Università degli Studi di Salerno

Dipartimento di Informatica

Progetto di Basi Dati II

Air Quality Insights:
Explore Pollution Levels in Real-Time

Professore
Genoveffa Tortora

Studenti
De Martino Angela 0522501589

Il mio progetto è intitolato "**AirQuality Insights: Explore Pollution Levels in Real Time**", si basa su quello che è il dataset "**World Air Quality Index by City and Coordinates**"[1] presente sulla piattaforma Kaggle ovvero una piattaforma che ospita competizioni di data science e fornisce dataset pubblici.

Più nello specifico, il dataset analizzato fornisce informazioni sulla qualità dell'aria in diversi paesi.

L'inquinamento atmosferico è una tematica di cui sentiamo parlare quotidianamente ed è un problema di grande rilevanza per la comunità, poiché ha un impatto significativo sulla salute umana, provocando gravi effetti sui polmoni, sul sistema cardiovascolare e sul clima.

Il dataset utilizzato è stato creato per consentire alla comunità scientifica e ai decisori politici di prendere decisioni informate per affrontare questo problema, specialmente nelle aree con elevata contaminazione atmosferica.

Questo dataset è stato ottenuto dalla combinazione di due set di dati: uno riguardante le informazioni sulle città e l'altro riguardante i livelli di inquinamento atmosferico. Unendo questi due, è possibile analizzare e confrontare gli indici di qualità dell'aria in diversi paesi in maniera efficiente.

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

Visualizzazione compatta del dataset

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

Il dataset comprende 17 mila righe di dati distribuite in 14 colonne.

Per comprendere gli attributi del dataset è importante introdurre alcuni termini fortemente legati al dominio di riferimento, ovvero quello dell'inquinamento atmosferico, ciò permetterà una comprensione completa delle implicazioni e delle misurazioni coinvolte.

- **PM2.5**: Questo termine indica particelle di dimensioni circa 2.5 micrometri che contengono metalli pesanti. Queste particelle possono essere dannose quando inalate, poiché possono penetrare profondamente nei polmoni, causando problemi respiratori e danni alla salute umana.
- **Ozono**: L'ozono è un gas che quando risulta essere presente in concentrazioni elevate nell'atmosfera, può provocare infiammazioni delle vie respiratorie e danneggiare sia la salute delle persone che la crescita delle piante.
- **Monossido di carbonio**: Questo gas, incolore ed inodore, è prodotto dalla combustione di combustibili fossili. L'inalazione di monossido di carbonio può portare a problemi legati al trasporto dell'ossigeno nel sangue, causando danni agli organi e mettendo a rischio la salute umana.

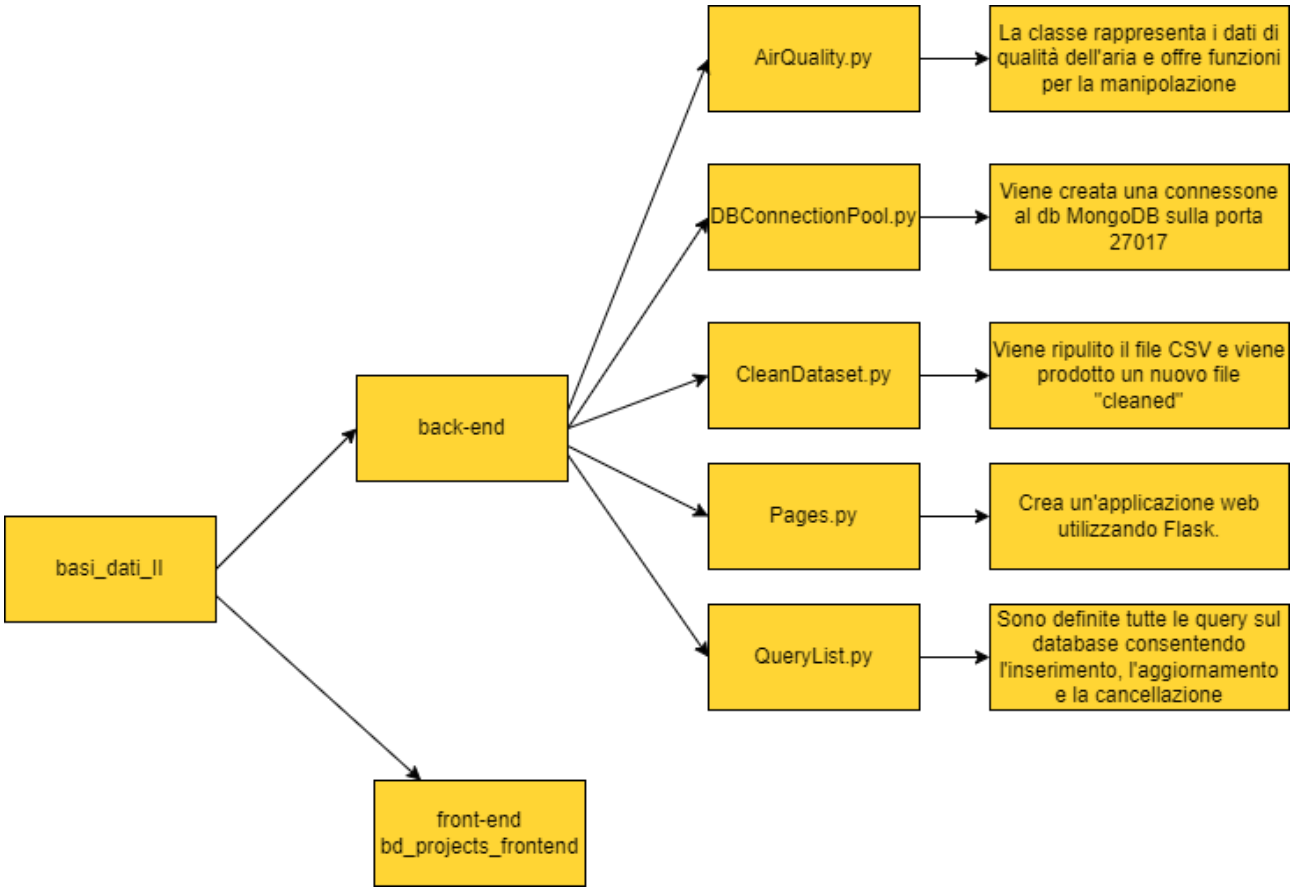
Una volta compresi i termini, passiamo all'analisi delle 14 colonne nella loro interezza.

1. **Country**: Rappresenta il nome del paese.
2. **City**: Rappresenta il nome della città.
3. **AQI Value**: Rappresenta il valore dell'Air Quality Index (AQI) per l'istanza specifica, un indice numerico che rappresenta la qualità dell'aria in una determinata area.
4. **AQI Category**: Rappresenta la categoria dell'Air Quality Index (AQI) per l'istanza specifica, che descrive il livello di inquinamento dell'aria.
5. **CO AQI Value**: Rappresenta il valore dell'Air Quality Index (AQI) specifico per il monossido di carbonio (CO) per l'istanza.
6. **CO AQI Category**: Rappresenta la categoria dell'Air Quality Index (AQI) specifica per il monossido di carbonio (CO) per l'istanza.
7. **Ozone AQI Value**: Rappresenta il valore dell'Air Quality Index (AQI) specifico per l'ozono per l'istanza.
8. **Ozone AQI Category**: Rappresenta la categoria dell'Air Quality Index (AQI) specifica per l'ozono per l'istanza.
9. **NO2 AQI Value**: Rappresenta il valore dell'Air Quality Index (AQI) specifico per l'NO2 per l'istanza.
10. **NO2 AQI Category**: Rappresenta la categoria dell'Air Quality Index (AQI) specifica per l'NO2 per l'istanza.
11. **PM2_5 AQI Value**: Rappresenta il valore dell'Air Quality Index (AQI) specifico per le particelle sottili PM2.5 per l'istanza.
12. **PM2_5 AQI Category**: Rappresenta la categoria dell'Air Quality Index (AQI) specifica per le particelle sottili PM2.5 per l'istanza.
13. **lat**: Rappresenta la latitudine geografica della città.
14. **lng**: Rappresenta la longitudine geografica della città.

Una volta terminata l'analisi del dataset, passiamo alla descrizione della struttura del progetto.

Il progetto è stato realizzato utilizzando l'ambiente di sviluppo IntelliJ siccome risulta essere una scelta ideale grazie alla sua versatilità e potenza nel supportare diversi linguaggi di programmazione.
Con una piattaforma semplice ma potente è stato possibile sviluppare un'applicazione bella e funzionale.

In particolare, è stata adottata una chiara separazione tra il back-end e il front-end dell'applicazione, seguendo una strutturazione ben definita: nella cartella principale del progetto "basi_dati_II" vi sono tutti i file per lo sviluppo back-end e poi, in un'apposita cartella chiamata "bd_projects_frontend" vi è tutto ciò che riguarda il front-end.



Visualizzazione semplificata dell'organizzazione delle cartelle in Intellij

Passiamo ora ad un'analisi a grana fine:

1. **Cartella principale**: Questa cartella contiene i file relativi al back-end del progetto tra cui:
    - AirQuality.py: Questo file definisce la classe AirQuality, che gestisce i dati relativi alla qualità dell'aria.
    - BDConnectionPool.py: Questo file si occupa della gestione della connessione al database, consentendo un accesso efficiente e sicuro ai dati.
    - CleanDataset.py: Questo file gestisce la pulizia dei dati nel dataset, garantendo che siano affidabili e pronti per l'analisi.
    - Pages.py: Il file Pages.py contiene il main del progetto, in cui vengono gestite le pagine dell'applicazione e le richieste dell'utente.
    - QueryList.py: Questo file contiene l'elenco delle query utilizzate per interrogare il database e ottenere i dati necessari.

2. **bd_projects_frontend**: Questa cartella contiene i file relativi al front-end del progetto tra i più importanti ci sono:

    - index.html: Questa è la pagina principale dell'applicazione che illustra il problema affrontato e fornisce una rappresentazione di quelli che sono gli effetti dell'inquinamento dell'aria.
      Da questa pagina iniziale si passa poi, mediante il bottone "procedi con le interrogazioni", ad effettuare le query al dataset.
    - test_query.html: Questa pagina contiene gli elementi necessari che consentono all'utente di effettuare le interrogazioni personalizzate sui dati e visualizzarne i risultati nell'apposita tabella (che è in grado di fornire anche un ordinamento alfabetico istantaneo cliccando sull'intestazione della colonna da ordinare).
    - style.css: Il file style.css contiene le regole di stile per l'applicazione, garantendo una presentazione coerente e piacevole.
    - script.js: Il file script.js contiene le funzionalità interattive dell'applicazione, come la gestione degli input dell'utente e l'aggiornamento dinamico dei dati visualizzati.

Nel contesto più specifico dell'implementazione della mia web app, ho optato per l'utilizzo del linguaggio di programmazione Python insieme a un database NoSQL particolarmente famoso ovvero **MongoDB**.

Questa scelta mi ha consentito di applicare le conoscenze apprese riguardo i database non relazionali ed ottenere una piattaforma che risulta essere efficiente nell'esecuzione di query ma anche piacevole da utilizzare.

MongoDB è risultato particolarmente performante perché fornisce molteplici vantaggi tra cui:

- Scalabilità orizzontale: consente di gestire facilmente grandi quantità di dati.
- Flessibilità dello schema: consente di adattare lo schema in base alle esigenze.
- Alta velocità e prestazioni
- Facilità d'integrazione

Per stabilire la comunicazione con il database, ho utilizzato la libreria **Pymongo**.
Pymongo è un potente strumento in Python che agevola l'interazione con MongoDB. Fornisce un'API intuitiva e ricca di funzionalità che si integra perfettamente con il linguaggio di programmazione Python, consentendo lo sviluppo di applicazioni robuste e scalabili basate su MongoDB.

Dopo aver stabilito la connessione con il database e definito la classe "AirQuality", ho pulito il file CSV originale utilizzando la libreria Pandas, e più nello specifico il metodo dropna, ottenendo così un nuovo file "cleaned" privo di valori mancanti e quindi coerente e affidabile.
Con questa solida base, ho concentrato gli sforzi sul cuore del progetto, concentrandomi sulle query CRUD.

Nel file "QueryList.py" ho definito le query CRUD (insertCountry, updateCountry, eightParameter, deleteCountry) che mi hanno permesso di eseguire operazioni fondamentali modo altamente efficiente: quando vengono eseguite le interrogazioni vengono estratti i dati provenienti dal front-end, vengono validati e successivamente eseguite le query sul database. Gli effetti dell'esecuzione si riflettono direttamente nella tabella visualizzata a schermo ed inoltre le modifiche apportate risultano essere permanenti e coincidono con i risultati attesi.
Particolare attenzione è stata dedicata alla query di ricerca, ovvero una query dinamica, che mi permette di calcolare dei range sulla base dei valori inseriti dall'utente. Nel file pages.py una volta estratti i dati necessari per la ricerca ho sfruttato gli operatori "$gt" (Greater Than) e "$lt" (Less Than) per effettuare ricerche basate su intervalli. Dopo aver eseguito questi calcoli, l'esecuzione procede alla chiamata della funzione "eightParameter" e quest'ultima utilizza i range precedentemente ottenuti per condurre una ricerca mirata nel database, consentendo di ottenere risultati specifici e rilevanti per le necessità dell'applicazione.

Inoltre, ho iniziato a pianificare sviluppi futuri per il progetto gettando le basi anche per interrogazioni più specifiche con query focalizzate su richieste di estrazione più dettagliate.

Una descrizione di tutte le query, CRUD e dettagliate, è il seguente:

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.

In sintesi, per orchestrare questa complessa interazione tra utenti, dati e visualizzazioni, è stato fondamentale l'impiego di Flask. Questo potente framework è stato adottato come robusta infrastruttura per sviluppare il back-end dell'applicazione. Sfruttando Flask, è stato possibile gestire con precisione le richieste HTTP provenienti dal front-end, grazie alla definizione accurata delle route dell'applicazione.

Questo processo ha consentito di stabilire chiaramente quale azione intraprendere in risposta a ciascuna specifica richiesta.

La flessibilità e l'efficienza di Flask hanno contribuito in modo significativo a creare un'esperienza utente fluida e una manipolazione efficace dei dati, offrendo una solida base per il funzionamento dell'intera applicazione.

Tutto questo ha creato le fondamenta per la piattaforma web "Air Quality Insights: Explore Pollution Levels in Real Time", che permette di esplorare i livelli di inquinamento in tempo reale.

In conclusione, l'implementazione di questa piattaforma ha effettivamente raggiunto gli obiettivi prefissati, offrendo un valore tangibile e concreto a tutti coloro che sono interessati ad approfondire l'analisi della qualità dell'aria.