

# Progetto Programmazione Avanzata

Castellani Filippo

A.A. 23/24

# Indice

<b>1</b>	<b>Introduzione</b>	<b>3</b>
1.1	Idea di base . . . . .	3
1.2	Inizializzazione del database . . . . .	3
<b>2</b>	<b>Applicazione</b>	<b>4</b>
2.1	Schermate dell'applicazione . . . . .	4
2.1.1	login . . . . .	4
2.1.2	Meteo . . . . .	5
2.1.3	Altri . . . . .	6
2.1.4	Bookmark . . . . .	7
2.1.5	Aggiungi Bookmark . . . . .	8
2.2	Interazione con il server . . . . .	9
2.3	Altre scelte Progettuali . . . . .	9
2.4	Unit Test . . . . .	9
<b>3</b>	<b>Server</b>	<b>10</b>
3.1	API . . . . .	10
3.2	Sessione . . . . .	11
3.3	Spring Data JPA . . . . .	11
3.4	Altre scelte Progettuali . . . . .	11

# 1 Introduzione

## 1.1 Idea di base

L'idea era quella di realizzare un servizio client-server che mostrasse all'utente le previsioni meteo di una determinata città o di una determinata coppia di coordinate. Inoltre si vuole dare la possibilità all'utente di usare dei bookmark con i quali salvare delle coordinate associandoli ad un nome. Il salvataggio di tali dati è gestito tramite un server con l'uso di un account così da poter usare i bookmark su più dispositivi.

Per tale progetto sono stati usati due servizi web:

- [Open-Meteo](#) per ricevere le condizioni meteo
- [Nominatim](#) per la geocodifica delle città

Il server ha un database che tiene traccia dei dati richiesti dagli utenti in passato così da usare i servizi sopra citati solo quando servono dei nuovi dati.

## 1.2 Inizializzazione del database

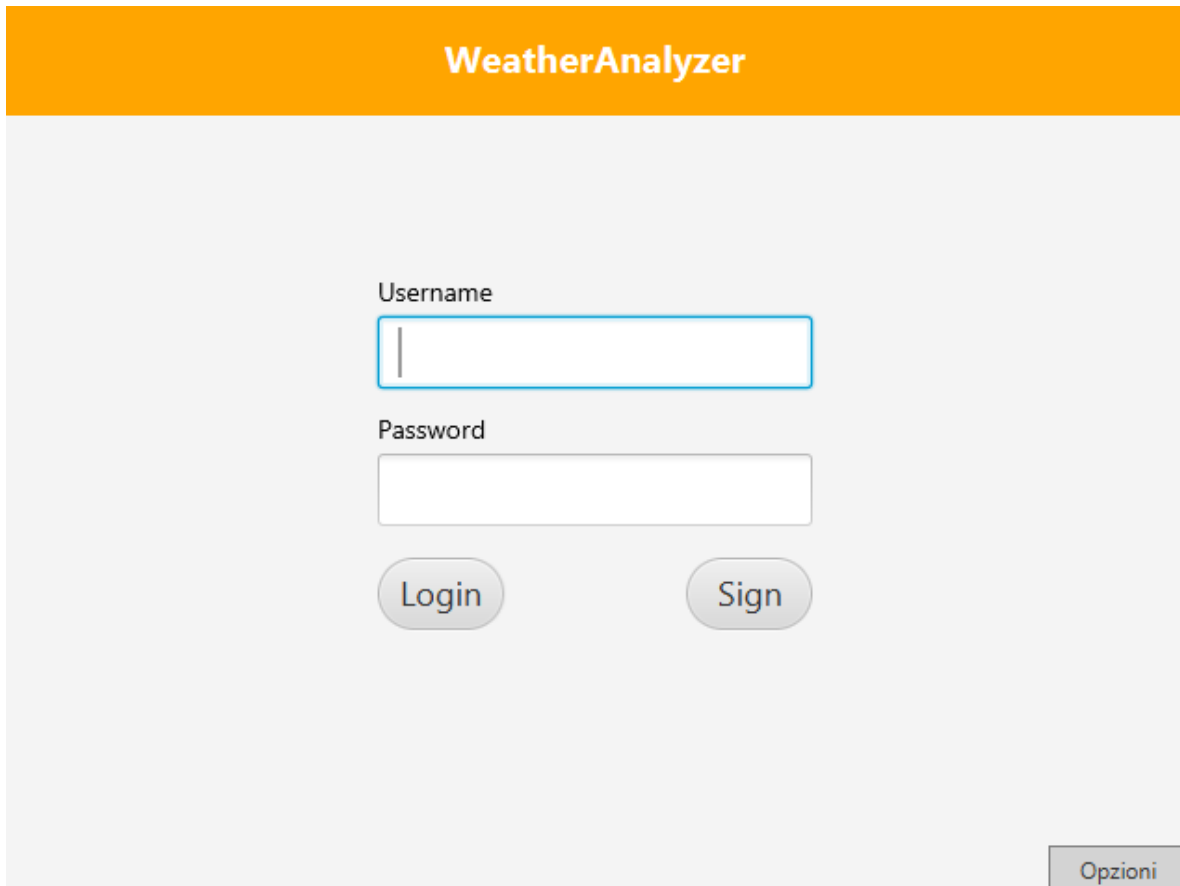
All'avvio il server prova a connettersi al database e cerca lo schema del servizio. Se lo schema non è presente esegue uno script sql (`init.sql`) per creare lo schema e le tabelle, poi le riempie con dei dati. Il server si connette al database con i seguenti parametri:

Indirizzo	localhost
Porta	3306
Username:	root
Password:	root

## 2 Applicazione

### 2.1 Schermate dell'applicazione

#### 2.1.1 login



The screenshot shows the login interface of the WeatherAnalyzer application. At the top, there is an orange header bar with the text "WeatherAnalyzer" in white. Below this, the main area has a light gray background. It contains two input fields: "Username" with a blue border and a cursor, and "Password" with a standard gray border. Below the password field are two rounded buttons labeled "Login" and "Sign". In the bottom right corner, there is a small button labeled "Opzioni".

All'apertura dell'applicazione viene richiesto il login con le credenziali dell'utente. Se non si è registrati lo si può fare con l'apposito pulsante. L'username deve essere alfanumerico e di massimo 10 caratteri, invece la password non deve contenere spazi e deve essere di al massimo 72 caratteri. Sia l'app che il server rifiutano richieste che non rispettano tali restrizioni. In basso a destra c'è un menù **Opzioni** che permette di cambiare lingua. Per poter procedere è necessario che il server sia in funzione. Il database contiene già un utente con dei dati di test le credenziali sono:

Username: default  
Password: default

### 2.1.2 Meteo

The screenshot shows the WeatherAnalyzer web application. It features a dark grey sidebar on the left with two buttons: 'Altri' and 'Bookmarks'. The main content area has a light grey background. At the top, there's an orange header with the text 'WeatherAnalyzer'. Below the header, there are input fields for 'Città' (containing 'Pisa'), 'Lat', and 'Lon'. To the right of these fields are two buttons: 'Richiedi' and 'Aggiorna'. Below the input fields, there are four data boxes: 'Temperatura' (14.6), 'Precipitazioni' (0.0), 'Umidità' (72), and 'Data' (19/01/2024 03:47). At the bottom right, there is a button labeled 'Opzioni'.

Città	Lat	Lon
Pisa		

Temperatura	Precipitazioni
14.6	0.0

Umidità	Data
72	19/01/2024 03:47

Buttons: Richiedi, Aggiorna, Opzioni

la schermata ci da la possibilità di richiedere l'ultimo record meteo disponibile sul database con **Richiedi** oppure di aggiornare il database con il meteo corrente usando il pulsante **Aggiorna**. Le richieste possono essere effettuate o per città o per coordinate in base a quale dei campi si riempie.

Il pulsante **Opzioni** ora permette di tornare al login con il tasto **Logout**.

A sinistra ci sono dei pulsanti con i quali si può navigare verso le altre schermate.

Nel Database ci sono dati per alcune delle principali città (Pisa, Roma, Milano...) e alcune coordinate (12,12 e 90,90)

### 2.1.3 Altri

# WeatherAnalyzer

Meteo

Bookmarks

Città

Lat

Lon

Pisa

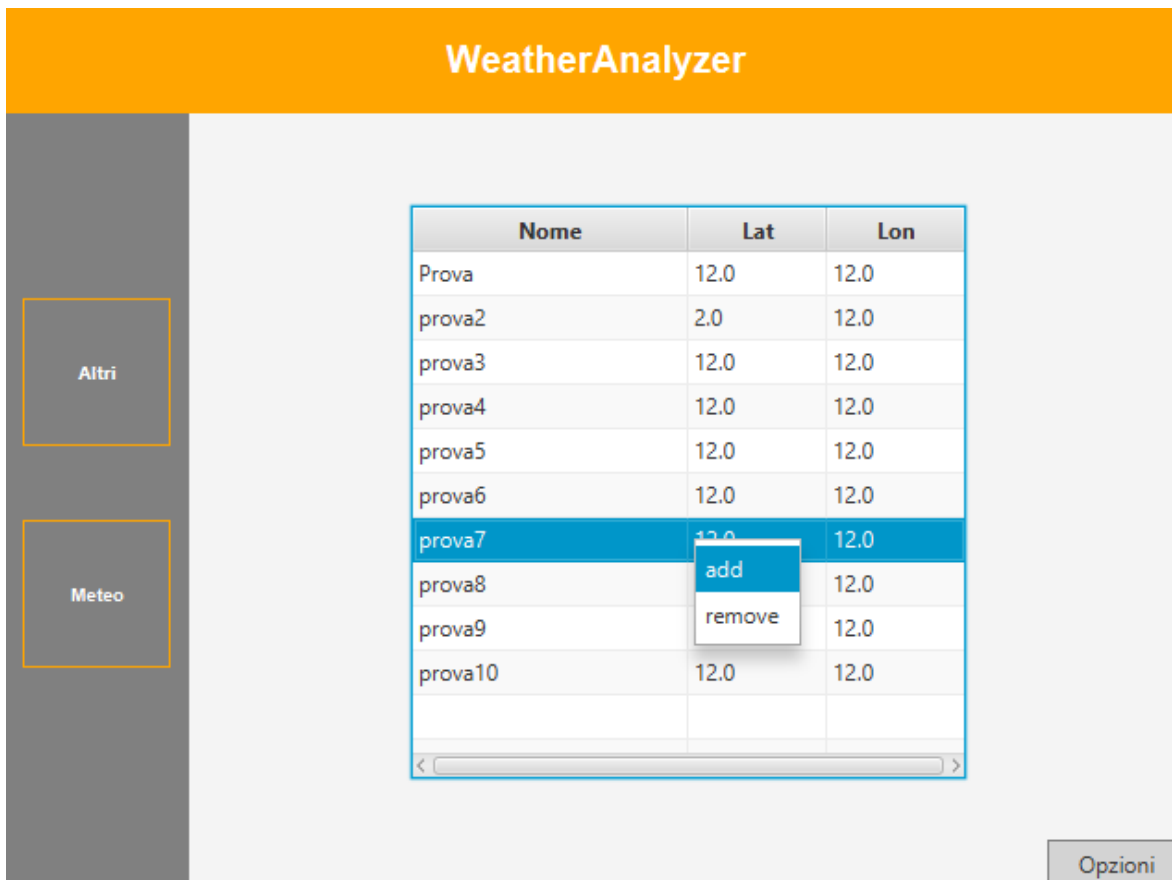
Temperatura	Precipit...	Umidità	Data
14.6	0.0	72	19/01/2024.
14.6	0.0	72	19/01/2024.
14.4	0.0	74	19/01/2024.
14.5	0.0	74	19/01/2024.
14.6	0.0	72	19/01/2024.
14.6	0.0	72	19/01/2024.
14.6	0.0	72	18/01/2024.
14.8	0.0	73	18/01/2024.

Richiedi

Opzioni

La schermata ci da la possibilità di richiedere (con il pulsante **Richiedi**) gli ultimi 10 record disponibili per una città o per una coppia di coordinate, la scelta del metodo funziona come nella precedente schermata. I pulsanti per cambiare schermata e quello per le **Opzioni** hanno le stesse funzionalità della precedente scena.

### 2.1.4 Bookmark



La schermata ci permette di visualizzare i nostri bookmark, i quali sono personali e salvati sul server così da poterli avere a disposizione su vari dispositivi. Al caricamento della pagina i bookmark verranno scaricati in automatico. Cliccando con il tasto destro sulla tabella si potranno o rimuovere o aggiungerne degli altri tramite un'apposita schermata. (Una volta eliminato il record dal server l'applicazione non aggiorna la pagina locale con una nuova richiesta, ma semplicemente rimuove il record anche localmente). I pulsanti laterali e quello delle **Opzioni** funzionano come nelle precedenti scene.

### 2.1.5 Aggiungi Bookmark

The screenshot shows the 'WeatherAnalyzer' application interface. The title bar is orange and contains the text 'WeatherAnalyzer'. A dark grey sidebar on the left contains two buttons: 'Altri' and 'Meteo'. The main content area is light grey and features three input fields: 'Nome' (containing 'Prova11'), 'Lat' (containing '12'), and 'Lon' (containing '12'). To the right of the 'Nome' field is a rounded button labeled 'Aggiungi'. To the right of the 'Lon' field is a rounded button labeled 'Indietro'. Below the input fields, the text 'Troppi bookmark' is displayed in red. In the bottom right corner, there is a small grey button labeled 'Opzioni'.

La schermata si apre dopo aver premuto **add** nella precedente scena. Qui possiamo inserire un nome e una coppia di coordinate per inviarle al server tramite la pressione del pulsante **Aggiungi**. Per annullare e tornare ai bookmark basta premere **Indietro**. Il server permette l'inserimento di un massimo di 10 bookmark ad utente, se si tenta di aggiungerne oltre la richiesta viene rifiutata e l'applicazione ce lo comunica. I pulsanti laterali e di **Opzioni** funzionano come nelle precedenti scene.



## 2.2 Interazione con il server

L'applicazione richiede ed invia dati al server tramite richieste Http di tipo `GET` e `POST`, queste ultime usate per l'invio di dati personali dell'utente. Le risposte e le richieste `POST` trasportano dati in formato `json` i quali vengono codificati e decodificati con la libreria `Gson`. Le risposte alle richieste `POST` seguono il formato della classe `Response` nel quale c'è un elemento di tipo stringa (potrebbero essere dei dati codificati in `json`) e un codice il quale ci dice se la richiesta è stata soddisfatta o meno. Il comportamento del client all'arrivo della risposta dipenderà da tale codice.

## 2.3 Altre scelte Progettuali

Il codice è diviso in controller (uno per schermata e uno per le richieste http) e in delle classi di utility, le quali sono le classi usate per contenere, inviare e ricevere dati.

Al login si crea una nuova sessione e il server ci invierà un token, il quale viene poi salvato localmente nel file `token.json` così da poterlo riutilizzare in seguito.

L'applicazione permette di cambiare lingua in ogni schermata, per farlo sono stati realizzati due file (`lingua-en.json` e `lingua-it.json`) in formato `json`.

Per le richieste http di tipo `GET` si è optato per l'utilizzo della libreria `HttpURLConnection`, invece per le richieste di tipo `POST` si è usata `HttpClient`.

L'applicazione tiene traccia di eventuali errori tramite l'utilizzo della libreria `Log4J` salvando i dati nella cartella `log`.

## 2.4 Unit Test

In fase di build si testa il metodo `changeLan` della classe `LoginController` per vedere se il file `json` della lingua viene caricato correttamente. Il test viene eseguito anche in fase di push su GitLab.

[Link](#) al GitLab dell'applicazione

## 3 Server

### 3.1 API

Il server è raggiungibile tramite le seguenti richieste:

POST /sign-in

per registrarsi

POST /login

per effettuare il login

GET /weather/city/{name}

per richiedere l'ultimo record meteo della città

GET /weather/city10/{name}

per richiedere gli ultimi 10 record meteo della città

GET /weather/coord={lat}&{lon}

per richiedere l'ultimo record meteo delle coordinate

GET /weather/coord10={lat}&{lon}

per richiedere gli ultimi 10 record meteo delle coordinate

GET /weather/update/city/{name}

per aggiornare l'ultimo record meteo della città

GET /weather/update/coord={lat}&{lon}

per aggiornare l'ultimo record meteo delle coordinate

POST /bookmark/add

per aggiungere un nuovo bookmark

POST /bookmark/remove

per rimuovere un determinato bookmark

POST /bookmark/get

per ricevere i propri bookmark

## 3.2 Sessione

Quando un utente effettua il login viene generata una sessione della durata di un'ora e all'utente viene assegnato un token che userà per effettuare le richieste. la sessione è salvata nel database del server. Se l'utente si connette ed ha già una sessione attiva allora viene usata quest'ultima, altrimenti se è scaduta se ne genera una nuova. Se l'utente dovesse eseguire richieste con un token scaduto sarà costretto a rieffettuare il login.

## 3.3 Spring Data JPA

Il server utilizza le librerie di **Spring**, in particolare per l'interazione con il database viene usata la libreria **JPA**. Per fare ciò si è realizzata per ogni tabella del database una classe nella cartella **table** ed il relativo **CrudRepository** nella cartella **repository**. Ogni interazione con il database viene effettuata utilizzando i metodi auto implementati del **repository** (ad eccezione della verifica dell'esistenza dello schema ed eventuale realizzazione all'avvio del server, il quale usa le librerie **SQL**).

## 3.4 Altre scelte Progettuali

Il codice è diviso in main, in controller (uno per il meteo, uno per i bookmark ed uno per gli utenti), in delle classi di utility (classi usate per contenere, inviare e ricevere dati), in table (per contenere ed inviare i dati dal database) e repository.

Le password sono codificate con l'utilizzo della libreria **BCrypt**.

Per le risposte http e le richieste di tipo **POST** si usa il formato **json**.

Il server stampa in console eventuali errori tramite l'utilizzo della libreria **Log4J**.

Le richieste ai servizi web utilizzati sono effettuate con la libreria **HttpURLConnection**.