

Desarrollo de aplicación híbrida con Ionic: AirSun

José Ángel Hernández García-Gango

Máster universitario de Desarrollo de aplicaciones para dispositivos móviles
Informática, multimedia y telecomunicación

Eduard Martín Lineros
Carles Garrigues Olivella

01/2020



Esta obra está sujeta a una licencia de
Reconocimiento-NoComercial-SinObraDerivada [3.0](https://creativecommons.org/licenses/by-nc-nd/3.0/es/)
[España de Creative Commons](https://creativecommons.org/licenses/by-nc-nd/3.0/es/)

FICHA DEL TRABAJO FINAL

Título del trabajo:	<i>Descripción del trabajo</i>
Nombre del autor:	<i>José Ángel Hernández García-Gango</i>
Nombre del consultor/a:	<i>Eduard Martin Lineros</i>
Nombre del PRA:	<i>Carles Garrigues Olivella</i>
Fecha de entrega (mm/aaaa):	01/2020
Titulación:	Máster universitario de Desarrollo de aplicaciones para dispositivos móviles
Área del Trabajo Final:	Informática, multimedia y telecomunicación
Idioma del trabajo:	<i>Castellano</i>
Palabras clave	<i>Ionic, contaminación atmosférica, desarrollo híbrido</i>
<p>Resumen del Trabajo (máximo 250 palabras): <i>Con la finalidad, contexto de aplicación, metodología, resultados y conclusiones del trabajo.</i></p>	
<p>El cambio climático es un problema de orden mundial que ahora está más presente que nunca en nuestro día a día. Es raro el informativo televisivo que no emita alguna noticia relacionada con el cambio climático. Lo mismo sucede con los periódicos generalistas, los medios informativos online o la radio. Igualmente, la ciudadanía está cada vez más concienciada con el medio ambiente. Y es que, el cambio climático es un problema que nos afecta a todos.</p> <p>Una de las causas relacionadas con el cambio climático es la contaminación del aire. El hombre, mediante la actividad industrial y el desarrollo del transporte, ha contribuido al deterioro progresivo del aire que respira y, por otro lado, ha dañado gravemente la capa de ozono, escudo fundamental frente a la radiación ultravioleta del Sol. Nuestros pulmones y nuestra piel sufren la irresponsabilidad de nuestros actos. El problema que pretende resolver nuestra aplicación es la falta de información que padece el ciudadano respecto a la calidad del aire y la radiación UV. Queremos ofrecer una información clara, actual e inmediata de dos factores que afectan directamente a nuestra salud cada vez que salimos a la calle.</p> <p>La presente memoria describe el proceso seguido para el desarrollo de la aplicación móvil AirSun. Se trata de una app híbrida multidispositivo desarrollada en Ionic. La aplicación recurrirá a servicios web públicos, así como al GPS del terminal, para ofrecer de forma inmediata una información clara e inequívoca de la contaminación del aire y del índice de radiación UV.</p>	

Abstract (in English, 250 words or less):

Climate change is a world-wide problem that is now more present than ever in our daily lives. It is uncommon to watch any news program on tv that does not broadcast any piece of news related to climate change. The same goes for general press, online media or radio. Also, citizens are increasingly aware of the environment. The truth is that climate change is a problem that affects all of us.

One of the causes related to climate change is air pollution. Man, through industrial activity and the development of transport, has contributed to the progressive deterioration of the air he breathes and, on the other hand, has severely damaged the ozone layer, a fundamental shield against ultraviolet radiation from the Sun. Our lungs and our skin suffer the irresponsibility of our actions. The problem that our application attempts to solve is the lack of information for citizens regarding air quality and UV radiation. We want to offer clear, current and immediate information on two factors that directly affect our health every time we go out.

This report describes the process followed for the development of the AirSun mobile application, a multi-device hybrid app developed at Ionic. The application will use public web services, as well as the terminal's GPS, to immediately provide clear and unambiguous information on air pollution and the UV radiation index.

Índice

1. Introducción.....	1
1.1 Contexto y justificación del Trabajo.....	1
1.2 Objetivos del Trabajo.....	3
1.3 Enfoque y método seguido.....	4
1.4 Planificación del Trabajo.....	6
1.5 Breve resumen de productos obtenidos.....	8
1.6 Breve descripción de los otros capítulos de la memoria.....	8
2. Análisis de usuarios y de competencia.....	10
2.1 Personas.....	10
2.2 Escenarios.....	15
2.3 Customer Journey Map.....	17
2.4 Competencia.....	19
3. Diseño y prototipado.....	22
3.1 Funcionalidades detectadas.....	22
3.2 Bocetos.....	24
3.3 Prototipos de baja fidelidad.....	30
3.4 Prototipos de alta fidelidad.....	34
4. Evaluación.....	39
4.1 Casos de uso.....	39
4.2 Diseño de la arquitectura.....	44
5. Implementación.....	45
5.1 Proyecto de Ionic.....	45
5.2 Pantallas.....	47
5.2.1 Principa.....	47
5.2.2 Nueva alerta.....	48
5.2.3 Lista de alertas.....	50
5.2.4 Editar alerta.....	52
5.3 Servicios.....	53
5.3.1 Web services.....	53
5.3.2 Servicio de almacenamiento local.....	54
5.3.3 Servicio de utilidades.....	55
5.4 Estado del proyecto.....	56
5.5 Actualización del estado del proyecto.....	57
6. Pruebas.....	60
6.1 Actualización de pruebas.....	69
7. Conclusiones.....	71
8. Glosario.....	73
9. Bibliografía.....	75
10. Anexos.....	77
10.1 Notas.....	77
10.2 Prerrequisitos para despliegue en un servidor local.....	77
10.3 Instalación y despliegue en navegador.....	77
10.4 Compilación del archivo APK.....	77
10.5 Manual de usuario.....	78

1. Introducción

1.1 Contexto y justificación del Trabajo

Según datos de la Organización Mundial de la Salud (OMS) “la contaminación atmosférica en las ciudades y zonas rurales de todo el mundo provoca cada año 4,2 millones de defunciones prematuras” [1](Organización Mundial de la Salud, 2018). Nuestros cielos se han vuelto nocivos. Cada vez que salimos a la calle ponemos en riesgo nuestra salud. Y es que la contaminación del aire que respiramos es la principal causante de una gran variedad de enfermedades: cánceres de pulmón, accidentes cerebrovasculares, asma, EPOC y otras afecciones respiratorias.

Si hablamos de la contaminación del aire también nos vemos obligados a mencionar un problema estrechamente relacionado, que preocupa desde los años setenta del pasado siglo: el deterioro de la capa de ozono. La capa de ozono es un escudo contra la radiación ultravioleta que proviene del Sol. El debilitamiento de esta barrera natural aumenta el efecto pernicioso que los rayos solares tienen sobre nuestro cuerpo. El aumento de los cánceres de piel o el de las enfermedades oculares relacionadas con la luz solar son algunos de los efectos relacionados con el deterioro de la capa de ozono. De nuevo la OMS nos poner en alerta, ya que la radiación ultravioleta solar es la “causante de 60 000 muertes prematuras” cada año [2](Organización Mundial de la Salud, 2006).

Los problemas derivados de la contaminación atmosférica son una preocupación mayor en las sociedades más desarrolladas. Cuestiones como el calentamiento global, el deterioro de la capa de ozono o el cambio climático —distintas caras del mismo problema— están de actualidad y copan un buen porcentaje de tiempo y extensión en los distintos espacios informativos. Recientemente, la joven activista medioambiental sueca Greta Thunberg sonrojaba a los principales dirigentes mundiales en un clamor contra su pasividad ante la problemática del cambio climático [3](La Vanguardia, 2019).

La concienciación es cada vez mayor en la ciudadanía. Un estudio reciente del Real Instituto Elcano señala que “la mayoría de los españoles piensa que el cambio climático es la principal amenaza del mundo” [4](El País, 2019b). La gente está concienciada frente a la crisis medioambiental de nuestro tiempo y la confianza en los científicos y en la ciencia es plena.

Por fortuna, la preocupación comienza a permear también en el ámbito político y, muy recientemente, el Consejo de Ministros aprobó un programa para reducir la contaminación atmosférica durante los próximos años y cumplir, de este modo, con las directrices de la Unión Europea [5](El País, 2019a).

Hay un estado de alerta permanente y una concienciación social generalizada. Podríamos aventurarnos a pensar que esta preocupación por el medio ambiente lleva al ciudadano a desear estar mejor informado de la calidad de aquello que respira cada día y de las posibles consecuencias que pueda tener para su salud. Es en este momento donde se nos manifiesta una necesidad a cubrir, una

necesidad, en esencia, informativa. ¿Qué es lo que respiro? ¿Empeorará mi asma si salgo al exterior? ¿Los rayos ultravioletas serán especialmente agresivos para mi piel? ¿Es recomendable que hoy me quede en casa?... Este tipo de preguntas es al que intentará responder de forma sencilla, rápida y clara la aplicación móvil que queremos desarrollar.

No nos proponemos innovar, ni crear un producto totalmente novedoso y revolucionario. En la actualidad existen páginas web y aplicaciones móviles de consulta que ya nos proporcionan información acerca de múltiples índices y factores relacionados con la calidad del aire y la luz solar. Entre ellas, podemos destacar *AirVisual*, una app de referencia en lo que a información sobre polución atmosférica se refiere. Pero también hay otras opciones en el mercado de aplicaciones móviles igual de válidas. Nos detendremos a analizar algunas de ellas en mayor detalle en el siguiente capítulo.

Pero, entonces, ¿qué es lo novedoso que queremos aportar ante una necesidad que parece cubierta? Si probamos con detenimiento la mayoría de estas aplicaciones nos daremos cuenta de que tienden a sobrecargar la información que muestran en pantalla, es decir, abruman al usuario con gran cantidad de datos. En este sentido, nuestra ventaja competitiva se basará en la diferenciación del producto. Nuestra app ofrecerá información sobre la calidad del aire y el índice de rayos ultravioleta; pero lo hará desde una aproximación diferente al problema.

Queremos proporcionar al usuario final una forma simple e inequívoca de cómo de saludable es el aire exterior del lugar en el que se encuentra. El acceso a esa información ha de ser rápido, directo, visual y no ha de abrumar con una retahíla de datos técnicos que, probablemente, no interese al usuario la mayoría de las ocasiones.

¿Hoy el aire exterior es bueno o está especialmente contaminado? ¿Ahora el índice de rayos ultravioleta es bajo o está por encima de la habitual? ¿Hoy es recomendable quedarse en casa o no hay inconveniente en salir al exterior? Nuestra aplicación quiere contestar a estas preguntas y quiere hacerlo de manera clara, directa e inmediata, mientras nos dirigimos a la puerta de casa para salir a la calle.

1.2 Objetivos del Trabajo

La aplicación móvil que proponemos desarrollar deberá cumplir con unos requerimientos funcionales y no funcionales básicos. Dichos requerimientos fundamentales podrán incrementarse a lo largo del desarrollo, según se reevalúen las necesidades iniciales.

Ref.	Requisito funcional
RF. 1	El sistema deberá mostrar de forma directa y sencilla el índice de contaminación del aire que hay en ese instante en el exterior del lugar en el que se encuentre.
RF. 2	El sistema deberá mostrar de forma directa y sencilla el índice de radiación ultravioleta que hay en ese instante en el exterior del lugar en el que se encuentre.
RF. 3	El usuario deberá poder recibir la información de una forma visual mediante código de colores significativo , que se correspondan con el grado de calidad del aire y/o el índice de radiación UV.
RF. 4	El sistema deberá enviar notificaciones automáticas al usuario en el caso en que los índices de calidad del aire o radiación UV sean especialmente perniciosos en un momento dado.
RF. 5	El sistema deberá geolocalizar automáticamente la posición del dispositivo en el momento de la consulta, minimizando la necesidad de intervención del usuario.
RF. 6	Durante la obtención de la posición y la consulta de los servicios web el sistema presentará en pantalla recomendaciones y consejos de salud para el usuario, amenizando la breve espera.

Ref.	Requisito no funcional
RNF. 1	La aplicación deberá poder instalarse en cualquier dispositivo móvil que cuente con la última versión de los sistemas operativos iOS o Android .
RNF. 2	El sistema deberá tener acceso al hardware de geolocalización del dispositivo para la obtención de la posición.
RNF. 3	El sistema deberá tener acceso a Internet para la consulta de los servicios que proporcionarán los datos.
RNF. 4	El sistema deberá ser fácil de usar, ágil en la recuperación y presentación de los datos, y claro en la información que ofrece al usuario.

1.3 Enfoque y método seguido

Hemos considerado a conciencia tanto la metodología como la estrategia a emplear en el desarrollo, valorando los pros y los contras de las diferentes opciones que teníamos a nuestro alcance para abordar el proyecto. Creemos que —atendiendo al tipo de aplicación y las circunstancias que rodearán su desarrollo— la mejor opción, en este caso, es recurrir a una metodología clásica de desarrollo *waterfall* y a una estrategia de desarrollo híbrido, que aproveche la flexibilidad que ofrecen las tecnologías web disfrutando, a su vez, de una experiencia cercana a la nativa.

¿Desarrollo web, nativo o híbrido?

Existen tres tipos principales de aplicaciones móviles: la aplicación web, la nativa y la híbrida. Como es lógico, cada una de ellas ofrece sus ventajas y sus contrapartidas.

Por un lado, tenemos las aplicaciones móviles basadas en las tecnologías web. Destacan por su facilidad y rapidez en el desarrollo. Por contra, su rendimiento es inferior al de una aplicación nativa y adolece de importantes limitaciones para acceder a las funcionalidades del hardware del dispositivo.

En el otro extremo encontramos las aplicaciones móviles nativas. Ofrecen la experiencia más fluida, muestran la apariencia propia de las guías de diseño de cada plataforma y permiten acceder a todos los recursos de hardware que ofrece el dispositivo móvil. Sin embargo, el coste de desarrollo es mucho mayor al tener que utilizar un lenguaje específico para cada plataforma, lo que implica tener que programar por separado para cada una de ellas.

Entre ambas tenemos la opción que hemos escogido para el desarrollo de nuestra aplicación. Nos referimos a las aplicaciones híbridas. Esta clase de desarrollo aúna lo mejor de las aplicaciones web y de las nativas. En el desarrollo híbrido recurrimos a tecnologías web como HTML, CSS y JavaScript, con el apoyo de algún framework; pero tenemos acceso a funcionalidades propias de las aplicaciones nativas, sin la necesidad de desarrollar varias veces para cada plataforma.

Nuestra aplicación es relativamente sencilla y no utiliza gráficos complejos ni recurre a procesos que necesiten un rendimiento elevado de hardware, por lo que no es necesario un desarrollo nativo. Además, deseamos que la aplicación sea multiplataforma y esté disponible tanto para el sistema operativo iOS como para Android. Por otro lado, los plazos para el desarrollo del proyecto son cortos y la fecha de entrega es inamovible. Considerando estas circunstancias, creemos que la mejor opción es un desarrollo híbrido con Ionic. Además, contamos con la ventaja de que ya poseemos experiencia previa en esta plataforma orientada al desarrollo de aplicaciones móviles híbridas.

¿Metodología *agile* o *waterfall*?

A la hora de adoptar una u otra metodología para el desarrollo nos ha sucedido algo parecido a la decisión sobre el tipo de aplicación. Tanto el desarrollo clásico

en cascada como la metodología agile más actual, cuentan con unas características muy marcadas, y la decisión de utilizar una u otra dependerá del tipo de desarrollo y las circunstancias que rodean al proyecto en el que nos embarquemos.

El desarrollo en cascada o *waterfall* es el desarrollo lineal tradicional. Tenemos unas fases bien definidas y no se suele comenzar una nueva hasta no tener completamente cerrada la anterior. Por lo general, tenemos una toma de requisitos bien definidos al inicio, una etapa de diseño, un desarrollo y, por último, una fase de pruebas antes de tener el producto listo para su distribución.

La metodología *agile* y sus diferentes ramas (*Scrum*, *Kanban*, etc.) está enfocada a lograr productos con mayor rapidez, sin dedicarle demasiado tiempo a la toma de requisitos y obteniendo un producto listo para producción después de cada iteración. Este método cíclico de iteraciones rápidas es el que está de moda actualmente, especialmente en desarrollos grandes.

Pero, en ese caso, ¿es la metodología *agile* la mejor para todos los desarrollos? Creemos que no necesariamente. Los métodos *agile* son muy recomendados en proyectos grandes y complejos, con unos requisitos que no hayan quedado completamente definidos de inicio, en los que existan muchas posibilidades de que el cliente introduzca modificaciones sobre la marcha. En cambio, la metodología *waterfall* se adapta muy bien a proyectos más pequeños, con menos desarrolladores, con unos requisitos muy claros y definidos desde el inicio, en los que no estén previstos grandes cambios durante el desarrollo, proyectos con un alcance y una duración bien determinada [6](Suzanna Haworth, 2019).

Nuestra aplicación móvil, por sus características y las circunstancias que van a rodear su desarrollo, responde mejor a una metodología *waterfall*. Es una aplicación que no excesiva complejidad; no hay gran cantidad de requisitos funcionales y quedarán bien definidos en las primeras fases del trabajo; el jefe de proyecto, diseñador, analista y desarrollador van a ser la misma persona; y disponemos de un plazo de tiempo muy limitado para todo el proceso y una fecha fija e inamovible para la entrega del producto final. Por tanto, el método de desarrollo en cascada es el más acertado para nuestro caso. Eso no significa que debamos asumir una versión plenamente ortodoxa y pura de una metodología en concreto. Durante todo el proceso de trabajo puede surgir la necesidad de realizar pequeñas revisiones de etapas ya finalizadas, modificaciones menores que por obligación u oportunidad deban introducirse, pero desde una perspectiva general del proyecto el método de desarrollo en cascada es el que mejor se adapta a nuestro trabajo.

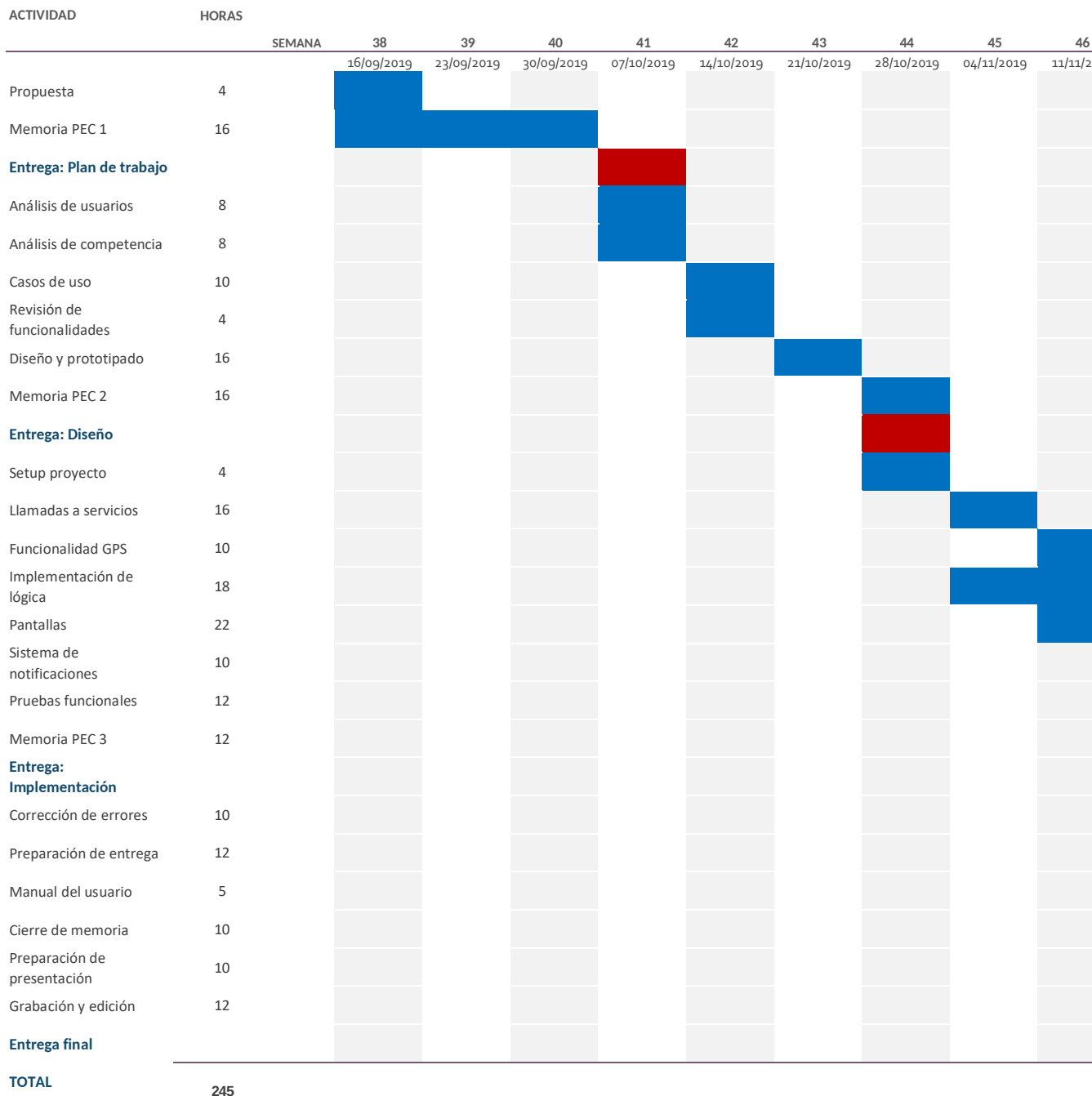
1.4 Planificación del Trabajo

Para la planificación del proyecto se han tenido en cuenta las fases en las que suele dividirse un desarrollo tradicional y las fechas de entrega establecidas para los diferentes hitos del proyecto.

Se ha procurado realizar una distribución de horas equilibrada a lo largo de las semanas que durará la realización del proyecto, pero no siempre ha sido posible o, al menos, no del todo. Creemos que es lógico que la fase de implementación tenga una carga de trabajo ligeramente superior a otras fases del proyecto. A esto hay que sumar que durante el proceso de implementación hay que ir elaborando en paralelo la memoria del proyecto.

No obstante, a pesar de que hay momentos con una carga de trabajo algo más elevada, creemos que la planificación que a continuación presentamos es viable y asumible por un solo desarrollador.

Planificación del proyecto



1.5 Breve resumen de productos obtenidos

Tras la conclusión del trabajo obtendremos tres productos:

- **Memoria.** El presente documento. En ella vendrá detallado todo el proceso de trabajo realizado durante todas las fases del desarrollo de la aplicación móvil.
- **Presentación.** Una breve presentación en vídeo del proyecto. Mediante recursos audiovisuales (diapositivas, voz en off, vídeo de demostración)

funcional, etc.) se intentará transmitir una idea clara y concisa de los aspectos más destacados del desarrollo y de la aplicación resultante.

- **Aplicación móvil.** Se trata del producto principal del proyecto, alrededor del que giran los dos anteriores. Al finalizar el trabajo obtendremos una aplicación móvil para iOS y Android. Esta aplicación, recurriendo al GPS del dispositivo y obteniendo información de *web services*, indicará al usuario de manera sencilla la calidad del aire y el índice de rayos UV que hay en el exterior del lugar en el que se encuentre.

1.6 Breve descripción de los otros capítulos de la memoria

Durante los próximos capítulos analizaremos pormenorizadamente las diferentes fases de la que consta el desarrollo de nuestro proyecto.

Primero, realizaremos un estudio de la competencia a la que se enfrenta nuestro producto en los mercados de aplicaciones más habituales. Analizaremos las mejores alternativas de software que actualmente ofrecen una funcionalidad similar a la aplicación móvil que pretendemos desarrollar. Veremos algunos de sus puntos fuertes pero, también, algunas de las debilidades que hemos detectado y que pretendemos explotar para crear una aplicación que ofrezca una clara ventaja competitiva respecto a las alternativas y, de este modo, convertirla en una opción atractiva para nuestros posibles usuarios.

Estos usuarios son los que abordaremos a continuación en nuestro estudio. Reflexionaremos sobre el público objetivo e imaginaremos algunos escenarios en los que nuestra aplicación pueda ser una herramienta de utilidad para estas personas. Además, elaboraremos algunos casos de uso que nos ayuden a entender mejor las necesidades de nuestros futuros usuarios y comprobar que la implementación de los requerimientos analizados con anterioridad satisfará tales necesidades.

Posteriormente, nos adentraremos en el proceso de diseño. Realizaremos algunos prototipos de distinta complejidad (*sketches*, *wireframes*, prototipos...) Aunque, si durante el proceso observamos que, atendiendo al nivel de complejidad del proyecto, no es necesaria la realización de alguno de estos productos y que podemos invertir ese tiempo en otras fases que necesiten mayor dedicación, procederemos en consecuencia. El objetivo es disponer de una guía que nos facilite la posterior implementación.

A continuación, entraremos en la fase de implementación. Los capítulos dedicados a esta etapa del proyecto abordarán cuestiones técnicas tales como la implementación de las llamadas a los *web services* públicos que nos proporcionarán la información, el acceso al GPS interno del dispositivo, la implementación de los procesos de notificaciones automáticas al usuario, el desarrollo de la lógica interna que procesará estos datos y los mostrará al usuario en pantalla, el desarrollo de estas pantallas... así como cualquier inconveniente que haya podido surgir durante el proceso.

Por último, dedicaremos las páginas finales de esta memoria a aquellos aspectos del proceso que hayamos podido pasar por alto, así como las conclusiones que extraigamos de toda esta experiencia.

2. Análisis de usuarios y de competencia

Nuestra aplicación se fundamentará en la filosofía de diseño centrado en el usuario (DCU) No sólo pretendemos resolver una necesidad de nuestro usuario objetivo, además queremos que la experiencia que tenga durante la utilización del producto sea plenamente satisfactoria. Con el objetivo de lograrlo vamos a recurrir a una serie de técnicas contrastadas durante el proceso de diseño. Antes siquiera de tomar un lápiz para elaborar el primer boceto, para asentar una idea sólo del producto que vamos a diseñar nos ayudaremos de métodos propios de DCU. Personas, escenarios, casos de uso, *journey maps*... son herramientas

fundamentales en la filosofía del DCU.

Para finalizar el capítulo, repasaremos algunas de las alternativas que ya ofrece el mercado a la hora de abordar el problema que pretendemos resolver con nuestra aplicación. Veremos los pros y los contras de estas propuestas, estudiaremos cómo poder mejorarlas y analizaremos la ventaja competitiva que tendrá respecto a la competencia. Siempre con el usuario presente en primer término en cada una de las decisiones que tomemos durante el proceso.

2.1 Personas

Una de las técnicas fundamentales en el DCU es la persona. Con ellas modelamos perfiles de usuario, que nos permiten tener una idea más clara de sus comportamientos, expectativas y actitudes ante el problema que desean resolver. La elaboración de las fichas de persona se aborda durante las primeras fases del diseño, antes de cualquier intento de elaborar un diseño o un prototipo. Durante esta fase, la ficha de persona nos facilita mantener la perspectiva del usuario durante el proceso, así como la forma que tendrá el usuario de utilizar el producto [7] (Nielsen, n.d.).

La información que podemos extraer de la persona es muy valiosa durante el proceso de diseño. Por esta razón, para el desarrollo de nuestra aplicación hemos elaborado tres fichas de persona. Todas ellas han sido creadas después de una labor previa de recopilación de información acerca de nuestros usuarios objetivos, mediante las técnicas de encuesta y entrevista personal. El resultado obtenido con estas fichas de persona nos facilitará el trabajo a la hora de abordar las siguientes fases del proceso de diseño.

Persona 1

Nombre: Luis Hernández Romero.

Profesión: Médico de familia.

Edad y estado civil: 35 años. Soltero

Descripción: Luis es un joven médico de salud al norte de la Comunidad de Madrid. Vive con sus padres a una vivienda unifamiliar cerca de la ciudad.

Adora su profesión y el trato diario con los pacientes. Le gusta salir a correr o a realizar alguna actividad física. Luis disfruta de la vida tranquila, rodeado de naturaleza. Es muy concienciado con los problemas medioambientales, especialmente una enfermedad respiratoria agravada por la contaminación.

Otra de las grandes pasiones de Luis es la música. Le gusta un poco el piano. Su gusto es muy variado, desde una banda de pop *indie* como de una banda de jazz hasta la música de su padre.



Necesidades: Como médico y amante de la naturaleza, Luis se preocupa por el aire que respira. Esto, junto a la enfermedad de su padre, fue el motivo que le llevó a buscar trabajo alejado de la ciudad.

Sin embargo, su pasión por la música le anima a acercarse a la ciudad los fines de semana, en busca de una oferta de ocio que no puede encontrar en su pequeña localidad.

Suele acudir solo a locales donde ponen la música que a él le gusta o con su padre al Auditorio Nacional a disfrutar de una obra clásica. Pero le encantaría disponer de una aplicación móvil que le alertara cuando los niveles de contaminación atmosférica son especialmente perjudiciales. De este modo, disfrutaría con más tranquilidad de sus escapadas a la ciudad.

Conocimientos tecnológicos: Como la mayoría de la gente de su generación, Luis está acostumbrado a utilizar la tecnología en su día a día. En la consulta, aunque se trate de un pequeño centro de salud, ya todo está informatizado. Por tanto, utiliza el ordenador en el trabajo y —ya en casa— emplea un *tablet* por comodidad.

Como la música es su gran afición, está suscrito a *Spotify* y *Tidal*. Siempre lleva las dos aplicaciones en su iPhone, en el *tablet* y en su smartwatch y, en conclusión, tiene soltura en el uso y la descarga de apps en dispositivos móviles.

Persona 2

Nombre: Marta Torrent Sainz.

Profesión: Community Manager.

Edad y estado civil: 37 años.
Soltera.

Descripción: Marta es licenciada en Periodismo. Está soltera y vive sola, con su perro, en pequeño un piso de alquiler en el centro de Barcelona.

Trabaja desde casa, a media jornada, como *community manager* en una compañía de desarrollo de videojuegos. Además, obtiene un sobre sueldo escribiendo, de vez en cuando, para un par de blogs de tecnología.



Cuando termina de trabajar cocina algo rápido para comer y a continuación saca a su perro a pasear. El resto de la tarde la dedica a descansar en el sofá mientras ve un capítulo de alguna serie de *Netflix* en su Smart Tv.

Los fines de semana suele ir de excursión por el campo. Marta padece asma y pasear por un entorno natural, en el que abunde el aire fresco y alejado de la contaminación de la ciudad, le sienta muy bien para su salud.

Necesidades: Marta procuró encontrar un empleo que le permitiese trabajar la mayor parte del tiempo desde casa. En su anterior empleo tardaba una hora en llegar a la oficina. En total, dos horas al día de transporte público y tráfico saturado, respirando un aire altamente contaminado. En ocasiones, normalmente los días de lluvia, su trayecto transcurría con normalidad. Pero otras veces, la contaminación atmosférica era tal que su problema de asma se resentía y el trayecto hasta su trabajo era una odisea.

Por suerte, desde hace unos años trabaja desde la tranquilidad de su dormitorio y evita la calle de la ciudad en la medida de lo posible. Sin embargo, Marta tiene un perro al que adora y por él está dispuesta a sacrificarse unos minutos al día para poder sacarlo a pasear.

Marta es una apasionada de la tecnología y no se separa de su *smartphone* en ningún momento. Por eso, le encantaría disponer de una aplicación que a la

hora de salir a la calle le informase de forma clara y sencilla cuáles son los niveles de contaminación atmosférica en la ciudad. De esta manera, podría elegir el momento ideal para pasear a su perro y salir a la calle mucho más tranquila.

Conocimientos tecnológicos: Marta trabaja en el mundo de Internet y las redes sociales. La tecnología está presente en su día a día. Su ordenador de escritorio, su Mac Book, su móvil o su iPad, son objetos cotidianos que utiliza tanto para el trabajo como para el ocio. Está habituada al uso de plataformas de contenidos digitales bajo demanda como *Netflix* y de vez en cuando compra aplicaciones para su móvil o su *tablet* en el correspondiente “store”.

Persona 3

Nombre: Marc Martín Cabanillas.

Profesión: Profesor de universidad.

Edad y estado civil: 54 años.
Casado.

Descripción: Marc es profesor de derecho en la Universidad Autónoma de Madrid. Vive con su mujer y su hija en un piso situado en un barrio al sur de la ciudad.

No dispone de mucho tiempo libre. Las clases en la facultad ocupan toda su mañana y por la tarde, cuando llega a casa, la mayoría de las veces sigue trabajando corrigiendo los ejercicios de sus alumnos y preparando la lección del día siguiente. No obstante, todas las noches procura encontrar algo de tiempo para su mayor afición: la literatura rusa del siglo XXI.

Marc es de tez muy clara y de niño le diagnosticaron fotosensibilidad. Tiene una piel muy delicada y siempre procura tomar las medidas oportunas para cuidarla. Siempre que va al campo o la playa lleva su gorra, una loción de elevado factor de protección y no se quita la camiseta hasta que se mete en el agua. Tiene revisiones periódicas en el dermatólogo y vigila cada lunar nuevo que detecta, porque sabe que es mejor prevenir que cuidar.

Necesidades: La mayor parte de los días de entre semana los pasa en la facultad o en casa, trabajando o dedicando su poco tiempo de ocio a las letras. No es de extrañar que llegado el fin de semana Marc quiera hacer algo



diferente. Además, su mujer y su hija siempre le animan a salir de casa y así poder disfrutar de algún plan en familia.

Marc siempre sale protegido al exterior, especialmente los días más soleados y, sobre todo, si realiza alguna escapada a la naturaleza. Marc estaría muy interesado en tener alguna forma rápida y sencilla de conocer el índice de rayos ultravioleta en un momento dado, así como recibir avisos si el índice alcanza valores demasiado elevados. Sabiendo estos datos, él y su piel disfrutarían con mayor tranquilidad del campo.

Conocimientos tecnológicos: Marc utiliza un ordenador portátil con asiduidad. Lo necesita, sobre todo, para su actividad laboral. Siempre lleva las presentaciones de cada lección en el portátil y en casa utiliza el ordenador para revisar el correo de la universidad y subir las calificaciones de sus alumnos a la web de la asignatura.

Pero más allá del típico paquete de ofimática, Marc no utiliza el ordenador para gran cosa. En general, no le interesa demasiado la tecnología y aunque dispone de un *smartphone* lo lleva, principalmente, para estar localizado.

2.2 Escenarios

Una vez que tenemos una aproximación de nuestro usuario real mediante la técnica de diseño de personas podemos centrar nuestra atención en los

escenarios de uso. Con el escenario conseguimos describir "de manera narrativa cómo utiliza un usuario el producto para lograr sus objetivos" [8] (Universitat Oberta & Catalunya, n.d.)

Las historias que surgen tras la descripción del escenario nos permiten entender mejor la relación entre los arquetipos de usuarios definidos durante la creación de las personas, el sistema con el que interactúan y el contexto en el que se produce dicha interacción. Al definir los escenarios en los que la persona utilizará nuestro producto tendremos una visión mucho más clara y precisa de la forma en la que el usuario se relacionará con nuestra aplicación y de las necesidades que esperará ver satisfechas durante el proceso. Con esta nueva información es probable que se nos ocurran ideas que no habíamos contemplado. En definitiva, la elaboración de escenarios nos ayudará a definir mejor las funcionalidades que ha de tener nuestra aplicación y nos facilitará las siguientes fases del diseño.

Escenario 1

Es un viernes por la tarde y Luis recuerda de repente que al día siguiente es el cumpleaños de su padre. Se ha olvidado por completo. Durante toda la semana ha estado tan enfrascado en su trabajo en la clínica que no ha tenido tiempo para mirar un regalo para su padre. ¡Ni siquiera se acordaba!

Luis toma su tablet para buscar en Internet un buen regalo y enmendar el olvido. Ve en Amazon unos cds de música clásica que a buen seguro encantarán a su padre, pero teme que el regalo no llegue a tiempo. A fin y al cabo, vive algo alejado de la ciudad.

De pronto, tiene una idea. Los sábados por la tarde suelen ofrecer conciertos en el Auditorio Nacional. Rápidamente visita la página web y descubre que justo el día del cumpleaños de su padre está programada la Quinta Sinfonía de Mahler en el auditorio y, además, aún quedan algunas entradas. Con un regalo así su padre quedará encantado.

Pero Luis tiene miedo de que la enfermedad respiratoria de su padre se agrave con el aire contaminado de la ciudad. Como los últimos días ha llovido en Madrid no debería suponer un riesgo significativo ir a la ciudad. No obstante, Luis para quedar más tranquilo toma su *smartphone* y abre la aplicación que mide la contaminación atmosférica del aire. En la app, de forma muy sencilla y rápida, activa una alerta para que le notifique de forma automática si la contaminación atmosférica supera los valores que ha marcado. Esta medida preventiva le reporta seguridad. Ahora sabe que podrá disfrutar de un día con su padre en la ciudad, sin preocupaciones.

Escenario 2

Marta acaba de publicar el último post del día para el blog de tecnología en el que trabaja. Está cansada, afuera hace frío y le gustaría quedarse tirada en el sofá, viendo algo divertido en Netflix, arropada con su manta favorita. Pero su perro pasea melancólico por la casa, señal de que está desesperado por salir a la calle.

Marta decide ser responsable con su mascota. Va a su habitación a vestirse. En la cómoda ve su mascarilla protectora. ¿Habrá demasiada contaminación hoy en la atmósfera? ¿Es necesario llevar la mascarilla? A Marta no le da vergüenza pasear por la calle con su mascarilla contra la contaminación. Sabe que su salud respiratoria es lo primero. Pero últimamente, en el parque por el que suele pasear a su perro, se ha encontrado en varias ocasiones con el vecino que le gusta.

Marta se da cuenta de que está dándole demasiadas vueltas al asunto cuando su indecisión tiene fácil solución. Toma su iPad y abre la aplicación sobre el índice de contaminación atmosférica. En pocos segundos la app detecta automáticamente su localización y muestra en pantalla de forma clara y muy visual la calidad del aire en ese instante. Parece que esta tarde no se tendrá que preocupar demasiado por la contaminación atmosférica. Marta deja la mascarilla en casa y sale tranquila a la calle con su perro.

Escenario 3

Por fin ha llegado el fin de semana para Marc. Como se acerca la época de exámenes en la universidad, ha pasado toda la semana ocupado impartiendo las clases a sus grupos de alumnos y por las tardes preparando las pruebas finales del semestre. Han sido días agotadores, pero por fin ha llegado el sábado y Marc ha decidido pasar un día en el campo con su esposa y su hija.

La predicción meteorológica indica que será un día de claros y nubes en la zona de la sierra de Madrid a la que tienen pensado ir. No parece que vaya a ser demasiado elevado el índice de rayos ultravioletas el día de hoy. No obstante, antes de salir Marc se equipa con su gorra con visera, que casi le cubre todo el rostro, y se embadurna bien con protección solar.

Cuando llegan al punto en el que comienza la excursión, Marc observa con recelo que apenas hay nubes en el cielo. Aunque el sol no incide con fuerza, no se fía. Rápidamente, saca del bolsillo del pantalón su smartphone y abre la aplicación que informa acerca del índice de rayos UV. Aunque Marc no es muy diestro con la tecnología, la app es muy sencilla y en unos segundos localiza su posición y muestra en pantalla de forma clara el índice de rayos ultravioleta en ese momento. Parece que Marc no tendrá que preocuparse y que su piel no sufrirá ningún daño, sin embargo, para asegurarse, crea una alerta para que la

app notifique de forma automática si el índice aumenta de valor durante la jornada. Ahora, por fin, Marc puede disfrutar tranquilo del paseo por el campo.

2.3 Customer Journey Map

El *customer journey map* es una herramienta de *Design Thinking* que refleja de forma visual el proceso que realiza el usuario desde que experimenta una necesidad hasta que concluye la utilización del servicio que utiliza para cubrir dicha necesidad. Dicho de otro modo, en el caso de una aplicación móvil el *journey map* describe de manera gráfica todos los pasos que da el usuario en la aplicación hasta que consigue su objetivo.

El journey map es una herramienta muy útil en la etapa de diseño pues refleja diversos aspectos del contexto en el que se produce la utilización del producto o servicio que describe. Entre otros datos, en *el journey map* suelen aparecer bien delimitadas todas las etapas del proceso, las necesidades que se pretenden cubrir en cada una de esas fases y las oportunidades que pueden surgir, así como la evolución de la actitud del usuario de inicio a fin. Toda esta información recopilada de una manera tan visual aporta una visión clara al equipo de desarrollo acerca del uso que se va a hacer del producto.

Con un *customer journey map* se logra visualizar un proceso que hasta este punto sólo se ha descrito de forma verbal y, en el caso de que intervenga un cliente o tengamos un equipo de desarrollo numeroso, permite que todas las partes interesadas participen de una visión común del proyecto [9] (Melone, 2018). De este modo el trabajo y la colaboración entre las partes se realiza de manera más eficiente.

En el caso de nuestro proyecto, no contamos con más desarrolladores ni tenemos encima a un cliente cambiándonos los requisitos cada día. Pero eso no significa que la elaboración de un *journey map* sea estéril. Estamos ante otra herramienta de la fase de diseño que contribuye a aportar más información sobre el problema que vamos a solucionar con nuestra aplicación y a mejorar la idea que podemos tener en esta fase sobre la interacción del usuario con nuestro producto.

Así, en nuestro *customer journey map*, vemos cómo una de las personas descritas al comienzo del capítulo necesita obtener información sobre la calidad del aire en un momento dado.

Luis Hernández



Médico de familia que trabaja en un centro de salud al norte de la Comunidad de Madrid

User goals

INFO ACTUAL

AÑADIR UNA ALERTA

SALIR DE LA APP

Consultar el índice de la calidad del aire en ese momento

Seleccionar un índice de contaminación atmosférica a partir del cual mostrar la alarma

Guardar la alarma junto a las anteriores

El usuario es pasivo en esta fase

Process and channels



Functions

Obtener y mostrar la información sobre la calidad del aire

Localizar automáticamente la posición del usuario y dar la opción de introducir las coordenadas de otro lugar a mano

Mostrar lista de alertas actuales

Función de alerta emergente en caso de sobrepasar el índice de contaminación guardado

Mental experience & thoughts

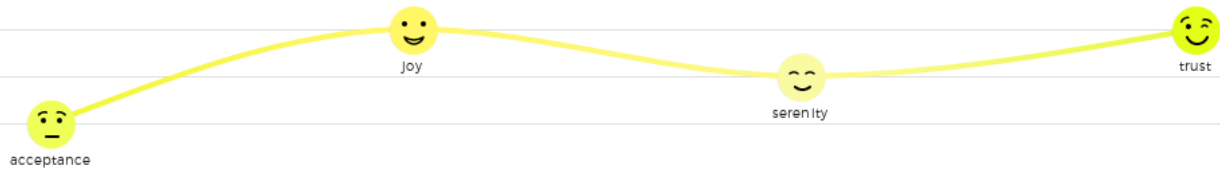
¿Por qué no puedo tener la información más a mano?

Es muy sencillo seleccionar una localización y un límite de contaminación

Tengo todas las alertas que necesito

¡Ahora puedo disfrutar de mis planes sin preocupación!

Experience



Ideas / Opportunities

Usar un widget en la pantalla principal del smartphone

¿Desplegable con nombres de lugares para la selección?

Opción de modificar y borrar alerta

¿Sonido de alarma junto al pop-up?

2.4 Competencia

Por último, con el objetivo de tener una visión más profunda de la problemática que pretendemos resolver y poder aportar una solución diferente, hemos realizado un breve estudio de las alternativas que ofrece el mercado de aplicaciones móviles.

Aunque es cierto que existe competencia, tanto en la *App Store* como en la *Play Store*, no hemos encontrado apenas aplicaciones enfocadas a proporcionar información sobre la calidad del aire y la radiación ultravioleta o, al menos, no hemos descubierto demasiadas que resulten realmente atractivas para el usuario. Además, tampoco hemos observado ninguna aplicación que aúne los dos factores medioambientales que pretende informar nuestra app.

Al final, para nuestro estudio hemos seleccionado dos aplicaciones móviles que, en nuestra opinión, destacan por encima de las demás. La primera aplicación móvil que hemos seleccionado es *AirVisual*.

AirVisual es un caso particular por varias razones. Por un lado, se trata de una API que ofrece en tiempo real datos acerca de la calidad del aire en todo el mundo. Dispone de un paquete básico de acceso gratuito a su API, que es el que utilizaremos como *web service* para obtener los datos que necesitará nuestra aplicación.

Por otro lado, *AirVisual App* es una app móvil muy cuidada en cuanto a diseño, que ofrece una colección de datos amplísima en relación a multitud de factores sobre la calidad del aire y, en definitiva, estamos ante la opción más profesional que existe en el mercado.



AirVisual App, como apuntamos, muestra en la pantalla del terminal móvil una cantidad ingente de datos y estadísticas acerca de la polución. Ninguna otra app ofrece más información. Y, precisamente, lo que es el punto fuerte de la aplicación es, as su vez, un punto débil. Después de probar el software creemos que tantos números, tantas gráficas de barras y tantos colores en pantalla pueden llegar a abrumar al usuario debido a la sobreinformación.

Nuestra app tiene presente el problema del que adolece AirVisual App y su diseño orientado al minimalismo tiene como objetivo aportar una solución alternativa al problema, ofreciendo una experiencia de usuario agradable, sencilla y directa.

La segunda aplicación que hemos seleccionado es UV Index Widget – Worldwide. Esta aplicación, que sólo está disponible en la App Store de Apple, se aproxima mucho más que AirVisual App a la idea de producto que deseamos ofrecer a nuestros usuarios. UV Index Widget – Worldwide ofrece información sobre el índice de radiación ultravioleta en lugar y un momento dado. Y, al contrario que AirVisual App, lo hace de forma sencilla, sin presentar un exceso de información en pantalla. Por tanto, resulta más visual y clara para el usuario, que también es notificado con alertas si el índice UV supera un nivel determinado.



Sin embargo, UV Index Widget – Worldwide ofrece solo información acerca de la radiación ultravioleta, lo que nos puede llevar a echar en falta la medición de algún factor medioambiental más. No obstante, a pesar de su excelente diseño y sencillez, el principal problema que vemos en UV Index Widget – Worldwide es que sólo está disponible para los dispositivos Apple, quedando los usuarios de Android sin esta excelente opción para medir el índice UV. Por nuestra parte, vemos una gran oportunidad y, aprovechando las ventajas de un desarrollo híbrido con Ionic, ofreceremos nuestra aplicación tanto para la plataforma iOS como para Android.

En resumen, aunque existen alternativas en el mercado para la problemática que deseamos resolver con nuestra aplicación, podemos aportar algo nuevo con nuestro producto. De este modo, creemos que podemos enfocar el desarrollo de nuestra app desde una perspectiva de ventaja competitiva por diferenciación [10] (Espinosa, 2017), proponiendo una opción más accesible de lo que ya ofrece el mercado de aplicaciones, mejorando la experiencia de usuario, poniendo a su disposición una alternativa que destaque por su sencillez e inmediatez de uso y, además, conformando una auténtica opción multiplataforma.

3. Diseño y prototipado

El propósito de esta fase es completar un prototipo de alta fidelidad de las pantallas que van a componer nuestra aplicación que nos sirva de guía visual durante la etapa de implementación.

Para llegar a obtener un prototipo primero repasaremos las funcionalidades detectadas a partir del análisis de los resultados obtenidos durante las actividades del capítulo anterior. Después, con la información extraída del estudio de los usuarios y de las funciones que implementará nuestra aplicación pasaremos a elaborar un primer boceto con el que disponer de un esqueleto en el que visualizar la disposición de los contenidos de la app. A continuación, recurriremos a las herramientas que ofrece *Draw.io* para diseñar un prototipo de baja fidelidad de cada una de las pantallas a partir de las ideas plasmadas en los bocetos. Por último, con la ayuda de *Proto.io* pasaremos ya a diseñar un prototipo de alta fidelidad, que mostrará visualmente una versión muy aproximada a lo que será el producto final.

3.1 Funcionalidades detectadas

Hemos realizado nuestro análisis de usuario empleando durante el proceso técnicas como las de personas, escenarios o el *journey map*. Todas estas técnicas nos han proporcionado una visión más clara de la funcionalidad que debe ofrecer nuestra aplicación móvil para poder satisfacer la demanda del usuario y resolver el problema que queremos solucionar. Llegados a este punto, con la información que hemos obtenido después del análisis de usuarios —y teniendo en cuenta los requisitos funcionales que ya detectamos en el primer capítulo— estamos en disposición de reevaluar la funcionalidad que vamos a implementar en nuestra aplicación. Como ya señalamos durante el análisis de la competencia, *AirSun* pretende diferenciarse del resto de aplicaciones del sector presentándose como una alternativa sencilla a la hora de obtener información sobre la calidad del aire y del índice de radiación ultravioleta. No queremos abrumar al usuario con un sinfín de datos, gráficos y estadísticas. La consulta del usuario ha de ser rápida y la información obtenida tiene que ser clara e inequívoca. Por este motivo, la lista de funcionalidades principales no será especialmente extensa, pero aquellas que se implementen cumplirán de manera eficiente con su objetivo.

A continuación, describimos las funciones principales que ofrecerá la aplicación *AirSun*.

Consultar información: en la pantalla principal de la aplicación el usuario obtendrá de manera clara y visual la información relativa a dos factores medioambientales relacionados con la salud: el índice de contaminación atmosférica y el índice de radiación ultravioleta. Ambos valores se obtendrán de la consulta de sendos servicios web. Los índices se mostrarán en pantalla con su valor numérico e irán acompañados de un código de color que refleje visualmente el grado de contaminación o de radiación ultravioleta que representa dicho número. De este modo, si obtenemos un valor elevado en alguno de los factores medioambientales que estamos midiendo lo reflejaremos sobre un fondo rojizo, estableciendo una asociación entre el valor mostrado y el color sobre el que aparece. Por tanto, los datos se muestran al usuario mediante un valor numérico

acompañado de un código de color, ayudando a que la consulta de la información sea lo más rápida e inequívoca posible.

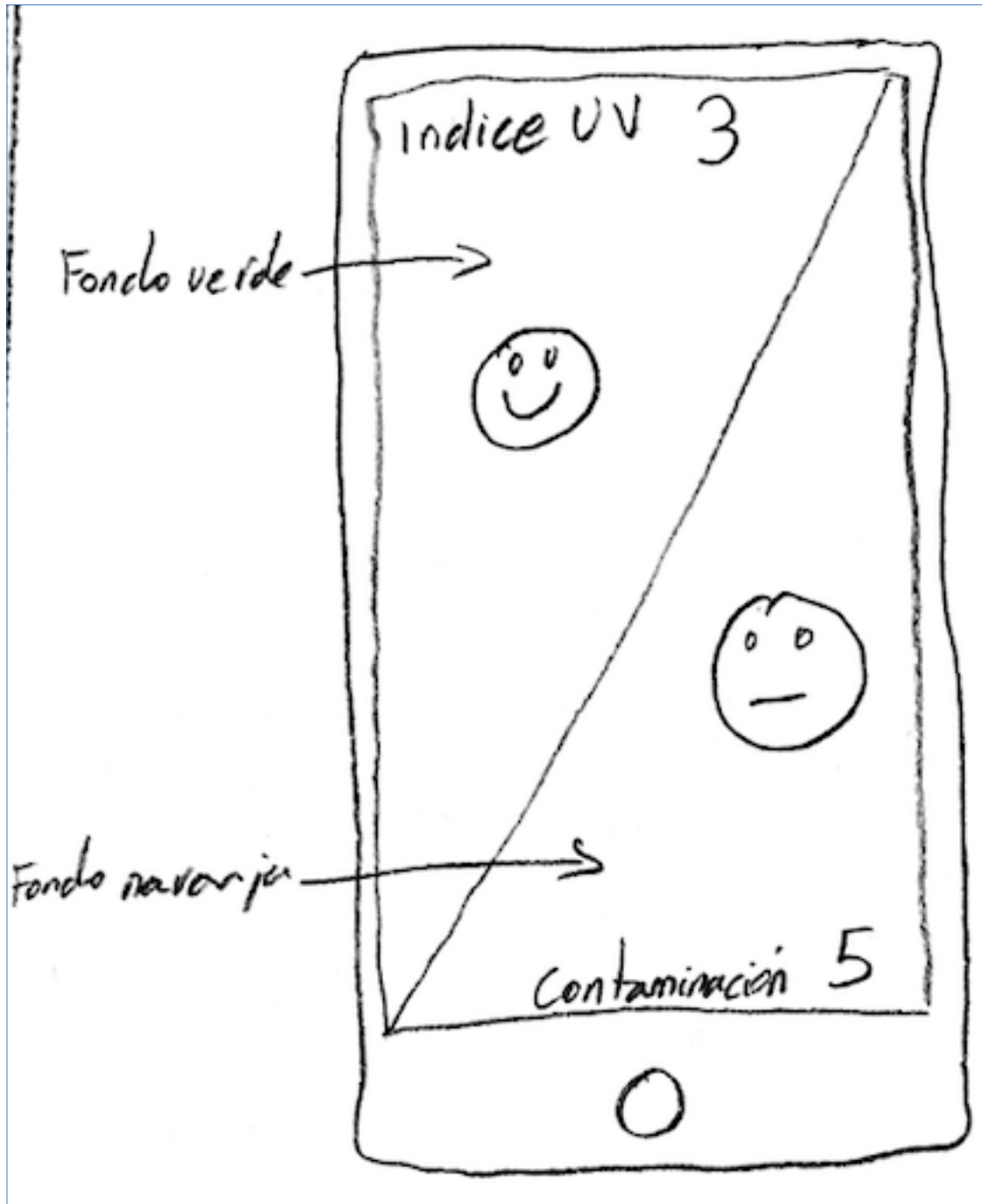
Crear una alerta: con esta función permitiremos al usuario configurar alertas personalizadas para que, en caso de cumplirse la casuística establecida en la alerta, reciba una alerta emergente en la pantalla del dispositivo móvil, aunque se encuentre fuera de la aplicación. El usuario indicará una localización, un valor máximo y el valor medioambiental sobre el que se creará la alerta.

Editar y borrar una alerta: una tercera pantalla mostrará el listado de alertas creadas por el usuario. Se indicará la localización, el factor medioambiental medido y el valor máximo al partir del cual se activará la alerta. Si el usuario desplaza un dedo sobre uno de los elementos de la lista aparecerán dos iconos que permitirán editar o eliminar la alarma seleccionada.

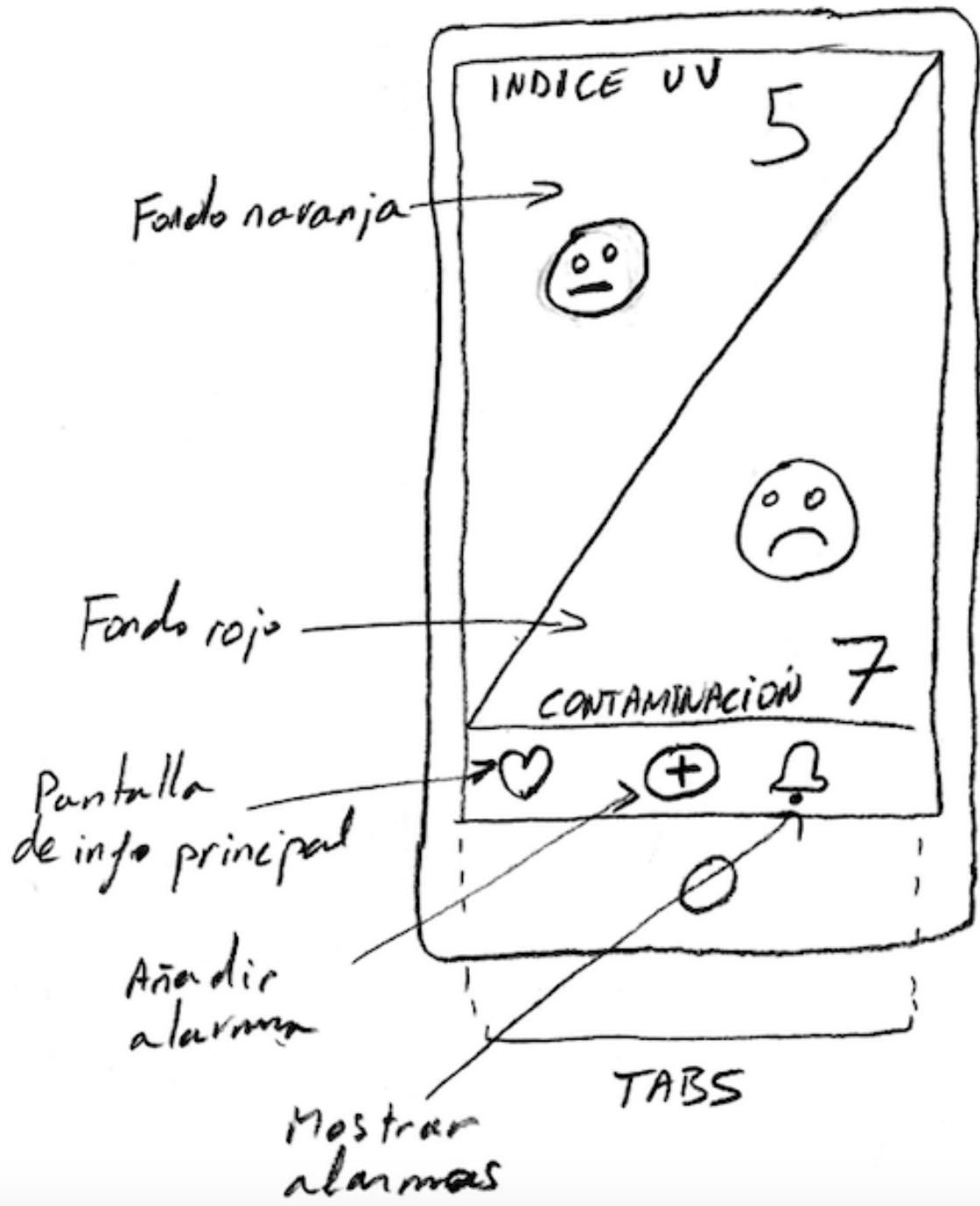
Éstas funciones constituyen la funcionalidad principal de la aplicación, pero dada su naturaleza, estamos ante una app que es susceptible de ver incrementada su funcionalidad en futuras iteraciones ofreciendo, por ejemplo, más indicadores medioambientales.

Antes de sentarnos a diseñar las pantallas de la app con un programa de diseño es aconsejable tomar lápiz y papel y crear una primera aproximación visual a la distribución que deseamos para los contenidos en cada una de las pantallas.

A continuación, mostramos los bocetos obtenidos en esta fase:



Pantalla de información (versión 1)

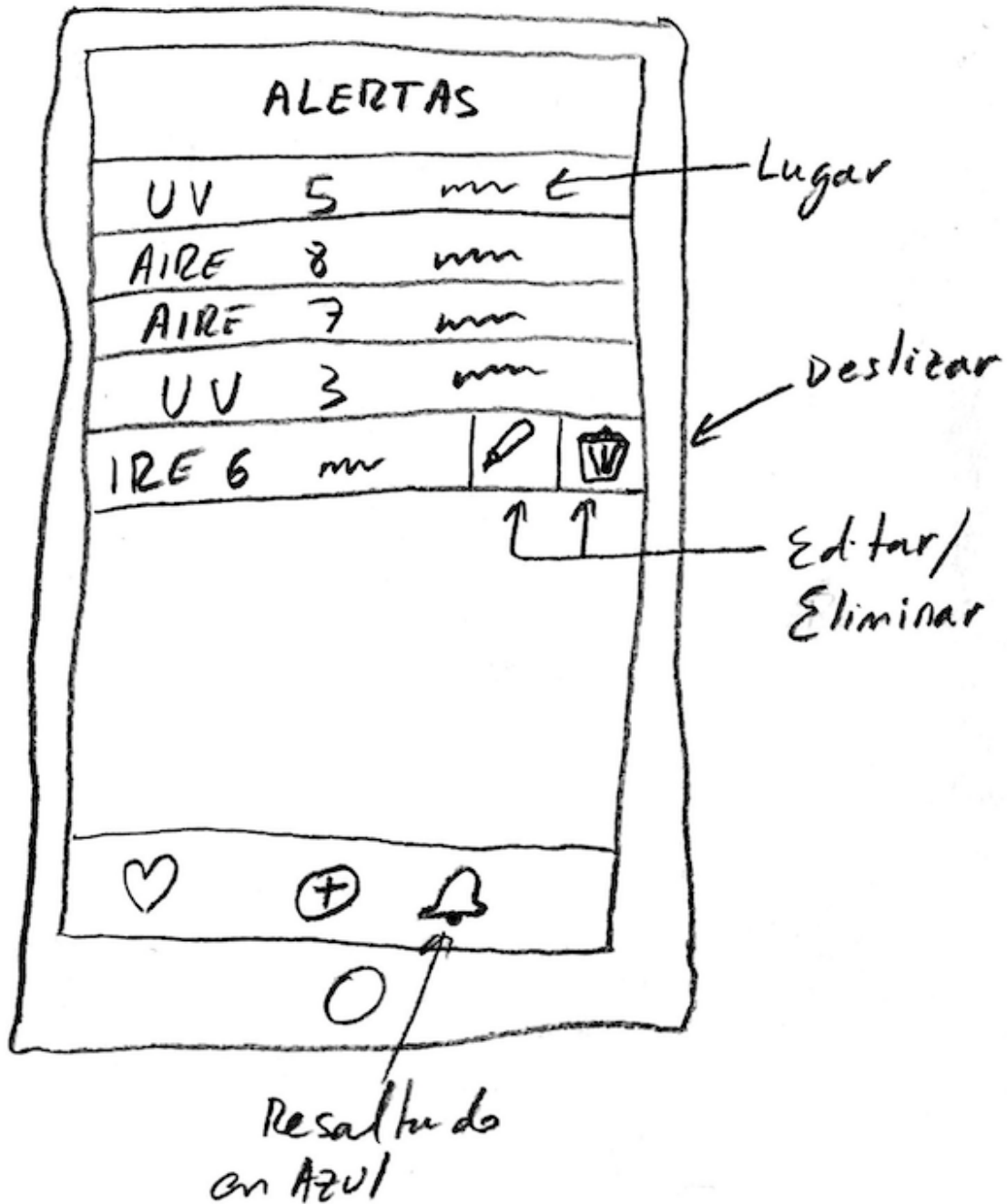


Pantalla de información (versión 1)

Los dos bocetos anteriores representan la pantalla principal de la aplicación. Es la pantalla que recibe al usuario y que muestra el índice de contaminación atmosférica y el índice de radiación ultravioleta.

Cada valor de la medición irá acompañado por un emoticono, de modo que refleje de forma gráfica el grado de contaminación y radiación. Además, el color del fondo de la sección estará relacionado con el valor obtenido, así factores de contaminación y de radiación ultravioleta bajos irán sobre un fondo de color verdoso, mientras que índices mayores aparecerán sobre una tonalidad rojiza.

Los *tabs* situados en el área inferior de la pantalla constituyen la navegación principal de la aplicación. A través de los iconos el usuario podrá desplazarse al resto de las pantallas.



Pantalla de alertas

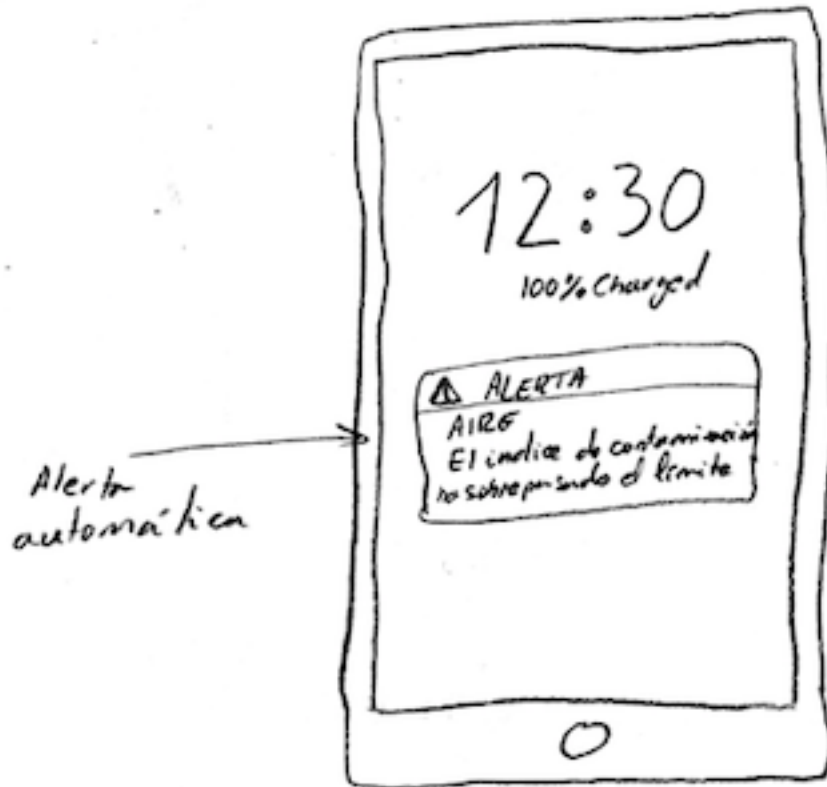
La pantalla anterior muestra todas las alertas programadas por el usuario. Es un listado en el que cada elemento de la lista contiene el factor ambiental medido, el grado a partir del cual se lanza la alerta y el lugar para el que se programa. Si el usuario desliza el dedo sobre uno de los elementos aparecerán dos iconos con las opciones de editar y eliminar la alerta.



Resaltado
en Azul

Pantalla de creación de alerta

El boceto anterior se corresponde con la pantalla de creación de alerta. En ella aparece un formulario en el que el usuario puede introducir un lugar, un factor ambiental y un índice a partir del cual se muestre la alerta. Dos botones ofrecen la posibilidad de guardar la alerta o cancelar el proceso.



Notificación de la alerta

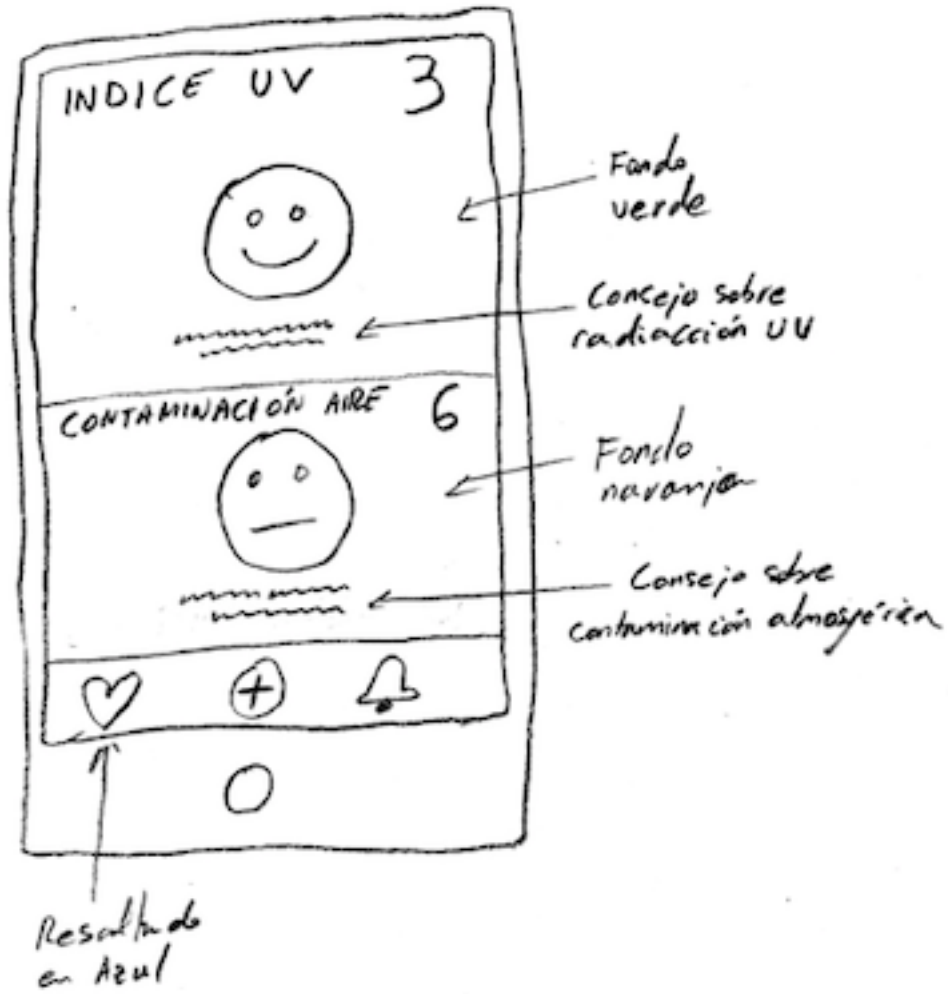
Por último, ya fuera de la aplicación, se lanza una notificación cuando se cumplen las condiciones establecidas en una de las alertas creadas por el usuario.

Una vez elaborados los primeros bocetos decidimos revisar la pantalla principal de la aplicación y realizar algunas modificaciones.

El cambio principal que hemos introducido sobre la pantalla de información tiene que ver con la división de las dos secciones informativas que componen la pantalla. En lugar de establecer una división diagonal hemos optado por una horizontal. Creemos que, de esta forma, no sólo la información mostrada es más clara para el usuario, sino que, además, puede facilitarnos la posterior implementación de la vista.

Por otro lado, decidimos que la pantalla principal quedaba algo vacía una vez distribuidos los contenidos que deseábamos mostrar. Para aportar algo más de carácter a la pantalla podríamos mostrar debajo de cada emoticono un consejo relacionado con el factor ambiental medido.

Por tanto, el boceto de la pantalla principal de la aplicación quedaría de la siguiente manera.

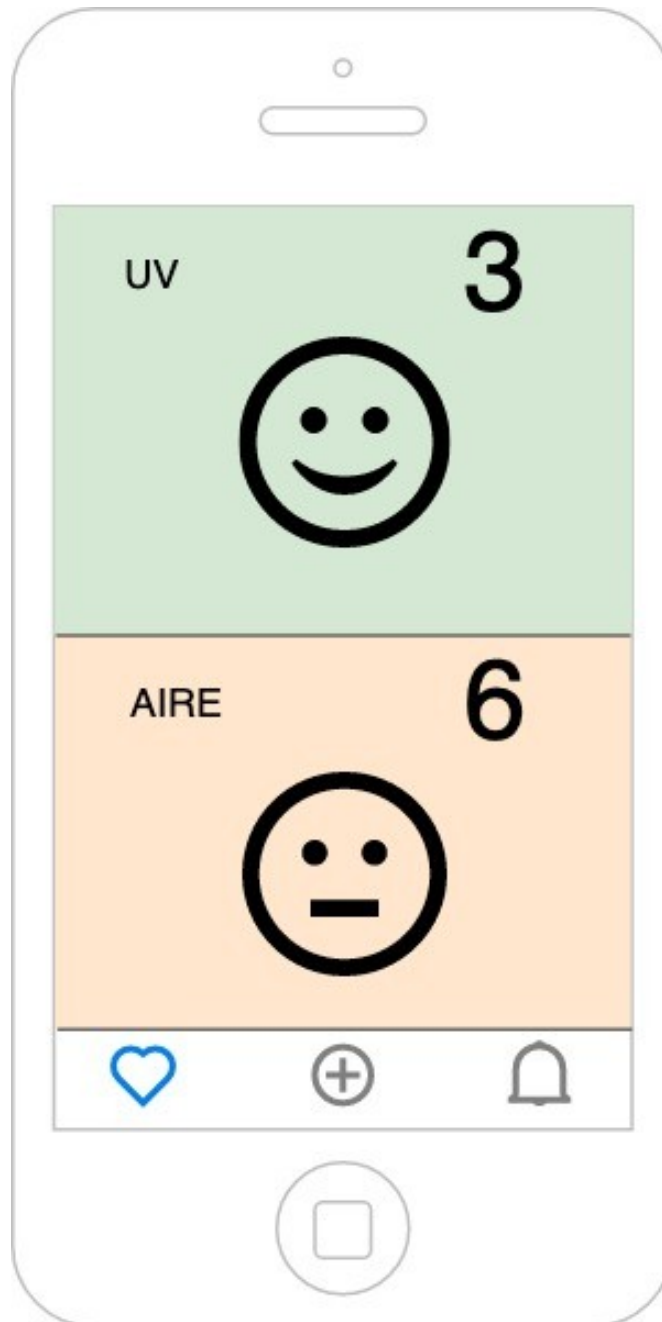


Pantalla de información (versión 2)

3.3 Prototipos de baja fidelidad

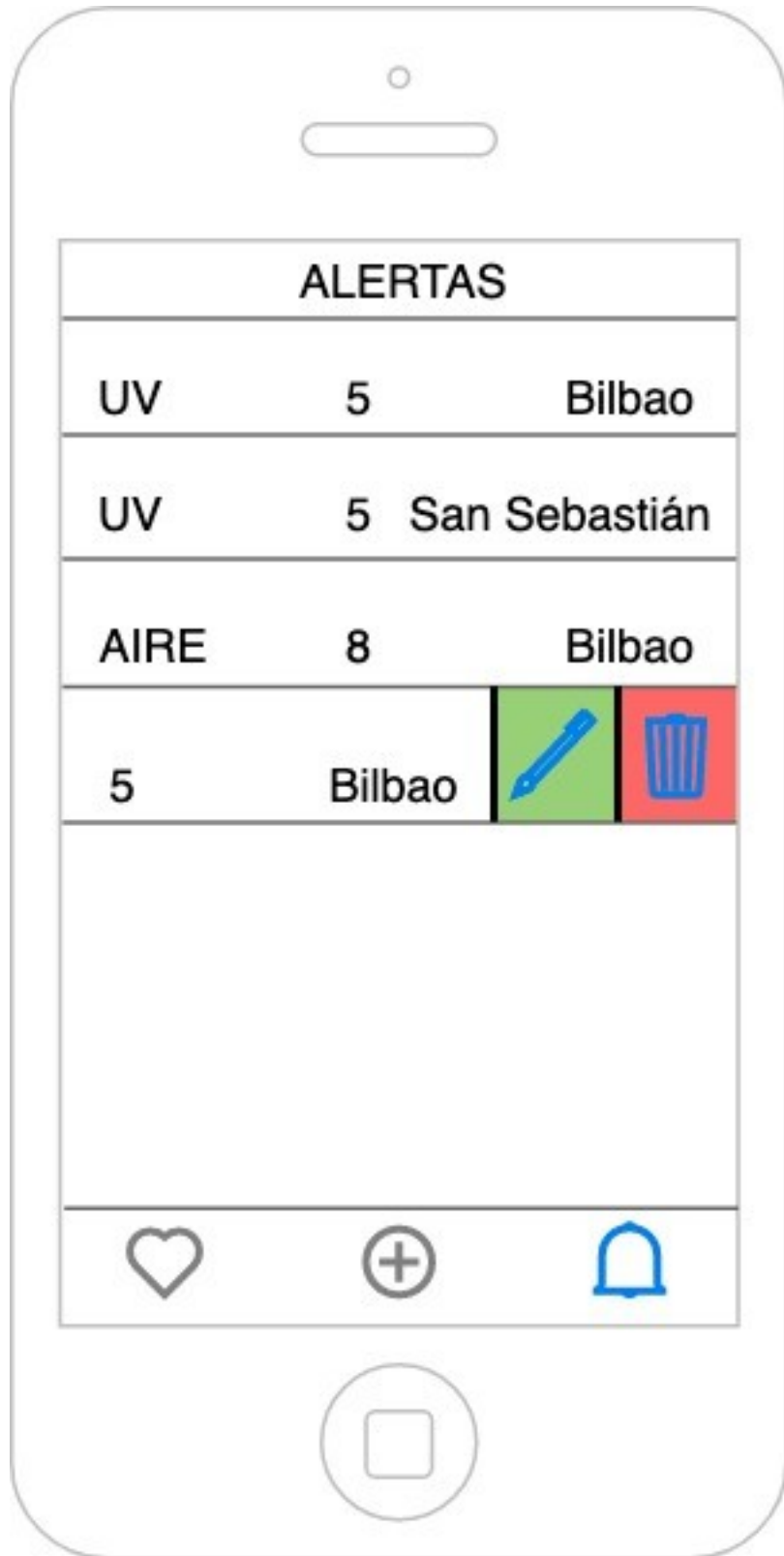
Una vez que hemos realizado los primeros bocetos y con una idea más clara de las pantallas que compondrán el producto final, recurrimos a la aplicación web *Draw.io* para llevar al siguiente paso los bocetos, elaborando unos *wireframes* que se acercarán más al aspecto que va a mostrar la app móvil.

Las pantallas diseñadas con la ayuda de *Draw.io* son las siguientes:



Pantalla de información

Cuando realizamos el prototipo de baja fidelidad de la pantalla principal nos damos cuenta de que no nos convence visualmente la decisión de mostrar un consejo debajo de cada emoticono. Por tanto, decidimos eliminarlo.



Pantalla de lista de alertas



Pantalla de creación de alerta



Notificación push de la alerta

3.4 Prototipos de alta fidelidad

Por último, los prototipos de alta fidelidad nos ofrecerán la versión más próxima al producto definitivo. Basándonos en estos prototipos hemos tomado las últimas decisiones de diseño antes de la fase de implementación. Para su elaboración hemos optado por la aplicación web Proto.io por su facilidad de uso y flexibilidad.

Los resultados obtenidos con el prototipado de alta fidelidad son los siguientes:

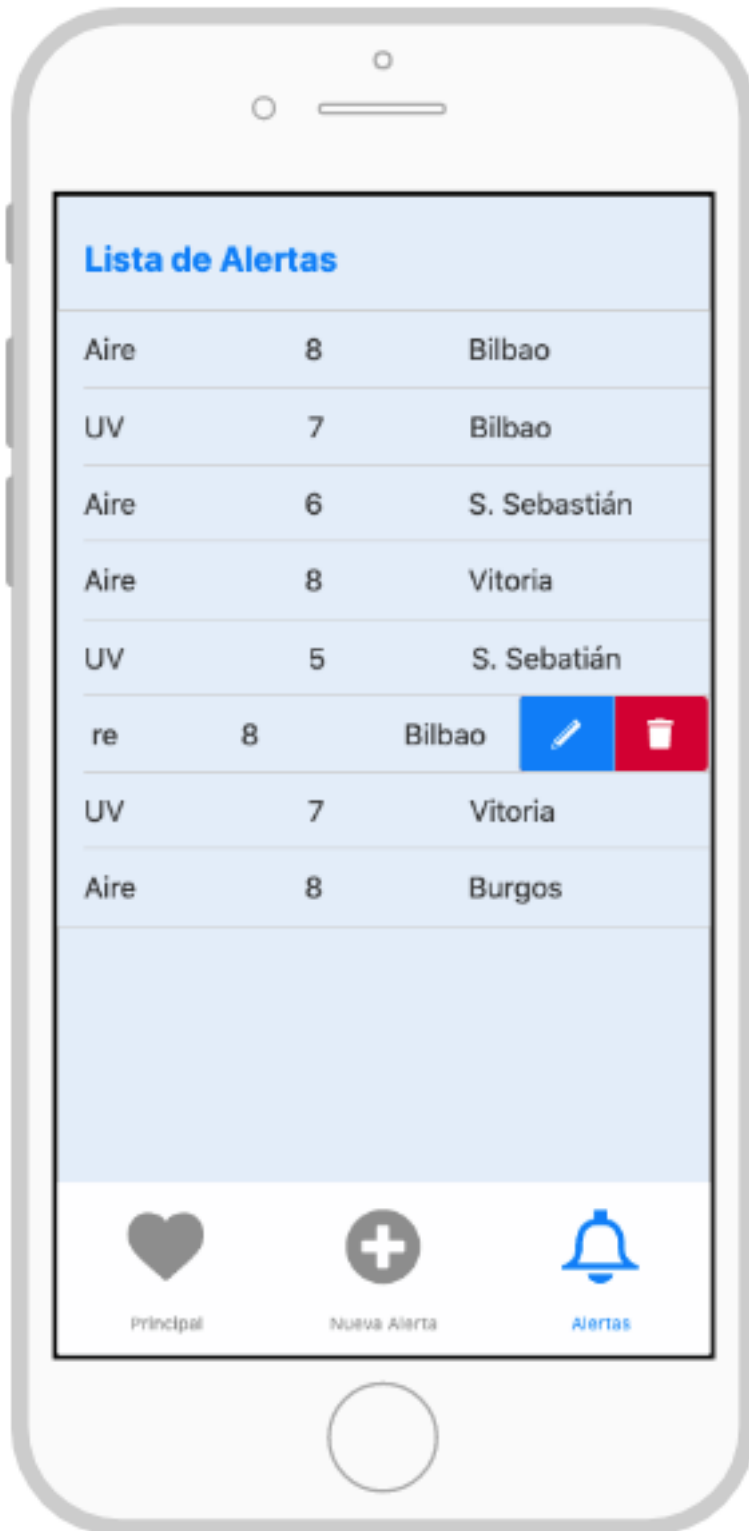


Pantalla de información

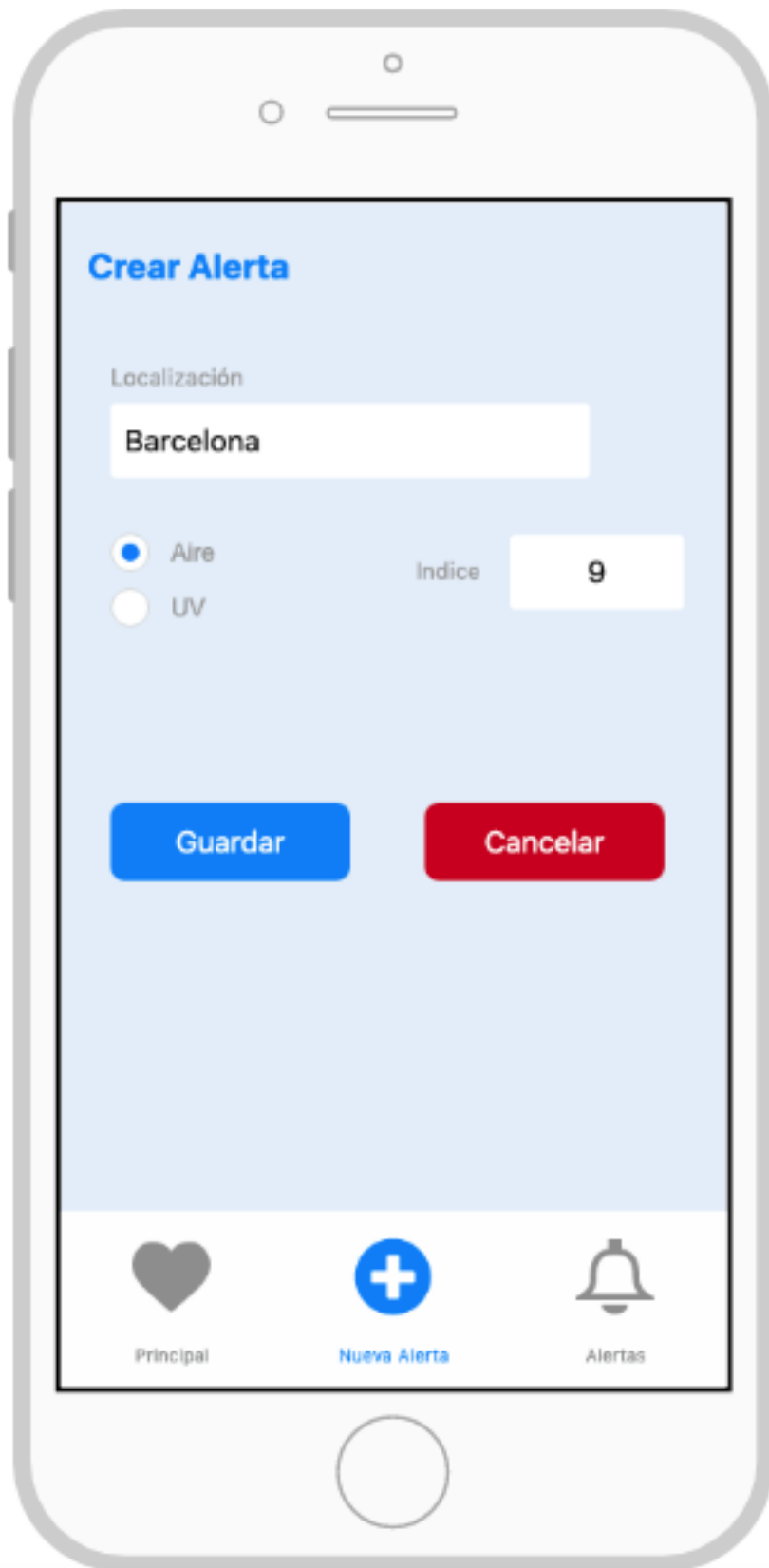


Pantalla de información

En la pantalla de información hemos optado por un diseño minimalista, en el que prescindimos de elementos textuales a la hora de mostrar los datos. Ya no aparecen las etiquetas de “AIRE” y “UV” que veíamos en el prototipo de baja fidelidad. Las hemos sustituido por unos iconos que aporten el mismo significado de manera más visual. También hemos cambiado los emoticonos por unas imágenes más estéticas, que hemos obtenido en la web de contenido gratuito <https://www.flaticon.com/>.



Pantalla de lista de alertas



Pantalla de creación de alerta



Pantalla de notificación push

4. Evaluación

4.1 Casos de uso

Los casos de uso constituyen una gran herramienta para observar las operaciones que realiza un usuario de forma secuencial en un sistema con un objetivo definido. Los casos de uso muestran la interacción del usuario con el sistema y sirven para especificar la funcionalidad que ha de tener este último. Con esta herramienta podemos definir los requerimientos del sistema desde el punto de vista del usuario [11] (Cevallos, 2015).

Para nuestra aplicación hemos detectado los siguientes casos de uso:

CU-001	Consultar índice de contaminación atmosférica
Actores	Usuario de la aplicación
Precondiciones	El sistema debe tener acceso al GPS del dispositivo y haber cobertura
Flujo	<ol style="list-style-type: none">1. El usuario accede a la aplicación.2. El sistema accede al GPS del dispositivo para obtener sus coordenadas geográficas.3. El sistema realiza una consulta a la API REST de AirVisual para obtener el índice de contaminación atmosférica en esas coordenadas geográficas.4. El sistema presenta la información en pantalla mediante un valor numérico, un emoticono y un color de fondo.
Postcondiciones	-

CU-002	Consultar índice de radiación ultravioleta
Actores	Usuario de la aplicación
Precondiciones	El sistema debe tener acceso al GPS del dispositivo y haber cobertura
Flujo	<ol style="list-style-type: none"> 1. El usuario accede a la aplicación. 2. El sistema accede al GPS del dispositivo para obtener sus coordenadas geográficas. 3. El sistema realiza una consulta a la API REST de OpenUV para obtener el índice de radiación ultravioleta en esas coordenadas geográficas. 4. El sistema presenta la información en pantalla mediante un valor numérico, un emoticono y un color de fondo.
Postcondiciones	-

CU-003	Añadir nueva alerta de contaminación atmosférica
Actores	Usuario de la aplicación
Precondiciones	-
Flujo	<ol style="list-style-type: none"> 1. El usuario pulsa el botón + en el tab de navegación. 2. El sistema presenta en pantalla un formulario. 3. El usuario rellena los campos de lugar, selecciona el check-box de Aire y establece un índice a partir del cual emitir la alerta. 4. El usuario pulsa el botón de guardar. 5. El sistema avisa al usuario de que la alerta ha sido guardada y le redirige a la pantalla del listado de alertas.
Postcondiciones	En el momento en el que se cumplan las condiciones de la alerta emite una notificación push al usuario. El sistema debe tener acceso al GPS del dispositivo y haber cobertura

CU-004	Añadir nueva alerta de radiación ultravioleta
Actores	Usuario de la aplicación
Precondiciones	-
Flujo	<ol style="list-style-type: none"> 1. El usuario pulsa el botón + en el tab de navegación. 2. El sistema presenta en pantalla un formulario. 3. El usuario rellena los campos de lugar, selecciona el check-box de UV y establece un índice a partir del cual emitir la alerta. 4. El usuario pulsa el botón de guardar. 5. El sistema guarda la alerta en memoria, avisa al usuario y le redirige a la pantalla del listado de alertas.
Postcondiciones	En el momento en el que se cumplan las condiciones de la alerta emite una notificación push al usuario. El sistema debe tener acceso al GPS del dispositivo y haber cobertura

CU-005	Cancelar creación de nueva alerta
Actores	Usuario de la aplicación
Precondiciones	-
Flujo	<ol style="list-style-type: none"> 1. El usuario pulsa el botón + en el tab de navegación. 2. El sistema presenta en pantalla un formulario. 3. El usuario rellena los campos de lugar, selecciona el check-box de Aire y establece un índice a partir del cual emitir la alerta. 4. El usuario pulsa el botón de cancelar. 5. El sistema redirige al usuario a la página principal.
Postcondiciones	-

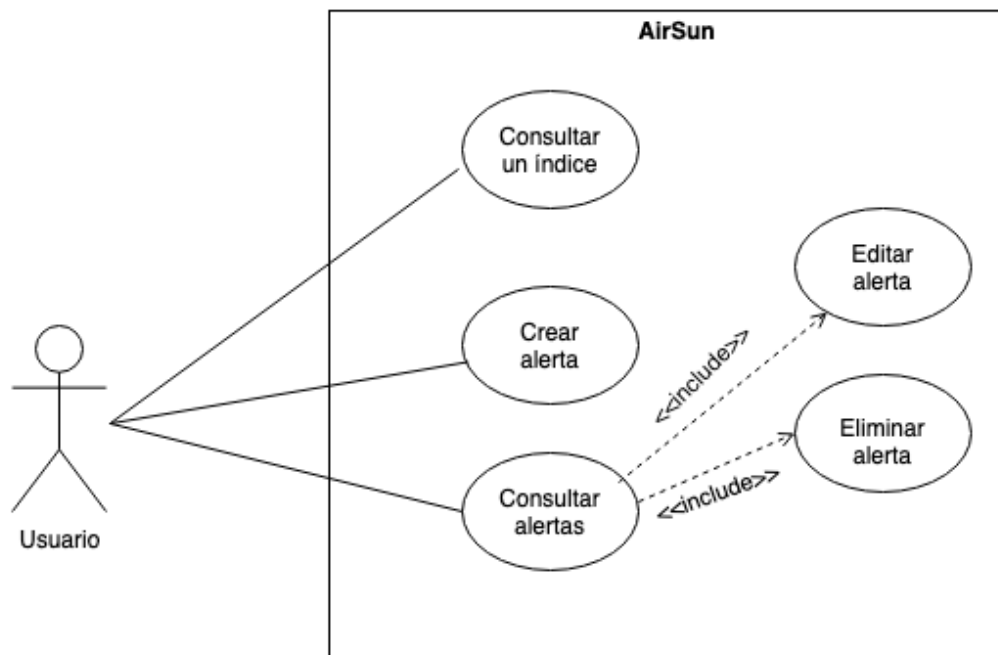
CU-006	Consultar alertas existentes
Actores	Usuario de la aplicación
Precondiciones	El usuario debe haber creado al menos una alerta
Flujo	<ol style="list-style-type: none"> 1. El usuario pulsa el botón del icono de la campana en el tab de navegación. 2. El sistema recupera de memoria el listado con las alertas creadas y lo muestra en pantalla.
Postcondiciones	-

CU-007	Eliminar alerta existente
Actores	Usuario de la aplicación
Precondiciones	El usuario debe haber creado al menos una alerta
Flujo	<ol style="list-style-type: none"> 1. El usuario pulsa el botón del icono de la campana en el tab de navegación. 2. El sistema recupera de memoria el listado con las alertas creadas y lo muestra en pantalla. 3. El usuario desplaza un dedo horizontalmente sobre uno de los elementos de la lista y pulsa el icono de la papelera. 4. El sistema elimina la alerta de memoria y avisa al usuario.
Postcondiciones	-

CU-008	Editar alerta existente
Actores	Usuario de la aplicación
Precondiciones	El usuario debe haber creado al menos una alerta
Flujo	<ol style="list-style-type: none"> 1. El usuario pulsa el botón del icono de la campana en el tab de navegación. 2. El sistema recupera de memoria el listado con las alertas creadas y lo muestra en pantalla. 3. El usuario desplaza un dedo horizontalmente sobre uno de los elementos de la lista y pulsa el icono del lápiz. 4. El sistema redirige al usuario a la pantalla de creación de alerta con los datos precargados. 5. El usuario modifica los datos. 6. El usuario pulsa el botón de guardar. 7. El sistema modifica la alerta en memoria, avisa al usuario mediante un mensaje y le redirige a la pantalla del listado de alertas.
Postcondiciones	-

CU-008	Error consulta de índice de contaminación
---------------	--

Actores	Usuario de la aplicación
Precondiciones	El sistema no tiene acceso al GPS del dispositivo o no hay cobertura de red
Flujo	<ol style="list-style-type: none"> 1. El usuario accede a la aplicación. 2. El sistema no logra acceder al GPS del dispositivo o a la red móvil. 3. El sistema muestra en pantalla el error.
Postcondiciones	-



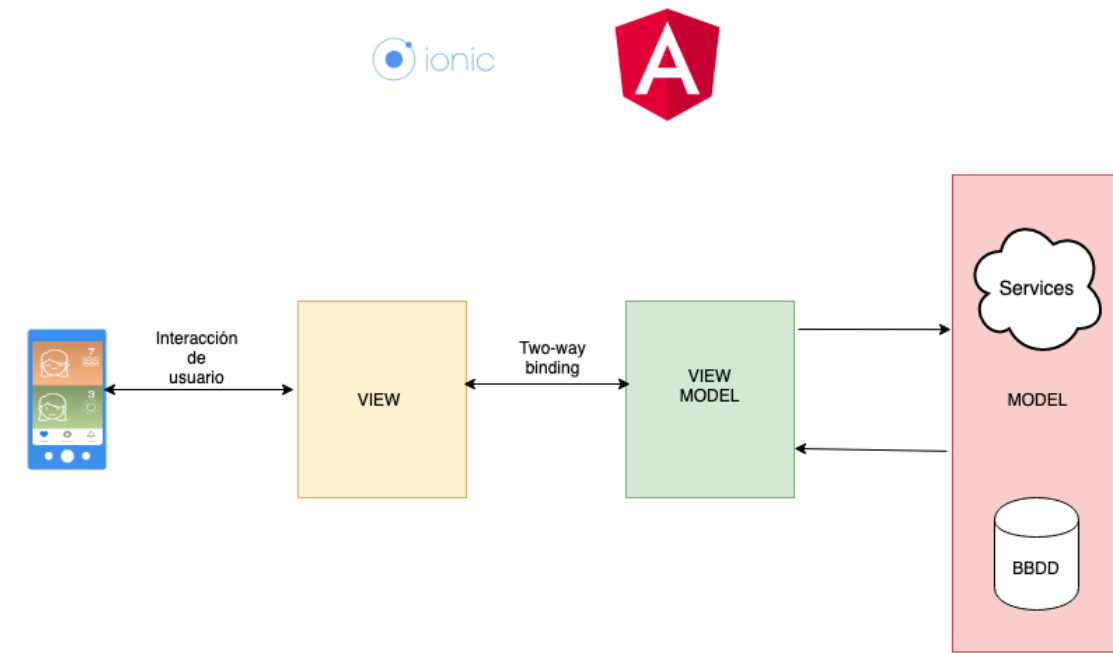
4.2 Diseño de la arquitectura

La aplicación AirSun será desarrollada con Ionic para aprovechar los beneficios que ofrece el desarrollo híbrido. La app estará disponible para los dos sistemas operativos predominantes en el universo móvil, pero no deseamos abordar dos desarrollos nativos por separado. Una plataforma como Ionic nos permitirá desarrollar una sola vez utilizando tecnologías web para luego desplegar el software final tanto en iOS como en Android.

Con Ionic utilizaremos el *framework* de Angular, que nos facilitará el desarrollo y nos permitirá organizar el proyecto entorno al patrón de diseño conocido como MVVM (Modelo Vista VistaModelo), similar al célebre MVC (Modelo Vista Controlador).

El patrón MVVM nos permitirá separar las diferentes capas de la aplicación, disociando la interfaz gráfica, la lógica de negocio y el modelo de datos. En el caso del patrón MVVM en Angular la interacción entre la vista y el controlador se producirá en ambos sentidos y el controlador, además de mostrar los datos en la vista observa si se produce algún cambio en la vista y actualiza el modelo de datos de forma automática [12] (José Jesús Pérez Rivas, 2015).

AirSun es un desarrollo relativamente sencillo basado en una plataforma que emplea un patrón de diseño ampliamente contrastado. Lo más destacable, en el caso de nuestra aplicación, lo encontramos en la capa de modelo de datos. Por un lado, necesitaremos un pequeño espacio para la persistencia de datos, que utilizaremos para almacenar las alarmas creadas por el usuario. Por otro lado, el sistema deberá invocar a dos servicios web de arquitectura REST para la obtención de la información relativa a los índices medioambientales que requiere el usuario.



5. Implementación

5.1 Proyecto de Ionic

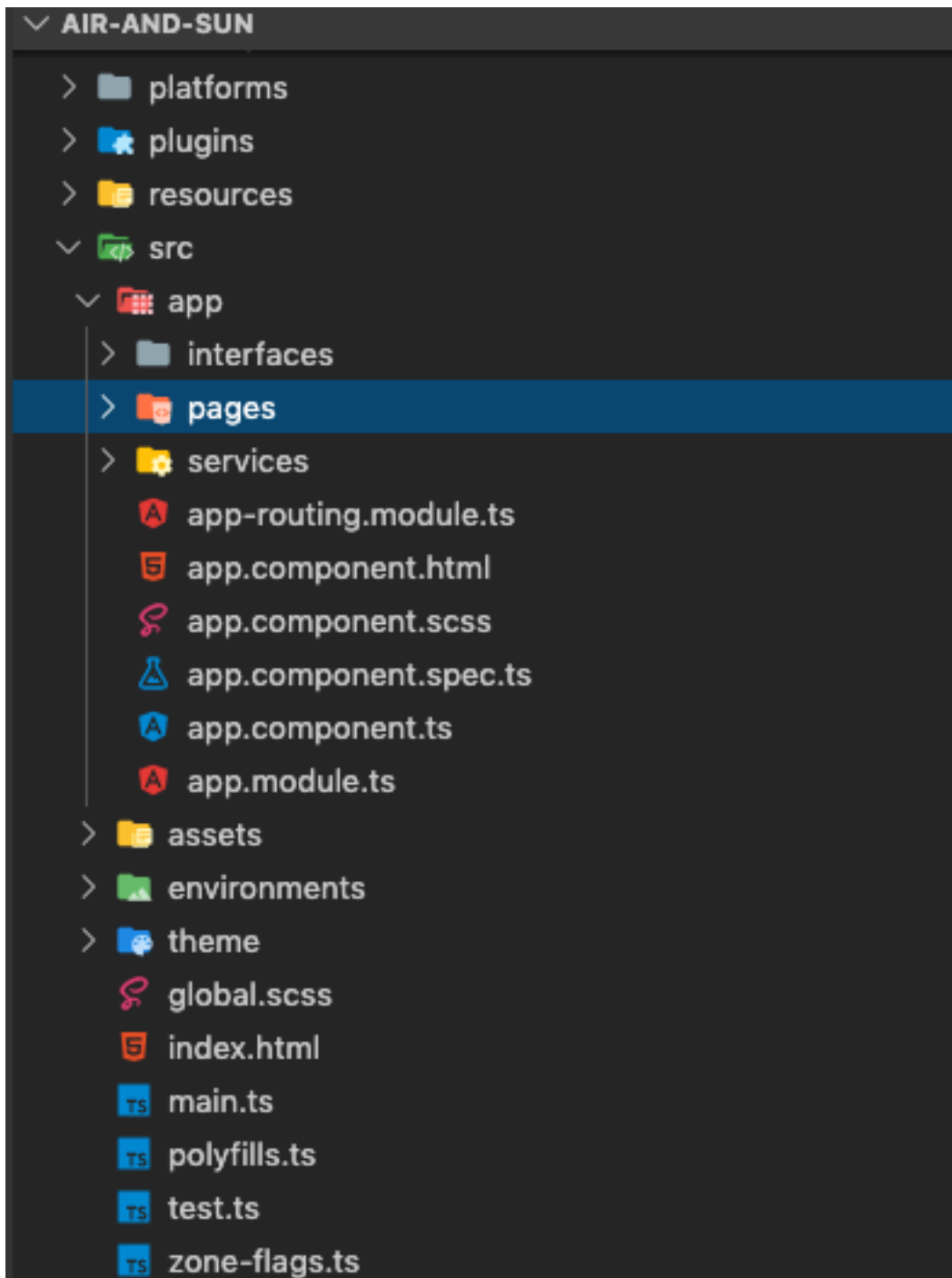
Una vez hemos realizado un análisis de nuestro *target* de usuarios, nos hemos adentrado en el diseño y prototipado de la parte visual de nuestra aplicación, hemos evaluado los casos de uso y analizado el diseño que tendrá la arquitectura de nuestra app, llega el momento de abordar la parte más técnica del proceso: la implementación.

Desde el primer momento teníamos claro que realizaríamos un desarrollo híbrido para que nuestra aplicación estuviera disponible al mayor número posible de dispositivos sin tener que realizar múltiples desarrollos. Es aquí donde interviene la magia de Ionic. Este kit de desarrollo de software proporciona las herramientas y servicios necesarios para el desarrollo de aplicaciones, tanto móviles como de escritorio, utilizando tecnologías propias del desarrollo web, como HTML, CSS y JavaScript. Cualquier desarrollador web puede crear una aplicación móvil tras un breve periodo de aprendizaje sin necesidad de tener que acometer la ardua tarea de formarse en tecnologías nativas como Android/Java o iOS/Swift. Para una aplicación que no exija un rendimiento elevado del hardware del dispositivo, las ventajas del desarrollo con Ionic son más que evidentes.

A la hora de crear el proyecto en Ionic se nos ofrece la posibilidad de utilizar **Angular** o **React** como *framework* para la interfaz de usuario. Tomamos la decisión de emplear Angular para el desarrollo al estar más familiarizados con el *framework* y disponer, tal vez, de más documentación en línea que su alternativa React.

Cuando abrimos el proyecto que acabamos de crear nos encontramos con la estructura típica de un proyecto Angular/Ionic. Para nuestro desarrollo respetamos dicha estructura y sólo añadimos algunas carpetas que nos facilitarán la labor de organizar nuestro código.

Por un lado, creamos una carpeta con el nombre *pages* que contendrá tanto la lógica como la parte visual de cada una de las páginas de nuestra app. Además, creamos una carpeta a la que llamamos *services* que contendrá las implementaciones necesarias para el acceso a los servicios web que consultará nuestra app y, posiblemente, alguna clase de utilidades que necesitaremos durante el desarrollo. Y, por último, la carpeta *interfaces* que nos ayudará a convertir las respuestas de los servicios web a un formato de objeto más manejable en nuestra aplicación.



Estructura de carpetas de la app en Visual Studio Code

5.2 Pantallas

Según hemos definido durante las etapas de diseño y prototipado, nuestra aplicación contará con tres pantallas a las que se accederá a través de un sistema de *tabs* que permitirá una navegación clara y sencilla a lo largo de la app. Estas tres páginas se corresponden la pantalla principal (que es la que ofrece la información acerca del índice de contaminación atmosférica y de radiación ultravioleta), la pantalla de creación de alertas y la pantalla de listado de alertas.

Además de estas pantallas principales, habrá una cuarta pantalla de edición de alerta a la que se accederá mediante un componente **ion-item-sliding** desde la página del listado de alertas.

5.2.1 Principal

El archivo *HTML* de la pantalla principal contiene dos componentes *ion-row* que dividen la pantalla en dos áreas diferenciadas, que muestran la información relativa a la calidad del aire y al índice de radiación ultravioleta respectivamente.

Tanto el valor del índice, como el icono del rostro del personaje, así como el color del fondo de cada área son dinámicos y se actualizan en función de los campos enlazados en el archivo *.ts* que contiene la lógica.

En el método **ngOnInit** de dicho archivo Typescript se llama al método para obtener la geolocalización del dispositivo. En cuanto se han recibido las coordenadas del dispositivo se procede a llamar con esos valores a los dos servicios que proporcionan los datos de calidad del aire e índice UV.

Por último, una vez que tenemos la información de sendos servicios REST se actualiza la pantalla para que refleje los datos recibidos mediante las propiedades que tenemos enlazadas.

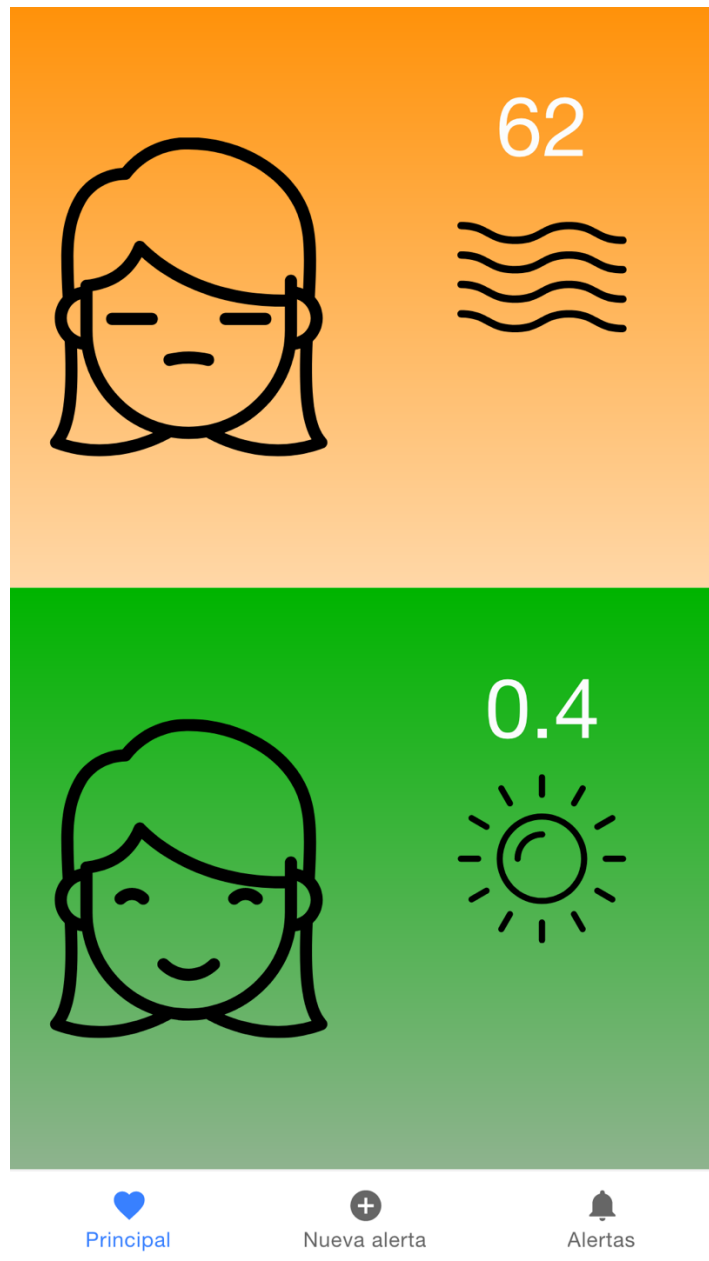
```
getLocation() {
  this.geolocation.getCurrentPosition().then((resp) => {
    this.latitud = resp.coords.latitude;
    this.longitud = resp.coords.longitude;

    // Call to air pollution web service
    this.loadAirInfo();

    // Call to UV index web service
    this.loadSunInfo();

    console.log(`Latitud ${this.latitud} Longitud ${this.longitud}`);
  }).catch((error) => {
    console.log('Error getting location', error);
  });
}
```

Cuando tenemos la geolocalización se llama a los servicios que proporcionan la información



Aspecto de la pantalla principal una vez implementada

5.2.2 Nueva alerta

La pantalla de nueva alerta nos presenta un formulario con los campos necesarios para ingresar el nombre de un lugar, la variable medioambiental a medir y el índice máximo a partir del cuál se activará la alerta. El formulario posee un sistema de validación y no permitirá guardar si no están rellenos los campos requeridos.

Dos botones nos ofrecen la posibilidad de guardar la alerta o descartarla y volver a la página de la que procedemos.

Nueva alerta


Localización


Indice


Variable ambiental

Aire

UV

 Principal

 Nueva alerta

 Alertas

Formulario para guardar una nueva alerta

En el archivo Typescript de la página destaca la llamada al servicio REST *Geocoder* que permite obtener las coordenadas geográficas de un lugar a partir del nombre del mismo. En el *request* de la llamada incluimos la cadena de texto del lugar que ha informado el usuario en el formulario.

Una vez que disponemos de todos los datos se procede a guardar la información de la nueva alarma en el *storage* local el dispositivo. Para ello hemos implementado un servicio que analizaremos más adelante.

```

onSaveAlert() {
  let latitude;
  let longitude;

  this.locationService.getGeocode(this.alerta.localizacion).subscribe( resp => {

    // Obtain coordinates for location
    latitude = resp.Response.View[0].Result[0].Location.NavigationPosition[0].Latitude;
    longitude = resp.Response.View[0].Result[0].Location.NavigationPosition[0].Longitude;
    latitude = Math.round(latitude * 100) / 100;
    longitude = Math.round(longitude * 100) / 100;

    // Add new alert with all data
    let newAlert = {localizacion: this.alerta.localizacion, indice: this.alerta.indice, variable: this.alerta.variable};
    this.dataLocal.saveAlert(newAlert);

    // Reset values
    this.alerta = {localizacion: '', indice: null, variable: 'Aire', lat: 0, lon: 0};

    // Go back
    this.navCtrl.back();
    this.presentToast("Alerta guardada");
  });
}

```

Método para guardar una nueva alerta previa llamada al servicio Geocoder



5.2.3 Lista de alertas

La última de las pantallas principales de la aplicación nos muestra el listado de las alertas que hay registradas hasta el momento.

La página está compuesta por un componente *ion-item-sliding* por cada registro de alerta almacenado en la base de datos local. Cada uno de los elementos de la lista contiene un componente *ion-item-options* que permite al usuario descubrir las opciones de editar y eliminar una alerta deslizando un dedo sobre uno de los elementos. Si se pulsa sobre el botón de eliminar, la alerta se borrará de la base de datos y si, por el contrario, se pulsa sobre el icono de editar, se abrirá una nueva página con un formulario sobre los que se cargarán los datos de la alarma seleccionada para que el usuario pueda modificarlos.

En el Typescript de la pantalla del listado de alertas podemos destacar la inclusión de un método que comprueba si se cumplen las condiciones para lanzar una notificación para cada una de las alertas de la lista. A este método se le llamará periódicamente de modo automático con la geolocalización actual del usuario y los índices medioambientales de esa posición para realizar dichas comprobaciones.

Además, podemos destacar la implementación de un método que proporciona la funcionalidad necesaria para emitir notificaciones al usuario con un mensaje personalizado. Este método aprovecha las prestaciones del plugin de **Cordova Local Notifications** y será invocado cada vez que se cumplan las condiciones estipuladas en una de las alertas de la base de datos local.

Lista de alertas		
Aire	4	Madrid
UV	5	Sevilla
UV	8	Barcelona
	Alicante	 
Aire	5	San Sebastián



Página de listado de alertas con opciones deslizables

```

// Checks if an alert should be fired for the current location
checkAlerts(maxDistance = 10, currentLat, currentLon, currentAirIndex, currentUvIndex) {
  this.alerts.forEach((alert) => {
    // Check if the user is in the radius of an alert
    let distance = this.utilsService.distanceFromTwoPoints(currentLat, currentLon, alert.lat, alert.lon);
    if(parseFloat(distance) < maxDistance) { // The user is inside the radius of an alert
      if(alert.variable == "Aire" && currentAirIndex > alert.indice) {
        this.sendNotification('AirSun', 'La contaminación atmosférica es superior a la alerta programada');
      }
      if(alert.variable == "UV" && currentUvIndex > alert.indice) {
        this.sendNotification('AirSun', 'La radiación ultravioleta es superior a la alerta programada');
      }
    }
  });
}

```

Método para comprobar si se cumplen las condiciones de una alerta

```
// Sends a notification to the user terminal
sendNotification(titulo, texto) {
  this.localNotifications.schedule({
    title: titulo,
    text: texto,
    trigger: { in: 5, unit: ELocalNotificationTriggerUnit.SECOND },
    foreground: true
  });
}
```

Método para emitir notificaciones al usuario

5.2.4 Editar alerta

Por último, está la pantalla para la edición de una alerta previamente seleccionada del listado. Se trata básicamente del formulario de creación de alerta, pero en este caso el campo de localización está deshabilitado y sólo se permite al usuario modificar el valor del índice máximo y la variable medioambiental a medir. Al pulsar sobre el botón de guardar no se añade una alerta nueva, sino que se modifican los datos de la alerta seleccionada.

Editar alerta

Localización
Sevilla

Índice
5

Variable ambiental

Aire

UV

Página de edición de alerta con campo de localización deshabilitado

5.3 Servicios

Para la implementación de la aplicación se han creado cinco clases auxiliares de servicios que facilitan la organización del proyecto y la realización del resto de tareas necesarias para el correcto funcionamiento de la app.

Llamadas a servicios web REST para la obtención de datos, operaciones con la base de datos local y utilidades auxiliares para el cálculo de distancias se han tratado como clases de servicios independientes que veremos a continuación.

5.3.1 Web Services

Nuestra aplicación obtiene de dos APIS los datos relativos al índice de contaminación atmosférica y el índice de radiación ultravioleta. Ambas APIS están expuestas como servicio web REST y se pueden consultar de forma sencilla y gratuita hasta determinado volumen de consultas.

OpenUV proporciona en tiempo real el índice de radiación ultravioleta para unas coordenadas geográficas determinadas. En nuestro servicio que accede a dicha API se llama al *endpoint* proporcionado por OpenUV incluyendo las coordenadas de latitud y longitud del terminal. Además, se incluye un *token* de autenticación en la cabecera.

```
getUvIndex(lat: number, lng: number){
  let headers = new HttpHeaders().set("x-access-token", SUNKEY);
  return this.http.get<ResponseUvIndex>(`https://api.openuv.io/api/v1/uv?lat=${lat}&lng=${lng}`, {headers});
}
```

Request para la consulta de la API de OpenUV

Por su parte, **AirVisual API** proporciona diversos datos sobre la contaminación atmosférica en cualquier zona del mundo. De entre la multitud de métodos que ofrece, nosotros llamamos a aquel que nos proporciona el índice de contaminación atmosférica de un punto geográfico incluyendo en la llamada solamente las coordenadas de latitud y longitud. Al igual que en OpenUV ha de incluirse una clave de autenticación.

```
getNearestCityDataGps(lat: number, lng: number){
  return this.http.get<ResponseNearestCity>(`https://api.airvisual.com/v2/nearest_city?lat=${lat}&lon=${lng}&key=${AIRKEY}`);
}
```

Request para la consulta de la API de AirVisual

La última API a la que recurrimos es la que ofrece **Here**, un servicio web que, entre otras funcionalidades, permite la posibilidad de convertir una dirección en coordenadas geográficas.

En la llamada incluimos la cadena con la localización que ha introducido el usuario en el campo de la alerta, un id de aplicación y un código de aplicación; estos dos últimos datos requeridos por la API de Here para su funcionamiento.

```
getGeocode(place: string){
  return this.http.get<ResponseLocation>(`https://geocoder.api.here.com/6.2/geocode.json?
  searchtext=${place}&app_id=${APPID}&app_code=${APPCODE}&gen=9`);
}
```

Request para la consulta de la API de Here

5.3.2 Servicio de almacenamiento local

Para el almacenamiento de las alertas creadas por el usuario hemos recurrido a la opción que proporciona Ionic con su **Ionic Storage**. Se trata de una forma sencilla de guardar información mediante pares de clave-valor y objetos JSON. Como no necesitamos un sistema de almacenamiento complejo para la información que deseamos guardar en nuestra aplicación, hemos decidido que la opción de Ionic Storage era la más práctica para nuestros requisitos.

En el servicio que hemos creado para almacenar y gestionar los datos hemos implementado métodos para guardar, eliminar, editar y recuperar alertas. Para el borrado y la edición de una alerta existente recurrimos al índice de la alerta que recuperamos del propio *array* de alertas. Posiblemente, el método más interesante sea el del guardado de una nueva alerta, pues antes de almacenarla en la base de datos realizamos una comprobación de que no existe la misma alerta, para evitar así duplicados.

```
saveAlert(alert: Alerta){
  // Avoids duplicating alerts
  let exists = false;
  this.alerts.forEach(function(bbddAlert) {
    if([bbddAlert.localizacion == alert.localizacion && bbddAlert.indice == alert.indice
    && bbddAlert.variable == alert.variable]){
      exists = true;
    }
  });

  // If the alerts is not on DDBB yet it is added
  if(!exists) {
    this.alerts.push(alert);
    this.storage.set('alerts', this.alerts);
  }
}
```

Método para la grabación de una alerta nueva

5.3.3 Servicio de utilidades

En último lugar, añadimos una clase de utilidades para la implementación de métodos auxiliares que nos proporcionen alguna funcionalidad extra que podamos necesitar en algún momento de nuestro desarrollo.

Por el momento, esta clase incluye un método que devuelve de la distancia en kilómetros entre dos coordenadas geográficas. Dicho método se utilizará cuando se haya de comprobar si el usuario se encuentra dentro del área seleccionada para una de sus alertas personalizadas y, en caso de superarse el índice establecido en la alerta, haya de lanzarse una notificación.

```
distanceFromTwoPoints(lat1, lon1, lat2, lon2) {  
  let rad = function(x) {return x*Math.PI/180;}  
  let R = 6378.137; //Earth radius in kms  
  let dLong = rad( lon2 - lon1 );  
  let dLat = rad( lat2 - lat1 );  
  let a = Math.sin(dLat/2) * Math.sin(dLat/2) + Math.cos(rad(lat1)) * Math.cos(rad(lat2))  
  * Math.sin(dLong/2) * Math.sin(dLong/2);  
  let c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));  
  let d = R * c;  
  return d.toFixed(2);  
}
```

Método para el cálculo en kilómetros de dos localizaciones

5.4 Estado del proyecto

El proceso de implementación de la aplicación transcurrió sin complicaciones durante la mayor parte del tiempo. Las funcionalidades definidas en las etapas anteriores eran implementadas en los plazos previstos y no se produjo ningún desvío respecto a la planificación preestablecida.

Todo el desarrollo iba bien hasta el momento de tener que implementar la funcionalidad que permitía al dispositivo consultar a intervalos de X minutos la geolocalización del dispositivo, para poder comprobar si se cumplían las condiciones para emitir alguna de las alertas registradas en la base de datos.

Investigando las posibilidades con las que contábamos para la implementación de esta funcionalidad comprendimos que la app, de algún modo, habría de seguir ejecutándose en segundo plano y no quedar pausada. Por desgracia, entre todas las herramientas que ofrece Ionic no existía ningún mecanismo que nos permitiera mantener la aplicación activa en segundo plano o, al menos, ninguno que nos ofreciera garantías.

La única alternativa real que encontramos es un *plugin* de **Cordova** llamado **Background Mode** [13] (“Background Mode,” n.d.) En teoría, soporta tanto Android como iOS y permite mantener la aplicación corriendo en segundo plano mientras “escucha” eventos que son lanzados cuando la aplicación cambia de estado.

Este *plugin* nos proporcionaba lo que necesitábamos para poder implementar la última funcionalidad de nuestra app. Sin embargo, al tratarse de un *plugin* poco popular, nos encontramos con escasa documentación acerca de su implementación. Por más que lo testeamos, tanto en máquinas virtuales como en dispositivos físicos reales, no logramos los resultados previstos y nos vimos en la obligación de desistir ante la inminente llegada de la fecha límite de entrega.

Por tanto, a pesar de las horas dedicadas a este último hito de nuestra fase de desarrollo, hemos prescindido temporalmente de esta funcionalidad, siendo la única de las planificadas que no se ha llegado a implementar por cuestiones técnicas. Tal vez, habríamos de abordar el problema desde otra perspectiva y buscar soluciones alternativas que no impliquen la utilización de *plugins* poco contrastados y sin ningún tipo de soporte técnico.

5.5 Actualización de implementación (20/12/2019)

Tal y como comentamos en el apartado anterior, nos vimos obligados a prescindir de una de las funcionalidades de nuestra aplicación de cara a la primera entrega de la implementación. Al no encontrar ninguna solución técnica que satisficiera adecuadamente las necesidades que pretende cubrir la aplicación y ante la proximidad de la fecha de entrega de la fase de implementación, decidimos aplazar temporalmente el desarrollo de dicha funcionalidad. Sin embargo, hemos seguido investigando el problema y, finalmente, hemos encontrado la solución que estábamos buscando.

Como señalábamos, nos hacía falta un mecanismo mediante el cual la aplicación pudiera acceder de forma periódica al hardware del terminal para obtener la geolocalización del dispositivo. Todo esto tendría que ejecutarse con la aplicación en segundo plano y sin la intervención del usuario.

Después de rechazar varias opciones, por impracticables o porque no cumplían con suficientes garantías con los resultados esperados, hemos encontrado una que sí satisface las necesidades de la aplicación y, además, lo hace sin impactar de forma negativa en los recursos del dispositivo. Se trata del *plugin* de Cordova **Background Geolocation** [14] (“Background Geolocation,” n.d.), que proporciona acceso a la geolocalización del dispositivo con la aplicación en segundo plano y ofrece diversos mecanismos para el ahorro de batería, como el activarse sólo cuando detecta movimiento.

Tras realizar una serie de pruebas y ver que funcionaba correctamente, hemos introducido la funcionalidad en nuestra aplicación creando un servicio exclusivo para el sistema de alertas (anteriormente en la página de *Lista de alertas*). Este servicio se activa cuando el usuario inicia la aplicación, aprovechando el método *ngOnInit()* de la página principal para invocarlo.

```
ngOnInit() {  
  // Obtain device location  
  this.getLocation();  
  
  // Init alerts service  
  this.alertsService.startBackgroundGeolocation();  
}
```

Invocación del servicio de alertas desde main.page.ts

En el archivo del servicio tenemos todos los métodos que utiliza el sistema de alertas, salvo el método que devuelve la distancia entre dos puntos geográficos, que se implementó en un servicio de utilidades aparte que facilita su posible reutilización futura.

El método *startBackgroundGeolocation()* es el principal del servicio de alertas. En él se configura el *plugin Background Geolocation*, se cargan las alertas del usuario almacenadas en la base de datos, se llaman a los servicios REST para la obtención de los datos de calidad del aire y radiación ultravioleta para la localización actual y, por último, se llama al método que cruza estos datos con las alertas y comprueba si alguna de ellas cumple con las condiciones y, por tanto, ha de emitirse una notificación al usuario.

```
async startBackgroundGeolocation() {
  const config: BackgroundGeolocationConfig = {
    desiredAccuracy: 10,
    stationaryRadius: 1,
    distanceFilter: 1,
    interval: 1200000, // check every 20 minutes
    debug: false, // enable this hear sounds for background-geolocation life-cycle.
    stopOnTerminate: false
  };

  // Load alerts from DDBB
  this.alerts = await this.dataLocal.loadAlerts();

  // BackgroundGeolocation Cordova plugin configuration
  this.backgroundGeolocation.configure(config).then(() => {
    this.backgroundGeolocation
      .on(BackgroundGeolocationEvents.location)
      .subscribe((location: BackgroundGeolocationResponse) => {

        this.airService.getNearestCityDataGps(location.latitude, location.longitude).subscribe( resp => {

          let airValue = resp.data.current.pollution.aqius;

          this.sunService.getUvIndex(location.latitude, location.longitude).subscribe( resp => {

            let sunValue = resp.result.uv;

            // Check whether current data satisfies any alert on DDBB
            this.checkAlerts(5, location.latitude, location.longitude, airValue, sunValue);

          });
        });
      });
  });

  // start recording location
  this.backgroundGeolocation.start();
});
```

Método principal del servicio de alertas

El método que comprueba las alertas del usuario utiliza la clase de utilidades para hallar la distancia en kilómetros entre la posición actual del dispositivo y cada una de las alertas personalizadas guardadas por el usuario. Si dicha distancia es menor que la establecida para el envío de una notificación (por defecto, 5 kilómetros) y los valores establecidos como mínimos en la alerta son superiores a los actuales, se envía una notificación al usuario aprovechando la funcionalidad que ofrece el *plugin Cordova Local Notifications* [15] (“Local Notifications,” n.d.).


```

// Checks if an alert should be fired for the current location
checkAlerts(maxDistance = 5, currentLat, currentLon, currentAirIndex, currentUvIndex) {

  this.alerts.forEach((alert) => {
    // Check if the user is in the radius of an alert
    let distance = this.utilsService.ditanceFromTwoPoints(currentLat, currentLon, alert.lat, alert.lon);

    if(parseFloat(distance) < maxDistance) { // The user is inside the radius of an alert

      if(alert.variable == "Aire" && currentAirIndex > alert.indice) {
        this.sendNotification('AirSun', `La contaminación atmosférica en ${alert.localizacion}
es superior a ${alert.indice}`);
      }

      if(alert.variable == "UV" && currentUvIndex > alert.indice) {
        this.sendNotification('AirSun', `La radiación ultravioleta en ${alert.localizacion}
es superior a ${alert.indice}`);
      }
    }
  });
}

```

Método que comprueba si se cumplen las condiciones en cada una de las alertas

```

// Sends a notification to the user terminal
sendNotification(titulo, texto) {
  this.localNotifications.schedule({
    title: titulo,
    text: texto,
    trigger: { in: 1, unit: ELocalNotificationTriggerUnit.SECOND },
    foreground: true
  });
}

```

Método para el envío de notificaciones

6. Pruebas

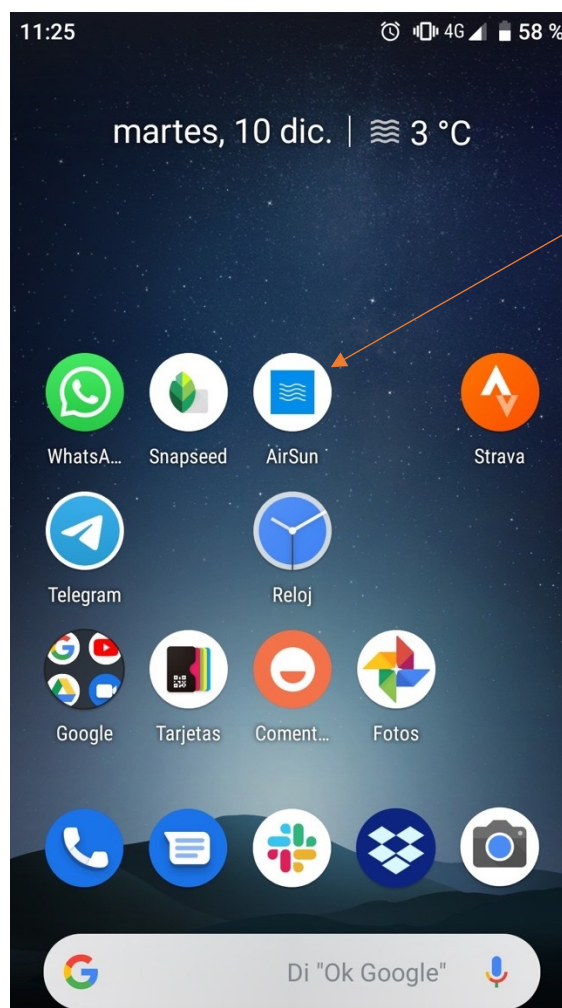
Después de la fase de implementación hemos procedido a realizar las pruebas pertinentes que aseguraran que la funcionalidad implementada tenía el comportamiento esperado. Valoramos la posibilidad de realizar pruebas unitarias con el *framework* de JavaScript **Jasmine**, pero después de los problemas sufridos con implementación del modo en segundo plano, con la demora que ello supuso, y observando que la fecha de entrega se acercaba, optamos por realizar unas pruebas basadas en casos de uso que nos aseguraran que la aplicación funcionaba de la forma que esperábamos.

A continuación, presentamos los resultados:

Caso 1 – Usuario desea consultar el índice de contaminación y/o la radiación UV

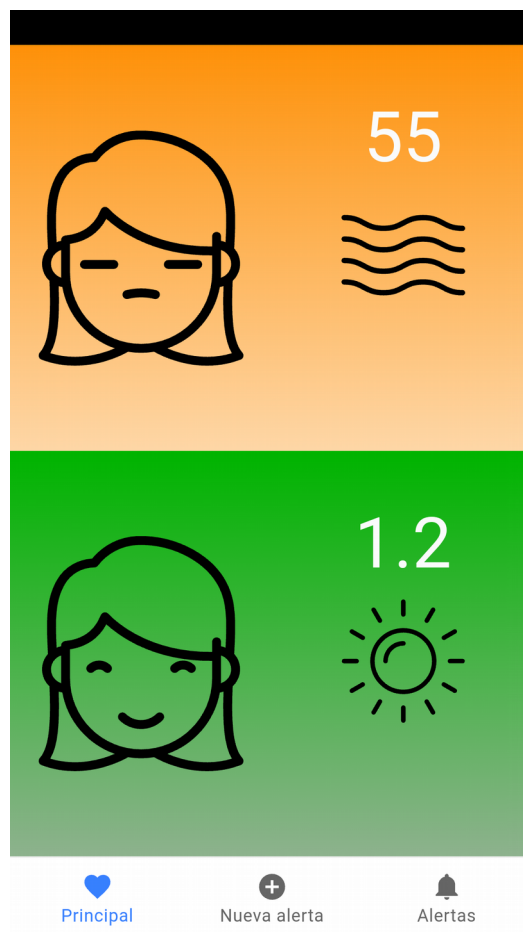
Resultado esperado: Al usuario se le presenta en pantalla el valor numérico del índice de contaminación atmosférica y la radiación ultravioleta, así como la representación de dicha medición mediante un código de colores y un emoticono.

Paso 1: El usuario clicca sobre el icono de la app en su dispositivo móvil.





Paso 2: El usuario recibe en pantalla la información en tiempo real acerca de la contaminación atmosférica y el índice de radiación UV del lugar en el que se encuentra.



Caso 2 – Usuario desea crear alerta para índice máximo de contaminación atmosférica en Barcelona.

Resultado esperado: Aparece la nueva alerta en el listado de alertas del usuario.

Paso 1: El usuario accede a la app y pulsa el botón del *tab* de “Nueva alerta”.

Paso 2: El usuario introduce la localización, el índice máximo a controlar por la alerta y selecciona “aire” en el *radiobutton*. A continuación, hace clic en el botón de guardar.

Nueva alerta

Localización
Barcelona

Indice
55

Variable ambiental

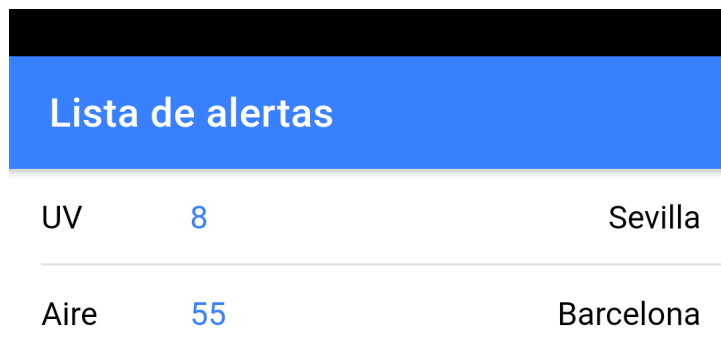
Aire

UV

GUARDAR CANCELAR

Principal Nueva alerta Alertas

Paso 3: Se carga automáticamente la pantalla de “Lista de alertas”. La alerta recién creada aparece en el listado.



Lista de alertas		
UV	8	Sevilla
Aire	55	Barcelona

Alerta guardada

Caso 3 – Usuario desea crear alerta para índice máximo de radiación ultravioleta en Madrid.

Resultado esperado: Aparece la nueva alerta en el listado de alertas del usuario.

Paso 1: El usuario accede a la app y pulsa el botón del *tab* de “Nueva alerta”.

Paso 2: El usuario introduce la localización, el índice máximo a controlar por la alerta y selecciona “UV” en el *radiobutton*. A continuación, hace clic en el botón de guardar.

The screenshot shows a mobile application form titled "Nueva alerta". It contains the following fields and controls:

- Localización:** A text input field containing "Madrid".
- Índice:** A text input field containing "3".
- Variable ambiental:** A group of radio buttons with two options: "Aire" (unselected) and "UV" (selected).
- Botones:** Two buttons at the bottom: "GUARDAR" (blue) and "CANCELAR" (red).

Four orange arrows point to the "Localización" field, the "Índice" field, the "UV" radio button, and the "GUARDAR" button.

At the bottom of the screen, there is a navigation bar with three items: "Principal" (heart icon), "Nueva alerta" (plus icon), and "Alertas" (bell icon).

Paso 3: Se carga automáticamente la pantalla de “Lista de alertas”. La alerta recién creada aparece en el listado.

The screenshot shows a mobile application screen titled "Lista de alertas" displaying a list of alerts in a table format:

UV	8	Sevilla
Aire	55	Barcelona
UV	3	Madrid

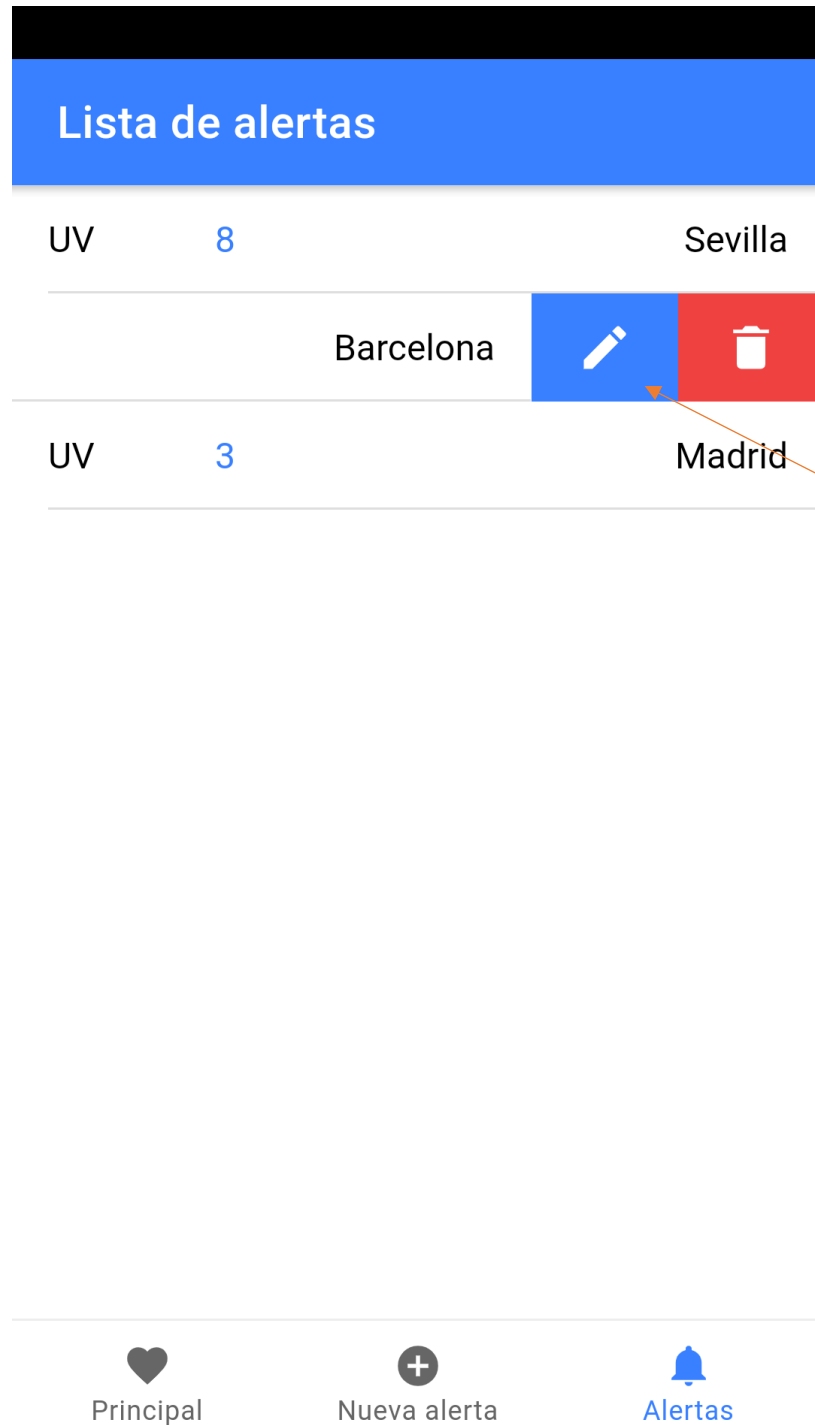
Alerta guardada

Caso 4 – Usuario desea editar una alerta y modificar el valor del índice máximo.

Resultado esperado: La alerta modificada se muestra con el nuevo valor en el listado de alertas del usuario.

Paso 1: El usuario accede a la app y pulsa el botón del *tab* de “Lista de alertas”.

Paso 2: El usuario desliza el dedo hacia la izquierda sobre el elemento de la lista a modificar. Sobre el elemento aparece un botón con el icono de un lápiz y lo pulsa.



Paso 3: El usuario modifica el valor del índice y pulsa el botón de guardar.

Editar alerta

Localización
Barcelona

Índice
45

Variable ambiental

Aire

UV

GUARDAR CANCELAR

Paso 4: Se carga automáticamente la pantalla de “Lista de alertas”. La alerta modificada aparece con el nuevo valor.

Lista de alertas		
UV	8	Sevilla
Aire	45	Barcelona
UV	3	Madrid

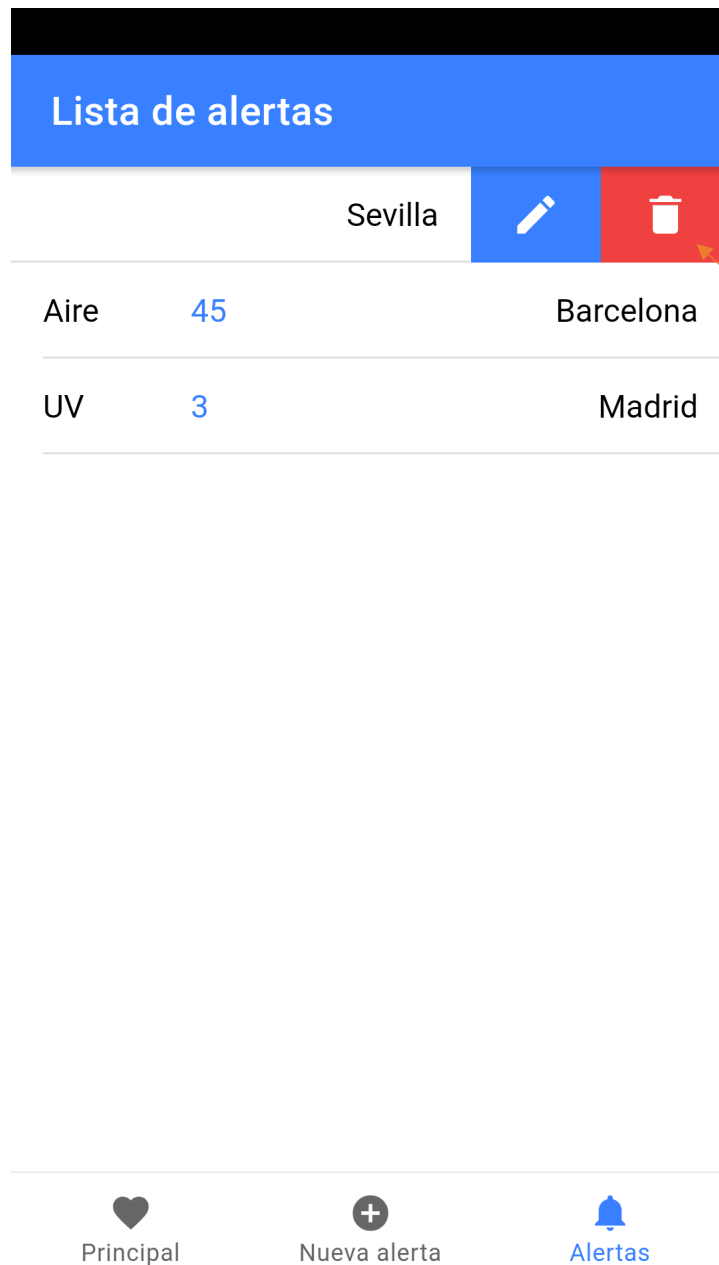
Principal Nueva alerta Alertas

Caso 5 – Usuario desea eliminar una alerta.

Resultado esperado: La alerta eliminada desaparece del listado de alertas del usuario.




Paso 1: El usuario accede a la app y pulsa el botón del *tab* de “Lista de alertas”.

Paso 2: El desliza el dedo hacia la izquierda sobre el elemento de la lista a eliminar. Sobre el elemento aparece un botón con el icono de una papelera y lo pulsa.



Paso 3: La alerta eliminada desaparece.

Lista de alertas		
Aire	45	Barcelona
UV	3	Madrid

 Principal  Nueva alerta  **Alertas**

6.1 Actualización de pruebas (20/12/2019)

Tras haber añadido la funcionalidad que nos faltaba hemos actualizado el capítulo dedicado a las pruebas, al igual que hiciéramos con el de implementación, para incluir la prueba realizada para testear esta nueva funcionalidad.

Caso 6 – Usuario se desplaza a la localización de una de sus alertas.

Resultado esperado: Las condiciones del lugar al que se desplaza cumplen con lo establecido en una de las alertas y el usuario recibe una notificación.

Paso 1: El usuario accede a la app y pulsa el botón del *tab* de “Nueva alerta”.

Paso 2: El usuario introduce la localización, el índice máximo a controlar por la alerta y selecciona “aire” en el *radiobutton*. A continuación, hace clic en el botón de guardar.

Nueva alerta

Localización
San Sebastián

Indice
9

Variable ambiental

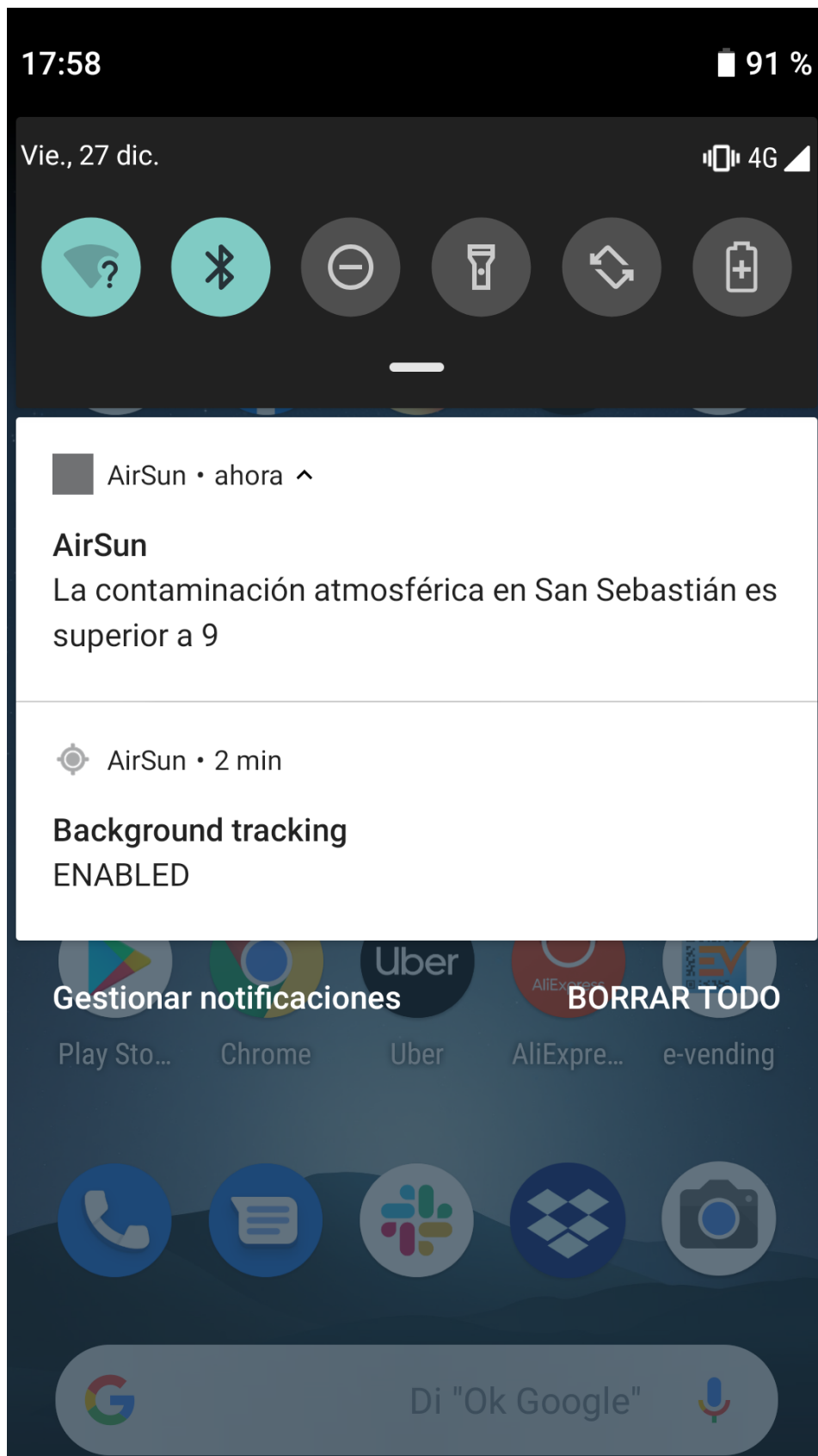
Aire

UV

GUARDAR CANCELAR

Principal Nueva alerta Alertas

Paso 3: El usuario se desplaza a la localización para la que ha establecido la alerta (en un radio de 5 kilómetros), como las condiciones se cumplen recibe la notificación.



7. Conclusiones

El desarrollo de la aplicación híbrida AirSun ha sido un proceso largo, complejo y no exento de inconvenientes. El camino ha estado lleno de obstáculos, especialmente durante la fase de implementación, y ha habido momentos de frustración como, por ejemplo, cuando no éramos capaces de implementar la funcionalidad para registrar en segundo plano la posición del dispositivo. Sin embargo, a pesar de los inconvenientes, la motivación por concluir el proyecto de manera satisfactoria no ha decaído en ningún momento y ese ímpetu nos ha impulsado para poder finalizar la aplicación tal y como la concebimos en un primer momento, incluida la ya mencionada funcionalidad que tanto se nos había resistido.

Lo primero que desearíamos destacar en este capítulo de reflexión es el aprendizaje que ha supuesto la elaboración de este trabajo. De hecho, si tuviéramos que quedarnos con una sólo palabra para definir esta aventura que toca a su fin sería, precisamente, «aprendizaje». Y es que, afrontar la tarea de desarrollar una aplicación móvil desde su fase de concepción —pasando por el análisis de los usuarios, el diseño y el prototipado—, hasta llegar a su implementación final, es toda una experiencia enriquecedora, que aporta en el ámbito formativo una serie de conocimientos que el mero estudio teórico de una materia no es capaz de proporcionar.

Entre aquello que hemos aprendido está la dificultad de mantener un proyecto fiel a su diseño inicial durante las siguientes etapas de desarrollo. A veces hay que hacer concesiones y desestimar ideas que parecían buenas en un inicio. Por tiempos, por razones técnicas o por no haber previsto algún inconveniente, nos vemos obligados a prescindir de alguna funcionalidad antes de la entrega del producto. En nuestro caso, creemos que hemos logrado mantener nuestra aplicación fiel al concepto inicial y aunque hemos sufrido dificultades durante la etapa de implementación consideramos el resultado final satisfactorio.

Entrando en una materia más técnica, hemos comprendido que, si bien un desarrollo híbrido multiplataforma aporta numerosas ventajas respecto al desarrollo nativo, también conlleva algunos inconvenientes. En lo referente a nuestra aplicación, hemos encontrado grandes complicaciones cuando hemos tenido que utilizar alguna de las funcionalidades que proporciona el hardware del dispositivo. En este sentido, creemos que para una aplicación sencilla, que no exija nada especial, el desarrollo híbrido con un framework como Ionic es más que suficiente; sin embargo, en el momento en el que alguna de las funcionalidades requiere el acceso al hardware del dispositivo, nos vemos obligados a ir más allá del propio entorno Ionic y recurrir a los plugins de Cordova, con los quebraderos de cabeza que puede llegar a suponer. Por tanto, considerando lo expuesto, creemos que hay que anticiparse y que se debe valorar a conciencia, ya en fases previas, si las tecnologías a utilizar durante el desarrollo son las adecuadas.

Otra lección que hemos aprendido tiene que ver con la metodología del desarrollo. El enfocar un proyecto de software desde una metodología *agile* tiene grandes ventajas. En nuestro caso, hemos realizado varias iteraciones entre las fases de desarrollo, lo que nos ha permitido mejorar el producto de forma gradual. El caso más significativo ha sido el de la consulta de la geolocalización en segundo plano. Como ya hemos mencionado, por motivos técnicos, no fue posible implementar dicha funcionalidad para la entrega de la

práctica vinculada a la fase de implementación. Sin embargo, una revisión posterior del problema y una actitud de constancia a la hora de investigar posibles soluciones nos ha llevado a completar la aplicación con todas las funcionalidades previstas al comienzo. De este modo, podemos considerar que hemos realizado el proyecto desde una perspectiva bastante fidedigna a lo que supone un desarrollo con metodología *agile* actualmente, con entregas progresivas completamente funcionales y que aportan mejoras de forma incremental en cada iteración.

Por otro lado, somos conscientes que aún en esta fase no estamos ante un producto perfecto. Observamos varias líneas de mejoras y nuevas funcionalidades que, tal vez, se podrían implementar en futuras iteraciones. Por citar algún ejemplo, nos gustaría que la aplicación fuera más personalizable por parte del usuario. Podríamos crear una página para la configuración de la aplicación, de modo que el usuario pudiera tomar decisiones como la frecuencia con la que la app debe comprobar la localización del dispositivo, el radio a partir del cual se comprueba una alerta en función de la localización establecida (actualmente fijo a cinco kilómetros) o desactivar temporalmente las notificaciones por completo si el usuario no desea ser molestado. En definitiva, se puede mejorar la aplicación otorgando al usuario final mayor control sobre cómo desea interactuar con el producto.

Para finalizar, no podemos concluir esta memoria sin destacar, frustraciones puntuales aparte, la gran satisfacción que nos ha reportado la elaboración del presente proyecto a lo largo de todas sus etapas y, especialmente, ver que el considerable esfuerzo realizado ha dado sus frutos en última instancia.

8. Glosario

Cordova. *Apache Cordova* es una plataforma de código abierto para el desarrollo móvil. Permite la creación de aplicaciones móviles mediante tecnologías propias del desarrollo web.

DCU. El Diseño Centrado en el Usuario es una metodología de diseño que enfoca la creación de productos en la solución de necesidades específicas de los usuarios.

Desarrollo híbrido. Alternativa intermedia al desarrollo nativo y al desarrollo web. Las aplicaciones híbridas son desarrolladas con lenguajes propios del desarrollo web para luego ejecutarse dentro de un contenedor nativo, que utiliza el motor del navegador del dispositivo.

Framework. Entorno de trabajo que sirve de esqueleto y estructura para organizar el desarrollo de una aplicación y que, en definitiva, facilita el proceso de implementación.

Geolocalización. Obtención de la localización geográfica de un objeto determinado, como puede ser un dispositivo móvil.

Journey Map. Herramienta de *Design Thinking* que refleja de forma visual el proceso que realiza el usuario desde que experimenta una necesidad hasta que concluye la utilización del servicio que utiliza para cubrir dicha necesidad.

Metodología agile. Metodología para el desarrollo de proyectos que implica múltiples iteraciones rápidas entre las distintas fases del proyecto, de modo que se obtiene un producto listo para producción después de cada iteración, que se va mejorando de forma incremental.

Radiación UV (Radiación ultravioleta). Un tipo de radiación electromagnética que forma parte de los rayos solares y cuyo exceso de exposición puede tener un efecto nocivo para la salud.

Request. En un servicio web es la petición que se manda a dicho servicio web que contiene la solicitud de información que se requiere de él.

Response. En un servicio web es la respuesta que devuelve el servicio ante una petición. Puede contener la información solicitada o un mensaje de error en el caso de que el proceso haya ido mal.

Tabs. Componente visual que facilita la navegación entre diferentes vistas de una aplicación.

Typescript. Lenguaje de programación que constituye un superconjunto de JavaScript, y que se utiliza para desarrollar aplicaciones tanto del lado del cliente como en servidor, estas últimas mediante Node.js.

Web Service. Conjunto de métodos con una determinada funcionalidad que pueden ser consumidos por otras aplicaciones a través de la Web mediante protocolos HTTP o HTTPS.

Wireframe. Esquema que representa la estructura visual de una aplicación y que muestra de forma esquemática el diseño y el ordenamiento de los elementos que la componen. Sirve de guía durante la fase de diseño de una aplicación.

9. Bibliografía

- [1] Organización Mundial de la Salud. (2018). Calidad del aire y salud. Accedido Octubre 6, 2019, desde [https://www.who.int/es/news-room/fact-sheets/detail/ambient-\(outdoor\)-air-quality-and-health](https://www.who.int/es/news-room/fact-sheets/detail/ambient-(outdoor)-air-quality-and-health)
- [2] Organización Mundial de la Salud. (2006). La carga mundial de morbilidad atribuible a la radiación ultravioleta solar (RUV). Accedido Octubre 6, 2019, desde <https://www.who.int/uv/publications/solaradgbd/es/>
- [3] La Vanguardia. (2019). Greta Thunberg avisa a los políticos: “Nos están fallando, no les perdonaremos.” Accedido Octubre 5, 2019, desde La Vanguardia website: <https://www.lavanguardia.com/natural/20190923/47584734916/greta-thunberg-cumbre-climatica-onu-antonio-guterres.html>
- [4] El País. (2019b). La mayoría de los españoles piensa que el cambio climático es la principal amenaza del mundo. Accedido Octubre 4, 2019, desde El País website: https://elpais.com/sociedad/2019/09/24/actualidad/1569321637_079127.html
- [5] El País. (2019a). España cumple al fin con Bruselas y envía el plan de calidad del aire con meses de retraso. Accedido Octubre 6, 2019, desde El País website: https://elpais.com/sociedad/2019/09/27/actualidad/1569580083_101700.html
- [6] Suzanna Haworth. (2019). Waterfall Vs Agile: ¿Cuál Metodología Debes Utilizar Para Tu Proyecto? Accedido Octubre 7, 2019, desde The Digital Project Manager website: <https://thedigitalprojectmanager.com/es/agile-frente-a-waterfall/>
- [7] Nielsen, L. (n.d.). Personas. Accedido Octubre 18, 2019, desde <https://www.interaction-design.org/literature/book/the-encyclopedia-of-human-computer-interaction-2nd-ed/personas>
- [8] Universitat Oberta, & Catalunya, D. (n.d.). Escenarios. Accedido Octubre 17, 2019, desde <http://design-toolkit.recursos.uoc.edu/es/escenarios/>
- [9] Melone, J. (2018). Journey mapping powers better design thinking. Accedido Octubre 21, 2019, desde <https://www.invisionapp.com/inside-design/journey-mapping-design-thinking/>
- [10] Espinosa, R. (2017). Ventaja Competitiva: qué es, claves, tipos y ejemplos. Accedido Octubre 28, 2019, desde <https://ingsoftwarekarlacevallos.wordpress.com/2015/06/04/uml-casos-de-uso/>
- [11] Cevallos, K. (2015). UML: Casos de Uso. Accedido Octubre 28, 2019, desde <https://ingsoftwarekarlacevallos.wordpress.com/2015/06/04/uml-casos-de-uso/>
- [12] José Jesús Pérez Rivas. (2015). Qué es y cómo empezar con Ionic Framework. Accedido Octubre 28, 2019, desde <https://www.phonegapSpain.com/que-es-y-como-empezar-con-ionic-framework/>

- [13] Background Mode. (n.d.). Accedido Diciembre 1, 2019, desde <https://ionicframework.com/docs/native/background-mode>
- [14] Background Geolocation. (n.d.). Accedido Diciembre 13, 2019, desde <https://ionicframework.com/docs/native/background-geolocation>
- [15] Local Notifications. (n.d.). Accedido Diciembre 3, 2019, desde <https://ionicframework.com/docs/native/local-notifications>

10. Anexos

10.1 Notas

El archivo .apk de la aplicación se encuentra en la carpeta APK.

El código del proyecto se encuentra en la carpeta air-and-sun-master.

El código también se encuentra disponible en GitHub (incluye README.md):

<https://github.com/joseanher81/air-and-sun>

10.2 Prerrequisitos para despliegue en un servidor local

- node.js
- ionic
- cordova

10.3 Instalación y despliegue en navegador

Ejecutar en el terminal, sobre la raíz del proyecto:

```
npm install
ionic serve
```

Importante; en caso de que falte algún plugin de Cordova probar con:

```
ionic cordova prepare
```

10.4 Compilación del archivo APK

- Crear las plataformas Android:

```
ionic cordova platform add android
```
- Crear los resources (copiando previamente nuestro fichero icon y splash):

```
ionic cordova resources
```
- Comprobar que existe carpeta “mipmap” con los archivos de icono y de splash en la siguiente ruta:

```
platforms ▶ android ▶ app ▶ src ▶ main ▶ res ▶ mipmap
```

- Compilar:

```
Debug
sudo ionic cordova build android --prod
```

```
Release
sudo ionic cordova build android --prod --release
```

10.5 Manual de usuario

AirSun es una aplicación móvil que proporciona al usuario, de manera clara y sencilla, información sobre el índice de contaminación atmosférica y el índice de radiación ultravioleta. La aplicación utiliza el GPS del dispositivo y varios servicios REST para obtener información en tiempo real de estos valores medioambientales en la localización del usuario. Además, permite al usuario configurar alertas para recibir notificaciones cuando los índices sobrepasen el valor máximo determinado para una localización establecida.

La información principal sobre el índice de contaminación atmosférica y el índice de radiación ultravioleta se muestra en la pantalla principal de la aplicación. Los valores se informan mediante un índice numérico, el icono de un rostro y un color de fondo significativo, que varían según la gravedad del índice medido.



Pantalla principal

La navegación en la aplicación se realiza mediante un sistema de *tabs* que nos permite desplazarnos a las pantallas de creación de nueva alerta y de listado de alertas existentes.

La creación de una alerta nueva es un proceso muy sencillo. Tan sólo hay que rellenar en los campos correspondientes el nombre del lugar para el que se quiere establecer la alerta, el valor del índice a partir del cual enviar una notificación al usuario y el factor medioambiental a medir.

Nueva alerta


Localización


Índice


Variable ambiental

Aire

UV

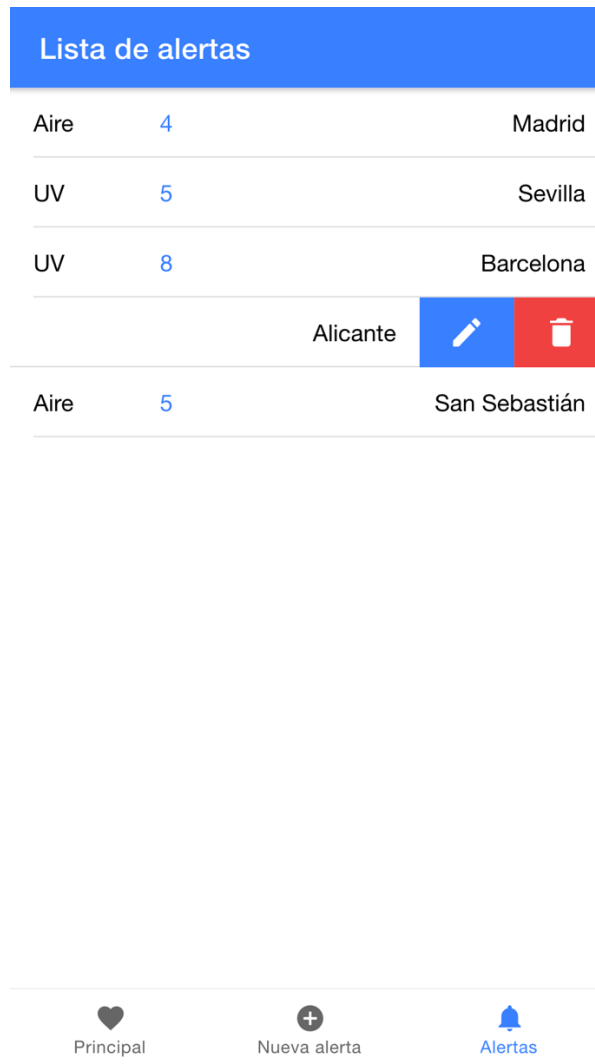
 Principal

 Nueva alerta

 Alertas

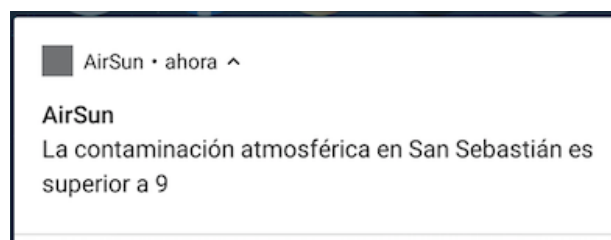
Pantalla de nueva alerta

En la pantalla de alertas encontraremos un listado con las alertas almacenadas en la memoria del dispositivo. Deslizando un dedo sobre uno de los elementos de la lista aparecerán las opciones para editar y borrar la alerta seleccionada.



Pantalla de listado de alertas

Si en algún momento el usuario se desplaza a una localización para la cual existe una alerta y se cumplen las condiciones medioambientales establecidas, se avisará al usuario mediante una notificación.



Notificación de alerta personalizada