MANUALE TECNICO

Tirocino Sync Lab

Autori:
Marangon Zaccaria

Azienda:
Sync Lab S.r.l.

December 11, 2024

Indice

1 Introduzione

1.1 Scopo del documento

Il seguente documento ha la scopo di fornire una panoramica generale del progetto, descrivendo le scelte architetturali e applicative effettuate durante lo sviluppo del prodotto richiesto durante il tirocinio curriculare, presso Sync Lab.

1.2 Informazioni sul progetto

Il progetto consiste nello sviluppo del backend e del frontend per un applicazione web che consenta di interagire con un motore LLM (Large Language Model) su cloud una volta effettuato l'accesso o l'eventuale registratione. Il progetto si compone di due parti principali, un frontend che generi un interfaccia utente per l'interazione con l'utente e un backend che si occupi di gestire la comunicazione con il motore LLM e la persistenza dei dati degli utenti.

1.3 Funzionalità del progetto

Il backend realizzato consente di eseguire operazioni relative alla gestione degli account e dei messagi inviati dagli utenti. Infatti è possibile eseguire tutte le opeazioni CRUD sugli account per i messaggi invece è possibile aggiungerli nel database relativo e visualizzare tutti inviati da uno specifico utente. Per quanto riguarda l'interazione con il motore LLM, il backend consente di comunicarci e di memorizzare le risposte generate dal motore stesso. Come ulteriore implementazione tutte le richieste utilizzano un sistema di autenticazione basato su JWT.
Il frontend, che comunica con il backend, implmenta le funzionalità relative alla registrazione e all'accesso, alla visualizzazione dei messaggi inviati e alla comunicazione con il motore LLM, ovviamente gestendo il token JWT ove presente.
Infine il database realizzato in PostgreSQL consente di memorizzare i dati relaivi agli utenti come email, password, username; Per quanto riguarda i messaggi inviati e ricevuti viene salvato il testo del messaggio, il mittente, il timestamp e l'account da cui il messaggio è stato inviato o ricevuto.

2 Scelte architetturali

2.1 Struttura del progetto

Il codice del progetto è stato organizzato in due separate per avere una netta separazione dei file tra il backend e il frontend. Questo scelta ha cosentito una maggiore chiarezza, modularità e ottimisticamente una facilitazione nella manutenzione del codice. Sia il backend che il frontend sono stati sviluppati con metodi analoghi quindi verrano spiegati di consenguenza prestanzo attenzione ove ci siano differenze. Si è deciso di adottare una archietettura a microservizi in quanto il progetto essendo di dimensioni relativamente ridotte avrebbe avuto più senso utilizzare una struttura monolitica per facilitarne lo sviluppo ma essendo l'architteura a microservizi adottata più di frequente si è deciso di utilizzare quest'ultima per avere una migliore resa dal punto di vista didattico. Infatti se si presta attenzione alla struttura sia del backend che del frontend si può notare come ogni componente sia nella sua specifica cartella o che sia comunque presente una forte separazione tra i vari file in base alla loro funzionalità.
Per quanto riguarda il backend non si è adottato il pattern architetturale MVC in quanto avrebbe inutilmente aumentato la complessità del codice, infatti si è cercato di utilizzare il più possibile l'archietettura a microservizi, per i motivi già ribaditi in precedenza, che di conseguenza aumenta la modularità e scalabilità del codice.
Il frontend tende anche esso ad adottare l'architettura a microservizi ove possibile, tuttavia, nel eventualità dove la struttura a microsevizi complicava il codice inutilmente si è optati per l'archietettura più adatto a tale scopo.

3 Scelte applicative

3.1 Framework e librerie utilizzate nel backend

Il backend è stato sviluppato utilizzando Java e Spring Boot, entrambi richiesti a priori dall'azienda. Le versioni utilizzate però sono state stabilite durante lo sviluppo del progetto, le versioni utilizzate infatti sono Java 23 e Spring Boot 3.3.5.
Durante lo sviluppo sono state utilizzati anche i seguenti strumenti:

- **Maven**: strumento di build e gestione delle dipendenze per progetti Java.
- **PostgreSQL**: sistema di gestione di database relazionali utilizzato per l'archiviazione degli account e dei messaggi. La scelta di PostgreSQL è stata fatta in quanto dovendo gestire più tabelle si è ritenuto più adatto rispetto ad un database NoSQL.
- **Docker**: piattaforma per la containerizzazione di applicazioni, in primo luogo utilizzata l'esecuzione del database e successivamente adotatta per l'esecuzione dell' intero progetto.

Sono state utilizzate anche le seguenti librerie relative a Spring Boot:

- **Spring starter web**: fornisce le dipendenze necessarie per lo sviluppo di applicazioni web RESTful.
- **Spring starter security**: implementa funzionalità di sicurezza per le applicazioni Spring Boot.
- **Spring starter validation**: offre strumenti necessari per validare i dati delle richieste utilizzando annotazioni Java Bean Validation.
- **Spring data JPA**: conesente l'accesso e la gestione di database relazionali tramite JPA (Java Persistence API).
- **PostgreSQL**: libreria necessaria per la connessione a database PostgreSQL.
- **Lombok**: utilizzata per ridurre il codice boilerplate tramite l'uso di annotazioni.
- **Langchain4j spring boot starter**: dipendenze per semplificare l'utilizzo di Langchain4j all'interno di progetti Spring Boot.
- **Langchain4j**: libreria per la connessione a nodi Langchain.
- **Json web token**: insieme di librerie necessarie per la gestione dei token JWT.

3.2 Framework e librerie utilizzate nel frontend

La parte del frontend utilizza Angular, richiesto sempre dall'azienda, e di conseguenza Node.js prerequisito per l'utilizzo di Angular. La versione utilizzata di Angular è la 18.2.11.
Nello sviluppo del frontend oltre ad Angular si è implementato anche PrimeNG, una libreria di componenti UI che garantisce un alta personalizzazione, grande varietà di componenti e una semplice integrazione in applicazioni Angular.

4 Deploy del progetto

Se si fosse intenzionati ad eseguire il progetto sarà sufficiente avere il codice sorgente, una volta eseguito il precedente passaggio basterà utilizzare un terminale, posizionarsi nella cartella del backend ed eseguire il comando: `docker-compose up --build .` Alternativamente sarà sufficiente aprire il progetto con un IDE ed eseguire il file `compose.yaml`.