Università degli studi di messina
DIPARTIMENTO DI SCIENZE MATEMATICHE E INFORMATICHE, SCIENZE FISICHE E SCIENZE DELLA TERRA
Corso di Laurea Triennale in Informatica

Progetto di programmazione a oggetti

Davide Mento
MAT. 527917

Anno accademico 2022/2023

Introduzione

Il progetto presenta lo sviluppo di un bot Telegram in linguaggio Java detto meteorologoBot. Il bot si avvale di una API pubblica in grado raccogliere dati da stazioni metereologiche e di fornire informazioni riguardanti le condizioni meteo di tutto il mondo. Sono disponibili vari comandi che permettono all'utente di ottenere le previsioni meteo della città inserita, inoltre, il bot presenta una funzionalità che avverte l'utente nel caso in cui ci dovessero essere condizioni favorevoli, o al contrario, allerte meteo. Questa funzionalità viene eseguita ogni ora per le allerte meteo, e ogni mezz'ora per il meteo favorevole. Il bot è inoltre in grado di memorizzare, attraverso un database, le città scelte dai singoli utenti per le notifiche.

Java

Per lo sviluppo in programmazione a oggetti è stato utilizzato Java, un linguaggio ricco di documentazione, API, scalabile, che permette lo sviluppo di applicazioni modulari. Inoltre, il vantaggio di Java è la sua capacità di funzionare su più piattaforme, grazie all'utilizzo della Java Virtual Machine (JVM), vantaggio da prendere in considerazione se si vuole integrare il bot all'interno di un server.

Telegram

Il progetto è sviluppato con l'utilizzo delle API offerte da Telegram. L'aspetto Open Source di Telegram comporta vari vantaggi riguardo la creazione di progetti, inoltre, le API di Telegram sono ricche di documentazione, rendendo lineare l'utilizzo di queste ultime.

MongoDB

Come database è stato utilizzato MongoDB, un database Open Source non relazionale di facile implementazione. Per rendere il progetto scalabile è stato utilizzato un database come MongoDB in quanto possiede una grande flessibilità data dalla capacità di gestire un grande volume di dati.

Visual Crossing

Visual Crossing è l'API utilizzata per la ricezione di dati metereologici. Questa REST API è Open Source, fornendo così un potente strumento anche per l'utilizzo in progetti personali. Questo servizio sfrutta stazioni metereologiche di tutto il mondo, organizza i dati e li spedisce al client attraverso il formato JSON, che viene poi letto e organizzato dal bot.

Maven

Maven è uno strumento di gestione dei progetti che sfrutta le cosiddette dipendenze. Si preferisce utilizzare un gestore di progetti in quanto un progetto Java richiede un numero elevato di librerie esterne, che, nei progetti di larga scala sono difficilmente gestiti localmente. Maven sfrutta le dipendenze, ovvero delle librerie che non vengono salvate localmente, bensì in un repository remoto, andando così a semplificare la gestione delle librerie.
La gestione delle dipendenze viene eseguita attraverso un file di configurazione xml, detto pom.xml. Durante la fase di compile vengono scaricate le dipendenze dal repository remoto.

Requisiti

Per lo sviluppo sono stati rispettati alcuni requisiti:

- Programmazione a oggetti
- Overloading e Overriding
- Incapsulamento
- Multi-Thread e gestione dei thread
- Gestione delle eccezioni
- Connessione ad un database
- Gestione di un project manager (Maven)
- Implementazione di una REST API (VisualCrossing)
- Ereditarietà

Comandi

Sono disponibili 5 comandi per l'utente:

- /start: avvia il bot e presenta il resto dei comandi all'utente.
- /forecast: presenta le previsioni meteo per il giorno corrente.
- /nextdayforecast: presenta le previsioni meteo per il giorno dopo a quello corrente.
- /setpreferredcity: questo comando assegna all'utente la città inserita, permettendo al bot di mandare messaggi nel caso di bel tempo o di allerte meteo.
- /setalert: permette all'utente di scegliere un'altra città diversa da quella selezionata in "/setpreferredcity" dalla quale ricevere informazioni per allerte meteo.

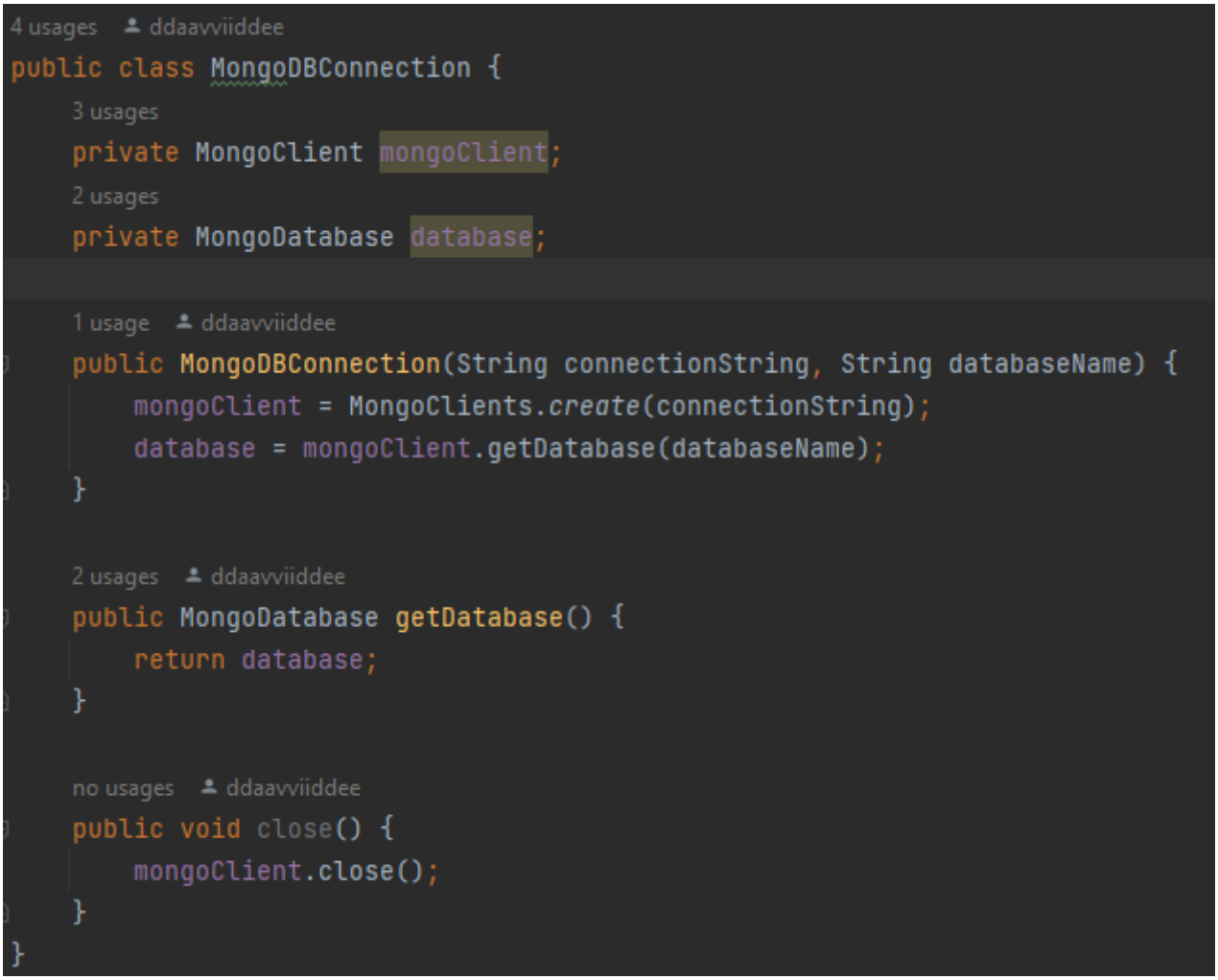
File utilizzati

Per il progetto sono stati utilizzati vari file:

Pom.xml: Il file pom.xml è un file di configurazione che rappresenta le dipendenze all'interno del project manager Maven. Maven è uno strumento che aiuta lo sviluppatore nella gestione dei progetti, andando a definire le dipendenze necessarie per il bot, per poi scaricarle automaticamente.

Main.java: il file main viene utilizzato nel momento in cui viene effettuata la build del progetto, andando ad avviare il bot utilizzando le API di Telegram.

MongoDBConnection.java: questo file si occupa della gestione dei metodi utili alla connessione al database, in questo il DB utilizzato è MongoDB.



```
4 usages  ddaavviiddee
public class MongoDBConnection {
    3 usages
    private MongoClient mongoClient;
    2 usages
    private MongoDatabase database;

    1 usage  ddaavviiddee
    public MongoDBConnection(String connectionString, String databaseName) {
        mongoClient = MongoClients.create(connectionString);
        database = mongoClient.getDatabase(databaseName);
    }

    2 usages  ddaavviiddee
    public MongoDatabase getDatabase() {
        return database;
    }

    no usages  ddaavviiddee
    public void close() {
        mongoClient.close();
    }
}
```

WeatherBot.java: questo file presenta tutti i metodi utili al funzionamento del bot, tra cui: gestione dei comandi, gestione dei thread, gestione delle chiamate all'API e gestione del database, e delle informazioni ricevute. Per l'invio dei messaggi è stata scelta la lingua inglese, dal momento che il bot è pensato per funzionare in tutto il mondo. La classe WeatherBot presenta ereditarietà in quanto estende la classe TelegramLongPollingBot, permettendo così l'utilizzo delle API fornite da Telegram.

WeatherAPI.java: questo file è dedicato alla costruzione della richiesta alla REST API. In questo caso, è stata utilizzata l'API fornita da Visual Crossing, in quanto Open Source con un limite di 1000 richieste giornaliere. La richiesta viene mandata attraverso il protocollo HTTP, mandando al server un URL contenente le informazioni inserite dall'utente. Nel momento in cui la richiesta viene mandata, questa viene eseguita, mandando così al bot un file JSON contenente tutte le informazioni riguardanti il meteo della città inserita.

```java
public class WeatherAPI {

    1 usage
    private final String apiKey;

    7 usages  ddaavviiddee
    public WeatherAPI(String apiKey) {
        this.apiKey = apiKey;
    }

    1 usage
    private static final String API_BASE_URL = "https://weather.visualcrossing.com/VisualCrossingWebServices/rest/services/timeline/";

    7 usages  ddaavviiddee *
    public String getWeatherInfo(String cityName, String apiKey) { // This method build the URL for the API call
        try {
            String apiUrl = API_BASE_URL + cityName + "?unitGroup=metric&key=" + apiKey;
            URL url = new URL(apiUrl);
            HttpURLConnection connection = (HttpURLConnection) url.openConnection();
            connection.setRequestMethod("GET");
            int responseCode = connection.getResponseCode();

            if (responseCode == HttpURLConnection.HTTP_OK) {
                BufferedReader reader = new BufferedReader(new InputStreamReader(connection.getInputStream()));
                String line;
                StringBuilder response = new StringBuilder();
                while ((line = reader.readLine()) != null) {
                    response.append(line);
                }
                reader.close();
                return response.toString();
            } else {
                System.out.println("Request error " + cityName + ". Response error: " + responseCode);
            }
        } catch (IOException e) {
            e.printStackTrace();
            System.out.println("Request error " + cityName + ". Exception: " + e.getMessage());
        }
        return null;
    }
}
```

Nel momento in cui una chiamata non dovesse funzionare, allora verrà mandato al gestore del bot un messaggio contente l'errore di risposta HTTP.

Conclusioni

Il progetto ha permesso un approccio pratico con la programmazione a oggetti, utilizzando servizi contemporanei come le API di Telegram e di Visual Crossing. Inoltre, si è studiato l'utilizzo dei databse non relazionali e la gestione di applicazioni multithreading. In aggiunta, l'approccio client-server con l'API Visual Crossing ha approfondito l'utilizzo del protocollo HTTP.