Grado en Ingeniería Informática

Trabajo Final de Grado

Desarrollo de un módulo de meteorología para PhotoPills en iOS

Autor:
Daniel Querol Beltrán

Supervisor:
Germán Marquès Marquès
Tutor académico:
Reyes Grangel Seguer

Fecha de lectura: 28 de juny de 2019
Curso académico 2018/2019

Resumen

En este documento se presenta el Trabajo Final de Grado (TFG) realizado durante la estancia en prácticas en la empresa Photopills, . Se describe el proceso de desarrollo mediante la metodología Scrum de un nuevo módulo de meteorología orientado al campo de la fotografía al aire libre que se integrará en la aplicación PhotoPills en su versión para sistemas iOS. Se incluye información sobre las diferentes fases del desarrollo del proyecto. Así como un estudio comparativo de los diferentes servidores meteorológicos con el fin de conocer la mejor opción para ser contratada como fuente de datos. Finalmente, se presentan los resultados obtenidos y se discuten las posibles mejoras.

Palabras clave

Desarrollo aplicaciones móvil, aplicación meteorológica iOS, meteorología, Swift, API rest.

Keywords

Mobile apps development, Weather app, iOS, Swift, API rest.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

1.1. Contexto

En este documento se describe el proyecto realizado por el estudiante Daniel Querol Beltrán como Trabajo Final de Grado (TFG) de Ingeniería Informática en el itinerario de Ingeniería del Software. Este proyecto se ha realizado durante la estancia en prácticas en la empresa Photopills, S.L. [10], propietaria de la aplicación para smartphones, también llamada, PhotoPills.

Esta empresa, con sede fiscal en Menorca, tiene como particularidad que sus empleados trabajan de forma remota, por lo que la estancia en prácticas se ha realizado a distancia, desde el domicilio del estudiante.

Photopills, S.L. nació tras el éxito de la aplicación para iOS[1] llamada PhotoPills, una aplicación dirigida a fotógrafos profesionales y aficionados exigentes que tiene como objetivo ayudar a la planificación de fotografías al aire libre.

La mayoría de fotógrafos siguen un proceso común para la realización de fotos. Empieza con una idea de una fotografía que se podría realizar en una localización concreta, seguida de una planificación para conocer el mejor momento para realizar la foto y finalmente su realización.

La app[2] PhotoPills ayuda a la planificación de fotografías. Incluye multitud de herramientas que permiten conocer el tono de la luz en un determinado momento del día, la trayectoria y posición del sol o la luna en el cielo, las horas de luz, la hora dorada[3], la posición del la Vía Láctea[4], las fases en las que se encuentra la luna, información sobre eclipses lunares o solares, etc.

En los últimos años la aplicación PhotoPills ha crecido mucho, incorporando nuevas funcio-

[1] Sistema operativo de los smartphones y tabletas de Apple.

[2] Aplicación informática diseñada para teléfonos móviles inteligentes.

[3] Momento del día en que la luz tiende a tener un color dorado. Se produce tras la salida del sol y antes de la puesta de sol.

[4] Galaxia en la que se encuentra el sistema solar. Desde la Tierra se puede observar en el cielo el centro galáctico en noches despejadas, con cielos ósculos sin contaminación lumínica.

nalidades, como el uso de realidad aumentada para visualizar la trayectoria del sol, la luna o el arco de la Vía Láctea. Actualmente está disponible la versión para smartphones Android. Todo esto ha hecho que cuenten con más de trescientos mil usuarios entre las dos plataformas (iOS y Android). Además de su gran éxito con la marca PhotoPills, que cuenta en sus redes sociales con más de 100 seguidores, especialmente activos en su cuenta de Instagram, donde realizan un concurso diario para elegir la mejor foto del día, las mejores fotos mensuales y la mejor foto del año.

En su afán de mejorar la aplicación y proporcionar a los usuarios más herramientas para la planificación de sus fotos, Photopills, S.L. propone realizar un proyecto informático para incorporar un nuevo módulo en su aplicación centrado en la meteorología.

1.2. Motivación

En la fotografía al aire libre se deben tener en cuenta aspectos como el paisaje, la dirección de la luz, la composición de los elementos fotografiados o la meteorología, entre muchos otros, si se quiere capturar una buena imagen. La meteorología es uno de los aspectos más importantes a tener en cuenta, no solo influirá en la realización de la actividad al aire libre, sino que será determinante para el resultado de la fotografía.

Pese a que se podría pensar que las situaciones de buen tiempo son la más buscadas para realizar fotografías, no siempre es así, dependiendo del estilo o el efecto que se quiera conseguir se buscan nubes de tormenta, cielos despejados, días lluviosos u otras adversidades meteorológicas.

El módulo de meteorología, unido al resto de módulos de la aplicación PhotoPills, proporcionará información sobre el estado actual del tiempo y la previsión. De esta forma, el usuario dispondría de toda la información necesaria para planificar sus fotos dentro de la app PhotoPills. Esto supondría una ventaja competitiva respecto a las otras aplicaciones destinadas a la planificación fotográficas, que no incluyen este tipo de información.

1.3. Objetivos

En esta sección se describe los objetivos generales del proyecto y objetivos específicos del producto. Además, se incluye el alcance, restricciones y riesgos del proyecto.

1.3.1. Objetivos generales del proyecto

El objetivo del proyecto propuesto por PhotoPills y objeto de este TFG es dotar a la aplicación PhotoPills, en su versión para iOS, de la información meteorológica necesaria para que los usuarios puedan planificar sus ideas fotográficas.

La aplicación PhotoPills está organizada en módulos o, como internamente se llaman, *píldo-*

ras. Existe la *píldora* de Planificación, que permite a los usuarios planificar fotos en una localización concreta usando un mapa, la *píldora* de Sol, con información sobre las fases a lo largo del día, la *píldora* de RA Noche, con una herramienta de realidad aumentada para poder localizar la Vía Láctea enfocando el móvil al cielo, entre muchas otras. Esta nueva funcionalidad meteorológica se incorporará como una nueva *píldora*, que se llamará *píldora* de Meteorología, y mostrará información sobre el estado actual del tiempo y el pronóstico para los días siguientes. Se diferenciará de otras aplicaciones meteorológicas en que incluirá, a parte de la información general, datos explícitamente orientados al campo de la fotografía, dando especial énfasis en parámetros que pueda ser de interés, como podría ser el porcentaje o la altura de las nubes, la visibilidad o el punto de rocío, entre otros parámetros.

1.3.2. Alcance del proyecto

En esta sección se pretende analizar el alcance desde el punto de vista funcional, organizacional e informático, indicando hasta donde llegará el proyecto, que partes incluirá y cuales no.

1.3.3. Alcance funcional

La *píldora* de Meteorología deberá incluir las siguientes funcionalidades:

- Permitirá al usuario elegir una localización para la que se quiera obtener información meteorológica.
- Permitirá al usuario gestionar las localizaciones, pudiendo dar de alta o baja una localización y listar las localizaciones guardadas.
- Se proporcionará al usuario información sobre el estado actual del tiempo para una localización concreta. Esta información incluirá datos como temperatura actual, temperatura mínima y máxima del día, velocidad y dirección del viento y un icono representativo de las condiciones meteorológicas.
- Mostrará al usuario un cronograma con el pronóstico para las próximas horas. El cronograma tendrá para cada hora un icono descriptivo con las condiciones meteorológicas y los parámetros de temperatura, dirección y velocidad del viento y precipitaciones.
- Se mostrará al usuario un resumen del pronóstico para los próximos días. En el que se incluirá para cada día la temperatura máxima y mínima, velocidad media del viento y precipitación acumulada.
- A partir de los datos de los pronósticos se representarán gráficas que permitirán a los usuarios entender mejor la evolución meteorológica. Se incluirán gráficas de la evolución de la temperatura, la humedad y la velocidad del viento.
- Con el fin de mostrar el pronóstico con todos los parámetros meteorológicos, se representará una tabla con información sobre: temperatura real, temperatura aparente, punto de rocío, velocidad y dirección del viento, velocidad de las ráfagas de viento, visibilidad,

humedad, presión, porcentaje de nubosidad, porcentaje de nubes bajas, medias y altas, probabilidad de precipitación, precipitación acumulada y nieve acumulada.

La aplicación no incluirá el cálculo de previsiones meteorológicas ni otros parámetros relacionados debido a su complejidad y los recursos necesarios. Por lo que se obtendrán los datos de un servidor meteorológico contratados.

Alcance organizacional

Photopills, S.L. es una pequeña empresa formada por tres trabajadores a tiempo completo y dos colaboradores. Tiene como sede fiscal Menorca, pero todos sus miembros trabajan desde casa, de forma remota. Por ese motivo, la estancia en prácticas se realizará de forma remota, manteniendo comunicación con el supervisor por videoconferencia y chat.

Su aplicación PhotoPills está disponible en las tiendas de iOS y Android. La compra de la app consiste en un único pago, que se realiza en estas tiendas en el momento de la descarga.

La aplicación está dirigida a fotógrafos, ya sea para uso aficionado como a nivel profesional. Por este motivo, la aplicación está diseñada para que sea intuitiva para los usuarios aficionados, y al mismo tiempo proporciona funcionalidades avanzadas para aquellos usuarios más exigentes y familiarizados con aspectos técnicos.

Alcance informático

Las *píldora* de Meteorología se conectarán a un proveedor de datos para obtener la información meteorológica. Por lo tanto, la aplicación se centrará en obtener y mostrar de forma clara la información sobre el estado actual y los pronósticos, pero no calculará evoluciones del tiempo u otros parámetros relacionados.

Las funcionalidades que se realizán en este proyecto deberán aplicarse a la versión de la aplicación para dispositivos con sistema operativo iOS, es decir, para iPhones e iPads. Por lo tanto, se deberá tener en cuenta los diferentes dispositivos que podrán ejecutar esta aplicación, desde las versiones más antiguas de iOS para las que la aplicación esté disponible, así como aprovechar el potencial y las resoluciones de las nuevas versiones de dispositivos de Apple.

El entorno de desarrollo usado para realizar este proyecto será un ordenador portátil Macbook Pro. Este requisito se debe a que solo el sistema operativo macOS tiene disponible el Xcode.[5] Los lenguajes de programación disponibles con los que se puede desarrollar aplicaciones nativas para este sistema operativo son Object C y Swift. Aunque la aplicación PhotoPills está desarrollada en Object C, en este caso se desarrolló con Swift por ser un lenguaje más moderno y existir una clara tendencia en los desarrolladores a usarlo. Esta combinación de lenguajes en la misma aplicación iOS no supone ningún problema, ya que son completamente compatibles.

[5]Entorno de desarrollo para la creación de aplicaciones para sistemas operativos de Apple (iOS, Apple TV, macOS, Apple Watch).

Destacar que en la aplicación PhotoPills los usuarios no se registran, sino que sus datos se mantienen en el dispositivo donde se ha instalado la aplicación, pudiendo hacer copias de seguridad y cargarlas en otros dispositivos.

1.3.4. Objetivos específicos del producto

Este proyecto es propuesto por Photopills, S.L. dando unas pautas sobre la información que debería mostrar la *píldora* de Meteorología, pero sin cerrarse a una lista concreta de requisitos que debe cumplir el producto final. Por este motivo el proyecto empieza con una fase de investigación para conocer mejor los requisitos técnicos y funcionales que deberá cumplir.

Tras una primera fase de investigación y elicitación con el propietario del producto, que en este caso es el supervisor de la estancia en prácticas, se obtienen los siguientes requisitos:

- La aplicación deberá mostrar información meteorológica general e información más avanzada que pueda ser relevante para los fotógrafos. Se deberá investigar que parámetros meteorológicos son los más importantes y la forma en que se deben mostrar.
- Se deberá mostrar información meteorológica sobre el estado actual y el pronóstico para los próximos días. Los pronósticos se deberán listar desglosados por horas y resumidos por días.
- La información meteorológica se deberá obtener de un servidor meteorológico especializado. Se deberá investigar empresas que proporcionen esta información a nivel global y cual de ellas es la más idónea para contratar.
- Se deberá optimizar las llamadas al servidor meteorológico para evitar sobrecostes.
- La aplicación deberá funcionar y mostrarse de forma correcta en cualquier dispositivo con sistema operativo iOS. Por lo tanto se deberá tener en cuenta las diferentes versiones de iPhone e iPad.
- Será necesario diseñar una interfaz que mantenga el estilo de la aplicación PhotoPills y muestre los datos de forma clara usando tablas y gráficos de líneas para mostrar la evolución del tiempo.
- Se deberán incorporar diferentes mapas del tiempo, como el de temperatura, nubosidad, radar de precipitaciones, etc.
- Los usuarios podrán dar de alta localizaciones para las que quieran obtener la información meteorológica. Estas se deberán guardar para que el usuario pueda volver a consultarla de forma rápida.

1.3.5. Costes y beneficios

Con el desarrollo de este nuevo módulo de meteorología para la aplicación PhotoPills, se quiere incluir en la app uno de los factores más importantes a tener en cuenta a la hora de planificar fotografías, que hasta el momento no se disponía. La empresa pretende seguir mejorando

la aplicación con el fin de obtener una ventaja competitiva respecto de la competencia. En los últimos años, PhotoPills ha logrado una ventaja respecto a la competencia gracias a las mejoras que se han añadiendo de forma constante, como la incorporación de herramientas de realidad aumentada o el lanzamiento de la versión PhotoPills para Android.

1.3.6. Recursos y restricciones

Para el desarrollo de este proyecto la empresa establece unas pautas sobre el uso del servidor meteorológico. Las peticiones de datos deben minimizarse al máximo, reduciendo el número de llamadas y de esta forma evitar posibles sobrecostes al tener que contratar tarifas más altas. En la mayoría de estos servicios, el precio depende del número de peticiones diarias, teniendo en cuenta que existen más de trescientos mil usuarios, un mal uso de las llamadas al servidor podría aumentar mucho el coste. Además, el servidor contratado deberá mostrar datos a nivel global, por lo que se descartan aquellos que proporcionan información solo para un país.

A parte de esta restricción, se cuenta con un número máximo de 300 horas para realizar el proyecto. Lo que supone un tiempo muy justo para todas las tareas que se deben realizar.

1.3.7. Análisis de riesgos

Los riesgos a los que se enfrenta este proyecto se listan a continuación:

- Desarrollar el nuevo módulo con todos los requisitos descritos en las 300 horas que dura la estada en prácticas. La metodología de trabajo empleada, descrita en el siguiente capítulo, ayudará a paliar este riesgo.
- La falta de experiencia del desarrollador, Daniel Querol, en el desarrollo de aplicaciones iOS pautas hacer que el desarrollo vaya lento o se cometan errores. Afortunadamente, el supervisor de prácticas, experto en este tema, orientará el desarrollo y resolverá dudas cuando se produzcan.
- El número de peticiones al servidor deberá reducirse al máximo para evitar sobrecostes en la contratación. Para ello se almacenaran los datos meteorológicos en caché y se reutilizarán si el usuario vuelve a consultarlos en un periodo corto de tiempo.

1.4. Estructura de la memoria

Este documento está estructurado en cinco capítulos. Está orientado a un lector con conocimientos técnicos en el desarrollo de software, por lo que no se explica extensamente todos los conceptos teóricos usados. En los diferentes capítulos se describe el desarrollo del proyecto, desde la especificación de requisitos hasta la fase final de implementación de software y test.

Los capítulos están organizados en secciones en los que se tratan aspectos relacionados con la realización del proyecto y el análisis realizado. En el capítulo 1 se describe la especificación

del proyecto y sus objetivos. En el capítulo 2 se explica la planificación temporal del proyecto, la metodología usada y el seguimiento realizado. En el capítulo 3 se describen las dos investigaciones realizadas para conocer los parámetros meteorológicos más influyentes para la realización de fotografías y la comparación entre los diferentes servidores meteorológicos candidatos a ser contratados. Aunque este capítulo podría formar parte del análisis, no se incluye en este, por ser muy extenso. En el capítulo 4 se describe la fase de análisis realizada antes de la implementación, la cual se describe en el capítulo 5, junto a los test de software realizados. Por último, en el capítulo 6 se exponen las conclusiones obtenidas del desarrollo del proyecto, se evalúa el estado en que se queda el producto y posibles mejoras que se podrían realizar en un futuro.

Destacar que en el documento se usan los términos de servicios meteorológicos y servidores meteorológicos de forma indistinta. En el primer caso se refiere al servicio que proporcionan las empresas o instituciones de este campo, mientras que el segundo se refiere al equipo concreto que proporciona esta información. En muchas ocasiones el uso es indistinto.

Capítulo 2

Planificación del proyecto

En este capítulo se detalla la metodología que se ha usado para la planificación del proyecto y los motivos de su elección. Además, se describe la planificación temporal, la estimación de costes, así como el seguimiento del trabajo realizado.

2.1. Metodología

Para el desarrollo del proyecto se ha elegido la metodología Scrum, con algunos modificaciones. A continuación, se realiza una introducción teórica breve y se exponen los motivos de su elección y los cambios que se han hecho respecto a su definición más estricta.

En la metodología Scrum el trabajo se divide en iteraciones llamadas Sprints. En estas iteraciones se eligen un conjunto de historias, que son requisitos de usuario a realizar definidas previamente en una lista conocida como Product Backlog o pila del producto. Las historias de usuario son descripciones de lo que quiere el cliente, pero desde el punto de vista del rol que las va a usar, es decir, el usuario de la aplicación PhotoPills. Al final de cada Sprint se revisan las historias de usuario y se clasifican como completadas o por hacer. En caso de ser del segundo tipo, pasan al siguiente Sprint donde podrán ser realizadas.

En esta metodología se definen los roles de Product Owner, ScrumMaster y equipo de desarrollo. El Product owner representa al cliente del producto que se va a desarrollar y se encarga de que el equipo trabaje adecuadamente desde la perspectiva del negocio. El Scrum master lidera el equipo para que cumpla la metodología.

Los motivos de su elección son:

- Scrum en una metodología ágil de trabajo que se caracteriza por tener un desarrollo del producto de forma incremental, en la que las diferentes etapas del desarrollo se solapan, en vez de completar una etapa tras otra, como se haría en una metodología tradicional. Esta forma de trabajo permite obtener resultados visibles, aunque el producto no esté completamente terminado. Esto supone un gran punto a favor en este proyecto ya que

uno de los riesgos es la limitación de tiempo disponible y no está asegurada la finalización del producto. Con esta metodología, aunque el trabajo no se terminase completamente, se podría tener un producto funcional.

- Esta metodología está espacialmente indicada para proyectos con requisitos poco definidos o cambiantes, como es el caso de este proyecto.
- La flexibilidad y la productividad caracterizan a esta forma de trabajo y es lo que se busca en este proyecto. Poder disponer de equipos auto-dirigidos y auto-organizados que puedan adaptarse bien al desarrollo.
- La metodología facilita la comunicación continua entre el Product Owner, que en este caso es el supervisor de las prácticas, y el desarrollador, que es el estudiante y autor de este documento.
- Finalmente, el motivo principal de su elección es que la empresa hace uso de esta metodología para el desarrollo de sus proyectos. Desde Photopills S.L. se ha aconsejado el uso de esta metodología.

En el caso de la empresa, se usa una metodología Scrum pero con alguna variación. Las reuniones diarias no se realizan, ya que los Sprints son cortos, de una semana de duración, y se da especial énfasis a las reuniones de planificación y revisión del Sprint realizadas semanalmente. A pesar de no hacer reuniones diarias, hay una comunicación constante vía chat, lo que permite resolver cualquier duda de forma rápida.

Al tratarse de una empresa pequeña, los roles de Product Owner y Scrum master se realizan por la misma persona, Germán Marquès, supervisor la estancia en prácticas. Mientras que el equipo de desarrollo está formado por una única persona, Daniel Querol.

2.2. Planificación

Como se ha explicado en el apartado anterior, la metodología usa iteraciones o sprints para dividir el desarrollo del proyecto. En este caso las iteraciones son de una semana. Al principio de cada iteración se realiza la reunión de planificación del sprint en la cual se elegirán las historias de usuario de la pila del producto y se definen las tareas a realizar en cada una de estas historias. Al final de cada iteración se comprueba en la reunión de revisión del sprint el trabajo realizado y las historias se establecerán como completadas o por completar, en este segundo caso pasan a la siguiente iteración. A continuación, se muestra la pila del producto con las épicas[1] que conformarán el desarrollo del proyecto:

- E1. Dada una localización concreta, el sistema obtendrá y mostrará los datos meteorológicos actuales y la previsión de forma detallada y estructurada.
- E2. El sistema permitirá gestionar las localizaciones (listar, dar de alta y baja).

[1] Historias de usuario muy grandes que deben ser divididas en historias más simples para poder ser desarrolladas de una forma más atómica.

- E3. Dado un mapa de una región, el sistema mostrará información meteorológica sobre el mapa.
- E4. El sistema deberá optimizar el número de consultas al servidor.

Teniendo en cuenta las épicas, en el Cuadro 2.1 se muestra la pila del producto inicial, con las historias de usuario y los requisitos técnicos. Para cada historia o tarea se estima las horas que conllevará su realización y además se prioriza usando una escala del 1 al 5. Donde 5 es una prioridad máxima y 1 es mínima. Para realizar esta estimación se ha tenido en cuenta aquellas historias de usuario cuyos resultados se quiere obtener antes.

Cuadro 2.1: Pila del producto inicial

Épica	Referencia	Descripción	Estimación	Prioridad
E1	HU01	Como usuario quiero ver información del estado meteorológico actual.	32h	4
	HU02	Como usuario quiero ver una cronología con la información meteorológica para las próximas 48 horas.	24h	2
	HU03	Como usuario quiero ver información meteorológica para los próximos 5 días.	24h	2
	HU04	Como usuario quiero obtener información meteorológica de los próximos 5 días en forma de gráfica.	24h	3
	HU05	Como usuario quiero obtener información meteorológica de los próximas 24 horas en forma de gráfica.	24h	3
	HU06	Como usuario quiero poder cambiar las unidades de medida en las que se muestra la información.	8h	4
	HU07	Como usuario quiero ver una tabla con todos los parámetros meteorológicos del pronóstico detallados.	24h	4
	T.1	Investigar los diferentes servidores meteorológicos.	40h	5
E2	HU08	Como usuario quiero poder elegir una localización para poder guardarla y obtener los datos meteorológicos.	16h	2
	HU09	Como usuario quiero disponer de una lista con todas las localizaciones guardadas para poder seleccionar una localización y obtener la información meteorológica.	10h	4
E3	HU010	Como usuario quiero ver información meteorológica sobre un mapa de una región.	32h	5
E4	T.2	El sistema deberá guardar en caché los datos meteorológicos para evitar consultas al servidor.	20h	5

Además de las historias de usuario y tareas mostradas en el Cuadro 2.1, se realiza una fase

de análisis previa con las siguientes tareas:

- Realizar una propuesta técnica del proyecto en la que se incluya información sobre el contexto y los objetivos del proyecto.
- Realizar un diseño inicial de las *interfaces* de usuario.
- Conocer la metodología de trabajo de la empresa.

2.3. Estimación de recursos y costes del proyecto

En este apartado se detalla y calcula el coste directo, indirecto y de contratación que supone el desarrollo del proyecto.

El coste directo incluye los recursos humanos, hardware y software, mientras que el coste indirecto está formado por aspectos relacionados con el alquiler, agua, luz, internet, etc. El coste de contratación está ligado con el impuesto de la seguridad social y otros impuestos, que en este caso se calculan como si fuese un proyecto profesional.

La estimación de los recursos, se ha realizado teniendo en cuenta los recursos humanos, de hardware y software. A continuación, se listan los recursos necesarios para el desarrollo del proyecto:

- Hardware. Para la creación de aplicaciones móviles nativas[2] para iOS es necesario disponer de un ordenador de Apple. Esto se debe a que el entorno de desarrollo Xcode, necesario para programar para esta plataforma, solo está disponible en el sistema operativo Mac OS de los ordenadores Apple. A parte del ordenador no es necesario ningún periférico ni equipo más. La aplicación se ejecuta sobre iPhones e iPads emulados por software, así que no es necesario disponer de estos dispositivos. Por tanto solo es necesario un ordenador Apple.
- Recursos humanos. Este proyecto se ha calculado inicialmente que costará unas 300 horas de realizar y será necesario un programador a tiempo completo.
- Software. Para este desarrollo no será necesario disponer de una nueva licencia de desarrollador de Apple, ya que Photopills, S.L. ya cuenta con una. Se usará esta licencia para publicar la actualización de la aplicación que contendrá la nueva *píldora* de Meteorología. A parte de la licencia, el entorno de desarrollo y los emuladores de dispositivos físicos son gratuitos.

A pesar de que se trata de una aproximación, los cálculos realizados siguen fuentes fiables que se citan a en cada caso.

- Costes directos.

[2] Aplicaciones que están dirigidas a un sistema operativo concreto.

- Hardware. El coste del equipo de desarrollo es de 2.000 €. Se estima que tiene una vida útil de unos 6 años. Teniendo en cuenta las 52 semanas al año, menos 4 semanas aproximadamente de vacaciones, trabajará un total 1.900 horas laborales anuales aproximadamente. Lo que hace un total de 11.400 horas de vida útil que costarán 2000 €. Prorrateando esta cifra a las 300 horas de este proyecto obtenemos que el coste del equipo será de **52 €**.
- Software. En el coste del software se tiene en cuenta la licencia de desarrollador de Apple, que es de 99 dolares. Pasando esta cifra en euros, a fecha actual de realización de este documento, supone 88 €. Teniendo en cuenta que la app Photopills cuenta con 16 *píldoras*, contando la nueva, el coste por *pildora* es de **5,5 €** por píldora.
- Recursos humanos. Según el portal especializado Indeed [9], el salario medio bruto mensual de un programador junior en el año 2019 es de 1.177 €. Teniendo en cuenta una media de 160 horas laborales mensuales, el salario por hora es de 7,3 euros. Por tanto el coste de las 300 horas del proyecto es de **2.206,8 €** aproximadamente.

- Costes indirectos. Estos costes incluyen el alquiler, agua, luz o internet, entre otros. En este caso todos los miembros de la empresa trabajan de forma remota, incluido el desarrollador de este proyecto, por lo que a la empresa no le supone ningún gasto indirecto.
- Costes de contratación. Además del sueldo bruto, la empresa debe pagar la Seguridad Social del trabajador. El coste de este impuesto es de 30,9 % del salario bruto mensual para contratos indefinidos y 32,1 % para contratos eventuales. Este último, en caso de ser a tiempo parcial, subiría un 1 %, aunque no es el caso de este proyecto. Teniendo en cuenta 2.206,8 €de salario bruto y el coste de la Seguridad Social del 32,1 %, el coste de contratación será de **708 €**.

Teniendo en cuenta todos los costes calculados, a continuación se muestra el coste total.

52 € (hardware) + 2.212,3 € (salario bruto)+ 708 € (seguridad social) = 2.972,3 €

2.4. Seguimiento del proyecto

En esta sección se detalla el control y seguimiento del proyecto durante el proceso de desarrollo y como se ha usado la metodología Scrum para definir las tareas a realizar en cada iteración. Además se indican las modificaciones y nuevas historias de usuario que se han añadido posteriormente a la planificación inicial.

Primera Quincena

En las primeras dos semanas de las prácticas no se define una iteración tipo Sprint, como se usa en Scrum. Se debe a que durante este tiempo se realiza la propuesta técnica y aun no se ha decidido qué metodología se usará ni están claros los requisitos del proyecto. Las tareas realizadas en este periodo se listan a continuación.

- Investigar aplicaciones meteorológicas para obtener ideas de los datos que muestran y la forma de mostrarlos, así como obtener ideas para el diseño de las *interfaces* de usuario.
- Conocer la metodología de trabajo de la empresa.
- Investigar que parámetros meteorológicos son de interés para los fotógrafos.
- Escribir la documentación de la propuesta técnica.
- Realizar un diseño inicial de las *interfaces* de usuario.

Debido a que el desarrollador tiene conocimientos previos de meteorología, no es necesario dedicar tiempo al conocimiento de esta materia. En esta etapa se hacen varias videoconferencia y la comunicación por chat es constante, lo que facilita la resolución de dudas.

Sprint 1

A partir de esta iteración se empieza a usar la metodología Scrum. En este *sprint* se lleva a cabo un fase de investigación y análisis para decidir el servidor meteorológico más conveniente para contratar. Para ello se crea una aplicación iOS prototipo en la que se realizan peticiones a los distintos servidores. Además, se comparan los datos meteorológicos de los pronósticos proporcionados por los diferentes servidores con los datos reales de diferentes estaciones meteorológicas para evaluar cual de estos servicios es más preciso a la hora de realizar predicciones. En el Cuadro 2.2 se listan las tareas seleccionadas de la pila del producto al inicio del Sprint.

Cuadro 2.2: Historias de usuario Sprint 1

Historia	Descripción
Tarea	Investigación de los diferentes servicios meteorológicos. Subtareas: 1. Seleccionar los servidores candidatos a investigar. 2. Leer documentación de cada servidor. 3. Crear un prototipo para hacer peticiones de datos a los servidores. 4. Probar peticiones de dados meteorológicos. 5. Estudiar la calidad de las previsiones. 6. Redactar la investigación y conclusiones.

Este sprint finaliza con la tarea completada, aunque la investigación realizada se va modificando y aportando nueva información durante todo el proceso de desarrollo del proyecto. Esta información está disponible en la sección 3.2 de esta memoria. En este periodo aún se nota la falta de experiencia con el entorno de desarrollo y el lenguaje de programación. Los errores son constantes, lo cual impide trabajar con mayor velocidad.

Sprint 2

En este sprint se realizan las primeras historias de usuario. Para ello se reutiliza parte del código usado en el prototipo del Sprint anterior, aunque en este caso los datos meteorológicos se muestran como se ha indicado en los diseños realizados en la primera quincena. Se eligen estas historias por tener mayor prioridad. En el Cuadro 2.3 se muestran la tarea e historia seleccionadas y las subtareas que conforman su realización.

Cuadro 2.3: Historias de usuario Sprint 2

Historia	Descripción
HU01	Como usuario quiero ver información sobre el estado actual de la meteorología. Tareas: 1. Crear pantalla donde mostrar los datos basada en el diseño realizado previamente. 2. Crear las clases y estructura del modelo necesarias para almacenar y gestionar los datos. 3. Obtener los datos meteorológicos del servidor. 4. Mostrar los datos de forma clara y estructurada, como se ha diseñado en las pantallas.
HU02	Como usuario quiero ver una cronología con la información meteorológica para las próximas 24 horas. Tareas: 1. Crear las clases y estructura del modelo necesaria para almacenar y gestionar los datos. 2. Obtener los datos meteorológicos del servidor. 3. Mostrar los datos de forma clara y estructurada en la pantalla creada en la HU01.

El sprint finaliza con las historias de usuario seleccionadas completadas. Se queda pendiente la realización de los test para la historia de usuario HU01, por lo que vuelve a entrar a la pila del producto como Error.

Cuadro 2.4: Historias de usuario añadidas a la pila del producto en el Sprint 2.

Épica	Referencia	Descripción	Estimación	Prioridad
E1	Error. 1	Realización de test para de la HU01: Como usuario quiero ver información del estado meteorológico actual.	8h	1

Sprint 3

El sprint 3 se seleccionan las historias que se muestran en el Cuadro 2.5. Se consigue completar las tres historias de usuario, pero sin la realización de los test de las historias HU03 y HU07, por lo que se añaden a la pila del producto como Error. Además, se queda pendiente la realización de varios cambios en las historias HU05 y HU06 que se harán el siguiente día, pero no pasan a formar parte de siguiente sprint, ya que son modificaciones que requieren poco tiempo.

Cuadro 2.5: Historias de usuario Sprint 3

Historia	Descripción
HU03	Como usuario quiero ver información meteorológica para los próximos 5 días. Tareas: 1. Crear la estructura y clases del modelo para almacenar la información. 2. Obtener los datos del servidor. 3. Crear el componente sobre el que se mostrarán los datos. 4. Mostrar los datos en el componente.
HU05	Como usuario quiero obtener información meteorológica de los próximos 5 días en forma de gráfica. Tareas: 1. Crear el componente sobre el que se mostrará la gráfica. 2. Dibujar la línea de la gráfica y las etiquetas. 3. Adaptar la gráfica para los diferentes parámetros meteorológicos.
HU07	Como usuario quiero ver una tabla con todos los parámetros meteorológicos del pronóstico detallados. Tareas: 1. Crear un componente sobre el que se mostrará la tabla. 2. Crear una tabla que se pueda desplazar horizontalmente. 3. Mostrar el encabezado de cada fila y columna y el contenido de la tabla.
HU06	Como usuario quiero poder cambiar las unidades de medida en las que se muestra la información. Tareas: 1. Crear la pantalla para mostrar las opciones de configuración. 2. Programar las funciones para hacer cambios entre unidades. 3. Mostrar el botón para ir a la pantalla de configuración desde todas las pantallas principales de la app.

Además, se añaden a la pila del producto las historias que se muestran en el Cuadro 2.6.

Cuadro 2.6: Historias de usuario añadidas a la pila del producto en el Sprint 3.

Épica	Referencia	Descripción	Estimación	Prioridad
E1	Error. 2	Realización de test historia de usuario HU06: Como usuario quiero poder cambiar las unidades de medida en las que se muestra la información.	8h	1
	Error. 3	Realización de test para de la HU03: Como usuario quiero ver información meteorológica para los próximos 5 días.	8h	1
	Error. 4	Realización de test para de la HU07: Como usuario quiero ver una tabla con todos los parámetros meteorológicos del pronóstico detallados.	8h	1
E6	HU011	Como usuario quiero disponer de la app en el idioma por defecto de mi smartphone.	20h	1

Sprint 4

En este sprint se elige una historia de usuario y una tarea que se muestran en el Cuadro 2.7, además se realiza un test que quedaba pendiente de una historia anterior.

Cuadro 2.7: Historias de usuario y tareas elegidos para el Sprint 4

Historia	Descripción
Error. 2	Realización de test historia de usuario HU06
HU08	Como usuario quiero poder elegir una localización para poder guardarla y obtener los datos meteorológicos. Tareas: 1. Añadir la opción de localización en la pantalla de configuración. 2. Crear la pantalla para mostrar las localizaciones en forma de lista. 3. Crear la pantalla para añadir nueva localización. 4. Crear la estructura de almacenaje de los datos de las localizaciones (nombre, latitud, longitud) 5. Crear la opción de borrado de una localización de la lista. 6. Crear la opción de actualización de una localización de la lista.
T.2	El sistema deberá guardar en caché los datos meteorológicos para evitar consultas al servidor.

Este sprint termina con la realización de la historia de usuario y la tarea y la solución del error, pero se deja los test por realizar, que pasarán a los próximos sprints. Además, se añade a la pila del producto la historia de usuario que se muestra en el Cuadro 2.8.

Cuadro 2.8: Historias de usuario añadidas a la pila del producto.

Épica	Referencia	Descripción	Estimación	Prioridad
E7	HU012	Como usuario quiero obtener los datos meteorológicos (actuales, gráficas, tabla) para la localización actual.	8h	1

Sprint 5

En el Sprint 5 se eligen las historias de usuarios que se muestran en el Cuadro 2.9 y se consigue realizarlas todas, menos la historia HU11 por falta de tiempo. Además, se ha dedicado tiempo a refactorizar parte del código para mejorar el rendimiento de la aplicación y se modifican algunos aspectos del diseño de las *interfaces* para mejorar la apariencia.

Cuadro 2.9: Historias de usuario y tareas elegidos para el Sprint 5

Historia	Descripción
HU010	Como usuario quiero obtener información sobre la meteorología sobre un mapa. Tareas: 1. Crear las clases que gestionarán las llamadas a las APIs de Mapas. 2. Crear una pantalla para mostrar el mapa. 3. Crear un menú para listar los mapas que se pueden mostrar. 4. Mostrar los datos meteorológicos sobre el mapa.
HU011	Como usuario quiero disponer de la app en el idioma por defecto de mi smartphone. Tareas: 1. Adaptar todos los textos de la aplicación para que se seleccione el idioma. 2. Crear los ficheros con las traducciones en cada idioma.
HU012	Como usuario quiero obtener los datos meteorológicos (actuales, gráficas, tabla) para la localización actual. Tareas: 1. Obtener la localización actual al iniciar la aplicación. 2. Guardar la localización actual como predeterminada.
Error. 1	Realización de test historia de usuario HU08: Como usuario quiero poder cambiar las unidades de medida en las que se muestra la información.
Error. 2	Realizar test HU 1. Obtención de los datos en formato JSON desde el servidor.

Resultado de las iteraciones

Como se ha descrito anteriormente, se realizan la gran mayoría de historias de usuario propuestas en un inicio. Además, se añaden a la pila del producto historias de usuario y errores, que en todos los casos pertenecen a historias a las que les falta realizar la fase de test. En el Cuadro 2.10 se muestra como ha quedado la pila del producto, con las historias y errores que se quedan sin completar. La historia HU04 que se muestra en el mismo cuadro, se rechazó al principio del desarrollo, ya que no aporta ninguna funcionalidad destacable para el usuario final.

Cuadro 2.10: Pila del producto final.

Épica	Referencia	Descripción	Estimación	Prioridad
E1	HU04	Como usuario quiero obtener información meteorológica de los próximos 5 días en forma de gráfica.	24h	3
E7	HU011	Como usuario quiero disponer de la app en el idioma por defecto de mi smartphone.	20h	1
E1	Error. 3	Realización de test para de la HU03: Como usuario quiero ver información meteorológica para los próximos 5 días.	8h	1
	Error. 4	Realización de test para de la HU07: Como usuario quiero ver una tabla con todos los parámetros meteorológicos del pronóstico detallados.	8h	1

Capítulo 3

Investigación

En esta sección se detalla la investigación realizada en este proyecto y los resultados obtenidos. Por una parte, se presenta un estudio realizado para conocer los parámetros meteorológicos de mayor interés para los fotógrafos. Por otra parte, se muestra una selección de servicios meteorológicos que pueden proporcionar los parámetros meteorológicos a nivel global y se muestra un estudio realizado para conocer la precisión con la que cada servicio meteorológico realiza sus pronósticos del tiempo y una comparación basada en sus características y precio.

3.1. Parámetros meteorológicos

En la aplicación se pretenden mostrar y representar los parámetros que puedan ser de utilidad para la planificación de fotografías al aire libre. Dependiendo de que foto quiera capturar, el fotógrafo buscará unas situaciones meteorológicas concretas. Por ejemplo, en el campo de la astro-fotografía[1] se buscan noches de cielos rasos, con baja humedad, lo que permite captar el cielo en todo su esplendor.

En las tiendas de aplicaciones, como App Store o Google Play, existen numerosas aplicaciones del campo de la meteorología, que nos permiten ver el estado actual del tiempo y el pronóstico para los próximos días. Para la *píldora* de Meteorología se quiere mostrar, a parte de los datos meteorológicos más comunes, parámetros de interés para los fotógrafos, como podría se la altura de las nubes o la visibilidad en kilómetros.

Para conocer los parámetros de mayor interés se ha realizado un estudio. Por una parte, se ha preguntado via mail a fotógrafos profesionales en que se fijan ellos a la hora de hacer fotos al aire libre y qué aplicaciones o webs usan para conocer la meteorología. Entre estos fotógrafos está Marco Grassi, Albert Dros, Francesco Gola, Josh Cripps y Nico Trinkhaus. Por otra parte, se ha discutido con los miembros de la empresa Photopills, que tienen amplios conocimientos de fotografía, que parámetros podían ser más interesantes y se ha investigado sobre los efectos meteorológicos que produce cada parámetro. Finalmente, se ha obtenido más información acerca

[1]Estilo fotográfico que consiste en fotografiar en cielo nocturno con el objetivo de captar objetos astronómicos, como estrellas, planetas o la Vía Láctea.

de estos parámetros meteorológicos en el libro *Manual de Meteorología* [6].

A continuación, se muestran una lista con los parámetros que se quiere obtener y los efectos que pueden producir, indicando porque se han seleccionado.

- Temperatura: la temperatura es uno de los parámetros principales que determinan el estado de la atmósfera. Produce fenómenos meteorológicos que pueden ser de interés para los fotógrafos. Se quiere conocer la evolución de las temperatura a lo largo de un periodo de tiempo, desglosado en horas, como podría ser las próximas 24 horas, así como la máxima y mínimas de temperatura, para los próximos días.
- Presión: se conoce como la fuerza que ejerce el aire sobre la superficie. A mayor altura sobre el nivel del mar, menor presión. No tiene una relación directa con la fotografía pero sí con las condiciones de buen o mal tiempo. La presión baja está relacionada con nubes y humedad, mientras que la presión alta suele producir cielos despejados. Puede tenerse en cuenta para fotografía de paisaje en altas montañas, ya que cuando hay altas presiones, la inversión térmica produce el "mar de nubes"que se puede observar desde las cimas. Se quiere conocer la presión atmosférica y su evolución.
- Humedad: la humedad relativa en aire puede ser de gran interés para la fotografía, ya que produce fenómenos meteorológicos como la niebla o el roció. Tiene una relación directa en la visibilidad y el punto de rocío. En la astro-fotografía se evitan las noches de humedad alta, ya que impide ver el firmamento en todo su esplendor. Se quiere conocer la humedad relativa en el aire y su evolución.
- Viento: en relación con este fenómeno se quieren obtener la velocidad y la dirección. Estos parámetros pueden ser de gran interés para las fotografías de larga exposición, no solo para saber si será posible realizar este tipo de fotos sin que el viento mueva el trípode, sino también para saber como puede quedar la composición de nubes en el resultado final. Dependiendo de lo que quiera capturar el fotógrafo, puede buscar situaciones de calma o viento en altura, entre otros. El viento también tiene efecto en la visibilidad, ya que una situación de viento disminuye la cantidad de partículas líquidas o sólidas, como polvo, en la atmósfera, haciendo que este aumente considerablemente. Se quiere conocer la velocidad y la dirección del viento, así como la velocidad de las rachas de viento.
- Nubes: uno de los parámetros más importantes, de este depende completamente el resultado final de cualquier fotografía en exterior, ya sea por su composición en la foto, que podría resultar un paisaje nublado o un cielo completamente despejado, como sus efectos en la luz y las sombras. De la nubosidad se quiere conocer el porcentaje de nubes en el cielo, el tipo de nube, ya sea más opaca o más ligera, y también la altura de la nube, desglosada en nubes bajas, medias y altas respecto de la superficie terrestre. La altura de la base de las nubes se puede estimar a partir de la temperatura de la superficie y el punto de rocío, pero se prefiere que esta información se proporcione directamente por el servició meteorológico.
- Precipitaciones: dentro de las precipitaciones se puede diferenciar diferentes parámetros. La presencia de algún tipo de precipitación, ya sea lluvia, nieve o granizo pueden impedir realizar fotos en exterior, aunque en algunos estilos fotográficos se buscan estas situaciones. Se quiere conocer el tipo de precipitación, la intensidad, la probabilidad de producirse y su evolución durante el tiempo. Además, de la nieve se quiere conocer la cantidad acumulada.

- Punto de rocío: permite conocer la temperatura en la que se empieza a condensar el vapor de agua presente en el aire. Es de interés, ya que permitirá saber si se producirá el roció sobre el suelo, la escarcha o la niebla baja que normalmente se forma en los valles por las mañanas después de una noche fría con cielos despejados y situaciones de viento en calma. Este parámetro puede ser calculado a partir de la temperatura y la humedad.
- Visibilidad: este parámetro es de gran interés para los fotógrafos de paisaje, ya que una gran visibilidad o una visibilidad a muchos kilómetros hará que aparezcan de forma nítida los objetos que estén a mucha distancia en el horizonte. La visibilidad depende de las partículas sólidas, cómo podría ser el polvo o la presencia de partículas contaminantes, así como partículas líquidas, como la humedad en el ambiente. Se quiere conocer la distancia de visibilidad y su evolución.

Sabiendo los parámetros meteorológicos que se quieren obtener, se han seleccionado aquellos servicios meteorológicos que proporcionan esta información.

3.2. Comparación de servicios meteorológicos

La realización de una aplicación para smartphones de meteorología requiere una fuente de datos de donde poder obtener la información que se mostrará al usuario. Estos datos son generados por meteorólogos a partir de modelos meteorológicos, cómo podría ser el modelo GFS (*Global Forecast System*) que proporciona la agencia americana NOAA[2]. De esta forma, se entiende que la aplicación se encargará únicamente de mostrar de una forma clara los datos obtenidos de los servidores meteorológicos y no de calcular pronósticos del tiempo.

En este apartado se describen, analizan y comparan los principales servicios meteorológicos que proporcionan datos a nivel global con el fin de conocer el mejor candidato para ser contratado. Para realizar el análisis y la comparación se han tenido en cuenta los diferentes factores, como los datos meteorológicos que proporcionan y su precisión, así como el precio y eficiencia de las peticiones de datos. Para analizar la precisión de los diferentes servicios se ha obtenido la previsión meteorológica para una localización donde existe una estación meteorológica y se han comparado los datos de la previsión con los datos reales que va captando la estación a lo largo del periodo de previsión.

Los servicios meteorológicos seleccionados para realizar el estudio deben cumplir de entrada un requisito: proporcionar datos meteorológicos de cualquier lugar del mundo. El motivo de este requisito se debe a que la aplicación Photopills tiene usuarios de todo el mundo, y no solo eso, sino que es usada por fotógrafos muy exigentes que viajan y exploran cualquier parte del planeta. Otros aspectos que se han tenido en cuenta para seleccionar los servicios meteorológicos son los datos que proporcionan, listados en la sección 3.1, entre los cuales se puede encontrar la temperatura, presión atmosférica, viento, humedad, etc.

Todos los servicios proporcionan los datos más básicos, que son la temperatura, velocidad del viento, dirección del viento, humedad relativa, presión atmosférica e información sobre precipi-

[2]Nacional Oceanic and Atmospheric Administration, (NOAA). Agencia nacional americana centrada en el estudio de las condiciones de los océanos y la atmósfera.

taciones. En cambio no todos proporcionan parámetros sobre la probabilidad de precipitaciones, niebla o tormenta, el punto de rocío, la visibilidad, el porcentaje de nubes de cubren el cielo, la velocidad o dirección de las ráfagas de viento. Algunos de estos parámetros, como el punto de rocío pueden ser calculados a partir de parámetros más básicos, pero otros solo podrán obtenerse si el servicio meteorológico los proporciona ya que son mucho más complejos de calcular.

3.2.1. Servidores Meteorológicos

En esta sección se describen los diferentes servidores meteorológicos que se han seleccionado como posibles candidatos para proporcionar la información meteorológica a la *píldora* de Meteorología.

Open Weather Map

Open Weather Map ofrece una API a la que podemos hacer diferentes tipos de llamadas:

- Obtener los datos meteorológicos actuales. En la documentación [3] indican que esta información se obtiene a partir de modelos y datos de más de cuarenta mil estaciones meteorológicas. Al trabajar con un modelo matemático pueden proporcionar datos de cualquier localización, incluso de aquellas en las que no tengan cerca una de las estaciones meteorológicas. Esta petición carece de algunos parámetros que son de interés para la aplicación, como son:
  - Nubes: no se indica la altura de las nubes, ni se desglosa en altas, medias y bajas. Tampoco indica ningun código descriptivo sobre el tipo[3] de nube.
  - Precipitaciones: no se indica la intensidad ni la probabilidad de precipitaciones.
  - Visibilidad: la respuesta con formato JSON no proporciona datos sobre este parámetros. Sorprendentemente si lo hace la respuesta a la misma petición, pero con formato XML.
  - Temperatura aparente: carece de este parámetro, aunque puede ser calculado fácilmente.
  - Punto de rocío: carece de este parámetro, al igual que la temperatura aparente, también puede ser calculado.
- Obtener el pronóstico meteorológico para los próximos cinco días con intervalos de tres horas o los próximos cuatro días con intervalos de una hora. Los parámetros devueltos son casi los mismos que en la petición de estado actual del tiempo. En este caso no se añade ningún parámetro nuevo y no se dispone de parámetros como la visibilidad, tampoco en formato XML, ni información de la precipitación en la última hora.
- Obtener la previsión de los próximos 16 días. En esta caso la respuesta proporciona datos resumidos para cada día. Se dispone de parámetros como la temperatura máxima y

[3]Existe una clasificación de diferentes tipos de nubes según su altura, forma o fenómeno meteorológico que las forma. Por ejemplo cirros, altoestratos, estratocúmulos, etc.

mínima, así como la temperatura que hará por la noche, por la mañana o por la tarde, la cantidad acumulada de lluvia o nieve.

- Obtener información sobre un mapa. Disponen de diferentes tipos de capas que se indican en el Cuadro 3.1.

Cuadro 3.1: Capas disponibles para aplicar sobre un mapa.

Capas
Precipitación acumulada
Intensidad de la precipitación
Volumen de nieve
Velocidad del viento a 10 m de altura
Velocidad del viento en color para indicar la intensidad y flechas para indicar la dirección
Presión atmosférica
Temperatura del aire a 2 metros de altura
Punto de rocío
Humedad relativa
Nubosidad

Algunas de las características que destacan de forma positiva de este servicio se listan a continuación.

- Se puede obtener información meteorológica de varias localizaciones en una misma llamada.
- Se puede obtener información meteorológica de varias localizaciones alrededor de una coordenada. Indicando unas coordenadas concretas, la respuesta devuelve información de las ciudades que están a una distancia que podemos configurar. La distancia puede ser circular a un punto dada por unas coordenadas o se pueden especificar cuatro coordenadas para delimitar una superficie rectangular.
- Dispone de una petición que devuelve un fichero con un gran volumen de datos del estado actual del tiempo y del pronóstico de muchas localizaciones en un mismo fichero. Estos ficheros pueden usarse para hacer peticiones directamente a estos datos y ahorrar llamadas al servidor. Esto podría ser interesante para aplicaciones con muchos usuarios, ya que de esta forma podríamos disponer de los datos en un servidor propio y realizar las llamadas a este servidor directamente y no al servidor de Open Weather Map.
- Los mapas disponen de capas del estado actual y del pronóstico y se pueden aplicar sobre Google Maps.

A continuación, se muestran los puntos negativos de este servicio. No se tienen en cuenta los parámetros de los que carece, ya que han sido indicados anteriormente.

- La información que se devuelve varía dependiendo del formato de la respuesta. El XML lleva más datos que el JSON.

- Parámetros de respuesta muy básicos, no destaca por ningún dato concreto. La información de la precipitación es muy escasa, ya que no se indica la probabilidad, la intensidad ni el volumen total.

Aeris

La documentación de Aeris [1] indica que este servicio proporciona una API a la que se pueden hacer las siguientes peticiones de datos:

- Obtener el estado actual del tiempo. La información se obtiene a partir de la estación meteorológica más cercana. Esto supone que en el caso de que se disponga de una estación a poca distancia, los datos serán muy fiables, mientras que en el caso de que la estación más cercana esté lejos no será así, ya que por ejemplo, la diferencia de altura puede alterar mucho los valores de los diferentes parámetros meteorológicos. Además, en el caso de que no se disponga de una estación en la zona, la respuesta a la petición no devuelve datos, sino un mensaje de error. A pesar de esto, se pueden obtener los datos del estado actual del tiempo mediante la petición del pronóstico, ya que devuelve la información desglosada por horas, desde la hora actual en adelante. La respuesta a esta petición dispone de todos los parámetros básicos y se añade alguno listado a continuación.
  - Nubes: dispone del porcentaje de nubes sobre el cielo y un código que indica el tipo de nube. También dispone de la altura de la base de las nubes.
  - Luz: porcentaje de luz estimado.
  - Nieve: disponen de información sobre la cantidad acumulada de nieve.
- Obtener el pronóstico del tiempo. Se dispone de información diaria o en intervalos configurables, por ejemplo de 1h, 3h o 24h. La información está disponible en un único tipo de llamada, es decir, es la misma petición la que obtiene una información del día completo o de una hora concreta, dependiendo de cómo se configure la petición se obtienen los datos desglosados por periodos. En este tipo de petición no se informa sobre la altura de las nubes, aunque eestacan algunos otros parámetros:
  - Precipitaciones: dispone de información de la probabilidad de precipitaciones.
  - Hielo: proporciona información sobre la cantidad de hielo acumulado.
  - Nieve: proporciona información sobre la cantidad de nieve acumulada.
  - Viento: dispone de información sobre la dirección mínima y máxima del viento, así como máximas de velocidad. También proporciona información sobre el viento a 80m de altura.
- Obtener información meteorológica sobre un mapa. Proporciona múltiples capas que se pueden aplicar sobre el Google Maps. En el Cuadro 3.2 se muestran las distintas capas que proporciona este servidor.

De este servicio destacan algunas características que se resumen a continuación.

Cuadro 3.2: Capas disponibles para aplicar sobre un mapa.

Capas
Radar de precipitaciones. Dispone del estado actual pero no del pronóstico.
Imágenes infrarrojas tomadas por satélite
Pronóstico de las nubes
Temperatura
Imágenes del satélite
Velocidad del viento
Flechas que indican la dirección del viento
Rayos en los últimos 15 minutos

- Las respuestas de peticiones disponen de todos los parámetros básicos, sin tener en cuenta la altura de las nubes, y añaden parámetros que pueden ser interesantes, como el código del tipo de nube o la probabilidad de precipitación.
- Dispone de muchos tipos de mapas, entre los que destacan el mapa de los rayos caídos en los últimos 15 minutos o el mapa con las flechas de la dirección del viento.

Algunas de los puntos negativos de este servicio son:

- Obtener el estado actual puede requerir varias peticiones. Debido a que si no se dispone de una estación cerca, se deberá hacer una llamada de tipo previsión.
- Una gran cantidad de los parámetros pueden devolver parámetros nulos.
- Tiene muchas datos que no están disponibles a nivel mundial, solamente Estados Unidos. Como información sobre las mareas o capas de pronóstico de precipitaciones.

World Weather Online

La API de este servicio proporciona información sobre el estado actual y el pronóstico según indica su documentación [5]. Aunque las peticiones se pueden realizar de forma separada, es posible configurar la petición para obtener el estado actual del tiempo y el pronóstico en una única respuesta, ahorrando así llamadas a la API. Este servició proporciona los parámetros básicos y además añade algunos parámetros que pueden ser de interés como la probabilidad que se produzca niebla, nieve o se produzca una tormenta eléctrica. A pesar de ello, no proporciona información sobre la altura de las nubes.

Algunos puntos destacables de esta API son:

- Dispone de información sobre la probabilidad de que ocurran determinados fenómenos. Como la probabilidad de niebla, nieve, lluvia o tormenta eléctrica.

- La peticiones son muy configurables. Se pueden solicitar los datos del estado actual del tiempo, pronóstico para los próximos días y pronóstico para las próximas horas en una misma petición.
- Dispone de una API, independiente de la API meteorológica, especializada en meteorología en la montaña, con información sobre el pronóstico desglosado en diferentes alturas, cota de nieve o nieve acumulada, y otra API con información sobre el estado de la mar, altura de las olas, mareas y swell[4].

Mientras que algunos puntos negativos que tiene este servicio son:

- No indica el tipo de precipitación ni la nieve acumulada en la API de meteorología.
- En caso de pedir la información meteorológica de unas coordenadas en el mar, no proporciona datos y devuelve un JSON con un error. Se debe a que estos datos están disponibles en su API de información marina, que no tiene servicio de prueba.
- No dispone de capas de información para aplicar sobre un mapa tipo Google Maps.
- Los datos de las respuesta JSON viene como cadena, lo que hace que sea un poco más costosa la carga de los datos.

Weatherbit

Este servicio proporciona información sobre el estado actual del tiempo y el pronóstico. Según indica su documentación [4], la API dispone de las peticiones descritas a continuación.

- Obtener el tiempo actual. Se basa en más de cuarenta y cinto mil estaciones meteorológicas que realizan observaciones. Según indican en la documentación, las peticiones devuelven la información más actualizada de la estación más cercana. No indica que sucede en el caso de no disponer de una estación relativamente cerca. Aunque en las numerosas pruebas realizadas para diferentes localizaciones siempre se ha obtenido datos del lugar.
- Obtener el pronóstico para los próximos cinco días en intervalos de tres horas. No dispone de periodos de tiempo más cortos. La respuesta solo proporciona los datos de cada de los intervalos y no dispone de un resumen diario.
- Obtener el pronóstico para las próximas 120 horas. Los intervalos en este caso son de una hora. No proporciona información resumida del día completo.

Los aspectos destacables de este servicio son la disponibilidad de información sobre la altura de las nubes, desglosada en bajas, medias y altas, y la información sobre las precipitaciones, detallando la cantidad de lluvia y/o nieve y la probabilidad de que se produzcan estas precipitaciones.

Mientras que el punto negativo que podemos encontrar son la no disponibilidad de capas para los mapas de información.

[4]Oleaje producido en alta mar por una borrasca y que se acerca a la costa.

Dark Sky

Este servicio proporciona una API que dispone de una sola llamada para realizar la petición de datos, según se indica en su documentación [2]. Estos datos se calculan a partir de modelos que obtiene de la Agencia Americana de Meteorología (NOAA).

- Obtener el pronóstico. Este tipo de llamada proporciona en un solo JSON los datos del estado actual del tiempo, calculado mediante un modelo, información sobre las precipitaciones que se producirán en los próximos minutos, pronóstico para las próximas horas y pronóstico para los próximos días. Dispone de todos los parámetros básicos menos la altura de las nubes. Además, añade información sobre la distancia a la tormenta más cercana.

Destacar de este servicio el ahorro de peticiones al servidor, ya que se obtienen todos los datos en una sola petición. Mientras que los mayores puntos negativo es que no dispone de mapas ni información sobre la altura de las nubes y además los pronósticos por horas solo contienen datos de las próximas.

3.2.2. Comparativa

Parámetros disponibles

En este apartado se pretende mostrar de una forma clara una comparativa de los parámetros que proporciona cada servidor nombrado previamente. En el Cuadro 3.3 se muestra qué parámetros están disponibles en cada servidor.

Cuadro 3.3: Parámetros disponibles en cada API.

	Open Weather Map	Aeis	World Weather Online	Weatherbit	Dark Sky
Temperatura	X	X	X	X	X
Temperatura aparente		X		X	X
Temperatura max y min	X	X	X	X	X
Punto de Rocío		X	X	X	X
Viento. Velocidad y Dir	X	X	X	X	X
Viento 80m altura		X			
Viento. ráfagas		X	X	X	X
Presión	X	X	X	X	X
Humedad	X	X	X	X	X
Visibilidad		X	X	X	X
Nubes. Porcentaje	X	X	X	X	X
Nubes. Descripción		X			
Nubes. Baja, media, alta				X	
Nubes. Altura base		X			
Precipitación. Total	X		X	X	
Precipitación. Probabilidad	X		X	X	X
Precipitación. Tipo	X			X	X
Precipitación. Intensidad					X
Nieve. Acumulada			X		
Niebla. Probabilidad			X		
Radiación solar				X	
Info. marina			X		
Mapas. Capas	X	X			
Mapas. Imagen			X		

Evaluación pronósticos

En esta sección se analiza la calidad de los pronósticos de los diferentes servicios descritos anteriormente. Para realizar las pruebas se han recogido datos de la previsión para los cinco días siguientes a la fecha de petición del pronóstico, con intervalos de tres horas entre datos y se han comparado con los datos de una estación meteorológica situada justo en la misma localización en la que se ha pedido la previsión.

Para realizar este estudio se ha realizado esta operación en cuatro localizaciones diferentes, que coinciden con las regiones donde la aplicación Photopills tiene más usuarios, estas localizaciones son Madrid (España), Portland (Estados Unidos), Auckland (Nueva Zelanda) y Windhoek (Namibia). Además, estas localizaciones en el momento de la realización del estudio están en inverno en el hemisferio norte y verano en el hemisferio sur, esto es destacable ya que en verano los pronósticos son más fáciles de realizar debido a que predominan las altas presiones y la estabilidad atmosférica.

Debido a la falta de parámetros, se descartan los servidores de World Weather Online y Dark Sky, por lo que finalmente se compara la previsión de Open Weather, Weatherbit y Aeris.

En el estudio se recogen datos de la temperatura, velocidad y dirección del viento y presión. Hubiese sido interesante poder recoger más parámetros, como el porcentaje de nubes, la humedad o la precipitación, pero el volumen de datos aumentaría mucho. Además, la transferencia de los datos reales de las diferentes estaciones meteorológicas a las hojas de cálculo donde se comparan con los pronósticos se realiza de forma manual, lo que hubiese requerido mucho tiempo. Al final de este estudio se comparan un total de 3.200valores de cuatro parámetros diferentes de las 4 localizaciones de los 4 servicios meteorológicos más los datos reales obtenidos de cada estación. A las Figuras 3.1 a 3.4 se muestran las gráficas resultantes para cada parámetro.

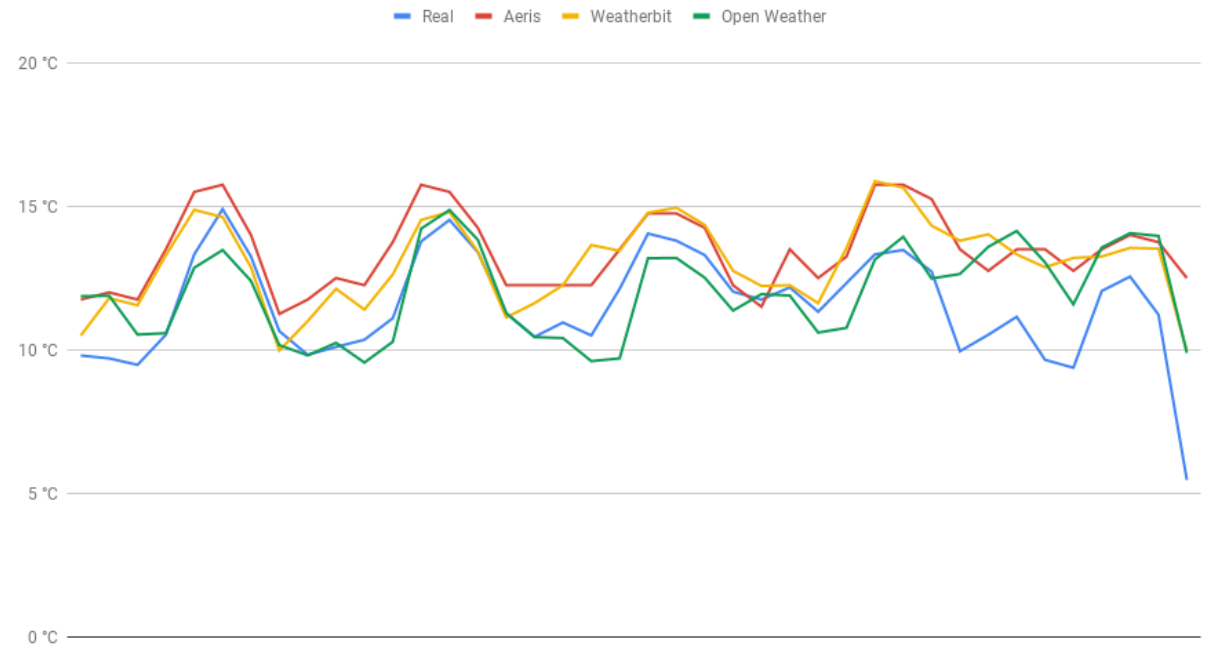

Figura 3.1: Gráfico de líneas con la evolución de las temperaturas

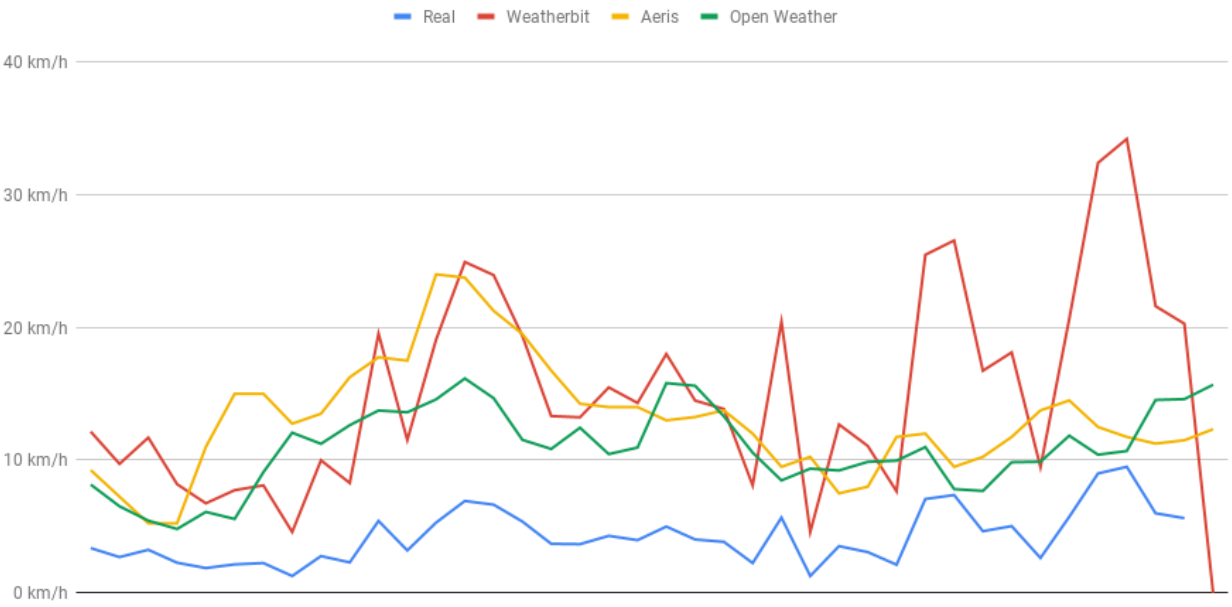

Figura 3.2: Gráfico de líneas de velocidad del viento en kilómetros por hora.

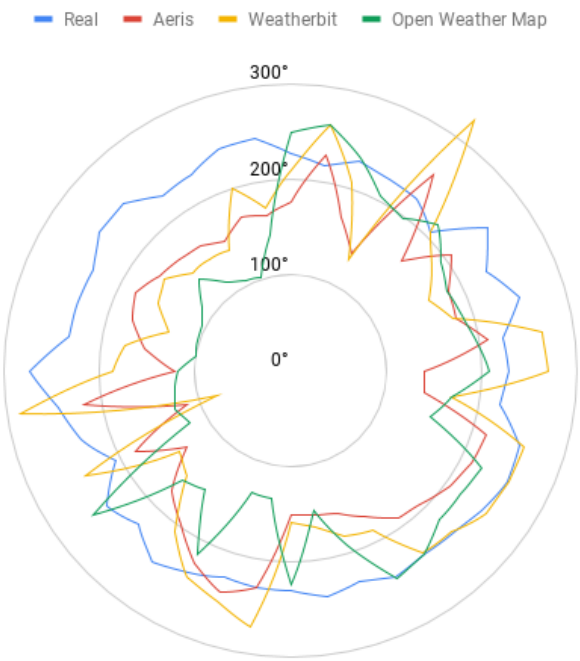

Figura 3.3: Gráfico radial con la evolución de la dirección del viento.

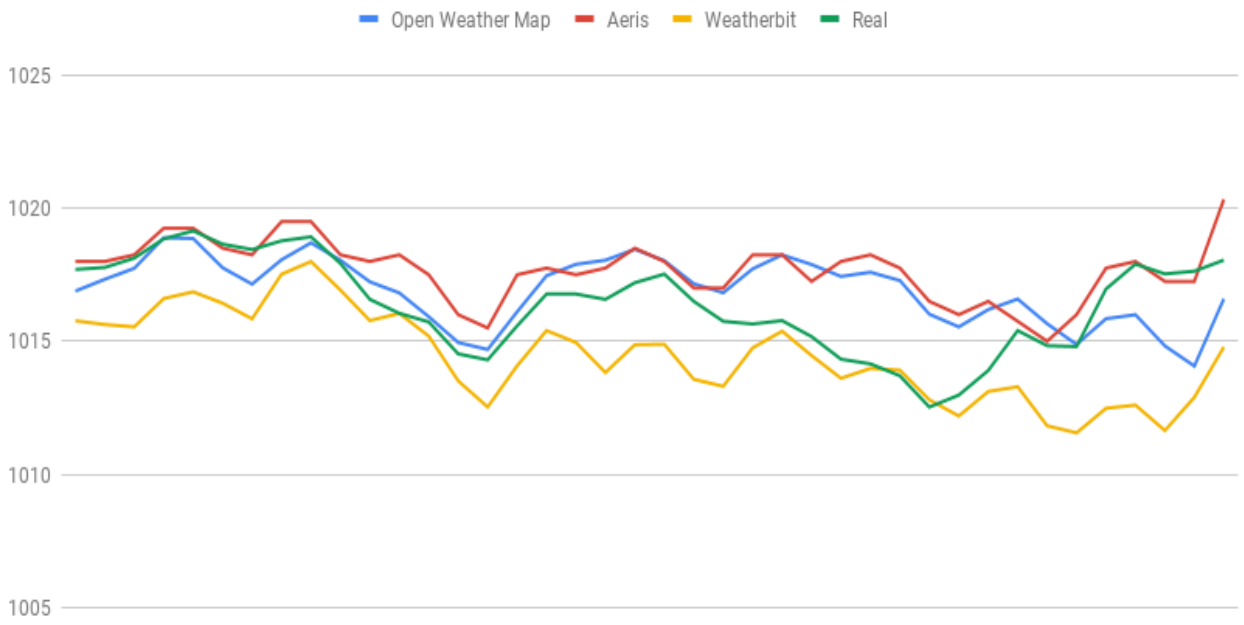

Figura 3.4: Gráfico de líneas con la evolución de la presión atmosférica.

Las gráficas mostradas proporcionan una idea de lo ajustadas que son las previsiones de cada servicio en la localización analizada. A pesar de ello, para una localización diferente podrían obtenerse resultados distintos. Un servicio que en esta localización da resultados muy ajustados

a la realidad podría no hacerlo en otra localización del planeta. Por lo tanto, aunque este estudio nos puede dar una idea general de cómo de precisos son cada servicio se debería realizar un estudio con muchas más localizaciones y parámetros para conocer realmente cual es el mejor haciendo pronósticos.

Cuadro 3.4: Desviaciones típicas de los diferentes parámetros comparados con los datos reales.

Parámetro	Aeris	Weatherbit	Open Weather Map
Temperatura (C)	1,330833888	1,097341336	0,8228189134
Velocidad del viento (km/h)	7,833957461	6,251851366	4,622387247
Dirección del viento (Grados)	41,5955564	38,72440824	47,38699267
Presión (hPa)	0,9484069703	1,380929457	0,9466244719

En el Cuadro 3.4 se muestran las desviaciones típicas de los diferentes parámetros comparados con el valor real obtenido de la estación meteorológica. La dirección del viento varía mucho respecto de los datos reales en todos los servidores debido a que es compleja de predecir.

Precios

En esta sección se analizan los precios de cada servicio teniendo en cuenta el número de peticiones necesarias para obtener los datos de tiempo actual y previsión. En el Cuadro 3.5 se desglosan las diferentes tarifas y precios de cada servidor.

Conclusiones del estudio

En este apartado se comparan los diferentes puntos fuertes y débiles de cada servicio.

Por un lado Weatherbit, que dispone de los pronósticos que se quieren para la *píldora* de Meteorología y es el único servicio que cumple con todos los parámetros que se quieren obtener, destacando la información sobre la altura de las nubes. Por contra, no dispone de mapas meteorológicos. Se podría paliar esta carencia contratando una API para los mapas y otra para los datos meteorológicos tipo texto. Esta solución tiene varios puntos negativos, por una parte encarece el precio final, ya que estamos contratando dos servicios en vez de uno, y por otro parte los datos tipo texto podría divergir de los datos mostrados sobre el mapa. Aunque este segundo punto no es tan negativo como podría parecer, ya que los datos que se muestran sobre un mapa son colores que indican la temperaturas o capas de nubes y no valores tipo texto, y como se puede ver, los diferentes servicios proporcionan datos bastante parecidos.

El servicio Aeris, que dispone de muchos de los parámetros básicos que se quieren obtener y además añade otros parámetros interesantes cómo información sobre el viento a 80m de altura o información sobre la nieve acumulada. Un punto fuerte de este servicio es la cantidad de información que se obtiene en las peticiones de pronósticos desglosados por horas, ya que devuelve 16 días desglosados en información de 24 horas. Aunque esta cantidad de datos hace

Cuadro 3.5: Tarifas y precios de cada servidor meteorológico.

Servicio	Peticiones Necesarias	Precio (Euros)	Servicio	Comentario
Open Weather Map	2	39,69 158,75	600 peticiones/min 3.000 peticiones/min	Incluye mapas básicos Incluye mapas avanzados
Aeris	2 o 3	20,29 70,58 123,52	- 100 peticiones/min max. 5000 diarias - 500 peticiones/min max. 50.000 diarias - 1.000 peticiones/min max. 100.000 diarias.	No incluye mapas. Tienen un precio especial por separado que empieza en los 39,7 euros.
World Weather Online	1	4,41 13,23	- 2.000 peticiones/día	Tienen una tarifa profesional en la que no indican el precio.
Weatherbit	2	30,88 149,1	- 50.000 peticiones/día - 500.000 peticiones/día	
DarkSky	1	0,000088	por petición	Las 1.000 peticiones/día gratis

que las respuestas a las peticiones sean un poco más lentas, permite que se puedan calcular los valores globales del día (temperatura máxima y mínima, precipitación acumulada, etc) a partir de estos datos, pudiendo ahorrar una llamada al servidor para obtener los datos resumidos por días, liberando así el servidor de llamadas y por lo tanto ahorrando dinero. Esto no se puede hacer con Weatherbit, ya que el pronóstico por horas devuelve datos para las 120 horas siguientes, y con estos datos no se puede calcular el resumen diario para los próximos 7 días. Tampoco se podría calcular en Open Weather Map por el mismo motivo. Además de ello dispone de mapas meteorológicos muy variados y con mucha nitidez a la hora de mostrar las zonas de información.

Por último, el servicio Open Weather, que aunque no dispone de todos los parámetros básicos, es uno de los que mejor pronósticos ha realizado en el estudio y dispone de mapas meteorológicos. A pesar de ello los mapas tienen muy mala calidad, ya que las zonas de información que se pintan sobre el mapa son muy generales.

Teniendo en cuenta los resultados obtenidos y los parámetros que proporciona cada servidor, el autor de este estudio y desarrollador de este proyecto elegiría Weatherbit. Esto se debe a que es el único servidor que proporciona la información sobre la altura de las nubes de forma desglosada. A pesar de ello, Photopills S.L. no se ha eligido un servidor concreto para ser contratado, por lo que el desarrollo del proyecto se ha realizado para que cualquier servidor de los descritos pueda funcionar.

Capítulo 4

Análisis y diseño del sistema

En este capítulo se describe la fase de análisis previa a la programación de la aplicación. Para ello se presentan diagramas y diseños que sirven de ayuda para entender mejor el proyecto y aportar más detalle sobre los requisitos.

Teniendo en cuenta la metodología usada, que ha sido Scrum, se han realizado diferentes diagramas para mostrar la estructura y la funcionalidad del proyecto.

4.1. Análisis del sistema

Algunos de los diagramas mostrados, como el diagrama de casos de caso, no se suelen usar en este tipo de metodologías ágiles, pero en este caso se añaden para ilustrar mejor el proyecto que se desarrolla.

4.1.1. Diagrama de casos de uso

El diagrama de casos de uso permite representar los diferentes roles del sistema y la funcionalidad con las que están asociados. Un rol puede ser un usuario, un equipo o un software concreto. Como ya se ha explicado, aunque no es necesario el uso de este tipo de diagramas en la metodología Scrum, se realiza para mejorar la comprensión la funcionalidad de la aplicación. En la Figura 4.1 se muestra el diagrama resultante.

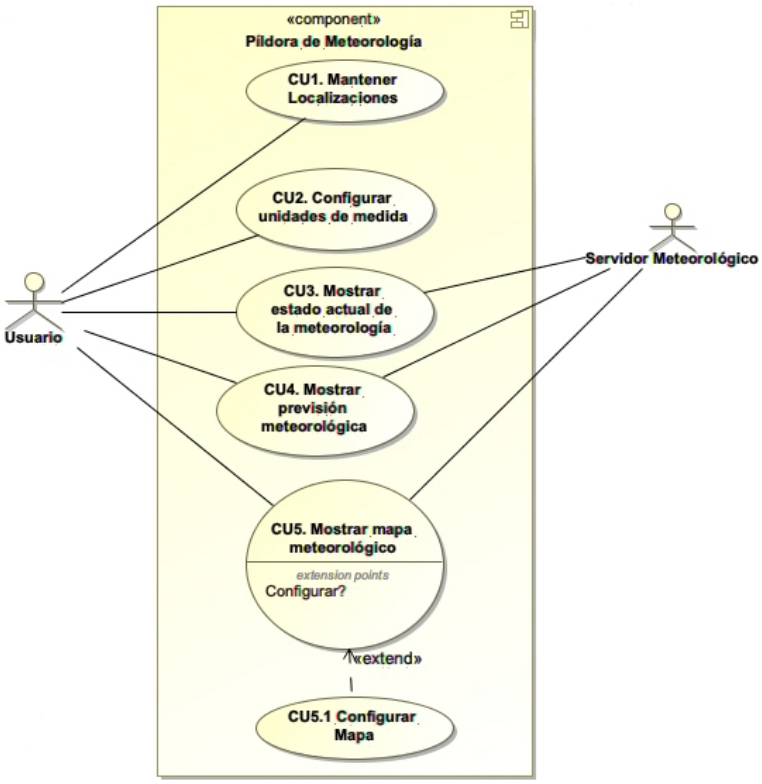

Figura 4.1: Diagrama de casos de uso.

4.1.2. Requisitos de datos

En este apartado se detallan los requisitos de datos que tiene la aplicación. La *píldora* de Meteorología no necesita grandes requisitos para el almacenamiento de los datos, ya que los datos que se van a guardar son relativamente sencillos y como no hay relación entre los diferentes datos, no será necesaria una estructura de almacenamiento relacional. En los Cuadros 4.1 a 4.3 se describen los requisitos de datos que se quieren mantener.

Cuadro 4.1: Especificación requisitos de datos de Estado Meteorológico.

Requisito de datos
Nombre: Estado Meteorológico
Datos específicos a mantener: fecha validez, temperatura, temperatura mínima, temperatura máxima, temperatura aparente, punto de rocío, dirección del viento, velocidad del viento, velocidad ráfagas de viento, precipitación, probabilidad de precipitación, humedad, visibilidad, nieve, presión atmosférica, porcentaje de nubes, porcentaje de nubes altas, porcentaje de nubes medias, porcentaje de nubes bajas y código icono.
Detalles: representa el estado meteorológico en un momento dado con todos los parámetros. El estado actual de la meteorología se mantiene con un objeto de este tipo. La previsión meteorológica se mantiene una lista de componentes de este tipo de dato.

Estos tres requisitos de datos permitirán mantener la información de la aplicación. Al usarse el paradigma de programación orientada a objetos, estos requisitos de datos se representarán

Cuadro 4.2: Especificación requisitos de datos de Localización.

Requisito de datos
Nombre: Localización
Datos específicos a mantener: nombre, coordenadas, Estado Meteorológico actual y lista Estado Meteorológico de la previsión.
Detalles: mantiene la información de un lugar del que se quiera obtener datos meteorológicos. La aplicación contendrá una lista de datos de este tipo para mantener los lugares guardados por el usuario.

Cuadro 4.3: Especificación requisitos de datos de Unidades de medida.

Requisito de datos
Nombre: Unidades de medida
Datos específicos a mantener: unidades temperatura, unidades viento, unidades visibilidad, unidades presión y unidades precipitación.
Detalles: este tipo de datos guarda las unidades de medida con la que se mantienen los datos meteorológicos en la aplicación. Permite mantener la configuración que el usuario establezca para las unidades en las que se mostraran los diferentes parámetros meteorológicos.

en la aplicación como tres clases diferentes, lo cual permitirá crear objetos de estos tipos. Serán almacenados en una base de datos no relacional[1] debido a que no se requieren estructuras jerárquicas por la simplicidad de la información guardada. En la siguiente sección se describe de forma más detallada como se almacenarán estos datos.

[1]Sistema de Gestión de Bases de Datos que no utiliza el sistema de relaciones entre datos de diferentes tablas y los datos almacenados no requieren estructuras como tablas.

4.1.3. Persistencia de datos

Los requisitos de almacenamiento de datos de la aplicación no son exigentes. Los tres tipos de datos que se quieren almacenar son los descritos en la sección anterior. Estos datos carecen de relación entre ellos, lo que hace innecesario el uso de esquemas de almacenamiento complejos o el uso de bases de datos relacionales. A continuación, se describe el motivo por el que se quiere almacenar cada tipo de dato:

- Estado Meteorológico. Se quiere almacenar el estado meteorológico durante un periodo de tiempo corto, de máximo 30 min, para ahorrar llamadas al servidor. De esta forma, si el usuario de la aplicación vuelve a consultar la información meteorológica en un periodo de tiempo inferior al definido, los datos no se pedirán al servidor, sino que se cargarán del almacenamiento local de la última consulta.

- Localización. Se quiere almacenar las localizaciones creadas por el usuario. Estas localizaciones estarán disponibles en una lista para poder volverse a consultar de forma rápida. También se podrán editar o borrar.

- Unidades de medida. Se quiere almacenar la configuración que el usuario establezca acerca de las unidades de medida con las que quiere consultar la información.

Estos tres tipos de datos se almacenarán en un sistema integrado que ofrece iOS llamado Core Data [7], que es un framework[2] usado para gestionar persistencia de los objetos de la aplicación. Core Data proporciona soluciones automatizadas, permitiendo guardar y recuperar objetos creados en tiempo de ejecución sin realizar configuraciones complejas.

4.2. Diagrama de clases a nivel de diseño

En esta sección se representa la estructura del proyecto desde el punto de vista de las clases que conforman el Modelo (Model-View-Controller). Este diagrama se ha perfeccionado conforme se han desarrollado las historias de usuario, creando las clases y relaciones necesarias en cada historia y refinando el diseño en casos puntuales donde se ha necesitado. Para simplificar, en este apartado se muestra el diseño final del diagrama de clases, con los atributos, relaciones y operaciones de cada clase. No se incluyen los métodos *setters* y *getters* para no hacer las clases excesivamente grandes.

Como se puede ver en la figura 4.2, se ha representado un diagrama de clases que conforman el modelo, en el que existe una clase de alto nivel y varias *interfaces* que facilitan la inyección de dependencias, haciendo que el software sea modular, fácilmente modificable y testeable. En el capítulo 5 de Implementación se detalla más cada parte de la implementación y se analizan los patrones de diseño usados y el motivo por el cual se han usado.

[2]Entorno de trabajo con un conjunto de herramientas software que facilitan el trabajo.

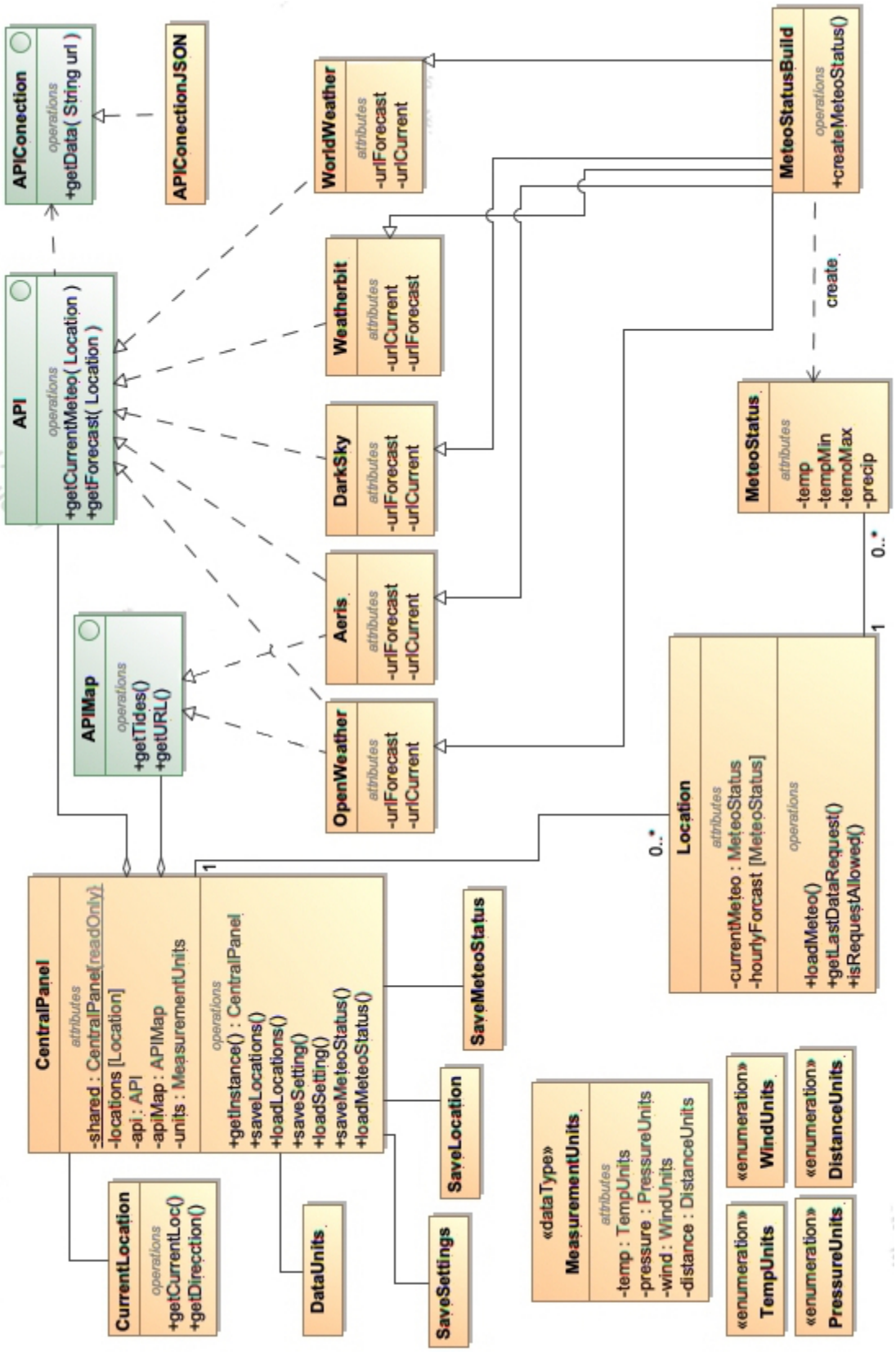

Figura 4.2: Diagrama de clases a nivel de diseño.

4.3. Diseño de la arquitectura del sistema

En este apartado se describen las arquitecturas usadas para la implementación de la *píldora* de Meteorología.

Como en la mayoría de proyectos de software, la aplicación está formada por distintos patrones arquitectónicos combinados. Por una parte, se usa la arquitectura cliente-servidor para la transferencia de los datos meteorológicos. Por otra, se usa el patrón arquitectónico Modelo-Vista-Controlador (MVC) para separar los datos y lógica de la aplicación de la representación y la gestión de eventos.

4.3.1. Arquitectura cliente-servidor

La *píldora* de Meteorología obtiene los datos del estado actual y el pronóstico de los servidores especializados. Esta forma de trabajo se conoce como cliente-servidor. Los servidores proporcionan unos servicios dados y los clientes demandan estos recursos.

Todos los servidores analizados en el Capítulo 3 disponen de una API Rest, que es un tipo de arquitectura de software para sistemas que comparten información a través de Internet. API Rest [11] se basa en la arquitectura cliente/servidor sin estado, en la cual las peticiones se realizan mediante una consulta HTTP[3] en la que se transmite toda la información necesaria para configurar la respuesta del servidor.

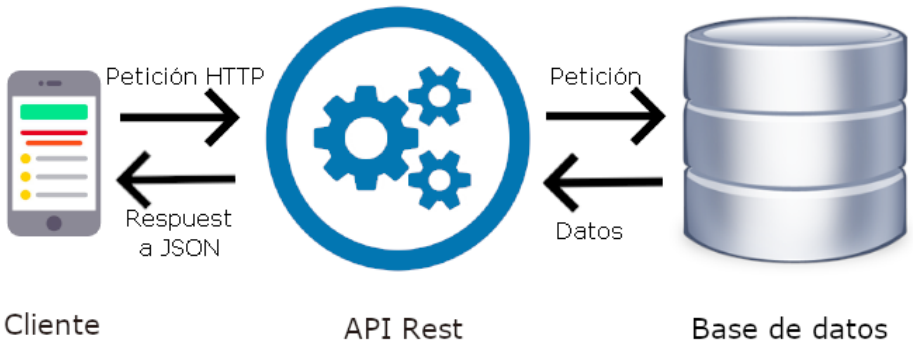

Figura 4.3: Descripción arquitectura API Rest.

Aunque la tecnología API Rest permite realizar operaciones de creación, lectura, actualización y borrado, en este proyecto solo se va a usar para obtener los datos del servidor. La dirección HTTP se va a configurar, dependiendo de la llamada, para obtener un fichero en formato JSON. Las configuración de las URLs[4] depende del servidor meteorológico usado, por

[3]Protocolo de transferencia de hipertexto. Es el protocolo de comunicación que permite la transferencia de información en Internet [8].

[4]Localizador de recursos uniforme o URL, son direcciones que apuntan a servicios localizados en la red.

lo que se debe consultar la documentación para conocer los diferentes tipos de llamadas. En la Figura 4.3 se puede ver el esquema de alto nivel con los componentes que participan en la comunicación API Rest.

4.3.2. Arquitectura Modelo-Vista-Controlador

El MVC o Modelo-Vista-Controlador es un patrón de construcción de software muy usado para la creación de aplicaciones con interfaz gráfica de usuario. Los componentes que forman el software se dividen en tres categorías, conocidos como Vista, Modelo y Controlador, con roles de trabajo distintos. Por una parte, la vista se encarga de la representación y la creación de la interfaz de usuario. Por otra, el controlador que se encarga de proporcionar los datos que la vista mostrará y de manejar los eventos que el usuario pueda hacer en la vista. En este caso el modelo está formado por la estructura jerárquica de clases mostrada en la Figura 4.2.

La parte de la vista y el controlador viene muy definida por la forma de trabajar en Swift. La vista se forma a través de ficheros XML que pueden construirse de forma gráfica usando la herramienta Storyboard integrada en el entorno de desarrollo Xcode. Cada una de las vistas tiene asociado un fichero controlador que se encarga de gestionar los eventos que el usuario realiza en la interfaz gráfica y reaccionar acordemente para mostrar los datos que se obtienen del modelo. A continuación, se listan las tres categorías descritas anteriormente con características propias del desarrollo iOS.

- Vista: formada por ficheros xib, que son un tipo de fichero que usa XML para almacenar el esquema de información de cómo se muestran los componentes (botones, imágenes, texto, etc) en pantalla.
- Controlador: clase que hereda de la superclase UIViewController. Recibe acciones y eventos de la vista y la actualizan.
- Model: representa los datos.

4.4. Diseño de la interfaz

En esta sección se muestran los diseños realizados previamente a la construcción de las *interfaces* en Xcode.

Con el fin de obtener ideas para el diseño, se testearon aplicaciones meteorológicas disponibles en la App Store[5]. A partir de estas ideas y un diseño propio se crearon los mockups[6] que se muestran en esta sección.

La *píldora* de Meteorología estará formada por cuatro pantallas principales donde se mostrará información meteorológica de diferentes modos y varios menús de configuración.

[5]Tienda virtual de aplicaciones en el sistema operativo iOS.

[6]Los mockups son maquetas o diseños usados para representar de forma gráfica como será un producto o interfaz.

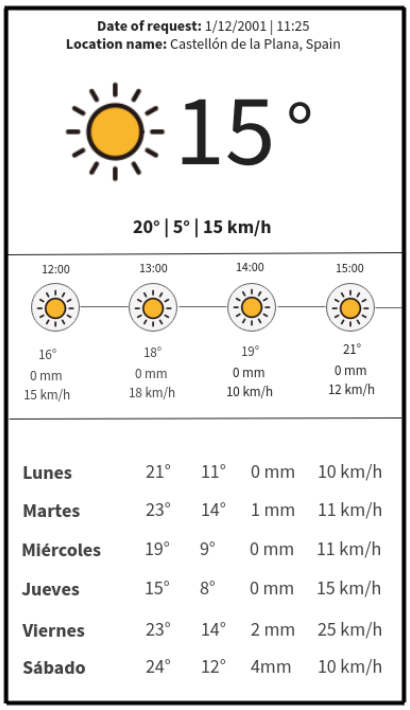

Figura 4.4: Diseño pantalla Info.

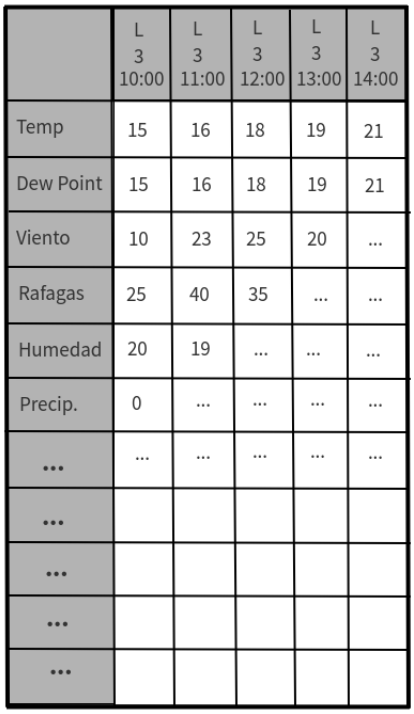

Figura 4.5: Diseño pantalla Tabla.

En la Figura 4.4 se muestra el diseño de la primera pantalla, llamada Info, que el usuario verá al entrar a la *píldora* de Meteorología. Se ha optado por dividir la pantalla en tres secciones. En la sección superior se muestran datos meteorológicos del estado actual, así como información sobre la localización que se está consultando. En la sección intermedia se muestra la evolución por horas. Con un desplazamiento a izquierda o a derecha se puede ver el pronóstico para los próximos días desglosado en horas. En la sección inferior se muestra un resumen del pronóstico para los próximos días. El orden de las secciones está pensado teniendo en cuenta el orden en que pasarán los eventos mostrados.

Como se puede observar, la interfaz de inicio muestra datos meteorológicos de carácter general, sin entrar en detalles. Se ha optado por hacerlo así para que el usuario pueda ver de forma rápida la evolución y en caso de que quiera ver más detalle, puede acceder a las demás pantallas.

En la Figura 4.5 se puede ver el diseño de la pantalla Tabla, donde se muestra información meteorológica más detallada. Se ha diseñado este cuadro para que el usuario pueda desplazarla horizontalmente y verticalmente, pero manteniendo fijas la columna y fila primeras. De esta forma el usuario puede saber siempre a la etiqueta y la hora a la que pertenece el dato que está viendo.Esta interfaz está diseñada con el fin de mostrar los datos específicos que los fotógrafos podrían tener en cuenta para planificar sus fotografías, ya que se mostrarán todos los parámetros que se detallan en la sección 4.1.

En la Figura 4.6 se puede ver el diseño de la pantalla Gráficos. En esta se mostrará la evolución de diferentes parámetros meteorológicos según su previsión. De esta manera el usuario podrá ver de forma rápida las tendencias que sigue la meteorología. Los gráficos de líneas estarán

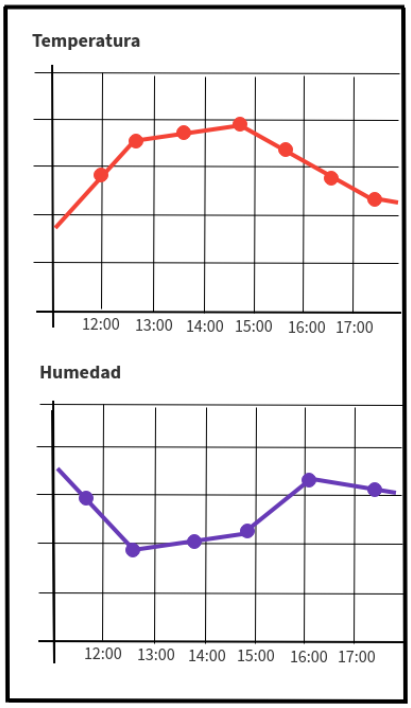

Figura 4.6: Diseño pantalla Gráficas.

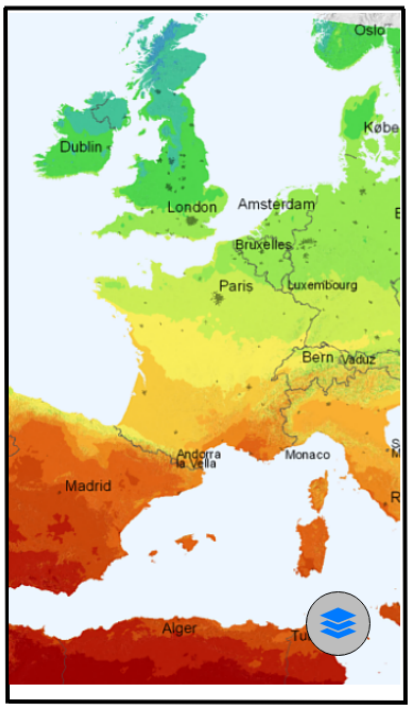

Figura 4.7: Diseño pantalla Mapa.

detallados a nivel de horas y se podrán desplazar horizontalmente con el fin de mostrar más información. Aunque en el diseño se muestren solo dos gráficos de línea, esta pantalla se podrá desplazar verticalmente y se podrán ver más gráficos de otros parámetros meteorológicos.

En la Figura 4.7 se muestra la pantalla Mapa. Esta interfaz mostrará un mapa meteorológico con diferente información que se aplicará por capas sobre el mapa. La capa podrá elegirse en el botón que aparece en la parte inferior derecha, donde aparecerá un menú con las diferentes opciones.

En la Figura 4.8 se muestra la pantalla de Configuración, en esta el usuario podrá elegir la localización para la que quiere obtener los datos meteorológicos y establecer las unidades de medida con la que se mostrarán los datos. En la sección superior se muestra la lista de localizaciones creadas por el usuario. La localización se podrá seleccionar tocando sobre el ítem de la lista y se podrá editar o borrar desplazando el ítem a la izquierda. En la parte inferior se muestra la sección de configuración de unidades de medida. Se sitúa en esta posición debido a que el usuario accederá muy pocas veces y conforme vaya creciendo la lista de la sección superior, esta cada vez aparecerá más abajo.

En la Figura 4.9 se muestra la pantalla de Localización, en esta el usuario podrá crear o editar localizaciones. En la parte superior el usuario podrá indicar el nombre de la localización y en la inferior colocar el pin sobre el lugar que se quiere guardar. Este diseño se ha inspirado en otras pantallas usadas dentro de la aplicación PhotoPills.

Figura 4.8: Diseño pantalla Configuración.

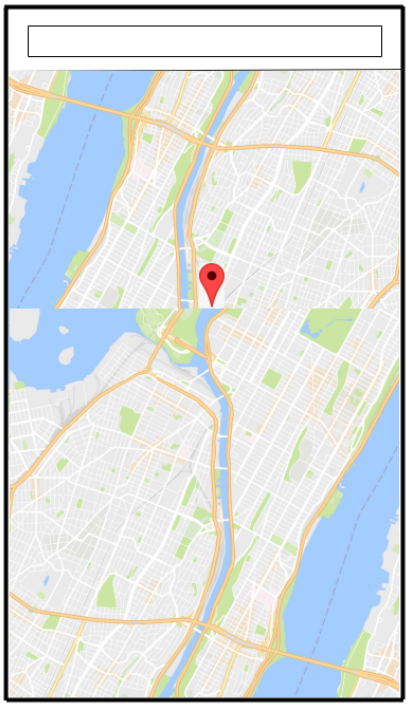

Figura 4.9: Diseño pantalla Localización.

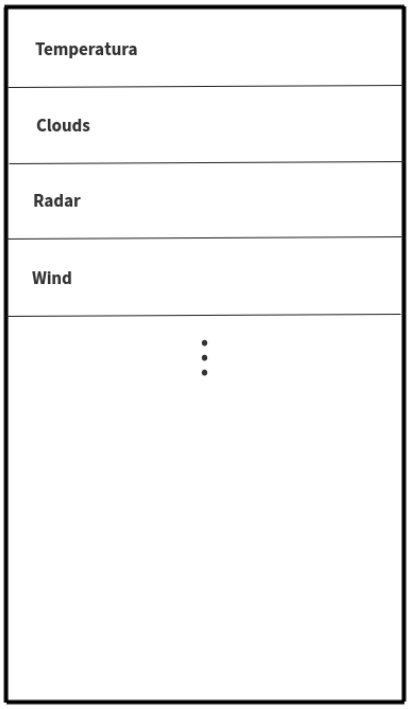

Figura 4.10: Diseño pantalla Configuración Mapa.

En la Figura 4.10 se muestra la pantalla de Configuración Mapa, en esta el usuario podrá elegir la capa de información que se mostrará sobre el mapa de la Figura 4.5. El usuario podrá elegir una opción seleccionado uno de los ítems de la lista.

Como se puede observar en todas las figuras, no existe opción de acceder a la pantalla de Configuración. Esto se debe a que estas opciones las establece iOS de forma nativa en la barra superior y en este caso no se modificará ese diseño y se seguirá las recomendaciones de diseño de Apple conforme al posicionamiento de estos botones en pantalla.

Para el diseño de estas *interfaces* se ha querido mantener los rasgos y estilo de la aplicación PhotoPills. A pesar de que no se muestra el color de los diferentes componentes, se tendrá en cuenta para el diseño final.

Para la navegación entre *interfaces* se utiliza un componente nativo de iOS llamado Tab Navigation. En este aparece una sección en el borde inferior con los cuatro ítems que representan las cuatro pantallas principales, Figuras 4.4 a 4.5, mostradas anteriormente. De esta forma se puede acceder a cualquier pantalla desde cualquier otra. En la Figura 4.11 se muestra como será la navegación.

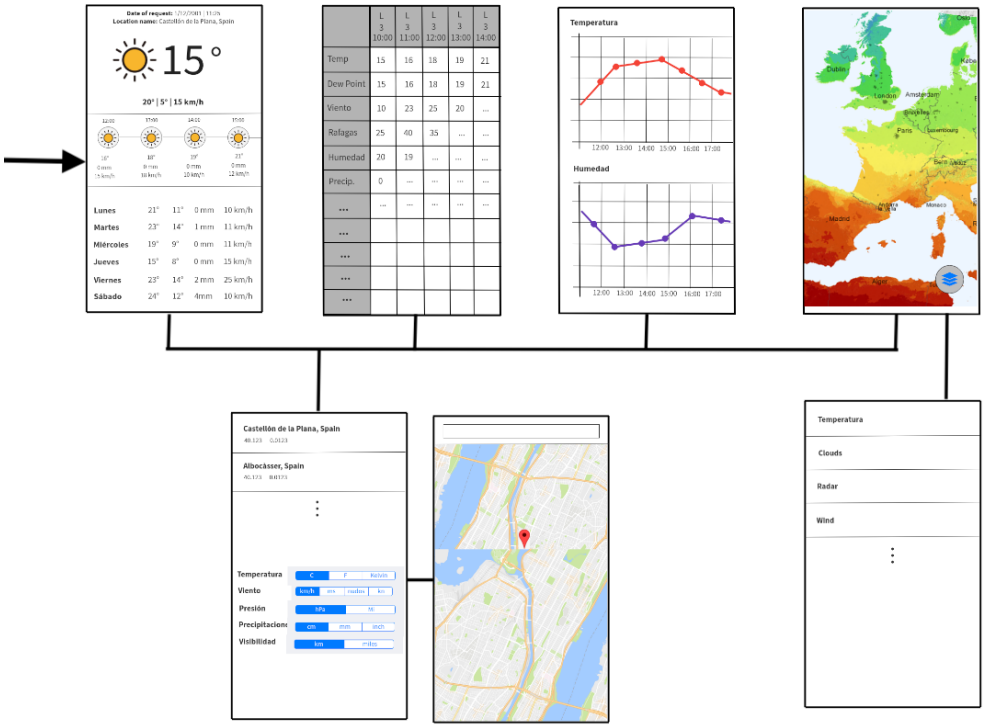

Figura 4.11: Navegación entre las diferentes pantallas.

Las líneas que aparecen en la Figura 4.11 representan las uniones de navegación entre cada pantalla. Todas son multidireccionales. El punto de inicio de la aplicación es la pantalla Info, que será la primera que verá el usuario al acceder a la aplicación. La pantalla de Configuración se podrá acceder desde cualquiera de las cuatro pantallas principales, ya que el botón de acceso aparecerá en la barra superior de todas ellas. Mientras que la pantalla de configuración del mapa solo podrá ser accesible desde la pantalla Mapa.

Capítulo 5

Implementación y pruebas

En esta capítulo se describe el resultado de la fase de implementación llevada a cabo en los diferentes Sprint que se han realizado para completar el proyecto y las pruebas realizadas en el código para comprobar que el comportamiento es el esperado.

5.1. Detalles de implementación

En esta sección se describen aspectos técnicos del desarrollo. Como ya se ha explicado anteriormente, el proyecto se divide en tres partes principales basadas en el patrón Modelo-Vista-Controlador. La sección de *Patrones de diseño* está orientada a describir la implementación del modelo. Mientras que la sección de *Interfaces de usuario* muestra las diferentes pantallas que forman la vista y se aprovecha para explicar las diferentes clases que forman el controlador.

Alguno de los aspectos descritos en esta sección ya han sido introducidos anteriormente, aunque en este caso se muestran con más detalle.

5.1.1. Patrones de diseño usados

En el desarrollo de este proyecto se han usado múltiples patrones de diseño que se describen en esta sección.

Los patrones de diseño describen técnicas para resolver problemas habituales en el desarrollo de software, permitiendo nombrar una solución usada de forma concisa, sin tener que hacer explicaciones extensas. De esta forma, si se indica que se ha usado un patron Observer[1] se está transmitiendo mucha información. La principal ventaja de su uso es que proporcionan soluciones muy probadas y permiten ahorrar tiempo de diseño ya que se puede detectar una solución a partir de contextos concretos.

[1]Permite notificar a un objeto subscrito el cambio de estado del objeto subscriptor.

Singleton

El Singleton o instancia única, es un patrón que permite limitar la creación de objetos de una clase. De esta forma una clase solo tiene una instancia, haciendo que todas las demás clases que utilicen este objeto estén trabajando con los mismos datos. Este patrón puede conllevar problemas en la ejecución multi-hilo[2] por el hecho de compartir la misma instancia, por eso se contemplan medidas para asegurar la integridad de los datos en todo momento.

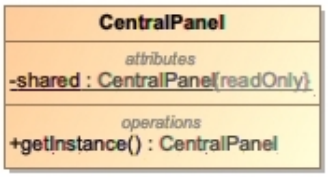

Figura 5.1: Clase *singleton* CentralPanel.

Como se muestra en la Figura 5.1 la clase CentralPanel incluye el patrón Singleton. Se ha querido que esta clase solo pudiese ser instanciada una única vez y su instancia fuese compartida por todas las demás clases que la usen. Para evitar problemas con el multi-hilo el atributo *shared* es una instancia de la clase CentralPanel que es inicializada por el hilo principal la primera vez que se usa. Además, CentralPanel también hace de clase de alto nivel a través de la cual se realiza la inyección de dependencias y es una especie de puerta de acceso desde el controlador al modelo.

Inyección de dependencias

La inyección de dependencias o en inglés Dependency Inyection, es un patrón de diseño o técnica muy usada en el paradigma de programación orientada a objetos. Permite suministrar objetos a una clase en lugar de ser esta clase la que cree dichos objetos [13]. Es habitual que esta técnica haga uso de *interfaces* para definir los objetos que se pueden instanciar ante una determinada declaración de variable. Como se muestra en la Figura 5.2, existe una clase llamada CentralPanel que se encarga de la inyección de dependencias. Los objetos que se pueden instanciar deben cumplir con la *interface* concreta.

[2]En la programación multi-hilo dos o más partes pueden ejecutarse simultáneamente.

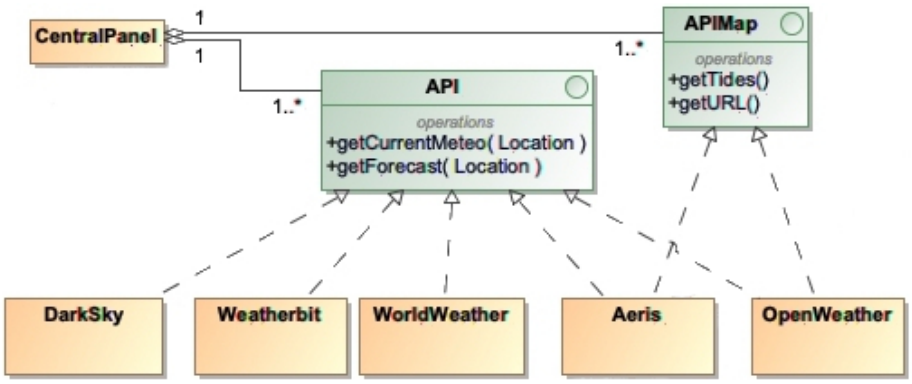

Figura 5.2: Inyección de dependencias en la clase CentralPanel.

Este patrón permite un diseño de la estructura en diferentes módulos, haciendo que el código esté menos acoplado, las pruebas de test y posibles cambios en un futuro. Permite hacer la inyección de dependencias falsas[3], aislando las diferentes clases. Incluso permite el cambio de instancia en tiempo de ejecución.

Builder

El patrón Builder simplifica la creación de objetos complejos que tengan muchos atributos, como es el caso de MeteoStatus y todos sus atributos que almacenan parámetros meteorológicos. Con el uso de este patrón se evita transferir todos estos parámetros a través del constructor de la clase o a través de métodos *setters*. Como se muestra en la Figura 5.3, en el proyecto se aplica este patrón para la creación de objetos de tipo MeteoStatus, que almacenan la información meteorológica en un determinado momento.

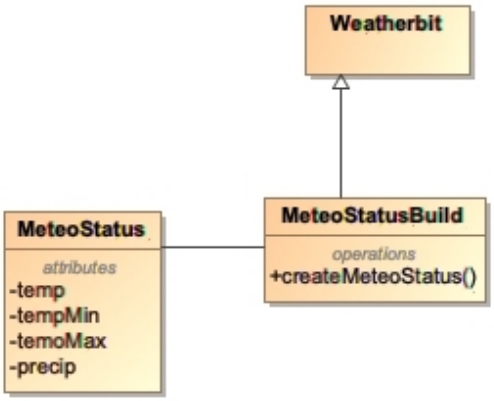

Figura 5.3: Inyección de dependencias en la clase CentralPanel.

[3]Técnica usada en las pruebas de software para emular el comportamiento de un componente.

Strategy

El patrón Strategy o estrategia nos permite definir una serie de algoritmos, en este caso en diferentes clases, y hacerlos intercambiables. Esto se debe a que las clases cumplen con una *interface* compartida, pero cada clase implementa los métodos de forma particular según su finalidad. En la Figura 5.2 del apartado de *Inyección de dependencias* se muestra el uso del patrón Strategy varias veces. Como se puede ver hay dos *interfaces* y las clases que las implementan. La *interface* API se encarga de definir unos métodos que deben implementar las clases que hacen las petición de datos al servidor concreto. Mientras que la *interface* APIMap hace lo mismo, pero en este caso define los métodos que se encargan, en las clases que los implementen, de hacer las peticiones de las imágenes que se colocan sobre los mapas meteorológicos. Esta estructura de clases que implementan las *interfaces* hace que se pueda intercambiar el comportamiento de la clase *CentralPanel* según el objeto que se inyecte en la clase CentralPanel. En este caso se podría cambiar el servidor del que se obtienen los datos incluso en tiempo de ejecución cambiando la instancia del objeto.

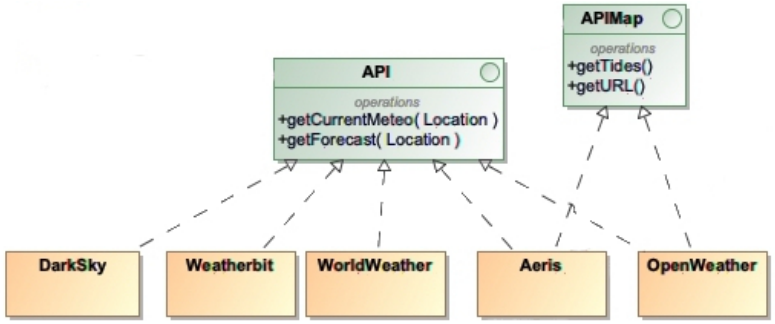

Figura 5.4: Uso de patrón Strategy.

En la Figura 5.4 se muestra otro ejemplo de uso del patrón. En este caso la *interface* APIConection define unos métodos que se usan, en las clases que implementen la interfaz, para descargar los datos del servidor, mientras que la clase APIConectionJSON se encarga de implementar estos métodos para los servidores que proporcionen los datos en ficheros JSON, que en el caso de este proyecto son todos los servidores. Con esta estructura, se puede obtener los datos de un servidor en formato XML sin realizar muchos cambios, bastaría con crear una clase que implementase los métodos de esta *interface*.

Closures

Los closures o cierres son bloques de código que se pasan como argumento en una función y se ejecutan dentro de esta en un momento determinado. Más que un patrón, son una técnica muy potente que puede usarse ante ciertas circunstancias.

Su uso se restringe a aquellos métodos encargados de obtener información que puedan tardar en obtenerla y retornarla, evitando que la aplicación se pare esperando la respuesta. En el caso de este proyecto se usan closures en el método que hace las peticiones de datos al servidor y en los métodos que cargan los datos de la base de datos. En el primer caso, cuando el método obtiene los datos JSON procedentes del servidor, genera una instancia de la clase que contendrá

esta información, en este caso MeteoStatus, y el método ejecuta el closure, respondiendo con la instancia creada. En el segundo caso el comportamiento es similar, pero los datos se obtienen de la base de datos.

5.1.2. Interfaces de usuario

En esta sección se presentan las diferentes *interfaces* de usuario que conforman la vista de la *píldora* de Meteorologia y los ficheros de lenguaje Swift que forman la parte del controlador de cada *interface* de usuario.

Para la navegación entre las diferentes pantallas se hace uso de una solución que proporciona Swift de forma nativa llamada *Tab Bar Controller*. Este tipo de navegación está basado en una barra inferior o superior en la que se muestran las diferentes pestañas que refencencian a cada pantalla. El usuario solo puede ver una pantalla a la vez y cuando elige una de las opciones de la barra de pestañas se cambia de pantalla usando animaciones que ya vienen establecidas por defecto.

Este tipo de navegación es muy usual en las aplicaciones iOS y se emplea en las otras *píldoras* de la aplicación PhotoPills. Aunque es muy editable, el *Tab Bar Controller* viene preconfigurado con funcionalidades y animaciones que ahorran mucho trabajo.

A continuación, se muestra cada pantalla, la forma en que se han construido y los controladores asociados que disponen. Se intenta no duplicar la información que se muestra en la sección *Diseño de la interfaz* del Capítulo 4 de *Análisis*.

Pantalla Meteo

En la Figura 5.5 se muestra la interfaz que contiene los datos meteorológicos generales. Esta vista, al igual que las demás, se ha construido usando la herramienta gráfica integrada en Xcode para el diseño de las *interfaces*, que internamente se transfiere a formato XML para guardar las propiedades de cada componente.

Esta vista es manejada por una clase controladora llamada ViewControllerMeteo, que hereda de la clase ViewController. Esta clase padre define una serie de comportamientos que son comunes en todos los controladores de vistas, como actualizar la vista, responder ante eventos del usuario, redimensionar la vista, etc.

Esta pantalla se divide en tres secciones. En la sección superior se muestra la información de la última petición de datos al servidor, la localización que se está consultando e información meteorológica del estado actual.

En la sección intermedia se muestra una lista de tipo UICollectionView con la previsión meteorológica por horas. Este tipo de lista es un objeto nativo de Swift encargado de administrar colecciones de elementos y presentarlos utilizando diseños personalizables. Cada elemento de la lista es una vista independiente, con su controlador asociado. El controlador principal de la vista

se encarga de crear cada elemento conforme se va haciendo *scroll* en la lista. Estos elementos se crean a partir de un fichero *xib* llamado HourlyCell que define la vista y hace de modelo. En este caso, el controlador de cada elemento hereda de la clase UICollectionViewCell, que implementa el comportamiento que es común en los elementos que forman las listas.

En la tercera sección se muestra un resumen meteorológico para los próximos días. Al igual que en la sección intermedia, se usa una lista de tipo UICollectionView formada por vistas independientes con su controlador asociado a cada una de estas. En este caso la vista se construye a partir de un fichero *xib* y el controlador asociado se define en la clase DailyCell, que hereda de la clase UICollectionViewCell.

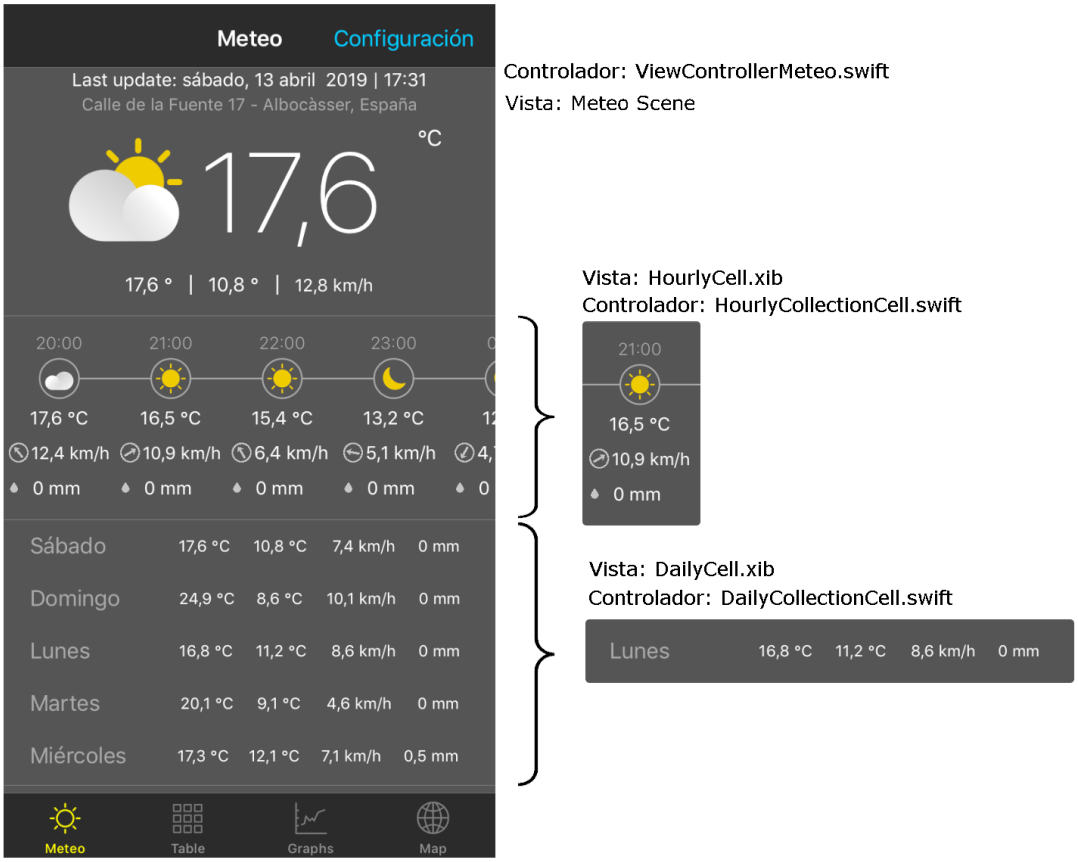

Figura 5.5: Pantalla en la que se muestra información meteorológica de forma resumida.

Pantalla Tabla

En la Figura 5.6 se muestra la pantalla que contiene los datos meteorológicos más específicos, como información del porcentaje de nubes en cada altura, velocidades del viento y rachas, punto de rocío, etc. Está formada por una tabla que puede deslizarse horizontal y verticalmente, pero siempre mantiene la primera fila y la primera columna fijas.

Para crear la tabla se hace uso de una clase llamada TableViewLayout que hereda de la

clase Swift llamada UICollectionViewLayout. En esta se define el comportamiento y la forma en que se muestran los datos. Además se definen los controladores para las celdas, que pueden ser de dos tipos. Uno llamado MeteoTableCell_Label para las celdas que muestran texto y otro llamado MeteoTableCell_Image para las celdas que muestran un icono.

Con el fin de mejorar el rendimiento y que el desplazamiento sea fluido, se reutilizan las celdas. Inicialmente los objetos tipo celda que son visibles en pantalla. Mientras el usuario hace *scroll* en la lista se asigna el texto o icono concreto a las celdas, evitando crear celdas nuevas y destruirlas cuando no están visibles. Toda esta gestión de creadión y asignado de valores a cada celda se lleva a cabo en el controlador principal de la vista, llamado ViewControllerTableMeteo. Este controlador, al igual que los demás controladores principales, hereda de la clase UIViewController, ya descrita anteriormente.

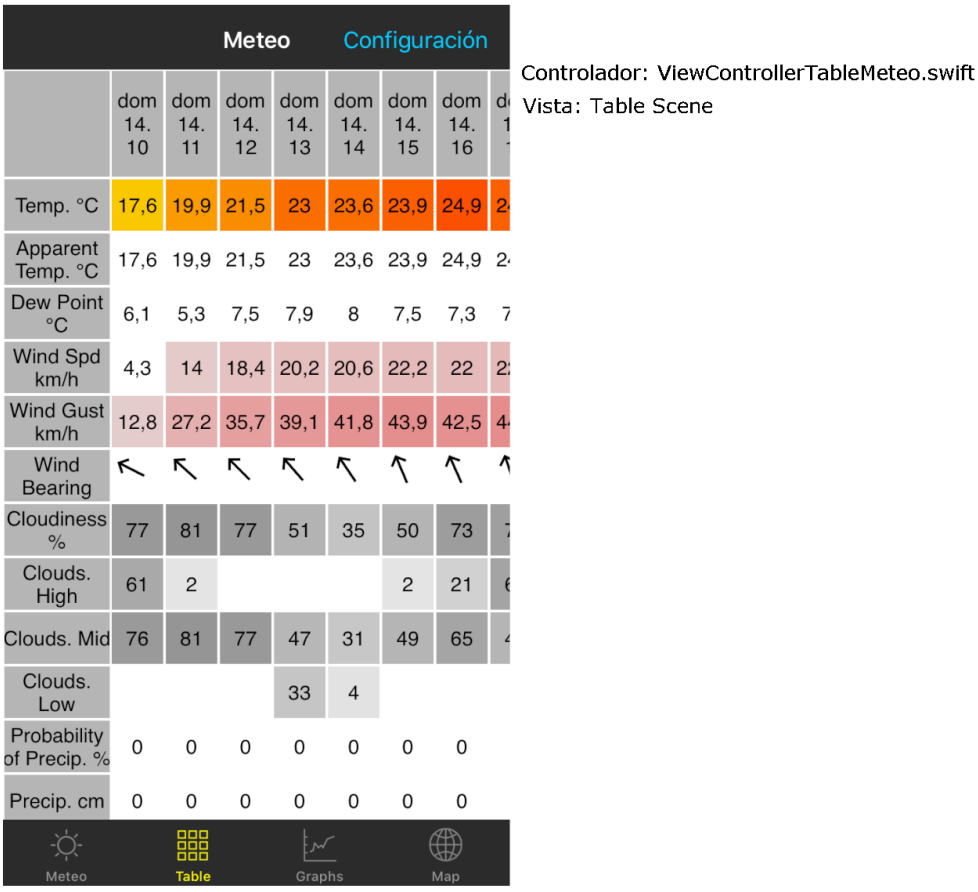

Figura 5.6: Pantalla en la que se muestra información meteorológica específica.

Pantalla Gráficos

En la Figura 5.7 se muestra la pantalla que contiene la evolución de la previsión mediante el uso de gráficos de líneas, lo que permite al usuario ver de forma rápida las tendencias de cada parámetro.

Para la creación de las gráficas se emplea una clase llamada GraphView, en esta se define el comportamiento necesario para dibujar las barras horizontales y verticales y la línea del gráfico, así como las deferentes etiquetas de texto. Para el dibujado de las gráficas se dispone de varias opciones. Se puede descargar un componente de software ya implementado que permita crear gráficas o dibujarlas desde cero mediante el uso de herramientas integradas en Swift. La primera opción ahorra mucho tiempo de desarrollo, pero la apariencia de la gráfica se ciñe a las opciones de configuración que tenga el componente descargado. En el caso de este proyecto se ha optado por la segunda opción para poder conseguir un gráfico de línea a medida, tal cual se ha dibujado en el diseño.

El dibujado de la gráfica hace uso del framework nativo de Swift llamada *Core Graphic*, en el que existen clases encargadas del facilitar la tarea de dibujado sobre un componente.

La pantalla mostrada en la Figura 5.7 está distribuida por secciones en las que se muestra una gráfica de líneas. Su controlador principal, ViewControllerGraphs, se encarga de inicializar y hacer la transferencia de los datos que se muestran. Cada gráfica se crea es una vista independiente, definida en un fichero *xib* y con un controlador asociado. Esta construcción modular permite añadir gráficas fácilmente a la vista principal.

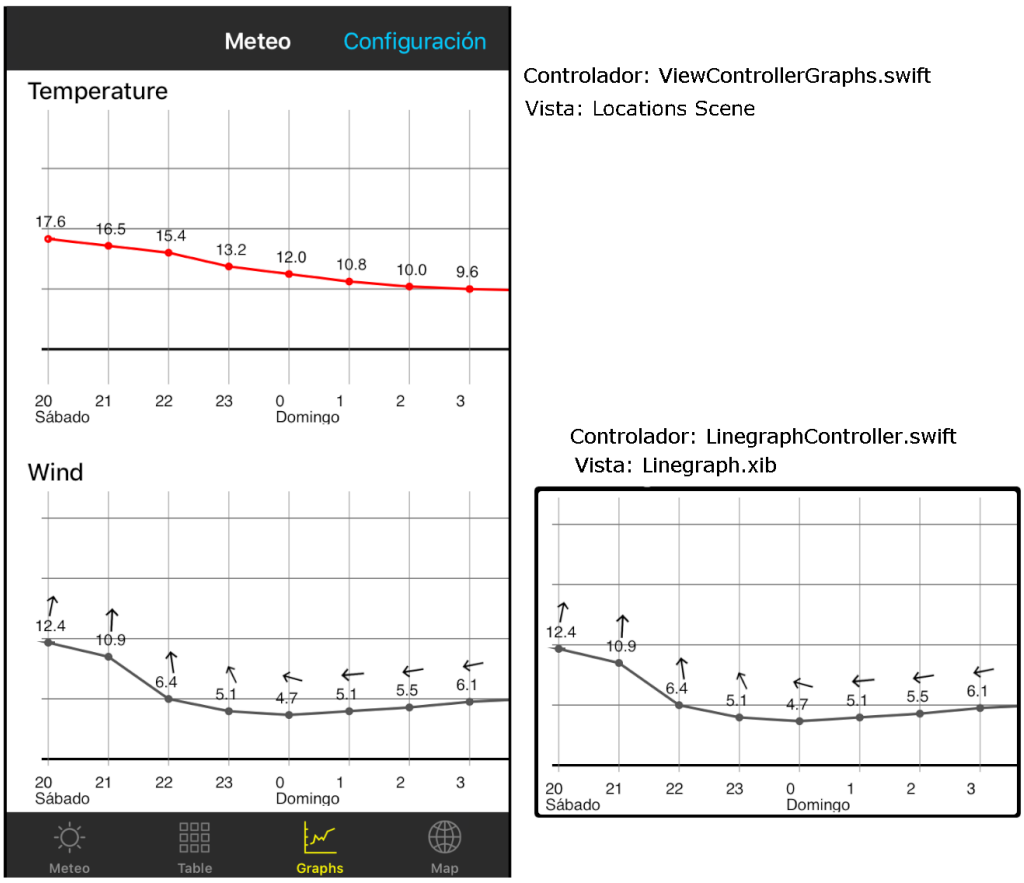

Figura 5.7: Pantalla en la que se muestra gráficos de líneas.

Pantalla Mapa

En la Figura 5.8 se muestra la pantalla que contiene el mapa meteorológico. El controlador de la vista, ViewControllerMap, usa un componente nativo de Swift llamado MapKitView que se encarga de mostrar un mapa. Este controlador localiza la región del mapa que se muestra y aplica la imagen semitransparente que contiene la información meteorológica.

Mediante el botón blanco que aparece en la parte inferior derecha se accede a la pantalla de configuración del mapa, que se explica en el siguiente apartado.

Las diferentes capas las transfiere el modelo, encargado de realizar la petición al servidor meteorológico y se aplican sobre la región del mapa conforme se obtienen.

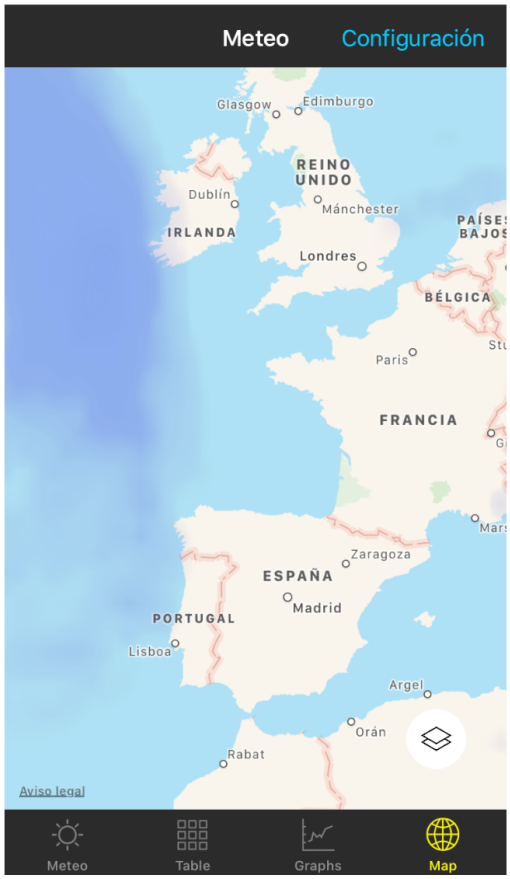

Figura 5.8: Pantalla donde se muestra la información meteorológica sobre un mapa.

Pantalla Configuración Mapa

La pantalla de configuración del mapa, Figura 5.9, muestra las diferentes opciones de capas de información meteorológica que se pueden obtener y el tipo de mapa (hibrido, terreno y satélite). Cuando se elige cualquiera de las opciones de esta vista, automáticamente se cierra esta pantalla y se aplican los cambios en el mapa. La vista principal tiene un controlador

asociado, ViewControllerMapConfiguation, que se encarga de inicializar la lista con todas las opciones. La lista es un componente UICollectionView y cada uno de los elementos tiene un controlador asociado, encargado de establecer el título y el icono. A diferencia de la pantalla Meteo, los diversos elementos de la lista no tienen una vista asociada. Esto se debe a que la simplicidad de cada elemento hace innecesario el uso de una vista independiente.

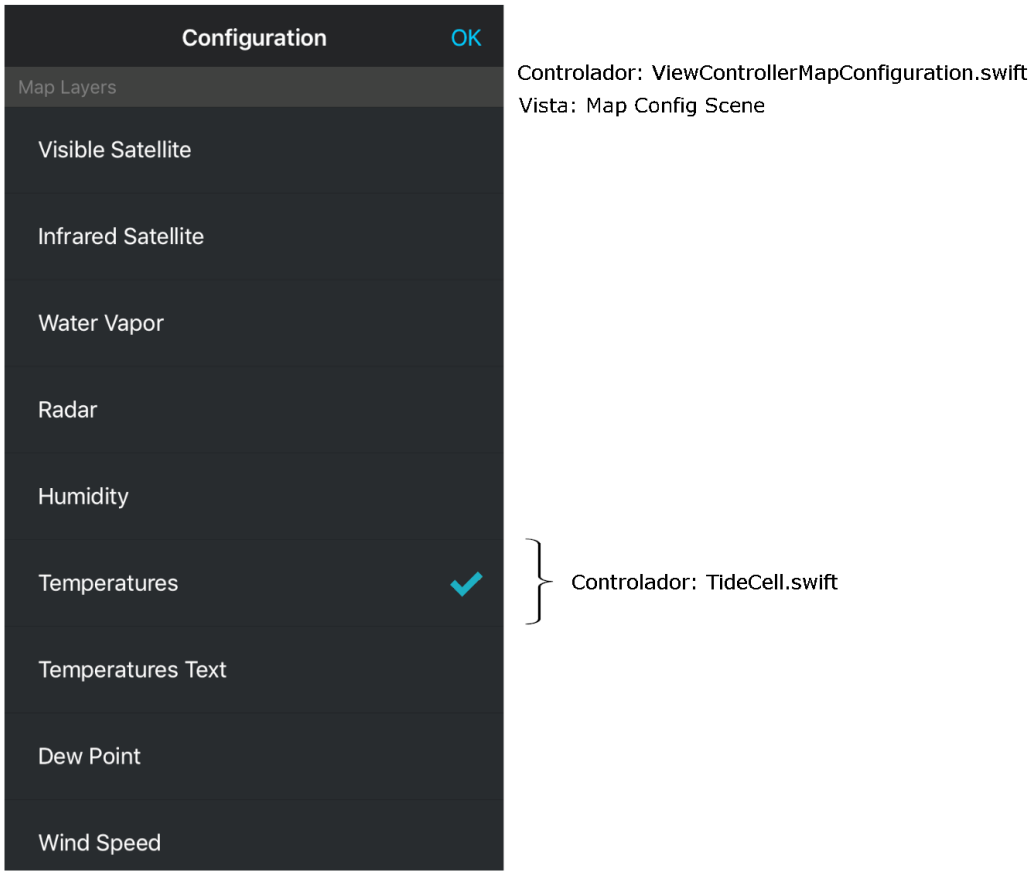

Figura 5.9: Pantalla de configuración del mapa.

Pantalla Configuración

En las pantallas principales de la aplicación, Fig 5.5 a 5.8, se muestra en la parte superior derecha un botón de Configuracion. Presionando este botón se accede a la pantalla de configuración, mostrada en la Figura 5.10, en la que se pueden diferenciar dos secciones. En la parte superior se muestra la lista de localizaciones creadas por el usuario y en la parte inferior la configuración de las unidades.

Las dos secciones están contenidas dentro de la misma lista, de tipo *UICollectionView*, pero con tipos de celda distintos. El primer tipo, asociado a unas controlador llamado LocationCell, es el que muestran un la información de la localización. Este tipo de celda son creadas dinámicamente por el controlador de la vista principal, ViewControllerLocation, a partir de los datos que el modelo proporciona. El segundo tipo de celda es un elemento estático que engloba todas las

opciones de unidades de medida. Este tipo de celda no se crea dinámicamente, es el controlador el que crea un único elemento de este tipo de forma estática. Esto se debe a que este elemento no cambia, siempre contiene el mismo contenido.

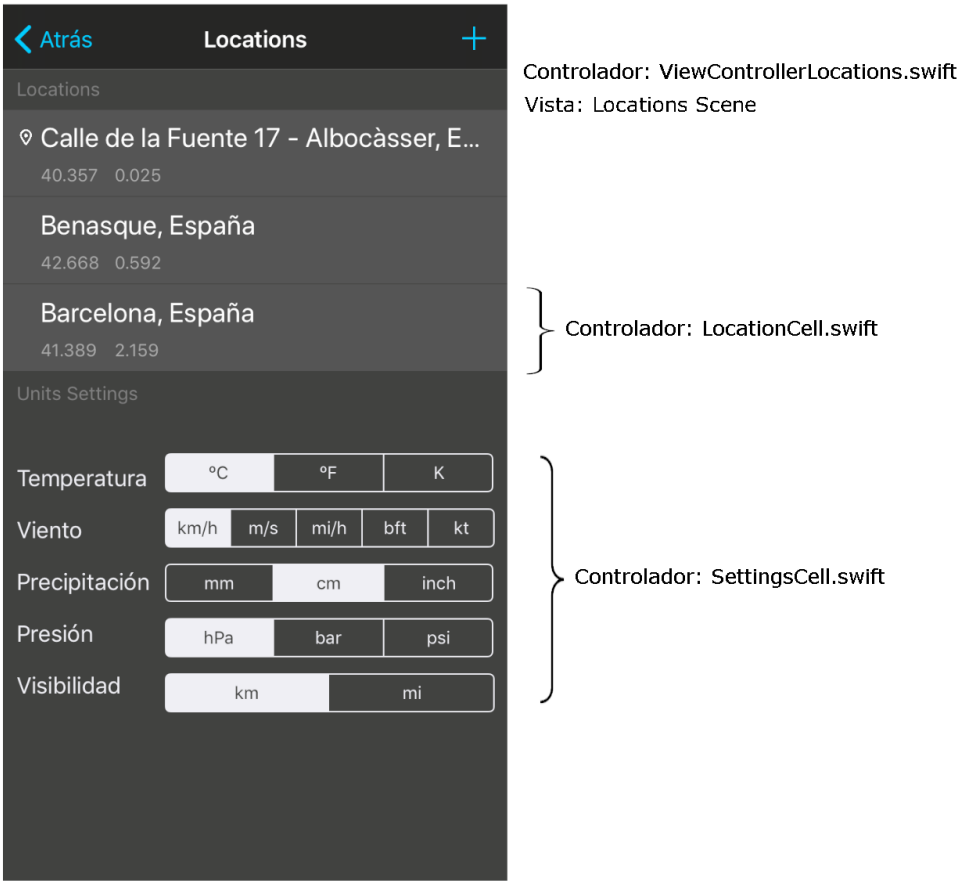

Figura 5.10: Pantalla para establecer la localización y las unidades.

Pantalla Nueva Localización

En la Figura 5.11 se muestra la pantalla en la que se añaden o editan las localizaciones. Esta vista está dividida en dos secciones. En la sección superior hay un campo de texto donde se introduce o modifica el nombre de la localización. Según donde esté colocado el pin en el mapa, el controlador de la vista, *ViewControllerNewLocation*, proporciona una sugerencia de nombre. Si se hace mucho *zoom* y se arrastra el pin sobre una calle concreta, sugiere el nombre de la calle, la población y el país, mientras que si se arrastra sin hacer *zoom*, el controlador sugiere la población y país o solamente el país. La gestión de estas recomendaciones de nombre la realiza una clase específica encargada de interpretar unas coordenadas y sugerir un nombre. La información sobre direcciones, poblaciones y países se obtiene a partir del paquete *MapKit* disponible en Swift.

En la sección inferior se muestra un mapa con un pin con una localización marcada. El funcionamiento del pin es el siguiente:

- Si se añade una nueva localización, el pin se coloca en el centro de la pantalla y sobre la localización actual. En caso de no tener la localización actual, se usa la última localización consultada o la localización por defecto. Si no hubiese ninguna de estas, se mostraría una localización predefinida en el código.
- Si se está editando una localización, el pin se coloca centrado en la pantalla y sobre la localización que se va a editar.

Para elegir una nueva posición de la localización sobre el mapa se puede arrastrar y soltar el pin donde se quiera.

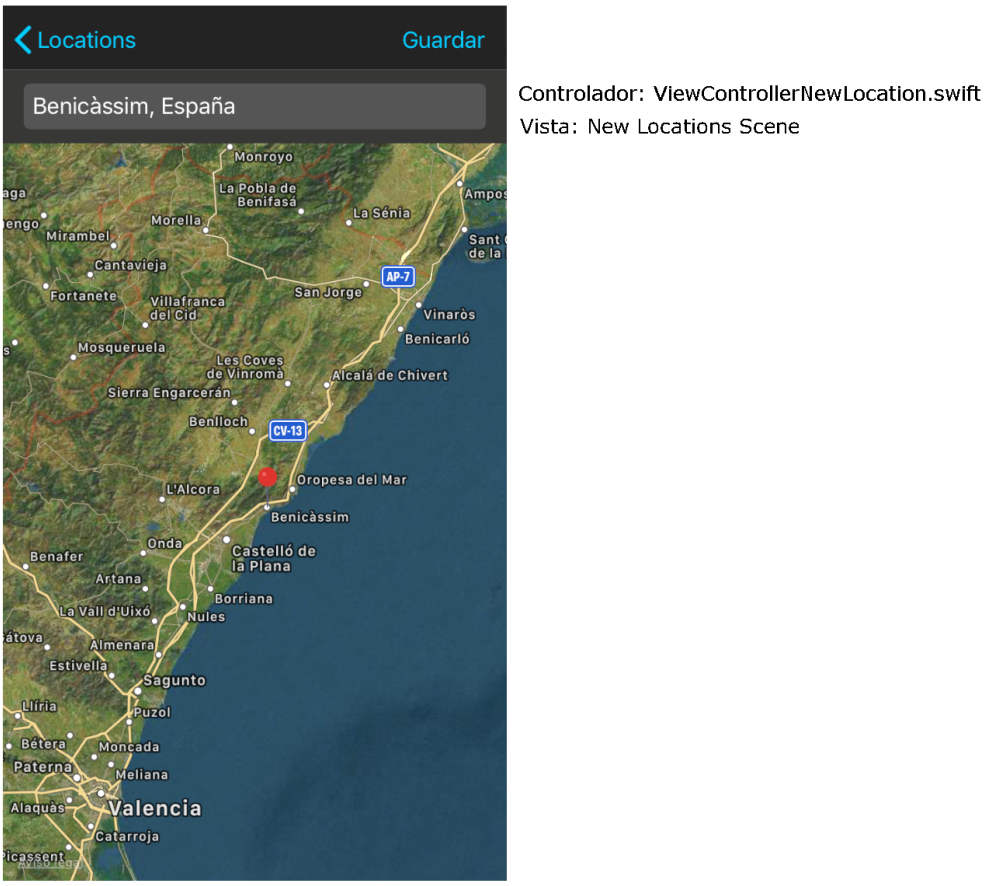

Figura 5.11: Pantalla para introducir nuevas localizaciones.

5.2. Verificación y validación

En esta sección se describen las diferentes pruebas que se han realizado para verificar y validar el funcionamiento de la *píldora* de Meteorología. Se debe tener en cuenta que en la pila del producto han quedado tareas por realizar relacionadas con pruebas de software. Pese a que las pruebas descritas en esta sección pueden garantizar el comportamiento correcto de las historias de usuario completadas, se deberían realizar pruebas más exhaustivas antes del lanzamiento de la *píldora*.

5.2.1. Pruebas unitarias

Este tipo de pruebas tienen como objetivo comprobar el correcto funcionamiento de una unidad de código [12]. En el Cuadro 5.1 se exponen las pruebas realizadas y una breve descripción.

Cuadro 5.1: Escenarios testeados en las pruebas unitarias.

Escenario: el usuario obtiene los datos meteorológicos del estado actual para una localización. **Historias testeadas:** HU01,8 **Comentario:** se muestran los datos de temperatura, temperatura mínima, temperatura máxima, velocidad del viento e icono.
Escenario: el usuario obtiene los datos meteorológicos del pronóstico para las próximas 48h. **Historias testeadas:** HU02, 8 **Comentarios:** la lista del pronóstico tiene 48 ítems.
Escenario: el pronóstico contiene todos los datos meteorológicos. **Historias testeadas:** HU02, 3, 4, 7, 8. **Comentarios:** los elementos del pronósticos contienen la hora, fecha, temperatura, dirección y velocidad del viento, velocidad de las rachas, humedad, porcentaje de nubes, presión, punto de rocío, temperatura aparente, precipitaciones, probabilidad de precipitaciones y nieve.
Escenario: el usuario obtiene los datos meteorológicos del pronóstico para los próximos 5 días. **Historias testeadas:** HU02, 8 **Comentarios:** la lista de *Estados Meteorológicos* tiene 5 ítems y estos ítems contienen la temperatura mínima y máxima, velocidad del viento y precipitación acumulada.
Escenario: el usuario el usuario obtiene todas las localizaciones guardadas. **Historias testeadas:** HU09 **Comentarios:** se comprueba que el número de localizaciones de la lista coincida con las creadas. Se comprueba que la lista no incluya una localización que ha sido borrada. Se comprueba que la lista contenga una localización cuyo nombre ha sido editado.
Escenario: el usuario puede cambiar las unidades con las que se muestran los diferentes parámetros meteorológicos. **Historias testeadas:** HU06 **Comentarios:** a partir de un valor numérico establecido se comprueba el cambio de unidades de temperatura, velocidad del viento, presión atmosférica, precipitación y distancia de visibilidad.
Escenario: el sistema carga los datos meteorológicos de caché si estos se han obtenido del servidor hace menos de 30 minutos. **Historias testeadas:** HU01, 2, 3, 4, 7 , T.2, **Comentarios:** a partir de un valor numérico establecido se comprueba el cambio de unidades de temperatura, velocidad del viento, presión atmosférica, precipitación y distancia de visibilidad.

5.2.2. Pruebas de integración

Para las pruebas de integración se emplean de dependencias falsas, también conocidos como mocks. La principal ventaja de su uso es que permiten simular dependencias reales sin que los demás componentes se den cuenta. Esto permite hacer pruebas con datos concretos, no dependiendo de la respuesta de un tercero, como podría ser el servidor meteorológico.

Una dependencia falsa puede substituir el objeto que se encarga de realizar la petición de datos meteorológicos al servidor por un objeto que cargase los datos desde un repositorio local, haciendo las pruebas más rápidas y con datos conocidos a priori. Esta técnica es completamente transparente para la clase o métodos que se están testeando.

La inyección de dependencias falsas se puede realizar de formar fácil gracias al diseño del modelo, en que las dependencias se inyectan en una clase de alto nivel llamada CentralPanel, explicada en el capítulo 4 de Análisis.

En la Figura 5.12 se muestra un diagrama de clases con las clases usadas para realizar la dependencia falsa. En este caso se ha substituido la clase Weatherbit, encargada de manejar las llamadas al servidor meteorológico Weatherbit y gestionar los datos de la respuesta JSON, por Weatherbit_Mock, que carga los datos directamente de un fichero local JSON. Como se puede ver, la nueva clase implementa la interfaz API, lo que hará posible la inyección de la dependencia en la clase CebtralPanel.

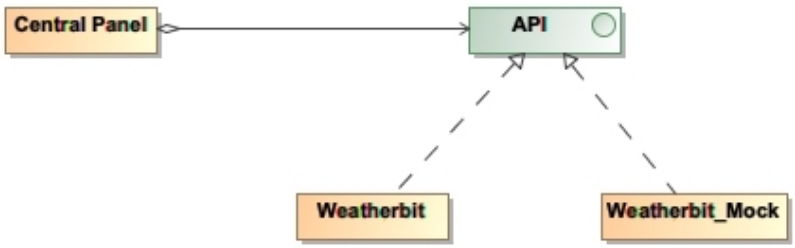

Figura 5.12: Inyección de dependencias falsas para los test de integración.

En el Cuadro 5.2 se describen los escenarios testeados e historias implicadas en cada uno.

Cuadro 5.2: Escenarios testeados en las pruebas de integración.

Escenario: El JSON del servidor contiene un mensaje de error. **Historias testeadas:** HU01, 2, 3, 4, 7. **Comentario:** Se carga de un fichero local un JSON con un mensaje cualquiera, pero sin datos estructurados tal cual espera el programa.
Escenario: Se cargan todos los datos de un *Estado Meteorológico* y se aplican cambios de temperatura. **Historias testeadas:** HU01 **Comentario:** Se carga de un fichero local un JSON un pronóstico y se comprueba que los parámetros coincidan una vez aplicado un cambio de unidades para la temperatura, velocidad del viento, visibilidad, precipitación acumulada y presión.
Escenario: Creación de una localización y obtención del estado actual meteorológico. **Historias testeadas:** HU01, 8 **Comentario:** Se crea una nueva localización y se hace una petición al servidor para obtener los datos. Se comprueba que lleve los parámetros de temperatura, temperatura mínima, temperatura máxima, velocidad del viento e icono.

5.2.3. Pruebas manuales

En el desarrollo del proyecto se realizan pruebas manuales, no automatizadas. El motivo de su realización es comprobar funcionalidades muy básicas o los aspectos relacionados con utilización de componentes que proporciona Swift. A continuación, se describen las pruebas realizadas y el motivo por el que no se han automatizado.

- Comprobación de los botones y translaciones de animaciones entre pantallas: se utilizan funciones y animaciones que proporciona Swift por defecto.
- Comprobación del pin para establecer la nueva localización: se comprueba que el nombre de localización que sugiere tiene sentido y el funcionamiento del *zoom* es correcto. La funcionalidad de estos métodos hace uso de métodos internos.
- Comprobar los límites de los colores de las celdas de la pantalla Tabla: se calcula el color a partir de un gradiente y se establecen unos límites para los valores meteorológicos. Por ejemplo, todos los valores inferiores a -30 o superiores a 45 muestran el mismo color. Se comprueba el funcionamiento con diferentes valores. También se comprueba que la visualización de la etiqueta se pueda leer con los diferentes colores de fondo.
- Comprobación de la dimensión de todas las vistas en los diferentes tipos de pantalla de iPhone e iPad.

Capítulo 6

Conclusiones

En este apartado expongo las conclusiones obtenidas del desarrollo de este proyecto desde un punto de vista personal y técnico, así como las posibles mejoras.

Resultados del proyecto

Para el desarrollo de este proyecto se contaba con 300 horas de estancia en prácticas en la empresa, en las que se debía realizar, a parte de la implementación y las pruebas, la fase de especificación para aclarar los objetivos. El tiempo era muy limitado y pese a que no se han implementado todas las historias de usuario, el resultado es positivo.

La parte del proyecto que ha quedado más incompleta ha sido la fase de pruebas. La mayoría de historias de usuario y tareas que han quedado en la pila del producto están relacionados con tests a realizar sobre el código. Además, las pruebas diseñadas no proporcionan suficientes garantías para su puesta en producción. Se deberían realizar más test y evaluar más escenarios, antes de publicar la actualización de PhotoPills con la *píldora* nueva. A pesar de ello, las pruebas realizadas son suficientes para garantizar el correcto funcionamiento de la aplicación, si bien para ponerla en producción se deberían realizar más pruebas.

Además de las pruebas, la interfaz de usuario actual no es la definitiva. El diseño de estas *interfaces* las ha hecho el estudiante de prácticas y no el diseñador de Photopills, S.L. Se ha intentado copiar el aspecto de las otras *píldoras*, pero el resultado obtenido, aunque cumple la funcionalidad, no transmite la apariencia de una aplicación profesional.

Uno de las claves desde el principio era hacer que esta aplicación mostrase información que fuese relevante para los fotógrafos, y hacerla "diferencial"de otras aplicaciones meteorológicas de uso general. Para ello se empleó tiempo investigando los parámetros que podían ser de interés para la planificación de las fotografías y los servidores que podían proporcionar esta información. En mi opinión, se ha cumplido satisfactoriamente este requisito.

Hasta la fecha la aplicación PhotoPills ha mostrado información como la trayectorias del sol, el tipo de luz en un momento determinado, la posición de la vía láctea según unas coordenadas,

etc. Esta información se calcula a partir de fórmulas y se puede garantizar que es correcta. Con la *píldora* de Meteorología no es así, la información meteorológica que se proporciona al usuario proviene de un servidor contratado y no se puede asegurar con certeza que se vaya a producir los fenómenos meteorológicos que se muestran en la *píldora*.

Aunque el usuario debería entender que no hay garantías en la información meteorológica, en alguna interfaz se podría informar de la empresa que proporciona esta información, con el objetivo de evadir la responsabilidad de PhotoPills en caso de que el pronóstico meteorológico no se cumpla.

Posibles mejoras

Pese a que la *píldora* de Meteorología cumple con las funcionalidades previstas inicialmente, se podrían añadir otras que la mejorarían.

Una de estas mejoras seria mejorar la pantalla del mapa meteorológico. En esta se muestran sobre el mapa los datos del estado actual del tiempo y no se dispone de ninguna opción para ver el pronóstico. Se podría mejorar la funcionalidad añadiendo la posibilidad de consultar el mapa en una hora determinada y así el usuario vería la evolución de previsión.

La aplicación PhotoPills cuenta con una lista de localizaciones, conocida como *puntos de interés*. Una posible mejora consistiría en integrar estos puntos de interés a la lista de localizaciones de la *píldora* de Meteorología. De esta forma, los usuarios podrían consultar la meteorología de estos puntos de interés sin tener que crear una localización en la *píldora* de Meteorología en el mismo lugar.

Conclusiones personales

Este proyecto ha supuesto un reto para mí. Pese a que unos meses antes de empezar la estancia en prácticas hice un curso sobre programación de aplicaciones iOS con Swift, no tenía mucha experiencia. En las primeras iteraciones me costaba trabajar con Xcode, un entorno de desarrollo que no había usado antes, y programar con Swift. Poco a poco fui resolviendo las dudas que tenía, con la ayuda del supervisor, Germán, y sobre todo consultando en Internet.

Haber realizado las prácticas en Photopills, S.L. ha supuesto una gran oportunidad. Como fotógrafo aficionado, hace varios años que uso PhotoPills y me considero muy fan de la marca. Esto ha sido una ventaja a la hora de realizar este proyecto, ya que no invertí nada de tiempo en el aprendizaje de la aplicación. Además de tener los conocimientos, el punto de vista fotográfico y el interés por este campo.

Haber podido desarrollar todas las fases del proyecto informático me ha permitido acabar de entender algunos conceptos que durante el grado he visto en diferentes asignaturas. Los conocimientos adquiridos sobre patrones de diseño en la asignatura EI1039, Diseño de Software, me han permitido crear una estructura más modular y desacoplada entre las diferentes clases. Además, para desarrollar el proyecto mediante la metodología Scrum he hecho uso de los co-

nocimientos aprendidos y las tareas realizadas en las asignaturas EI1032, Análisis de Software, y EI1050, Métodos Ágiles. Para el diseño y la realización de los diferentes test han sido útiles las competencias adquiridas en las asignatura EI1031, Validación y Verificación, y EI1048, Paradigmas del programario. Aunque estas asignaturas nombradas no han sido las únicas cuyos conocimientos han sido empleados, hay muchas otras que han sido fundamentales para poder tener las habilidades necesarias para desarrollar este proyecto.

La experiencia de realizar las prácticas a distancia ha sido positiva, aunque en muchas ocasiones hubiese preferido estar en una oficina con más gente. Primero, el redimiendo de trabajo creo que seria mayor rodeado de gente que está trabajando. Segundo y más importante, por el factor social. No solo por aprender de más gente, sino por poder relacionarse y hablar con los demás.

Este proyecto me ha permitido unir mi pasión por la fotografía y el desarrollo de software. Aprendiendo muchas cosas no solo a nivel técnico, sino también de fotografía y meteorología. Por ello estoy contento con el resultado obtenido, ha sido muy gratificante poder crear desde cero este producto.

Bibliografía

[1] Documentación aeris. https://www.weatherbit.io/api.

[2] Documentación dark sky. https://darksky.net/dev.

[3] Documentación open weather map. https://openweathermap.org/api.

[4] Documentación weatherbit. https://www.weatherbit.io/api.

[5] Documentación world weather online. https://www.worldweatheronline.com/developer/.

[6] Francesc Vilar Bonet Carles García Sellés. Manual de meteorología. la montaña. https://es.wikipedia.org/wiki/Prueba_unitaria.

[7] Core Data. Referencia core data. https://developer.apple.com/documentation/coredata.

[8] HTTP. Descripción del protocolo http. https://es.wikipedia.org/wiki/Protocolo_de_transferencia_de_hipertexto.

[9] HTTP. Salarios medios programador junior. https://www.indeed.es/salaries/Programador/a-junior-Salaries.

[10] Photopills. Web photopills. https://www.photopills.com.

[11] Wikipedia. Descripción api rest. https://es.wikipedia.org/wiki/Transferencia_de_Estado_Representacional.

[12] Wikipedia. Descripción pruebas unitarias. https://es.wikipedia.org/wiki/Prueba_unitaria.

[13] Wikipedia. Inyección de dependencias. https://es.wikipedia.org/wiki/Inyecci\unhbox\voidb@x\bgroup\let\unhbox\voidb@x\setbox\@tempboxa\hbox{o\global\mathchardef\accent@spacefactor\spacefactor}\accent19o\egroup\spacefactor\accent@spacefactor\futurelet\@let@token\penalty\@M\hskip\z@skipn_de_dependencias.