

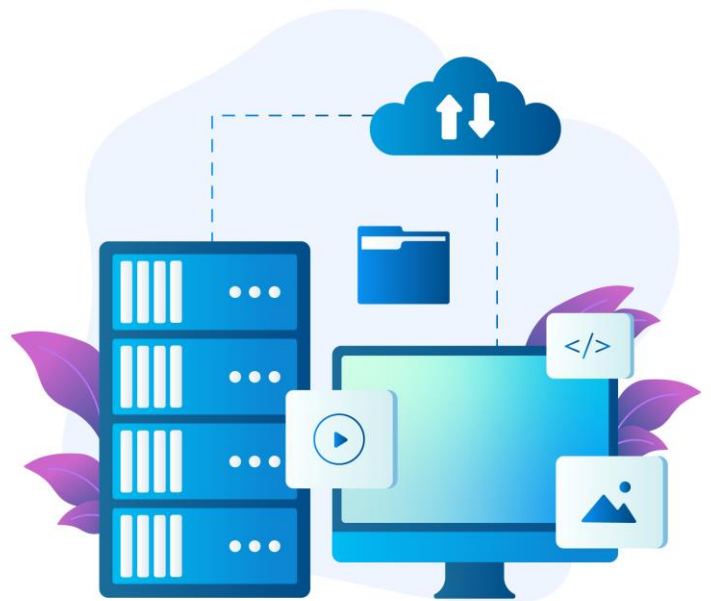
Project report

« Distributed Systems »

Presented by :



Selim Ben Jeddí



2022-2023



Report outline

- 1- Presentation of the project*
- 2- Project Architecture*
- 3- Project planning with scrum*
- 4- Realization of the project*

1- Presentation of the project

The goal of the project is to create a weather web application using Docker and modern technologies such as: Python for the management of the back end side and with its Flask library for the Front end, Kafka to control the flow of data between the database data and the application and cassandra for data storage.

The application contains advanced features like email notifications, history, recommendation system and feed management using Kafka.

All these technologies are included in a development environment based on Docker containers in order to facilitate the implementation.

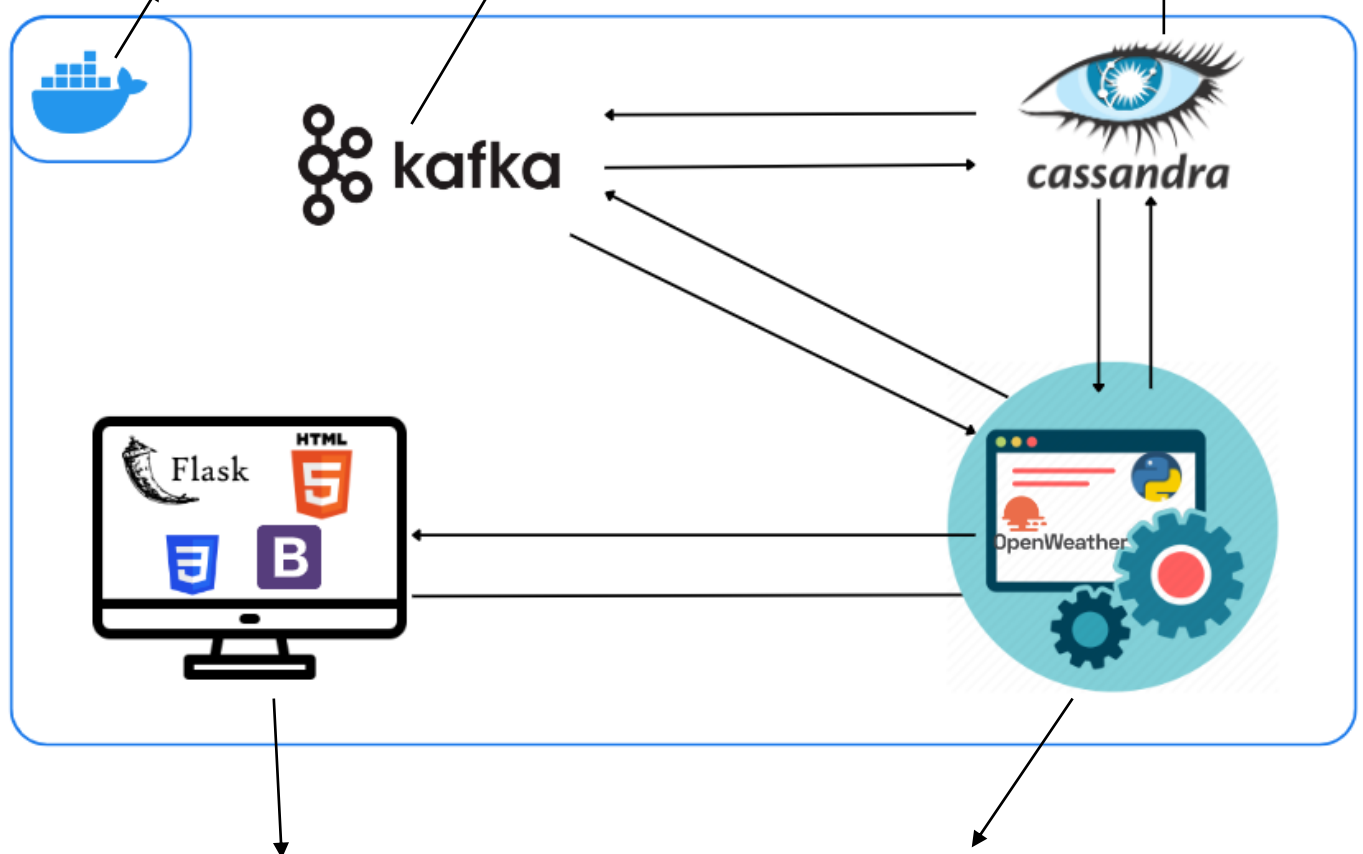
2- Project Architecture

Docker : built by images installed from Docker-compose and Docker File. In our project we distinguish 4 images and thus 4 containers: Zookeeper (on which kafka is based), Kafka, Python and cassandra

Kafka : acts as an intermediary between the back-end and Cassandra. Composed of producer who receives the request from the back-end it produces the message and places it in topic so that the consumer consumes the topic and manages the database

Cassandra : NoSQL database that contains 2 tables: users and meteorological data.

These tables are manipulated by the CQLEngine library in python to avoid writing queries in the code and presented the code in the form of classes



Front-end: The work in the Front-end is based on the Flask library of python mainly which allows to manage the Html pages containing the graphic interfaces, with the use of bootstrap for the CSS.

Back-end Python : The work in the back-end is distributed in the form of 2 python files: the first containing 21 processing functions with 2 classes with a main to run the python and kafka servers, and the second for the management of notifications by email.

3- Project planning with Scrum

Product Backlog:

In order to determine it, we use the MoSCoW method which is frequently used in projects with an iterative life cycle.

Priority: A characteristic that indicates how important user stories are to each other.

This method is based on:

- **M** for " **Must have**" (**vital**) : these tasks are essential to the success of the project. They must therefore be carried out as a priority and this is non-negotiable.

- **S** for " **Should have**" (**essentiel**) : this category groups the important tasks to be carried out as soon as the "Must have" tasks are completed. They bring real added value to the project and contribute to the achievement of objectives, but unlike the "Must haves", they can be deferred over time.

ID	User Story	Priorité
1	As a user I can consult the weather data in real time for a specific city, including temperature, humidity, pressure atmosphere and precipitation.	Must Have
2	As a user I can search weather data by city by using a map.	Should Have
3	As a user I can display the search results and benefit from a suggestion system following the weather.	Must Have
4	As a user, I can register on the application in order to benefit from advanced features.	Must Have
5	As a subscriber I can authenticate myself in order to access my account	Must Have
6	As a subscriber I can have a list of favorite cities	Should Have
7	As a subscriber I can receive daily email notifications on my favorite cities with alerts in case of extreme weather conditions.	Should Have
8	As a subscriber I can	Must Have

	consult the history of my searches by city and days with a maximum of 16 days.	
--	--	--

In order to set up this work, we divided this backlog into 3 Sprints:

Sprint 1 (17 Days): From January 30, 2023 to February 15, 2023

Sprint 2 (25 Days): From February 16, 2023 to March 12, 2023

Sprint 3 (21 Days): From March 13, 2023 to April 2, 2023

Sprint 1 :

In this sprint, we focused on the documentation of new technologies, the preparation of the working environment on Docker and we started the front and back end of the recovery of weather data by city.

Sprint Backlog :

ID	User Story	Tasks	Upcoming User Stories (Sprint2)
1	Documentation	<ul style="list-style-type: none"> - Familiarity with Docker, Kafka and Cassandra concepts 	<ul style="list-style-type: none"> - As a user I can search weather data by city by using a map.
2	Installation of the working environment	<ul style="list-style-type: none"> - Installation of the docker (with technical difficulties related to the installation of the docker on Windows 11) - Creation of the project with the different images: Zookeeper, Kafka, Cassandra and python from the docker-compose - Installation of python libraries using the docker File - Test containers 	<ul style="list-style-type: none"> - As a user I can display the search results and benefit from a suggestion system following the weather. - As a user, I can register on the application in order to benefit from advanced features. - As a subscriber I can authenticate myself in order to access my account
3	As a user I can consult the weather data in real time for a specific city, including	<ul style="list-style-type: none"> - Recovery of the Open Weather API key - Development of the python back 	

	temperature, humidity, pressure atmosphere and precipitation.	end for the recovery of weather data in real time by city - Development of the graphical interface for the search of cities.	
--	---	---	--

Realisation :

Docker-Compose

```

docker-compose.yml
1  version: "3"
2
3  services:
4    zookeeper:
5      image: 'bitnami/zookeeper:latest'
6      container_name: zookeeper
7      ports:
8        - '2181:2181'
9      environment:
10       - ALLOW_ANONYMOUS_LOGIN=yes
11    kafka:
12      image: 'bitnami/kafka:latest'
13      container_name: kafka
14      ports:
15        - '9092:9092'
16      environment:
17        - KAFKA_BROKER_ID=1
18        - KAFKA_LISTENERS=PLAINTEXT://:9092
19        - KAFKA_ADVERTISED_LISTENERS=PLAINTEXT://kafka:9092
20        - KAFKA_ZOOKEEPER_CONNECT=zookeeper:2181
21        - ALLOW_PLAINTEXT_LISTENER=yes
22      depends_on:
23        - zookeeper

```

```

24    cassandra:
25      image: 'cassandra:latest'
26      container_name: cassandra
27      ports:
28        - '9042:9042'
29      environment:
30        - CASSANDRA_CLUSTER_NAME=WeatherDataCluster
31      volumes:
32        - cassandra_data:/var/lib/cassandra
33    python:
34      build: ./myapp
35      container_name: python
36      ports:
37        - '5000:5000'
38      environment:
39        - KAFKA_SERVER=kafka:9092
40        - CASSANDRA_SERVER=cassandra:9042
41      depends_on:
42        - kafka
43        - cassandra
44      volumes:
45        - ./myapp:/app
46    volumes:
47      cassandra_data:

```

Docker-File

```

myapp > Dockerfile > ...
1  FROM python:3.8
2
3  WORKDIR /app
4
5  COPY requirements.txt requirements.txt
6  RUN pip install -r requirements.txt
7
8  COPY . .
9
10 CMD ["python", "app.py"]
11
12

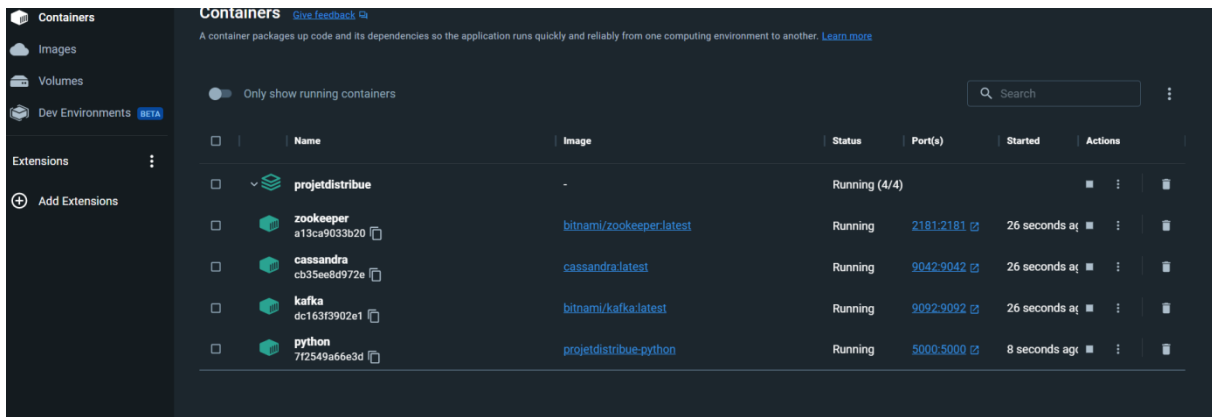
```

```

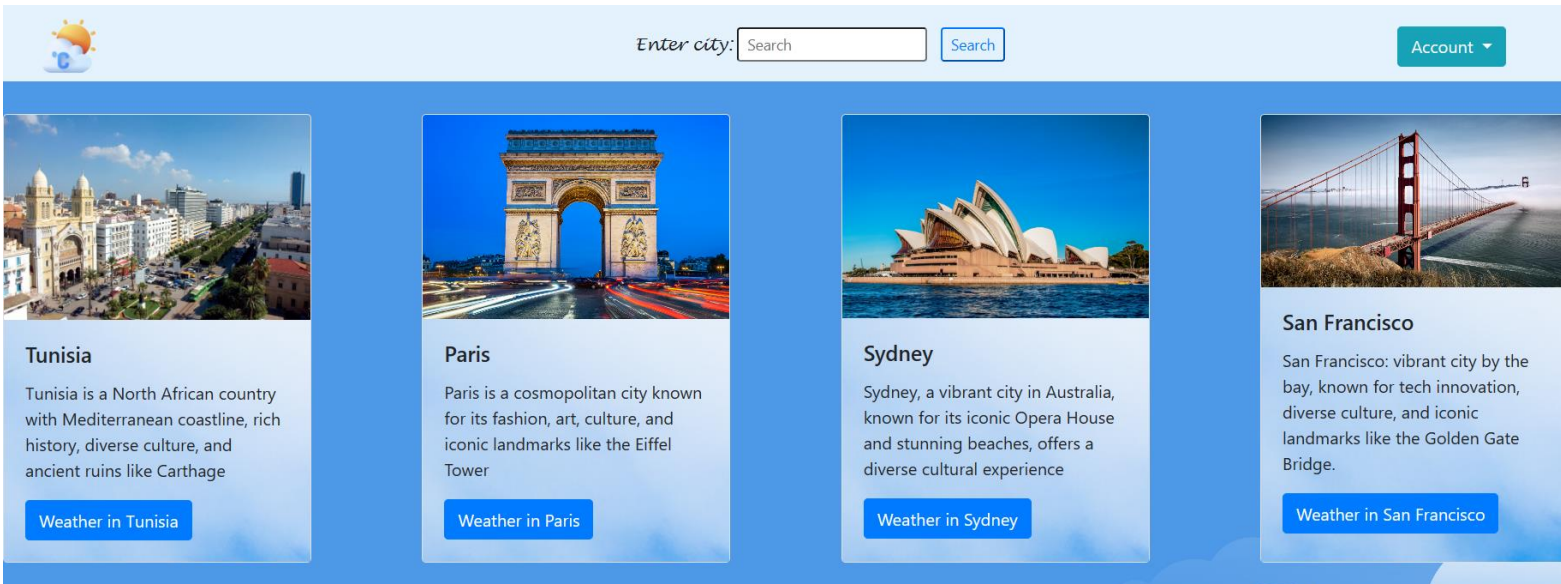
myapp > requirements.txt
1  Flask
2  requests
3  kafka-python
4  pandas
5  numpy
6  cassandra-driver
7  flask-wtf
8  cqlengine
9  schedule

```

Docker container



Home user interface and city search



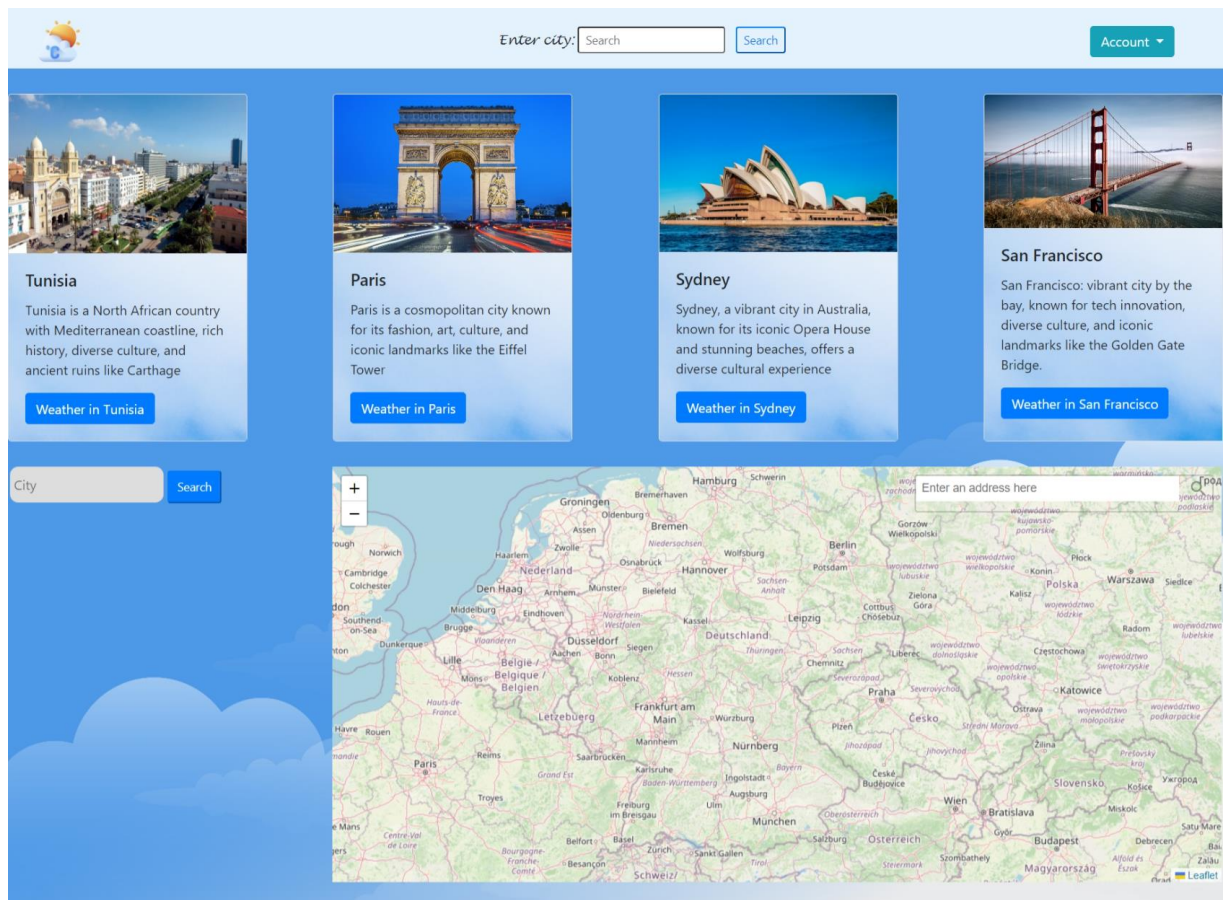
Sprint 2 :

After mastering our development environment, in this sprint we focused on the most important features of our application such as the display of weather data, the suggestion system, map search and authentication.

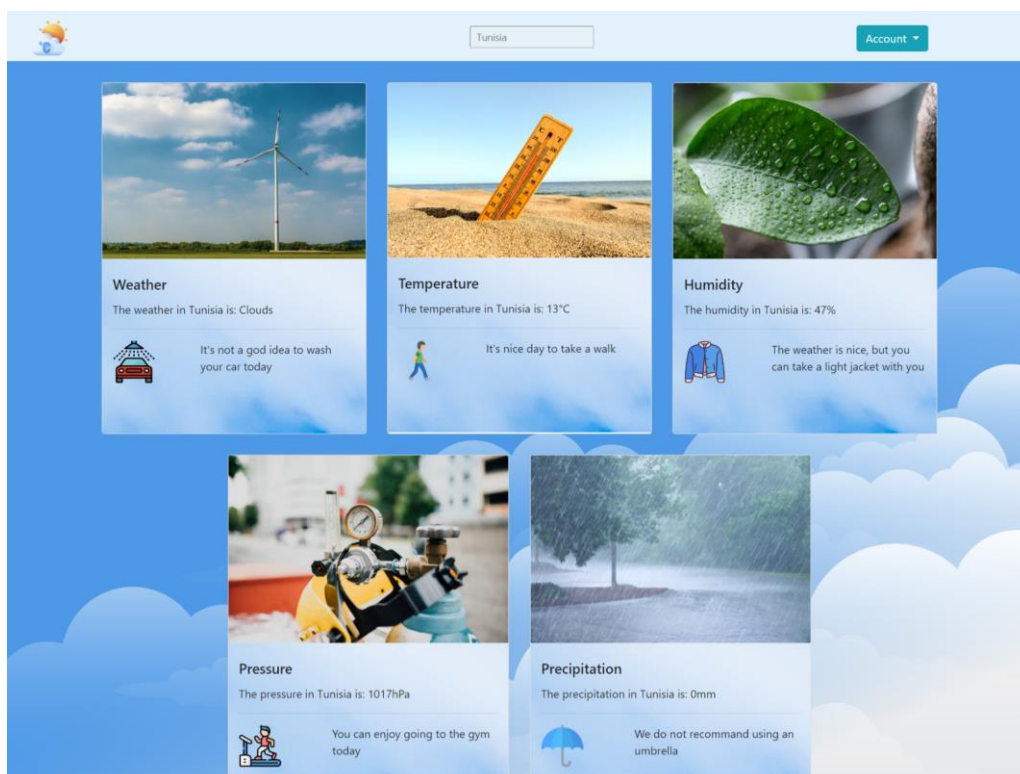
ID	User Story	Tasks	Upcoming User Stories (Sprint3)
1	As a user I can search weather data by city by using a map.	<ul style="list-style-type: none"> - Development of the front-end side of the world map by city - Retrieval of the city from the map 	<ul style="list-style-type: none"> - En tant que abonné je peux avoir une liste des villes favorites - As a subscriber I can receive daily notifications by

		<p>and processing at the back-end level by calling the meteorological data search function</p> <ul style="list-style-type: none"> - Return of data on the front End using the Flask library 	<p>email on my favorite cities with alerts in case of extreme weather conditions.</p> <p>- As a subscriber I can consult the history of my searches by city and days with a maximum of 16 days.</p>
2	As a user I can display the search results and benefit from a suggestion system following the weather.	<ul style="list-style-type: none"> - Retrieval of the entered city where to search by user map - Call to the Open Weather API to determine real-time data - Data processing for suggestion system - Display of requested meteorological data 	
3	- As a user, I can register on the application in order to benefit from advanced features.	<ul style="list-style-type: none"> - Development of the user account creation interface - Creation of the Cassandra database (with the Weather_Key key, and the user table) - Development of the back-end side (class creation and adding functionality to the table using CQLEngine) - Test adding users to the database 	
4	- As a subscriber I can authenticate myself in order to access my account	<ul style="list-style-type: none"> - Development of the user authentication interface - Recovering user data in the back end of the table in order to recover the account - Loading user account data (Back end processing and 	

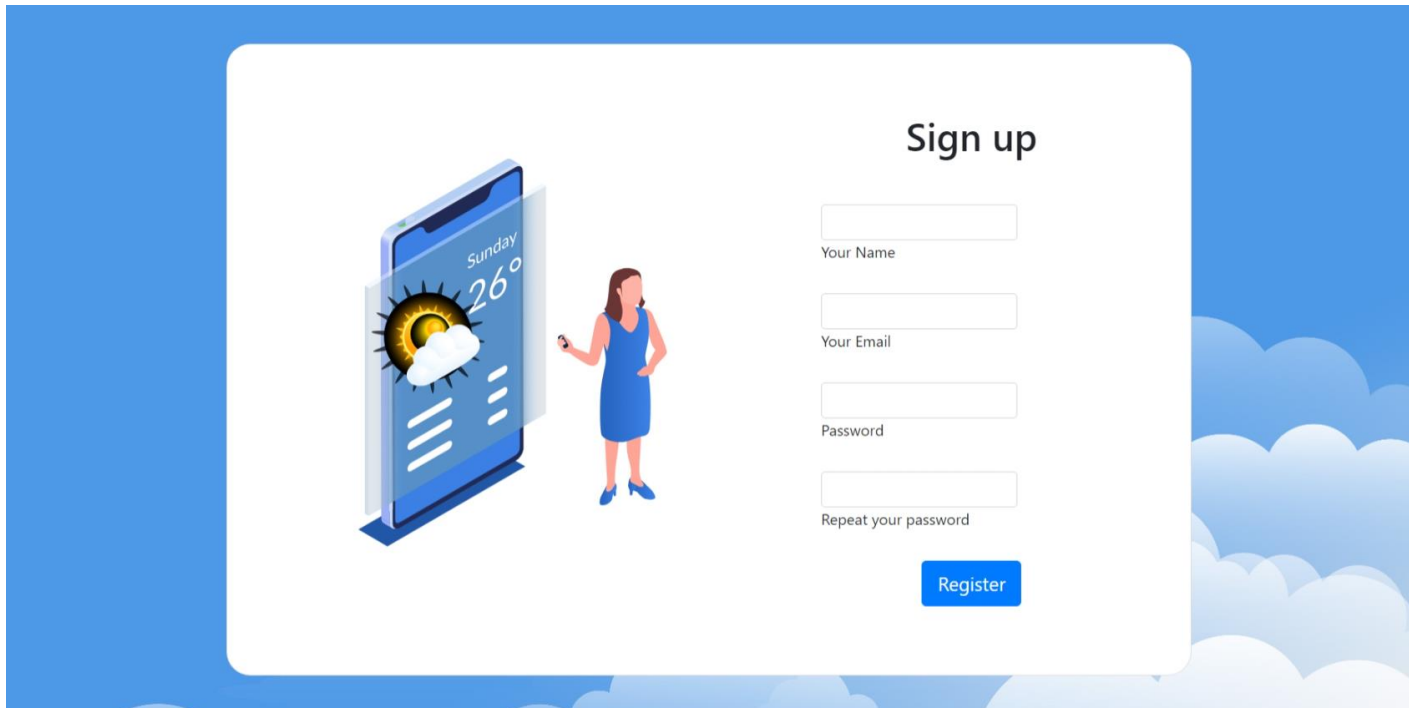
Réalisation : Home user interface and city search with map



Weather data results display interface



Account creation interface



The mockup for the account creation interface features a blue background with a white rounded rectangle in the center. On the left, there is an illustration of a woman in a blue dress standing next to a large smartphone. The phone's screen displays a weather app with a sun and cloud icon, the text 'Sunday', and '26°'. To the right of the illustration, the heading 'Sign up' is positioned above four input fields. The first field is labeled 'Your Name', the second 'Your Email', the third 'Password', and the fourth 'Repeat your password'. A blue 'Register' button is located at the bottom right of the form area.

Sign up

Your Name

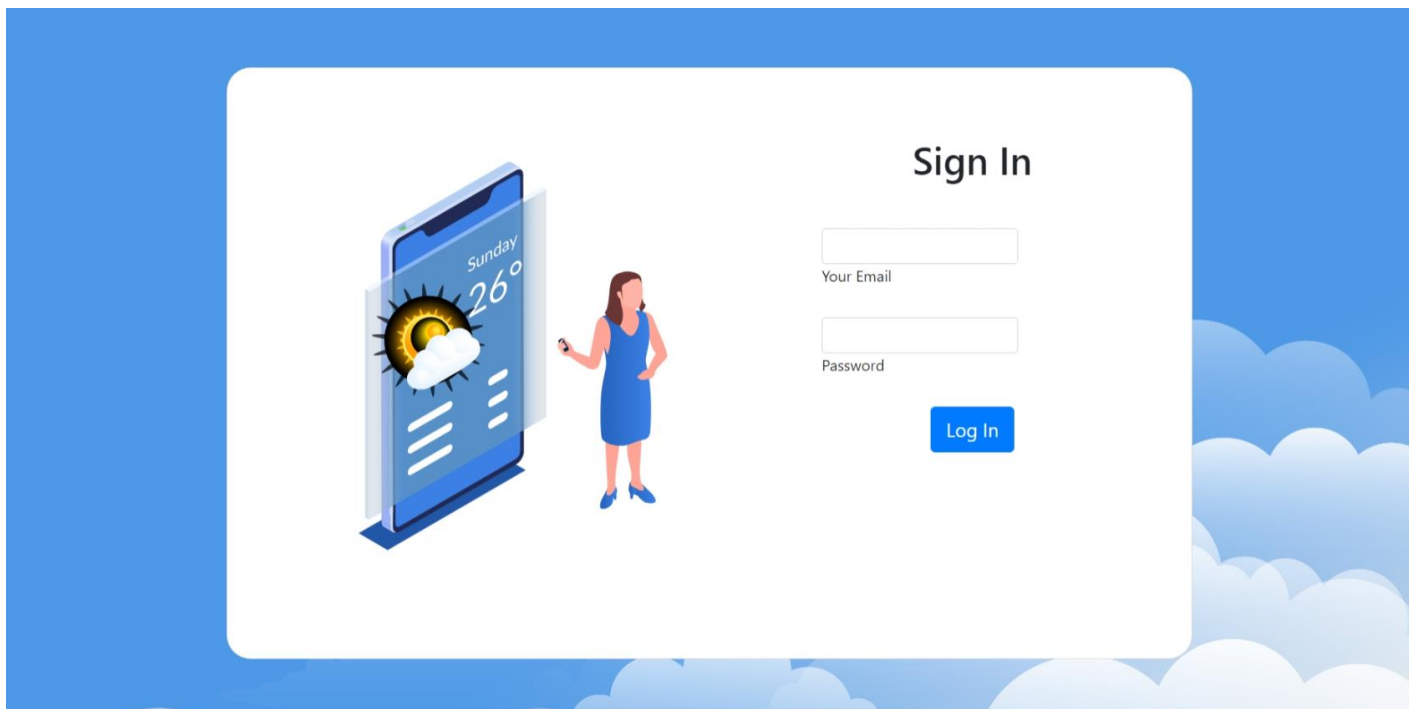
Your Email

Password

Repeat your password

Register

Authentication interface



The mockup for the authentication interface features a blue background with a white rounded rectangle in the center. On the left, there is an illustration of a woman in a blue dress standing next to a large smartphone. The phone's screen displays a weather app with a sun and cloud icon, the text 'Sunday', and '26°'. To the right of the illustration, the heading 'Sign In' is positioned above two input fields. The first field is labeled 'Your Email' and the second 'Password'. A blue 'Log In' button is located at the bottom right of the form area.

Sign In

Your Email

Password

Log In

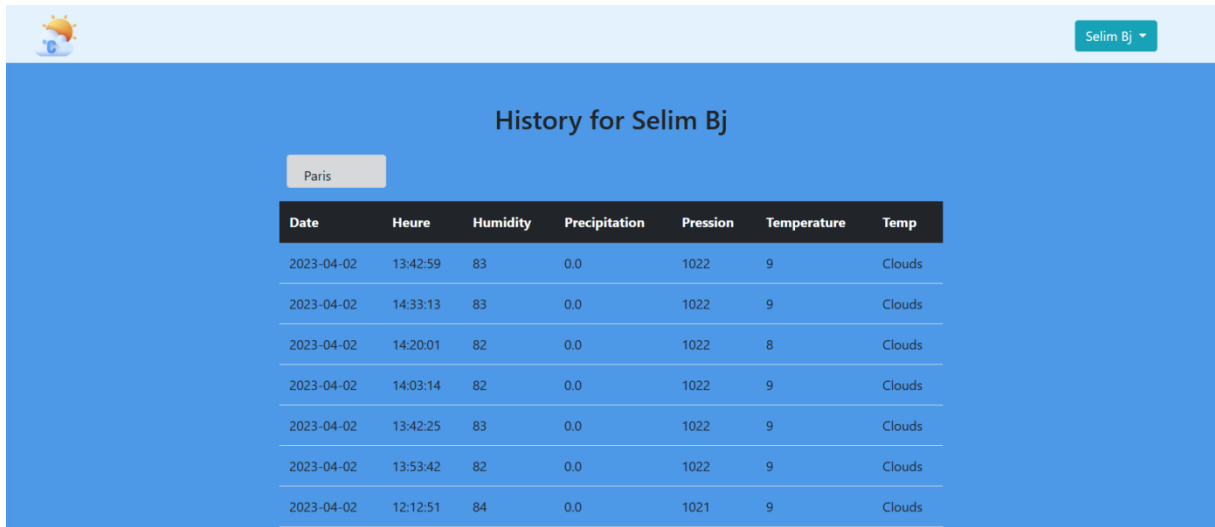
Sprint 3 :

Once we're done with the basic functionality, in this sprint we're addressing the concept of Kafka and its relationship to the other components where it's going to be involved in recording user history in the Weather_data table, and we'll implement the e-mail notification function and the notion of favourites.

ID	User Story	Tasks
1	- As a subscriber I can consult the history of my searches by city and days with a maximum of 16 days.	<ul style="list-style-type: none"> - Development of the back-end side for recording in the database - Development of the topic, kafka producer and consumer and its implementation in our system - Development of back-end processing to ensure the history of only 15 Days for each city - Link between the user table and the weather_data table in order to extract the corresponding user data - Development of the graphical interface for displaying the history by user and by city with dates.
2	- As a subscriber I can have a list of favorite cities	<ul style="list-style-type: none"> - Recovery of the user's favorite city by a button on the front End - Development of the back-end in order to add the user's city to the list of his favorites - Development of the graphical interface to display the list of the user's favorites with its real-time weather data.
3	- As a subscriber I can receive daily notifications by email on my favorite cities with alerts in case of extreme weather conditions.	<ul style="list-style-type: none"> - Development of the back-end processing of the 2nd python file for sending notifications by email using the smtp service. - Retrieval of the list of the user's favorite cities with their meteorological data - Sends email notifications to users every 2 hours - Back-end processing to send recommendations in extreme weather conditions.
4	- Report	- Writing of the explanatory report.

Realisation :

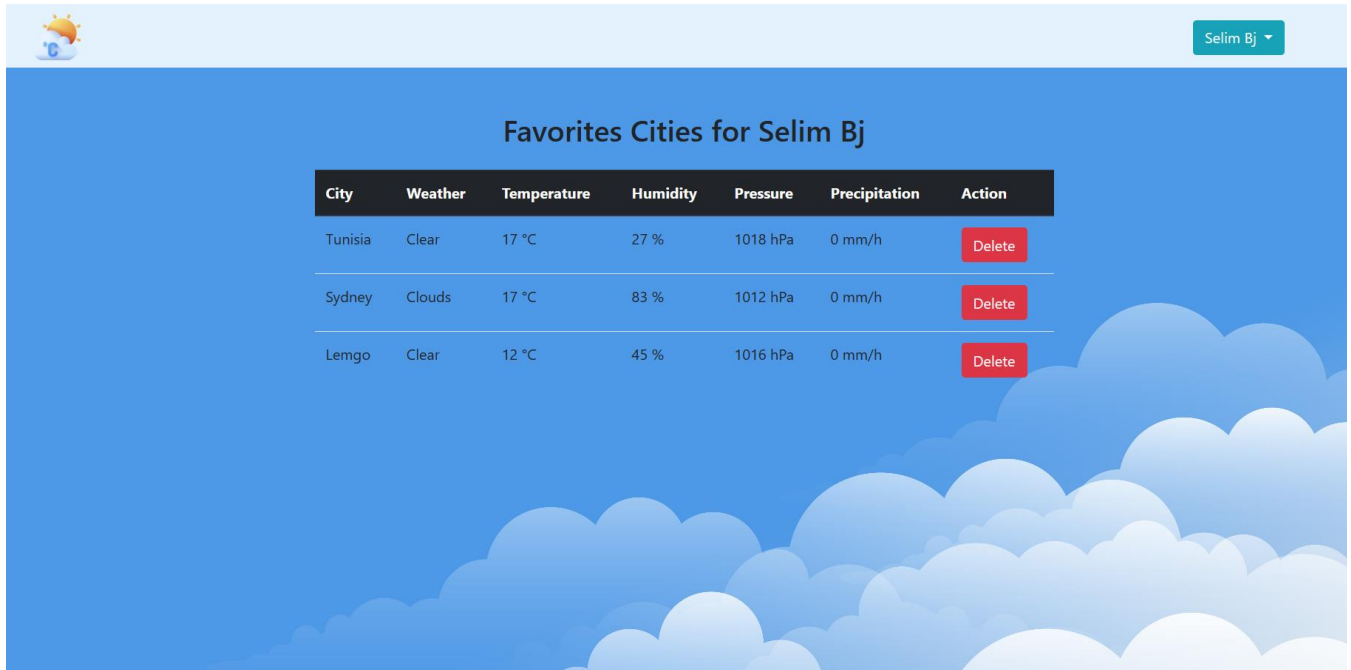
User history display interface



The interface shows a header with a weather icon and a user dropdown menu. The main content area is titled 'History for Selim Bj' and features a city selector set to 'Paris'. Below this is a table with seven columns: Date, Heure, Humidity, Precipitation, Pression, Temperature, and Temp. The table contains eight rows of weather data for Paris on April 2, 2023.

History for Selim Bj						
Paris						
Date	Heure	Humidity	Precipitation	Pression	Temperature	Temp
2023-04-02	13:42:59	83	0.0	1022	9	Clouds
2023-04-02	14:33:13	83	0.0	1022	9	Clouds
2023-04-02	14:20:01	82	0.0	1022	8	Clouds
2023-04-02	14:03:14	82	0.0	1022	9	Clouds
2023-04-02	13:42:25	83	0.0	1022	9	Clouds
2023-04-02	13:53:42	82	0.0	1022	9	Clouds
2023-04-02	12:12:51	84	0.0	1021	9	Clouds

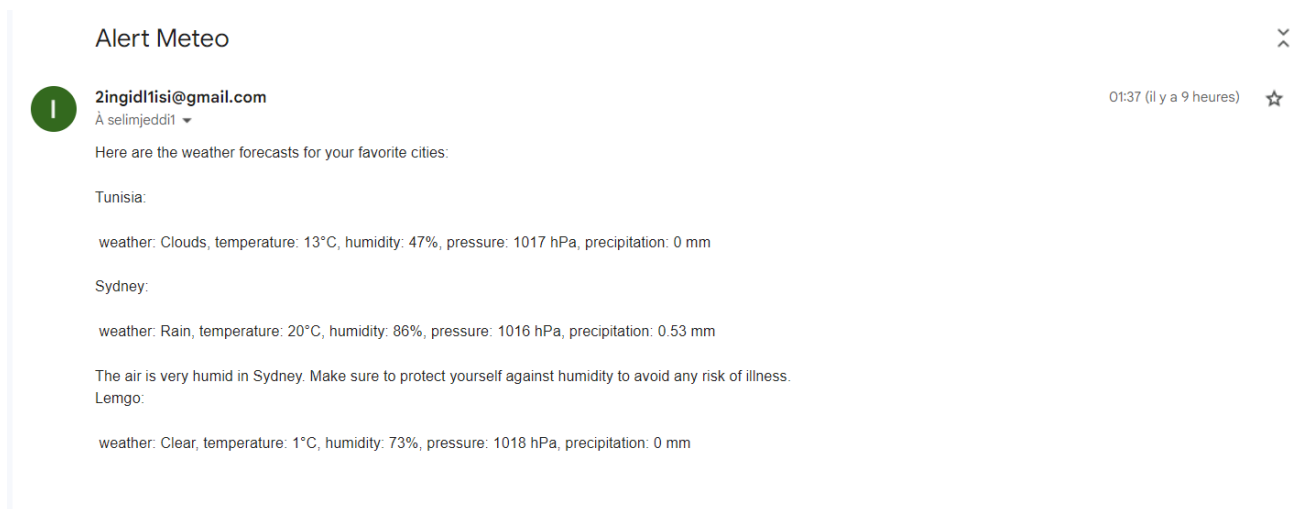
User favorites display interface



The interface shows a header with a weather icon and a user dropdown menu. The main content area is titled 'Favorites Cities for Selim Bj' and contains a table with seven columns: City, Weather, Temperature, Humidity, Pressure, Precipitation, and Action. The table lists three favorite cities: Tunisia, Sydney, and Lemgo, each with a 'Delete' button.

City	Weather	Temperature	Humidity	Pressure	Precipitation	Action
Tunisia	Clear	17 °C	27 %	1018 hPa	0 mm/h	Delete
Sydney	Clouds	17 °C	83 %	1012 hPa	0 mm/h	Delete
Lemgo	Clear	12 °C	45 %	1016 hPa	0 mm/h	Delete

Receiving notifications by email



Conclusion :

In conclusion, this Distributed Systems project made it possible to put into practice theoretical knowledge acquired during engineering training. The choice of modern technologies such as Docker, Python Flask, Kafka and Cassandra made it possible to develop a Weather web application with advanced features such as email notifications, history, recommendation system and flow management. The Scrum method was used to plan and organize the work in sprints, which made it possible to better manage the time and the tasks to be carried out.