



UNIVERSIDAD POLITÉCNICA DE MADRID

Escuela Técnica Superior de Ingeniería de Sistemas Informáticos

Máster Universitario en Ingeniería Web

Trabajo Fin de Máster

Aplicación Móvil Meteorológica – WeatherYr

Autor

Bastián Tobar Mori

Tutor

Francisco Javier Gil Rubio

Enero de 2025



AGRADECIMIENTOS.

A mi familia, por su apoyo incondicional, a mi hija Leonor por acompañarme en esta aventura, y a mi pareja Yocelin por apoyarme en todo momento. A mi tutor, Francisco Javier Gil Rubio, por su tiempo y dedicación en este proyecto. ¡Muchas gracias!



RESÚMEN.

Este Trabajo de Fin de Máster tiene como objetivo aplicar y consolidar los conocimientos adquiridos a lo largo del Máster en Ingeniería Web mediante el desarrollo de una aplicación móvil denominada WeatherYr.

WeatherYr es una aplicación meteorológica que permite conocer distintos parámetros climáticos según un punto buscado. Entre las variables disponibles se incluyen la temperatura mínima y máxima, la velocidad dirección del viento, presión atmosférica, calidad del aire, entre otros.

Para la implementación del sistema se ha utilizado una arquitectura cliente-servidor con las siguientes tecnologías:

- Android Studio para el desarrollo de la capa de presentación, proporcionando una interfaz amigable para los usuarios.
- Spring Boot para la capa de negocio, encargada del procesamiento de la información y la integración con la API meteorológica.
- YR.no Api, el servicio del Instituto Meteorológico de Noruega, para la obtención de datos meteorológicos en tiempo real.
- Nominatim Api, servicio que permite hacer geocodificación, es decir, convertir una dirección o nombre de lugar en coordenadas geográficas (latitud y longitud).
- Air Quality Api, servicio que proporciona información en tiempo real sobre la calidad del aire en una ubicación determinada
- Sunrise-Sunset Api es un servicio gratuito que te permite obtener las horas de salida y puesta del sol (sunrise & sunset), así como otros datos astronómicos relacionados, para una ubicación determinada.
- NASA APOD Api (Astronomy Picture of the Day) es un servicio gratuito que proporciona diariamente una imagen o video del espacio junto con una explicación astronómica, accesible por fecha.
- Firebase se utiliza para la gestión de datos y almacenamiento en la nube, permitiendo funcionalidades como la consulta de pronósticos actuales, y pronósticos que estén dentro de las próximas 48 horas.

El desarrollo de la aplicación se enfoca en la eficiencia, la escalabilidad y la usabilidad, garantizando un acceso rápido y preciso a la información meteorológica. Además, se han implementado pruebas para asegurar la fiabilidad del sistema y su correcto funcionamiento en diferentes dispositivos Android.

PALABRAS CLAVES.

Android, Spring, Firebase, Datos meteorológicos.



ABSTRACT.

The aim of this Master's Thesis is to apply and consolidate the knowledge acquired throughout the Master's Degree in Web Engineering by developing a mobile application called WeatherYr.

WeatherYr is a weather application that allows users to find out different climate parameters for a specific location. The variables available include minimum and maximum temperature, wind speed and direction, atmospheric pressure, air quality, among others.

A client-server architecture was used to implement the system, employing the following technologies:

- Android Studio for the development of the presentation layer, providing a user-friendly interface.
- Spring Boot for the business layer, responsible for processing information and integrating with the weather API.
- YR.no API, the Norwegian Meteorological Institute's service, for obtaining real-time meteorological data.
- Nominatim API, a service that allows geocoding, i.e., converting an address or place name into geographical coordinates (latitude and longitude).
- Air Quality API, a service that provides real-time information on air quality at a given location.
- Sunrise-Sunset API is a free service that allows you to obtain sunrise and sunset times, as well as other related astronomical data, for a specific location.
- NASA APOD API (Astronomy Picture of the Day) is a free service that provides a daily image or video of space along with an astronomical explanation, accessible by date.
- Firebase is used for data management and cloud storage, enabling features such as checking current forecasts and forecasts for the next 48 hours.

The development of the application focuses on efficiency, scalability, and usability, ensuring fast and accurate access to weather information. In addition, tests have been implemented to ensure the reliability of the system and its correct operation on different Android devices.

KEYWORDS

Android, Spring, Firebase, Weather data.



CONTENIDO

AGRADECIMIENTOS	2
RESÚMEN	3
ABSTRACT.....	4
1. INTRODUCCIÓN	1
1.1 CONTEXTO Y MOTIVACIÓN.....	2
1.2 OBJETIVOS	2
1.2.1 OBJETIVO GENERAL.....	2
1.2.2 OBJETIVOS ESPECÍFICOS.....	2
1.3 ESTRATEGIAS DE DESARROLLO.....	3
1.4 ESTRUCTURA DEL DOCUMENTO.....	3
2. MARCO TEÓRICO.	4
2.1 APLICACIONES METEOROLÓGICAS EXISTENTES.....	4
2.1.1 ACCUWEATHER:.....	4
2.1.2 THE WEATHER CHANNEL.....	4
3. CARACTERÍSTICAS DE “WEATHERYR”	5
3.1 APIS METEOROLÓGICOS Y DATOS CLIMÁTICOS.....	5
3.2 APIS UTILIZADAS EN “WEATHERYR”	6
3.3 FUNCIONAMIENTO E INTEGRACIÓN EN WEATHERYR.	12
4. FRAMEWORKS Y TECNOLOGÍAS SELECCIONADAS PARA WEATHERYR	13
4.1 FRAMEWORK.....	13
4.2 ANDROID STUDIO	13
4.3 SPRING BOOT.....	13
4.4 FIREBASE.....	14
4.5 VENTAJAS DE UTILIZAR ESTE STACK.....	14
4.6 ARQUITECTURA DE LA APLICACIÓN	15
4.6.1 CAPA DE PRESENTACIÓN – CLIENTE ANDROID (ANDROID STUDIO).....	15
4.6.2 CARACTERÍSTICAS PRINCIPALES DEL CLIENTE ANDROID.....	16
4.6.3 CAPA DE LÓGICA DE NEGOCIO – BACKEND CON SPRING BOOT.....	16
4.6.4 CAPA DE DATOS – FIREBASE.....	17
4.6.4.1 VENTAJAS DE USO DE FIREBASE.	17
5. REQUISITOS Y DISEÑO.....	18
5.1 MODELO DE DOMINIO.....	19



5.2 CASOS DE USO	20
5.2.1 DIAGRAMA DE CASOS DE USO.....	20
5.2.2 ESPECIFICACIÓN DE CASOS DE USO.....	21
5.2.2.1 CASO DE USO LOGIN.	21
5.2.2.2 CASO DE USO LOGOUT.....	22
5.2.2.3 CASO DE USO MENÚ PRINCIPAL.....	23
5.3 DIAGRAMA DE CONTEXTO.	24
6. IMPLEMENTACIÓN.....	25
6.1 DESARROLLO DEL BACKEND CON AUTENTICACIÓN Y ALERTAS METEOROLÓGICAS.	25
6.1.2 ARQUITECTURA DEL BACKEND.....	25
6.1.2.1 TECNOLOGÍAS UTILIZADAS.	25
6.1.3 IMPLEMENTACIÓN DE LA AUTENTICACIÓN.	26
6.1.4 FUNCIONALIDADES DEL BACKEND.....	26
6.1.5 FUNCIONALIDADES DEL BACKEND.....	27
6.2 ARQUITECTURA DEL FRONTEND.....	28
6.2.1 TECNOLOGÍAS UTILIZADAS.....	28
6.2.2 FUNCIONALIDADES DEL FRONTEND	28
6.3 INTEGRACIÓN DE FUNCIONALIDADES CLAVE.....	29
6.3.1 GRÁFICOS METEOROLÓGICOS.....	29
6.3.2 MAPAS.....	30
6.3.3 PRONÓSTICO DE 48 HORAS.....	30
7. PRUEBAS Y RESULTADOS.....	31
7.1 PRUEBAS DEL BACKEND.	31
7.1.1 MÉTODOS DE PRUEBA.	32
7.1.2 LISTA DE ENDPOINTS TESTEADOS.....	34
7.2 PRUEBAS DEL FRONTEND.....	35
7.2.1 RESULTADOS	35
8. CONCLUSIONES Y MEJORAS.	45
8.1 CONCLUSIONES.....	45
8.2 MEJORAS Y FUTURAS AMPLIACIONES.	46
9. ANEXOS.	48
10. REFERENCIAS.	67



1. INTRODUCCIÓN

En la era digital actual, la demanda de aplicaciones móviles que proporcionen información precisa y en tiempo real es más alta que nunca. El campo de la meteorología no es una excepción, y con el avance de la tecnología, las personas buscan formas más accesibles y confiables de obtener datos climáticos. Este Trabajo de Fin de Máster en Ingeniería Web explora el desarrollo de "WeatherYr", una aplicación móvil avanzada diseñada para ofrecer información meteorológica detallada y actualizada.

WeatherYr se propone como una solución tecnológica que integra varios componentes de software y hardware para mejorar la interacción del usuario con datos meteorológicos precisos. El objetivo principal de este proyecto es aplicar y consolidar los conocimientos adquiridos durante el Máster, a través del desarrollo de una aplicación que no solo cumpla con las expectativas funcionales, sino que también aborde desafíos como la escalabilidad, la eficiencia y la usabilidad.

Para la implementación del sistema, se adoptó una arquitectura cliente-servidor utilizando tecnologías actualmente demandadas en el mercado. Android Studio fue seleccionado para el desarrollo de la capa de presentación, ofreciendo una interfaz intuitiva y amigable para los usuarios. La capa de negocio, desarrollada con Spring Boot, se encarga del procesamiento de la información y la integración con las distintas API's meteorológica, garantizando una aplicación no solo funcional sino también escalable y confiable.

Además, Firebase se utiliza para la gestión de datos y almacenamiento en la nube, permitiendo funcionalidades como la consulta de pronósticos actuales, y pronósticos que estén dentro de las próximas horas.

La finalidad de este documento es detallar el desarrollo y la implementación de la aplicación WeatherYr, destacando la metodología de investigación, los desafíos técnicos superados y las estrategias adoptadas para asegurar una aplicación eficiente y orientada al usuario.



1.1 CONTEXTO Y MOTIVACIÓN.

La relevancia de la información meteorológica se ha intensificado en nuestra sociedad, convirtiéndose en un recurso crítico para la toma de decisiones en diversos sectores. Desde la planificación de actividades recreativas hasta la gestión de operaciones en agricultura y transporte, el acceso a datos climáticos precisos y actualizados es esencial para anticipar y adaptarse a las condiciones atmosféricas. A pesar del progreso en tecnología móvil y la disponibilidad de APIs para datos meteorológicos, muchas aplicaciones existentes aún presentan desafíos significativos. Estos incluyen interfaces de usuario complejas, necesidad de configuraciones exhaustivas, lo que puede obstaculizar la experiencia del usuario.

1.2 OBJETIVOS.

1.2.1 OBJETIVO GENERAL.

Desarrollar una aplicación móvil para la consulta y análisis de datos meteorológicos, que integre de manera efectiva la información en tiempo real, el almacenamiento en la nube, y las visualizaciones gráficas interactivas, mejorando así la accesibilidad y la experiencia del usuario.

1.2.2 OBJETIVOS ESPECÍFICOS.

1. Diseñar e implementar una interfaz intuitiva que facilite la visualización de los datos climáticos.
2. Optimizar la interacción con distintas APIs para garantizar la recepción de datos climáticos exactos y actualizados.
3. Integrar servicios de autenticación para permitir el almacenamiento y uso seguro de datos.
4. Implementar interfaces para simplificar la comprensión de tendencias climáticas a lo largo del tiempo.



1.3 ESTRATEGIAS DE DESARROLLO.

Para establecer una estrategia de desarrollo que permita cumplir con los objetivos mencionados en mi proyecto de fin de máster para la aplicación WeatherYr, he realizado y estructurado un plan con los siguientes pasos:

- 1. Investigación y análisis preliminar:** En este punto se debe determinar las necesidades específicas de los usuarios y los requisitos técnicos para la aplicación. Además, evaluar otras aplicaciones similares para identificar fortalezas y áreas de mejora. Finalmente, después de investigar y analizar los datos anteriores se debe decidir y escoger las herramientas y tecnologías más adecuadas para el desarrollo del proyecto (Android Studio, Spring Boot, Firebase, APIs meteorológicas).
- 2. Definir el diseño de la arquitectura:** Aquí se debe definir tanto la arquitectura general, así como también el diseño de Backend y Frontend.
- 3. Desarrollo e implementación:** Se debe llevar a cabo el desarrollo del Backend y Frontend.
- 4. Pruebas y Validación:** Es necesario realizar test de pruebas unitarios para garantizar el correcto funcionamiento de la aplicación.
- 5. Documentación del servidor mediante cliente Swagger.**
- 6. Ajustes finales:** Mejoras de rendimiento y usabilidad.

1.4 ESTRUCTURA DEL DOCUMENTO.

El presente trabajo final de título, está organizado de la siguiente manera:

- 1. Introducción:** Presenta el contexto y motivación, objetivos generales y específicos, y estrategia de desarrollo del proyecto.
- 2. Marco Teórico:** Describe los conceptos fundamentales y tecnologías utilizadas.
- 3. Requisitos y Diseño:** Explica los casos de uso y modelo de dominio.
- 4. Implementación:** Detalla la implementación del Frontend y Backend.
- 5. Pruebas y Resultados:** Describe las pruebas realizadas y los resultados obtenidos.
- 6. Conclusiones y Mejoras:** Analiza los resultados alcanzados y propone mejoras futuras.
- 7. Anexos y Referencias:** Incluye documentación complementaria y bibliografía.



2. MARCO TEÓRICO.

Según AccuWeather, el mundo de hoy cambia muy rápidamente y el clima se está volviendo más errático, tenemos infinitas repercusiones del calentamiento global, y el comportamiento y las expectativas de los consumidores están cambiando.

Es por eso que la proliferación de aplicaciones meteorológicas ha aumentado ya que estas permiten que los usuarios accedan a datos climáticos en tiempo real desde cualquier dispositivo móvil. Existen diversas aplicaciones que ofrecen pronósticos detallados, mapas interactivos y alertas climáticas. La elección de la API adecuada dependerá de las necesidades específicas del consumidor, como la precisión requerida, la cobertura geográfica y las características adicionales que se pueda necesitar.

A continuación, decidí presentar 2 de las aplicaciones más utilizadas en el mercado según un estudio de la y una comparación con la aplicación realizada “WeatherYr”, destacando sus características y diferencias.

2.1 APPLICACIONES METEOROLÓGICAS EXISTENTES.

2.1.1 ACCUWEATHER:

Es una de las aplicaciones más reconocidas en el ámbito meteorológico. Su servicio se basa en su propia API de pronósticos y proporciona información detallada como temperatura, índice UV, precipitaciones y calidad del aire. Una de sus características más valoradas es el pronóstico minuto a minuto, que ofrece predicciones precisas para la próxima hora. Sin embargo, sus desventajas es que muchas de sus funciones avanzadas requieren una suscripción premium y la versión gratuita contiene bastantes anuncios publicitarios.

2.1.2 THE WEATHER CHANNEL.

Propiedad de IBM, esta aplicación ofrece pronósticos detallados con modelos predictivos avanzados. Se distingue por sus mapas interactivos de radar y cobertura global del clima. Además, proporciona alertas meteorológicas en tiempo real, lo que la convierte en una opción ideal para quienes requieren actualizaciones constantes. No obstante, al igual que AccuWeather, la mayoría de sus características avanzadas están bloqueadas detrás de un modelo de suscripción, lo que puede limitar su accesibilidad.



3. CARACTERÍSTICAS DE “WEATHERYR”

Si bien las aplicaciones mencionadas anteriormente ofrecen información meteorológica de calidad, WeatherYr introduce algunos elementos diferenciadores que la distinguen dentro de su segmento:

1. Interfaz optimizada y sin publicidad:

Muchas aplicaciones gratuitas incluyen anuncios o restringen funcionalidades clave bajo un modelo de suscripción. WeatherYr ofrece una experiencia de usuario libre de publicidad y sin restricciones en sus funciones principales.

2. Gráficos en tiempo real:

La integración de gráficos facilita la visualización de datos meteorológicos, proporcionando a los usuarios una interpretación sencilla y clara de las condiciones climáticas y tendencias a lo largo del tiempo.

3. Interfaz intuitiva y de fácil uso:

Se priorizará el diseño de una interfaz gráfica clara, accesible y coherente, que facilite la interacción del usuario con el sistema sin necesidad de conocimientos técnicos avanzados. Se aplicarán principios de usabilidad y diseño centrado en el usuario para garantizar una experiencia fluida. En conclusión, WeatherYr busca proporcionar una alternativa más accesible, intuitiva y eficiente para los usuarios que requieren información meteorológica específica en Madrid, integrando tecnologías actuales en el mercado, para mejorar la experiencia de consulta.

3.1 APIS METEOROLÓGICOS Y DATOS CLIMÁTICOS.

La obtención de datos meteorológicos en tiempo real es un aspecto clave para cualquier aplicación climática. Las Apis meteorológicas permiten acceder a información precisa sobre temperatura, humedad, velocidad del viento y otros parámetros sin necesidad de infraestructura propia de monitoreo climático. Gracias a estas Apis, las aplicaciones móviles pueden ofrecer pronósticos actualizados, mejorar la planificación de actividades y proporcionar alertas en caso de condiciones adversas.

Para el desarrollo de WeatherYr, se optado por nutrir la información de distintas Api, de las cuales podemos generar un reporte detallado del clima según un punto específico.



A continuación, se detallan las Apis empleadas.

3.2 APIS UTILIZADAS EN “WEATHERYR”

a) YR.no Api – LocationForecast 2.0 (Compact Version).

Para la implementación de WeatherYr, se ha optado por utilizar la API LocationForecast 2.0 de YR.no, un servicio proporcionado por el Instituto Meteorológico de Noruega. Esta API destaca por su precisión y fiabilidad, ya que proviene de una entidad meteorológica reconocida a nivel internacional.

Datos que proporciona la Api.

La API LocationForecast 2.0 (versión compact) devuelve una gran cantidad de datos en formato JSON, organizados en series temporales con predicciones horarias. Algunos de los parámetros más importantes que se extraen y almacenan en Firebase son:

- Temperatura mínima y máxima (°C)
- Velocidad y dirección del viento (m/s y grados)
- Probabilidad de precipitación (%)
- Humedad relativa (%)
- Presión atmosférica (hPa)
- Condiciones generales del tiempo (descripción de nubes, sol, lluvia, etc.)

Ejemplo de consulta a la Api

Una petición típica a la Api LocationForecast 2.0 se realiza mediante una URL con las coordenadas de la ubicación deseada (en este caso, Madrid):

- GET <https://Api.met.no/weatherApi/locationforecast/2.0/compact?lat=40.4168&lon=-3.7038>

Nota: Para evitar bloqueos por parte del servicio, YR.no requiere que las solicitudes incluyan un User-Agent personalizado con el nombre de la aplicación.



b) API de Nominatim – OpenStreetMap

Nominatim es un motor de búsqueda de geocodificación y geocodificación inversa basado en los datos abiertos y colaborativos de OpenStreetMap (OSM). Su función principal es transformar direcciones textuales en coordenadas geográficas (latitud y longitud), y viceversa. Es una herramienta esencial para aplicaciones que necesitan ubicar puntos en un mapa a partir de nombres de lugares o, inversamente, identificar lugares a partir de coordenadas.

Funcionalidades Principales

- Geocodificación (Forward Geocoding): Proceso de convertir una dirección postal completa, un nombre de ciudad, un punto de interés o cualquier cadena de texto que describa una ubicación, en sus correspondientes coordenadas geográficas (latitud y longitud).
 - Ejemplo: "Torre Eiffel, París" se transforma en -33.4377591, -70.650445 (latitud, longitud).
- Geocodificación Inversa (Reverse Geocoding): Proceso de convertir un par de coordenadas geográficas (latitud y longitud) en una dirección postal legible o un nombre de lugar cercano.
 - Ejemplo: Las coordenadas -32.8000578, -71.145221 se transforman en "Hijuelas, Valparaíso, Chile".

Interfaz de la Api

La Api de Nominatim es accesible a través de solicitudes HTTP GET y devuelve las respuestas en formatos como JSON (el más comúnmente utilizado) o XML. Las consultas se realizan añadiendo parámetros a la URL base del servicio.

- URL Base Pública (ejemplo): <https://nominatim.openstreetmap.org/>
- Parámetros de Consulta Típicos:
 - q: Para búsquedas de texto libre (geocodificación).
 - lat, lon: Para búsquedas por coordenadas (geocodificación inversa).
 - format: Define el formato de la respuesta (ej., json).
 - limit: Limita el número de resultados devueltos.

Ejemplo de Uso (Geocodificación)

Solicitud (HTTP GET):

<https://nominatim.openstreetmap.org/search?q=Hijuelas%2C+Valparaiso%2C+Chile&format=json&limit=1>



c) Api de Calidad del Aire: OpenWeatherMap Air Quality Api

Es un servicio que proporciona acceso a datos históricos, actuales y de pronóstico sobre la contaminación del aire a nivel global. Permite integrar información detallada sobre el Índice de Calidad del Aire (AQI) y las concentraciones de diversos contaminantes atmosféricos en sus aplicaciones.

Funcionalidades Principales

- **Datos Actuales de Calidad del Aire:** Permite obtener el AQI actual y las concentraciones de los principales contaminantes para una ubicación geográfica específica (mediante latitud y longitud).
- **Pronóstico de Calidad del Aire:** Ofrece la posibilidad de consultar el AQI y las concentraciones de contaminantes para las próximas horas o días.
- **Componentes de Contaminantes:** Proporciona datos detallados sobre las concentraciones de gases clave y partículas, incluyendo:
 - CO (Monóxido de Carbono)
 - NO (Monóxido de Nitrógeno)
 - NO₂ (Dióxido de Nitrógeno)
 - O₃ (Ozono)
 - SO₂ (Dióxido de Azufre)
 - PM2.5 (Partículas finas con un diámetro aerodinámico de 2.5 micrómetros o menos)
 - PM10 (Partículas respirables con un diámetro aerodinámico de 10 micrómetros o menos)
 - NH₃ (Amoníaco)
- **Índice de Calidad del Aire (AQI):** Proporciona un valor numérico y una categoría descriptiva del AQI (ej., "Bueno", "Moderado", "Poco Saludable"), lo que facilita la interpretación de la calidad del aire para el usuario final.

Interfaz de la Api

La Api se accede a través de solicitudes HTTP GET a endpoints específicos, utilizando la latitud y longitud como parámetros principales. Requiere una clave API (API Key), que se obtiene al registrarse en OpenWeatherMap.

- **URL Base (ejemplo):** <http://Api.openweathermap.org/>
- **Endpoint Típico (Actual):** /data/2.5/air_pollution
- **Parámetros de Consulta Típicos:**
 - lat: Latitud.
 - lon: Longitud.
 - appid: Tu clave API de OpenWeatherMap.



Políticas de Uso y Consideraciones

- **Requisito de Clave Api:** A diferencia de la instancia pública de Nominatim, la API de OpenWeatherMap requiere siempre una clave API válida para autenticar las solicitudes.
- **Límites del Plan Gratuito:** OpenWeatherMap ofrece un plan gratuito que tiene límites en el número de llamadas por minuto y por día. Para aplicaciones con un alto volumen de usuarios, es necesario considerar la suscripción a un plan de pago.
- **Actualización de Datos:** Los datos de calidad del aire se actualizan con una frecuencia que depende de las estaciones de monitoreo subyacentes, pero suelen ser razonablemente recientes para aplicaciones de monitoreo en tiempo real.

Ejemplo de Uso (Datos Actuales de Calidad del Aire)

Solicitud (HTTP GET):

GET

https://Api.openweathermap.org/data/2.5/air_pollution?lat={latitud}&lon={longitud}&appid={tu_Api_key}

d) Api Astronómica: Sunrise-Sunset.org Api

Es un servicio web sencillo y gratuito diseñado específicamente para calcular los tiempos de amanecer, atardecer, el mediodía solar y las diferentes fases del crepúsculo para cualquier ubicación geográfica y fecha.

Funcionalidades Principales

Esta Api se especializa en proporcionar datos horarios para los siguientes eventos astronómicos diarios:

- **sunrise:** La hora en que el sol aparece por el horizonte por la mañana.
- **sunset:** La hora en que el sol desaparece por el horizonte por la tarde.
- **solar_noon:** La hora en que el sol alcanza su punto más alto en el cielo (mediodía solar).
- **day_length:** La duración total del día, desde el amanecer hasta el atardecer.
- **civil_twilight_begin** y **civil_twilight_end:** El inicio y fin del crepúsculo civil.
- **nautical_twilight_begin** y **nautical_twilight_end:** El inicio y fin del crepúsculo náutico.
- **astronomical_twilight_begin** y **astronomical_twilight_end:** El inicio y fin del crepúsculo astronómico.
- **status:** Indica el estado de la solicitud (ejemplo: "OK" para éxito).



Interfaz de la Api

La Api de Sunrise-Sunset.org se accede mediante solicitudes HTTP GET. No requiere una clave API para su uso estándar, lo que simplifica su implementación.

- **URL Base:** <https://Api.sunrise-sunset.org/>
- **Endpoint Típico:**
 - /json
- **Parámetros de Consulta:**
 - lat (Float, Requerido): Latitud de la ubicación.
 - lng (Float, Requerido): Longitud de la ubicación.
 - date (String, Opcional): Fecha en formato YYYY-MM-DD. Si se omite, se usa la fecha actual.
 - formatted (Integer, Opcional): 0 para tiempos en formato ISO 8601 (UTC), 1 (por defecto) para tiempos en formato de 12 horas AM/PM.

Consideraciones Importantes

- **Zona Horaria (UTC):** Todos los tiempos devueltos por la Api están en UTC (Tiempo Universal Coordinado). La aplicación cliente es responsable de convertir estos tiempos a la zona horaria local del usuario si es necesario para una visualización correcta.
- **Uso Libre:** La API es gratuita y generalmente no tiene límites de tasa estrictos documentados para usos razonables.

Ejemplo de Uso:

Solicitud (HTTP GET):

GET <https://Api.sunrise-sunset.org/json?lat=-32.8000578&lng=-71.145221&formatted=0>

e) Api de Contenido Astronómico: NASA – Astronomy Picture of the Day (APOD)

La Api de NASA – Astronomy Picture of the Day (APOD) es un servicio ofrecido por la NASA que proporciona una nueva imagen o fotografía astronómica cada día, acompañada de una breve explicación escrita por un astrónomo profesional. Esta Api es una fuente rica y accesible de contenido visual y educativo relacionado con el espacio, la astronomía y la exploración espacial, ideal para integrar en aplicaciones que buscan ofrecer una experiencia diaria inspiradora y con base científica.



Principales funciones:

La API APOD permite acceder a la "Imagen Astronómica del Día" para una fecha específica o para la fecha actual. Los datos proporcionados para cada entrada incluyen:

- **date:** La fecha de la imagen en formato YYYY-MM-DD.
- **explanation:** Una descripción concisa y educativa de la imagen o fenómeno astronómico.
- **hdurl:** La URL de la imagen en alta definición.
- **media_type:** Indica el tipo de contenido (principalmente "image" para fotografías o "video" para videos de YouTube/Vimeo).
- **service_version:** La versión de la API utilizada.
- **title:** El título de la imagen del día.
- **url:** La URL de la imagen en resolución estándar (o la miniatura del video, si media_type es "video").
- **copyright (Opcional):** Información de derechos de autor, si aplica.

Fuente de Datos y Alcance

Las imágenes y explicaciones son seleccionadas y producidas por el personal del Astronomy Picture of the Day en la NASA, lo que garantiza la calidad y veracidad del contenido.

Interfaz de la API

La API de APOD se accede mediante solicitudes HTTP GET a su endpoint principal. Es fundamental que cada solicitud incluya una clave API (`Api_key`) para la autenticación. La NASA proporciona una "`DEMO_KEY`" para fines de prueba, pero recomienda obtener una clave personal para cualquier aplicación en producción.

- **URL Base:** <https://api.nasa.gov/planetary/apod>
- **Parámetros de Consulta:**
 - `Api_key` (String, Requerido): Tu clave API de la NASA.
 - `date` (String, Opcional): La fecha de la imagen deseada en formato YYYY-MM-DD. Si se omite, se devuelve la imagen del día actual.
 - `hd` (Boolean, Opcional): true para solicitar la URL de la imagen en alta definición (`hdurl`), false para la resolución estándar (`url`). Por defecto es false.
 - `count` (Integer, Opcional): Si se especifica, devuelve un número aleatorio de imágenes. No se puede usar con `date`.
 - `thumbs` (Boolean, Opcional): Si es un video, devuelve la URL de la miniatura.



Políticas de Uso y Consideraciones

- **Clave Api Requerida:** A diferencia de Apis sin autenticación, cada llamada a la Api de APOD debe incluir una Api_key válida.
- **Límites de Tasa:** La API tiene límites de tasa (típicamente 50 solicitudes por día para la DEMO_KEY y 1000 solicitudes por hora para una clave personal registrada). Es crucial gestionar estas solicitudes para evitar bloqueos.
- **Unicidad Diaria:** La API está diseñada para proporcionar una "Imagen del Día". Si necesitas un flujo continuo de imágenes astronómicas, esta API no es la más adecuada por sí sola, pero es excelente para una característica diaria.
- **Contenido:** Principalmente imágenes y ocasionalmente videos.

Ejemplo de Uso

Solicitud (HTTP GET):

GET https://Api.nasa.gov/planetary/apod?Api_key=DEMO_KEY&date=2023-01-20&hd=true

3.3 FUNCIONAMIENTO E INTEGRACIÓN EN WEATHERYR.

A continuación, se detalla cómo se realizará la integración de las Apis con la aplicación móvil.

1. Consulta a la Api desde el backend:
 - La aplicación no accede directamente a las Apis desde el frontend. En su lugar, se realiza una petición HTTP desde el backend desarrollado en Spring Boot.
2. Filtrado de datos relevantes:
 - Dado que las Apis devuelven JSON extenso con múltiples valores, se extraen únicamente los parámetros más importantes para la aplicación.
3. Almacenamiento en Firebase:
 - Los datos obtenidos se almacenan en Firebase Realtime Database, lo que permite que el Frontend acceda a consultas recientes sin necesidad de realizar múltiples peticiones a la API.
4. Visualización en la aplicación:
 - El Frontend de la aplicación en Android Studio obtiene los datos desde Firebase y los muestra en la interfaz, con gráficos descriptivos para mejorar la experiencia del usuario.



4. FRAMEWORKS Y TECNOLOGÍAS SELECCIONADAS PARA WEATHERYR

4.1 FRAMEWORK

Para el desarrollo de WeatherYr, se ha propuesto utilizar los siguientes frameworks para la capa de cliente, servidor y base de datos.

4.2 ANDROID STUDIO

Android Studio es el entorno de desarrollo integrado (IDE) oficial para el desarrollo de aplicaciones Android. Está basado en IntelliJ IDEA y es desarrollado por Google. Entre sus particularidades destacan: Soporta el desarrollo en Java, Kotlin y C++, incluye herramientas avanzadas como el Android Emulator, el Layout Editor y el APK Analyzer, tiene integración con Gradle, lo que permite gestionar dependencias y automatizar la compilación del código. Por otro lado soporta Jetpack y Material Design, facilitando el diseño de interfaces modernas. También ofrece herramientas de profiling y debugging para optimizar el rendimiento de las aplicaciones. Finalmente permite la integración con Firebase y otras APIs de Google.

4.3 SPRING BOOT

Spring Boot es un framework basado en Java que facilita la creación de aplicaciones backend con Spring Framework. Su objetivo es reducir la configuración manual y permitir el desarrollo de servicios RESTful y microservicios de manera rápida. Entre sus particularidades destacan que proporciona una estructura de autoconfiguración, eliminando la necesidad de configuraciones extensas en XML, incluye un servidor embebido (como Tomcat o Jetty), permitiendo ejecutar aplicaciones sin necesidad de desplegarlas manualmente. Además, es compatible con JPA/Hibernate para la persistencia de datos en bases de datos relacionales.

Por otro lado, soporta la creación de APIs RESTful, integrándose con JSON, XML y otros formatos de intercambio de datos. También posee herramientas de seguridad integradas mediante Spring Security, y finalmente es ideal para arquitecturas de microservicios, ya que puede integrarse con Docker y Kubernetes.



4.4 FIREBASE

Es una plataforma en la nube de Google que proporciona un backend completo para el desarrollo de aplicaciones web y móviles. Ofrece servicios como bases de datos en tiempo real, autenticación, almacenamiento y hosting. Entre sus características podemos mencionar las siguientes:

- Firebase Authentication permite autenticación de usuarios mediante correo, Google, Facebook, entre otros.
- Cloud Firestore y Realtime Database para el almacenamiento de datos en la nube en tiempo real.
- Cloud Functions permite ejecutar código backend sin necesidad de servidores.
- Firebase Hosting ofrece alojamiento para aplicaciones web.
- Firebase Cloud Messaging (FCM) permite enviar notificaciones push a dispositivos Android e iOS.
- Firebase Analytics proporciona métricas detalladas sobre el comportamiento de los usuarios en la app.

4.5 VENTAJAS DE UTILIZAR ESTE STACK

1. Eficiencia en el desarrollo: Android Studio proporciona herramientas avanzadas como el Layout Editor, el Emulador de Android y el soporte para Jetpack, lo que acelera el desarrollo de la interfaz de usuario. Por otro lado, Spring Boot elimina configuraciones manuales complejas, gracias a su autoconfiguración y servidor embebido, permitiendo desarrollar APIs backend de manera rápida. Finalmente, Firebase reduce la necesidad de desarrollar infraestructura backend desde cero, ya que ofrece servicios como autenticación, bases de datos en tiempo real y notificaciones push listas para usar.

2. Arquitectura escalable y flexible: Spring Boot facilita la creación de microservicios, permitiendo escalar el backend de forma modular según la demanda, y Firebase nos permite escalar automáticamente el almacenamiento de datos y la autenticación de usuarios sin preocuparse por la infraestructura.



3. Integración y sincronización de datos en tiempo real: Con Firebase Realtime Database o Cloud Firestore, la aplicación móvil puede obtener actualizaciones de datos en tiempo real sin necesidad de realizar peticiones manuales. Esto es útil para aplicaciones de mensajería, notificaciones en vivo, seguimiento de pedidos o cualquier sistema que requiera datos en tiempo real.

4. Seguridad y autenticación sin complicaciones: Firebase Authentication nos permite implementar inicio de sesión con Google, Facebook, correo y otros métodos de autenticación con muy pocas líneas de código. Además, Spring Security en Spring Boot proporciona un sistema robusto para proteger el Backend con OAuth2, JWT o autenticación basada en sesiones.

5. Reducción de costos y mantenimiento: Firebase ofrece un plan gratuito con recursos suficientes para aplicaciones pequeñas y medianas, evitando la necesidad de contratar servidores dedicados. Además, Spring Boot es de código abierto y permite implementar backend en la nube sin licencias costosas. Finalmente, el stack completo es compatible con DevOps, permitiendo despliegues automatizados con CI/CD mediante GitHub Actions, Firebase Hosting, Kubernetes y Sonarcloud.

6. Compatibilidad con múltiples plataformas: Spring Boot permite crear una API REST consumible tanto por una aplicación móvil Android como por una aplicación web. Finalmente, Firebase tiene SDKs para Android, iOS y Web, lo que facilita la integración de múltiples clientes en el mismo backend.

4.6 ARQUITECTURA DE LA APLICACIÓN

El desarrollo de WeatherYr se basa en una arquitectura cliente-servidor, en la que la aplicación móvil Android interactúa con un backend desarrollado en Spring Boot, el cual se encarga de procesar y almacenar los datos meteorológicos obtenidos desde YR.no. Para garantizar un rendimiento óptimo, la información procesada se almacena en Firebase, permitiendo acceso en tiempo real, autenticación de usuarios y notificaciones push mediante Cloud Messaging.

La arquitectura de WeatherYr está compuesta por tres capas principales, cada una con un propósito específico:

4.6.1 CAPA DE PRESENTACIÓN – CLIENTE ANDROID (ANDROID STUDIO).

La capa de presentación es el componente con el que interactúa el usuario. Ha sido desarrollada en Android Studio utilizando Java/Kotlin, siguiendo un diseño modular basado en el patrón MVVM (Model-View-ViewModel). Este enfoque permite una mejor separación de responsabilidades y facilita la escalabilidad de la aplicación.



4.6.2 CARACTERÍSTICAS PRINCIPALES DEL CLIENTE ANDROID.

1. Interfaz gráfica intuitiva:

- Diseñada con Material Design para garantizar una experiencia de usuario óptima.
- Uso de gráficos interactivos y listas dinámicas para mostrar datos meteorológicos.

2. Consumo de datos desde el backend:

- La app realiza peticiones REST al backend en Spring Boot para obtener información meteorológica almacenada en Firebase.
- Se emplea Retrofit, una librería eficiente para la comunicación con APIs REST.

3. Autenticación de usuarios con Firebase Authentication:

- Permite inicio de sesión con correo electrónico.
- Gestión de sesiones seguras para personalizar la experiencia del usuario.

4. Notificaciones Push con Firebase Cloud Messaging (FCM):

- Envío de alertas meteorológicas en tiempo real a los usuarios.
- Posibilidad de activar o desactivar notificaciones desde la configuración de la aplicación.

4.6.3 CAPA DE LÓGICA DE NEGOCIO – BACKEND CON SPRING BOOT.

El backend de WeatherYr ha sido desarrollado con Spring Boot, un framework basado en Java que facilita la creación de APIs REST escalables y eficientes.

Funciones principales del backend:

1. Obtención de datos meteorológicos desde las APIs:

- Se realizan peticiones a las APIs para obtener pronósticos detallados.
- Se filtran y estructuran los datos antes de ser almacenados en Firebase Realtime Database.

2. Exposición de APIs REST para el Cliente Android:

- Se implementan endpoints REST para que la app móvil pueda consultar datos meteorológicos actualizados.

3. Integración con Firebase para almacenamiento de datos:

- Una vez procesados, los datos climáticos se almacenan en Firebase Realtime Database, asegurando acceso rápido.



4. Gestión de autenticación con Firebase Authentication:

- o El backend verifica los usuarios autenticados antes de permitir el acceso a ciertas funcionalidades.

5. Envío de notificaciones push con Firebase Cloud Messaging (FCM):

- o Se gestionan alertas meteorológicas y se envían notificaciones a los usuarios cuando se detectan eventos climáticos importantes.

4.6.4 CAPA DE DATOS – FIREBASE.

Firebase es el servicio de almacenamiento y gestión de datos en la nube utilizado en WeatherYr. Su integración con Spring Boot y Android permite una comunicación fluida y en tiempo real.

Servicios de Firebase utilizados:

1. Firebase Realtime Database:

- o Almacena los datos meteorológicos obtenidos desde las Apis.
- o Permite que el cliente Android consulte información sin depender de las Apis externas.
- o Sincronización en tiempo real, asegurando que los usuarios siempre tengan datos actualizados.

2. Firebase Authentication:

- o Gestiona el inicio de sesión con correo electrónico y contraseña.
- o Garantiza acceso seguro a los datos del usuario.

3. Firebase Cloud Messaging (FCM):

- o Permite el envío de notificaciones push para alertas meteorológicas.
- o Integrado con el backend en Spring Boot, que gestiona la activación de las notificaciones.

4.6.4.1 VENTAJAS DE USO DE FIREBASE.

El uso de Android Studio, Spring Boot y Firebase proporciona múltiples beneficios:

1. Eficiencia y rendimiento:

- o La arquitectura cliente-servidor reduce la carga en la aplicación móvil.
- o Firebase permite acceso rápido a los datos sin necesidad de peticiones repetitivas a las Apis externas.



2. Escalabilidad:
 - Gracias a Spring Boot, el backend puede ampliarse fácilmente con nuevas funcionalidades.
 - Firebase maneja grandes volúmenes de datos sin afectar el rendimiento.
3. Seguridad y autenticación:
 - Firebase Authentication garantiza un acceso seguro a los datos de los usuarios.
 - Las comunicaciones entre cliente y servidor están protegidas.
4. Notificaciones en tiempo real:
 - Con Firebase Cloud Messaging, los usuarios reciben alertas meteorológicas en tiempo real.
5. Experiencia de usuario optimizada:
 - La interfaz en Android Studio se ha diseñado para ofrecer una navegación intuitiva.
 - Se han implementado gráficos y listas dinámicas para mejorar la visualización de datos.

5. REQUISITOS Y DISEÑO.

Esta sección aborda los Requisitos y Diseño de WeatherYr., esenciales para cumplir con los objetivos del proyecto. Para ello, se definen los requisitos no funcionales, algunos de los cuales ya se derivan de las tecnologías seleccionadas.

Los requisitos funcionales serán abordados siguiendo la metodología RUP (Rational Unified Process), y se detallan las interacciones del sistema, fundamentales para la resolución de los requisitos específicos planteados.

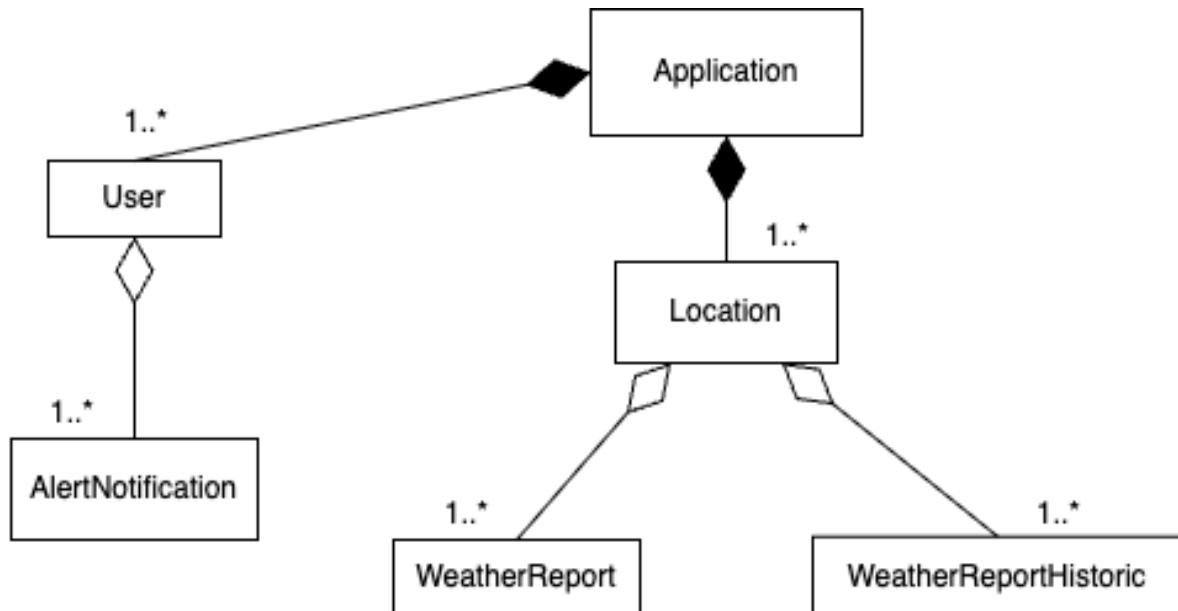
Se incluye la presentación de los siguientes elementos:

- **Modelo de Dominio:** Representando la estructura conceptual del negocio y los elementos clave del sistema.
- **Diagramas de Casos de Uso:** Ilustrando las funcionalidades del sistema desde la perspectiva de sus usuarios.
- **Especificaciones de Casos de Uso:** Describiendo detalladamente cada una de las interacciones y flujos del sistema.
- **Diagrama de Contexto:** Delimitando el alcance del sistema y sus interacciones con entidades externas.



5.1 MODELO DE DOMINIO.

El modelo de dominio define las entidades principales del sistema y sus relaciones. Se ha identificado una estructura centralizada en torno a la entidad Application, que gestiona usuarios y datos meteorológicos.

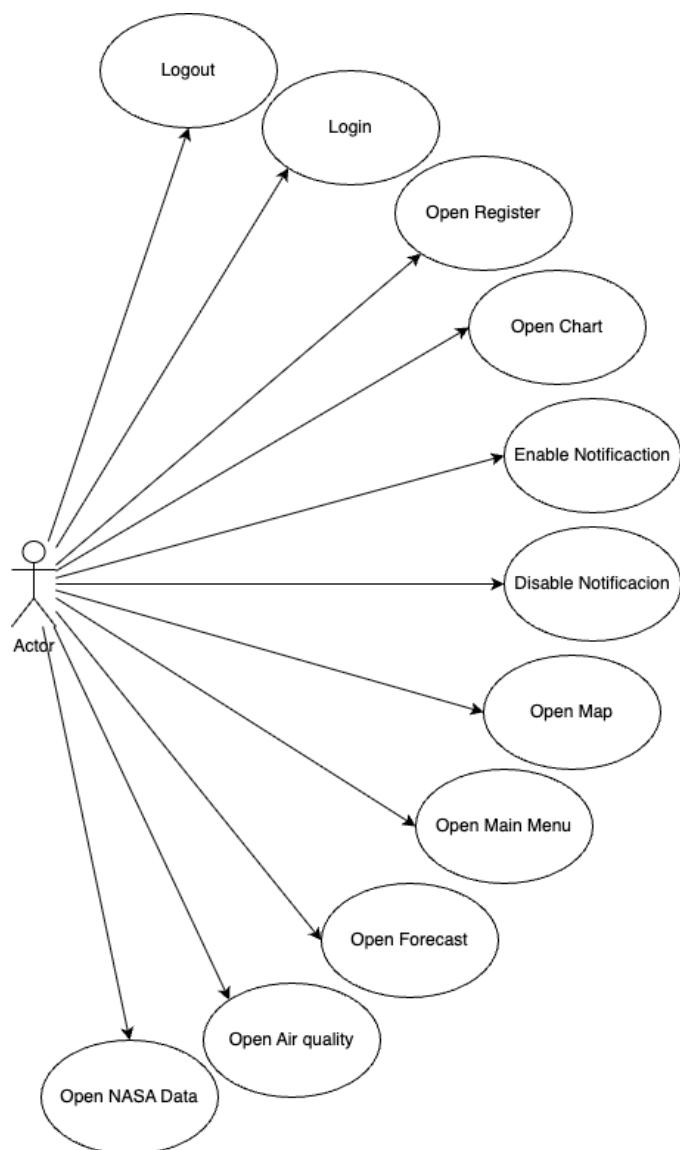




5.2 CASOS DE USO

5.2.1 DIAGRAMA DE CASOS DE USO

El siguiente diagrama ilustra los principales casos de uso del sistema, donde el usuario puede interactuar con diversas funcionalidades como autenticarse, visualizar gráficos y mapas, gestionar notificaciones y cerrar sesión.



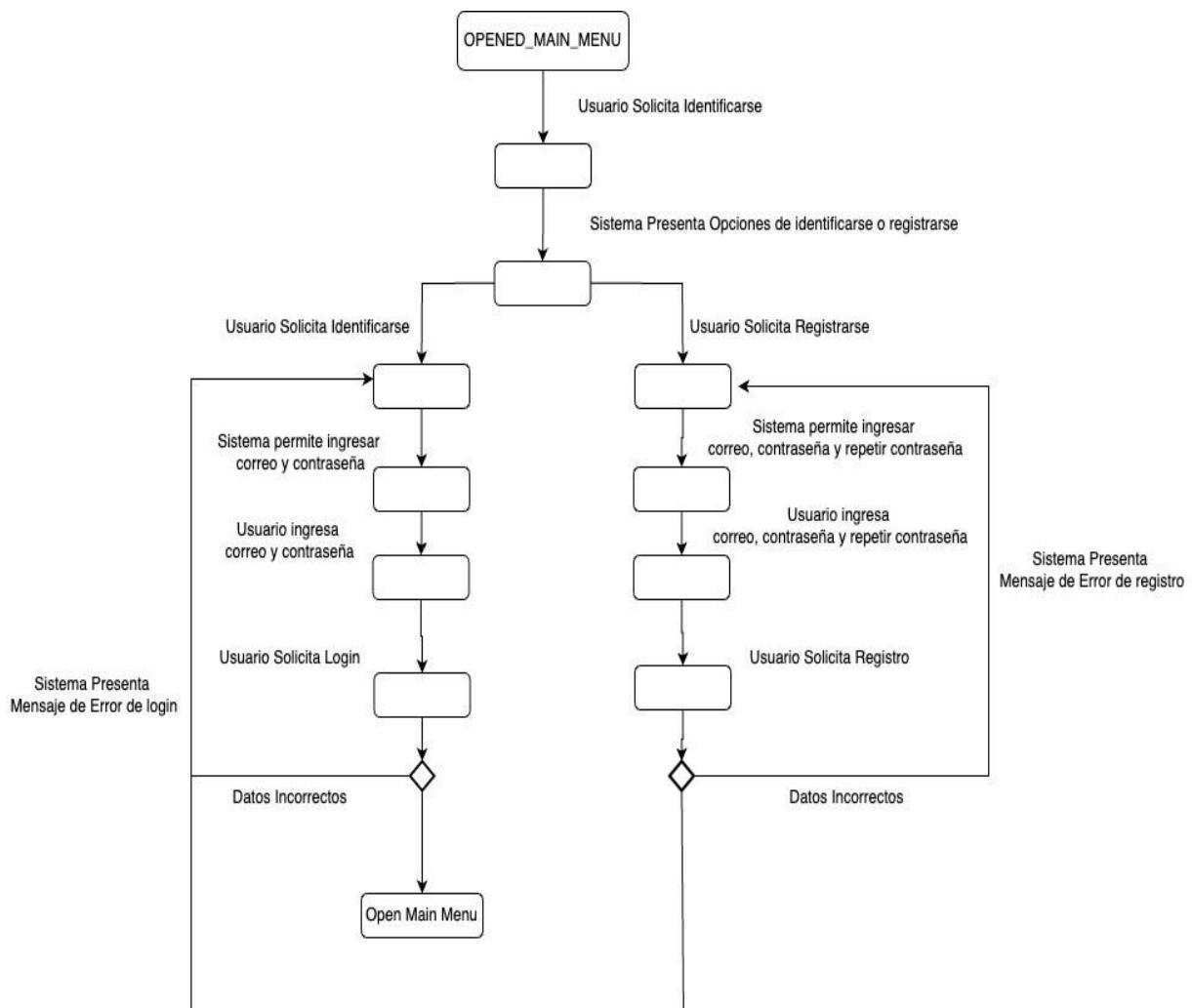


5.2.2 ESPECIFICACIÓN DE CASOS DE USO.

A continuación, se especifican los siguientes casos de uso:

5.2.2.1 CASO DE USO LOGIN.

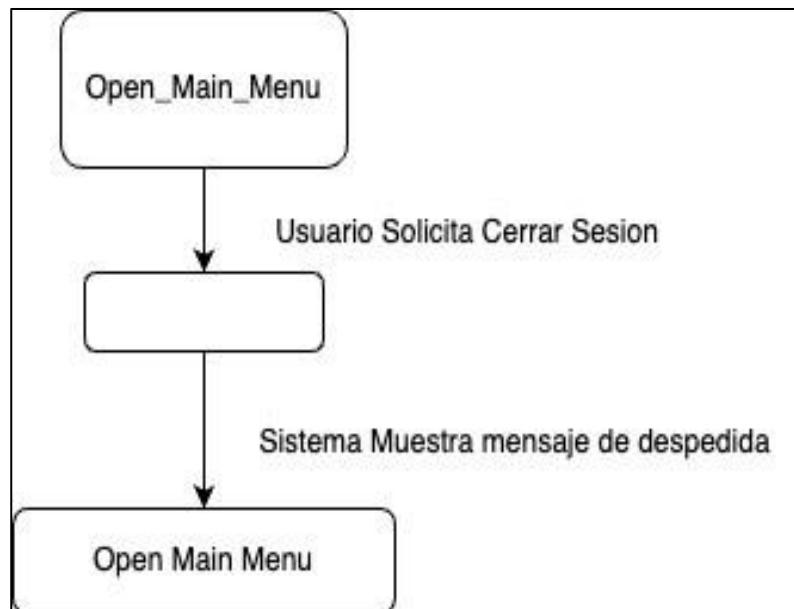
- Actor: Usuario
- Descripción: Permite al usuario autenticarse en el sistema.
- Flujo principal:
 1. El usuario solicita identificarse.
 2. El sistema presenta opciones de login o registro.
 3. El usuario ingresa credencial.
 4. El sistema verifica las credenciales.
 5. Si son correctas, el usuario accede al menú principal; si no, el sistema muestra un mensaje de error.





5.2.2.2 CASO DE USO LOGOUT.

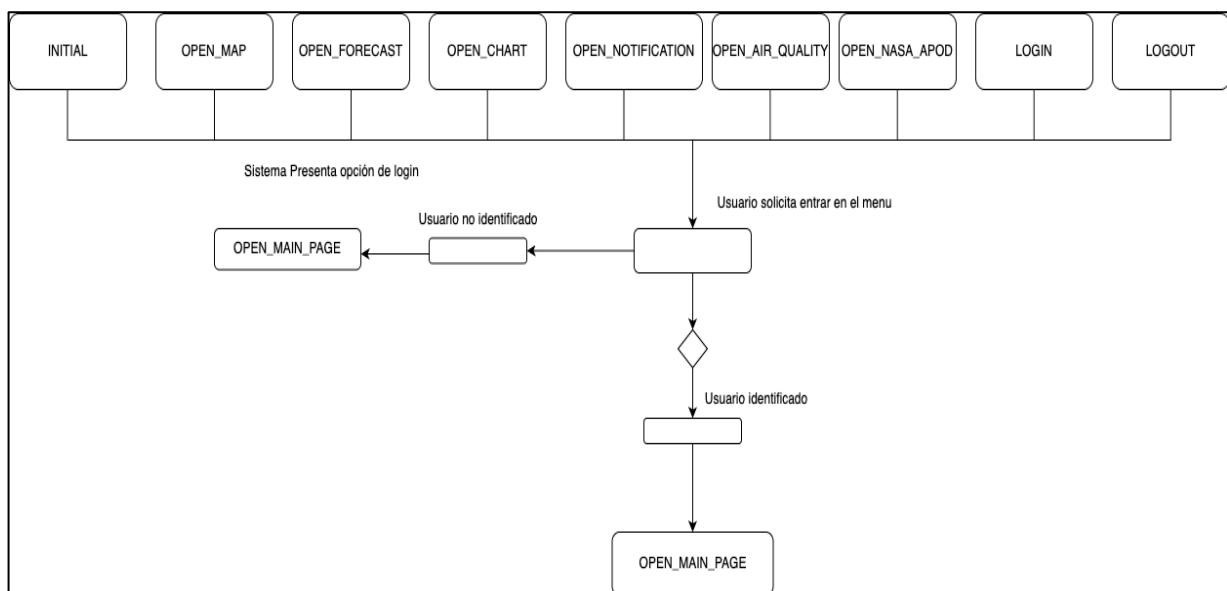
- Actor: Usuario
- Descripción: Permite al usuario cerrar sesión.
- Flujo principal:
 1. El usuario solicita cerrar sesión desde el menú principal.
 2. El sistema muestra un mensaje de confirmación.
 3. El usuario es redirigido a la pantalla inicial.





5.2.2.3 CASO DE USO MENÚ PRINCIPAL.

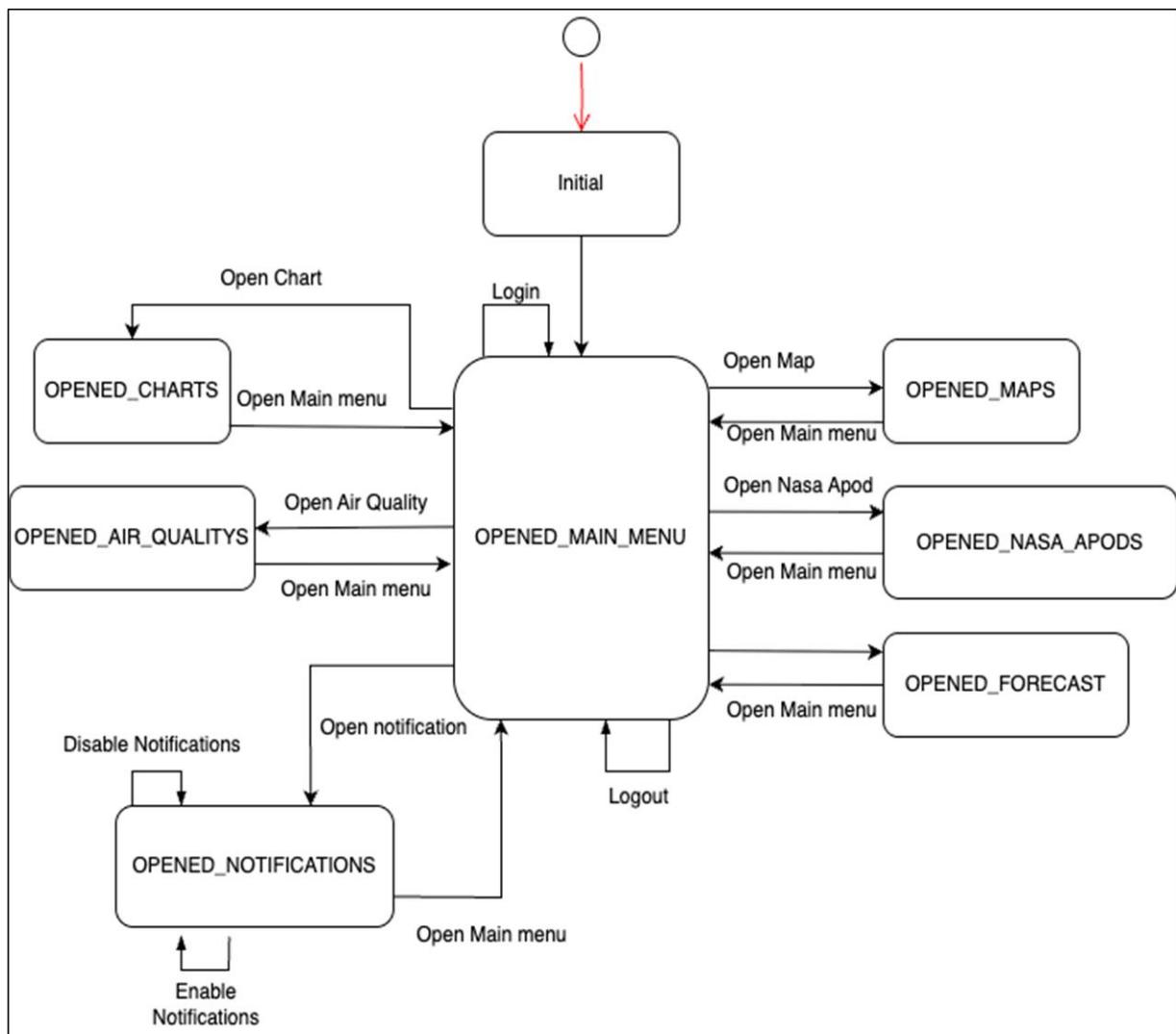
- **Actor:** Usuario
- **Descripción:** Permite al usuario acceder al menú principal desde donde puede navegar por la aplicación.
- **Flujo principal:**
 1. El usuario solicita ingresar al menú.
 2. El sistema verifica si el usuario está autenticado.
 3. Si el usuario está autenticado, se muestra la página principal; de lo contrario, se redirige a la pantalla de login.





5.3 DIAGRAMA DE CONTEXTO.

El diagrama de contexto muestra la relación entre el sistema y los actores externos. En este caso, el usuario es el único actor, interactuando con la aplicación a través de diversas funcionalidades.





6. IMPLEMENTACIÓN

6.1 DESARROLLO DEL BACKEND CON AUTENTICACIÓN Y ALERTAS METEOROLÓGICAS.

El backend de WeatherYr está desarrollado en Spring Boot con un enfoque reactivo, lo que permite manejar de manera asíncrona la obtención de datos meteorológicos desde las APIs externas, la integración con Firebase Realtime Database y la gestión de notificaciones mediante Firebase Cloud Messaging (FCM). Además, incluye un sistema de autenticación con Firebase Authentication y alertas meteorológicas automatizadas para los usuarios.

6.1.2 ARQUITECTURA DEL BACKEND.

El backend sigue una arquitectura de microservicios, donde:

1. La aplicación Android (cliente) envía peticiones REST para obtener datos meteorológicos y manejar notificaciones.
2. El backend (servidor) obtiene, procesa y almacena información meteorológica en Firebase.
3. Firebase Realtime Database almacena los datos de clima y preferencias de los usuarios.
4. Firebase Cloud Messaging (FCM) envía notificaciones personalizadas.

6.1.2.1 TECNOLOGÍAS UTILIZADAS.

- Spring Boot: Framework principal para la gestión de servicios REST.
- Spring WebFlux: Permite un procesamiento reactivo eficiente.
- Firebase Realtime Database: Almacena datos climáticos y preferencias de usuarios.
- Firebase Authentication: Gestiona el acceso seguro de los usuarios.
- Firebase Cloud Messaging (FCM): Maneja notificaciones push personalizadas.
- WebClient: Se usa para consumir la API de YR.no de manera asíncrona.
- Cloud Translation: Se usa para traducir la información obtenida de la API de NASA – Astronomy Picture of the Day (APOD).



6.1.3 IMPLEMENTACIÓN DE LA AUTENTICACIÓN.

Se maneja mediante Firebase Authentication, lo que permite:

1. Registro de nuevos usuarios con email y contraseña.
2. Inicio de sesión con validación de credenciales.
3. Generación y validación de tokens JWT para autenticación segura.
4. Gestión del token de Firebase Cloud Messaging (FCM) para notificaciones personalizadas.

ENDPOINTS DE AUTENTICACIÓN

Método	Endpoint	Descripción
POST	/auth/register	Registra un nuevo usuario en Firebase.
POST	/auth/login	Autentica a un usuario y devuelve un token.
POST	/auth/update-fcm-token	Guarda o actualiza el token de FCM del usuario

6.1.4 FUNCIONALIDADES DEL BACKEND.

1. Obtención de datos meteorológicos:

- o Consulta las Apis externas y obtiene la información para punto en específico se procesan y almacena los datos en Firebase.
- o Extrae datos como temperatura, humedad y velocidad del viento, calidad del aire, hora de salida y puesta del sol, duración del día.

2. Almacenamiento de datos en Firebase:

- o Utiliza Firebase Realtime Database para almacenar información meteorológica en tiempo real.
- o Reduce la dependencia de consultas constantes a las Apis Externas.

3. Gestión de notificaciones y alertas meteorológicas:

- o Verifica si el usuario tiene activadas las notificaciones en Firebase.
- o Genera alertas meteorológicas y las envía mediante Firebase Cloud Messaging (FCM).



ENDPOINTS DE INFORMACION METEOROLOGICO

Método	Endpoint	Descripción
GET	/weather/wind-map/last	Obtiene el último punto del mapa de viento
GET	/weather/location	Obtiene las coordenadas geográficas para una ubicación.
GET	/weather/hourly	Obtiene pronósticos últimas 48 horas
GET	/weather/historical	Obtiene registros históricos de datos meteorológicos para una ubicación.
GET	/weather/full-report	Obtiene reporte meteorológico completo para una ubicación específica

6.1.5 FUNCIONALIDADES DEL BACKEND.

Cómo funcionan las alertas meteorológicas:

1. Se verifica si el usuario tiene activadas las notificaciones en Firebase.
2. Se consulta el último dato de clima instantáneo almacenado.
3. Se generan alertas meteorológicas basadas en las condiciones actuales.
4. Si se detecta un evento crítico (ej. tormenta fuerte, riesgo de granizo), se envía una notificación push al usuario.

ENDPOINT PARA GENERAR ALERTAS METEOROLÓGICAS

Método	Endpoint	Descripción
GET	/Api/alerts/{userId}	Genera alertas basadas en el último dato de clima y envía notificación si es necesario.

Los usuarios solo recibirán alertas si tienen activadas las notificaciones, se evalúan eventos climáticos en tiempo real.



6.2 ARQUITECTURA DEL FRONTEND.

El cliente Android está desarrollado bajo arquitectura MVVM (Model-View-ViewModel):

1. Capa de Presentación

- Maneja la interfaz de usuario y la navegación.
- Utiliza Fragments para organizar las vistas de clima, historial y configuración de notificaciones.
- Implementa gráficos descriptivos para representar los datos meteorológicos.

2. Capa de Lógica de Negocio.

- Gestiona la obtención y procesamiento de datos antes de enviarlos a la UI.
- Maneja la comunicación con el backend y Firebase.
- Permite la actualización automática de los datos sin afectar el rendimiento de la aplicación.

6.2.1 TECNOLOGÍAS UTILIZADAS

- **Android Studio:** Plataforma de desarrollo principal.
- **Firebase Authentication:** Autenticación segura de usuarios.
- **Firebase Cloud Messaging (FCM):** Envío de notificaciones personalizadas.
- **Retrofit:** Consumo de APIs REST del backend.
- **MPAndroidChart:** Visualización de datos meteorológicos en gráficos.

6.2.2 FUNCIONALIDADES DEL FRONTEND

1. Autenticación de Usuarios

- Permite a los usuarios registrarse e iniciar sesión de manera segura.
- Maneja sesiones autenticadas para acceder al servicio.

2. Consulta de Datos Meteorológicos

- Muestra el clima actual en tiempo real.
- Permite visualizar pronósticos del tiempo, mapas de viento, temperatura y presión.
- Facilita el acceso rápido a información relevante sin necesidad de múltiples consultas a la API externa.



3. Visualización de Datos en la UI

- Presenta gráficos con tendencias meteorológicas.
- Organiza la información en fragmentos de fácil navegación.
- Incluye gráficos para facilitar la experiencia del usuario.

4. Gestión de Notificaciones Push

- Las notificaciones solo se envían si el usuario las tiene activadas.
- La aplicación permite activar o desactivar las notificaciones en la configuración.
- Si están activadas, el usuario recibe alertas meteorológicas en tiempo real.
- Se integran con Firebase Cloud Messaging (FCM) para asegurar su entrega eficiente.

5. Consumo de Apis REST

- Se comunica con el backend para obtener datos meteorológicos actualizados.
- Maneja la autenticación en cada solicitud para garantizar la seguridad de los datos.
- Sincroniza los datos almacenados en Firebase con los obtenidos de la Api externa.

6.3 INTEGRACIÓN DE FUNCIONALIDADES CLAVE.

En WeatherYr, la integración entre el backend y frontend permite una experiencia fluida para los usuarios, combinando datos meteorológicos en gráficos, mapas y un pronóstico de las próximas horas.

Esta sección explica cómo se conectan las principales funcionalidades dentro de la aplicación.

6.3.1 GRÁFICOS METEOROLÓGICOS.

La aplicación permite visualizar la evolución de temperatura, velocidad del viento, precipitaciones y presión atmosférica en gráficos descriptivos.

Cómo se integran backend y frontend:

1. Backend

- Obtiene datos desde el servidor los cuales están almacenados en firebase y los almacena en Firebase Realtime Database.
- Expone un endpoint REST (/weather/historical) que devuelve predicciones meteorológicas históricas.



2. Frontend

- Consulta los datos mediante Retrofit y los transforma en gráficos con MPAndroidChart.
- Muestra gráficos de líneas, barras y pastel en GraficFragment.java. GraficFragment

6.3.2 MAPAS.

Los usuarios pueden ver mapas descriptivos de viento, temperatura y presión para analizar patrones meteorológicos en tiempo real.

Cómo se integran backend y frontend:

1. Backend

- Obtiene datos del último mapa de viento disponible y los almacena en Firebase.
- Expone un endpoint REST (/weather/wind-map/last) con la información más reciente.

2. Frontend

- Muestra los datos en MapFragment.java, donde se carga un WebView con un mapa interactivo. MapFragment
- El archivo windy_map.html utiliza la API de Windy en su versión gratuita, la cual proporciona datos de viento, temperatura y presión atmosférica en tiempo real. windy_map

6.3.3 PRONÓSTICO DE 48 HORAS.

La aplicación puede proporcionar un pronóstico meteorológico con una anticipación de 48 horas.

Cómo se integran backend y frontend:

1. Backend

- Almacena las consultas en Firebase Realtime Database.
- Expone un endpoint REST (/weather/full-report) que permite obtener un reporte meteorológico.

2. Frontend

- Consulta los datos mediante Retrofit y los muestra en ForecastFragment
- Organiza la información en tarjetas, facilitando su interpretación.



7. PRUEBAS Y RESULTADOS.

Para garantizar el correcto funcionamiento de WeatherYr, se realizaron pruebas en backend y frontend, abarcando validaciones de seguridad, integración y experiencia de usuario. Se utilizaron herramientas como Swagger, Postman, JUnit y pruebas end-to-end en Android Studio con el backend levantado y conectado a Firebase.

7.1 PRUEBAS DEL BACKEND.

El backend fue sometido a diferentes niveles de prueba para garantizar su estabilidad y correcto funcionamiento.



7.1.1 MÉTODOS DE PRUEBA.

1.- Cliente Swagger.

- Se agregaron tags y descripciones en cada endpoint para estructurar la documentación.
- Se generó documentación automática de los endpoints con Swagger para facilitar pruebas manuales.

A continuación, se adjunta una captura del cliente swagger completo. Además, se pueden encontrar en los anexos, al final de este documento, capturas en detalle del cliente.

Weather App API 1.0 OAS3

/v3/api-docs

API para autenticación y datos meteorológicos

Servers Authorize

Autenticación Endpoints para autenticación y registro de usuarios en Firebase

POST /auth/update-fcm-token	Actualizar el token de FCM del usuario	
POST /auth/register	Registrar un nuevo usuario	
POST /auth/login	Autenticar usuario	

weather-controller

POST /weather/historical/save	Guarda manualmente un registro histórico de datos meteorológicos.	
GET /weather/wind-map/last	Obtener el último punto del mapa de viento	
GET /weather/location	Obtiene las coordenadas geográficas para una ciudad y país dados.	
GET /weather/hourly	Obtener pronósticos por hora (últimas 48 horas)	
GET /weather/historical	Obtiene registros históricos de datos meteorológicos para una ubicación.	
GET /weather/full-report	Obtiene un reporte meteorológico completo (actuales, pronóstico por hora, mapa de viento y calidad del aire) para una ubicación específica.	

alert-controller

GET /api/alerts/{userId}	Generar alertas meteorológicas basadas en el último dato de clima	
---------------------------------	---	--



2. Pruebas con Postman

- Se verificó la correcta autenticación con JWT en los endpoints protegidos.
- Se probaron solicitudes REST para confirmar la correcta obtención de datos desde Firebase y las APIs externas.

A continuación, se adjunta captura del REST de alertas meteorológicas.

The screenshot shows a Postman interface with the following details:

- Request URL:** http://localhost:8080/api/alerts/2TuQOQJ3ZmXEFgw9wvpTxOwm0nZ2
- Method:** GET
- Headers:** (7)
 - Accept: gzip, deflate, br
 - Connection: keep-alive
 - Authorization: Bearer eyJhbGciOiJSUzI1NiIsImtpZCI6ImE0MzRmMzFkN2Y3NV
- Body:** (Pretty) "1": "¡Alerta! Baja humedad relativa, riesgo de sequedad."
- Response Status:** 200 OK (1821 ms, 501 B)

3. Pruebas Automatizadas con Junit e integración con SonarCloud

- Se desarrollaron pruebas unitarias e integración con JUnit y Mockito.
- Se validaron controladores, servicios y configuraciones de seguridad.
- Se alcanzó una cobertura del 80% en sonarCloud.



*Imagen obtenida desde la interfaz web de SonarCloud



7.1.2 LISTA DE ENDPOINTS TESTEADOS.

Método	Endpoint	Descripción
POST	/auth/register	Registra un usuario en Firebase.
POST	/auth/login	Autentica al usuario y genera un token JWT.
POST	/auth/update-fcmtoken	Guarda el token de notificaciones push en Firebase.
POST	/weather/historical/save	Guarda manualmente el registro histórico de datos meteorológicos
GET	/weather/hourly	Retorna el pronóstico por hora.
GET	/weather/windmap/last	Obtiene el último mapa de viento disponible.
GET	/weather/location	Obtiene las coordenadas geográficas para una ciudad y país dados.
GET	/weather/historical	Obtiene registros históricos de datos meteorológicos para una ubicación.
GET	/weather/full-report	Obtiene un reporte meteorológico completo para una ubicación específica.
GET	/Api/alerts/{userId}	Genera alertas meteorológicas si el usuario tiene notificaciones activadas.



7.2 PRUEBAS DEL FRONTEND.

Las pruebas del frontend se realizaron en Android Studio, utilizando emuladores. Se llevaron a cabo pruebas end-to-end, con el backend levantado y conectado a Firebase.

Áreas probadas:

1. Flujo Completo de la Aplicación (End-to-End)

- Se levantó el backend y se ejecutó la aplicación en un emulador.
- Se validó la comunicación con Firebase y la correcta obtención de datos meteorológicos.

2. Consumo de Apis REST

- Se probó la autenticación con JWT para garantizar seguridad en las solicitudes.
- Se verificó la obtención de datos climáticos y su actualización en tiempo real.

3. Visualización de Datos Meteorológicos

- Se validó la correcta actualización de los gráficos en la pantalla de pronóstico de 48 horas.
- Se probó la representación de los mapas de viento usando la API de Windy.

4. Notificaciones Push

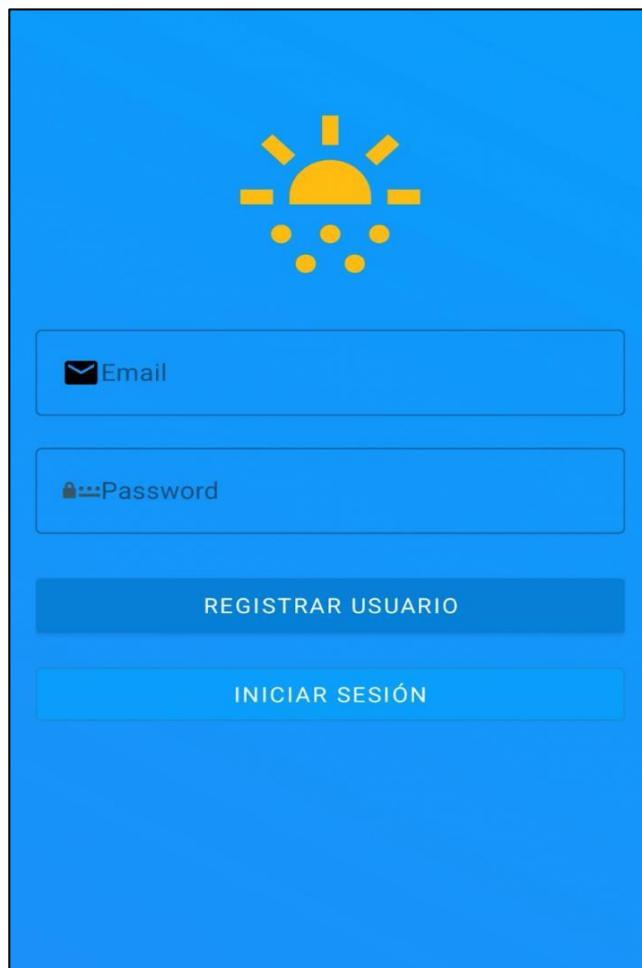
- Se verificó que solo se enviarán alertas si el usuario tenía notificaciones activadas.
- Se validó la funcionalidad de activar/desactivar alertas meteorológicas en la configuración.

7.2.1 RESULTADOS .

Esta sección presenta una serie de capturas de pantalla que ilustran los resultados y funcionalidad de la aplicación posterior a la ejecución de las pruebas.

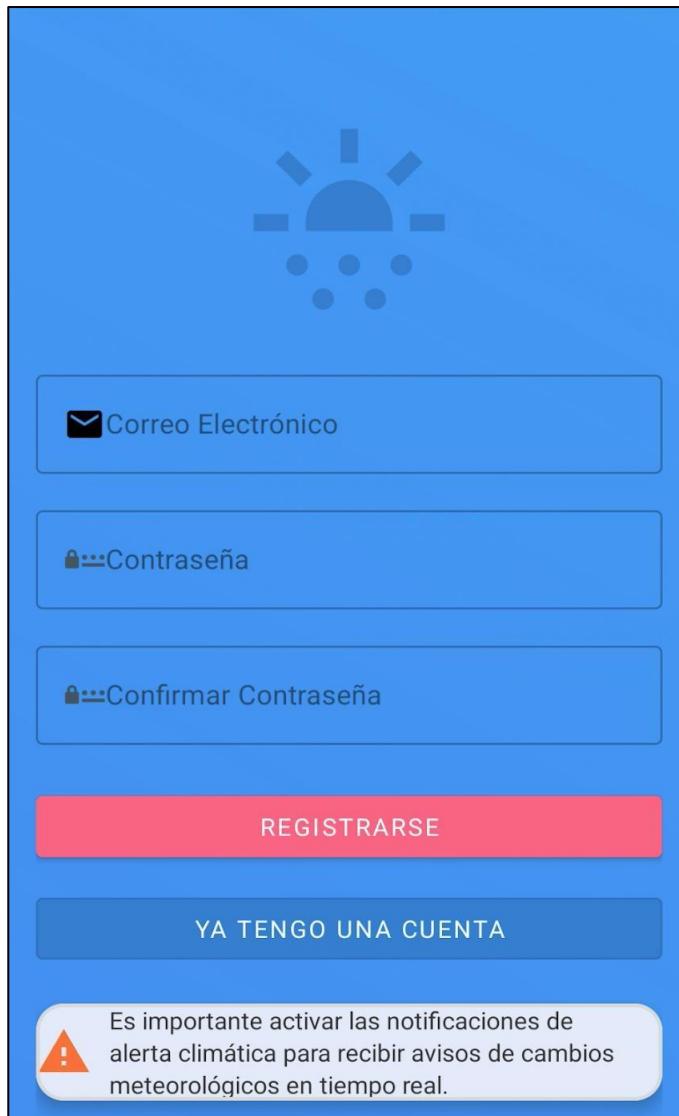
Captura 1: Pantalla de Inicio de Sesión

La captura de pantalla que se presenta a continuación muestra la pantalla de inicio de sesión y registro de la aplicación. Entre sus funciones están el ingreso de credenciales, donde hay un campo para ingresar el correo electrónico (Email), con un ícono de sobre, y un campo para ingresar la contraseña o password con un ícono de candado. En caso de no estar registrado previamente, presenta un botón de "registrar usuario" para crear una nueva cuenta en la aplicación. Finalmente presenta un botón "iniciar sesión" para acceder a la aplicación con las credenciales.



Captura 2: Pantalla de Registro de usuario

Esta captura muestra las funciones de Ingreso de datos para registro. Presenta un campo para introducir un email, acompañado de un ícono de sobre, un campo para ingresar una clave segura, con un ícono de candado, y luego un campo para re-ingresar la contraseña y evitar errores. Finalmente presenta un Botón "REGISTRARSE" en color rosa que permite completar el registro tras ingresar los datos correctamente, y de lo contrario un Botón "YA TENGO UNA CUENTA" en color azul que dirige al usuario a la pantalla de inicio de sesión si ya está registrado.





Captura 3: Menú Principal y Visualización de Datos Climáticos

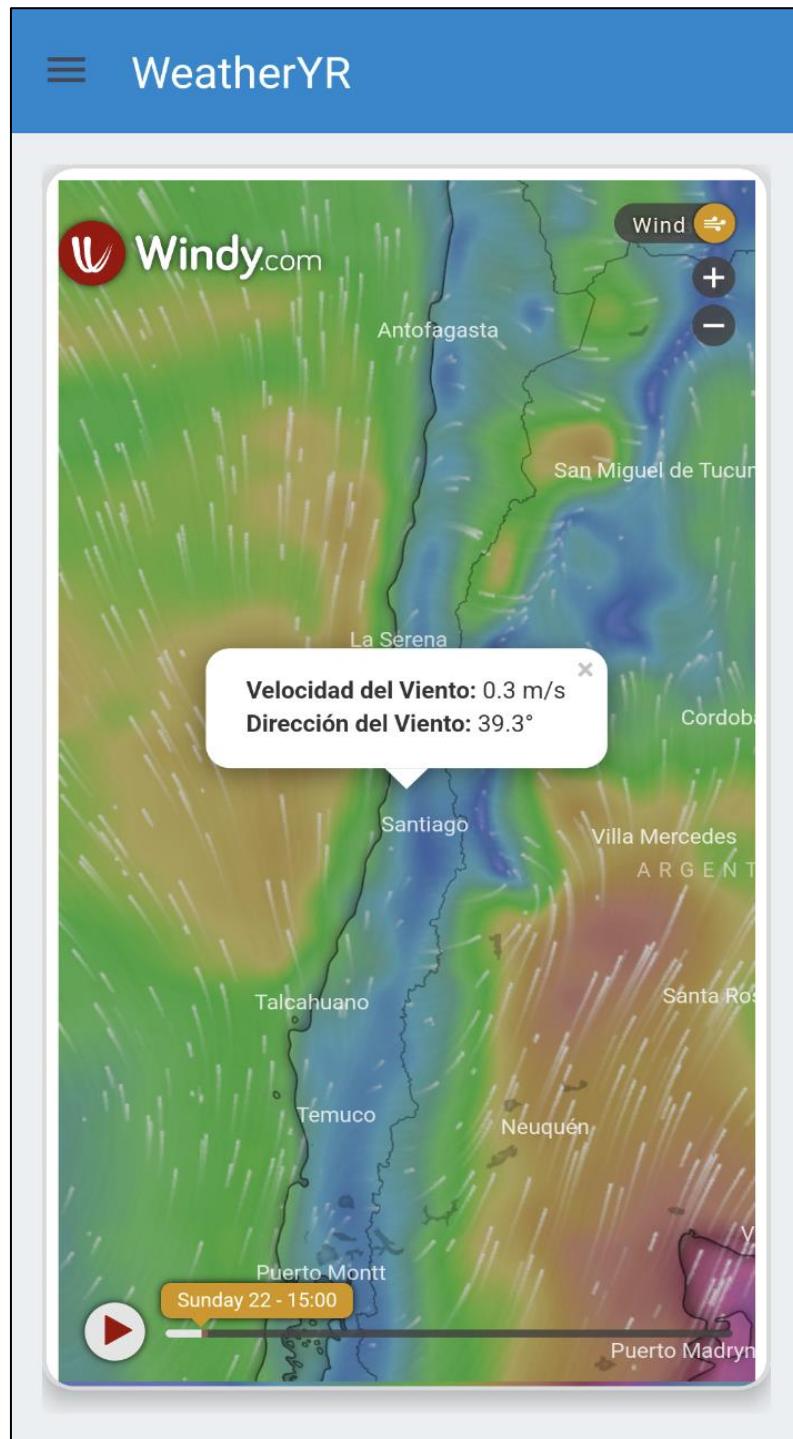
En esta captura se presenta la información obtenida por el servicio REST /full-report, donde los datos son obtenidos y presentados en la siguiente interfaz que está diseñada con íconos de Fontawesome y card de material en la cual se presentan distintos datos correspondientes a la comuna de Hijuelas, Chile.





Captura 4: Visualización de Mapas

En la siguiente captura se presenta el mapa de viento proporcionado por la Apis Windy.com, no obstante, Windy nos proporciona 2 opciones más de mapas en la versión gratuita utilizada, éstos corresponden a mapas de presión y temperatura.





Captura 5: Visualización de Pronósticos.

En esta captura se presenta una serie de card que nos presentan un pronóstico de las próximas horas, donde puede apreciar la hora exacta, temperatura, presión y velocidad del viento.

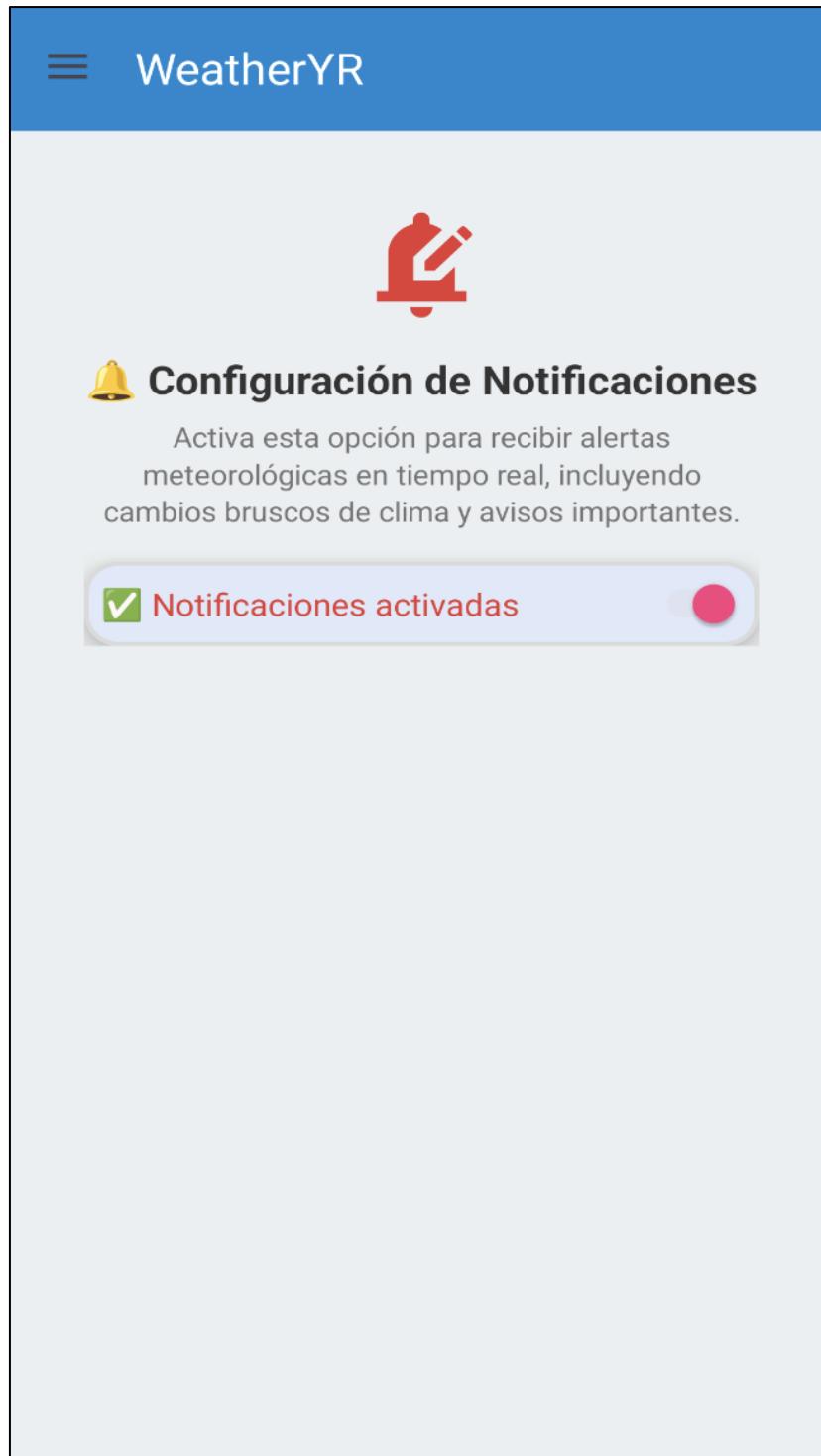
The screenshot shows a weather forecast for the next eight hours. Each hour is represented by a card with the following information:

Hora	Icono	Condición	Temperatura	Velocidad del Viento	Presión
19:00	🌙	Despejado Noche	14.9°C	2.6 m/s	0.0 mm
20:00	🌙	Despejado Noche	14.8°C	2.7 m/s	0.0 mm
21:00	🌙	Despejado Noche	14.1°C	2.0 m/s	0.0 mm
22:00	🌙	Despejado Noche	12.2°C	1.0 m/s	0.0 mm
23:00	🌙	Despejado Noche	11.4°C	1.5 m/s	0.0 mm
00:00	🌙	Despejado Noche	10.2°C	1.8 m/s	0.0 mm
01:00	🌙	Despejado Noche	9.2°C	1.9 m/s	0.0 mm
02:00	🌙	Despejado Noche	8.4°C	2.0 m/s	0.0 mm



Captura 6: Configuración de Notificaciones

En esta captura se presenta un switch que nos permite activar o desactivar las notificaciones de alertas climáticas.





Captura 7: Calidad del aire

Esta captura presenta un informe de calidad del aire y un indicador con una pequeña descripción y además se presenta un gráfico con la concentración de contaminantes.





Captura 8: Dato astronómico diario.

Esta captura de pantalla muestra la imagen proporcionada por la API NASA APOD (Astronomy Picture of the Day), la cual corresponde a la imagen astronómica del día publicada diariamente por la NASA."

The screenshot shows the WeatherYr app interface. At the top, there is a blue header bar with the app's logo and name. Below it, a white card displays the title "Imagen Astronómica del Día (NASA APOD)" in bold black text. Underneath the title, the specific image is shown with the caption "A Berry Bowl of Martian Spherules". The date "22 de junio de 2025" is displayed next to the image, along with the credit "© NASA". The image itself is a photograph of a reddish-brown, textured surface on Mars, featuring numerous small, dark, spherical objects (Martian spherules) scattered across it. A small text at the bottom left of the image reads "... PT, Curiosity Rover". Below the image, there is a detailed explanatory text in Spanish: "¿Cómo se crearon estas inusuales esférulas marcianas? Miles de inusuales esférulas grises hechas de hierro y roca, llamadas arándanos, fueron encontradas incrustadas en rocas circundantes cerca del lugar de aterrizaje del rover Opportunity en Marte en 2004. Para ayudar a investigar su origen, Opportunity encontró una superficie llamada Berry Bowl con una hendidura rica en orbes marcianos. El Berry Bowl se muestra aquí".



Captura 9: Dato astronómico diario.

En esta captura se presentan datos astronómicos correspondientes a Hijuela, como la hora de salida y puesta del sol, la duración del día y el mediodía solar.

The screenshot displays the WeatherYr app interface with the following information:

- Header:** WeatherYR
- Section: Datos astronómicos**
 - Date: domingo 22 de junio
 - Location: *Hijuelas, Provincia de Quillota, Región de Valparaíso, Chile*
- Sunrise and Sunset:**
 - Salida del sol (HL): **08:45 a. m.**
 - Puesta del sol (HL): **06:47 p. m.**
- Detalles Solares:**
 - Mediodía Solar: **01:46 p. m.**
 - Duración del Día: **10:02:04**
- Tiempos de Crepúsculo:**
 - Crepúsculo Civil:**
 - Inicio: **08:19 a. m.**
 - Fin: **07:13 p. m.**
 - Crepúsculo Náutico:**
 - Inicio: **07:48 a. m.**



8. CONCLUSIONES Y MEJORAS.

Esta sección presenta una reflexión final sobre el desarrollo de WeatherYr, destacando los logros alcanzados y las posibles mejoras futuras para optimizar la aplicación.

8.1 CONCLUSIONES.

La aplicación ha sido optimizada para garantizar una experiencia fluida y confiable para los usuarios. Se corrigieron errores clave, como el envío incorrecto del token de autenticación en algunas solicitudes, asegurando que Retrofit lo incluya en cada petición. Además, se solucionó un problema donde las alertas meteorológicas se enviaban incluso cuando el usuario tenía desactivadas las notificaciones.

Las pruebas end-to-end confirmaron que la integración entre frontend, backend y Firebase funciona correctamente. El backend pasó todas las pruebas de integración con Firebase y la API de YR.no, mientras que el frontend fue validado en dispositivos emulados, garantizando un rendimiento estable.

A continuación, se mencionan todas las funciones ejecutadas correctamente por la aplicación, cumpliendo así con los objetivos planteados al inicio del proyecto.

Integración completa entre backend y frontend

- Se estableció una comunicación fluida entre Spring Boot, Firebase y Android Studio.
- Se logró obtener un reporte completo según la búsqueda ingresada desde el frontend, gracias a la integración con APIs externas. Las respuestas de estas APIs se procesan y almacenan en Firebase para su rápida recuperación.

Autenticación y Seguridad

- Se implementó Firebase Authentication para gestionar usuarios de manera segura.
- Se protegieron endpoints mediante JWT, asegurando el acceso solo a usuarios autenticados.

Visualización de Datos Meteorológicos

- Se integraron gráficos para mostrar tendencias meteorológicas de 48 horas.
- Se implementaron mapas de viento con Windy API (versión gratuita) para representar condiciones climáticas

Notificaciones Push

- Se habilitaron notificaciones meteorológicas, solo si el usuario tiene alertas activadas.
- Se permitió a los usuarios activar o desactivar las alertas según su preferencia.



8.2 MEJORAS Y FUTURAS AMPLIACIONES.

Si bien la aplicación cumple con sus objetivos principales, existen diversas mejoras y ampliaciones que podrían implementarse en el futuro para optimizar su funcionalidad y experiencia de usuario.

1. Ampliación de la Cobertura Meteorológica

- Actualmente, la aplicación está optimizada para la ciudad de Madrid.
 - Se podría agregar la opción de seleccionar múltiples ubicaciones y guardar preferencias.
2. Optimización del Manejo de Datos
- Mejorar la eficiencia del almacenamiento en Firebase, optimizando las consultas a datos históricos.
 - Implementar un mecanismo de caché en el cliente para reducir la dependencia del backend.

2. Mejoras en la Visualización de Datos

- Agregar nuevas vistas de gráficos más detalladas, como comparaciones semanales o mensuales.
- Incluir más capas en el mapa de Windy API, como precipitación, humedad o temperatura a diferentes altitudes.

3. Integración con Widgets y Wearables

- Desarrollar un widget para que los usuarios puedan consultar el clima sin abrir la aplicación.
- Explorar la compatibilidad con smartwatches para recibir alertas y visualizar datos meteorológicos.

4. Mejora de la Experiencia de Usuario

- Personalización de la interfaz según las preferencias del usuario.
- Inclusión de un modo oscuro para mejorar la accesibilidad y reducir consumo de energía.

5. Ampliación de la Funcionalidad de Notificaciones

- Actualmente, las alertas meteorológicas se envían de forma general si el usuario tiene activadas las notificaciones.



- Se podría agregar un sistema de personalización, donde los usuarios puedan seleccionar qué tipo de alertas desean recibir, ésto permitiría una experiencia más personalizada y reduciría notificaciones irrelevantes para el usuario. Algunas opciones de alertas que se podrían personalizar podrían ser las siguientes:
 - Tormentas
 - Lluvias intensas
 - Altas temperaturas
 - Bajas temperaturas
 - Fuertes vientos

Para finalizar, el desarrollo de WeatherYr ha sido una oportunidad para aplicar diversas tecnologías en un entorno real. Se logró construir una aplicación estable y funcional, con una arquitectura bien definida y una integración eficiente entre backend, frontend y Firebase.

Sin embargo, la aplicación tiene un gran potencial de crecimiento, y futuras mejoras que permitirán ampliar su alcance y funcionalidades.



9. ANEXOS.

Anexo 1: Detalle cliente swagger. Modelo de Request, Response y Modelos.

The screenshot shows the Swagger UI interface for the Weather App API. At the top, it displays the date and time (2/2/25, 16:37) and the version (1.0 OAS3). The URL /v3/api-docs is highlighted in a green box. To the right, there is an 'Explore' button. Below this, the title 'Weather App API' is shown with its version 1.0 and specification OAS3. A sub-section '/v3/api-docs' is visible. The main content area is titled 'Autenticación' (Authentication), which describes endpoints for authentication and user registration in Firebase. It includes a 'Try it out' button. Below this, there is a detailed view of a POST endpoint for updating FCM tokens. The endpoint is '/auth/update-fcm-token' with the description 'Actualizar el token de FCM del usuario'. It requires an Authorization header (string type). The request body is specified as application/json. An example value is provided, showing a JSON object with 'additionalProp1' and 'additionalProp2' fields. The bottom of the interface shows the URL 'localhost:8080/webjars/swagger-ui/index.html#/alert-controller/getWeatherAlerts' and a page number '1/12'.



2/25, 16:37

Swagger UI

```
{
    "additionalProp3": "string"
}
```

Responses

Code	Description	Links
200	Token de FCM actualizado correctamente	No links
	Media type application/json Controls Accept header.	
	Example Value	
	{ "message": "Token de FCM actualizado con éxito" }	
400	Error en la actualización	No links
	Media type application/json	
	Example Value	
	{ "error": "Token de FCM es requerido" }	

POST /auth/register Registrar un nuevo usuario ^

Registra un usuario en Firebase con email y contraseña

Parameters **Try it out**

No parameters

Request body required

localhost:8080/webjars/swagger-ui/index.html#/alert-controller/getWeatherAlerts 2/12



2/25, 16:37

Swagger UI

application/json

JSON con los datos del usuario

Example Value Schema

```
{ "email": "user@example.com", "password": "password123" }
```

Responses

Code	Description	Links
200	Usuario registrado exitosamente	No links
	Media type application/json Controls Accept header.	
	Example Value	
	{ "message": "Usuario registrado con UID: abc123" }	
400	Error en el registro	No links
	Media type application/json	
	Example Value	
	{ "error": "Email y contraseña son obligatorios" }	

POST /auth/login Autenticar usuario

localhost:8080/webjars/swagger-ui/index.html#/alert-controller/getWeatherAlerts

3/12



2/25, 16:37 Swagger UI

Inicia sesión en Firebase con email y contraseña y devuelve un token JWT

Parameters

No parameters

Request body required

application/json

JSON con el email y contraseña del usuario

Example Value Schema

```
{  
  "email": "string",  
  "password": "string"  
}
```

Responses

Code	Description	Links
200	Usuario autenticado exitosamente	<i>No links</i>
	Media type application/json Controls Accept header.	
	Example Value	
	{ "token": "Bearer abcdef123456" }	
400	Credenciales inválidas o error de autenticación	<i>No links</i>
	Media type application/json	
	Example Value	

localhost:8080/webjars/swagger-ui/index.html#/alert-controller/getWeatherAlerts

4/12



weather-controller

POST /weather/historical/save Guarda manualmente un registro histórico de datos meteorológicos.

Este endpoint es para guardar una entrada histórica específica. En un sistema real, esto podría ser automático.

Parameters

No parameters

Request body required

application/json

Objeto HistoricalWeatherEntry a guardar

Example Value | Schema

```
{
  "id": "NsAcX_Y6Z-yE9fc0e8c",
  "location": {
    "name": "string",
    "latitude": 0,
    "longitude": 0,
    "countryCode": "string"
  },
  "recordedAt": "2025-05-28T15:30:00.123456Z",
  "instantWeatherSnapshot": {
    "airTemperature": 0,
    "relativeHumidity": 0,
    "airPressureAtSeaLevel": 0,
    "windSpeed": 0,
    "cloudArealFraction": 0
  },
  "hourlyForecasts": [
    {
      "time": "string",
      "airTemperature": 0,
      "windSpeed": 0,
      "precipitationAmount": 0
    }
  ],
  "forecast": {
    "start": "2025-05-28T15:30:00.123456Z",
    "end": "2025-05-28T16:30:00.123456Z",
    "temperature": 0,
    "humidity": 0,
    "pressure": 0,
    "wind": 0
  }
}
```

Responses

Code	Description	Links
201	Registro histórico guardado exitosamente	No links



Code	Description	Links
201	Registro histórico guardado exitosamente	No links
400	Solicitud inválida (ej. datos faltantes)	No links
500	Error interno del servidor al guardar	No links



GET /weather/wind-map/last Obtener el último punto del mapa de viento

Devuelve solo el registro más reciente del mapa de viento almacenado en Firebase.

Parameters

No parameters

Responses

Code	Description	Links
200	Registro más reciente obtenido exitosamente	No links
204	No hay registros disponibles	No links

Media type application/json

Controls Accept header.

Example Value | Schema

```
{ "features": [ { "geometry": { "coordinates": [ -3.7838, 40.4168 ] }, "properties": { "windSpeed": 12, "windDirection": 270 } } ] }
```

Media type */*

Example Value | Schema

```
string
```



GET /weather/location Obtiene las coordenadas geográficas para una ciudad y país dados.

Usa un servicio de geocodificación para convertir un nombre de ciudad y país en latitud y longitud.

Parameters

Name Description

addressQuery * required string (query) Cadena de búsqueda de la dirección (ej. 'Santiago, Chile') Example : Santiago, Chile

Santiago, Chile

Responses

Code	Description	Links
200	Coordenadas obtenidas exitosamente	No links
400	Parámetros de entrada inválidos o ubicación no encontrada	No links

Media type application/json

Example Value Schema

```
{ "cityName": "Santiago", "latitude": -33.4489, "longitude": -70.6693, "countryName": "Chile" }
```



Code	Description	Links
200	Coordenadas obtenidas exitosamente	No links
	<p>Media type</p> <div style="border: 1px solid #ccc; padding: 2px; display: inline-block;">application/json</div> <p>Controls Accept header.</p> <p>Example Value Schema</p> <pre>{ "cityName": "Santiago", "latitude": -33.4489, "longitude": -70.669, "countryName": "Chile" }</pre>	
400	Parámetros de entrada inválidos o ubicación no encontrada	No links
	<p>Media type</p> <div style="border: 1px solid #ccc; padding: 2px; display: inline-block;">application/json</div> <p>Example Value Schema</p> <pre>{ "statusCode": 403, "message": "Acceso denegado", "details": "Usuario no tiene permisos" }</pre>	
500	Error interno del servidor	No links
	<p>Media type</p> <div style="border: 1px solid #ccc; padding: 2px; display: inline-block;">application/json</div> <p>Example Value Schema</p> <pre>{ "statusCode": 403, "message": "Acceso denegado", "details": "Usuario no tiene permisos" }</pre>	



GET /weather/hourly Obtener pronósticos por hora (últimas 48 horas)

Devuelve los registros de pronósticos horarios almacenados en Firebase dentro de las últimas 48 horas.

Parameters

No parameters

Responses

Code	Description	Links
200	Datos obtenidos exitosamente	No links
500	Error interno del servidor	No links

Media type

application/json

Controls Accept header.

Example Value | Schema

```
[  
  {  
    "time": "2025-01-31T10:00:00Z",  
    "temperature": 18.3,  
    "humidity": 75,  
    "windSpeed": 10  
  },  
  {  
    "time": "2025-01-31T11:00:00Z",  
    "temperature": 19.1,  
    "humidity": 73,  
    "windSpeed": 12  
  }]
```



GET /weather/historical Obtiene registros históricos de datos meteorológicos para una ubicación.

Recupera los registros históricos de datos meteorológicos para una latitud y longitud dadas.

Parameters

Try it out

Name	Description
latitude * required number(\$double) (query)	Latitud de la ubicación <i>Example : -33.4489</i> <input type="text" value="-33.4489"/>
longitude * required number(\$double) (query)	Longitud de la ubicación <i>Example : -70.6693</i> <input type="text" value="-70.6693"/>
limit integer(\$int32) (query)	Número máximo de registros a recuperar (por defecto: 10) <i>Default value : 10</i> <i>Example : 5</i> <input type="text" value="5"/>

Responses

Code	Description	Links
200	Registros históricos obtenidos exitosamente	No links

Media type

Controls Accept header.
Example Value : Schema



Code	Description	Links
200	Registros históricos obtenidos exitosamente	No links
	<p>Media type</p> <div style="border: 1px solid #ccc; padding: 2px; display: inline-block;">application/json</div> <p>Controls Accept header.</p> <p>Example Value Schema</p> <pre>{ "id": "-NsAcX_Y6Z-yE9fc0e8c", "location": { "name": "string", "latitude": 0, "longitude": 0, "countryCode": "string" }, "recordedAt": "2025-05-28T15:30:00.123456Z", "instantWeatherSnapshot": { "airTemperature": 0, "relativeHumidity": 0, "airPressureAtSeaLevel": 0, "windSpeed": 0, "cloudAreaFraction": 0 }, "hourlyForecasts": [{ "time": "string", "airTemperature": 0, "windSpeed": 0, "precipitationAmount": 0 }], }</pre>	
400	Parámetros de latitud o longitud inválidos	No links
	<p>Media type</p> <div style="border: 1px solid #ccc; padding: 2px; display: inline-block;">application/json</div> <p>Example Value Schema</p> <pre>{ "statusCode": 403, "message": "Acceso denegado", "details": "Usuario no tiene permisos" }</pre>	
500	Error interno del servidor al recuperar registros	No links
	<p>Media type</p> <div style="border: 1px solid #ccc; padding: 2px; display: inline-block;">application/json</div>	



GET /weather/historical Obtiene registros históricos de datos meteorológicos para una ubicación.

Recupera los registros históricos de datos meteorológicos para una latitud y longitud dadas.

Parameters

Name **Description**

latitude * required
number(\$double)
(query)
Example : -33.4489

-33.4489

longitude * required
number(\$double)
(query)
Example : -70.6693

-70.6693

limit
integer(\$int32)
(query)
Default value : 10
Example : 5

5

Responses

Code	Description	Links
200	Registros históricos obtenidos exitosamente	No links

Media type
application/json ▾
Controls Accept header.

Example Value Schema



Code	Description	Links
200	Registros históricos obtenidos exitosamente	No links
	<p>Media type</p> <p>application/json</p> <p>Controls Accept header.</p> <p>Example Value Schema</p> <pre>{"latitude": 0, "longitude": 0, "countryCode": "string" }, "recordedAt": "2025-05-28T15:30:00.123456Z", "instantWeatherSnapshot": { "airTemperature": 0, "relativeHumidity": 0, "airPressureAtSeaLevel": 0, "windSpeed": 0, "cloudAreaFraction": 0 }, "hourlyForecasts": [{ "time": "string", "airTemperature": 0, "windSpeed": 0, "precipitationAmount": 0 }], "airQualitySnapshot": { "latitude": 0, "longitude": 0, "components": { "additionalParam1": 0 } }</pre>	
400	Parámetros de latitud o longitud inválidos	No links

GET /weather/full-report Obtiene un reporte meteorológico completo (actuales, pronóstico por hora, mapa de viento y calidad del aire) para una ubicación específica.

Combina llamadas a servicios de geocodificación y pronóstico meteorológico para proporcionar un conjunto completo de datos. Incluye manejo de caché para mejorar el rendimiento.

Parameters

Name	Description
addressQuery <small>* required</small>	Cadena de búsqueda de la dirección (ej. 'Calle Falsa 123, Springfield, USA' o 'Hijuelas, Valparaíso, Chile')
(query)	Hijuelas, Valparaíso, Chile

Responses

Curl

```
curl -X 'GET' \
'http://localhost:8080/weather/full-report?addressQuery=Hijuelas%20Valparaíso%20Chile' \
-H 'Accept: application/json' \
-H 'Authorization: Bearer eyJhbGciOiJSUzI1NiIsImtpZCI6IjNzjA1Mzk0Mzk2OTEzYTc4ZHM4MGY0MjcwMzM4NjM2NDa2MTBhZGMiLCJ0eXAiOiJKV1QiLCJ4eXAiOiJkV1QiLCJleHBvcnkiOiJhdHRwczovL3N1Y3VyZXRoVa2VuLmdvb2dsZS5jb20vYmFjZDgiLCJpYXJhZGFnZWxpbWUiOiJ1c2VycmluZCIsImFtciI6ImFtciJ9'
```

Request URL

```
http://localhost:8080/weather/full-report?addressQuery=Hijuelas%20Valparaíso%20Chile
```



Code	Description	Links
200	Datos meteorológicos obtenidos exitosamente	No links
	<p>Media type</p> <div style="border: 1px solid #ccc; padding: 2px; display: inline-block;">application/json</div> <p>Controls Accept header.</p> <p>Example Value Schema</p> <pre>{ "location": { "cityName": "Santiago", "latitude": -33.4489, "longitude": -70.6693, "countryName": "Chile" }, "instantWeather": { "airTemperature": 15, "relativeHumidity": 70, "airPressureAtSeaLevel": 1012.5, "windSpeed": 5, "cloudAreaFraction": 50 }, "hourlyForecasts": [{ "time": "2025-01-01T10:00:00Z", "airTemperature": 16, "windSpeed": 4.5, "precipitationAmount": 0 }, { "time": "2025-01-01T11:00:00Z", "airTemperature": 17, "windSpeed": 4.5, "precipitationAmount": 0 }] }</pre>	
400	Parámetros de entrada inválidos o ubicación no encontrada	No links
	<p>Media type</p> <div style="border: 1px solid #ccc; padding: 2px; display: inline-block;">application/json</div> <p>Controls Accept header.</p> <p>Example Value Schema</p> <pre>{ "statusCode": 403, "message": "Acceso denegado", "details": "Usuario no tiene permisos" }</pre>	No links
500	Error interno del servidor	No links



Schemas

^

AirQuality ✓ {
description: Datos de calidad del aire para la ubicación.
latitude > [...]
longitude > [...]
components > {...}
timestamp > [...]
aqiCategory > [...]
aqi > [...]
}

HistoricalWeatherEntry ✓ {
description: Registro histórico de datos meteorológicos para un lugar y tiempo específicos
id > [...]
location* LocationCoordinates > {...}
recordedAt* > [...]
instantWeatherSnapshot InstantWeather > {...}
hourlyForecasts > [...]
airQualitySnapshot AirQuality > {...}
}

HourlyForecast ✓ {
description: Detalles del pronóstico por hora
time > [...]
airTemperature > [...]
windSpeed > [...]
precipitationAmount > [...]
}

InstantWeather ✓ {
description: Detalles de los datos instantáneos
airTemperature > [...]
relativeHumidity > [...]
airPressureAtSeaLevel > [...]
windSpeed > [...]
cloudAreaFraction > [...]
}



```
LocationCoordinates ✕ {  
    description: Coordenadas de la ubicación solicitada.  
    name  
    latitude  
    longitude  
    countryCode  
    > [...]  
}
```

```
ErrorResponse ✕ {  
    description: Modelo de error para respuestas HTTP  
    statusCode  
    message  
    details  
    > [...]  
}
```

```
LoginRequestDto ✕ {  
    email  
    password  
    > [...]  
}
```

```
Geometry ✕ {  
    description: Representa la geometría de un punto  
    type  
    coordinates  
    > [...]  
}
```

```
Properties ✕ {  
    description: Propiedades del punto, incluye velocidad y dirección del viento  
    windSpeed  
    windDirection  
    time  
    > [...]  
}
```



<pre>WindMap ✓ { description: Represents a collection of points for the interactive map type > [...] features > [...] }</pre>
<pre>WindMapPoint ✓ { description: Represents a point of data in the wind speed map geometry Geometry > {...} properties Properties > {...} type > [...] }</pre>
<pre>NASAApodInfo ✓ { description: Information about the Daily Astronomical Image (APOD) from NASA title > [...] explanation > [...] url > [...] thumbnailUrl > [...] copyright > [...] mediaType > [...] date > [...] }</pre>
<pre>Results ✓ { description: Details of astronomical times sunrise > [...] sunset > [...] solar_noon > [...] day_length > [...] civil_twilight_begin > [...] civil_twilight_end > [...] nautical_twilight_begin > [...] nautical_twilight_end > [...] astronomical_twilight_begin > [...] astronomical_twilight_end > [...] }</pre>
<pre>WeatherResponse ✓ { description: Consolidated response containing all weather, air quality and wind map data for a location. location LocationCoordinates > {...} currentWeather InstantWeather > {...} hourlyForecasts > [...] airQuality AirQuality > {...} windMap WindMap > {...} astronomicalTimes Results > {...} nasaApod NASAApodInfo > {...} }</pre>



Anexo 2: Pruebas en postman

The screenshot displays two separate API requests in the Postman application:

Request 1: User Registration

- Method: POST
- URL: <http://localhost:8080/auth/register>
- Body (JSON):

```
1 {
2   "email": "user@example.com",
3   "password": "password123"
4 }
```

Response 1: Status: 400 Bad Request | Time: 490 ms | Size: 529 B

```
1 {
2   "error": "Error al registrar usuario: The user with the provided email already exists (EMAIL_EXISTS)."
3 }
```

Request 2: Instant Weather Data

- Method: GET
- URL: <http://localhost:8080/weather/instant/last>
- Headers (8):

Key	Value
User-Agent	PostmanRuntime/7.43.0
Accept	application/json
Accept-Encoding	gzip, deflate, br
Connection	keep-alive
accept	application/json
Authorization	Bearer eyJhbGciOiJSUzI1NiIsImtpZCI6ImE0MzMzMzFkN2Y3NWRn2QyZjQ0Yjg...
Key	

Response 2: Status: 200 OK | Time: 3.16 s | Size: 559 B

```
1 {
2   "airTemperature": 1.1,
3   "relativeHumidity": 0.0,
4   "airPressureAtSeaLevel": 0.0,
5   "windSpeed": 1.5,
6   "cloudAreaFraction": 0.0
7 }
```



10. REFERENCIAS.

- [1] “Welcome to the MET Weather API”. Welcome to the MET Weather API. Accedido el 18 de enero de 2025. [En línea]. Disponible: <https://www.met.no/>
- [2] “News”. MET Weather API. Accedido el 15 de enero de 2025. [En línea]. Disponible: <https://docs.Api.met.no/>
- [3] “Proven Accuracy | The Weather Company”. The Weather Company. Accedido el 21 de enero de 2025. [En línea]. Disponible: <https://www.weathercompany.com/provenaccuracy/>
- [4] “Ni Apple Weather ni Accuweather: un estudio meteorológico confirma cuál es la mejor app para el tiempo”. Meristation. Accedido el 21 de enero de 2025. [En línea]. Disponible: https://as.com/meristation/betech/ni-apple-weather-ni-accuweather-unestudio-meteorologico-confirma-cual-es-la-mejor-app-para-el-tiempo/?utm_source=com
- [5] “Windy API - Home”. Windy API - Home. Accedido el 23 de enero de 2025. [En línea]. Disponible: <https://Api.windy.com/>
- [6] Stackscale, “Bases de datos NoSQL: características y tipos”. Accedido el 15 de enero de 2025. [En línea]. Disponible: <https://www.stackscale.com/es/blog/bases-de-datos- nosql/>
- [7] Google, “Firebase”. Accedido el 8 de enero de 2025. [En línea]. Disponible: <https://firebase.google.com/>
- [8] Google, “Documentación de Firebase, Authentication”. Accedido el 10 de enero 2025. [En línea]. Disponible: <https://firebase.google.com/docs/auth>
- [9] Programación y Más, “Android: ¿Qué es MVC, MVP y MVVM?”, PYM. Accedido el 15 de enero de 2025. [En línea]. Disponible: <https://programacionymas.com/blog/android- mvc-mvp-mvvm>
- [10] A. Leiva, ” MVP para Android: Cómo organizar la capa de presentación”, DevExpert, 3 diciembre 2019. Accedido el 15 de enero de 2025. [En línea]. Disponible: <https://devexpert.io/mvp-android/>
- [11] “MVVM en Android con Kotlin, LiveData y View Binding – Android Architecture Components”, Curso Kotlin, 22 abril 2021. Accedido el 15 de enero de 2025. [En línea]. Disponible: <https://cursokotlin.com/mvvm-en-android-con-kotlin-livedata-y-view- binding-android-architecture-components/>



- [12] “JUnit 5”. Accedido el 27 de enero de 2025. [En línea]. Disponible: <https://junit.org/junit5/>
- [13] “Log” Developer Android. Accedido el 18 de enero de 2025. [En línea]. Disponible: <https://developer.android.com/reference/android/util/Log>
- [14] “API Documentation & Design Tools for Teams | Swagger”. API Documentation & Design Tools for Teams | Swagger. Accedido el 3 de febrero de 2025. [En línea]. Disponible: <https://swagger.io/>
- [15] “SonarQube Cloud”. SonarCloud Online Code Review as a Service Tool | Sonar. Accedido el 22 de enero de 2025. [En línea]. Disponible: <https://sonarcloud.io/documentation>
- [15] “Build app server send requests | Firebase Cloud Messaging”. Firebase. Accedido el 15 de enero de 2025. [En línea]. Disponible: <https://firebase.google.com/docs/cloudmessaging/send-message>
- [16] “Content Manager 8.5.0”. IBM - United States. Accedido el 15 de enero de 2025. [En línea]. Disponible: <https://www.ibm.com/docs/es/content-manager/8.5.0?topic=Apisunderstanding-clientserver-architecture-java-only>