Bolocan Crina-Maria

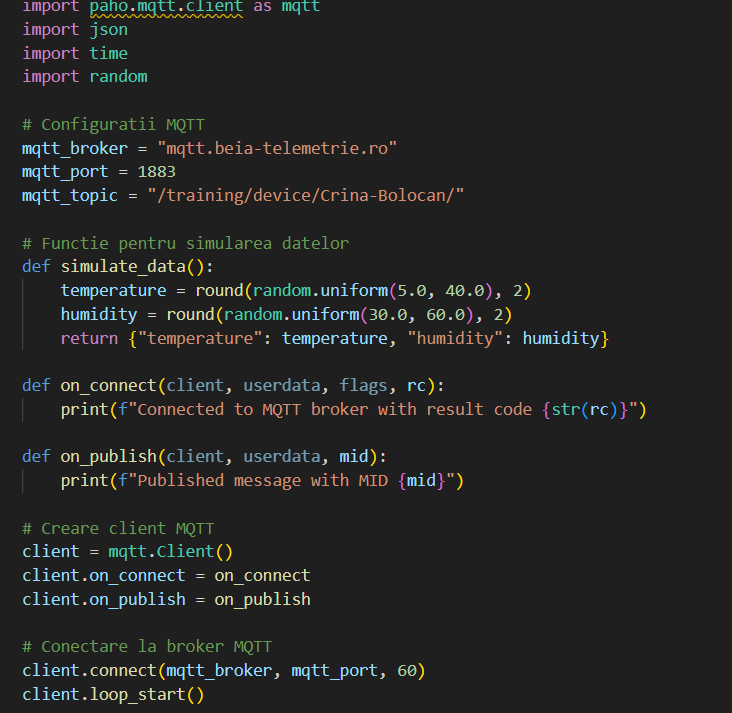
Pre-practica Task

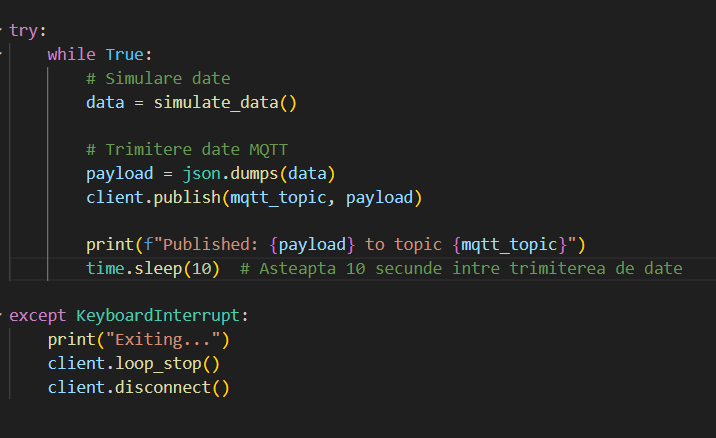
1. Description

Document in a GDoc, make a presentation in English and post a code in Github for connecting IoT devices (raspberry pi, esp, pycom, arduino, libelium, galileo, android, etc.) via MQTT to the database, storage on the blockchain (Ethereum, IOTA, Hyperledger, etc.), orchestrating services with Arrowhead using docker and viewing in a mobile application, using Node-Red for data processing, viewing data in Grafana, setting notifications in Grafana, configuring local data storage in a folder with candidate name and auto-start resynchronization of local data with the cloud in case of power or data network failure, data query via chatbot or voicebot.

1. Generating random

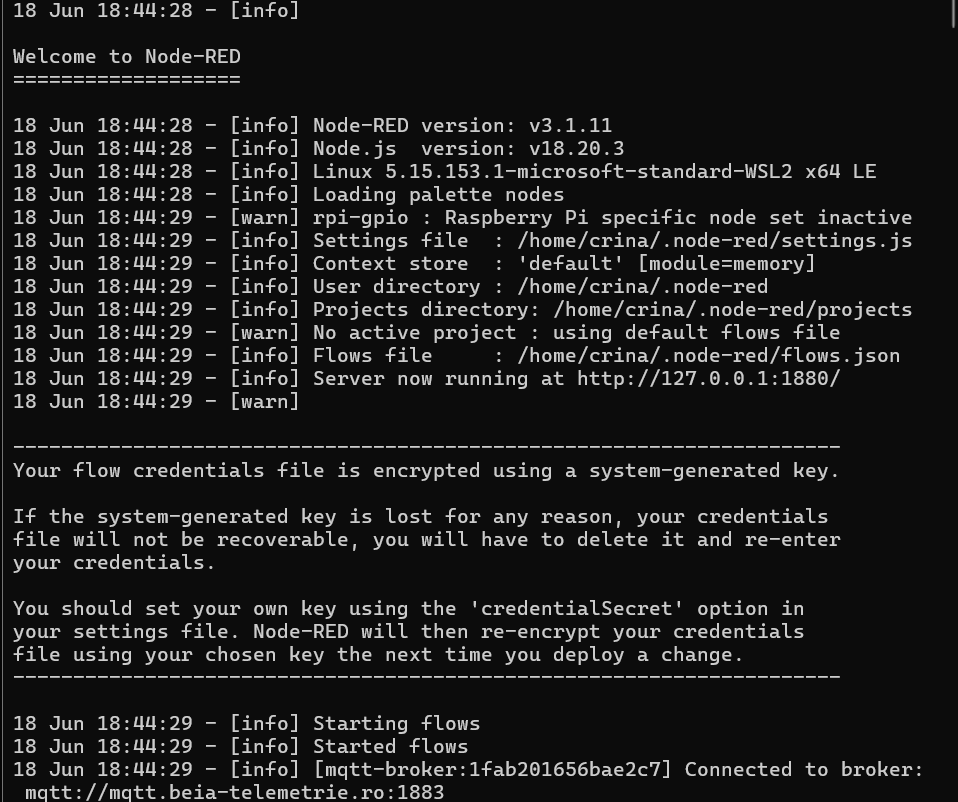
Due to the lack of a special device, I generated data for both temperature and humidity randomly. The code in python generates simulated data for temperature and humidity using random values within specified ranges (temperature: 5.0 to 40.0, humidity: 30.0 to 60.0) and return a dictionary containing temperature and humidity. After that, I callback function called when the MQTT client connects to the broker. Setup: creates an instance of the MQTT client (mqtt.Client()), connects to the MQTT broker (mqtt\_broker, mqtt\_port) with a keep-alive interval of 60 seconds, converts the data dictionary to JSON format (payload = json.dumps(data)), publishes the JSON payload to the MQTT topic (mqtt\_topic) using client.publish().





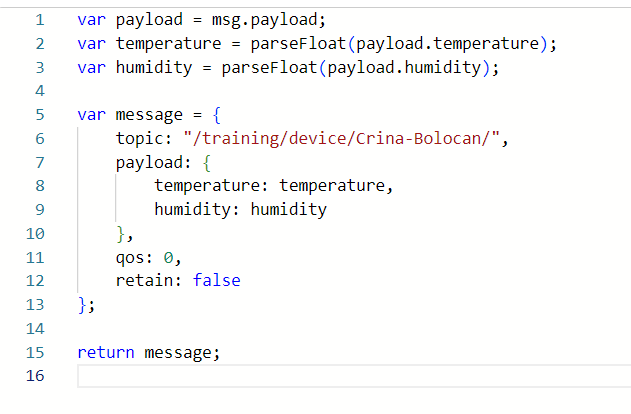
1. Node-red:

I successfully installed it from https://nodejs.org/en and I generated the IP address so I can access it:

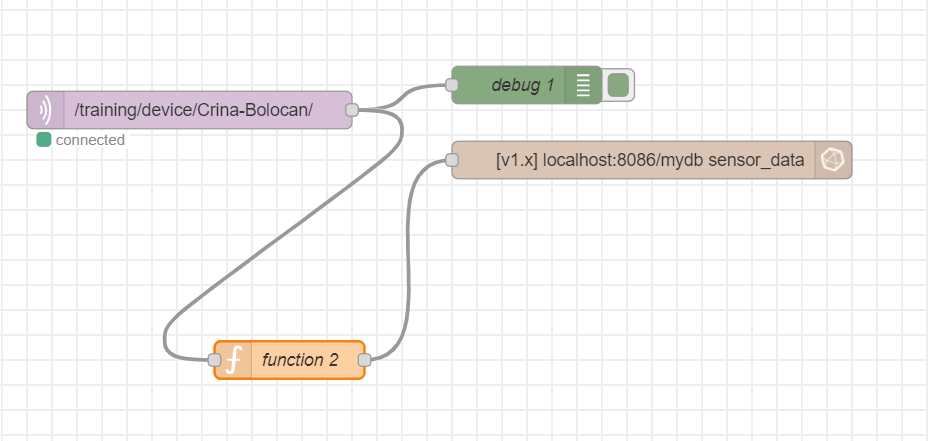


Steps:

* Configured a „mqtt in” node to subscribe to the MQTT topic /training/device/Crina-Bolocan/.
* Set the server to mqtt.beia-telemetrie.ro.
* Created a function node to process the parsed JSON data.
* The function extracts the temperature and humidity values and formats them into a new message suitable for InfluxDB.



* Add a node „influxdb out” o write the temperature and humidity data to InfluxDB.
* Connecting nodes.



 This is the output from debug node, to check the data.

1. Grafana

Steps:  
  
1. Data Source Configutraion:

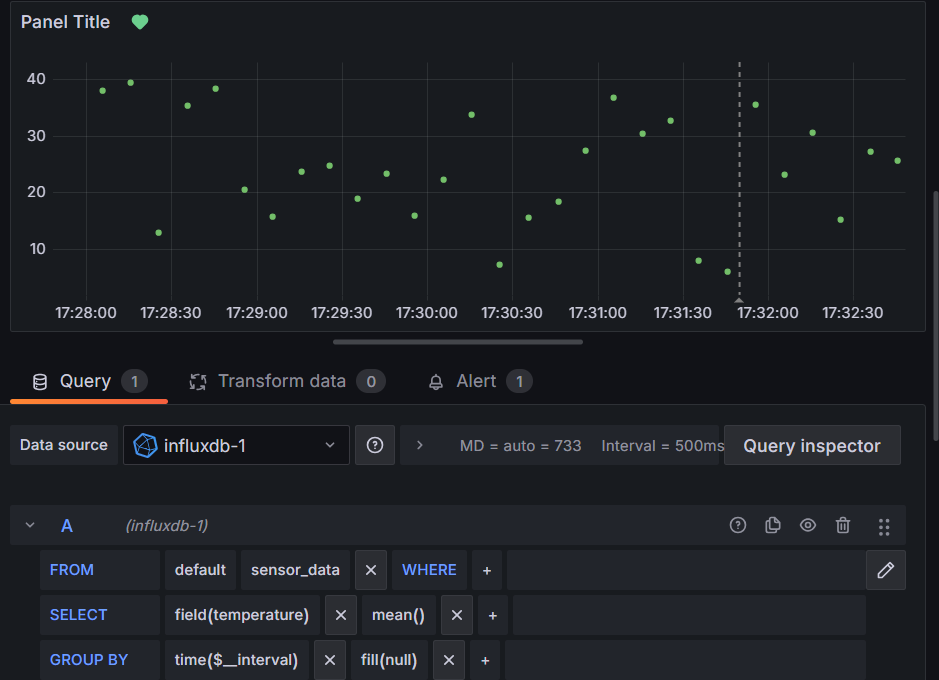
* Added InfluxDB as a data source in Grafana.
* Configured the InfluxDB URL (e.g., http://localhost:8086) and database name (e.g., mydb).
* Specified authentication details if required.

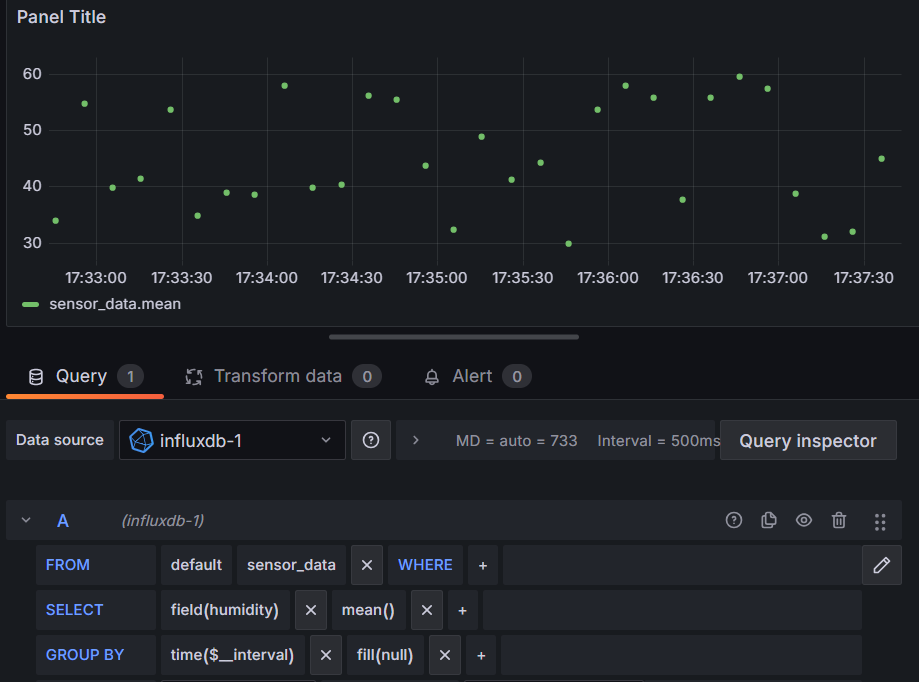
2. Creating a Dashboard:

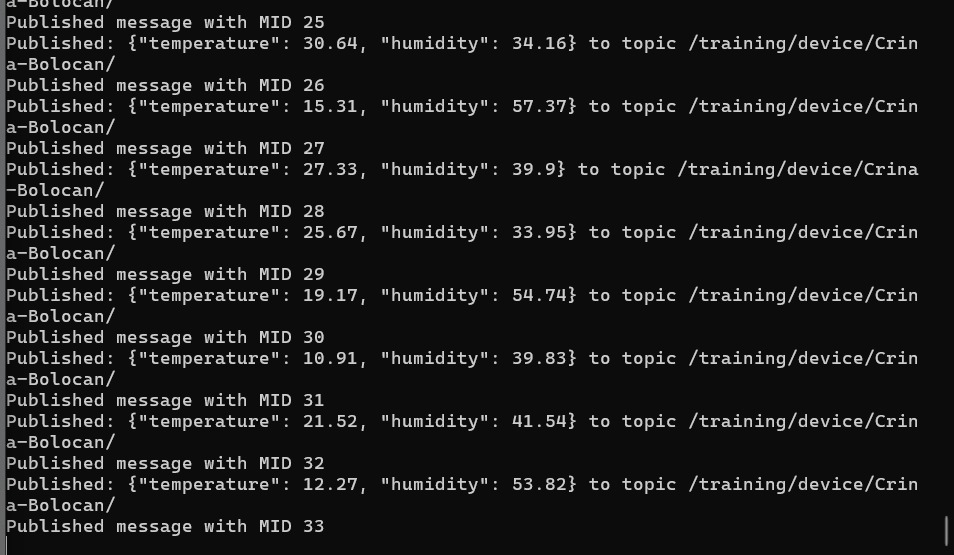
* Created a new dashboard in Grafana.

3. Adding Panels:

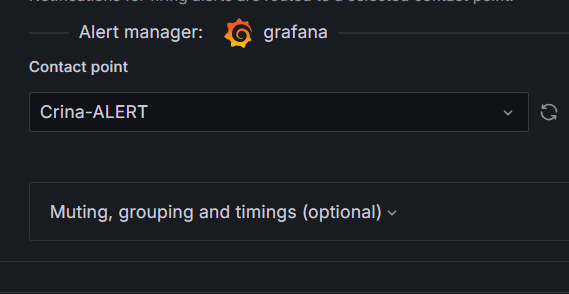
* Added two panels to the dashboard: one for temperature and one for humidity.



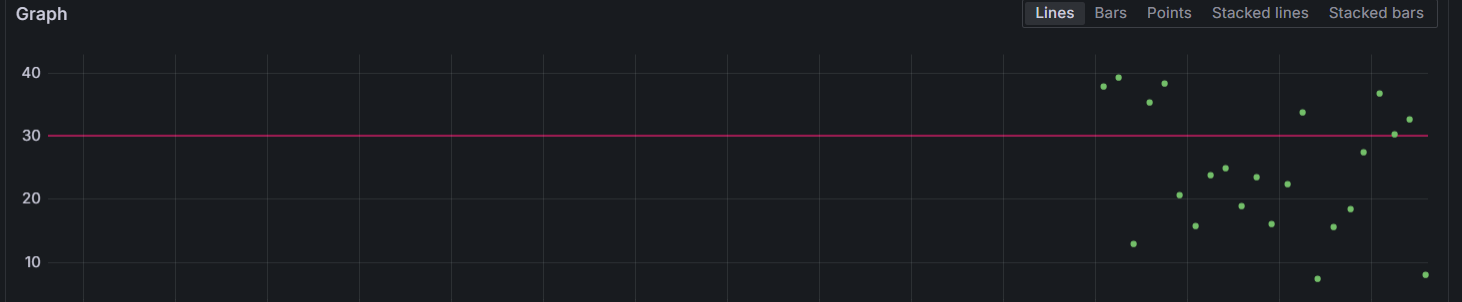


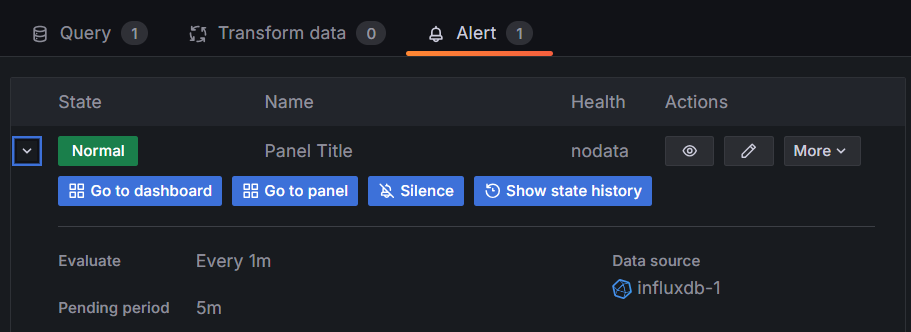


Setting up the alert and notification in Grafana:

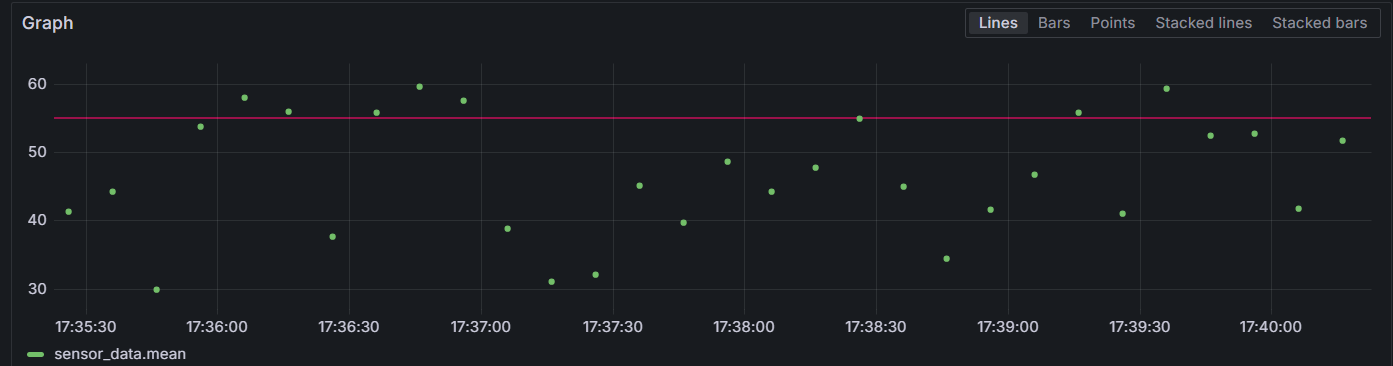


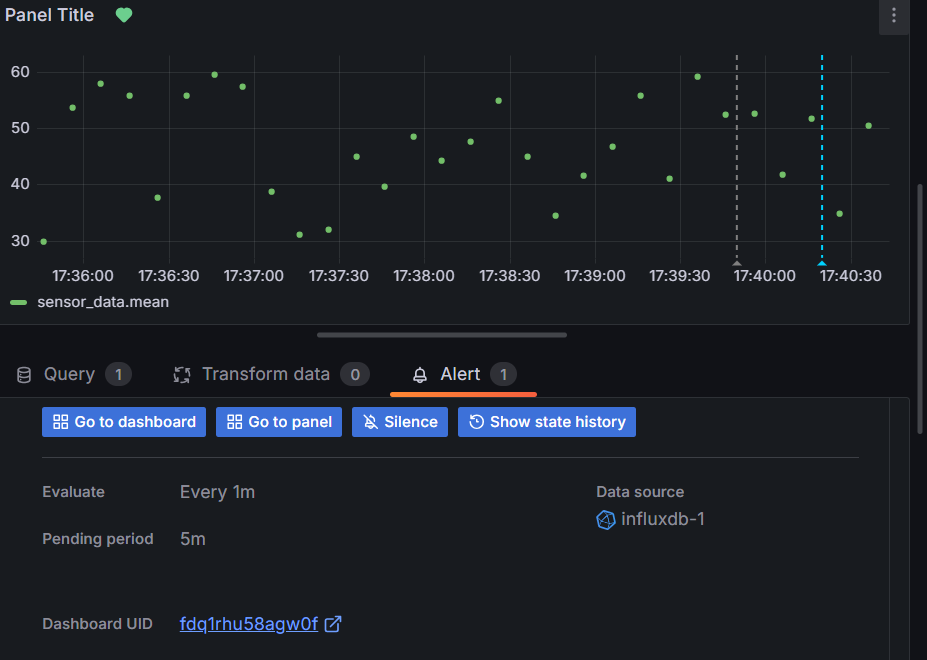
Alert value is for temperature: if is > 30.





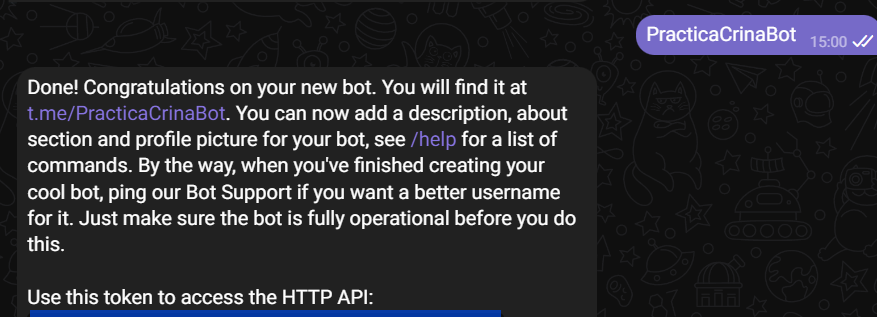
Alert value for humidity is 55.



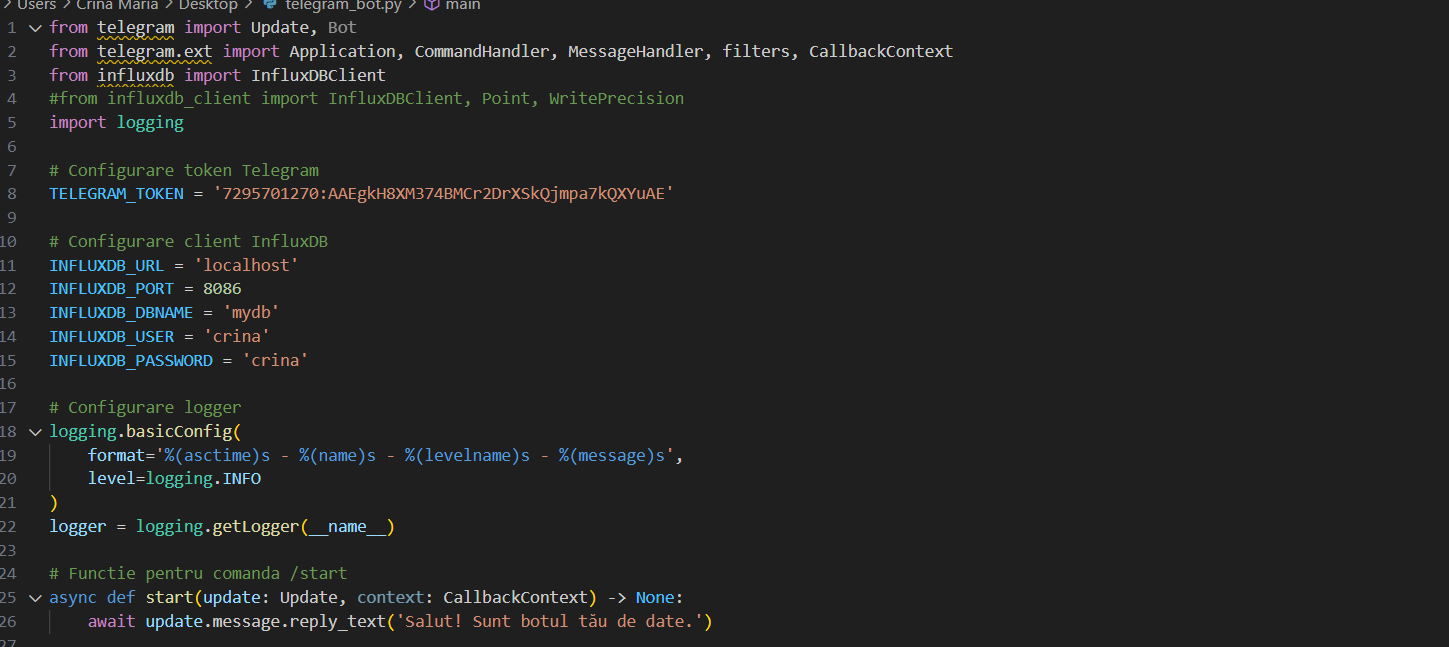


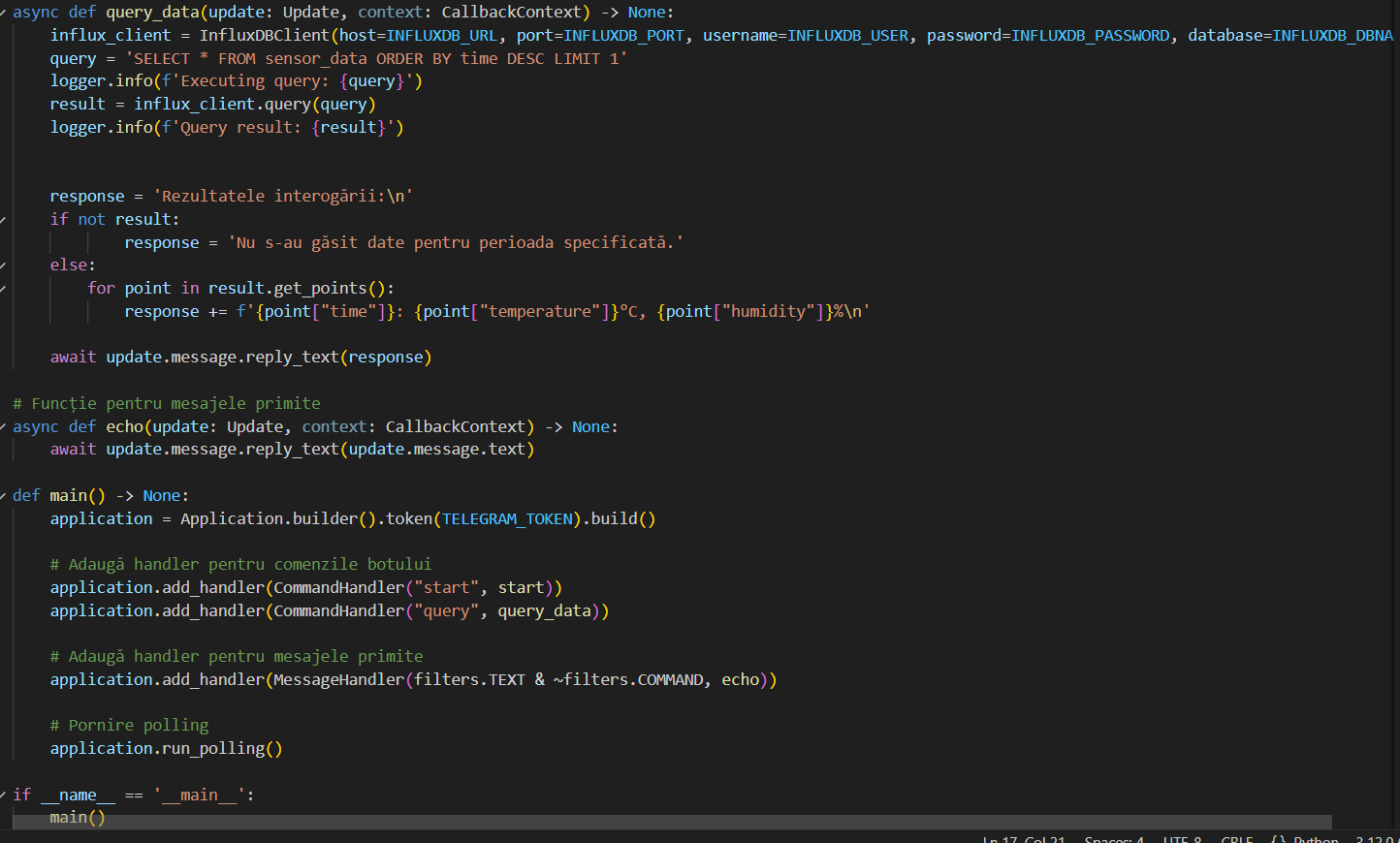
1. Creating a telegram ChatBot:

Firstly, I need to download Telegram and create an account on my phone and connect on telegram web. I searched for „BotFather”. I created a newbot with command /newbot.



After that, I made a python code so I can use my bot to tell me the info about the data:





The code help me to connect with the newbot and give me info about currrent data.



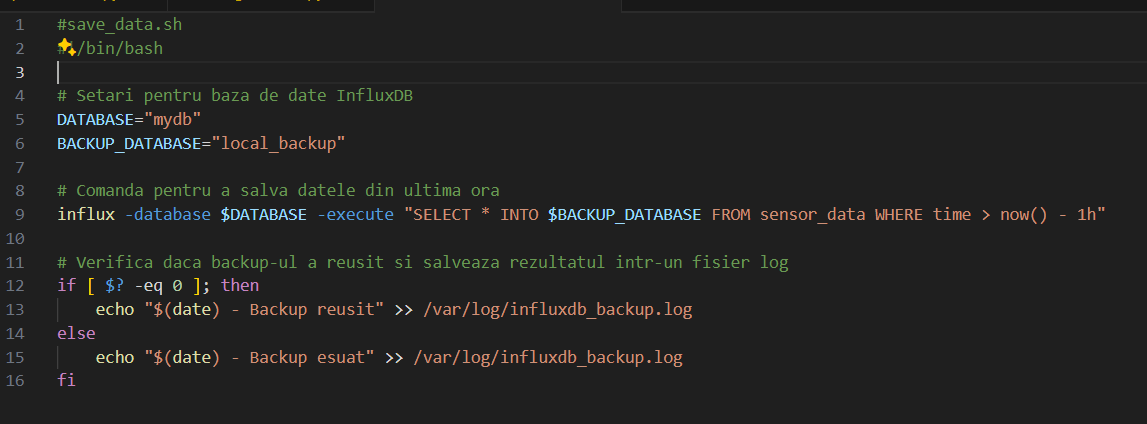
1. Resynchronizing Data in Case of Outage

Local Backup Configuration

Create a script to save data locally in case of an outage and resynchronize it with the InfluxDB database when the connection is restored.

*Backup Script:*

A shell script (save\_data.sh) is created to perform a local backup of data. This script uses the influx command-line tool to execute a query that copies data from the sensor\_data measurement to a local\_backup measurement for the last hour.





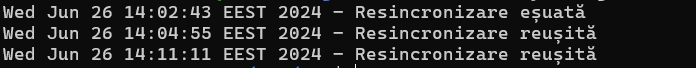
*Periodic Execution:*

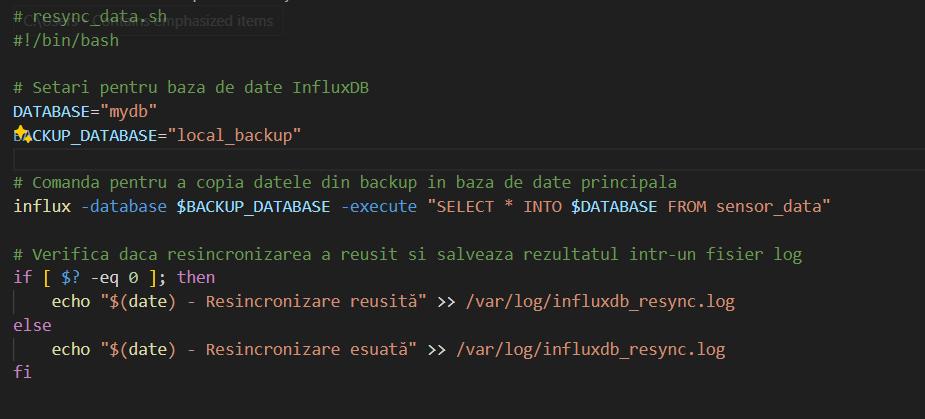
The script is scheduled to run periodically using crontab, which is a job scheduling system available on Unix-like operating systems.

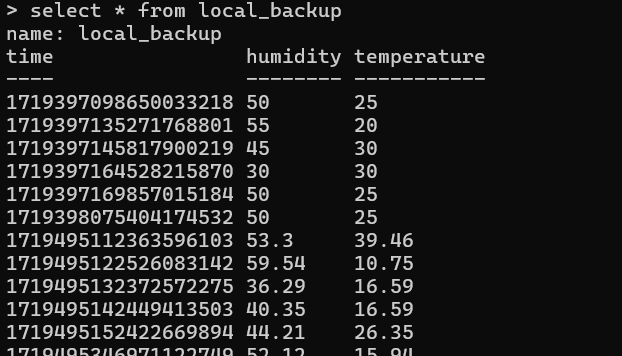
The crontab entry 0 \* \* \* \* /path/to/save\_data.sh ensures that the backup script runs at the start of every hour, thereby regularly saving the latest data locally.

*Resynchronization:*

When the connection to InfluxDB is restored, the locally backed-up data can be resynchronized with the main database. This ensures that no data is lost during an outage, and all data is eventually stored in the InfluxDB database as intended.





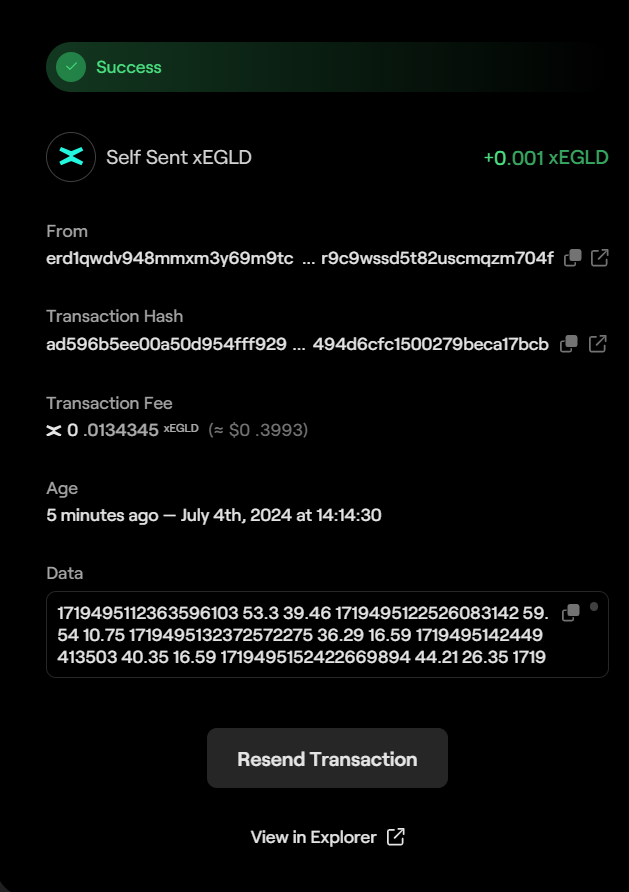


1. Blockchain storage part:

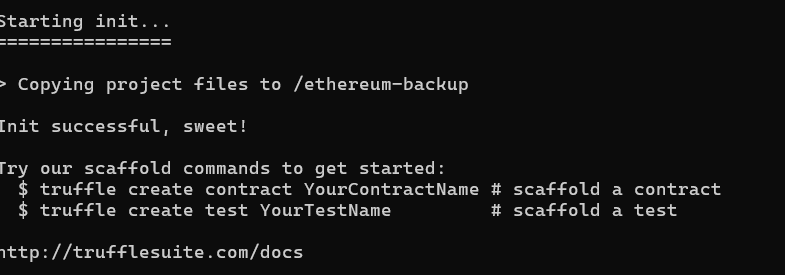
Storing data on the blockchain has become essential due to the limitations of traditional cloud-based storage methods. The conventional approach involves centralizing data on a single server, making it vulnerable to attacks and often lacking encryption.

Initialized the truffle = popular development framework for Ethereum and other blockchain platforms. It simplifies and streamlines the process of building decentralized applications (DApps) and smart contracts. Truffle provides a suite of tools that make it easier for developers to write, deploy, and manage smart contracts on the Ethereum blockchain.

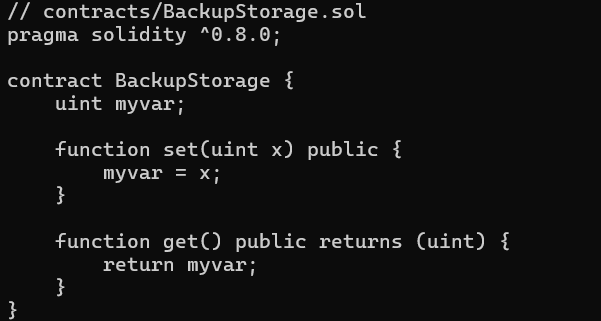
I used XPortal and MultiversX mobile app to get a data transaction.



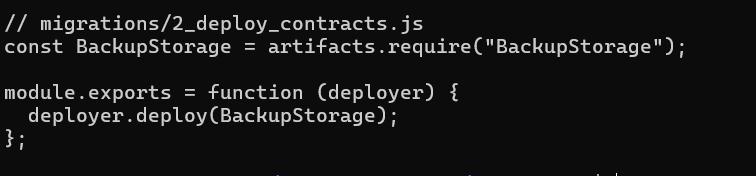
Another try with someting else:



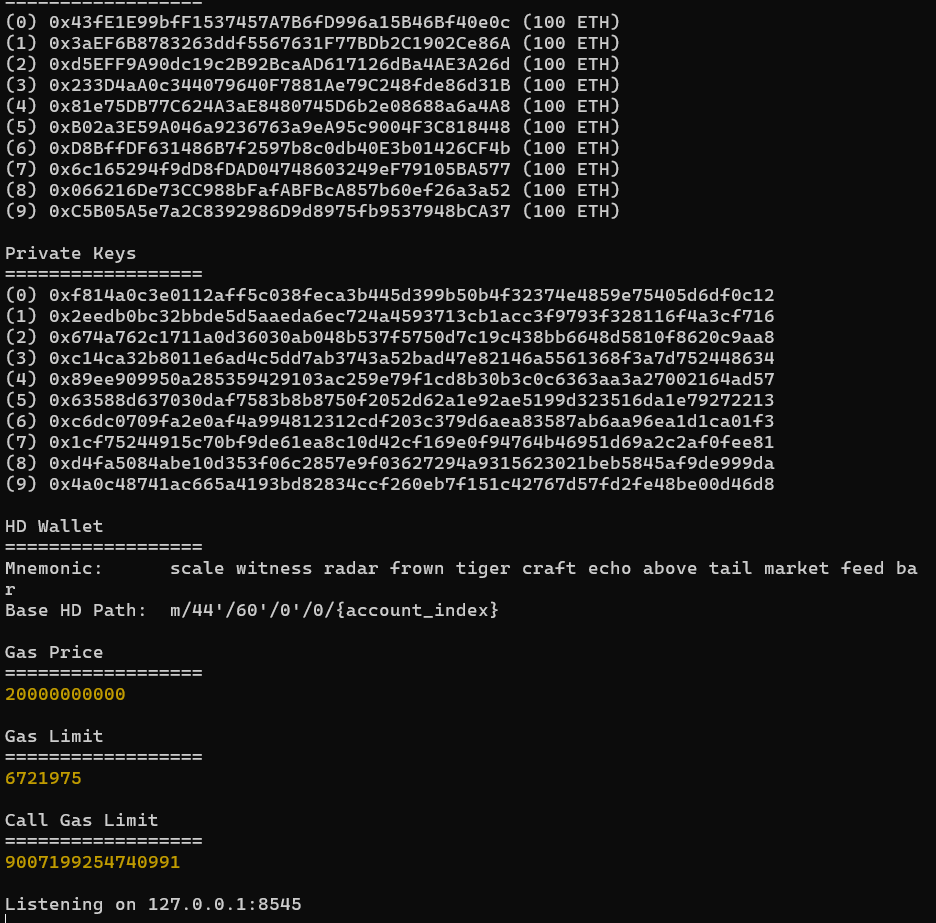
In directory contracts/ I created the BackupStorage for a simple example:

****

I also created a file “2\_deploy\_contracts.js” in the migrations/ directory which is used as a directive that allows us to deploy the SimpleStorage contract to the blockchain:

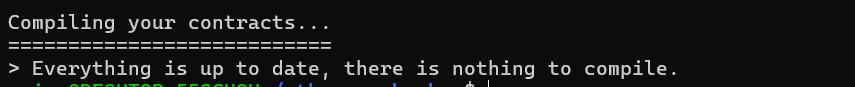
****

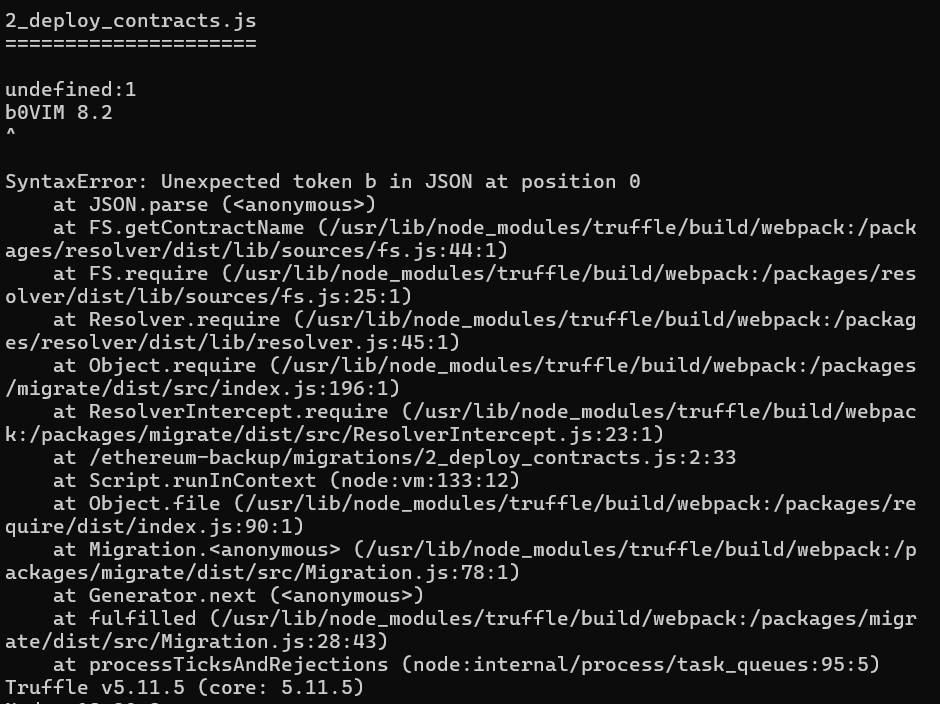
ganache-cli:

****

truffle compile

truffle migrate

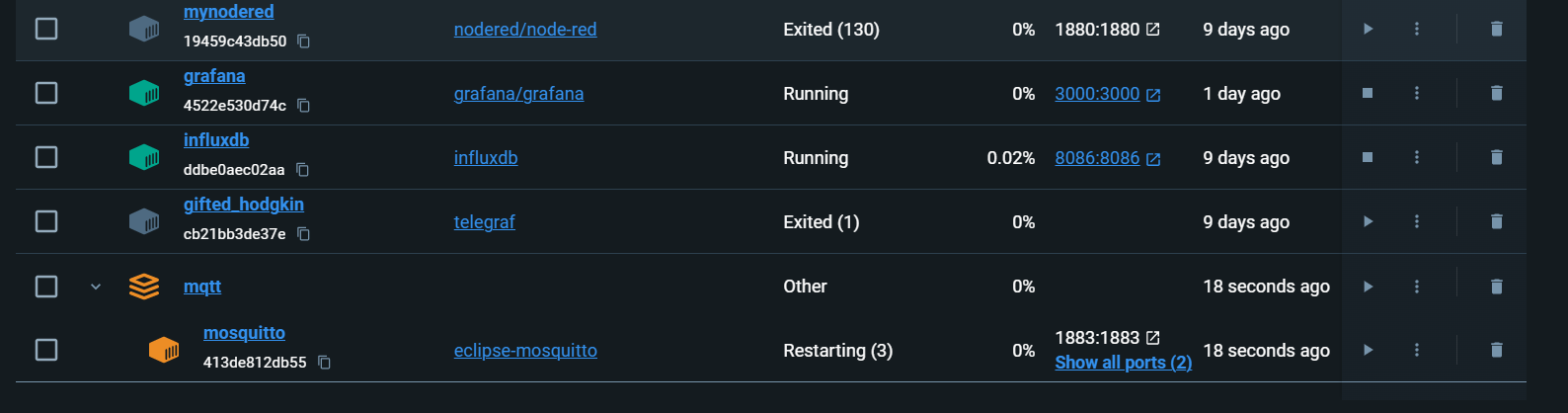
****

**Issue:   
**

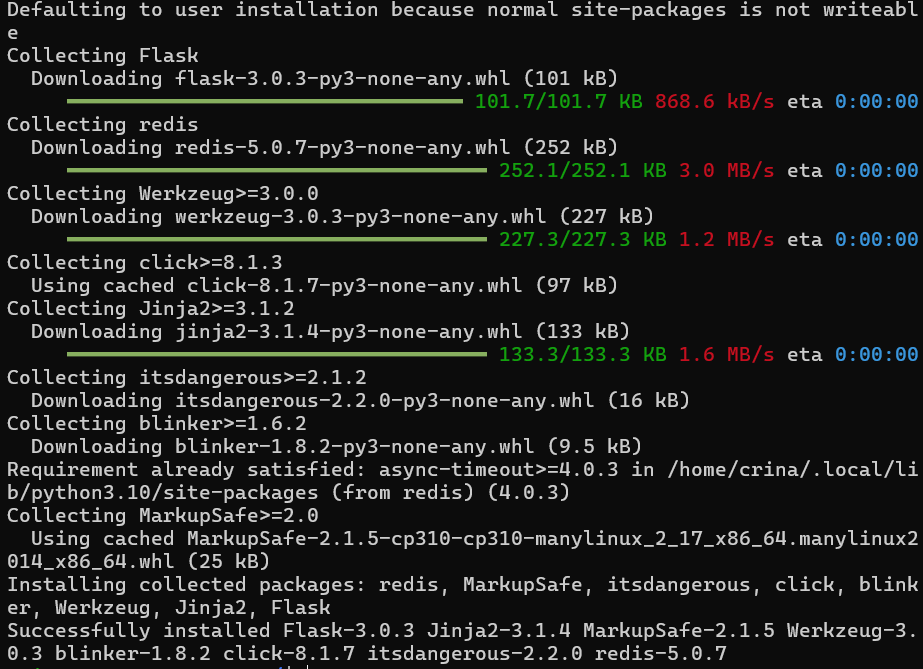
1. **Docker**

**I already had Docker installed, so I went straight to creating the containers. „Docker is a set of platform as a service (PaaS) products that uses OS-level virtualization to deliver software in packages called containers. The software that hosts the containers is called Docker engine.**

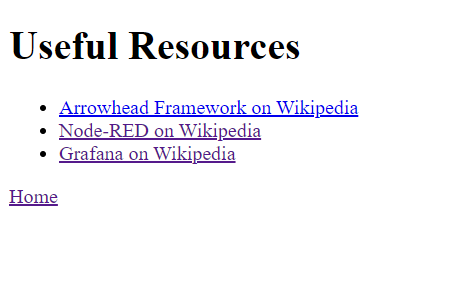
**A container is a lightweight, standalone, and executable software package that includes everything needed to run a piece of software, including the code, runtime, system tools, system libraries, and settings. Containers provide a consistent and isolated environment for applications to run across different computing environments, such as development, testing, and production.”**

****

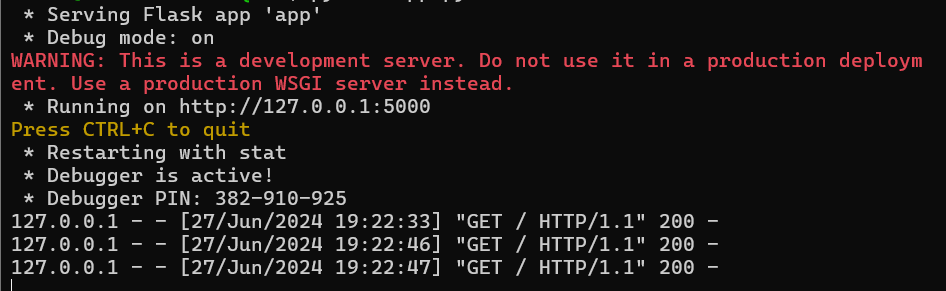
**Flask installed:**

****

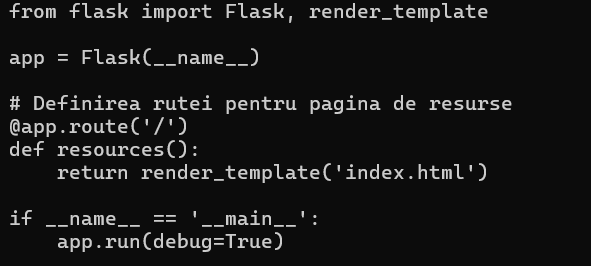
**This is the web page I created, so we can see the appilication running, but it’s simple without part of Arrowhead.**

****

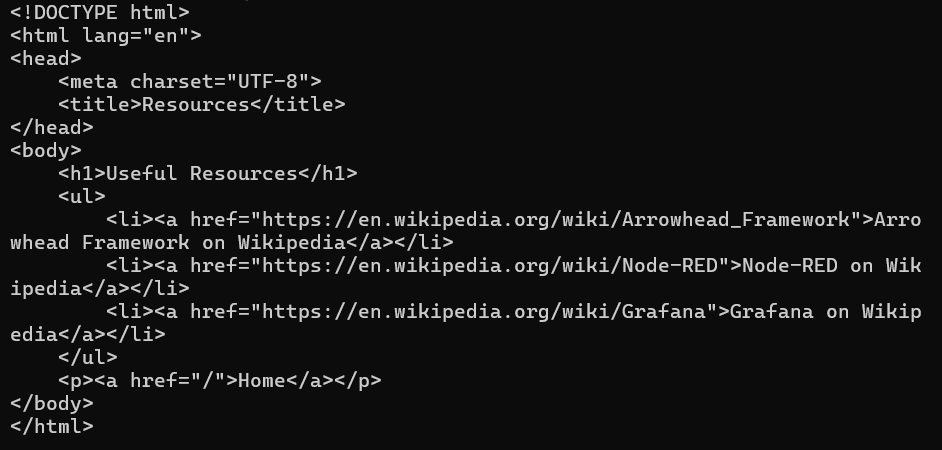
**This is the output of the script for the app, so we can see that it’s working. „python app.py”**

****

**The code:**

****

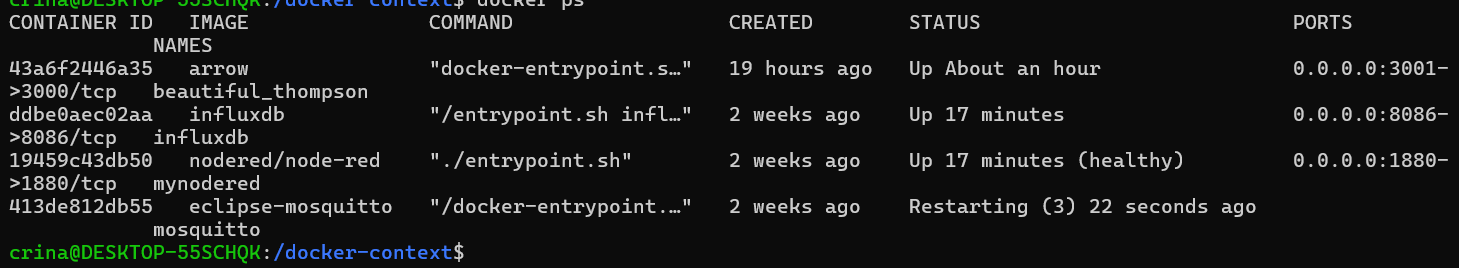
**The HTML code for the web page:**

****

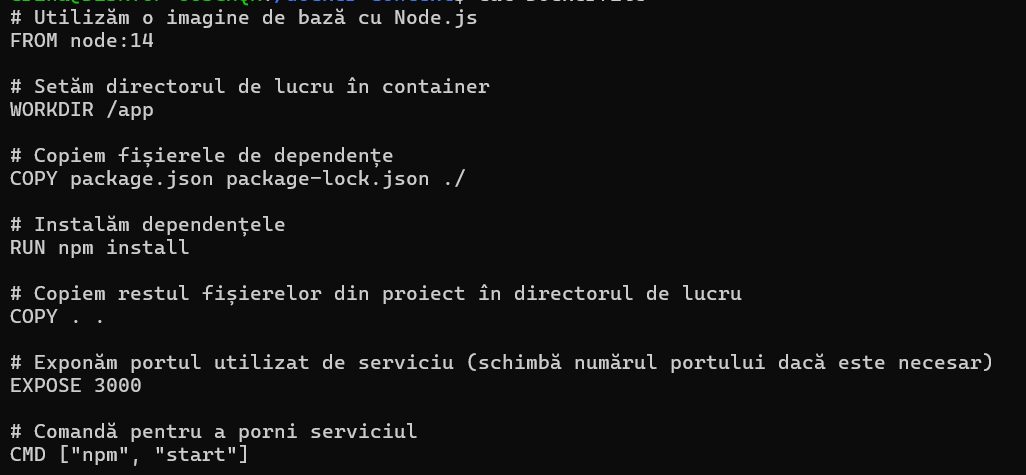
1. Arowhead:

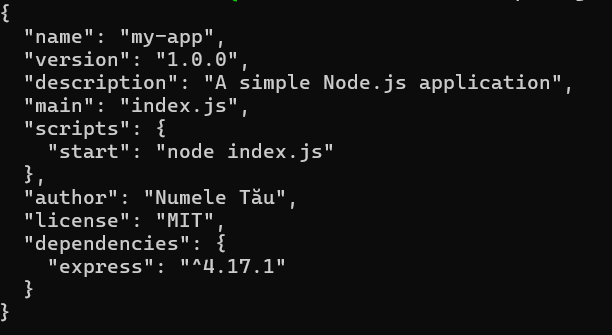
The Arrowhead Framework enables the design and implementation of automation systems in application domains such as production, smart cities, e-mobility, energy, and buildings.

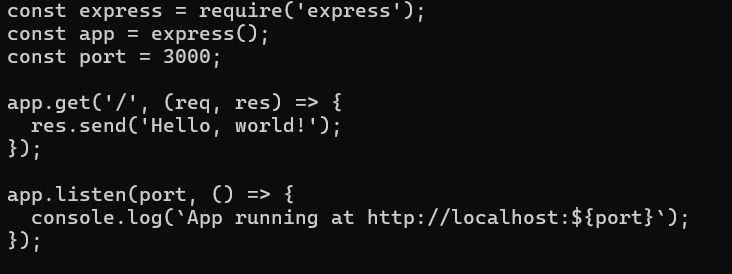
It was created to efficiently address Industry 4.0 requirements, primarily through scalable, secure, and flexible information sharing that enables system interoperability and integration.



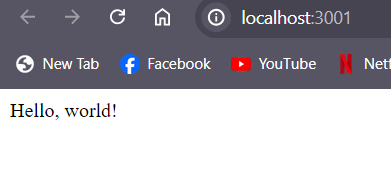
I created the arrowhead part only for Grafana to see if it’s working.

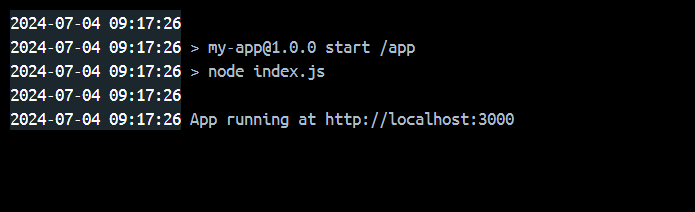






This is the first web page that I created using the arrowhead part, after that I want to connect with the Flask part that I already created (you can see the photo in the Docker part in the file).





Github link: [git@github.com:crinabolocan/Practica-BEIA.git](mailto:git@github.com:crinabolocan/Practica-BEIA.git)

<https://github.com/crinabolocan/Practica-BEIA.git>