UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Desarrollo de una aplicación que muestre datos de contaminación ambiental

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Xie Qiu, Zhe Hao

Tutor/a: Vidal Oriola, Germán Francisco

CURSO ACADÉMICO: 2022/2023

Resum

La contaminació ambiental o atmosfèrica és un problema generalitzat en les grans ciutats. Normalment es mesuren els nivells de partícules (PM2.5 i PM10), diòxid de nitrogen (NO2) i diòxid de sofre (SO2).

Amb aquest Treball de Fi de Grau es pretén desenvolupar una aplicació mòbil nativa senzilla que permeta la consulta dels valors d'aquests contaminants per a una ubicació específica. L'objectiu pricipal és proporcionar als usuaris informació actualitzada sobre la qualitat de l'aire i altres contaminants en temps real, perquè puguen prendre decisions informades. En la seua versió inicial, l'aplicació permetrà als usuaris consultar els valors dels contaminats tant per a la seua ubicació com per a ubicacions marcades com a favorites.

El desenvolupament de l'aplicació seguirà una metodologia àgil *Scrum*, la qual cosa garantirà l'eficiència i flexibilitat en el procés de desenvolupament.

Paraules clau: aplicació mòbil, contaminació ambiental, qualitat de l'aire, Scrum

Resumen

La contaminación ambiental o atmosférica es un problema generalizado en las grandes ciudades. Normalmente se mide el nivel de partículas (PM2.5 y PM10), el dióxido de nitrógeno (NO2) y el dióxido de azufre (SO2).

Con este Trabajo de Fin de Grado se pretende desarrollar una aplicación móvil nativa sencilla que permita la consulta de los valores de dichos contaminantes para una ubicación específica. El objetivo principal es proporcionar a los usuarios información actualizada sobre la calidad del aire y otros contaminantes en tiempo real, para que puedan tomar decisiones informadas. En su versión inicial, la aplicación permitirá a los usuarios consultar los valores de los contaminantes tanto para su ubicación como para ubicaciones marcadas como favoritas.

El desarrollo de la aplicación seguirá una metodología ágil *Scrum*, lo que garantizará la eficiencia y flexibilidad en el proceso de desarrollo.

Palabras clave: aplicación móvil, contaminación ambiental, calidad del aire, Scrum

Abstract

Environmental or atmospheric pollution is a widespread problem in large-sized cities. The levels of particles (PM2.5 and PM10), nitrogen dioxide (NO2) and sulfur dioxide (SO2) are typically measured.

This Final Degree Project aims to develop a simple native mobile application that allows users to check the values of these pollutants for a specific location. The main objective is to provide users with up-to-date information on air quality and other pollutants in real time, enabling them to make informed decisions. In its initial version, the application will allow users to check pollutant values for both their current location and marked favorite locations.

The application development will follow an agile *Scrum* methodology, ensuring efficiency and flexibility throughout the development process.

Key words: mobile app, environmental pollution, air quality, Scrum

Índice general

Índice general II
Índice de figuras IV
Índice de tablas V

1 Introducción 1
1.1 Motivación . . . 1
1.2 Objetivos . . . 2
1.3 Estructura de la memoria . . . 2
2 Estudio estratégico 4
2.1 Análisis de las aplicaciones . . . 4
2.1.1 BreezoMeter . . . 4
2.1.2 AirCare . . . 6
2.1.3 AirVisual . . . 8
2.1.4 Plume Labs . . . 9
2.2 Comparativa de las aplicaciones . . . 11
2.3 Propuesta . . . 12
3 Metodología de trabajo 13
3.1 Enfoque tradicional . . . 13
3.2 Enfoque ágil . . . 14
3.3 Scrum . . . 14
4 Análisis y especificación de requisitos 17
4.1 Especificación de Requisitos Software . . . 17
4.1.1 Propósito . . . 17
4.1.2 Ámbito y descripción del producto . . . 17
4.1.3 Modelo de Dominio . . . 18
4.1.4 Límites del Sistema . . . 19
4.1.5 Restricciones del Sistema . . . 20
4.1.6 Lista de Características del Sistema . . . 20
4.1.7 Diagramas de Casos de Uso . . . 23
4.1.8 Pruebas de Aceptación . . . 26
5 Arquitectura y diseño de la aplicación 27
5.1 Arquitectura multicapa . . . 27
5.2 Diseño de las interfaces de usuario . . . 28
6 Implementación 32
6.1 Tecnología utilizada . . . 32
6.1.1 React Native . . . 32
6.1.2 Expo . . . 33
6.1.3 Node.js y Express.js . . . 34
6.1.4 MongoDB . . . 34
6.1.5 Git . . . 34
6.2 Aspectos relevantes de la implementación . . . 35
6.2.1 Desarrollo del frontend . . . 35

6.2.2 Integración de la API de AQI . . . 37
6.2.3 Comunicación entre el cliente y servidor . . . 40
7 Pruebas **41**
7.1 Metodología ATDD . . . 41
7.2 Pruebas unitarias . . . 42
8 Despliegue y mantenimiento **44**
8.1 Mantenimiento y evolución de software . . . 44
8.2 Lanzamiento del producto . . . 44
8.3 Descripción del producto final . . . 45
8.3.1 Pantallas Login y Registro . . . 45
8.3.2 Pantallas Inicio y Buscar . . . 45
8.3.3 Pantalla Ajustes . . . 47
8.3.4 Pantalla Detalles . . . 47
9 Conclusiones y trabajo futuro **49**
9.1 Conclusiones . . . 49
9.2 Relación con las asignaturas cursadas . . . 50
9.3 Trabajo futuro . . . 51
Bibliografía **52**

Apéndices

A Pruebas de Aceptación **54**
B Relación con los ODS **62**

Índice de figuras

2.1 Pantallas de BreezoMeter . . . 5
2.2 Pantallas de BreezoMeter . . . 6
2.3 Pantallas de Aircare . . . 7
2.4 Pantalla del chatbot inteligente . . . 7
2.5 Pantallas de AirVisual . . . 8
2.6 Pantallas de AirVisual . . . 9
2.7 Pantallas de Plume Labs . . . 10
2.8 Pantallas de Plume Labs . . . 11

3.1 Diagrama del enfoque tradicional o en cascada . . . 13
3.2 Diagrama del enfoque ágil . . . 14
3.3 Marco de trabajo Scrum . . . 15
3.4 Tablero canvan . . . 15
3.5 Tablero canvan . . . 15

4.1 Diagrama del modelo de dominio . . . 18
4.2 Diagrama de contexto . . . 20
4.3 Diagrama de casos de uso principales . . . 24
4.4 Diagrama de casos de uso para la gestión de ciudades . . . 24
4.5 Diagrama de casos de uso para la gestión de usuarios . . . 25
4.6 Diagrama de casos de uso para la gestión de las notificaciones . . . 25

5.1 Arquitectura en tres capas . . . 27
5.2 Mockups Pantalla de inicio y ajustes . . . 29
5.3 Mockups Pantalla de inicio y ajustes . . . 29
5.4 Mockups Pantalla de búsqueda . . . 30
5.5 Mockup Pantalla de detalles . . . 31
5.6 Mockup Notificación . . . 31

6.1 Arquitectura simplificada de React Native . . . 33

7.1 Ciclo del enfoque ATDD . . . 42

8.1 Pantallas de Clean Breath . . . 45
8.2 Pantallas de Clean Breath . . . 46
8.3 Pantalla de Ajustes . . . 47
8.4 Pantalla de Detalles . . . 48
8.5 Pantalla de inicio actualizada . . . 48

Índice de tablas

2.1 Tabla comparativa de características 11

4.1 Atributos de la clase Usuario 19
4.2 Atributos de la clase Ciudad 19
4.3 Descripción del requisito Acceder a la pantalla de inicio 21
4.4 Descripción del requisito Acceder a ajustes 21
4.5 Descripción del requisito Buscar una ciudad 21
4.6 Descripción del requisito Añadir ciudad a la lista de favoritos 21
4.7 Descripción del requisito Eliminar ciudad de la lista de favoritos 22
4.8 Descripción del requisito Consultar los detalles de la calidad del aire de una ciudad 22
4.9 Descripción del requisito Registrarse como usuario 22
4.10 Descripción del requisito Acceder como invitado 22
4.11 Descripción del requisito Iniciar sesión 23
4.12 Descripción del requisito Activar/desactivar notificación diaria 23
4.13 Descripción del requisito Activar/desactivar alertas 23

A.1 Prueba de aceptación *Vista de la pantalla principal* 54
A.2 Prueba de aceptación *Barra inferior de navegación* 54
A.3 Prueba de aceptación *Menú superior* 54
A.4 Prueba de aceptación *Activar notificaciones diarias* 55
A.5 Prueba de aceptación *Desactivar notificaciones diarias* 55
A.6 Prueba de aceptación *Ajustar hora de la notificación diaria* 55
A.7 Prueba de aceptación *Se muestra la ubicación* 56
A.8 Prueba de aceptación *Se muestra el índice de calidad del aire en la tarjeta* . . . 56
A.9 Prueba de aceptación *El color se corresponde con la calidad del aire* 56
A.10 Prueba de aceptación *Se muestra el índice de calidad del aire* 56
A.11 Prueba de aceptación *Se muestran datos de otros contaminantes* 57
A.12 Prueba de aceptación *Se muestra recomendaciones de salud* 57
A.13 Prueba de aceptación *El color cambia según la calidad del aire* 57
A.14 Prueba de aceptación *El usuario puede buscar ciudades por el nombre* 57
A.15 Prueba de aceptación *Se muestran las opciones que coinciden con la búsqueda* . 58
A.16 Prueba de aceptación *El usuario puede guardar una ciudad como favorito* . . . 58
A.17 Prueba de aceptación *El usuario puede eliminar una ciudad como favorito* . . . 58
A.18 Prueba de aceptación *El usuario puede ver las ciudades guardadas en la pantalla de inicio* 59
A.19 Prueba de aceptación *El usuario introduce un nombre de usuario no válido* . . 59
A.20 Prueba de aceptación *El usuario introduce una contraseña no válida* 59
A.21 Prueba de aceptación *Las contraseñas introducidas no coinciden* 60
A.22 Prueba de aceptación *Registro correcto* 60
A.23 Prueba de aceptación *Inicio fallido* 60
A.24 Prueba de aceptación *Inicio de sesión con éxito* 61
A.25 Prueba de aceptación *Inicio de sesión con éxito* 61
A.26 Prueba de aceptación *Cerrar sesión* 61

B.1 Tabla de grado de relación del trabajo con los ODS 62

CAPÍTULO 1
Introducción

En las últimas décadas, la contaminación se ha convertido en un tema cada más importante y en un desafío global. La contaminación ambiental o atmosférica está siendo una preocupación de suma importancia, especialmente en las grandes y medianas ciudades. La creciente preocupación por los efectos nocivos en la salud humana hace necesario el desarrollo de herramientas que proporcionen información precisa y actualizada sobre la calidad del aire, ayudando de esta manera a los ciudadanos a tomar decisiones informadas sobre su salud y bienestar.

En este contexto, el presente Trabajo de Fin de Grado se centra en el desarrollo de una aplicación móvil que permita la consulta de la calidad del aire así como de los valores de los diferentes contaminantes en una ubicación. Esta aplicación tiene como objetivo proporcionar a los usuarios información relevante, permitiéndoles tomar decisiones informadas y adoptar medidas para proteger su bienestar.

1.1 Motivación

Hoy en día, ya existen numerosas plataformas y aplicaciones que aportan información sobre la calidad del aire. Sin embargo, creemos que este tipo de herramientas, que contribuyen al bienestar de todos, siempre son bienvenidas. Nos motiva de la misma manera poder abordar esta problemática a través de la tecnología. Estamos comprometidos en desarrollar una herramienta que permita a los usuarios acceder a datos precisos y relevantes sobre la calidad del aire. Creemos que al proporcionar información detallada y actualizada, estamos capacitando a los usuarios para la toma de decisiones adecuadas sobre la protección de su bienestar y del entorno.

Nuestro objetivo final es ofrecer una aplicación móvil funcional y fácil de usar. Estamos entusiasmados con la posibilidad de desarrollar una herramienta que sea realmente útil y de ayuda en la mejora de la calidad de vida, respetando el medio ambiente.

Este Trabajo de Fin de Grado representa una oportunidad valiosa para aplicar nuestros conocimientos en el desarrollo de aplicaciones móviles y trabajar en pro de una causa socialmente relevante.

1.2 Objetivos

El presente trabajo propone una solución software para: i) proporcionar información detallada sobre la calidad del aire y sus contaminantes ii) informar al usuario de alteraciones drásticas en la calidad del aire, para dotar así a los usuarios de los conocimientos necesarios para tomar decisiones responsables.

En consecuencia, para asegurar el adecuado desarrollo y funcionamiento de la aplicación, es crucial cumplir con una serie de requerimientos claramente definidos:

- Desarrollar una herramienta funcional que aporte información sobre la calidad del aire y proporcione recomendaciones simples que ayuden a los usuarios en el día a día.
- Diseñar una interfaz agradable, intuitiva y fácil de usar.
- Implementar un sistema de notificaciones que informe sobre la calidad del aire y/o alerte de casos extremos en la zona que afecte al usuario.
- Aplicar la metodología ágil *Scrum* en el proceso de desarrollo de la aplicación, asegurando eficiencia y flexibilidad en la entrega de funcionalidades y mejoras continuas.
- Promover la conciencia ambiental y fomentar hábitos saludables en la sociedad.

1.3 Estructura de la memoria

A continuación, describimos la organización de los contenidos restantes de la memoria:

- **Capítulo 2. Estudio estratégico:** en este capítulo se realizará un estudio de las soluciones existentes a esta problemática. Se analizarán las características principales de cada solución y se planteará la propuesta de la solución.
- **Capítulo 3. Metodología de trabajo:** aquí se describirá el enfoque tradicional y el enfoque ágil de la metodología Scrum utilizada para el desarrollo de la aplicación. Se explicarán los principios y prácticas adoptadas, así como la organización y la planificación de los *sprints*.
- **Capítulo 4. Análisis y especificación de los requisitos:** en este capítulo se detallarán los principales requisitos del sistema. Se definirán los casos de uso, diagramas de clases, estableciendo las bases para el diseño de la aplicación.
- **Capítulo 5. Arquitectura y diseño de la aplicación:** en este apartado se presentará la arquitectura de la aplicación, y la descripción de los componentes principales.
- **Capítulo 6. Implementación:** aquí se presentarán las tecnologías utilizadas y se detallarán diferentes aspectos y funcionalidades implementadas.
- **Capítulo 7. Pruebas:** en este capítulo se presentará brevemente el conjunto de pruebas que se han desarrollado para la identificación de errores y fallos de ejecución. Asimismo, se comentarán las medidas tomadas para garantizar la calidad del sistema.

- **Capítulo 8. Despliegue y mantenimiento:** en este capítulo se explicará el proceso de despliegue de la aplicación. Se discutirán las tareas de mantenimiento necesarias para asegurar el correcto funcionamiento de la aplicación.
- **Capítulo 9. Manual de uso:** presenta una pequeña *demo* con el fin de que el usuario se familiarice con el uso de la aplicación, mostrando capturas del producto final.
- **Capítulo 10. Conclusiones y trabajo futuro:** Por último, en este capítulo se reflexionará sobre los resultados obtenidos. También se sugerirán posibles mejoras para el futuro.

Estos capítulos estructurarán la memoria del trabajo, en los cuales se documentan todo el proceso del desarrollo del trabajo.

CAPÍTULO 2
Estudio estratégico

La creciente preocupación por la contaminación ambiental ha impulsado la búsqueda de soluciones innovadoras que aborden los desafíos relacionados con la calidad del aire. En este contexto, analizar las diferentes soluciones propuestas hasta la fecha se convierte en una tarea crucial para comprender mejor la problemática. Este estudio estratégico no solo nos permite evaluar la eficacia de las soluciones, sino que también nos proporciona un punto de vista sólido para identificar posibles mejoras y oportunidades.

En este apartado, exploraremos cuatro aplicaciones diferentes existentes en el mercado actual en las plataformas iOS y Android. A través de este análisis, buscamos identificar los puntos fuertes y las limitaciones de cada solución.

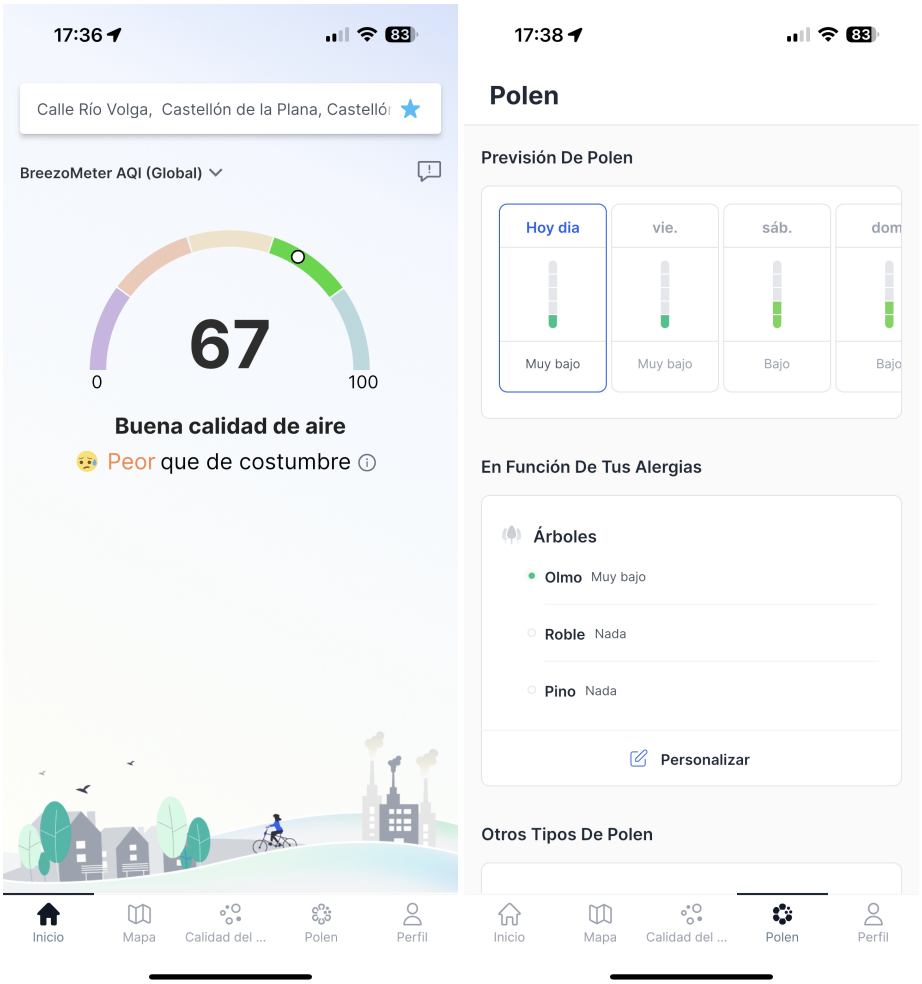
2.1 Análisis de las aplicaciones

2.1.1. BreezoMeter

La aplicación BreezoMeter cuenta con un diseño agradable e intuitivo y permite a los usuarios explorar de manera visual los niveles de calidad del aire en diferentes ubicaciones (ver Figura 2.1a).

BreezoMeter cuenta con un mapa interactivo en el que los usuarios pueden desplazarse y ver la calidad del aire en distintas zonas. La calidad del aire está representada con diferentes colores.

Además, la aplicación cuenta con una pestaña en la que los usuarios pueden acceder a información sobre diversos tipos de polen presentes en su área (ver Figura 2.1b). Esta característica resulta especialmente útil para aquellas personas que padecen alergias.

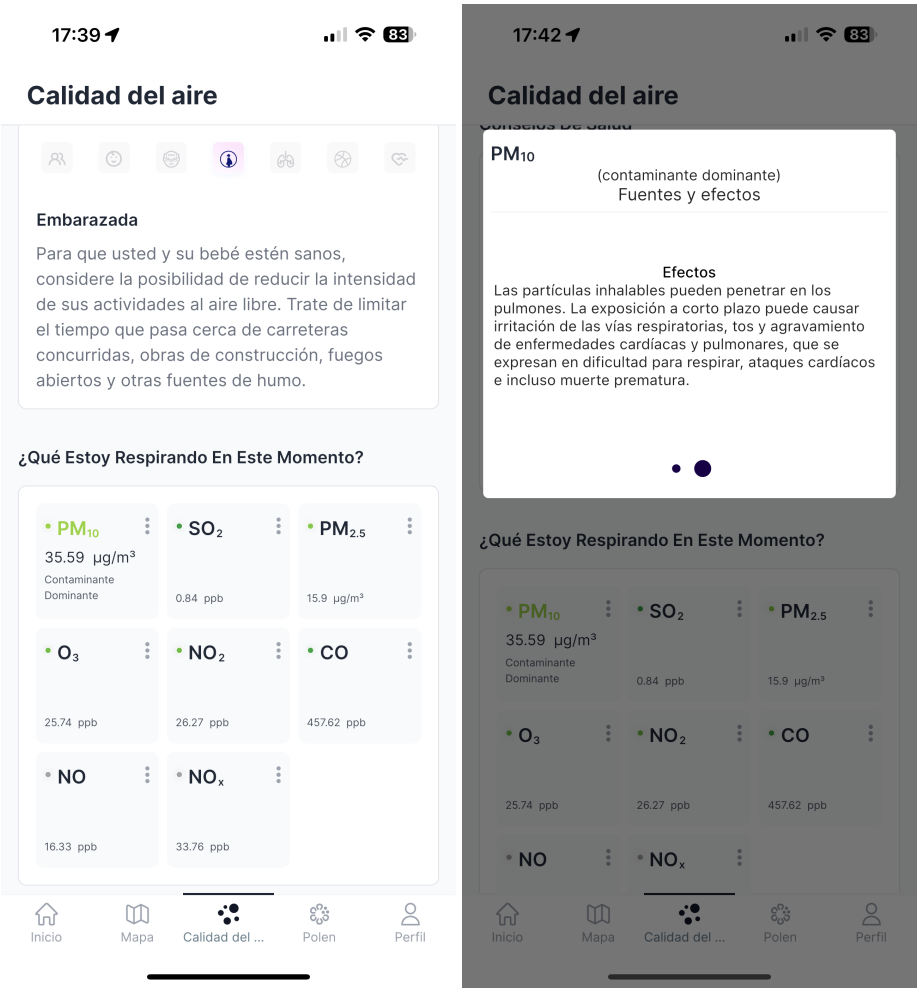


(a) Pantalla de inicio de BreezoMeter **(b)** Pantalla de información sobre el polen

Figura 2.1: Pantallas de BreezoMeter

Una característica destacada es la presentación de datos sobre diversos contaminantes atmosféricos, junto con información relevante sobre sus fuentes principales y los posibles efectos (ver Figura 2.2b). Además, la aplicación incluye una sección que proporciona consejos de salud dirigidos a una amplia audiencia, desde niños hasta personas embarazadas (ver Figura 2.2a).

Por último, pero igualmente relevante, BreezoMeter permite a los usuarios consultar la calidad del aire utilizando diversos índices, como el ICA, utilizado en España, o el IQA, común en Francia, o el suyo propio, BreezoMeter AQI. Sin embargo, en algunos casos, no se muestra un valor numérico correspondiente al índice de calidad del aire, lo que podría suscitar dudas.



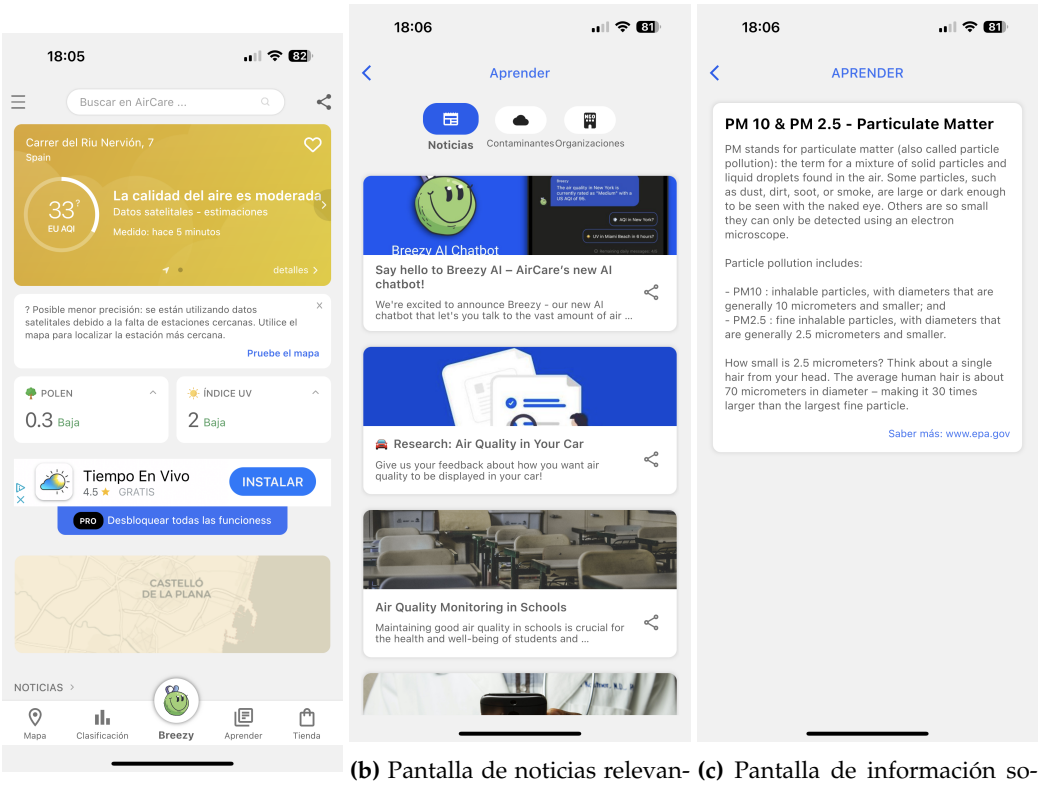
(a) Pantalla de consejos de salud y datos de contaminantes **(b)** Ventana de información relevante sobre los contaminantes

Figura 2.2: Pantallas de BreezoMeter

2.1.2. AirCare

AirCare presenta una diseño intuitivo (ver Figura 2.3a). Sin embargo, la apliación incluye publicidad en la versión gratuita, que se puede eliminar mediante una suscripción.

En la versión gratuita, AirCare permite a los usuarios consultar la calidad del aire y los datos de los principales contaminantes en diversas ubicaciones. La aplicación permite consultar una clasificación mundial o por continentes de los países con el peor índice de calidad del aire. Además, incluye una sección de noticias e información acerca de los contaminantes (ver Figuras 2.3b y 2.3c).



(a) Pantalla de inicio de AirCare **(b)** Pantalla de noticias relevantes **(c)** Pantalla de información sobre contaminantes

Figura 2.3: Pantallas de Aircare

Si el usuario decide adquirir la suscripción ofrecida, además de eliminar los anuncios, obtendrán acceso a un chatbot inteligente propio (ver Figura 2.4), aunque está en fase beta. Además, tendrá acceso a más datos, como pronósticos e historial más extensos. También tendrá la posibilidad de crear un *widget* en la pantalla de inicio del teléfono móvil.

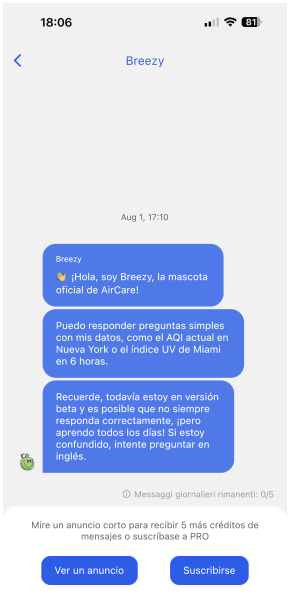
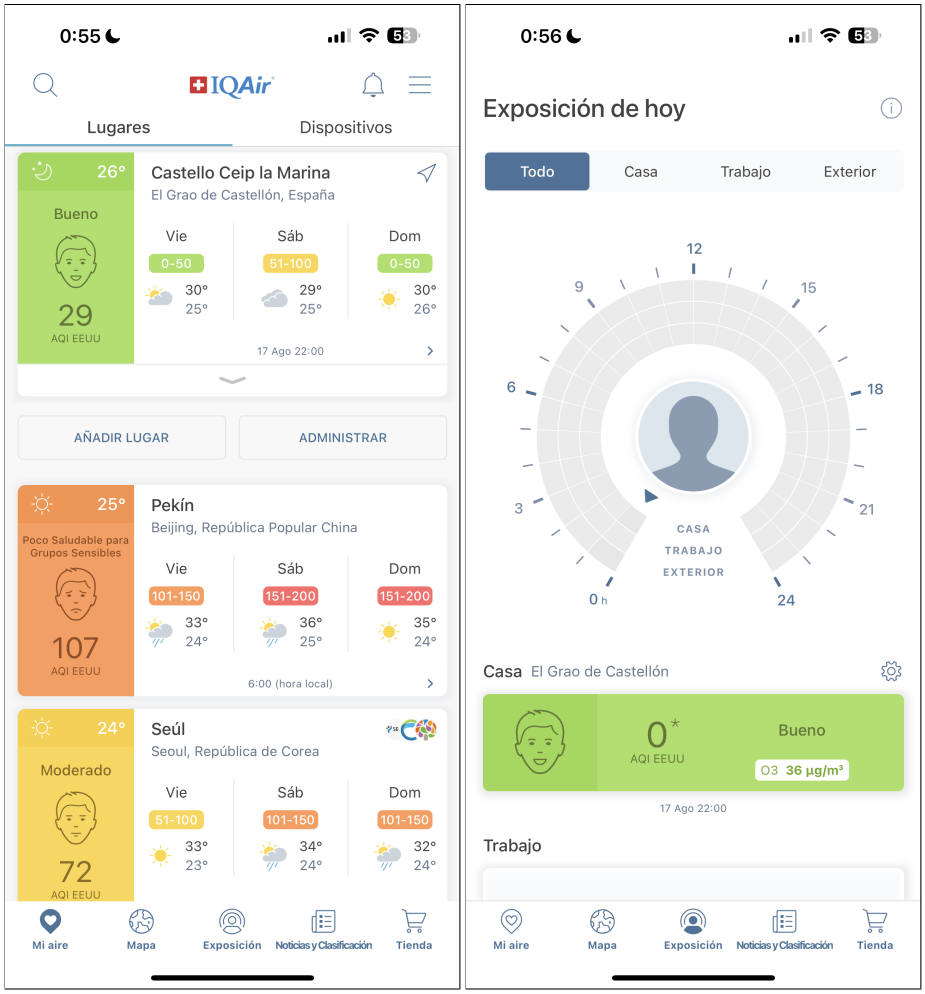


Figura 2.4: Pantalla del chatbot inteligente

2.1.3. AirVisual

AirVisual es una de las aplicacióne más descargadas de la categoría, también es de las más completas en este análisis. Cuenta con un diseño más elaborado que incorpora más elementos visuales a primera vista (ver Figura 2.5a).

Como aspecto distintivo, AirVisual presenta una característica única: realiza un seguimiento del usuario a lo largo del día (ver Figura 2.5b). Esto permite a los usuarios revisar su exposición diaria, tanto en el exterior como en interiores.

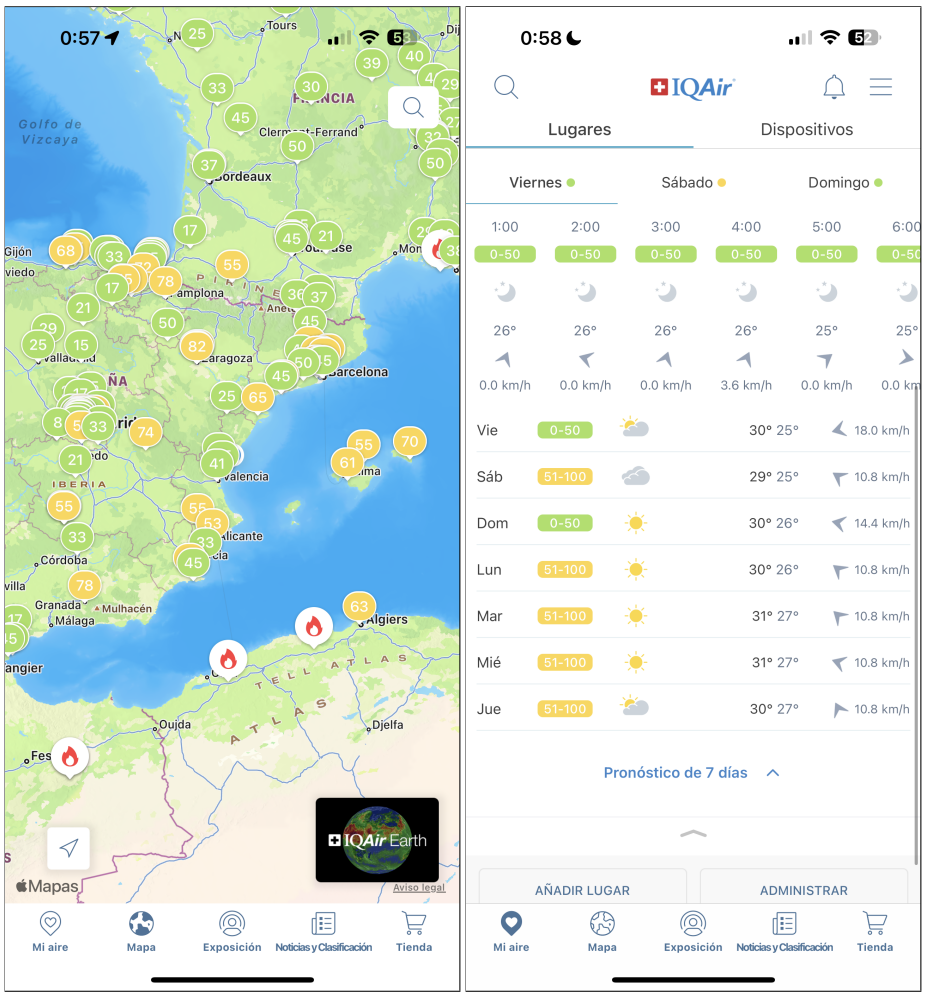


(a) Pantalla de inicio de AirVisual **(b)** Pantalla de exposición

Figura 2.5: Pantallas de AirVisual

AirVisual también cuenta con un mapa interactivo en el que se muestra de manera visual y numérica el índice de calidad del aire en diferentes zonas. Asimismo, también muestra lugares con alto riesgo de incendio (ver Figura 2.6a).

La aplicación ofrece un pronóstico de hasta una semana, que proporciona información tanto sobre la calidad del aire como sobre las condiciones climáticas (ver Figura 2.6b). Además, los usuarios pueden acceder a una sección que muestra una clasificación de las ciudades con la peor calidad del aire, así como un apartado de noticias.



(a) Mapa interactivo **(b)** Pronóstico semanal

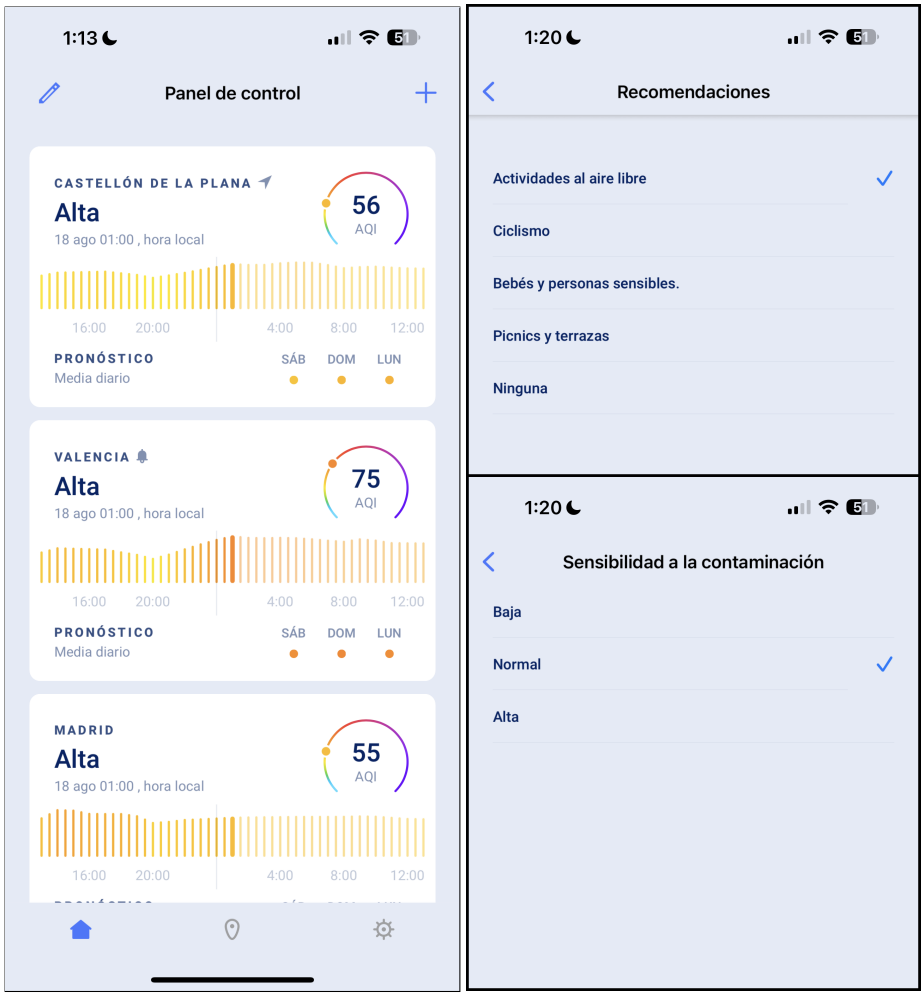
Figura 2.6: Pantallas de AirVisual

La solución, como se ha podido comprobar, es de las más completas. La aplicación, aunque no cuenta con publicidad, sí tiene un apartado en el que se muestra diferentes productos propios a la venta, como son sensores de monitoreo o purificadores de aire.

2.1.4. Plume Labs

Plume Labs destaca por su diseño minimalista y elegante, ofreciendo al usuario una experiencia de uso agradable (ver Figura 2.7a).

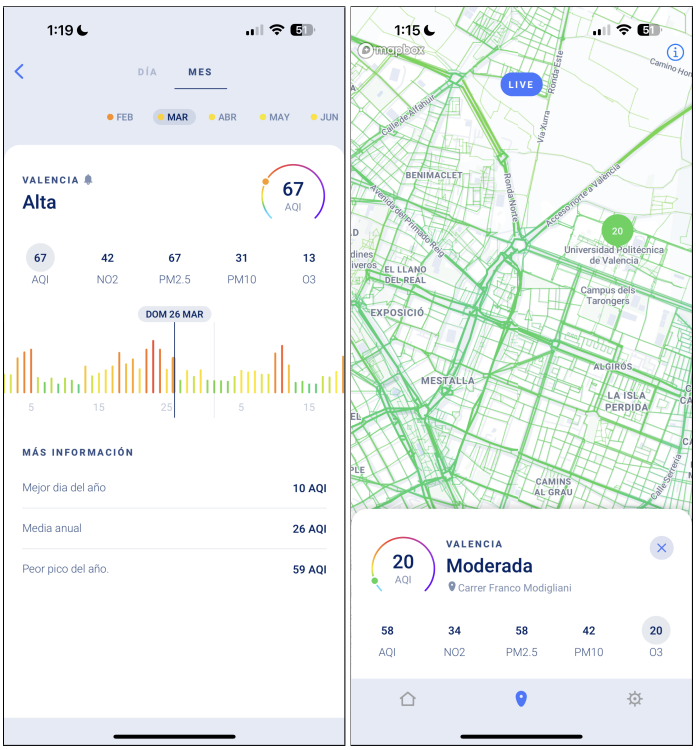
Una característica destacada es la posibilidad del usuario para ajustar su sensibilidad a la contaminación y seleccionar recomendaciones específicas que desearía recibir (ver Figura 2.7b). Esto permite que la aplicación ofrezca una experiencia más personalizada y adaptada a las preferencias de cada uno.



(a) Pantalla de inicio de Plume Labs **(b)** Pantalla de preferencias

Figura 2.7: Pantallas de Plume Labs

Para ofreceer información más completa, Plume Labs ofrece a los usuarios acceso a datos históricos y un pronóstico de hasta tres días (ver Figura 2.8a). Además, la aplicación incluye un mapa interactivo que permite a los usuarios filtrar por contaminantes para obtener consultas más detalladas (ver Figura 2.8b).



(a) Pantalla de pronóstico **(b)** Mapa interactivo con filtros

Figura 2.8: Pantallas de Plume Labs

2.2 Comparativa de las aplicaciones

Como parte de nuestro proceso de análisis, se ha creado una tabla comparativa que destaca las características de cada una. A continuación, se muestra la tabla que resume las características más importantes que presenta cada aplicación mencionada y las características que incluirá la aplicación:

Tabla 2.1: Tabla comparativa de características

Caracterítica	BreezoMeter	AirCare	AirVisual	Plume Labs	Incluido como requisito
Notificaciones sobre la calidad del aire	Sí	Sí*	Sí	Sí	Sí*
Consejos sobre la salud	Sí	No	Sí	Sí	Sí
Registro de usuario	Sí	Sí*	Sí	Sí	Sí
Versión de pago	No	Sí	No	No	No
Múltiples índices a seleccionar	Sí	Sí	Sí	Sí	No
Clasificación de ciudades	No	Sí	Sí	No	Sí
Mapa interactivo	Sí	Sí	Sí	Sí*	Sí
Sección de noticias e información relevante	Sí	Sí	Sí	No	Sí

* *Característica incluida con ciertas modificaciones*

2.3 Propuesta

Tras estudiar las diferentes aplicaciones disponibles en el mercado actual y comparar las características de las distintas soluciones propuestas (ver Tabla 2.1), sacamos las siguientes conclusiones, de cara a la primera versión del producto:

- El producto será multiplataforma, es decir, funcionará adecuadamente en las plataformas de iOS y Android.
- La aplicación presentará un diseño simple e intuitivo, permitiendo al usuario familiarizarse con la aplicación en el menor tiempo posible. Se le mostrará al usuario datos númericos de diferentes contaminantes, junto con el índice de calidad del aire. Además, se asignará un color diferente a cada rango del índice, facilitando la identificación visual de la calidad del aire.
- La aplicación proporcionará consejos de diversos tipos según la calidad del aire, lo que apoyará la toma de decisiones de los usuarios. Por ejemplo, podría sugerir el uso de una mascarilla en situaciones específicas.
- El usuario tendrá la posibilidad de acceder a la aplicación como invitado, aunque con ciertas limitaciones. También puede optar por registrarse de manera gratuita y tener acceso a todas las funciones ofrecidas.
- Los usuarios tendrán la capacidad de configurar las notificaciones que deseen recibir de la aplicación. Estas incluirán una notificación diaria, así como alertas para cambios significantes en la calidad del aire en la ubicación del usuario.
- Algunas de las características mencionadas en la Tabla 2.1 han sido consideradas de menor prioridad y, por lo tanto, se han excluido de esta primera versión del producto debido a limitaciones de tiempo.

CAPÍTULO 3
Metodología de trabajo

En el ámbito del desarrollo de software, existen diversas formas o metodologías que sirven como herramienta de gestión del proceso de desarrollo, proporcionando estructura y eficiencia, fomentando, al mismo tiempo, la colaboración entre miembros del equipo. Generalmente, estas metodologías descomponen el proceso en diferentes etapas, como planificación, ejecución, seguimiento y cierre. No obstante, estas pueden variar según el enfoque o la metodología aplicada. A continuación, se presentarán dos enfoques ampliamente utilizados: el enfoque tradicional o en cascada y el enfoque ágil.

3.1 Enfoque tradicional

El enfoque tradicional, también conocido como modelo en cascada, ha sido ampliamente utilizado durante décadas en el desarrollo de software [1]. Este consiste en un procedimiento lineal y secuencial, es decir, las diferentes fases de desarrollo se ejecutan tan solo una vez. Además, no se avanza a la siguiente etapa hasta finalizar la anterior (ver Figura 3.1). Este enfoque es útil en proyectos donde los requisitos son estables y bien definidos desde el principio. Sin embargo, presenta limitaciones cuando surgen cualquier tipo de cambios, ya sea un nuevo requisito o un error, en cualquiera de las etapas, ya que cualquier modificación requiere volver atrás y repetir las fases anteriores, cosa que resulta una pérdida significativa de tiempo y de recursos, tanto humanos como materiales.

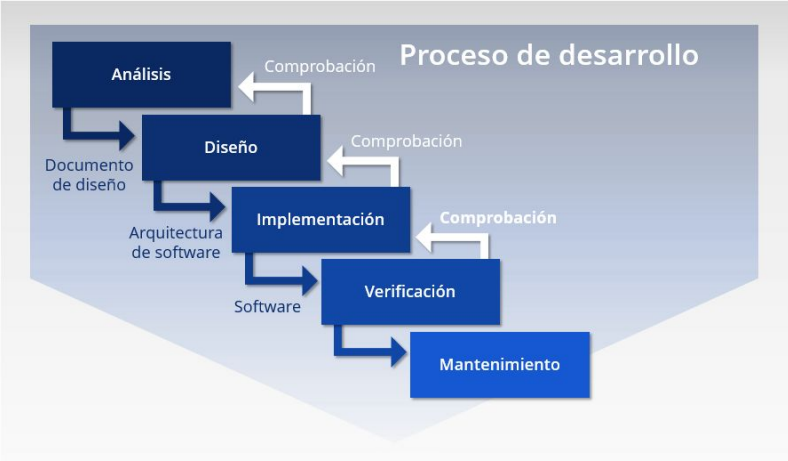


Figura 3.1: Diagrama del enfoque tradicional o en cascada

3.2 Enfoque ágil

Por otro lado, el enfoque ágil ha ganado popularidad en los últimos años debido a su capacidad de adaptación y flexibilidad. Por lo general, las metodologías ágiles se caracterizan por un procedimiento iterativo, es decir, el proceso de desarollo se realiza por iteraciones con entregas incrementales de funcionalidades (ver Figura 3.2). Estas iteraciones llamadas "sprints", generalmente tienen una duración de 1 a 4 semanas. Al final de cada sprint, se entrega un incremento funcional del producto. En algunas metodologías también se promueve la comunicación entre los miembros del equipo, así como la adaptación continua en función de los cambios y las necesidades del proyecto que vayan surgiendo. Algunas de las metodologías ágiles más utilizadas son Extreme Programming [2], Scrum [3] y Kanban [4]

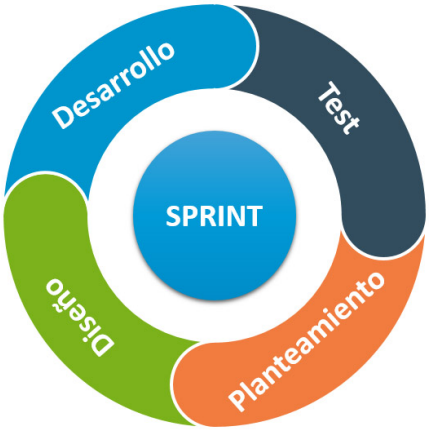


Figura 3.2: Diagrama del enfoque ágil

3.3 Scrum

Para este trabajo, se ha seguido el enfoque ágil Scrum. La elección de Scrum se basa en su capacidad para responder a los cambios, permitiendo una mayor flexibilidad y adaptación durante el desarrollo. Scrum se ha aplicado mediante la planificación de sprints, en este caso de una duración de dos semanas, la realización de revisiones diarias de seguimiento (*Daily Stand-ups*) y la revisión periódica del progreso, realizada al final de cada sprint (*Sprint Reviews*) (ver Figura 3.3).

Una de las características de Scrum es la definición de un *Backlog*, que consiste en una lista ordenada (según la prioridad) de los *items* a abordar. Estos *items*, que también pueden ser llamadados Historias de Usuario (HU) o Unidades de Trabajo (UT) son generalmente funciones a implementar en el proyecto, aunque también pueden ser tareas de otro tipo, como tareas asociadas a la arquitectura del producto o tareas técnicas como el *refactoring* [5].

Los items tienen un flujo de trabajo o desarrollo (*Workflow*): Definición de los requisitos, Implementación o Desarrollo y *Testing*. La Definición de requisitos abarca un peso relevante ya que en él se debe concretar la Definición de *DONE* (que indica cuando un elemento está completado), por lo general mediante Pruebas de Aceptación. Esto se traduce en un enfoque de desarrollo basado en pruebas de aceptación, conocido como ATDD (*Acceptance Test-Driven Development* en inglés) [23].

Las Pruebas de Aceptación consisten en pruebas realizadas para demostrar que el sistema cumple con el comportamiento esperado desde la perspectiva del usuario. Estas pruebas de aceptación representan una solución intermedia, ya que, por un lado, pueden describir en detalle el comportamiento del sistema y, por otro lado, son más fáciles de administrar que extensos textos narrativos o plantillas [6] .

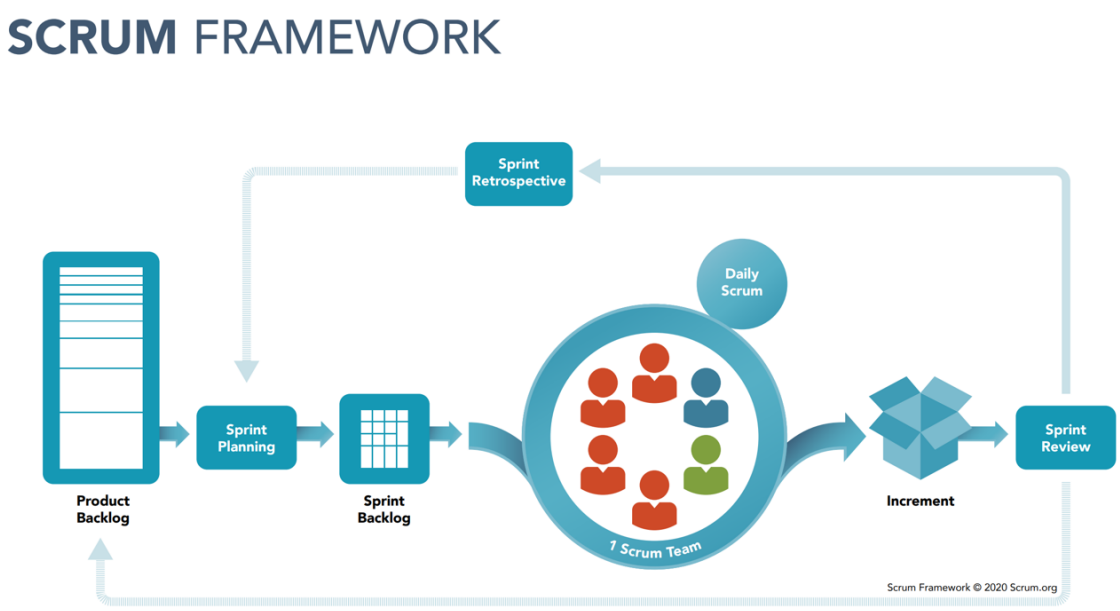


Figura 3.3: Marco de trabajo Scrum

Con el objetivo de facilitar el seguimiento del proyecto se ha decidido en utilizar Trello [7], una herramienta de gestión de proyectos en línea. Trello se basa en un sistema de tableros, columnas o listas y tarjetas que, en esta ocasión, las columnas serían las distintas fases y las tarjetas serían los items (ver Figuras 3.4 y 3.5).

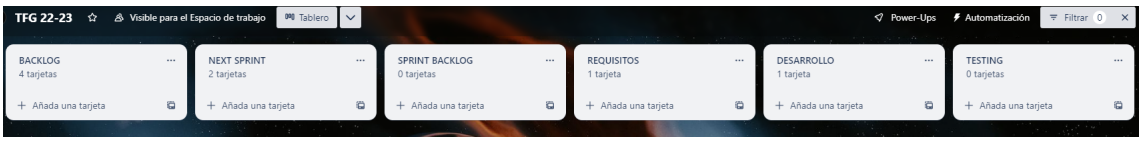


Figura 3.4: Tablero canvan

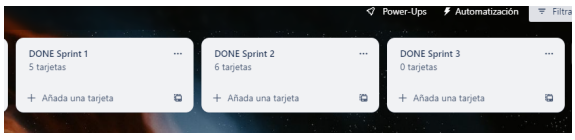


Figura 3.5: Tablero canvan

La adopción de Scrum como metodología de trabajo ha permitido una mayor transparencia en el progreso del proyecto y la capacidad de adaptarse a las necesidades cambiantes. Este enfoque ha proporcionado un marco sólido para el desarrollo, facilitando una entrega más eficiente.

Es importante destacar que no se ha seguido rigurosamente las reglas propuestas por la metodología Scrum. En su lugar, se han realizado ciertas adaptaciones para ajustarlas a las necesidades de este proyecto, concretamente en la especificación de los requisitos.

En resumen, el enfoque ágil, en este caso Scrum, se ha seleccionado como metodología de trabajo para este proyecto, permitiendo una mayor adaptabilidad y entrega incremental de funcionalidades. El uso de Scrum ha proporcionado una base consistente para el desarrollo de la aplicación.

CAPÍTULO 4
Análisis y especificación de requisitos

"La parte más difícil de construir un sistema de software es decidir con precisión qué construir. [...] Por lo tanto, la función más importante que los ingenieros de software realizan para sus clientes es la extracción y el refinamiento iterativos de los requisitos del producto."[8].

4.1 Especificación de Requisitos Software

Como se ha explicado en la Sección 3.3, en esta fase no se ha seguido por completo las pautas propuestas por Scrum. En su lugar, se ha optado por una combinación con un enfoque más tradicional, específicamente siguiendo algunas de las guías propuestas por el estándar IEEE/ANSI Std. 830-1998 [10].

Siguiendo el estándar IEEE/ANSI Std. 830-1998, se han establecido, de manera tradicional, el propósito, ámbito y restricciones del sistema, entre otros. Además, siguiendo las guías de Scrum, en cada iteración o sprint, antes de proceder con la implementación de cada ítem, se han desarrollado prototipos o *mock-ups* y se han definido las pruebas de aceptación de cada uno.

A continuación, se presentarán los contenidos mencionados junto con sus correspondientes diagramas o tablas, cuando aplique.

4.1.1. Propósito

El objetivo de esta Especificación de Requisitos de Software (ERS) es establecer los requerimientos que se incluirán en el MVP (Producto Mínimo Viable) de la aplicación, con el propósito de facilitar y orientar su desarrollo.

4.1.2. Ámbito y descripción del producto

En esta primera versión, la aplicación adoptará el nombre *Clean Breath*, compuesto por las palabras en inglés: *Clean* (limpio) y *Breath* (respiración).

Clean Breath tiene como propósito ofrecer a sus usuarios una plataforma, donde puedan consultar la calidad del aire en su área y los niveles de contaminantes atmosféricos. Esto les proporciona información esencial para tomar decisiones informadas sobre su salud y bienestar.

Contará con una interfaz limpia y minimalista con la información relevante necesaria. Un diseño elegante permitirá a los usuarios acceder eficientemente a datos sobre la calidad del aire. La combinación de una experiencia de usuario intuitiva y una representación visual efectiva consolidará aún más la utilidad y la practicidad de Clean Breath.

Los usuarios tendrán la opción de acceder a la aplicación como invitados, aunque de manera limitada. En este caso, se minimizará el almacenamiento de información sobre el usuario. También tendrán la opción de registrarse para tener acceso a todas las funcionalidades ofrecidas.

4.1.3. Modelo de Dominio

Con el fin de proporcionar una mejor compresión del contexto, se ha elaborado un diagrama de clases siguiendo las convenciones propuestas por UML (*Unified Modeling Language*) [9] (ver Figura 4.1).

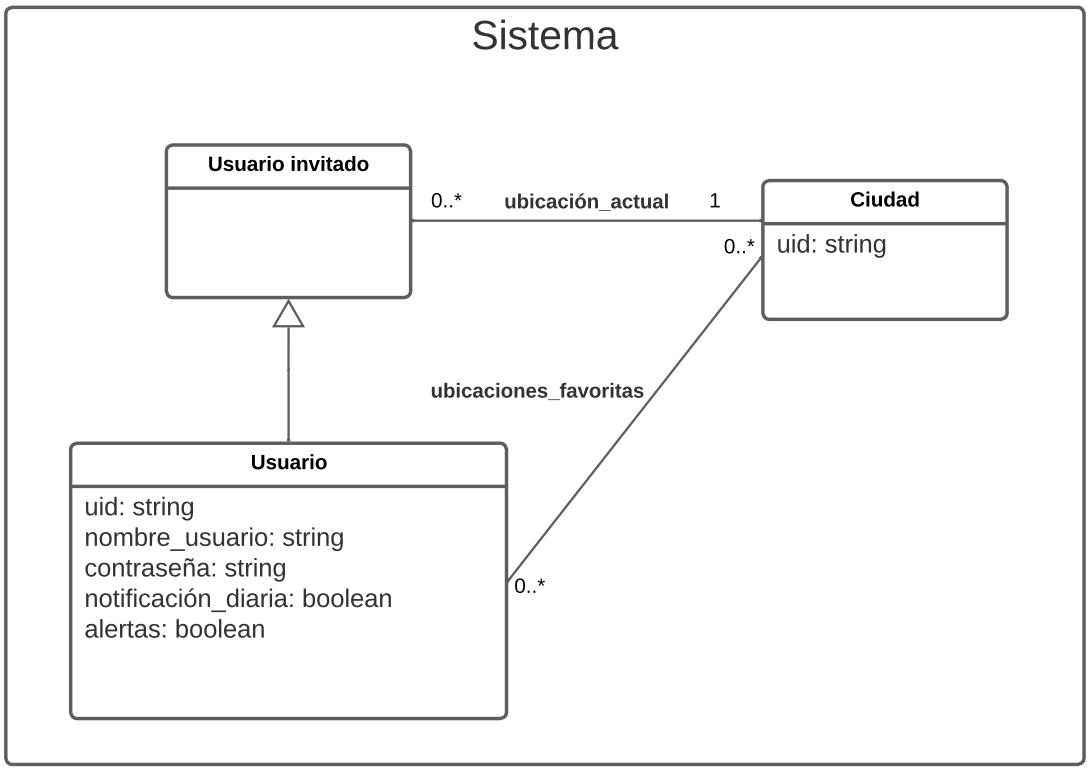


Figura 4.1: Diagrama del modelo de dominio

A continuación, se detalla el significado de cada clase con sus respectivos atributos.

- **Usuario invitado**: usuario que accede a la aplicación sin haberse registrado previamente.
- **Usuario**: usuario registrado en el sistema.

Tabla 4.1: Atributos de la clase Usuario

Atributos	Descripción
uid *string*	Cadena de caracteres única que identifica al usuario internamente.
nombre_usuario *string*	Identificador que una persona decide poner una vez se registra en la aplicación. Deberá ser única y será requerida para el inicio de sesión.
contraseña *string*	Código secreto que solamente conoce el usuario y será requerido para el inicio de sesión.
notificacion_diaria *boolean*	Variable que puede tomar los valores true o false, indicando si el usuario desea o no recibir una notificación diaria.
alertas *boolean*	Variable que puede tomar los valores true o false, indicando si el usuario desea o no recibir alertas en casos de condiciones atmosféricas extremas.

- **Ciudad**

Tabla 4.2: Atributos de la clase Ciudad

Atributos	Descripción
uid *string*	Identificador único de la ciudad usado para la obtención de los datos

4.1.4. Límites del Sistema

Los límites del sistema son representados mediante una diagrama de contexto, en el se muestran los diferentes actores que interactúan con el sistema (ver Figura 4.2). En este caso los actores identificados son los siguientes:

- **Usuario invitado**: usuario que accede a la aplicación (con ciertos limites) sin haberse registrado previamente.
- **Usuario**: usuario registrado en el sistema con acceso a todas las funcionalidades.
- **API externo**: corresponde con la API de AQI (*Air Quality Index*) integrada con la aplicación, que es la fuente de los datos mostrados en la aplicación.

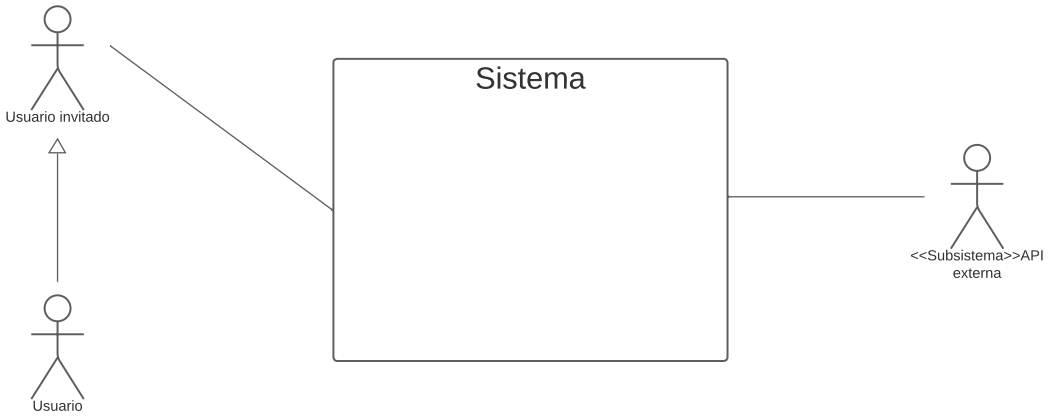


Figura 4.2: Diagrama de contexto

4.1.5. Restricciones del Sistema

En este caso concreto, la única restricción que presenta la aplicación es que requiere de una conexión a internet para su correcto funcionamiento. En caso contrario, no sería posible la obtención de datos sobre la calidad del aire.

4.1.6. Lista de Características del Sistema

A continuación se muestra el listado de las características que componen el sistema y sus respectivas descripciones. En esta ocasión, serán también los ítems a abordar durante el desarrollo. Para una mejor organización y mayor comprensión, las características se han agrupado según el área al que afectan: Principales del sistema, Gestión de ciudades, Gestión de usuarios y Gestión de ciudades

- Características principales
  - Acceder a la pantalla de inicio (ver Tabla 4.3)
  - Acceder a ajustes (ver Tabla 4.4)
- Gestión de ciudades
  - Buscar una ciudad (ver Tabla 4.5)
  - Añadir ciudad a la lista de favoritos (ver Tabla 4.6)
  - Eliminar ciudad de la lista de favoritos (ver Tabla 4.7)
  - Consultar los detalles de la calidad del aire de una ciudad (ver Tabla 4.8)
- Gestión de usuarios
  - Registrarse como usuario (ver Tabla 4.9)
  - Acceder como invitado (ver Tabla 4.10)
  - Iniciar sesión (ver Tabla 4.11)

- Gestión de notificaciones
  - Activar/Desactivar notificación diaria (ver Tabla 4.12)
  - Activar/Desactivar alertas (ver Tabla 4.13)

Tabla 4.3: Descripción del requisito Acceder a la pantalla de inicio

Actor: USUARIO INVITADO	**Ítem: ACCEDER A LA PANTALLA DE INICIO**
Descripción	Acceso la pantalla principal mediante el cual, puede acceder al resto de las funciones.
Precondiciones	Ninguna
Entradas	Ninguna
Salidas	Pantalla de inicio con acceso al resto de las funciones de la aplicación.
Flujo principal	1. Iniciar la aplicación

Tabla 4.4: Descripción del requisito Acceder a ajustes

Actor: USUARIO INVITADO	**Ítem: ACCEDER A AJUSTES**
Descripción	Acceso la pantalla de ajustes de la aplicación.
Precondiciones	Ninguna
Entradas	Ninguna
Salidas	Listado de configuraciones de la aplicación.
Flujo principal	1. Pulsar la sección de ajustes

Tabla 4.5: Descripción del requisito Buscar una ciudad

Actor: USUARIO INVITADO	**Ítem: BUSCAR UNA CIUDAD**
Descripción	El usuario puede buscar cualquier ciudad por su nombre.
Precondiciones	Ninguna
Entradas	El nombre de la ciudad.
Salidas	Lista de localidades o ciudades que coincidan con la búsqueda realizada.
Flujo principal	1. Pulsar la sección de buscar 2. Introducir el nombre de una ciudad 3. Pulsar el botón de buscar

Tabla 4.6: Descripción del requisito Añadir ciudad a la lista de favoritos

Actor: USUARIO	**Ítem: AÑADIR CIUDAD A LA LISTA DE FAVORITOS**
Descripción	El usuario puede guardar una ciudad como favorito.
Precondiciones	El usuario debe haberse registrado.
Entradas	Ciudad seleccionada.
Salidas	La ciudad se muestra como favorito.
Flujo principal	1. Seleccionar una ciudad 2. Pulsar el botón de favorito

Tabla 4.7: Descripción del requisito Eliminar ciudad de la lista de favoritos

Actor: USUARIO	**Ítem: ELIMINAR CIUDAD DE LA LISTA DE FAVORITOS**
Descripción	El usuario puede eliminar una ciudad de la lista de favoritos.
Precondiciones	El usuario debe haberse registrado. La ciudad debe estar en la lista de favoritos.
Entradas	Ciudad seleccionada.
Salidas	La ciudad deja de mostrarse como favorito.
Flujo principal	1. Seleccionar una ciudad 2. Pulsar el botón de favorito

Tabla 4.8: Descripción del requisito Consultar los detalles de la calidad del aire de una ciudad

Actor: USUARIO INVITADO	**Ítem: CONSULTAR LOS DETALLES DE LA CALIDAD DEL AIRE DE UNA CIUDAD**
Descripción	El usuario puede acceder a una pantalla en la que se muestra información más detallada sobre una ciudad
Precondiciones	Ninguna
Entradas	Ciudad seleccionada.
Salidas	Una pantalla que muestra más información, como los datos de los diferentes contaminantes.
Flujo principal	1. Seleccionar una ciudad

Tabla 4.9: Descripción del requisito Registrarse como usuario

Actor: USUARIO	**Ítem: REGISTRARSE COMO USUARIO**
Descripción	El usuario puede registrarse en al aplicación con nombre de usuario y contraseña.
Precondiciones	Ninguna
Entradas	El nombre de usuario y la contraseña.
Salidas	El usuario accede a la aplicación.
Flujo principal	1. Seleccionar la opción de registrarse. 2. Rellenar los campos necesarios. 3. Pulsar el botón de registrarse.

Tabla 4.10: Descripción del requisito Acceder como invitado

Actor: USUARIO INVITADO	**Ítem: ACCEDER COMO INVITADO**
Descripción	El usuario puede acceder a la aplicación como invitado, es decir, sin registrarse.
Precondiciones	Ninguna
Entradas	Ninguna
Salidas	El usuario accede a la aplicación con ciertos límites.
Flujo principal	1. Seleccionar la opción de acceder como invitado.

Tabla 4.11: Descripción del requisito Iniciar sesión

Actor: USUARIO	**Ítem: INICIAR SESIÓN**
Descripción	El usuario puede acceder a la aplicación con nombre de usuario y contraseña
Precondiciones	El usuario debe haberse registrado en la aplicación.
Entradas	El nombre de usuario y la contraseña.
Salidas	El usuario accede a la aplicación.
Flujo principal	1. Rellenar los campos necesarios 2. Pulsar el botón de iniciar sesión

Tabla 4.12: Descripción del requisito Activar/desactivar notificación diaria

Actor: USUARIO	**Ítem: ACTIVAR/DESACTIVAR NOTIFICACIÓN DIARIA**
Descripción	El usuario puede activar o desactivar las notificaciones diarias de la aplicación.
Precondiciones	El usuario debe haberse registrado en la aplicación.
Entradas	La hora en la que quiere recibir la notficación.
Salidas	Ninguna
Flujo principal	1. Acceder a la pantalla de ajustes. 2. Activar o desactivar las notificaciones diarias. 3. En caso de activar, seleccionar la hora de la notificación.

Tabla 4.13: Descripción del requisito Activar/desactivar alertas

Actor: USUARIO	**Ítem: ACTIVAR/DESACTIVAR ALERTAS**
Descripción	El usuario puede activar o desactivar las alertas de la aplicación.
Precondiciones	El usuario debe haberse registrado en la aplicación.
Entradas	Ninguna
Salidas	Ninguna
Flujo principal	1. Acceder a la pantalla de ajustes. 2. Activar o desactivar las notificaciones diarias.

4.1.7. Diagramas de Casos de Uso

En este apartado se explica mediante la técnica Casos de Uso cuáles son las acciones que pueden realizar los usuarios de la aplicación, así como la funcionalidad que tiene la misma para que los usuarios de la app realicen dichas acciones.

4.1.7.1. Casos de uso principales

Tras iniciar la aplicación, se accederá de manera predeterminada a la pantalla de inicio. Desde esta pantalla, el usuario podrá ver información de su ubicación actual y de las ciudades que haya guardado como favoritos. También tendrá acceso a la pantalla de ajustes donde se le mostrará distintas configuraciones del sistema (ver Figura 4.3).

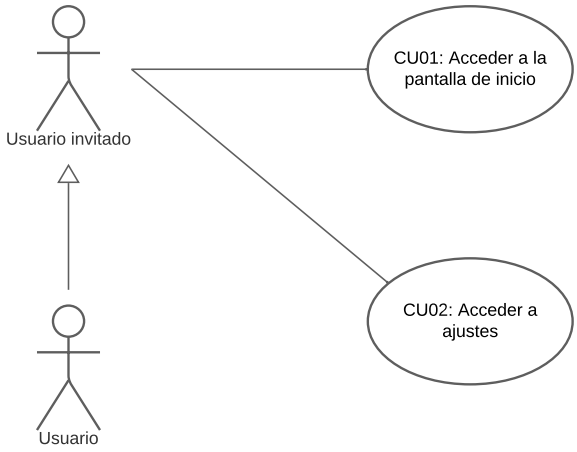


Figura 4.3: Diagrama de casos de uso principales

4.1.7.2. Gestión de ciudades

En la pantalla de inicio de la aplicación, seleccionando una ciudad, el usuario puede acceder a una nueva pantalla donde se muestra información detallada. Asimismo, desde la pantalla de detalles, si el usuario está registrado, tendrá la opción de guardar o eliminar una ciudad de la lista de favoritos (ver Figura 4.4). También tendrá acceso, desde la pantalla de inicio, a la sección de búsqueda de ciudades.

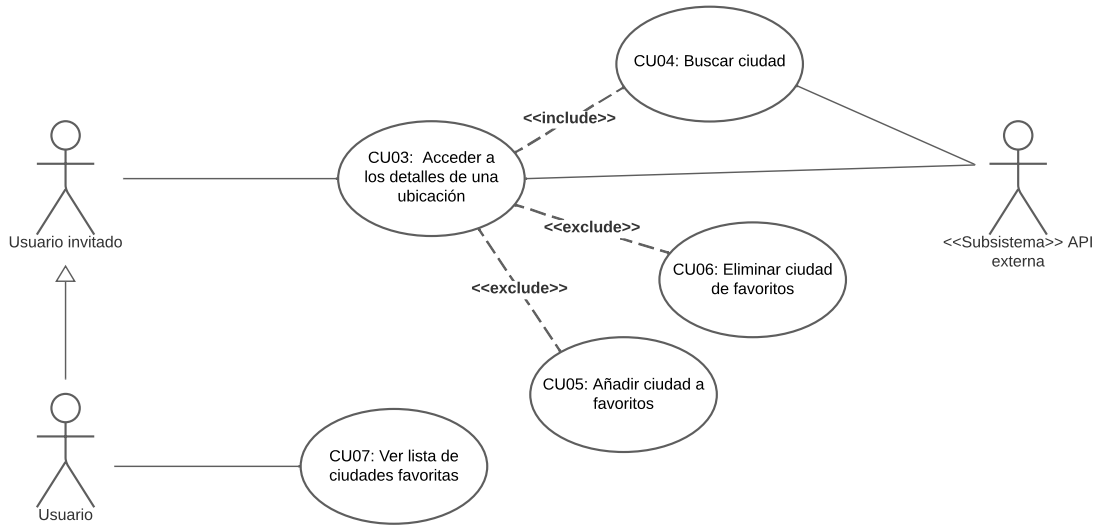


Figura 4.4: Diagrama de casos de uso para la gestión de ciudades

4.1.7.3. Gestión de usuarios

Para usar la aplicación, el usuario puede acceder como invitado o registrarse con un nombre de usuario y contraseña (ver Figura 4.5). En el primer caso, tendrá un acceso limitado de la aplicación, por ejemplo no podrá guardar ciudades como favoritos.

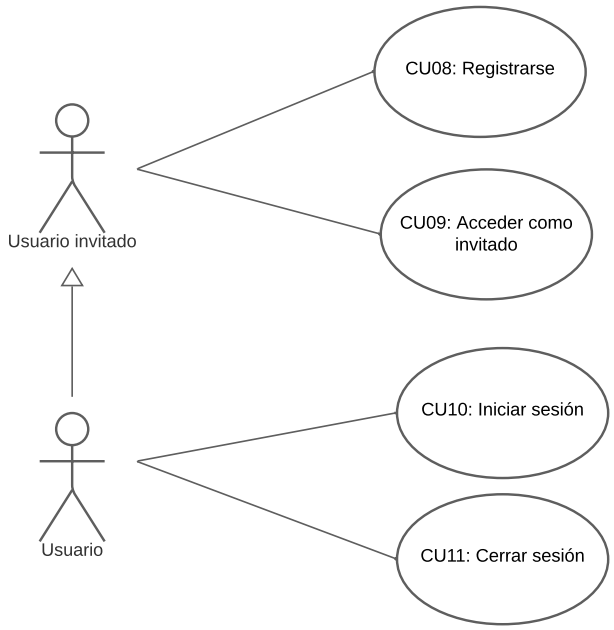


Figura 4.5: Diagrama de casos de uso para la gestión de usuarios

4.1.7.4. Gestión de notificaciones

En la pantalla de ajustes, el usuario tendrá acceso a diferentes configuraciones del sistema. Estos serían la activación y desactivación de las notificaciones y las alertas. Estas funciones también están limitadas a los usuarios registrados (ver Figura 4.6).

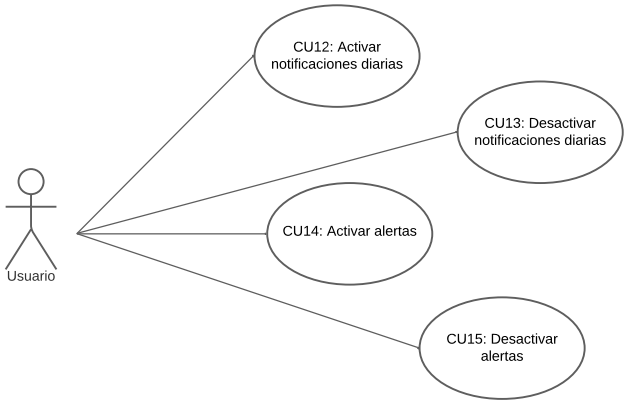


Figura 4.6: Diagrama de casos de uso para la gestión de las notificaciones

4.1.8. Pruebas de Aceptación

En esta sección, se detallan las pruebas de aceptación que se han diseñado para validar la funcionalidad de la aplicación. Estas pruebas han sido cuidadosamente estructuradas para evaluar la aplicación y para asegurarse de que cumple con las expectativas de los usuarios finales. Las pruebas de aceptación representan un paso crucial, ya que garantizan que la aplicación sea eficaz y satisfactoria.

Siguiendo el enfoque ágil, las pruebas de aceptación se han ido diseñando y definiendo a lo largo del transcurso de los sprints, antes de implementar cada ítem. Las pruebas sirven también como guía para el desarrollador, ya que puede describir diversos escenarios y el comportamiento de la aplicación en dichos casos.

Para consultar todas las pruebas véase Apéndice A. A continuación, se describe una de las pruebas de aceptación definidas, incluidos sus criterios de éxito y procedimientos de ejecución:

En la Tabla A.23 se describe el escenario en el que un usuario no consigue iniciar sesión mediante nombre de usuario y contraseña. En la prueba de aceptación, se establecen las condiciones o requisitos previos para que esta situación sea posible, que en este caso serían: ingreso de datos incorrectos o que el usuario aún no se ha registrado en el sistema. A continuación, se enumeran los pasos necesarios para recrear el escenario, donde el usuario, al abrir la aplicación, rellena los campos obligatorios: nombre de usuario y contraseña. Por último, se especifica el resultado esperado, que consiste en que la aplicación debería mostrar un mensaje de error indicando que los datos introducidos son incorrectos.

Una vez que la prueba de aceptación está definida, el ítem avanza a la fase de desarrollo. Al finalizar la implementación, o incluso durante ella, el desarrollador deberá validar las pruebas siguiendo los pasos definidos en ella. Cuando el desarrollador supere exitosamente todas las pruebas, el ítem podrá avanzar a la siguiente fase, que es el *testing*. En esta etapa, uno o varios testers vuelve a ejecutar las pruebas para garantizar el funcionamiento correcto del ítem.

A parte de la información descrita, también se muestra información sobre el ítem al que corresponde la prueba *Inicio de sesión* y la característica a la que pertenece *Gestión de usuarios*.

CAPÍTULO 5
Arquitectura y diseño de la aplicación

La arquitectura y el diseño son elementos esenciales para el desarrollo exitoso de cualquier sistema software, ya que determinan cómo se organizan los componentes y cómo se cumplen los objetivos establecidos.

5.1 Arquitectura multicapa

La arquitectura multicapa, es un enfoque común en el diseño de sistemas software. En ella, se divide las diferentes responsabilidades y componentes de la aplicación en capas o niveles separados [11].

La principal ventaja de este enfoque, también la principal razón por su elección, es que el desarrollo puede realizarse en varios niveles, lo que permite que cualquier cambio que ocurra afecte solo al nivel necesario. Esto se debe a que el acoplamiento informático queda reducido hasta una interfaz de paso de mensajes.

Concretamente, la división de capas ha sido la siguiente: capa de Presentación, capa de Negocio y capa de Datos (ver Figura 5.1).

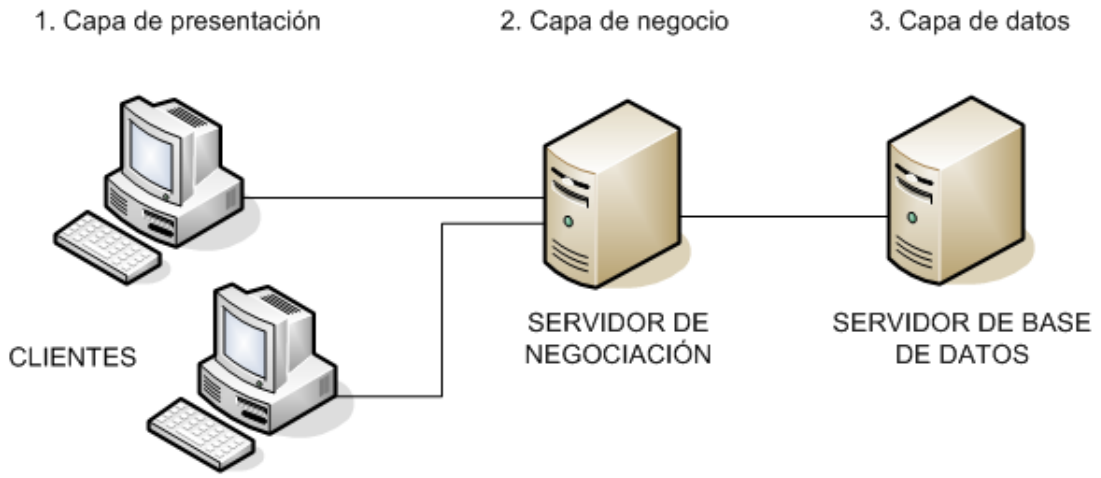


Figura 5.1: Arquitectura en tres capas

1. **Capa de Presentación.** La capa de Presentación o cliente es la que se muestra al usuario y es con la que interactúa. Aquí, se desarrollan las interfaces de usuario, en

las cuales se representará visualmente la información relativa a la calidad del aire y los diferentes contaminantes. Los componentes visuales y la lógica de presentación se gestionan aquí.

2. **Capa de Negocio.** La capa de Negocio o servidor se encarga de la lógica que subyace al sistema. Aquí es donde se reciben y se procesan las peticiones del usuario enviadas desde la capa de presentación. Se denomina capa de negocio (o incluso de lógica de negocio) porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica también con la capa de datos para la recuperación y el almacenamiento de datos.

3. **Capa de Datos.** Aquí es donde se almacenan y gestionan los datos de la aplicación, como los datos del usuario. Esta capa recibe solicitudes de almacenamiento y de recuperación desde la capa de negocio.

En conjunto, estas tres capas trabajan juntas para crear una aplicación móvil funcional. La capa de Presentación proporciona una experiencia visual al usuario, la capa de Negocios se encarga de la lógica y el flujo de la aplicación, y por último, la capa de Datos garantiza que los datos se guarden y recuperen de manera fiable.

Esta arquitectura contribuye a la separación de responsabilidades, facilitando el mantenimiento y la escalabilidad del sistema. Cada capa tiene su propia función y se comunica con las demás de manera organizada, ofreciendo, así, una experiencia coherente.

5.2 Diseño de las interfaces de usuario

A continuación, se presentarán los bocetos o *Mockups* de las interfaces de usuario. El diseño de las interfaces se han realizado utilizando la herramienta Figma, un editor de gráficos vectorial [12]. Siguiendo las pautas del enfoque ágil, se han ido diseñando estos bocetos a medida que se implementaba cada ítem.

Al iniciar la aplicación, el usuario tendrá varias opciones para acceder a la aplicación: registrarse, iniciar sesión o acceder como invitado (ver Figura 5.2).

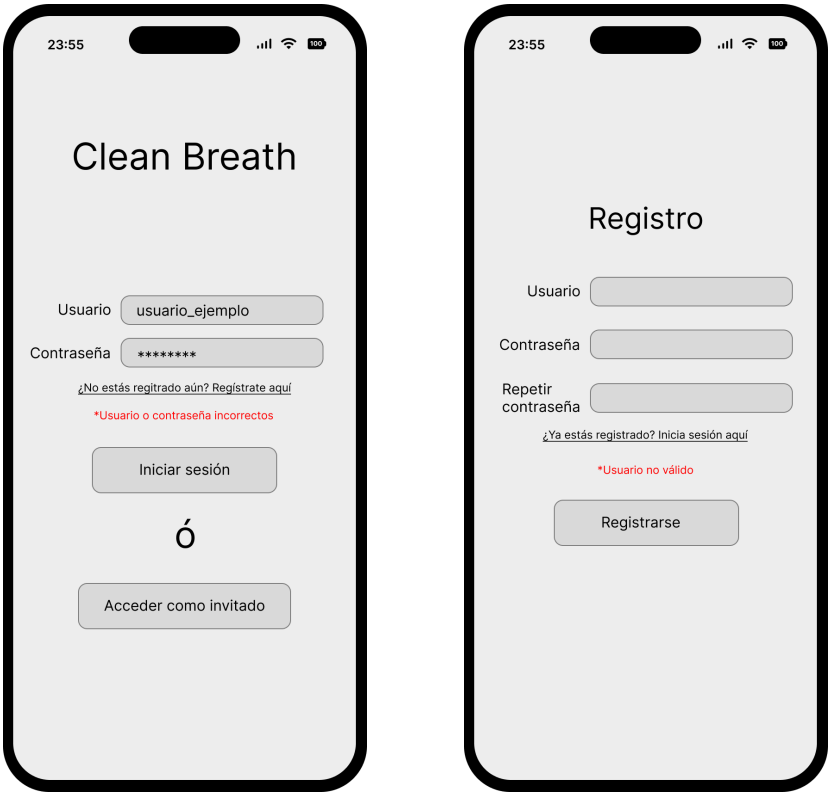
Desde la pantalla de inicio (ver Figura 5.3a), el usuario tendrá acceso a las siguientes funcionalidades:

- Vista de una tarjeta con información básica (ubicación e índice de calidad del aire) de la ubicación actual del usuario. Si el usuario se ha registrado, verá todas las tarjetas de las ubicaciones que haya guardado como favoritos. Según la calidad del aire, estas tarjetas irán cambiando de color (desde verde hasta granate).

- Acceso a información más detallada de una ubicación pulsando a la tarjeta que desee.

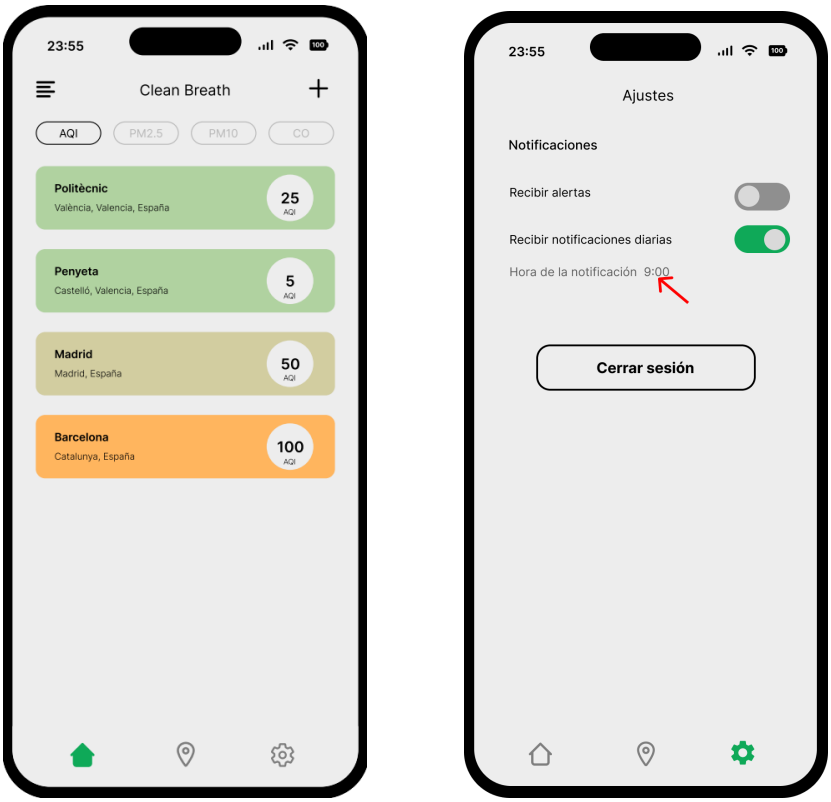
- Acceso a la pantalla de búsqueda pulsando al icono más (+) situado en la parte superior derecha de la pantalla.

- Acceso a la pantalla de ajustes desde la barra de navegación situada en la parte inferior de la pantalla. Desde ajustes, además de cerrar sesión, el usuario podrá gestionar las notificaciones de la aplicación (ver Figura 5.3b).



(a) Pantalla de inicio de sesión **(b)** Pantalla de registro

Figura 5.2: Mockups Pantalla de inicio y ajustes



(a) Pantalla de inicio **(b)** Ajustes

Figura 5.3: Mockups Pantalla de inicio y ajustes

Desde la pantalla de búsqueda (ver Figura 5.4a), el usuario podrá realizar búsquedas introduciendo el nombre de una ciudad en el cuadro de texto. Después de pulsar el botón de búsqueda, se mostrará una lista de opciones que coincidan con la palabra introducida (ver Figura 5.4b). Pulsando cualquiera de las opciones, el usuario podrá acceder la información detallada de la ubicación correspondiente.

(a) Mockup Pantalla de búsqueda **(b)** Mockup Resultados de búsqueda

Figura 5.4: Mockups Pantalla de búsqueda

En la pantalla de detalles de una ubicación (ver Figura 5.5), además de mostrarle al usuario la información básica ya mostrada desde la pantalla de inicio, podrá consultar los valores de otros contaminantes atmosféricos. La aplicación, según el índice de calidad del aire, proporcionará recomendaciones/consejos sobre la salud, ayudando al usuario a tomar la decisión más adecuada. Al igual que con las tarjetas de la pantalla de inicio, el color en la pantalla de detalles también irá cambiando según la calidad del aire.

Y por último, en caso de que el usuario haya activado las notificaciones, tanto las notificaciones diarias como las alertas, el usuario recibirá una notificación (ver Figura 5.6), con un mensaje corto y conciso indicándole la situación de la calidad del aire en ese momento.

Figura 5.5: Mockup Pantalla de detalles

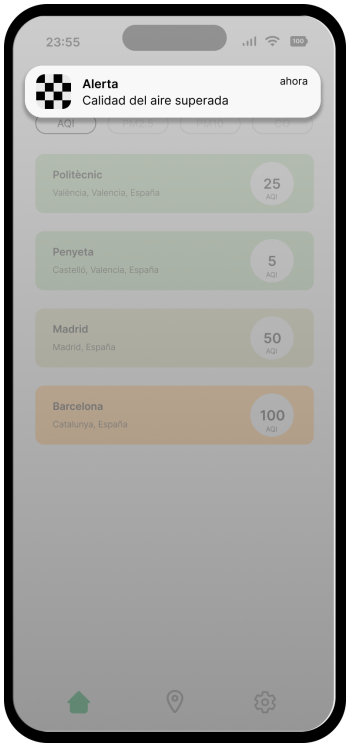


Figura 5.6: Mockup Notificación

CAPÍTULO 6
Implementación

En este capítulo se documentan las tecnologías y herramientas usadas para el desarrollo de la aplicación. También se hará una breve descripción de los aspectos técnicos más importantes.

6.1 Tecnología utilizada

6.1.1. React Native

React Native, utilizado para el desarrollo del *frontend*, es un marco de desarrollo de código abierto. que permite crear aplicaciones nativas para dispositivos iOS y Android utilizando el lenguaje de programación JavaScript y la biblioteca React [13]. A diferencia del desarrollo tradicional de aplicaciones para dispositivos móviles, donde se necesita codificar por separado para las distintas plataformas, React Native permite compartir gran parte del código, agilizando enormemente el desarrollo de aplicaciones multiplataformas.

Con React Native, los desarrolladores pueden crear interfaces de usuario mediante la combinación de componentes que representan los elementos visuales en la pantalla. Los componentes son como las piezas de un rompecabezas, que se pueden juntar y acoplar para construir la interfaz de usuario. Cada componente puede representar un elemento visual o una parte funcional de la aplicación, como un botón, un cuadro de texto o una imagen [14].

Internamente, React Native se basa en tres principales componentes: *JavaScript Thread*, *Bridge* o *Shadow Thread* y *Main Thread* (ver Figura 6.1).

- ***JavaScript Thread*:** o hilo de JavaScript es donde se se ejecuta el código JavaScript de la aplicación. Aquí es donde se desarrolla la lógica, se define las interfaces de usuario y se manejan los eventos.
- ***Bridge*:** actúa como un puente intermediario entre el código de JavaScript y el código nativo de iOS y Android. Permite que los componentes escritos en JavaScript se comuniquen con los módulos nativos de la plataforma. Por ejemplo, un componente de "Vista"se traduce en un contenedor nativo de la plataforma, permitiendo que la aplicación se sienta como una aplicación nativa.
- ***Main Thread*:** se encarga de manejar las operaciones de la interfaz de usuario. Cuando se reciben actualizaciones o instrucciones del *JavaScript Thread* a través del *Bridge*, los componentes nativos procesan esas instrucciones y actualizan la interfaz de usuario.

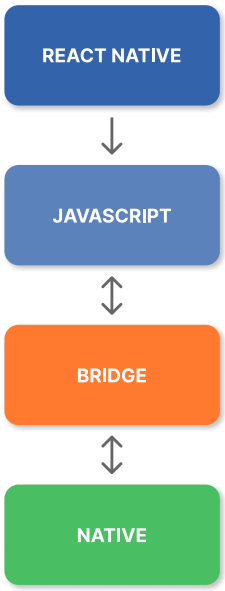


Figura 6.1: Arquitectura simplificada de React Native

Los componentes pueden tener estados, que son como almacenes de datos que pueden cambiar con el tiempo. Cuando el estado de un componente cambia debido a eventos, interacciones del usuario o actualizaciones de datos, la interfaz de usuario se actualiza automáticamente para reflejar esos cambios.

Además, React Native admite *hot reloading*, es decir, los cambios en el código JavaScript se reflejan en la aplicación en ejecución sin tener que reiniciarla. Esto facilita el proceso de desarrollo y depuración.

6.1.2. Expo

Junto con React Native, se ha utilizado Expo, que es un *framework* que simplifica la creación de aplicaciones móviles desarrolladas en React Native [15]. Ofrece servicios que reducen la complejidad involucrada en el desarrollo móvil tradicional, ayudando a los desarrolladores a centrarse en el desarrollo de la aplicación, sin tener que lidiar con las configuraciones técnicas de las distintas plataformas.

Gracias a Expo, los desarrolladores no necesitan configurar y mantener los SDKs (en inglés: *Software Development Kit*) nativos ni ajustes específicos de iOS y Android, simplificando, así la configuración inicial del desarrollo.

Una característica distintiva es que, mediante su propia aplicación móvil Expo Go, los desarrolladores pueden visualizar los cambios que van introduciendo en un emulador o en un dispositivo móvil. Esto permite a los desarrolladores observar y ajustar cómo los cambios se reflejan en tiempo real.

Además, Expo proporciona APIs que permiten acceder a funciones nativas de los dispositivos, como la cámara o el GPS, sin necesidad de escribir código nativo. Por otro lado, simplifica la carga y el uso de recursos multimedia.

Recapitulando, Expo es una plataforma que simplifica el desarrollo al abstraer la complejidad técnica, permitiendo a los desarrolladores enfocarse en la creación de aplicaciones.

6.1.3. Node.js y Express.js

Para el desarrollo del *backend* de la aplicación, se ha decidido usar Express.js. Express.js o simplemente Express es un framework de Node.js minimalista y rápido, que proporciona características y herramientas para desarrollar servidores escalables [16].

Express proporciona herramientas que ayudan en la definición de rutas, gestión de solicitudes y respuestas HTTP, gestión de *middlewares*, etc. Además, Express es muy extensible, ofreciendo la posibilidad de agregar funciones personalizadas utilizando middlewares de terceros.

Una de las características más apreciadas de Express es su capacidad para crear APIs *RESTful*. De esta manera, los desarrolladores pueden construir servicios web que sigan principios de buen diseño y permitan una comunicación sencilla entre diferentes aplicaciones y plataformas.

En resumen, Express es un framework ligero que facilita la creación de aplicaciones web y APIs en Node.js. Su orientación hacia la simplicidad y la modularidad lo convierte en una herramienta poderosa para la construcción de aplicaciones web eficientes y escalables.

6.1.4. MongoDB

MongoDB es un sistema de base de datos NoSQL que destaca por su enfoque en la escalabilidad, la flexibilidad y la capacidad de manejar datos no estructurados o semiestructurados [17].

En lugar de guardar los datos en tablas, como se hace en las bases de datos relacionales, MongoDB utiliza un modelo de documentos para almacenar y organizar la información. Esto quiere decir que guarda los datos en documentos. Estos documentos son almacenados en BSON, que es una representación binaria de JSON, haciendo que la integración de los datos sea más fácil.

Los documentos pueden presentar estructuras diferentes, conteniendo múltiples campos con valores de distintos tipos. De este modo, se elimina la necesidad de crear esquemas rígidos como en las bases de datos relacionales, permitiendo una mayor adaptación a las necesidades del sistema.

Resumiendo, MongoDB es una base de datos NoSQL que ofrece escalabilidad, flexibilidad y capacidad de adaptación a las necesidades cambiantes de las aplicaciones.

6.1.5. Git

Para la gestión de versiones del proyecto, se ha optado por el uso de Git, un software nos que permite controlar y gestionar las versiones del código fuente de la aplicación [18].

La principal diferencia entre Git y cualquier otro VCS (del inglés: *Version Control System*) es la forma en la que manejan los datos. Git los gestiona como un conjunto de copias instantáneas. Cada vez que se confirma un cambio toma una foto del aspecto de todos los archivos en ese momento y guarda una referencia. Para ser eficiente, si los archivos no se han modificado, guarda un enlace al archivo anterior idéntico que ya tiene almacenado.

Git presenta una arquitectura distribuida. A diferencia de otros sistemas de control de versiones que tienen un único espacio para todo el historial de versiones del software, como CVS (*Concurrent Versions System*) o Subversion, en Git la copia de trabajo de ca-

da desarrollador es también un repositorio que puede contener el historial completo de todos los cambios.

En este caso, el proyecto se alojará en un repositorio creado en GitHub, un servicio gratuito de *hosting* de repositorios en la nube.

6.2 Aspectos relevantes de la implementación

Llegados a este punto, se puede apreciar que la aplicación desarrollada no presenta una complejidad muy grande. Esto se debe a que la mayor parte de los datos que utilizamos provienen de una API externa. En otras palabras, solamente será necesario encargarnos de representar los datos en las interfaces de usuario una vez los recibimos de la API.

De esta manera, la responsabilidad principal del desarrollo recae en dos áreas principales: en primer lugar, en el diseño de las interfaces de usuario (la capa de Presentación); y en segundo lugar, en la correcta integración del sistema con la API externa, así como en la comunicación fluida entre el cliente y el servidor.

6.2.1. Desarrollo del frontend

Con el objetivo de sequir buenas prácticas de codificación, asegurando la mantenibilidad, escalabilidad y reutilización eficiente del código, se ha optado por la creación y estructuración de interfaces a través de componentes o módulos reutilizables.

Al dividir la interfaz de usuario en componentes, fue posible modularizar el diseño y la funcionalidad de la aplicación en pequeñas unidades independientes. Cada componente se responsabiliza de una parte específica de la interfaz, promoviendo una mayor claridad en el código. Así también , se estaría cumpliendo con el *Principio de Segregación de la Interfaz*, perteneciente a los principios SOLID [19].

Este enfoque conlleva, también, una reducción en la complejidad del código. Al contar con distintos compomentes para diferentes aspectos de la interfaz, se reduce el número de líneas de código dentro de cada archivo. De este modo, se facilita la compresión y mantenimiento del mismo. Asimismo, el sistema es más escalable, ya que las modificaciones en un componente tienen un impacto limitado en el resto del sistema.

Los componentes reutilizables actúan como bloques de construcción que pueden unirse y acoplarse para dar forma a la interfaz de usuario. Además, la modularización fomenta la reutilización, ya que los diferentes módulos o componentes pueden implementarse en otras partes de la aplicación, minimizando la duplicación de código.

En el Fragmento de código 6.1 se puede visualizar el código correspondiente al componente *SavedLocationCard*. Dentro de la instrucción *return*, se especifica qué datos y cómo se deben presentar al usuario, estructurados mediante el uso del componente visual *View*. En este caso, muestra una tarjeta con información básica sobre una localidad: nombre de la localidad y de la ciudad y el índice de calidad del aire. Además, se aborda cómo debe comportarse en caso de que la obtención de los datos requeridos se retrase o se cancele debido a algún problema o inconveniente.

Los datos necesarios son obtenidos mediante la función *fetchFeed(...)* ejecutada en *useEffect()*. *useEffect* es un *hook* de React, utilizada para construir interfaces interactivas. Se utiliza para realizar acciones que ocurren después de que un componente se monta en el DOM (*Document Object Model*) (por ejemplo, hacer una solicitud a una API).

Luego, el Fragmento de código 6.2 es el componente que se encarga de crear la lista de tarjetas que se mostrarán al usuario. En el componente se realiza un mapeo de los *cityIDs* utilizando el método *.map()*. Por cada *cityID* en la lista, se renderiza un componente *SavedLocationCard* del Fragmento de código 6.1. Se le pasa a cada tarjeta el *cityId* correspondiente y una función *handleNavigate* para dirigir al usuario a la pantalla de los detalles de la ubicación cuando se hace clic en la tarjeta.

En resumen, la adopción de este enfoque de desarrollo ha permitido crear una aplicación móvil más mantenible, escalable y reutilizable. No solo simplifica la comprensión de los archivos, sino que también favorece la inclusión de modificaciones futuras.

```
1  import { useState, useEffect } from "react";
2  import { View, Text, ActivityIndicator, TouchableOpacity } from "react-
     native";
3
4  import styles from "./savedlocationcard.style";
5  import { fetchFeed } from "../../../../hook/useFetch";
6  import { convertToGlobalSchema } from "../../../../hook/dataExtractor";
7  import { COLORS } from "../../../../constants";
8
9  const SavedLocationCard = ({ cityId, handleNavigate }) => {
10   const [info, setInfo] = useState({});
11   const [isLoading, setIsLoading] = useState(true);
12   const [error, setError] = useState(null);
13
14   useEffect(() => {
15     fetchFeed({ cityId: cityId }).then(({ data, isLoading, error }) => {
16       setIsLoading(isLoading);
17       setError(error);
18       setInfo(convertToGlobalSchema(data));
19     });
20   }, []);
21
22   return (
23     <View>
24       {isLoading ? (
25         <View style={{ padding: 25 }}>
26           <ActivityIndicator size="large" color={COLORS.gray} />
27         </View>
28       ) : error ? (
29         <View style={styles.errorContainer}>
30           <Text style={styles.errorText}>
31             {"Error inesperado. Por favor, intentelo de nuevo mas tarde."
   }
32           </Text>
33         </View>
34       ) : (
35         <TouchableOpacity
36           style={styles.container(info.aqi)}
37           onPress={handleNavigate}
38         >
39           <View>
40             <Text style={styles.locationText}>{info.location}</Text>
41             <Text style={styles.cityText}>{info.city}</Text>
42           </View>
43           <View style={styles.infoContainer}>
44             <Text style={styles.infoText(info.aqi)}>{info.aqi}</Text>
45           </View>
46         </TouchableOpacity>
47       )}
48     </View>
49   );
50 };
```

```
51
52   export default SavedLocationCard;
```

Fragmento de código 6.1: Código del componente *SavedLocationCard* correspondiente a las tarjetas con información básica

```
1  import { View } from "react-native";
2  import { useRouter } from "expo-router";
3
4  import styles from "./savedlocations.style";
5  import SavedLocationCard from "../../common/cards/saved/SavedLocationCard";
6
7  const SavedLocations = ({ cityIDs }) => {
8    const router = useRouter();
9
10   return (
11     <View style={styles.cardsContainer}>
12       {cityIDs?.map((id, index) => (
13         <SavedLocationCard
14           key={index}
15           cityId={id}
16           handleNavigate={() => {
17             router.push({
18               pathname: `/details/${id}`,
19               params: {
20                 cityId: id,
21               },
22             });
23           }}
24         />
25       ))}
26     </View>
27   );
28 };
29
30 export default SavedLocations;
```

Fragmento de código 6.2: Código del componente *SavedLocations* correspondiente la lista de las tarjetas

6.2.2. Integración de la API de AQI

Como se ha mencionado anteriormente, la integración de la API de AQI en la aplicación es esencial [20]. Con el objetivo de lograr resultados óptimos, se ha implementado una estrategia de integración de datos virtual.

Esta estrategia de integración, se basa en la creación de un *wrapper*. Este el que se encarga de realizar las consultas a la fuente de datos, en este caso, la API de AQI, y posteriormente transformar los datos obtenidos para que se ajusten al esquema deseado en nuestra aplicación. Esta transformación se lleva a cabo mediante un mapeo semántico, donde los datos recuperados se ajustan y organizan. En esta ocasión, el wrapper está formado por dos archivos (ver Fragmentos 6.3 y 6.4).

Primero, en el Fragmento de código 6.3 se encuentra código que hace la llamada a la API para conseguir los datos necesarios.

En él definen dos funciones, *feedEndpoint* y *searchEndpoint*, que generan las URL de la API en función de los parámetros proporcionados.

La función *apiCall* es el núcleo de este fragmento, ya que es la que maneja las solicitudes HTTP. Dentro de ella, en el bloque *try-catch*, se realiza la solicitud HTTP. Devuelve

un objeto que contiene tres propiedades: *data* (los datos de la respuesta de la API), *isLoading* (indicando si la solicitud está en curso) y *error* (cualquier error que ocurra durante la solicitud).

Finalmente, se exportan dos funciones, *fetchLocations* y *fetchFeed*. *fetchLocations* se emplea para buscar ubicaciones basadas en una palabra clave, mientras que *fetchFeed* se usa para obtener información sobre la calidad del aire de una ciudad específica.

Por otro lado, en el Fragmento de código 6.4 se encuentra el código responsable de realizar la transformación.

En primer lugar, se llama a la función *getGlobalSchema* qye crea y devuelve un objeto con valores predeterminados que representan una estructura deseada para los datos de calidad del aire.

Luego, *convertToGlobalSchema*, utilizando el esquema global generado por la función *getGlobalSchema* como punto de partida, extrae los datos relevantes del objeto pasados como parámetro y los asigna a las propiedades correspondientes en el esquema global.

Por ejemplo, la función divide el nombre de la ubicación para separar la ciudad y la ubicación exacta. También asigna valores para el índice de calidad del aire (AQI) y los valores de diferentes contaminantes, como PM2.5, PM10, O3, NO2, SO2 y CO.

La principal razón de esta transformación de los datos reside en el hecho de que la API de AQI proporciona un conjunto de datos con un tamaño considerable, incluyendo información que, en esta primera versión de la aplicación, no es necesaria. De este modo, se facilita la presentación y procesamiento de los datos en la aplicación y, el resultado final es un objeto que sigue un formato consistente, simplificando su manejo dentro de la aplicación.

Sin embargo, esta estrategia tiene un inconveniente: no almacenamos los datos localmente, lo que significa que la aplicación depende de la conectividad a internet. A pesar de este inconveniente, la estrategia sigue siendo beneficiosa en términos de eficiencia y escalabilidad.

```
1  import axios from "axios";
2
3  import { apiKey } from "../constants";
4
5  const feedEndpoint = (params) => `https://api.waqi.info/feed/${params.
       cityId}/?token=${apiKey}`;
6  const searchEndpoint = (params) => `https://api.waqi.info/v2/search/?token=
       ${apiKey}&keyword=${params.keyword}`;
7
8  const apiCall = async (endpoint) => {
9    var data = [];
10   var isLoading = false;
11   var error = null;
12
13   const options = {
14     method: "GET",
15     url: endpoint,
16   };
17
18   try {
19     isLoading = true;
20     const response = await axios.request(options);
21     data = response.data.data;
22     isLoading = false;
23   } catch (err) {
24     console.log("ERROR: ", err);
```

```
    error = err;
  } finally {
    isLoading = false;
  }

  return { data, isLoading, error };
};

export const fetchLocations = (params) => {
  return apiCall(searchEndpoint(params));
};
export const fetchFeed = (params) => {
  return apiCall(feedEndpoint(params));
};
```

Fragmento de código 6.3: Código de llamada a la API de AQI

```
function getGlobalSchema() {
  return {
    location: "",
    city: "",
    dominant: "",
    aqi: 0,
    pollutants: [
      { name: "PM2.5", value: 0 },
      { name: "PM10", value: 0 },
      { name: "O3", value: 0 },
      { name: "NO2", value: 0 },
      { name: "SO2", value: 0 },
      { name: "CO", value: 0 },
    ],
  };
}

export function convertToGlobalSchema(data) {
  var global = getGlobalSchema();

  let fullLocation = data.city.name;

  global.location = fullLocation.split(",")[0];
  global.city = fullLocation.split(",").slice(1).join(",").trimStart();
  global.dominant = data.dominentpol;
  global.aqi = data.aqi;

  global.pollutants[0].value =
    data.iaqi.pm25 === undefined ? "-" : data.iaqi.pm25.v;
  global.pollutants[1].value =
    data.iaqi.pm10 === undefined ? "-" : data.iaqi.pm10.v;
  global.pollutants[2].value =
    data.iaqi.o3 === undefined ? "-" : data.iaqi.o3.v;
  global.pollutants[3].value =
    data.iaqi.no2 === undefined ? "-" : data.iaqi.no2.v;
  global.pollutants[4].value =
    data.iaqi.so2 === undefined ? "-" : data.iaqi.so2.v;
  global.pollutants[5].value =
    data.iaqi.co === undefined ? "-" : data.iaqi.co.v;

  return global;
}
```

Fragmento de código 6.4: Código para el *mapping* semántico

6.2.3. Comunicación entre el cliente y servidor

La comunicación entre el servidor y el cliente también es importante en nuestro sistema. Utilizamos una arquitectura basada en solicitudes y respuestas a través de protocolos HTTP para lograr una interacción fluida.

Utilizamos métodos HTTP estándar como GET, POST y DELETE para transmitir estas solicitudes. Por ejemplo, cuando el cliente necesita datos de un usuario, envía una solicitud GET al servidor y este los devuelve al cliente en forma de una respuesta HTTP.

Para gestionar esta comunicación de manera ordenada, mantenemos un conjunto de puntos finales (*endpoints*) en el servidor, cada uno asociado con una función o recurso específico. Esto permite al cliente dirigir sus solicitudes al endpoint correspondiente según la acción que desee llevar a cabo.

En resumen, la comunicación entre el servidor y el cliente se basa en un enfoque de solicitudes y respuestas a través de protocolos HTTP. Al seguir un patrón de diseño cliente-servidor logramos una interacción eficiente entre ambas partes en nuestro sistema.

CAPÍTULO 7
Pruebas

Garantizar la calidad de un sistema software es imprescindible, ya que cualquier defecto en el producto final genera insatisfacción en el usuario final, además de la posibilidad de causar un daño irremediable. En casos extremos, un simple error de código puede desencadenar consecuencias catastróficas.

Podemos encontrar numeroros ejemplos de casos en los que un simple fallo o defecto software provoca la pérdida de una gran cantidad de dinero, o incluso un accidente. Por ejemplo, Nissan, fabricante reconocido de automóviles, tuvo que retirar más de un millón de automóviles debido a un defecto de software que causaba el malfuncionamiento del airbag [21]. Otro ejemplo: la explosión del cohete Ariane 5 tras 40 segundos de su lanzamiento en el año 1996, por el simple error de convertir un número almacenado en coma flotante a un numero entero produciendo un *overflow* o desbordamiento. Como consecuencia supuso una pérdida de hasta 500 millones de euros [22].

7.1 Metodología ATDD

Como ya se ha mencionado en la Sección 3.3, se ha llevado a cabo un enfoque de desarrollo basado en pruebas de aceptación (ATDD por sus siglas en inglés). Como indica su nombre, consiste en la definición de una serie de pruebas de aceptación que validan y verifican que el sistema cumple con los requisitos definidos y las espectativas de los usuarios finales.

ATDD [23], al igual que sus hermanas TDD (*Test Driven Development*) y BDD (*Behaviour Driven Development*), es una técnica que usan los equipos ágiles para realizar las pruebas del sistema. De igual manera, en estas técnicas se priorizan las pruebas y estas se definen antes de implementar un requisito o funcionalidad del sistema.

El ciclo de de desarrollo basado en pruebas de aceptación consiste en lo siguiente: i) el equipo discute en colaboración los criterios de aceptación ii) se diseñan y definen los casos de pruebas iii) el equipo de desarrollo implementa los ítems correspondientes iv) por último se testea los nuevos ítems (ver Figura 7.1).

Las principales diferencias [24] entre estas tres técnicas es que, ATDD se centra en la colaboración entre los diferentes roles (desarrolladores, usuarios, testers, etc.) para establecer los casos de prueba. Estos casos ayudan a asegurar que el software cumpla con los criterios de aceptación, permitiendo una comprensión compartida de las expectativas del software desde el inicio.

En cambio, TDD se centra en la definición de pruebas unitarias. El objetivo es guiar el diseño del software y asegurar que cada parte del código esté cubierta. TDD se enfoca más en la lógica interna del sistema.

Por último, BDD se centra en el comportamiento del software desde la perspectiva del usuario. Las especificaciones se expresan en lenguaje natural. Y ayuda a garantizar que el software cumpla con los requisitos y funcionalidades esperados.

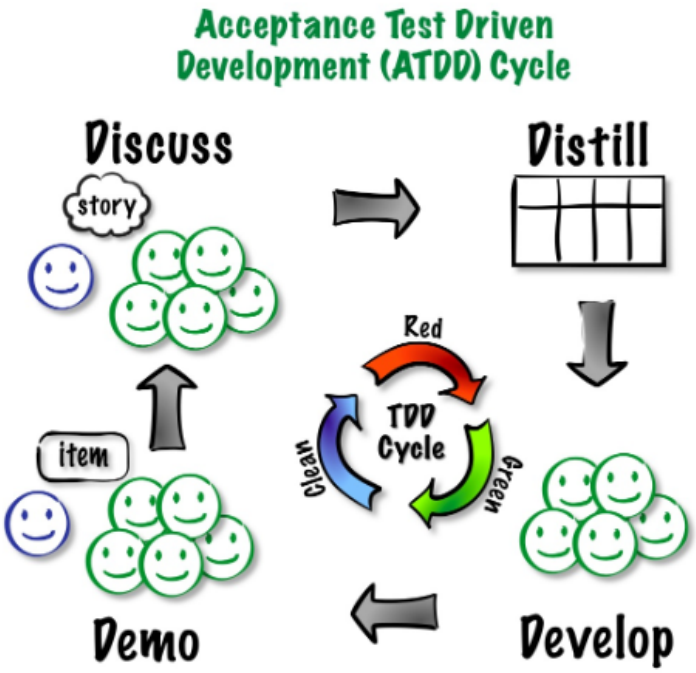


Figura 7.1: Ciclo del enfoque ATDD

Cabe mencionar que, generalmente, una vez definidas, las pruebas suelen automatizarse para agilizar el proceso de testeo. Sin embargo, en esta situación particular, debido a las restricciones inherentes al desarrollo de una aplicación nativa para dispositivos móviles, las pruebas se han realizado manualmente.

La principal ventaja del ATDD es que ayuda a asegurar que todos los miembros del proyecto entiendan qué hay que hacer. Asegura que todos los miembros tengan la misma definición de qué significa cada elemento, una necesidad, requisito, etc., y cuando el ítem está finalizado.

En resumen, el ATDD es una técnica que se centra en crear escenarios de aceptación antes de empezar con la implementación, lo que mejora la comprensión de los requisitos y garantiza que el producto final cumpla con las expectativas del usuario.

7.2 Pruebas unitarias

Con la finalidad de garantizar el correcto funcionamiento de la lógica del sistema, se aplicaron, junto con las pruebas de aceptación, pruebas unitarias.

Las pruebas unitarias son pruebas destinadas a evaluar la funcionalidad individual de un componente o unidad de código específico de forma aislada. Así, se podría evitar errores en la lógica del sistema, de la misma manera se evitarían errores humanos en el código.

Para realizar las pruebas se ha hecho uso Jest, un framework de testing de JavaScript [25]. Jest ofrece diversas ventajas, como una sintaxis sencilla y una amplia gama de utilidades integradas para asistir en la creación de las pruebas.

A continuación se muestran algunos ejemplos de pruebas realizadas para comprobar las distintas funcionalidades del sistema. Concretamente, se muestran las pruebas que verifican el correcto funcionamiento del sistema de favoritos (agregar o eliminar alguna ciudad de favoritos) y de activar las notificaciones diarias (ver Fragmento de código 7.1).

La primera prueba, denominada *Add to favorites*, evalúa la función *AddToFavorites*. Esta función se invoca con dos argumentos: *admin* y *@1234567890*. La expectativa es que esta función agregue con éxito *@1234567890* a la lista de favoritos, y el resultado esperado es un *array* que contiene varios elementos, incluyendo *@1234567890*.

La segunda prueba, llamada *Delete from favorites*, se enfoca en la función *DeleteFromFavorites*. Al igual que en la prueba anterior, se proporcionan los argumentos *admin* y *@1234567890*. El objetivo es verificar si la función elimina *@1234567890* de la lista de favoritos, y se espera un resultado que sea igual al *array* especificado.

Por último, la tercera prueba mostrada, titulada *Enable daily notification*, evalúa, como indica su nombre, la función *EnableDailyNotification*. En este caso, se pasa *admin* y una fecha cualquiera ("1995-12-17T03:24:00") como argumentos. La expectativa es que esta función habilite notificaciones diarias y devuelva *OK* como resultado.

En conjunto, estas pruebas unitarias ayudan a garantizar que las funciones de servicio cumplan con sus respectivas funcionalidades y produzcan los resultados esperados. Esto es fundamental para mantener la calidad y confiabilidad de una aplicación o sistema.

```
1  import { jest, expect, test } from "@jest/globals";
2
3  import {
4    AddToFavorites,
5    DeleteFromFavorites,
6    EnableDailyNotification,
7  } from "../services/services";
8  test("Add to favorites", async () => {
9    const response = await AddToFavorites("admin", "@1234567890");
10   expect(response).toStrictEqual(["@6633", "@6640", "@10114", "@1234567890"
       ]);
11 });
12
13 test("Delete from favorites", async () => {
14   const response = await DeleteFromFavorites("admin", "@1234567890");
15   expect(response).toStrictEqual(["@6633", "@6640", "@10114"]);
16 });
17
18 test("Enable daily notification", async () => {
19   const response = await EnableDailyNotification(
20     "admin",
21     "1995-12-17T03:24:00"
22   );
23   expect(response).toBe("OK");
24 });
```

Fragmento de código 7.1: Fragmento de código en el que se definen varios tests *SavedLocationCard* correspondiente a las tarjetas con información básica

CAPÍTULO 8
Despliegue y mantenimiento

8.1 Mantenimiento y evolución de software

En los últimos años, en el ciclo de vida del software, el mantenimiento del software está adquiriendo cada vez más y más importancia. En términos de recursos requeridos y costos, ya se ha convertido en la actividad principal.

Gracias a seguir al enfoque ágil Scrum y la aplicación de la técnica ATDD, y el seguimiento de los principios SOLID y la aplicación de unas buenas prácticas de codificación, se ha logrado simplificar en gran medida la tarea de mantenimiento del software.

Por un lado, Scrum brinda la flexibilidad necesaria para adaptarse a cambios y mejoras continuas. Con la realización de revisiones periódicas, se puede detectar de forma precoz nuevas necesidades e incluso los errores no detectados anteriormente.

Por otro lado, ATDD ha establecido una base sólida de pruebas que garantizan la integridad de nuestro código. Además, seguir los principios para escribir código limpio y comprensible asegura la legibilidad, mantenibilidad y escalabilidad del software.

Como consecuencia, el proceso de mantenimiento se desarrolla de manera más eficiente, con la capacidad de realizar ajustes, correcciones y mejoras sin sacrificar la calidad ni aumentar los riesgos. Aunque se haya necesitado una mayor inversión de tiempo al inicio del proyecto, esta se refleja en una reducción de costos y esfuerzos a largo plazo.

8.2 Lanzamiento del producto

Llegando a la etapa final, nos encontramos con el último paso: el despliegue del producto final.

Para el frontend, Expo proporciona numerosas herramientas para el proceso de construcción o *build* de la aplicación final, y de despliegue, tanto para la plataforma iOS como Android. Esto agiliza y garantiza un proceso de despliegue sin complicaciones.

Para el lado del servidor y la base de datos, habría que seguir los pasos tradicionales de despliegue. A continuación, se muestran algunos de los pasos más importantes que se deberían de seguir:

1. Seleccionar un proveedor de servicios en la nube, como podría ser AWS (*Amazon Web Services*), Azure (de Microsoft) o *Gloogle Cloud*

2. Configurar y ajustar las variables de entorno, certificados SSL, etc.

3. Implementar medidas de seguridad adecuadas para garantizar la integridad de los datos.

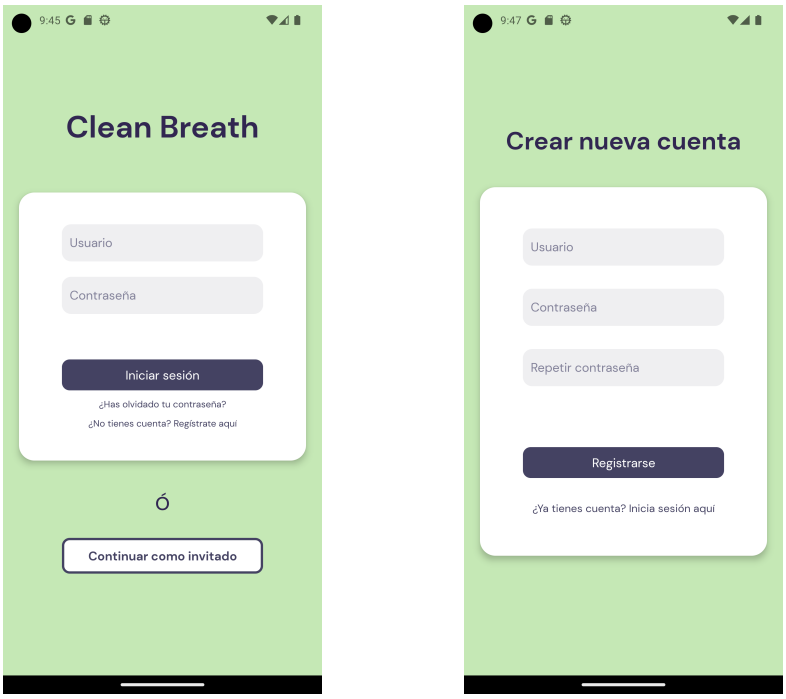
8.3 Descripción del producto final

En este capítulo, se presentará una guía de todas las funciones y características de la aplicación móvil. A través de capturas de pantalla y descripciones, les ayudaremos a navegar por la aplicación y comprender su funcionalidad.

8.3.1. Pantallas Login y Registro

Tras iniciar la aplicación, la primera pantalla mostrada al usuario es la correspondiente al *Login* (ver Figura 8.1a). Desde esta pantalla, el usuario puede iniciar sesión rellenando los campos de *Usuario* y *Contraseña*, en caso de haberse registrado previamente, o acceder a la aplicación como invitado pulsando el botón *Continuar como invitado*.

El usuario también tiene la opción de registrarse. Para ello, tendrá que pulsar en el botón *¿No tienes cuenta? Regístrate aquí*, el cual le llevará a la pantalla de registro (ver Figura 8.1b). Aquí, el usuario puede crear una nueva cuenta completando los campos requeridos: un nombre de usuario, una contraseña y repetir la contraseña.



(a) Pantalla de login de la aplicación **(b)** Pantalla de login de la aplicación

Figura 8.1: Pantallas de Clean Breath

8.3.2. Pantallas Inicio y Buscar

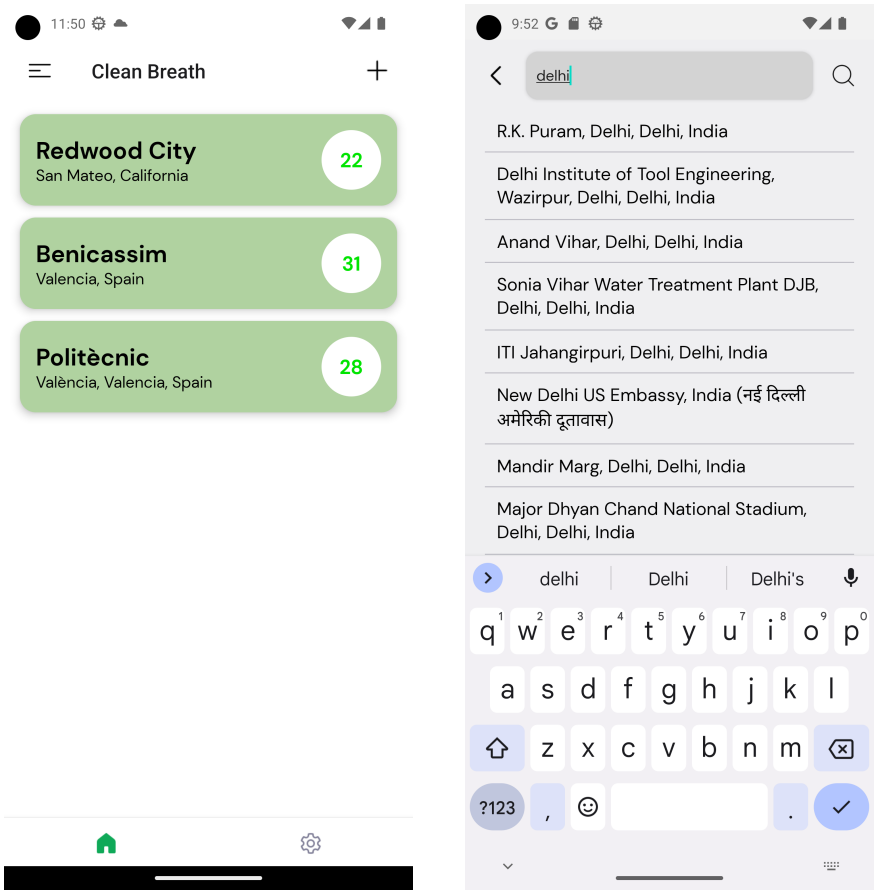
Ya iniciado sesión, ya sea con usuario y contraseña o como invitado, el usuario accederá a la pantalla de inicio de la aplicación (ver Figura 8.1a).

En esta pantalla, el usuario observará una barra superior donde encontrará en la parte derecha un botón con el icono de lupa mediante el cual podrá acceder a la pantalla de

búsqueda. Y en la parte inferior está la barra de navegación con el cual podrá ir a la pantalla de ajustes o volver a la pantalla de inicio.

El usuario verá en esta pantalla tarjetas de ciudades de diferentes colores, según la calidad del aire. En estas tarjetas se muestra el nombre de la localidad y el índice de calidad del aire. Si el usuario está registrado, también se le muestra las ciudades que previamente haya guardado como favoritas.

En la pantalla de búsqueda, el usuario puede buscar cualquier ciudad con introducir el nombre o parte de ella. Pulsando en el botón con el icono de lupa se realizará la búsqueda y se mostrará todos los resultados que coincidan con la palabra introducida (ver Figura 8.2b).

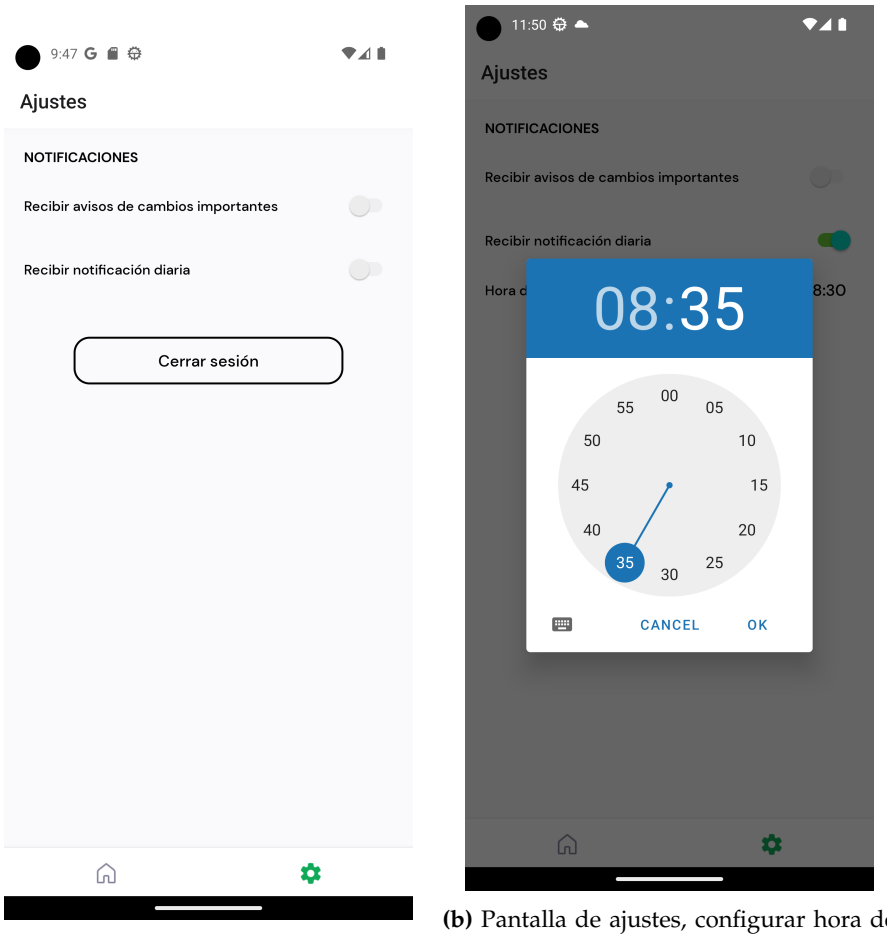


(a) Pantalla de inicio **(b)** Pantalla de búsqueda

Figura 8.2: Pantallas de Clean Breath

8.3.3. Pantalla Ajustes

En la pantalla de ajustes, el usuario podrá configurar las notificaciones de la aplicación (activar o desactivar) y cerrar sesión, que devuelve al usuario a la pantalla de login (ver Figura 8.3a). Al activar las notificaciones diarias, el usuario tendrá la opción de configurar, también, la hora en la que desee recibir la notificación (ver Figura 8.3b). Si el usuario ha accedido como invitado, solamente tendrá la opción de cerrar sesión.



(a) Pantalla de ajustes

(b) Pantalla de ajustes, configurar hora de notificación

Figura 8.3: Pantalla de Ajustes

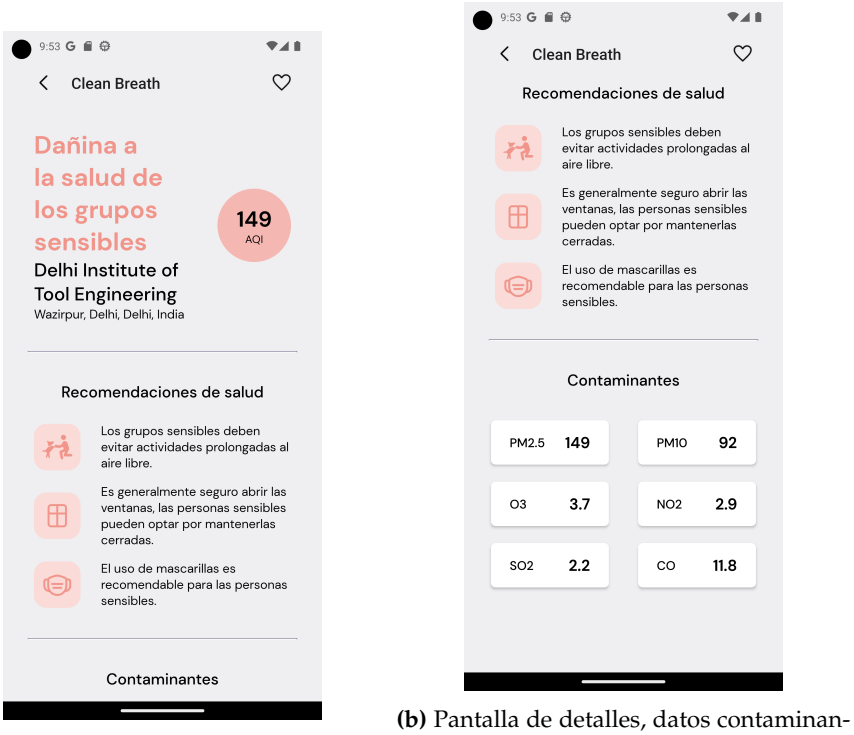
8.3.4. Pantalla Detalles

Seleccionando o pulsando una ciudad, ya sea de la pantalla de inicio como de los resultados de la búsqueda, el usuario puede acceder a la pantalla de detalles de la ciudad. En esta pantalla, a parte de la información básica mostrada en la tarjeta, se le mostrará consejos de salud y datos de los contaminantes presentes en el área (ver Figuras 8.4a y 8.4b).

A parte de representar la condición atmosférica mediante colores, se incluye un mensaje para evitar cualquier confusión, que dice en este caso *Dañina a la salud de los grupos sensibles*. Los valores de los contaminantes atmosféricos están convertidos al estándar AQI de la EPA de EE. UU [26].

En la parte superior derecha, el usuario tendrá la opción de guardar o eliminar una ciudad de la lista de favoritos (si está registrado) pulsando en el botón con el icono de corazón. O puede volver atrás pulsando el botón con el icono de una flecha en la parte

izquierda. Cuando se guarde o elimine una ciudad de favoritos, esto se reflejará también en la pantalla de inicio (ver Figura 8.5).



(a) Pantalla de detalles

(b) Pantalla de detalles, datos contaminantes

Figura 8.4: Pantalla de Detalles

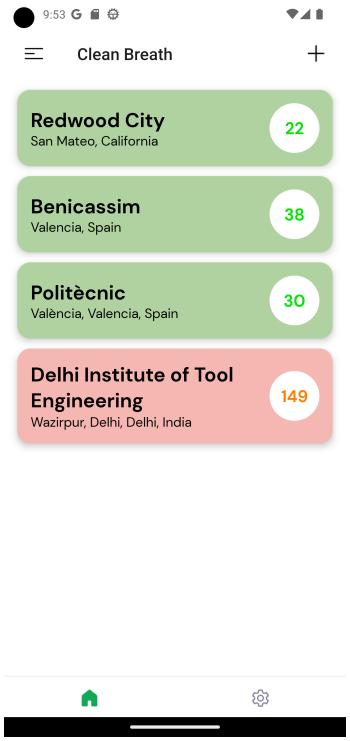


Figura 8.5: Pantalla de inicio actualizada

CAPÍTULO 9
Conclusiones y trabajo futuro

9.1 Conclusiones

Durante la realización de este Trabajo de Fin de Grado, se ha abordado el desarrollo de una aplicación móvil nativa para la consulta de datos de la contaminación ambiental. A través de un enfoque de desarrollo ágil y con la utilización de tecnologías modernas, se ha logrado alcanzar los objetivos establecidos, obteniendo conocimientos valiosos sobre el proceso de desarrollo de aplicaciones.

En el transcurso del proyecto, se ha experimentado cómo la elección de las tecnologías y herramientas, así como la estrategia del proceso de desarrollo tienen un impacto crucial en los resultados. Por un lado, la elección de Scrum ha proporcionado agilidad y adaptabilidad al proceso. Esta metodología ha demostrado ser una guía sólida para mantener un ritmo adecuado de trabajo. Asimismo, la técnica ATDD ha garantizado la calidad del código y el cumplimiento de los requisitos acordados.

Por otro lado, la elección de utilizar la combinación de las tecnologías MongoDB, Express.js, React Native y Node.js ha aportado ventajas significativas. La coherencia entre las tecnologías ha simplificado sustancialmente el proceso de desarrollo, eliminando la necesidad de aprender varios lenguajes al mismo tiempo. Por ejemplo, el uso de React Native con Expo ha permitido un desarrollo eficiente de una aplicación multiplataforma.

Gracias al análisis y estudios de contenidos académicos y recursos digitales, y a la gran comunidad activa de desarrolladores detrás las tecnologías implementadas, ha sido posible conseguir llevar a cabo este proyecto. Esta diversidad de fuentes ha sido primordial para guiar el desarrollo del proyecto.

En conclusión, este proyecto ha demostrado el potencial de la tecnología para abordar desafíos sobre la contaminación ambiental. La aplicación desarrollada se presenta como un paso inicial en esta dirección, y su impacto podría ser aún mayor con mejoras continuas.

Como análisis final, el desarrollo de este Trabajo de Fin de Grado ha sido un recorrido en el que destaca la convergencia de los conocimientos adquiridos durante la formación en la escuela. La unión de tecnologías con enfoques ágiles ha resultado en una solución eficaz con un impacto notorio. Asimismo, revela un poder indiscutible para abordar desafíos del mundo real.

9.2 Relación con las asignaturas cursadas

Es innegable que los conocimientos obtenidos durante el Grado de Ingeniería Informática han desempeñado un papel crucial en la elaboración de este trabajo. Especialmente, las asignaturas cursadas en la rama de Ingeniería de Software han sido de gran relevancia. A continuación, se realizará un repaso de las asignaturas que más han influido en el desarrollo de este trabajo:

- **PSW (Proceso de Software):** ha tenido un impacto significativo en la configuración del proceso de desarrollo de la aplicación. A través de esta asignatura, se ha adquirido conocimientos sólidos de los diversos enfoques y metodologías de desarrollo disponibles, lo que ha facilitado la elección de Scrum como la metodología principal. También ha proporcionado una base sólida para tomar decisiones en la planificación y ejecución de este trabajo.
- **AER (Análisis y Especificación de Software):** ha sido fundamental para comprender la importancia de esta fase en el proceso de desarrollo de software. Además, en esta asignatura se ha explorado los estándares más utilizados en la industria. Gracias a los conocimientos adquiridos en AER, ha sido posible la identificación y especificación de los requisitos principales de la aplicación.
- **IEI (Integración e Interoperabilidad):** ha proporcionado una perspectiva sobre la importancia de la integración de datos. En esta asignatura, se han adquirido conocimientos que resultaron vitales para lograr la integración de las APIs empleadas en la aplicación.
- **AVD (Análisis, Validación y Depuración de software):** ha tenido un impacto importante en la implementación de la técnica de ATDD, resultando en un sistema con una reducida incidencia de errores. También ha sido crucial para llevar a cabo un análisis exhaustivo, validar con precisión y depurar de manera efectiva, contribuyendo significativamente a la calidad y fiabilidad del producto final.

Por último, es importante destacar que también se han requerido competencias transversales que han sido desarrolladas durante el transcurso del Grado. Aquellas que han tenido un mayor impacto en este proyecto son:

- **Aprendizaje permanente:** esta competencia me ha proporcionado la capacidad de adaptarme constantemente a nuevas tecnologías y enfoques en el campo de la informática. A través de esta habilidad, he sido capaz de mantenerme actualizado con las últimas tendencias y adoptar herramientas y metodologías innovadoras, lo que ha enriquecido el proceso de desarrollo de este proyecto y su resultado final.
- **Instrumental específica:** gracias a esta competencia, he sido capaz de aplicar de manera efectiva diversas tecnologías y herramientas en el desarrollo de este proyecto, asegurando la implementación exitosa de funcionalidades clave y garantizando la calidad del producto final.
- **Pensamiento crítico:** gracias a esta habilidad, he sido capaz evaluar de manera objetiva y reflexiva los enfoques, decisiones y soluciones en el desarrollo. También ha enriquecido mi capacidad para abordar problemas complejos y encontrar soluciones en cada etapa de este trabajo.

9.3 Trabajo futuro

Si bien el desarrollo de la aplicación se ha completado satisfactoriamente, existen aún numerosas oportunidades de mejora.

En términos de la metodología empleada, Scrum destaca en entornos de trabajo que implican una colaboración activa entre los diferentes miembros del equipo (programadores, testers, diseñadores...) y los clientes. Sin embargo, podría haber sido más beneficioso optar por una metodología más adecuada para proyectos llevados a cabo por un equipo de trabajo más reducido, en este caso, por una persona. Ya que en este contexto de trabajo, no se podrían aprovechar al máximo los beneficios que podría ofrecer Scrum.

En cuanto a características que quedaron pendientes, destaca la ausencia de un mapa interactivo que permitiera a los usuarios visualizar el índice de calidad del aire en su ubicación y en las ciudades marcadas como favoritas. Así, se habría mejorado la experiencia del usuario al ofrecer una forma más visual de acceder a los datos de la calidad del aire.

También habría sido interesante la incorporación de más de fuentes de datos. Dado que el propósito de la aplicación es proporcionar información sobre la contaminación ambiental para ayudar en la toma de decisiones por parte de los usuarios, esta ampliación habría permitido a la aplicación ofrecer datos más precisos y actualizados, incrementando así la fiabilidad de la aplicación.

Por último, en relación con la experiencia del usuario, se habría mejorado mediante la incorporación de interfaces de usuario más elaboradas y dinámicas. El diseño con animaciones y transiciones fluidas habría enriquecido la experiencia de uso de la aplicación, haciéndola más atractiva.

Bibliografía

[1] El modelo en cascada: Desarrollo secuencial de software. (2019, marzo 11). IONOS Digital Guide. Visitar https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/el-modelo-en-cascada/ Fecha de consulta: 2023, julio 17.

[2] Rosado-Gómez, A., Quintero-Duarte, A., Meneses-Guevara, C. D. (2012, diciembre 30). Desarrollo ágil de software aplicando programación extrema. *Revista Ingenio*, 5(1), 24-29.

[3] Trigás Gallego, M. (2012, junio 18). Metodología Scrum. Visitar https://openaccess.uoc.edu/bitstream/10609/17885/1/mtrigasTFC0612memoria.pdf

[4] Yacelga, A. R. L., Cabrera, M. A. C. (2022, abril 1). Uso de tableros Kanban como apoyo para el desarrollo de las metodologías ágiles. *Universidad y Sociedad*, 14(S2), 208-214.

[5] Letelier, P. (2021, noviembre 24). ¿Qué tienes en tu Backlog (y en tus Sprints)? Visitar http://agilismoatwork.blogspot.com/2021/11/que-contiene-tu-backlog-y-sprints.html Fecha de consulta: 2023, julio 21.

[6] Letelier, P. (2011, diciembre 14). Gestión «Ágil» de Requisitos. Visitar http://agilismoatwork.blogspot.com/2011/12/gestion-de-requisitos-agil.html Fecha de consulta: 2023, julio 21.

[7] Gestiona los proyectos de tu equipo desde cualquier lugar | Trello. Visitar https://trello.com/es

[8] Brooks, F., Kugler, H. (1987, abril). No Silver Bullet – Essence and Accident in Software Engineering. (pp. 1-14).

[9] Sparks, G. (2000). Una introduccion al UML. *El Modelo Lógico*. Visitar http://sparxsystems.com.ar/downloads/whitepapers/El_Modelo_Logico.pdf Fecha consulta: 2023, agosto 24.

[10] IEEE Recommended Practice for Software Requirements Specifications. (1998, octubre 20). IEEE Std 830-1998, 1-40. Visitar https://doi.org/10.1109/IEEESTD.1998.88286

[11] América. (2013, noviembre 19). BASE DE DATOS: ARQUITECTURA MULTICAPA. http://sistemadedatos2013.blogspot.com/2013/11/arquitectura-multicapa.html Fecha de consulta: 2023, agosto 14.

[12] Figma: The Collaborative Interface Design Tool. Visitar https://www.figma.com/ Fecha de consulta: 2023, agosto 25.

[13] React Native · Learn once, write anywhere. Visitar https://reactnative.dev/ Fecha de consulta: 2023, julio 15.

[14] AltexSoft (2021, mayo 24). The Good and the Bad of React Native App Development. Visitar https://www.altexsoft.com/blog/react-native-pros-and-cons/ Fecha de consulta: 2023, agosto 10.

[15] Expo. (s. f.). Visitar https://expo.dev/ Fecha de consulta: 2023, julio 27.

[16] MDN. (2023, julio 26). Express/Node introduction—Learn web development Visitar https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction Fecha de consulta: 2023, agosto 3.

[17] MongoDB: La Plataforma De Datos Para Aplicaciones. (s. f.). Visitar https://www.mongodb.com/es Fecha de consulta: 2023, agosto 3.

[18] Git. Visitar https://git-scm.com/ Fecha de consulta: 2023, agosto 24.

[19] Martín, M. J. (2018, noviembre 21) SOLID: Los 5 principios que te ayudarán a desarrollar software de calidad. *Profile Software Services*. Visitar https://profile.es/blog/principios-solid-desarrollo-software-calidad/ Fecha de consulta: 2023, agosto 24.

[20] API - Air Quality Programmatic APIs. Visitar https://aqicn.org/api/es/

[21] Hirsch, J. (2014, marzo 26). Nissan recalls 1 million cars to fix air bags that may not trigger. *Los Angeles Times*. Visitar https://www.latimes.com/business/la-xpm-2014-mar-26-la-fi-hy-nissan-recalls-altima-sentra-pathfinder-leaf-20140326-sto html Fecha de consulta: 2023, agosto 16.

[22] IBÁÑEZ «ALVY», Á. (2014, junio 4). El error de software que convirtió un lanzamiento espacial en carísimos fuegos artificiales. *RTVE.es*. Visitar https://www.rtve.es/noticias/20140604/error-software-convirtio-lanzamiento-espacial-carisimos-fuegos-artificiales/948262.shtml Fecha de consulta: 2023, agosto 16.

[23] Garzas, J. (2015, julio 21). ¿Qué es eso de ATDD? Javier Garzas. Visitar https://www.javiergarzas.com/2015/07/que-es-eso-de-atdd.html Fecha de consulta: 2023, agosto 16.

[24] JOSEPABLOSARCO (2015, marzo 31). TDD vs BDD vs ATDD. Testing en Español Visitar https://josepablosarco.wordpress.com/2015/03/31/tdd-vs-bdd-vs-atdd/ Fecha de consulta: 2023, agosto 16.

[25] Jest, biblioteca de JavaScript para crear, ejecutar y estructurar pruebas. Visitar https://jestjs.io/es-ES/ Fecha de consulta: 2023, agosto 25.

[26] US EPA, O. (2020, abril 13). National Ambient Air Quality Standards (NAAQS) for PM [Other Policies and Guidance]. Visitar https://www.epa.gov/pm-pollution/national-ambient-air-quality-standards-naaqs-pm Fecha de consulta: 2023, septiembre 1.

APÉNDICE A
Pruebas de Aceptación

Tabla A.1: Prueba de aceptación *Vista de la pantalla principal*

CARACTERÍSTICA: Principal	**ITEM: Pantalla principal**
PRUEBA DE ACEPTACIÓN: Vista de la pantalla principal	
Condiciones	
Pasos	El usuario accede a la aplicación ya sea con o sin cuenta.
Resultado	El usuario puede ver la pantalla de inicio de la aplicación con la información correspondiente

Tabla A.2: Prueba de aceptación *Barra inferior de navegación*

CARACTERÍSTICA: Principal	**ITEM: Pantalla principal**
PRUEBA DE ACEPTACIÓN: Barra inferior de navegación	
Condiciones	
Pasos	El usuario accede a la aplicación ya sea con o sin cuenta.
Resultado	En la parte inferior de la pantalla se muestra una barra de navegación por el cual el usuario puede acceder a ajustes.

Tabla A.3: Prueba de aceptación *Menú superior*

CARACTERÍSTICA: Principal	**ITEM: Pantalla principal**
PRUEBA DE ACEPTACIÓN: Menú superior	
Condiciones	
Pasos	El usuario accede a la aplicación ya sea con o sin cuenta.
Resultado	En la parte superior de la pantalla se muestra un menú por el cual el usuario tiene acceso a la búsqueda de ciudades.
Observaciones	Solamente es visible en la pantalla de inicio, es decir, no será visible en la pantalla de ajustes.

Tabla A.4: Prueba de aceptación *Activar notificaciones diarias*

CARACTERÍSTICA: Principal	**ITEM: Pantalla Ajustes**
PRUEBA DE ACEPTACIÓN: Activar notificaciones diarias	
Condiciones	El usuario debe haber iniciado sesión previamente. El usuario debe haber aceptado recibir notificaciones. El usuario aún no ha activado las notificaciones diarias.
Pasos	1. El usuario pulsa en Ajustes. 2. El usuario pulsa el botón para activar las notificaciones diarias.
Resultado	El usuario recibirá una notificación diaria en la hora predeterminada (9:00)

Tabla A.5: Prueba de aceptación *Desactivar notificaciones diarias*

CARACTERÍSTICA: Principal	**ITEM: Pantalla Ajustes**
PRUEBA DE ACEPTACIÓN: Desactivar notificaciones diarias	
Condiciones	El usuario debe haber iniciado sesión previamente. El usuario debe haber activado las notificaciones diarias.
Pasos	1. El usuario pulsa en Ajustes. 2. El usuario pulsa el botón para desactivar las notificaciones diarias.
Resultado	El usuario dejará de recibir notificaciones diarias.

Tabla A.6: Prueba de aceptación *Ajustar hora de la notificación diaria*

CARACTERÍSTICA: Principal	**ITEM: Pantalla Ajustes**
PRUEBA DE ACEPTACIÓN: Ajustar la hora de la notificación diaria	
Condiciones	El usuario debe haber iniciado sesión previamente. El usuario debe haber activado las notificaciones diarias.
Pasos	1. El usuario pulsa en Ajustes. 2. El usuario ajusta la hora que desea recibir las notificaciones diarias.
Resultado	El usuario recibirá las notificaciones en la hora seleccionada.

Tabla A.7: Prueba de aceptación *Se muestra la ubicación*

CARACTERÍSTICA: Gestión de ciudades	**ITEM: Tarjeta con la información básica de una ciudad**
PRUEBA DE ACEPTACIÓN: Se muestra la ubicación	
Condiciones	
Pasos	1. El usuario accede a la aplicación. 2. El usuario permanece en la pantalla de inicio.
Resultado	Se muestra una tarjeta con la ubicación (ciudad, provincia y el país).

Tabla A.8: Prueba de aceptación *Se muestra el índice de calidad del aire en la tarjeta*

CARACTERÍSTICA: Gestión de ciudades	**ITEM: Tarjeta con la información básica de una ciudad**
PRUEBA DE ACEPTACIÓN: Se muestra el índice de calidad del aire en la tarjeta	
Condiciones	
Pasos	1. El usuario accede a la aplicación. 2. El usuario permanece en la pantalla de inicio
Resultado	En la tarjeta se muestra el índice de calidad del aire actual.

Tabla A.9: Prueba de aceptación *El color se corresponde con la calidad del aire*

CARACTERÍSTICA: Gestión de ciudades	**ITEM: Tarjeta con la información básica de una ciudad**
PRUEBA DE ACEPTACIÓN: El color se corresponde con la calidad del aire	
Condiciones	
Pasos	1. El usuario accede a la aplicación. 2. El usuario permanece en la pantalla de inicio
Resultado	El color de la tarjeta cambia según el índice de calidad del aire.

Tabla A.10: Prueba de aceptación *Se muestra el índice de calidad del aire*

CARACTERÍSTICA: Gestión de ciudades	**ITEM: Pantalla con información detallada de la calidad del aire**
PRUEBA DE ACEPTACIÓN: Se muestra el índice de calidad del aire	
Condiciones	
Pasos	1. El usuario accede a la aplicación. 2. El usuario pulsa en la ubicación que desea.
Resultado	Se abre una nueva pantalla donde el usuario puede ver el índice de calidad del aire.

Tabla A.11: Prueba de aceptación *Se muestran datos de otros contaminantes*

CARACTERÍSTICA: Gestión de ciudades	**ITEM: Pantalla con información detallada de la calidad del aire**
PRUEBA DE ACEPTACIÓN: Se muestran datos de otros contaminantes	
Condiciones	
Pasos	1. El usuario accede a la aplicación. 2. El usuario pulsa en la ubicación que desea.
Resultado	Se abre una nueva pantalla donde el usuario puede ver datos de otros contaminantes.
Observaciones	Cabe la posibilidad de que algún dato no esté disponible. En tal caso, se mostrará "para el contaminante que corresponga

Tabla A.12: Prueba de aceptación *Se muestra recomendaciones de salud*

CARACTERÍSTICA: Gestión de ciudades	**ITEM: Pantalla con información detallada de la calidad del aire**
PRUEBA DE ACEPTACIÓN: Se muestra recomendaciones de salud	
Condiciones	
Pasos	1. El usuario accede a la aplicación. 2. El usuario pulsa en la ubicación que desea.
Resultado	Se abre una nueva pantalla donde la aplicación proporciona ciertas recomendaciones sobre la salud en relación con la calidad del aire del momento.

Tabla A.13: Prueba de aceptación *El color cambia según la calidad del aire*

CARACTERÍSTICA: Gestión de ciudades	**ITEM: Pantalla con información detallada de la calidad del aire**
PRUEBA DE ACEPTACIÓN: El color cambia según la calidad del aire	
Condiciones	
Pasos	1. El usuario accede a la aplicación. 2. El usuario pulsa en la ubicación que desea.
Resultado	El tono de color de la pantalla de detalles cambia según la calidad del aire. Por ejemplo, el color será verde si la calidad es buena.

Tabla A.14: Prueba de aceptación *El usuario puede buscar ciudades por el nombre*

CARACTERÍSTICA: Gestión de ciudades	**ITEM: Búsqueda de ciudades**
PRUEBA DE ACEPTACIÓN: El usuario puede buscar ciudades por el nombre	
Condiciones	
Pasos	1. El usuario accede a la aplicación. 2. El usuario pulsa en el botón de búsqueda en el menú superior.
Resultado	Se abre una nueva ventana con una entrada de texto donde el usuario puede introducir el nombre de la ciudad que desee.

Tabla A.15: Prueba de aceptación *Se muestran las opciones que coinciden con la búsqueda*

CARACTERÍSTICA: Gestión de ciudades	**ITEM: Búsqueda de ciudades**
PRUEBA DE ACEPTACIÓN: Se muestran las opciones que coinciden con la búsqueda	
Condiciones	
Pasos	1. El usuario accede a la aplicación. 2. El usuario pulsa en el botón de búsqueda en el menú superior. 3. El usuario introduce el nombre de la ciudad que desea buscar. 4. El usuario pulsa el botón de búsqueda.
Resultado	Se le muestra al usuario una lista de opciones que coinciden con el nombre o texto introducido.

Tabla A.16: Prueba de aceptación *El usuario puede guardar una ciudad como favorito*

CARACTERÍSTICA: Gestión de ciudades	**ITEM: Añadir ciudad a favoritos**
PRUEBA DE ACEPTACIÓN: El usuario puede guardar una ciudad como favorito	
Condiciones	El usuario debe haber iniciado sesión previamente. Y la ciudad no ha sido añadida a favoritos previamente.
Pasos	1. El usuario accede a la aplicación. 2. El usuario pulsa en el botón de búsqueda en el menú superior. 3. El usuario busca la ciudad que desea. 4. El usuario pulsa en la ciudad desea. 5. El usuario pulsa en el botón en forma de corazón en la parte superior.
Resultado	La ciudad se añade correctamente.

Tabla A.17: Prueba de aceptación *El usuario puede eliminar una ciudad como favorito*

CARACTERÍSTICA: Gestión de ciudades	**ITEM: Añadir ciudad a favoritos**
PRUEBA DE ACEPTACIÓN: El usuario puede eliminar una ciudad como favorito	
Condiciones	El usuario debe haber iniciado sesión previamente. Y la ciudad debe haber sido añadido como favorito previamente.
Pasos	1. El usuario accede a la aplicación. 2. El usuario pulsa en la ciudad que desea. 3. El usuario pulsa en el botón en forma de corazón en la parte superior.
Resultado	La ciudad se elimina correctamente.

Tabla A.18: Prueba de aceptación *El usuario puede ver las ciudades guardadas en la pantalla de inicio*

CARACTERÍSTICA: Gestión de ciudades	**ITEM: Añadir ciudad a favoritos**
PRUEBA DE ACEPTACIÓN: El usuario puede ver las ciudades guardadas en la pantalla de inicio	
Condiciones	El usuario debe haber iniciado sesión previamente.
Pasos	El usuario accede a la aplicación.
Resultado	Se muestra en la pantalla de inicio una lista de tarjetas de las ciudades que el usuario ha guardado como favoritos.
Observaciones	Puede darse el caso de que no se muestren las ciudades recién guardadas. En tal caso, el usuario deberá recargar la pantalla arrastrando hacia abajo.

Tabla A.19: Prueba de aceptación *El usuario introduce un nombre de usuario no válido*

CARACTERÍSTICA: Gestión de usuarios	**ITEM: Registro**
PRUEBA DE ACEPTACIÓN: El usuario introduce un nombre de usuario no válido	
Condiciones	
Pasos	1. El usuario inicia la aplicación. 2. El usuario pulsa en el botón para registrarse. 3. El usuario introduce un nombre de usuario no válido (menos de 3 caracteres o más de 20 caracteres). Por ejemplo: aa.
Resultado	Se muestra un mensaje de error indicando que el nombre de usuario introducido no es válido.

Tabla A.20: Prueba de aceptación *El usuario introduce una contraseña no válida*

CARACTERÍSTICA: Gestión de usuarios	**ITEM: Registro**
PRUEBA DE ACEPTACIÓN: El usuario introduce una contraseña no válida	
Condiciones	
Pasos	1. El usuario inicia la aplicación. 2. El usuario pulsa en el botón para registrarse. 3. El usuario introduce una contraseña no válida (menos de 6 caracteres). Por ejemplo: 12345.
Resultado	Se muestra un mensaje de error indicando que la contraseña introducida no es válida.

Tabla A.21: Prueba de aceptación *Las contraseñas introducidas no coinciden*

CARACTERÍSTICA: Gestión de usuarios	**ITEM: Registro**
PRUEBA DE ACEPTACIÓN: Las contraseñas introducidas no coinciden	
Condiciones	
Pasos	1. El usuario inicia la aplicación. 2. El usuario pulsa en el botón para registrarse. 3. El usuario introduce una contraseña válida (de al menos de 6 caracteres). 4. El usuario introduce una contraseña distinta en el campo de Repetir contraseña.
Resultado	Se muestra un mensaje de error indicando que las contraseñas introducidas no coinciden.

Tabla A.22: Prueba de aceptación *Registro correcto*

CARACTERÍSTICA: Gestión de usuarios	**ITEM: Registro**
PRUEBA DE ACEPTACIÓN: Registro correcto	
Condiciones	Todos los datos introducidos son correctos.
Pasos	1. El usuario inicia la aplicación. 2. El usuario pulsa en el botón para registrarse. 3. El usuario rellena los campos requeridos. 4. El usuario pulsa el botón de registrarse.
Resultado	El usuario se registra correctamente y accede a la aplicación.

Tabla A.23: Prueba de aceptación *Inicio fallido*

CARACTERÍSTICA: Gestión de usuarios	**ITEM: Inicio de sesión**
PRUEBA DE ACEPTACIÓN: Inicio fallido	
Condiciones	Los datos introducidos son incorrectos. O el usuario no se ha registrado aún.
Pasos	1. El usuario inicia la aplicación. 2. El usuario introduce un nombre de usuario y una contraseña.
Resultado	Se muestra un mensaje de error indicando que el nombre de usuario o contraseñan son incorrectos

Tabla A.24: Prueba de aceptación *Inicio de sesión con éxito*

CARACTERÍSTICA: Gestión de usuarios	**ITEM: Inicio de sesión**
PRUEBA DE ACEPTACIÓN: Inicio de sesión con éxito	
Condiciones	El usuario se ha registrado previamente. Y los datos introducidos son correctos.
Pasos	1. El usuario inicia la aplicación. 2. El usuario introduce un nombre de usuario y una contraseña.
Resultado	El usuario inicia sesión con éxito y accede a la aplicación

Tabla A.25: Prueba de aceptación *Inicio de sesión con éxito*

CARACTERÍSTICA: Gestión de usuarios	**ITEM: Iniciar sesión como invitado**
PRUEBA DE ACEPTACIÓN: Inicio de sesión con éxito	
Condiciones	
Pasos	1. El usuario inicia la aplicación. 2. El usuario pulsa el botón de iniciar sesión como invitado.
Resultado	El usuario inicia sesión con éxito y accede a la aplicación
Observaciones	El usuario accede a la aplicación con ciertas limitaciones

Tabla A.26: Prueba de aceptación *Cerrar sesión*

CARACTERÍSTICA: Gestión de usuarios	**ITEM: Cerrar sesión**
PRUEBA DE ACEPTACIÓN: Cerrar sesión	
Condiciones	El usuario debe haber iniciado sesión previamente.
Pasos	1. El usuario pulsa en Ajustes. 2. El usuario pulsa el botón de cerrar sesión.
Resultado	Se cierra sesión con éxito.

APÉNDICE B
Relación con los ODS

En 2015, la Organización de las Naciones Unidas (ONU) aprobó la Agenda 2030 sobre el Desarrollo Sostenible, una oportunidad para que los países y sus sociedades emprendan un nuevo camino con el que mejorar la vida de todos. La Agenda define un total de 17 Objetivos de Desarrollo Sostenible (ODS) de aplicación universal para impulsar el crecimiento económico, el compromiso con las necesidades sociales y la protección del medio ambiente. Cada objetivo tiene metas específicas que deben alcanzarse en los próximos 15 años.

A continuación, se muestra una tabla con todos los 17 objetivos y el grado de relación de cada uno con este trabajo.

Tabla B.1: Tabla de grado de relación del trabajo con los ODS

Objetivos de Desarrollo Sostenible	**Alto**	**Medio**	**Bajo**	**No procede**
ODS 1. **Fin de la pobreza.**				x
ODS 2. **Hambre cero.**				x
ODS 3. **Salud y bienestar.**	x			
ODS 4. **Educación de calidad.**				x
ODS 5. **Igualdad de género.**				x
ODS 6. **Agua limpia y saneamiento.**				x
ODS 7. **Energía asequible y no contaminante.**			x	
ODS 8. **Trabajo decente y crecimiento económico.**				x
ODS 9. **Industria, innovación e infraestructuras.**	x			
ODS 10. **Reducción de las desigualdades.**				x
ODS 11. **Ciudades y comunidades sostenibles.**	x			
ODS 12. **Producción y consumo responsables.**		x		
ODS 13. **Acción por el clima.**	x			
ODS 14. **Vida submarina.**				x
ODS 15. **Vida de ecosistemas terrestres.**		x		
ODS 16. **Paz, justicia e instituciones sólidas.**				x
ODS 17. **Alianzas para lograr objetivos.**				x

Observando la Tabla B.1, se puede afirmar que este proyecto de desarrollo de una aplicación móvil, destinada a proporcionar información sobre la calidad del aire y los niveles de contaminantes diferentes ubicaciones, presenta una sólida relación con varios ODS.

El ODS 3, *Salud y bienestar*, se ve directamente influenciado por la aplicación. La calidad del aire tiene un impacto directo en la salud humana, al exponerse a contaminantes que puede agravar o incluso provocar enfermedades respiratorias y cardiovasculares. Proporcionar datos precisos y actualizados ayuda a los usuarios a tomar decisiones informadas sobre su exposición ambiental, en última instancia, mejorando la salud y el bienestar.

El ODS 9, *Industria, innovación e infraestructura*, se relaciona intrínsecamente con el desarrollo de este proyecto. La adopción de tecnologías como React Native y Node.js muestra cómo la innovación tecnológica contribuye a un desarrollo más sostenible. Aprovechando estas herramientas, la aplicación desarrollada, mejora la eficiencia en la consulta de datos de la calidad del aire, aumentando, al mismo tiempo, la accesibilidad a la información.

La información sobre la calidad del aire puede influir en decisiones clave sobre la planificación urbana. De este modo, la aplicación está promoviendo el ODS 11, *Ciudades y comunidades sostenibles*. Al aumentar la conciencia sobre la importancia de la calidad del aire, esta aplicación apoya el desarrollo de comunidades más saludables.

Más moderadamente, esta aplicación se relaciona con el ODS 13, *Acción por el clima*, ya que al monitorear los niveles de contaminantes, promueve de manera indirecta una mayor comprensión de cómo nuestras acciones pueden afectar a la contaminación atmosférica, pudiendo también contribuir a la preservación del clima global.

Del mismo modo, el usuario, al estar más consciente de las consecuencias de sus acciones, la aplicación también puede influir en las decisiones de consumo afectando las emisiones y huella ecológica. Estamos hablando de una relación con el ODS 12, *Producción y consumo responsables*.

Asimismo, el ODS 15, *Vida de ecosistemas terrestres*, también puede verse influido. La contaminación atmosférica no solo tiene efectos en los seres humanos sino que también puede afectar negativamente en la salud de los ecosistemas terrestres. Al permitir una mayor comprensión de los niveles de contaminantes, esta aplicación puede desempeñar un papel crucial en la conservación de estos ecosistemas.

Por último, el ODS 7, *Energía asequible y no contaminante*, puede tener una relación más periférica. Si bien la aplicación no se centra en estas áreas, la implementación de tecnologías modernas y el potencial de la aplicación para influir en decisiones de movilidad podrían tener un impacto indirecto en la promoción de energías limpias y de la movilidad sostenible.

En resumen, esta aplicación móvil nativa establece una relación directa y significativa con los ODS 3, 9, 11 y 13. Además, su influencia en los ODS 12 y 15 es relevante pero menos directa, y su conexión con el ODS 7 es más periférica. En conjunto, este proyecto muestra cómo la tecnología y la información pueden desempeñar un papel clave en promover un desarrollo sostenible y en la búsqueda de un futuro más saludable.