



# INFORME FINAL DE PROYECTO

Período Académico (2025)

FACULTAD DE INFORMATICA Y ELECTRONICA

CARRERA DE INGENIERIA EN TECNOLOGIAS DE LA INFORMACIÓN

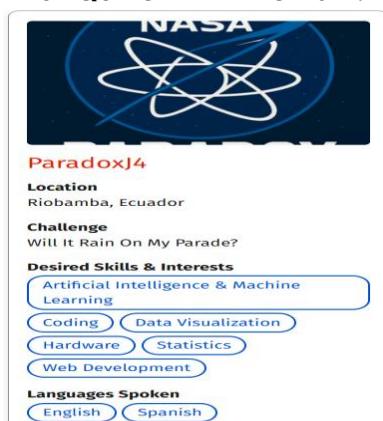
URKUSC

# CONTENIDO

## Índice

- 1. DATOS GENERALES EQUIPO “PARADOX J4”
  - 1.1 DATOS DEL CAPITÁN
  - 1.2 DATOS DE LOS MIEMBROS
- 2. DESARROLLO
  - 2.1 INTRODUCCIÓN
- 3. OBJETIVOS
- 4. DURACIÓN
- 5. METODOLOGÍA MOBILE-D Y PATRÓN DE DISEÑO MVC
  - 5.1 Metodología Mobile-D
  - 5.2 Patrón de Diseño MVC
  - 5.3 Matices adicionales y posibles extensiones
  - 5.4 Consideraciones específicas
- 6. RESULTADOS OBTENIDOS EN LA APLICACIÓN MÓVIL
- 7. CONCLUSIONES
- 8. RECOMENDACIONES
- 9. ANEXO

## 1. DATOS GENERALES EQUIPO "PARADOX J4":



**ParadoxJ4**

**Location**  
Riobamba, Ecuador

**Challenge**  
Will It Rain On My Parade?

**Desired Skills & Interests**  
Artificial Intelligence & Machine Learning  
Coding Data Visualization  
Hardware Statistics  
Web Development

**Languages Spoken**  
English Spanish

### DATOS DEL CAPITAN

Nombre: Juan Stalin Quezada Ruiz  
Teléfonos: 0990903557

E- Mail: [juan.quezada@epoch.edu.ec](mailto:juan.quezada@epoch.edu.ec)

### DATOS DE LOS MIEMBROS

Nombre: NASA Space Apps Challenge - Riobamba  
Ámbito Empresa / Institución: Evento tecnológico global  
Provincia: Chimborazo  
Cantón: Riobamba  
Dirección: Escuela Superior Politécnica del Chimborazo

Nombres	Correos
Lisbeth Aracelly Escudero Coba	<a href="mailto:Lisbeth.escudero@epoch.edu.ec">Lisbeth.escudero@epoch.edu.ec</a>
Juan Stalin Quezada Ruiz	<a href="mailto:juan.quezada@epoch.edu.ec">juan.quezada@epoch.edu.ec</a>
Jordy Ricardo Carrión Chávez	<a href="mailto:Jordy.ricardo25@gmail.com">Jordy.ricardo25@gmail.com</a>
Edison Efrain Remache Gualacio	<a href="mailto:Efrain.remache@epoch.edu.ec">Efrain.remache@epoch.edu.ec</a>
Jordy Joel Vele Carrera	<a href="mailto:jordy.vele@epoch.edu.ec">jordy.vele@epoch.edu.ec</a>
Jordi Armando Touriz Garnica	<a href="mailto:jordi.touriz@epoch.edu.ec">jordi.touriz@epoch.edu.ec</a>

## 2. DESARROLLO

### INTRODUCCIÓN

El NASA Space Apps Challenge es un hackatón global que fomenta la innovación tecnológica utilizando datos de observación de la Tierra de la NASA. Participar en este evento de dos días permitió al estudiante aplicar conocimientos teóricos y habilidades técnicas en el desarrollo de una aplicación innovadora. Este informe documenta las actividades realizadas, los resultados obtenidos y las lecciones aprendidas durante el concurso, centrado en el desarrollo de "WeatherLens", una aplicación diseñada para analizar probabilidades de condiciones climáticas.

A lo largo del documento, se describen las tareas ejecutadas, el cumplimiento de los objetivos establecidos y una reflexión sobre la experiencia, incluyendo recomendaciones para futuros participantes.

### 3. OBJETIVOS

#### Objetivo General

Desarrollar una aplicación innovadora utilizando datos de observación de la Tierra de la NASA, integrando conocimientos técnicos para crear un panel personalizado que permita a los usuarios analizar la probabilidad de condiciones climáticas específicas en ubicaciones y fechas seleccionadas durante el NASA Space Apps Challenge.

#### Objetivos Específicos

- Investigar y seleccionar fuentes de datos climáticos de la NASA para alimentar la aplicación con información relevante.
- Desarrollar un backend y scripts para procesar datos y generar salidas personalizadas, incluyendo la opción de descargar archivos de datos.
- Implementar una interfaz interactiva que permita a los usuarios ingresar ubicaciones y fechas, visualizando probabilidades de condiciones climáticas.

### 4. DURACIÓN

Fecha de Inicio

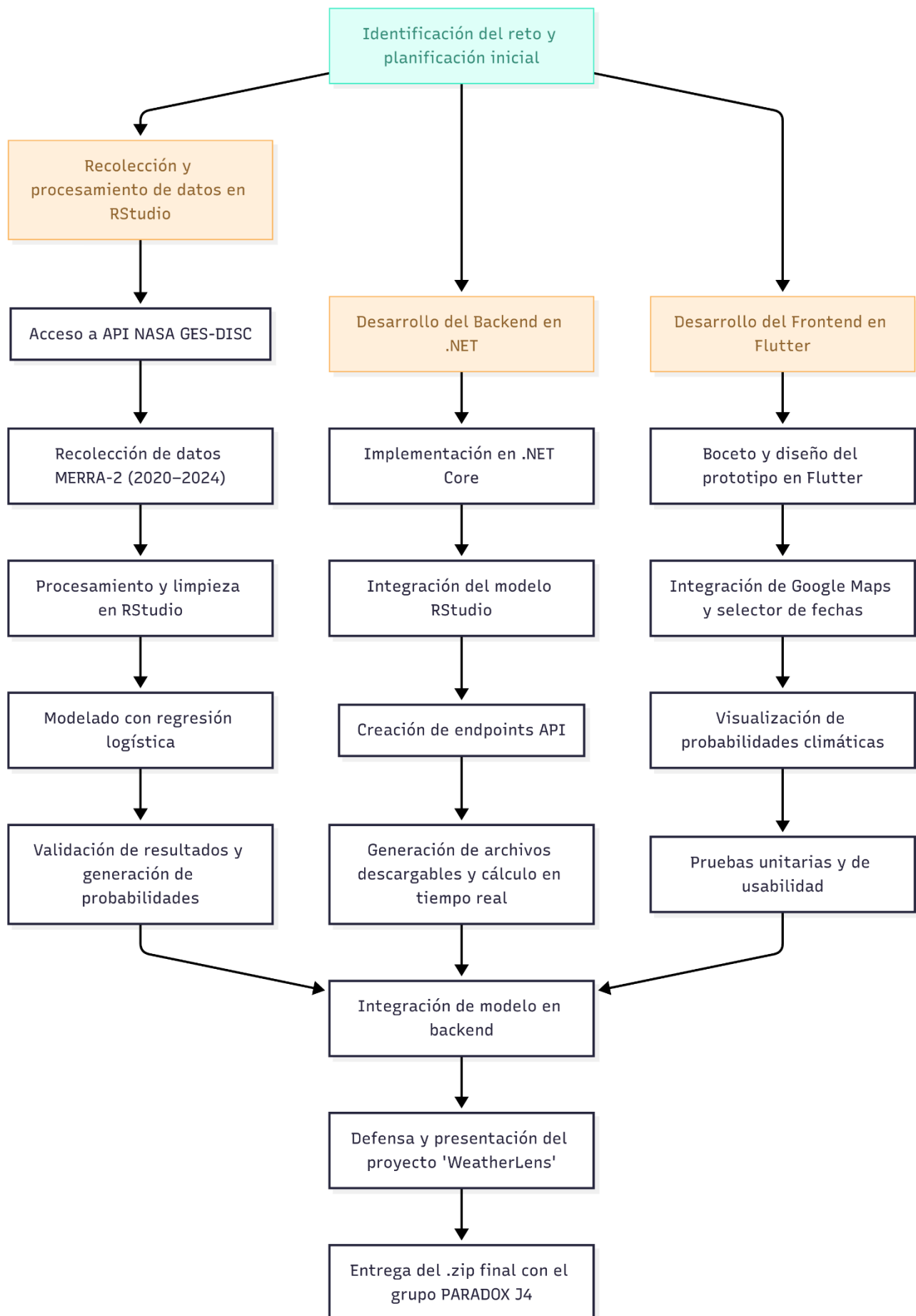
04 / 10 / 2025

Fecha de finalización

05 / 10 / 2025

#### Enfoque Inicial

El equipo PARADOX J4 abordó el desarrollo de "WeatherLens" en el NASA Space Apps Challenge 2025 seleccionando el reto "Climate Change Knowledge", enfocado en analizar probabilidades de condiciones climáticas extremas. Inicialmente, anticipamos recibir una base de datos predefinida, pero la caída temporal de la página NASA nos llevó a explorar alternativas, optando por la API GESDISC tras revisar su documentación. Utilizamos Postman para probar esta API, identificando parámetros clave como temperatura, humedad, viento y radiación, adaptándonos a los nombres específicos de NASA. Este proceso inició con peticiones exitosas en Postman, organizando datos mediante una aplicación de consola en .NET para descargarlos en carpetas. En RStudio, tras una limpieza exhaustiva, creamos una base con datos de 50 estados de EE.UU. (2020-2024), entrenando un modelo de regresión probabilístico con cinco estados propensos a desastres (Florida, Mississippi, Louisiana, Texas, Alabama) por su representatividad, dejando espacio para escalar. Finalmente, integramos estos datos en un backend .NET con endpoints REST y un frontend Flutter, desarrollando una app móvil Android con interfaz interactiva.

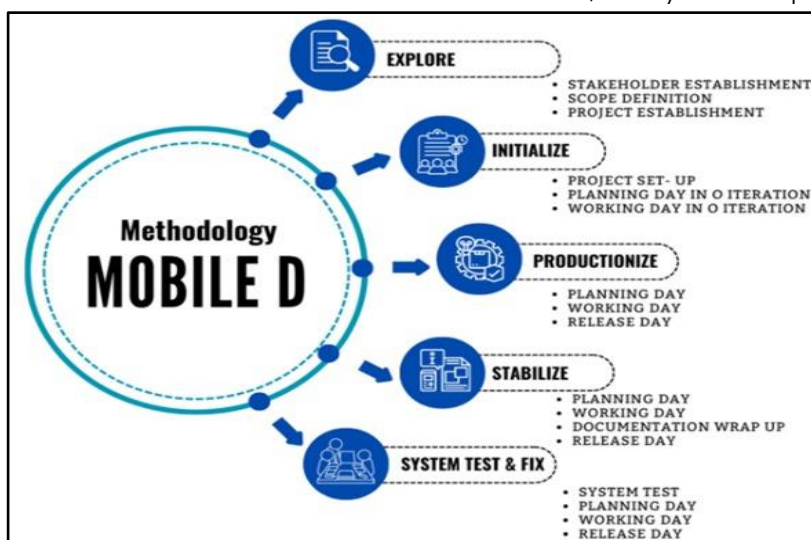


## 5. Metodología de desarrollo y patron de diseño MVC

### - Metodología Mobile-D

La metodología Mobile-D, un enfoque ágil optimizado para proyectos móviles, fue seleccionada para el desarrollo de "WeatherLens" durante el NASA Space Apps Challenge (4-5 de octubre de 2025) debido a su capacidad para gestionar proyectos en entornos de alta presión con plazos cortos. Mobile-D estructura el desarrollo en cinco fases iterativas, adaptadas al hackatón de 48 horas, permitiendo una planificación efectiva, ejecución rápida y validación continua. A continuación, se detallan las fases aplicadas y su implementación específica:

- **Exploración:** Analizamos los retos del NASA Space Apps Challenge, seleccionando el desafío de cambio climático (Climate Change Knowledge) por su relevancia y la disponibilidad de datos de la NASA. Realizamos un brainstorming para definir requisitos clave: una aplicación que prediga condiciones climáticas extremas (ej. calor >32°C, alta precipitación) con un mapa interactivo y descargas de datos. Identificamos fuentes como la API GESDISC (MERRA-2) y la API de Google Maps. Se documentaron ideas iniciales con fotos del equipo (Anexo: Imagen 1).
- **Inicialización:** Dividimos al equipo "PARADOX J4" de seis integrantes en tres subequipos: frontend (tres personas para la interfaz en Flutter), backend (dos personas para el procesamiento en .NET) y modelo de regresión probabilístico (2 persona para el análisis en RStudio). Esta división se basó en las fortalezas de cada miembro (TI y Estadística).
- **Producción:** Los subequipos trabajaron en paralelo. El equipo de frontend creó un boceto inicial y prototipos en Flutter. El equipo de backend configuró la API REST en .NET y procesó datos de RStudio. El equipo de predicción desarrolló un modelo de regresión probabilístico en RStudio para calcular probabilidades climáticas. Iteramos rápidamente.
- **Estabilización:** Refinamos la aplicación, corrigiendo errores como fallos en la carga de datos de la API GESDISC y problemas de sincronización entre el mapa de Flutter y las predicciones. Se optimizó el rendimiento del backend para reducir latencia en las consultas API. Se realizaron revisiones cruzadas para asegurar la cohesión entre frontend, backend y modelo.
- **Pruebas:** Ejecutamos pruebas unitarias (ej. validación de endpoints en .NET, widgets en Flutter) y pruebas de usuario simuladas para verificar la usabilidad del mapa interactivo y la precisión de las predicciones. Los resultados se documentaron en GitHub, incluyendo capturas de pruebas exitosas.



### Patrón de Diseño MVC

- **Model:**
  - El Model encapsula los datos y la lógica de negocio subyacente del sistema "WeatherLens". Incluye los datos generados por el modelo predictivo de variabilidades climáticas procesado

en RStudio, exportados o integrados al backend .NET como archivos CSV, bases de datos o respuestas API. También abarca los datos geográficos de la API de Google Maps (coordenadas, marcadores, capas geográficas), que se combinan con las predicciones climáticas. El backend .NET actúa como el núcleo del Model, procesando las salidas de RStudio y sirviéndolas a través de una API REST, gestionando la persistencia y la lógica de acceso a datos para alimentar la aplicación.

- View:
  - La View representa la interfaz de usuario en Flutter, encargada de renderizar el mapa interactivo de Google Maps y mostrar los datos climáticos (por ejemplo, superposiciones de temperatura, precipitación o alertas climáticas en el mapa). Los widgets de Flutter se actualizan dinámicamente según los datos recibidos del Controller, reflejando las condiciones climáticas seleccionadas por el usuario (ej. calor >32°C). Aunque no es reactiva por sí misma como en MVVM, la View depende del Controller para recibir actualizaciones y presenta la información en un formato visual accesible, como gráficos o texto, adaptado a usuarios planificando actividades al aire libre.
- Controller:
  - El Controller actúa como intermediario entre el Model y la View, gestionando la lógica de interacción y el flujo de datos en "WeatherLens". En Flutter, el Controller maneja solicitudes al backend .NET para obtener los datos climáticos procesados en RStudio, transforma las predicciones (por ejemplo, mapeando datos a marcadores o capas en Google Maps) y controla el estado de la interfaz (cargando, error, datos listos). Esto se puede implementar utilizando un sistema de control de estado como setState o un framework como GetX, común en Flutter para conectar la View con el Model a través del Controller, asegurando una comunicación unidireccional eficiente.

Matices Adicionales y Posibles Extensiones:

- Arquitectura Limpia (Clean Architecture):
  - Aunque MVC estructura bien la separación de responsabilidades, la integración de un backend .NET con un modelo probabilístico en RStudio sugiere beneficios al aplicar principios de Clean Architecture. Esto incluye:
    - Capa de Dominio: El modelo probabilístico en RStudio como la lógica central de predicciones climáticas, independiente de tecnologías específicas.
    - Capa de Datos: El backend .NET como repositorio, procesando datos de RStudio y exponiéndolos vía API.
    - Capa de Presentación: La interfaz Flutter con MVC, consumiendo datos para la View.
  - Esta arquitectura mejora la escalabilidad y modularidad, especialmente con múltiples fuentes de datos como GESDISC o Google Maps.
- Patrón Repository (en el backend):
  - En el backend .NET, el uso de un patrón Repository abstrae el acceso a los datos generados por RStudio (ej. predicciones almacenadas en bases de datos o archivos), facilitando la integración con la API que el Controller utiliza. Esto centraliza la lógica de datos y simplifica las actualizaciones o cambios en las fuentes.
- Integración con Google Maps:
  - La API de Google Maps como fuente externa de datos geográficos refuerza la necesidad de un Controller que coordine los datos climáticos del backend con las coordenadas del mapa. El Controller transforma y pasa esta información a la View, asegurando una integración fluida que alinea las predicciones con las ubicaciones seleccionadas por el usuario.

Consideraciones Específicas:

- Procesamiento en RStudio: Si el modelo en RStudio genera datos estáticos (ej. archivos CSV) que el backend .NET sirve directamente, el flujo es directo: RStudio → .NET → API → Controller → View. Para

predicciones en tiempo real, el backend podría integrar scripts de R mediante R.NET o una API ejecutable, añadiendo complejidad al Model y requiriendo ajustes en el Controller para manejar actualizaciones dinámicas.

- **Gestión de Estado en Flutter:** Dado que la View depende del Controller para reflejar cambios (ej. actualizaciones en el mapa al recibir nuevas predicciones), se puede usar setState o un framework como GetX para gestionar el estado, asegurando que la interfaz responda a las interacciones del usuario de manera eficiente.
- **Escalabilidad:** La estructura MVC, combinada con Clean Architecture en el backend, permite expandir "WeatherLens" (ej. añadiendo más variables climáticas o fuentes como GPM) sin comprometer la organización del código. El Controller puede manejar nuevas lógicas de negocio, y la View se adapta a interfaces más complejas, manteniendo la aplicación mantenible y escalable.

## 6. RESULTADOS OBTENIDOS EN LA APLICACION MOVIL

Durante el NASA Space Apps Challenge en Riobamba, el equipo "PARADOX J4" desarrolló "WeatherLens" en un período de dos días, aprovechando datos de observación de la Tierra de la NASA para crear una herramienta predictiva de condiciones climáticas. Las actividades se llevaron a cabo en un entorno colaborativo, superando desafíos técnicos y adaptándonos a limitaciones de tiempo. A continuación, se detalla una tabla con las actividades principales, integrando un enfoque técnico, creativo y reflexivo sobre el impacto de la experiencia:

N.	ACTIVIDADES	OBSERVACIONES
1	<p><b>Identificación del reto del concurso y planificación inicial:</b> Se seleccionó el reto relacionado con el cambio climático (Climate Change Knowledge), enfocado en analizar probabilidades de condiciones climáticas extremas.</p> <ul style="list-style-type: none"> <li>- Principales dificultades incluyeron la caída temporal de la página de la NASA para acceder a datos, lo que nos obligó a buscar alternativas. Inicialmente exploramos datos externos de fuentes oficiales, pero optamos manejar datos de la página de la NASA como al principio.</li> </ul> <p><a href="#">GES DISC How-To's: Cómo acceder a la API de series temporales de barras de datos de hidrología usando Python</a></p>	Ver Anexos: Imagen 1, imagen 2, imagen 3, imagen 4
2	<p><b>Recolección de datos proporcionados por la NASA:</b> Utilizamos la API "NASA GES-DISC" (Goddard Earth Sciences Data and Information Services Center) para obtener conjuntos de datos climáticos como MERRA-2, que incluyen variables como temperatura, precipitación y viento.</p> <ul style="list-style-type: none"> <li>- <b>Principales dificultades</b> involucraron el manejo de la API, ya que requería autenticación y parámetros específicos; usamos Postman para probar solicitudes y visualizar respuestas basadas en notebooks de Jupyter. Enlace principal: <a href="https://disc.gsfc.nasa.gov/">https://disc.gsfc.nasa.gov/</a> (fuente oficial de datos climáticos de la NASA).</li> </ul>	Ver Anexos: Imagen 5, Imagen 6.
3	<p><b>Construcción de la base de datos:</b> Se creó una estructura de datos basada en conjuntos obtenidos de los 50 estados de EE.UU. Se desarrolló un script en .NET para extraer y organizar variables meteorológicas</p>	Ver Anexos: Imagen 7, imagen 8, imagen 9.



	durante un período de 5 años (ej. 2020-2024). Este enfoque se justificó por la disponibilidad de datos granulares en EE.UU., permitiendo un análisis representativo de patrones climáticos regionales que podrían escalarse globalmente, evitando sesgos en áreas con datos incompletos.	
4	<b>Procesamiento de los datos en RStudio:</b> Se empleó RStudio para limpiar, analizar y modelar datos climáticos obtenidos de la API GESDISC (MERRA-2), enfocándose en los cinco estados con mayor incidencia de desastres naturales en los últimos años: Florida, Mississippi, Louisiana, Texas, y Alabama. Se utilizó un modelo de regresión probabilístico debido a su robustez para predecir probabilidades de eventos climáticos extremos (ej. calor >32°C o precipitación alta) a partir de variables binarias, permitiendo incorporar múltiples factores como temperatura, humedad, velocidad del viento, radiación corta y radiación larga. La elección del modelo se justificó por su capacidad para manejar correlaciones entre variables y generar predicciones precisas, validadas con un análisis estadístico que mostró coeficientes significativos (ej. temperatura: 1.000, radiación larga: 0.946). Referencia del modelo: Basado en técnicas de modelado binomial de RStudio, adaptadas de la documentación de la NASA Earthdata.	Ver Anexos: imagen 10, imagen 11, imagen 12, imagen 13, imagen 14.
5	<b>Desarrollo del backend:</b> Se implementó en .NET para manejar el procesamiento de datos en tiempo real, integrando el modelo creado en RStudio. Esto incluyó endpoints para consultas de API, cálculo de probabilidades y generación de archivos descargables, asegurando escalabilidad y seguridad en el manejo de datos sensibles.	Ver Anexos: Imagen 15, Imagen 16.
6	<b>Diseño del frontend y creación del icono de la aplicación:</b> Se elaboró un boceto inicial, para la interfaz en Flutter, incluyendo un mapa interactivo de Google Maps, un selector de fechas y un panel de resultados con gráficos y opciones de descarga. El diseño priorizó usabilidad con colores contrastantes (azul para precipitación, rojo para calor). El icono de "WeatherLens" se generó con IA (ej.leonardo.IA), ya que, como equipo de TI y Estadística, carecemos de experiencia en diseño gráfico, optimizando tiempo y asegurando un resultado profesional con elementos climáticos (globo, nubes, lente) en tonos azules y rojos.	Ver Anexos: Imagen 17, Imagen 18, Imagen 19, imagen 20.
7	<b>Desarrollo del frontend:</b> Se implementó la interfaz de "WeatherLens" en Flutter, utilizando el patrón de diseño MVVM para garantizar modularidad y escalabilidad. El frontend incluyó un mapa interactivo de Google Maps para seleccionar ubicaciones, un selector de fechas y un panel de resultados con gráficos reactivos que muestran probabilidades climáticas (ej. calor >32°C). Se realizaron pruebas unitarias (ej. validación de widgets) y de usabilidad (ej. interacción con el mapa) para asegurar funcionalidad y una experiencia de usuario fluida en dispositivos móviles.	Ver Anexos: Imagen 21, Imagen 22, imagen 23.
8	<b>Defensa del proyecto:</b> Se preparó y presentó una demostración oral de "WeatherLens" ante los jueces del NASA Space Apps Challenge en Riobamba. La presentación destacó la funcionalidad de la aplicación, incluyendo el mapa interactivo, las predicciones climáticas basadas en datos de la NASA (API GESDISC) y la opción de descarga de datos. Se mostró una demo en vivo, navegando por la interfaz en Flutter y	

	explicando el modelo probabilístico de RStudio integrado en el backend .NET.	
9	Entrega del proyecto realiza con el grupo <b>"PARADOX J4"</b> en un .zip comprimido	

### Beneficios para los Usuarios en Comparación con Opciones Tradicionales de la NASA

"WeatherLens" ofrece un valor único frente a herramientas tradicionales como MERRA-2 de GESDISC o Earthdata Search, que exigen autenticación y conocimientos técnicos para acceder a APIs. Nuestra app móvil Android entrega predicciones probabilísticas en tiempo real, eliminando la necesidad de navegar sitios complejos o procesar datos manualmente, beneficiando a planificadores de eventos al aire libre con alertas visuales y descargas personalizadas (ej. 60% probabilidad de calor >32°C). A diferencia de apps genéricas de clima, WeatherLens atiende a usuarios generales y técnicos, como científicos de NASA, con un modelo de regresión probabilístico basado en análisis estadístico en RStudio. Esto permite planificar a largo plazo (meses o años) y descargar reportes (CSV/JSON) útiles para investigaciones, superando las limitaciones de pronósticos cortoplacistas y enfoques científicos no orientados a móviles.

### Escalabilidad del Proyecto

La escalabilidad de "WeatherLens" se fundamenta en un diseño flexible con potencial para adaptarse a diversos contextos geográficos y funcionales en el futuro. Partiendo de datos de estados de EE.UU., el proyecto se puede expandir a ciudades o regiones ajustando latitud/longitud mediante el script .NET existente. Esta capacidad podría enriquecerse con la implementación de roles de usuario (superusuarios, técnicos, científicos) que requieran seguridad y delimitación, así como la adición de nuevas secciones como mapas de calor. El algoritmo basado en R tiene el potencial de alimentarse con bases de datos de otras agencias aeroespaciales (ej. brasileña o europea) para mejorar la precisión de las predicciones. Además, se podrían incorporar funcionalidades como notificaciones push, login vía APIs de Google y reportes PDF. La transición a una app web, similar a GESDISC, con permisos de ubicación para centrar el mapa, es una posibilidad viable. Al usar Flutter como framework, se puede escalar a iOS, aplicaciones de escritorio (Windows o Linux) o web, dependiendo de la compatibilidad de paquetes, ampliando su alcance futuro. El modelo, entrenado con 3200 tuplas por variable (solo un estado), se diseñó para escalar a 50 estados y más, con un futuro potencial en Machine Learning, aunque limitado por el tiempo del reto y recursos computacionales.

## 7. CONCLUSIONES:


- Se fortalecieron competencias en desarrollo colaborativo y manejo de datos reales de la NASA, permitiendo la creación de una aplicación funcional que integra frontend, backend y modelos predictivos para analizar riesgos climáticos.
- La superación de dificultades técnicas, como caídas de servidores y manejo de APIs complejas, mejoró la resiliencia del equipo y validó la utilidad de herramientas como Postman y RStudio en entornos de tiempo limitado.
- El proyecto contribuyó a la misión del NASA Space Apps Challenge al promover soluciones basadas en datos abiertos, demostrando cómo la tecnología puede empoderar a usuarios en la toma de decisiones informadas sobre el cambio climático.

## 8. RECOMENDACIONES:

- Para futuros participantes, se recomienda explorar múltiples fuentes de datos de respaldo ante posibles caídas de páginas de la NASA, como APIs alternativas o datasets pre-descargados, para evitar interrupciones en la recolección.
- Es aconsejable documentar exhaustivamente el manejo de APIs (ej. GESDISC) desde el inicio, incluyendo pruebas con herramientas como Postman, ya que los datos a menudo están incompletos o requieren parámetros específicos que no son intuitivos.
- Se sugiere planificar divisiones de equipo claras desde el principio, combinando perfiles de TI y Estadística para equilibrar desarrollo técnico y análisis predictivo, maximizando la eficiencia en hackatones cortos.


## ANEXOS

### TEAM MEMBERS




**Juan Stalin Quezada Ruiz**  
@juanquezada  
Ecuador


Team Owner




**EDISON EFRAIN REMACHE GUALACIO**  
@edisonrg\_21  
Ecuador




**Jordy Ricardo Carrión Chávez**  
@jordy-ricardo25  
Ecuador



**Jordy Joel Vele Carrera**  
@\_lawgoth\_  
Ecuador



**Lisbeth Aracelly Escudero Cobra**  
@aracellyec\_2025  
Ecuador



**Jordi Armando Touriz Garnica**  
@jordit  
Ecuador

You are a team member.

[Leave Team](#)

#### Team Information

**Local Event**  
Riobamba, Ecuador

**Challenge**  
Will It Rain On My Parade?

**Desired Skills & Interests**

Artificial Intelligence & Machine Learning

Coding Data Visualization Hardware

Statistics Web Development

**Languages Spoken**

English Spanish

Imagen 1

https://disc.gsfc.nasa.gov/information/howto?title=How%20to%20Access%20the%20Hydrology%20Data%20Rods%20Ti... Inicio sesión

**DATOS DE LA TIERRA** Buscar un DAAC

# DISCO DE GES

12 Retroalimentación Migración a la nube Ayuda

Procedimientos Ingrese la búsqueda (p)

[Composición atmosférica](#), [ciclos del agua](#) y [la energía y variabilidad climática](#)

**Debido a la interrupción de la financiación del gobierno federal, la NASA no está actualizando este sitio web. Lamentamos sinceramente este inconveniente.**

[Volver a los procedimientos](#)

## Cómo acceder a la API de series temporales de barras de datos de hidrología usando Python

Descargue el [Jupyter Notebook](#) complementario

**Visión general:**

En este cuaderno se describe el acceso a la API de series temporales de barras de datos de hidrología mediante Python. Consulta una cuadrícula más cercana a Newton, IL, y consulta dos meses de superficie NLDAS por hora y totales de precipitación del suelo superior de 0-100 cm. Estas variables se trazan usando Matplotlib.

**Procedimiento:**

Cómo acceder a la API de series temporales de barras de datos de hidrología usando Python

Windows Buscar 8°C Prac. despejado 0:56 05/10/2025

Imagen 2

climateknowledgeportal.worldbank.org/country/ecuador/era5-historical

**Climate Change Knowledge Portal**  
For Development Practitioners and Policy Makers

Climate Change and Development Understand Climate Data Explore Our Data [Download Data](#)

LOCATION: Ecuador [Explore other countries](#)

CLIMATE VARIABLE: Humedad relativa

1950 1960 1970 1980 1990 2000 2010 2023

Category	Relative Humidity (%)
1950	84.6
1951	86.08
1952	84.79
1953	85.13
1954	84.3
1955	84.89
1956	86.91
1957	85.58

HIDE Data Table Print Download

Windows Buscar 11:32 04/10/2025

Imagen 3

2025 NASA Space Apps Challenge

# Will It Rain On My Parade?

+ Join the Challenge

Details

Resources

Teams

## Event

2025 NASA Space Apps Challenge

## Difficulty

Intermediate

## Subjects

Coding Data Analysis Data Visualization Forecasting Software  
Weather Web Development

## SUMMARY

If you're planning an outdoor event—like a vacation, a hike on a trail, or fishing on a lake—it would be good to know the chances of adverse weather for the time and location you are considering. There are many types of Earth observation data that can provide information on weather conditions for a particular location and day of the year. Your challenge is to construct an app with a personalized interface that enables users to conduct a customized query to tell them the likelihood of “very hot,” “very cold,” “very windy,” “very wet,” or “very uncomfortable” conditions for the location and time they specify.

*imagen 4*

## How to Access the Hydrology Data Rods Time Series API Using Python

### Overview

This notebook describes accessing the [Hydrology Data Rods Time Series API](#) using Python. It queries a grid nearest Newton, IL, and queries two months of hourly NLDAS hourly surface and top 0-100cm soil precipitation totals. These variables are then plotted using Matplotlib.

### Prerequisites

This notebook was written using Python 3.9, and uses these libraries and files:

- [requests](#) (version 2.22.0 or later)
- [Pandas](#)
- [Matplotlib](#)
- [Seaborn](#) (optional, used for our plot theme)

### Optional Anaconda Environment YAML:

This notebook can be run using the 'nasa-gesdisc' [YAML file](#) provided in the 'environments' subfolder.

### Import modules

```
[1]: import requests
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
plt.title='Hydrology Data Rods'
plt.style.use('darkgrid')
```

*Imagen 5*

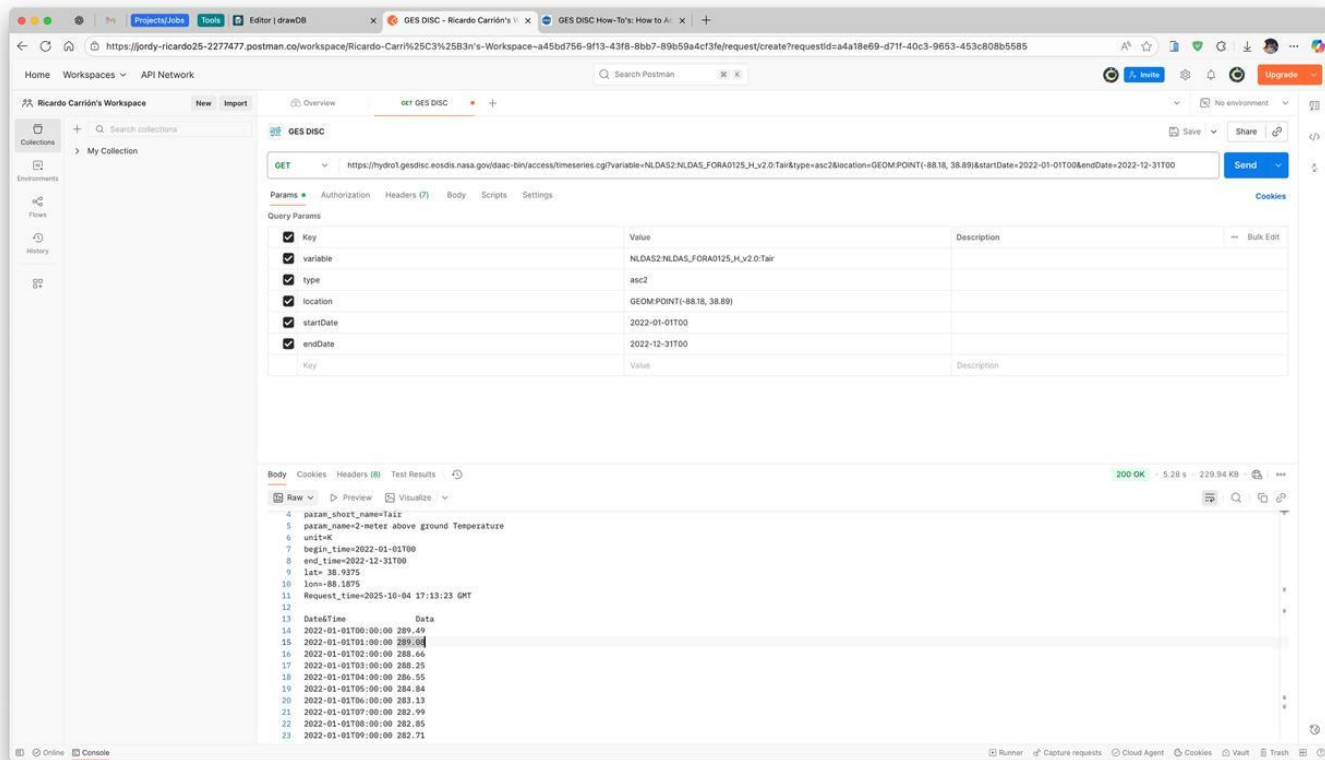


Imagen 6

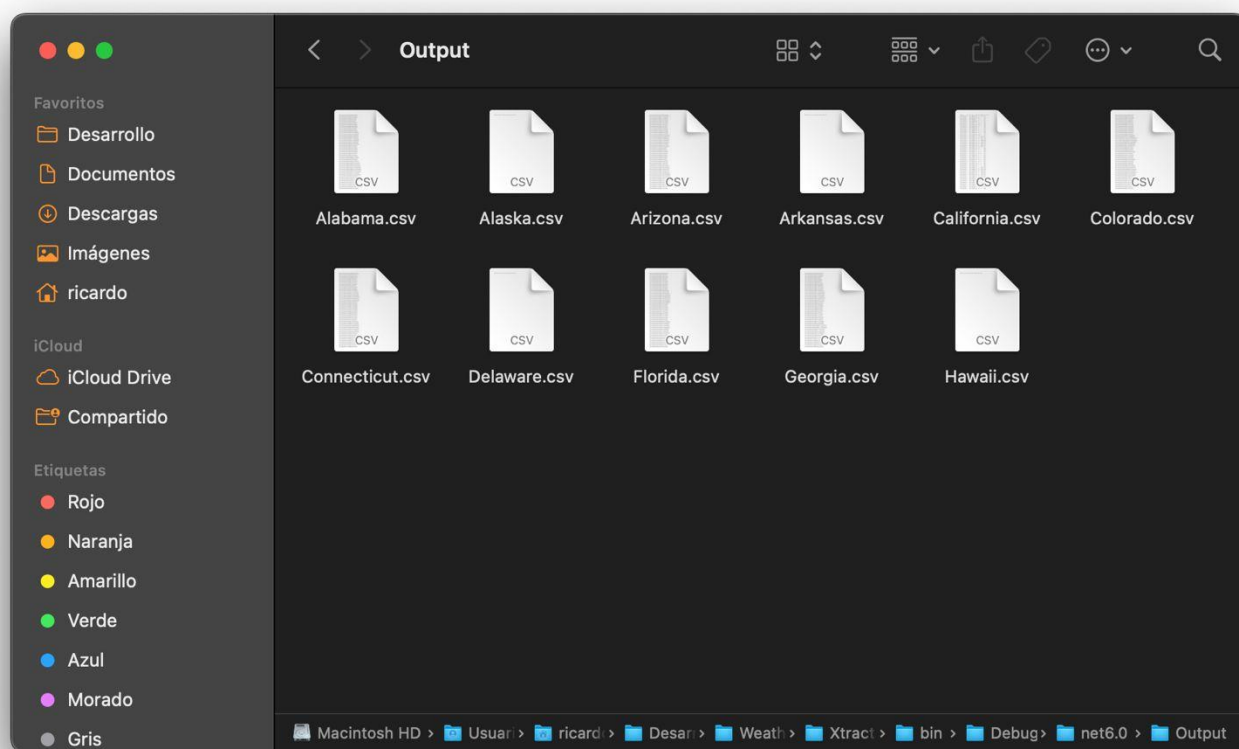


Imagen 7



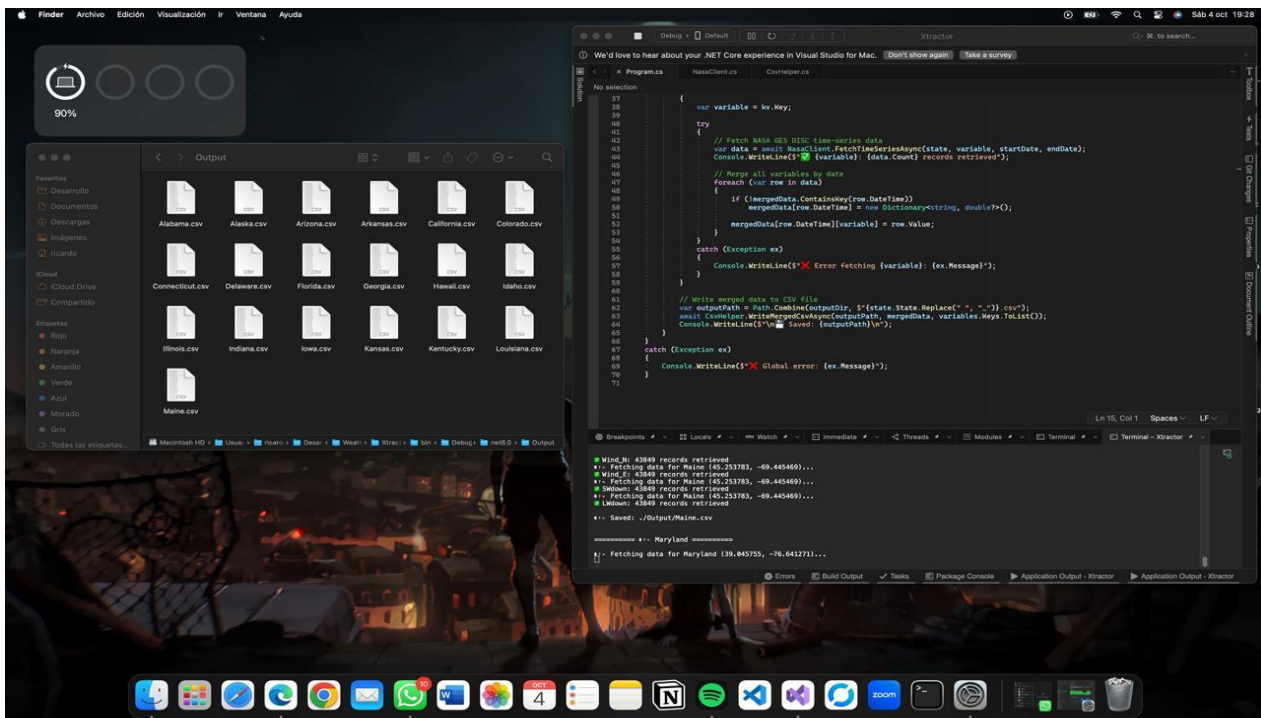


Imagen 8

Alabama.csv							
Date&Time	Rainf	Tair	Qair	Wind_N	Wind_E	SWdown	LWdown
2020-01-01T00:00:00	0	279.17	0.0045406	0.13	3.2	0	239.89
2020-01-01T01:00:00	0	278.25	0.0043671	0.22	3.15	0	239.89
2020-01-01T02:00:00	0	277.33	0.0041933	0.31	3.1	0	239.9
2020-01-01T03:00:00	0	276.41	0.0040192	0.41	3.05	0	236
2020-01-01T04:00:00	0	275.83	0.003923	0.05	2.9	0	236
2020-01-01T05:00:00	0	275.26	0.0038267	-0.32	2.75	0	236.01
2020-01-01T06:00:00	0	274.68	0.00373	-0.68	2.6	0	239.82
2020-01-01T07:00:00	0	274.27	0.0036904	-0.58	2.56	0	239.82
2020-01-01T08:00:00	0	273.86	0.0036506	-0.48	2.52	0	239.82
2020-01-01T09:00:00	0	273.45	0.0036108	-0.39	2.48	0	282.78
2020-01-01T10:00:00	0	273.9	0.0037078	0.05	1.6	0	282.78
2020-01-01T11:00:00	0	274.34	0.0038049	0.48	0.73	0	282.77
2020-01-01T12:00:00	0	274.78	0.0039019	0.91	-0.14	0	243.98
2020-01-01T13:00:00	0	276.13	0.0040135	0.74	-0.1	16.856	243.98
2020-01-01T14:00:00	0	277.47	0.004125	0.57	-0.06	155.784	243.97
2020-01-01T15:00:00	0	278.82	0.0042367	0.4	-0.01	294.88	272.91
2020-01-01T16:00:00	0	280.86	0.0042898	0.67	-0.4	397.632	272.9
2020-01-01T17:00:00	0	282.9	0.0043431	0.94	-0.78	486.35	272.9
2020-01-01T18:00:00	0	284.95	0.0043964	1.21	-1.16	512.114	286.6
2020-01-01T19:00:00	0	285.68	0.0043551	1.81	-0.79	468.175	286.6
2020-01-01T20:00:00	0	286.41	0.0043139	2.4	-0.41	380.262	286.6
2020-01-01T21:00:00	0	287.14	0.0042727	3	-0.04	198.32	334.27
2020-01-01T22:00:00	0	286.05	0.0044939	2.85	-0.46	94.704	334.28
2020-01-01T23:00:00	0	284.96	0.0047154	2.7	-0.88	0	334.28
2020-01-02T00:00:00	0	283.87	0.0049369	2.56	-1.3	0	331.81
2020-01-02T01:00:00	0	283.72	0.0050272	2.45	-0.92	0	331.81
2020-01-02T02:00:00	0	283.58	0.0051176	2.35	-0.54	0	331.81
2020-01-02T03:00:00	0	283.43	0.0052079	2.25	-0.15	0	355.33
2020-01-02T04:00:00	0	283.39	0.0053353	2.54	-0.44	0	355.33
2020-01-02T05:00:00	0	283.35	0.0054623	2.83	-0.72	0	355.33
2020-01-02T06:00:00	0	283.3	0.0055897	3.12	-1	0	361.18
2020-01-02T07:00:00	0	283.13	0.0060012	2.96	-1.43	0	361.18
2020-01-02T08:00:00	0.0188	282.95	0.0064132	2.8	-1.87	0	361.18
2020-01-02T09:00:00	0.0388	282.78	0.0068248	2.64	-2.3	0	363.85
2020-01-02T10:00:00	0	283.08	0.0070144	2.63	-3.6	0	363.85
2020-01-02T11:00:00	0.0064	283.37	0.007204	2.62	-4.89	0	363.85
2020-01-02T12:00:00	0.0752	283.67	0.0073936	2.61	-6.19	0	377.31
2020-01-02T13:00:00	2.9972	284.18	0.0076361	3.56	-6.34	4.984	377.3
2020-01-02T14:00:00	1.9828	284.69	0.0078792	4.52	-6.49	47.296	377.3

Imagen 9

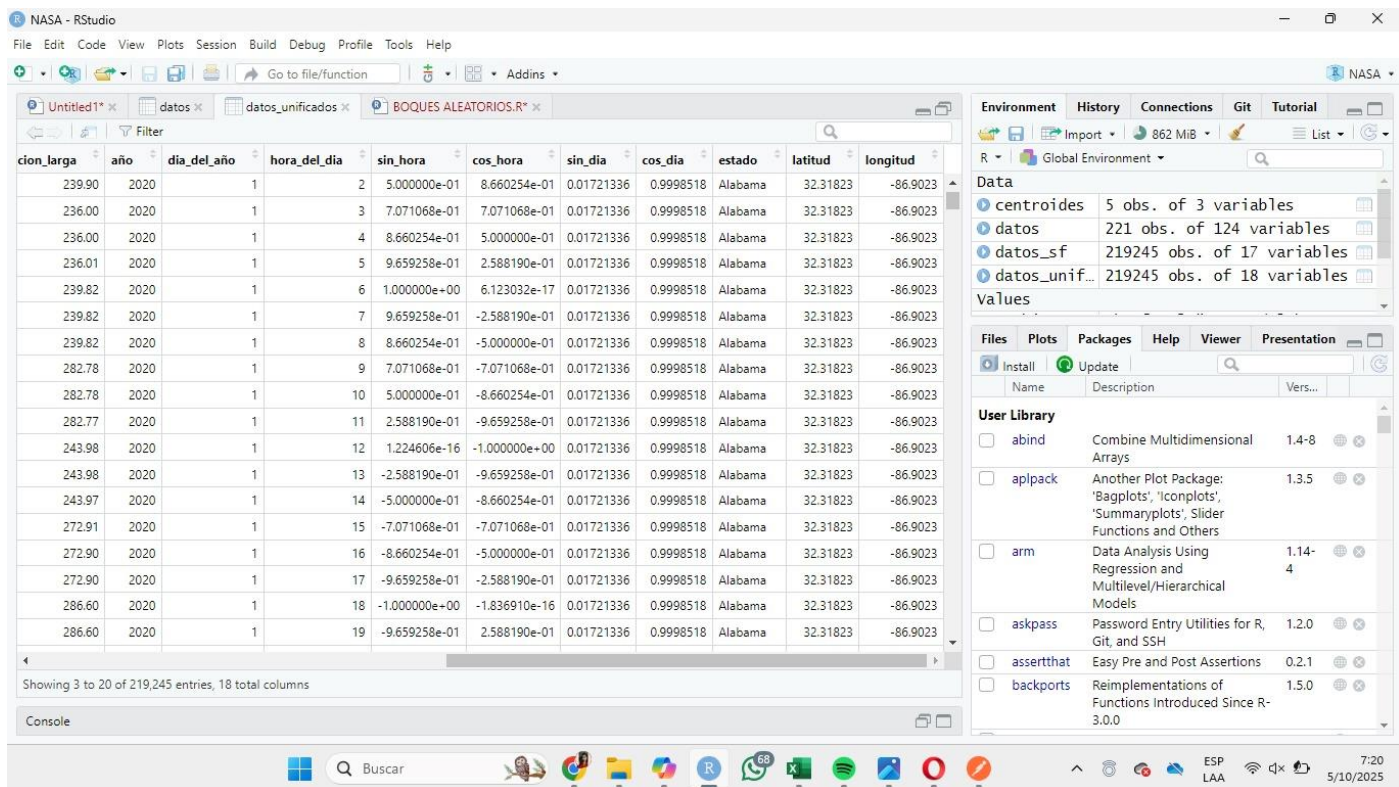


Imagen 10

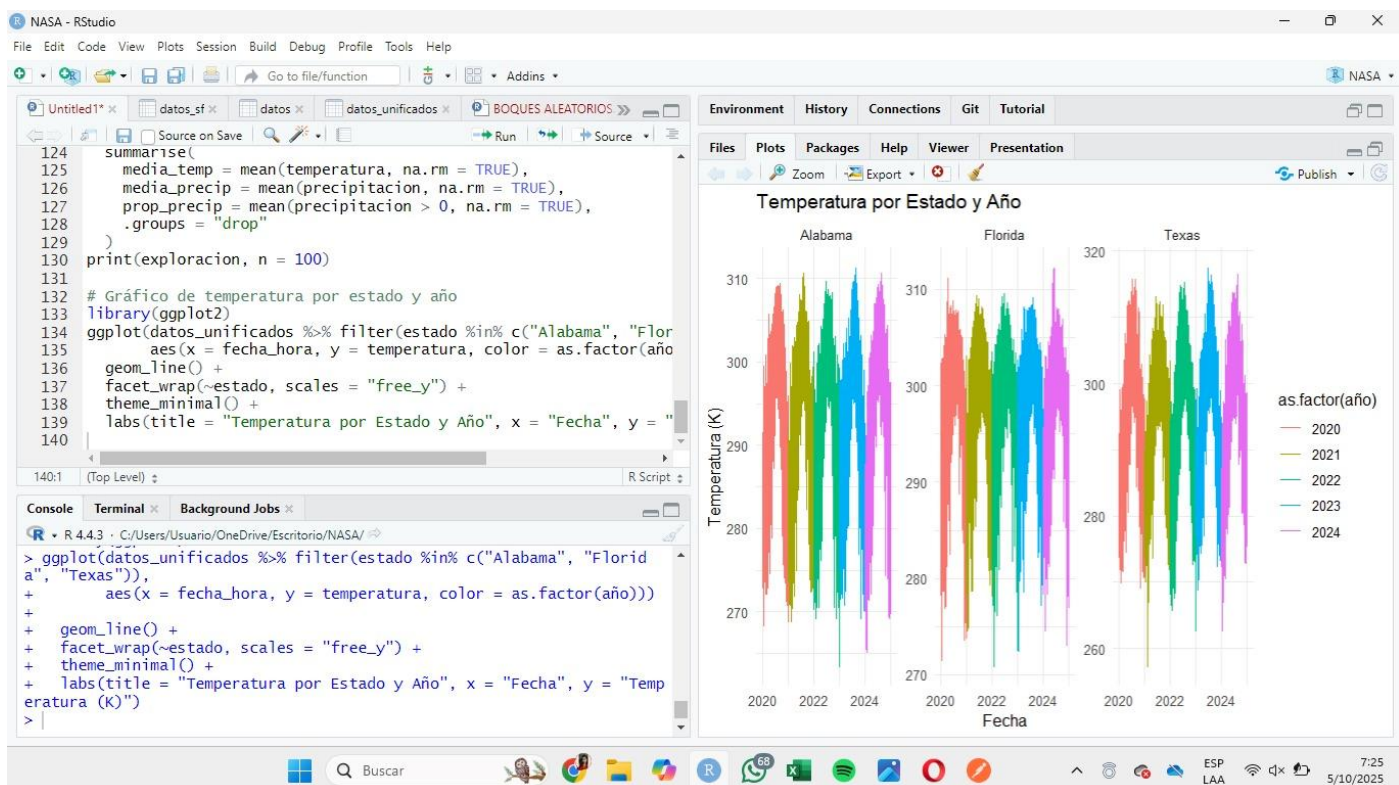


Imagen 11



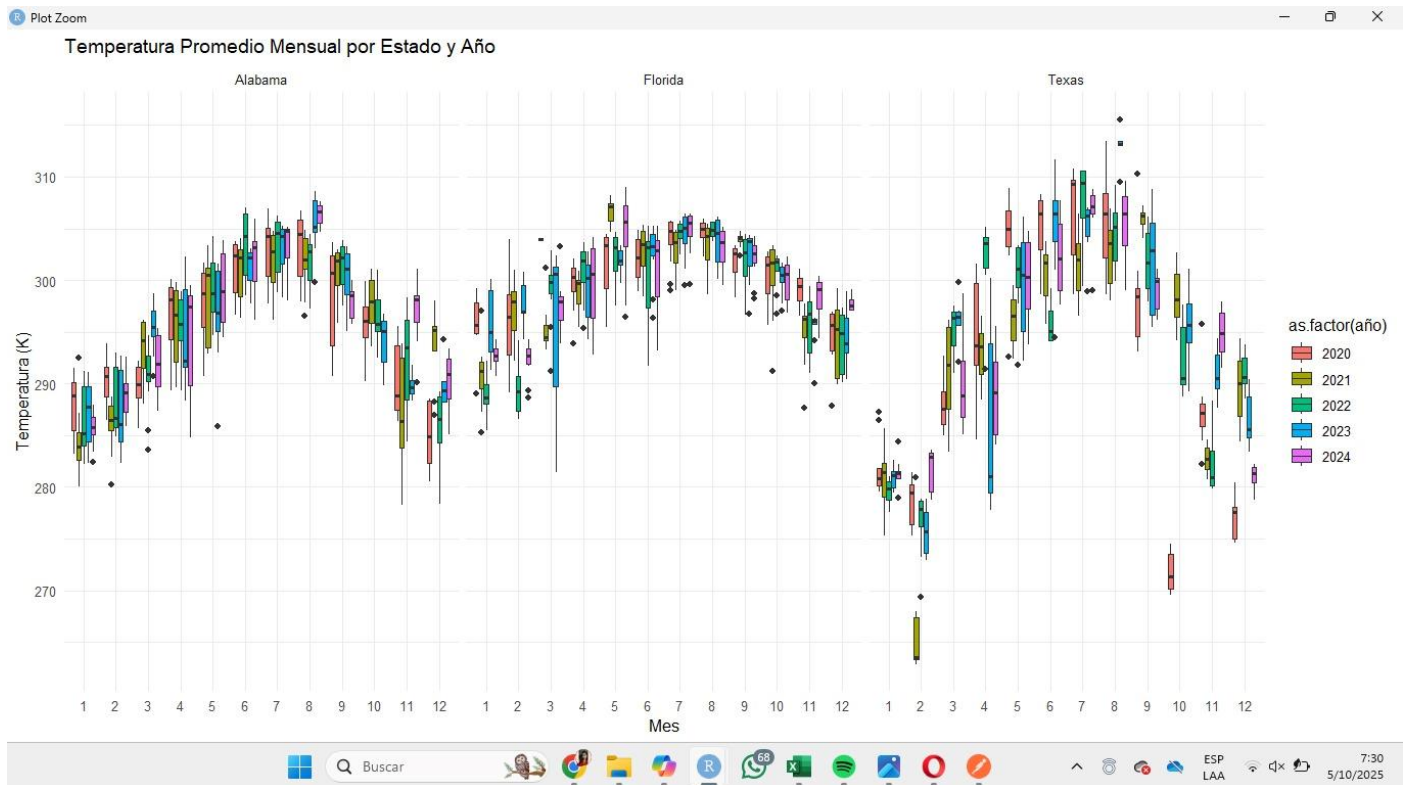


Imagen 12

## Predicción de Condiciones Climáticas

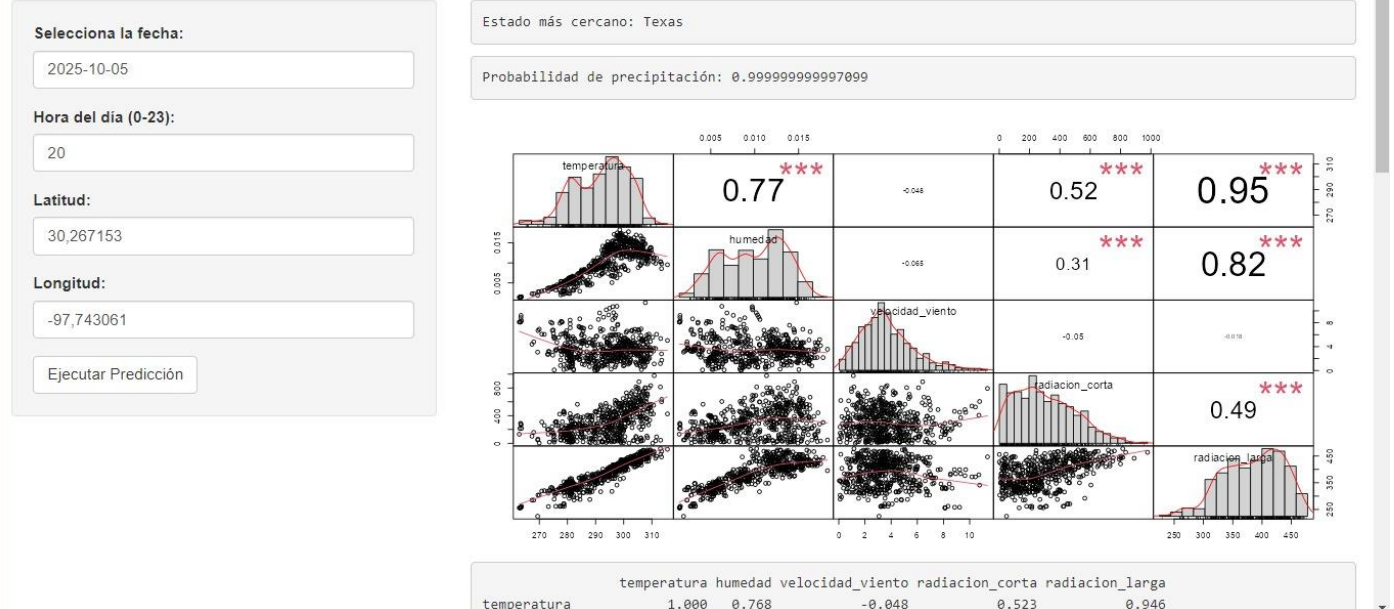
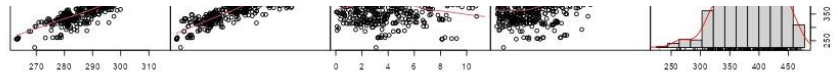


Imagen 13



	temperatura	humedad	velocidad_viento	radiacion_corta	radiacion_larga
temperatura	1.000	0.768	-0.048	0.523	0.946
humedad	0.768	1.000	-0.065	0.307	0.817
velocidad_viento	-0.048	-0.065	1.000	-0.050	-0.018
radiacion_corta	0.523	0.307	-0.050	1.000	0.488
radiacion_larga	0.946	0.817	-0.018	0.488	1.000

Call:  
glm(formula = formula, family = binomial(link = "logit"), data = datos\_estado)

Coefficients:

	Estimate	Std. Error	z value	Pr(> z )
(Intercept)	2.657e+01	1.313e+06	0	1
temperatura	1.080e-10	5.413e+03	0	1
humedad	-7.971e-07	8.833e+06	0	1
velocidad_viento	5.145e-10	7.718e+03	0	1
radiacion_corta	-2.469e-12	2.162e+02	0	1
radiacion_larga	1.509e-11	1.091e+03	0	1
sin_hora	-2.516e-09	1.200e+05	0	1
cos_hora	1.247e-10	3.381e+04	0	1
sin_dia	-1.776e-09	2.684e+04	0	1
cos_dia	-3.406e-09	5.950e+04	0	1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 0.0000e+00 on 560 degrees of freedom

Imagen 14

```

var mergedData = new Dictionary<string, Dictionary<string, double?>>();
foreach (var kv in variables)
{
    var variable = kv.Key;
    try
    {
        // Fetch NASA GES DISC time-series data
        var data = await NasaClient.FetchTimeSeriesAsync(state, variable, startDate, endDate);
        Console.WriteLine($"✅ {variable}: {data.Count} records retrieved");
        // Merge all variables by date
        foreach (var row in data)
        {
            if (!mergedData.ContainsKey(row.DateTime))
            {
                mergedData[row.DateTime] = new Dictionary<string, double?>();
            }
            mergedData[row.DateTime][variable] = row.Value;
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine($"❌ Error fetching {variable}: {ex.Message}");
    }
}

// Write merged data to CSV file
var outputPath = Path.Combine(OutputDir, $"{state.State.Replace(" ", "_")}.csv");
await CsvHelper.WriteMergedCsvAsync(outputPath, mergedData, variables.Keys.ToList());
Console.WriteLine($"💾 Saved: {outputPath}");
}
catch (Exception ex)
{
    Console.WriteLine($"❌ Global error: {ex.Message}");
}
}

```

```

*~ Fetching data for Alabama (32.318231, -86.902298)...
*~ Wind_E: 43849 records retrieved
*~ SWdown: 43849 records retrieved
*~ Fetching data for Alabama (32.318231, -86.902298)...
*~ LWdown: 43849 records retrieved
*~ Saved: ../Output/Alabama.csv

*~ Alaska
*~ Fetching data for Alaska (63.588753, -154.493062)...
*~ Rainf: 0 records retrieved
*~ Fetching data for Alaska (63.588753, -154.493062)...

```

Imagen 15

```

1  using Xtractor.Clients;
2  using Xtractor.Helpers;
3
4  // Define the NASA variables to extract
5  var variables = new Dictionary<string, string>
6  {
7      { "Rainf", "Rainf (Precipitation) [kg/m²/s]" },
8      { "Tair", "Tair (Air temperature at 2 m) [K]" },
9      { "Qair", "Qair (Specific humidity) [kg/kg]" },
10     { "Wind_M", "Wind_M (Northward wind component) [m/s]" },
11     { "Wind_E", "Wind_E (Eastward wind component) [m/s]" },
12     { "SWdown", "SWdown (Downward shortwave radiation) [W/m²]" },
13     { "LWdown", "LWdown (Downward longwave radiation) [W/m²]" }
14 };
15
16 // Time range for the dataset
17 var startDate = "2020-01-01T00";
18 var endDate = "2025-01-01T00";
19
20 // Output directory
21 var outputDir = "/Output";
22 Directory.CreateDirectory(outputDir);
23
24 try
25 {
26     // Load U.S. states and coordinates from CSV
27     var states = await CsvHelper.ReadCsvAsync("../Assets/USA_States.csv");
28
29     foreach (var state in states)
30     {
31         Console.WriteLine($"***** {state.State} *****\n");
32
33         // Dictionary: DateTime + values by variable
34         var mergedData = new Dictionary<string, Dictionary<string, double?>>();
35
36         foreach (var kv in variables)
37         {
38             var variable = kv.Key;
39
40             try

```

\*\*\*\*\* Alabama \*\*\*\*\*

Fetching data for Alabama (32.318231, -86.902298)...

Rainf: 43849 records retrieved

Fetching data for Alabama (32.318231, -86.902298)...

Build successful.

Imagen 16

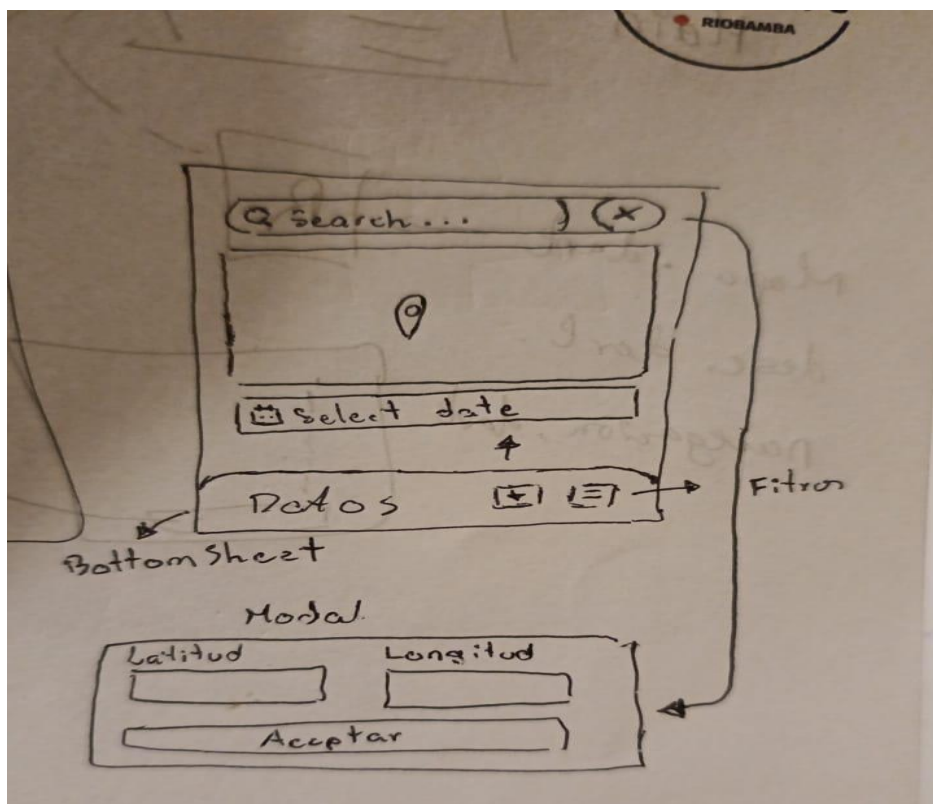


Imagen 17

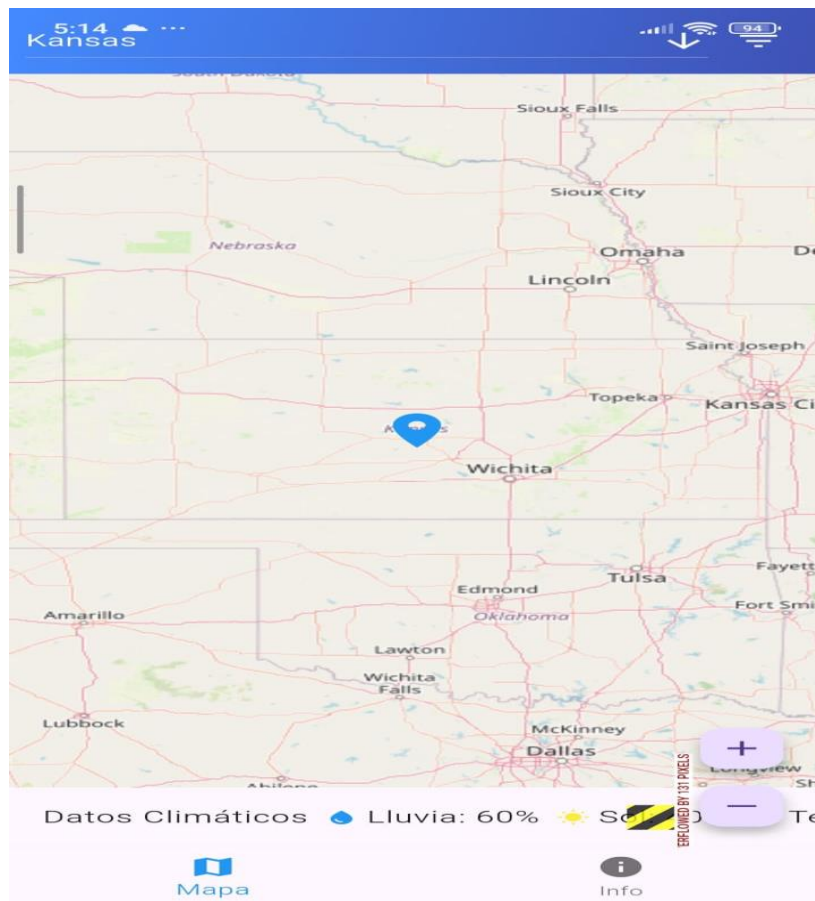


Imagen 18

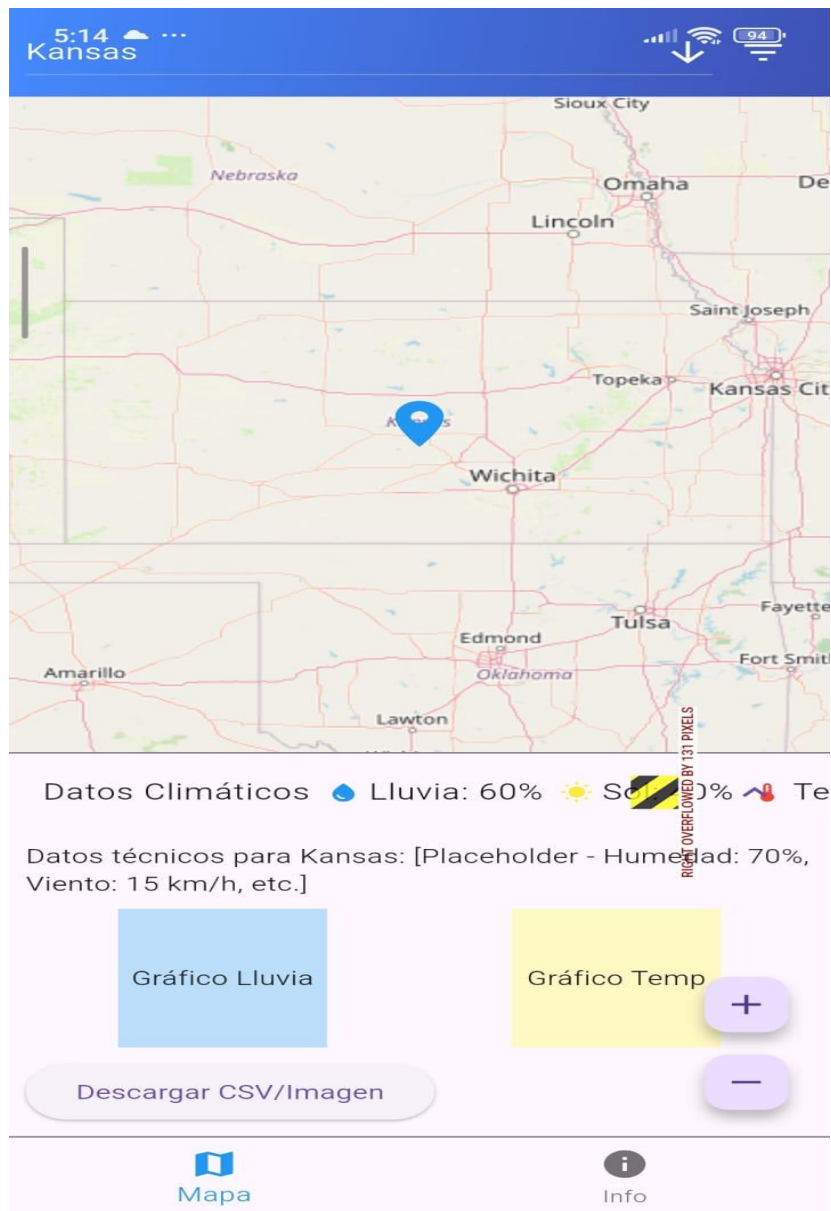


Imagen 19



Imagen 20



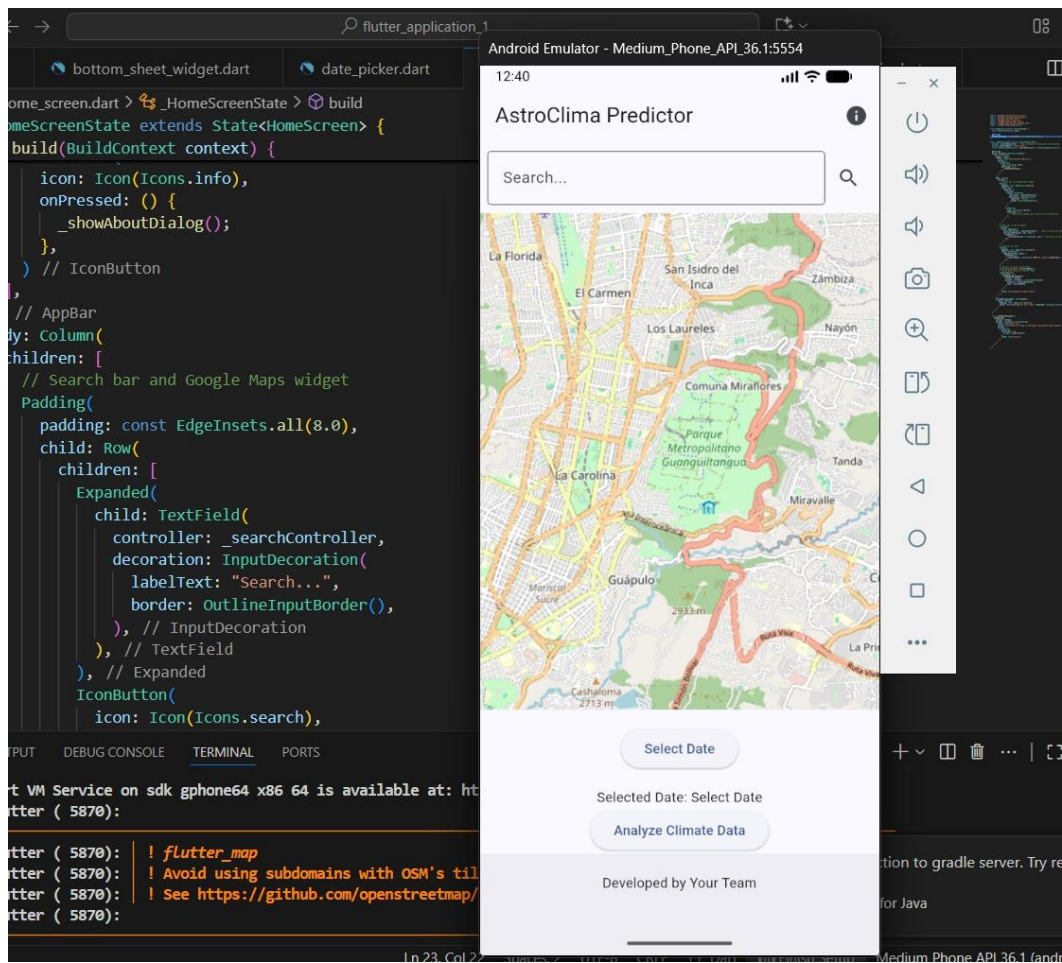


Imagen 21

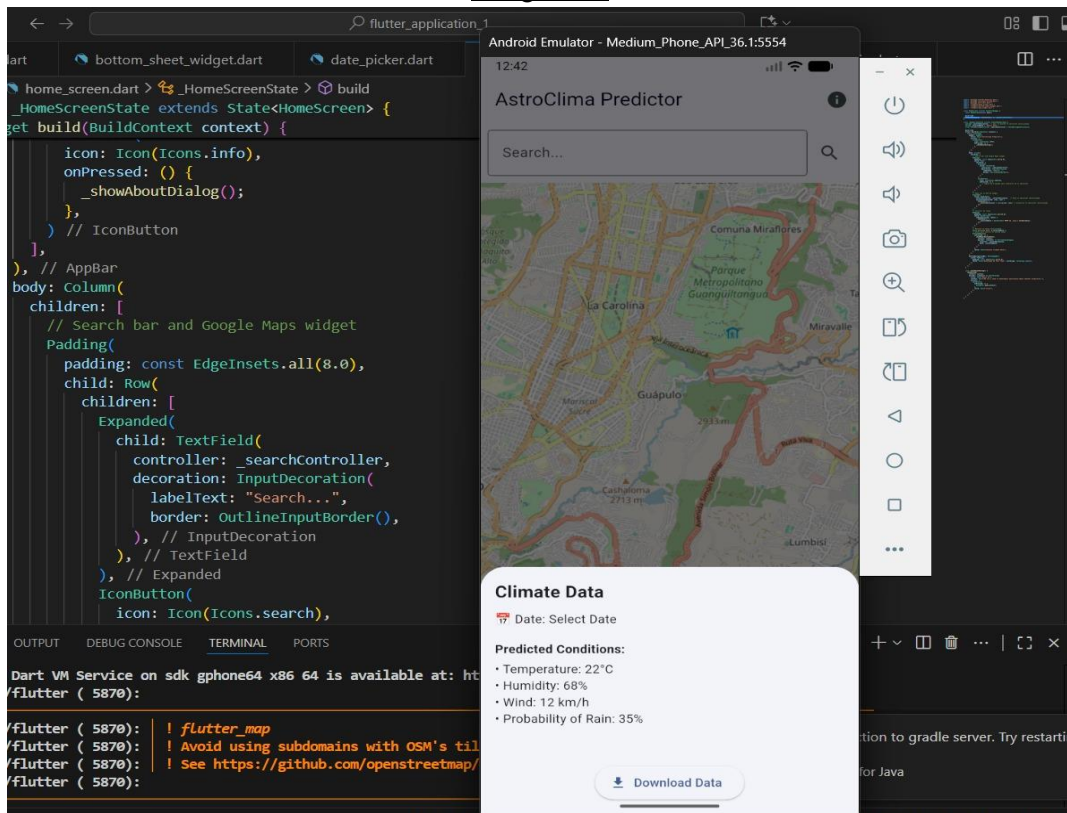


Imagen 22

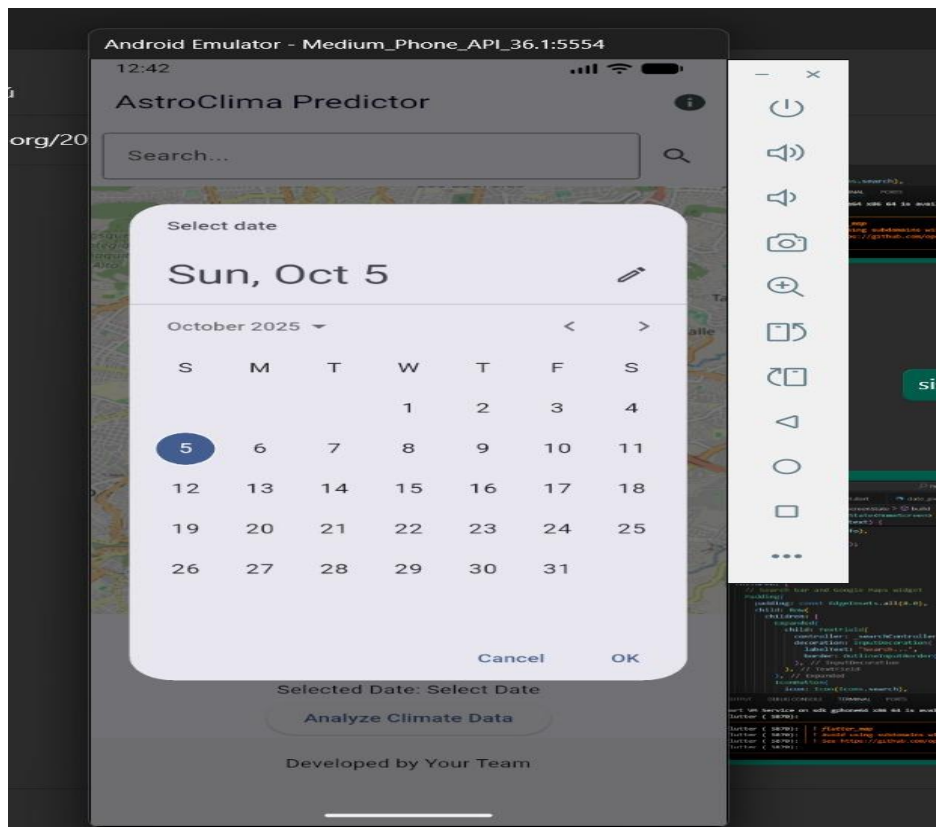


Imagen 23