

# NUESTRO PROYECTO



# ROLES

ROLES	PERSONA
PRODUCT OWNER	LEMOS SOFÍA
SCRUM MASTER	DAINE LUCAS
TEAM MEMBER	BARRIOS PABLO
TEAM MEMBER	FUENTES SERGIO
TEAM MEMBER	ADRIEL

# SOBRE EL PROYECTO

Consta de una página web que permite al usuario ingresar una ciudad con la finalidad de ver la temperatura / el tiempo correspondiente. Esto lo hace mediante la conexión con una API. También, el programa tiene la opción de variar el Sistema de Medidas que utiliza (Celcius, Farenheit y Kelvin) para que el usuario elija el de su preferencia.

Todos los datos son recopilados para posteriormente mostrarle al usuario el historial de temperaturas en una determinada ciudad y su posible pronóstico en días posteriores.

La interfaz va a ser pensada para ser atractiva pero, a su vez, simple e intuitiva. Así ampliaremos mucho más el público objetivo.



## SPRINT 1 – NUESTROS CRITERIOS DE ACEPTACION

1

Cuando introduzco el nombre de una ciudad válida, veo el clima actual de esa ciudad (temperatura, humedad, viento, etc.).

2

Si introduzco una ciudad inexistente o incorrecta, la aplicación muestra un mensaje de error adecuado.



**SPRINT 1**  
SUGERENCIA

**HISTORIA 1**  
Consultar el clima  
de una ciudad.

**HISTORIA 2**  
Menú de  
consultas

**HISTORIA 3**  
Manejar errores  
de API.

**TAREAS**

1. Crear la Base de Datos. (E)
2. Crear el Buscador de Ciudades.
3. Vincularlos y mostrar error de ser necesario.

**TAREAS**

1. Crear un Menú Recursivo con opción de “Salir”.

**TAREAS**

1. Hacer que el programa reintente la conexión.
2. Crear un mensaje de “Error”.

**EPIC**

Implementación de la consulta del clima y del menú interactivo.



**SPRINT 2**  
SUGERENCIA

**HISTORIA 4**  
Cambiar entre  
Celsius y  
Fahrenheit

**TAREAS**

1. Mostrar por pantalla la opción de elegir una medida.
2. Hacer que esa elección se mantenga hasta que se indique lo contrario.

**EPIC**

Implementación de la funcionalidad de cambio de unidades de medida y el historial de consultas.

**HISTORIA 5**  
Historial de  
ciudades y climas

**TAREAS**

1. Guardar el Historial de búsqueda en la Base de Datos.
2. Mostrar por pantalla cuando el usuario lo pida.



## SPRINT 3 SUGERENCIA

**HISTORIA 6**  
Ver el pronóstico  
de 5 días  
posteriores

**HISTORIA 7**  
Que se ejecute en  
Docker

### TAREAS

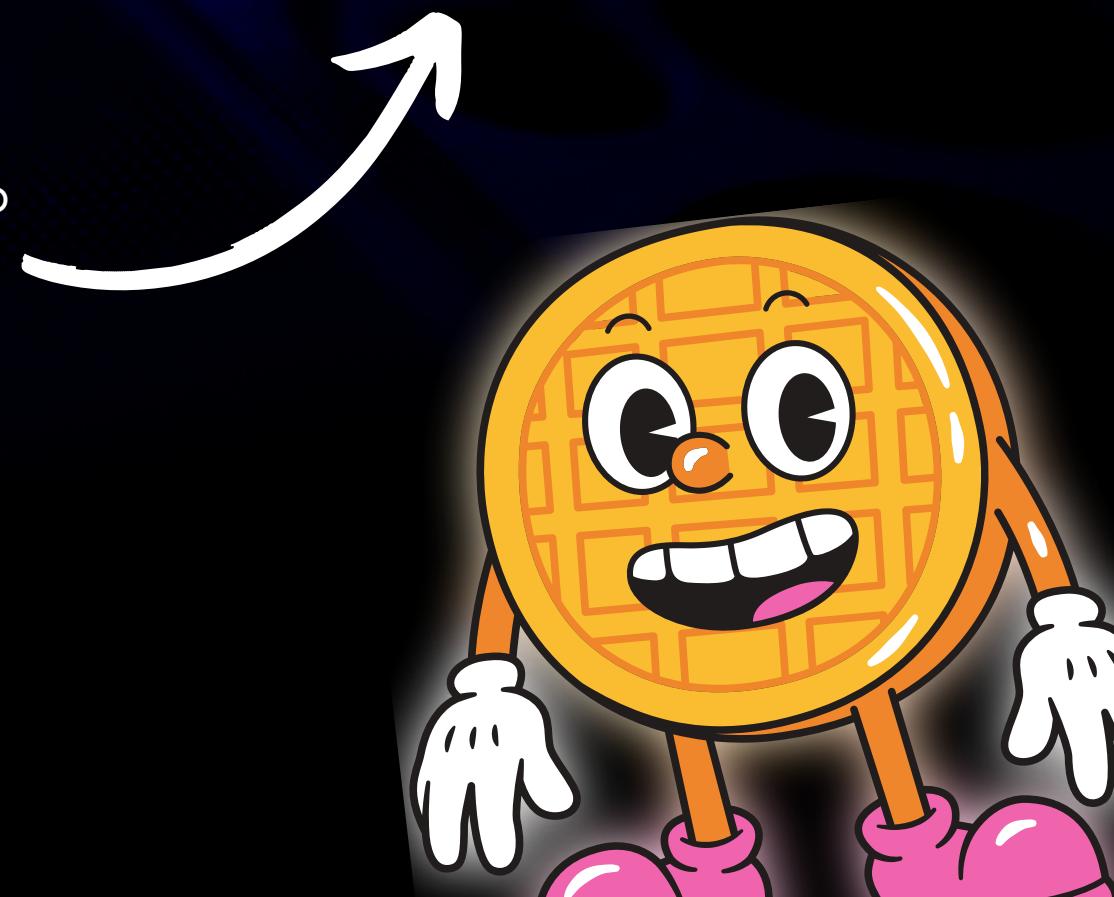
1. Darle la opción al usuario de consultar por días posteriores.
2. Cargar en la Base de Datos.
3. Que sea Capaz de mostrarse por pantalla.
4. Mostrar en caso de haber un “Error”.

### EPIC

Implementación del pronóstico  
del clima y contenedorización  
de la aplicación.

### TAREAS

1. Coordinar con todo el equipo para usar el mismo tipo de contenedor.





# NUESTRO BACKLOG

Buscar BP EI LD SL Versión ▾ Epic ▾ Filtros rápidos ▾ Insights Ver configuración

Tablero Sprint 1 7 oct - 21 oct (3 incidencias) 0 0 0 Completar sprint ⋮

DETALLE	TIPO	ESTADO	TAREAS POR HACER	ASIGNACIÓN
SPLS-1 Consulta de clima por nombre de ciudad	RELEASE-UNO	GESTIÓN DE CONSULTAS	TAREAS POR HACER	BP
SPLS-2 Investigación y selección de API	RELEASE-UNO	FINALIZADA		EI
SPLS-3 Implementar manejo de errores	RELEASE-UNO	FINALIZADA		LD
SPLS-4 Implementar la lógica de consulta	RELEASE-UNO	FINALIZADA		SL
SPLS-5 Documentación	RELEASE-UNO	EN CURSO		BP
SPLS-6 Pruebas unitarias y funcionales	RELEASE-UNO	EN CURSO		EI
SPLS-7 Implementación de menú interactivo	RELEASE-UNO	GESTIÓN DE CONSULTAS	TAREAS POR HACER	LD
SPLS-8 Gestión de errores por fallos de conexión y API	RELEASE-UNO	GESTIÓN DE ERRORES	TAREAS POR HACER	SL



# NUESTRO CÓDIGO

```
dockerfile > ...
1 FROM python:3
2 WORKDIR /clima
3 COPY requirements.txt ./
```

4 RUN pip install --no-cache-dir -r requirements.txt

5 COPY clima.py .

6 CMD [ "python","./clima.py" ]

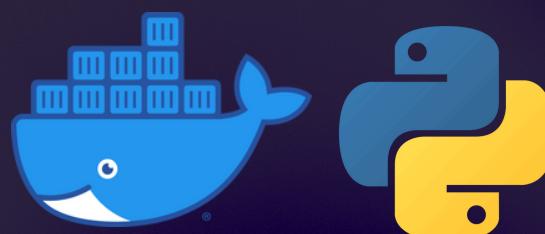
Go Run Terminal Help ← → 🔍 Proyecto Integrador

dockerfile clima.py ✘ .env .gitignore README.md RELEASE.txt requirements.txt

```
clima.py > ...
1 from dotenv import load_dotenv
2 import os
3 import requests
4
5 nombre_ciudad=input("Ingrese la ciudad para obtener su pronóstico: ")
6 nombre_pais=input("Ingrese el pais de la ciudad: ")
7
8 load_dotenv()
9 api= os.getenv('API')
10 print(api)
11 #api="abf47be9918bb6eb6fb7cdd893089636"
12 unidad_de_medida="metric"#tiene distintas variantes, por default es kelvin, y el resto son metric(celsius) e imperial(fahrenheit)
13 url= f"https://api.openweathermap.org/data/2.5/weather?q={nombre_ciudad},{nombre_pais}&appid={api}&units={unidad_de_medida}"
14 #Holis
15 respuesta_api=requests.get(url)
16 #validacion de la respuesta de la api
17 if respuesta_api.status_code == 200:
18     clima=respuesta_api.json()#guardo el archivo en formato json q me permite navegar como si fuese un diccionario por la data
19     print(clima['main'])#obtengo los datos de la llave
20 else:
21     print(f"Error: {respuesta_api.status_code}")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Python

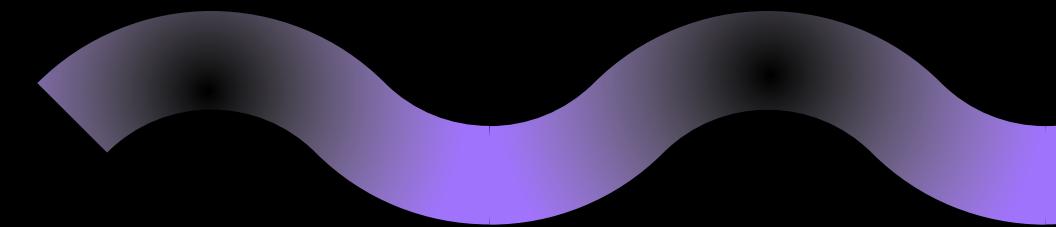
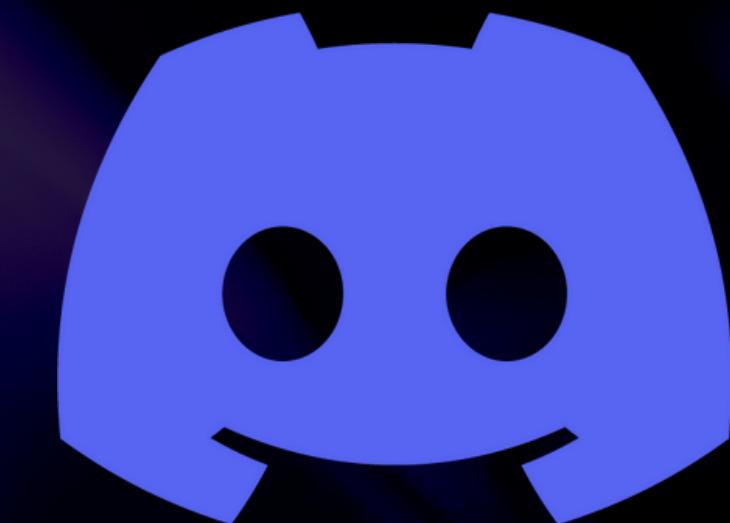
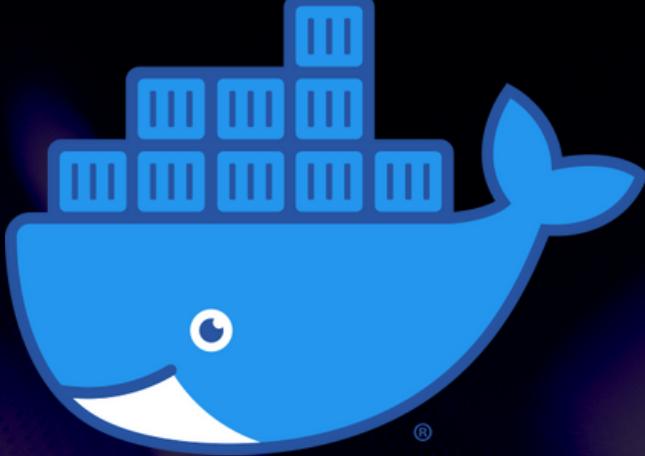


```
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 6 (delta 0), reused 6 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (6/6), 1.74 KiB | 35.00 KiB/s, done.
From https://github.com/ByAkko317/ScrumbL-cookies
 * branch            developingSofi -> FETCH_HEAD
   8e589b4..bc963b9  developingSofi -> origin/developingSofi
Updating 8e589b4..bc963b9
Fast-forward
  Errores1.py | 69 ++++++=====
  1 file changed, 69 insertions(+)
  create mode 100644 Errores1.py
PS C:\Users\Pablo\OneDrive\Escritorio\UTN\Materias\2024\ORG EMPRESARIAL 2024\Proyecto Integrador>
```

0 Live Share

Ln 9, Col 22 Spaces: 4 UTF-8 CRLF

# HERRAMIENTAS UTILIZADAS





# GRACIAS POR SU ATENCIÓN

