linea orizzontale

**Dudniak Oleksandr - Stage svolto in Magenta Srl.**

**8-26 novembre 2021**

# AreaGraph

## OBIETTIVI

### Studiare, progettare e sviluppare un'applicazione software per la visualizzazione su web di dati relativi alla qualità dell’aria, ricavati da un database.

# 

## DESCRIZIONE

Il risultato finale e’ stato semplificato a causa del poco tempo disponibile, ma racchiude tutti i punti dell’obiettivo, ovvero:

### Creare un software per la visualizzazione **web**:

### Selezionare una **centralina** tra quelle disponibili (su una **mappa** integrata)

### Scegliere un **intervallo di tempo** a piacere

### Dati i valori giusti, ottenere i dati dal backend che li richiede al database primario

### Visualizzare in un **grafico** i dati relativi agli **inquinanti dell’aria** nel periodo scelto

Il design del sito e’ stato creato usando [Bootstrap 4](https://getbootstrap.com/docs/4.0/getting-started/introduction/) e CSS puro,

quasi del tutto *responsive* e *client-friendly*.

Per il *backend* e’ stato utilizzato il framework [**Flask**](https://flask.palletsprojects.com/) di[**Python**](https://www.python.org/)**,**

E per il *frontend* e’ stato utilizzato il framework [**AngularJS**](https://angularjs.org/) **(V13.0.1)**, di [**NodeJS**](https://nodejs.org/it/)

# 

# 

## DATABASE

Utilizza [***postgreSQL***](https://www.postgresql.org/)(e Timescale DB)

### Tabelle e view necessari per lo sviluppo:

* Schema **public** :
* **Tables**>**station**:
  + Informazioni sulle centraline
* **Tables**>**sensor:**
  + Informazioni sui sensori
* **Views**>**station\_data\_hourly\_avg** :
  + **bucket**: *data dell'inserimento dati, in formato UTC che comprende valori raccolti dall’ora precedente fino a quella inserita*
  + **station\_id***: riferimento esterno alla chiave primaria della tabella station*
  + **sensor\_id***riferimento esterno alla chiave primaria della tabella sensor*

# 

## AMBIENTE DI SVILUPPO & OS

L’**IDE** principalmente utilizzato nel progetto e’ [**VSCode**](https://code.visualstudio.com/), con la grande disponibilità di estensioni di utility (*come l’intellisense di un linguaggio, o l’integrazione di github, ecc.* ), e il terminale integrato.

Per quanto riguarda il **sistema operativo**, e’ stato utilizzato per la maggior parte **Windows 10**, ma per fare test prima del processo di deploy ,che usa [**gunicorn**](https://gunicorn.org/) (modulo di python), e’ stato utilizzato **Linux** perche’ e’ disponibile solo su questo OS.

## 

## 

## STRUTTURA DEL REPOSITORY

*Le 3 principali cartelle sono*:

### **Backend**

* + **backend**: *cartella contenente tutta la parte logica*
  + **.env** : *file che contiene variabili d’ambiente per python*
  + **main.py** : *file per avviare l’app localmente per test*
  + **requirements.txt** : *moduli necessari per python*

### **Frontend**:

* + *Ha la struttura di un generico progetto Angular, con la separazione in varie cartelle di*:
    - **Componenti :**
      * **~Primari~**
      * **~Secondari~**
    - **~Servizi~**
    - Variabili d’ambiente (~**environments~**)
      * **\*Importante\*:**
        + **Seguire le istruzioni del file README.md** (nella root directory del repository) **per creare il file mancante**
    - **~Assets~** : file statici, come il logo.

### **Heroku**:

* + *cartella creata per velocizzare il processo di deployment.*

*La cartella* ***extra*** *contiene dei file d’immagine del logo e favicon, e il loro progetto Photoshop*

# 

## BACKEND

La parte del backend e' stata strutturata in modo che sia facile da essere caricata su Heroku.

Ecco perché contiene la cartella **backend.**

### All’interno di questa cartella, si trova:

#### Cartelle **static** e **templates :** *servono solamente se si avvia il backend in locale*, dove **static** contiene l’icona del sito, e **templates** contiene una semplice pagina html (questi si vedono andando al root path del sito).

#### **\_\_init\_\_.py** : file python che fa diventare la cartella **backend** un modulo di python, e contiene:

* + Inizializzazione dell’applicazione **Flask**
  + Inizializzazione di [**Flask Restful**](https://flask-restful.readthedocs.io/) e riferimento delle classi (che gestiscono le richieste) a un endpoint
  + Inizializzazione di **CORS** ([Cross-Origin Resource Sharing](https://en.wikipedia.org/wiki/Cross-origin_resource_sharing))
  + Endpoint dell’**autorizzazione** al backend, che usa [**JWT**](https://jwt.io/)
  + E meno importante, la customizzazione dei messaggi d’errore

#### **controllers.py** : file python che racchiude la parte principale della logica:

* + Caricamento delle variabili d’ambiente dal file **.env**
  + *Funzione* di **accesso** e **disconnessione** dal database, quindi il collegamento viene eseguito quando serve e chiuso alla fine dell’operazione.
  + *Funzione* dell’**esecuzione della query**
  + *Funzione* che controlla che alla richiesta dell’endpoint ci sia un **token** di autorizzazione e che sia **valido**
  + *Funzioni* utili per la gestione/validazione delle stazioni
  + ***Classi*** che gestiscono le **richieste** ai vari **endpoint**

#### **queries.py** : file python che contiene tutte le query da richiedere al database principale, creato principalmente per una migliore lettura del file **controllers.py**

# 

# 

## FRONTEND

Come scritto in precedenza, ha la struttura di un generico progetto Angular, quindi c’e’:

* **src/app/routes/ :** Sono i componenti *primari*/**routes** dell’app, che a loro volta contengono =>
* **src/app/components/ :** Componenti secondari, che possono essere riutilizzati in diverse parti ( ad esempio la **navbar** ).
* **src/app/services/ :** Servizi **TitleManagementService** e **HttpRequestService** , dove il primo si occupa di gestire il titolo della pagina, e il secondo di effettuare le richieste http al backend e ottenere dati.
  + L'utilità di questi servizi e’ quella di creare una sola istanza (quindi non sprecare memoria) che condivide stessi attributi e metodi tra diversi componenti (sincronizzato).
* **src/environments/ :** Variabili d’ambiente.
* **src/utils/** contiene:
  + Le **interfacce** per **tipizzare** i dati ricevuti dalle richieste http
  + La **classe** utile per creare il grafico di **echarts**
* **src/assets/ :** File statici (immagini), come il logo del sito,ecc. .

## PROCESSO DEL DEPLOY

Per il **deploy** e’ stato scelto [**Heroku**](https://www.heroku.com/), un servizio cloud per creare e gestire applicazioni (dentro dei container), con il piano ***gratuito***.

Usa **git** per gestire le applicazioni.

Il piano ***gratuito*** offre **550** ore *complessive* di utilizzo al mese (che si possono controllare [qui](https://dashboard.heroku.com/account/billing))

Fortunatamente per non sprecare le ore, l’app va in **idle mode/standby** dopo **30 minuti di inutilizzo**, e si riattiva automaticamente quando viene utilizzata.

### **PASSAGGI**:

#### Registrarsi sul [sito](https://signup.heroku.com/login)

#### Installare [Heroku CLI](https://devcenter.heroku.com/articles/heroku-cli#download-and-install)

#### Eseguire il login dal *terminale* [ **heroku login** ]

#### Dentro la cartella **Heroku/** creare un nuovo repository git [ **git init** ]

#### Sempre nella cartella creare l’app di heroku:

* + **heroku create <nome dell’app> --region=eu**
    - **Attenzione**: Se si crea prima l’app, per assegnare il repository ad heroku eseguire:
    - **heroku git:remote -a <nome dell’app>**

#### Creare e configurare il file **environment.dev.ts** in **Frontend/src/environments/** con il seguente codice:

| export const environment = {  apiUrl: 'https://<nome dell'app di heroku>.herokuapp.com/api/',  user: '<**jwt\_user** del file .env>',  passw: '<**jwt\_password** del file .env>',  }; |
| --- |

#### Creare e configurare il file **.env** in **Backend/**

(prendendo di riferimento .env-example):

| #Database  host="<host>"  port="<port>"  db="<database name>"  user="<username>"  password="<password>"  #------------------------------------------------  #JWT  secret\_key="<secret key for pyJWT>"  #Credentials  jwt\_user="<username>"  jwt\_password="<password>"  #------------------------------------------------ |
| --- |

#### Da **Backend/** copiare:

* la cartella **backend**
* il file **.env**
* dentro la cartella **Heroku/**

#### In **Frontend/** eseguire:

#### **ng build --prod --build-optimizer**

#### Copiare tutti i file

* da **Frontend/dist/Frontend**
* a **Heroku/frontend**

#### Aggiornare il repository:

* + **git add --all**
  + **git commit -m `Initial commit`**

#### Caricare i dati su Heroku:

* + **git push heroku master**