UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

Sistema de Navegación Semiautónomo en Interiores

Presentado por Mario Bartolomé Manovel
en Universidad de Burgos — 3 de junio de 2018

Tutores: Dr. Alejandro Merino Gómez
Dr. César Ignacio García Osorio
Dr. José Francisco Díez Pastor

Índice general

Índice de figuras

Índice de tablas

Apéndice A

Plan de Proyecto Software

A.1. Introducción

La planificación es, sin duda, una parte fundamental en el ciclo de vida de un proyecto. Con el uso de metodologías ágiles para el seguimiento del proyecto, la planificación inicial se vuelve voluble y cambiante, de forma que la mayor parte de aproximaciones serán precisamente eso, aproximaciones.

En este anexo se detallan los recursos necesarios para el desarrollo del proyecto mediante un estudio de la planificación temporal, y un estudio de la viabilidad del mismo.

El primer apartado, **planificación temporal**, establece un calendario de estimaciones. Cabe destacar que dicho calendario es totalmente flexible. Si que se ha mantenido la estructura habitual estableciendo una fecha de inicio y una fecha de finalización, y se han tenido en cuenta la duración de las tareas, pero estas son extendibles entre diferentes fases, o *sprints*.

El segundo apartado, **estudio de viabilidad**, detalla la viabilidad del proyecto a través de dos subapartados. El primero detallará la viabilidad económica del mismo, teniendo en cuenta los costes y beneficios previstos. El segundo apartado, relacionado con la viabilidad legal, detallará el contexto legal en el que se regula este proyecto.

A.2. Planificación temporal

Para la realización de este proyecto se ha seguido una metodología ágil basada en Scrum. Dicha metodología aporta flexibilidad, muy necesaria durante el desarrollo software. Si bien está metodología no se ha seguido por completo, ya que el proyecto no ha sido desarrollado por un equipo grande, si que se ha seguido en esencia la forma que la metodología propone:

- El desarrollo se ha llevado a cabo mediante iteraciones, llamadas *sprints* y revisiones de las mismas.
- La duración de los *sprints* fue de dos semanas. Se consideró hacerlos de una única semana, pero la dificultad del desarrollo de los algoritmos, y sobre todo el amplio estudio que estos requerían, hizo que nos decantásemos por una duración bisemanal.
- Al finalizar cada *sprint* se procedía a realizar una reunión, en la que se comentaba el estado del desarrollo, y se planificaba la siguiente iteración.
- Durante las reuniones de revisión/planificación, se establecían las tareas a realizar en el siguiente *sprint*.
- La monitorización del proyecto se realiza mediante gráficos *burndown*, aunque no son especialmente útiles en fases iniciales, donde el número de *issues* no es alta, y su finalización es muy variable.

Se ha realizado una valoración de la duración de cada *issue* haciendo uso de *story points*, que son una manera de aproximar la duración máxima de las mismas. La estimación temporal dada a los *story points* se ha realizado como una correspondencia a días, es decir, una tarea marcada con 2 *story points* se estima que finalizará al cabo de 2 días de trabajo.

De esta forma se flexibiliza todavía más la metodología, a riesgo de perder precisión. Al no darse una correspondencia absoluta entre los *story points* asignados y la duración, se puede relegar una tarea poco importante para finalizarla en el número de días asignados. Desde luego una tarea sencilla se podrá finalizar antes, y una compleja después, dependiendo de la dedicación, en horas, dedicada a ella.

Sprint 1. 06/11/17 - 20/11/17

Este primer *sprint* fue una toma de contacto, y realizado antes del comienzo del semestre, con la intención de tener claro el desarrollo inicial del proyecto. En él se establecieron los requisitos del mismo, y los diferentes algoritmos a implementar. La descomposición de las tareas puede ser consultada en Sprint 1

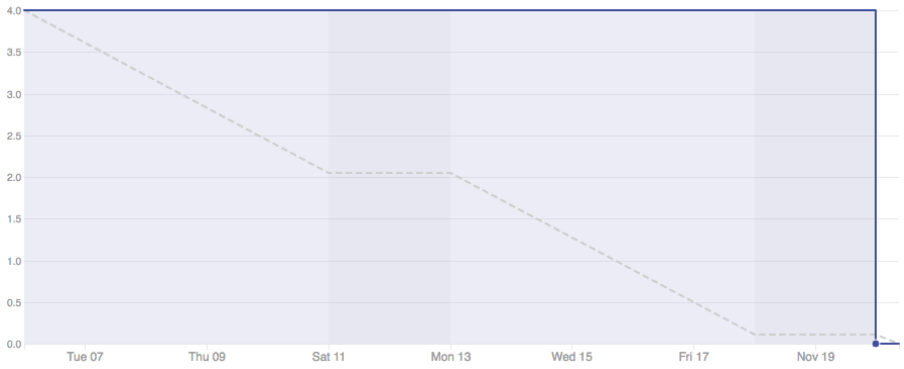

Figura A.1: Gráfico burndown.

Como puede verse en la figura A.1, se relegó el cierre de estas tareas de documentación hasta el final del *sprint*. Se estimaron 4 *story points*, pero se dedicaron al menos 8 días a la búsqueda de información. Dada la carga de trabajo al final del semestre no se pudo dedicar más tiempo al mismo.

Sprint 2. 20/11/17 - 04/12/17

Este segundo *sprint* fue dedicado a continuar con la obtención de información, y realizado antes del comienzo del semestre. El Dr. Alejandro Merino Gómez sugirió la utilización de algoritmos de Campos Potenciales para implementar el sistema de evasión de obstáculos, y proveyó de mucha información relacionada con el mismo. La descomposición de las tareas puede ser consultada en Sprint 2.

Figura A.2: Gráfico burndown.

Como puede verse en la figura A.2, este *sprint* si que siguió con mayor precisión la forma ideal del gráfico *burndown*. Se estimaron 10 *story points*, pero se dedicaron 9 días a la búsqueda de información. Dada la carga de trabajo al final del semestre no se pudo dedicar más tiempo al mismo.

Sprint 3. 4/12/17 - 15/01/18

Este tercer *sprint*, más largo por la existencia de las vacaciones de navidad, fue dedicado a continuar con la obtención de información, y realizado antes del comienzo del semestre. El Dr. Alejandro Merino Gómez sugirió la búsqueda de algún sistema operativo de tiempo real, en previsión de que el sistema operativo de la RaspberryPi no fuera lo suficientemente rápido en sus cambios de contexto. El Dr. José Francisco Díez Pastor proporcionó información sobre búsqueda de rutas. La descomposición de las tareas puede ser consultada en Sprint 3.

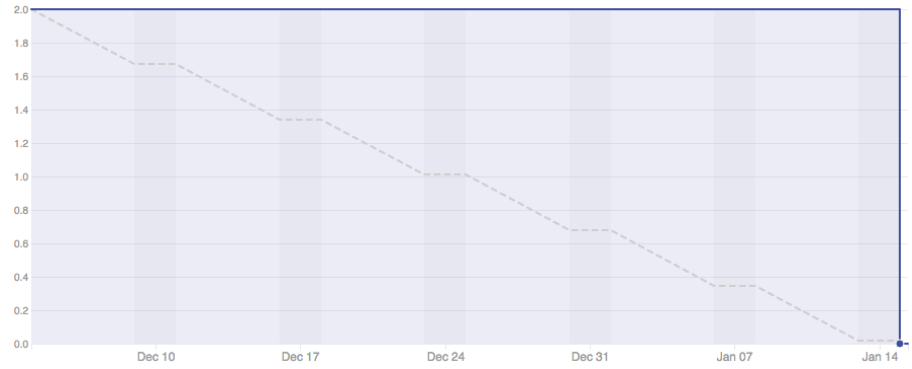

Figura A.3: Gráfico burndown.

Como puede verse en la figura A.3, este *sprint* no que siguió la forma ideal del gráfico *burndown*, dadas las pocas tareas creadas. Se estimaron 2 *story points*, pero se dedicaron al menos 10 días a la búsqueda de información. Dada la carga de trabajo al final del semestre, y el periodo vacacional, no se pudo dedicar más tiempo al mismo.

Sprint 4. 17/01/18 - 31/01/18

Este cuarto *sprint*, fue dedicado a conseguir la comunicación entre la RaspberryPi y el drone. Se creó la primera implementación del algoritmo de Campos Potenciales, posteriormente sustituido por *VFH*. En este caso, durante la preparación del *sprint*, expliqué a los tutores las diferentes formas de comunicación que se podían valorar. La descomposición de las tareas puede ser consultada en Sprint 4.

Figura A.4: Gráfico burndown.

Como puede verse en la figura A.4, este *sprint* se ajustó bastante bien a la forma ideal del gráfico *burndown* dada la cantidad de tareas. Se estimaron 9 *story points*, pero se dedicaron al menos 12 días a la consecución del protocolo implementado.

Sprint 5. 31/01/18 - 14/02/18

Este quinto *sprint*, fue dedicado a conseguir la emisión de vídeo desde la RaspberryPi. Durante la preparación de este *sprint*, el Dr. César Ignacio García Osorio sugirió realizar un *refactor* de algunas partes del código ya desarrollado para mejorar su legibilidad y mantenibilidad. La descomposición de las tareas puede ser consultada en Sprint 5.

Figura A.5: Gráfico burndown.

Como puede verse en la figura A.5, este *sprint* no se ajustó demasiado a la forma ideal del gráfico *burndown* dada la longitud de algunas de las tareas creadas. Se estimaron 14 *story points*, y se dedicaron 14 días a lograr los objetivos marcados.

Sprint 6. 14/02/18 - 08/03/18

Este sexto *sprint*, fue dedicado a realizar una implementación que sustituyese al algoritmo de Campos Potenciales. Durante la preparación de este *sprint*, se propuso crear otro algoritmo, dados los problemas que presentaba el algoritmo de Campos Potenciales. El Dr. César Ignacio García Osorio mostró preocupación por las latencias existentes en el feed de vídeo proveniente de la RaspberryPi, así que también se comprobó el código, y se buscaron formas de lograr que este fuese más fluido. La descomposición de las tareas puede ser consultada en Sprint 6.

Figura A.6: Gráfico burndown.

Como puede verse en la figura A.6, este *sprint* no se ajustó a la forma ideal del gráfico *burndown* dada la existencia de una única tarea, y su longitud. Se estimaron 8 *story points*, pero realmente se dedicaron 14 días a lograr los objetivos marcados.

Sprint 7-8. 08/03/18 - 12/04/18

Este septimo-octavo *sprint* se creó con longitud doble, debido a la necesidad de generar una primera release y la cantidad de trabajo requerido para ella. Fue dedicado a finalizar la implementación del VFH, que sustituyese al algoritmo de Campos Potenciales. Se creó una página web que permitía a un usuario iniciar sesión y visualizar el vídeo proveniente del drone asignado, así como su control remoto. Durante este *sprint* se implementó la parte central de la arquitectura de este proyecto.

La descomposición de las tareas puede ser consultada en Sprint 7-8.

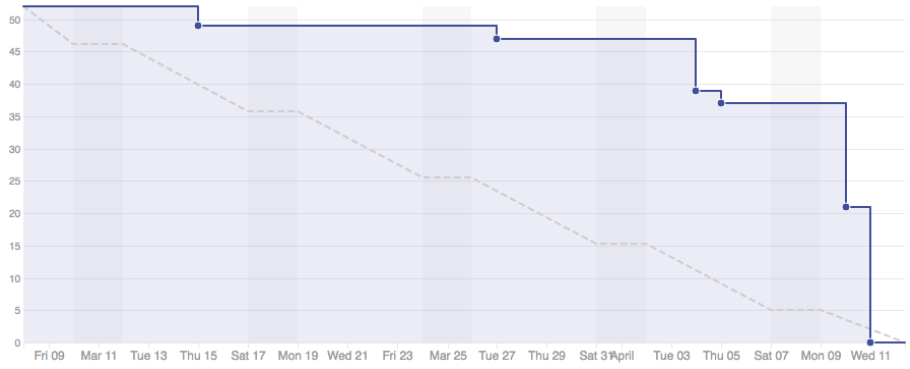

Figura A.7: Gráfico burndown.

Como puede verse en la figura A.7, este *sprint* no se ajustó a la forma ideal del gráfico *burndown* dada la longitud de las tareas finales, que provocan que el escalado del gráfico no sea el apropiado. Se estimaron 52 *story points*, pero realmente se dedicaron 14 días a lograr los objetivos marcados. Obviamente, existió solapamiento de tareas. La release se finalizó correctamente y está disponible para su uso como versión *alpha*.

Sprint 9. 12/04/18 - 26/04/18

Este noveno *sprint* fue dedicado a comenzar con la implementación del Filtro de Partículas, y crear los scripts de instalación de los que carecía la primera release.

La descomposición de las tareas puede ser consultada en Sprint 9.

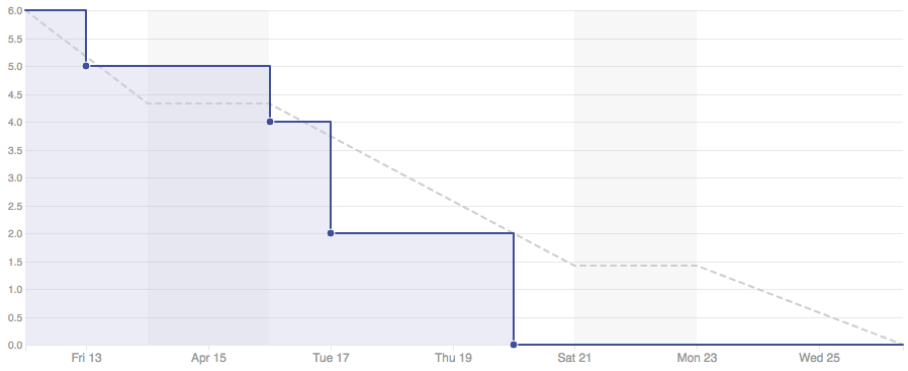

Figura A.8: Gráfico burndown.

Como puede verse en la figura A.7, este *sprint* se ajustó bastante bien a la forma ideal del gráfico *burndown*. Sin embargo al final del mismo, se aprecia la desaparición de tareas. Esto es debido a pasar la implementación del Filtro de Partículas al siguiente *sprint*. No existe una forma de *extender* a duración de una tarea entre *sprints* de forma que se presente en ambos. Se estimaron 6 *story points*, pero realmente se dedicaron 14 días a lograr los objetivos marcados, además del comenzar con el desarrollo del Filtro de Partículas. En un principio se estimó la duración del desarrollo del Filtro de Partículas en 8 *story points*.

Sprint 10. 26/04/18 - 11/05/18

Este décimo *sprint* fue dedicado a la implementación del Filtro de Partículas.

Tal y como se ha explicado con anterioridad, no existe una forma de establecer la misma *issue* en varios *sprints*, de manera que el gráfico *burndown* de este periodo de tiempo se encuentra vacío. El Filtro de Partículas no se finalizó durante el *sprint*, y por lo tanto fue extendido al siguiente.

La duración del desarrollo del Filtro de Partículas se extendió a 21 *story points*.

Sprint 11. 11/05/18 - 25/05/18

Este undécimo *sprint* fue dedicado a continuar con la implementación del Filtro de Partículas, y establecer el hardware restante en el drone. Además, se implementaron los controladores PID necesarios para la segunda release del proyecto.

La descomposición de las tareas puede ser consultada en Sprint 11.

Figura A.9: Gráfico burndown.

Como puede verse en la figura A.9, este *sprint* se ajustó muy bien a la forma ideal del gráfico *burndown*, dada la gran cantidad de tareas creadas. Sin embargo al final del mismo, se aprecia la desaparición de tareas. Esto es debido a pasar la implementación del Filtro de Partículas al siguiente *sprint*, debido a la necesidad de probarlo correctamente. No existe una forma de *extender* a duración de una tarea entre *sprints* de forma que se presente en ambos. Se estimaron 27 *story points*, pero realmente se dedicaron 14 días a lograr los objetivos marcados, además del continuar con el desarrollo del Filtro de Partículas.

Sprint 12. 25/05/18 - 8/06/18

Este duodécimo *sprint* está siendo dedicado a finalizar la documentación del proyecto, y a las pruebas en un entorno real.

La descomposición de las tareas puede ser consultada en Sprint 12, aunque no se encuentra cerrado ni finalizado.

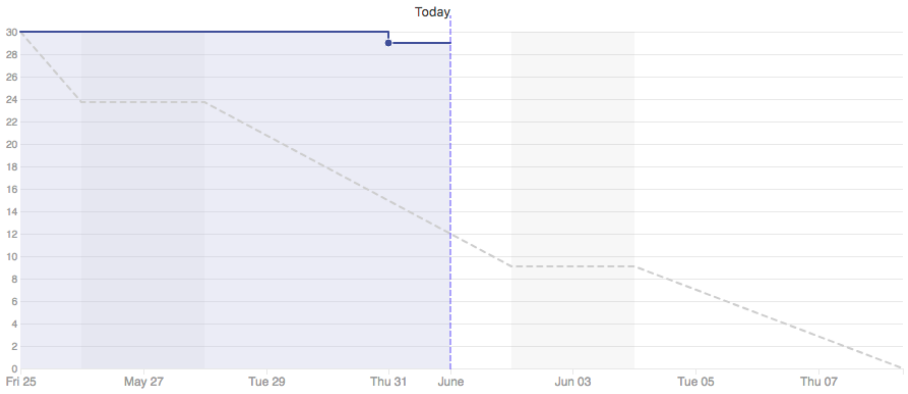

Figura A.10: Gráfico burndown.

Por el momento se han estimado 30 *story points*, entre los que se cuentan los 21 del Filtro de Partículas, y 8 de ellos se corresponden con la adquisición de métricas de SonarCloud.

A.3. Estudio de viabilidad

Viabilidad económica

En este apartado se detallan los costes y beneficios que podría suponer el proyecto, de desarrollarse con una intención comercial.

Costes

De personal: Para el desarrollo del proyecto tan solo se ha empleado un desarrollador, empleado a tiempo completo durante cuatro meses, de febrero a mayo.

Concepto	Coste
Salario mensual neto	1102€
Retención IRPF (13.55 %)	271€
Seguridad Social (29.9 %)	598€
Salario mensual bruto	2000€
Total 4 meses	8000€

Tabla A.1: Costes de personal de desarrollo

La retribución a la Seguridad Social se ha calculado como un 23,60 % por contingencias comunes, más un 5,50 % por desempleo de tipo general, más un 0,20 % para el Fondo de Garantía Salarial y má un 0,60 % de formación profesional. En total un 29,9 % que se aplica al salario bruto. Ver [1].

Se estima también la ayuda prestada por los tres tutores, dividiendo un crédito a partes iguales entre ellos. Para lo cual se utilizará la tabla disponible en [2], y haciendo el cálculo de la siguiente manera:

- Sueldo base Ayudante Doctor: 1815,61€. 14 pagas anuales. Total 25418,54€al año. El cual imparte 24 créditos, de forma que tiene un coste por crédito de 1059,1€al año. Quedando así en 353,04€por los 4 meses.

- Sueldo base Contratado Doctor Permanente: 2325,24€. 14 pagas anuales. Total 32553,36€al año. El cual imparte 32 créditos, de forma que tiene un coste por crédito de 1017,3€al año. Quedando así en 339,1€por los 4 meses.

- Sueldo base Contratado Doctor Básico: 2001,27€. 14 pagas anuales. Total 28017,8€al año. El cual imparte 24 créditos, de forma que tiene un coste por crédito de 1167,41€al año. Quedando así en 389,14€por los 4 meses.

De esta forma, y tal y como se ha establecido la división a partes iguales del crédito correspondiente:

Concepto	Coste
Ayudante Doctor	$353{,}04 * 0{,}3\overline{3} = 117{,}68$€
Contratado Doctor Permanente	$339{,}1 * 0{,}3\overline{3} = 113{,}04$€
Contratado Doctor Básico	$389{,}14 * 0{,}3\overline{3} = 129{,}72$€
Total 4 meses	360,44€

Tabla A.2: Costes totales de asesoría

De esta manera, los costes totales de personal ascienden a:

Concepto	Coste
Desarrolladores	8000€
Asesoría Doctores	360,44€
Total 4 meses	8360,44€

Tabla A.3: Costes totales de personal

De hardware: Para el desarrollo del proyecto se han empleado numerosos elementos de hardware que se detallan a continuación:

Concepto	Coste	coste amortizado
Ordenador portátil	800€	66,67€
Emisora	177,79€	11,86€
Drone	Desglose	
RaspberryPi 3B	33,74€	2,25€
Tarjeta de memoria	21,99€	1,47€
Carcasa RaspberryPi	10,99€	0,74€
Cámara PiNoIR	28,99€	1,94€
LED IR	9,99€	0,67€
Controladora de vuelo	38,99€	2,6€
Motores EMAX mt2216	47,84€	3,19e
ESC 20A EMAX	30,53€	2,04€
Hélices de repuesto	10,34€	0,69€
Cargador Baterías	18,63€	1,25€
Sensores Ultrasonidos HCSR04	8,79€	0,59€
Chasis drone con PCB	19,99€	1,34€
Tren de aterrizaje	5,89€	0,4€
Cables	7,09€	0,48€
Tornillos	17,15	1,15€
Hilo de estaño	9,9€	0,67€
Soldador JBC 11W	41,95€	2,8€
Total	1340,58€	89,38€

Tabla A.4: Costes de hardware

De software: Para el desarrollo de este proyecto, se valoran los costes del siguiente software, cuya amortización se ha establecido en 2 años.

Concepto	Coste	coste amortizado
PyCharm IDE	79,6€	13,27€
WebStorm IDE	51,6€	8,6€
GitHub Privado	23,43€	3,91€
SonarCloud Privado	40€	6,67€
Total	194,63€	32,45€

Tabla A.5: Costes de software

Varios: Los siguientes, son costes variados:

Concepto	Coste
Alquiler de oficina	600€
Internet	94,4€
Alquiler nave de pruebas	600€
Total	1294,4€

Tabla A.6: Costes variados

Totales: El total de los costes es el siguiente:

Concepto	Coste
Personal	8360,44 €
Hardware	1340,58 €
Software	194,63 €
Varios	1294,4 €
Total	11190,05€

Tabla A.7: Costes totales del proyecto

Beneficios

Por el momento, se trata de un proyecto con fines académicos, y con unos gastos relativamente elevados para la etapa tan temprana en que se encuentra.

Sin embargo, los beneficios podrían ser muy elevados. Se va a considerar un sistema de suscripción anual al servicio de vigilancia mediante drones. Dicho servicio proporciona vigilancia mediante drones, los cuales estarán supervisados por personal humano, y su mantenimiento. El servicio se detalla en la siguiente tabla:

Entorno	Dimensiones	Drones	Coste
Nave	$> 2000m^2$	1	8000 €/año
Nave grande	$> 4000m^2$	2	15000€/año
Grupo de naves	$> 15000m^2$	4	28000€/año

Tabla A.8: Monetización del proyecto.

La finalidad de establecer las dimensiones y el número de drones, se corresponde con la necesidad de realizar una vigilancia adecuada de un recinto de dimensiones determinadas. Es decir, no es lo mismo cubrir una superficie de $1000m^2$ que una de $2000m^2$, por supuesto teniendo en cuenta la distribución interior de las mismas. El último elemento de la tabla A.8, se corresponde con un grupo de naves ubicadas en el mismo recinto, de forma que los drones tendrían acceso a cada una de las naves, y a un canal de tránsito entre ellas.

Los costes serían rentables para las empresas, dado que contratar personal de seguridad acorde a la dimensionalidad del negocio es más costoso que mantener los drones establecidos en la tabla A.8.

Punto de equilibrio El punto de equilibrio establece el momento en el que se alcanza a cubrir la inversión inicial. Los cálculos propuestos a continuación son una estimación aproximada:

Asumiendo el coste de personal de estos 4 meses en los 8360,44€resultantes de la tabla A.3, podemos suponer uno coste anual de 25081,32€por empleado[1]. Se estima que para llevar a cabo este proyecto de forma correcta, y ágil, se deberían emplear a tiempo completo, al menos ,3 desarrolladores y 2 ingenieros en electrónica. Suponiendo unos gastos anuales de personal de desarrollo de 125406,6€.

Se establece que un vigilante/piloto es capaz de atender hasta 8 drones. De forma que se añade otro empleado por cada 8 drones construidos.

[1]No se ha eliminado el coste de los tutores.

El coste de elaborar un drone con las características detalladas en la tabla A.4 asciende, sin contar con la necesidad de una emisora por drone, y un ordenador portátil por drone, a 362,79€. Sin embargo, teniendo en cuenta que la tecnología empleada no es tan precisa como se desearía, se añade un margen que permitiría mejorar el hardware empleado, estableciendo el total de elaboración de un drone en 550€.

Los costes de mantenimiento y reparación de un drone, basados en la experiencia personal con estos dispositivos, y en la tecnología empleada, se establece en 150€anuales por dispositivo. El coste del seguro de responsabilidad civil de la empresa se estima en 265€por dispositivo, con una cobertura mínima de 300000€, tal y como se estipula en [3], y teniendo en cuenta que este seguro cubriría daños al dispositivo (lo cual permite acotar mejor las estimaciones dadas). Para más información ver tablas de costes.

Además, se incluye un gasto de daño social generado, ver A.3, valorado en 30000€al año, en concepto de formación para los afectados.

De esta forma, se establece el siguiente resumen de costes, asumiendo que las suscripciones anuales generadas son siempre la suscripción más simple de que se dispone, y que establece un ingreso de 8000€anuales:

Concepto	Coste anual
Desarrolladores	125406,6€
Piloto/Vigilante	25081,32€* drone/8
Hardware	550€* drone
Mantenimiento	150€* drone
Seguros	265€* drone
Formación	30000€
Total anual	$(125406{,}6 + 965 * drone + 30000 + 25081{,}52 * drone/8)$€

Tabla A.9: Gastos totales

Y por lo tanto el cálculo sería, donde x = drone por simplificar:

$$8000x - (125406{,}6 + 965x + 30000 + \frac{25081{,}52x}{8}) \approx 40 \text{ drones/año} \quad \text{(A.1)}$$

para cubrir los costes generados.

Responsabilidad Social

Con el auge de la mecanización y automatización de los puestos de trabajo, es obvio que los seres humanos iremos perdiendo cabida en empleos que un sistema informatizado podría realizar con mayor eficiencia, y sobre todo de forma más económica.

Este proyecto, tiene un gran potencial de causar un daño social importante. El puesto de vigilante nocturno ha provisto de seguridad en muchos entornos laborales, y ha contribuido a generar riqueza protegiendo los bienes de las diferentes empresas que hacen uso de este personal.

Por tanto, si este proyecto avanza y llega a ser utilizado en sustitución de personal humano, se propone impartir formación en pilotaje de este tipo de drones a los trabajadores afectados, para que puedan pasar a formar parte de la empresa. De esta forma, el efecto de esta tecnología supondrá una mejora en sus condiciones de trabajo y una evolución en materia de seguridad, tanto para las empresas como para estos trabajadores.

Viabilidad legal

En esta sección se aborda el contexto legal en el que se encuentra el proyecto, así como todo lo relacionado con las licencias.

Dado que el desarrollo de casi la totalidad de las implementaciones de este proyecto ha sido propia, esta sección será breve en lo que a licencias se refiere.

Software

El proyecto consta de las siguientes dependencias software:

Dependencia	Versión	Descripción	Licencia
Numpy	1.14.2	Se trata de la librería de cálculo científico por excelencia en Python.	BSD
SciPy	1.0.1	Librería de múltiples herramientas para cálculo científico.	BSD
PySerial	3.4	Librería que proporciona acceso a puertos serie.	BSD
Bluetin_Echo	0.1.1	Librería que proporciona una clase para usar los sensores de ultrasonidos HC-SR04.	BSD
UV4L	1.14	Conjunto de drivers para captura de vídeo, y framework WebRTC basado en OpenWebRTC.	BSD

Tabla A.10: Dependencias en backend.

Dependencia	Versión	Descripción	Licencia
Flask	1.0	Se trata de un framework para desarrollo de aplicaciones web en Python.	BSD
Eventlet	0.22.1	Librería de comunicación concurrente para Python.	MIT
SQLAlchemy	1.2.6	ORM para Python.	MIT

Tabla A.11: Dependencias en frontend.

De esta forma, el código desarrollado deberá tener una licencia compatible con MIT y BSD, ambas muy permisivas al respecto. La monetización del proyecto se basa en un sistema de suscripción al mismo, pero la existencia de competencia redunda en una mejora de los productos y servicios, ya que fuerza a las empresas a mantenerse al día. Por tanto, se libera todo el código, tanto del backend como del frontend, bajo licencia BSD de 3 clausulas.

Puede leerse un resumen de la licencia en BSD-3 Clause license

Documentación, imágenes y vídeos

La documentación generada, así como las imágenes utilizadas (salvo dos presentadas en artículos académicos listados en la bibliografía, y sobre las que se ha aportado el crédito correspondiente) y los vídeos han sido de generación propia.

Por lo tanto se establece una licencia *Creative Commons Attribution 4.0 International* (CC-BY-4.0), que permite compartir, adaptar y distribuir para cualquier propósito la obra, incluso con fines comerciales, siempre y cuando se de la debida atribución de la autoría del trabajo.

Puede leerse un resumen de la licencia en CC-BY 4.0.

Resumen

A modo de resumen de las licencias dadas al material del proyecto, se añade la siguiente tabla:

Recurso	Licencia
Código Backend	BSD
Código Frontend	BSD
Documentación	CC-BY-4.0
Imágenes	CC-BY-4.0
Vídeos	CC-BY-4.0

Tabla A.12: Resumen de licencias.

Marco legal

Más allá de tratarse de únicamente un proyecto software, el uso de drones entra en un marco legal muy delicado en España.

La legislación al respecto es de reciente creación, y difiere entre países. A continuación se especifica el marco legal en el que se engloba este proyecto, y las diferentes repercusiones que tiene sobre el uso del mismo.

Legislación sobre Aeronaves

Según el Real Decreto 1036/2017, disponible en BOE - RD1036/2017:

- Para operar un drone **de forma profesional**, es necesario ser dado de alta por AESA[2] como operador de drones.
- No está permitido sobrepasar los 120m de altura.
- Se permite el vuelo sobre aglomeraciones de personas y en zonas urbanas, siempre y cuando la aeronave no sobrepase los 10 kg. Se debe guardar una distancia de seguridad de 50 m con edificios y personas. El piloto no podrá encontrarse a más de 100 m de distancia.
- Se permite el **vuelo fuera del alcance visual** con aeronaves de menos de 2 kg, y que cuenten con medios para detectar y evitar a otros usuarios del espacio aéreo.
- La aeronave deberá contar con una placa identificativa, que conste de nombre del fabricante, tipo, modelo, numero de serie, y nombre del operador y datos de contacto.

Dado el marco establecido en la anterior lista, y dado que el drone creado no sobrepasa los 2 kg de peso, tan solo sería necesario contar con personal cualificado, y con una placa identificativa del dispositivo. Además, tal y como se estipula, se hace referencia a drones con medios para evitar colisiones, lo cual enmarca este proyecto.

Pese a no mentarse el uso en recintos privados, y acogiendo el proyecto a la normativa más restrictiva, parece sencillo y razonable cumplir con la legislación vigente.

GDPR

La reciente implantación de la GDPR[3], lleva consigo una serie de restricciones y notificaciones a tener en cuenta:

- Se debe señalar públicamente que la zona cuenta con sistemas de vídeo vigilancia.
- Asimismo se debe informar que los datos pueden ser utilizados con fines de mejora del producto.
- Se informará del no tratamiento de datos biométricos recogidos durante sesiones de grabación.

[2] Agencia Estatal de Seguridad Aérea
[3] Reglamento General de Protección de Datos de la Unión Europea

- Se informará de los derechos de cancelación y de acceso a los datos en cualquier momento.
- La transmisión de vídeo se hará de forma cifrada (es requisito de WebRTC, de manera que no es complejo de cumplir), bien mediante HTTPS o tuneles TLS.
- Derecho al olvido. Permite que una persona captada por el sistema de vigilancia pueda ejercer su derecho a que los datos sean eliminados, de forma que se detenga su procesamiento o compartición.

Apéndice B

Especificación de Requisitos

B.1. Introducción

En este anexo se recogen las especificaciones de requisitos requeridas para este proyecto. Estas definirán el comportamiento esperado del sistema.

B.2. Objetivos generales

En este proyecto se ha tratado de llevar a cabo los siguientes objetivos de carácter general:

- Diseñar un drone capaz de recorrer un espacio, en el que existan obstáculos, de forma segura.
- Diseñar un sistema de acceso al drone de forma segura. Tanto para controlarlo de forma remota, como para activa los mecanismos de control automatizados.
- Diseñar una interfaz web que permita la visualización en tiempo real de la cámara del drone, así como su control remoto por un operador.

B.3. Catalogo de requisitos

Los siguientes requisitos se derivan de los objetivos generales arriba dispuestos:

Requisitos Funcionales

RF-1 Acceso al sistema: El usuario debe poder iniciar sesión en el sistema.

RF-2 Cierre de sesión: El usuario debe poder cerrar sesión en el sistema.

RF-3 Gestión del sistema: El usuario debe ser capaz de gestionar el sistema completo que hay a bordo del drone.

RF-3.1 Gestión del streaming de vídeo: El usuario debe poder iniciar/detener el streaming de vídeo.

RF-3.2 Grabaciones de vídeo: El usuario debe poder iniciar/detener y descargar las grabaciones de vídeo.

RF-3.3 Gestión de controles del drone: El usuario debe poder activar/desactivar el control manual del dispositivo.

RF-3.3.1 Seguridad automatizada: El drone debe presentar mecanismos de autopreservación. Debe protegerse frente a obstáculos.

RF-3.4 Gestión de control automático del drone: El usuario debe poder activar/desactivar los mecanismos de control automatizado.

RF-3.5 Registro de acciones: Las acciones llevadas a cabo por el usuario deben guardarse en un registro.

RF-4 Gestión del backend: El usuario debe poder acceder a la equipo que controla el backend.

RF-4.1 Gestión del sistema operativo: El usuario debe poder llevar a cabo las tareas de gestión del sistema operativo como crea conveniente.

RF-5 Registro de sucesos: Intentos de intrusión deben ser registrados.

RF-6 Reconocimiento de dispositivos: La aplicación web debe ser capaz de reconocer un dispositivo (joystick o emisora) conectado al equipo, para llevar a cabo el control del drone.

Requisitos no funcionales

RNF-1 Usabilidad: El frontend ha de ser sencillo de utilizar e intuitiva. En el caso de producirse errores, estos deben ser claramente explicados, o registrados.

RNF-2 Rendimiento: Tanto el frontend, como el backend deben funcionar de forma eficiente. Retardos elevados podrían dar lugar a problemas muy serios en la ejecución de las ordenes del usuario.

RNF-3 Capacidad y Escalabilidad: Tanto el frontend como el backend deben ser escalables. El frontend debe ser capaz de soportar una asiduidad grande de usuarios, así como permitir añadir múltiples agentes a cada usuario. El backend, debe permitir añadir controladores sin que haya que reestructurar todo el sistema.

RNF-4 Disponibilidad: El sistema al completo debe ser robusto. El frontend debe estar disponible para su uso el mayor tiempo que sea posible. El backend debe ser capaz de recuperarse de errores, y determinar acciones a tomar en el caso de los mismos.

RNF-5 Seguridad: Tanto el acceso al frontend como al backend deben hacerse de forma segura. En ningún momento el drone debe guardar imágenes de la zona vigilada. El almacenamiento de contraseñas se hará de forma cifrada, y la comunicación entre el backend y el frontend deberá ser segura.

B.4. Especificación de requisitos

Esta sección detallará el diagrama de casos de uso, y detallará cada uno de ellos.

Actores

- **Vigiliante:** Actor a cargo de la gestión del sistema en producción. Interactúa con el FrontEnd.

- **Admin:** Actor a cargo de la gestión del sistema en desarrollo o en caso de resolución de problemas en producción. Interactúa con el BackEnd.

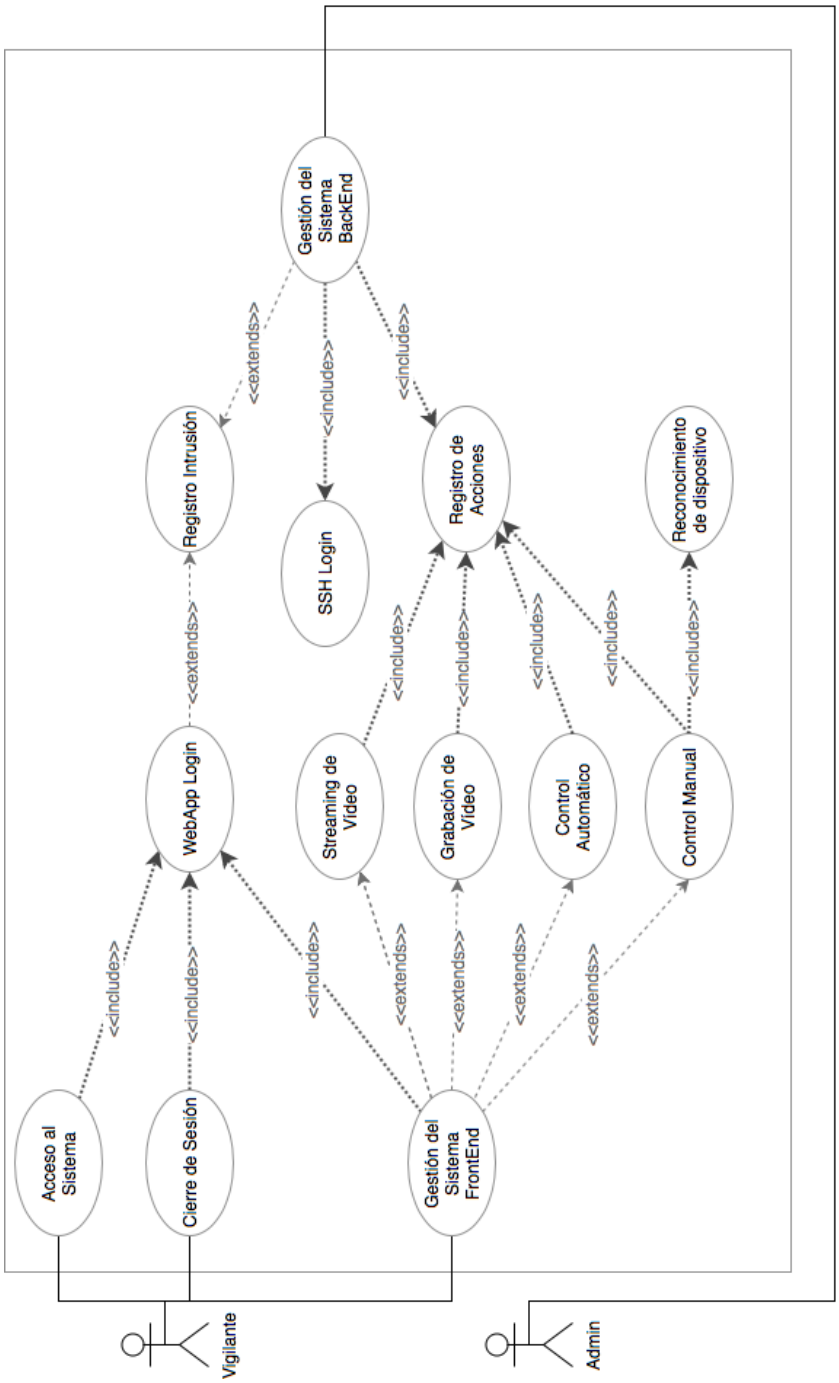

Figura B.1: Diagrama de Casos de Uso.

Casos de Uso

CU-01	**Acceso al sistema**
Actor	Vigilante
Requisitos asociados	RF-1, RF-5
Descripción	Permite al usuario iniciar sesión en el sistema vía web.
Precondiciones	La base de datos está activa. El usuario dispone de cuenta activa.
Acciones	1. El usuario accede a la página web. 2. El usuario introduce sus credenciales. 3. Se muestra la página principal con el logo y los botones asociados al streaming de vídeo.
Postcondición	El sistema habrá asignado al usuario el drone correspondiente.
Excepciones	▪ Contraseña/Usuario incorrectos (mensaje). ▪ No existen drones asociados (mensaje).
Importancia	Alta.

Tabla B.1: CU-01. Acceso al sistema

CU-02	**Cierre de sesión**
Actor	Vigilante
Requisitos asociados	RF-2, RF-5
Descripción	Permite al usuario cerrar sesión en el sistema vía web.
Precondiciones	La base de datos está activa. El usuario dispone de cuenta activa. El usuario tiene iniciada una sesión.
Acciones	1. El usuario accede a la aplicación web. 2. El usuario pulsa sobre el enlace 'Logout' situado en la parte superior izquierda. 3. Se muestra la página de inicio de sesión con un mensaje de despedida.
Postcondición	El sistema habrá dado por cerrada la sesión de usuario. Se redirige al usuario a la página de inicio de sesión.
Excepciones	El usuario no tenía sesión activa. (redirección a login).
Importancia	Alta.

Tabla B.2: CU-02. Cierre de sesión

CU-03	**Gestión del sistema**
Actor	Vigilante
Requisitos asociados	RF-1, RF-3
Descripción	Permite al usuario gestionar las acciones del drone vía web.
Precondiciones	La base de datos está activa. El usuario dispone de cuenta activa. El usuario tiene iniciada una sesión. El usuario tiene asignado un drone.
Acciones	1. El usuario conecta un dispositivo con el número de canales necesarios. 2. Se informa sobre el reconocimiento de dicho dispositivo y se muestran los valores de los canales activos. 3. Se muestra el botón de activación del control manual.
Postcondición	El sistema habrá asignado al usuario el drone correspondiente. La aplicación web habrá reconocido el dispositivo conectado. Se dará la posibilidad de activar el control manual.
Excepciones	Dispositivo no reconocido.
Importancia	Alta.

Tabla B.3: CU-03. Gestión del sistema

CU-04	**Gestión del streaming de vídeo**
Actor	Vigilante
Requisitos asociados	RF-3, RF-3.1 RF-3.5
Descripción	Permite al usuario activar/desactivar el feed de vídeo proveniente del drone.
Precondiciones	La base de datos está activa. El usuario dispone de cuenta activa. El usuario tiene iniciada una sesión. El usuario tiene asignado un drone.
Acciones	El usuario pulsa sobre el botón 'Start/Stop Streaming'
Postcondición	La aplicación web inicia/detiene el vídeo.
Excepciones	Drone no disponible (mensaje).
Importancia	Alta.

Tabla B.4: CU-04. Gestión del streaming de vídeo

CU-05	**Grabación de vídeo**
Actor	Vigilante
Requisitos asociados	RF-3, RF-3.2 RF-3.5
Descripción	Permite al usuario grabar el feed de vídeo proveniente del drone.
Precondiciones	La base de datos está activa. El usuario dispone de cuenta activa. El usuario tiene iniciada una sesión. El usuario tiene asignado un drone.
Acciones	El usuario pulsa sobre el botón 'Record Video'
Postcondición	La aplicación web comienza con la grabación del feed de vídeo.
Excepciones	Feed de vídeo no disponible (mensaje).
Importancia	Baja.

Tabla B.5: CU-05. Grabación de vídeo

CU-06	**Control manual del drone**
Actor	Vigilante
Requisitos asociados	RF-3, RF-3.3 RF-3.5, RF-3.3.1, RF-6
Descripción	Permite al usuario controlar el drone de forma manual.
Precondiciones	La base de datos está activa. El usuario dispone de cuenta activa. El usuario tiene iniciada una sesión. El usuario tiene asignado un drone. El usuario ha conectado un dispositivo adecuado.
Acciones	El usuario pulsa sobre el botón 'Enable/Disable Manual'
Postcondición	La aplicación web solicita la activación del control manual al sistema. El sistema se conecta al drone y transmite la información proveniente de la aplicación web.
Excepciones	Drone no disponible (mensaje).
Importancia	Alta.

Tabla B.6: CU-06. Control manual

CU-07	**Gestión de controles automáticos**
Actor	Vigilante
Requisitos asociados	RF-3, RF-3.4 RF-3.5
Descripción	Permite al usuario activar/desactivar los mecanismos de control automatizado del drone.
Precondiciones	La base de datos está activa. El usuario dispone de cuenta activa. El usuario tiene iniciada una sesión. El usuario tiene asignado un drone.
Acciones	1. El usuario selecciona los mecanismos de control automatizado que desea emplear. 2. El usuario pulsa sobre el botón 'Enable auto'
Postcondición	La aplicación web solicita la activación de los controles automatizados al sistema. El sistema se conecta al drone y transmite la información proveniente de la aplicación web.
Excepciones	▪ Drone no disponible (mensaje). ▪ Mecanismo automatizado no reconocido (mensaje).
Importancia	Media.

Tabla B.7: CU-07. Gestión de controles automáticos

CU-08	**Gestión del backend**
Actor	Admin
Requisitos asociados	RF-4, RF-5
Descripción	Permite al usuario gestionar el equipo que controla el drone.
Precondiciones	El usuario dispone de credenciales de acceso al sistema.
Acciones	1. El usuario inicia sesión en el sistema. 2. Se informa sobre el registro de intentos de acceso no autorizado. 3. Se muestra prompt del sistema a la espera de órdenes.
Postcondición	El sistema operativo habrá registrado el inicio de sesión del usuario. El sistema operativo habrá iniciado sesión con el usuario solicitado.
Excepciones	Acceso denegado, pubkey. (mensaje)
Importancia	Alta.

Tabla B.8: CU-08. Gestión del backend.

Apéndice C

Especificación de diseño

C.1. Introducción

En este anexo se definirá la forma en que se han resuelto los requisitos expuestos anteriormente. Se dará una visión detallada de la arquitectura del sistema, diferenciando la aplicación web y el agente, o drone.

C.2. Diseño de datos

WebApp

La aplicación visible por el usuario final maneja los siguientes datos, o entidades:

- Usuario: Un usuario dispone de un nombre, apellidos, una dirección de correo electrónico, un nombre de usuario (que utilizará para iniciar sesión), y una contraseña cifrada. Se genera un id automáticamente para cada usuario.
- Drone: Un drone dispone de un nombre, una localización, un controlador (que se corresponde con el id de usuario que lo controla), un puerto para el control remoto y un puerto para el feed de vídeo Se genera un id automáticamente para cada drone.
- Registro de sesión: La aplicación web guarda registros del inicio/cierre de sesión por parte de un usuario. Dispone del identificador de usuario y de un timestamp.

- Registro de intrusión: La aplicación web guarda registros de inicios de sesión infructuosos. Dispone del usuario utilizado, y un timestamp.
- Registro de acciones: La aplicación web guarda registros de las acciones llevadas a cabo por cada usuario. Dispone de un identificador del usuario que realiza la acción, un timestamp y un identificador de la acción realizada.
- Acciones: Se trata de las acciones que realizan los usuarios. Dispone de un identificador y una descripción de la acción.

Agente

El agente basa su funcionamiento en el uso de las siguientes entidades:

- Controladores PID: Los controladores PID se encargan de proveer de valores para los diferentes canales automatizados del agente. Disponen de los coeficientes de las distintas componentes proporcional, integral y derivativa, así como de un límite superior y un límite inferior.
- Sensores: Los sensores de distancia disponen de un pin de disparo, un pin de recepción del eco, y un ángulo.
- Control Remoto: Se trata de un pequeño servidor encargado de recibir las órdenes de un usuario remoto. Dispone de una dirección y un puerto en el que escuchar.
- MSPio: Se trata de la implementación del protocolo MultiWiiSerial-Protocol, el cual permite establecer comunicación con la controladora de vuelo. Dispone de un puerto serie y una velocidad de transferencia (en baudios).
- Evasión de obstáculos: Se trata de la implementación del algoritmo VFH. Dispone de las siguientes entidades:
  - Malla de histograma: Una representación del entorno del agente. Dispone de las medidas de los sensores, distancias máxima y mínima a cubrir, el mapa del entorno, el tamaño de la ventana a cubrir, el tamaño de las celdas que componen la malla, una medida del error de los sensores y su apertura.
  - Histograma polar: Una representación basada en sectores del entorno del agente. Dispone de una malla de histograma y el ancho de los sectores.

- Capa de salida: La capa de salida es la que realiza el cálculo de la orientación necesaria. Dispone de un histograma polar, un umbral de navegación segura, y la cantidad de sectores del histograma polar que se requieren para considerar un espacio como 'amplio'.

- Sistema de localización: Se trata de la implementación del algoritmo del Filtro de Partículas. Dispone de una representación del entorno del agente, medidas de error para el control de rotación, de desplazamiento y de toma de medidas de los sensores, y el número de sectores a generar en cada celda.

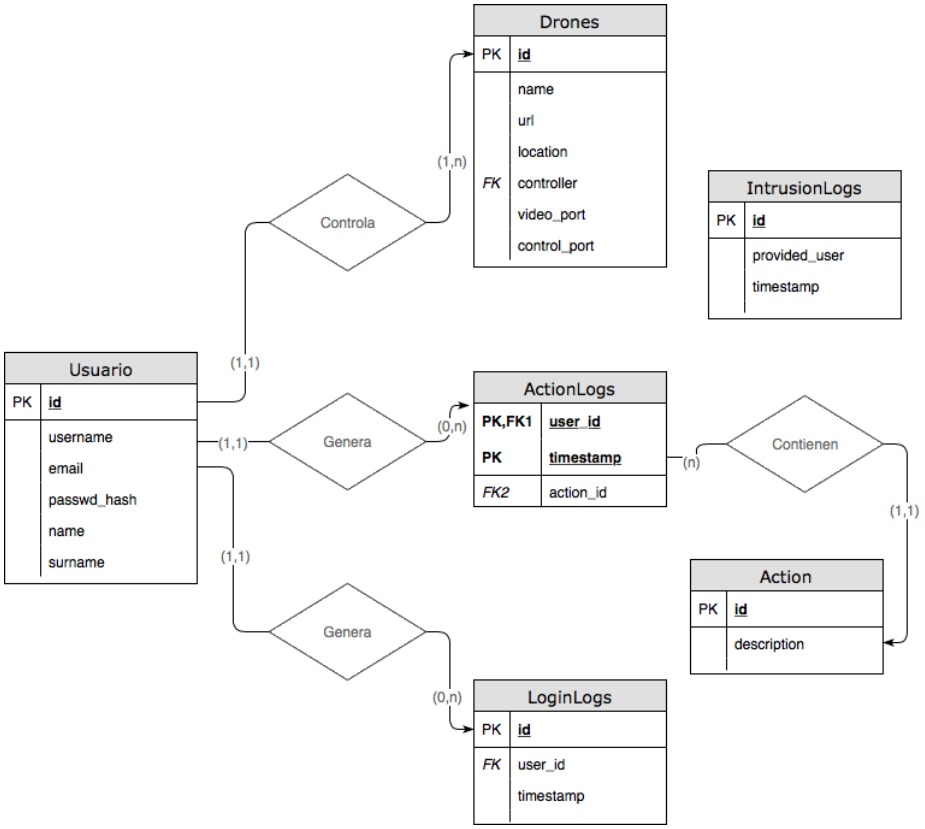

Figura C.1: Diagrama E/R para WebApp.

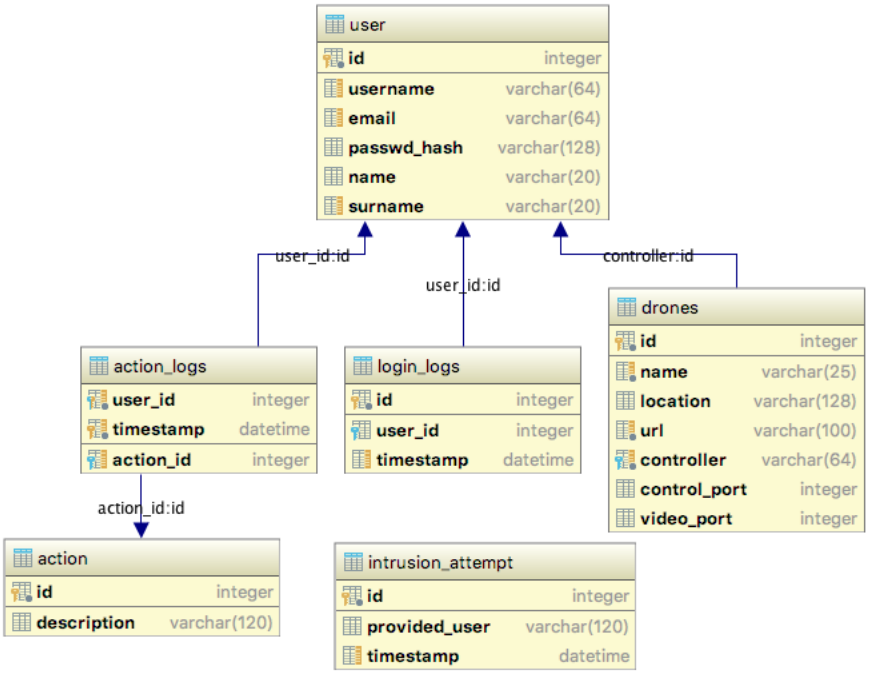

Figura C.2: Diagrama Relacional WebApp. Generado mediante PyCharm.

C.3. Diseño procedimental

En este apartado se recoge el diagrama secuencial correspondiente a la totalidad del sistema. En él, se representa el flujo de ejecución que puede o no, ver subsección C.4, seguir el sistema creado.

C.4. Diseño arquitectónico

En esta sección se dará una aclaración al diagrama C.3, el cual da una idea general del funcionamiento del sistema. Para lograr un código fácil de mantener, y de escalar dado el potencial de crecimiento de las tecnologías relacionadas con este proyecto, se han generado una serie de clases intermedias que actúan a modo de *wrapper*. La interfaz de la aplicación se ha generado siguiendo el conocido patrón MVC.

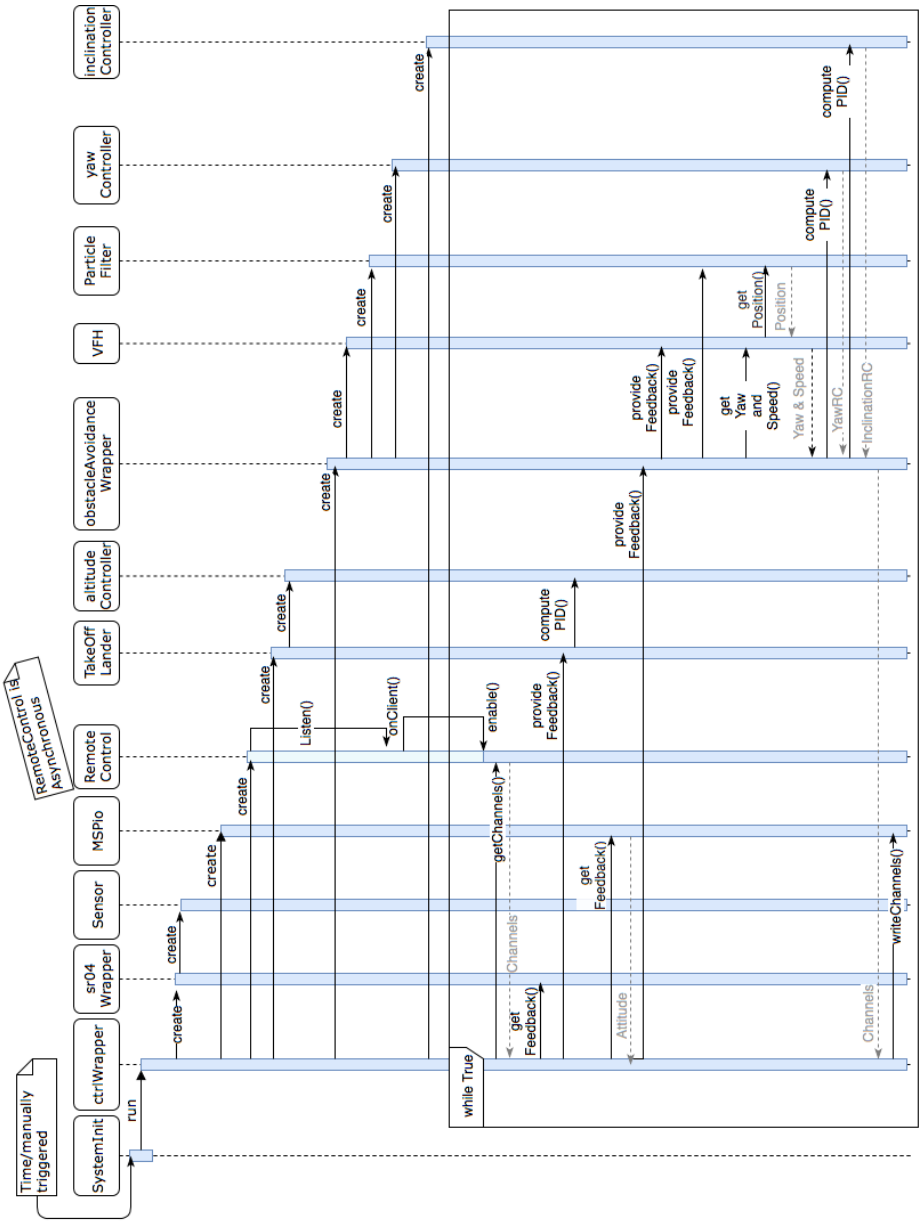

Figura C.3: Diagrama de secuencia del sistema.

Patrón MVC

Para la parte representada por la aplicación web, se ha hecho uso del habitual patrón MVC (*Model View Controller*), con la intención de separar la capa de lógica de la aplicación (*Controller*), de la interfaz de usuario (*View*) y de los datos manejados (*Model*).

El patrón sigue la siguiente disposición:

- **Model**: El modelo de la aplicación representa los datos a los que la aplicación accederá. Almacena los mismos y los gestiona. En el caso de este proyecto se trata de la base de datos de la aplicación Flask.

- **View**: La vista de la aplicación muestra al usuario una interfaz gráfica sobre la que trabajar. Al recibir acciones, se encarga de solicitar al controlador las medidas necesarias. En el caso de este proyecto se corresponde con la interfaz web de la aplicación Flask.

- **Controller**: El controlador podría considerarse el punto de comunicación entre el modelo y la vista. Se encarga de sincronizar los datos almacenados en el modelo con lo presentado en la vista, y de recibir los eventos generados por la vista y actuar en consecuencia.

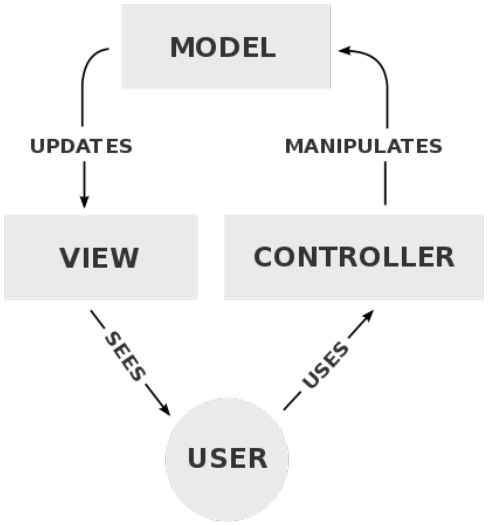

Figura C.4: Patrón MVC. Imagen de Wikipedia.

Patrón Catálogo

El patrón catálogo no es tan habitual. Se utiliza sobre todo en lenguajes en los que se permite pasar una función o método como parámetro. Por ello en Python, donde todos los elementos son objetos de primera clase, es posible hacer uso de este patrón de forma muy sencilla.

Permite establecer el comportamiento de una instancia más genérica, estableciendo los métodos generales de esta a métodos especializados de la clase que encapsula. En este caso, se encarga de proporcionar un acceso genérico a los métodos especializados de cada controlador automatizado.

Este patrón es el núcleo de funcionamiento del sistema de control del agente. En el diagrama C.3 puede verse como se instancian numerosos controladores (*RemoteControl, TakeOffLander* y *obstacleAvoidanceWrapper*). Dichos controladores, en el momento de ser instanciados por el *ctrlWrapper* se encapsulan en una clase interna denominada *PrioritizedController*. Dicha clase, podría ser considerada como la capa externa de un patrón proxy de protección[1], el cual permite comprobar si el controlador está activo, si requiere de información de los sensores para su funcionamiento, y si requiere ser bloqueado para usarse (para protegerse de accesos simultáneos en el caso de tener un comportamiento asíncrono). Además, provee de información sobre la *prioridad* del controlador.

La prioridad de un controlador es lo que hace tan flexible el sistema. El diagrama C.3 presenta un uso secuencial de los diferentes *PrioritizedController* generados, sin embargo, el orden de ejecución de cada uno, puede cambiar de forma dinámica. El *ctrlWrapper* basa el procesamiento de los canales de control (Acelerador, Balanceo, Inclinación, Rotación, Armar/Desarmar... etc) en la prioridad de los controladores.

Por ejemplo: Supongamos que existen dos controladores, uno remoto y manual (este es el usuario conectado al drone a través de la interfaz web) con una prioridad igual a 5, y un controlador de rotación y velocidad autónomo, con una prioridad igual a 10.

La solicitud de los valores de los canales, se realiza por orden de prioridad de menor a mayor. De manera que el controlador automático de rotación y velocidad sobreescribirá el input del usuario (solo en los canales controlados), ya que tiene una prioridad superior. Supongamos que el usuario no desea eso. Prefiere ser él quien controle por completo el sistema, así que sencillamente arrancaría el sistema estableciéndose a si mismo como el controlador más prioritario. Sobreescribiendo de esta forma al controlador automatizado.

[1] El patrón proxy de protección controla el acceso al objeto original

Arquitectura general

La arquitectura de la parte visible del proyecto, la aplicación web, no presenta una complejidad elevada. Sin embargo el sistema de control del agente no es tan intuitivo. Por ello, se presenta el siguiente diagrama C.5 a modo aclaratorio.

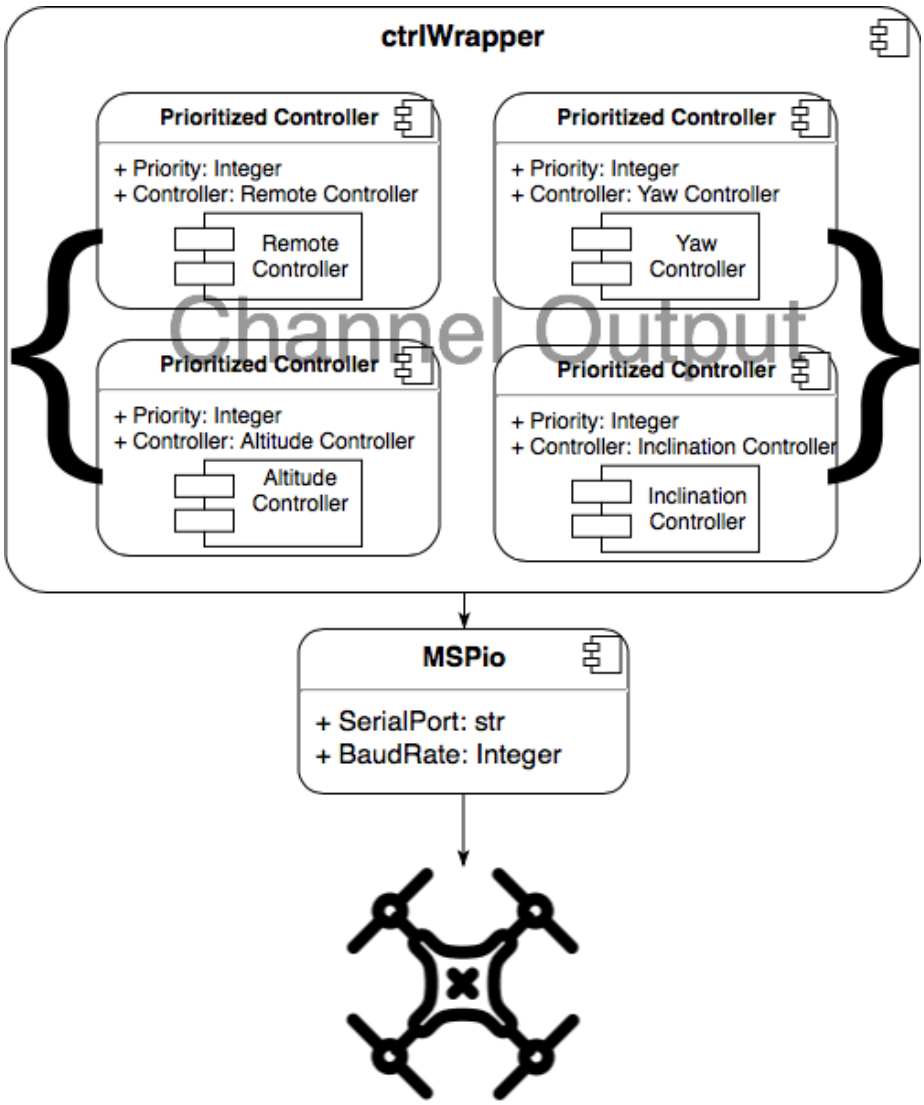

Figura C.5: Encapsulación de controladores mediante controladores priorizados.

El uso de esta arquitectura presenta varias ventajas:

- Los controladores son independientes de la implementación del sistema de control.

- La arquitectura generada puede ser utilizada para controlar cualquier sistema basado en varios sistemas de control que se quieran priorizar.

- Permite llevar a cabo pruebas con mayor seguridad, dado que se puede hacer uso de un controlador manual solo cuando este se active, de manera que los controladores automatizados en pruebas puedan ser interrumpidos en cualquier momento.

Diseño de paquetes

Para organizar las diferentes partes del proyecto, se ha generado una estructura de paquetes que diferencia la ubicación de los mismos (agente o servidor) de forma sencilla.

En la figura C.6 puede verse la estructura de paquetes seguida en el agente. Los paquetes contienen:

- **algorithms**: Las implementaciones de los algoritmos empleados.

- **autoControllers**: Los diferentes sistemas de control automatizados implementados.

- **comms**: Los sistemas de comunicación implementados. Tanto entre el agente y la RaspberryPi, como entre el usuario y la RaspberryPi.

- **sensors**: Las implementaciones de adquisición de datos de los sensores usados.

- **systemControl**: Contiene la implementación del sistema de control del agente.

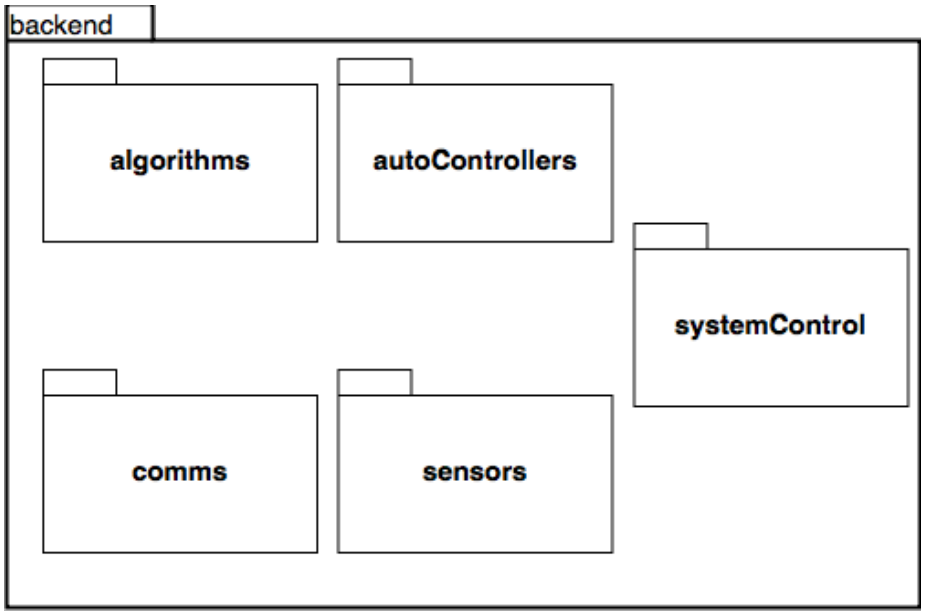

Figura C.6: Estructura de los paquetes en el agente.

En la figura C.7 puede verse la estructura de paquetes seguida en el servidor web. Los paquetes contienen:

- **controller**: Contiene la implementación del controlador de la aplicación web.
- **model**: Contiene la implementación del modelo de datos utilizado por el controlador.
  - *migrations*: Contiene un mecanismo para realizar operaciones de upgrade/downgrade de las diferentes versiones de la base de datos.
    - versions: Contiene las versiones de la base de datos
- **view**: Contiene la vista de la aplicación web.
  - *static*: Contiene los elementos estáticos de la aplicación web.
  - *templates*: Contiene las plantillas que representan la interfaz de la aplicación web.

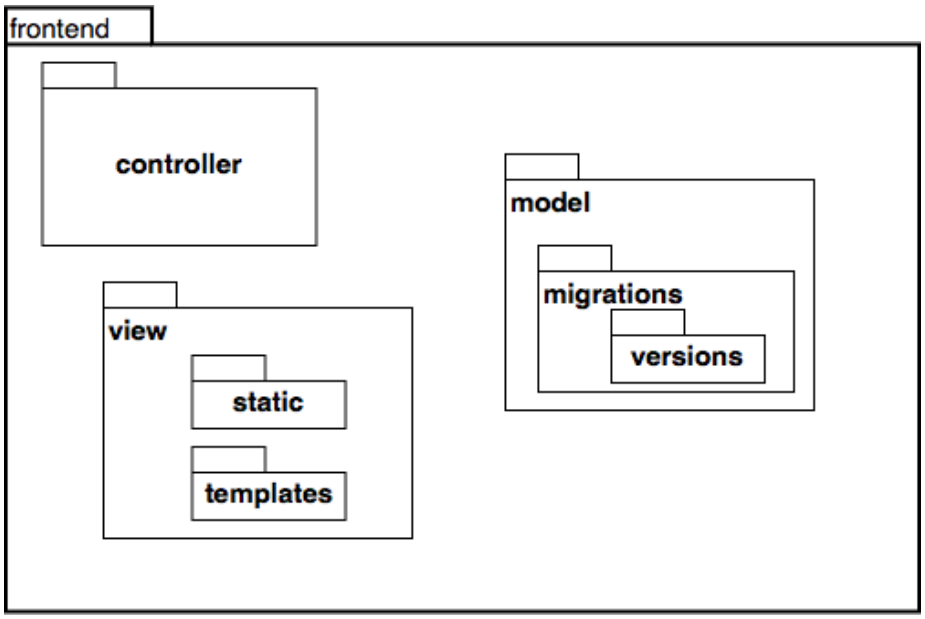

Figura C.7: Estructura de los paquetes en el servidor web.

Diseño de clases

Siguiendo con la nomenclatura utilizada en el diseño de la estructura de paquetes, se dará a continuación las características de cada clase implementada:

FrontEnd

- `controller.__init__`: Establece una serie de opciones necesarias para el arranque de la aplicación web. Rutas de archivos, instancia la aplicación en sí, establece la configuración, la base de datos, el servicio de login y la capa de sockets utilizada para comunicar la interfaz de usuario de forma asíncrona con la aplicación.

- `controller.routes`: Se trata de la clase encargada del control de la aplicación web. Responde a las solicitudes de los usuarios realizando las acciones correspondientes, y establece la comunicación necesaria entre el agente y el usuario.

- `model.config`: Se trata de una clase que contiene elementos de configuración de la aplicación web. Tal como la ubicación de la base de

datos a utilizar, la clave secreta para generar los tokens de la aplicación, etc.

- **`model.models`**: Se trata de la definición de las entidades a utilizar por el ORM utilizado (SQLAlchemy).
- **`model.migrations.env`**: Se trata de un script generado por Alembic (una extensión para SQLAlchemy que permite realizar versionado de las bases de datos), y que permite actualizar, o revertir, los cambios realizados basándose en las diferentes versiones generadas.
- **`model.migrations.versions.XXXXXX`**: Se trata de los diferentes scripts que usará Alembic para actualizar o revertir los cambios realizados en la base de datos.
- **`view.login`**: Un formulario simple a utilizar por el controlador a la hora de obtener los datos de login.
- **`frontend.droneControlWebUI`**: El instanciador del servidor web. Se encarga de ejecutar la aplicación web mediante los parámetros especificados.

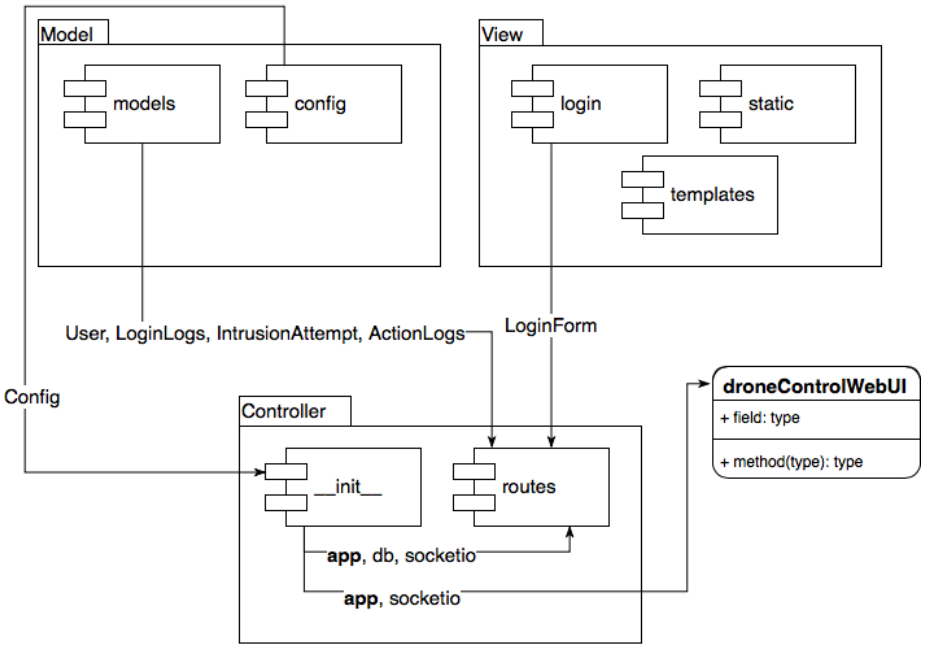

Figura C.8: Diagrama de clases del frontend.

BackEnd

- **`algorithms.Geometry`**: Se trata de un módulo auxiliar utilizado por el Filtro de Partículas para realizar cálculos geométricos de forma vectorizada haciendo uso de Numpy.
- **`algorithms.ParticleFilter`**: Se trata de la implementación del Filtro de Partículas. Se encarga de obtener la posición del agente en el plano.
- **`algorithms.VFH`**: Se trata de la implementación del sistema de evasión de obstáculos.
- **`autoControllers.abcControllerPID`**: Se trata de una metaclase[2] que define el contrato que deben subscribir las clases que implementen controladores PID.
- **`autoControllers.altitudeController`**: Se trata de una implementación de la metaclase abcControllerPID, destinada al control de altitud.
- **`autoControllers.inclinationController`**: Se trata de una implementación de la metaclase abcControllerPID, destinada al control de inclinación.
- **`autoControllers.yawController`**: Se trata de una implementación de la metaclase abcControllerPID, destinada al control de la orientación.
- **`autoControllers.obsAvoidanceWrapper`**: Se trata de una clase implementada para encapsular los algoritmos de evasión de obstáculos, y generar el cálculo del control de orientación e inclinación para alimentar los controladores PID correspondientes.
- **`autoControllers.takeOffLanding`**: Se trata de una clase implementada para encapsular el controlador de altitud. Permite tanto el despegue como el aterrizaje, y genera el cálculo del control de altitud para alimentar al controlador PID correspondiente.
- **`comms.MultiWiiProtocol`**: Esta es la implementación del protocolo MultiWiiSerial Protocol. Dicho protocolo es utilizado por las herramientas de configuración de las controladoras de vuelo de los drones

[2]En Python las interfaces se denominan *metaclases*, aunque las capacidades de estas son más extendidas, servirá para referirnos a ellas de aquí en adelante.

actuales, que ejecutan un firmware basado en el controlador MultiWii. Se encarga de llevar a cabo la comunicación entre el drone y la RaspberryPi, permitiendo el flujo de información en ambas direcciones.

- **comms**.RemoteControl: Se trata del servidor encargado de recibir las órdenes remotas del usuario.
- **sensors**.sr04Wrapper: Se trata de una pequeña clase que encapsula el acceso a los sensores de ultrasonidos utilizados para el cálculo de distancias, para el sistema de evasión de obstáculos.
- **sensors**.Sensor: Se trata de una pequeña clase que representa un sensor de ultrasonidos, incluyendo el acceso a la instancia de la clase *Echo* (perteneciente a la librería *Bluetin_Echo*), así como el ángulo en el que está orientado el sensor, y los pines a los que está conectado.
- **systemControl**.ctrlWrapper: Se trata de la implementación del sistema de control del drone. Se posiciona como un mecanismo de separación entre el drone y sus sensores, y los diferentes controladores. Se encarga de coordinar, de forma priorizada, los diferentes mecanismos de control del agente, así como de suministrar a aquellos que así lo requieran la información de los diferentes sensores a los que tiene acceso.

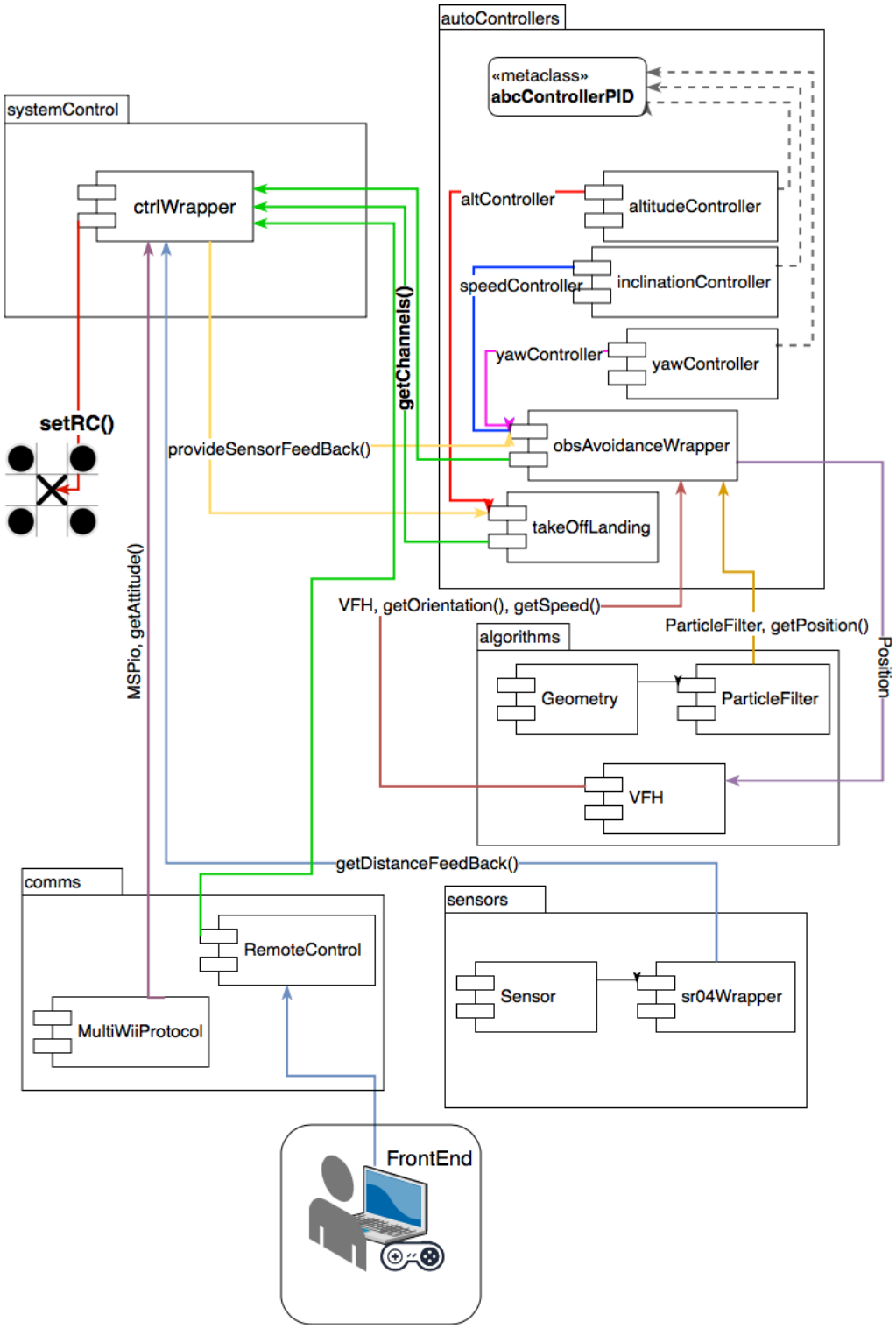

Figura C.9: Diagrama de clases del backend.

Apéndice D

Documentación técnica de programación

D.1. Introducción

D.2. Estructura de directorios

D.3. Manual del programador

D.4. Compilación, instalación y ejecución del proyecto

D.5. Pruebas del sistema

Apéndice E

Documentación de usuario

E.1. Introducción

E.2. Requisitos de usuarios

E.3. Instalación

E.4. Manual del usuario

Bibliografía

[1] S. Social, "Bases y tipos de cotización 2018," 2018, [Internet; descargado 1-junio-2018]. [Online]. Available: http://www.seg-social.es/Internet_1/Trabajadores/CotizacionRecaudaci10777/Basesytiposdecotiza36537/index.htm

[2] UBU, "Retribuciones segun ii convenio de pdi laboral ano 2017," 2017, [Internet; descargado 2-junio-2018]. [Online]. Available: http://www.ubu.es/sites/default/files/portal_page/files/pdi_laboral_2017_2.pdf

[3] J. C. R, "Ley de navegacion aerea," 2001, [Internet; descargado 2-junio-2018]. [Online]. Available: http://www.fomento.gob.es/NR/rdonlyres/B8761680-F6C0-48CC-ABCD-3F5CD7480008/71900/RD_37_2001.pdf