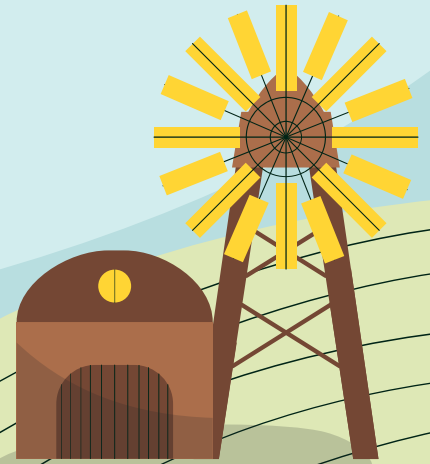
Air Quality Insights: Explore Pollution Levels in Real-Time

World Air Quality Index by City and Coordinates[1]

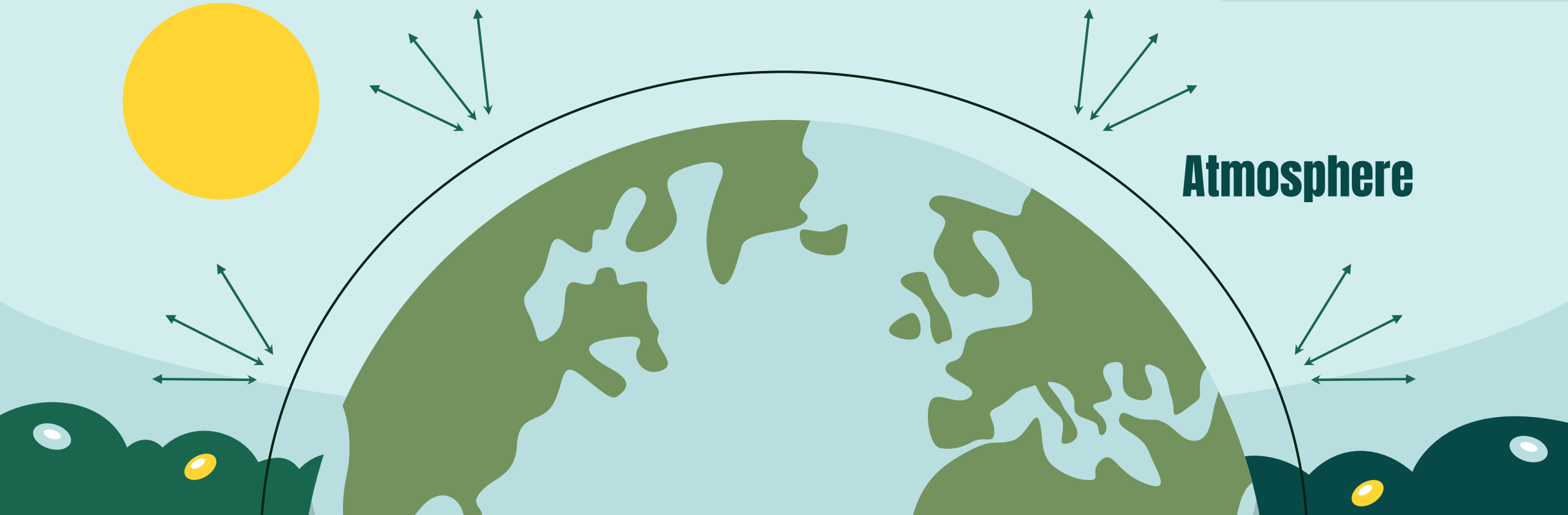
Progetto di Basi Dati II
Realizzato da
Angela De Martino
0522501589



[1] https://www.kaggle.com/datasets/adityaramachandran27/world-air-quality-index-by-city-and-coordinates?resource=download

Il problema

L'inquinamento **atmosferico** è una seria minaccia per la salute umana e l'ambiente a causa delle emissioni inquinanti da attività umane, con effetti negativi sui polmoni, il sistema cardiovascolare e il clima globale. La raccolta di dati sull'inquinamento atmosferico è essenziale per monitorare l'impatto e **adottare misure efficaci per ridurlo.**

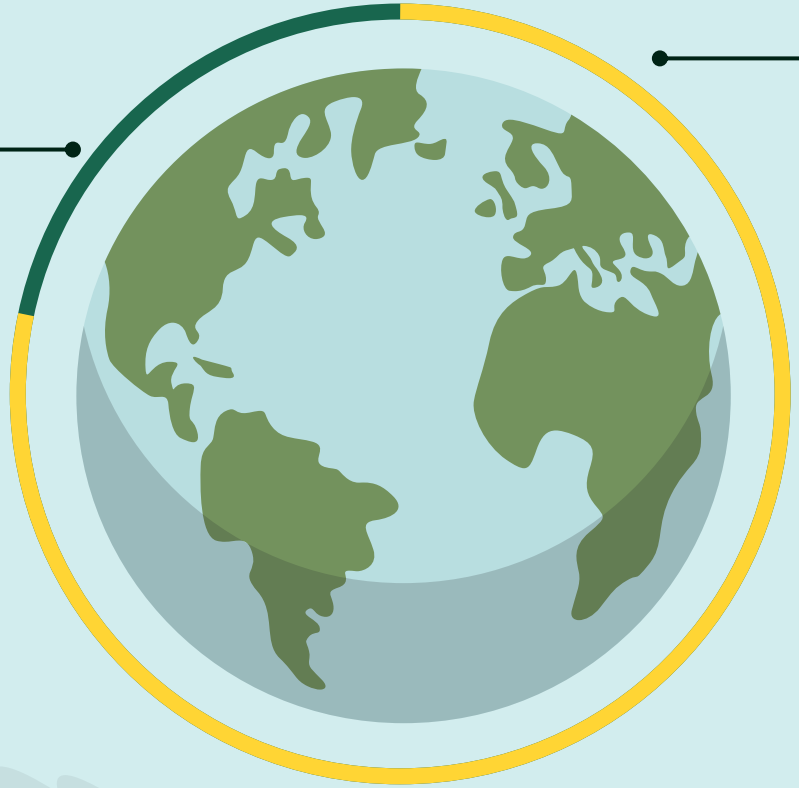


Qual è l'obiettivo?

Il progetto si propone di sviluppare una web app con l'obiettivo **di fornire un supporto alle istituzioni nel processo decisionale volto alla riduzione dell'inquinamento atmosferico nelle aree ad elevata contaminazione**. Attraverso un'accurata valutazione dell'impatto ambientale e sulla salute umana, gli enti saranno in grado di adottare provvedimenti basati su evidenze concrete per affrontare il fenomeno. L'obiettivo finale del progetto è quindi l'implementazione di una soluzione efficace volta alla riduzione dell'inquinamento atmosferico, contribuendo così al miglioramento della salute umana e del benessere ambientale.

Fusione dei set di dati

Informazioni sulle città con le corrispondenti coordinate di latitudine e longitudine



Contente i dati sui livelli di inquinamento atmosferico nei pasi di tutto il mondo.

Struttura del dataset

16696 tuple

AQI and Lat Long of Countries.csv (1.41 MB)

Detail | Compact | Column

10 of 14 columns

Country	City	AQI Value	AQI Categ...	CO AQI Va...	CO AQI Ca...	Ozone AQI...
Russian Federation	Praskoveya	51	Moderate	1	Good	36
Brazil	Presidente Dutra	41	Good	1	Good	5
Brazil	Presidente Dutra	41	Good	1	Good	5
Italy	Priolo Gargallo	66	Moderate	1	Good	39
Poland	Przasnysz	34	Good	1	Good	34
United States of America	Punta Gorda	54	Moderate	1	Good	14
United States of America	Punta Gorda	54	Moderate	1	Good	14

14 colonne

Spiegazione dei termini comuni dell'indice di qualità dell'aria (AQI)

PM2,5

Si riferisce a minuscole particelle o goccioline presenti nell'aria con un'ampiezza pari o inferiore a 2,5 micrometri. Possono essere dannose per la salute umana se inalate, soprattutto in alte concentrazioni.

Ozono (O3)

L'ozono è un gas che può formarsi nell'atmosfera attraverso una reazione chimica tra la luce solare e altri inquinanti. Alti livelli di ozono possono essere dannosi per la salute umana, in particolare per chi ha problemi respiratori.

Monossido di carbonio (CO)

Il CO è un gas incolore e inodore prodotto dalla combustione incompleta dei combustibili fossili. Livelli elevati di CO possono essere tossici per l'uomo e possono causare mal di testa, vertigini e nausea.

Colonne del dataset

1. **Country**
Rappresenta il nome del paese associato ai dati relativi all'indice di qualità dell'aria (AQI).

2. **City**
Indica il nome della città a cui si riferiscono i dati sull'indice di qualità dell'aria.

3. **AQI Value**
È il valore numerico dell'indice di qualità dell'aria per una determinata città o area.

4. **AQI Category**
Indica la categoria di qualità dell'aria associata al valore dell'indice di qualità dell'aria.

5. **CO AQI Value:**
Rappresenta il valore numerico dell'indice di qualità dell'aria per il monossido di carbonio (CO) per una città specifica.

6. **CO AQI Category**
Indica la categoria di qualità dell'aria associata al valore dell'indice di qualità dell'aria per il monossido di carbonio (CO).

7. **Ozone AQI Value**
Rappresenta il valore numerico dell'indice di qualità dell'aria per l'ozono (O3) per una determinata città o area.

8. **Ozone AQI Category**
Indica la categoria di qualità dell'aria associata al valore dell'indice di qualità dell'aria per ozono (O3).

9. **NO2 AQI Value**
Rappresenta il valore numerico dell'indice di qualità dell'aria per l'NO2 per una determinata area o città.

10. **NO2 AQI Category**
Indica la categoria di qualità dell'aria associata al valore dell'indice di qualità dell'aria per l'NO2.

11. **PM2.5 AQI Value**
Rappresenta il valore numerico dell'indice di qualità dell'aria per il PM2.5 per una determinata città o area.

12. **PM2.5 AQI Category**
Indica la categoria di qualità dell'aria associata al valore dell'indice di qualità dell'aria per il PM2.5.

13. **lat**
Rappresenta la latitudine geografica della città o dell'area corrispondente.

14. **lng**
Rappresenta la longitudine geografica della città o dell'area corrispondente.

Organizzazione dei file

basi_dati_II

back-end

front-end
bd_projects_frontend

AirQuality.py → La classe rappresenta i dati di qualità dell'aria e offre funzioni per la manipolazione

DBConnectionPool.py → Viene creata una connessone al db MongoDB sulla porta 27017

CleanDataset.py → Viene ripulito il file CSV e viene prodotto un nuovo file "cleaned"

Pages.py → Crea un'applicazione web utilizzando Flask.

QueryList.py → Sono definite tutte le query sul database consentendo l'inserimento, l'aggiornamento e la cancellazione

Perchè MongoDB?

Scalabilità orizzontale

Consente di scalare orizzontalmente il database, distribuendo i dati per gestire carichi di lavoro elevati e garantire prestazioni ottimali.

Flessibilità dello schema

MongoDB è un database non relazionale basato su documenti, che consente di aggiungere, rimuovere o modificare campi dei documenti senza dover seguire uno schema rigido.

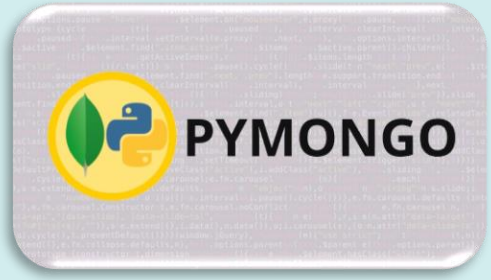
Alta velocità e prestazioni

È progettato per garantire alte prestazioni, grazie al suo modello di dati orientato ai documenti e all'ottimizzazione per operazioni di lettura e scrittura veloci.

Facilità d'integrazione

Fornisce driver e librerie di supporto per una vasta gamma di linguaggi di programmazione, semplificando l'integrazione con le applicazioni web.

DBConnectionPool.py




```python
import pymongo
#Creo una connessione al database locale MongoDB eseguito sulla porta 27017

15 usages
def connection_pool():
        client = pymongo.MongoClient("mongodb://localhost:27017")
        #accedo alla collezione presente all'interno del db AirQuality
        col = client["Air_Quality"]["Air_Quality"]
        #restituisco la collezione
        return col
```

Pymongo è una libreria Python che consente di comunicare con il database MongoDB e di interagire con esso attraverso il linguaggio di programmazione Python. È uno strumento potente e molto popolare per sviluppare applicazioni Python che utilizzano MongoDB come database sottostante.

Offre un'API intuitiva e ricca di funzionalità che consente di sfruttare al meglio le potenzialità di MongoDB e sviluppare applicazioni robuste e scalabili.

Classe AirQuality.py

Metodo Init (inizializzo gli attributi)

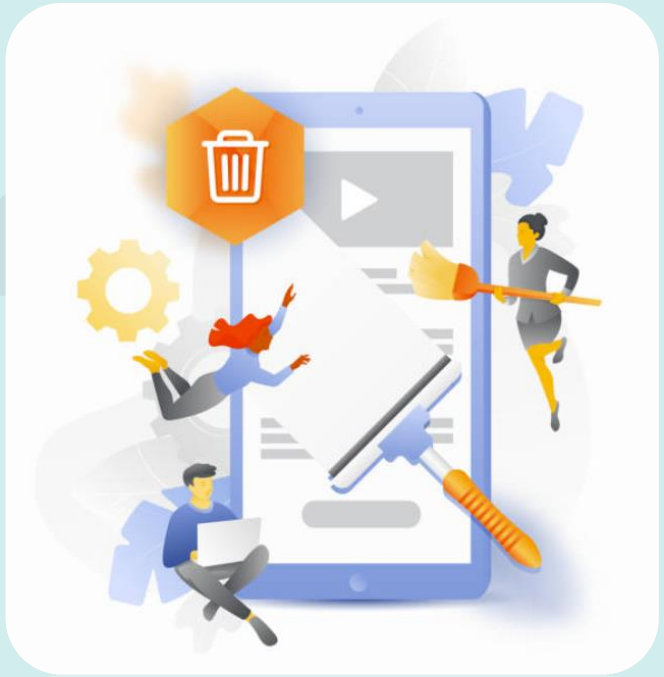
```python
# Class AirQuality per il dataset
5 usages
class AirQuality:
    #Inizializzo gli attributi dell'oggetto AirQuality utilizzando i
    def __init__(self, air_quality):
        self.id = air_quality["_id"]
        self.Country = air_quality["Country"]
        self.City = air_quality["City"]
        self.aqi_value = air_quality["AQI Value"]
        self.aqi_category = air_quality["AQI Category"]
        self.co_aqi_value = air_quality["CO AQI Value"]
        self.co_aqi_category = air_quality["CO AQI Category"]
        self.ozone_aqi_value = air_quality["Ozone AQI Value"]
        self.ozone_aqi_category = air_quality["Ozone AQI Category"]
        self.no2_aqi_value = air_quality["NO2 AQI Value"]
        self.no2_aqi_category = air_quality["NO2 AQI Category"]
        self.pm25_aqi_value = air_quality["PM2_5 AQI Value"]
        self.pm25_aqi_category = air_quality["PM2_5 AQI Category"]
        self.lat = air_quality["lat"]
        self.lng = air_quality["lng"]
```

Metodo dump (restituisce l'oggetto AirQuality come dizionario)

```python
def dump(self):
    # Restituisci l'oggetto AirQuality come un dizionario
    return {
        #"id": self.id,
        "Country": self.Country,
        "City": self.City,
        "AQI Value": self.aqi_value,
        "AQI Category": self.aqi_category,
        "CO AQI Value": self.co_aqi_value,
        "CO AQI Category": self.co_aqi_category,
        "Ozone AQI Value": self.ozone_aqi_value,
        "Ozone AQI Category": self.ozone_aqi_category,
        "NO2 AQI Value": self.no2_aqi_value,
        "NO2 AQI Category": self.no2_aqi_category,
        "PM2_5 AQI Value": self.pm25_aqi_value,
        "PM2_5 AQI Category": self.pm25_aqi_category,
        "lat": self.lat,
        "lng": self.lng
    }
```

checkFormato verifica se l'oggetto è stato correttamente inizializzato

Pulizia dei dati



La libreria Pandas è stata usata per pulire un dataset sulla qualità dell'aria.
Sono state eliminate le righe che contenevano valori mancanti e poi sono stati salvati in un nuovo file CSV pulito.
Questa operazione è utile per assicurarsi che i dati siano completi e coerenti prima di analizzarli o visualizzarli.

pandas

Flask

Flask è stato utilizzato come framework per lo sviluppo del backend dell'applicazione.
È stato impiegato per gestire le richieste HTTP provenienti dal frontend, definendo le route dell'applicazione che stabiliscono quali azioni eseguire in risposta a una specifica richiesta.
Inoltre, Flask ha fornito funzionalità per la gestione dei template, consentendo la generazione dinamica delle pagine web.

Inserimento pages.py

```python
# Inserimento
@app.route("/test_query/insert_country", methods=['POST', 'GET'])
def insert_country():
    country = request.form['ins-country']
    city = request.form['ins-city']
    aqi_value = request.form['ins-aqi-value']
    aqi_category = request.form['ins-aqi-category']
    co_aqi_value = request.form['ins-co-value']
    co_aqi_category = request.form['ins-co-category']
    ozone_aqi_value = request.form['ins-ozone-value']
    ozone_aqi_category = request.form['ins-ozone-category']
    no2_aqi_value = request.form['ins-no2-value']
    no2_aqi_category = request.form['ins-no2-category']
    pm25_aqi_value = request.form['ins-pm25-value']
    pm25_aqi_category = request.form['ins-pm25-category']
    lat = request.form['ins-lat']
    lng = request.form['ins-lng']

    # Crea un dizionario con i valori estratti
    object = {
        "_id": "",
        "Country": country,
        "City": city,
        "AQI Value": aqi_value,
        "AQI Category": aqi_category,
        "CO AQI Value": co_aqi_value,
        "CO AQI Category": co_aqi_category,
        "NO2 AQI Value": no2_aqi_value,
        "NO2 AQI Category": no2_aqi_category,
        "PM2_5 AQI Value": pm25_aqi_value,
        "PM2_5 AQI Category": pm25_aqi_category,
        "lat": lat,
        "lng": lng
    }

    #creo l'oggetto AirQuality ed invoco checkFormato
    air_quality = AirQuality.AirQuality(object)
    if AirQuality.checkFormato(air_quality):
        Query.insertCountry(air_quality)
        return render_template("test_query.html", response="Paese inserito correttamente", list_air=fullTable())
    else:
        return render_template("test_query.html", response="Valori mancanti", list_air=getResult())
```

```python
# InsertCounry mi permette di inserisci un paese con tutti i campi necessari per la memorizzazione in MongoDB
1 usage
def insertCountry(country):
    col = Cp.connection_pool()
    col.insert_one({
        "Country": country.Country,
        "City": country.City,
        "AQI Value": country.aqi_value,
        "AQI Category": country.aqi_category,
        "CO AQI Value": country.co_aqi_value,
        "CO AQI Category": country.co_aqi_category,
        "Ozone AQI Value": country.ozone_aqi_value,
        "Ozone AQI Category": country.ozone_aqi_category,
        "NO2 AQI Value": country.no2_aqi_value,
        "NO2 AQI Category": country.no2_aqi_category,
        "PM2_5 AQI Value": country.pm25_aqi_value,
        "PM2_5 AQI Category": country.pm25_aqi_category,
        "lat": country.lat,
        "lng": country.lng
    })
```

QueryList.py

Aggiornamento pages.py

```python
@app.route("/test_query/update_country", methods=['POST', 'GET'])
def update_country():
    country = request.form['upd-country']
    new_country = request.form['upd-country2']
    city = request.form['upd-city']
    aqi_value = request.form['upd-aqi-value']
    aqi_category = request.form['upd-aqi-category']
    co_value = request.form['upd-co-value']
    co_category = request.form['upd-co-category']
    ozone_value = request.form['upd-ozone-value']
    ozone_category = request.form['upd-ozone-category']
    no2_value = request.form['upd-no2-value']
    no2_category = request.form['upd-no2-category']
    pm25_value = request.form['upd-pm25-value']
    pm25_category = request.form['upd-pm25-category']
    lat = request.form['upd-lat']
    lng = request.form['upd-lng']

    # Creazione dell'oggetto per l'aggiornamento
    obj = {
        "_id": "",
        "Country": country,
        "City": city,
        "AQI Value": aqi_value,
        "AQI Category": aqi_category,
        "CO AQI Value": co_value,
        "CO AQI Category": co_category,
        "Ozone AQI Value": ozone_value,
        "Ozone AQI Category": ozone_category,
        "NO2 AQI Value": no2_value,
        "NO2 AQI Category": no2_category,
        "PM2_5 AQI Value": pm25_value,
        "PM2_5 AQI Category": pm25_category,
        "lat": lat,
        "lng": lng
    }
    print(obj)
    if country == "":
        return render_template("test_query.html", response="Errore! Nessun paese inserito.", list_air=getResult())
    else:
        # Trova i dati nel database
        result = Query.findCountryByName(country)
        value = []
        new = AirQuality.AirQuality(obj)
        for i in result:
            value.append(AirQuality.AirQuality(i))
        if len(value) != 0:
            if new_country == "":
                new_country = country
            if new.Country == "":
                new_country = value.Country
                new.Country = value.Country
            new.co_aqi_value = value[0].co_aqi_value
        if new.co_aqi_category == "":
            new.co_aqi_category = value[0].co_aqi_category
        if new.ozone_aqi_value == "":
            new.ozone_aqi_value = value[0].ozone_aqi_value
        if new.ozone_aqi_category == "":
            new.ozone_aqi_category = value[0].ozone_aqi_category
        if new.no2_aqi_value == "":
            new.no2_aqi_value = value[0].no2_aqi_value
        if new.no2_aqi_category == "":
            new.no2_aqi_category = value[0].no2_aqi_category
        if new.pm25_aqi_value == "":
            new.pm25_aqi_value = value[0].pm25_aqi_value
        if new.pm25_aqi_category == "":
            new.pm25_aqi_category = value[0].pm25_aqi_category
        if new.lat == "":
            new.lat = value[0].lat
        if new.lng == "":
            new.lng = value[0].lng

        # Aggiorna i dati nel database
        Query.updateCountry(new_country, new)
        print("Aggiornati")
        return render_template("test_query.html", response="Aggiornamento effettuato", list_air=fullTable())
    else:
        return render_template("test_query.html", response="Aggiornamento non effettuato.",
                               list_air=getResult())
        print("Non aggiornati")
```

```python
# UpdateCountry è la funzione per aggiornare un paese in base al nome della città indicata
1 usage
def updateCountry(country, country1):
    col = Cp.connection_pool()
    myquery = {"Country": country}
    newvalues = {
        "$set": {
            "Country": country1.Country,
            "City": country1.City,
            "AQI Value": country1.aqi_value,
            "AQI Category": country1.aqi_category,
            "CO AQI Value": country1.co_aqi_value,
            "CO AQI Category": country1.co_aqi_category,
            "Ozone AQI Value": country1.ozone_aqi_value,
            "Ozone AQI Category": country1.ozone_aqi_category,
            "NO2 AQI Value": country1.no2_aqi_value,
            "NO2 AQI Category": country1.no2_aqi_category,
            "PM2_5 AQI Value": country1.pm25_aqi_value,
            "PM2_5 AQI Category": country1.pm25_aqi_category,
            "lat": country1.lat,
            "lng": country1.lng
        }
    }
    col.update_one(myquery, newvalues)
```

QueryList.py

Ricerca pages.py

```python
# Ricerca paese
@app.route("/test_query/find_query", methods=['POST', 'GET'])
def find_query():
    # Verifica se la richiesta è di tipo POST
    if request.method == 'POST':
        # Estrae i valori dai campi del modulo inviato con la richiesta
        country = request.form['find-country']
        city = request.form['find-city']
        aqi_min = request.form['find-aqi-min']
        aqi_max = request.form['find-aqi-max']
        aqi_category = request.form['find-aqi_category']
        co_min = request.form['find-co-min']
        co_max = request.form['find-co-max']
        aqi_co_category = request.form['find-aqi-co_category']
        lat = request.form['find-lat']
        lng = request.form['find-lng']
temp1 = 0
temp2 = 0
# Se il nome del paese rappresenta una stringa vuota viene assegnato un dizionario {"$ne": None}.
if country == "":
    country = {"$ne": None}

# Qui gestisco i casi in cui ho uno o entrambi i valori di aqi_min e aqi_max forniti come stringhe vuote.
# A seconda di questi valori, viene costruita una condizione di filtro per il campo "tot_aqi" utilizzando operatori di confronto
# come "$ne" (diverso da null), "$gt" (maggiore di) e "$lt" (minore di) nel contesto delle query al database.
```

"$gt" e "$lt"

```
if aqi_min == "":
    if aqi_max == "":
        tot_aqi = {"$ne": None, "$ne": None}
    else:
        temp2 = float(aqi_max)
        tot_aqi = {"$ne": None, "$lt": temp2}
else:
    temp1 = float(aqi_min)
    if aqi_max == "":
        tot_aqi = {"$gt": temp1, "$ne": None}
    else:
        temp2 = float(aqi_min)
        tot_aqi = {"$gt": temp1, "$lt": temp2}
if co_min == "":
    if co_max == "":
        tot_co = {"$ne": None, "$ne": None}
    else:
        temp2 = float(co_max)
        tot_co = {"$ne": None, "$lt": temp2}
else:
    temp1 = float(co_min)
    if co_max == "":
        tot_co = {"$gt": temp1, "$lt": temp2}
    else:
        temp2 = float(co_max)
        tot_co = {"$ne": None, "$lt": temp2}
```

```python
# Lista delle query
# EightParameter è la query dinamica per la ricerca che mi permette di calcolare i range sulla base dei valori inseriti dall'utente.
1 usage
def eightParameter(country, city, tot_aqi, aqi_category, tot_co, aqi_co_category, lat, lng):
    col = Cp.connection_pool()
    query = col.find({"$and": [
        {"Country": country},
        {"City_name": city},
        {"AQI": tot_aqi},
        {"CO": tot_co},
        {"AQI_category": aqi_category},
        {"AQI_CO_category": aqi_co_category},
        {"Latitude": float(lat)},
        {"Longitude": float(lng)}
    ]
    })
    return query
```

QueryList.py

Cancellazione pages.py

```python
@app.route("/test_query/delete_country", methods=['POST', 'GET'])
def delete_query():
    if request.method == 'POST':
        # Ottieni il nome del paese da eliminare
        country = request.form['demo-country-id']

        if country == "":
            # Se il nome del paese è vuoto, mostra un messaggio di errore
            return render_template("test_query.html", response="Nessun paese rilevato da eliminare",
                                   list_air=getResult())
        else:
            # Cerca il paese nel database
            result = Query.findCountryByName(country)
            value = None
            for i in result:
                value = result
            if value is not None:
                # Se il paese è presente nel database, elimina il documento corrispondente
                Query.deleteCountry(country)
                return render_template("test_query.html", response="Eliminazione effettuata", list_air=fullTable())
                print("Eliminazione effettuata")
            else:
                # Se il paese non è presente nel database, mostra un messaggio di errore
                return render_template("test_query.html", response="Eliminazione non effettuata", list_air=getResult())
                print("Eliminazione non effettuata")
```

```python
# DeleteCountry è la funzione che elimina un paese sulla base del nome
1 usage
def deleteCountry(country):
    col = Cp.connection_pool()
    col.delete_one({"Country": country})
```

QueryList.py

1.eightParameter: Esegue una query dinamica per la ricerca in base a otto parametri di ricerca (country, city, tot_aqi, aqi_category, tot_co, aqi_co_category, lat, lng) all'interno del database. Restituisce i risultati corrispondenti alle condizioni specificate.

2.insertCountry: Inserisce un nuovo paese nel database con tutti i campi associati necessari per la memorizzazione dei dati relativi all'Air Quality Index (AQI) e le relative categorie.

3.updateCountry: Aggiorna i dati di un paese nel database in base al nome della città indicata.

4.deleteCountry: Elimina un paese dal database in base al nome.

5.countryAlphabetica: Trova tutti i paesi nel database e li ordina in ordine alfabetico per nome del paese.

6.findCountryByName: Trova i paesi nel database in base al nome del paese.

7.findCountryByAQICategory: Trova i paesi nel database in base alla categoria dell'Air Quality Index (AQI).

8.findCountryByAQIValue: Trova i paesi nel database con il valore di AQI compreso tra i valori min e max specificati.

9.findCountryByCoordinates: Trova i paesi nel database in base alle coordinate di latitudine e longitudine specificate.

10.findCountryByCOAQIValue: Trova i paesi nel database con il valore di AQI del monossido di carbonio (CO) compreso tra i valori min e max specificati.

11.findCountryByOzoneAQICategory: Trova i paesi nel database in base alla categoria dell'Air Quality Index dell'ozono.

12.findCountryByNO2AQIValue: Trova i paesi nel database con il valore di AQI del biossido di azoto (NO2) compreso tra i valori min e max specificati.

13.findCountryByPM25AQICategory: Trova i paesi nel database in base alla categoria dell'Air Quality Index delle particelle sottili (PM2.5).

14.findCountryByPM25AQIValue: Trova i paesi nel database con il valore di AQI delle particelle sottili (PM2.5) compreso tra i valori min e max specificati.

15.findCountryByLatitudeOrLongitude: Trova i paesi nel database in base alla coordinata di latitudine o longitudine specificata.

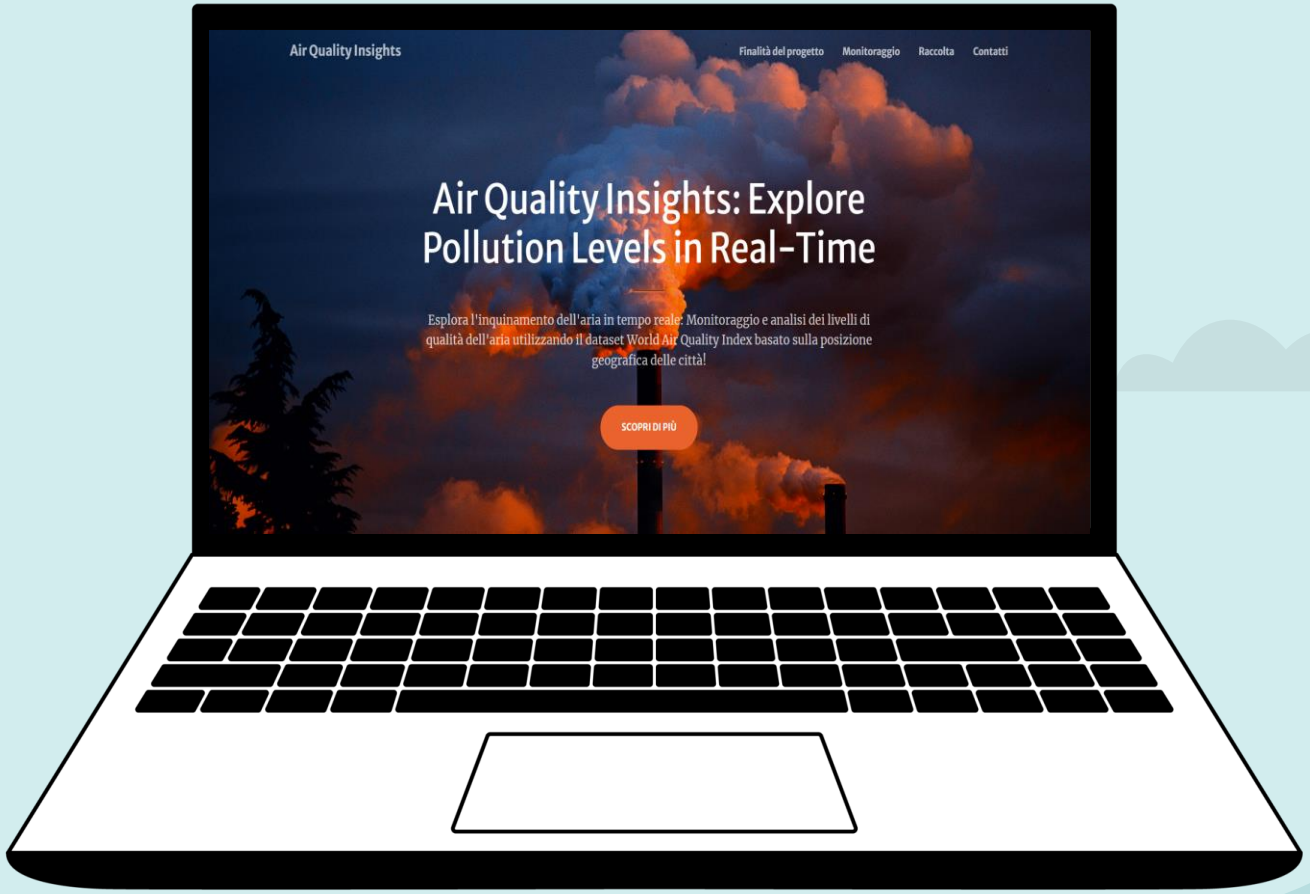
Front-end

La parte grafica del progetto è stata sviluppato utilizzando HTML, CSS e JavaScript.

- HTML ha permesso di strutturare il contenuto delle pagine web,
- CSS è stato utilizzato per definire lo stile e l'aspetto visivo dell'interfaccia.
- JavaScript è stato impiegato per aggiungere interattività e funzionalità dinamiche alle pagine, gestendo eventi, richieste al server e manipolando il contenuto HTML e CSS.

L'interfaccia utente è stata progettata con l'impiego di questi strumenti personalizzarne l'aspetto e definendo colori, layout, sfondi e tipografia. JavaScript ha consentito l'interazione con gli utenti, gestendo la compilazione dei moduli e rispondendo a eventi come clic sui pulsanti o lo scorrimento della pagina.

Il front-end ha comunicato con il back-end attraverso richieste HTTP, inviando richieste al server per il recupero o l'aggiornamento dei dati. JavaScript si è occupato di inviare queste richieste al back-end utilizzando le API fornite e di elaborare le risposte ricevute per aggiornare l'interfaccia utente.



Sviluppi futuri

Aggiunta di strumenti interattivi

Aggiungi **strumenti e grafici interattivi per visualizzare i dati in modo più efficace e comprensibile**. Ad esempio integrando grafici a barre, grafici a torta o grafici a dispersione per evidenziare le tendenze, le relazioni e le metriche chiave all'interno dell'applicazione.

Analizzare diversi tipi d'inquinamento

Ampliare l'applicazione per coprire una varietà di problemi ambientali, consentendo agli utenti di monitorare e comprendere una **vasta gamma di impatti ambientali** per promuovere una maggiore consapevolezza e adozione di comportamenti sostenibili.

Sviluppo mobile

Estendere la web app con una **versione mobile** dedicata per consentire agli utenti di accedere e utilizzare l'applicazione da dispositivi mobili, offrendo un'esperienza user-friendly e accessibile ovunque.

Grazie per l'attenzione

Codice presente su Github al seguente link https://github.com/AngelaDeM/AirQuality/tree/main/basi_dati_II