

UNIVERSIDAD POLITÉCNICA DE MADRID
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA
AERONÁUTICA Y DEL ESPACIO
GRADO EN INGENIERÍA AEROESPACIAL

TRABAJO FIN DE GRADO

**Diseño de una Interfaz Web para el Prototipado de un Dron
con Carga de Pago Configurable en CAD**

AUTOR: César Eduardo JIMÉNEZ GÁMEZ

ESPECIALIDAD: Vehículos Aeroespaciales (VA)

COTUTOR: Luis IZQUIERDO MESA

TUTOR DEL TRABAJO: Sergio ÁVILA SÁNCHEZ

Julio de 2022

Agradecimientos

*A mi incansable compañera de trabajo, gran amiga e increíble **compañera de vida**. Sin ti, Virgi, trabajare en este proyecto no habría sido lo mismo. Tu luz me ha iluminado a lo largo de este camino en muchos momentos difíciles. Gracias por este fantástico año lleno de aprendizajes, experiencias, crecimiento, alegrías y amor junto a ti. Te amo mucho muchísimo mi vida. ¡Yo más!*

*A mi **madre** por enseñarme lo que es el amor y por regalarme su apoyo incondicional a lo largo de toda mi vida. Gracias por regalarme tu vida, por ser la madre que siempre he necesitado. Y, sobre todo, gracias por tu valentía, por haber sacado a esta familia adelante incluso en los momentos más difíciles. Brindo por mi madre, la mejor que esta vida me pudo dar.*

*A mis **abuelos** por siempre estar a mi lado apoyándome, cuidándome, queriéndome y preocupándose por mi bienestar. Gracias por estar siempre a mi lado. Os quiero.*

*A mi **hermanita** por hacerme reír y regalarme su compañía desde que tengo uso de razón, gracias. Gracias por siempre haber cuidado de mí, por siempre haber jugado conmigo desde que era un niño, por regalarme tu apoyo y por quererme incondicionalmente. Eres increíble loquita.*

*A mi **padre** por enseñarme sobre el apasionante mundo de la ingeniería, los negocios y a pensar de una manera diferente. Aunque siempre hayas estado lejos, te quiero.*

*Al mejor amigo del hombre, mi perro. Siempre te llevaré en mi corazón **Blacky**.*

*Doy gracias a la vida por hacer aparecer en mi camino a los mejores **amigos** que esta vida me ha brindado jamás, amigos que han sido, son y serán para toda la vida. Gracias por todo lo reido, por todo lo aprendido, por todo lo sufrido y, sobre todo, por todo lo vivido y que queda por vivir. Gracias Iván, Rafa, Diego, Dani y Pablo.*

Y, por último, pero no por ello menos importante. A todas las personas que de una u otra forma me han ayudado, apoyado, inspirado y acompañado a lo largo de estos últimos años de mi vida. Doy también las gracias, en especial, a mis alumnos y a mis mentores por inspirarme y ayudarme a crecer como persona.

*A todas las personas mencionadas: **OSAMO**.*

Gracias a mis tutores, Sergio Ávila y Luis Izquierdo, por brindarme la oportunidad de realizar este TFG.

*César Eduardo Jiménez Gámez
Alcalá de Henares, 30 de agosto de 2022*

Índice General

Introducción	9
1.1. Motivación	9
1.2. Resumen y Objetivos del Trabajo.....	12
1.3. Estructuración del Documento.....	13
Estado del Arte	17
2.1. Introducción.....	17
2.2. Caracterización de un VTOL	17
2.3. Estudios Relacionados	24
2.4. Necesidades de la Industria de Drones.....	25
Metodología	28
3.1. Introducción.....	28
3.2. Trabajo realizado sobre el VTOL.....	29
3.2.1. Toma de Medidas	30
3.2.2. Creación del Modelo 3D	34
3.3. Análisis Funcional de la Aplicación Web	36
3.3.1. El Día de la Prueba de Vuelo	36
3.3.2. Análisis Funcional.....	38
3.4. Diseño del HMI de la Aplicación Web.....	40
3.5. Testeo y Validación del Trabajo	41
Implementación	43
4.1. Introducción.....	43
4.2. Diseño Inicial del Dron.....	43
4.2.1. Planos obtenidos.....	44
4.2.2. Realización de Sketches Inmersivos.....	46
4.2.3. Diseño del Fuselaje	50
4.2.4. Diseño de la Tapa 1	62
4.2.5. Diseño de la Tapa 2	65
4.2.6. Diseño de la Tapa 3	67
4.2.7. Diseño de la Tapa 4	69
4.2.8. Diseño de la Tapa Inferior	71
4.2.9. Diseño de los Patines del Tren de Aterrizaje	72
4.2.10. Diseño de los Tubos Estructurales	74
4.2.11. Montaje del dron en Assembly Design.....	75

4.3. Toma de Datos en el Trabajo de Campo	76
4.3.1. Outputs solicitados por el “cliente”	76
4.3.2. Inputs necesarios	76
4.4. Diseño del HMI y Redacción del Análisis Funcional	79
4.4.1. Diseño del HMI	79
4.4.2. Códigos de HTML5, JavaScript y CSS	81
4.4.2.1. Secciones (tabs)	81
4.4.2.2. Acordeones	83
4.4.2.3. Biblioteca jQuery	84
4.4.2.4. Envío de datos con JSON	85
4.4.2.5. Envío de datos al servidor local Flask en Python	88
4.4.2.6. Elemento HTML <iframe>	89
4.4.2.7. Biblioteca three.js	91
4.4.2.8. Escritura de resultados finales	99
4.4.3. Presentación del Flujograma	101
Testeo y Validación	107
5.1. Validación del Modelo del Dron en Catia	107
5.2. Testeo de la Interfaz Web	109
5.3. Conclusiones Generales de la Validación	112
Conclusiones y Futuras Líneas de Investigación	114
6.1. ¿Se han Conseguido Alcanzar los Objetivos Propuestos Inicialmente?	114
6.1.1. Diseño del Modelo 3D del Dron (Objetivo 1)	115
6.1.2. Desarrollo de la Página Web (Objetivo 2)	117
6.1.3. Inputs/Outputs para Calcular el Centro de Gravedad (Objetivo 3)	119
6.1.4. Aprendizajes (Objetivo 4)	120
6.2. Futuras Líneas de Investigación y de Desarrollo	120
Bibliografía	126
7.1. Sobre el uso de “three.js” en JavaScript	126
7.2. Fighter VTOL Fixed-wing Aircraft	126
7.3. JavaScript, HTML5 y CSS	127
7.4. JSON y Servidor Local Flask	128
7.5. Documentos relativos al Estado del Arte	129
7.6. Tutoriales de Catia V5	129

Nomenclatura

CDG	Centro de Gravedad
HMI	Human Machine Interface
DAVE	Departamento de Aeronaves y Vehículos Espaciales
OEW	Operational Empty Weight
MPL	Maximum Payload
VTOL	Vertical Take-Off and Landing
eVTOL	VTOL eléctrico
ODM	Open Drone Map
GUI	Graphical User Interface

Capítulo 1

Introducción

1.1. Motivación

La realización del proyecto descrito en las próximas páginas de este documento tuvo su comienzo en febrero de 2022, en la Escuela Técnica Superior de Ingeniería Aeronáutica y del Espacio (ETSIAE).

Durante ese mes, mi compañera de trabajo Virginia Bueno Gómez y yo, César, nos encontrábamos trabajando en las prácticas del DAVE. Ambos teníamos la idea de hacer de nuestro TFG un proyecto ambicioso que juntase algunas de las disciplinas que más nos habían llamado la atención a lo largo de los últimos años de carrera: programación, automatización de procesos, aerodinámica, diseño e impresión 3D aplicados a la ingeniería aeroespacial. Además, con la idea de poner en práctica otros conocimientos adquiridos a lo largo de la carrera en el ámbito de la “Gestión de Empresas y Proyectos” y de los “Sistemas de Producción Aeroespacial”, buscábamos desarrollar un proyecto que pudiera ser atractivo para el mercado.

Algunas de nuestras primeras ideas fueron encaminadas hacia la creación y comercialización de un nuevo modelo de dron recreativo: un coche volador, un ala volante, etc. También surgió la idea de crear una página web titulada “Diseña tu dron”, donde los usuarios pudieran diseñar su propio dron de una forma simple y sencilla en función de las actuaciones de vuelo deseadas.

Con estas ideas en mente, comenzamos a asistir como oyentes a clases de “Python” y clases de “HTML y JavaScript” [45] [47] [49]. También asistimos a una asignatura llamada “Aeroingenia”, destinada a enseñar a sus alumnos cómo diseñar y fabricar un dron capaz de cumplir una determinada misión. Las siguientes imágenes (*figura 1.1*) describen algunos de los aprendizajes adquiridos durante la asistencia a estas asignaturas, entre ellos: lecciones sobre creatividad aplicada al diseño (clase de Aeroingenia) y lecciones sobre programación en HTML [45], JavaScript [22][45] y CSS [45] (ejemplo de calculadora web desarrollada con

estos lenguajes).

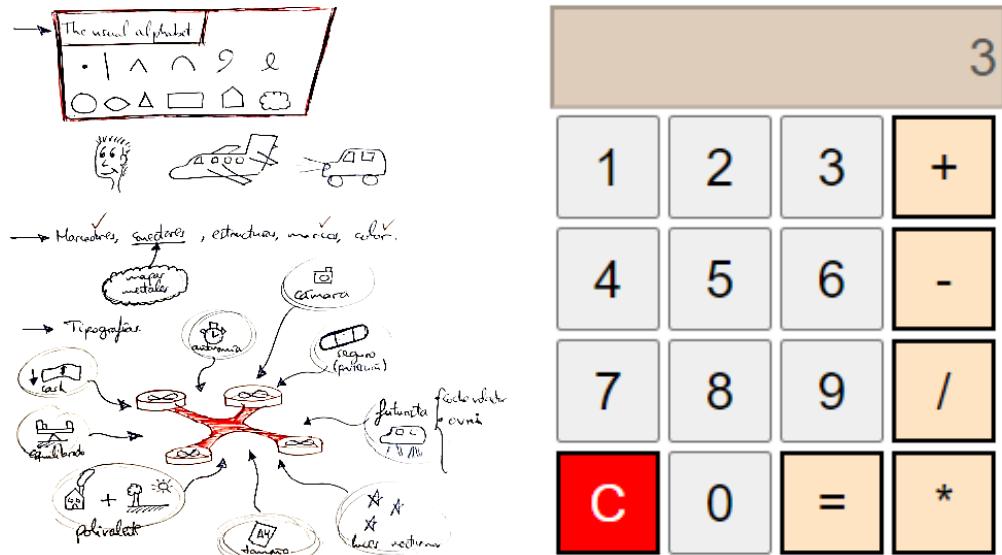


Figura 1.1: Ejemplos de aprendizajes adquiridos sobre diseño y creatividad en Aeroingeniería, a la izquierda, y sobre programación en HTML y JavaScript, a la derecha.

Como dice la frase: *"La suerte es lo que sucede cuando la preparación se encuentra con la oportunidad."*, Voltaire, sucedió lo inesperado. Tras una de las clases de “HTML y JavaScript” mantuvimos una conversación con Luis Izquierdo Mesa, nuestro profesor de la asignatura, actual tutor de este TFG y actual “cliente” para el que se ha desarrollado este proyecto. Nos comentó que tanto él como sus compañeros de trabajo estaban haciendo pruebas con un VTOL de la empresa Make Fly Easy: *“Fighter VTOL Fixed-wing Aircraft”* (figura 1.2) [11][12][13]. El fin principal que tenían era el de investigar y desarrollar tecnologías que permitieran realizar misiones de vigilancia aérea.





Figura 1.2: Imágenes del dron “Fighter VTOL Fixed-wing Aircraft”. La primera, tomada en el DAVE, y la segunda, una foto promocional de la aeronave.

Teniendo en cuenta nuestros intereses, Luis nos contó sobre algunas de las necesidades que tenían él y su equipo. Las ideas que nos planteó nos resultaron atractivas, más teniendo en cuenta que la investigación planteada podía tratarse de un “Océano Azul” pues no existía una tecnología similar conocida que permitiera hacer lo solicitado por el cliente. Fue en ese preciso momento cuando decidimos aceptar su propuesta y desarrollar este proyecto.

Tras varias reuniones, se logró elaborar un listado de objetivos concretos que al cliente le interesaba alcanzar gracias a la investigación plasmada en estas páginas. Estos se pueden separar en diferentes grupos en función de las disciplinas involucradas:

1. Primero, se muestran los objetivos relacionados con el **diseño, aerodinámica y sistemas de producción aeroespacial**:
 - a. Diseño de un modelo inicial en Catia basado en un dron real, más concretamente del “*Fighter VTOL Fixed-wing Aircraft*”.
 - b. Parametrización del modelo obtenido.
 - i. Parametrización del fuselaje para acoplar diferentes cargas de pago en el lugar más apropiado.
 - ii. Diseño de un “gimbal” parametrizable en función de la cámara elegida.
 - iii. Parametrización de diferentes configuraciones con la hélice tractora delantera o trasera.
 - iv. Parametrización del perfil alar en función de las actuaciones requeridas por el usuario para cada vuelo.
 - c. Desarrollo de un sistema de producción en serie del modelo parametrizable, empleando para ello métodos como la impresión 3D, moldeo por inyección, etc.
2. A continuación, se describen los fines relacionados con el campo de la **programación, la automatización de procesos y el diseño de producto**:
 - a. Desarrollo de una página web de alta usabilidad que permitiera a los usuarios visualizar el modelo obtenido en Catia desde todos sus ángulos, pudiendo: ocultar, mostrar y señalar diferentes partes del mismo.

- b. Debido a la necesidad de emplear diferentes cámaras para las misiones de vigilancia del dron, se busca lograr el cálculo automatizado de la posición del CDG del dron cargado, en función de la distribución de carga introducida a través de la interfaz web mencionada. También se desea mostrar los resultados obtenidos al usuario.
 - c. Creación de un servidor local capaz de enlazar esta interfaz de usuario con un código en Python que pudiera de automatizar procesos en Catia V5, proporcionando los resultados requeridos y devolviéndolos a la interfaz web.
 - d. Desarrollo del código en Python necesario para alcanzar los objetivos anteriores.
3. **Entender la forma de aplicar e integrar** frameworks, bibliotecas y tecnologías ya existentes con el fin de alcanzar los objetivos propuestos. Además, se busca construir nuevas tecnologías capaces de realizar las tareas ya citadas, poco desarrolladas anteriormente.

1.2. Resumen y Objetivos del Trabajo

La ya citada lista de objetivos del cliente resultó ser bastante larga y densa. Por este motivo, de cara a la realización de este proyecto, tanto Virginia como yo decidimos centrar nuestros esfuerzos en la consecución de los objetivos más novedosos de la actualidad, poco desarrollados o investigados en la industria de drones actual, por lo menos, públicamente. De forma más detallada, decidimos enfocar el proyecto en los puntos **1.a., 2.a., 2.b., 2.c., 2.d., 3.** También se decidió añadir un apartado que permitiera al usuario hacer cálculos preliminares de la autonomía y el alcance del dron durante su misión, en función de su carga útil y de las baterías empleadas.

Por último, antes de entrar en materia, me gustaría hacer una breve puntualización sobre este trabajo. El proyecto en el que se basa este TFG ha sido desarrollado por dos personas: Virginia Bueno Gómez (mi compañera) y César Eduardo Jiménez Gámez (yo). Cada uno de nosotros, durante la fase de desarrollo se ha centrado más concretamente en una de las dos partes de este proyecto, dividiendo al mismo en Backend y Frontend. De esta forma, yo me he especializado en el Frontend de la página web, mientras que Virginia lo ha hecho en el Backend de la misma. Así pues, de un mismo proyecto de fin de grado, han emanado dos Trabajos de Fin de Grado complementarios:

- El TFG que usted está leyendo en este preciso momento, cuyo principal contenido trata el Frontend del proyecto (**objetivos 2.a., 2.b. y 3 principalmente**).
- El TFG desarrollado por Virginia, que trata el Backend del mismo proyecto (**objetivos 2.c., 2.d. y 3 principalmente**). Cabe destacar que este TFG será presentado el próximo curso, en 2023.

Si bien es cierto, ambos hablaremos en nuestros TFG sobre el objetivo **1.a.**, dividiendo este en 2 partes. Mientras yo me centro más en explicar cómo diseñar el fuselaje del dron y las

partes que lo componen, Virginia se centrará más en explicar cómo diseñar el ala, la cola y otros componentes necesarios.

Por último, deseo señalar que en el siguiente enlace de GitHub https://github.com/Cesitar22/TFG_Cesar_Eduardo_Jimenez_Gamez se pueden encontrar todos los códigos explicados y empleados a lo largo de la realización de este TFG.

En el siguiente punto aclararemos cuáles son los contenidos de este trabajo.

1.3. Estructuración del Documento

En el [Capítulo 2](#), descubriremos cuál es el punto de desarrollo actual de la industria de drones, en especial de las tecnologías VTOL y haremos hincapié en sus grandes ventajas frente a otras tipologías de drones. Añadido a esto, hablaremos sobre el estado del arte de las tecnologías destinadas al cálculo y análisis del CDG en drones, entendiendo cuáles son los problemas derivados de un mal centrado del dron, cuáles son los estudios e investigaciones que se han hecho al respecto (véase un ejemplo de estas investigaciones en la *figura 1.3*) y por qué aparece hoy en día la necesidad de disponer de sistemas que permitan parametrizar la distribución de carga en drones.

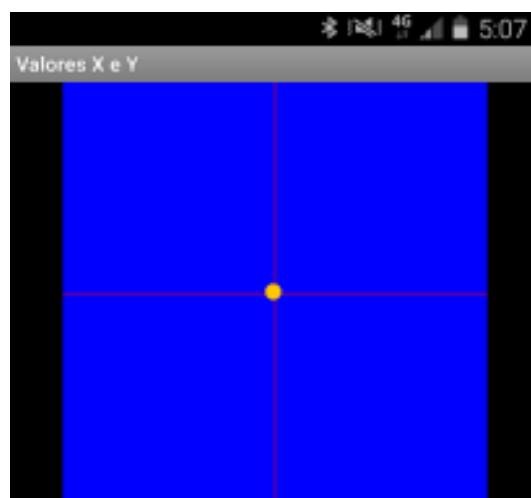


Figura 1.3: Imágenes de un estudio relativo al cálculo automatizado del CDG de un dron. Realizado por [Diego Sebastian Jiménez Gallegos](#) en la UAB.

En el **Capítulo 3**, explicaremos las metodologías seguidas para cumplir con éxito las expectativas deseadas de este proyecto y se hablará también acerca de las posibles mejoras de los procedimientos aplicados. Primero, trataremos los aspectos referentes a la realización del modelo en Catia del dron “*Fighter VTOL Fixed-wing Aircraft*” (en la *figura 1.4* se nos puede ver a Virginia y a mí realizando mediciones sobre este dron), desarrollado para nuestro “cliente”. Después, comprenderemos cuál ha sido el trabajo de campo necesario para definir completamente las necesidades requeridas por nuestro cliente. Tras esto, haremos algunos comentarios acerca del análisis funcional de la aplicación desarrollada, y hablaremos sobre el diseño del HMI. Al finalizar la lectura de este capítulo, quedarán entendidos los razonamientos que hay detrás de cada decisión en este proyecto, entendiéndose además la forma en que los usuarios deberán interactuar con la página web para obtener los resultados buscados (centrado del dron y sugerencias acerca del mismo) en función de sus datos introducidos (distribución de carga de pago). Por último, pero no menos importante, explicaremos cuales han sido las pruebas realizadas para asegurar el correcto funcionamiento de la página web desarrollada. También explicaremos el testeo hecho sobre el modelo del VTOL desarrollado en Catia V5.



Figura 1.4: Imágenes de Virginia y César tomando medidas sobre el VTOL modelo.

En el **Capítulo 4**, mostraremos cuál ha sido la implementación de todos los métodos explicados en el capítulo 3. Detallaremos paso por paso el proceso (toma de medidas, realización de planos, trabajo de campo y creación del modelo 3D) seguido para obtener el modelo 3D en Catia principalmente del fuselaje (y otras piezas relacionadas) del dron de nuestro cliente. Se muestra en la *figura 1.5* el resultado de este trabajo, es decir, el modelo 3D final.

Además, estudiaremos en profundidad cual es la forma de “pensar” y de “operar” de los códigos de programación empleados para dotar a la página web de la máxima usabilidad posible. Mostraremos y explicaremos en detalle los códigos de programación (y su

interconexión) empleados para alcanzar los objetivos buscados respecto al Frontend del sistema. Hablaremos del papel de los grandes protagonistas que dan imagen y funcionalidad a la página web desarrollada: HTML, JavaScript, CSS y el servidor local programado en Python. Por último, pasaremos a hablar más en detalle del framework y las bibliotecas más importantes que se han utilizado, como son: Flask, three.js y jQuery.

De esta forma, quedará completamente definido el diagrama de flujo (o fluograma) de la aplicación web, tras lo cual analizaremos, también en profundidad, el diseño del HMI adoptado con el fin de alcanzar la máxima usabilidad posible. Se puede ver un ejemplo del HMI resultante en la *figura 1.5* también.

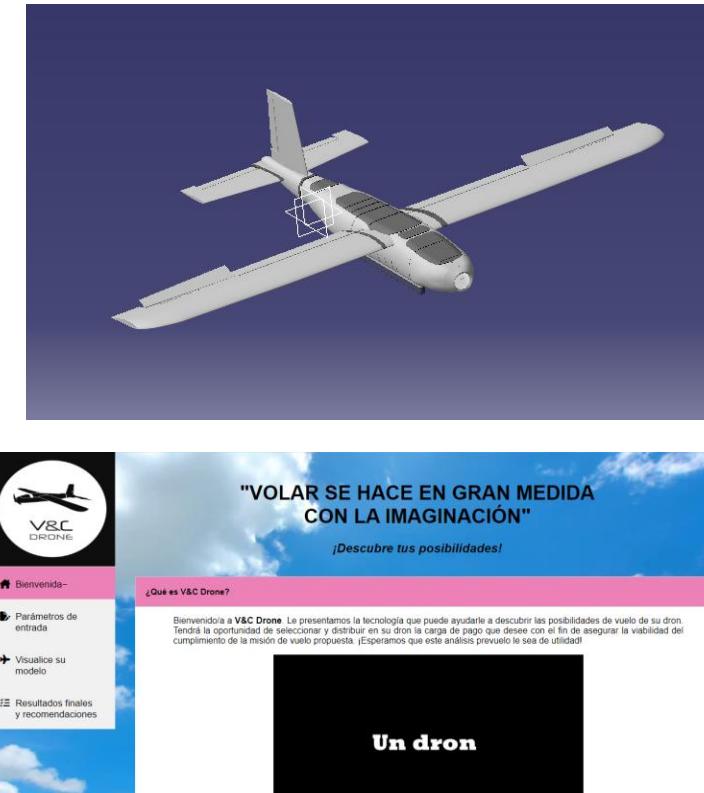


Figura 1.5: Imágenes del modelo 3D del dron completado y de la primera sección de la página web desarrollada.

En el [Capítulo 5](#), testearemos y validaremos el modelo en Catia mediante la impresión 3D de una de sus piezas (ver *figura 1.6*), y el sistema usuario-servidor a través de un ejemplo de uso real. Debatiremos sobre los aspectos positivos y negativos del proyecto desarrollado, dando entrada al [Capítulo 6](#), en el cual plantearemos las conclusiones de esta investigación y haremos hincapié tanto en los objetivos que se han logrado alcanzar como en los que no se han conseguido completar, abriendo de esta forma nuevas líneas de investigación (se muestra un ejemplo en la *figura 1.6*) para continuar con este innovador y ambicioso proyecto.

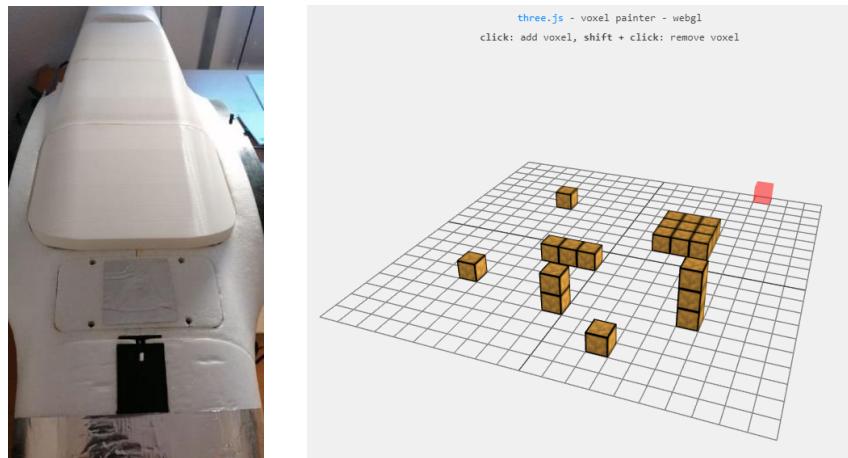


Figura 1.6: Imágenes relacionadas con el testeo del modelo 3D en Catia y con una de las futuras líneas de investigación más prometedoras.

Capítulo 2

Estado del Arte

2.1. Introducción

En el presente capítulo, entenderemos por qué este proyecto se puede considerar dentro de un mercado del tipo “Océano Azul” (mercado sin competencia). Comprenderemos cuál es la situación actual de la industria de los drones, cuáles son sus aplicaciones actuales y futuras más demandadas y cuál es la perspectiva de este mercado en los próximos años. Además, analizaremos cuál es el papel de los drones VTOL y por qué suscitan tanto interés en el mundo actual.

Añadido a esto, indagaremos en las diferentes investigaciones publicadas que existen en la actualidad relacionadas con la creación de sistemas capaces de calcular de una forma sencilla y práctica, para un usuario cualquiera, el CDG de un dron en función de su distribución de carga útil.

Finalmente, al terminar este capítulo comprenderemos de qué manera puede servir el sistema desarrollado en este trabajo para cubrir y solucionar algunas de las necesidades actuales y futuras de la industria mundial de los drones.

2.2. Caracterización de un VTOL

2.2.1. Contexto Mundial Actual de la Industria

Originalmente, los drones fueron creados como alternativas más seguras y baratas a los aviones militares tripulados. Actualmente, cuando se escucha la palabra “drón”, se suele pensar en los drones de tipo militar y los de tipo recreativo. Sin embargo, la proyección a futuro de la industria de drones va mucho más allá.

Se trata de una de las industrias con mayores perspectivas de crecimiento para los próximos años. Estas son algunas de las estimaciones económicas [63][66][67][68][70] futuras del mercado de drones (*figura 2.1*):



Figura 2.1: Estimación del crecimiento global del mercado de drones por región, desde la actualidad hasta 2027.

- **A nivel mundial.**

- Se estima que el mercado mundial de los drones alcanzará en 2025 un valor de 42.800 millones de dólares.
- En la actualidad, se estima que el mercado global de soluciones que utilizan drones mueve alrededor de 127 mil millones de dólares.

- **En Europa y Estados Unidos.**

- Se estima que la flota de drones industriales en Europa y Estados Unidos será de 50.000 millones de dólares para el año 2050 y más de 1 millón de unidades serán vendidas.
- Para 2023, América del Norte seguirá siendo el mayor mercado del mundo para los RPAS comerciales, con una demanda total de unidades del 32,3%.
- Para 2023, el aumento de demanda de RPAS comerciales seguirá en crecimiento, habiéndose alcanzado para entonces un envío de unidades de 2,91 millones de unidades.

- **En España.**

- El Plan Estratégico para el Desarrollo del sector civil de los drones en España estima que la flota de drones de uso profesional podría superar las 51.400 aeronaves en 2035 y alcanzar las 53.500 en 2050. Estos hechos llevarían a un impacto económico de 1.220 millones de euros en 2035 y de 1.520 en 2050.

El gran desarrollo de los RPAS (Remotely Piloted Aircraft Systems) de las últimas décadas responde a una creciente necesidad de: inmediatez, tecnificación y masificación de los servicios, entre ellos la mensajería y el transporte.

Primero fue en el ámbito militar, y ahora en el civil. Los drones han experimentado un enorme auge en uso y popularidad debido a su capacidad de realizar una gran variedad de tareas de forma económica, eficiente y veloz. Son vehículos muy versátiles, de menor tamaño y con bajos costes de mantenimiento. Destaca que su uso es mucho más

económico que el de aeronaves similares que sí estén tripuladas.

Queda claro que los drones tienen una elevada capacidad para impulsar la eficiencia de las operaciones, para mejorar el análisis de datos de las empresas y para reducir costes de operación. Teniendo en cuenta además su precisión y seguridad, se piensa que el futuro de los drones estará principalmente dominado por aplicaciones de tipo comercial. Muchas de estas aplicaciones [69][71], incluidas también en la *figura 2.2*, se pueden ver a continuación:

- **Seguros.** Determinación de las causas, obligaciones, responsabilidades y el daño de un desastre natural y otros eventos.
- **Mapeo 3D.** Empleo en compañías de construcción, agrícolas y mineras para ofrecer “vistas de pájaro” de las áreas de interés. Cartografía aérea.
- **Entrega.** Propósitos médicos en áreas de difícil acceso por tierra, siendo menos costoso que el uso de helicópteros. También aparece el reparto a domicilio con drones.
- **Inspecciones** de líneas eléctricas, torres de telecomunicaciones, grandes infraestructuras, aerogeneradores, plataformas petrolíferas, edificios, grúas, tuberías, paneles solares, etc. Son muy valiosos para inspeccionar áreas difíciles de alcanzar o en entornos contaminados.
- **Transmisión de datos.** Amplificación de las señales de red de internet para acceder a lugares remotos como desiertos o para eventos a gran escala donde la cobertura es inadecuada. Una red totalmente inalámbrica en el cielo sería menos costosa, menos perjudicial y tomaría menos tiempo al construirla que la infraestructura terrestre.
- **Colección de videos.** Seguridad, patrullas aéreas, mayor velocidad de reacción ante situaciones críticas, eliminación de exposición humana en situaciones peligrosas, producción cinematográfica.
- **Termografía.** Determinación de la temperatura superficial de los objetos a distancia, sin necesidad de contacto físico directo. Detección de “hotspots” en paneles solares.
- **Fotogrametría.** Creación de modelos 3D de alta precisión con menos costo que con un avión o un helicóptero y con más datos del terreno que la topografía convencional. Se obtienen “nubes de puntos” de lo escaneado.
- **Control de obras.** Crear planos de edificios, plantas, fachadas, etc. Evaluación del estado de edificios. Peritaje. Planificación, supervisión y seguridad.
- **Misiones de servicio público.** policía (ej. operaciones de búsqueda y rescate), bomberos (extinción de incendios), etc.
- **Otras aplicaciones.** Agricultura, periodismo, eficiencia energética, documentación de la vida silvestre, etc.

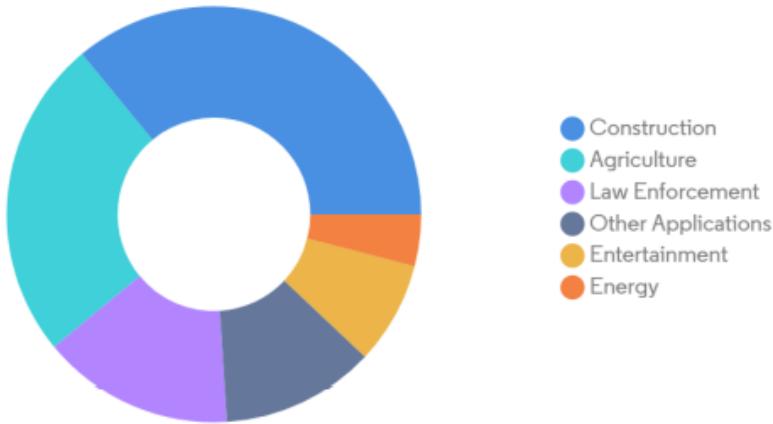


Figura 2.2: Ingresos relativos generados por el mercado de drones por aplicación en 2021.

Aunque los drones más típicos son controlados remotamente por un piloto, los drones completamente autónomos ya están en las últimas fases de desarrollo, cerca de ser una realidad. Estos serán capaces de sustituir los métodos tradicionales de operación de muchas actividades comerciales, conllevando este hecho las siguientes ventajas:

- Reducción del tiempo de realización de muchas actividades comerciales.
- Reducción de costos.
- Mejora en el análisis de datos.
- Mayor comprensión y predicción del rendimiento operativo.
- Aparición de nuevas oportunidades y creación de nuevos modelos de negocio.
- Creación de nuevos puestos de trabajo relacionados: programadores, desarrolladores de software, administradores de redes, etc.

Si bien es cierto que estas tecnologías harán desaparecer muchos otros puestos de trabajo, los beneficios derivados del empleo de drones de uso comercial son muchos más, y sus aplicaciones son casi infinitas, siendo en la actualidad una de las tecnologías con más proyección a futuro. Por este motivo, el día 8 de junio de 2017 el *Economist* [77] escribió:

“Tratar de imaginar cómo evolucionarán los drones, y los usos que se les dará, es como intentar pronosticar la evolución de la informática en la década de 1960 o los teléfonos móviles en los años ochenta. Su potencial como herramientas de negocios estaba claro en ese momento, pero la tecnología se desarrolló de forma inesperada. Lo mismo seguramente será cierto para los drones.”

2.2.2. Tipologías de Drones

Con todo lo comentado anteriormente, es de esperar que para diferentes necesidades hayan surgido a lo largo de la historia diferentes configuraciones típicas [64] para los drones. Estas se pueden dividir en tres:

- **Drones multirrotor (*figura 2.3*).** Presentan grandes ventajas como su alta maniobrabilidad, bajos precios y su facilidad de uso. Sin embargo, tienen ciertas desventajas como son su bajo alcance de vuelo y su alta dependencia de las condiciones atmosféricas para una operación segura. Cabe destacar que son los drones más empleados hoy en día.



Figura 2.3: Ejemplos de drones multirrotor. Destaca el proyecto “Nixie”, a la derecha, por su originalidad.

- **Drones de Ala fija (*figura 2.4*).** Entre sus mayores ventajas encontramos su amplio rango de vuelo, gran estabilidad, mayor eficiencia en el vuelo lineal y la posibilidad de planeo frente a la pérdida del motor. Si bien es cierto, presentan algunas desventajas como son la necesidad de disponer de una zona con amplio espacio para aterrizar y despegar, su elevado coste, su dificultad en el manejo del vuelo, su mayor tamaño y la necesidad de requerir una mayor formación por parte del piloto para dirigirlos. Sobre todo, destaca la capacidad de estos drones para permanecer en el aire durante varias horas gracias al planeo, siendo estos los drones aerodinámicamente más eficientes.



Figura 2.4: Ejemplo de dron de ala fija.

- **Drones VTOL (*figura 2.5*).** Esta tipología de dron presenta innumerables ventajas y pocos inconvenientes. Entre las ventajas están el amplio rango de vuelo, su gran estabilidad, la ventaja de permitir un vuelo lineal eficiente, la recuperación segura en caso de pérdida de potencia del motor, la mayor capacidad de carga, mejor maniobrabilidad y la falta de necesidad de disponer de campos de despegue y aterrizaje habilitados. Por otro lado, entre sus inconvenientes solo destacan el elevado costo de este tipo de drones y la elevada especialización necesaria para pilotarlos, debido a sus características especiales.



Figura 2.5: Ejemplo del dron VTOL “*Fighter VTOL Fixed-wing Aircraft*”, utilizado como base en este trabajo. En este enlace puede encontrarse su vídeo promocional:
<https://www.youtube.com/watch?v=ji-tbCiAR5U>

2.2.3. La Llegada del VTOL

Como ya se mencionó en el *Capítulo 1* de este trabajo, el dron utilizado como base de partida se trata de un VTOL. Es por este motivo por el que se ha decidido dedicarle un apartado especial a este tipo de drones.

El VTOL es un dron que fusiona las ventajas de los drones multirrotor y de los drones de ala fija. Por un lado, las hélices se emplean para su despegue y aterrizaje vertical, sin la necesidad de tener que utilizar una pista, pudiendo realizar dichas maniobras en cualquier zona (igual un dron multirrotor). Por otro lado, el ala le proporciona la estabilidad y la posibilidad de ser utilizado en grandes extensiones de terreno, es decir, le dan una mayor autonomía (como la que tendría un dron de ala fija).

En la búsqueda de la eficiencia, eficacia, autonomía y versatilidad, se pueden encontrar muchos diseños diferentes, innovadores e incluso arriesgados. Como se ha dicho, los vehículos VTOL buscan combinar la alta autonomía de las aeronaves de ala fija con las capacidades de despegue y aterrizaje vertical de los drones multirrotor. Algunas de las soluciones aplicadas [65] para lograr este efecto combinado son:

- **Tail Sitters.** “*Cantas*” (*figura 2.6*), de Checo New Space Technologies.



Figura 2.6: Ejemplo de dron VTOL de tipo “Tail Sitter”.

- **Multirrotor + fixed wing.** *Pelican* (figura 2.7), de Dronetech.



Figura 2.7: Ejemplo de dron VTOL de tipo “Multirrotor + fixed wing”.

- **Hybrid between helicopter and autogiro.** *GT20 Gyrotrak* (figura 2.8) de Airial Robotics.



Figura 2.8: Ejemplo de dron VTOL de tipo “Hybrid between helicopter and autogiro”.

- **Tilt Motors.** *Vehículo de germandrones* (figura 2.9).



Figura 2.9: Ejemplo de dron VTOL de tipo “Tilt Motors”. Puede verse este dron en funcionamiento siguiendo el enlace:
https://www.youtube.com/watch?v=PBg6o_GN8kk

- **Tilt Wing.** Dufour Aerospace VTOL Aircarft [75] (figura 2.10) de la empresa Dufour Aerospace[76], de Suiza.



Figura 2.10: Ejemplo de dron VTOL de tipo “Tilt Wing”. Puede verse este dron en funcionamiento siguiendo el enlace:

<https://www.youtube.com/watch?v=RwAwMujKOXU>

Por último, cabe mencionar ciertos proyectos llamativos donde los VTOL se están viendo involucrados. Entre ellos destacan dos:

- Destaca el proyecto la **empresa suiza Dufour Aerospace** (ver Figura 2.10.), una de las primeras (ej. Airbus, etc.) que ha sacado un eVTOL destinado a ser el “taxi volador del futuro”. Se trata del Dufour Aerospace VTOL Aircarft. He aquí donde radica la importancia del despegue y aterrizaje vertical, pudiendo este aparato realizar estas maniobras en zonas urbanizadas sin la necesidad de disponer de pistas adecuadas para su realización.
- Más que un proyecto que esté en marcha actualmente, como continuación del ejemplo de Amazon (incluido más adelante) con sus entregas de última milla empleando drones multirrotor, se puede pensar que en un futuro los drones VTOL puedan ser los adecuados para realizar entregas y envíos a medio y largo alcance de forma autónoma.

2.3. Estudios Relacionados

Como bien se ha comentado, la industria de los drones se trata de uno de los mercados con mayor potencial de expansión del mundo en la actualidad. A pesar de esto, los estudios publicados relativos al cálculo previo al vuelo del CDG de un dron, en función de su distribución de carga útil, no son muchos.

En cuanto a este campo de estudio, la investigación conocida que tiene más relación con lo desarrollado a lo largo del proyecto aquí presente es el TFG sobre el “Cálculo del centro de gravedad con Arduino para Android” [74] realizado por Diego Sebastian Jiménez Gallegos en la Universitat Autònoma de Barcelona (UAB). El resumen del trabajo de Diego Sebastián,

citado textualmente, es:

“Se presenta el diseño y la implementación de un sistema para medir el centro de gravedad de drones utilizando Arduino y visualizando los datos en un dispositivo con Android. Medir el centro de gravedad nos permitirá llevar un control de cómo se distribuye la carga en un dron, y, por lo tanto, nos puede ayudar a identificar errores que se pueden producir en este a causa de una mala distribución e incluso problemas con la sujeción de la carga.”

El objetivo de su trabajo era crear un sistema capaz de devolver el CDG del dron y sus cargas de pago, dando el resultado respecto al dron. También debía poder mostrar este resultado en una aplicación instalada en un dispositivo Android. Se recuerdan las imágenes mostradas en la *figura 1.3*, las cuales muestran estos resultados logrados por Diego.

Sin embargo, a día de hoy, aún no existen (por lo menos no abiertos al público) sistemas de confianza para distribuir bien el peso en los drones destinados al transporte de cargas, tal que se pueda llevar un control sobre los posibles fallos que se pueden producir por una mala distribución de la carga útil. Sí es cierto que en aviones y helicópteros se conoce de manera bastante fiable la posición del CDG antes de cada vuelo mediante el uso de tablas, gráficas y esquemas de carga. Sin embargo, este tipo de estudios tan exhaustivos para elaborar dichas tablas y gráficas, hoy en día todavía no se aplican (públicamente) en la mayor parte del mercado de drones. Es por este motivo por el que este proyecto se puede considerar dentro de un “*Océano Azul*”, por no tener casi competencia conocida en el mercado actual de drones y por poder ofrecer la posibilidad tanto a empresas como a particulares de disponer de un sistema usable, visual y sencillo capaz de proporcionarles una buena estimación del estado de carga de sus drones.

2.4. Necesidades de la Industria de Drones

En última instancia, toda aplicación comercial a la que sirven los drones debe ser lo más beneficiosa posible, económicamente hablando. Es por ello por lo que se intenta constantemente reducir los costes derivados de la operación de estos vehículos.

Las preocupaciones que tienen más relevancia en la actualidad para los operadores de la industria de drones son las que inciden directamente en los costes de operación. Estos asuntos están relacionados con problemas relativos a:

- Carga útil.
- Eficiencia aerodinámica.
- Alcance del vuelo.
- Ahorro energético.
- Autonomía.

En la industria ya se están realizando avances significativos que ayudan a mejorar las problemáticas asociadas al ahorro energético, aumento de la autonomía e incremento del alcance. Estas son algunas de las vías de desarrollo actuales:

- Drones alimentados con combustibles alternativos.
- I+D en materiales compuestos para reducir el peso de la aeronave.

Ahora, tal y como comenta Diego Sebastian en su TFG, respecto a la preocupación relacionada con la carga útil, estos son algunos de los posibles problemas que pueden derivar de una mala distribución de la carga de pago:

1. En drones multirrotor, con el fin de mantener la estabilidad del vuelo, puede haber **motores más cargados que se desgasten antes** y, por lo tanto, fallen antes que el resto de los motores.
2. Los **giros** pueden ser **más difíciles** de hacer con una mala distribución de carga.
3. El **sistema de sujeción de la carga puede no ser el adecuado**, haciendo que esta se desplace y aparezcan fallos durante el vuelo.

Estos tres ítems pueden terminar convirtiéndose en accidentes, malfuncionamiento de los aparatos y pérdidas económicas a gran escala para las empresas, más teniendo en cuenta la gran proliferación de la industria en tantos sectores.

Por estos motivos, Virginia y yo, en el proyecto desarrollado (recomiendo consultar el TFG de Virginia en el futuro), proponemos el diseño de un software que permita simular la situación real de carga del dron estudiado a través de una interfaz web desarrollada con HTML y JavaScript. Esto permitirá a los usuarios conocer si su distribución de carga es o no correcta antes de realizar el vuelo, evitando todos los tipos de accidentes mencionados.

2.4.1. ¿Por qué le puede Interesar a la Industria este Proyecto?

Tener la oportunidad de hacer ensayos prevuelo que permitan determinar la mejor posición posible para alojar las cargas de pago necesarias en el interior de un dron, puede ser muy interesante para todas las empresas que tengan el objetivo de automatizar procesos mediante el uso de una flota de drones. Aplicar estrategias preventivas de daños y accidentes, como la presentada en este trabajo, les permitirá reducir sus costes de mantenimiento y posibles pérdidas económicas relacionadas.

Por ejemplo, dos famosos proyectos en los que podría ser muy interesante aplicar este software son:

1. **Amazon Prime Air (figura 2.11).** Patentado por Amazon, se trata de un ambicioso proyecto que pretende poder entregar pedidos en 30 minutos mediante el uso de drones autónomos guiados por GPS, siendo una tecnología muy útil para realizar entregas en zonas alejadas o de difícil acceso, como puede ser el caso de islas o plataformas petrolíferas. Este proyecto reducirá el costo por entrega y el tiempo de entrega de los pedidos, estimándose un ahorro del 40% en los costos de envío, que a su vez podría traducirse en un aumento del 15-20% en el margen de beneficio y, por tanto, una disminución del precio de los pedidos del 15-20%.

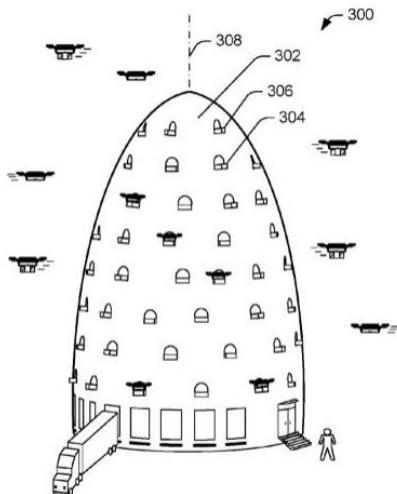


Figura 2.11: Imagen real de un dron mensajero de Amazon Prime Air y boceto de una “colmena de drones” del mismo proyecto.

2. **Movilidad Aérea Urbana (UAM).** En España, al igual que en otros muchos países, y debido al continuo crecimiento de la población, se plantea la posibilidad de desarrollar una red de drones destinados a la movilidad aérea urbana, tanto para logística como para personas. En esta categoría se encuentra el dron de Dufour Aerospace, mostrado en la *figura 2.10*.

En proyectos como estos, la carga de pago y su distribución siguen patrones ciertamente aleatorios, pudiendo depender de cada entrega y de cada viaje (esto dependerá del dron en concreto). Aplicar este tipo de software antes de cada vuelo podría ayudar a resolver los problemas derivados de las malas distribuciones de carga.

Este es el principal motivo por el que el desarrollo de una página web fácil de utilizar y sencilla de entender como la presentada, puede ser de mucha utilidad para las futuras y presentes aplicaciones comerciales de los drones, permitiendo a cualquier empleado poder hacer uso de esta herramienta.

Capítulo 3

Metodología

3.1. Introducción

Con el fin de alcanzar los objetivos propuestos en el *Capítulo 1*, se ha empleado la *Metodología en Cascada*, cuyas fases son las mostradas en la *figura 3.1*, a continuación:

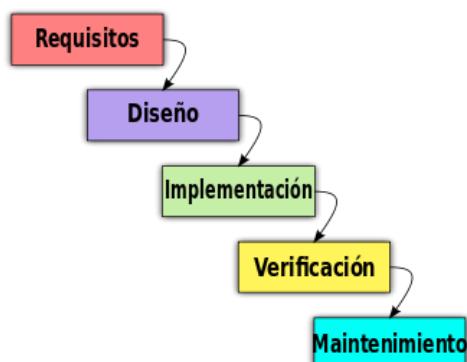


Figura 3.1: Fases de la metodología en cascada.

En la etapa de requisitos se identifican los objetivos perseguidos por el proyecto. Estos ya se han visto con bastante detalle en el *Capítulo 1*. En el *Capítulo 2* también se ha mostrado cuál es la motivación comercial principal de este trabajo.

Un buen resumen de todos los objetivos perseguidos en este trabajo (dedicado al Frontend del proyecto), aunándolos en uno solo, sería el siguiente:

"Creación de una interfaz web (HTML5+Javascript) que permita a sus usuarios realizar cálculos del centro de gravedad y las actuaciones de un dron VTOL en función de la distribución de carga de pago del mismo. Esta página web servirá como capa de comunicación entre un modelo 3D del dron -actualmente utilizado en misiones de cartografía, y previamente modelado en una herramienta de CAD- y los cálculos realizados en Python."

En la fase del diseño se hará un primer planteamiento - un boceto - del diseño de la página web a desarrollar, es decir, el diseño del HMI. Tras ello se comenta cuáles han sido las herramientas empleadas para llevar a buen puerto este proyecto. También se hablará sobre las decisiones y alternativas de desarrollo más importantes que se han seguido.

También, se detallará la estrategia planteada para desarrollar el modelo 3D del VTOL en Catia V5, detallando cuáles han sido los módulos y herramientas principalmente utilizados para alcanzar con buen éxito esta tarea.

Esta etapa de la metodología será la que se describirá en los diferentes apartados de este capítulo.

Ahora, la **fase de implementación** viene determinada por la creación del software utilizado para alcanzar los objetivos propuestos. Primero se muestra el proceso de creación del fuselaje del modelo 3D del dron estudiado. Luego se hablará de los códigos programados en HTML5, JavaScript, CSS y el servidor en Python que envía los datos y devuelve los resultados. El estudio detallado de esta fase del proceso puede verse en el “*Capítulo 4. Implementación*”

Llegamos ya casi al final de la metodología aplicada en este trabajo, la **fase de verificación**, detallada en el “*Capítulo 5. Testeo y Validación*”. Principalmente se comprobará que el sistema HMI desarrollado funciona correctamente. Para ello se mostrará la forma de funcionar de la aplicación a través de un ejemplo real de uso, a modo de prueba. Se comprobará que todos los resultados implementados en la salida del servidor se muestran correctamente y también se asegurará que ningún tipo de entrada de datos que el usuario introdujera produzca problemas. También se verá el testeo realizado sobre el modelo en Catia del VTOL.

Por último, la **fase de mantenimiento** se trata de un proceso de mejoras, de puesta a punto y de comprobaciones exhaustivas para asegurar el correcto funcionamiento de todo el sistema. Esta es la fase también conocida como de “operaciones”, donde se realizan más y más comprobaciones y se utiliza el sistema desarrollado, asegurando su fiabilidad. Sin embargo, dado que este es un trabajo cerrado -una vez terminado el proyecto, todas las acciones sobre él finalizan- en principio esta fase no tendrá lugar en ningún momento. Si bien, en el “*Capítulo 6. Conclusiones*” se plantean posibles ideas de mejora de la aplicación web desarrollada, abriendo la posibilidad a futuras líneas de investigación y de desarrollo.

Ahora, empezaremos con el análisis de la fase de diseño.

3.2. Trabajo realizado sobre el VTOL

En este primer apartado de la metodología, explicaremos desde el inicio cuál ha sido el procedimiento seguido para obtener el modelo 3D en Catia V5 del dron “Fighter VTOL Fixed-wing Aircraft”.

3.2.1. Toma de Medidas

El primer paso para hacer el modelo consistió en elaborar un plan que nos permitiera conocer todas las medidas características y necesarias que nos sirvieran para construir el modelo 3D en Catia V5.

La primera idea que se tuvo fue la de realizar un fotoescaneado del VTOL utilizando el software ODM (cuyo logo puede verse en la *figura 3.2*), capaz de generar una nube de puntos del dron completo en el ordenador. Sobre esta nube de puntos, se podrían tomar todas las medidas necesarias con bastante precisión sobre la propia pantalla del ordenador, lo cual proporcionaría bastante precisión en la toma de todas las medidas. Para crear esta nube de puntos, fue necesario tomar una gran cantidad de fotografías del dron desde todos sus ángulos, algunas de las cuales se pueden visualizar en las *figuras 3.3, 3.4 y 3.5*.



Figura 3.2: Logo del software ODM.

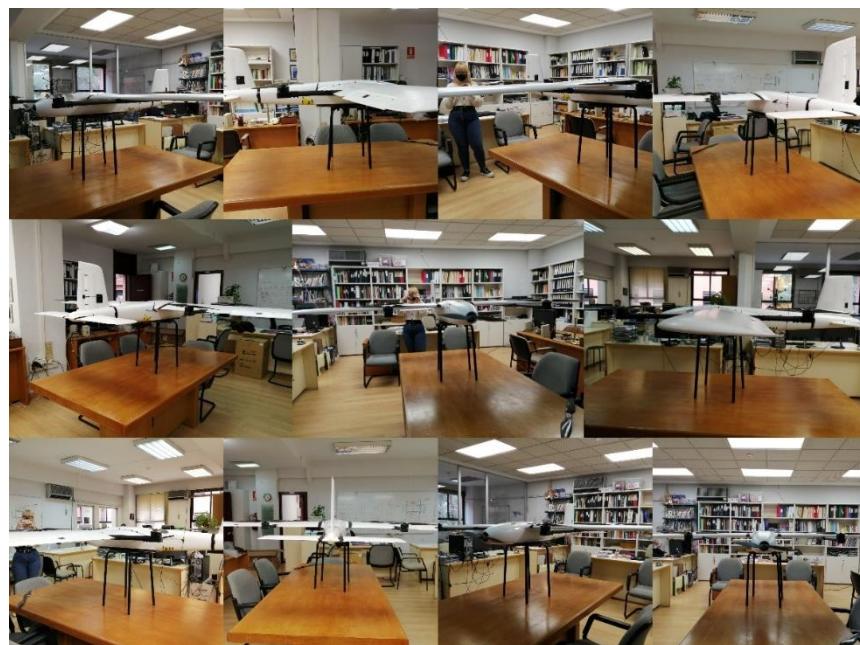


Figura 3.3: Collage número 1 de algunas de las 87 fotografías tomadas durante la realización del fotoescaneado.

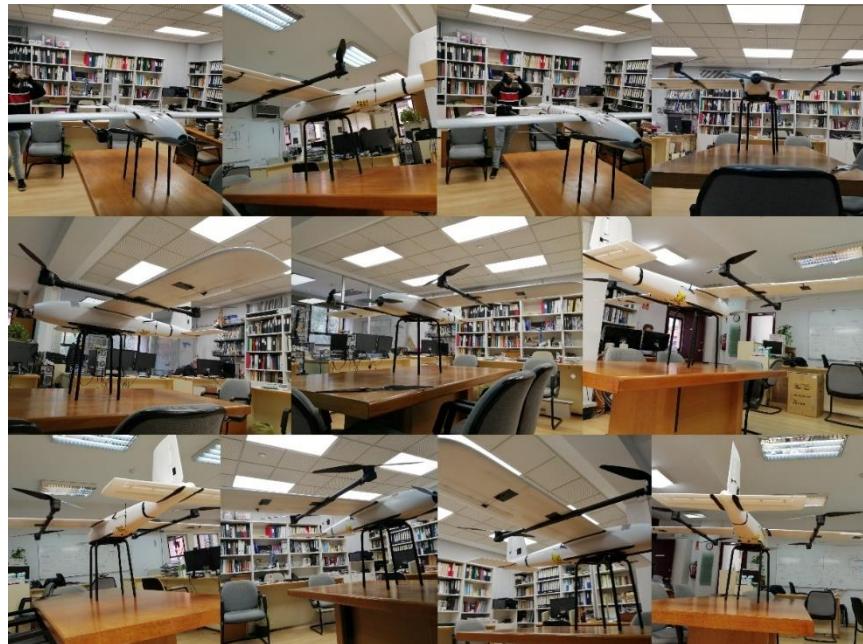


Figura 3.4: Collage número 2 de algunas de las 87 fotografías tomadas durante la realización del fotoescaneado.



Figura 3.5: Imágenes de Virginia y César mientras tomaban las fotografías destinadas a realizar el fotoescaneado.

Sin embargo, este intento de fotoescaneado fracasó cuando la nube de puntos obtenida quedó emborronada, muy probablemente a causa del fondo de las imágenes, del cambiante ángulo de toma de las fotos, y debido a la falta de espacio en el DAVE para realizar mejor dichas fotografías. Se estima que el fondo fue el principal problema, pues puede observarse que se trataba de un fondo muy heterogéneo que podía “confundir” al programa.

Tras este intento fallido y, con el fin de asegurar la continuación del proyecto, se decidió dejar el dron en el DAVE durante una semana completa con el fin de tomar todas sus medidas características de forma manual (se recuerda ver la *figura 1.4*) y poder estudiar todos los detalles posibles del dron con más detenimiento. Si bien esta opción proporcionaría una menor precisión en las mediciones, dadas las circunstancias y objetivos del trabajo, el error cometido se consideraría aceptable.

Las medidas de las dimensiones de las piezas fueron realizadas con un calibre digital de 0.01mm de precisión y con un metro de 1mm de precisión. Por otro lado, las medidas de los pesos de todas las partes del VTOL se tomaron con una báscula de 0.001kg de precisión. De esta forma, se pudieron obtener los pesos característicos del aparato, como son su OEW y su MPL.

A pesar de la bondad de muchas de las medidas, debe quedar patente la gran dificultad existente al medir muchas de las partes del dron, hecho que afectó sobre todo a la toma de medidas de las secciones del fuselaje. Ante este problema, se tomarían dos tipos de soluciones distintas dependiendo de la pieza:

- Utilizar más adelante la serie de fotografías tomadas del dron para realizar “Sketches Inmersivos” [86] en Catia V5, siempre que estas estuvieran en verdadera magnitud y fueran de la calidad suficiente. En el “Capítulo 4. Implementación” se explicará detalladamente en qué consiste esta forma de proceder.
- Hacer inferencias y aproximaciones basadas en las medidas que sí se pudieron tomar con buen grado de certeza. Esta fue la solución adoptada para lidiar con el problema achacado a las secciones del fuselaje. Tal y como puede verse en la *figura 3.6*:



Figura 3.6: Imagen en detalle del fuselaje.

Se observa que este fuselaje es de sección variable. Este hecho supone un problema a la hora de tomar sus medidas para hacer su modelado 3D, pues solo se conocen con buen grado de certeza las secciones trasera y delantera (una circunferencia en este caso). En la *figura 3.7* pueden verse imágenes en verdadera magnitud de estas dos secciones mencionadas:



Figura 3.7: Imagen de las secciones trasera y delantera del dron VTOL.

Las secciones intermedias, con las herramientas de medición disponibles, eran imposibles de conocer completamente. Al mismo tiempo tampoco se podían hacer sketches inmersivos de estas secciones, pues evidentemente resultaba imposible obtener fotos del corte de las secciones. Por ello, se debieron inferir y estimar tanto las medidas como las formas de dichas secciones.

Una primera aproximación fue considerar que la longitud del fuselaje se podía separar en dos partes, según esta fotografía (*figura 3.8*):



Figura 3.8: Dos partes inicialmente consideradas del fuselaje.

Se consideró que la “PARTE 1” consistía en secciones que se asemejaban a circunferencias y que experimentaban una variación de su sección muy pequeña hasta la línea amarilla divisoria. Mientras, se consideró que la “PARTE 2” consistía en secciones que se asemejaban a la sección trasera del fuselaje, aumentadas con factores de escala hasta la línea divisoria.

Tras aplicar los factores de escala [85] considerados correctos en base al resto de medidas características de la aeronave (sí conocidas), el resultado obtenido fue el que mostrado en la siguiente figura (*figura 3.9*):

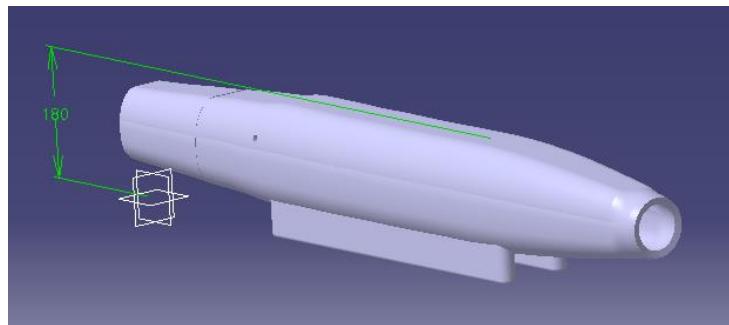


Figura 3.9: Fuselaje obtenido mediante la aplicación de factores de escala a las secciones trasera y delantera.

Este se trataba, evidentemente de un resultado bastante insatisfactorio, pues el fuselaje de esta imagen (*figura 3.9*) tiene muy pocos parecidos con el fuselaje real del dron, destacando lo excesivamente “estrecho” que resultó ser. Debido a la poca similitud con el fuselaje real, se desechó la aproximación realizada. Se desechó el uso de los factores de escala y la hipótesis de las dos partes diferenciadas. El procedimiento final adoptado para diseñar el fuselaje es la descrita en el “*Capítulo 4. Implementación*.”

Durante la toma de medidas a lo largo de la semana, se fueron confeccionando manualmente, con papel y lápiz, los planos preliminares de todas las piezas y partes representativas del dron. En el “*Capítulo 4. Implementación*” se muestran los planos del fuselaje y de las tapas que lo conforman.

Si desea ver los planos del ala, la cola y otras partes representativas del dron, le recomiendo revisar el TFG de Virginia dentro de unos meses.

3.2.2. Creación del Modelo 3D

Una vez terminados los planos necesarios, se comenzó a desarrollar el modelo 3D del dron mediante el programa informático de diseño Catia V5. Se eligió dicha herramienta debido a su versatilidad, sus múltiples opciones de diseño y su sencillez en el manejo. Para la creación de todas las piezas del dron explicadas en este trabajo, se utilizarían los siguientes módulos del programa:

3. Módulo Part Design de CATIA V5 (*ver figura 3.10*).

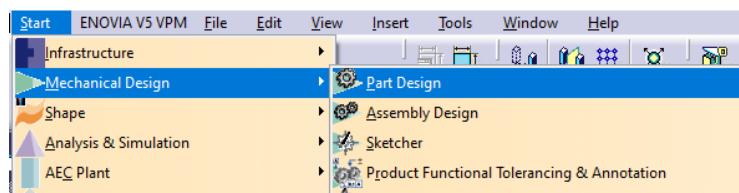


Figura 3.10: Acceso al módulo Part Design de Catia V5.

4. Módulo Assembly Design de CATIA V5 (*ver figura 3.11*).

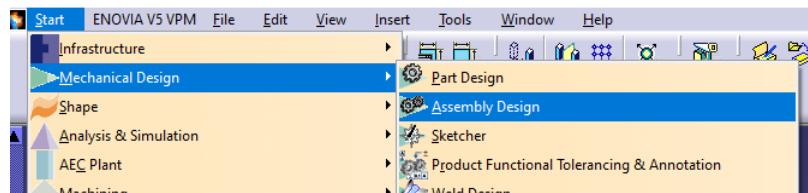


Figura 3.11: Acceso al módulo Assembly Design de Catia V5.

5. Módulo Generative Shape Design de CATIA V5 (*ver figura 3.12*).

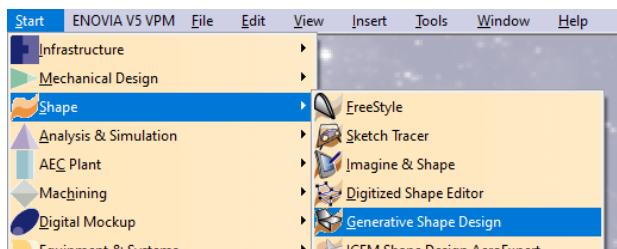


Figura 3.12: Acceso al módulo Generative Shape Design de Catia V5.

6. Módulo Sketch Tracer en CATIA V5 (*ver figura 3.13*).

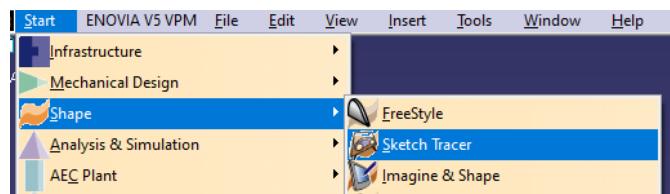


Figura 3.13: Acceso al módulo Sketch Tracer de Catia V5.

Estas son, además, las herramientas más utilizadas de cada uno de los módulos mencionados:

- Respecto al módulo Part Design, de izquierda a derecha nos encontramos con, respectivamente, las herramientas (*ver figura 3.14*): Thick Surface, Mirror, Apply Material, Sketch, Plane, Edge Fillet, Pad [79], Pocket, Multi-Pad, Close Surface y Multi-Pocket.



Figura 3.14: Herramientas utilizadas en el módulo Part Design.

- Respecto al módulo Assembly Design tenemos (*ver figura 3.15*): Coincidence Constraint, Contact Constraint, Offset Constraint y Angle Constraint.



Figura 3.15: Herramientas utilizadas en el módulo Assembly Design.

- Del módulo Generative Shape Design usamos (*ver figura 3.16*): Multi-Sections Surface [78], Join, Fill [81], Translate y Rotate.



Figura 3.16: Herramientas utilizadas en el módulo Generative Shape Design.

- Por último, en cuanto al módulo Sketch Tracer las únicas herramientas que aplicaremos serán (*ver figura 3.17*): Create an Immersive Sketch y Top View.



Figura 3.17: Herramientas utilizadas en el módulo Sketch Tracer.

En el “*Capítulo 4. Implementación*” se desarrollará más concretamente el uso dado a cada una de estas herramientas con el fin de crear el modelo 3D del fuselaje del VTOL y las otras piezas comentadas en dicho capítulo.

3.3. Análisis Funcional de la Aplicación Web

3.3.1. El Día de la Prueba de Vuelo

Una vez terminado el modelo 3D del VTOL, mi compañera Virginia y yo asistimos de la mano de nuestro tutor, Luis Izquierdo, a una prueba de vuelo de la aeronave en un aeródromo de Sigüenza, Guadalajara (*ver figura 3.18*).



Figura 3.18: Imagen de Virginia, César y el VTOL en el aeródromo de Sigüenza.

El objetivo principal de la misión a realizar aquel día, por parte de Luis y de sus compañeros de trabajo, era el de testear unos sistemas de comunicación tierra-aire, buscando lograr la transmisión de imágenes desde la cámara alojada en el dron hasta un ordenador en tierra. No obstante, mi compañera y yo asistimos a dicha misión con el objetivo de entender al completo cuáles podían ser las necesidades pendientes de resolver por el equipo que, además, estuviesen directamente relacionadas con la carga útil del dron y su distribución.

Lo primero que se estudió fue el mecanismo de medición prevuelo del CDG del dron realizado en pista, recomendado por el fabricante. La forma de hacerlo consistía, tal y como se puede percibir en la *figura 3.19*, en la elevación del dron sujetado por una cuerda que se debía colocar en la sección del dron donde debería estar situado el CDG (típicamente situado al 25% de la cuerda). En caso de encabritar o picar en exceso la aeronave durante dicho levantamiento, sería necesario redistribuir la carga con el fin de situar el CDG en una posición más aceptable.

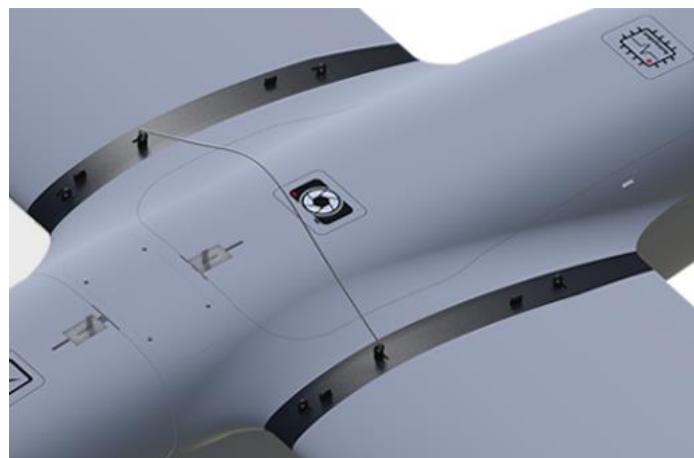


Figura 3.19: Mecanismo de medición del centro de gravedad del VTOL. Imagen tomada del sitio web <https://es.aliexpress.com/item/10000223165284.html>

Lo segundo que se pudo observar aquel día fue el siguiente problema. Nuestro tutor y sus compañeros buscaban introducir en el interior del dron unas placas electrónicas necesarias para cumplir con la misión propuesta, aparte de las baterías. En determinado momento, finalizando el proceso de carga, se percataron de que una de las placas no cabía correctamente. Por este motivo se vieron en la necesidad de tener que recolocar las baterías, desplazando el CDG hacia un punto de mayor inestabilidad. Finalmente, aunque todo entró dentro del espacio de carga del dron, las piezas quedaron algo mal colocadas, pudiendo sufrir daños durante la operación.

Nuestro tutor ya nos había comentado con anterioridad su ilusión por disponer de un sistema que pudiera permitir establecer un correcto posicionamiento del CDG del dron antes siquiera de cargarlo en la pista. Conocer estos datos con una simple simulación de ordenador podía ser de mucha utilidad para evitar problemas repentinos como el comentado. Por este motivo, tras este trabajo de campo con Luis, se comprobó que disponer de un sistema como el diseñado durante este proyecto - complementario a la rudimentaria medición del CDG recomendada por el fabricante - era necesario.

3.3.2. Análisis Funcional

Gracias al trabajo de campo descrito anteriormente se consiguió definir cuáles eran los datos necesarios a introducir por el usuario, para luego poder devolverle los resultados buscados. Además, se hizo una primera aproximación de la manera en que se lograría alcanzar los objetivos propuestos: proporcionar resultados sobre las actuaciones del dron y calcular su centrado en función de su distribución de carga útil.

En el *Capítulo 4* se detallarán cuáles son estos inputs, cuáles son los outputs y, lo más importante de todo, cuál es el fluograma de la aplicación resultante. Se incluirán también capturas de todos los códigos programados con el editor de texto “Visual Studio Code” a lo largo del proyecto.

Finalizando este apartado, cabe aclarar que en este trabajo no se tratarán las fórmulas ni métodos empleados para calcular los resultados de salida. Para poder ver todos estos cálculos y fórmulas, deberá dirigirse en unos meses al TFG escrito por Virginia. A continuación, se muestran algunas imágenes (*figuras 3.20, 3.21, 3.22 y 3.23*) descriptivas del proceso funcional del Frontend del sitio web, explicado más en detalle en el “Capítulo 4. Implementación.”:

"VOLAR SE HACE EN GRAN MEDIDA CON LA IMAGINACIÓN"

¡Descubre tus posibilidades!

1/2 - Defina las cargas de pago

Con el objetivo de calcular el **centro de gravedad** de su dron, necesitaremos que introduzca los datos de las cargas de pago que desee añadir. Se piden la posición (X, Y, Z), las dimensiones (A, B, H) y la masa de cada carga.

Carga 1: Nombre 1 X1 cm Y1 cm Z1 cm A1 cm B1 cm H1 cm Masa 1 kg
 Carga 2: Nombre 2 X2 cm Y2 cm Z2 cm A2 cm B2 cm H2 cm Masa 2 kg
 Carga 3: Nombre 3 X3 cm Y3 cm Z3 cm A3 cm B3 cm H3 cm Masa 3 kg

Añadir carga Eliminar la última carga Guardar y cargar datos

Cualquier aclaración que pueda necesitar en este apartado, se encuentra en la sección "Bienvenida", en "¿Cómo funciona V&C Drone?".

2/2 - Calcule el alcance y la autonomía

Figura 3.20: Inputs para introducir los datos de las cargas de pago del usuario.

Cualquier aclaración que pueda necesitar en este apartado, se encuentra en la sección "Bienvenida", en "¿Cómo funciona V&C Drone?".

2/2 - Calcule el alcance y la autonomía

Con el objetivo de calcular el **alcance** y la **autonomía** de su dron, necesitaremos que introduzca los siguientes datos:

N _{mot} :	Número de motores	Q:	Capacidad de la batería mA·h
I _{mot} :	Corriente máxima por motor A	C _{rate} :	Tasa C de la batería h ⁻¹
I _{equip} :	Corriente máxima otros equipos A	DR:	Regla de descarga de la batería %
V _{bat} :	Tensión nominal de la batería V	L _{flying} :	Carga de vuelo %
V ₀ :	Velocidad de vuelo km/h	Guardar y Calcular	

Cualquier aclaración que pueda necesitar en este apartado, se encuentra en la sección "Bienvenida", en "¿Cómo funciona V&C Drone?".

Figura 3.21: Inputs para introducir los datos del dron y sus baterías.

"VOLAR SE HACE EN GRAN MEDIDA CON LA IMAGINACIÓN"

¡Descubre tus posibilidades!

1/1 - Clique aquí para ver sus resultados

- La **cantidad de cargas** introducidas es de - cargas. Los resultados obtenidos con esta distribución de carga son los siguientes.
- Aproximadamente, la **autonomía** de su vuelo será de:

- h

- Aproximadamente, el **alcance** de su vuelo será de:

- km

Figura 3.22: Imagen 1 del diseño de la sección de los outputs.

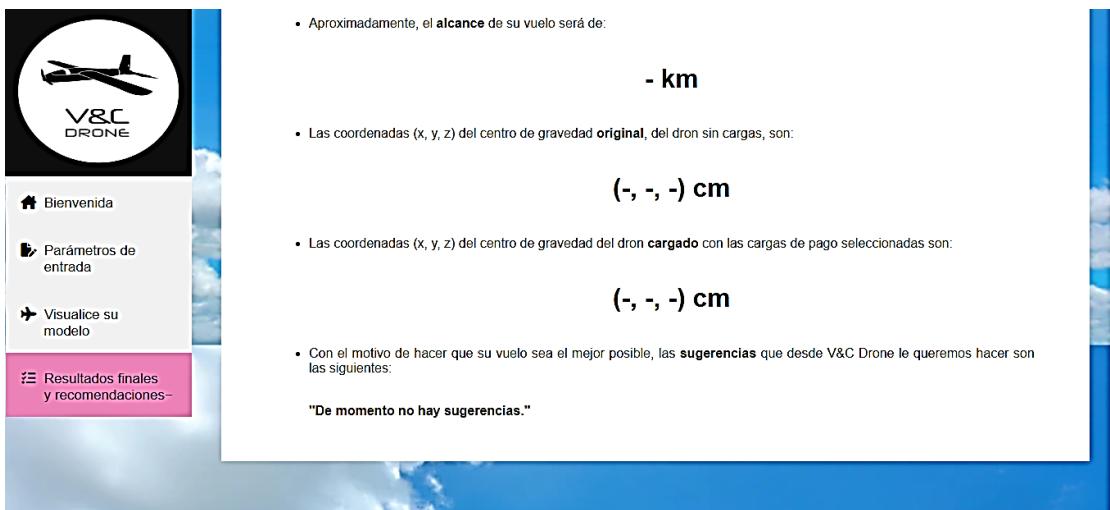


Figura 3.23: Imagen 2 del diseño de la sección de los outputs.

3.4. Diseño del HMI de la Aplicación Web

Respecto al diseño del HMI, desde un principio, el principal objetivo ha sido el de alcanzar la máxima usabilidad posible, haciendo de esta una aplicación web sencilla de utilizar.

Por este motivo, se optó por incorporar las secciones mínimas necesarias para hacer funcionar la página web, permitiendo al mismo tiempo entender correctamente el funcionamiento de la misma por parte de los usuarios. Un primer diseño fue el mostrado a continuación, en la figura 3.24:

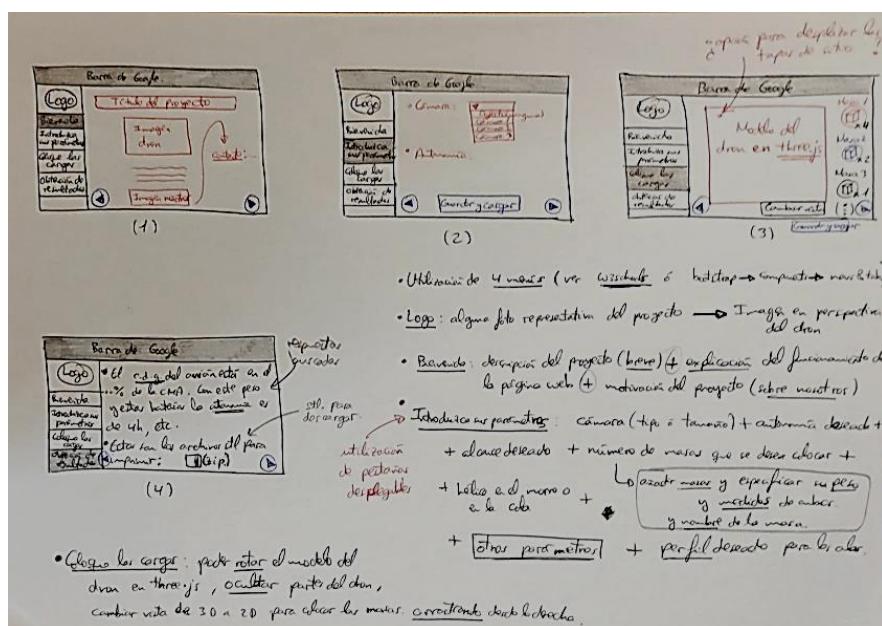


Figura 3.24: Boceto del primer diseño del HMI.

La búsqueda de la simplicidad y la claridad llevó a la utilización de elementos como secciones, acordeones [16font], botones, utilización de “swal”, uso de three.js, jQuery y llevó a tomar otras muchas decisiones que más adelante se comentarán. Destaca el uso de las bibliotecas:

- **three.js** (*figura 3.25*): empleada para crear la animación en 3D del dron en el navegador web. (Ver referencias de la [1] a la [10])
- **jQuery** (*figura 3.25*): empleada para simplificar el manejo de eventos, interactuar con el HTML y hacer una página web más interactiva y animada. [40] [41] [39] [46]



Figura 3.25: Logos de las bibliotecas utilizadas en JavaScript para este proyecto.

3.5. Testeo y Validación del Trabajo

La validación de este proyecto consistirá en la ejecución de un ejemplo de uso real de la aplicación web por parte de un usuario ficticio. A medida que este usuario avance por la página web, se irá documentando con imágenes la demostración real de uso de la misma, comprobando que las diferentes funciones de JavaScript, la comunicación con el servidor y la disposición de los elementos en la página HTML funcionan correctamente.

Además, se validará el modelo 3D realizado en Catia V5 a través de una experiencia de aplicación real. Ante la necesidad de disponer de una nueva tapa de las pertenecientes al dron por motivos que se explicarán más adelante, se estudiará el resultado alcanzado mediante su impresión 3D (ver *figura 3.26*). Se comentará el resultado obtenido, comprendiendo lo aceptable de las aproximaciones realizadas durante el proceso de diseño y la toma de medidas.

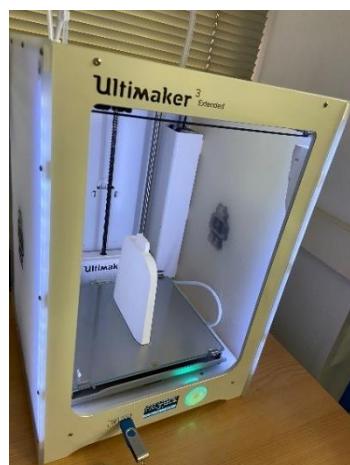


Figura 3.26: Impresora 3D imprimiendo la tapa del dron.

Cabe decir que, para entender y leer sobre los parámetros de impresión 3D utilizados, las características de la impresión y los fallos y aciertos relacionados con dichos parámetros, en el futuro usted deberá dirigirse al TFG de Virginia para poder leer este contenido.

Capítulo 4

Implementación

4.1. Introducción

En este capítulo se mostrará paso por paso la forma de lograr los objetivos propuestos. Del capítulo anterior ya se conocen los métodos utilizados en el desarrollo de este proyecto; ahora se entenderá en detalle la aplicación práctica de dichos métodos.

Primero, se mostrará el resultado final del diseño en Catia V5 del dron. Se mostrarán los planos obtenidos y las piezas diseñadas parte por parte.

Más adelante, se proporcionarán capturas de los códigos programados en HTML5, JavaScript, CSS y el servidor en Python empleado. También, se analizará el fluograma de todo el sistema y se hablará de la manera en que el HMI se ha implementado.

4.2. Diseño Inicial del Dron

Como se dijo en el capítulo anterior, mediante el uso de un calibre digital, una báscula y un metro se tomaron todas las medidas posibles directamente sobre el dron. Al mismo tiempo se hicieron los planos del mismo y se tomaron fotografías de todas las vistas posibles con el fin de luego poder hacer comparaciones entre el VTOL real y el modelo en Catia.

Como ya se ha comentado anteriormente, las piezas cuyo diseño trataremos en este TFG son:

- El **fuselaje**.
- Cinco **tapaderas** del fuselaje llamadas: tapa 1, tapa 2, tapa 3, tapa 4 y tapa inferior.
- **Patines** del tren de aterrizaje.
- **Tubos** estructurales del interior del fuselaje.

Los procesos de diseño del resto de las piezas (semialas, cola y otras piezas más) se podrán encontrar próximamente en el TFG de mi compañera Virginia.

4.2.1. Planos obtenidos

Tras realizar todas las medidas pertinentes de las piezas comentadas, se obtuvieron los siguientes planos del fuselaje y los patines (*figura 4.1*), y de sus tapas (*figura 4.2*):

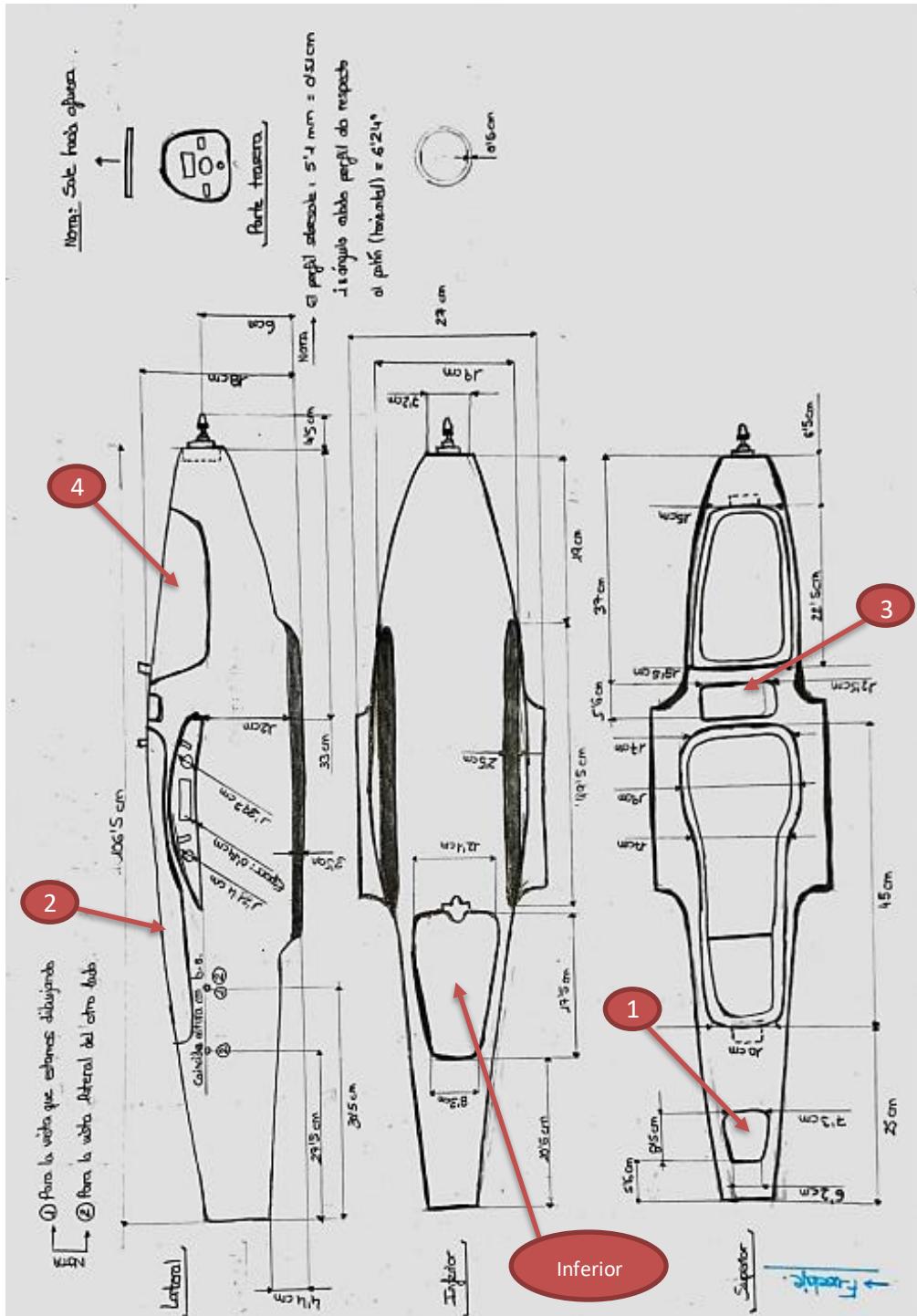


Figura 4.1: Planos del fuselaje y los patines del “*Fighter VTOL Fixed-wing Aircraft*”.

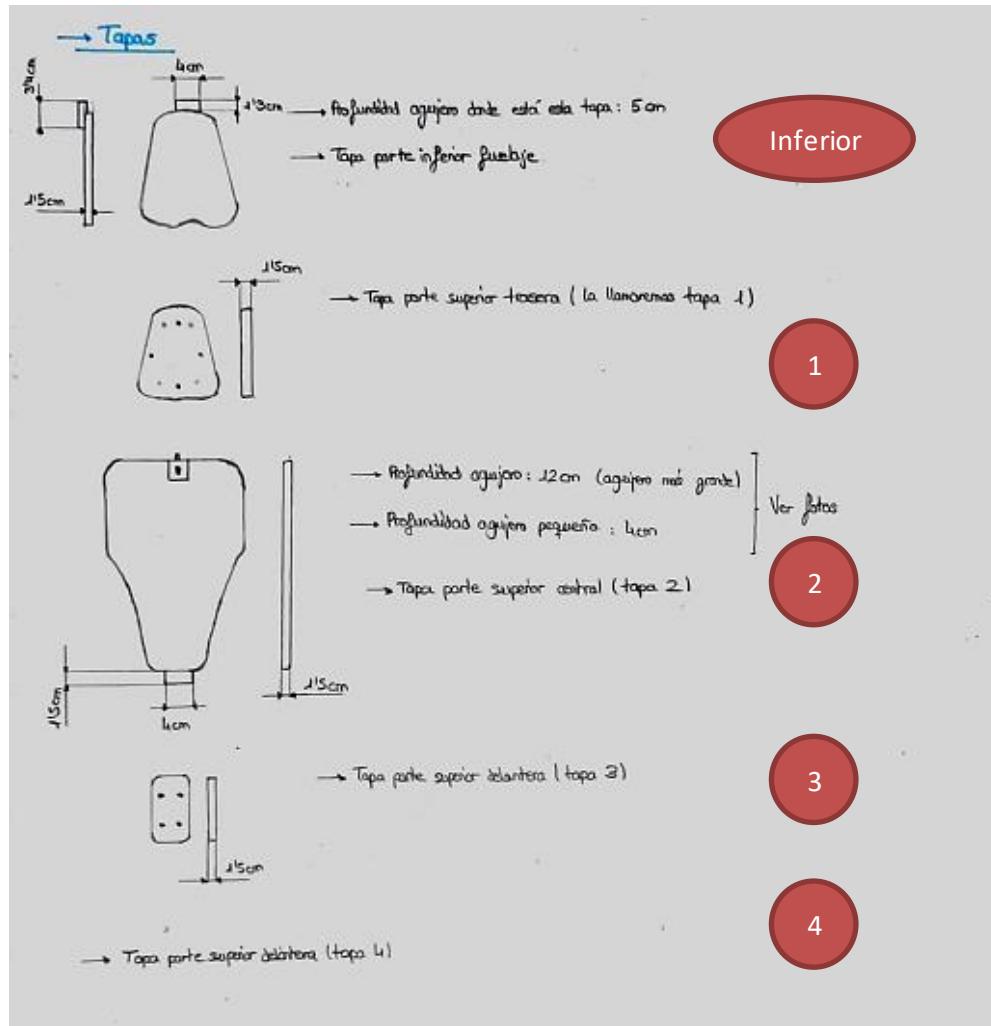


Figura 4.2: Planos de cuatro de las cinco tapas del fuselaje.

Aunque no es de interés para este TFG (sí será relevante en el TFG de Virginia), se incluye en la figura 4.3 una imagen que incluye los pesos medidos de todas las partes del VTOL.

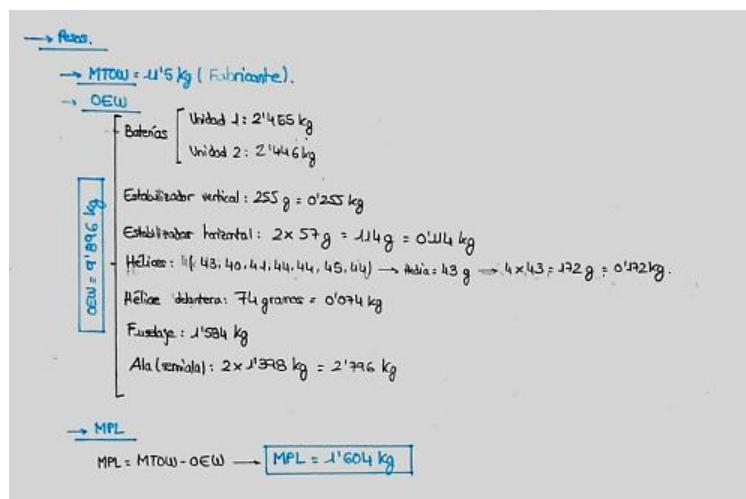


Figura 4.3: Pesos característicos del “Fighter VTOL Fixed-wing Aircraft”.

Además, debe tenerse en cuenta que los planos de las medidas características (ver *figura 4.4*) proporcionados por el fabricante del dron también han sido de ayuda, a modo orientativo de las aproximaciones realizadas. Estos planos son:

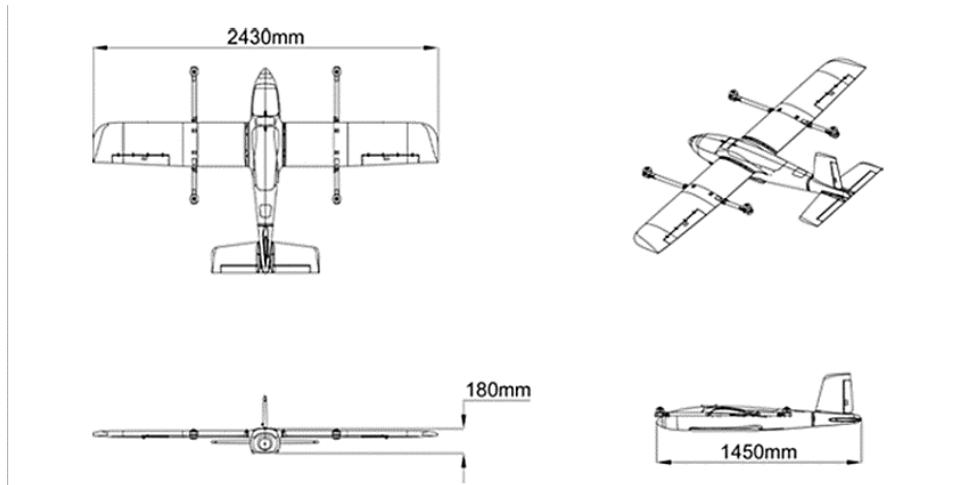


Figura 4.4: Dimensiones características de la aeronave, proporcionadas por el fabricante en el sitio web <https://es.aliexpress.com/item/10000223165284.html>

Una vez obtenidos los planos donde se incluyen todas las medidas representativas del dron que se pudieron medir de manera directa, se comenzó a hacer el diseño 3D de las piezas ya mencionadas. A continuación, se detalla dicho proceso, pieza por pieza.

4.2.2. Realización de Sketches Inmersivos

La sinuosidad de la forma de las tapas del dron imposibilitó la toma de muchas de las mediciones necesarias para hacer un modelo 3D realista con las herramientas utilizadas. Es por este motivo por el que se decidió buscar una alternativa mejor.

Dado que las fotos de estas tapas tenían bastante calidad, mi compañera y yo emprendimos la búsqueda de alguna herramienta en Catia que nos permitiera hacer un sketch de la silueta de la pieza. Finalmente, Virginia dio con la solución: “Sketches Inmersivos”. Esta es la manera de referirse en Catia a la realización de un sketch sobre una imagen previamente cargada en Catia.

La primera prueba de esta solución la realizó Virginia, haciendo un sketch inmersivo de la sección delantera de la cola (igual que la sección trasera del fuselaje). De esta forma, fue capaz de convertir la imagen de la izquierda de la *figura 4.5* en un sketch (imagen de la derecha de la *figura 4.5*) bastante aproximado a la forma de la sección real.

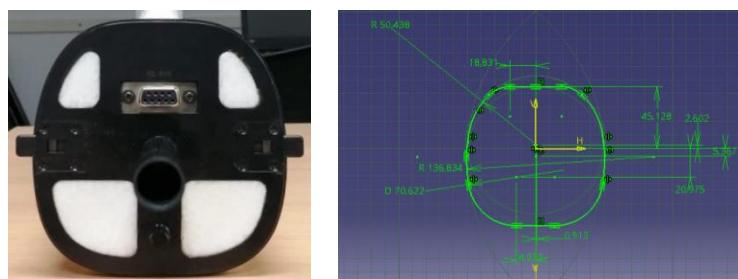


Figura 4.5: Primer sketch inmersivo realizado por Virginia Bueno Gómez.

Aprovechando el “know how” obtenido gracias a Virginia, procedí con la realización de los sketches inmersivos de todas las tapas señaladas en las *figuras 4.1* y *4.2*. Las fotos de las que se disponía de las tapas eran las mostradas en la *figura 4.6*:

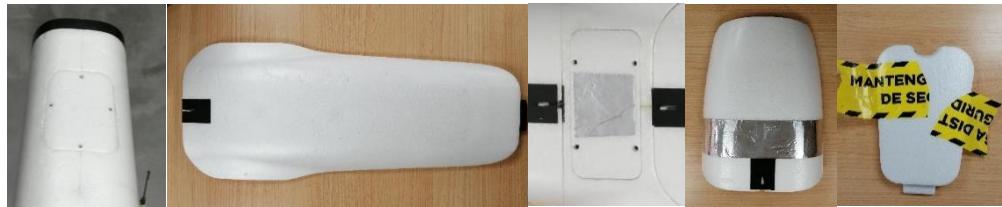


Figura 4.6: Imágenes reales de las tapas del dron mostradas por orden, siendo de izquierda a derecha: 1, 2, 3, 4 e inferior.

De esta forma, el procedimiento aplicado a la tapa 2, por ejemplo, para obtener un sketch bastante aceptable de su silueta es el siguiente.

Ya en el módulo “Sketch Tracer”, primero se debe pulsar el icono de “Top View” (cualquier vista en 2D es válida) y luego se ha de clicar sobre la herramienta “Create an Immersive Sketch” comentada en la metodología. Tras ello se abrirá una pestaña que nos permitirá seleccionar la imagen de, en este caso, la tapa 2. Seleccionamos la imagen y aparecerá el menú mostrado en la *figura 4.7*:

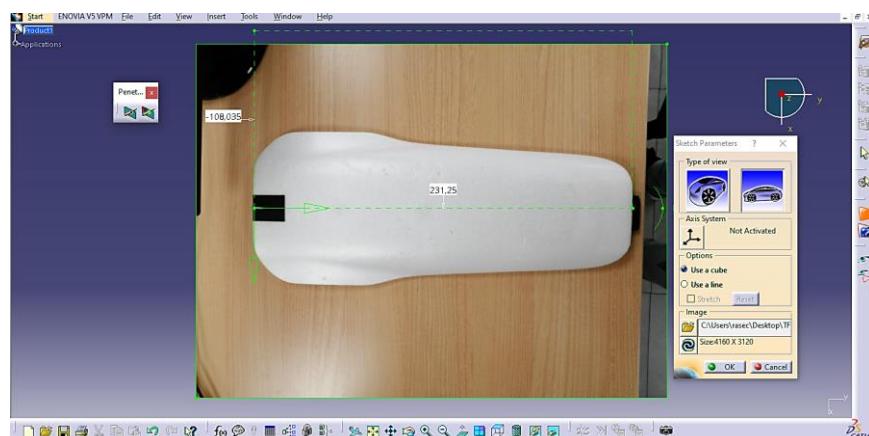


Figura 4.7: Imagen de la tapa 2 como paso previo a la realización del sketch inmersivo.

Como puede verse en la *figura 4.7*, se mantiene la selección “use a cube” y, a continuación, se desplazan los ejes verdes de la figura hasta el origen de coordenadas que sea de nuestro interés. Como puede verse, aparecen dos medidas en los dos ejes verdes mostrados. Dado que de la tapa 2 se tiene - tal y como puede verse en los planos de las *figuras 4.1* y *4.2* - la medida de su longitud (45cm) deberemos hacer que la medida del “eje verde” sobre la longitud de la imagen importada tenga la misma medida. Esta será la referencia para tomar el resto de las medidas sobre la tapa. Haciendo doble clic sobre la medida que aparece por defecto, deberemos escribir los 450mm correspondientes. Tras ello pulsaremos OK en el cuadro abierto, y ya habremos finalizado todas las tareas necesarias en este módulo para hacer sketches inmersivos, quedando la imagen de la *figura 4.8*.

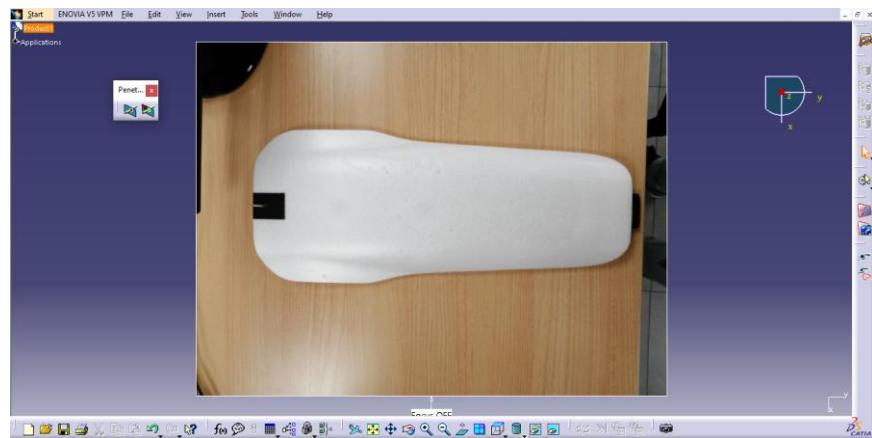


Figura 4.8: Imagen de la tapa 2 tras poner la medida de referencia.

Ahora haremos doble clic sobre “Product1” en el árbol y nos dirigiremos al módulo “Part Design”, donde utilizaremos la herramienta “Sketch”. El plano que debemos seleccionar para hacer el sketch será el mismo que el plano donde se encuentra la imagen colocada, tal y como se puede ver en la siguiente *figura 4.9*.

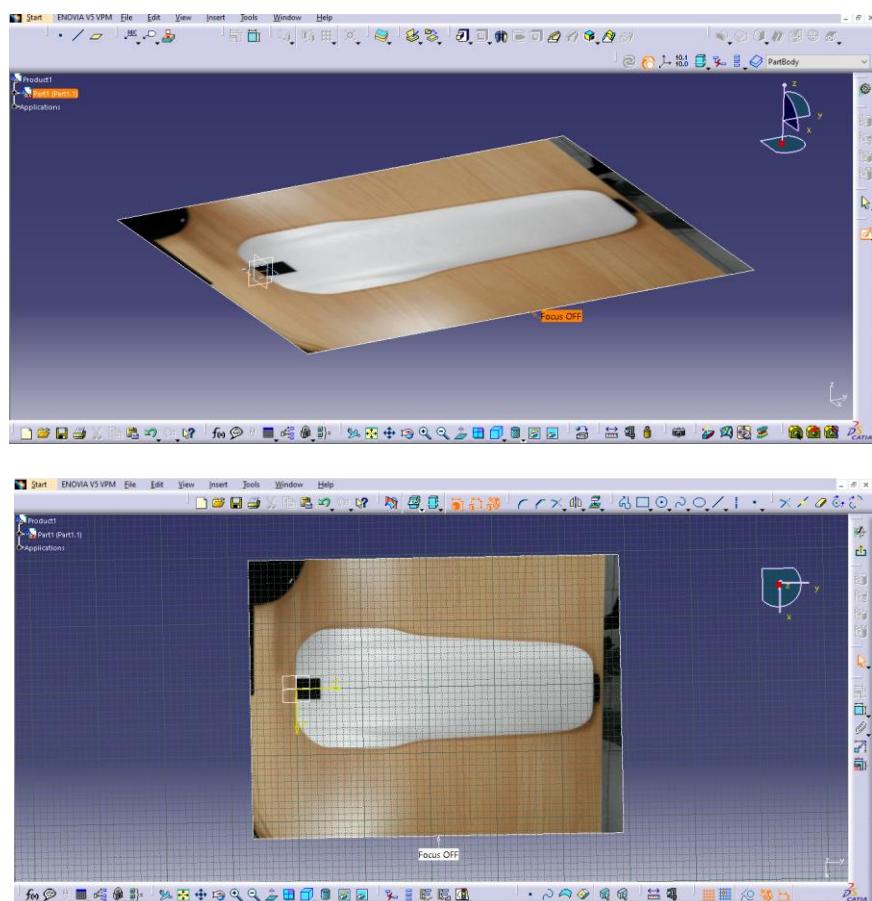


Figura 4.9: Imagen de la tapa 2 en el módulo “Part Design”, antes de finalmente hacer el sketch inmersivo sobre ella.

Sobre este plano colocaremos las rectas, arcos de circunferencia y curvas que sean necesarios, empleando las acotaciones que nos permitan seguir de la manera más acertada posible la silueta de la imagen de la tapa 2.

Tras completar este sketch y volver a la imagen 3D en el “Part Design”, se hace clic derecho sobre la imagen y se borra, obteniendo lo mostrado en la *figura 4.10*.

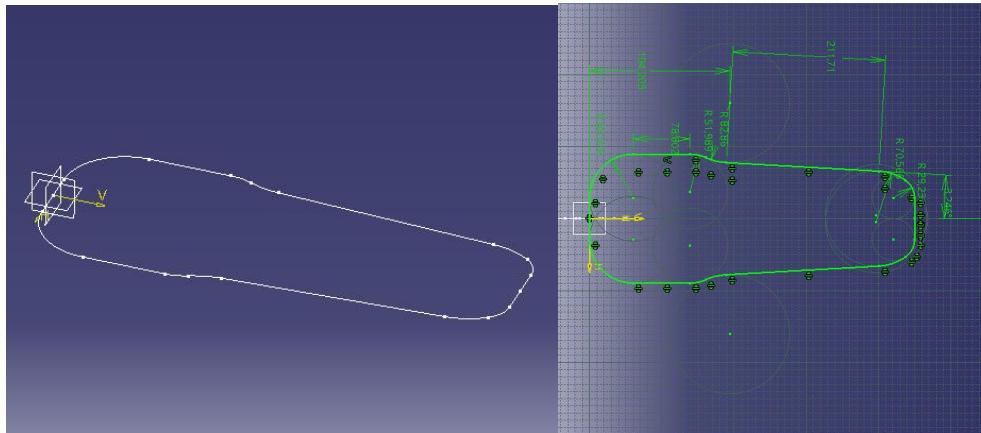


Figura 4.10: Imágenes del sketch inmersivo de la tapa 2 tras haber eliminado su foto.

Aplicando exactamente el mismo procedimiento con la imagen de la tapa 4, se obtiene un sketch bastante aproximado a la silueta real, como se puede ver en la *figura 4.11*.

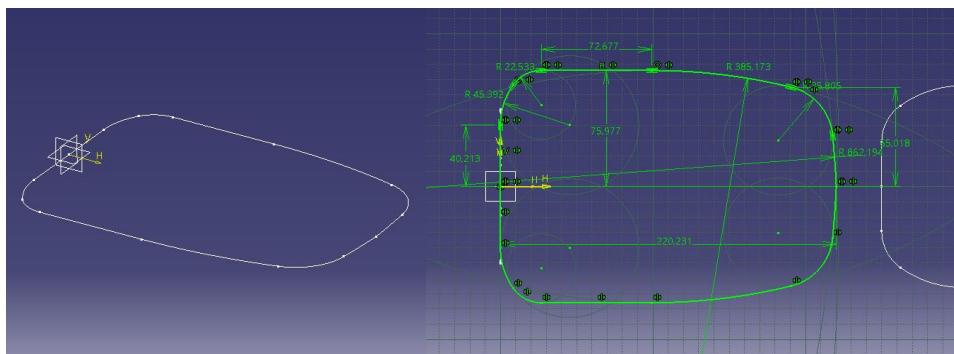


Figura 4.11: Imágenes del sketch inmersivo de la tapa 4 tras haber eliminado su foto.

Y de la misma forma se obtienen los sketches de las tapas 1, 3 e inferior, respectivamente (*figura 4.12*):

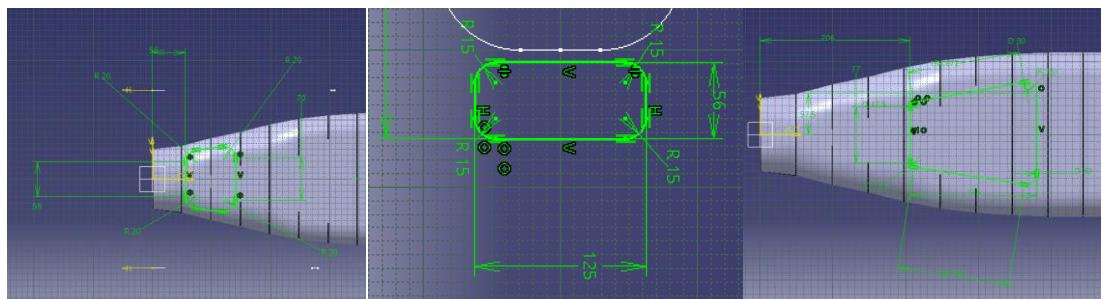


Figura 4.12: Imágenes de los sketches inmersivos de las tapas 1, 3 e inferior tras haber eliminado la imagen modelo y haberlos colocado en el modelo del dron.

4.2.3. Diseño del Fuselaje

El primer detalle a destacar en este apartado es que el diseño completo del fuselaje [82][83][84] se ha realizado única y exclusivamente con el empleo de los módulos “Part Design” y “Generative Shape Design”.

Se recuerda del *Capítulo 3* que el primer intento de hacer el fuselaje fue un intento fallido. En aquel caso se intentó aplicar factores de escala a la sección trasera del fuselaje (cuya foto se puede ver en la *figura 3.7*) para crear algunas de las secciones intermedias, derivando las restantes secciones de la sección delantera, también mediante la aplicación de factores de escala.

Tras aquel primer intento, se concluyó que las secciones del fuselaje eran de sección variable, es decir, las secciones cambiaban suavemente a lo largo de su longitud. No se mantenía la proporción en las medidas (como se había considerado en el primer diseño). Teniendo esto en cuenta, se pudo desarrollar el fuselaje definitivo mostrado en esta la *figura 4.13*.

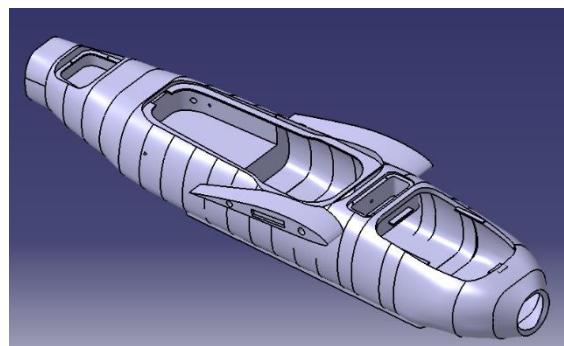


Figura 4.13: Imagen del modelo del fuselaje definitivo.

Los pasos seguidos para conseguir desarrollar con éxito el fuselaje mostrado en la *figura 4.13* son los que se describen en los próximos párrafos.

El primer paso consistió en la realización de un esquema del “esqueleto” del fuselaje a modo de referencia, es decir, se colocaron sketches de partes del fuselaje cuya posición y cuyas medidas eran conocidas con detalle, situando el origen de coordenadas de Catia en la sección trasera del fuselaje. Estos sketches fueron los siguientes: sección trasera, sección intermedia de máxima altura, sección de morro, sketch inmersivo de la tapa 2, sketch inmersivo de la tapa 4 y sketch inmersivo de la tapa 3. Se puede observar este “esqueleto” inicial en la *figura 4.14*, a continuación.

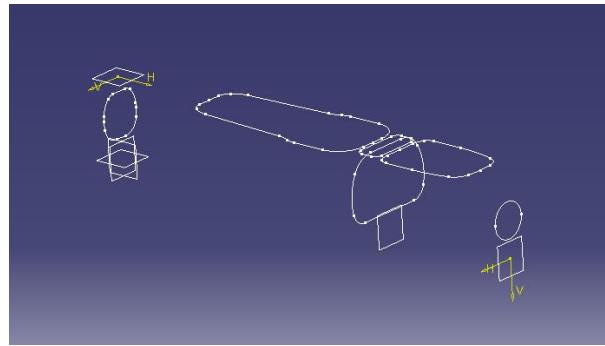


Figura 4.14: Primera imagen del “esqueleto” de referencia del fuselaje.

Respecto a esta figura, se debe comentar que el sketch inmersivo utilizado como sección trasera del fuselaje es el obtenido en la *figura 4.5* por Virginia, pues esta sección trasera del fuselaje coincide con la sección delantera de la cola pues ambas partes del dron deben encajar. Además, el sketch de la sección intermedia de máxima altura y el de la sección de morro del dron se pueden ver en la siguiente *figura 4.15*.

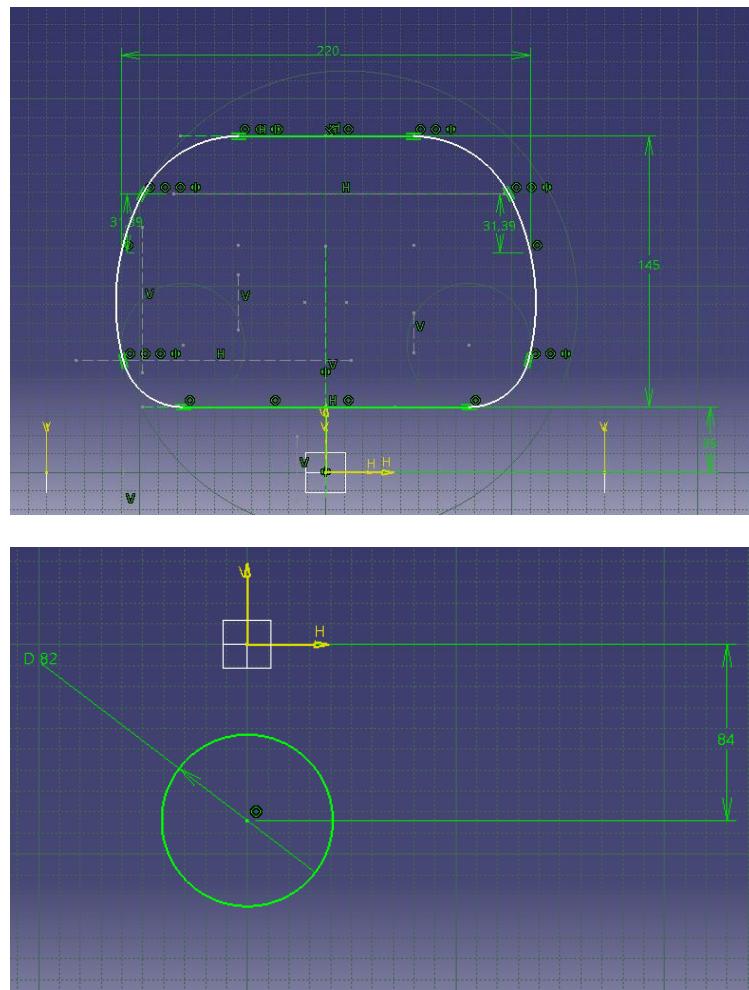


Figura 4.15: Sketch intermedio de la sección de máxima altura y sketch de la sección de morro.

Se debe comentar que el sketch intermedio dibujado (ver imagen superior de la *figura 4.15*) tiene una altura de 145mm y su base se sitúa a 35 mm de altura (el espacio ocupado por los patines), siendo el resultado de su suma igual a la altura máxima del dron indicada por el fabricante, de 18cm. Este sketch se ha realizado utilizando como base el sketch de la sección trasera, deformando sus curvas “a ojo” hasta conseguir el ancho, alto y forma adecuados.

Siguiendo las indicaciones dadas en los planos de la *figura 4.1*, las posiciones donde se colocaron todos estos sketches de la *figura 4.14* se muestra a continuación en la *figura 4.16*.

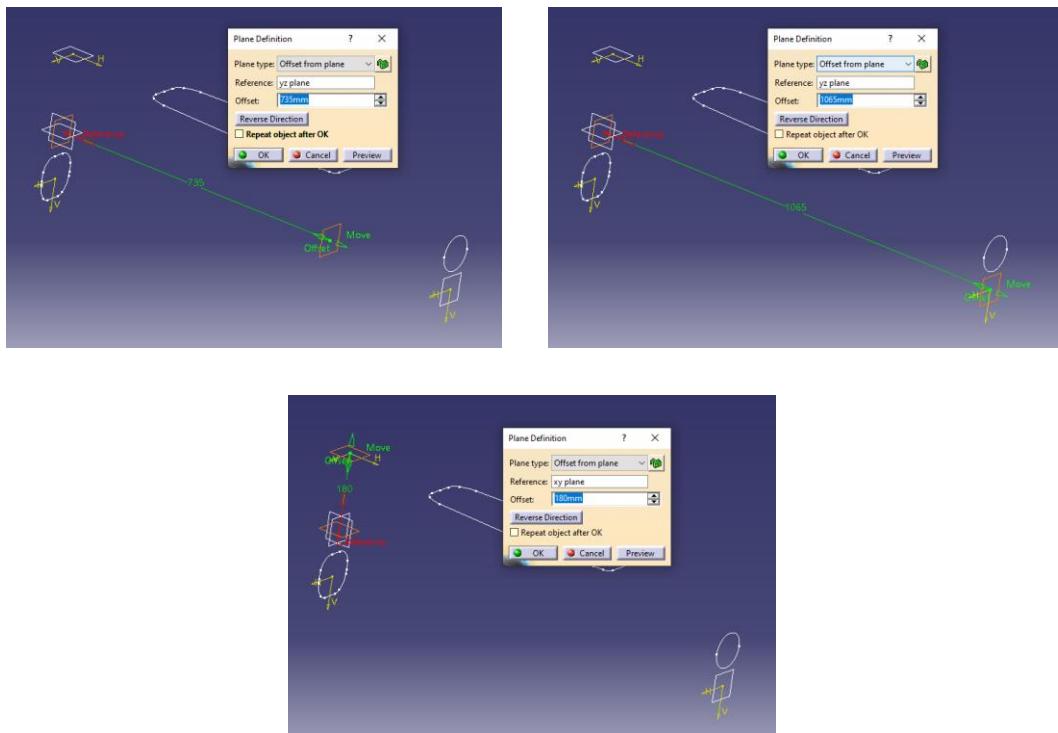


Figura 4.16: Imágenes sobre el posicionamiento (respecto de la sección trasera) de los planos utilizados para hacer los sketches mencionados, respectivamente a 735mm el plano intermedio, a 1065mm el plano del morro y a 180mm el plano de máxima altura.

En las imágenes reales del dron (ver figuras 3.6, 3.7 y 3.8) se observa que este tiene unos pequeños agujeros en el fuselaje para permitir la conexión de algunas antenas externas con el interior, tanto a estribor como a babor. Por este motivo (hacer estos agujeros en pasos posteriores) se incluyen dos sketches a los laterales, tal y como puede verse en la *figura 4.17*.

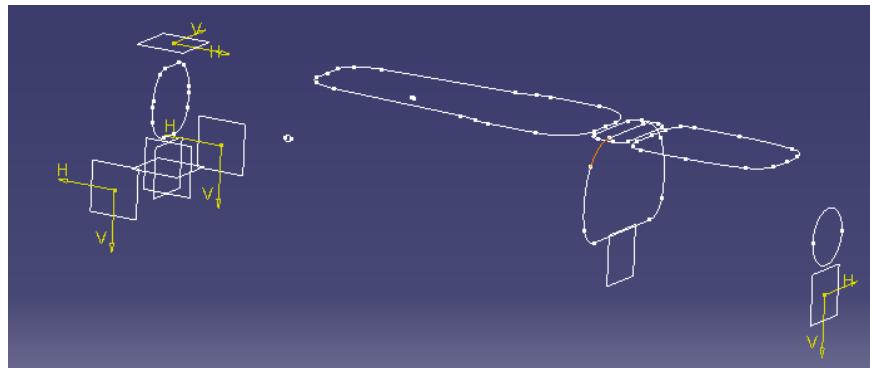


Figura 4.17: Segunda imagen del “esqueleto” de referencia del fuselaje.

En este punto del diseño solo se tienen tres secciones del fuselaje. Para hacer que la unión de las secciones dé como resultado una superficie cuya evolución longitudinal sea “suave”, “lenta” y “gradual”, es necesario introducir muchas más secciones con pequeñas variaciones en sus medidas (“por tanteo”) que sirvan de transición entre la sección trasera y la sección intermedia, y también entre la sección intermedia y la sección de morro. Por ello, se coloca una cantidad considerable de secciones equiespaciadas entre estas tres secciones, pudiéndose ver el “esqueleto” del dron actualizado en la siguiente *figura 4.18*.

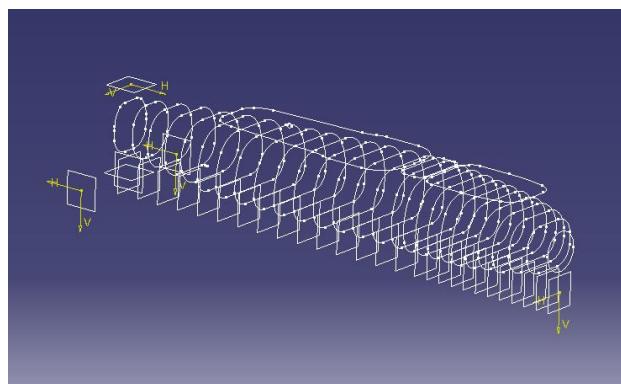


Figura 4.18: Tercera imagen del “esqueleto” de referencia del fuselaje.

Se puede contar que el total de secciones presentes en el dron es de 27 secciones en total. Para referirnos a ellas a partir de ahora, la sección 1 será la sección del morro y la sección número 27 será la sección de cola. Una vez asignada esta nomenclatura a las secciones mostradas en la *figura 4.18*, es necesario aclarar lo siguiente: el equiespaciado empleado entre la sección de morro y la intermedia de referencia es menos que el empleado entre esta última y la sección de cola. Esto se debe, primero, a las diferencias en longitud entre “sección intermedia-sección de morro” y “sección intermedia-sección trasera”, siendo esta última mucho mayor, haciendo innecesario introducir cambios en las secciones de manera tan “continuada”. El segundo motivo por el que se decidió elegir estos equiespaciados es por lo acusado del cambio de forma de las secciones, siendo la variación “sección intermedia-sección de morro” mucho más acusada que la variación “sección intermedia-sección trasera”, siendo necesario introducir muchas más secciones por unidad de longitud que permitan alcanzar una variación suave de las secciones del fuselaje.

En el módulo “Generative Shape Design” se aplica la herramienta “Fill” a las secciones desde la número 9 a la 27 (ver *figura 4.19*), dejando el resto de las secciones sin rellenar.

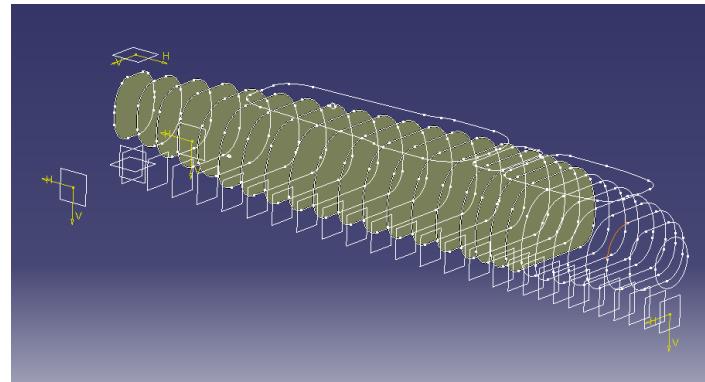


Figura 4.19: Aplicación de la herramienta Fill a las secciones desde la 9 a la 27.

Tras ello se emplea la herramienta “Multi-Sections Surface” entre pares de secciones, es decir, se aplica entre la sección 1 y la 2, luego se aplica entre la 2 y la 3, y así sucesivamente hasta la 27 (ver figura 4.20).

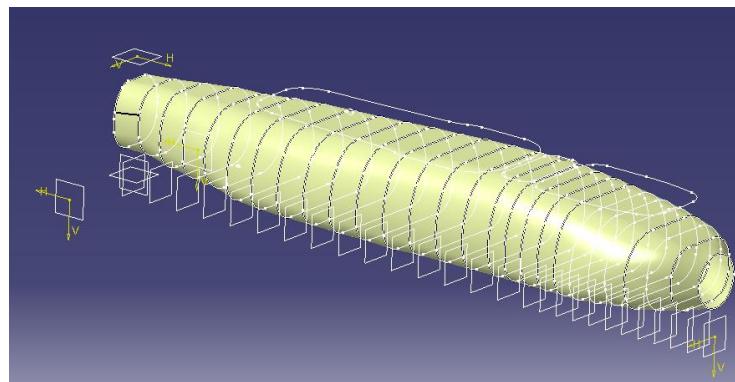


Figura 4.20: Aplicación de la herramienta Multi-Sections Surface dos a dos secciones.

A continuación, se aplica la herramienta “Join” sucesivamente cada 2 “Fill” y 1 “Multi-Sections Surface”, creando “rodajas” (es decir, tramos) de fuselaje. Puede verse un ejemplo de creación de un “Join” en la figura 4.21.

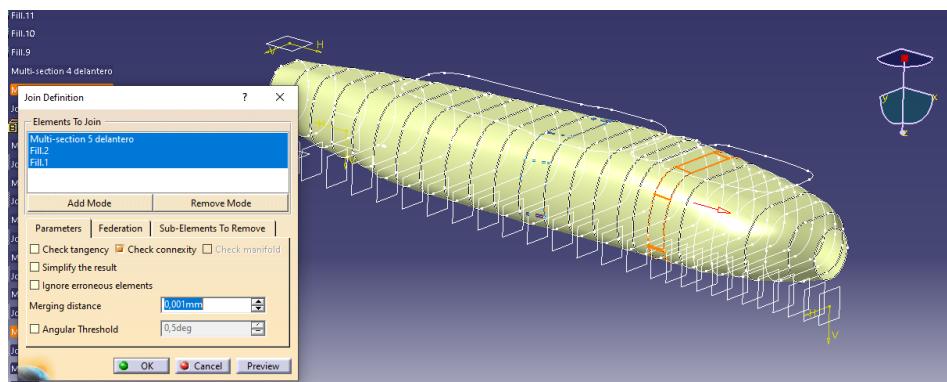


Figura 4.21: Creación de los elementos Join para cada “rodaja” de fuselaje, desde la sección 9 hasta la 27.

Y, además, se procede a crear un “Join” de todas las “Multi-Sections Surface” creadas, desde la sección 1 hasta la 27. Este procedimiento puede verse reflejado en la *figura 4.22*.

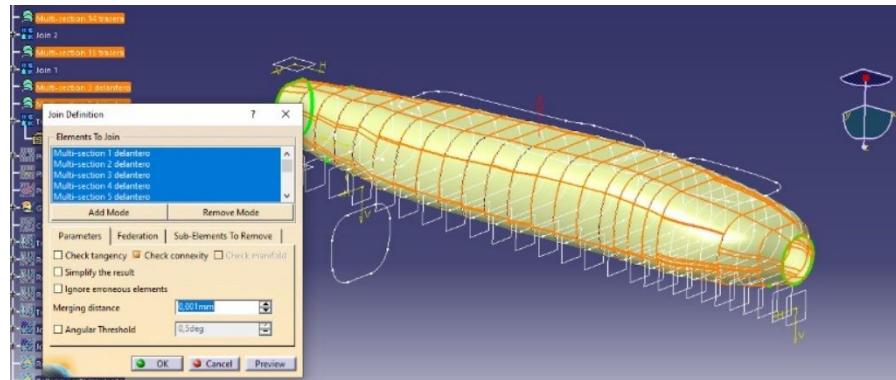


Figura 4.22: Creación del elemento “Join” de todos los elementos “Multi-sections Surface” del fuselaje. Se crea un “revestimiento unificado”.

Finalmente, tras solo haber estado trabajando con superficies, se procede a añadir material al modelo del dron, empleando dichas superficies y los “Join” como referencia para hacerlo. Se utilizan las herramientas del módulo “Part Design”: “Thick Surface” y “Close Surface”. La primera añade un espesor de material a la superficie seleccionada, mientras que la segunda coge una superficie cerrada (por este motivo se hicieron los “Join”) y la rellena completamente de material. Los cuadros de control/selección de estas dos herramientas pueden verse en la *figura 4.23*.

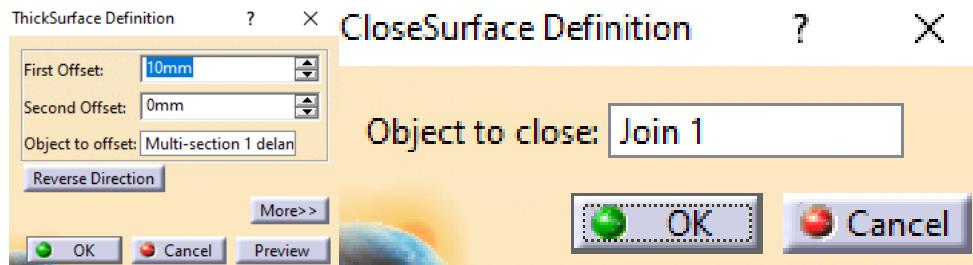


Figura 4.23: Ventanas de selección para definir los tramos de fuselaje llenado de material y los tramos con espesores de material.

Así, tras pulsar el botón OK en cada una de estas ventanas para cada una de las “rodajas” de fuselaje, luego esconder las superficies y además utilizar la herramienta “Pocket” para hacer los agujeros de las antenas comentados en la *figura 4.17*, se obtiene la siguiente *figura 4.24*.

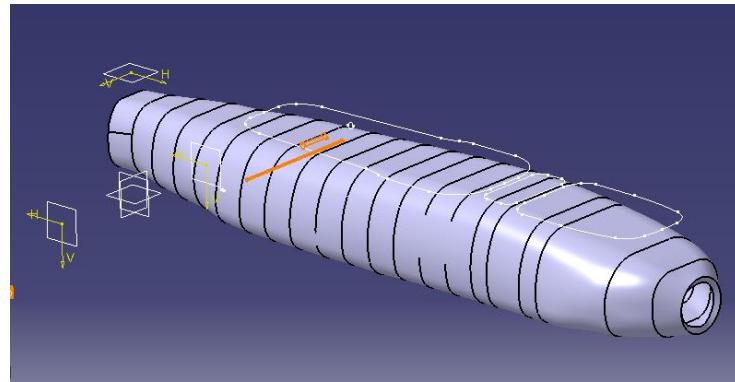


Figura 4.24: Primera imagen del cuerpo sólido del fuselaje

En este punto del diseño, ya puede observarse lo bondadosa de la estrategia de diseño seguida, pues este cuerpo del fuselaje sí tiene un gran parecido con el fuselaje real del dron base (ver *figuras 1.2 y 3.6* ha modo comparativo). A continuación, los próximos pasos del diseño se centrarán en el “detalle” del diseño del fuselaje. Se añadirán los huecos, recovecos, agujeros y otras formas que caracterizan el fuselaje de este VTOL. Por ello, el primer paso será hacer los huecos para alojar las tapas del dron mediante la herramienta “Pocket”. También, en el caso de las tapas 1 e inferior, se empleará esta herramienta para “vaciar” el material del dron, creando los espacios de carga que dichas tapas guardan.

Así, pueden observarse en la *figura 4.25* los huecos obtenidos de las tapas 2 y 4:

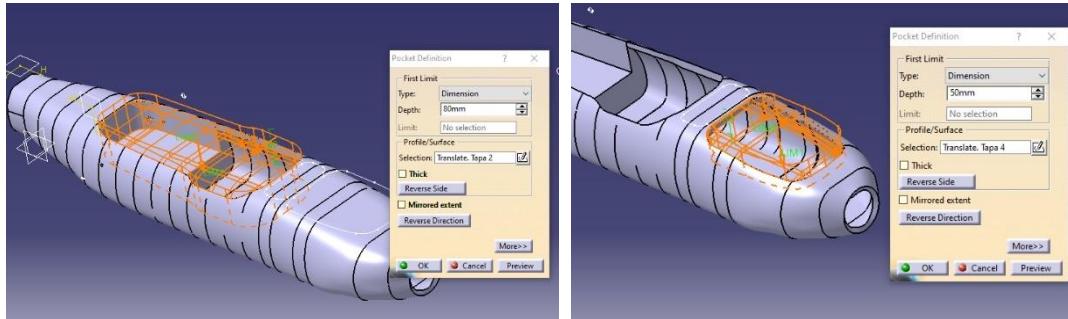


Figura 4.25: Empleo de la herramienta “Pocket” para hacer los huecos de las tapas 2 y 4.

Los vaciados de material correspondientes a las posiciones de la tapa 1 y la tapa inferior, empleando también la herramienta “Pocket” se pueden observar en la *figura 4.26*, a continuación:

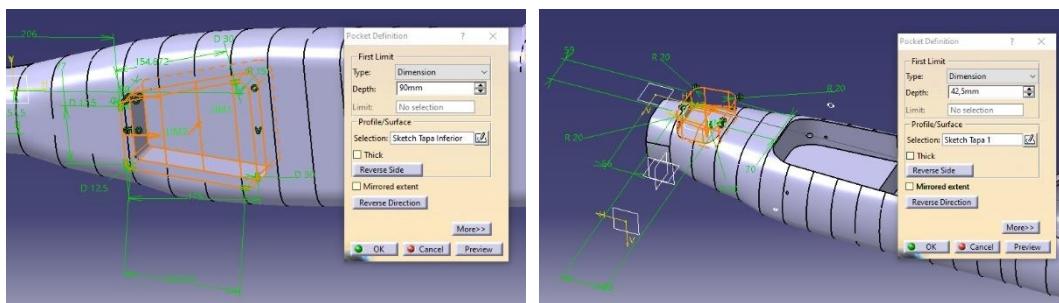


Figura 4.26: Empleo de la herramienta “Pocket” para hacer los vaciados de la tapa 1 y de la tapa inferior.

Empleando la herramienta “Multi-Pocket”, se modela la sección trasera del fuselaje, tal y como se ve en la siguiente imagen (*figura 4.27*):

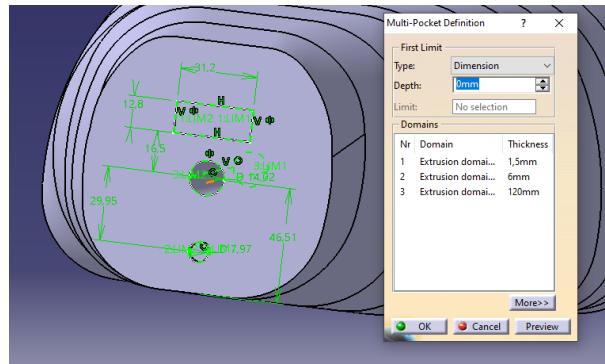


Figura 4.27: Empleo de la herramienta “Multi-Pocket” para configurar la sección trasera del fuselaje.

Por otro lado, para crear el espacio de carga cubierto por la tapa 3, se ha seguido el siguiente procedimiento.

Se utiliza el sketch inmersivo de la tapa 3 para crear un sketch de las mismas proporciones, aumentando este en un determinado factor de escala. Este sketch es el colocado en la parte inferior del “cajón” que se desea crear. Tras aplicar un “Pad” a este sketch para crear este “bloque de material”, puede verse el resultado de su aplicación en la *figura 4.28*. El siguiente paso será aplicar un “Pocket” al sketch inmersivo de la tapa 3, es decir, el sketch situado por encima del cajón, pudiendo ver el resultado de este “Pocket” en la *figura 4.29*.

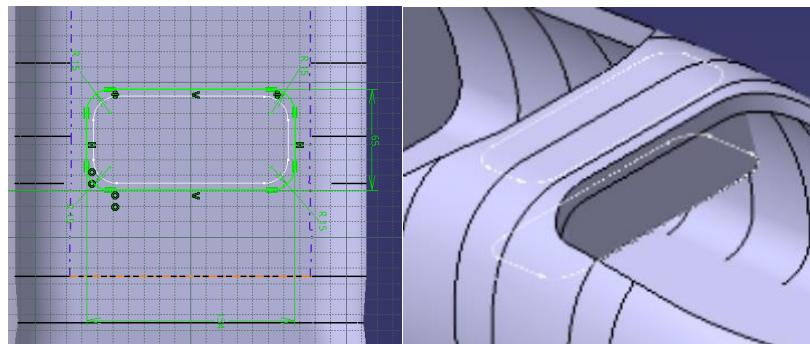


Figura 4.28: Primer paso (“Pad”) en la creación del espacio de carga cubierto por la tapa 3.

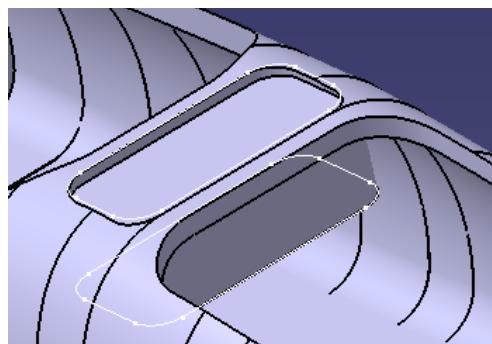


Figura 4.29: Segundo paso (“Pocket”) en la creación del espacio de carga cubierto por la tapa 3.

Ahora se emplea otro sketch de igual proporción que el sketch inmersivo de la tapa 3, esta vez con un factor de escala menor que 1. Se utiliza este sketch para, mediante un “Pocket” poder vaciar el material del cajón, como puede verse en la *figura 4.30*.

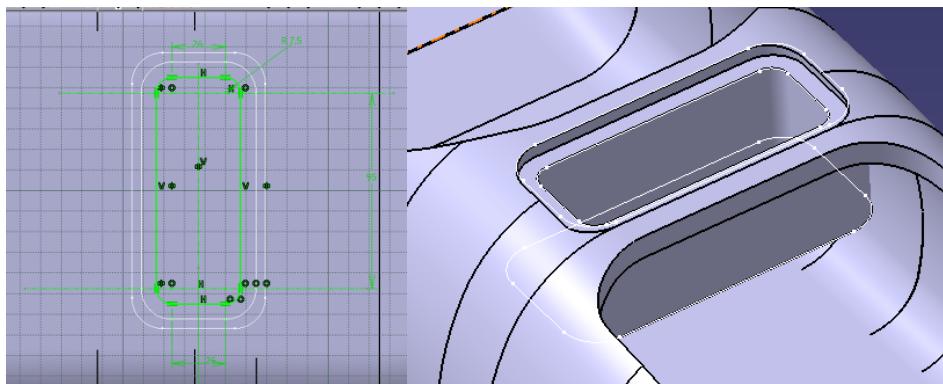


Figura 4.30: Tercer paso (“Pocket”) en la creación del espacio de carga cubierto por la tapa 3.

Por último, se añaden cuatro agujeros necesarios para poder atornillar la tapa 3 al fuselaje más delante (ver *figura 4.31*). También se añade un pequeño orificio que permite pasar cables desde el interior de este cajón al resto del fuselaje (ver *figura 4.32*).

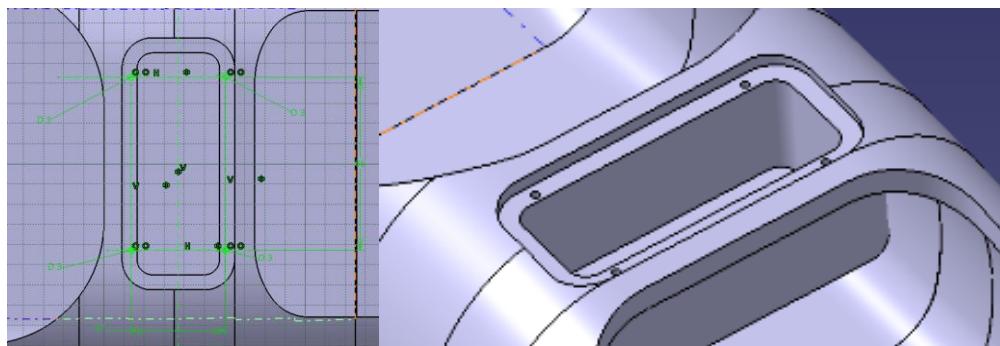


Figura 4.31: Realización de agujeros para tornillos mediante la herramienta “Pocket”.

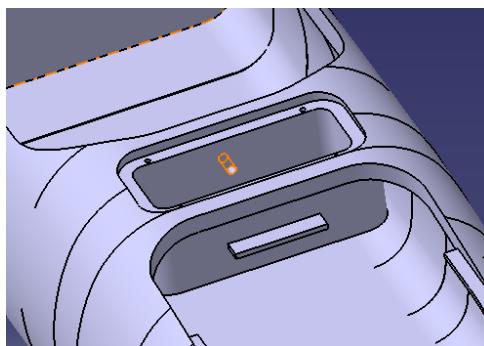


Figura 4.32: Resultado de la creación del orificio del cajón diseñado, también mediante el uso de la herramienta “Pocket”.

Una vez finalizado el cajón de la tapa 3, se procede a hacer la estructura del fuselaje que se unirá a los patines del tren de aterrizaje. Para ello se crea el sketch de los patines, se hace un “Pad” del mismo y se hacen dos agujeros en uno de los lados (con un “Pocket”), donde encajarán los ejes del patín correspondiente. Por último, se hace el simétrico de estos agujeros, añadiéndose estos al otro lado del fuselaje. Estas imágenes de la *figura 4.33* recogen lo descrito en este párrafo:

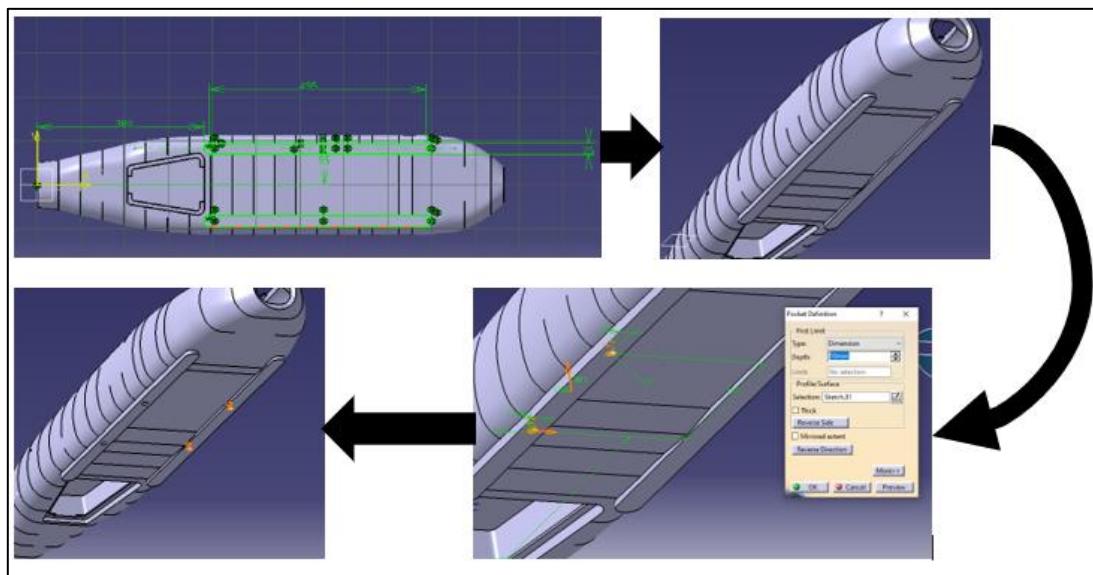


Figura 4.33: Procedimiento de creación de la estructura en que encajarán los patines.

Por otro lado, en el hueco inferior del fuselaje (cubierto por la tapa inferior) se deben construir las siguientes estructuras con el fin de asegurar un correcto acople de la tapa inferior en el lugar. Por ello, se deben añadir dos estructuras: un hueco para introducir la pestaña de la tapa y una estructura soporte para la tapa.

Con el fin de hacer el hueco donde se alojará la pestaña de la tapa inferior, se crea un sketch rectangular en la pared donde se encuentra este hueco. Tras ello, se aplica un “Pocket” a dicho sketch y se obtiene el resultado mostrado en la *figura 4.34*. Después, se debe crear la estructura soporte de la tapa, para lo cual se crea un sketch basado en el sketch inmersivo de la tapa inferior al cual se le hacen varias acomodaciones para obtener el resultado más parecido posible al real. Se hace un “Pad” de dicho sketch y se obtiene el resultado definitivo de este espacio de carga, ya preparado para poder colocar la tapa inferior. El proceso de creación de estas estructuras se puede observar en las siguientes imágenes, recogidas en la *figura 4.34*.

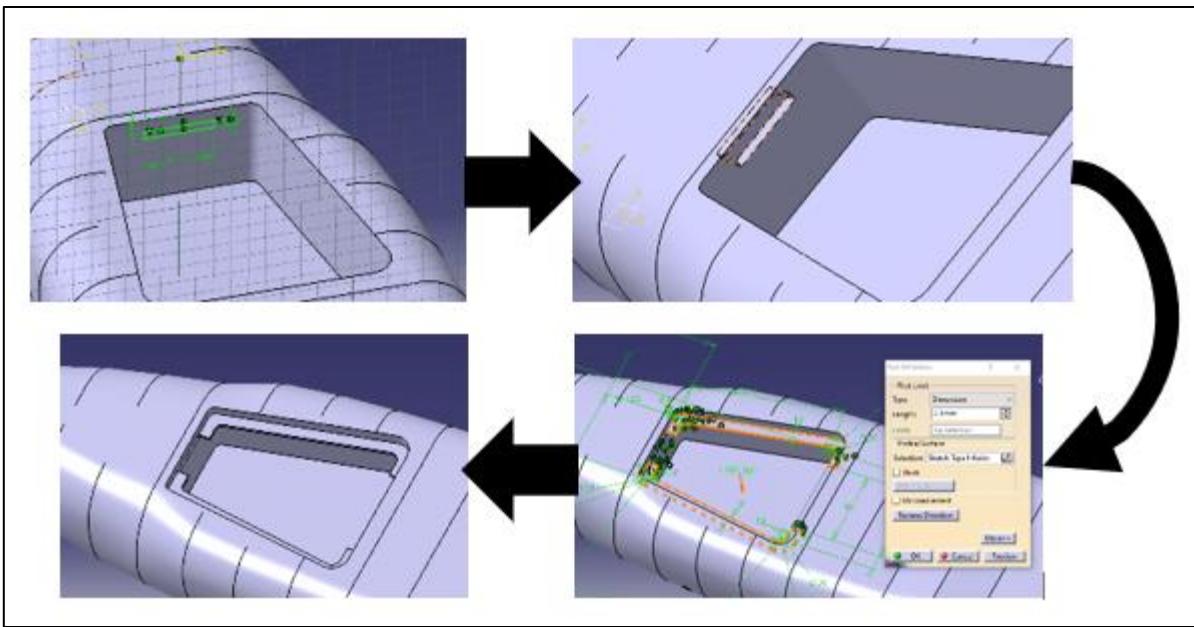


Figura 4.34: Proceso de diseño del espacio de carga de la tapa inferior.

Ahora hablaremos sobre las modificaciones a introducir en el espacio de carga cubierto por la tapa 1. De forma similar al caso anterior, en este caso también se debe colocar una estructura que sirva de apoyo para esta tapa, tras lo cual se harán los agujeros de los tornillos indicados.

El proceso seguido para realizar estas modificaciones en el espacio de carga de la tapa 1 ha seguido un proceso distinto al mostrado en la *figura 4.34*. Como puede observarse en la *figura 4.26*, el “Pocket” aquí realizado se encargó de comer la cantidad de material necesaria hasta llegar a la cara superior del borde empleado de apoyo. Ahora se realizan dos “Pocket”, el primero para eliminar el material sobrante que permitirá crear el espacio de carga definitivo (se emplea un sketch reducido del sketch inmersivo de la tapa 1, con un factor de escala menor que 1), y el segundo para eliminar el material sobrante que quedará por debajo de la pestaña buscada (se utiliza un sketch igual al sketch inmersivo de la tapa 1, situándolo por debajo de la pestaña). Por último, se hacen los agujeros para atornillar la tapa al fuselaje, también utilizando un “Pocket”. Se puede observar en la *figura 4.35* un esquema del proceso seguido para ajustar este espacio de carga.

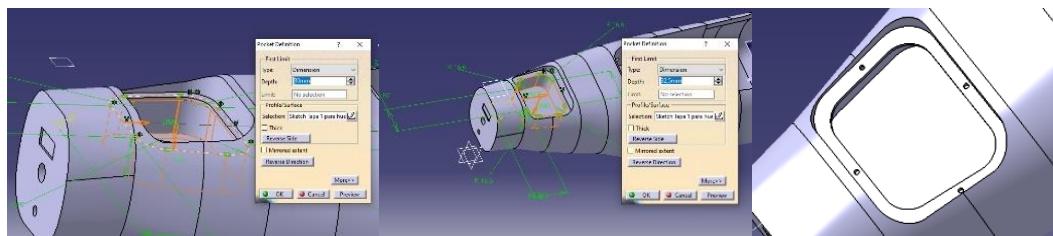


Figura 4.35: Esquema del proceso seguido para diseñar el espacio de carga de la tapa 1.

Para diseñar el espacio de carga cubierto por la tapa 2 también se deben incluir tanto un hueco para introducir la pestaña de seguridad de la tapa como un apoyo simple que permita acoplar bien la tapa en el espacio destinado. El procedimiento seguido para hacer este diseño es similar al descrito en casos anteriores, aplicando las mismas ideas y técnicas. En la *figura 4.36* se muestran los resultados alcanzados para el espacio de la tapa 2:

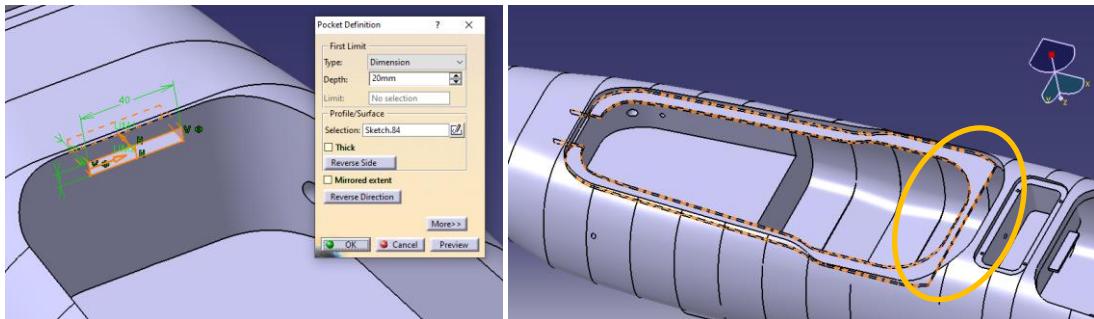


Figura 4.36: Esquema del proceso seguido para diseñar el espacio de carga de la tapa 2.

Destaca el hecho de que el borde de apoyo de la tapa 2 tuvo que ser recortado en las cercanías a la tapa 3, como puede observarse gracias a la marca incluida en la *figura 4.36*. Este hecho tuvo lugar porque el tubo soporte de borde de ataque invadió el espacio del borde de apoyo, y fue necesario recortarlo.

Por último, en cuanto a la tapa 4, en el fuselaje se deben añadir unas pequeñas solapas que sirvan de apoyo a la tapa. Lograr hacer un apoyo como el real (deformado con la estructura) se consideró como un proceso bastante complejo de realizar debido a la forma del fuselaje, sin reportar tampoco un mayor beneficio para los fines de este trabajo tener uno u otro apoyo. Se optó entonces por utilizar unas simples pestañas que, en definitiva, cumplieran con la misma función. De esta forma, primero se añadieron las cuatro pestañas necesarias y luego se incluyó un agujero en la parte más cercana al morro para permitir el encaje de la pestaña de la tapa 4. Puede observarse este proceso en la *figura 4.37*:

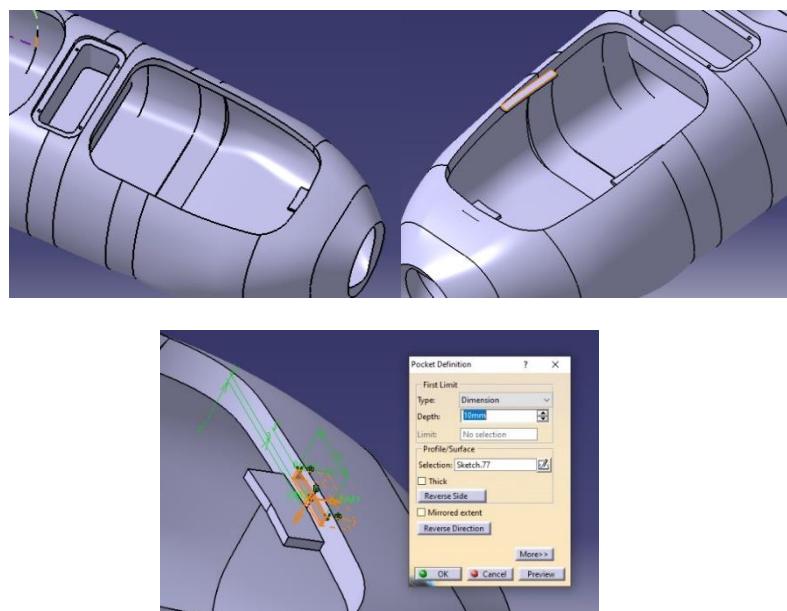


Figura 4.37: Esquema del proceso seguido para diseñar el acople de la tapa 2 al fuselaje.

4.2.4. Diseño de la Tapa 1

La forma de proceder para diseñar cada una de las tapas es bastante similar para todas ellas. En todas se parte de la *figura 4.22* pues, una vez llegado ese punto de diseño del fuselaje, se procedió a realizar copias del archivo “.CATPart” con el fin de poderlo usar como base para hacer las cinco tapaderas del dron. Es decir, la *figura 4.22* será el punto de partida utilizado para diseñar las tapas de este VTOL. Si se “esconden” los elementos sobrantes de dicha figura, se obtiene la *figura 4.38* aquí mostrada:

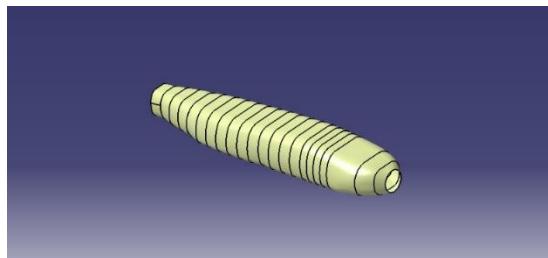


Figura 4.38: Punto de partida del diseño de todas las tapas del VTOL.

A continuación, se empleará la herramienta “Thick Surface” para dar espesor a los tramos del fuselaje correspondientes a la posición longitudinal de la tapa correspondiente. En el caso de la tapa 1, se aplica esta herramienta en los tramos de fuselaje que se pueden ver en la *figura 4.39* aquí mostrada:

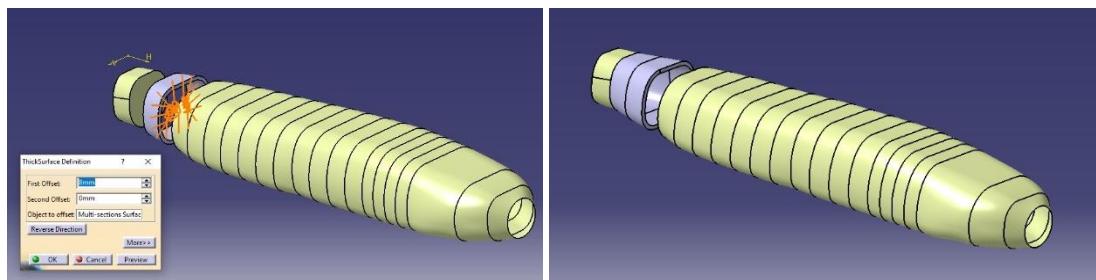


Figura 4.39: Aplicación de la herramienta “Thick Surface” a la zona de interés.

Una vez completado este paso, se esconden las superficies no necesarias en la creación de esta tapa, obteniéndose la *figura 4.40*:

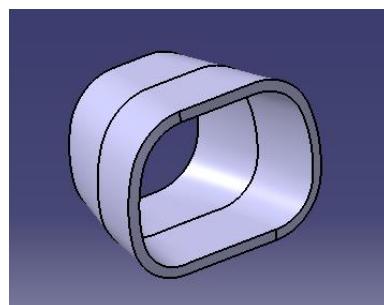


Figura 4.40: Tramos del fuselaje que contienen a la tapa 1.

A continuación, se utiliza el sketch de la tapa 1, situado en un plano superior a la pieza

obtenida en la *figura 4.40*, obteniéndose la *figura 4.41*:

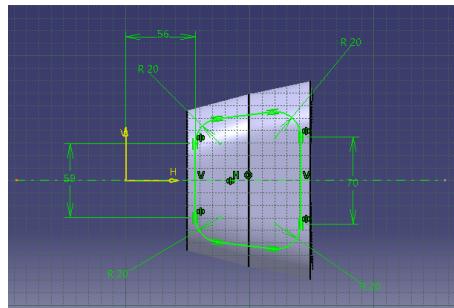


Figura 4.41: Colocación del sketch de la tapa 1 sobre los tramos del fuselaje.

Tras este paso, el siguiente objetivo es eliminar el material sobrante de la pieza. Para ello se emplea la herramienta “Pocket” dos veces, con el fin de eliminarlo, tal y como se puede observar en las siguientes *figuras 4.42* y *4.43*:

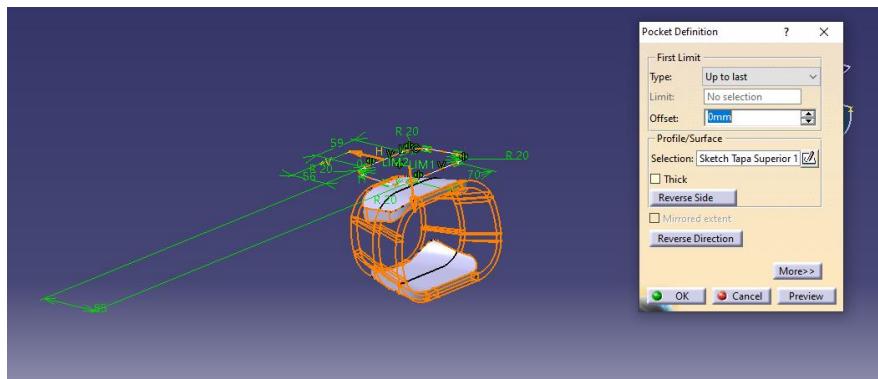


Figura 4.42: Primer paso en la eliminación de material sobrante de la tapa 1.

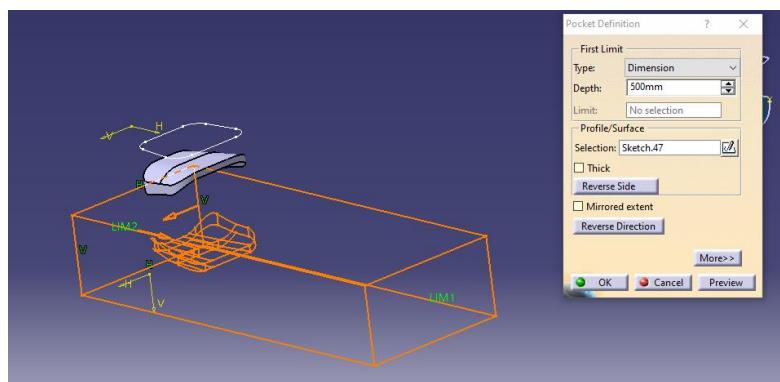


Figura 4.43: Segundo paso en la eliminación de material sobrante de la tapa 1.

En este punto, casi se ha conseguido obtener la forma definitiva de la tapa 1. Lo siguiente es dotar a dicha tapa de unas superficies de apoyo planas (con el fin de poderse apoyar la tapa sobre el fuselaje), para lo cual también se utiliza la herramienta “Pocket”, como se ve en la *figura 4.44*:

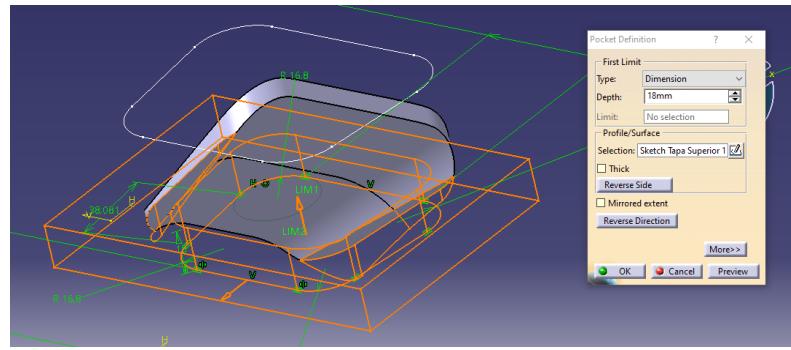


Figura 4.44: Creación de dos de los apoyos planos de la tapa 1.

Tras hacer esto, se añaden a la tapa dos apoyos más, para lo cual se emplea la herramienta “Pad” (ver *figura 4.45*). Además, se realizan cuatro perforaciones en la tapa para permitir el paso de los tornillos que sujetan dicha tapa al fuselaje, utilizando también la herramienta “Pocket” (ver *figura 4.46*).

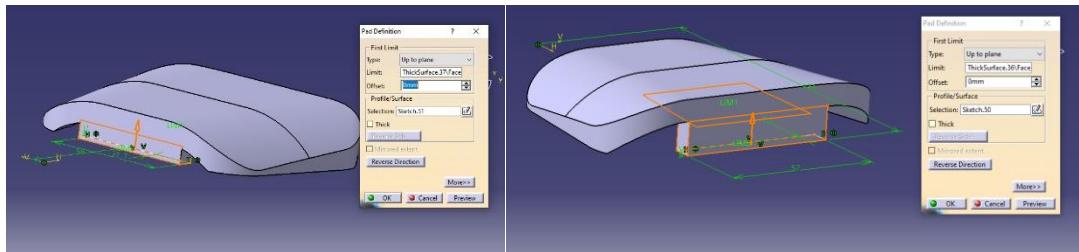


Figura 4.45: Creación de los otros dos apoyos planos de la tapa 1.

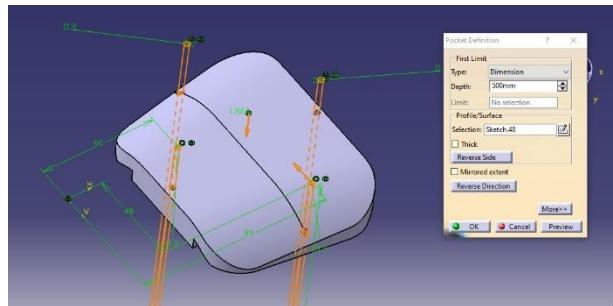


Figura 4.46: Realización de los agujeros para tornillos con la herramienta “Pocket”.

Habiendo realizado ya todas las operaciones necesarias para elaborar el modelo 3D de esta tapa, en la *figura 4.47* se muestra una imagen de la tapa 1 completada.

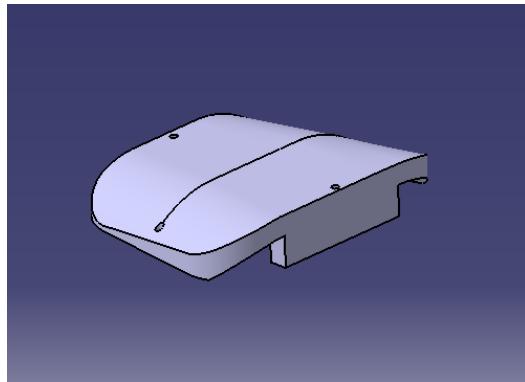


Figura 4.47: Tapa 1 completada.

4.2.5. Diseño de la Tapa 2

Aquí, igual que en el diseño de la tapa anterior, se parte de la *figura 4.38* para realizar este diseño. Sin embargo, los tramos de fuselaje a los que se aplica la herramienta “Thick Surface” son distintos, siendo en este caso los mostrados en la siguiente *figura 4.48*.

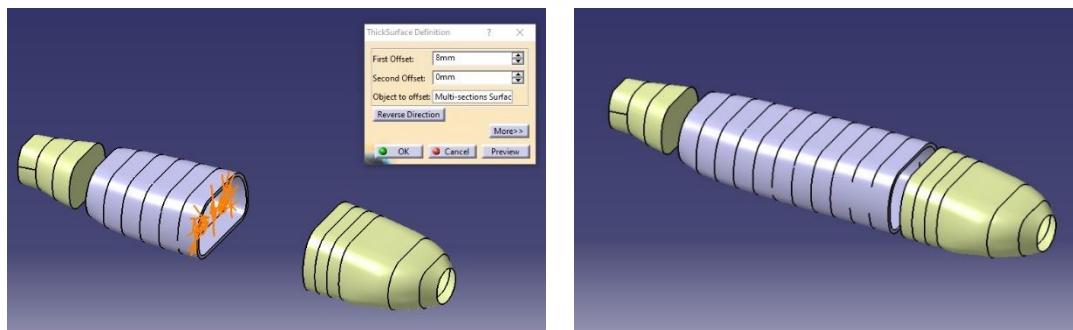


Figura 4.48: Aplicación de la herramienta “Thick Surface” a la zona de interés de la tapa 2.

Del mismo modo, tras esconder las superficies sobrantes, se obtiene lo mostrado en la *figura 4.49*.

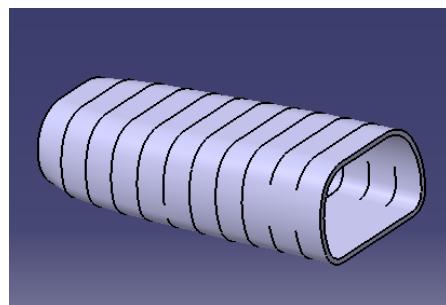


Figura 4.49: Tramos del fuselaje que contienen a la tapa 2.

Tras este proceso, se elimina el material sobrante de la misma forma que se hizo para la tapa 1. Para ello, en este caso, se debe utilizar el sketch immersivo de la tapa 2 que se mostró en la *figura 4.10*. Se muestra en la *figura 4.50* el procedimiento seguido para quitar el material sobrante, efectivamente, un procedimiento igual al seguido para la tapa 1.

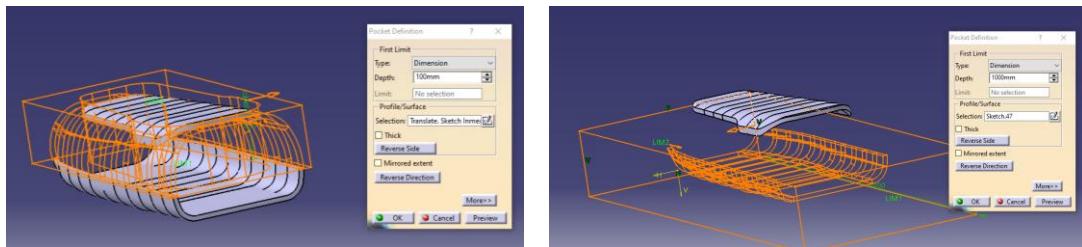


Figura 4.50: Pasos en la eliminación de material sobrante de la tapa 2.

Tras esto, se añade a la tapa una pestaña de anclaje al fuselaje, tal y como puede verse en la *figura 4.51*.

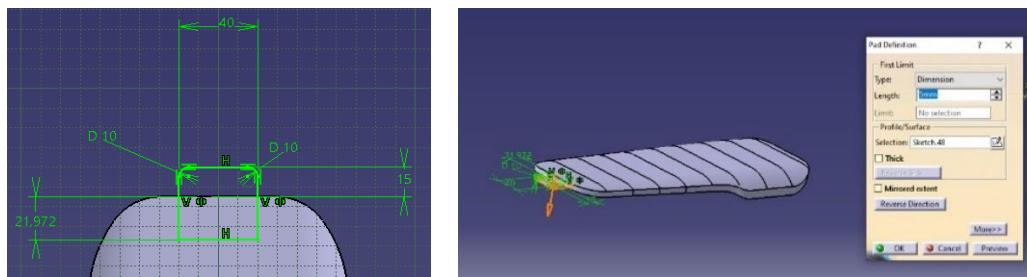


Figura 4.51: Creación de una pestaña de unión al fuselaje.

Ahora se deben incluir las superficies de apoyo planas a esta tapa, permitiendo su correcto apoyo sobre las pestañas del fuselaje. Para lograr esto, se utiliza al sketch inmersivo de esta tapa número 2 como base para crear otro sketch similar (ver *figura 4.52*), que servirá para lograr este objetivo.

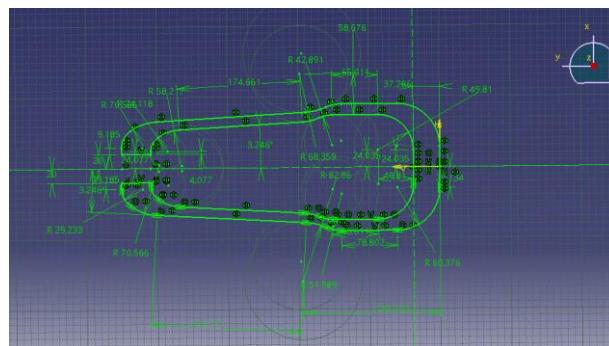


Figura 4.52: Sketch inmersivo de la tapa 2 modificado para crear la superficie de apoyo de la tapa 2.

Con el sketch mostrado en la *figura 4.52* se aplica la herramienta “Pocket” y se obtiene el resultado mostrado en la siguiente *figura 4.53*.

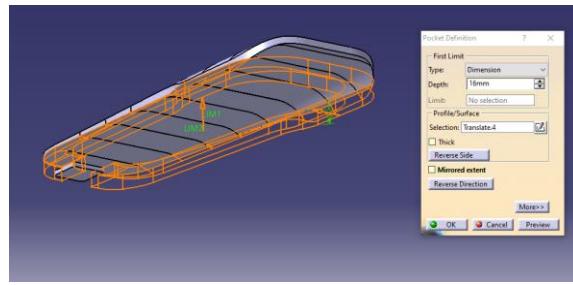


Figura 4.53: Obtención de una superficie de apoyo sobre el fuselaje para la tapa 2.

Finalmente, el resultado final del modelo 3D de la tapa 2 es el que se muestra en la *figura 4.54*.

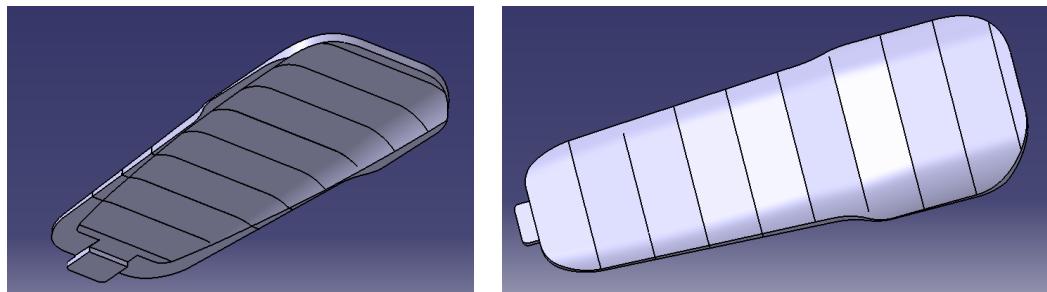


Figura 4.54: Dos vistas del resultado final de la tapa 2.

4.2.6. Diseño de la Tapa 3

En el diseño de esta tapa, todos los primeros pasos son iguales que los seguidos para las tapas anteriores. Por este motivo, se muestra un resumen auto explicativo de imágenes (*figura 4.55*) de los pasos iniciales del diseño de esta tapa.

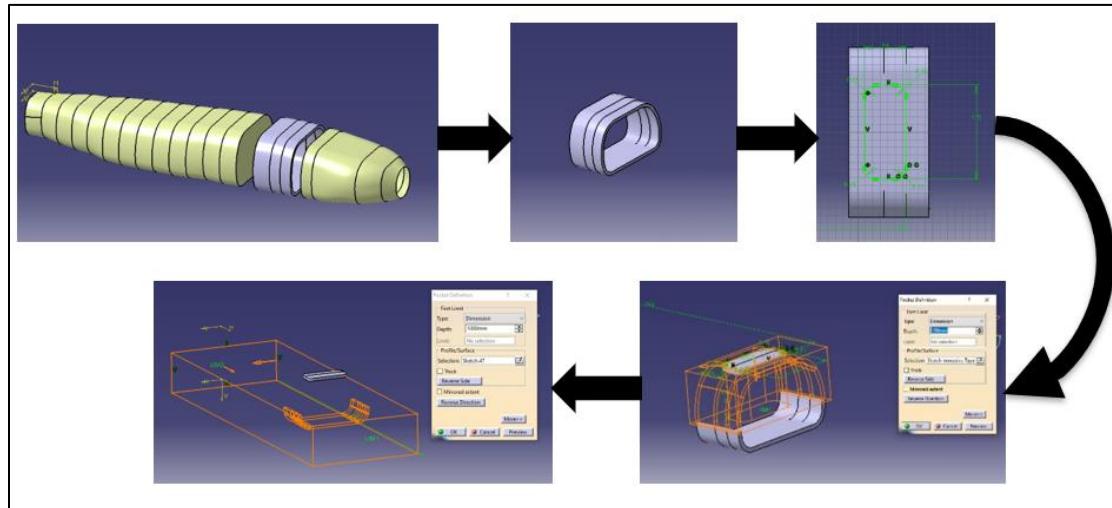


Figura 4.55: Primera parte del proceso de diseño de la tapa 3.

A continuación, se hacen cuatro taladros (herramienta “Pocket”) por los que pasarán los tornillos de sujeción de la tapa al fuselaje. El sketch empleado y la aplicación del “Pocket” se muestran en la siguiente *figura 4.56*.

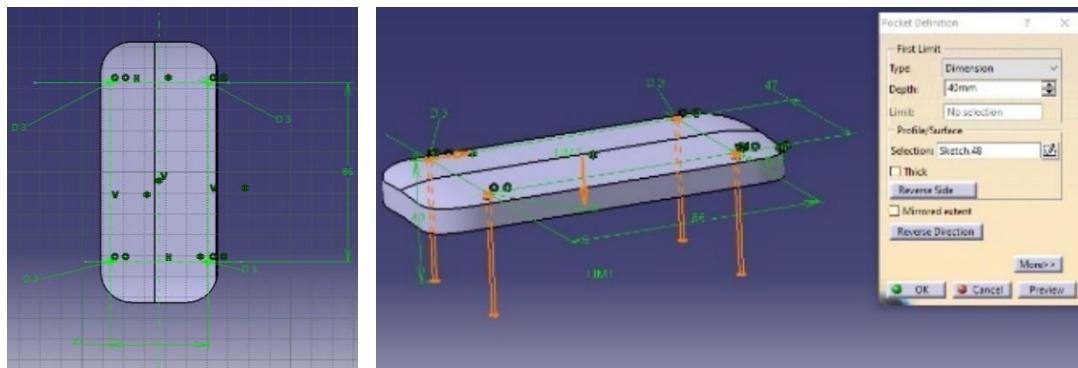


Figura 4.56: Taladros de la tapa 3.

Por último, se retira material de la base de la tapa mediante un “Pocket” de tal forma que el contorno de la tapa sirva como superficie de apoyo al colocarse esta en el fuselaje. Ver *figura 4.57*.

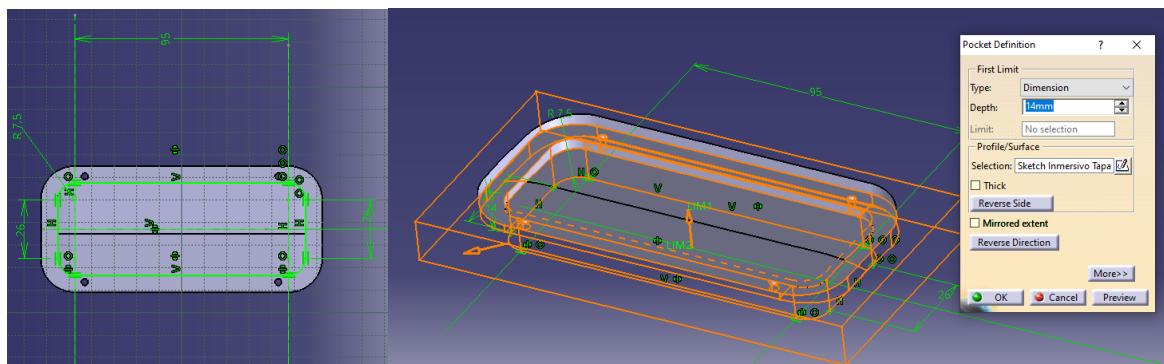


Figura 4.57: Creación del apoyo plano para la tapa 3.

Finalmente, se llega al resultado final logrado, mostrado en la *figura 4.58*, a continuación:

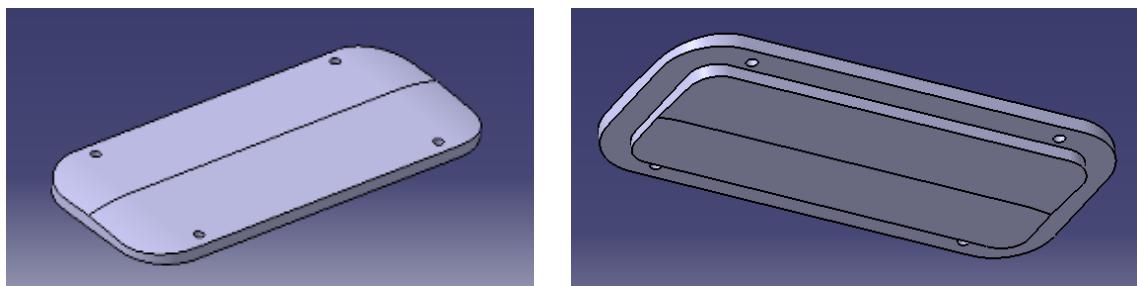


Figura 4.58: Resultado final de la tapa 3.

4.2.7. Diseño de la Tapa 4

De la misma forma que se ha hecho con las fases iniciales de la tapa 3, a continuación, se muestra un resumen visual (ver *figura 4.59*) de las primeras etapas del proceso de creación de la tapa 4.

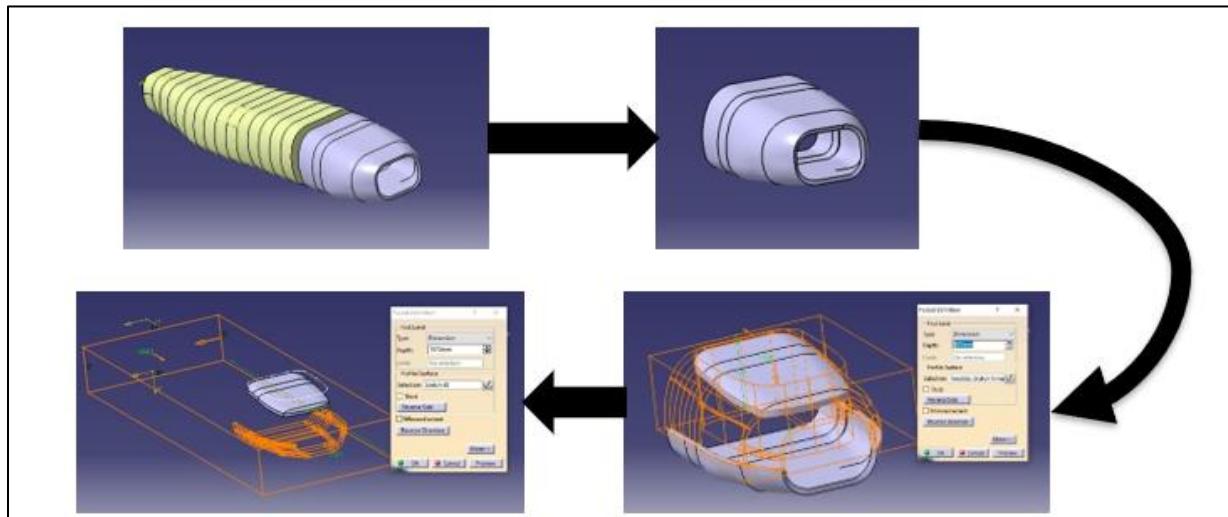


Figura 4.59: Primera parte del proceso de diseño de la tapa 4.

Tras estos pasos, se utiliza el sketch inmersivo de la tapa 4 (mostrado en la *figura 4.11*) para diseñar las superficies de apoyo de la tapa. Se modifica este sketch y se crean cuatro “patas”, mediante la herramienta “Multi-Pad” que servirán para apoyar la tapa en el fuselaje. En la siguiente imagen (*figura 4.60*) se muestran el sketch modificado y una imagen de la aplicación del “Multi-Pad” a dicho sketch.

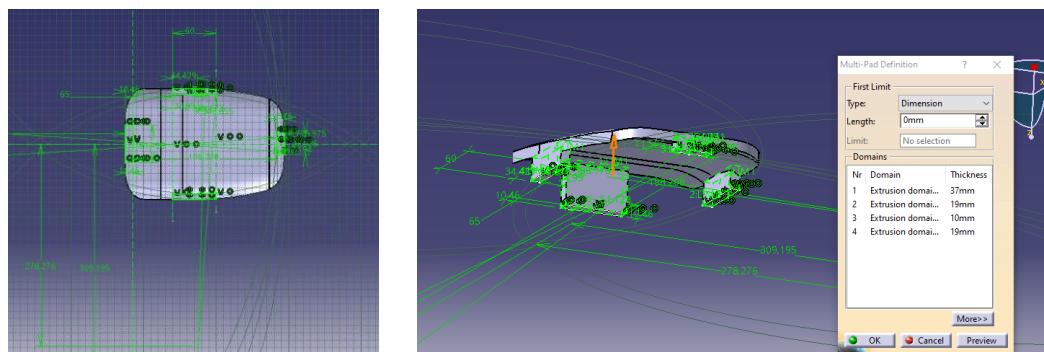


Figura 4.60: Creación de los apoyos de la tapa 4 para el fuselaje.

Si se observan las fotografías y planos del dron mostrados se observa que esta tapa no se coloca paralelamente al suelo, sino que, debido a la variación de las secciones del fuselaje en esa zona, esta tapa forma un cierto ángulo con el suelo. Es por esto por lo que se debe apoyar cada una de las “patas” a diferentes alturas del suelo. Por este motivo se aplicó un “Multi_Pocket” a las patas creadas, consiguiendo elevar el contacto del apoyo lo necesario con el fin de garantizar un buen apoyo sobre el fuselaje. Se muestra el resultado obtenido en la *figura 4.61*.

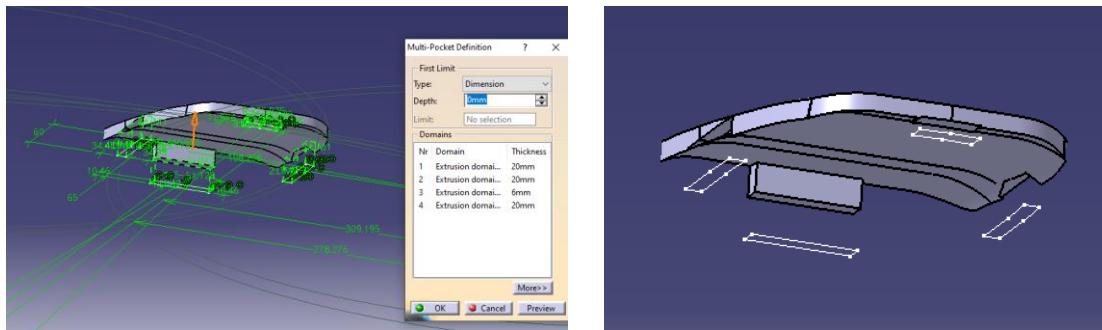


Figura 4.61: Segunda parte de la creación de los apoyos de la tapa 4 para el fuselaje.

Por último, se puede ver en la figura 4.62 el proceso de creación de una pestaña de encaje en el fuselaje mediante un “Pad”:

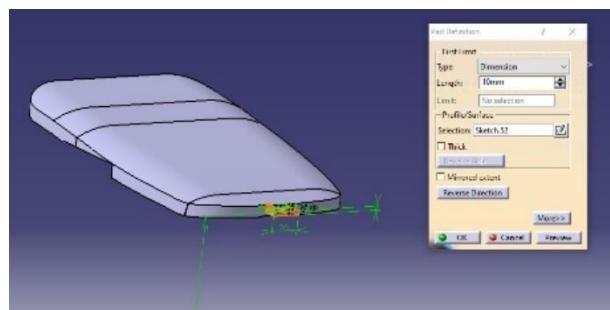


Figura 4.62: Creación de la pestaña de la tapa 4.

Finalmente se muestra en la *figura 4.63* el resultado obtenido para esta pieza (tapa 4):

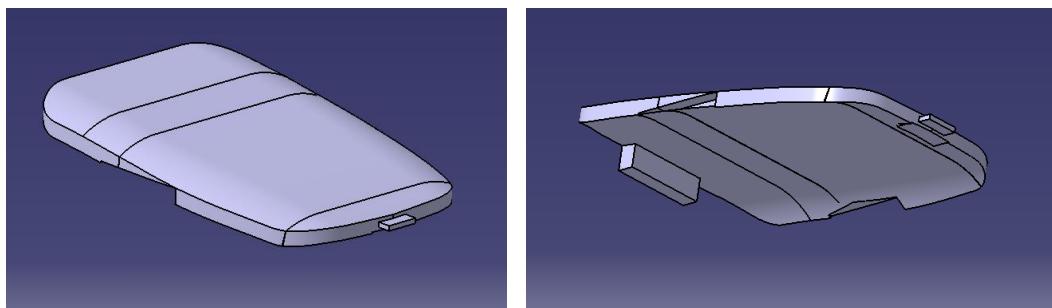


Figura 4.63: Resultado final de la tapa 4.

4.2.8. Diseño de la Tapa Inferior

Tal y como se ha hecho con las anteriores tapaderas, la tapa de la parte inferior del fuselaje se comienza haciendo de forma similar, como puede verse en la siguiente *figura 4.64*.

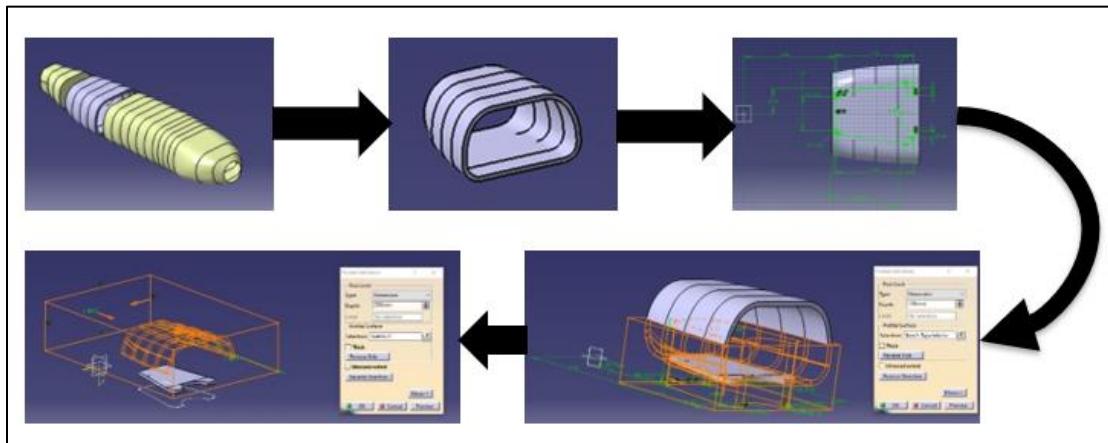


Figura 4.64: Primera parte del proceso de diseño de la tapa inferior.

Después de seguir los pasos mostrados en la figura anterior, se diseña una pestaña de cierre para la tapa (ver *figura 4.65*).

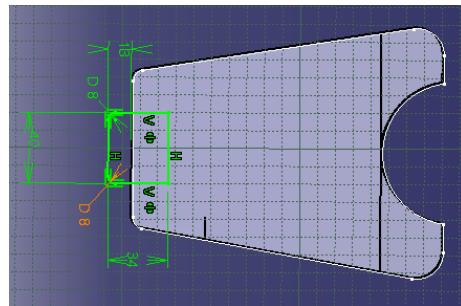


Figura 4.65: Creación de la pestaña de la tapa inferior.

En conclusión, el diseño final obtenido para la tapa inferior se muestra en la siguiente *figura 4.66*.

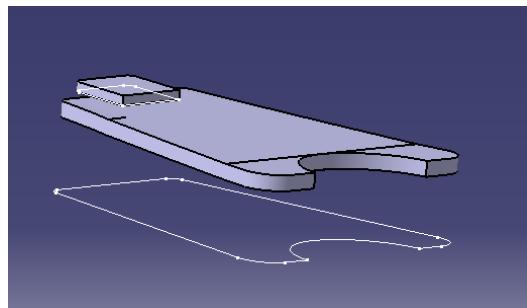


Figura 4.66: Resultado final de la tapa inferior.

4.2.9. Diseño de los Patines del Tren de Aterrizaje

El proceso de diseño de los patines de este dron es bastante sencillo, y consta de los siguientes pasos. Cabe destacar, antes de empezar con el procedimiento seguido, que solo es necesario diseñar uno de los patines, pues el otro es simétrico y no es necesario diseñar los dos.

El primer paso del procedimiento aquí seguido es hacer el sketch del patín, mostrado en la figura 4.67.

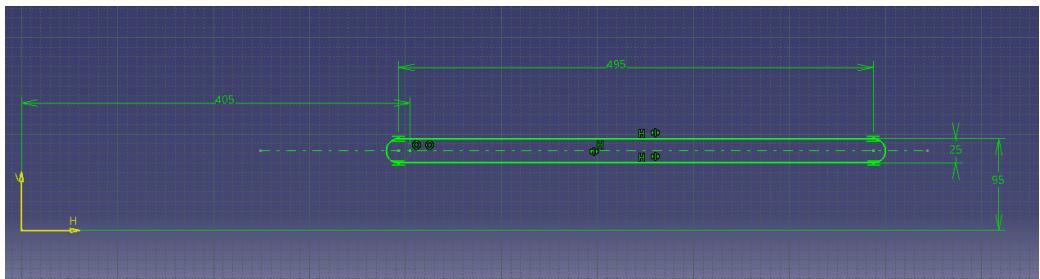


Figura 4.67: Sketch del patín.

Luego, se utiliza la herramienta “Pad” para dar volumen a este sketch (ver figura 4.68), con una altura de 30mm.

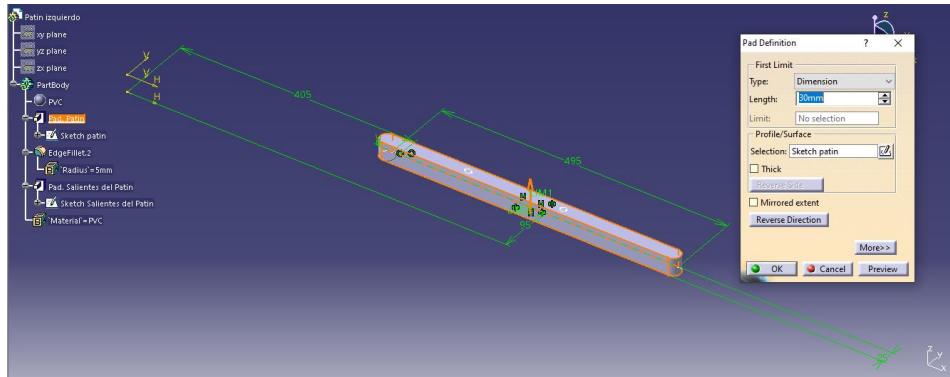


Figura 4.68: “Pad” del sketch del patín.

Además, se emplea la herramienta “Edge Fillet” para redondear el borde inferior (ver figura 4.69), con un radio de 5mm.

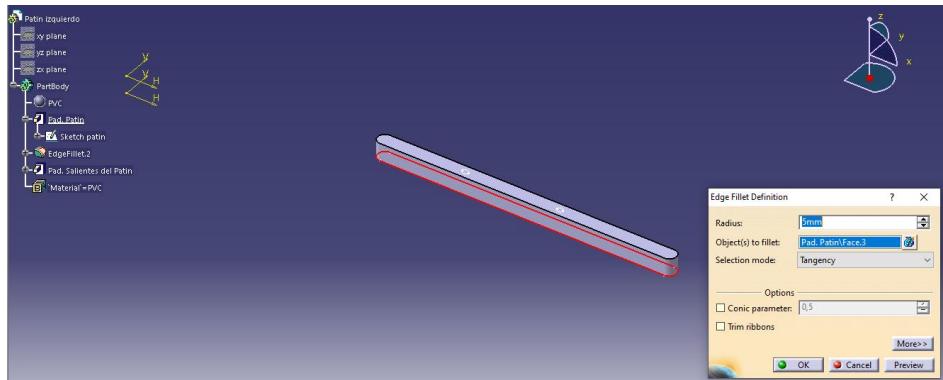


Figura 4.69: Aplicación de “Edge Fillet” al patín.

Después, se hacen unos pequeños salientes (ver figura 4.70) que servirán para encajar el patín en el cuerpo del fuselaje.

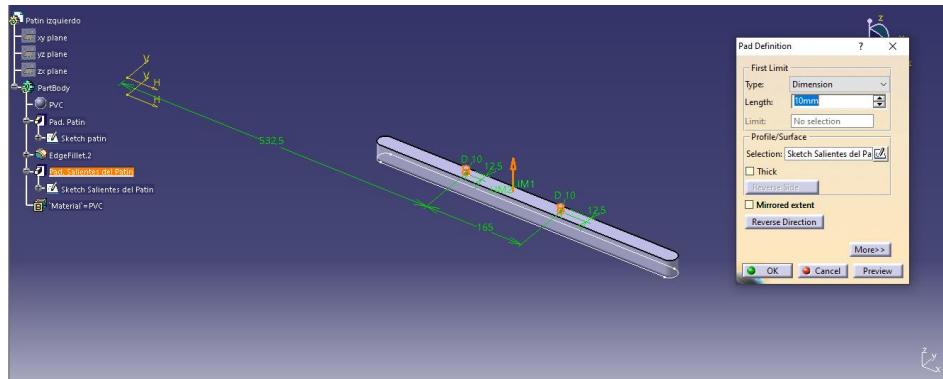


Figura 4.70: Creación de las pestañas de encaje mediante la herramienta “Pad”.

Tras ello se le añade el material (el cual no es representativo, pues solo será de utilidad para poderle asignar un color al patín en el “Assembly Design”) al patín con la herramienta “Apply Material”.

Esta es la imagen final del patín diseñado (*figura 4.71*):

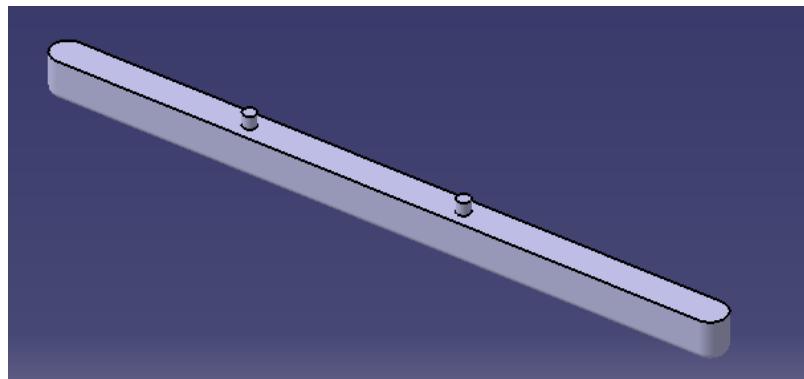


Figura 4.71: Patín definitivo.

4.2.10. Diseño de los Tubos Estructurales

Los tubos estructurales a los que se hace referencia en este apartado son los tubos negros marcados en la siguiente imagen (*figura 4.72*), destinados a albergar los tubos de encaje de fibra de carbono de las semialas, soportando esfuerzos transmitidos por dichos tubos.



Figura 4.72: Patín definitivo.

Durante el proceso de toma de mediciones, el espesor de estos tubos resultó ser imposible de conocer sin tener que desmontar pieza por pieza el dron. Por ello, se estimó un grosor considerado como “suficiente”. Sin embargo, el diámetro interior de estos tubos sí podía conocerse a través del diámetro exterior de los tubos de las semialas, los cuales deben encajar en los primeros.

El diseño de estos tubos se ha hecho de una forma muy sencilla, consistiendo solamente en tres pasos: hacer un “Sketch” y aplicarle a este un “Pad” con la altura necesaria. En las siguientes imágenes se resume este breve proceso para ambos tubos - en la *figura 4.73* se muestra el tubo más cercano al borde de ataque (BA) de la ala, mientras que en la *figura 4.74* se muestra el tubo más cercano al borde de salida (BS) de la ala -.

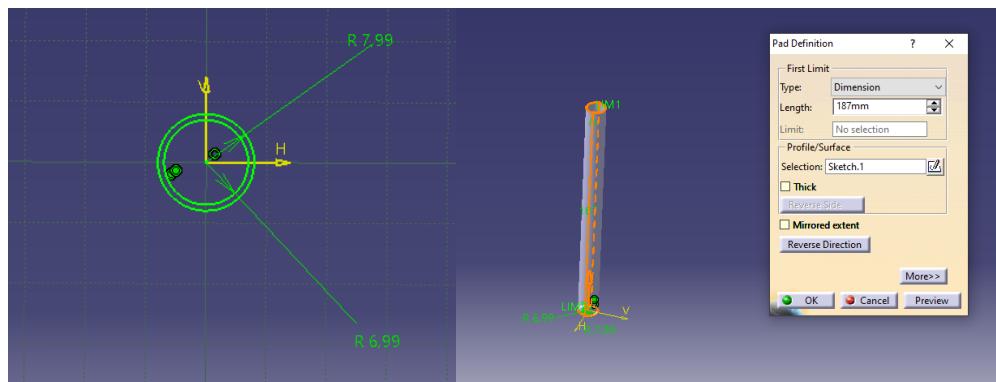


Figura 4.73: Tubo estructural más cercano al borde de ataque (BA) de la ala.



Figura 4.74: Tubo estructural más cercano al borde de salida (BS) del ala.

Destaca que la única diferencia entre ambos tubos se encuentra en sus diámetros, siendo el tubo de borde de salida más estrecho que el de borde de ataque.

4.2.11. Montaje del dron en Assembly Design

En este apartado se muestra el ensamblaje resultante de todas las piezas diseñadas en Catia en los apartados anteriores, para lo cual se ha utilizado el módulo “Assembly Design”. Dentro de este módulo, con el fin de establecer las “constraints” adecuadas entre todas las piezas diseñadas con el fin de saber si todas estas encajan entre ellas o si, por el contrario, hay piezas que no encajan y fuera necesario volver a diseñarlas. Para establecer dichas “constraints” se han empleado las herramientas: Coincidence Constraint, Contact Constraint y Offset Constraint.

Tras completar el ensamblaje de todas las piezas mostradas en este TFG, el resultado final obtenido es el mostrado en la *figura 4.75*, a continuación:

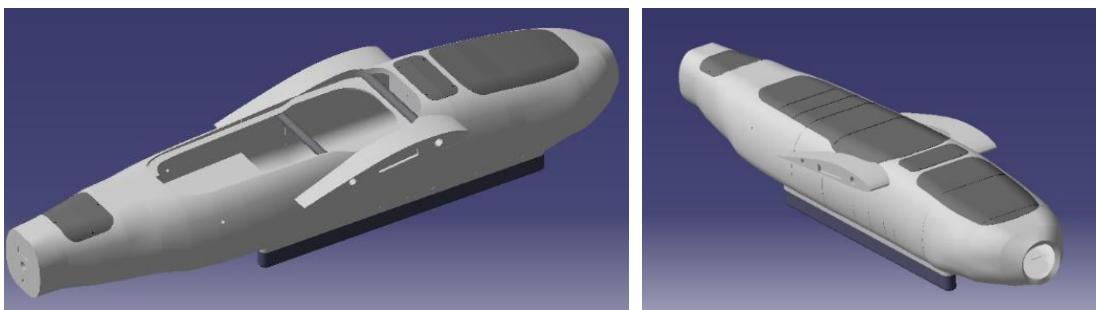


Figura 4.75: Ensamblaje de las piezas del dron diseñadas en este TFG.

Cabe destacar que la presencia de los pequeños tramos de ala en los laterales del fuselaje que se muestran en la *figura 4.75* son elementos añadidos al fuselaje por Virginia, pudiendo consultar su realización en unos meses, en su TFG complementario al aquí escrito. Uniendo ambos trabajos, el resultado final es el ensamblaje del dron completo puede verse en la *figura 1.5* de este trabajo.

4.3. Toma de Datos en el Trabajo de Campo

En el *Capítulo 3* se explicaron las conclusiones a las que se llegó el “Día de la Prueba de Vuelo”. Se entendieron y definieron las necesidades específicas detrás de este proyecto. Además, se mostraron brevemente las secciones de la aplicación web destinadas a la introducción de datos y salida de resultados (*figuras 3.20, 3.21, 3.22, 3.23*). En este apartado, se explicará en detalle el proceso de selección de las variables de salida (outputs) y de los parámetros de entrada (inputs).

4.3.1. Outputs solicitados por el “cliente”

Los valores de salida que el “cliente” deseaba conocer desde un inicio eran los siguientes:

- Posición del CDG del dron tras cargar el dron.
- Posición del CDG del dron sin cargar.

A los cuales, mi compañera de trabajo y yo, añadimos los siguientes outputs que, sin un coste de recursos adicional excesivo, darían al usuario una información bastante valiosa. Estos outputs añadidos son:

- Alcance de la aeronave.
- Autonomía de la aeronave.
- Sugerencias y comentarios relativos al posicionamiento del CDG y a la carga útil del vuelo.

Habiendo definido estos valores de salida, se procedió al planteamiento de una estrategia que permitiera obtener estos resultados. En el “Backend” desarrollado por Virginia, se decidió calcular el CDG del modelo del dron vacío mediante el uso de Catia V5, empleando este resultado (y con ciertos conocimientos de física) para calcular el CDG del dron cargado. Así mismo, los cálculos del alcance y la autonomía responderían a cálculos realizados en el “Backend” basados en un determinado modelo físico. Todos estos son los aspectos debidamente explicados que se podrán encontrar en el TFG de Virginia.

Por otro lado, respecto al “Frontend”, la preocupación principal sería diseñar el HMI que permitiese a los usuarios visualizar de manera sencilla estos resultados. Esto terminaría por llevarnos hacia el uso de la biblioteca “three.js” con el fin de mostrar gráficamente estos resultados del CDG sobre el dron cargado, aparte de mostrar en una sección de “resultados” los valores numéricos de todos los outputs deseados (ver *figuras 3.22 y 3.23*).

4.3.2. Inputs necesarios

Ahora, se responde la siguiente pregunta: “¿cuáles son los datos de entrada que deberá introducir un usuario para obtener los outputs citados?”

Con el fin de obtener las sugerencias, el CDG del dron cargado y el CDG del dron sin cargar, los únicos inputs necesarios serían aquellos que permitieran definir y colocar en el interior

del modelo 3D del dron las cargas de pago que el usuario necesitase. En los siguientes puntos se recogen los datos de entrada que el usuario debería poder introducir:

- Definir la cantidad de cargas de pago que se deseen introducir en el dron.
- Definir el tamaño de cada carga. Con el fin de hacer un modelo genérico para todo tipo de carga estas se modelizan mediante prismas rectangulares, tal que estos pudieran servir como representación del “Bounding Box” de cualquier tipo de carga.
- Definir la posición de cada carga en el interior del dron.
- Definir la masa de cada carga.

Teniendo en cuenta los datos solicitados en cuanto a tamaño y posición, se decidió emplear el siguiente esquema mostrado en la *figura 4.76.*, donde el “punto rojo” es la esquina del prisma respecto a la que se mide la posición del mismo.

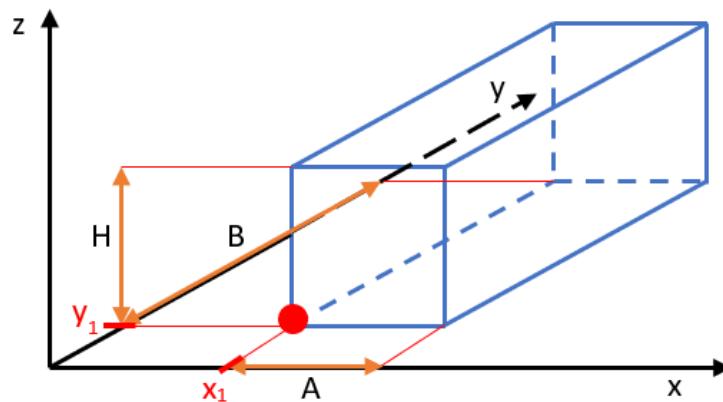


Figura 4.76: Esquema de los datos de entrada para una carga.

Debe comentarse que los ejes de referencia de la *figura 4.76.* con los que tratará el usuario se encuentran situados en este punto del modelo 3D del dron (ver *figura 4.77*), y en las direcciones indicadas en esta imagen:

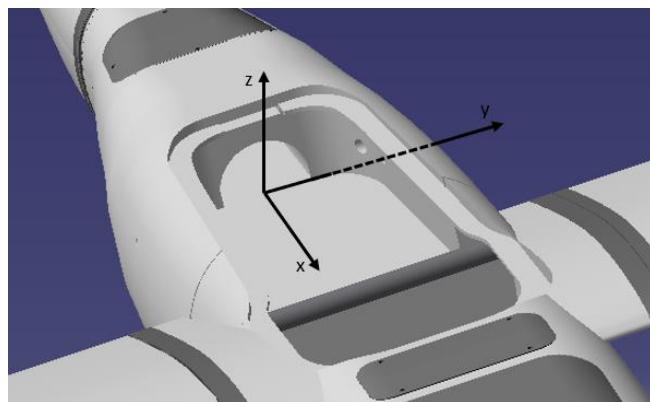


Figura 4.77: Situación de los ejes de referencia utilizados por el usuario para colocar las cargas de pago en el interior del dron

Como se puede observar en la *figura 4.76*, se deben incluir un total de seis datos por cada carga para definir su tamaño y su posición, siendo estos datos:

- Distancia A. Lado del prisma según la dirección x.
- Distancia B. Lado del prisma según la dirección y.
- Distancia H. Lado del prisma según la dirección z.
- Posición X₁ del “punto rojo” señalado en la *figura 4.76*.
- Posición Y₁ del “punto rojo” señalado en la *figura 4.76*.
- Posición Z₁ del “punto rojo” señalado en la *figura 4.76* (aunque esta coordenada Z₁ no se muestra en la *figura 4.76*, sí es un dato necesario).

Es importante señalar varios aspectos. El primero es que estos seis datos de cada carga se piden al usuario en centímetros (cm). El segundo aspecto a destacar es que la creación tridimensional (realizada en el “Backend” mediante el uso de Catia) de estos prismas siempre se realiza colocando en primer lugar el “punto rojo” dibujado según las coordenadas introducidas por el usuario, tras lo cual se procede a crear el resto del prisma, haciendo partir desde este “punto rojo” (esquina del prisma) los lados del mismo según las direcciones positivas de los ejes señalados en la *figura 4.76* y *4.77*.

Así, gracias al diseño del HMI (ver *figura 3.20*), explicado más adelante, se permitiría al usuario introducir: datos de la masa de cada carga (en kilogramos), los seis datos antes citados y, además, asignar un nombre a cada una de las cargas definidas. Más adelante, se mostrará cuál fue el código aplicado para permitir al usuario introducir la cantidad de cargas que este deseara.

Todos estos datos de entrada mencionados servirían para calcular, por un lado, los outputs relativos al CDG y las sugerencias. Por otro lado, en la *figura 3.21*, pueden verse los datos solicitados para realizar los cálculos relativos a la autonomía y el alcance de la aeronave [50], los cuales son:

- N_{mot}: número de motores utilizados durante la fase de vuelo estudiada.
- I_{mot}: corriente máxima que consume un solo motor, pedida en amperios (A). Típicamente pueden tratarse del orden de 10A, dependiendo del dron.
- I_{equip}: corriente máxima consumida por los equipos del dron (excluyendo sus motores), también pedida en amperios (A). Típicamente pueden tratarse del orden de 2A, dependiendo del dron.
- V_{bat}: tensión nominal de la batería, solicitada al usuario en voltios (V). Un valor típico puede ser de 7,4V.
- V₀: velocidad de vuelo durante la fase de crucero, solicitada en km/h. Un valor típico para el VTOL estudiado en este proyecto es de 70km/h, siendo esta la velocidad de operación recomendada por el fabricante.

- Q : capacidad de la batería en $\text{mA}\cdot\text{h}$, siendo $50000\text{mA}\cdot\text{h}$ un valor típico.
- C_{rate} : tasa máxima de descarga sin que la batería se dañe o sobrecaliente, solicitada en h^{-1} (en estrecha relación con su capacidad Q). Valor típico de 10h^{-1} .
- DR : regla de descarga de la batería. Se trata del máximo porcentaje de carga recomendable para la batería, en tanto por ciento. Un valor típico suele ser del 80%.
- L_{flying} : peso de la carga de pago en tanto por ciento. Este valor se obtiene de dividir el peso total de la carga de pago introducida entre el peso total del vuelo (el peso del dron más el peso de las cargas de pago), multiplicando por 100 a dicho valor resultante. Un valor típico puede ser del 30%.

Estos nueve datos solicitados al usuario permitirán calcular el alcance y la autonomía del vuelo como un proceso independiente del cálculo del CDG y las sugerencias relativas al mismo. Esto es, existen dos ramas de ejecución independientes en la aplicación.

Las ecuaciones y el proceso de cálculo para convertir todos estos datos de entrada en los resultados buscados se podrán ver en el futuro TFG de mi compañera Virginia.

4.4. Diseño del HMI y Redacción del Análisis Funcional

4.4.1. Diseño del HMI

En la *figura 3.24* del *Capítulo 3* se mostró el boceto de una primera idea de diseño del HMI (Human Machine Interface) que se emplearía para interactuar con el usuario y ofrecerle el servicio buscado. Este primer diseño nos pareció adecuado tanto a Virginia como a mí debido a la sencillez y usabilidad que podía propiciar. Así, aquel diseño inicial de la página web sería el adoptado como definitivo a lo largo del proyecto. A continuación, en las *figuras 4.78, 4.79 y 4.80* se muestra un collage de todas las vistas de la aplicación según vamos avanzando por ella a través de las secciones, de los acordeones y utilizando la rueda del ratón. En este punto, me gustaría invitar al lector a visitar el enlace de GitHub incluido en el *Capítulo 1* de este trabajo de forma que, empleando los códigos allí presentes pudiera ver este diseño del HMI de primera mano.

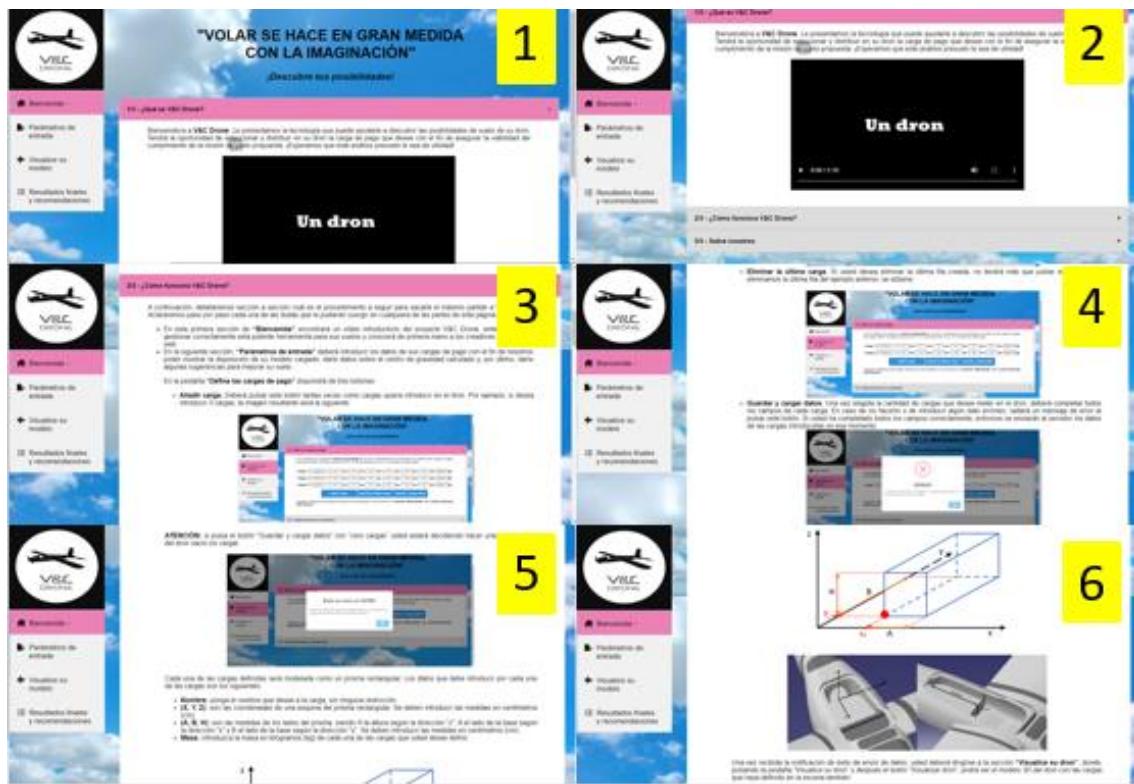


Figura 4.78: Collage número 1 del HMI de la aplicación web desarrollada.

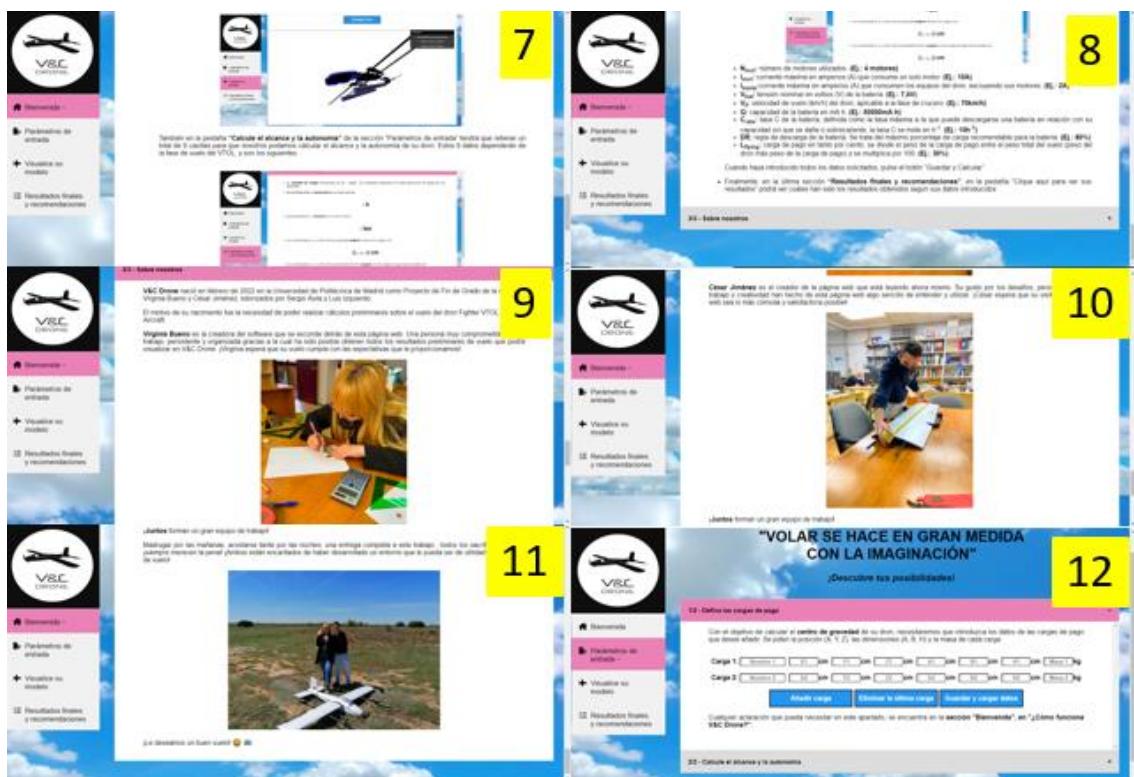


Figura 4.79: Collage número 2 del HMI de la aplicación web desarrollada.



Figura 4.80: Collage número 3 del HMI de la aplicación web desarrollada.

En los próximos apartados se explican los códigos empleados para hacer el diseño y dotar de funcionalidad a la aplicación mostrada en estas imágenes.

4.4.2. Códigos de HTML5, JavaScript y CSS

HTML [35] [36] [15] es el lenguaje de marcado utilizado para estructurar y desplegar los contenidos de la página web, mientras que JavaScript es el lenguaje de programación empleado para añadir características interactivas a dicho sitio web, y CSS es el lenguaje utilizado para diseñar y dar estilo a la página web. A continuación, se mostrarán y explicarán las partes más representativas de los códigos tanto de JavaScript como de HTML y de CSS empleados para desarrollar este proyecto. Los puntos en los que podemos dividir esta explicación son los siguientes:

4.4.2.1. Secciones (tabs)

Existen muchas formas de crear tabs (o secciones) en HTML. La aquí seleccionada es una de las enseñadas por “W3Schools” en su página web. A continuación, se muestra en la figura 4.81 el código HTML5 empleado para crear el menú de secciones de esta página web, y su resultado a la derecha:

```
<div class="tab">
  <button class="tablinks" onclick="opensection(event, 'Bienvenida')" id="defaultOpen">
    | <i class="fa-solid fa-house-chimney"></i>&nbsp;&nbsp;Bienvenida
  </button>
  <button class="tablinks" onclick="opensection(event, 'Parametros')">
    | <i class="fa-solid fa-file-pen"></i>&nbsp;&nbsp;Parámetros de <br> &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;entrada
  </button>
  <button class="tablinks" onclick="opensection(event, 'Modelo')">
    | <i class="fa-solid fa-plane"></i>&nbsp;&nbsp;Visualice su <br> &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; modelo
  </button>
  <button class="tablinks" onclick="opensection(event, 'Resultados')">
    | <i class="fa-solid fa-list-check"></i>&nbsp;&nbsp;Resultados finales <br>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
    y recomendaciones
  </button>
</div>
```

Bienvenida -
Parámetros de entrada
Visualice su modelo
Resultados finales y recomendaciones

Figura 4.81: Creación de tabs en HTML5.

Las cosas más importantes a resaltar de este código son las siguientes:

- El “button” al que se asigna el atributo ‘id = “defaultOpen”’, será la sección abierta por defecto al abrirse la página web. Esto es gracias al método mostrado en la *figura 4.82*.

```
document.getElementById("defaultOpen").click();
```

Figura 4.82: Método empleado para “disparar” un clic sobre el “id” seleccionado.

- Destaca el uso de la *librería Fontawesome*[23], utilizada para incluir los dibujos (avión, lista, casa, etc.) en los tabs, decorando así la imagen y mejorando la estética de la página.

A continuación, se muestra el resto del código HTML, observándose que todo el código se encuentra agrupado en varias etiquetas “div” (ver *figura 4.83*) asociadas al nombre asignado a cada tab, de forma que al clicar sobre uno de los “tabs” se selecciona el “div” correspondiente y se muestra su contenido, ocultando el contenido del resto de los “div”.

```
43 > <div id="Bienvenida" class="tabcontent" background="cielo_azul.jpg"> ...
207
208
209 > <div id="Parametros" class="tabcontent"> ...
310
311
312 > <div id="Modelo" class="tabcontent"> ...
331
332
333 > <div id="Resultados" class="tabcontent"> ...
372 </div>
```

Figura 4.83: Código completo de HTML agrupado en etiquetas “div”.

En la siguiente *figura 4.84* se muestra la función de JavaScript empleada para abrir cada sección. Además, en la *figura 4.85* se enseñan las líneas de código utilizadas en el CSS.

```
42 //ABRIR UNA SECCIÓN EN LA PÁGINA WEB
43 function opensection(evt, SectionName) {
44     var i, tabcontent, tablinks;
45     tabcontent = document.getElementsByClassName("tabcontent");
46     for (i = 0; i < tabcontent.length; i++) {
47         tabcontent[i].style.display = "none";
48     }
49     tablinks = document.getElementsByClassName("tablinks");
50     for (i = 0; i < tablinks.length; i++) {
51         tablinks[i].className = tablinks[i].className.replace(" active", "");
52     }
53     document.getElementById(SectionName).style.display = "block";
54     evt.currentTarget.className += " active";
55 }
```

Figura 4.84: Función de JavaScript para abrir la sección elegida.

```
.tab button:hover {
    background-color: #ddd;
}

.tab button.active {
    background-color: #rgb(236, 129, 183);
}

.tabcontent {
    padding: 0px 12px;
    text-align: center;
    border: 0px solid #rgb(197, 15, 15);
    width: 100%;
    border-left: none;
    height: 100%;
}
```

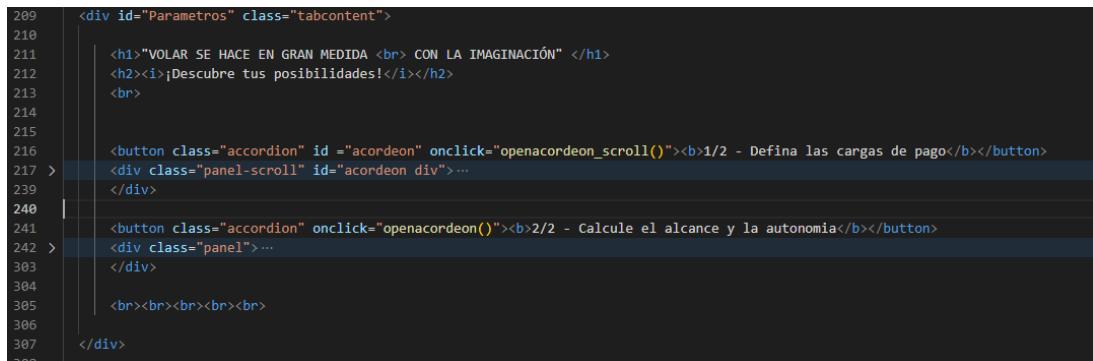
```
.tab button {
    display: block;
    background-color: inherit;
    color: black;
    padding-top: 22px;
    padding-right: 10%;
    padding-bottom: 16px;
    padding-left: 8%;
    width: 100%;
    border: none;
    outline: none;
    text-align: left;
    cursor: pointer;
    transition: 0.3s;
    font-size: 17px;
}
```

```
.tab {
    position: fixed;
    top: 33%;
    float: left;
    border: 1px solid #rgb(236, 233, 233);
    background-color: #f1f1f1;
    width: 17%;
    height: 0px;
```

Figura 4.85: Códigos de CSS aplicados al diseño de los tabs.

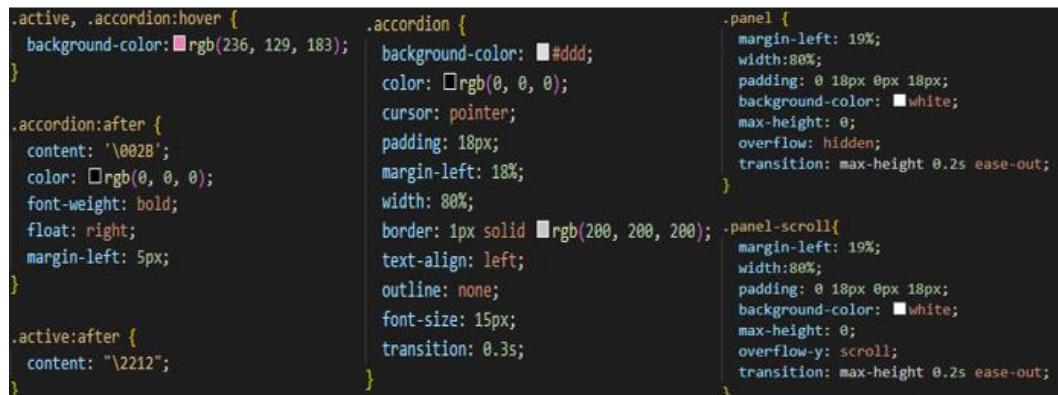
4.4.2.2. Acordeones

La creación de los acordeones aquí empleados es también procedente de las lecciones de “W3Schools”. El código HTML, el CSS y un ejemplo del resultado obtenido tras su aplicación se muestran respectivamente en las *figuras 4.86, 4.87 y 4.88*. En la *figura 4.86* se observa que el acordeón está compuesto por dos partes: un botón (abre y cierra el acordeón) y un panel (muestra el contenido del acordeón).



```
209 <div id="Parametros" class="tabcontent">
210   <h1>"VOLAR SE HACE EN GRAN MEDIDA <br> CON LA IMAGINACIÓN" </h1>
211   <h2><i>¡Descubre tus posibilidades!</i></h2>
212   <br>
213
214
215   <button class="accordion" id ="acordeon" onclick="openacordeon_scroll()"><b>1/2 - Defina las cargas de pago</b></button>
216   <div class="panel-scroll" id="acordeon div">...
217   </div>
218
219   <button class="accordion" onclick="openacordeon()"><b>2/2 - Calcule el alcance y la autonomía</b></button>
220   <div class="panel">...
221   </div>
222
223   <br><br><br><br><br>
224
225 </div>
226
227
228
```

Figura 4.86: Código HTML5 que muestra dos acordeones dentro del tab “Parametros”.



```
.active, .accordion:hover {
  background-color: #rgb(236, 129, 183);
}

.accordion:after {
  content: '\002B';
  color: #rgb(0, 0, 0);
  font-weight: bold;
  float: right;
  margin-left: 5px;
}

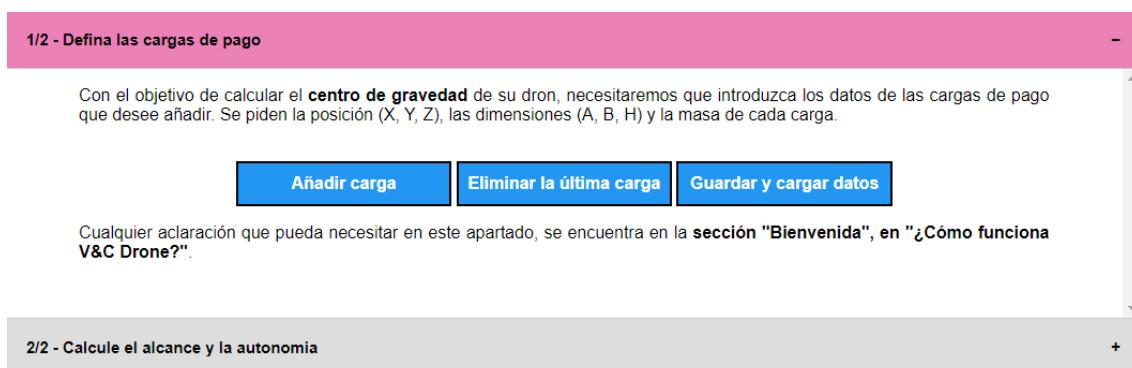
.active:after {
  content: "\2212";
}

.accordion {
  background-color: #ddd;
  color: #rgb(0, 0, 0);
  cursor: pointer;
  padding: 18px;
  margin-left: 18%;
  width: 80%;
  border: 1px solid #rgb(200, 200, 200);
  text-align: left;
  outline: none;
  font-size: 15px;
  transition: 0.3s;
}

.panel {
  margin-left: 10%;
  width:80%;
  padding: 0 18px 0px 18px;
  background-color: #white;
  max-height: 0;
  overflow: hidden;
  transition: max-height 0.2s ease-out;
}

.panel-scroll{
  margin-left: 19%;
  width:80%;
  padding: 0 18px 0px 18px;
  background-color: #white;
  max-height: 0;
  overflow-y: scroll;
  transition: max-height 0.2s ease-out;
}
```

Figura 4.87: Códigos de CSS aplicados al diseño de los acordeones.



1/2 - Defina las cargas de pago

Con el objetivo de calcular el **centro de gravedad** de su dron, necesitaremos que introduzca los datos de las cargas de pago que deseé añadir. Se piden la posición (X, Y, Z), las dimensiones (A, B, H) y la masa de cada carga.

Añadir carga Eliminar la última carga Guardar y cargar datos

Cualquier aclaración que pueda necesitar en este apartado, se encuentra en la sección "Bienvenida", en "¿Cómo funciona V&C Drone?".

2/2 - Calcule el alcance y la autonomía

Figura 4.88: Ejemplo de aplicación de los acordeones.

De la *figura 4.87* anterior, es importante destacar la diferencia entre la clase “panel” y la clase “panel-scroll”. Se observa que esta diferencia radica, principalmente, en el

“overflow”. Todos los acordeones de la página web tienen un contenido fijado, y el acordeón se ajusta a dicho contenido al abrirse. Sin embargo, en el acordeón “Defina las cargas de pago” (*figura 4.88*) se tiene la posibilidad de, una vez abierto el acordeón, modificar el contenido del panel. Por este motivo será necesario encontrar alguna manera de hacer que el panel aumente o disminuya de tamaño según se añadan o se quiten cargas con los botones. La solución que se encontró a este problema fue modificar la función original “openacordeon()” del JavaScript a otra “openacordeon_scroll()” para este acordeón en concreto, donde puede observarse que el único cambio se encuentra en la última línea a. Lo conseguido con este cambio es el siguiente efecto: el panel del acordeón cambia su altura dinámicamente según cambia su contenido, hasta llegar a tener una altura de 500px, momento en el que se activa el empleo del “scroll” en el interior del panel.

```
//ABRIR UN ACORDEÓN DE LA PÁGINA WEB
function openacordeon(){
    var acc = document.getElementsByClassName("accordion");
    var i;

    for (i = 0; i < acc.length; i++) {
        acc[i].addEventListener("click", function() {
            this.classList.toggle("active");
            var panel = this.nextElementSibling;
            if (panel.style.maxHeight) {
                panel.style.maxHeight = null;
            } else {
                panel.style.maxHeight = panel.scrollHeight + "px";
            }
        });
    }
}

//ABRIR EL ACORDEÓN PARA DEFINIR LAS CARGAS DE PAGO
function openacordeon_scroll(){
    var acc = document.getElementsByClassName("accordion");
    var i;

    for (i = 0; i < acc.length; i++) {
        acc[i].addEventListener("click", function() {
            this.classList.toggle("active");
            var panel = this.nextElementSibling;
            if (panel.style.maxHeight) {
                panel.style.maxHeight = null;
            } else {
                panel.style.maxHeight = 550 + "px";
            }
        });
    }
}
```

Figura 4.89: Código del JavaScript encargado de abrir el panel de los acordeones.

4.4.2.3. Biblioteca jQuery

Ahora hablaremos sobre una de las partes más interesantes del código en JavaScript de este trabajo: el diseño del HMI para la definición de las cargas de pago, cuya interfaz de usuario puede verse en la *figura 3.20*. Destaca el funcionamiento dinámico de esta estructura HTML, pues al pulsar los botones “Añadir carga” o “Eliminar la última carga” se añade o elimina una fila completa de inputs. No existe un límite de cargas a añadir, lo que da mucha versatilidad al uso de la aplicación para el usuario. Sobre esto, destaca el uso de la biblioteca jQuery [20] para lograr este efecto, utilizando el siguiente código (*figura 4.90*):

```
6 //SE CREA LA RESPUESTA A LOS BOTONES DE LA CREACIÓN Y ELIMINACIÓN DE CARGAS
7 $(document).ready(function() {
8
9     $("#add").click(function(){
10         var contador = $("div[class='Cargas']").length + 1 ;
11
12         $(this).before(
13             "<div class='Cargas'><label for='Carga "+ contador +"><b>Carga "+ contador +"</b></label><input type='text' id='Carga "+ contador +
14             "' class='meterdatos_cargas_nombre' placeholder='Nombre '+ contador +' />&nbsp;&nbsp;<input type='number' id='Coordenada_X_Carga ' +
15             contador +' class='meterdatos_cargas' placeholder='X'+ contador +' /><label for='Coordenada_X_Carga '+ contador +
16             "'><b>cmc</b></label>&nbsps;&nbsps;<input type='number' id='Coordenada_Y_Carga '+ contador +' class='meterdatos_cargas' placeholder='Y'+
17             contador +' /><label for='Coordenada_Y_Carga '+ contador +'><b>cmc</b></label>&nbsps;&nbsps;<input type='number' id='Coordenada_Z_Carga ' +
18             contador +' class='meterdatos_cargas' placeholder='Z'+ contador +' /><label for='Coordenada_Z_Carga '+ contador +
19             "'><b>cmb</b></label>&nbsps;&nbsps;<input type='number' id='Longitud_del_lado_h_Carga '+ contador +
20             "' class='meterdatos_cargas' placeholder='A '+ contador +' /><label for='Longitud_del_lado_h_Carga '+ contador +
21             "'><b>cmc</b></label>&nbsps;&nbsps;<input type='number' id='Longitud_del_lado_v_Carga '+ contador +
22             "' class='meterdatos_cargas' placeholder='B '+ contador +' /><label for='Longitud_del_lado_v_Carga '+ contador +
23             "'><b>cmc</b></label>&nbsps;&nbsps;<input type='number' id='Altura_Carga '+ contador +' class='meterdatos_cargas' placeholder='H'+
24             contador +' /><label for='Altura_Carga '+ contador +'><b>cmc</b></label>&nbsps;&nbsps;<input type='number' id='Masa_Carga '+ contador +
25             "' class='meterdatos_cargas' placeholder='Masa '+ contador +' /><label for='Masa_Carga '+ contador +
26             "'><b>kg</b></label> <br> <br> </div> ");
27     });
28
29     $(document).on('click', '#delete', function(){
30
31         $(".Cargas").last().remove();
32
33     });
34});
```

Figura 4.90: Código jQuery en JavaScript para añadir y eliminar filas de cargas.

El código HTML5 donde se crean estos botones es el mostrado en la siguiente *figura 4.91*:

```

216 <button class="accordion" id = "acordeon" onclick="openacordeon_scroll()"><b>1/2 - Defina las cargas de pago</b></button>
217 <div class="panel-scroll" id="acordeon div">
218   <p>
219     Con el objetivo de calcular el <b>centro de gravedad</b> de su dron, necesitaremos que introduzca los datos
220     de las cargas de pago que deseé añadir. Se piden la posición (X, Y, Z), las dimensiones (A, B, H) y la masa
221     de cada carga.
222   </p>
223 
224   <div class="botones">
225     <br>
226     <input type="button" class="botones_cargas" id="add" value="Añadir carga"> </input>
227     <input type="button" class="botones_cargas" id="delete" value="Eliminar la última carga">
228     <input id="calculando" type="button" class="botones_cargas" onclick="meteCargaPago()" value="Guardar y cargar datos">
229     <br>
230   </div>
231 
232   <p>
233     Cualquier aclaración que pueda necesitar en este apartado, se encuentra en la <b>sección "Bienvenida", en
234     <a href="#">¿Cómo funciona V&C Drone?</a>.</p>
235   <br><br>
236 
237 </div>
238

```

Figura 4.91: Creación en HTML5 de los botones de la *figura 3.20*.

En el código de la *figura 4.90* cada vez que se pulsa el botón “Añadir carga” se crea un “div” completo en el que se incluyen las etiquetas de texto y los inputs necesarios para definir la nueva carga. Fue necesario introducir un “contador” capaz de contar el número de cargas presentes en cada momento, contando la cantidad de elementos que perteneciesen a la clase “Cargas”. De esta manera, con dicho contador se podrían leer y guardar todos los datos de las cargas empleando bucles, independientemente del número que hubiera. Por último, se explica que cuando se clica el botón cuyo identificador es “delete” se borra el último elemento que se haya añadido, es decir, la última carga.

4.4.2.4. Envío de datos con JSON

En la siguiente *figura 4.92* se muestra un ejemplo con dos cargas de pago donde se han llenado todos los campos necesarios.

1/2 - Defina las cargas de pago

Con el objetivo de calcular el **centro de gravedad** de su dron, necesitaremos que introduzca los datos de las cargas de pago que deseé añadir. Se piden la posición (X, Y, Z), las dimensiones (A, B, H) y la masa de cada carga.

Carga 1:	Batería 1	0	cm	0	cm	0	cm	19	cm	5	cm	5	cm	2.5	kg
Carga 2:	Placa 1	20	cm	0	cm	-5.5	cm	10	cm	5	cm	1	cm	0.5	kg

Añadir carga Eliminar la última carga Guardar y cargar datos

Cualquier aclaración que pueda necesitar en este apartado, se encuentra en la **sección "Bienvenida"**, en "[¿Cómo funciona V&C Drone?](#)".

2/2 - Calcule el alcance y la autonomía

Figura 4.92: Ejemplo de definición de cargas en el HTML.

Una vez el usuario pulsa el botón “Guardar y cargar datos”, tal y como puede verse en la *figura 4.91*, la función “meteCargaPago()” del JavaScript es llamada. Las siguientes *figuras 4.93, 4.94, 4.95 y 4.96* muestran el código completo de esta función, la cual se encarga de crear un JSON [56][57][59][60][61] con los datos introducidos por el usuario, comprobar la validez de los valores introducidos y luego mandar la petición al servidor, llamando a la función “Enviar_cargas()”.

```
//LECTURA, COMPROBACIÓN Y ENVÍO DE DATOS AL SERVIDOR DE LOS VALORES INTRODUCIDOS PARA LAS CARGAS DE PAGO
function meteCargaPago(){

let mijson = [];

let num_cargas = document.getElementsByClassName("Cargas").length;
if (num_cargas == 1) {
    document.getElementById("numero_cargas").innerHTML = "La <b>cantidad de cargas</b> introducidas es de " + num_cargas +
    " carga. Los resultados obtenidos con esta distribución de carga son los siguientes.";
} else {
    document.getElementById("numero_cargas").innerHTML = "La <b>cantidad de cargas</b> introducidas es de " + num_cargas +
    " cargas. Los resultados obtenidos con esta distribución de carga son los siguientes.";
}

let numero_cargas = {numero_cargas:num_cargas};
mijson.push(numero_cargas);

if(num_cargas == 0){

    // var dictstring = JSON.stringify(mijson);
    // var type = "JSON";
    // var filename = "Datos de las cargas.json";
    // var file = new Blob([dictstring], {type: type});
    // var a = document.createElement("a");
    // url = URL.createObjectURL(file);
    // a.href = url;
    // a.download = filename;
    // document.body.appendChild(a);
    // a.click();
}
}

```

Figura 4.93: Primera imagen de la función “meteCargaPago()” del JavaScript.

```
let str = JSON.stringify(mijson);
Enviar_cargas(str);
swal("Este es solo un AVISO", "No se ha introducido ninguna carga de pago, por lo que en la siguiente sección el dron se visualizará vacío.")

} else[]

for (let i = 1; i <= num_cargas; i++) {

let text = i.toString();

var nombre_carga = document.getElementById("Carga " + text).value;
var coord_x = document.getElementById("Coordenada_X_Carga " + text).value;
var coord_y = document.getElementById("Coordenada_Y_Carga " + text).value;
var coord_z = document.getElementById("Coordenada_Z_Carga " + text).value;
var lado_h = document.getElementById("Longitud_del_lado_h_Carga " + text).value;
var lado_v = document.getElementById("Longitud_del_lado_v_Carga " + text).value;
var altura = document.getElementById("Altura_Carga " + text).value;
var masa_carga = document.getElementById("Masa_Carga " + text).value;

let cargaPago = getCargaPago(nombre_carga, coord_x, coord_y, coord_z, lado_h, lado_v, altura, masa_carga);

mijson.push(cargaPago);
}

// var dictstring = JSON.stringify(mijson);
// var type = "JSON";
// var filename = "Datos de las cargas.json";
// var file = new Blob([dictstring], {type: type});
// var a = document.createElement("a");
// url = URL.createObjectURL(file);

```

Figura 4.94: Segunda imagen de la función “meteCargaPago()” del JavaScript.

```

// a.href = url;
// a.download = filename;
// document.body.appendChild(a);
// a.click();

let str = JSON.stringify(mijson);

for (let j = 1; j <= num_cargas; j++) {

    let j_text = j.toString();

    var coord_x_comprobar = mijson[j].x;
    var coord_y_comprobar = mijson[j].y;
    var coord_z_comprobar = mijson[j].h_off;
    var lado_h_comprobar = mijson[j].long_lado_h;
    var lado_v_comprobar = mijson[j].long_lado_v;
    var altura_comprobar = mijson[j].l_pad;
    var masa_carga_comprobar = mijson[j].masa;

    if (coord_x_comprobar === null || coord_x_comprobar ==='' || coord_y_comprobar === null || coord_y_comprobar ==='' || coord_z_comprobar === null || coord_z_comprobar ==='' || lado_h_comprobar === null || lado_h_comprobar ==='' || lado_v_comprobar === null || lado_v_comprobar ==='' || altura_comprobar === null || altura_comprobar ==='' || masa_carga_comprobar === null || masa_carga_comprobar ==='') {

        swal("ERROR", "Le falta un campo por completar en los datos de la Carga " + j_text + " o ha introducido un valor no válido.", "error");
        return;
    } else if (lado_h_comprobar < 0.1 || lado_v_comprobar < 0.1 || altura_comprobar < 0.1){

        swal("ERROR", "La longitud de alguno de los lados de la Carga " + j_text +
        " es demasiado pequeña o tiene un valor negativo. Se deben introducir valores superiores a 0.1cm.", "error");
    }
}

```

Figura 4.95: Tercera imagen de la función “meteCargaPago()” del JavaScript.

```

    return;

} else if (masa_carga_comprobar < 0.0001){

    swal("ERROR", "El valor de la masa de la Carga " + j_text +
    " es demasiado pequeño o tiene un valor negativo. Se debe introducir un valor superior a 0.0001kg.", "error");
    return;

} else if (j === num_cargas){

    Enviar_cargas(str);

}

}

```

Figura 4.96: Cuarta imagen de la función “meteCargaPago()” del JavaScript.

Si se descomentan las últimas líneas de la *figura 4.94* (y las primeras de la *figura 4.95*), al pulsar el botón “Guardar y cargar datos” de la *figura 4.92*, se descargará un archivo JSON [25] que contendrá los datos introducidos por el usuario. Una imagen de este archivo generado es la que se muestra en la siguiente *figura 4.97*:

```

[{"numero_cargas":2},
 {"name":"Bateria 1","x":"0","y":"0","h_off":"0","long_lado_h":"19","long_lado_v":"5","l_pad":"5","masa":"2.5"},
 {"name":"Placa 1","x":"20","y":"0","h_off":"-5.5","long_lado_h":"10","long_lado_v":"5","l_pad":"1","masa":"0.5"}]

```

Figura 4.97: JSON generado con los valores de las cargas de pago introducidas en la *figura 4.92*.

De esta forma, en un primer bucle (*figura 4.94*) se crea el JSON y en un segundo bucle (*figuras 4.95 y 4.96*) se comprueban todos los datos introducidos por el usuario, devolviendo un mensaje de error en caso de haber algún campo vacío o mal rellenado. En caso de estar todo bien rellenado se llama a la función “Enviar_cargas()”, la cual hará una petición al servidor.

Para enviar los mensajes de error, avisos y mensajes de éxito, destaca el uso de la [librería sweetalert](#) [24], la cual añade un estilo más decorativo que los simples “alert()” de JavaScript. Un ejemplo de estos mensajes sucedería al dejar un campo sin completar, tal y como puede verse en la *figura 4.98*, a continuación:



Figura 4.98: Mensaje de error con el uso de la librería sweetalert.

4.4.2.5. Envío de datos al servidor local Flask en Python

Al realizar la llamada a la función “Enviar_cargas()”, a esta se le pasa el JSON generado (ver *figura 4.97*) como un string [30][31][33], es decir, una cadena de texto. Se convierte el JSON en una variable de tipo string utilizando el método `JSON.stringify()` mostrado en la *figura 4.95*. Tras esto, tal y como puede verse en la *figura 4.99*, se realiza una petición “await fetch” [29] [32] [52] [53] (método solo aplicable en funciones asíncronas) mediante la cual se manda el string del JSON al servidor en Python. En este momento el código de JavaScript espera la respuesta del servidor para continuar con la siguiente línea de código.

```
//FUNCIÓN QUE ENVÍA EL JSON DE LAS CARGAS DE PAGO AL SERVIDOR FLASK
async function Enviar_cargas(myjson){
    clic = clic + 1;

    const response = await fetch(`http://127.0.0.1:5000/datos_cargas/${myjson}`);
    const data = await response.json();

    let y_cdg_dron: number = parseFloat(data[0]).toFixed(2);
    let y_cdg_dron = Number(parseFloat(data[1]).toFixed(2));
    let z_cdg_dron = Number(parseFloat(data[2]).toFixed(2));
    let x_cdg_total = Number(parseFloat(data[3]).toFixed(2));
    let y_cdg_total = Number(parseFloat(data[4]).toFixed(2));
    let z_cdg_total = Number(parseFloat(data[5]).toFixed(2));
    let string_sugerencias = data[6];

    document.getElementById("cdg_original").innerHTML = "("+ x_cdg_dron + ", " + y_cdg_dron + ", " + z_cdg_dron + ") cm";
    document.getElementById("cdg_cargado").innerHTML = "("+ x_cdg_total + ", " + y_cdg_total + ", " + z_cdg_total + ") cm";
    document.getElementById("sugerencias").innerHTML = "<b>&quot;" + string_sugerencias + "&quot;</b>";

    swal("PERFECTO", "Los datos se han enviado y cargado correctamente. Ya puede ver sus resultados.", "success");
}
```

Figura 4.99: Función “Enviar_cargas()” que manda la petición y recibe la respuesta del servidor local Flask en Python.

Puede verse en la siguiente *figura 4.100* el código del servidor Flask en Python que recibe la petición “await fetch” de JavaScript. Se accede a la ruta indicada en JavaScript y se ejecuta la función asociada a dicha ruta. Primero vuelve a convertir el string enviado en un JSON y este se envía a la función en Python encargada de hacer los cálculos necesarios, tal que la respuesta de dicha función se devuelve al JavaScript mediante el “return” del final de la fusión.

```

1  from flask import Flask, request, jsonify, json
2  from flask_cors import CORS
3  from CDG_CP_pycatia_FUNCION import Calculo_cdg, Calcular_AyA
4
5  app = Flask(__name__)
6  cors = CORS(app)
7
8  @app.route("/datos_cargas/<myjson>")
9  def postME1(myjson):
10     datos = json.loads(myjson)
11     resultado = Calculo_cdg(datos)
12     resultado = jsonify(resultado)
13     return resultado
14
15
16  @app.route("/datos_AyA/<ourjson>")
17  def postME2(ourjson):
18    datos_AyA = json.loads(ourjson)
19    resultado_AyA = Calcular_AyA(datos_AyA)
20    resultado_AyA = jsonify(resultado_AyA)
21    return resultado_AyA
22
23
24  if __name__ == "__main__":
25      app.run(debug=True)

```

Figura 4.100: Servidor Flask en Python.

Observando la *figura 4.100* queda claro lo comentado con anterioridad sobre la independencia de los dos procesos de cálculo: la autonomía y el alcance, por un lado; los cálculos de CDG y las sugerencias por el otro.

El proceso de cálculo de la autonomía y el alcance sigue un camino similar, pero con sus propias funciones. En este caso, por ser una cantidad de datos de entrada definida de antemano, no es necesario el empleo de bucles para leer y comprobar los datos, simplificándose bastante estos procesos. Sin embargo, el envío de la petición al servidory la espera de la respuesta se realizan de formas completamente análogas.

4.4.2.6. Elemento HTML <iframe>

Una vez enviados los datos de las cargas al servidor, se permite al usuario visualizar el modelo 3D del VTOL a través lo que se conoce como “iframe”, un elemento HTML que permite insertar un documento HTML dentro de un documento HTML principal. Esto se muestra en la siguiente *figura 4.101*:



Figura 4.101: Iframe del HTML de three.js embebido en el HTML de la página web principal.

El código del HTML es bastante sencillo, pues básicamente consiste en darle al “iframe” el atributo “src” (aparte de otros muchos posibles) con la dirección del HTML que queremos incrustar en el HTML principal. El código de la *figura 4.102* indica como debe llevarse esta acción a cabo:

```
<div id="Modelo" class="tabcontent">
<h1>"VOLAR SE HACE EN GRAN MEDIDA <br> CON LA IMAGINACIÓN" </h1>
<h2><i>¡Descubre tus posibilidades!</i></h2>
<br>
|<button class="accordion" onclick="openacordeon()"><b>1/1 - Visualice su dron</b></button>
<div class="panel">
<p>
    ¡Bienvenido a la primera pestaña de resultados! Si usted ya ha incluido los datos de sus cargas de pago en la pestaña de "Parámetros de entrada", pulse el siguiente botón para visualizar su dron cargado:
</p>
<input id="calculando_" type="button" class="botones_cargas" onclick="divLogin()" value="Visualizar dron"> </input>
<br><br><br>
</div>

<iframe id="viewer" name="viewer" allow="fullscreen" allow="xr-spatial-tracking" src="./indexAnimaciones.html" style="display: unset;" scrolling="no"></iframe>
<br><br><br><br><br>
</div>
```

Figura 4.102: Llamada al iframe del HTML del dron desde el HTML de la página web

En cuanto a lo sucedido al clicar el botón “Visualizar dron” (ver *figura 4.101*), se deben destacar algunos puntos relevantes respecto del CSS utilizado para añadir a la acción ejecutada un efecto bastante visual y atractivo. Como puede verse en la *figura 4.103*, la altura definida para el “iframe” en el CSS es de 0 píxeles y tiene una transición de altura de 0,4 segundos. Esto es así porque al pulsar el botón “Visualizar dron” de la *figura 4.101*, se llama a la función “divLogin()” del JavaScript mostrada en la *figura 4.104*.

```
#viewer {
    width: 80.15%;
    margin-left: 17.97%;
    margin-top: 0%;
    background-color: #white;
    height: 0px;
    transition: height 0.4s;
    border: 0px solid;
    border-color: #rgb(236, 129, 183);
}
```

Figura 4.103: CSS del iframe.

```
function divLogin(){
    if(clic==0){
        document.getElementById("viewer").style.height = "0%";
        document.getElementById("viewer").style.border = "0px solid";
        swal("ATENCIÓN", "Antes de poder visualizar su modelo, debe clicar el botón 'Guardar y cargar datos' en la pestaña" +
        "'Parámetros de entrada', sección 'Defina las cargas de pago'.", "error");
    } else{
        document.getElementById("viewer").style.height = "85%";
        document.getElementById("viewer").style.border = "1.5px solid";
        swal("PERFECTO", "A continuación se abrirá una nueva pestaña para que usted pueda visualizar su dron.", "success");
    }
}
```

Figura 4.104: Función “divLogin()” del JavaScript de la aplicación.

Así, lo que puede observarse en la *figura 4.104* es que, en caso de haberse pulsado previamente el botón “Guardar y cargar datos” de la *figura 4.92*, la variable “clic” tomará el valor unidad y se redefinirá la altura del iframe hasta el 85% de la altura. Teniendo en cuenta el efecto de transición aplicado en el CSS, quedará un efecto bastante visual.

4.4.2.7. Biblioteca three.js

El “iframe” mostrado en la *figura 4.101* se trata de un HTML (*indexAnimaciones.html*) que se puede leer en el navegador de manera independiente. Así, si abrimos esta página web en el navegador, observamos lo mostrado en la *figura 4.105*.

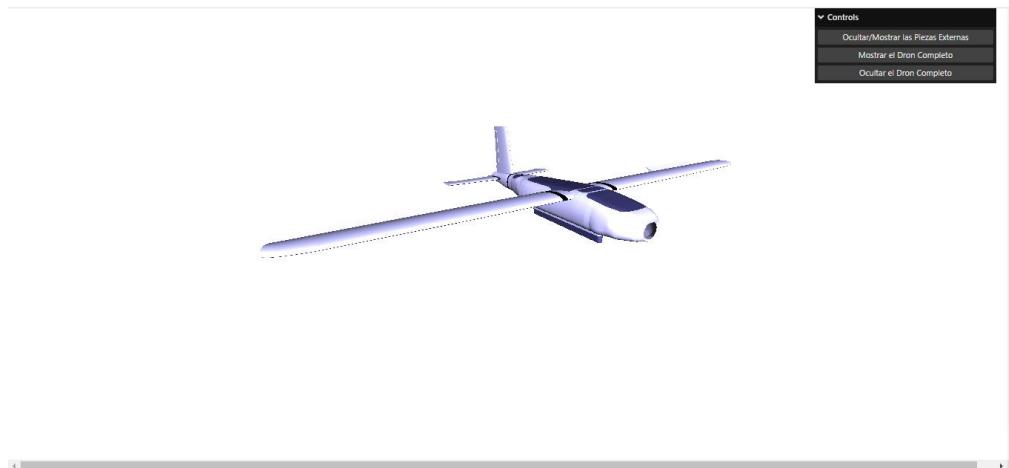


Figura 4.105: Se muestra el contenido del *indexAnimaciones.html*

Este código se trata de un HTML (ver *figura 4.106*) que en realidad es muy sencillo y lo único que tiene es un “div” en el cual se renderiza la imagen mostrada del dron y una llamada al archivo de JavaScript encargado de darle forma, estilo y funcionalidad a la escena empleada para mostrar el dron en la página web.

```
<!DOCTYPE html>
<html lang="en">
<head>

    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, user-scalable=no, minimum-scale=1.0, maximum-scale=1.0">
    <title>Threejs STL+Animaciones</title>

</head>
<body>

    <div id="canvas"></div>
    <script type="module" src="./indexAnimaciones.js"></script>

</body>
</html>
```

Figura 4.106: Se muestra el código del *indexAnimaciones.html*

A continuación, se estudia detenidamente el código del *indexAnimaciones.js*, que es el archivo de JavaScript que permite crear la escena mostrada en la *figura 4.105*. Esta escena se ha hecho con el empleo de la biblioteca *three.js* de JavaScript, empleada para crear y mostrar gráficos animados por computadora en 3D en un navegador web como puede ser, por ejemplo, Google Chrome.

La estructura principal de cualquier programa realizado con three.js incluye los siguientes elementos:

- Escena. Es como si fuera un lienzo donde podemos colocar los elementos gráficos deseados.
- Cámara observando la escena. Las hay de dos tipos: cámara en perspectiva y cámara ortográfica. A continuación, se muestra un esquema (ver *figura 4.107*) sobre los parámetros de la cámara en perspectiva, que ha sido la utilizada en este trabajo.

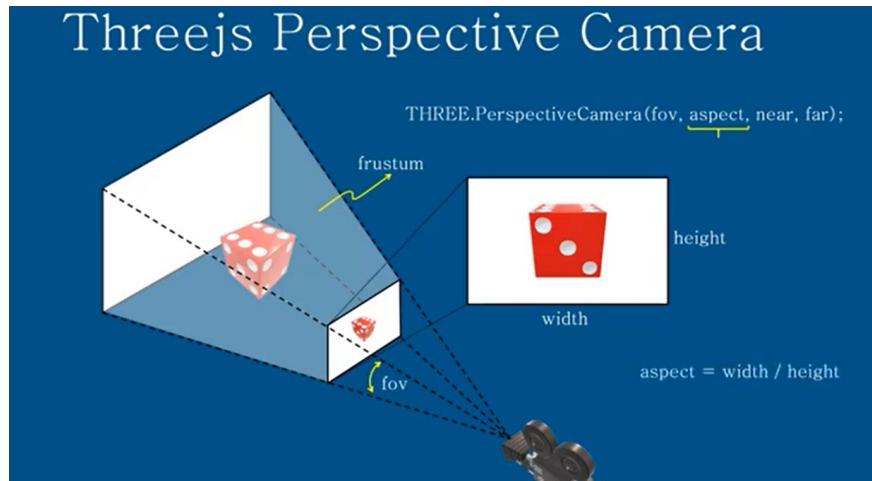


Figura 4.107: Ilustración tomada del canal de YouTube <https://www.youtube.com/c/MonkeyWit> sobre el uso de la cámara en perspectiva.

- Renderer que toma todo lo que está en la escena, bajo el foco de la cámara, y lo muestra.

En una escena a su vez, podemos tener:

- Luces.
- Grupos de objetos.
- Formas (los llamados “mesh”), compuestos a su vez por una geometría, un material y una textura.

En este trabajo en concreto, el JavaScript para la construcción de la escena de three.js comienza con la importación de los módulos necesarios (ver *figura 4.108*) y una extensa declaración de variables (ver *figuras 4.109* y *4.110*):

```

1 //-----IMPORTACIÓN DE MÓDULOS-----
2
3
4 import * as THREE from './Archivosthreejs/three.module.js';
5 import {STLLoader} from './Archivosthreejs/STLloader.js';
6 import {OrbitControls} from './Archivosthreejs/OrbitControls.js';
7 import {EffectComposer} from './Archivosthreejs/EffectComposer.js'; //Añadir efectos de animacion
8 import {RenderPass} from './Archivosthreejs/RenderPass.js';
9 import {OutlinePass} from './Archivosthreejs/OutlinePass.js'; //Resaltar cada elemento
10 import { GUI } from './Archivosthreejs/lil-gui.module.min.js';
11

```

Figura 4.108: Importación de módulos para three.js.

```

12 //----- DECLARACIÓN DE VARIABLES -----
13
14 let scene, camera, renderer, composer, outlinePass;
15
16 let selectedObjects = [];
17
18 let Fuselaje, Semiala_izquierda, Semiala_derecha, Timon_Semiala_derecha, Timon_Semiala_izquierda, Cola_estab_vertical, Timon_estab_vertical,
19 Estab_horizontal_derecho, Timon_estab_horizontal_derecho, Estab_horizontal_izquierdo, Timon_estab_horizontal_izquierdo;
20
21 let Pieza_negra_seccion_fuselaje_cola, Pieza_negra_fuselaje_semiala_izquierda, Pieza_negra_fuselaje_semiala_derecha, Tubo_soporte_BA,
22 Tubo_soporte_BS, Pieza_negra_semiala_derecha, Pieza_negra_semiala_izquierda, TuboBS_de_Semiala_izquierda, TuboBA_de_Semiala_izquierda,
23 Volumen_de_fuselaje_derecho, Volumen_de_fuselaje_izquierdo, Tubo_cola_fuselaje, Pieza_negra_seccion_cola_fuselaje,
24 let Pieza_negra_timon_estab_horizontal_derecho: any negra_fuselaje_estab_horizontal_izquierdo,Pieza_negra_estab_horizontal_derecho,
25 Pieza_negra_timon_estab_horizontal_derecho, Tubo_estab_horizontal_derecho, Tubo_estab_horizontal_izquierdo,
26 Pieza_negra_timon_estab_horizontal_izquierdo, Pieza_negra_estab_horizontal_izquierdo;
27
28 let Patin_izquierdo, Patin_derecho, Tapa1, Tapa2, Tapa3, Tapa4, Tapa_inferior, Pestana_de_union_derecha, Pestana_de_union_izquierdo;
29
30 let Cargas_pago, cdg_DRON, cdg_TOTAL;
31
32 let nombres_blancas = [
33   Fuselaje, Semiala_izquierda, Semiala_derecha, Timon_Semiala_derecha, Timon_Semiala_izquierda, Cola_estab_vertical, Timon_estab_vertical,
34   Estab_horizontal_derecho, Timon_estab_horizontal_derecho, Estab_horizontal_izquierdo, Timon_estab_horizontal_izquierdo
35 ];
36
37 let nombres_negras = [
38   Pieza_negra_seccion_fuselaje_cola, Pieza_negra_fuselaje_semiala_izquierda, Pieza_negra_fuselaje_semiala_derecha, Tubo_soporte_BA,
39   Tubo_soporte_BS, Pieza_negra_semiala_derecha, Pieza_negra_semiala_izquierda, TuboBS_de_Semiala_izquierda, TuboBA_de_Semiala_izquierda,
40   TuboBS_de_Semiala_derecha, TuboBA_de_Semiala_derecha, Tubo_cola_fuselaje, Pieza_negra_seccion_cola_fuselaje,
41   Pieza_negra_fuselaje_estab_horizontal_derecho, Pieza_negra_fuselaje_estab_horizontal_izquierdo,Pieza_negra_estab_horizontal_derecho,
42   Pieza_negra_timon_estab_horizontal_derecho, Tubo_estab_horizontal_derecho, Tubo_estab_horizontal_izquierdo,
43   Pieza_negra_timon_estab_horizontal_izquierdo, Pieza_negra_estab_horizontal_izquierdo
44 ];
45
46 let nombres_otras = [
47   Patin_izquierdo, Patin_derecho, Tapa1, Tapa2, Tapa3, Tapa4, Tapa_inferior, Pestana_de_union_derecha, Pestana_de_union_izquierdo
48 ];
49
50 let nombre_carga_pago = [Cargas_pago];
51
52 let nombre_cdg_dron = [cdg_DRON];
53
54 let nombre_cdg_total = [cdg_TOTAL];
55

```

Figura 4.109: Declaración de variables y su almacenamiento en arrays para luego cargar los objetos y añadirlos a la escena.

```

56 //----- ALMACENAMIENTO DE LAS URLs DE LOS ARCHIVOS STL -----
57
58 let url_blancas = [
59   "./ArchivosSTL/Fuselaje_AlicatPart.stl", "./ArchivosSTL/Semiala_izquierda_AlicatPart.stl", "./ArchivosSTL/Semiala_derecha_AlicatPart.stl",
60   "./ArchivosSTL/Timon_semiala_derecha_AlicatPart.stl", "./ArchivosSTL/Timon_semiala_izquierda_AlicatPart.stl", |
61   "./ArchivosSTL/Cola_estab_vertical_AlicatPart.stl", "./ArchivosSTL/Timon_estab_vertical_AlicatPart.stl",
62   "./ArchivosSTL/Estabilizador_horizontal_derecho_AlicatPart.stl", "./ArchivosSTL/Timon_estab_horizontal_derecho_AlicatPart.stl",
63   "./ArchivosSTL/Estabilizador_horizontal_izquierdo_AlicatPart.stl", "./ArchivosSTL/Timon_estab_horizontal_izquierdo_AlicatPart.stl"
64 ];
65
66 let url_negras = [
67   "./ArchivosSTL/Pieza_negra_seccion_fuselaje_cola_AlicatPart.stl",
68   "./ArchivosSTL/Pieza_negra_fuselaje_semiala_izquierda_AlicatPart.stl", "./ArchivosSTL/Pieza_negra_fuselaje_semiala_derecha_AlicatPart.stl",
69   "./ArchivosSTL/Tubo_soporte_BA_AlicatPart.stl", "./ArchivosSTL/Tubo_soporte_BS_AlicatPart.stl",
70   "./ArchivosSTL/Pieza_negra_semiala_derecha_AlicatPart.stl", "./ArchivosSTL/Pieza_negra_semiala_izquierda_AlicatPart.stl",
71   "./ArchivosSTL/Tubo_BS_de_semiala_izquierda_AlicatPart.stl", "./ArchivosSTL/Tubo_BS_de_semiala_derecha_AlicatPart.stl",
72   "./ArchivosSTL/Tubo_BS_de_semiala_derecha_AlicatPart.stl", "./ArchivosSTL/Tubo_BA_de_semiala_derecha_AlicatPart.stl",
73   "./ArchivosSTL/Tubo_cola_fuselaje_AlicatPart.stl", "./ArchivosSTL/Pieza_negra_seccion_cola_fuselaje_AlicatPart.stl",
74   "./ArchivosSTL/Pieza_negra_fuselaje_estab_horizontal_derecho_AlicatPart.stl",
75   "./ArchivosSTL/Pieza_negra_fuselaje_estab_horizontal_izquierdo_AlicatPart.stl",
76   "./ArchivosSTL/Pieza_negra_estab_horizontal_derecho_AlicatPart.stl",
77   "./ArchivosSTL/Pieza_negra_timon_estab_horizontal_derecho_AlicatPart.stl",
78   "./ArchivosSTL/Tubo_estab_horizontal_derecho_AlicatPart.stl", "./ArchivosSTL/Tubo_estab_horizontal_izquierdo_AlicatPart.stl",
79   "./ArchivosSTL/Pieza_negra_timon_estab_horizontal_izquierdo_AlicatPart.stl",
80   "./ArchivosSTL/Pieza_negra_estab_horizontal_izquierdo_AlicatPart.stl"
81 ];
82
83 let url_otras = [
84   "./ArchivosSTL/Patin_izquierdo_AlicatPart.stl", "./ArchivosSTL/Patin_derecho_AlicatPart.stl",
85   "./ArchivosSTL/Tapa_1_AlicatPart.stl", "./ArchivosSTL/Tapa_2_AlicatPart.stl", "./ArchivosSTL/Tapa_3_AlicatPart.stl",
86   "./ArchivosSTL/Tapa_4_AlicatPart.stl", "./ArchivosSTL/Tapa_inferior_AlicatPart.stl", "./ArchivosSTL/Pestana_union_derecha_AlicatPart.stl",
87   "./ArchivosSTL/Pestana_union_izquierda_AlicatPart.stl"
88 ];
89
90 let url_carga_pago = ["./ArchivosSTL/Parte_CargaPago.stl"];
91
92 let url_cdg_dron = ["./ArchivosSTL/Parte_CuboCDG_ORIGINAL.stl"];
93
94 let url_cdg_total = ["./ArchivosSTL/Parte_CuboCDG_CARGADO.stl"];
95

```

Figura 4.110: URLs de almacenamiento de los objetos que se van a cargar en la escena.

Como puede verse en las dos figuras anteriores, tanto los nombres (variables) asignados a cada uno de los objetos a cargar (piezas del dron, representaciones del CDG y cargas de pago) como las URL de estos objetos se almacenan en “arrays” de JavaScript a causa del método empleado para cargar todas estas piezas, mediante el empleo de bucles.

También podemos darnos cuenta de lo siguiente: todas las piezas cargadas son de formato STL. Three.js es capaz de leer muchos tipos de formatos 3D de piezas, sin embargo, el uso de STL es especialmente conveniente por las características de las tecnologías aplicadas en el Frontend y en el Backend. Todos los archivos de piezas del dron que se cargan en la escena, así como el CDG de solo el dron, son archivos estáticos, es decir, no sufren ningún tipo de cambio durante la ejecución de los códigos empleados. Estos archivos simplemente son leídos y cargados en la escena desde un inicio. Por otro lado, las URL correspondientes a las variables creadas con el nombre “carga_pago” y las creadas con el nombre “cdg_total” son archivos dinámicos, es decir, cambian y se crean nuevamente cada vez que el usuario hace una petición al servidor desde la página web.

Además, es importante decir que las piezas a las que nos referimos como “blancas” las llamamos así por el color que les hemos añadido en three.js, mientras que las llamadas como “negras” son porque el color que se les da en la animación es negro también, siendo “otras” las piezas a las que se les ha asignado otro color diferente.

Sin embargo, si se compara la animación de three.js en la *figura 4.105* con imágenes reales del dron (ver *figura 3.8*, por ejemplo) se observa que las tapas del dron en principio deberían ser “blancas”. Sin embargo, se han metido en “otros” con el fin de poder aplicar diferentes efectos de post-procesado, a las tapas por un lado y al conjunto fuselaje-semialas-cola por otro lado. Se busca poder “esconder” las piezas “blancas” con el fin de aumentar mucho la visibilidad del interior del dron, manteniendo la referencia de las tapas.

Los siguientes pasos seguidos por el código son: definición de los colores que se asignarán a una pieza o grupo de piezas, definición de variables necesarias para el funcionamiento de algunas funcionalidades de la escena y llamada a las dos funciones principales de la escena (ver *figura 4.111*).

```

96 //-----DECLARACIÓN DE COLORES APLICADOS A LOS STL EN LA ESCENA-----
97
98 let color_blancas = {color: "rgb(190, 190, 255)"};
99 let color_negras = {color: "rgb(0, 0, 0)"};
100 let color_otras = {color: "rgb(50, 50, 100)"};
101 let color_carga_pago = {color: "rgb(0, 50, 100)"};
102 let color_cdg_dron = {color: "rgb(255, 0, 0)"};
103 let color_cdg_total = {color: "rgb(0, 255, 0)"};
104 let colores_piezas = [color_blancas, color_negras, color_otras, color_carga_pago, color_cdg_dron, color_cdg_total];
105
106 //-----DECLARACIÓN DE VARIABLES DE CONTROL NECESARIAS PARA ESCENA-----
107
108 let loader = new STLLoader();
109 const raycaster = new THREE.Raycaster();
110 const mouse = new THREE.Vector2();
111
112 //-----LLAMADA A LAS FUNCIONES-----
113
114 init();
115 animate();
116

```

Figura 4.111: Declaración de colores empleados para los objetos y llamada a las funciones principales.

Ahora, se explicará el funcionamiento de estas funciones. La primera de ellas es la función “animate()” en la *figura 4.112*, la cual está corriendo en bucle. Esta función se encarga de, constantemente, coger la escena y lo que haya en ella, ver cómo está colocada la cámara

en cada momento y renderizar dicha escena, es decir, mostrar la escena actualizada en función de las acciones que pudieran haberse realizado, por ejemplo, con el ratón. Además, renderiza el “composer”, una variable que almacena los efectos de animación de la escena. En definitiva, esta función permitirá ver en todo momento la versión actualizada de la escena.

```

117 //-----
118
119 function animate(){
120     requestAnimationFrame(animate);
121     renderer.render(scene, camera);
122     composer.render();
123     renderer.clearDepth();
124 }
125

```

Figura 4.112: Código de la función “animate()”.

En las próximas *figuras 4.113, 4.114, 4.115 y 4.116* se muestran todas las partes de la función “init()”, pudiendo observarse: la creación de la escena, la definición de la cámara, su posicionamiento mediante el método “camera.position.set()” y el punto hacia donde dirigirla para mirar mediante el método “camera.lookAt()”. Además, se añade la cámara a las dos “capas” (“layers”, en inglés) que aparecen en la escena. Se definen cuatro luces direccionales mediante el bucle de las líneas 143 a la 152, tras lo cual se renderiza la escena y se carga en el “canvas” definido en el HTML de la *figura 4.106*.

```

126 //-----  

127  

128 function init(){  

129     //-----INICIALIZACIÓN DE ESCENA-----  

130     scene = new THREE.Scene();  

131  

132     camera = new THREE.PerspectiveCamera(75, window.innerWidth / window.innerHeight, 0.1, 10000);  

133     camera.position.set(0,0,10);  

134     camera.lookAt(0,0,0);  

135  

136     camera.layers.enable( 50 );    // enabled by default  

137     camera.layers.enable( 40 );  

138     scene.add( camera );  

139  

140     var posiciones = [10, -10];  

141     for (let i = 0; i < 3; i++) {  

142         let light_50 = new THREE.DirectionalLight("rgb(110, 110, 110)");  

143         let light_50_2 = new THREE.DirectionalLight("rgb(110, 110, 110)");  

144         let position_1 = posiciones[i];  

145         light_50.position.set(0,0,position_1);  

146         light_50_2.position.set(position_1, 0, 0);  

147         light_50.layers.enable( 50 );  

148         light_50_2.layers.enable( 50 );  

149         camera.add( light_50, light_50_2 );  

150     }  

151  

152     //-----  

153     renderer = new THREE.WebGLRenderer({  

154         alpha: true,                      // hace background blanco/transparente  

155         antialias: false                 // mejora calidad de las líneas  

156     });  

157  

158     renderer.setSize(window.innerWidth, window.innerHeight);  

159     document.getElementById("canvas").appendChild(renderer.domElement);  

160  

161     //-----  

162     //-----BUCLES DE CARGA DE PIEZAS (POR GRUPOS DE COLORES Y DE CAPAS)-----  

163  

164     for (let i = 0; i < nombres_blancas.length; i++) {  

165         loader.load(url_blancas[i], (model)=>  

166             nombres_blancas[i] = new THREE.Mesh(model,new THREE.MeshLambertMaterial(colores_piezas[0]));  

167             nombres_blancas[i].scale.set(0.1, 0.1, 0.1);  

168             nombres_blancas[i].position.set(-1,2,0);  

169             nombres_blancas[i].rotation.set(-Math.PI/2.5, -Math.PI/30, -Math.PI/2.8);  

170             nombres_blancas[i].layers.set(50);  

171             scene.add(nombres_blancas[i]);  

172         });
173     }

```

Figura 4.113: Parte 1 del código de la función “init()”.

Lo siguiente en esta función es cargar todas las piezas almacenadas en los “arrays” inicialmente citados. Se añade color, se escala, se posiciona y se rota a cada una de las piezas para luego añadirlas a la escena. Solo a las piezas “blancas”, llamadas “externas” de aquí en adelante se les añade a un “layer” identificado con el número 50. Los CDG de solo el dron y CDG total (dron más cargas de pago) se añaden a otro “layer” identificado con el 40.

En determinado momento, solicité a mi compañera Virginia que generara los STL de todas las piezas del dron desde Catia y mediante el uso de Python, de forma automática. Dicha generación de STLs debía cumplir el requisito de estar referidos a los mismos ejes. Ella logró referir todos los STL a los “ejes globales” de Catia, de forma que todas las piezas del dron estaban ya listas para ser cargadas en la página web. Recomiendo leer el TFG de Virginia dentro de unos meses para poder entender mejor este proceso.

```

174  for (let i = 0; i < nombres_negras.length; i++) {
175    loader.load(url_negras[i], (model)=>{
176      nombres_negras[i] = new THREE.Mesh(model,new THREE.MeshLambertMaterial(colores_piezas[1]));
177      nombres_negras[i].scale.set(0.1, 0.1, 0.1);
178      nombres_negras[i].position.set(-1,2,0);
179      nombres_negras[i].rotation.set(-Math.PI/2.5, -Math.PI/30, -Math.PI/2.8);
180      scene.add(nombres_negras[i]);
181    });
182  }
183  for (let i = 0; i < nombres_otras.length; i++) {
184    loader.load(url_otras[i], (model)=>{
185      nombres_otras[i] = new THREE.Mesh(model,new THREE.MeshLambertMaterial(colores_piezas[2]));
186      nombres_otras[i].scale.set(0.1, 0.1, 0.1);
187      nombres_otras[i].position.set(-1,2,0);
188      nombres_otras[i].rotation.set(-Math.PI/2.5, -Math.PI/30, -Math.PI/2.8);
189      scene.add(nombres_otras[i]);
190    });
191  }
192  for (let i = 0; i < nombre_carga_pago.length; i++) {
193    loader.load(url_carga_pago[i], (model)=>{
194      nombre_carga_pago[i] = new THREE.Mesh(model,new THREE.MeshLambertMaterial(colores_piezas[3]));
195      nombre_carga_pago[i].scale.set(0.1, 0.1, 0.1);
196      nombre_carga_pago[i].position.set(-1,2,0);
197      nombre_carga_pago[i].rotation.set(-Math.PI/2.5, -Math.PI/30, -Math.PI/2.8);
198      nombre_carga_pago[i].layers.set(40);
199      scene.add(nombre_carga_pago[i]);
200    });
201  }
202  for (let i = 0; i < nombre_cdg_dron.length; i++) {
203    loader.load(url_cdg_dron[i], (model)=>{
204      nombre_cdg_dron[i] = new THREE.Mesh(model,new THREE.MeshLambertMaterial(colores_piezas[4]));
205      nombre_cdg_dron[i].scale.set(0.1, 0.1, 0.1);
206      nombre_cdg_dron[i].position.set(-1,2,0);
207      nombre_cdg_dron[i].rotation.set(-Math.PI/2.5, -Math.PI/30, -Math.PI/2.8);
208      nombre_cdg_dron[i].layers.set(40);
209      scene.add(nombre_cdg_dron[i]);
210    });
211  }
212  for (let i = 0; i < nombre_cdg_total.length; i++) {
213    loader.load(url_cdg_total[i], (model)=>{
214      nombre_cdg_total[i] = new THREE.Mesh(model,new THREE.MeshLambertMaterial(colores_piezas[5]));
215      nombre_cdg_total[i].scale.set(0.1, 0.1, 0.1);
216      nombre_cdg_total[i].position.set(-1,2,0);
217      nombre_cdg_total[i].rotation.set(-Math.PI/2.5, -Math.PI/30, -Math.PI/2.8);
218      nombre_cdg_dron[i].layers.set(40);
219      scene.add(nombre_cdg_total[i]);
220    });
221  }

```

Figura 4.114: Parte 2 del código de la función “init()”.

En la siguiente imagen de código (*figura 4.115*) se muestra la creación de la GUI y la definición de los procesos que deben suceder al pulsar cada uno de los botones de dicha interfaz. Destaca que, mediante el uso de dos “layer” diferentes, se consigue hacer aparecer el CDG total y CDG del dron en todo momento.

Además, en la siguiente figura, se incluyen los “controles del ratón” y el “post-procesado” de la escena. Respecto a este último destaca el efecto “OutlinePass”, que permite a los usuarios resaltar el borde de las piezas señaladas con el ratón. De esta forma, estarían ya añadidos los efectos de imagen buscados: resaltar piezas y esconder/mostrar piezas.

```

223 //-----PROGRAMACIÓN DE LAS ACCIONES APLICADAS A LAS "CAPAS" AL CLICAR UNO U OTRO BOTÓN-----
224 const layers = (
225   'Ocultar/Mostrar las Piezas Externas': function() {
226     camera.layers.toggle(50);
227     camera.layers.enable(40);
228   },
229   'Mostrar el Dron Completo': function () {
230     camera.layers.enableAll();
231   },
232   'Ocultar el Dron Completo': function () {
233     camera.layers.disableAll();
234   }
235 );
236
237
238
239
240
241 //-----AGREGAR Graphical User Interface Y CONTROLES DEL RATÓN-----
242
243 const gui = new GUI();
244 gui.add(layers, 'Ocultar/Mostrar las Piezas Externas');
245 gui.add(layers, 'Mostrar el Dron Completo');
246 gui.add(layers, 'Ocultar el Dron Completo');
247
248 let control = new OrbitControls(camera, renderer.domElement);
249 control.enablePan = false;
250 control.minDistance = 2;
251 // control.maxDistance = 19;
252 control.enableDamping = true;
253 control.dampingFactor = 0.05;
254 control.maxPolarAngle = Math.PI; //opción por defecto, es el máximo
255 control.target.set(3, 2, 5);
256
257 //-----POST-PROCESSING-----
258
259 composer = new EffectComposer(renderer);
260 const renderPass = new RenderPass(scene, camera);
261 composer.addPass(renderPass);
262
263 outlinePass = new outlinePass(new THREE.Vector2(window.innerWidth, window.innerHeight), scene, camera, layers);
264 outlinePass.edgeStrength = 8;
265 outlinePass.visibleEdgeColor.set("#0000ff");
266 outlinePass.hiddenEdgeColor.set("#ff0000");
267 composer.addPass(outlinePass);
268
269 window.addEventListener("resize", onWindowResize);
270 renderer.domElement.style.touchAction = "none";
271 renderer.domElement.addEventListener("pointermove", onPointerMove);
272

```

Figura 4.115: Parte 3 del código de la función “init()”.

Finalmente, se incluyen al final del código las funciones (ver *figura 4.116*) involucradas en el correcto funcionamiento de la escena. Estas funciones no han sido programadas durante este proyecto, sino que se han tomado de otros proyectos de three.js, pues a día de hoy no existe ningún manual minucioso sobre el uso de esta biblioteca. Por tanto, hoy en día, la mejor forma de aprender a manejarla es ir a la página web <https://threejs.org/> y descargar todos los códigos (ver *figura 4.117*) de ejemplo que tienen ahí alojados. Leyendo y entendiendo estos códigos de ejemplo es como, hoy en día, se elaboran nuevas escenas. Es por esto por lo que las funciones de la *figura 4.116* se han tomado directamente de algunos de los ejemplos base utilizados para desarrollar esta escena.

```

273 //----> FUNCIONES PARA LAS ANIMACIONES
274
275 async function onPointerMove(event){
276   if(event.isPrimary === false) return;
277
278   mouse.x = (event.clientX/window.innerWidth)*2 - 1;
279   mouse.y = -(event.clientY/window.innerHeight)*2 + 1;
280
281   checkIntersection();
282 }
283
284 sync function checkIntersection() {
285   raycaster.setFromCamera(mouse, camera);
286   const intersects = raycaster.intersectObject(scene, true);
287
288   if (intersects.length > 0) {
289     const selectedObject = intersects[0].object;
290     addSelectedObject(selectedObject);
291     outlinePass.selectedObjects = selectedObjects;
292   } else {
293     //outlinePass.selectedObjects = [];
294   }
295 }
296
297 sync function addSelectedObject(object){
298   selectedObjects = [];
299   selectedObjects.push(object);
300 }
301
302 sync function onWindowResize(){
303   camera.aspect = window.innerWidth / window.innerHeight;
304   camera.updateProjectionMatrix();
305   renderer.setSize( window.innerWidth, window.innerHeight );
306   composer.setSize( window.innerWidth, window.innerHeight );
307 }

```

Figura 4.116: Parte 3 del código de la función “init()”.

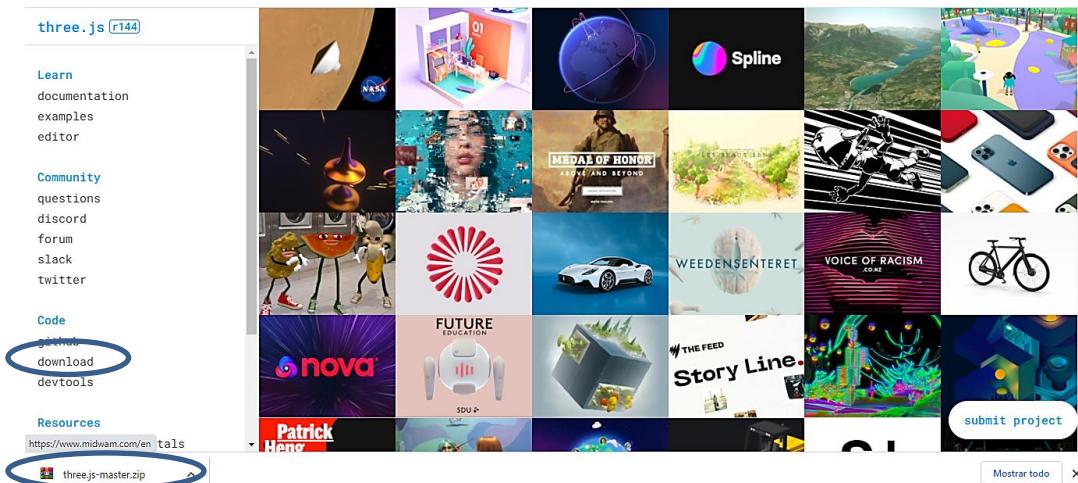


Figura 4.117: Proceso de descarga de todos los códigos de ejemplo de la página web <https://threejs.org/>

Para ver los resultados obtenidos mediante el empleo de la GUI, debe leerse el “*Capítulo 5. Testeo y Validación.*”

Durante el desarrollo de la animación en three.js se trató de aplicar el efecto mostrado en https://mediawing.jp/js/three.js/examples/webgl_postprocessing_ssrr.html donde se ve que introduciendo un valor de IOR = 1, puede hacerse “desaparecer” una pieza con un clic. Sin embargo, esta animación “ssrrPass”, aparte de no responder de manera precisa a las órdenes del ratón, no puede ponerse en conjunción con la animación “OutlinePass”. Es decir, un efecto de post-procesado desinhibe la actuación del otro efecto y viceversa, de forma que en la escena solo renderizará el efecto situado más abajo en el código. Por estos

problemas se decidió emplear “layers” para solo dar a algunos objetos la propiedad de esconderse/mostrarse, permitiendo todavía utilizar el efecto “OutlinePass” en el resto de los objetos de la escena. Este comportamiento anómalo que impide tener dos efectos a la vez funcionando sobre una misma pieza es indicado por la organización de three.js en su página web, pero todavía sigue sin resolverse.

4.4.2.8. Escritura de resultados finales

La forma de sacar por pantalla los resultados numéricos buscados para que el usuario pudiera verlos cómodamente es la siguiente. En la sección “Resultados finales y recomendaciones” de la página web, el usuario se encuentra con la siguiente vista (ver figura 4.118) cuando todavía no ha o completado o “guardado y cargado” los datos requeridos en ninguna de las dos opciones de cálculo de la página.

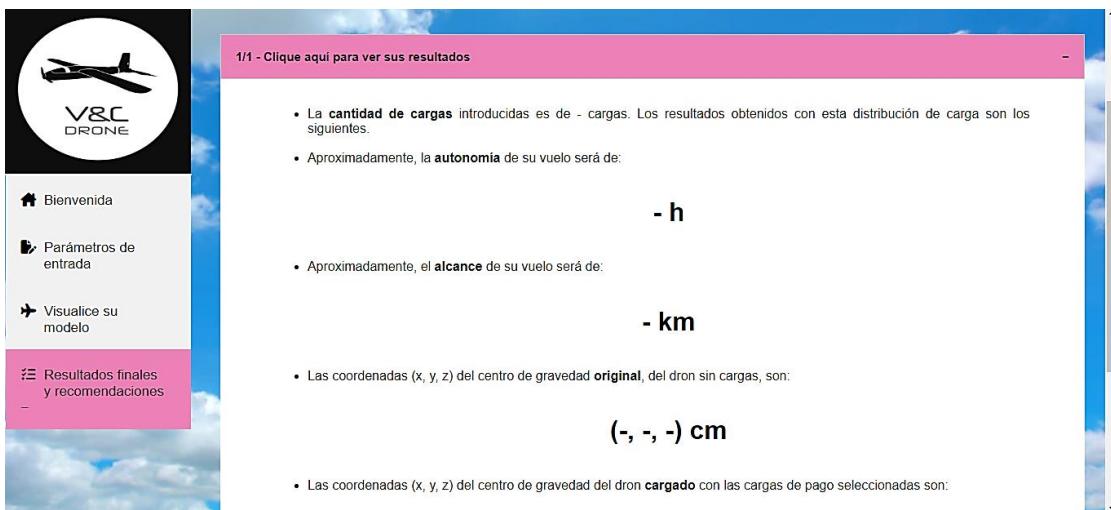


Figura 4.118: Sección de resultados de la página web.

El código HTML del acordeón que propicia esta vista es el mostrado en la figura 4.119:

```
<button class="accordion" onclick="openacordeon()><b>1/1 - Clique aquí para ver sus resultados</b></button>
<div class="panel">
<br>
<ul>
  <li id="numero_cargas">
    La <b>cantidad de cargas</b> introducidas es de - cargas. Los resultados obtenidos con esta distribución de carga son los siguientes.
  </li>
  <br>
  <li>Aproximadamente, la <b>autonomía</b> de su vuelo será de: </li>
  <h1 id="horas" class="resultado">- h</h1>
  <li>Aproximadamente, el <b>alcance</b> de su vuelo será de: </li>
  <h1 id="kilometros" class="resultado">- km</h1>
  <li>Las coordenadas (x, y, z) del centro de gravedad <b>original</b>, del dron sin cargas, son: </li>
  <h1 id="cdg_original" class="resultado">(-, -, -) cm</h1>
  <li>
    Las coordenadas (x, y, z) del centro de gravedad del dron <b>cargado</b> con las cargas de pago seleccionadas son:
  </li>
  <li>
    <h1 id="cdg_cargado" class="resultado">(-, -, -) cm</h1>
  </li>
  <br>
  Con el motivo de hacer que su vuelo sea el mejor posible, las <b>sugerencias</b> que desde V&C Drone le queremos hacer son las siguientes:
  <br>
  <p id="sugerencias"><b>"De momento no hay sugerencias."</b></p>
</ul>
<br><br>
</div>
```

Figura 4.119: Código HTML que estructura la figura 4.118.

Donde se observa que a cada etiqueta (o “tag”) con la clase “resultado” se le ha añadido un identificador. Así, si vemos el código (ver *figura 4.120*) del envío de datos y recepción de resultados del JavaScript correspondiente a, por ejemplo, los cálculos de la autonomía y el alcance, se puede entender cómo podemos pasar de la *figura 4.118* a la *figura 4.121*.

```
//FUNCIÓN QUE ENVÍA EL JSON DE LA AUTONOMÍA Y EL ALCANCE AL SERVIDOR FLASK
async function Enviar_AyA(ourjson){

    const response_AyA = await fetch(`http://127.0.0.1:5000/datos_AyA/${ourjson}`);
    const data_AyA = await response_AyA.json();

    let autonomia_def = parseFloat(data_AyA[0]);
    let alcance_def = parseFloat(data_AyA[1]);

    let autonomia_definitiva = Number(autonomia_def.toFixed(3));
    let alcance_definitivo = Number(alcance_def.toFixed(3));

    document.getElementById("horas").innerHTML = autonomia_definitiva + " h";
    document.getElementById("kilometros").innerHTML = alcance_definitivo + " km";

    swal("PERFECTO", "Los datos se han enviado y cargado correctamente. Ya puede ver sus resultados.", "success");
}

}
```

Figura 4.120: Función de JavaScript que envía los datos necesarios para calcular la autonomía y el alcance, devolviendo estos valores y escribiéndolos en el HTML.



Figura 4.121: HMI de la aplicación, que muestra los resultados de autonomía y alcance obtenidos.

Tras haber introducido los datos típicos mencionados en la sección de “Parámetros de entrada” de la aplicación, estos fueron los resultados obtenidos, viendo como el método “document.getElementById().innerHTML” mostrado en la *figura 4.119* es capaz de escribir lo que le pidamos en el elemento HTML con el “id” indicado. Este es el paso final de la implementación en este trabajo.

4.4.3. Presentación del Fluograma

En este apartado se muestra el diagrama de flujo (o fluograma) de la aplicación web diseñada en este proyecto, a partir de la *página 104*. Se pretende utilizar este diagrama como apoyo principal para realizar el análisis funcional de la página web.

Destaca que el grado de detalle empleado en este caso ha sido superficial, es decir, el diagrama se centra en explicar la funcionalidad de la aplicación desde el punto de vista de la utilización de la misma por parte de un usuario cualquiera. Sin embargo, sí es cierto que en algunas partes del diagrama se han incluido algunos pasos que en principio un usuario no podría ver, pues por medio del HMI se “esconde” la lógica que ocurre detrás de la interfaz de usuario. A continuación, se describirá paso a paso cada uno de los caminos seguidos desde la apertura de la aplicación hasta la obtención final de los resultados buscados.

El diagrama de flujo [87][88][89] mostrado comienza con el “Inicio de la página web”, siendo este el momento en que la abrimos por primera vez o la refrescamos. Una vez dentro de la aplicación, se recomienda bastante al usuario leer las instrucciones y comentarios incluidos en la sección de “Bienvenida”, pues servirá para ayudarle a entender completamente el funcionamiento, si bien simple, de la página. Tras esto, el usuario puede clicar en la siguiente sección: “Parámetros de entrada”, donde se le mostrarán dos acordeones, es decir, dos opciones (por ello se emplea un selector múltiple en este fluograma). El usuario puede elegir abrir el primer acordeón (1) destinado a la realización de cálculos sobre el CDG del dron y su distribución de cargas, donde tendrá que definir todas sus cargas de pago. También, puede seleccionar el segundo acordeón disponible (2), destinado al cálculo de la autonomía y el alcance de la aeronave en función de los datos de las baterías, velocidad de vuelo, etc. En definitiva, a raíz de este punto, se abren dos posibles ramas de cálculo que son independientes la una de la otra pues, como se mostró en la *figura 4.100*, el servidor montado dispone de dos rutas por las que le pueden llegar los datos necesarios. A continuación, se muestran los pasos seguidos desde la entrada de datos hasta la obtención de los resultados, en función de la rama (o camino) seguida.

4.4.3.1. El usuario elige el camino 1 (“Defina las cargas de pago”)

1. Primero, el usuario debe introducir manualmente la cantidad de cargas que desea tener mediante los botones “Añadir carga” y “Eliminar la última carga”, introduciendo sus datos, tal y como se muestra en la *figura 4.122*. Entonces el usuario se pregunta si ha introducido todas las cargas que deseaba; si la respuesta es negativa entonces deberá seguir utilizando estos botones e introduciendo los datos necesarios; si la respuesta es positiva, entonces sale del bucle y se dirige al paso dos.

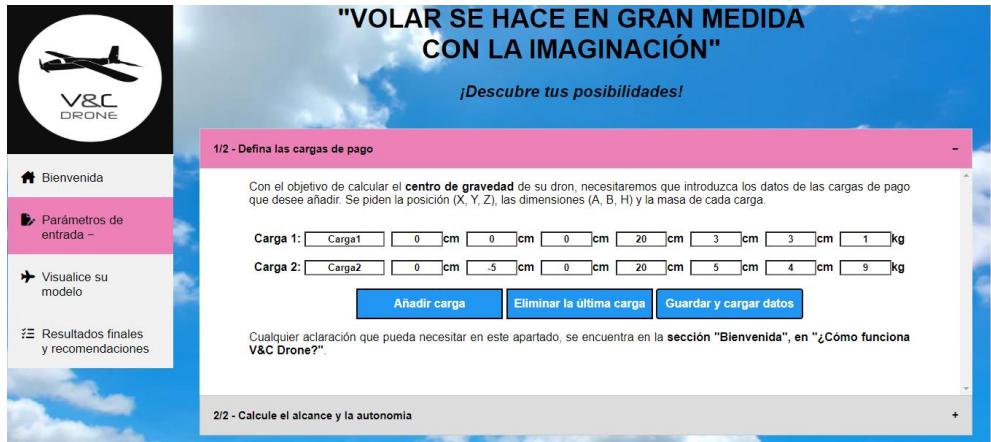


Figura 4.122: Datos introducidos por un usuario en el apartado de la definición de cargas de pago.

2. Ya con todas las cargas deseadas y sus datos introducidos, se debe confirmar que el servidor que va a recoger estos datos está encendido (“escuchando”); en caso de no estarlo, deberá encenderse. Este es un procedimiento que, en principio, el usuario final del proyecto no tendrá que hacer en fases avanzadas del mismo
3. Con el servidor encendido, se pulsa el botón “Guardar y Calcular”.
4. Entonces el programa se pregunta si el número de cargas introducidas por el usuario es nulo, en cuyo caso se envía un mensaje de “aviso” al usuario para advertirle de que la operación que ha realizado le va a llevar a ver el dron vacío (sin cargas), enviándose los datos al servidor. Si el número de cargas es distinto de cero, el programa evaluará si alguno de los datos introducidos es inválido o si falta por introducir algún dato. Si hay algún error de este tipo, saltará un mensaje de error y no se enviará nada al servidor; si todos los datos enviados son correctos, se enviarán al servidor.
5. Tras aproximadamente 45 segundos de espera (tiempo de realización de todas las operaciones del Backend), el servidor devuelve los resultados esperados de CDG, sugerencias y comentarios, y genera los ficheros STL del “CDG total actualizado” y “de las cargas de pago”, que van a ser leídos por el archivo “indexAnimaciones.js” de JavaScript al cargar el dron. Cuando vuelve la respuesta, salta un mensaje de éxito.
6. El usuario deberá dirigirse a la sección “Visualice su modelo” donde, tras clicar sobre el acordeón y pulsar el botón “Visualizar dron”, este podrá ver su modelo.
7. Si el usuario decidiese cambiar algo de su diseño en este punto, como cambiar la distribución de las cargas o su peso, etc. deberá dirigirse a la sección de “Parámetros de entrada” y comenzar de nuevo el proceso. Si el usuario está conforme con su modelo, entonces deberá continuar con los siguientes pasos.
8. Emplear el clic izquierdo del ratón para rotar la escena y la rueda del mismo para hacer zoom/alejar. Se recomienda al usuario pulsar los botones de la GUI en este orden: “Ocultar/mostrar las piezas externas” (aumentará mucho la visibilidad de la escena y del interior del dron, manteniendo la referencia de las tapas) → “Ocultar

el dron completo” → “Ocultar/mostrar las piezas externas” (solo se mostrarán el fuselaje, la cola y las semialas; sin las tapas, ni patines ni otras piezas pequeñas) → “Mostrar el dron completo”.

9. El usuario debe dirigirse ahora a la sección titulada “Resultados finales y Recomendaciones” para ver por pantalla sus resultados del CDG del dron sin cargar, del CDG del dron cargado, y ver las sugerencias prestadas. De esta forma se habrá llegado al final de esta rama del flujoograma.

4.4.3.2. El usuario elige el camino 2 (“Calcular el alcance y la autonomía”)

1. Primero, el usuario debe introducir manualmente los datos requeridos para hacer estos cálculos, como se muestra en la figura 4.123.

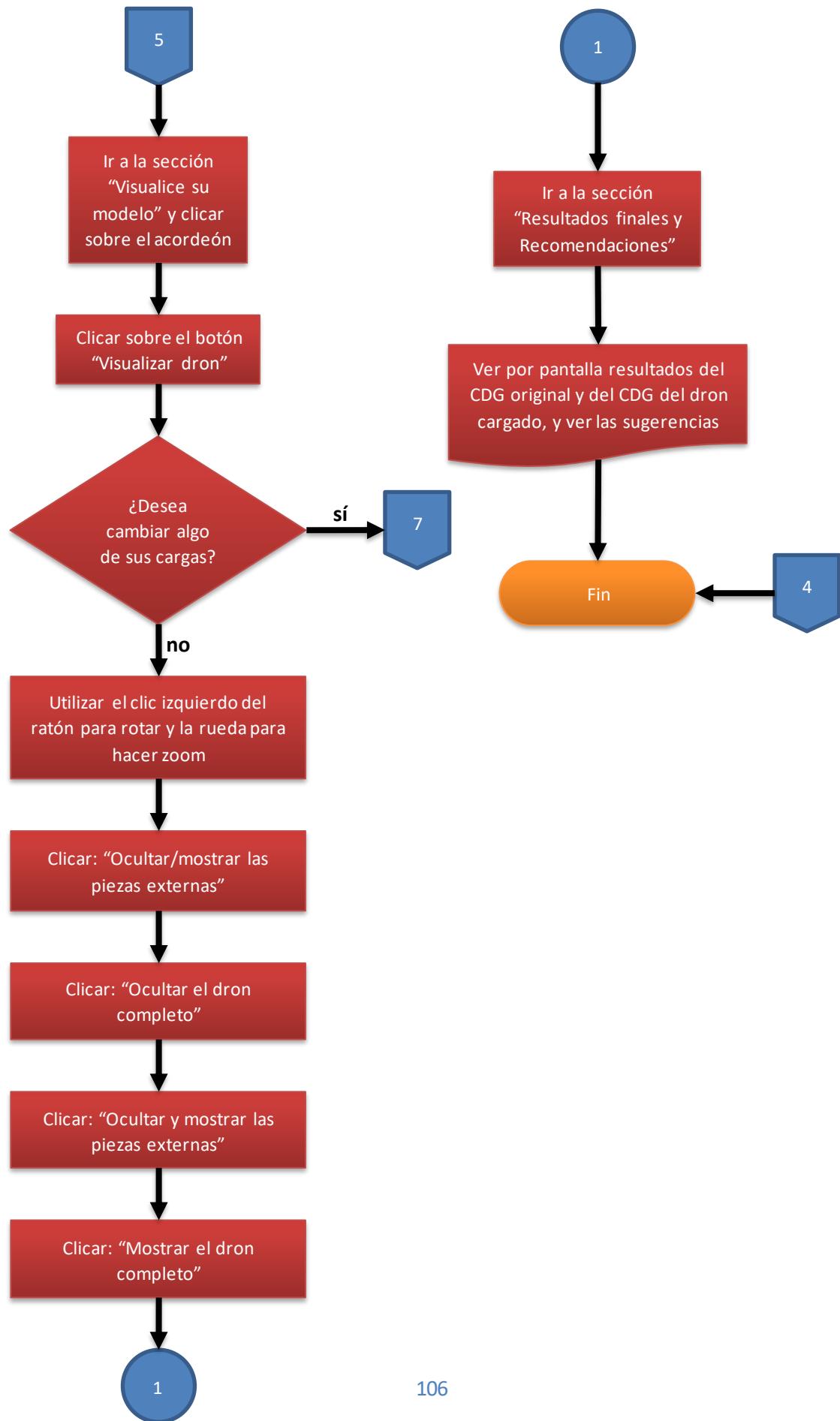
Figura 4.123: Datos introducidos por un usuario en el apartado del cálculo del alcance y la autonomía.

2. Luego, se debe comprobar que el servidor que va a recoger estos datos esté encendido; en caso de no estarlo, deberá encenderse. Cabe destacar que este es un procedimiento que en principio el usuario final del proyecto no tendrá que hacer en fases avanzadas del mismo.
3. Con el servidor activo, se pulsa el botón “Guardar y Calcular”.
4. Antes de enviarse los datos al servidor, el programa comprueba internamente que todos los datos introducidos son válidos. En caso de haber algún dato erróneo o algún campo a falta de llenar, se devolverá un mensaje de error al usuario que le hará tener que volver al paso 1, reescribiendo correctamente los datos que hayan dado error. Si todos los datos fueran correctos entonces se envían al servidor.
5. El servidor devuelve los resultados esperados de la autonomía y el alcance y lanza un mensaje de éxito.
6. Nos dirigimos a la sección “Resultados finales y recomendaciones” para visualizar los resultados obtenidos, finalizando el flujo del diagrama en este punto.

Funcionalidad de la Aplicación







Capítulo 5

Testeo y Validación

8.1. Validación del Modelo del Dron en Catia

Durante la misión realizada el “Día de la Prueba de Vuelo” uno de los enganches de la tapa 2 del dron se rompió. Este hecho tuvo lugar unos segundos después de que el dron iniciase el vuelo de crucero, y tuvo como consecuencia la pérdida de la tapa mencionada.

La tapa 2 salió despedida del aparato y cayó en algún punto del aeródromo de Sigüenza. Por suerte, el dron pudo mantener la estabilidad y aterrizar sin problema. A pesar de las batidas de búsqueda organizadas aquel día, tras dos horas de búsqueda, la tapa se dio por perdida.

En ese momento se plantearon varias alternativas en lo referente al próximo vuelo programado para la semana siguiente al suceso. Algunas de ellas fueron:

- Comprar un nuevo recambio de la pieza perdida.
- Utilizar algún elemento (ej. un trozo de contrachapado) que más o menos pudiera sustituir la tapa.
- Utilizar un modelo en 3D de la tapa perdida para hacer su impresión 3D.

Tras un tiempo de deliberación por parte del equipo presente aquel día, se decidió seguir para adelante con la tercera opción.

En aquel momento, Virginia y yo ya habíamos completado el modelo 3D de todas las partes del dron en Catia, entre ellas, el modelo de la tapa 2. Fue este el motivo por el cual Luis Izquierdo nos encargó realizar la impresión 3D de esta pieza en el Departamento de Aeronaves y Vehículos Espaciales (DAVE), en la ETSIAE.

Más adelante, en el TFG de Virginia se explicará en detalle cuáles fueron los programas utilizados para imprimir, parámetros de impresión y los problemas y aciertos durante la impresión. En este trabajo se explicarán los problemas posteriores a la impresión que están relacionados con los parámetros de la misma, por ejemplo, el peso excesivo de la pieza.

En este TFG se tratará la bondad del modelo obtenido en Catia V5, lo fiable de las medidas empleadas para realizarlo y el éxito/fracaso en la colocación de la tapa impresa en el dron real.

Con el motivo de reducir al máximo posible el error cometido en las medidas de la tapa 2 durante su primera medición, se decidió realizar una segunda medición de las dimensiones de la pieza con el fin de reducir lo máximo posible el error cometido en el diseño del modelo. Reduciendo el error absoluto de las medidas, aumentaba la confianza en el hecho de que el modelo de la tapa pudiera encajar en el dron real.

Algunas de las segundas medidas tomadas no coincidieron totalmente con las primeras mediciones por lo que se decidió hacer una media de las mismas y utilizar los valores obtenidos. Finalmente, el modelo mostrado en la *figura 4.54* de la tapa 2 se imprimió, tal y como se muestra en la *figura 3.26*. de este documento.

El proceso de validación del modelo 3D de la tapa y, por tanto, del fuselaje diseñado en Catia (pues la tapa deriva del diseño previo del fuselaje) se puede observar en la *figura 5.1*. En esta figura se observa una imagen de la tapa 2 impresa y otras tres imágenes donde se coloca la tapa impresa en el hueco correspondiente a la tapa original perdida.



Figura 5.1: Comprobación de la validez del modelo 3D de la Tapa 2 en el dron real.

Gracias a estas tres últimas imágenes se puede afirmar que, efectivamente, la tapa 2 impresa no encajó donde debería haber entrado. Sin embargo, sí es cierto que esta pieza no encaja en el hueco correspondiente por muy pocos milímetros de error, pudiéndose incluso lijar la tapa y conseguir colocarla.

Definitivamente, la conclusión que se puede extraer de esta validación es que la forma y las dimensiones de la tapa del modelo son prácticamente iguales a la tapa real. El error en las medidas puede venir propiciado por la falta de instrumentación más adecuada para tomar ciertas medidas de difícil acceso y por el propio error de los instrumentos de medida. Por tanto, a pesar de estos errores, se puede considerar que el modelo desarrollado en este TFG se trata de un modelo bastante válido, debido a la forma y las dimensiones prácticamente iguales que se obtuvieron. Si bien es cierto, sería necesario realizar más comprobaciones con el fin de asegurar la fiabilidad de esta afirmación. De momento, con la comprobación realizada, se piensa que se puede tratar y que se trata de un modelo válido.

8.2. Testeo de la Interfaz Web

Para testear el funcionamiento de la interfaz web se procede a analizar paso por paso la experiencia con el uso de la aplicación de un usuario cualquiera (no profesional). Cabe destacar que la experiencia, en principio, debería ser bastante similar a lo descrito por el diagrama de flujo del capítulo anterior.

Como primer paso nuestro usuario ficticio comenzaría por abrir la página web. Porque así está diseñado por defecto, continuaría su incursión en la página web por el tab de “Bienvenida” (ver figura 5.2). Se espera que un usuario razonable leyera todo lo indicado en esta sección, es decir, los tres acordeones de la misma, de forma que el usuario pudiera pasar a la siguiente sección con una buena comprensión sobre el funcionamiento de la aplicación.

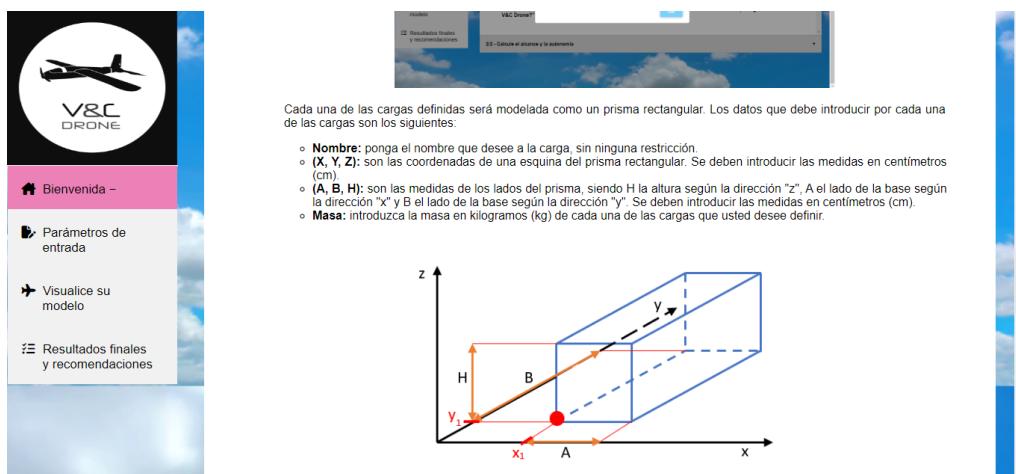


Figura 5.2: Esta imagen representa el proceso de lectura de la sección “Bienvenida”.

Una vez finalizada la lectura de la sección “Bienvenida”, el usuario debería clicar sobre la sección “Parámetros de entrada”. Una vez aquí podrá abrir ya sea el primer o el segundo acordeón, indistintamente, dependiendo de lo que desee hacer. Suponiendo que pinche sobre el primer acordeón “Defina las cargas de pago”, se abrirá el menú mostrado en la siguiente figura 5.3:

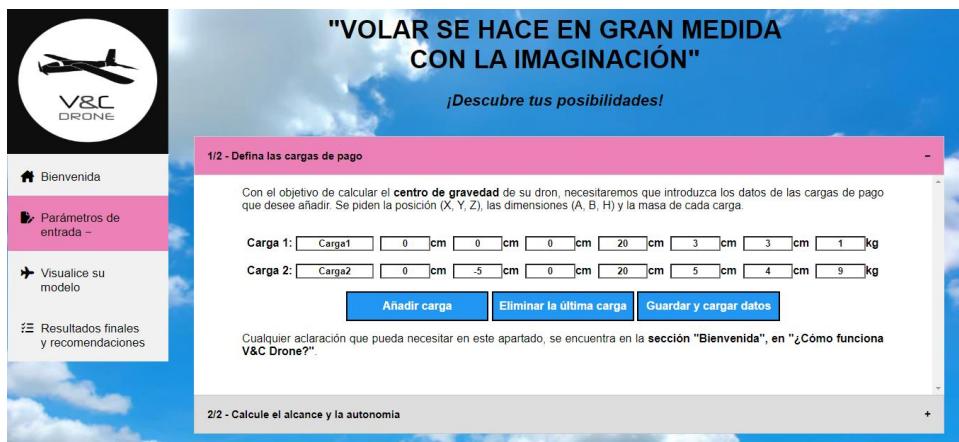


Figura 5.3: Menú de entrada de datos de las cargas de pago del usuario.

Una vez introducidos estos datos, el usuario, si así lo desea, podría pasar a introducir en el segundo acordeón los datos pedidos para calcular la autonomía y el alcance del dron, tal y como puede verse en la siguiente *figura 5.4*:

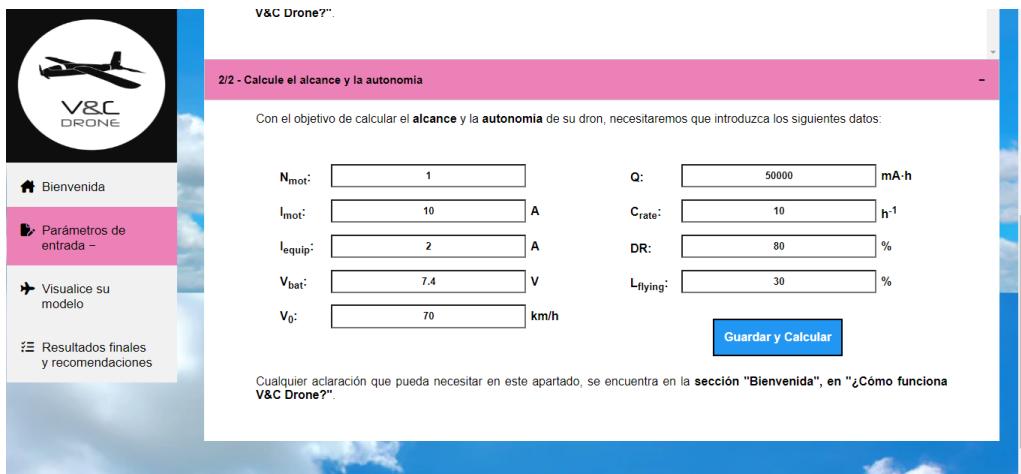


Figura 5.4: Menú de entrada del dron y de las baterías del usuario.

Una vez introducidos todos los datos se procede a pulsar el botón “Guardar y Calcular” de, por ejemplo, el “Cálculo de la autonomía y el alcance”, tras lo cual el usuario podría ver los resultados de la autonomía y el alcance en la sección “Resultados finales y recomendaciones”, como se muestra en la *figura 5.5*:



Figura 5.5: Menú que muestra los resultados del usuario, en función de los datos introducidos en la *figura 5.4*.

A continuación, nuestro usuario ficticio también debería pulsar el botón “Guardar y cargar datos” del acordeón “Defina las cargas de pago”, obteniendo en este caso dos tipos de resultados. Por un lado, obtendría los valores numéricos del CDG del dron y el CDG total, al igual que las sugerencias y comentarios, los cuales se mostrarían también en la sección “Resultados finales y recomendaciones”, de forma similar a como se mostraban los resultados de alcance y autonomía en la *figura 5.5*. Sin embargo, en este caso también se obtendría el modelo 3D de estos resultados, el cual podría verse en la sección “Visualice su modelo”. Por ejemplo, para una sola carga de lados 20x3x3cm, y con la esquina de referencia colocada en el punto definido como (0,0,0), si se le da una masa de 1kg, se

obtendría el siguiente resultado, mostrado en las *figuras 5.6 y 5.7*:



Figura 5.6: Modelo 3D del dron con una carga en su interior, donde se observa el CDG total (en verde) retrasado respecto al CDG del dron (en rojo) a causa de la carga.

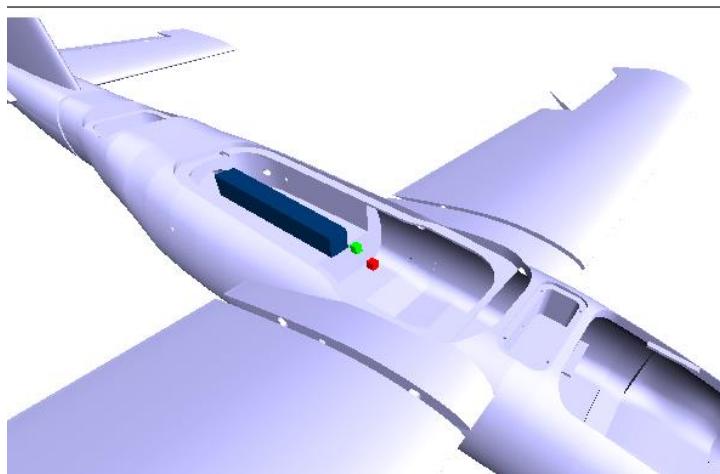


Figura 5.7: Detalle de la *figura 5.6*.

En las dos figuras anteriores puede observarse, tras emplear los controles de GUI para facilitar la visibilidad de la escena: el CDG del dron sin cargar en color rojo, el CDG total (del dron más la carga de pago) en color verde y la carga de pago mencionada en color azul.

Se observa que el CDG total se va hacia la cola (por detrás del CDG del dron) debido al peso de la carga incluida en el dron.

Aquí, finalmente, terminaría el recorrido del usuario, habiendo obtenido todos los resultados buscados de forma bastante satisfactoria, visual y sencilla.

8.3. Conclusiones Generales de la Validación

Algunas de las conclusiones generales más importantes que se pueden comentar en base a la validación del modelo y de la página web, son las siguientes:

- El modelo de Catia V5 del dron, a pesar de la falta de precisión en la toma de muchas medidas, es un modelo aceptable que tiene muchas similitudes con el dron real.
- La utilización de sketches inmersivos para hacer diseños de piezas que busquen imitar de alguna manera un diseño real ya existente es una muy buena garantía de que el diseño final será lo más parecido posible al real.
- El diseño HMI de la página web la convierte en una página bastante sencilla de utilizar y de aplicar, es decir, muy usable.
- Aunque se esperaba poder hacer la entrada de datos mucho más interactiva desplazando bloques sobre un tablero con three.js también, la opción elegida más tarde ha resultado ser lo suficientemente satisfactoria para cumplir con los objetivos marcados.
- El tiempo de espera de alrededor de 45 segundos en el cálculo de los CDG, las sugerencias y generación de STLs puede ser un tiempo muy largo para muchos de los usuarios. Este detalle será tratado más detenidamente en el futuro TFG de Virginia.
- Algunas veces la página web se refresca a causa del navegador web utilizado cuando el tiempo de espera para el “fetch” es demasiado largo. De momento, no se ha conseguido evitar este refresco de la página mediante el uso de JavaScript.
- La comunicación entablada entre JavaScript y Python ha resultado ser un completo éxito que ha permitido delegar en el lado del Backend las tareas más complejas de realizar.
- La integración de todas las tecnologías a tener en cuenta durante la realización de este novedoso proyecto ha sido un éxito desde el punto de vista funcional.
- Finalmente, cabe decir que los objetivos más importantes planteados al inicio del trabajo se han conseguido cumplir. Respecto al Frontend, se ha conseguido crear un sistema entrada, envío y recepción de datos capaz de cumplir con los objetivos deseados.
- Se ha conseguido alcanzar un uso bastante satisfactorio en el uso de three.js, pudiendo servir como base para realizar proyectos más avanzados. Los objetivos específicos solicitados por el cliente en cuanto a esta área del trabajo se han cumplido, pudiendo hacer zoom, rotar, señalar y mostrar/esconder piezas.
- Por último, como apreciación personal quiero destacar lo útil de este proyecto de cara a mi futuro profesional, habiendo aprendido sobre los lenguajes de

programación más demandados a día de hoy, entre ellos JavaScript.

Capítulo 6

Conclusiones y Futuras Líneas de Investigación

Finalmente, ya hemos llegado al capítulo final de este trabajo. A continuación, veremos cuáles son las preguntas a las que pretendemos proporcionar respuesta en este capítulo.

En un primer apartado, responderemos la siguiente pregunta: “*¿Se han conseguido alcanzar los objetivos propuestos inicialmente?*”. Contestando esta cuestión nos daremos cuenta del éxito o fracaso de la presente investigación, pudiendo además hacer una pequeña recapitulación de los puntos más importantes de este trabajo. Además, podremos analizar el “grado de éxito” alcanzado, enlazando esto directamente con el siguiente apartado de este capítulo.

En el segundo apartado, en función del grado de éxito alcanzado en la presente investigación, explicaremos cuáles pueden ser algunos de los caminos más interesantes a recorrer por futuras investigaciones relacionadas con la aplicación web desarrollada durante este proyecto.

6.1. **¿Se han Conseguido Alcanzar los Objetivos Propuestos Inicialmente?**

Para comenzar con este apartado, primero debemos recordar cuales eran los objetivos perseguidos por este TFG, definidos en el “*Capítulo 1. Introducción*”. Estos eran cuatro:

1. Diseñar un modelo de dron en Catia basado en un dron real, más concretamente el “Fighter VTOL Fixed-wing Aircraft”.
2. Desarrollo de una página web de alta usabilidad que permitiera a los usuarios visualizar el modelo obtenido en Catia desde todos sus ángulos. Se busca también poder: ocultar, mostrar y señalar diferentes partes de este.

3. Lograr el cálculo automatizado de la posición del CDG del dron cargado, en función de la distribución de carga introducida a través de la interfaz web diseñada. También se desea mostrar los resultados obtenidos al usuario.
4. Entender la forma de aplicar e integrar los frameworks y tecnologías ya existentes con el fin de alcanzar los objetivos propuestos. Además, se busca construir nuevas tecnologías capaces de realizar tareas relacionadas poco desarrolladas con anterioridad.

Se recuerda al lector que este ha sido un proyecto realizado en completa colaboración con Virginia, de forma que el resto de los objetivos comentados en la Introducción podrán encontrarse en el futuro próximo en su TFG. A continuación, estudiaremos el “grado de éxito” alcanzado en cada uno de estos cuatro objetivos.

6.1.1. Diseño del Modelo 3D del Dron (Objetivo 1)

A continuación, se comparan dos imágenes (*figura 6.1*) en perspectiva del dron real y del modelo 3D en Catia V5 obtenido.



Figura 6.1: Imágenes del “Fighter VTOL Fixed-wing Aircraft” real y del modelo 3D en Catia V5 basado en dicho dron.

Salvo la diferencia en los colores de las piezas, se observa que todas las partes del modelo 3D son muy similares a las del dron real. La forma del ala, el fuselaje, cola y medidas características son prácticamente iguales, habiendo alcanzado un diseño global de la aeronave bastante satisfactorio. Si bien es cierto, puede observarse que las hélices, las barras que las sujetan y su mecanismo de plegado, nunca se llegaron a diseñar en Catia V5 por no ser un aspecto crucial de cara a la consecución del resto de objetivos del proyecto. Sin embargo, esta decisión llevaría a tener un menor OEW en el modelo diseñado, en comparación al real. Para entender cómo se solucionó este problema adicional de cara al cálculo del CDG del dron, en el futuro próximo, usted deberá consultar el TFG de Virginia.

Más concretamente, en este TFG se ha mostrado el proceso de creación del fuselaje, sus tapas y su patín de aterrizaje. Ahora, se muestran imágenes comparativas de estas partes

(figura 6.2 y 6.3), entre el modelo 3D obtenido y el “Fighter VTOL Fixed-wing Aircraft” real.

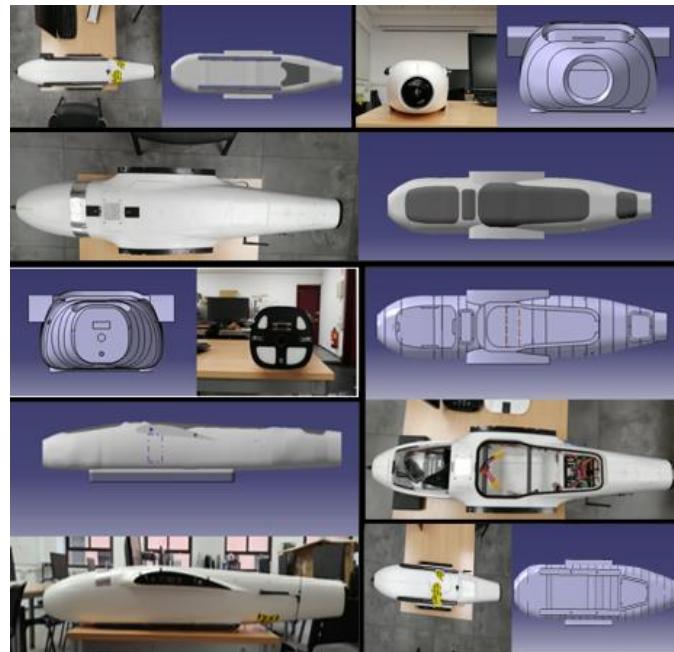


Figura 6.2: Collage comparativo de las vistas de la aeronave “Fighter VTOL Fixed-wing Aircraft” y de su modelo realizado en Catia V5.



Figura 6.3: Collage comparativo de las vistas de las tapas del fuselaje reales y las diseñadas en Catia V5.

Evidentemente, puede verse que estas partes del modelo creado y del dron base son

bastante similares, siendo todas las medidas características prácticamente las mismas.

Sí es cierto que muchos pequeños detalles de muchas de las piezas se han aproximado o cambiado de forma que la funcionalidad siguiera siendo la misma o, por lo menos, muy similar. Estas aproximaciones se han permitido y se consideran como correctas debido a que los objetivos generales de este trabajo lo permiten, no siendo de crucial importancia llegar a hacer un diseño perfecto del VTOL empleado como base.

Además, gracias a lo comentado en el capítulo de “*Testeo y Validación*”, queda evidenciada la buena aproximación de estas piezas hechas en Catia a las piezas reales, habiéndose creado un modelo bastante satisfactorio.

6.1.2. Desarrollo de la Página Web (Objetivo 2)

Como ya se ha visto a lo largo de este trabajo, una vez finalizado el modelo 3D deseado en Catia, se buscó visualizar el dron desde la aplicación web desarrollada. Para ello se empleó *three.js*, buscando además añadir efectos de postprocesado.

Tras crear la escena, situar la cámara, poner las luces, cargar cada pieza como un STL independiente y añadir color a cada una de ellas, se obtuvo el siguiente resultado mostrado en la *figura 6.4*:



Figura 6.4: Imagen del dron en “*three.js*” en la sección de resultados de la página web.

Aquí ya se añadieron los efectos llamados “*OrbitControls*”, tal que, sin más que utilizar la rueda del ratón y el clic izquierdo del mismo, el usuario puede hacer zoom y rotar el dron, ofreciendo una buena visualización del mismo.

A continuación, se añadió el postprocesado. Lo primero que se buscó hacer fue añadir el efecto de resaltado (“*OutlinePass*”). Se obtuvo el siguiente resultado (*figura 6.5*):

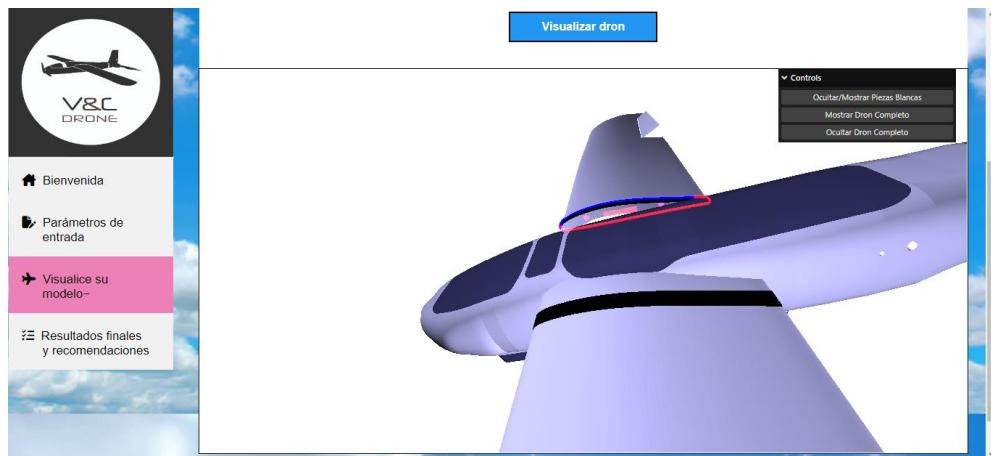


Figura 6.5: Demostración del efecto de postprocesado “OutlinePass” en “three.js”.

Tras esto, el siguiente paso fue lograr incluir el efecto “ocultar/mostrar”, tal que con un simple clic se pudiera esconder las partes seleccionadas de la aeronave. Esto se consiguió utilizando una GUI, tal y como se muestra en las siguientes imágenes de la figura 6.6:

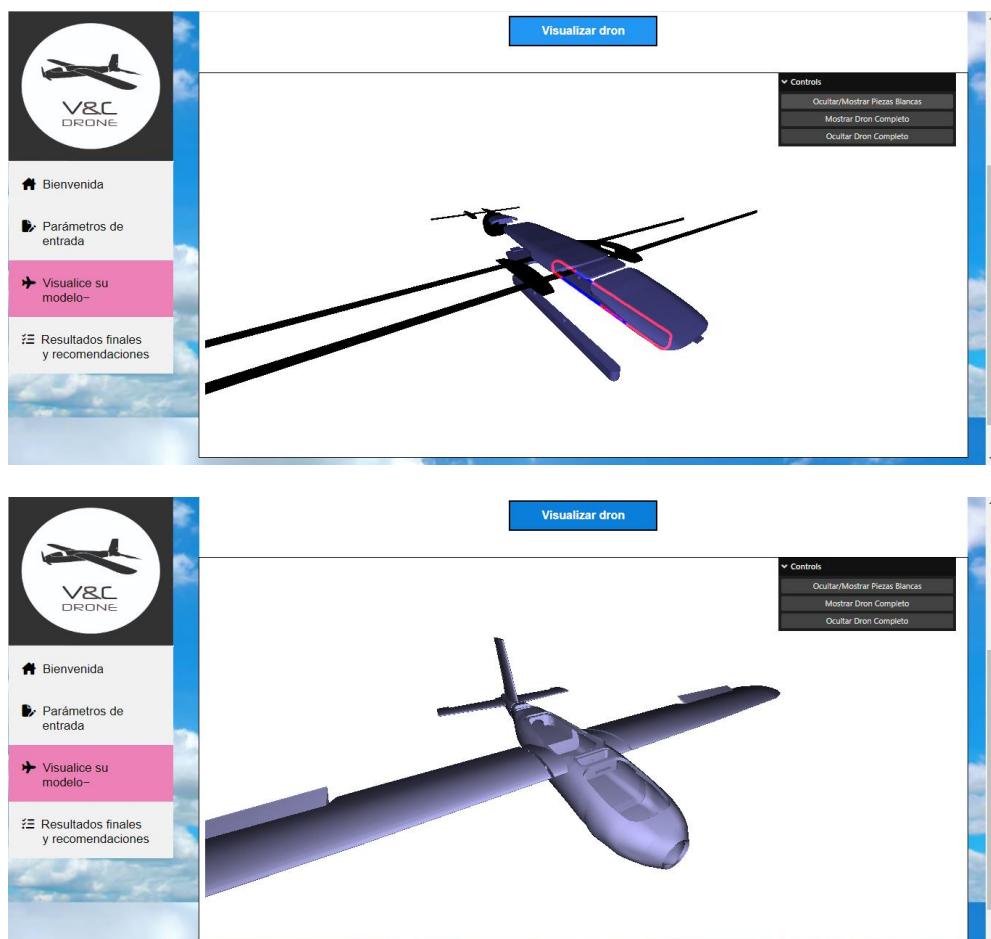


Figura 6.6: Demostración del efecto de postprocesado “Layers” en “three.js” a través del uso de la GUI.

Puede comprobarse que el resultado obtenido es bastante satisfactorio pues se ha tenido éxito en todo lo relativo a la presentación del dron en la página web, cumpliendo con todas las expectativas del cliente.

A pesar del buen resultado logrado, es conveniente comentar lo inesperado del comportamiento de three.js al incluir más de una animación a una misma pieza de la escena. Three.js solo atribuye a cada pieza la última animación que se le haya asignado en el código de JavaScript, por lo que no se ha podido asignar a dos piezas al mismo tiempo la animación "OutlinePass" y la animación "Layers". Este asunto es comentado por el propio three.js, quien en sus tutoriales online habla sobre este comportamiento del postprocesado.

6.1.3. Inputs/Outputs para Calcular el Centro de Gravedad (Objetivo 3)

Respecto a este objetivo cabe decir que, si bien se ha hecho una aplicación funcional que permite alcanzar los resultados buscados y deseados, no se la ha logrado dotar de la usabilidad que el cliente esperaba tener.

Se pretendía lograr desarrollar una aplicación similar a lo comentado en la primera línea de investigación del próximo apartado "Futuras Líneas de Investigación y de Desarrollo", el cual se recomienda leer. Por tanto, los resultados obtenidos en cuanto a este objetivo no han sido del todo satisfactorios en cuanto a la usabilidad, pero sí lo han sido en cuanto a la funcionalidad.

Se ha logrado enviar los datos de entrada necesarios para realizar el cálculo automatizado de la posición del CDG, pero no se ha logrado cumplir esta tarea de la forma ansiada (ver siguiente apartado). La forma de introducir los datos de entrada definitiva ha sido esta (ver figura 6.7):

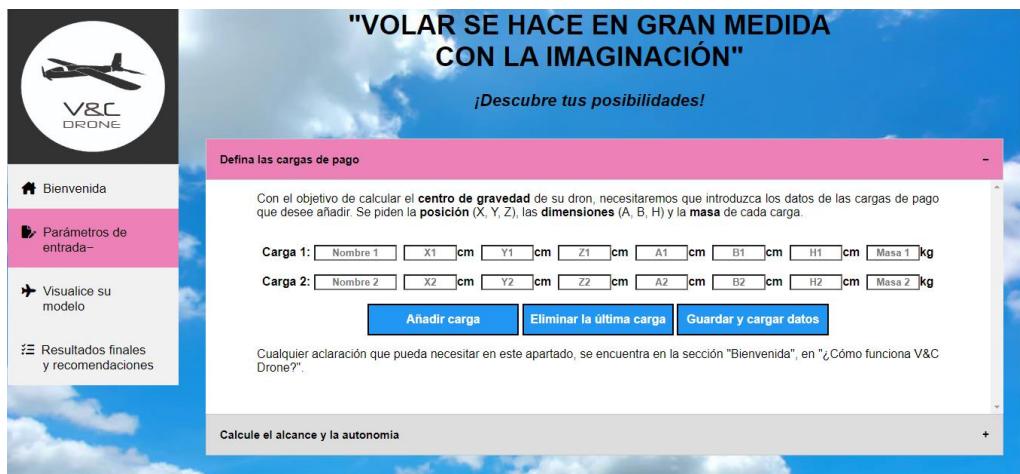


Figura 6.7: HMI para la introducción de los datos de las cargas de pago.

Se quería haber eliminado de este HMI la posición (X, Y, Z) de cada carga para mandar estos tres datos de una manera más gráfica.

Por último, en cuanto a este objetivo, la parte del “envío de datos” al servidor y “vuelta de datos” desde el servidor sí se ha logrado de manera bastante satisfactoria. Se ha logrado dar al usuario la posibilidad de introducir todas las cargas de pago deseadas (sin límite) mediante el uso de jQuery.

Finalmente, se ha logrado recibir correctamente los datos de salida de la función en Python alojada en el servidor, tal que se muestran al usuario los resultados numéricos obtenidos, así como, lo más interesante de todo, el modelo 3D que puede observarse en las *figuras 5.6* y *5.7*.

6.1.4. Aprendizajes (Objetivo 4)

Por último, solo queda comentar la efectividad de jQuery y de three.js para conseguir completar con un buen éxito funcional esta aplicación.

Sobre jQuery destaca su alta capacidad para añadir dinamismo y funcionalidad a la página web de una forma bastante sencilla. Al mismo tiempo, destaca la facilidad de uso de three.js, una vez comprendido su manejo, para crear figuras y animaciones en nuestra aplicación web.

Finalmente, cabe decir que se ha conseguido lograr de forma bastante satisfactoria la completa integración de los archivos .js, .css, .html, .py con el fin de alcanzar la funcionalidad solicitada. En definitiva, se podría afirmar que este proyecto puede ser, también, un buen demostrador tecnológico, como base para futuras investigaciones.

6.2. Futuras Líneas de Investigación y de Desarrollo

Teniendo en cuenta el apartado anterior, ahora nos planteamos la siguiente pregunta: “¿Cuáles pueden ser las futuras líneas de investigación más interesantes?”. Existen bastantes posibilidades para continuar con el desarrollo de esta aplicación web, las cuales comentaremos en los próximos párrafos.

Seguidamente, se proporciona un listado de las ideas más interesantes pendientes de desarrollar y/o mejorar a raíz de este proyecto. Estos caminos de investigación se han incluido en esta lista con un único objetivo en mente: “**aumentar lo máximo posible la USABILIDAD e INTERACTIVIDAD de la aplicación web**”. De esta forma, las ideas de desarrollo propuestas son las siguientes:

1. Siguiendo este enlace https://threejs.org/examples/#webgl_interactive_voxelpainter puede tomarse esta animación como punto de partida para desarrollar la siguiente idea mostrada en la *figura 6.10*.

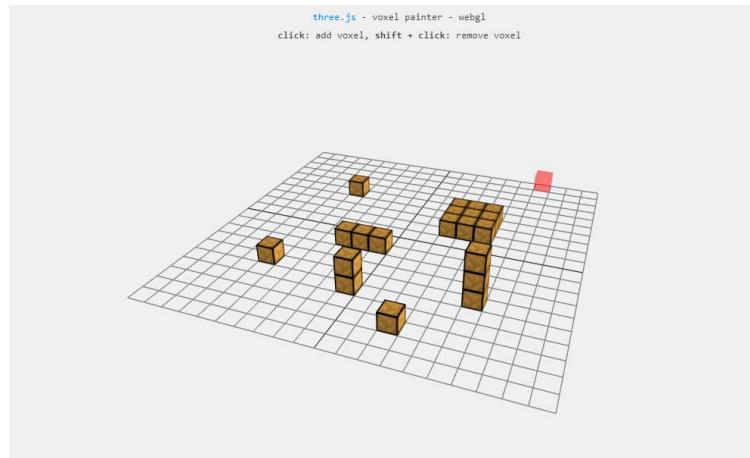


Figura 6.10: Animación de three.js llamada webgl_interactive_voxelpainter.

Como puede deducirse de la imagen anterior, esta animación de three.js permite al usuario añadir cajas cúbicas al hacer clic sobre el tablero, pudiendo además apilar estas cajitas según el cliente desee. Con ello, se plantea desarrollar una pestaña interactiva similar para la página web de este proyecto, tal que esta modificación cumpla las siguientes especificaciones:

- Deberán incluirse unos <input> que permitan a los usuarios introducir las dimensiones (largo y ancho) del tablero de la figura. Además, se pedirá la altura del espacio donde las cargas deben ir alojadas. De esta forma, quedará definido el “Bounding Box” del espacio de carga. Además, sería recomendable eliminar la cuadrícula incluida en el tablero.
- De los <input> incluidos en la sección “Parámetros de entrada → Defina las cargas de pago” solo se mantendrán aquellos destinados a introducir el nombre, dimensiones y peso de cada carga de pago, eliminándose aquellos que indican la posición (x, y, z) de las cargas. Sería de interés añadir también otro <input> que permitiera poder indicar cuántas cargas de cada tipo hay. Y, por último, sería una idea bastante buena poder permitir al usuario elegir de qué color quiere cada tipo de carga definida.
- Una vez el usuario haya introducido su distribución de cargas, deberán aparecer unos iconos en el margen de la figura del <iframe>, representando cada uno de los tipos de cargas definidas. De esta forma, el usuario podría seleccionar y arrastrar sus cargas sobre el tablero definido, colocándolas a su gusto, e incluso pudiéndolas girar.
- Por último, para lograr el objetivo aquí expuesto debería poderse tener una recarga automática y constante de los <iframe>, pudiendo además ver ambas figuras – la animación del dron con las cargas alojadas en su interior y la animación del tablero de carga - al mismo tiempo. De esta manera, podría crearse una experiencia bastante cómoda y visual para el usuario, colocando sus cargas de pago en el tablero indicado, y automáticamente pudiendo observar cómo quedan alojadas en el interior del dron.
- Este sería el esquema del HMI resultante (ver *figura 6.11*):

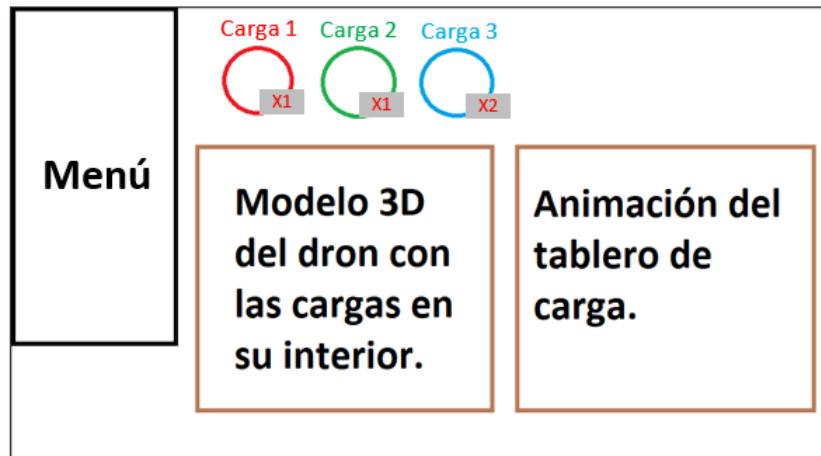


Figura 6.11: Esquema del HMI para la primera línea de investigación.

2. Respecto a los acordeones empleados en el desarrollo de este trabajo, sería interesante lograr tener inicialmente todos los acordeones de la página web abiertos nada más acceder a la misma. Hablando de los acordeones, respecto al acordeón de título “*Defina las cargas de pago*” sería una buena opción hacer que el panel se ajustase siempre al contenido, sin necesidad de hacer “scroll” pasada una cierta cantidad de píxeles de altura del panel.
3. En fases avanzadas del proyecto, una opción muy interesante sería poder solicitar al cliente que cargue su propio dron en la aplicación (ver la *figura 6.12*), cargando los CATPart y el Assembly de su dron en un zip. La aplicación cogería estos archivos y mostraría su dron en la página a través de three.js, momento a partir del cual el usuario debería definir el espacio de carga útil en el interior de su modelo de dron, para después continuar con lo indicado en el punto 1 de este apartado.

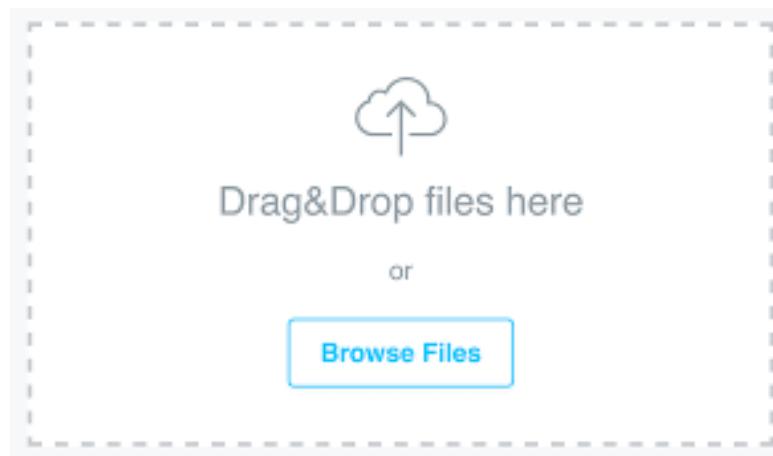


Figura 6.12: Ejemplo de la idea del tercer punto de investigación, donde los usuarios suben las piezas de sus drones.

También, se debe tener en cuenta que para cargar el dron del usuario sería necesario enviar una petición al servidor para convertir todos los CATPart en STL's, automatizando además el proceso de carga y de lectura de los STL en three.js, para

cualquier cantidad de piezas.

4. Otra idea de desarrollo es la de dar al usuario la posibilidad de indicar con el ratón dónde se encuentra la posición ideal de su CDG según el fabricante. También podría definir con el ratón el “Bounding Box” de espacio de carga útil. Todo esto, haciéndolo en la animación del dron.
5. Desarrollar una versión responsive de la interfaz web (ver la *figura 6.13*).



Figura 6.13: Imagen explicativa del término “responsive” en programación.

6. Profesionalizar más el Frontend de la página web, añadiendo efectos más atractivos y creando una verdadera experiencia sensorial desde la primera sección hasta la última de la página.
7. Investigar la manera de dar al usuario la oportunidad de asignar un color diferente a cada una de las cargas de pago definidas por él. Para ello, se podría optar por cargar un STL por cada cubo de manera que se le pudiera asignar un color a cada uno, en lugar de generar un STL para todos los cubos tal y como se ha hecho en este proyecto.
8. Asegurar la compatibilidad de todo el código programado para otros navegadores webs distintos de Google Chrome.
9. Desarrollar una función que permita superponer dos efectos diferentes en `three.js` para la misma pieza.
10. En los inputs de tipo número de la aplicación, los usuarios pueden escribir números decimales bien con (,) o con (.). Cuando JavaScript hace cálculos sí distingue entre punto (sí aceptado) y coma (no leído como número decimal) para escribir números decimales. Sin embargo, las entradas de tipo numérico permiten escribir un número decimal de ambas formas, con punto o con coma, no identificándolo como un error tampoco. El asunto está en que JSON creado llevaría la coma en cuestión en uno o más datos de entrada, dando también problemas en Python pues este lenguaje tiene el mismo criterio en cuanto a separadores decimales que JavaScript. Tras hacer varias pruebas para intentar eliminar la posibilidad de escribir una coma en el interior del input o tratar de buscar comas en los datos introducidos y en caso de haberlas, sustituirlas por puntos ninguna de estas opciones se ha logrado implementar en la

versión actual, por lo que se propone como vía de investigación añadir el código necesario al JavaScript que permita solucionar este problema.

Finalmente, querido lector, me despido de usted. Espero que haya disfrutado del proyecto realizado por Virginia y por mí. ¡Virginia y yo le mandamos mucho ánimo y mucha fuerza con sus proyectos! Un saludo y...¡gracias por el tiempo dedicado y la energía puestos para leer estas páginas que tanto me costaron escribir! ¡Gracias por haber llegado hasta el final!



¡MUCHO ÉXITO!

César Eduardo Jiménez Gámez
Alcalá de Henares, 30 de agosto de 2022

Bibliografía

7.1. Sobre el uso de “three.js” en JavaScript

- [1] **Postprocesado**, <https://threejs.org/docs/#examples/en/postprocessing/EffectComposer>
- [2] **Postprocesado**, <https://threejs.org/docs/#manual/en/introduction/How-to-use-post-processing>
- [3] **Renderizado**, <https://threejs.org/docs/#api/en/renderers/WebGLRenderer.alpha>
- [4] **Renderizado**, <https://threejs.org/docs/#api/en/renderers/WebGLRenderer>
- [5] **Cargar archivos STL**, https://threejs.org/examples/?q=load#webgl_loader_stl
- [6] **Manual de uso**, <https://threejs.org/docs/index.html#manual/en/introduction/Creating-a-scene>
- [7] **Medición del Bounding Box de un STL**, <https://products.aspose.app/3d/es/measurement/stl>
- [8] **Ejemplo de uso**, <https://programmerclick.com/article/22171715815/>
- [9] **Fondo transparente en three.js**, <https://code-examples.net/es/q/138bbc6>
- [10] **Canal de MonkeyWits sobre three.js**, <https://www.youtube.com/c/monkeywit>

7.2. Fighter VTOL Fixed-wing Aircraft

- [11] **Fighter VTOL Fixed-wing Aircraft, Make Fly Easy**, <https://n9.cl/w3bw5>
- [12] **Fighter VTOL Fixed-wing Aircraft, Make Fly Easy**, <https://n9.cl/xicy7>
- [13] **Fighter VTOL Fixed-wing Aircraft, Make Fly Easy**, <https://n9.cl/qtnl3>
- [14] **Hélices de Make Fly Easy**, <https://n9.cl/n1rkh>

7.3. JavaScript, HTML5 y CSS

- [15] **HTML5**, <https://www.youtube.com/watch?v=N6MM4nSueOg>
- [16] **CSS y JavaScript, acordeón**, <https://www.youtube.com/watch?v=rrRkPOnJaY8>
- [17] **JavaScript, form**, <https://www.youtube.com/watch?v=j3ixg2cPI54>
- [18] **JavaScript, redondeo de decimales**, <https://www.youtube.com/watch?v=gX9n3vtPuLo>
- [19] **JavaScript, return**, <https://www.youtube.com/watch?v=kti9atdox3I>
- [20] **Canal de YouTube de Kiko Palomares sobre desarrollo web**,
<https://www.youtube.com/c/KikoPalomares>
- [21] **Canal de YouTube HolaMundo**, <https://www.youtube.com/c/HolaMundoDev>
- [22] **Canal de YouTube sobre desarrollo web de “jonmircha”**,
<https://www.youtube.com/c/jonmircha>
- [23] **Font Awesome**, <https://fontawesome.com/>
- [24] **Uso de “swal”**, <https://sweetalert.js.org/guides/>
- [25] **Descargar archivo en JavaScript**, <https://code-boxx.com/create-save-files-javascript/>
- [26] **CSS, efectos de imagen**, <https://www.enriquejros.com/efectos-imagenes-css/>
- [27] **Acordeones, JavaScript**, <https://evolvingweb.ca/blog/building-accordions-html5-details-tag-no-javascript-needed>
- [28] **Delftstack, JavaScript**, <https://www.delftstack.com/es/howto/javascript/javascript-import-vs-require/>
- [29] **Fetch, JavaScript**, <https://flask-es.readthedocs.io/patterns/javascript/>
- [30] **Strings, JavaScript**, <https://www.cloudera.com/javascript-convert-string-variable-name/>
- [31] **Strings, JavaScript**, <https://www.codespeedy.com/convert-string-into-variable-name-in-javascript/>
- [32] **Fetch, JavaScript**, <https://flask.palletsprojects.com/en/2.1.x/patterns/javascript/>
- [33] **Strings, JavaScript**, <https://www.neoguias.com/como-encontrar-caracter-en-cadena-javascript/>
- [34] **Strings, JavaScript**, <https://altocodigo.blogspot.com/2019/07/javascript-convertir-numero-string-y.html>
- [35] **Manual de HTML**, <http://www.estrellateyarde.org/manual-de-html/manual-de-html-enlaces>
- [36] **Manual de HTML y CSS**, <https://www.um.es/docencia/barzana/IAGP/HTML1.html>
- [37] **Fondo de imagen, HTML**, <https://desarrolloweb.com/articulos/1384.php>
- [38] **Selectores de validez CSS**, <https://elcssar.com/css/valid-invalid>

- [39] **JavaScript, jQuery**, <https://n9.cl/5nij0>
- [40] **jQuery**, <http://www.forosdelweb.com/f179/boton-crear-nuevo-input-formulario-1124389/>
- [41] **jQuery**, <https://www.baulphp.com/agregar-mas-campos-jquery-con-botón-quitar/>
- [42] **jQuery, acordeón**, <https://n9.cl/0yewu3>
- [43] **jQuery, acordeón**, <https://n9.cl/7ayu1>
- [44] **JavaScript, mostrar y ocultar elementos**, <https://platzi.com/tutoriales/1050-programacion-basica/178-mostrar-y-ocultar-div-con-javascript-y-css3/>
- [45] **w3schools**, <https://www.w3schools.com/>
- [46] **jQuery**, <https://api.jqueryui.com>
- [47] **Foro de preguntas stackoverflow**, <https://stackoverflow.com/questions>
- [48] **Bootstrap**, <https://getbootstrap.com/docs/5.1/getting-started>
- [49] **Manuales para desarrollo web**, <https://developer.mozilla.org/es/docs/Web>
- [50] **Cálculo de autonomía y alcance**, <https://www.translatorscafe.com/unit-converter/ES/calculator/multicopter-lipo-battery/>

7.4. JSON y Servidor Local Flask

- [51] **Relación entre Python, JSON y JavaScript**, <https://www.makeuseof.com/tag/python-javascript-communicate-json/>
- [52] **Explicación de Fetch API**, <https://rahulbaran.hashnode.dev/how-to-send-json-from-javascript-to-flask-using-fetch-api>
- [53] **Explicación de Fetch API**, <https://es.javascript.info/fetch>
- [54] **Rutas en Flask**, <https://pythonbasics.org/flask-tutorial-routes/>
- [55] **Uso de await en funciones asíncronas**, <https://flask.palletsprojects.com/en/2.1.x/async-await/>
- [56] **Lectura de JSON en el servidor**, <https://n9.cl/1prm>
- [57] **Explicación detallada de lectura de JSON en el servidor**,
<https://stackoverflow.com/questions/29987323/how-do-i-send-data-from-js-to-python-with-flask>
- [58] **Ejemplo de servidor 1**, <https://www.youtube.com/watch?v=dIhg8HMZTOk>
- [59] **Explicación de JSON**, <https://www.youtube.com/watch?v=nIctNyBGQcE>
- [60] **Explicación de JSON**, <https://www.youtube.com/watch?v=ak4238Z9BIA>
- [61] **Explicación de JSON**, https://www.youtube.com/watch?v=rJesac0_Ftw

- [62] **Lectura de JSON con D3.js**, <https://www.tutorialsteacher.com/d3js/loading-data-from-file-in-d3js>

7.5. Documentos relativos al Estado del Arte

- [63] **Datos comerciales estadísticos**, <https://www.toptal.com/finance/market-research-analysts/los-drones-comerciales-estan-revolucionando-las-operaciones-comerciales>
- [64] **Ventajas y desventajas de diferentes drones**, <https://www.pasionporvolar.com/reflexiones-en-el-mundo-de-los-drones-vtol/>
- [65] **Varias configuraciones de VTOL**, <https://www.embention.com/es/producto/uav-vtol-hibridas/>
- [66] **Aplicaciones comerciales de los drones**, <https://ideingenieria.es/servicios/transformacion-digital-4-0/drones/#ventajas>
- [67] **Datos comerciales estadísticos**, <https://capital.es/2021/04/21/la-industria-espanola-del-dron-alcanzara-los-1-500-millones-en-2050/>
- [68] **Datos comerciales estadísticos**, <https://thedronesland.com/industria-de-los-drones-crecimiento/>
- [69] **Aplicaciones de los drones**, <https://n9.cl/7n3kr>
- [70] **Impacto en la economía de los drones**, <https://hablemosdeempresas.com/empresa/drones-economia-mercado-laboral/>
- [71] **Aplicaciones de los drones**, <https://exodronics.com/es/sectores-drones-ejercen-mayor-impacto/>
- [72] **Mercado de los drones**, <https://www.mordorintelligence.com/es/industry-reports/drones-market>
- [73] **TFG de referencia**, <https://n9.cl/ozv86>
- [74] **TFG de referencia**, https://ddd.uab.cat/pub/tfg/2016/tfg_49537/Articulo-centro-gravedad.pdf
- [75] **eVTOL de Dufour Aerospace**, <https://www.xataka.com/vehiculos/este-evtol-avion-electrico-capaz-despegar-atterrizar-vertical-demuestra-video>
- [76] **Empresa Dufour Aerospace**, <https://www.dufour.aero/>
- [77] **Cita del Economist**, <https://www.economist.com/technology-quarterly/2017-06-08/civilian-drones>

7.6. Tutoriales de Catia V5

- [78] **Herramienta Multi-Sections Surface**, http://catiadoc.free.fr/online/cfyug_C2/cfyugloft.htm
- [79] **Herramienta Pad**, http://catiadoc.free.fr/online/prtug_C2/prtugbt0501.htm

- [80] **Combinar dos sólidos**, <https://youtu.be/Jvr9g6fSxIQ>
- [81] **Herramienta Fill**, <https://youtu.be/vUk9j9Yk9WM>
- [82] **Diseño de fuselaje, ejemplo 1**, <https://youtu.be/l4GngAw6lGY>
- [83] **Diseño de fuselaje, ejemplo 2**, <https://youtu.be/ZYrPAoNygc>
- [84] **Diseño de fuselaje, ejemplo 3**, <https://www.youtube.com/watch?v=sFYjLp-ukrg>
- [85] **Aplicar un factor de escala**, <https://youtu.be/17OeoOZVOdA>
- [86] **Realizar sketches inmersivos**, <https://www.youtube.com/watch?v=O-Aw1HUesqU>

7.7. Fluajogramas

- [87] **Realización de fluajogramas**, <https://www.youtube.com/watch?v=lffiTrZR4E>
- [88] **Realización de fluajogramas**, <https://www.areatecnologia.com/diagramas-de-flujo.htm>
- [89] **Realización de fluajogramas**, <https://www.heflo.com/es/blog/modelado-de-procesos/significado-simbolos-diagrama-flujo/>