

PollutionStat

Progetto di
Diego Daolio (matricola: 152316)
Mattia Mora (matricola:152354)

1. DESCRIZIONE DEL PROGETTO

Il progetto ha lo scopo di fornire un prototipo di sistema in grado di aiutare a studiare gli effetti collaterali dei vari agenti inquinanti nella vita di tutti i giorni e in varie posizioni del globo.

I'obiettivo è quello di permettere agli utenti delle varie città di poter controllare la presenza di determinati agenti inquinanti, grazie a rilevazioni fatte quotidianamente da sensori, e verificare come esse possano influenzare la salute della popolazione locale attraverso lo studio delle malattie più frequenti nella zona. .

Per ogni città viene creata una pagina di statistiche che comprendono:

1. La generazione di grafici per visionare l'andamento dei determinati agenti inquinanti rilevati dai sensori (importante segnalare che ogni sensore rileva un solo agente inquinante)
2. Le malattie più frequenti nella zona con il conteggio di tutti i casi e dati specifici della patologia come indice di morte o durata media.
Lo scopo è di poter ricondurre la presenza di determinate patologie alla presenza di specifici agenti inquinanti.
3. Visualizzare i piani adottati per diminuire la quantità di uno o più agenti inquinanti presenti sul territorio.

Agli utenti autorizzati viene data la possibilità di controllare come altre città stiano provvedendo a diminuire la presenza eccessiva di determinati inquinanti attraverso l'utilizzo di "Piani"; se questi ultimi dovessero portare riscontri positivi potrebbero essere adottati da altre città per risolvere gli stessi problemi.

È stato deciso di mostrare anche i piani con esito negativo, si ritiene utile mostrare le soluzioni fallimentari in modo che non vengano più adottate da altre città e non si perda ulteriore tempo.

Ogni utente può anche visualizzare sulla mappa i sensori a lui assegnati e le corrispondenti rilevazioni.

Ci sono due tipologie di sensori: acquisiti ed atmosferici, che possono misurare la quantità di diversi agenti inquinanti in base al tipo di inquinamento (nell'aria o nell'acqua).

Per permettere tutti questi servizi in maniera efficiente, è stato creato un sistema così strutturato:

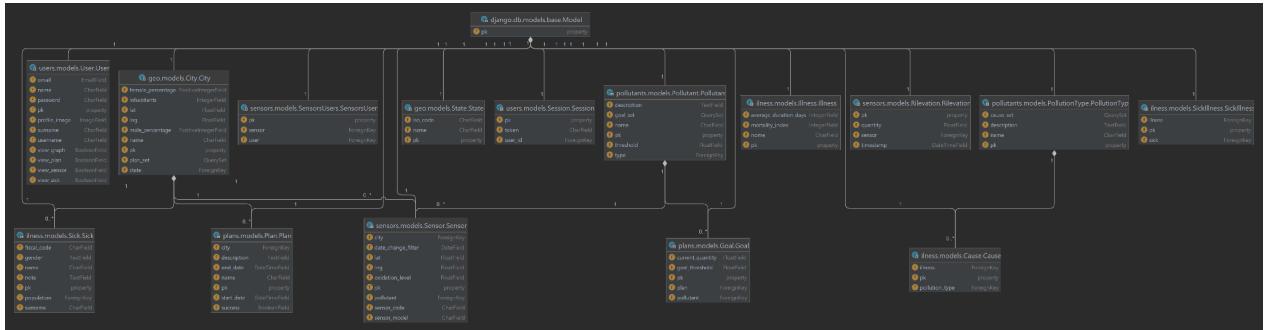


Diagramma consultabile al link:

<https://drive.google.com/file/d/1f174pYjUUChBRcjCoBMKjQAJqG-rXfVg/view?usp=sharing>

2. TECNOLOGIE USATE E MOTIVAZIONE

È stato fatto uso di Django come framework per lo sviluppo del backend.

Utilizzando la sua solida architettura ed ampia gamma di funzionalità, è stato possibile implementare la logica di back-end, creare un database e ottenere un administrator page automaticamente generata per modificare e aggiungere i dati.

Per il frontend, è stato scelto React JS, libreria open-source per costruire interfacce utente interattive, grazie a quest'ultima abbiamo sviluppato un'interfaccia dinamica per visualizzare i dati sull'inquinamento di alcune aree.

I componenti grafici sono delle librerie “MUI” e “Grommet” di React JS.

Per coordinare la comunicazione tra backend e frontend, è stato adottato GraphQL come linguaggio di query, la libreria adottata per la sua implementazione si chiama graphene-django.

GraphQL ha permesso di ottimizzare le prestazioni dell'applicazione e di migliorare l'efficienza nella trasmissione dei dati tra il frontend e il backend, oltre che diminuire i tempi di sviluppo

3. ORGANIZZAZIONE LOGICA DELL'APPLICAZIONE

In Django è stato organizzato il codice utilizzando sette app separate, ognuna con uno specifico scopo.

Ogni app presenta una cartella “models” contenente i modelli django e una cartella “schemas” contenente gli schema per graphQL.

- L'app "Geo" si occupa di gestire le informazioni relative alle città e agli stati. Qui sono presenti modelli per rappresentare le entità come città e stati, e funzioni per interagire con tali dati; ad esempio per ottenere l'elenco delle città.
- L'app "illness" gestisce le malattie legate all'inquinamento. All'interno di questa app sono presenti modelli per rappresentare le diverse malattie. È possibile utilizzare questa app per recuperare informazioni sulle malattie associate all'inquinamento o per eseguire operazioni come l'aggiunta di una nuova malattia al sistema.
- L'app "plans" si occupa di gestire i piani adottati dalle città per affrontare l'inquinamento. Qui sono definiti modelli per rappresentare i diversi piani, come ad esempio piani di riduzione delle emissioni o programmi di monitoraggio dell'aria. Questa app fornisce funzionalità per creare, modificare e visualizzare i piani adottati dalle città.
- L'app "pollutants" gestisce l'elenco degli agenti inquinanti e il tipo di inquinamento, ad esempio l'inquinamento atmosferico o acquatico. In questa app sono inoltre presenti modelli per rappresentare gli agenti inquinanti e le loro proprietà. È possibile utilizzare questa app per ottenere informazioni sugli agenti inquinanti presenti nel sistema o per aggiungere nuovi agenti inquinanti.
- L'app "sensors" si occupa di gestire i sensori. Qui sono presenti modelli per rappresentare i sensori e le relative informazioni. L'app fornisce funzionalità per l'acquisizione, l'archiviazione e l'elaborazione delle rilevazioni oltre che dei sensori.
- L'app "users" gestisce i dati degli utenti dell'applicazione. Questa app contiene modelli per rappresentare gli utenti e le loro informazioni come email e password. Fornisce funzionalità per la gestione degli account utente, inclusa l'autenticazione, la registrazione e il profilo degli utenti.

Con questa organizzazione logica a livello di codice il backend può gestire in modo separato e organizzato i dati relativi alle città, alle malattie, ai piani, agli agenti inquinanti, ai dati dei sensori e agli utenti facilitandone la manutenzione.

4. SCELTE FATTE

Durante lo sviluppo del progetto ci siamo trovati di fronte a diverse decisioni e scelte da fare. Nonostante l'uso di GraphQL, che semplifica la logica dell'API, è stato deciso di adottare un approccio personalizzato, scrivendo query apposite negli schema, per affrontare alcune situazioni specifiche:

- Per ottenere i set di dati necessari per i grafici degli inquinanti, è stata introdotta una logica personalizzata nello schema "Pollutant".

Partendo dai dati delle rilevazioni è stata creata una funzione che restituisse un JSON strutturato in questo modo:

```
{ "NOME_INQUINANTE1": [LISTA_INDAGINI_INQ1],  
  "NOME_INQUINANTE2": [LISTA_INDAGINI_INQ2], ... }.
```

Questo approccio ha permesso di fornire facilmente i dati necessari per generare i grafici degli inquinanti nella pagina di statistiche della web app.

- Per gestire l'autenticazione degli utenti, è stata implementata una logica basata sui token di sessione.

Una volta inseriti nome utente e password, l'utente riceve un token che rappresenta la sua sessione attiva, questo token viene salvato nel browser dell'utente tramite i cookie.

A ogni successivo accesso alle pagine dell'app o tentativo di bypassare il processo di login, si controlla se il token esiste e se corrisponde a una sessione valida; in caso contrario, l'utente viene reindirizzato alla pagina di login.

Per la gestione delle password è stato utilizzato l'algoritmo di hash sha256, la password inserita dall'utente (così come il token di sessione) viene criptata e salvata nel database.

Questa ha permesso di implementare un sistema di autenticazione sicuro e di

garantire l'accesso solo agli utenti autorizzati.

- Implementazione di ruoli: è stato deciso di rendere libera la registrazione ma inserendo alcune limitazioni in visualizzazione.

Un utente si può registrare mediante l'apposita pagina ma avrà solo i permessi per visualizzare le città sulla mappa e i grafici nella pagina delle statistiche.

Mediante l'administrator page è possibile assegnare agli utenti anche altre autorizzazioni, quali la visione dei piani delle città, delle malattie e dei sensori.

- Filtro sensore-utente: Nel progetto è prevista una relazione M-N tra sensori e utenti in modo che si possa decidere se e quali sensori mostrare alle varie utenze.

5. TEST FATTI

È stata sfruttata la funzionalità di Django per scrivere test automatizzati, ma siccome abbiamo utilizzato Graphene-Django li abbiamo scritti mediante "GraphQLTestCase".

Ci siamo concentrati principalmente sulla verifica delle funzionalità offerte nel frontend, assicurandoci che esse non vengano compromesse da sviluppi futuri.

Per ogni app di Django sono stati scritti alcuni test, all'avvio di ogni test vengono create anagrafiche fintizie nel database di testing con lo scopo di verificare il corretto funzionamento di alcune query e/o mutation.

Di seguito i testi più significativi:

- App illness:

Viene testata una query specifica che permette agli utenti di ottenere l'elenco delle malattie registrate in una determinata città.

Questa query consente agli utenti di verificare quali malattie sono state segnalate o registrate nella città di loro interesse.

- App plans:

Viene testata una query che restituisce i piani adottati da una specifica città.

Questa query consente agli utenti di ottenere informazioni sui piani o strategie messi in atto dalla città per affrontare la problematica dell'inquinamento.

- App pollutants:

Viene testata una query specifica che restituisce gli agenti inquinanti presenti in una

determinata città.

Questa query consente agli utenti di ottenere un elenco degli agenti inquinanti che sono presenti nell'ambiente di una specifica città.

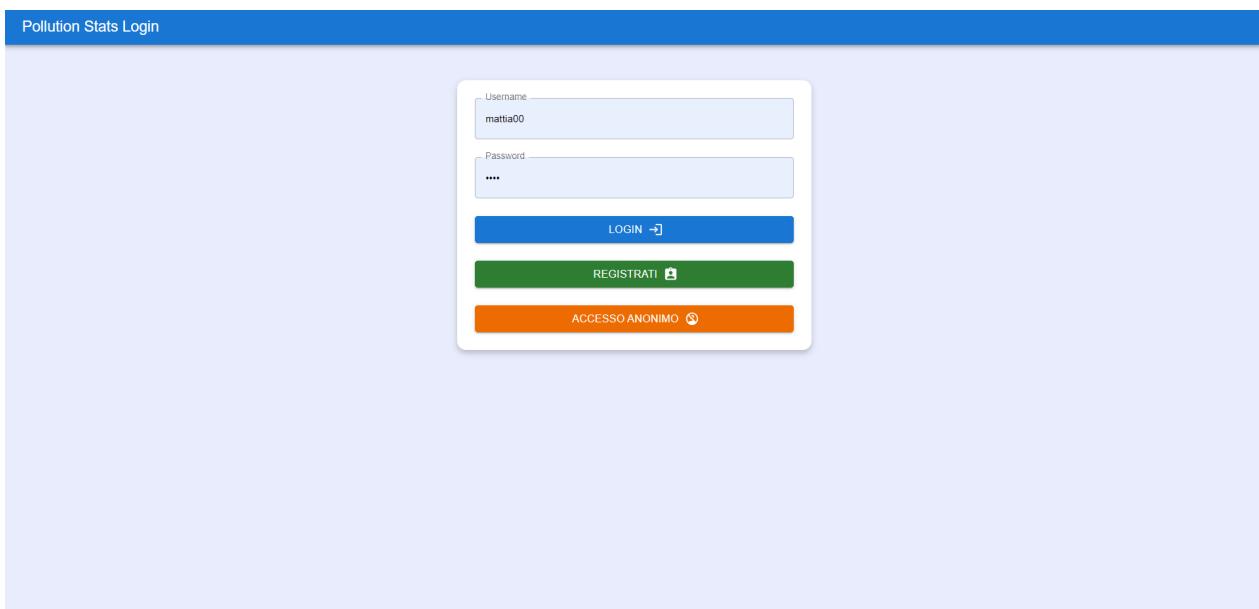
- App sensors:

Viene testata una query specifica che ritorna le rilevazioni effettuate da un sensore specifico e una che restituisce i sensori associati a un utente.

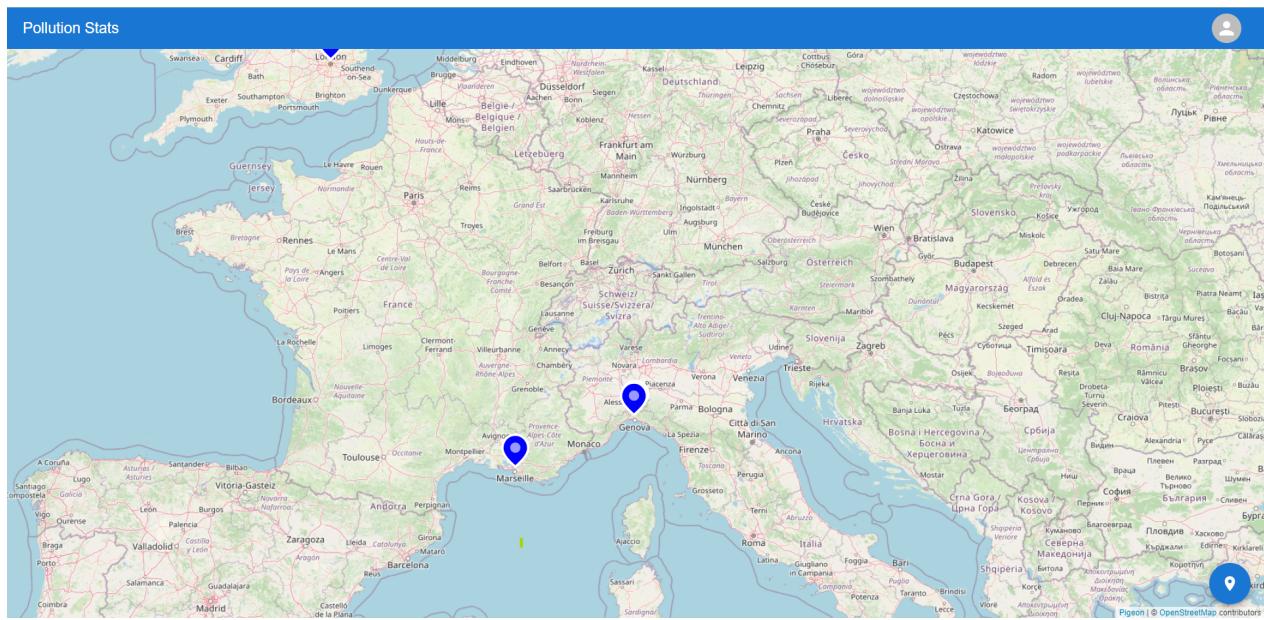
La prima query consente agli utenti di ottenere i dati o le rilevazioni registrate da un determinato sensore, come ad esempio le misurazioni di inquinamento dell'aria o dell'acqua, mentre la seconda query consente agli utenti di ottenere l'elenco dei sensori che sono collegati o assegnati al loro account utente.

6. RISULTATO

All'avvio dell'applicazione, viene visualizzata una schermata di login che offre agli utenti la possibilità di accedere al sistema o di registrarsi come un nuovo utente.



Dopo la login si presenterà una mappa con vari “marker” e un bottone in fondo sulla destra per riportarci automaticamente nel luogo in cui ci troviamo (se si da il consenso a condividere la posizione).

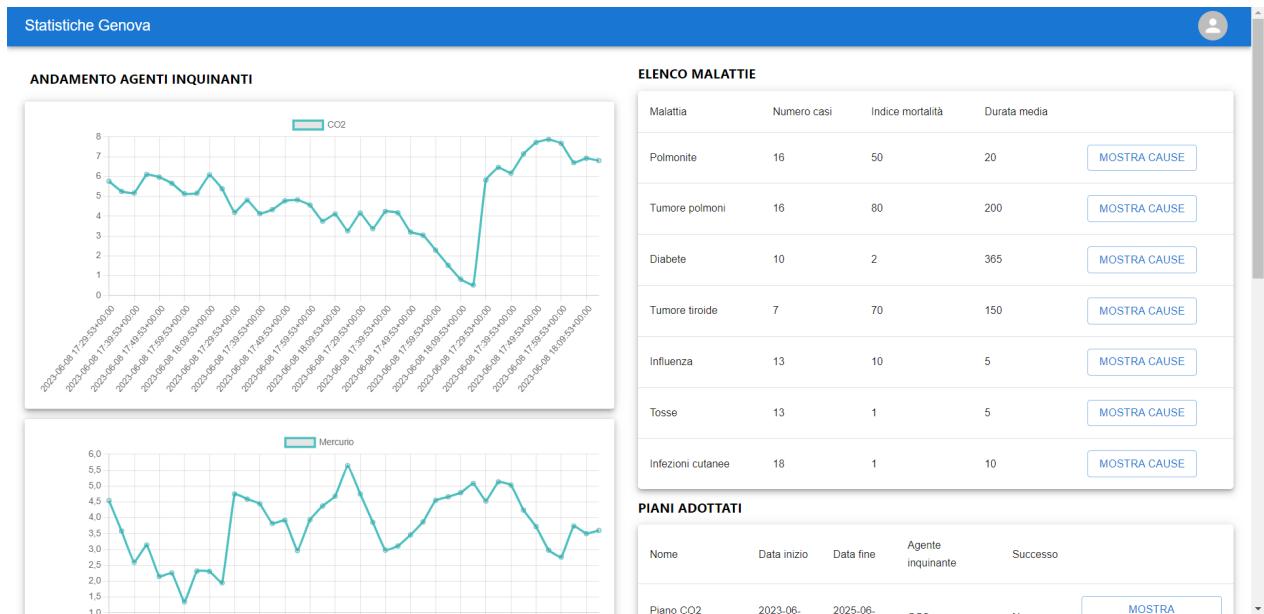


Distinguiamo due tipologie di Marker:

Marker BLU: Segnalano la città e permettono di condurre l'utente direttamente alla pagina delle statistiche.

Marker ROSSI: Segnalano un sensore, se cliccati riportano tutte le rilevazioni di quel sensore. Un sensore è visibile solo agli utenti a cui sono associati.

Pagine delle statistiche



Sulla sinistra verranno mostrati in colonna i grafici dell'andamento di tutti gli agenti inquinanti presenti rilevati nella zona,

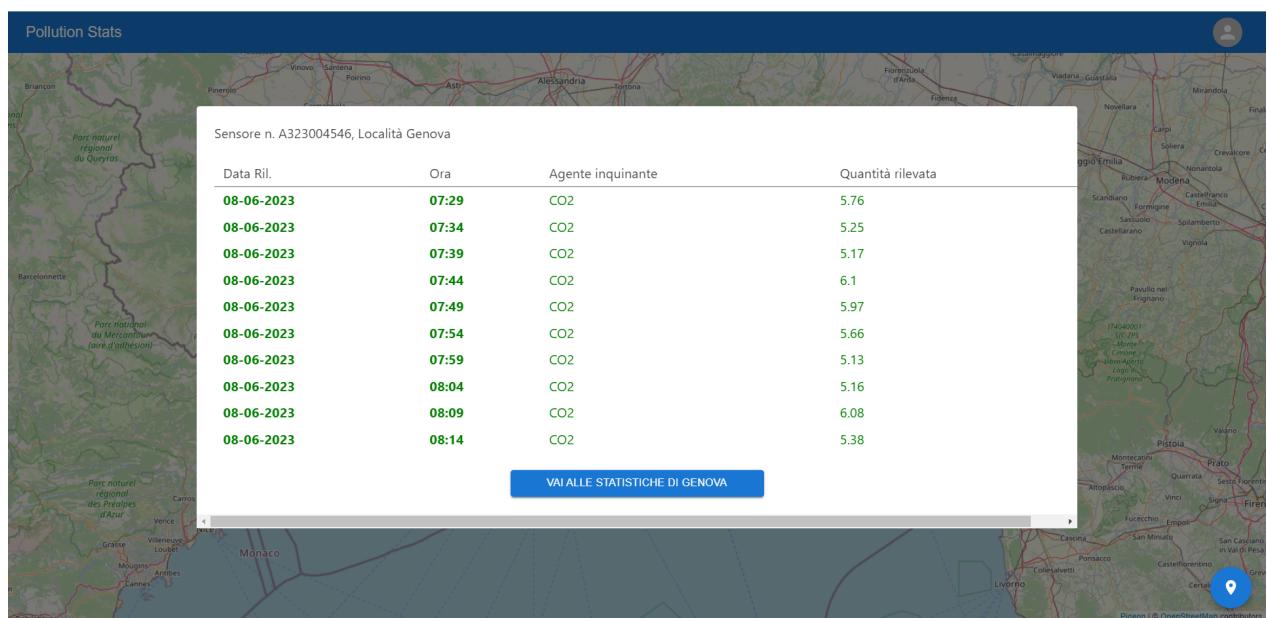
Menù delle rilevazioni per un sensore:

Le rilevazioni possono essere di tre colori:

Verde: La quantità rilevata è minore della soglia massima impostata nel **piano** attualmente utilizzato dalla città per quell'agente inquinante.

Rossa: La quantità rilevata è maggiore della soglia massima impostata nel **piano** attualmente utilizzato dalla città per quell'agente inquinante.

Nera: La città non ha piani attualmente in vigore per quell'agente inquinante



è inoltre presente il bottone “*Vai alle statistiche di <città>*” che riconduce alla pagina di statistiche generali della città in cui è collocato il sensore

In alto a destra è presente un’icona contenente l’immagine profilo dell’utente loggato, cliccandoci sarà possibile accedere alla pagina di personalizzazione del profilo.



mattia

mora

mattia.mora56@gmail.com

Trascina un'immagine o **carica**

AGGIORNA

LOGOUT

Per creare le anagrafiche delle città, dei piani, degli agenti inquinanti ecc... viene sfruttata la pagina di Admin generata automaticamente dal framework django.

Django administration

Site administration

WELCOME ADMIN VIEW SITE / CHANGE PASSWORD / LOG OUT

AUTHENTICATION AND AUTHORIZATION

- Groups
- Users

GEO

- Cities
- States

ILLNESS

- Causes
- Illnesses
- Sick Illnesses
- Sicks

PLANS

- Goals
- Plans

POLLUTANTS

- Pollutants
- Pollution types

Recent actions

My actions

- utenteimaeisistito boniclaudio.mora56@gmail.com User
- mattia00 mattia.mora56@gmail.com User
- mattia00 mattia.mora56@gmail.com User
- + A323004546 mattia00 Sensors users
- matteo67 mattia.mora56@gmail.com User
- + A323004517 mattia00 Sensors users
- + A323004516 mattia67 Sensors users
- ddfs City

7. PROBLEMI

I principali problemi incontrati sono stati i seguenti:

1. **Gestione delle autorizzazioni:** L'applicazione richiede un sistema di gestione delle autorizzazioni per garantire che solo gli utenti autorizzati possano accedere a determinate funzionalità o dati sensibili. Abbiamo affrontato il problema implementando un sistema di autenticazione e gestendo accuratamente i permessi degli utenti, consentendo loro di accedere solo alle risorse appropriate.
2. **Gestione degli errori:** Durante lo sviluppo, abbiamo affrontato diversi scenari di errore, come errori di rete, errori di validazione dei dati e errori del server. Abbiamo dedicato tempo all'implementazione, per ogni funzione coinvolta in questi errori, di funzionalità per intercettare gli errori e restituire alert adeguati.
3. **Progettazione del frontend:** è stato necessario creare un'interfaccia user-friendly facilmente utilizzabile dagli utenti, inoltre, il frontend, deve adattarsi a dispositivi diversi, garantendo una visualizzazione ottimale su schermi di varie dimensioni e risoluzioni.

Per eventuali sviluppi futuri invece dovremmo risolvere i seguenti problemi:

1. **Integrazione dei dati:** Leggere i dati dei sensori e renderli accurati può essere difficile a causa della precisione limitata dei sensori stessi, delle condizioni ambientali che possono influire sui risultati, del rumore presente nei dati e dei tempi di risposta dei sensori. Sarà importante effettuare la calibrazione e la manutenzione regolare dei sensori per migliorare la precisione.
2. **Ottimizzazione delle query:** Con un numero crescente di dati e richieste da gestire, subentrerà il problema dell'ottimizzazione delle query per garantire prestazioni efficienti. Sarà necessario impostare degli indici corretti e gestire, con librerie come "Timescale" di postgres, le serie storiche delle rilevazioni dei sensori.