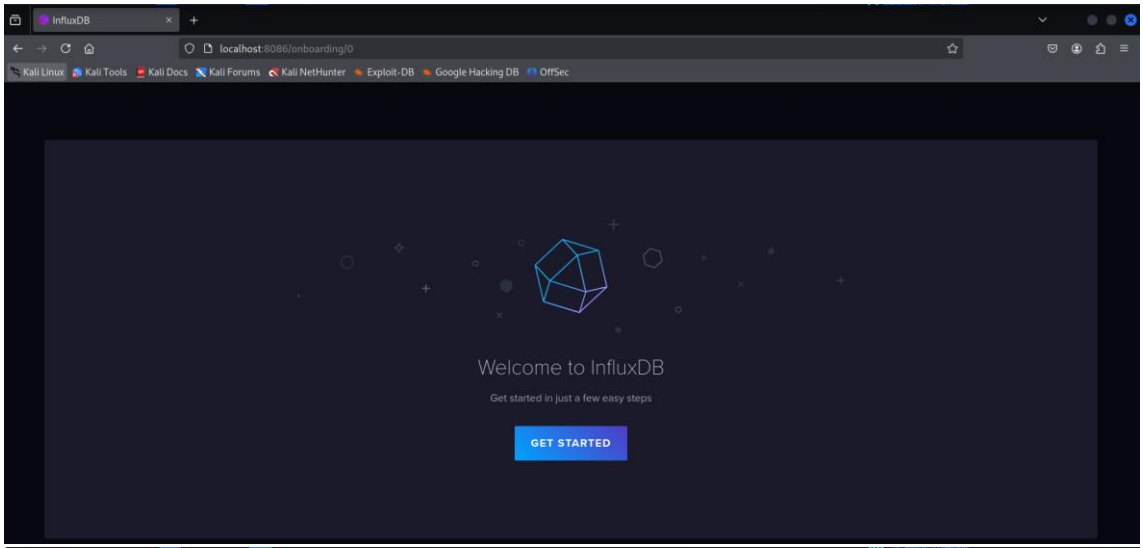
Que hacer

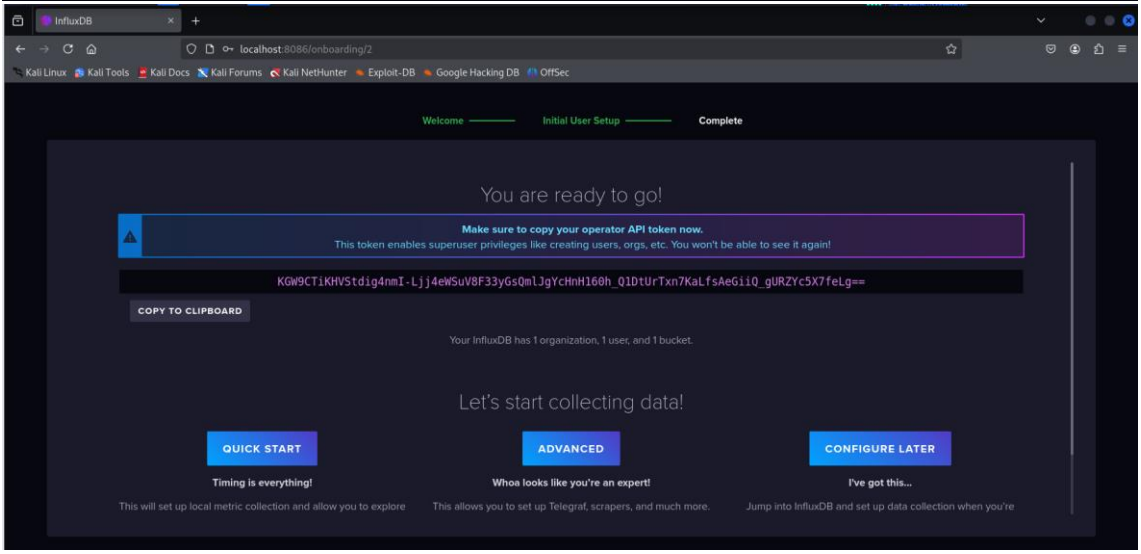
- Primera semana
    - Analizar la estructura de los componentes utilizados en los códigos de Python que te proporcioné.
    - Crear una base de datos InfluxDB usando Docker.

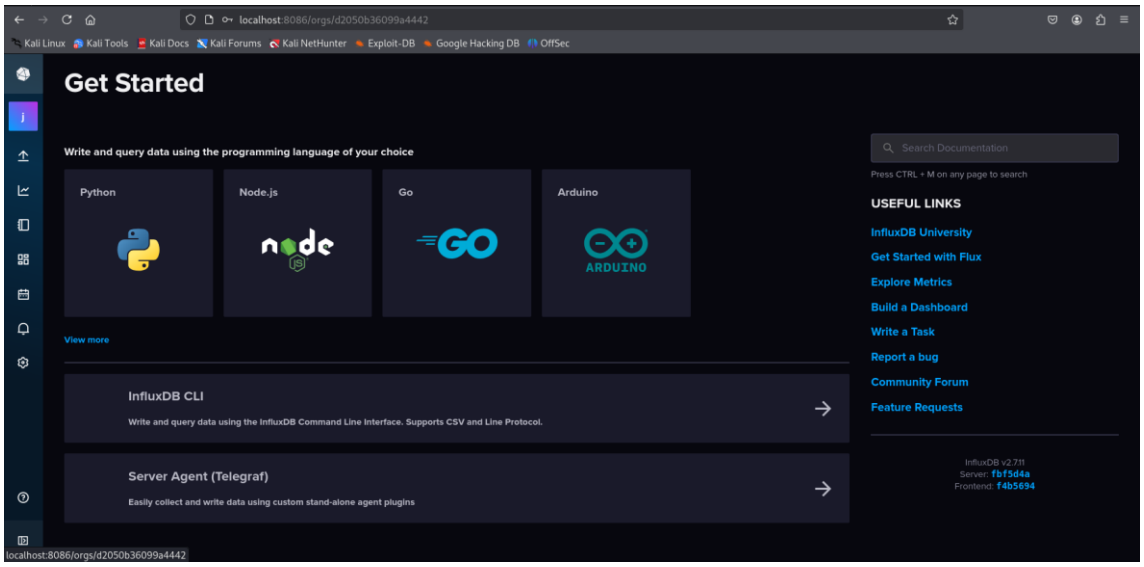
```
┌──(kali㉿joelhersie)-[~]
└─$ sudo docker pull influxdb:2.7
```

```
┌──(kali㉿joelhersie)-[~]
└─$ sudo docker run -d --name=influxdb -p 8086:8086 -v influxdb_data:/var/lib/influxdb2 influxdb:2.7
368cde8add7b76339ffa3ec0df36f277b0f6b64b5204ead84db04389b95a73af
```

```
┌──(kali㉿joelhersie)-[~]
└─$ sudo docker start influxdb
influxdb
```







- Probar todos los códigos que se encuentran en la carpeta InfluxDB.
  - Recuerda que la suscripción debe tener código ejecutado en el servidor.

sensor.py

```
┌──(kali㉿joelhersie)-[~/Downloads/Testing Code]
└─$ source prueba/bin/activate

┌──(prueba)─(kali㉿joelhersie)-[~/Downloads/Testing Code]
└─$ python3 sensor.py
Temperatura enviada: 21.99°C
```

WebSocketServer.py

```
┌──(prueba)─(kali㉿joelhersie)-[~/Downloads/Testing Code]
└─$ pip install websockets
```

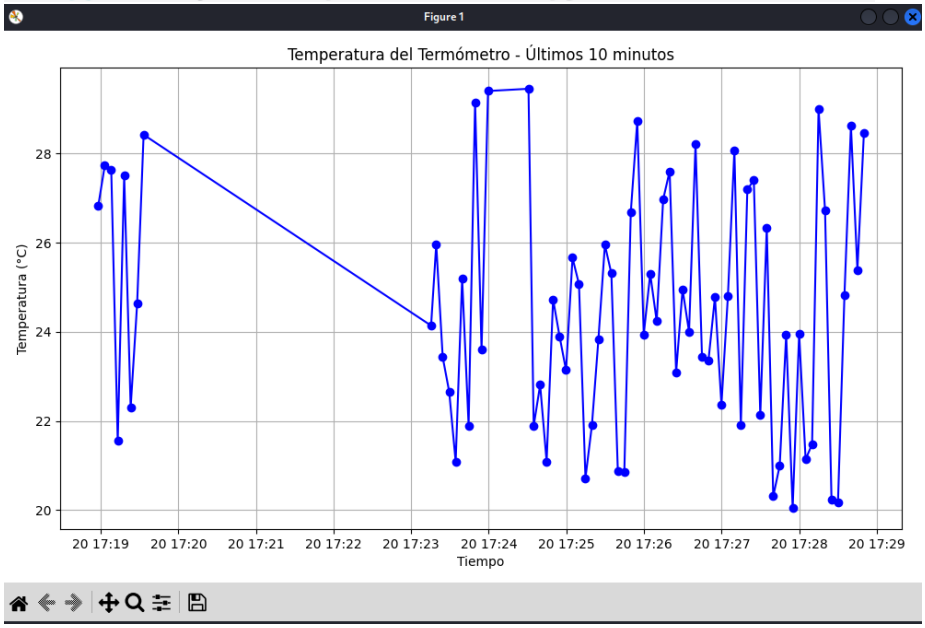
```
┌──(prueba)─(kali㉿joelhersie)-[~/Downloads/Testing Code]
└─$ python3 WebSocketServer.py
Servidor WebSocket iniciado en ws://0.0.0.0:8765
```

LeerWebSocket.py

```
┌──(prueba)─(kali㉿joelhersie)-[~/Downloads/Testing Code]
└─$ python3 LeerWebSocket.py
Conectado al servidor WebSocket
```

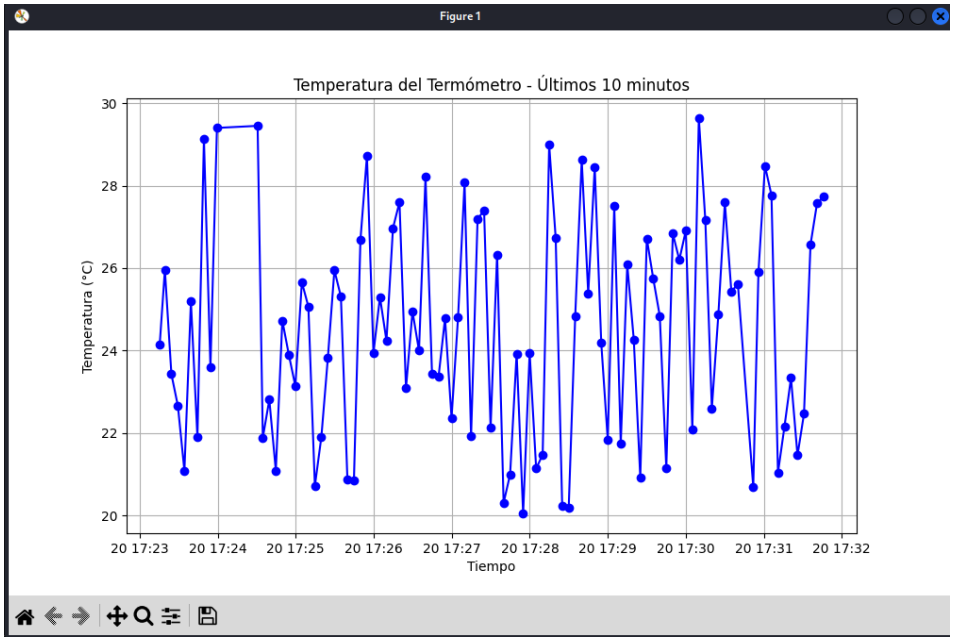
Plot_temperature_data.py

```
┌──(prueba)─(kali㉿joelhersie)-[~/Downloads/Influxdb]
└─$ python3 plot_temperature_data.py
```



plot_temperature_data_real_time.py

```
┌──(prueba)─(kali㉿joelhersie)-[~/Downloads/Influxdb]
└─$ python3 plot_temperature_data_real_time.py
```



Media.py

```
┌──(prueba)─(kali㉿joelhersie)-[~/Downloads/Testing Code]
└─$ python3 media.py
Media de temperatura (últimos 2 minutos): 24.954166666666666°C
```

Lector.py

```
┌──(prueba)─(kali㉿joelhersie)-[~/Downloads/Testing Code]
└─$ python3 lector.py
Última temperatura registrada: 20.48°C
```

- Crear un video mostrando que todo funciona correctamente.

  # VIDEO PRIMERA SEMANA

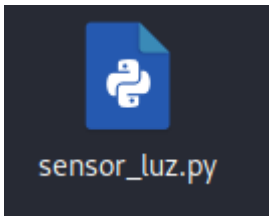
  https://youtu.be/4s1u8RrRBP8

- Segunda semana

  - Simular una granja IoT con Python, InfluxDB y WebSocket usando componentes.
  - Crear 2-3 sensores.


```python
import random
import time
from influxdb_client import Point
from influxdb_client.client.write_api import SYNCHRONOUS
from connection_component import InfluxDBConnection

def simulate_humidity_data():
    """Simula el envio de datos de humedad cada 5 segundos."""
    connection = InfluxDBConnection(
        url="http://localhost:8086",
        token="NcvCQPUwfIujXntIuLFfttKVZMpfCLvCSEU10dQwfxJnAd9SEsIpu1NZkStPCiFaYW2oxWWvxjARzGkCdBC3qg==",
        org="jhs",
        bucket="jhs"
    )

    client = connection.get_client()
    write_api = connection.get_write_api(client)

    try:
        while True:
            # Genera datos de humedad aleatoria entre 30% y 70%
            humidity = round(random.uniform(30, 70), 2)
            point = Point("hygrometer").field("humidity", humidity)
            write_api.write(bucket=connection.bucket, org=connection.org, record=point)
            print(f"Humedad enviada: {humidity}%")
            time.sleep(5)  # Simula el envio cada 5 segundos
    except KeyboardInterrupt:
        print("Simulación detenida.")

if __name__ == "__main__":
    simulate_humidity_data()

```

```
┌──(prueba)─(kali㉿joelhersie)-[~/Downloads/Influxdb]
└─$ python3 sensor_temp.py
Humedad enviada: 36.26%
Humedad enviada: 50.62%
Humedad enviada: 61.71%
```

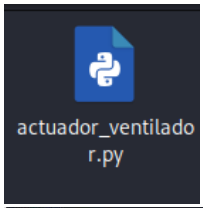


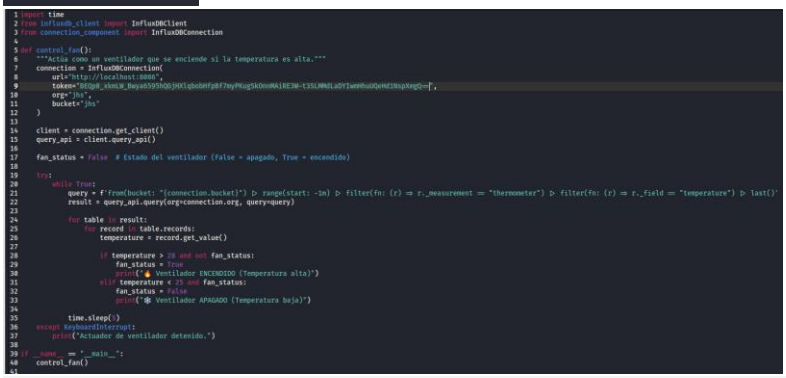

```python
import random
import time
from influxdb_client import Point
from influxdb_client.client.write_api import SYNCHRONOUS
from connection_component import InfluxDBConnection

def simulate_light_data():
    """Simula el envío de datos de luminosidad cada 5 segundos."""
    connection = InfluxDBConnection(
        url="http://localhost:8086",
        token="NcvCQPUwfIujXntIuLFfttKVZMpfCLvCSEU10dQwfxJnAd9SEsIpu1NZkStPCiFaYW2oxWWvxjARzGkCdBC3qg==",
        org="jhs",
        bucket="jhs"
    )

    client = connection.get_client()
    write_api = connection.get_write_api(client)

    try:
        while True:
            # Simula la luz entre 100 y 1000 lux
            light = round(random.uniform(100, 1000), 2)
            point = Point("light_sensor").field("lux", light)
            write_api.write(bucket=connection.bucket, org=connection.org, record=point)
            print(f"Luminosidad enviada: {light} lux")
            time.sleep(5)
    except KeyboardInterrupt:
        print("Simulación detenida.")

if __name__ == "__main__":
    simulate_light_data()
```

```
┌──(prueba)─(kali㉿joelhersie)-[~/Downloads/Influxdb]
└─$ python3 sensor_luz.py
Luminosidad enviada: 302.92 lux
Luminosidad enviada: 201.05 lux
Luminosidad enviada: 880.54 lux
```

- Simular 2 actuadores con Python (esto debe ser una suscripción).

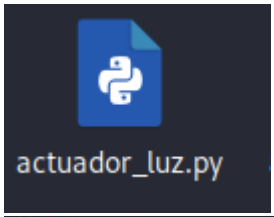
Actuador temperatura:





```
┌──(prueba)─(kali㉿joelhersie)-[~/Downloads/Influxdb]
└─$ python3 actuador_ventilador.py
🔥 Ventilador ENCENDIDO (Temperatura alta)
❄ Ventilador APAGADO (Temperatura baja)
```

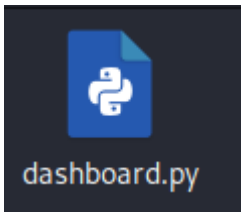
Actuador luz:



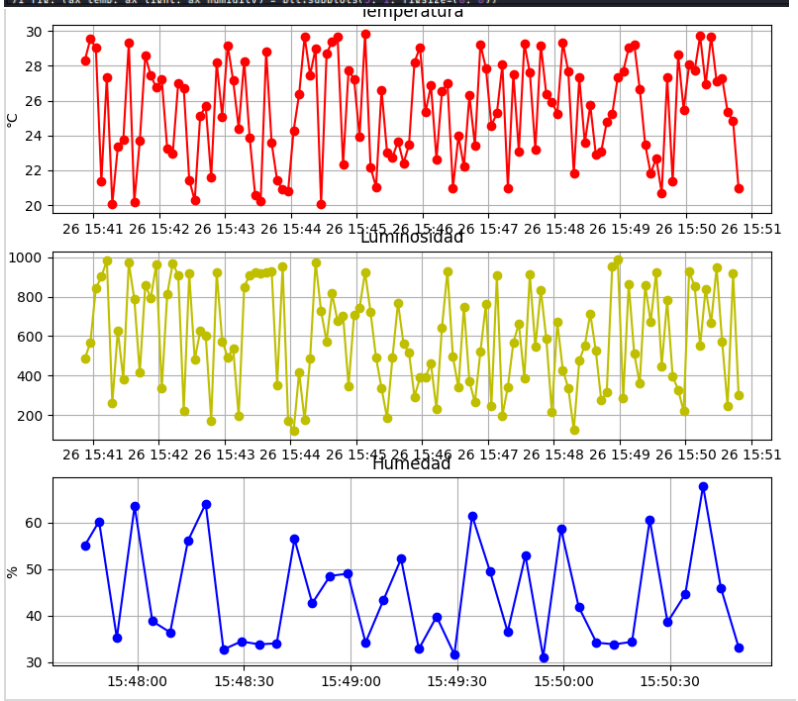

```python
import time
from influxdb_client import InfluxDBClient
from connection_component import InfluxDBConnection

def control_light():
    """Actúa como una luz que se enciende si la luminosidad es baja."""
    connection = InfluxDBConnection(
        url="http://localhost:8086",
        token="StpdsQrPrK_eRYDVcxNF3gFxSe0pPoKbtLe-qsxfZIl_fW-kxb2dvZT_gIpK-tS_U7hZipu4oNJaPMN7mMDgrw==",
        org="jhs",
        bucket="jhs"
    )

    client = connection.get_client()
    query_api = client.query_api()

    light_status = False  # Estado de la luz (False = apagada, True = encendida)

    try:
        while True:
            query = f'from(bucket: "{connection.bucket}") |> range(start: -1m) |> filter(fn: (r) => r._measurement == "light_sensor") |> filter(fn: (r) => r._field == "lux") |> last()'
            result = query_api.query(org=connection.org, query=query)

            for table in result:
                for record in table.records:
                    luminosity = record.get_value()

                    if luminosity < 300 and not light_status:
                        light_status = True
                        print("💡 Luz ENCENDIDA (Ambiente oscuro)")
                    elif luminosity > 600 and light_status:
                        light_status = False
                        print("🌙 Luz APAGADA (Ambiente iluminado)")

            time.sleep(5)
    except KeyboardInterrupt:
        print("Actuador de luz detenido.")

if __name__ == "__main__":
    control_light()
```

```
┌──(prueba)─(kali㉿joelhersie)-[~/Downloads/Influxdb]
└─$ python3 actuador_luz.py
💡 Luz ENCENDIDA (Ambiente oscuro)
🌙 Luz APAGADA (Ambiente iluminado)
```

- Generar algunos dashboards utilizando gráficos.

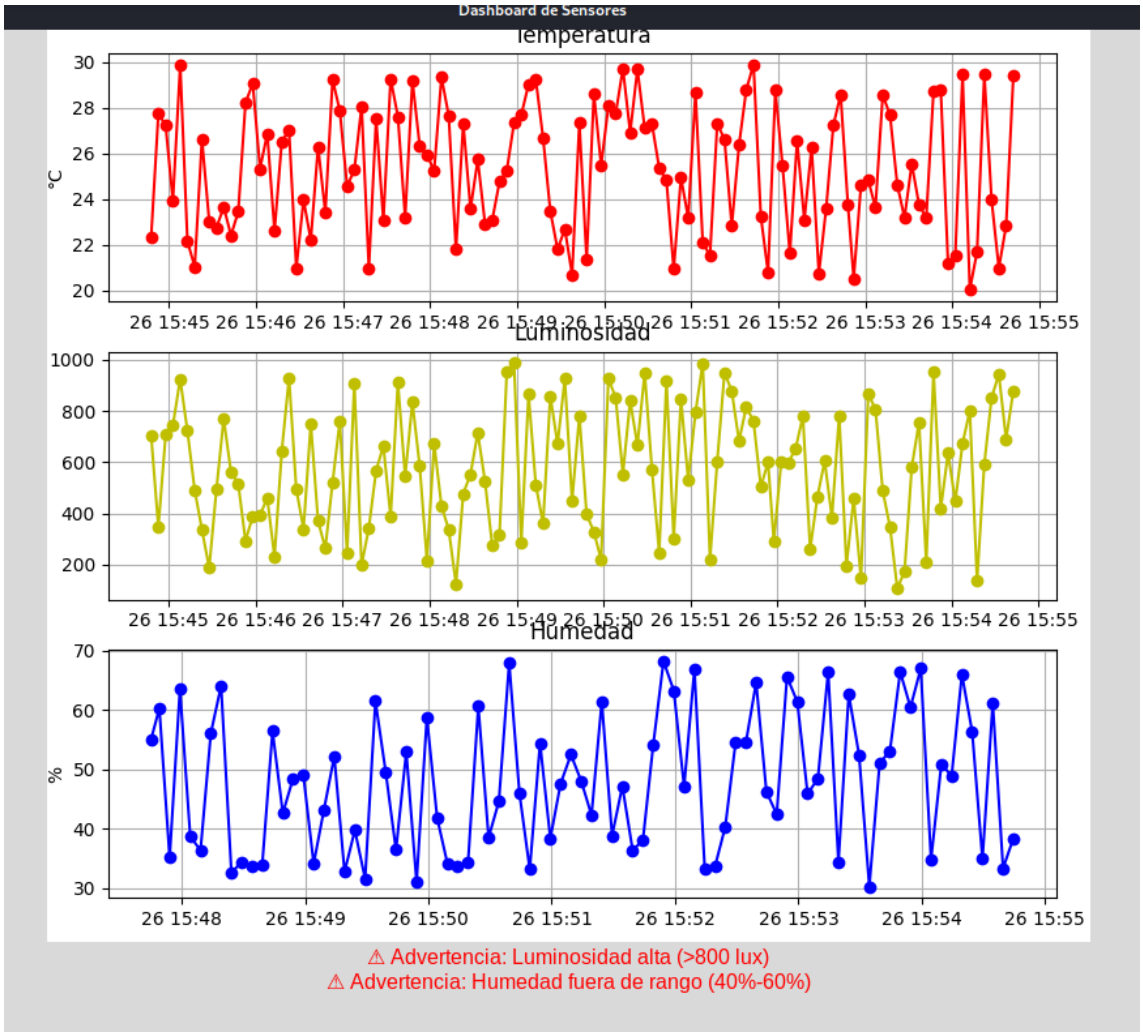



```python
import tkinter as tk
from tkinter import ttk
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from connection_component import InfluxDBConnection

def fetch_data(measurement, field):
    """Consulta los datos de un sensor en InfluxDB."""
    connection = InfluxDBConnection(
        url="http://localhost:8086",
        token="BEQp0_xkmLW_Bwya6595hQGjHXlqbobHfpBf7myPKugSkOnnMAiRE3W-t3SLMMdLaDYIwmHhuUQeHd1NspXmgQ==",
        org="jhs",
        bucket="jhs"
    )
    client = connection.get_client()
    query_api = connection.get_query_api(client)

    query = f'''
    from(bucket: "{connection.bucket}")
        |> range(start: -10m)
        |> filter(fn: (r) => r._measurement == "{measurement}" and r._field == "{field}")
        |> yield(name: "sensor_data")
    '''

    tables = query_api.query_data_frame(query)
    if tables.empty:
        return None

    df = tables[['_time', '_value']].rename(columns={"_time": "Time", "_value": field.capitalize()})
    df['Time'] = pd.to_datetime(df['Time'])
    df.set_index('Time', inplace=True)
    return df

def update_graphs():
    """Actualiza los gráficos con nuevos datos."""
    ax_temp.clear()
    ax_light.clear()
    ax_humidity.clear()

    temp_data = fetch_data("thermometer", "temperature")
    light_data = fetch_data("light_sensor", "lux")
    humidity_data = fetch_data("hygrometer", "humidity")

    if temp_data is not None:
        ax_temp.plot(temp_data.index, temp_data['Temperature'], marker='o', linestyle='-', color='r')
        ax_temp.set_title("Temperatura")
        ax_temp.set_ylabel("°C")
        ax_temp.grid(True)

    if light_data is not None:
        ax_light.plot(light_data.index, light_data['Lux'], marker='o', linestyle='-', color='y')
        ax_light.set_title("Luminosidad")
        ax_light.set_ylabel("Lux")
        ax_light.grid(True)

    if humidity_data is not None:
        ax_humidity.plot(humidity_data.index, humidity_data['Humidity'], marker='o', linestyle='-', color='b')
        ax_humidity.set_title("Humedad")
        ax_humidity.set_ylabel("%")
        ax_humidity.grid(True)

    canvas.draw()
    root.after(5000, update_graphs)  # Actualiza cada 5 segundos

# Configurar la interfaz gráfica
root = tk.Tk()
root.title("Dashboard de Sensores")
root.geometry("900x700")

fig, (ax_temp, ax_light, ax_humidity) = plt.subplots(3, 1, figsize=(8, 8))
```



- Generar algunas alertas.

He editado el dashboard, y he creado alertas ahí, se verán cuando sobre pase la temperatura, luz o humedad que hemos indicado.




```python
def check_alerts(temp, light, humidity):
    """Muestra alertas si los valores están fuera de los límites normales."""
    alerts = []
    if temp is not None:
        if temp > 35:
            alerts.append("⚠ ALARMA: Temperatura crítica (>35°C)")
        elif temp > 30:
            alerts.append("⚠ Advertencia: Temperatura alta (>30°C)")

    if light is not None:
        if light > 900:
            alerts.append("⚠ ALARMA: Luminosidad crítica (>900 lux)")
        elif light > 800:
            alerts.append("⚠ Advertencia: Luminosidad alta (>800 lux)")

    if humidity is not None:
        if humidity < 30 or humidity > 70:
            alerts.append("⚠ ALARMA: Humedad crítica (<30% o >70%)")
        elif humidity < 40 or humidity > 60:
            alerts.append("⚠ Advertencia: Humedad fuera de rango (40%-60%)")
```

VIDEO SEGUNDA SEMANA:

https://youtu.be/P50N0AF8xAs

What is to be delivered

- Primera semana: Solo el video
- Segunda semana: El código (solo el código, no el proyecto) y un video mostrando cómo funciona todo.