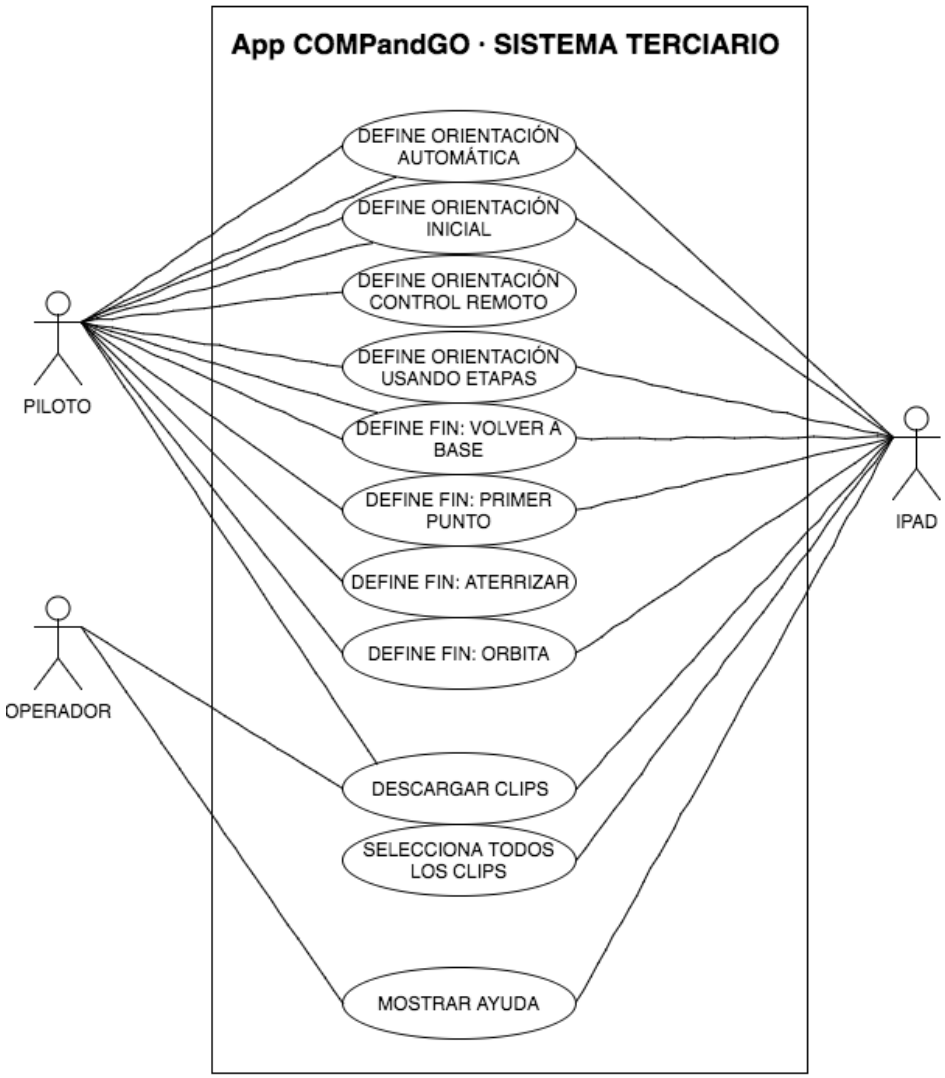


Imagen 5: Diagrama de casos de uso: Sistema terciario.

Actores	Casos de Uso
Piloto: Piloto de la aeronave. **Operador:** Gestor de la empresa de drones dada de alta como tal en AESA ***iPad*:** Dispositivo sobre el el que se ejecuta la App	**Define orientación automática** (Piloto, *iPad*), **Define orientación inicial** (Piloto, *iPad*), **Define orientación control remoto** (Piloto, *iPad*), **Define orientación usando etapas** (Piloto, *iPad*), **Define fin: volver a base** (Piloto, *iPad*), **Define fin: primer punto** (Piloto, *iPad*), **Define fin: aterrizar** (Piloto, *iPad*), **Define fin: orbita** (Piloto, *iPad*), **Descargar clips** (Piloto, Operador, *iPad*), **Selecciona todos los clips** (*iPad*), **Mostrar ayuda** (Operador, *iPad*)

Tabla 8: Casos de uso: Sistema terciario.

2.2.4 *DJI Phantom 4*· La elección de un Dron para nuestro desarrollo

El *Phantom 4* es el dron comercial con mayor desarrollo tecnológico y mejores prestaciones presentado hasta ahora. Presentado en Nueva York el pasado 1 de marzo de 2016 [19], representa el producto más esperado dentro del mercado de drones en los últimos tiempos.

Imagen 6: Dron *DJI Phantom 4*: Vista frontal.

2.2.4.1 Conceptos del Hardware *DJI Phantom 4*

Algunas de sus prestaciones desde el punto de vista de la navegación aérea son [20]:

- Sus dimensiones y su peso (1380 g) en relación a sus cuatro propulsores de última generación lo convierten en una aeronave extremadamente veloz, alcanzando los 20 m/s , con unas velocidades de ascenso de 6 m/s y de descenso de 4 m/s.

Imagen 7: Dron *DJI Phantom 4*: Vista en perspectiva isométrica.

- Su nuevo sistema de baterías inteligentes de 5350 mAh de tipo LiPo 4S y 462 g de peso permiten vuelos de hasta 28 minutos de duración, con telemetría en tiempo real del estado de la misma.

Imagen 8: Dron *DJI Phantom 4*: Batería.

- Las cámara representa un elemento fundamental de este dron con respeto a sus competidores en el mercado ya que permite grabaciones de vídeo en calidades de Ultra HD (4K) de 24p y 25p, a bitrates de hasta 60 Mbps y capacidad de grabación en SuperSlowMotion de hasta 120 fps. Por otra parte graba nativamente en Codecs MP4/MOV (MPEG-4 AVC / H.264) y realiza fotografías de 12 Megapixel en formato nativo Adobe DNG RAW. El revolucionario desarrollo de su lente permite evitar hasta un 95% la aberración producida por su gran angular.

Imagen 9: Dron *DJI Phantom 4*: Cámara.

- El *Phantom 4* posee un sistema de sensores acústicos junto con dos cámaras complementarias a la principal (situadas en los laterales) y que detectan objetos a distancias muy próximas, dotando al sistema de un sofisticadísimo control anti-choques.

Imagen 10: Dron *DJI Phantom 4*: Sensores.

- Desde el nuevo control remoto y conectándolo a un dispositivo móvil (mediante un cable USB-C), permite una transmisión de señal a la aeronave hasta una distancia de 5 km, con una recepción en tiempo real de vídeo a 720p de calidad, junto con una detalladísima telemetría.

  Incorpora funciones tan sofisticadas como el seguimiento automático de objetivos, el regreso a base de manera autónoma en caso de pérdida de recepción de datos de satélites o un sistema de piloto automático que se activa ante cualquier incidencia que pudiera producirse a lo largo de la misión de vuelo.

Imagen 11: Dron *DJI Phantom 4*: Mando a distancia.

2.2.4.2 Desarrollo de Software para *DJI Phantom 4*

2. Contexto y Estudio preliminar | 2.2 Especificación de requisitos | 2.2.4 *DJI Phantom 4*· La elección de un Dron para nuestro desarrollo

- *DJI* pone al alcance de los desarrolladores un completo SDK que incluye Frameworks, clases y métodos para el control de buena parte del hardware de sus aeronaves [21].

  Tras un proceso de alta en su área de desarrollo [22], un usuario puede acceder a un área de descarga GitHub [23] y a una completa documentación sobre su API, así como intercomunicarse, mediante su foro oficial, con una amplísima comunidad de desarrolladores. Por otra parte, existen a disposición de los desarrolladores diferentes ejemplos de integración de sus tecnologías con los entornos de desarrollo iOS y Android.

Imagen 12: DJI proporciona un SDK denominado Mobile SDK

- *DJI* facilita también una herramienta de depuración llamada *DJI Bridge App*, que se usa para interconectar vía TCP el Xcode y su simulador con el control remoto del *Phantom* (y en consecuencia con el dron) [24].

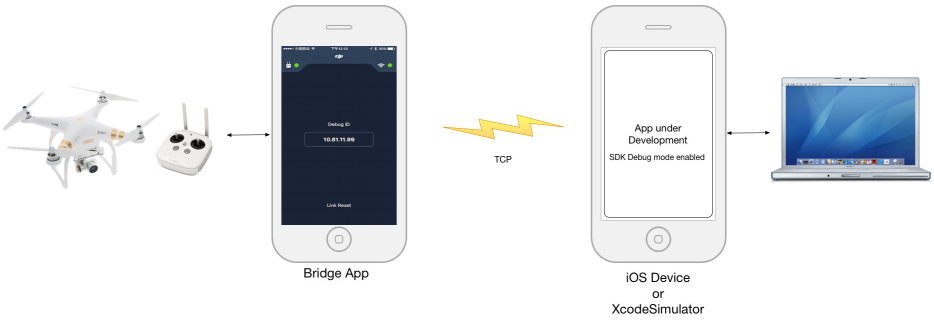


Imagen 13: Esquema de conexiones Xcode => Simulador => Mando a distancia => Dron.

- Mediante el simulador *DJI Assistant 2*, podemos testear directamente el comportamiento de nuestra aplicación sin necesidad de realizar vuelos reales con la aeronave. A través de este software se obtienen logs del comportamiento del dron en cada instante de tiempo y en cada punto geográfico. Este utilidad es especialmente relevante, como se expondrá más adelante en esta memoria, en los procesos de validación experimental de la App.

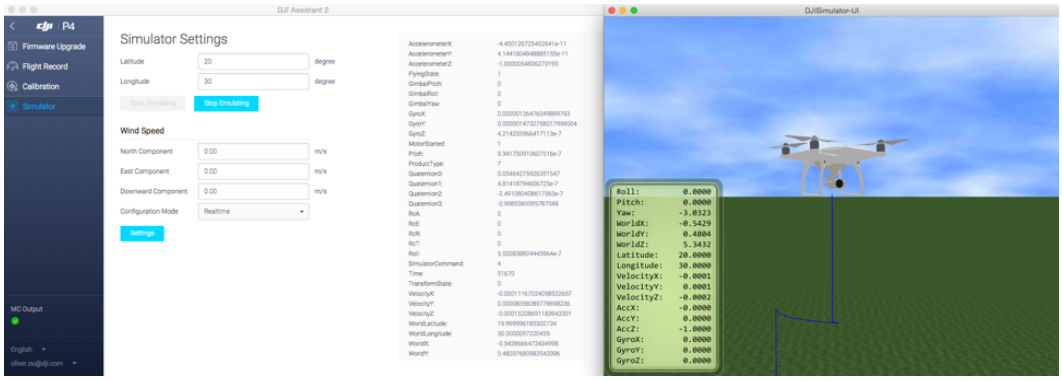


Imagen 14 La interface del simulador *DJI Assistant 2*.

2.3 Entorno de Desarrollo

La App objeto de desarrollo de este TFG es exclusivamente compatible con drones de la firma *DJI* y ha sido concebida para ejecutarse en dispositivos móviles *iPad* de la firma *Apple*.

Todos los dispositivos portátiles de esa firma utilizan el mismo OS (en distintas versiones, por supuesto). Las tabletas (*iPad*), teléfonos inteligentes (*iPhone*) y reproductores digitales de audio portátiles (*iPod*) funcionan bajo iOS.

De cara a desarrollar Apps para estos dispositivos, *Apple* facilita de forma gratuita a sus usuarios el IDE (entorno de desarrollo integrado) Xcode. Con esa herramienta a nuestro alcance, una de las opciones más sólidas a escoger como lenguaje de programación es Objective-C.

En los siguientes apartados detallaremos esta cadena de elecciones, que pasan de una a otra como fichas de dominó.

DJI -> *iPad* -> iOS -> Xcode -> Objective-C

2.3.1 iPad · La elección del dispositivo

Como hemos visto, para sacar el máximo rendimiento de las posibilidades del dron, se hace imprescindible conectarle a su control remoto un dispositivo móvil. Este soporte permite recibir en tiempo real la imagen captada por la cámara del dron. Así mismo, será el nexo de unión entre el dron y el entorno de desarrollo de nuestra App. El dispositivo móvil que se ha escogido para el desarrollo de este proyecto es el *iPad*.

Imagen 15: *iPad Air 2.*

El *iPad* es una tableta (dispositivo electrónico portátil con pantalla táctil, según la R.A.E.) comercializada por la firma *Apple*. La razón de haberse decantado por un dispositivo de esta firma ha sido el acuerdo comercial entre la casa fabricante *DJI* y la mencionada *Apple* para la comercialización del modelo de dron *Phantom* 4 y sus accesorios. Este acuerdo ha traído como consecuencia la especial calidad en la compatibilidad entre los dispositivos de ambas firmas.

En ese mismo orden de cosas, los principales dispositivos de *Apple* compatibles con el *Phantom* 4 y que podían ser susceptibles de ser candidatos para ser elegidos eran el *iPhone* y el *iPad*. En este punto la balanza se decantó a favor del *iPad* por su mayor potencia de procesador y por las dimensiones de su pantalla.

La familia de modelos de *iPad* es ya lo suficientemente extensa (ya se encuentran en su 4ª generación) como para que la elección del modelo concreto más adecuado sea un paso importante. Las distintas Apps que se comercializan necesitan decodificar la señal de vídeo en tiempo real y, por tanto, se requiere mucha potencia de CPU. Es por ello que la baraja de candidatos queda reducida a aquellos *iPad*s con procesadores más potentes (A7, A8 ó A9, en todas sus variedades).

Finalmente, el elegido ha sido un *iPad Air 2*, que se encuentra en la gama media de tabletas *Apple* compatibles. Sus principales características son [25]:

- Pantalla: Se trata de una pantalla Retina, compuesta por tres capas fusionadas (cubierta de vidrio, sensor táctil y LCD), dotada de un acabado antirreflectante (especialmente útil cuando se trabaja con el dron en un día soleado y sin parasol) y retroiluminada mediante LED. Si la fuerza se mide en números, los de esta pantalla son:
    - Dimensiones: 240 x 169,5 x 6,1 (mm)
    - Tamaño: 9,7 pulgadas (diagonal)
    - Peso: 444 g
    - Resolución: 2048 x 1536
    - 3,1 millones de píxeles.
    - 264 píxeles por pulgada.
- Procesador: A8X con arquitectura de 64 bits.
- Conectividad:

  - Wi-Fi (802.11ª/b/g/n/ac); con doble banda (2,4 GHz y 5 GHz).
  - Bluetooth 4,2
  - 4G
- Batería:
  - Recargable integrada de polímeros de litio de 27,3 vatios/hora
  - Hasta 9 horas de navegación por Internet a través de redes móviles
- Sistema Operativo: iOS 9.

2.3.2 iOS · La elección del Sistema Operativo

Siguiendo con la concatenación de decisiones, la elección del *iPad* condiciona la elección del sistema operativo. iOS es el sistema operativo de los dispositivos móviles de *Apple*, que es la responsable de la fabricación tanto de iOS como del hardware, por lo que la integración entre ambos es tal que permite que se aprovechen al máximo las prestaciones de las distintas familias de *iPhone* y *iPad*.

iOS es una derivación del sistema operativo OSX, incluido por *Apple* en sus ordenadores de sobremesa y portátiles. OSX a su vez no deja de ser una implantación del sistema operativo UNIX, aunque se puede precisar este particular. Digamos que OSX se basa en Darwin, el cual está derivado en parte de FreeBSD 5, que finalmente integra servicios de UNIX. Todo un árbol genealógico de la historia viva de la informática.

Como si de una receta de cocina se tratase, y sin hacer una relación exhaustiva de todos los ingredientes, diremos que OSX aprovechó de UNIX la interface de líneas de comandos (llamada Terminal), la gestión avanzada de la memoria y la multitarea de FreeBSD 5, a los que *Apple* añadió distintos componentes de los que es propietaria, como una interfaz gráfica de usuario llamada Aqua y el Finder. Si bien es cierto que está escrito en C y C++, principalmente está desarrollado en Objective-C.

Con el lanzamiento al mercado del primer modelo de *iPhone* (en 2007), este ya venía dotado de un sistema operativo derivado de OSX y que se denominó *iPhone* OS. En junio de 2010 llegaría el cambio de denominación de este sistema operativo a iOS 4, coincidiendo con el lanzamiento al mercado del *iPhone* 3G. En el momento de redactarse este TFG, la versión más actual es iOS 9.

Las actualizaciones de iOS son gratuitas y se realizan a través o bien de la tienda online de aplicaciones de *Apple* (App *Store*) o bien del software iTunes. A través de esos mismos canales se tiene acceso a un amplísimo catálogo de Apps. A fecha del pasado 21 de marzo, **Costello** (2016), cifraba el número aproximado de aplicaciones en iOS en 1.500.000, de las cuales 1.000.000 eran compatibles con el *iPad* [14].

Las Apps más populares para la gestión de drones se encuentran en el App *Store*. Todas ellas buscan aprovechar al máximo las posibilidades que ofrece iOS, como el hecho de que sea multitarea, que permita conexiones muy sencilla con otros dispositivos de *Apple* (mediante servicios incorporados como *Handoff, AirDrop, AirPlay, iCloud*, etc.) o que permita un control gestual multitáctil sobre la pantalla.

2.3.3 Xcode · La elección del entorno de Desarrollo

De cara a desarrollar una App, independientemente del sistema operativo de trabajo, es necesario emplear un entorno de desarrollo integrado (IDE). No deja de ser un software que se tiene que instalar en el ordenador, aunque hoy en día gracias al cloud computing los hay también disponibles en línea (se ejecutan por medio de navegadores de internet).

Imagen 16: Icono de la aplicación de Xcode para el sistema operativo OSX.

Un programador para OSX o iOS puede optar por emplear herramientas de código abierto (como *Netbeans* y *Eclipse*) o bien IDE. nativos, esto es, proporcionados por el desarrollador del sistema operativo. *Apple* suministra al efecto Xcode junto a OSX (y por tanto de modo gratuito).

Xcode es la más completa de las herramientas de desarrollo de aplicaciones para dispositivos iOS. Incluye todo lo necesario para programar aplicaciones móviles. Para obtenerlo basta con tener una cuenta de usuario de *Apple* y descargarlo a través del *Apple Store*, en la sección *Categorías* y en la subsección *Para desarrolladores*. En junio de 2016 la versión más actual es Xcode 7.3.1.

Mediante Xcode se puede compilar código C, C++, Objective-C, Objective-C ++, Swift, *Apple*Script o Java. Los compiladores que emplea para ello forman parte de la colección llamada GCC, perteneciente al proyecto GNU (que busca crear el primer OS completamente libre). Para desarrollar y conectar entre sí los diferentes componentes de un proyecto, Xcode emplea Frameworks, que facilitan esas relaciones y dota al proyecto de un estructura muy determinada, la cual determina el método de trabajo a seguir. Por tanto, es necesario introducir ese tipo de Frameworks (desarrollados por *Apple* o por terceros) en los distintos proyectos.

La interface de Xcode, en idioma inglés en su totalidad, se divide en las siguientes zonas de trabajo:

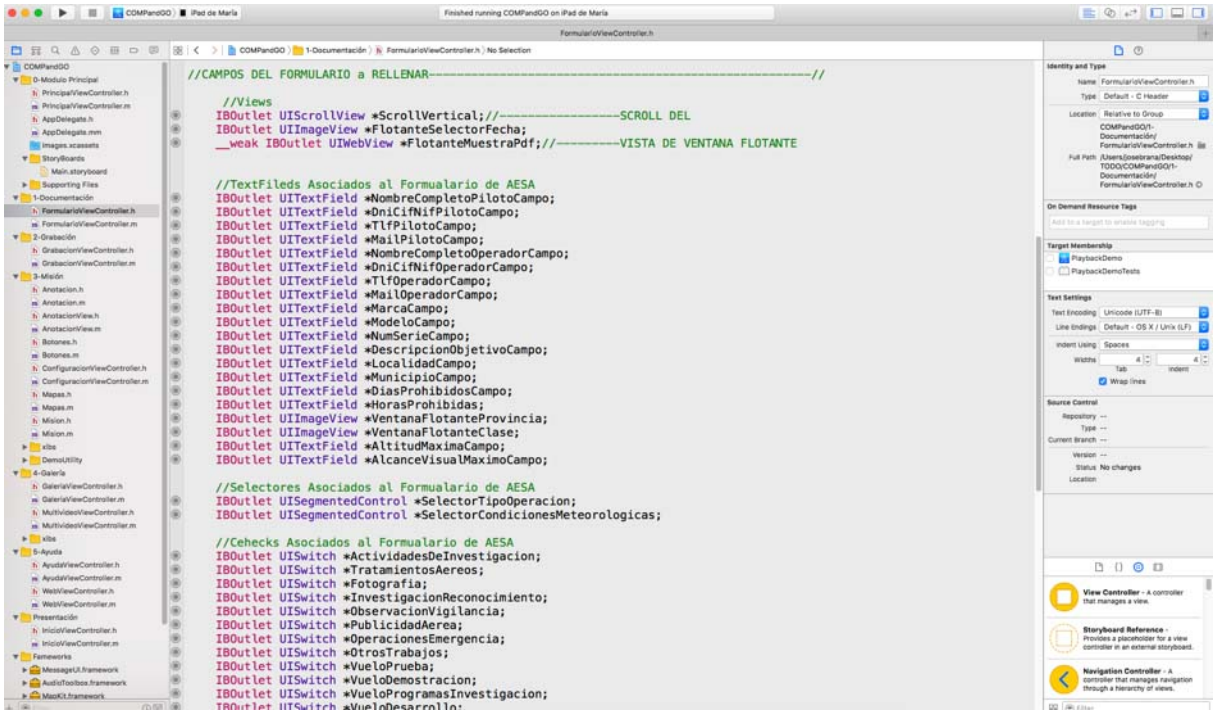


Imagen 17: Interface de Xcode 7.3.1.

- La barra de menús.
- La barra de herramientas, que incluye de izquierda a derecha lo siguiente:
  - Botón de Play: para lanzar la compilación del proyecto.
  - Selector del dispositivo donde vamos a compilarlo. Pueden ser tanto dispositivos reales (conectados físicamente al ordenador) como dispositivos virtuales, a los que podemos acceder a través de un Simulador. Existe un espacio más adelante dentro de esta memoria reservado para este Simulador.

  - Mostar el editor estándar: muestra el código de un archivo.
  - Mostrar el editor de la contraparte: muestra el código del archivo directamente relacionado con el anterior.
  - Mostrar la versión del editor.
  - Ocultar / mostrar el editor principal de archivos (que se corresponde con el área vertical izquierda de la pantalla).
  - Ocultar / mostrar la consola de errores (área inferior central).
  - Ocultar / mostrar las utilidades (área vertical derecha).
- Explorador de archivos del proyecto: área vertical izquierda de la pantalla.
- Zona de código / Storyboard: ocupa el centro de la pantalla. El diseño de la interface de una aplicación se hace de modo gráfico, gracias a la herramienta Storyboard, lo que facilita enormemente el proceso, además de hacerlo muy intuitivo.
- Zona de utilidades: área vertical derecha. En su parte superior hay una zona de edición (que incluye una extensísima ayuda en inglés), mientras que en la parte inferior está la biblioteca de objetos.

2.3.4 Objective-C · La elección del lenguaje de programación

Objective-C es el principal lenguaje de programación usado cuando se programa software para OSX y iOS. Está basado en C y se puede definir como un superset de C, es decir, que coge C como base y se le añaden una serie de funcionalidades extra que permiten que C adquiera la capacidad de estar orientado a objetos y un tiempo de ejecución dinámico [2]. Su sintaxis permite al usuario crear clases, objetos, variables de instancia, métodos, etc. Su herencia de C provoca que también puedan utilizarse construcciones, expresiones, punteros, etc. Con Objective-C puede compilarse cualquier programa escrito completamente en C. Dentro de una clase de Objective-C se puede utilizar sin problema código en C.

En Objective-C los detalles de implementación del código están separados de las declaraciones. A la hora de desarrollar una aplicación, nos encontraremos con la siguiente organización de archivos:

- Archivos de cabecera: Con extensión .h, contienen las declaraciones, que así permanecen separadas de los detalles de implementación del código. En detalle, incluyen las clases, las funciones, los tipos de datos y la declaración de constantes.

- Archivos fuente: Con extensiones .m y .mm. Incluyen los detalles de implementación del código fuente.
    a. Los .m pueden contener código en C y en Objective-C.
    b. Los .mm además de contener código en C y Objective-C pueden incluir código en C++. Solamente hay que utilizarlos en este caso.

2.3.4.1 Mensajes entre objetos

La mayoría de los procesos que ocurren dentro de una App creada con Objective-C son resultado de la comunicación entre objetos mediante mensajes. Los objetos se envían mensajes unos a otros continuamente. Cuando un objeto le envía un mensaje a otro se dice que lo invoca.

La estructura correcta de un mensaje para que un objeto invocado la comprenda es la siguiente [2]:

[Receptor Selector: Argumento]

[aQuiénLeEstamosEnviandoEl Mensaje loQueQueremosQueHaga: conQuéLoHaga]

[Puntero nombreDelMétodo: argumentosAUtilizar]

Lo primero que llama la atención a los programadores que toman su contacto con este lenguaje son los corchetes. No hay que darles mayor importancia. Es la convención de escritura escogida por Objective-C, sin más.

- La primera parte es el puntero, lo que apunta hacia el objeto que va a recibir el mensaje.
- La segunda parte es el nombre del Método. Es aquello que queremos que haga el objeto invocado, que podrá hacerlo o no dependiendo de su herencia, que es la que nos dará las opciones de lo que ese objeto invocado es capaz de hacer.
- La tercera y última parte son los argumentos a utilizar por el objeto invocado. En definitiva los requisitos, los cuales deben ser de un tipo que pueda aceptar el objeto invocado.

El receptor también puede ser una clase, ya que las clases poseen métodos. El receptor no tiene que garantizar una respuesta al mensaje. Si no responde, simplemente puede retornar un puntero nulo (nil).

Esta estructura de los mensajes logra que el código en Objective-C sea más robusto que el de otros lenguajes de programación que se apoyan en C.

2.3.4.2 Delegados

En el lenguaje Objective-C se puede dar el caso de que un objeto invocado no pueda responder a un mensaje porque no tenga el método adecuado implementado. ¿Qué ocurre entonces?

Un delegado (delegate) es un objeto que hace de representante de otro objeto. Podemos tener pues un objeto principal y su delegado. En el caso planteado de que un mensaje quede sin respuesta por el Receptor, se le enviará a su delegado para que lo responda [2].

2.3.4.3 Propiedades delegadas

Las propiedades declaradas son una notación empleada por Objective-C para reducir la cantidad de código redundante. Reemplazan a las declaraciones y, opcionalmente, a los métodos de acceso.

Para usar una propiedad declarada primero hay que declararla en el archivo de cabecera y después hay que implementarla en el archivo fuente.

2.3.5 Registro en programas de Desarrollo

Normalmente la capacidad para instalar Apps de creación propia en dispositivos queda acotada a aquellos usuarios que lo solicitan formalmente al desarrollador del hardware o del OS. Esa solicitud trae implícita la aceptación de un acuerdo de licencia y un coste anual. Esto le sirve al suministrador del servicio como filtro y como sistema de control de las App que se crean el mundo para funcionar en sus dispositivos.

A ese proceso de alta se le denomina registro en programas de desarrollo.

2.3.5.1 Registro en Programas de Desarrollo *Apple*

En el caso que nos ocupa en este proyecto, tenemos que tener presente que para instalar un App de programación propia en un *iPad*, *Apple* exige darse de alta como desarrollador en su *Apple Developer Program*.

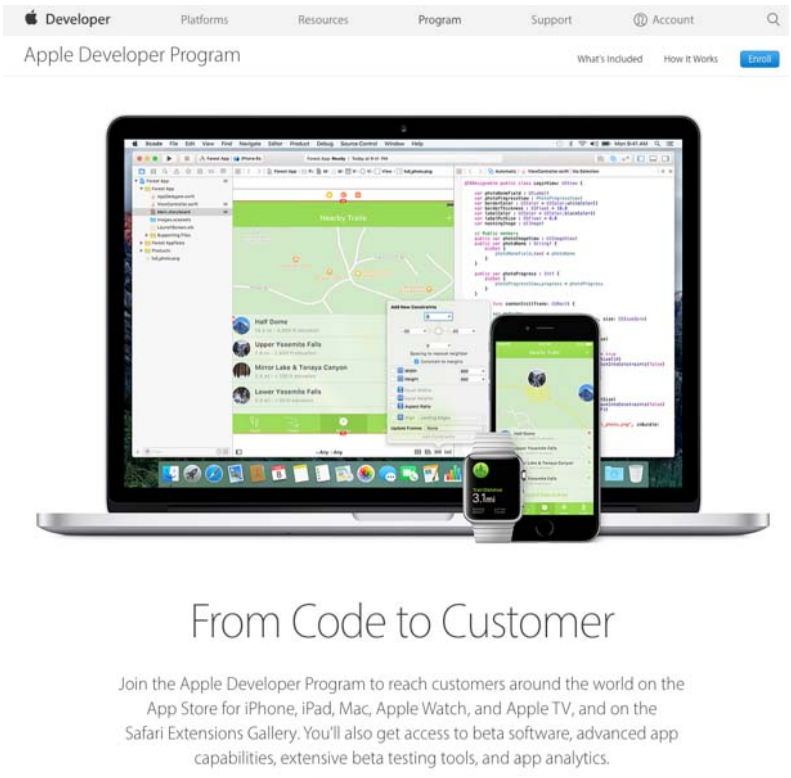


Imagen 18: Página web del *Apple Developer Program*.

Para ello hay que acudir a la web de *Apple Developer* [26], ingresar con los datos de una cuenta de usuario de *Apple* y aceptar un acuerdo legal. Tras estos pasos ya tendremos acceso a nuestra área de desarrolladores. Desde ahí, ya tendremos a nuestro alcance varias opciones, como pueden ser un enlace a la descarga de Xcode, una sección de documentación, foros oficiales de ayuda, etc.

Sin embargo en ese punto aún no se tiene el estatus de desarrollador *Apple*. De hecho el proceso todavía no se ha puesto en marcha como tal. Habrá que hacer clic en la parte inferior de la pantalla sobre el icono de *Join the Apple Developer Program*. A continuación

vendrán sucesivos clics sobre Enroll, Start your Enrollment, y se comenzará a facilitar una serie de datos personales (incluida nuestra identificación como individuo, autónomo, empresa o empresa estatal) y de pago para ingresar el coste de la licencia anual ($99 para una licencia individual y $299 para una de empresa). *Apple* enviará un correo de confirmación. Tras responderlo correctamente ya se habrá hecho efectiva el alta.

Si se accede de nuevo a la cuenta de *Apple Developer* se observa que se tiene acceso a nuevas secciones. Una de ellas es el vehículo que permite al desarrollador poder instalar Apps en dispositivos portátiles. Se trata de la sección *Certificates, IDs & Profiles*, donde se detallan los certificados iOS que se poseen.

Las acciones a realizar en este apartado y que son imprescindibles para poder instalar una App en el *iPad* son:

1. Activación del dispositivo *Apple* donde se instalará la App. Se solicitará el código alfanumérico del *Hardware UUID*. Ese dato se puede obtener gracias a iTunes. Al conectar el dispositivo a un terminal donde esté instalado iTunes, podremos encontrar la información buscada en el *Resumen* del dispositivo. Se trata de un dato único para cada dispositivo, con lo que *Apple* se garantiza prácticamente que cada usuario de este servicio es el legítimo propietario del dispositivo.
2. Obtención de un certificado de desarrollo. Desde la sección Certificates se pueden gestionar los distintos certificados que poseamos. Para crear cualquiera de estos certificados en general y el *iOS Development Certificate* en particular, se debe utilizar la aplicación de OSX *Acceso a llaveros*. Desde el menú *Acceso a llaveros => Asistente para certificados => Solicitar un certificado de una autoridad de certificación*. Tras facilitar los datos de acceso a la web de *Apple Developer*, podremos descargar el certificado al disco local. Luego bastará con subir este certificado a la mencionada sección *Certificates*.
3. Creación de un perfil del tipo *Provisioning Profile*. Debe crearse el correspondiente a *iOS App Development* (como curiosidad, esta es la sección donde se crean los *Distribution Provisioning Profile*, necesarios para la distribución de las App en el *App Store*). Este perfil sirve para poder utilizar nuestros certificados de desarrollo para firmar nuestras aplicaciones. A estas alturas ya se han terminado las acciones dentro de la web de *Apple* Developer.
4. Ya sólo quedan un par de pasos, pero esta vez dentro del propio software Xcode. El primero se realiza desde el menú Xcode => Preferences… => Accounts. Ahí se

tienen que añadir los mismos datos de la cuenta de usuario de *Apple* que se han empleado a lo largo de todo este proceso.

5. Por último, habrá que acudir al menú Xcode => Windows => Devices => icono de Settings => *Show Provisioning Profiles*… Ahí se deberá añadir el *Provisioning Profile* obtenido anteriormente.

Aquí finaliza el proceso. Si se ha llevado a buen término, el desarrollador ya podrá instalar su App en el dispositivo *Apple* de su propiedad.

2.3.5.2 Registro en Programas de Desarrollo *DJI*

De manera análoga a *Apple*, la empresa *DJI* exige un registro oficial a los programadores que van a desarrollar Apps para gestionar las distintas características de sus drones. En el caso de *DJI* este registro es gratuito y ni siquiera va asociado a un ID de alguna aeronave de *DJI*.

El registro se hace desde la web de *DJI Developer* [22]. Solamente exige de un nombre de usuario, una contraseña y una dirección de correo electrónico, a la que se enviará un correo de confirmación. Tras utilizar los datos de acceso escogidos, se llega a la página de perfil del usuario.

Desde allí se pueden solicitar unos datos imprescindibles para el correcto funcionamiento de la App a desarrollar. El proceso es el siguiente.

1. Pulsar sobre el botón *Create App*.
2. Aparecen en pantalla dos columnas: *Mobile SDK* y *Onboard SDK*. Nos quedamos en la primera (que viene seleccionada por defecto).
3. Hay que cumplimentar una serie de campos:
    a. APP name: escribimos el nombre de nuestra App. En nuestro caso ***COMPandGO***.
    b. Software platform: escogemos iOS.
    c. Package name: es el mismo término que en Xcode se denomina *Bundle Identifier*. Cuando se crea un proyecto nuevo en Xcode se tienen que facilitar una serie de datos. Entre ellos están:
        i. *Product Name*: el nombre de la App. En nuestro caso ***COMPandGO***.

ii. Organization Identifier: se suelen poner las siglas en mayúsculas. Se recomienda que sean tres letras, para distinguir claramente nuestro Bundle Identifier del de los Frameworks, etc. De *Apple*, los cuales siempre tienen dos letras mayúsculas. En nuestro caso se ha escogido TUT.

iii. *Bundle Identifier*: se construye con el siguiente patrón: Organization Identifier.*Product Name*. Así, el de nuestra App sería TUT.***COMPandGO***.

d. Category: Lista desplegable con opciones sobre las distintas categorías de aplicaciones prácticas que va a tener nuestra App.

e. Description: una breve descripción de la aplicación.

4. Pulsar el botón de *Create*.
5. Ahora en la página del perfil de usuario parecerá una nueva App creada. Haciendo clic sobre su nombre aparecerán los datos que hemos aportado para su creación, acompañados de otro nuevo y muy importante. Se llama *App Key* y consiste en un código alfanumérico de 24 caracteres.

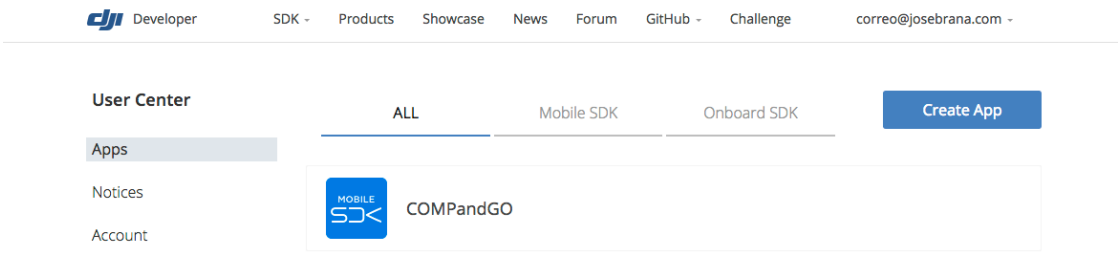


Imagen 19: *DJI Developer*: Cuenta de usuario: App ***COMPandGO*** creada y visible en la sección de Apps del usuario.

El permiso que concede *DJI* a desarrollar la App con el nombre escogido ligará biunívoca e indivisiblemente a dicha APP Key con el Bundle Identifier de la misma. Así *DJI* se asegura en todo momento la identificación de la App.

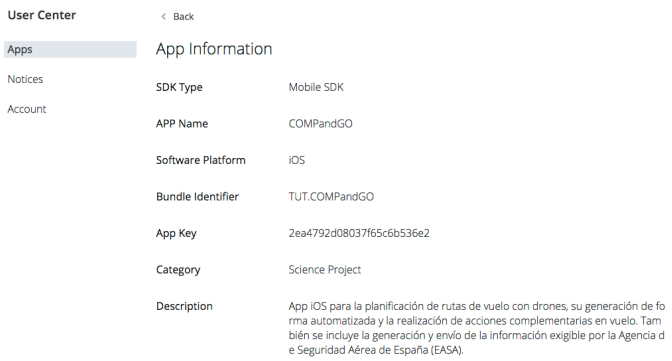


Imagen 20: *DJI Developer*: Cuenta de usuario: App ***COMPandGO***: Información de la App.

3 DISEÑO DE LA SOLUCIÓN

3.1 Diagramas

Para facilitar la compresión de la estructura a desarrollar, se realizarán diagramas de flujo de proceso, diagrama de clases y diagrama de secuencias.

3.1.1 Diagrama de flujos de proceso

Gráfico preliminar de conectividad, flujos y estados de la App [27].

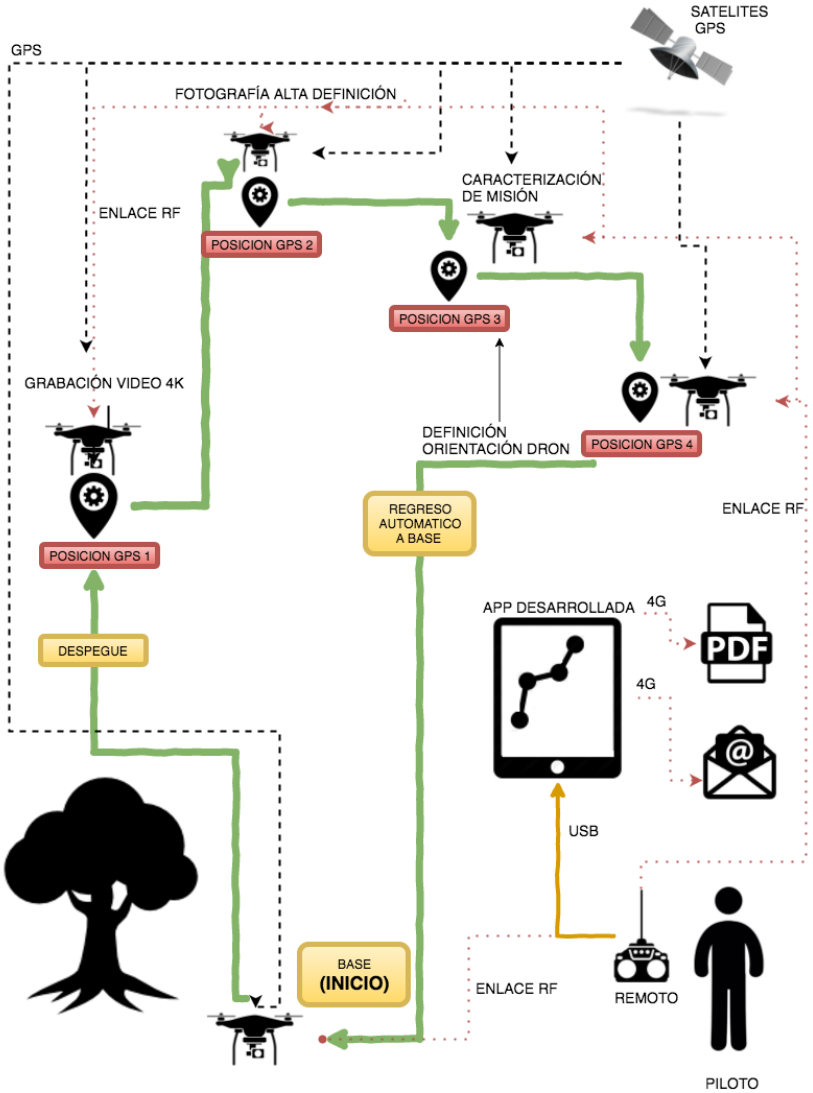


Imagen 21: Gráfico preliminar de la conectividad, flujos y estados de ***COMPandGO***.

3.2 Diseño de Marca asociada al desarrollo

Para completar el desarrollo de nuestro software, se ha diseñado un plan de marketing preliminar, explorando las posibilidades de explotación comercial de una App de características similares a la que nos ocupa.

Respecto al branding (o búsqueda de una conexión emocional entre un producto y una marca comercial), se ha tratado de aunar en un único concepto las ideas de composición (***COMP***) y de acción (***GO***). Por otra parte era deseable que la marca comercial tuviera vínculos personales con las señas de identidad del AUTOR. En ese sentido y al tratarse de un desarrollo software netamente asturiano, lo ideal era encontrar un vocablo que tuviera arraigo con la cultura asturiana. En este sentido podemos observar que si la palabra ***COMPandGO*** se leyese como si se tratase de una palabra en español, sonaría casi homófonamente como la palabra “compango”, que como es universalmente sabido es la palabra que en asturiano se usa para nombrar al acompañamiento cárnico de la fabada asturiana.

Finalmente se opta por utilizar la palabra ***COMPandGO*** como nombre comercial vinculado al desarrollo de nuestra App.

3.2.1 Naming-Check

Una vez decidida la marca y dada la importancia que tiene la disponibilidad de registros en las diferentes redes sociales para la distribución de cualquier bien comercial, así como para facilitar una vía de contacto entre los clientes (potenciales o de facto) y los desarrolladores, se ha realizado una breve prospección de la disponibilidad en esos canales de la marca comercial seleccionada (***COMPandGO***).

Para ello se ha empleado la herramienta online *namecheckr.com* [28]. Como se observa en la **Imagen 22**, están disponibles para su registro la práctica totalidad de los dominios vinculados a las redes sociales más importantes.

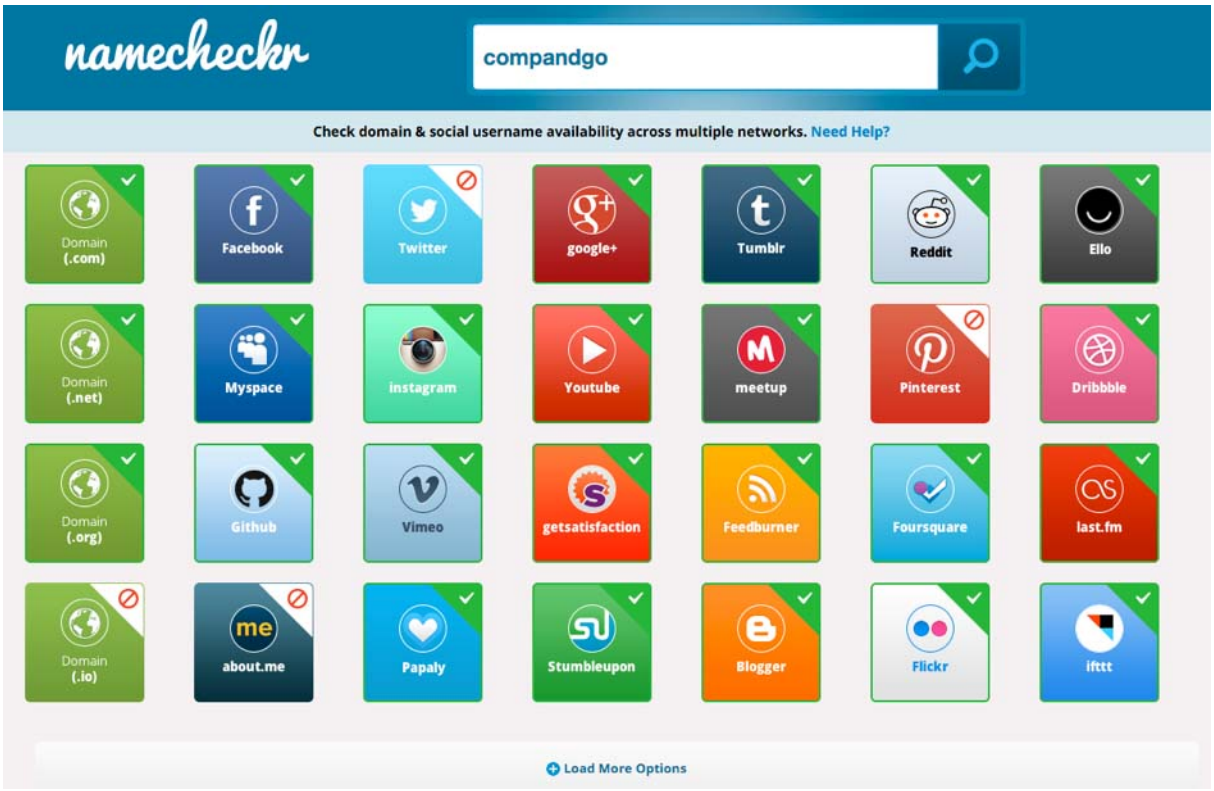


Imagen 22: *namecheckr.com*: Disponibilidad del dominio y del nombre de usuario ***COMPandGO***.

3.2.2 Diseño de un logo para la App

Uno de los elementos en los que la industria de las tecnologías de la información y la comunicación ha puesto mayor énfasis en los últimos años es el desarrollo de experiencias de usuario (UX). Un factor importante dentro de esa UX es el diseño gráfico de las distintas partes con las que interactúa el usuario de un servicio software.

En ese sentido, el icono de una aplicación es un elemento nucelar de su diseño, mediante el cual un posible cliente establece un primer vínculo emocional con la aplicación y, en muchos casos, condiciona su interés. La relación entre la cantidad de descargas de un aplicación y la calidad del diseño gráfico de su icono confirma que la utilización de un diseño gráfico acertado multiplica las posibilidades de éxito en el marketing de una App.

A tal efecto, para el desarrollo de este TFG se ha diseñado un icono de formas planas, en la línea de la tendencia actual tanto de las aplicaciones iOS como en otros portales de distribución (*Chrome Web Store*, etc.).

Se ha trabajado sobre un *Adobe Photoshop CS7* diseñando un icono que representara la esencia de las dos funcionalidades principales que recoge el sistema: la composición de rutas y el vuelo de un dron cuadricóptero.

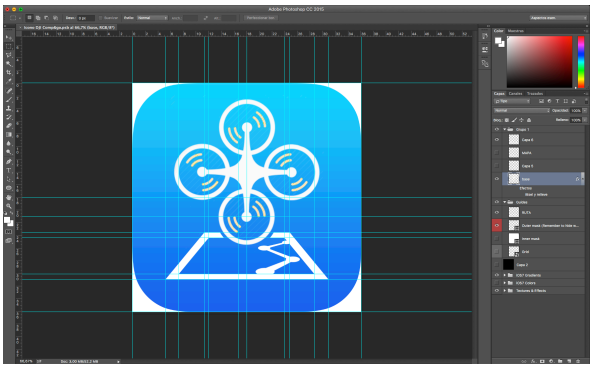


Imagen 23: Proceso del diseño del icono de ***COMPandGO***.

Imagen 24: Diseño final del icono para ***COMPandGO***.

3.2.3 Generación de iconos

Una vez diseñado el icono de la aplicación, es necesario exportarlo a distintas resoluciones y tamaños para su distribución en los distintos dispositivos móviles. En el caso particular de nuestro desarrollo, nos hemos centrado en la creación de una familia de iconos para las distintas generaciones de dispositivos móviles *iPad* y *iPhone*.

Imagen 25: El icono de ***COMPandGO*** se exporta a distintas resoluciones y tamaños.

3.3 Diseño de Mockups de la Interface

Para el diseño de los distintos prototipos de interface se ha utilizado la herramienta *Balsamiq Mockups 3* [27]. Se han generado y clasificado los mockups en tres grupos funcionales:

- Mockups del subsistema *Rutas y Misiones*.
- Mockups del subsistema *Galería*.
- Mockups del subsistema *Documentación*.

También se ha realizado un esquema general de la conexión y relaciones entre todos los mockups.

3.3.1 Diseño general de Mockups

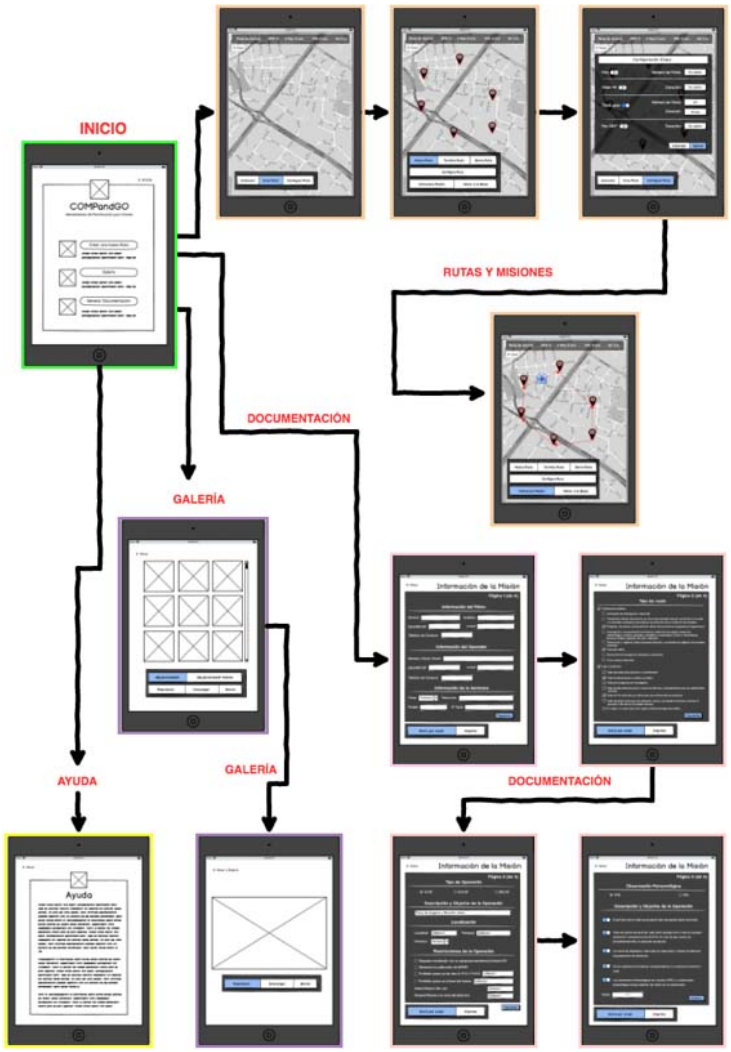


Imagen 26: Diseños generales de Mockups de la interface.

3.3.2 Diseño general de Mockups: Rutas y misiones

Inicio

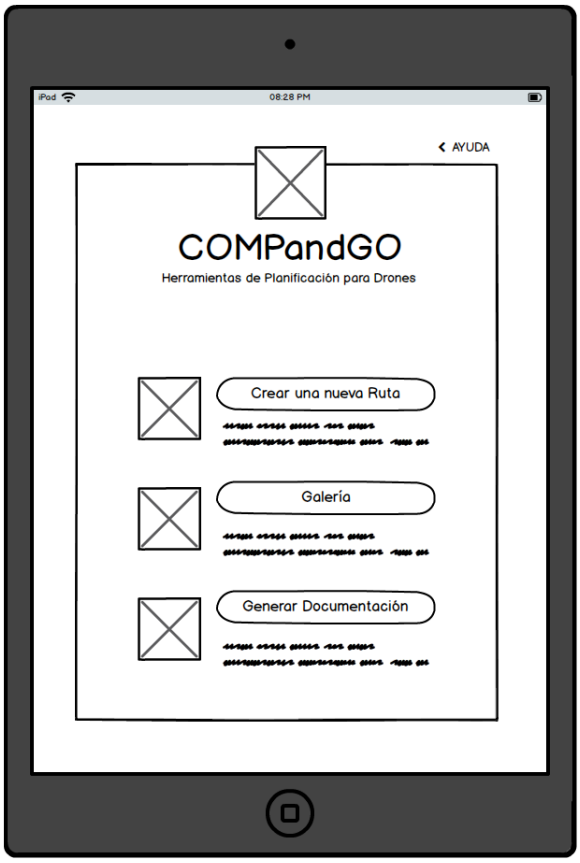


Imagen 27: Mockup del subsistema de Menú principal.

Inicio→Ayuda

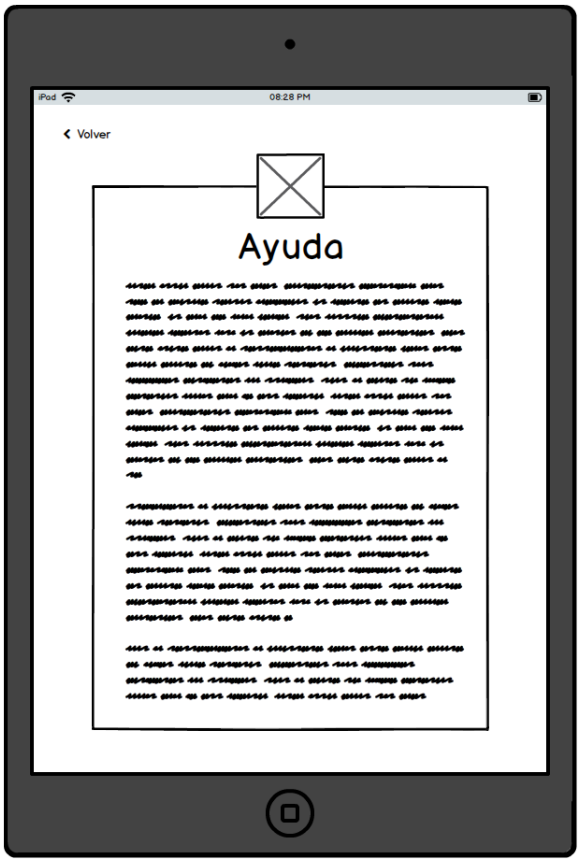


Imagen 28: Mockup del subsistema secundario de Ayuda.

PANTALLA 1: INICIO	PANTALLA 2: AYUDA
Descripción: Pantalla de inicio que presenta un menú con tres opciones, las cuales agrupan las funcionalidades del sistema. Se presentan tres botones junto a tres iconos, cada uno de ellos con una breve descripción: *Crear una nueva Ruta, Galería y Generar Documentación*. En la esquina superior derecha, un botón de *Ayuda* nos enlaza a una pantalla con instrucciones del manejo de la aplicación.	**Descripción:** Pantalla de texto con instrucciones de uso de la aplicación, así como detalle de las posibles configuraciones de rutas, etapas y misiones.
Requisitos Funcionales asociados: RF-001 al RF-006, RF-018, RF-021 al RF-023, RF-025, RF-026, RF-028, RF-031, RF-035	**Requisitos Funcionales asociados:** RF-035
Requisitos No Funcionales asociados: RNF-001, RNF-003 al RNF-006	**Requisitos No Funcionales asociados:** RNF-001, RNF-003, RNF-004, RNF-005, RNF-006

Tabla 9: Pantallas de los subsistemas de Menú principal y Ayuda: Descripción y requisitos asociados.

Inicio→Crear Ruta→Acércate

Imagen 29: Mockup del subsistema de Creación de Rutas, mientras se hace un zoom de aproximación.

Inicio→Crear Ruta→Acércate→Marcar Ruta

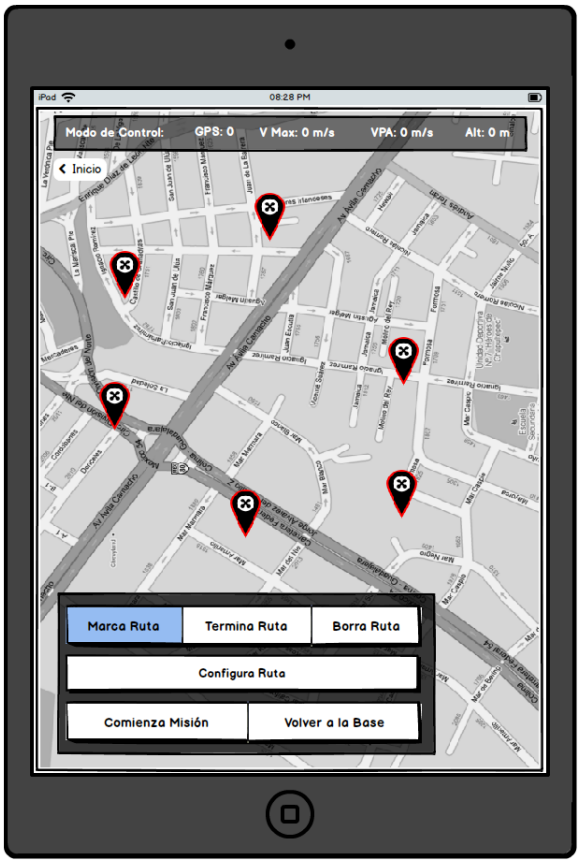


Imagen 30: Mockup del subsistema de Creación de Rutas, mientras se marcan las estaciones de una ruta o misión.

PANTALLA 3: ACÉRCATE	PANTALLA 4: CREAR RUTA
Descripción: En la pantalla vemos un plano de cercanía, tanto de la ubicación del usuario como de su entorno. En la parte inferior izquierda, dos botones nos permiten *Crear una Ruta* y *Configurarla*. En la parte superior, un banner nos facilitará información sobre el *Modo de Control*, *número de satélites GPS* a los que se está conectado, *Velocidad Máxima* configurada, *Velocidad en modo de Piloto Automático* y la *Altitud*. También se dispone de una enlace directo a la pantalla de *Inicio*.	**Descripción:** Mediante interacción táctil, y sobre la pantalla anterior, se añaden localizadores sobre el mapa que definirán cada una de las etapas. Junto con las funciones anteriores, ahora se dispone de los botones *Marcar Ruta* (para iniciar el proceso de marcado), *Terminar Ruta* (para finalizarlo) y *Borrar* Ruta (elimina el conjunto de los localizadores), así como dos botones más para *Comenzar la Misión* y *Volver a la Base*.
Requisitos Funcionales asociados: RF-001 al RF-006, RF-018, RF-023	**Requisitos Funcionales asociados:** RF-001 al RF-006, RF-018, RF-021 al RF-023
Requisitos No Funcionales asociados: RNF-001, RNF-003 al RNF-006	**Requisitos No Funcionales asociados:** RNF-001, RNF-003 al RNF-006

Tabla 10: Pantallas del subsistema de Creación de Rutas: Descripción y requisitos asociados.

Inicio→Crear Ruta→Acércate→Configura Ruta

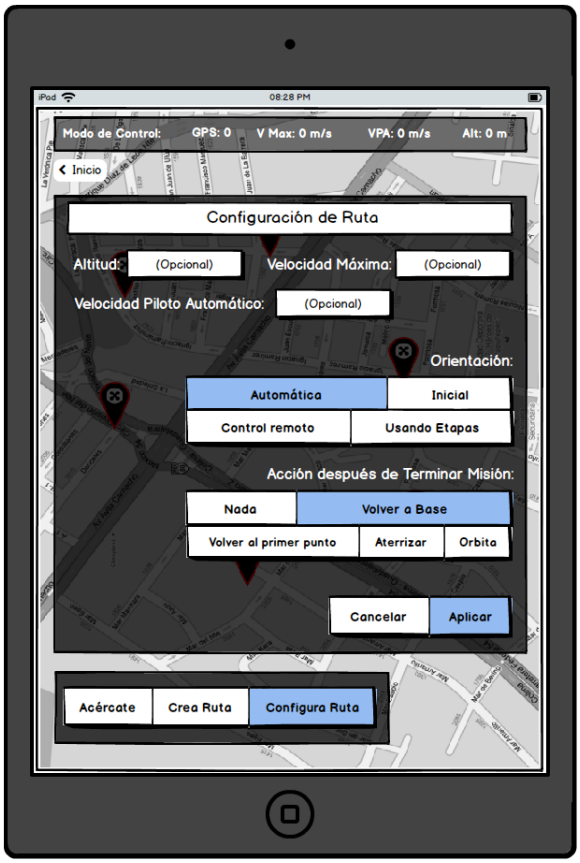


Imagen 31: Mockup del subsistema de Creación de Rutas, mientras se configura una ruta.

Inicio→ Crear Ruta→ Comienza Misión

Imagen 32: Mockup del subsistema de Creación de Rutas, mientras se ejecuta la misión.

PANTALLA 5: CONFIGURA RUTA	PANTALLA 6: COMENZAR MISIÓN
Descripción: Nos permite introducir, desde teclado, los siguientes datos necesarios para Configurar la Ruta. Los tres primeros son: *Altitud*, *Velocidad Máxima* y *Velocidad en Piloto Automático*. Respecto a la Orientación de la aeronave, se definen 4 posibilidades: *Automática*, *Inicial*, a definir por *Control Remoto* y *Usando* como referencia la posición de las distintas *Etapas*. Existen cuatro botones más para definir qué hacer depués de Terminar la Misión: *Nada, Volver a la Base, Volver al primer punto, Aterrizar u Orbitar*.	**Descripción:** Tras comenzar la misión, un icono que representa al dron nos indica la posición de la aeronave y su desplazamiento entre las distintas etapas.
Requisitos Funcionales asociados: RF-001 al RF-018, RF-021 al RF-023	**Requisitos Funcionales asociados:** RF-001 al RF-006, RF-018, RF-021 al RF-023
Requisitos No Funcionales asociados: RNF-001, RNF-003 al RNF-006	**Requisitos No Funcionales asociados:** RNF-001, RNF-003 al RNF-006

Tabla 11: Pantallas del subsistema de Creación de Rutas: Descripción y requisitos asociados.

3.3.3 Diseño general de Mockups: Galería

Inicio→ Galería→ Seleccionar **Inicio→ Galería→ Reproducir**

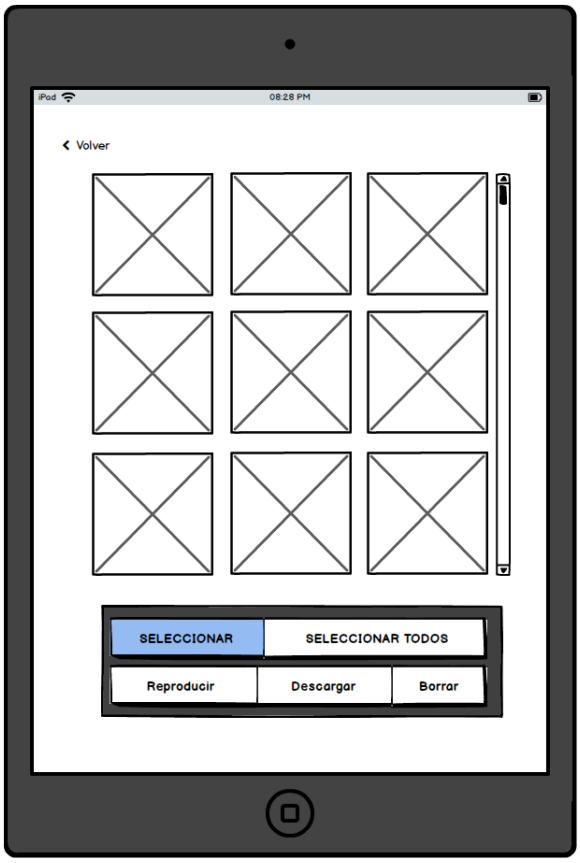


Imagen 33: Mockup del subsistema de Galería, mientras se selecciona un archivo multimedia.

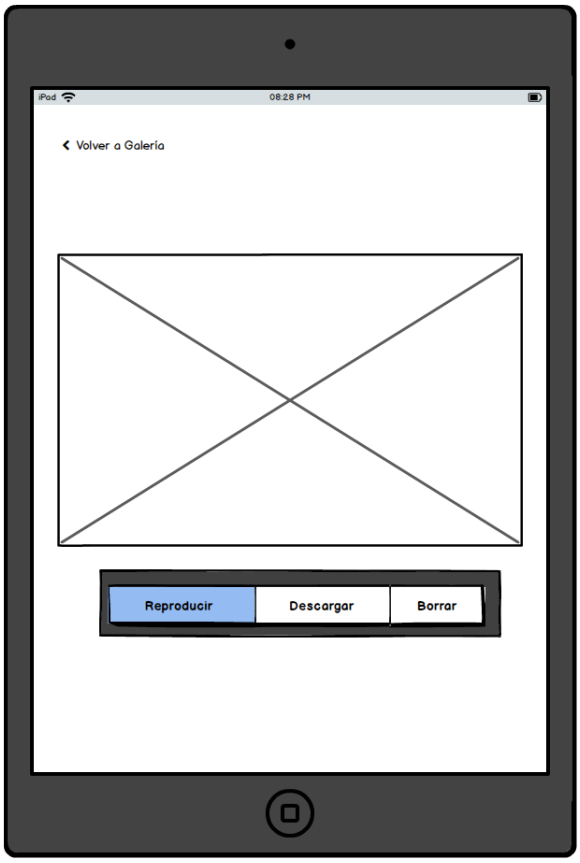


Imagen 34: Mockup del subsistema de Galería, mientras se visualiza un archivo multimedia.

PANTALLA 7: GALERÍA	PANTALLA 8: VER CLIP
Descripción: Visualización en mosaico de las miniaturas de los clips de vídeo o de las capturas fotográficas generadas durante la misión. En la parte inferior, se localizan dos botones que permiten una *Selección* individual de un clip o una *Selección* de *Todos* los clips. A continuación, los objetos seleccionados pueden ser *Reproducidos*, *Descargados* o *Borrados*.	**Descripción:** Se trata de la pantalla de Reproducción de cada clip individual. El tamaño de visualización estará acotado por el ancho del dispositivo, así como su orientación *Landscape o Portrait*.
Requisitos Funcionales asociados: RF-025 al RF-030	**Requisitos Funcionales asociados:** RF-025, RF-029, RF-030
Requisitos No Funcionales asociados: RNF-001 al RNF-006	**Requisitos No Funcionales asociados:** RNF-001 al RNF-007

Tabla 12: Pantallas del subsistema de Galería: Descripción y requisitos asociados.

3.3.4 Diseño general de Mockups: Documentación

Inicio→ Generar Documentación→ Información 1 **Inicio→ Generar Documentación→ Información 2**

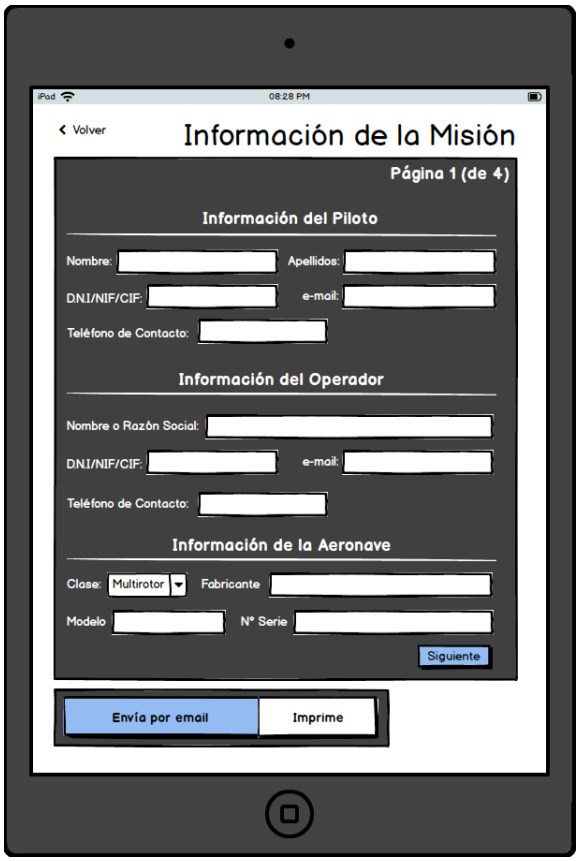


Imagen 35: Mockup del subsistema de Documentación, concretamente del primer bloque apartados.

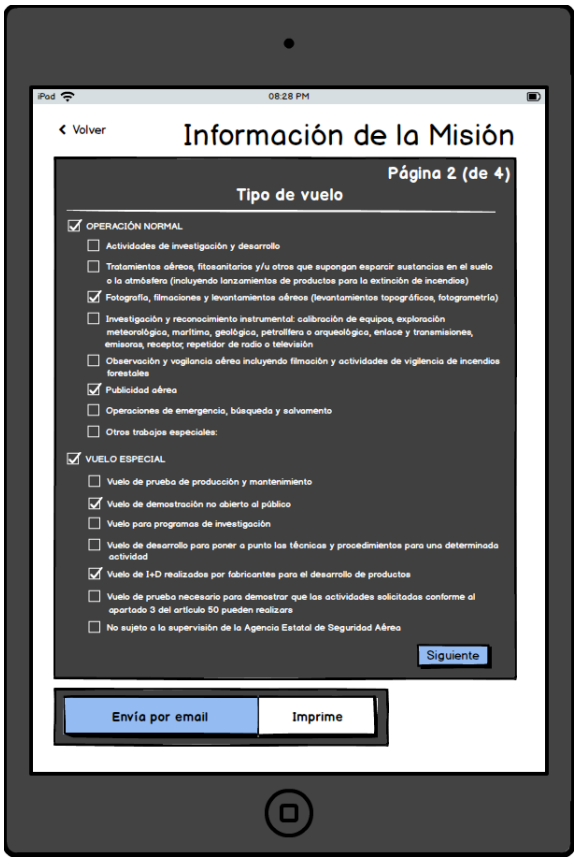


Imagen 36: Mockup del subsistema de Documentación, concretamente del segundo bloque apartados.

PANTALLA 9: INFORMACIÓN DE LA MISIÓN (1 de 4)	PANTALLA 10: INFORMACIÓN DE LA MISIÓN (2 de 4)
Descripción: Formulario a través del cual el usuario introduce información del *Piloto*, del *Operador* y de la *Aeronave*. En la parte inferior izquierda, dos nuevas funcionalidades (*Enviar por Email* e *Imprimir*) son accesibles desde sendos botones. Por último, en la esquina inferior derecha, se encuentra un botón que permite pasar a la página siguiente del formulario.	**Descripción:** Formulario a través del cual el usuario marca características del *Tipo de Vuelo*, a través de las cuales quedará definido como un vuelo de *Operación Normal* o un *Vuelo Especial*.
Requisitos Funcionales asociados: RF-038 al RF-041	**Requisitos Funcionales asociados:** RF-038 al RF-041
Requisitos No Funcionales asociados: RNF-001 al RNF-006	**Requisitos No Funcionales asociados:** RNF-001 al RNF-006

Tabla 13: Pantallas del subsistema de Documentación: Descripción y requisitos asociados.

Inicio→ Generar Documentación→ Información 3

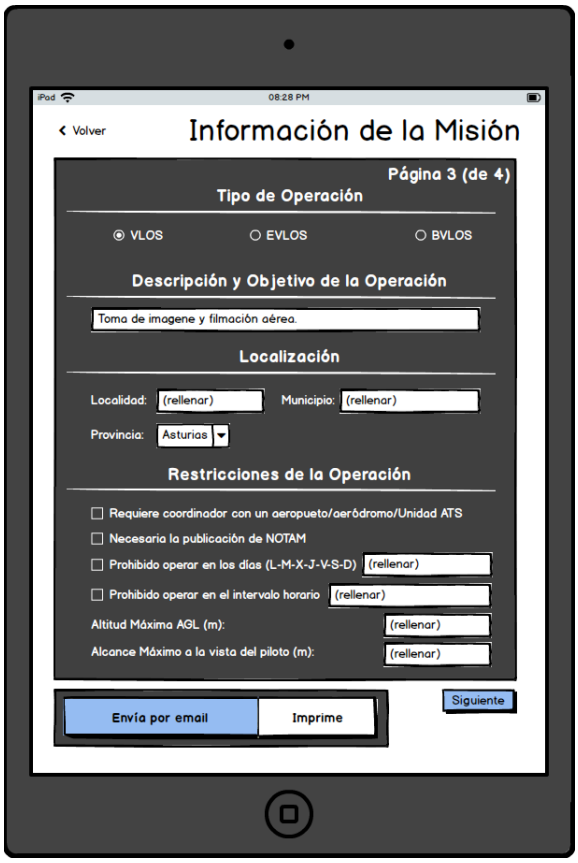


Imagen 37: : Mockup del subsistema de Documentación, concretamente del tercer bloque apartados.

Inicio→ Generar Documentación→ Información 4

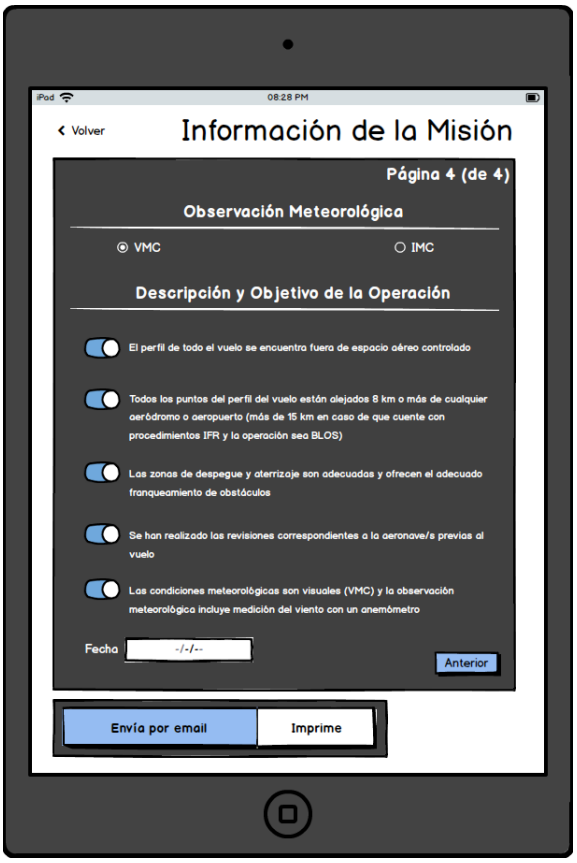


Imagen 38: : Mockup del subsistema de Documentación, concretamente del cuarto bloque apartados.

PANTALLA 11: INFORMACIÓN DE LA MISIÓN (3 de 4)	PANTALLA 12: INFORMACIÓN DE LA MISIÓN (4 de 4)
Descripción: Formulario a través del cual el usuario selecciona e introduce información del *Tipo de Operación*, de la *Descripción* y *Objetivo* de la misma, de la *Localización* y de sus posibles *Restricciones*.	**Descripción:** Formulario a través del cual el usuario selecciona información relativa a la *Observación Meteorológica* y a la *Descripción y Objetivo de la Operación*.
Requisitos Funcionales asociados: RF-038 al RF-041	**Requisitos Funcionales asociados:** RF-038 al RF-041
Requisitos No Funcionales asociados: RNF-001 al RNF-006	**Requisitos No Funcionales asociados:** RNF-001 al RNF-006

Tabla 14: Pantallas del subsistema de Documentación: Descripción y requisitos asociados.

3.4 Tratamiento del sonido

Se han añadido distintos efectos de sonido en aras de mejorar la experiencia del usuario en su navegación por las distintas secciones de la App. Se ha tratado con ello de hacerla más agradable, pero también más intuitiva, ya que los sonidos aparecen cuando el usuario de la App interactúa con los distintos botones de la interface, confirmándole que efectivamente ha pulsado un determinado botón. Este recurso se ha convertido hoy en día en cotidiano, pero continúa siendo muy efectivo para que el usuario se cerciore de que su gesto ha sido "recibido" por la interface y que debe esperar una reacción como consecuencia.

Para ello se ha tenido que trabajar con un Framework específico, denominado AutoToolBox. Los archivos de sonido utilizados proceden de librerías gratuitas con licencia Creative Commons (libre de derechos de autor). Para el tratamiento de sonido se han empleado distintos servicios online tanto para la conversión de formatos a m4a (el usado preferentemente en Xcode) [30], como para el acceso a bancos de sonidos [29].

Así nuestra aplicación cuenta con sonidos independientes para los siguientes botones/acciones: Accede a sección, Vuelve, Graba, Fotografía, Crea Pdf…

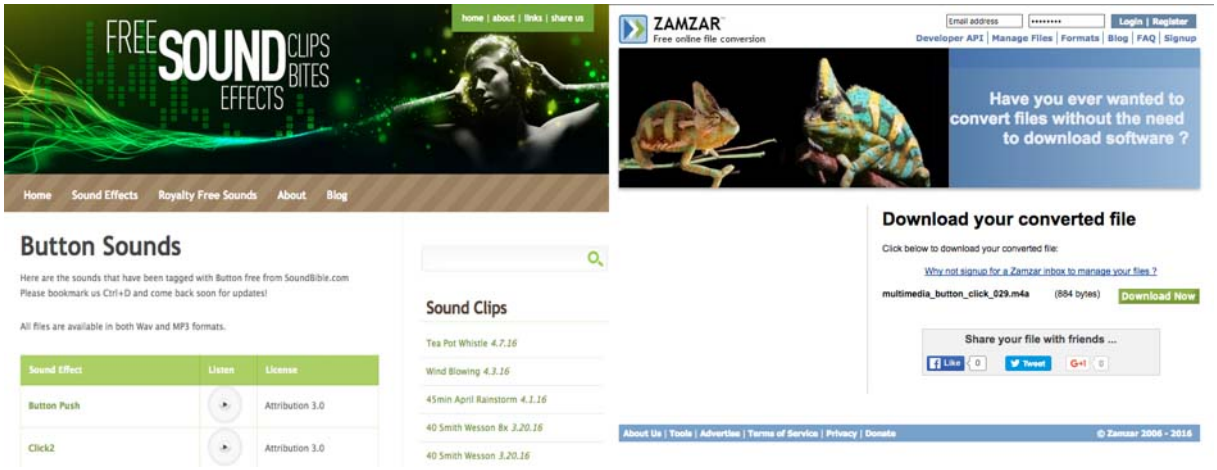


Imagen 39: Webs de librerías de archivos de audio libres de derechos de autor [29] y de tratamiento de sonidos [30].

4 DESCRIPCIÓN TÉCNICA DE LA SOLUCIÓN

4.1 Tecnologías utilizadas en la implementación

Para la implementación de nuestra aplicación vamos a contar con una serie de Frameworks, desarrollados principalmente por *Apple* y por *DJI*, que nos facilitarán la programación de nuestro código.

También se utilizarán librerías de otros desarrolladores (todas ellas software libre) para simplificar algunas de las tareas que realizará nuestra aplicación.

4.1.1 Frameworks utilizados

Para el desarrollo de nuestro sistema se utilizarán los siguientes Frameworks:

1. **DJI Mobile SDK 3.2:** nos permitirá acceder a las clases vinculadas con…
    a. el control de la cámara de nuestro dron.
    b. el control de la reproducción y grabación de archivos de vídeo.
    c. las operaciones sobre la tarjeta MicroSD.
    d. la geolocalización del dron a través de los distintos satélites GPS.
    e. los controles de aeronavegabilidad.
    f. los sistemas de seguridad.
2. **VídeoPreviewer Framework:** nos permitirá…
    a. el acceso en tiempo real al buffer de vídeo de la cámara del dron.
    b. su decodificación.
    c. la manipulación de las imágenes obtenidas.
3. **MapKit Framework**: nos permitirá…
    a. integrar mapas directamente en ventanas y vistas.
    b. realizar anotaciones sobre esos mapas.
4. **UIKit Framework**: proporciona infraestructura crucial para construir y gestionar aplicaciones iOS.
5. **AudioToolBox Framework**: Para la reproducción de sonidos vinculados a acciones.

4.1.2 Clases utilizadas

Las **principales** clases que se utilizarán a lo largo de nuestro desarrollo son las siguientes:

1. **DJIWaypoint**: esta clase representa a los objetos que en esta memoria hemos denominado como waypoints, estaciones, etapas o puntos intermedios de paso para un dron a la hora de efectuar su misión de vuelo. Es decir, una operación de vuelo de un dron consta de múltiples objetos waypoint. El usuario también puede definir acciones a realizar por el dron en cada waypoint.
2. **DJICamera**: se encarga de gestionar y reproducir los contenidos visuales generados por la cámara del dron. Proporciona para cambiar los ajustes de la cámara e interactuar sobre ella.
3. **DJIAircraft**: contiene componentes de la aeronave tales como la batería, el gimbal, el control de vuelo o la cámara.
4. **DJIFlightController**: esta clase contiene componentes del control de vuelo y proporciona métodos para enviar diferentes métodos a dicho control de vuelo.
5. **DJIPlaybackManager**: usada para interactuar con el sistema de reproducción de la cámara.
6. **DJIRemoteController**: esta clase representa el controlador remoto de la aeronave. Proporciona métodos para cambiar los ajustes del mando de control remoto (dispositivo que puede tener GPS, radio, botones, salidas y entradas de vídeo, batería, etc.).
7. **DJIBattery**: esta clase gestiona y proporciona información en tiempo real del estado de la batería.
8. **MKMapItem**: encapsula información sobre un punto específico de un mapa.
9. **MKLocalSearchRequest**: utilizado para usar mapas basados en determinados criterios de búsqueda.

4.1.3 Métodos utilizados

Se enumerarán a continuación los métodos que se han empleado durante el desarrollo de ***COMPandGO***.

1. **viewDidLoad**: se llama a este método después de que el controlador de vistas sea cargado en memoria. Se ejecuta al cargar la App.
2. **didReceiveMemoryWarning**: método al que se llama en caso de problemas d memoria.
3. **compruebaLongitud**: comprueba el número máximo de caracteres permitidos en los cuadros de texto del formulario.
4. **mailComposeController**: controlador del correo electrónico.
5. **crearPdf**: crea el archivo Pdf.
6. **dibujaFondoPdf**: dibuja el fondo de la página del archivo Pdf.
7. **imagenFondo**: define la imagen de fondo del Pdf (área que ocupa, dónde se empieza a dibujar, etc.).
8. **dibujaTexto**: coloca los textos de cada campo del Pdf dentro de su cuadro de texto correspondiente.
9. **pickerView**: permite seleccionar una opción dentro de una lista desplegable (por ejemplo, una provincia española de las 52 existentes.
10. **viewDidAppear**: notifica al viewController que una vista ha sido añadida (a la jerarquía de vistas). Se ejecuta al cargar la App.
11. **viewWillDisappear**: notifica al viewController que una vista ha sido eliminada (de la jerarquía de vistas). Al salir de la App, se libera.
12. **muestraAlertaConTitulo**: es un método para mostrar un alerta por pantalla cuando se produce un error. Por ejemplo un error de registro, tomando una foto, cuando hay un error de grabación de vídeo…
13. **registerApp**: comprobación de la APP Key del registro en el programa de desarrollo de *DJI* (método delegado de *DJI*).
14. **sdkManagerProductDidChangeFrom**: comprueba si el hardware *DJI* ha sufrido algún cambio (método delegado de *DJI*).
15. **sdkManagerDidRegisterAppWithError**: método para informar si hay errores en el registro de la APP (relacionados con el Bundle y la APP Key del registro de desarrollo) (método delegado de *DJI*).
16. **componentWithKey**: chequeo de la cámara de *DJI* (método delegado de *DJI*).

17. **conectaCamara**: método para conectar la cámara. Siempre que la cámara se desconecte o que algo haya cambiado en ella, lo necesitaremos para reconectarnos.
18. **cámara**: recibe datos de la cámara en formato H264 y se los envía al Video Previewer para decodificar.
19. **cámara**: mantiene actualizado el estado de la cámara en caso de que se hayan producido cambios en los ajustes.
20. **calculaSegundos**: reformatea los segundos en formato de tiempo UNIX (su origen de tiempos es el 1 de enero de 1970).
21. **alerta**: es una alerta de cámara.
22. **iniciaDatosMapa:** inicializa los arrays de datos de posiciones y orientaciones.
23. **muestraEstadoDeAlertas**: muestra el estado de las alertas.
24. **dismissStatusAlertView**: deshabilita las alertas.
25. **actualizaAlertaDeStatusConTitulo**: permite cambiar el mensaje de alerta al cambiar el estatus.
26. **vaciaPrevioDeVideo**: vacía el previo de vídeo.
27. **inicializaArrayDeDescargas**: inicializa los procesos de escritura y archivo temporal de descarga.
28. **IniciaBarraDeEstado**: inicializa el contador de la barra de estado del proceso de descarga.
29. **paraContador**: detiene el contador de la barra de estado del proceso de descarga.
30. **reseteaDescarga**: si se ha producido un error en la descarga, la resetea.
31. **descargaVideo**: descarga el vídeo seleccionado al dispositivo portátil.
32. **grabaImagenDescargada**: guarda en un archivo de imagen o de vídeo los datos obtenidos del proceso de descarga.
33. **noSePuedeGuardarLaImagen**: si se produce un error al guardar en un archivo los datos obtenidos en la descarga.
34. **coloca8Videos**: muestra en pantalla un mosaico con 8 vistas previas de archivos multimedia capturados.
35. **playbackManager**: gestor de reproducción (método delegado de *DJI*).
36. **updateUIWithPlaybackState**: refresca la imagen (método delegado de *DJI*).

4.2 Implementación del desarrollo Modelo-Vista-Controlador (MVC)

Resulta importante comprender el funcionamiento interno de nuestro código, su estructura arquitectónica. Para hacer este tipo de aplicaciones usamos el patrón de diseño Modelo-Vista-Controlador (MVC) [2]. Consiste en dividir en tres capas nuestra aplicación:

- Capa Modelo: La lógica de la aplicación, cómo funciona en su interior.
- Capa Vista: La presentación, lo que el usuario ve en pantalla.
- Capa Controlador: La información de todo lo que tenemos dentro de nuestra App.

El MVC es el patrón de diseño por el que nos vamos a guiar.

- Define en la aplicación tres roles. A cada objeto se le asigna el rol de Modelo, el de Vista o el de Controlador. Cada grupo de objetos perteneciente a un mismo rol tiene asignado una de las tres capas ya mencionadas.
- Define cómo se comunican las capas de objetos entre sí.

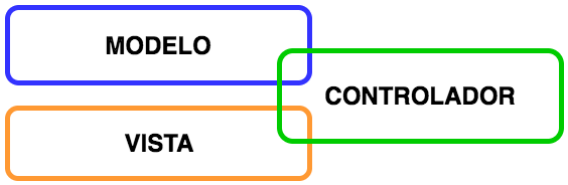


Imagen 40: Modelo-Vista-Controlador:: Esquema de cómo se comunican las tras capas de objetos entre sí [2].

- La capa Vista sólo puede comunicarse con la Capa Controlador. Lo hace a través de Outlets, mediante los cuales gestiona la interface del usuario. Pueden ser:
  - Acciones (Actions), mediante las que la Vista comunica al Controlador lo que está haciendo el usuario a través de la interface.
  - Delegados (Delegates), mediante los cuales el Controlador domina lo que pasa en la Vista.
  - Fuente de datos (Data Source), que es semejante al anterior.
- La Capa Modelo también puede comunicarse únicamente con la Capa Controlador, que es la encargada de sacar en pantalla la información del Modelo.

- Es decir, las capas Modelo y Vista no se comunican directamente entre ellas. Se relacionan siempre a través del Controlador. Si tuvieran relación directa afectaría a la eficacia del código y elevaría la dificultad a la hora de encontrar errores durante la tarea de su depuración.

4.2.1 Modelos

En el siguiente apartado se listan, a modo de resumen, las cabeceras de los principales métodos y acciones del subsistema implementado, así como unos breves comentarios de sus funciones.

4.2.1.1 Implementación del subsistema Cámara


```
//REGISTRO EN PROGRAMA DE DESARROLLO (COMPROBACIÓN DE APPKEY)-----------------------------
- (void)registerApp{}

//COMPROBACIÓN DE PRODUCTO-----------------------------------------------------------------
-(void) sdkManagerProductDidChangeFrom_DJIBaseProduct* _Nullable) oldProduct
to_DJIBaseProduct* _Nullable) newProduct{}

//REGISTRO DE ERRORES DE DJI---------------------------------------------------------------
- (void)sdkManagerDidRegisterAppWithError_NSError *)error{}

//DELEGADO DEL OBJETO CAMARA (CHEQUEO DE LA CÁMARA DE DJI)---------------------------------
-(void) componentWithKey_NSString *)key changedFrom_DJIBaseComponent *)oldComponent
to_DJIBaseComponent *)newComponent {}

//CONECTA CON LA CÁMARA--------------------------------------------------------------------
- (DJICamera*) conectaCamara {}

//RECIBE DATOS DE LA CAMARA (H264) Y SE LOS ENVIA AL VÍDEO PREVIEWER (PARA DECODIFICAR)----
-(void) Cámara_DJICamera *)camera didReceiveVídeoData_uint8_t *)vídeoBuffer
length_size_t)size{}

//MANTENER ACTUALIZADO EL ESTADO DE LA CAMARA (Por si hubo cambios de Settings)------------
-(void) Cámara_DJICamera*)camera didUpdateSystemState_DJICameraSystemState*)systemState
{}

//CALCULA SEGUNDOS (por diferencia con 1970, según sistema estándar)-----------------------
- (NSString *)CalculaSegundos_int)seconds
{}

//CAPTURA IMAGEN---------------------------------------------------------------------------
- (IBAction)AccionCaptura_id)sender {}

//CAPTURA VÍDEO----------------------------------------------------------------------------
- (IBAction)AccionGraba_id)sender {}

//INTERCAMBIA MODO DE VÍDEO/FOTO-----------------------------------------------------------
- (IBAction)AccionCambioDeModo_id)sender {}
```

Código 1: Código de implementación del subsistema de Cámara de ***COMPandGO***.

4.2.1.2 Implementación del subsistema Geolocalización


```
//Inicia los datos de la Mision
-(void)initData {}
//Inicia los datos de la Interface
-(void) initUI {}
//Registro en el programa de desarrollo de DJI
-(void) registerApp{}
//Control de Errores programa de desarrollo de DJI
#pragma mark DJISDKManagerDelegate Methods
- (void)sdkManagerDidRegisterAppWithError_NSError *_Nullable)error{}
//Registro de Cambios de producto en DJI
- (void)sdkManagerProductDidChangeFrom_DJIBaseProduct *_Nullable)oldProduct to_DJIBaseProduct
*_Nullable)newProduct{}
//Geolocaliza y acerca mapa
- (void)focusMap{}
//Actualiza Localizacion
-(void) startUpdateLocation{}
//Añade Etapa
- (void)addWaypoints_UITapGestureRecognizer *)tapGesture{}
//Configuracion de Etapa
- (void)cancelBtnActionInDJIWaypointConfigViewController_ConfiguracionViewController
*)waypointConfigVC{}
//Ejecuta Después de terminar Etapa
- (void)finishBtnActionInDJIWaypointConfigViewController_ConfiguracionViewController
*)waypointConfigVC{}
- (void)stopBtnActionInGSButtonVC_Botones *)GSBtnVC{}
//Acción borra marcas en Mapa
- (void)clearBtnActionInGSButtonVC_Botones *)GbtnVC{}
//Registro en el programa de desarrollo de DJI
- (void)focusMapBtnActionInGSButtonVC_Botones *)GSBtnVC{}
//Comienza Vuelo
- (void)startBtnActionInGSButtonVC_Botones *)GSBtnVC{}
//Intercambia Modo
- (void)switchToMode_DJIGSViewMode)mode inGSButtonVC_Botones *)GSBtnVC{}
//Añade Botón
- (void)addBtn_UIButton *)button withActionInGSButtonVC_Botones *)GSBtnVC{}
//Gestor de Localizaciones
- (void)locationManager_CLLocationManager *)manager didUpdateLocations_NSArray *)locations{}

#pragma mark MKMapViewDelegate Method
- (MKAnnotationView *)mapView_MKMapView *)mapView viewForAnnotation_id
<MKAnnotation>)annotation{}
#pragma mark DJIFlightControllerDelegate
- (void)flightController_DJIFlightController *)fc
didUpdateSystemState_DJIFlightControllerCurrentState *)state
```

Código 2: Código de implementación del subsistema de Geolocalización de ***COMPandGO***.

4.2.1.3 Implementación del subsistema Galería


```
#pragma mark Gestión de Alertas
//MÉTODO PARA MOSTRAR ALERTAS
- (void)MuestraAlertaConTitulo_NSString *)title withMessage_NSString *)message{}
//ALERTAS DE CAMARA
- (void)Alerta_UIAlertView *)alertView clickedButtonAtIndex_NSInteger)buttonIndex{}
//MUESTRA ESTADO DE ALERTAS
-(void) MuestraEstadoDeAlertas {}
//DESAHABILITA ALERTAS
-(void) dismissStatusAlertView {}
- (void)ActulizaAlertaDeStatusConTitulo_NSString *)title message_NSString *)message {}
//REGISTRO PROGRAMA DE DESARROLLO DJI (COMPROBACIÓN DE API COMPandGO)
- (void)registerApp{}
//COMPRUEBA POSIBLES CAMBIOS DE PRODUCTO
-(void) sdkManagerProductDidChangeFrom_DJIBaseProduct* _Nullable) oldProduct
to_DJIBaseProduct* _Nullable) newProduct{}
//MUESTRA POSIBLES ERRORES DE REGISTRO
- (void)sdkManagerDidRegisterAppWithError_NSError *)error{}

#pragma mark Control de Cámara
//VACIA PREVIO DE VÍDEO
- (void)VaciaPrevioDeVídeo {}
//CONECTA CAMARA
- (DJICamera*) conectaCamara {}
//RECIBE DATOS DE LA CAMARA (H264) Y SE LOS ENVIA AL VÍDEO PREVIEWER (PARA DECODIFICAR)
- (void)Cámara_DJICamera *)camera didReceiveVídeoData_uint8_t *)vídeoBuffer
length_size_t)size{}
//MANTENER ACTUALIZADO EL ESTADO DE LA CAMARA (Por si hubo cambios de Settings-
(void)Cámara_DJICamera *)camera didUpdateSystemState_DJICameraSystemState *)systemState{}

#pragma mark Descarga
//INICIALIZA ARRAY DE DESCARGAS
- (void)InicializaArrayDeDescargas{}
//INICIALIZA BARRA DE ESTADO (NUMÉRICA)
- (void)ActualizaBarraDeEstado_NSTimer *)updatedTimer{}
//INICIALIZA BARRA DE ESTADO (NUMÉRICA)
- (void)IniciaBarraDeEstado{}
//PARA CONTADOR
- (void)ParaContador{}
//RESETEA DESCARGA
- (void)ReseteaDescarga{}
//DESRCARGA VÍDEO
-(void) DescargaVídeo{}
//GRABA IMAGEN DESCARGADA
- (void)GrabaImagenDescargada{}
//NO PUEDE GUARDAR LA IMAGEN
- (void)NoPuedeGuardarLaImagen_UIImage *)image didFinishSavingWithError_NSError *)error
contextInfo_void *)contextInfo{}
//CALCULA SEGUNDOS (por diferencia con 1970, según sistema estándar)
- (NSString *)TransformaSegundos_int)seconds{}

#pragma mark – PlayBack de Vídeos
//INICIALIZA MODO COLOCA 8 VÍDEOS EN PANTALLA
- (void) Coloca8Vídeos{}

//GESTOR DE REPRODUCCIÓN (DELEGADO DE DJI)
- (void)playbackManager_DJIPlaybackManager *)playbackManager
didUpdatePlaybackState_DJICameraPlaybackState *)playbackState{}

//REFRESCA IMAGEN (DELEGADO DE DJI)
- (void)updateUIWithPlaybackState_DJICameraPlaybackState *)playbackState{}

#pragma mark Acciones de Botones
- (IBAction)AccionVerUno_id)sender {}
- (IBAction)AccionVerTodos_id)sender {}
- (IBAction)AccionMultiPrevio_id)sender {}
```

```
- (IBAction)AccionBotonDePlay_id)sender {}
- (IBAction)AccionBotonPara_id)sender {
- (IBAction)AccionBotonSeleccionaUno_id)sender {}
- (IBAction)BotonAccionBorra_id)sender {}
- (IBAction)BotonAccionDescarga_id)sender {}
- (IBAction)AccionBotonSeleccionaTodos_id)sender {


#pragma mark – Acciones de Sonido

- (IBAction)SonidoVuelve_id)sender {AudioServicesPlaySystemSound(BotonTipoVuelve);}
- (IBAction)SonidoTipo1_id)sender {AudioServicesPlaySystemSound(BotonTipo1);}
- (IBAction)SuenaClick_id)sender {AudioServicesPlaySystemSound(SuenaClick);}
```

Código 3: Código de implementación del subsistema de Galería de ***COMPandGO***.

4.2.1.4 Implementación del subsistema Información AESA

4. Descripción técnica de la solución	4.2 Implementación Modelo VC	4.2.1 Modelos

```
#pragma mark – Comprobaciones
//COMPROBACIONES DE TAMAÑOS MÁXIMOS PERMITIDOS EN LOS CAMPOS DEL FORMULARIO- (void)
compruebalongitud{}

#pragma mark – E-Mail
//COMTROLADOR DE EMAIL
- (void) mailComposeController_MFMailComposeViewController *)controller
didFinishWithResult_MFMailComposeResult)result error_NSError *)error}

#pragma mark – Caracteriza el PDF
//CREAR PDF
- (void)CrearPdf_NSString *)rutaArchivo{}

//DIBUJA FONDO PDF
-(void)dibujaFondoPdf{}

//DEFINE IMAGEN DE FONDO DEL PDF
-(void)imagenFondo{}

#pragma mark – Rellena el PDF
//DIBUJA TEXTO
-(void)dibujaTexto{}

//PROVINCIA Y CLASE
- (void) pickerView_UIPickerView *)pickerView didSelectRow_NSInteger)row
inComponent_NSInteger)component{}

#pragma mark – Acciones PDF
//GENERAR PDF
//
- (IBAction)GenerarPdfBoton_id)sender {}

//ENVÍA EMAIL
- (IBAction)EnviarEmail_id)sender {}

#pragma mark – Acciones Oculta/Muestra
- (IBAction)IntroduceFechaBoton_id)sender{}
- (IBAction)AccionCierraFlotanteClase_id)sender {}
- (IBAction)AccionCierraFlotanteProvincia_id)sender {}
- (IBAction)cambiaClase_id)sender {}
- (IBAction)CambiaProvincia_id)sender {}
- (IBAction)OcultarTeclado_id)sender {}
- (IBAction)OcultaFlotantePdf_id)sender {}
- (IBAction)CierraFechaBoton_id)sender {}
- (IBAction)MuestraFlotantePdf_id)sender {}
```

Código 4: Código de implementación del subsistema de Documentación de ***COMPandGO***.

4.2.2 Controladores

La relación de los controladores utilizados, clasificados según los subsistemas a los que pertenecen, es la siguiente:

- Subsistema de LaunchScreen:
  - **InicioViewController**
- Subsistema de Menú principal:
  - **PrincipalViewController**
- Subsistema de Documentación:
  - **FormularioViewController**
- Subsistema de Cámara:
  - **GrabacionViewController**
- Subsistema de Creación de Rutas:
  - **Mision**
  - **Anotacion**
  - **AnotacionView**
  - **Mapas**
  - **Botones**
  - **ConfiguracionViewController**
- Subsistema de Galería:
  - **GaleriaViewController**
  - **MultiPantallaViewController**
- Subsistema secundario de Ayuda (local y online):
  - **AyudaViewController**
  - **WebViewController**

Diagrama de interrelación de clases y métodos asociados a las clases pertenecientes a los subsistemas de LaunchScreen y Menú Principal, de Cámara y de Documentación [33].

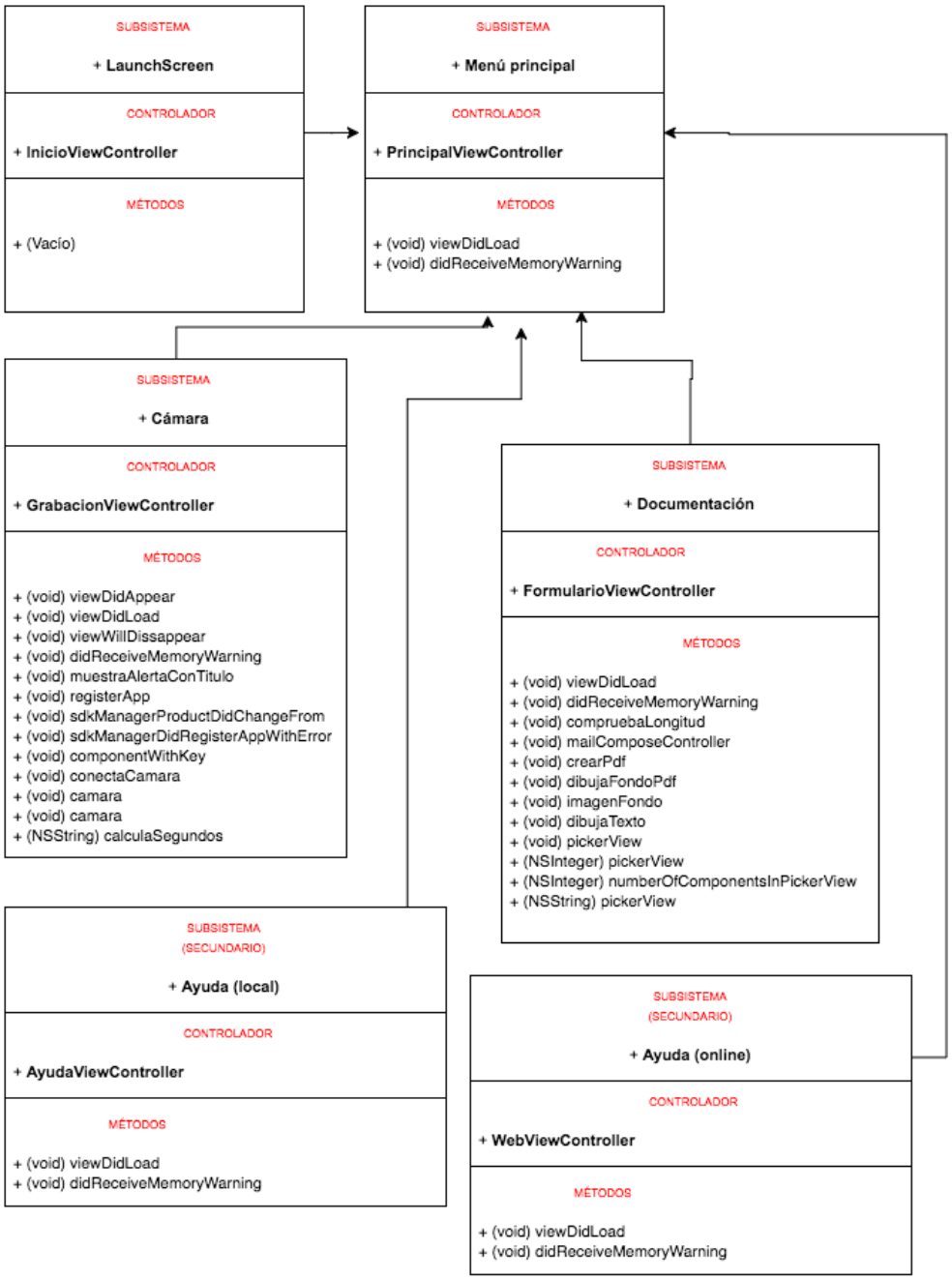


Imagen 41: Diagrama de clases de los subsistemas LaunchScreen y Menú principal, Cámara y Documentación.

Diagrama de interrelación de clases y métodos asociados a las clases pertenecientes al subsistema de Creación de Rutas [33].

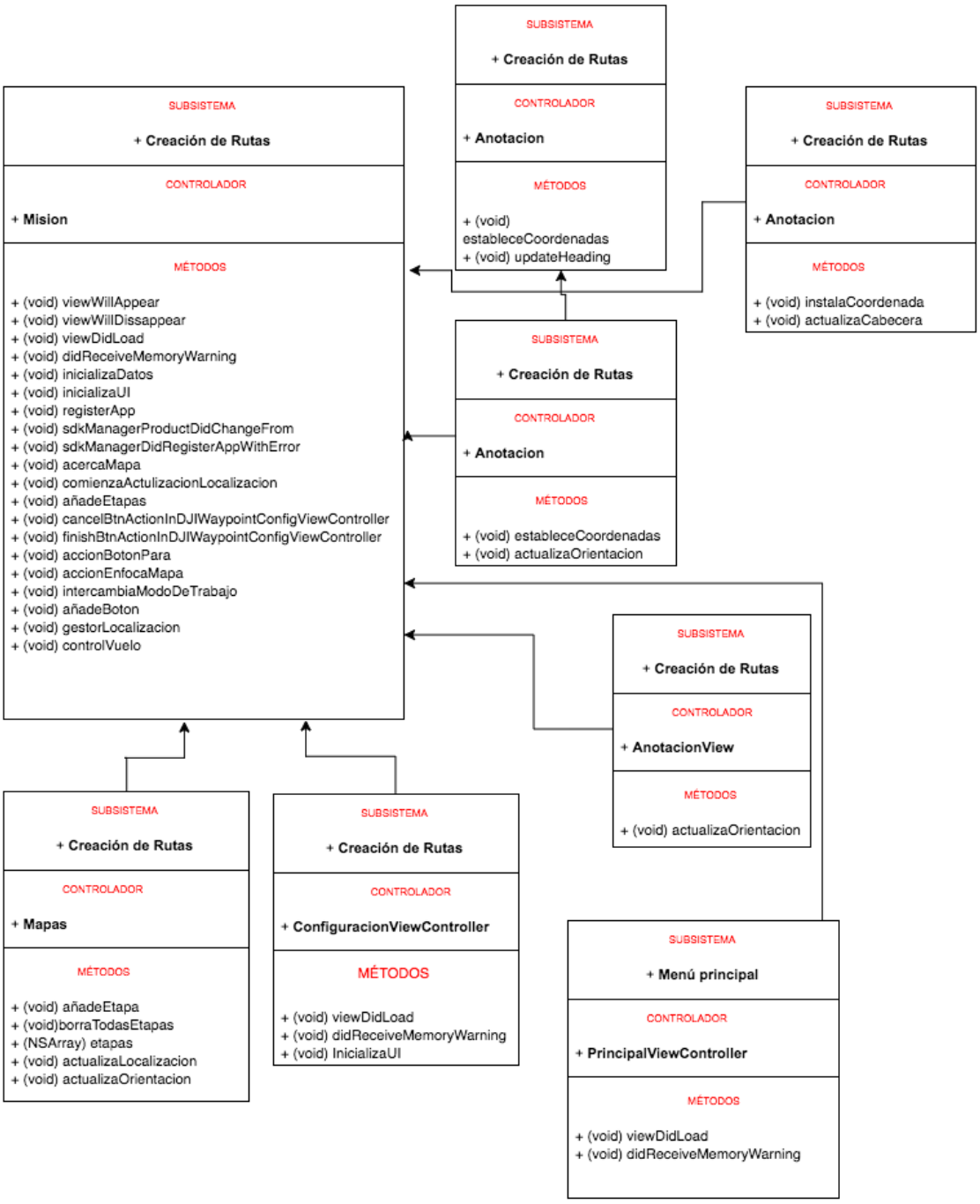


Imagen 42: Diagrama de clases del subsistema Creación de Rutas.

Diagrama de interrelación de clases y métodos asociados a las clases pertenecientes al subsistema de Galería y al subsistema secundario de Ayuda [33].

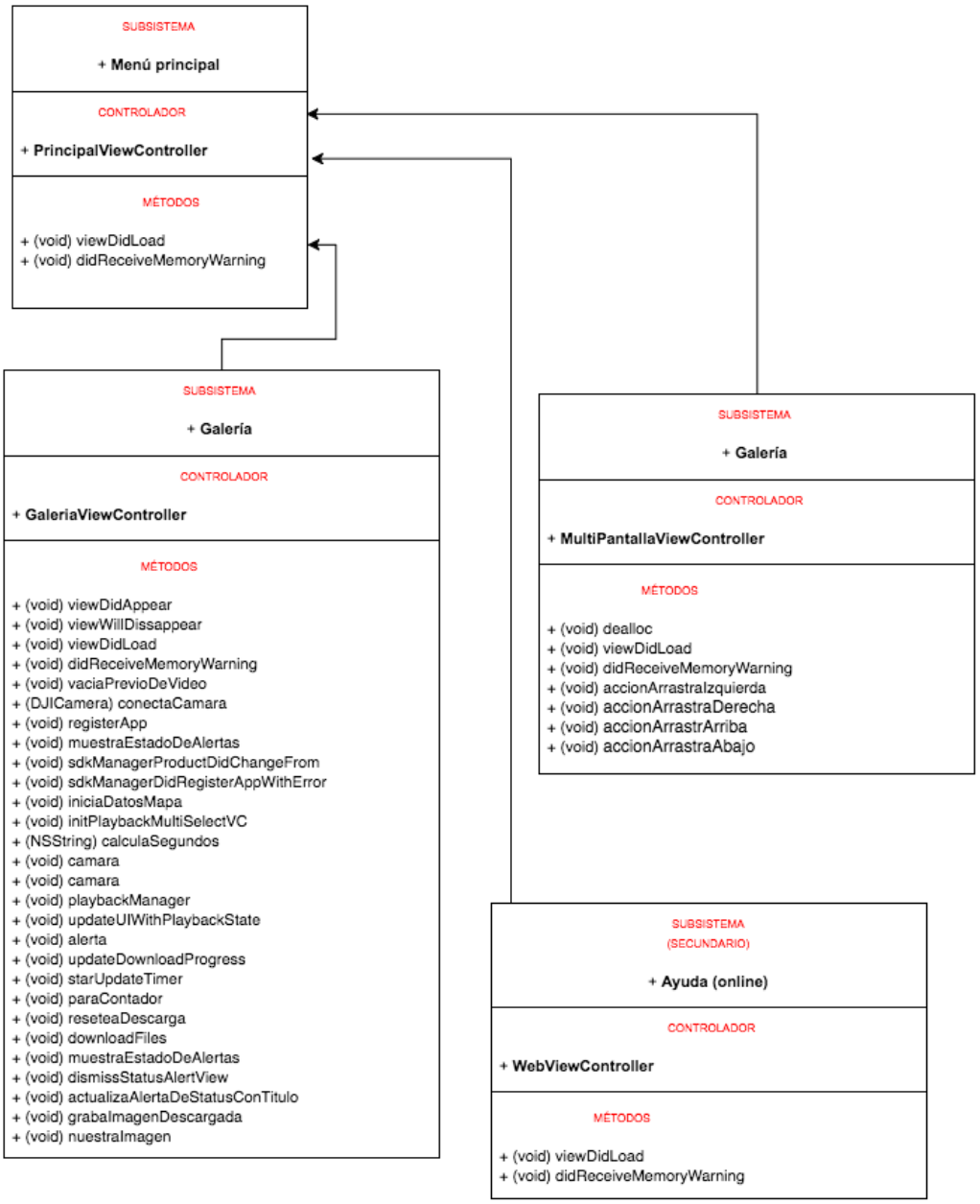


Imagen 43: Diagrama de clases del subsistema Galería y del subsistema secundario Ayuda.

4.2.3 Vistas

Se ha trabajado con un total de 8 vistas principales en un único StoryBoard, tal y como se puede apreciar en en el mapa de conexiones siguiente.

Imagen 44: mapa de conexiones: Storyboard con las 8 vistas principales.

Cada vista principal se corresponde con los siguientes subsistemas:

Vista nº	Subsistema
1	LaunchScreen
2	Menú principal
3	Documentación
4	Cámara
5	Creación de rutas
6	Galería
7	Ayuda (local)
8	Ayuda (online)

Tabla 6: Correspondencia entre las vistas principales y los subsistemas de **COMPandGO**.

Como complemento a estas vistas (y en UIView independientes) se han diseñado he implementado a modo de ventanas emergentes (pop-ups) varias vistas secundarias, útiles en el proceso de configuración de los ajustes de nuestra aplicación. A continuación, a modo de ejemplo, algunas de estas vistas secundarias de la vista de Creación de Rutas.

Imagen 45: Vista secundaria de la de Creación de Rutas (pop-up).

Imagen 46: Vista secundaria de la de Creación de Rutas (pop-up).

Imagen 47: Vista secundaria de la de Creación de Rutas (pop-up).

5 PRUEBAS DE VALIDACIÓN

5.1 Validación técnica de la solución

De cara a comprobar el correcto funcionamiento de ***COMPandGO***, se realizarán una serie de pruebas, que dividiremos en dos grandes grupos:

- El grupo de las pruebas de funcionamiento ordinario de la App a nivel de experiencia del usuario. Son las que aplicaremos a cada subsistema de ***COMPandGO***. Veremos que serán pruebas denominadas de caja negra.
- El grupo de las pruebas a nivel del código de la App. Son pruebas de integración y rendimiento, también denominadas como pruebas de caja blanca.

5.1.1 Pruebas de cada subsistema

Las pruebas de caja negra son aquellas en las que se efectúa una ejecución normal de la aplicación. Consisten en comprobar que el desempeño de la App es correcto en todos sus aspectos, obteniéndose las respuestas correctas a cada acción.

De cara a plasmar el proceso de validación del correcto funcionamiento del conjunto de la aplicación ***COMPandGO***, se ha optado por mostrar pues cómo se han testeado cada uno de los distintos subsistemas que la componen. Estos subsistemas son:

1. Subsistema de LaunchScreen y Menú principal.
2. Subsistema de Documentación.
3. Subsistema de Cámara.
4. Subsistema de Creación de Rutas.
5. Subsistema de Galería.
6. Subsistema secundario de Ayuda (local y online).

5.1.1.1 Pruebas del subsistema de LaunchScreen y Menú principal

Imagen 48: Prueba de confirmación de carga del subsistema de LaunchScreen.

El LaunchScreen carga correctamente, mostrando en pantalla los gráficos acotados respecto a los márgenes de la pantalla, tanto en orientación *Portrait* como *Landscape*. La interface exige tocar la pantalla para acceder al Menú principal de la aplicación mediante una transición Flip Horizontal, reproduciéndose un sonido.

Imagen 49: Prueba de confirmación de carga del subsistema de Menú principal.

A continuación se llega al Menú principal de opciones, desde donde podemos acceder a los cuatro subsistemas principales de la App. En la esquina superior derecha, dos iconos nos permiten acceder a la página de ayuda o volver atrás.

5.1.1.2 Pruebas del subsistema de Documentación

El primer icono (de los cuatro que destacan en el Menú principal) nos permite acceder a la sección de Documentación, en la que podemos rellenar a través de un formulario todos los campos necesarios para la generación automática de la documentación oficial de AESA. Para desplazarnos por la pantalla (debido a la gran cantidad de información) contamos con un scroll vertical. Si escogemos la orientación *Landscape*, a la parte derecha de los campos podemos leer una breve ayuda.

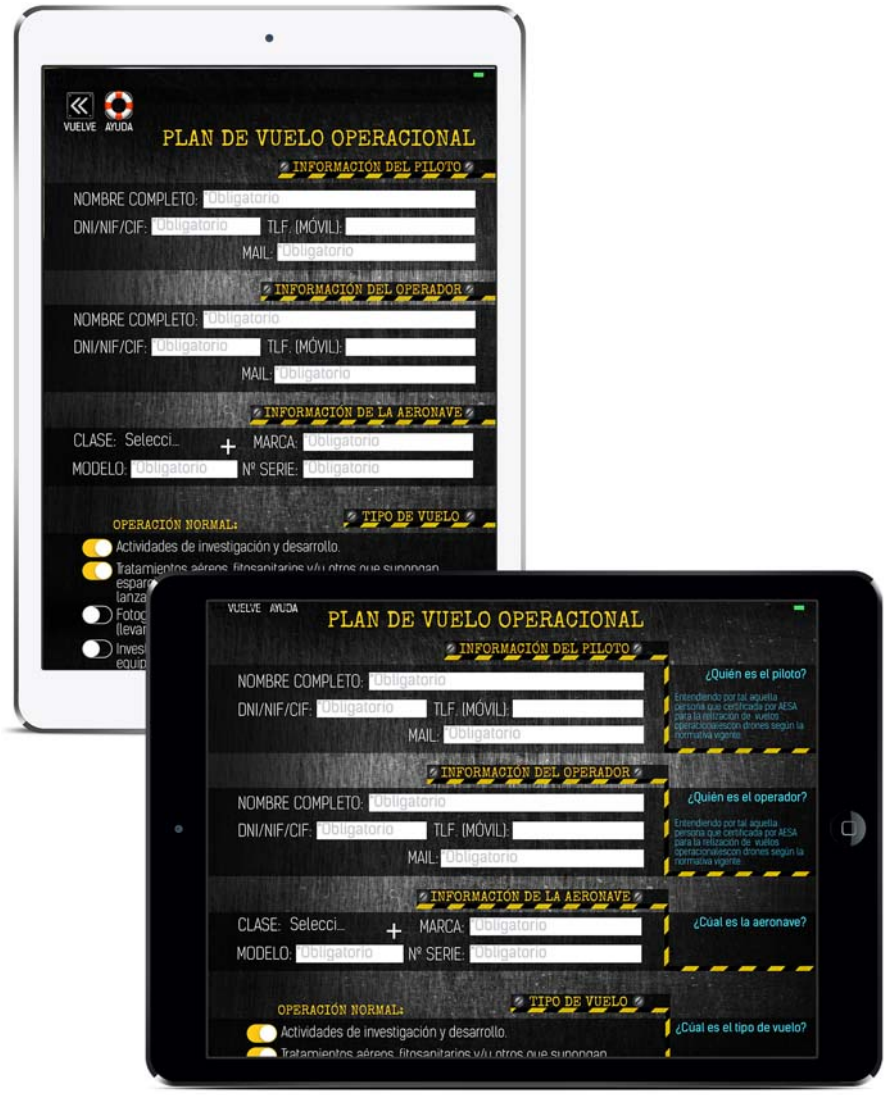
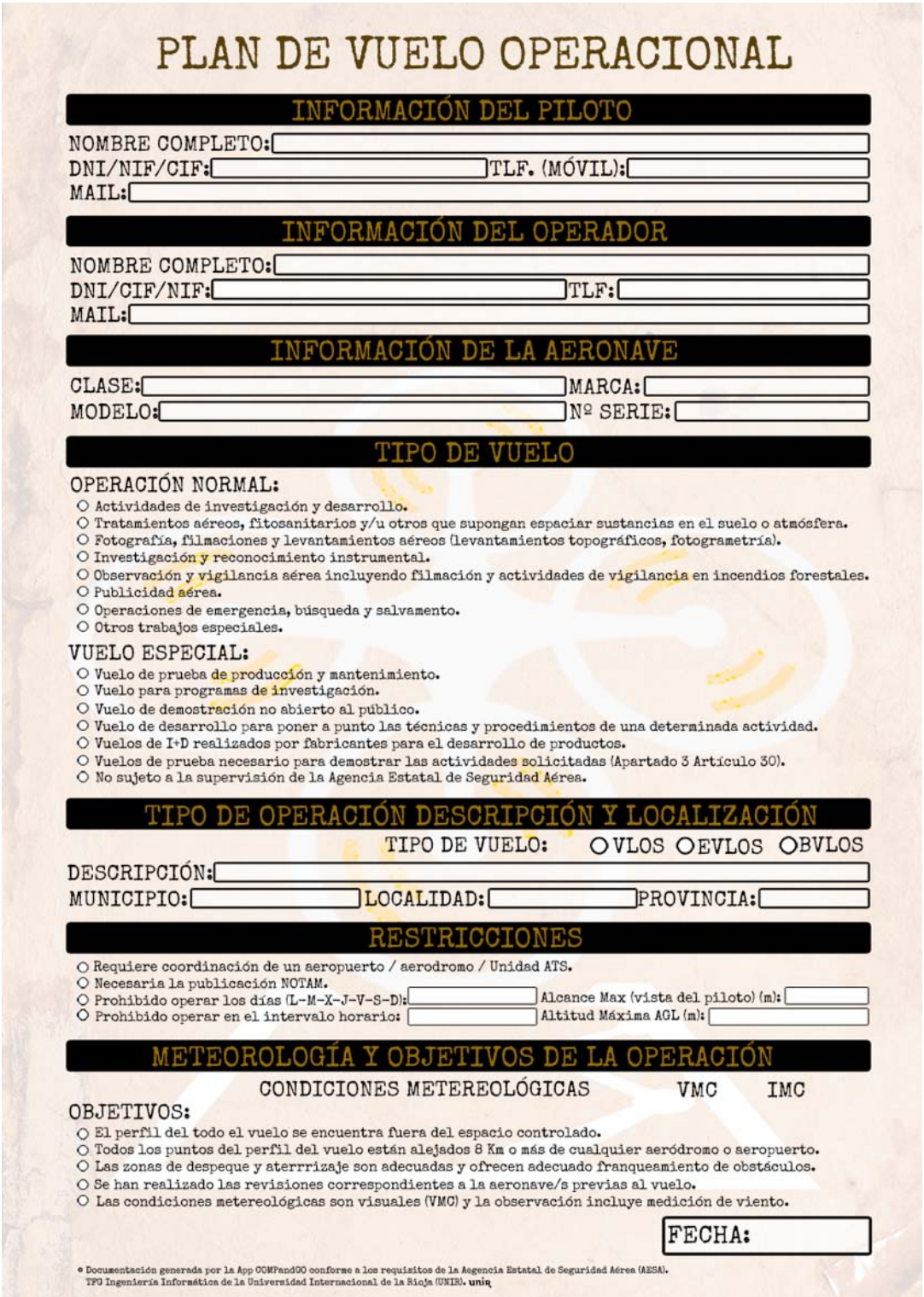


Imagen 50: Prueba de confirmación de carga del subsistema de Documentación.

A medida que vamos rellenando el documento, iremos descendiendo gracias al scroll vertical, hasta llegar a la parte inferior del documento. Ahí disponemos de tres botones que nos permitirán crear el archivo Pdf oficial con nuestros datos, comprobarlo directamente en la pantalla o enviarlo por email.

Imagen 51: Prueba de confirmación de carga del subsistema de Documentación.



PLAN DE VUELO OPERACIONAL

INFORMACIÓN DEL PILOTO

NOMBRE COMPLETO:
DNI/NIF/CIF: TLF. (MÓVIL):
MAIL:

INFORMACIÓN DEL OPERADOR

NOMBRE COMPLETO:
DNI/CIF/NIF: TLF:
MAIL:

INFORMACIÓN DE LA AERONAVE

CLASE: MARCA:
MODELO: Nº SERIE:

TIPO DE VUELO

OPERACIÓN NORMAL:

- ○ Actividades de investigación y desarrollo.
- ○ Tratamientos aéreos, fitosanitarios y/u otros que supongan espaciar sustancias en el suelo o atmósfera.
- ○ Fotografía, filmaciones y levantamientos aéreos (levantamientos topográficos, fotogrametría).
- ○ Investigación y reconocimiento instrumental.
- ○ Observación y vigilancia aérea incluyendo filmación y actividades de vigilancia en incendios forestales.
- ○ Publicidad aérea.
- ○ Operaciones de emergencia, búsqueda y salvamento.
- ○ Otros trabajos especiales.

VUELO ESPECIAL:

- ○ Vuelo de prueba de producción y mantenimiento.
- ○ Vuelo para programas de investigación.
- ○ Vuelo de demostración no abierto al público.
- ○ Vuelo de desarrollo para poner a punto las técnicas y procedimientos de una determinada actividad.
- ○ Vuelos de I+D realizados por fabricantes para el desarrollo de productos.
- ○ Vuelos de prueba necesario para demostrar las actividades solicitadas (Apartado 3 Artículo 30).
- ○ No sujeto a la supervisión de la Agencia Estatal de Seguridad Aérea.

TIPO DE OPERACIÓN DESCRIPCIÓN Y LOCALIZACIÓN

TIPO DE VUELO: ○ VLOS ○ EVLOS ○ BVLOS

DESCRIPCIÓN:
MUNICIPIO: LOCALIDAD: PROVINCIA:

RESTRICCIONES

- ○ Requiere coordinación de un aeropuerto / aerodromo / Unidad ATS.
- ○ Necesaria la publicación NOTAM.
- ○ Prohibido operar los días (L-M-X-J-V-S-D): Alcance Max (vista del piloto) (m):
- ○ Prohibido operar en el intervalo horario: Altitud Máxima AGL (m):

METEOROLOGÍA Y OBJETIVOS DE LA OPERACIÓN

CONDICIONES METEREOLÓGICAS VMC IMC

OBJETIVOS:

- ○ El perfil del todo el vuelo se encuentra fuera del espacio controlado.
- ○ Todos los puntos del perfil del vuelo están alejados 8 Km o más de cualquier aeródromo o aeropuerto.
- ○ Las zonas de despeque y aterrrizaje son adecuadas y ofrecen adecuado franqueamiento de obstáculos.
- ○ Se han realizado las revisiones correspondientes a la aeronave/s previas al vuelo.
- ○ Las condiciones metereológicas son visuales (VMC) y la observación incluye medición de viento.

FECHA:

* Documentación generada por la App COMPandGO conforme a los requisitos de la Agencia Estatal de Seguridad Aérea (AESA).
TFG Ingeniería Informática de la Universidad Internacional de la Rioja (UNIR). unir

Imagen 52: Prueba de confirmación de carga del subsistema de Documentación: Pdf generado por ***COMPandGO***.

La aplicación nos enviará con nuestros datos este archivo Pdf con las información requerida por AESA.

5.1.1.3 Pruebas del subsistema de Cámara

Imagen 53: Prueba de confirmación de carga del subsistema de Cámara.

El segundo icono del Menú principal da acceso al subsistema Cámara, desde donde obtendremos imágenes en tiempo real de la cámara del dron. En un selector de modo de trabajo, situado en la esquina superior derecha, podremos elegir entre el modo fotografía o el modo vídeo. También podremos acceder desde aquí a los subsistemas de la Galería o de la Ayuda.

5.1.1.4 Pruebas del subsistema de Creación de Rutas

A través del tercer botón del Menú principal, podremos acceder al subsistema Creación de Rutas, desde el que podremos definir una posible ruta de nuestra aeronave. En la esquina superior izquierda dos botones nos permitirán acercarnos a nuestra posición actual en el mapa y añadir sobre él las etapas que deseemos que realice. También podremos ver información en tiempo real del vuelo en los dos marcadores que se han diseñado para tal efecto en la parte inferior.

Imagen 54: Prueba de confirmación de carga del subsistema de Creación de Rutas.

Desde el botón Ajustes podremos también configurar las características del vuelo entre cada una de las etapas, así como lo que hará el dron después de terminar la misión (modo de Vuelta a la base o vuelta a casa).

Imagen 55: Prueba de confirmación de carga del subsistema de Creación de Rutas: Sección Ajustes.

5.1.1.5 Pruebas del subsistema de Galería

Imagen 56: Prueba de confirmación de carga del subsistema de Galería: Un archivo seleccionado.

El tercer subsistema es el de Galería, que nos permite acceder a las grabaciones de vídeo y fotografías almacenadas en la tarjeta MicroSD del Dron, así como reproducirlas, descargarlas a nuestro dispositivo portátil (*iPad*) o borrarlas. También es posible acceder desde esta misma pantalla a la cámara de la aeronave.

5.1.1.6 Pruebas del subsistema secundario de Ayuda (local y online)

5. Pruebas de validación	5.1 Validación técnica de la solución	5.1.1 Pruebas de cada subsistema

Imagen 57: Prueba de confirmación de carga del subsistema secundario de Ayuda (local).

Desde la sección de Ayuda dispondremos de unas breves indicaciones de cómo funciona cada subsistema de la aplicación, así como un acceso a la web de la App.

Finalmente si pulsamos sobre el botón de Visite nuestra web, nos llevará a la página corporativa de la App, en la que encontraremos documentación ampliada y vídeos explicativos (aún no realizados) de la aplicación en funcionamiento.

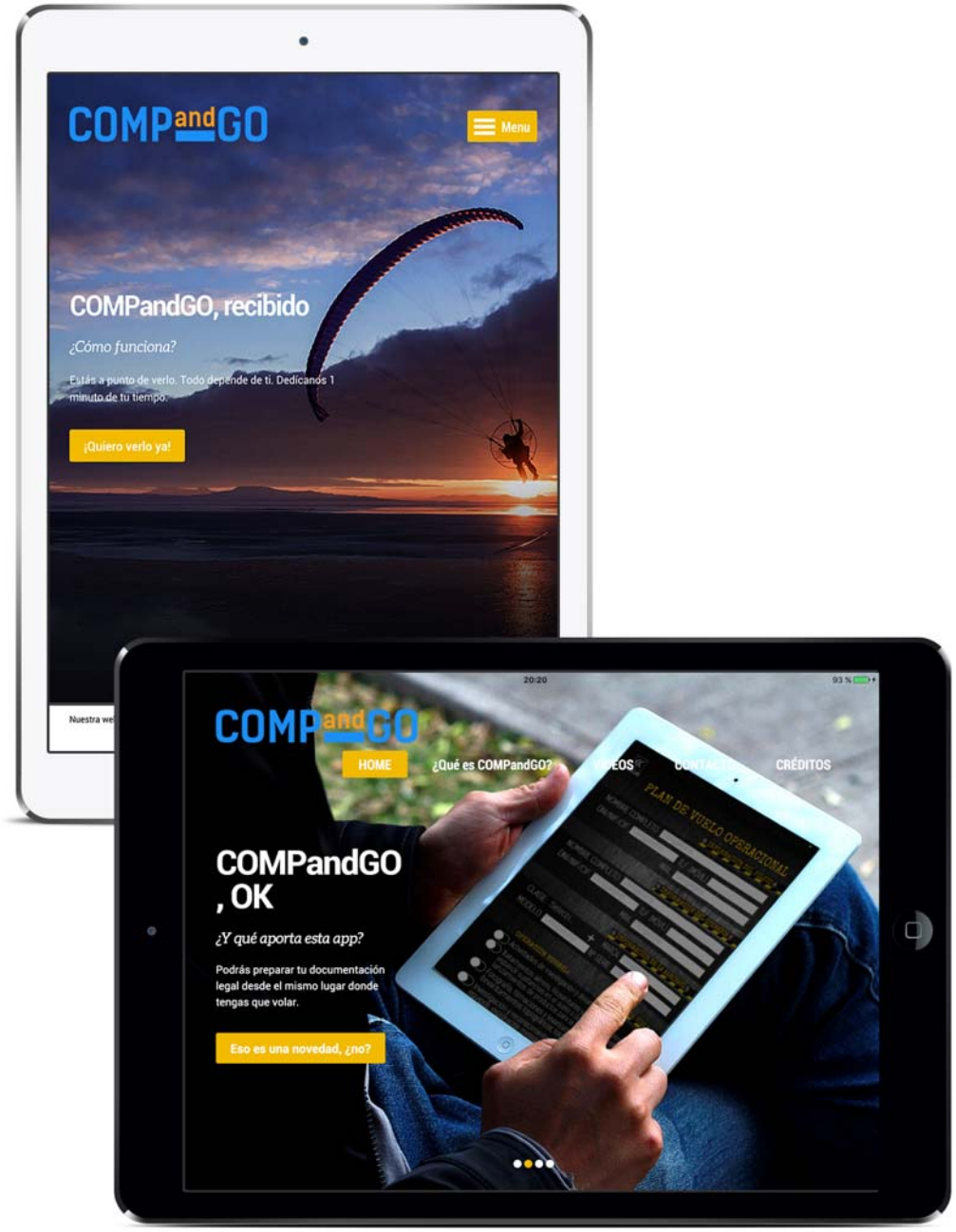


Imagen 58: Prueba de confirmación de carga del subsistema secundario de Ayuda (online).

5.1.2 Pruebas de integración y rendimiento

El segundo tipo de pruebas son las pruebas de caja blanca. Son aquellas que se realizan ya a nivel del código propiamente dicho y se encargan de comprobar que el funcionamiento "interno" de la App es el correcto.

5.1.2.1 Integración y depurado

Para realizar las pruebas de integración y depurado del código fuente de ***COMPandGO*** se emplearán herramientas que el propio Xcode pone a nuestra disposición a tal efecto.

La primera de ellas se trata de *Static analyzer*, que podemos encontrar en el menú Xcode => Product => Analyze. Su cometido es doble:

- Analizar la totalidad del código fuente: análisis de todas las líneas de código, así como de sus interdependencias.
- Presentarnos un informe sobre los problemas que a priori podrían presentarse provocando un funcionamiento irregular de nuestra App. Esos problemas pueden ser de varios tipos, achacables a nosotros (errores de pureza o elegancia del código, errores lógicos de implementación, etc.) e incluso ser ajenos a nuestro desempeño (errores en los Frameworks ue se usan en el proyecto).

Los resultados arrojados por *Static Analyzer* han sido de utilidad para detectar incongruencias en definiciones de datos. Esta herramienta ha sido de uso recurrente a lo largo de todo el proceso de programación.

La segunda herramienta que nos aporta Xcode para el cometido que nos ocupa se llama *Energy impact gauge*. Permite ver en términos de eficiencia energética, los consumos de memoria, CPU, GPU y resto de hardware. Para trabajar con ella una vez que la App se está ejecutando (en el simulador o en el iPad), hay que acceder al menú Xcode => View => Navigators => Show Debug Navigator. En el caso de ***COMPandGO*** los resultados arrojados por esta herramienta no reportan información conflictiva o susceptible de un mantenimiento perceptivo.

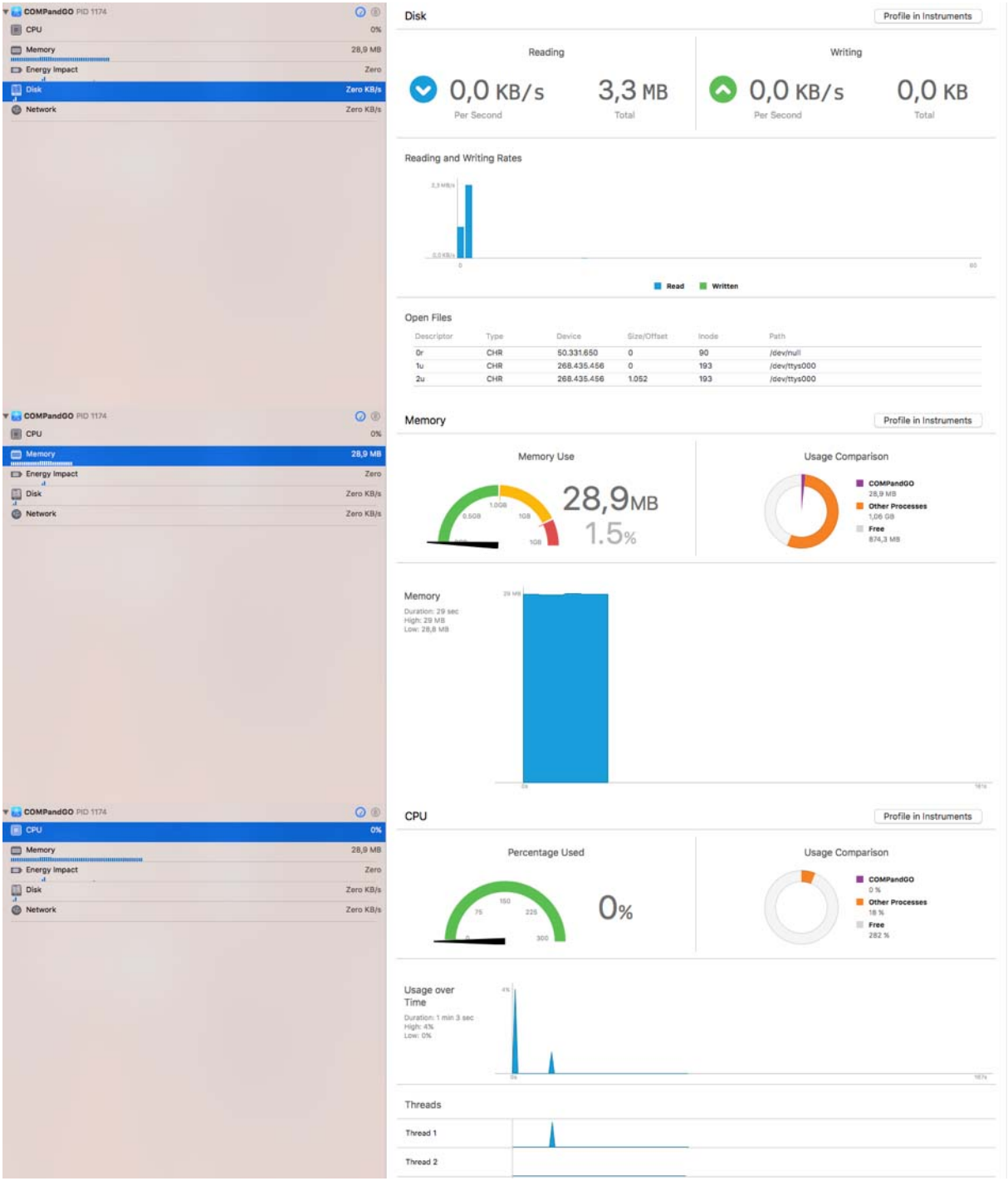


Imagen 59: Mosaico de imágenes de eficiencia de ***COMPandGO***.

5.1.2.2 Landing Page y QR-Codes

Una landing page es una página que ayuda a convertir a los usuarios de un contenido de su interés en clientes finales. Básicamente consiste en ofrecer algo del interés de un usuario, invitarle a que visite una página de destino, poner a su disposición un formulario de contacto (o lo que es lo mismo, de facilitarnos sus datos). Esa información personal de un potencial cliente es muy valiosa, y más si nos da una oportunidad perfecta para contactar con él de forma directa y personalizada.

Por ejemplo, un usuario conoce una App a través de una tienda de aplicaciones o de una entrada de un blog, donde evidentemente tiene que encontrar a su alcance un enlace que le ofrezca la posibilidad de ampliar la información. Ese enlace le llevará a la landing page, donde encontrará fácilmente un formulario de contacto, que deberá cumplimentar para recibir algo a cambio (tal vez una descarga gratuita, o ayuda personalizada, etc.).

Se ha considerado de interés la creación de una landing page como complemento al proyecto de la App desarrollada. Los objetivos son:

- Aumentar la información disponible sobre la App, dedicando para ello todo el espacio que se considere necesario, sin las limitaciones y restricciones que existen en las tiendas online de aplicaciones, donde el espacio destinado a estos particulares suele ser limitado.
- Informar a los usuarios de la naturaleza del origen del proyecto. Reflejar cómo se asienta sobre los cuatro pilares que son la UNIR, el código libre (GitHub), Xcode + *iPad* (*Apple*) y los drones + Frameworks de *DJI*.
- Alojar material multimedia (fotografías y vídeos) ilustrativo.
- Ampliar la ayuda para explicar el funcionamiento de la App.
- Ofrecer un canal de contacto entre usuario y desarrollador, pensado como vehículo para enviar sugerencias, preguntas y demás aportes que supongan un valor añadido tanto para el emisor como para el receptor.

El alojamiento de la landing page se ha realizado en un subdominio de la web del desarrollador, siendo su página de inicio la siguiente:

http://josebrana.com/compandgo/

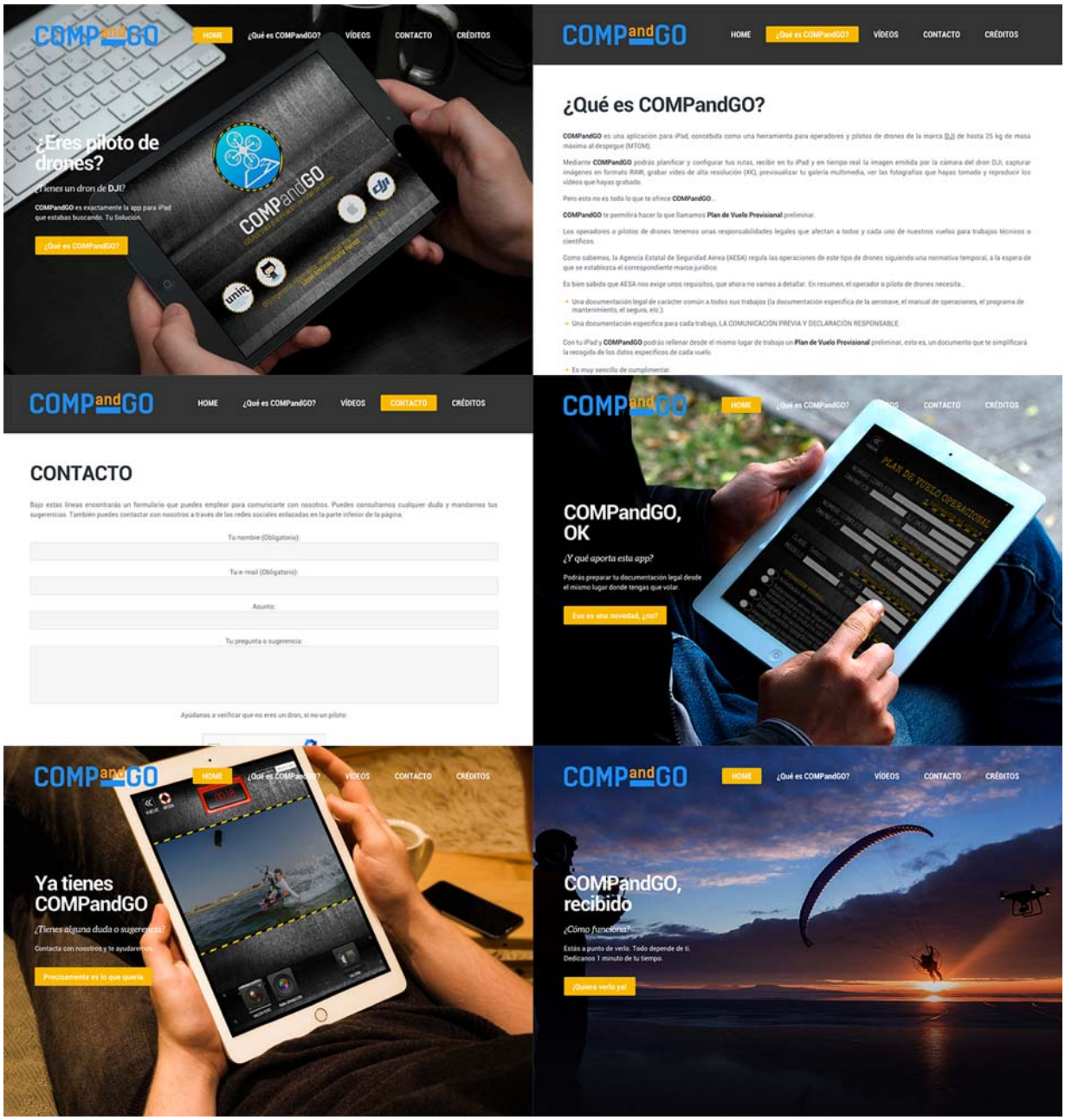


Imagen 60: Mosaico con capturas de pantalla de la landing page (www.josebrana.com/compandgo).

Se ha elegido WordPress como el sistema de gestión de contenidos (CMS) de la web (esto es, la interface que permite gestionar los datos de la base de datos alojada en el hosting del dominio). Se caracteriza por estar concebida como plataforma de publicación personal (no en vano se encuentra entre las primeras elecciones de los bloggers) y enfocada a primar los valores estéticos, la máxima compatibilidad con los estándares web y la facilidad de uso. Según **Rodríguez** (2016), WordPress representa hoy en día el 23,4 % de todas las webs (sin ceñirse a las bitácoras; hablamos de todo tipo de webs, como tiendas, páginas presenciales, de servicios, etc.) y el 60,7 % de webs con CMS [15].

En la elección del diseño han primado los siguientes aspectos:

- Claridad y brevedad.
- Diseño *Responsive* o adaptativo: Se trata de buscar la correcta visualización de los contenidos en todo tipo de dispositivos, independientemente de su tamaño de pantalla (teléfonos inteligentes, tabletas, etc.).
- Alojamiento de contenidos multimedia, especialmente fotografías de alta calidad y vídeos de alta resolución.
- Formulario de contacto: en una página individual, para reflejar la importancia que tiene para nosotros.

Hoy en día el uso cotidiano de dispositivos móviles dotados de cámara (tabletas, teléfonos inteligentes, etc.) ha ampliado enormemente las posibilidades de interacción entre el terminal y el mundo que le rodea. Una manera sencilla de aprovechar este aspecto para realizar acciones automáticas en estos aparatos es el empleo de QR codes (códigos QR).

Los códigos QR no dejan de ser una evolución de los códigos de barras, aunque esta vez extendidos hasta las dos dimensiones, ya que esta vez la información que incluyen está codificada dentro de un cuadrado. Permiten automatizar acciones como abrir direcciones URL, enviar un email, ubicar unas coordenadas geográficas en una web de mapas, etc.

Para la confección de este proyecto se han generado dos QR codes. Para ello se ha empleado la herramienta gratuita *delivr* [31]. En concreto se han generado los correspondientes al acceso directo al GitHub del código fuente del proyecto y a la landing page. Ahondando en este último caso, también se ha empleado el servicio de *delivr* que permite llevar un control estadístico del uso del código QR de acceso a la landing page (número de accesos, etc.).

Imagen 61: Código QR de acceso al GitHub donde está alojado el código fuente de **COMPandGO**.

Imagen 62: Código QR de acceso a la landing page (www.josebrana/compandgo).

5.1.3 Simuladores

5.1.3.1 Simuladores de *iPad* en Xcode

Ya hemos dedicado a Xcode, el entorno de desarrollo de la aplicación, su correspondiente apartado en esta memoria. Este entorno lógicamente permite compilar los códigos programados. Esta compilación puede hacerse de dos maneras:

- En dispositivos reales: Conectados físicamente al ordenador. Para ello es imprescindible darse de alta como *Apple* Developer y unirse específicamente al programa de desarrolladores de *Apple*.
- En dispositivos virtuales, a los que podemos acceder a través de un Simulador.

El Simulador es un software que se carga a parte del Xcode, aunque se lanza desde éste (razón por la que no los calificamos como independientes). Nos permite simular la conexión del Xcode con diversos dispositivos de las distintas familias de *Apple*. Así, con una instalación básica de Xcode 7 podemos elegir entre cinco tipos de *iPad* (desde el *iPad 2* hasta al *iPad Pro*) y siete modelos de *iPhone* (desde el *iPhone 4S* hasta el *iPhone 6*+). No obstante, y de manera también gratuita, podemos acceder a simuladores de otros modelos de esas familias, así como a los de otros dispositivos como el *Apple TV*.

El proceso de desarrollo de una App para cualquier dispositivo móvil en general provoca que al programador le surja la necesidad habitual de tener que probar la evolución de la aplicación instalándola muchas veces en el dispositivo de destino. Y es que recurrir siempre y en todo momento a la compilación en dispositivos virtuales mediante el Simulador no despeja todas las dudas que le pueden surgir al programador, cosa que sí hace la compilación en el dispositivo genuino. Por ejemplo, si las funcionalidades de la App requieren acceder a la cámara del dispositivo portátil (el Simulador no tiene ni siquiera acceso a una webcam incorporada al ordenador del programador).

Durante este proyecto se ha recurrido en innumerables ocasiones a la compilación de la App en un *iPad Air 2* con pantalla retina, de cara a comprobar que la calidad de la resolución de las distintas pantallas de la App era la más adecuada para que un usuario potencial pueda desempeñar un trabajo profesional con la calidad prometida en nuestras especificaciones (fotografías de alta calidad en formato RAW y vídeos 4K de alta resolución).

5.1.3.2 Simuladores de *DJI* (*Assistant2* v1.0.4)

Ya hemos comentado la gran ayuda que ha supuesto el simulador de Xcode para emular la instalación de la App en dispositivos móviles de *Apple*. Y por otra parte, hemos reflejado sus limitaciones cuando se necesitaba testear la comunicación de la App con la cámara del dron.

Algo parecido nos ocurre en este nuevo apartado. Primero comentemos la necesidad que nos surgió, después cómo se solventó con la ayuda de un nuevo simulador y finalmente las limitaciones que le aún y con todo le encontramos al remedio.

Una de las funciones de la App es la configuración de rutas de vuelo, para lo cual es necesario el manejo de mapas, posicionamiento GPS, etc. En la configuración se pueden regular parámetros del vuelo de la aeronave, etc. ¿Cómo comprobar si funciona todo correctamente una vez que la App esté instalada en el *iPad*? ¿Saliendo a pilotar el dron, desplazándose hasta una zona donde el vuelo sea legal, etc.? Y lo que es peor, puede que la App manifieste un error desde el principio y haya que regresar en el acto. ¿Y habría que repetir este larguísimo procedimiento cada vez que queramos probar la App? Sería del todo inviable.

Por suerte, *DJI* pone a disposición de los usuarios un simulador de vuelo para sus drones, que permite ser gobernado manejando el propio control remoto del dron.

Este simulador se llama *Assistant2 v1.0.4*. Se puede descargar de forma gratuita desde la web de *DJI*, en el apartado reservado a las descargas para *Phantom 4* [32].

Tras instalarlo, lo arrancamos y nos pide que conectemos el dron a nuestro ordenador mediante un cable miniUSB. Una vez conectado, le quitamos las hélices (por precaución) y lo encendemos. *DJI Assistant* lo detectará como dispositivo conectado.

Imagen 63: *DJI Assistant* detecta que se ha conectado al equipo un dron encendido.

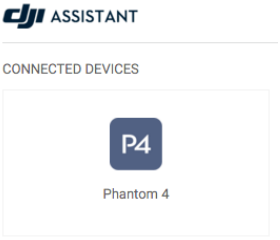


Imagen 64: El dron *DJI Phantom 4* aparece en la lista de equipos conectados.

En este punto es muy importante que no toquemos las palancas que regulan el despegue del dron. Si por despiste las manipulamos y las llevamos a la posición de despegue, los motores se pondrían en marcha (aunque no despegaría, ya que le hemos quitado las hélices con anterioridad; pero si se hubiera producido un doble despiste, entonces el dron sí despegaría). Haciendo doble clic sobre el dispositivo, llegamos a una sección donde tenemos varias opciones. Podremos actualizar el firmware del dron, calibrarlo o entrar al simulador. En los ajustes del vuelo simulado, podremos definir las coordenadas del lugar donde queremos simular el vuelo y ajustar la velocidad del viento. Una vez ajustados estos parámetros, pulsamos sobre *Start Simulating*.

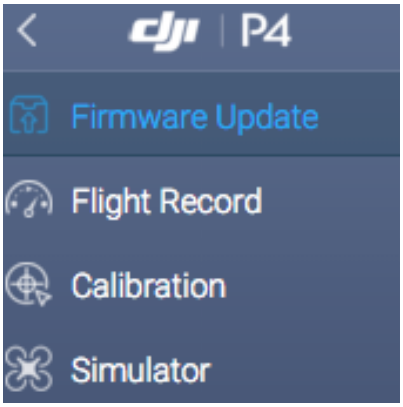


Imagen 65: Opciones para trabajar con el dron detectado.

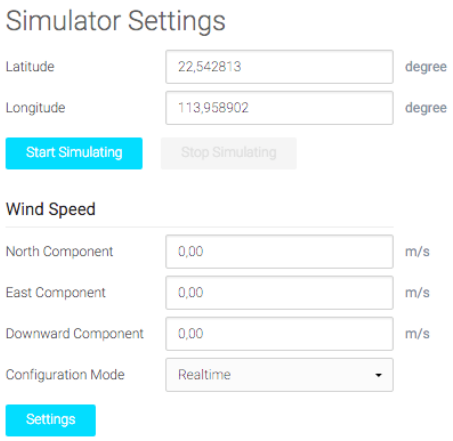


Imagen 66: Parámetros de lo ajustes del vuelo simulado.

Aparecerá una nueva pantalla en nuestro ordenador (la del *DJI Assistant* no se cerrará) donde veremos un dron. Y ahora sí. Como si de un vídeo juego se tratara, podemos comenzar a controlarlo con el mando del dron. Si pinchamos sobre esta ventana con el botón derecho del ratón, nos saldrán unas opciones. Una es muy interesante, y no es otra que la posibilidad de que el dron al desplazarse deje tras de sí una trazadora de color azul, Así será muy fácil identificar sus movimientos. Mientras volamos, la pantalla del *DJI Assistant* nos mostrará la telemetría del vuelo.

Más aún, si antes de despegarlo le conectamos el *iPad* al mando del dron y ejecutamos ***COMPandGO***, podremos configurar una misión de vuelo. Tras configurarla, veremos en la pantalla del *iPad* el mapa con las coordenadas geográficas definidas en *DJI Assistant*, el icono del dron y a la propia aeronave desempeñar la misión que se le ha ordenado.

Por último, de nuevo la limitación obvia que tiene el simulador es que no podemos probar la cámara del dron (la calidad de sus imágenes, la resolución de sus vídeos, etc.).

Inevitablemente, si queremos testear este particular, sí que habrá que realizar vuelos reales al aire libre.

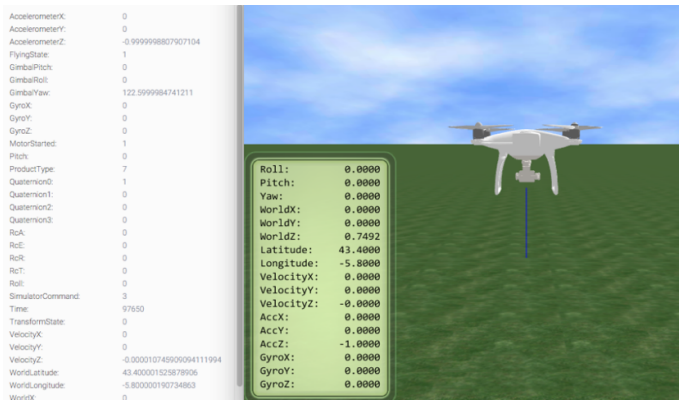


Imagen 67: El vuelo simulado comienza con el despegue del dron (su coordenada Z aumenta).

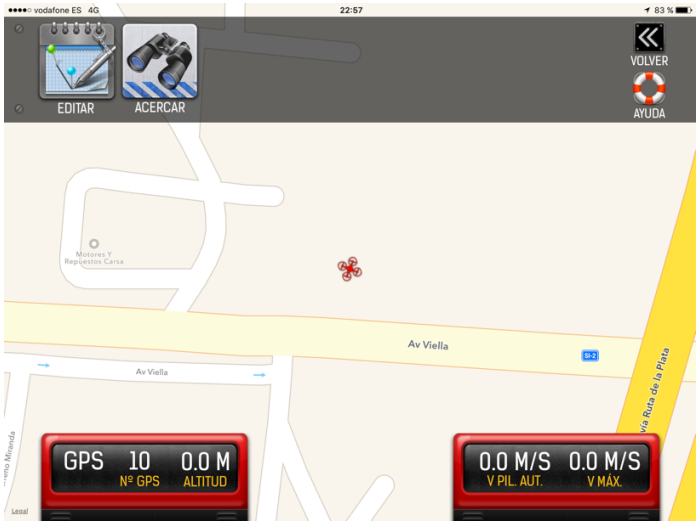


Imagen 68: La interface de ***COMPandGO*** muestra en la pantalla del *iPad* cómo ejecuta la misión el dron.

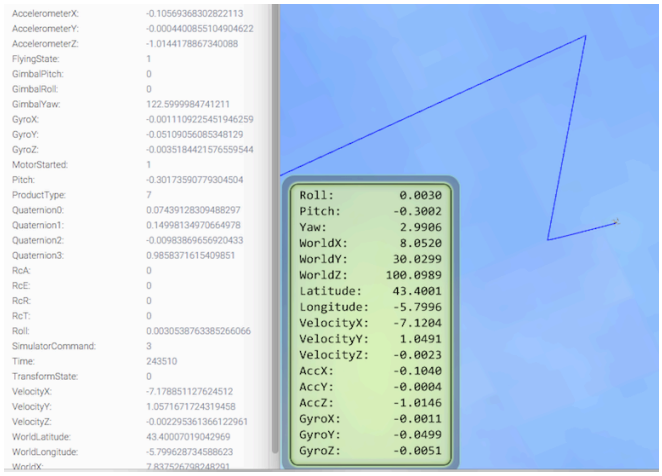


Imagen 69: La interface de *DJI Assistant* muestra la telemetría y el vuelo del dron (trazadora azul)

5.1.4 Pruebas de campo con dron *Phantom 4*

Las pruebas de campo con el dron también están englobadas dentro del tipo de pruebas de caja blanca, ya que en ellas se va a comprobar el principal aspecto que no podíamos testear vía simulador; esto es, se realizan pruebas en las que se accede a la señal capturada por la cámara del dron.

Las pruebas resultan satisfactorias, ya que podemos capturar fotografías y vídeos, que se almacenan en la tarjeta microSD del dron *DJI Phantom4*. De igual manera, esos archivos multimedia capturados se pueden descargar a la aplicación de *Fotos* del *iPad*.

5.1.4.1 Resultados de los vuelos de validación (logs)

Completando las pruebas de campo, se adjuntan las capturas de pantalla de los vídeos de alta resolución capturados con la cámara del dron, acompañados de los datos de telemetría asociados. Como era de esperar, la señal entre el *iPad* donde se ejecuta ***COMPandGO*** y el dron no se ha interrumpido en ningún momento, incluso aunque el dron llega a alcanzar una altura aproximada de 110 m.

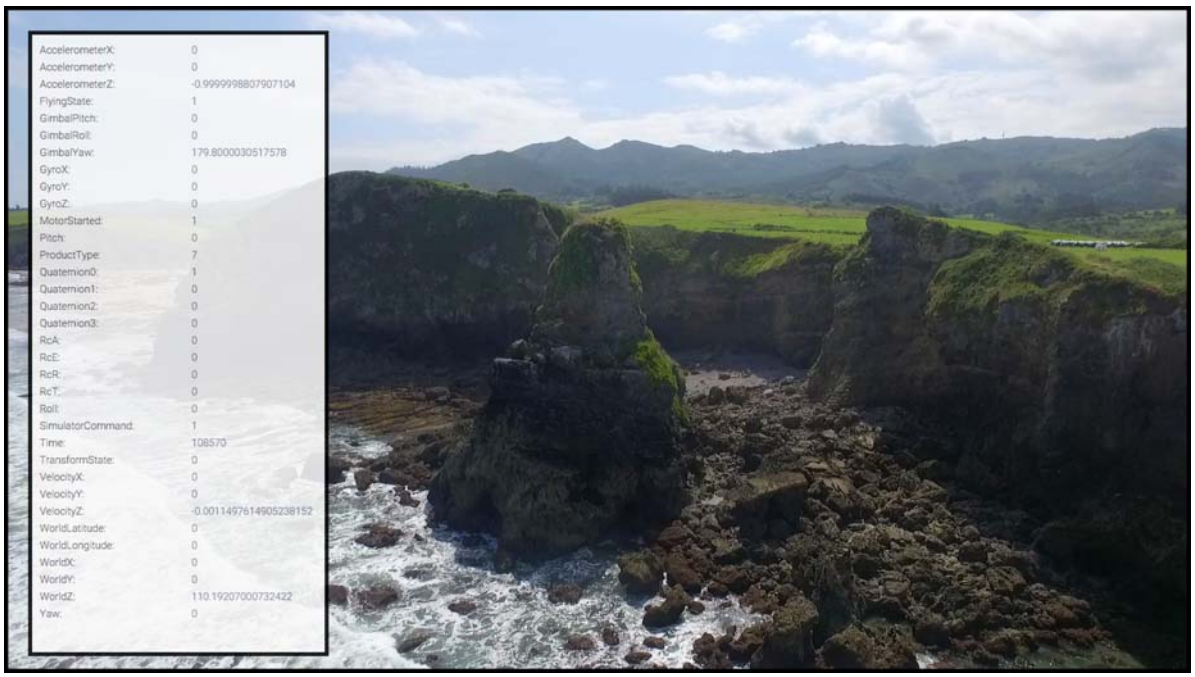


Imagen 70: Vuelo de validación 1.

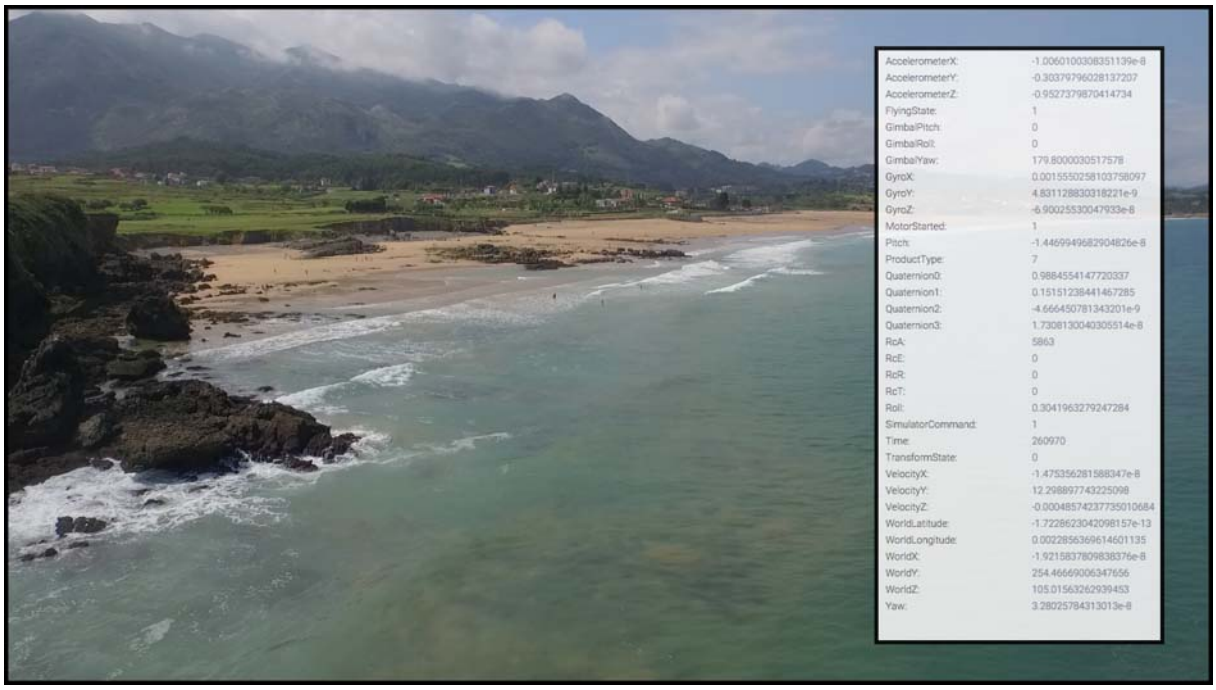


Imagen 71: Vuelo de validación 2.

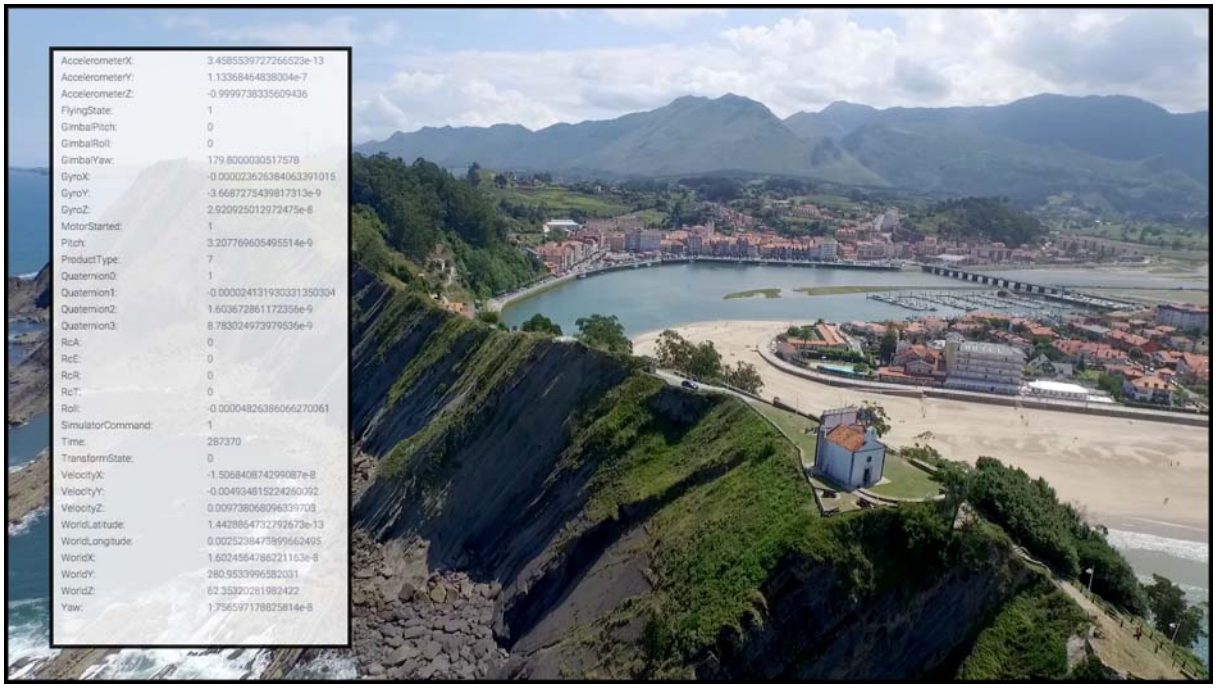


Imagen 72: Vuelo de validación 3.

5.2 Proceso de distribución en *Apple Store*

Este proceso es muy similar al seguido para conseguir instalar nuestras aplicaciones en los dispositivos móviles de *Apple*. Análogamente a aquél, consistirá en acceder a nuestra cuenta de *Apple Developer*, obtener un certificado de distribución (*iOS Distribution Certificate;* en aquella ocasión se trataba del *iOS Development Certificate*, pero el proceso es análogo), creación de un *Distribution Provisioning Profile* e ingreso de los archivos obtenidos en Xcode.

1. Acceso a la cuenta de *Apple Developer*.
2. Obtención de un certificado de distribución. S debe utilizar la aplicación de OSX Acceso a llaveros. Desde el menú *Acceso a llaveros* => *Asistente para certificados* => *Solicitar un certificado de una autoridad de certificación*. Tras facilitar los datos de acceso a la web de *Apple Developer*, podremos descargar el certificado al disco local. Luego bastará con subir este certificado a sección *Certificates* de nuestra cuenta de Apple *Developer*.
3. Creación de un perfil del tipo *Provisioning Profile*. Debe crearse el correspondiente a *iOS App Distribution.* Este perfil sirve para poder utilizar nuestros certificados de distribución para firmar nuestras aplicaciones.
4. Ahora desde la interface de Xcode, primero se va a menú Xcode => Preferences… => Accounts. Ahí se tienen que añadir los mismos datos de la cuenta de usuario de *Apple* que se han empleado a lo largo de todo este proceso.
5. Por último, habrá que acudir al menú Xcode => Windows => Devices => icono de Settings => *Show Provisioning Profiles*… Ahí se deberá añadir el *Provisioning Profile* obtenido anteriormente.

Aquí finaliza el proceso. Si se ha llevado a buen término, el desarrollador ya tendrá su perfil funcionando para poder distribuir sus aplicaciones.

6 CONCLUSIONES

6.1 Análisis de la consecución de objetivos iniciales

Considero que los objetivos generales y específicos planteados tentativamente al inicio del TFG han sido cumplidos. No deja de ser cierto que la última fase del mismo se ha caracterizado por una continua búsqueda de detalles a pulir y mejorar, lo que provoca que no cesen de brotar ideas sobre cómo optimizar el sistema que se está desarrollando. En ocasiones algunas de esas ideas se han reservado para considerarlas a posteriori como futuras mejoras de la App (algunas de ellas se detallarán en el punto siguiente), pero en otras su puesta en práctica supondría reiniciar desde la base todo el TFG (no dejan de alinearse con los típicos pensamientos de "sabiendo lo que sé ahora y si tuviera que empezar de nuevo..."). Supongo que es algo inherente a todo proceso creativo que exija mucha implicación del autor, donde solamente el plazo de entrega fija el punto y final definitivo.

La consecución de los objetivos se ha realizado sobreponiéndose a no pocos obstáculos. Aunque la información de *DJI* es amplia y sus tutoriales de acceso a subsistemas son muy completos y detallados, han sido varias las trabas con las que me he encontrado a lo largo de los últimos meses:

Por un lado el hecho de haber elegido un dron que se ha presentado en público hace tan sólo tres meses (en marzo de 2016) [19] ha complicado muchísimo el desarrollo, ya que la actualización del SDK no contempla muchos de los servicios que en los modelos anteriores *Phantom 2* y *Phantom 3* funcionaban correctamente y que han quedado deprecados para el nuevo modelo *Phantom 4*.

Otro inconveniente de importancia surgió con la imposibilidad de contar con el hardware necesario desde el comienzo del desarrollo. A pesar de que el *DJI Phantom 4* fue presentado el pasado mes de marzo de 2016 [19], problemas de stock impidieron la venta de las primeras unidades en España hasta finales de abril. En concreto el dron que se ha utilizado para la realización de este TFG se recibió a mediados del mes de mayo, lo que supuso un período de incertidumbre más prolongado de lo previsto. Ya se ha comentado en esta memoria la utilidad de los simuladores de *iPad* que ofrece Xcode para comprobar la correcta compilación de la App. Sin embargo, en el caso de ***COMPandGO*** siempre resultó relevante si el código programado permitía el acceso a la cámara del dron, y ese aspecto no

lo aportaba el simulador de *iPad*, para ello era imprescindible conectar el *iPad* al dron. Y recordamos que el desempeño del otro simulador utilizado, el *DJI Assistant 2*, requería también de la conexión al *DJI Phantom 4*. Así las cosas, el tiempo del que se ha dispuesto para realizar las pruebas de la cámara ha sido mucho más limitado de lo previsto. No han sido pocas las ocasiones en las que la App compilaba correctamente pero luego a la hora de acceder a la cámara se obtenía solamente un pantallazo en negro. Y en este punto es importante reseñar la dificultad añadida que supone ser consciente que la App no recibe la señal de la cámara del dron y al mismo tiempo saber que compila correctamente y que para la App Xcode no arroja ningún warning leve ni ningún error grave. Es decir, la búsqueda de los errores supuso en muchos casos un desafío de considerable entidad, ya que se hizo "a ciegas", sin alertas de fallos en los softwares empleados (que siempre suelen ser muy esclarecedoras y otorgan una pista de por dónde empezar a corregir).

Quizás del hecho de haber superado tantos escollos dependa en gran medida el grado de satisfacción final.

Un objetivo cumplido, no sólo inherente al desarrollo de ***COMPandGO*** si no quizás al conjunto de mis estudios en Ingeniería Informática, ha sido el cambiar mi propio paradigma de trabajo como desarrollador. Y en especial en lo relativo al uso de IDE Xcode y Objective-C, lenguaje del que algunos aspectos me han resultado especialmente novedosos, como lo relativo a la interrelación de clases, la herencia y la definición de sus delegados.

6.2 Líneas de trabajo futuras y posibles nuevas funcionalidades

En un futuro mantenimiento perfectivo se plantean una serie de modificaciones sobre este desarrollo que en opinión de este autor completarían el trabajo realizado.

Por un lado, respecto a la documentación generada por ***COMPandGO***, quizás sería conveniente que se pudiera adjuntar un sencillo mapa de la ruta planificada. Dicho desarrollo no exigiría en principio un complejo desarrollo adicional, ya que las clases implementadas (especialmente el Framework de mapas de *Apple*, en combinación con el Framework de *DJI*) nos facilita la información precisa de la situación de la aeronave en cada momento. Así mismo ya se han implementado funciones para adjuntar un archivo al correo electrónico con el formulario generado y esas mismas funciones se podrían reutilizar para el envío de dicha información.

Sería también interesante poder establecer un sistema de almacenamiento de los perfiles de usuario, de tal manera que se pudieran cargar automáticamente al sistema aquellos datos que pudieran ser comunes a la mayor parte de los vuelos de un usuario, tales como los datos específicos de la aeronave o sus datos personales, ya que se entiende que una buena parte de los usuarios del sistema no poseen más de un dron y entre los diferentes planes de vuelo operacionales preliminares las diferencias suelen radicar en las características geográficas, técnicas, meteorológicas o sencillamente de visibilidad de dicho vuelo.

Otra funcionalidad que sería interesante añadir a la aplicación es la capacidad para capturar imágenes panorámicas de 360º (Pan 360º). El principal problema es que la cámara del dron no tiene total libertad de rotación en torno al eje Z de la aeronave, si no que es muy limitada.

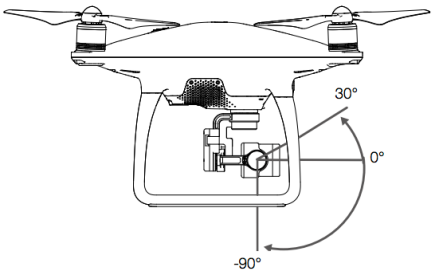


Imagen 73: La cámara del *DJI Phantom 4* rota 120º alrededor del eje Y (desde +30º a -90º).

Entrando en detalles, la cámara del dron sólo tiene un ángulo de rotación de 120º alrededor del eje Y (desde +30º a -90º) de la aeronave y otro de 30º (desde -15º a +15º) alrededor del eje Z [20]. ¿Por qué no tiene más en este último caso? Porque en un *DJI Phantom 4* una rotación mayor haría que en las imágenes capturadas se viera la estructura (soportes) del dron (situación que no se produce en otros modelos de drones con un diseño distinto en los que la cámara está más expuesta a los golpes accidentales). De modo que para hacer una Pan 360º en nuestro caso de un *DJI Phantom 4* no podemos optar por una solución que incluya una sola imagen en una única toma mediante la rotación de 360º la cámara, si no que tendríamos que pensar en una solución que pasara por rotar toda la aeronave 360º. El *DJI Phantom 4* tendría que realizar pues, por ejemplo, cuatro rotaciones de 90º, capturar cuatro imágenes y componerlas en una sola imagen panorámica. Una solución que por el momento se ha escapado de nuestros objetivos para este TFG.

Otro aspecto de interés que incrementaría la utilidad de nuestra App sería que fuera capaz de acceder a los datos de telemetría que almacena el dron, permitiera verlos en pantalla y almacenarlos. Abundando es este aspecto, también sería de interés que esos datos se pudieran cotejar con otros. Por ejemplo, sería útil comparar ciertas magnitudes de un vuelo

realizado con los valores máximos y mínimos que establece la ley. Así, se podría conocer el dato de la altura máxima alcanzada en una misión de vuelo y averiguar si ese valor se ha salido del rango de alturas que permite la normativa vigente (120 m de altura) [17].

También queda pendiente un trabajo de "adelgazamiento" de la interface gráfica, pues si bien en la actualidad todo el arte se ha trabajado a altísima calidad, en una buena parte de los casos se podrían mejorar los ratios de compresión de los recursos gráficos, mediante un cambio de formato en algunos casos y mediante el reescalado o cambio de compresiones en otro. De esta manera, el tamaño de la interface gráfica de usuario sería menor y en consecuencia el tamaño del archivo (producto) final de la App también disminuiría. Esta consideración también es aplicable al tratamiento de los sonidos.

En un proceso de depurado, se podrían definir una jerarquía de clases y métodos delegados que evitaran, en algunos casos, la utilización de código con un alto grado de similitud en los distintos subsistemas implementados. Así por ejemplo, mediante el programa de desarrollo de *DJI*, y previo acuerdo de desarrollo formal, se puede evitar una duplicidad de API Keys en los subsistemas de la Cámara y Galería, algo que con el programa de desarrollo básico o estándar no se puede hacer, ya que una buena parte son "requeridos".

Queda pues pendiente una depuración definitiva del sistema, que como todo desarrollo es susceptible de ser mejorado desde el mismo momento de su implantación y aceptación.

6.3 Reflexión personal

Considero que el fin ulterior de intentar llevar a buen puerto este Trabajo de Fin De Grado es conseguir desarrollar de manera autónoma e integral un proyecto informático de cierta envergadura, que sirva como colofón a la adaptación de mis estudios de Ingeniería Informática a las nuevas titulaciones de Grado.

La realización de este TFG ha supuesto un enorme reto personal, dado que desde hace más de 20 años mi trayectoria profesional como realizador de televisión me ha desvinculado del mundo de la programación. Después de tantos años alejado del campo donde desempeñé mis estudios universitarios, volver a establecer una relación estrecha con el ámbito académico suponía un desafío muy importante para mí. Y no sólo por el tiempo transcurrido, si no por el profundo cambio que han experimentado tanto las metodologías de la enseñanza como la materia objeto del estudio en estos 20 años. Huelga comentar lo

terriblemente cambiante que es la Ingeniería Informática (lenguajes de programación, entornos de trabajo, dispositivos, etc.), ya que su continua evolución se nota casi de año en año. Extrapolar esos cambios casi vertiginosos desde el "curso a curso" hasta los 20 años vista… No hay mucho que añadir al respecto.

También ha sido interesante en esta nueva etapa de formación, especialmente y como máximo exponente en el desarrollo de este TFG, la perspectiva vital de aproximación al estudio. He encontrado que el nivel de satisfacción personal es inversamente proporcional a la, digamos, urgencia en cumplir con una etapa que casi viene marcada desde la infancia. La sucesión de acontecimientos que son los estudios primarios, los superiores, los universitarios y la entrada urgente en el mercado laboral prácticamente no otorgan tiempo para reposar y madurar lo conseguido. Cuando se acomete la decisión de iniciar un proceso de formación con el objetivo final de aprender una materia sin urgencias, con el firme propósito de, sobre todo, saber y ampliar conocimientos, entonces la satisfacción es mucho mayor.

Mi idea de buscar un desarrollo de este tipo surgió de la necesidad detectada en mi entorno laboral, como realizador de una cadena de televisión pública, al observar que había un conflicto de intereses entre la necesidad de cumplir con los requisitos exigidos por AESA para poder realizar grabaciones dentro del espacio aéreo español y la de improvisar y actuar con celeridad a la hora de esas filmaciones, especialmente en sucesos de actualidad para informativos (por ejemplo, un incendio o una accidente en una autopista). Después de tantas y tantas horas y jornadas de trabajo, queda satisfecho el reto intelectual de afrontar un desarrollo integral de un proyecto como ***COMPandGO***, surgido a raíz de esa necesidad que apareció en mi vida y al que en primera instancia no podía imaginar que fuese capaz de encontrarle (o al menos plantearle) una posible solución.

¿El resumen? El objetivo final fue, es y será aprender, saber. Sentirse capaz de poder alcanzar algo que conlleve un esfuerzo. Muchas de las mayores satisfacciones de mi vida también las he obtenido así y me siento orgulloso de ello. No es poco.

6.4 Publicación en repositorio de código libre tipo Git

Por último, y tal como se ha mencionado en anteriores apartados, se procede al almacenamiento del código fuente de la App ***COMPandGO*** en un repositorio de código libre tipo Git. El área de descarga alojada en el sitio web de GitHub se encuentra en el siguiente enlace:

http://github.com/josebrana/1-COMPandGO-DESARROLLO

Así mismo, en la landing page del proyecto se alojarán una serie de vídeos introductorios y explicativos del funcionamiento de ***COMPandGO***. Concretamente los vídeos estarán disponibles en la sección VÍDEOS de esa web. La dirección exacta se encuentra en el enlace indicado a continuación:

http://josebrana.com/compandgo/videos/

BIBLIOGRAFÍA / REFERENCIAS

[1] Guillen, Paulo (Diseñador gráfico) (2013) *Variations 3*. [Package of icons, Serie Premium] Disponible en http://guillendesign.deviantart.com/art/VARIATIONS-3-261509672

[2] Berganza, L. (Formador). (2014). *Objective-C*. [Curso formativo] Disponible en https://www.vídeo2brain.com/es/cursos/objective-c

[3] Berganza, L. (Formador). (2014). *Objective-C · Aprende a usar el lenguaje de desarrollo de iOS*. [Curso formativo] Disponible en https://www.vídeo2brain.com/es/cursos/objective-c

[4] Berganza, L. (Formador). (2014). *Xcode: Apps para iPhone y iPad*. [Curso formativo] Disponible en https://www.vídeo2brain.com/es/cursos/xcode-Apps-para-iphone-y-ipad

[5] González, M. (Formador). (2010). *Desarrollo de aplicaciones para iPhone y Objective-C*. [Curso formativo] Disponible en https://www.vídeo2brain.com/es/cursos/ desarrollo-de-aplicaciones-para-iphone-y-objective-c

[6] Solís, C. (Formador). (2016). *GitHub para programadores*. [Curso formativo] Disponible en https://www.vídeo2brain.com/es/cursos/ github-para-programadores

[7] DJI Forum (2016, 8 de junio). Areas not to fly in due to GPS outages: dates and maps included. Mensaje publicado en http://forum.dji.com/forum.php?mod=viewthread&tid=54828&extra=page%3D1%26filter%3Dtypeid%26typeid%3D372%26orderby%3Ddateline%26typeid%3D372%26orderby%3Ddateline

[8] DJI Developer Forum (2016, 13 de enero). [iOS] Custom Mission – questions. Mensaje publicado en http://forum.dev.dji.com/forum.php?mod=viewthread&tid=32036&extra=page%3D1%26filter%3Dtypeid%26typeid%3D306

[9] Fly School Air Academy (s.f.) (2016, 13 de febrero). *Marco regulatorio de la operación de drones en Europa*. Mensaje publicado en http://escuela.flyschool.es/mod/forum/discuss.php?d=358

[10] Jiménez, M. (2015). *El auge de los drones trae nuevos negocios*.

Recuperado el 2 de abril de 2016 de http://cincodias.com/cincodias/2015/04/02/tecnologia/1427998183_788282.html

[11] Rohan, M. (2016). *Small Drones Market worth 10.04 Billion USD by 2020*. Recuperado el 3 de abril de 2016 de http://www.marketsandmarkets.com/PressReleases/small-uav.asp

[12] Confidente, E. (2016). *La nueva regulación española sobre drones vuela (muy) bajo*. Recuperado el 22 de abril de 2016 de http://blogs.elconfidencial.com/espana/el-confidente/2016-04-22/drones-espana-regulacion-aesa-easa_1187211/

[13] Isla visual (s.f.) (2012). *Diferencias entre SCRUM y XP.* Recuperado el 23 de abril de 2016 de http://www.islavisual.com/articulos/desarrollo_web/diferencias-entre-scrum-y-xp.php

[14] Costello, S. (2016). *How many Apps are in the App Store?.* Recuperado el 21 de abril de 2016 de http://ipod.about.com/od/iphonesoftwareterms/qt/Apps-in-App-store.htm

[15] Rodríguez, L. *Estadísticas que hacen de WordPress el CMS más popular del mundo*. Recuperado el 6 de junio de 2016 de https://www.40defiebre.com/estadisticas-wordpress/

[16] Apple Inc. (s.f.) (2016) . Maps for Developers. Disponible en https://developer.apple.com/maps/

[17] AESA (s.f.). *¿Qué requisitos tiene que cumplir un piloto para poder volar un dron?* Recuperado el 19 de abril de de 2015 de http://www.seguridadaerea.gob.es/lang_castellano/cias_empresas/trabajos/rpas/faq/default.aspx#05

[18] AESA (s.f.). *Registro de declaración responsable de operador de aeronaves RPA's*. Recuperado el 5 de julio de 2016 de http://www.seguridadaerea.gob.es/media/4305572/listado_operadores.pdf

[19] Phantom-four.com (s.f.) (2015). *DJI Phantom 4 launch*. Recuperado el 2 de abril de 2016 de http://www.phantom-four.com/dji-phantom-4-launch-2/

[20] Dà-Jiāng Innovations Science and Technology Co., Ltd. (2016). *Phantom 4. Especificaciones*. Recuperado el 2 de abril de 2016 de

https://www.dji.com/es/product/phantom-4/info

[21] Dà-Jiāng Innovations Science and Technology Co., Ltd. (s.f.) (2016). *DJI Mobile SDK for iOS*. Disponible en http://developer.dji.com/mobile-sdk/

[22] Dà-Jiāng Innovations Science and Technology Co., Ltd. (s.f.) (2016). *DJI Developer*. Disponible en https://developer.dji.com/

[23] Dà-Jiāng Innovations Science and Technology Co., Ltd. (s.f.) (2016). *DJI Mobile SDK for iOS*. Disponible en https://github.com/dji-sdk/Mobile-SDK-iOS

[24] Dà-Jiāng Innovations Science and Technology Co., Ltd. (s.f.) (2016). *DJI Bridge App Tutorial*. Recuperado el 24 de junio de 2016 de https://developer.dji.com/mobile-sdk/documentation/ios-tutorials/BridgeAppDemo.html

[25] Apple Inc. (s.f.) (2014) *iPad Air 2 specs*. Recuperado el 2 de abril de 2016 de http://www.apple.com/es/ipad-air-2/specs/

[26] Apple Inc. (s.f.) (2016) *Apple Developer Program*. Disponible en https://developer.apple.com/programs/

[27] *Balsamiq Mockups Website*. Disponible en https://balsamiq.com/

[28] *Namecheckr Website*. Disponible en https://www.namecheckr.com/

[29] *Soundbible Website.* Disponible en http://soundbible.com/suggest.php?q=button+sounds&x=0&y=0

[30] *Zamzar Website*. Disponible en http://www.zamzar.com/

[31] *Delivr Website*. Disponible en https://delivr.com/

[32] *DJI Assistant 2 v1.0.4*. Disponible en http://www.dji.com/es/product/phantom-4/info#downloads

[33] *Draw.io Website*. Disponible en https://www.draw.io/