UNIVERSITÀ DI PISA

Progetto Programmazione Avanzata

Corso di Laurea in Ingegneria Informatica
Anno Accademico 2024-2025

Bellini Matteo

1. Introduzione

1.1. Caso d'Uso

L'applicazione EzPlanner permette agli utenti di tenere traccia dei loro impegni futuri memorizzandoli su un server. Per ogni attività sarà possibile specificare la data a cui questa fa riferimento, un topic e se necessario anche una descrizione. EzPlanner mette a disposizione dei suoi utenti un'interfaccia come quella di un calendario dove sarà possibile controllare rapidamente se sono presenti, ed eventualmente in quale numero, attività memorizzate in una certa data.

1.2. Struttura dell'applicazione

EzPlanner è un'applicazione distribuita, costituita da un server che interagisce con un database SQL e da un client che comunica con il server. Il client interagisce con il server tramite il protocollo HTTP e i dati inviati vengono codificati mediante il formato JSON.

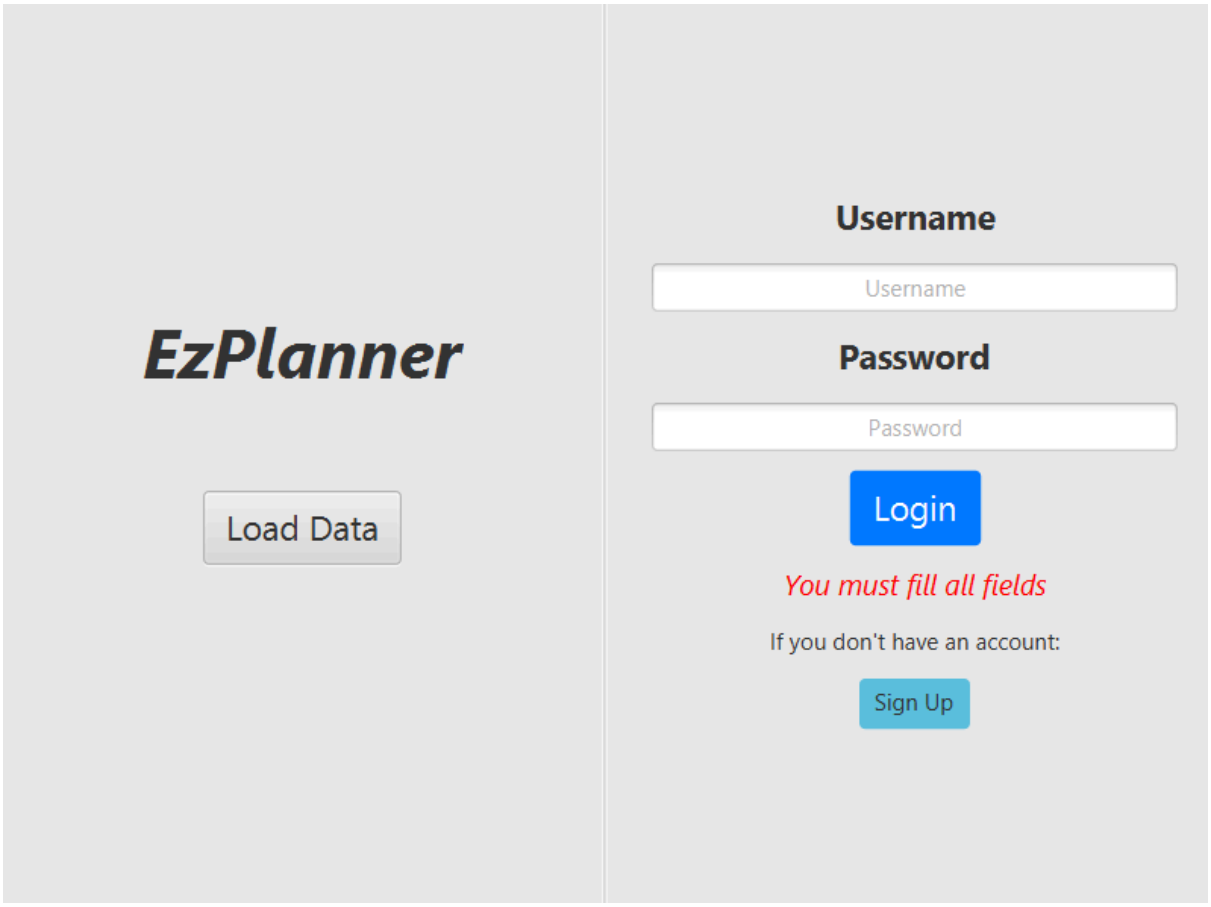
1.3. Avvio dell'applicazione

Per poter utilizzare l'applicazione è necessario avviare prima il server e successivamente il client. Il server si occuperà di creare il database e le relative tabelle se non già presenti; in seguito su richiesta del client sarà possibile popolarlo con alcuni dati a condizione che le tabelle siano vuote.
Il popolamento aggiunge due utenti ed alcune attività; gli username sono: *mb*, *marioR* e la password è *prova* per entrambi gli account.

2. Lato Client

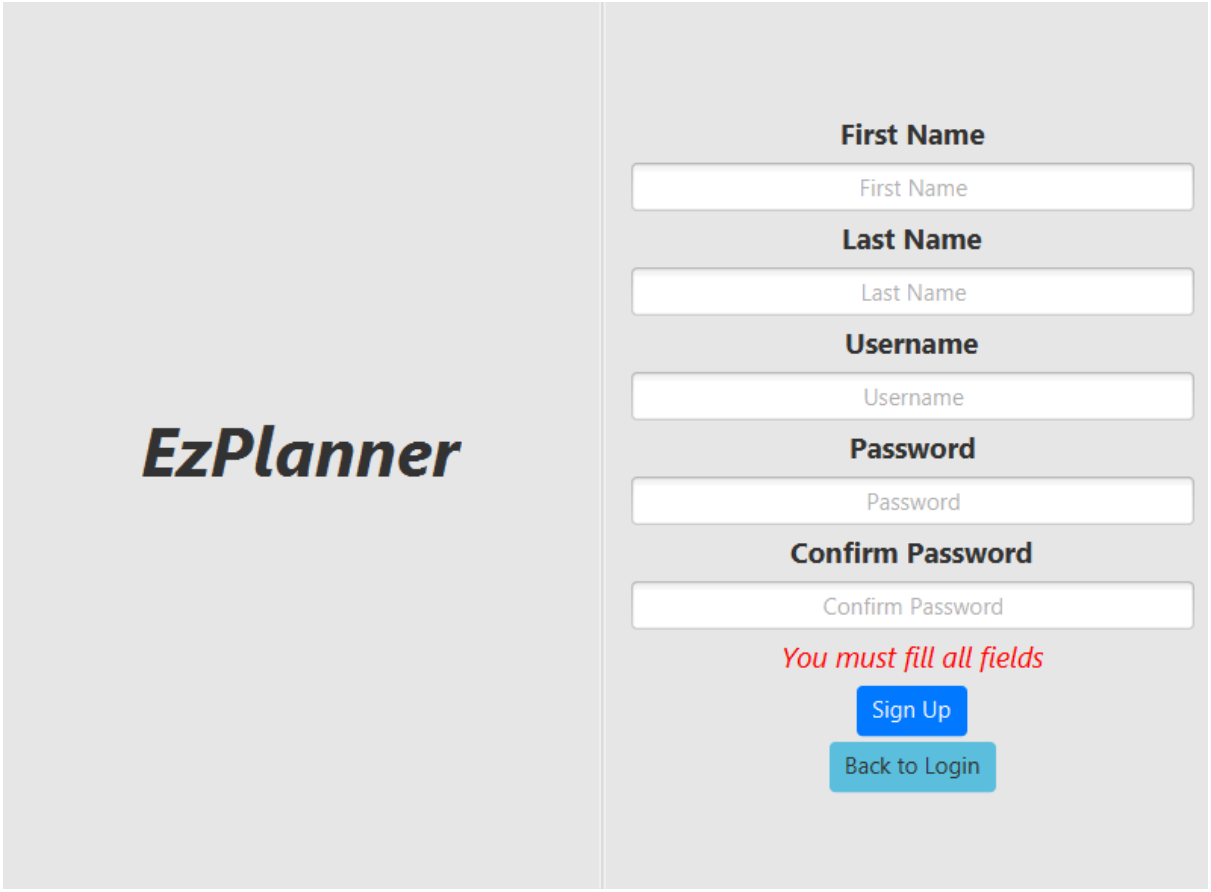
2.1. Schermate

2.1.1. Schermata Login



La schermata presenta il bottone per popolare il database, i campi dove inserire username e password per il login ed il pulsante che permette di passare alla schermata di registrazione. Nei casi in cui i campi non sono riempiti o i valori sono errati viene mostrato un messaggio di errore di colore rosso.

2.1.2. Schermata SignUp



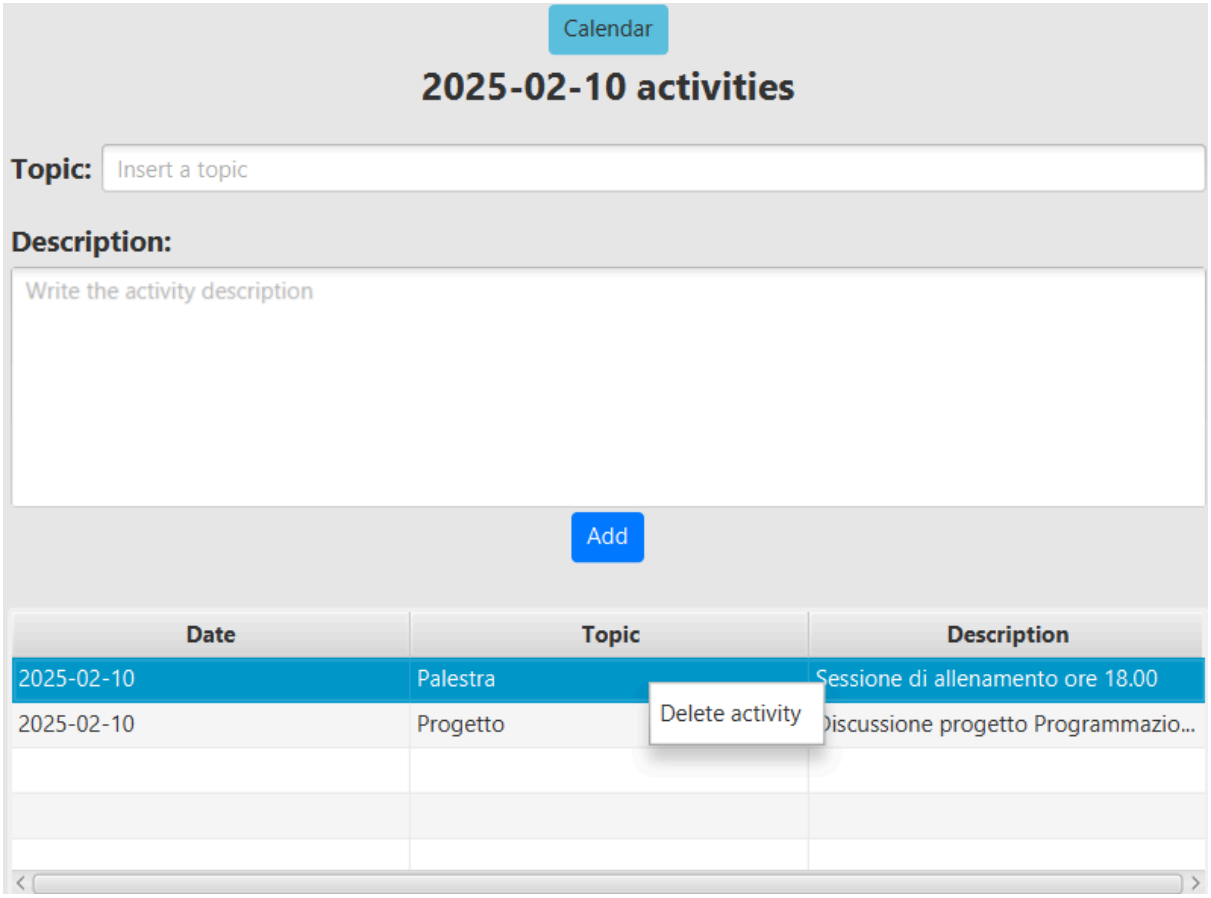
La schermata presenta i campi da riempire necessari per definire un nuovo utente. Nel caso in cui non siano stati riempiti tutti i campi o alcuni siano errati verrà mostrato un messaggio informativo di colore rosso. Sarà possibile tornare alla schermata di login tramite la pressione dell'apposito tasto.

2.1.3. Schermata Home

Questa schermata permette di visualizzare un calendario dove per ogni giorno viene mostrato il numero di attività che l'utente ha memorizzato. Nella parte alta è possibile scegliere quale mese visualizzare selezionandolo tramite il DatePicker oppure tramite la pressione dei bottoni freccia che porta la data selezionata avanti o indietro di un mese. Il pulsante reset permette di riportare la schermata alla data odierna. La data selezionata avrà sfondo celeste sul calendario.

La pressione del tasto Go permetterà di passare alla schermata di gestione delle attività per la data selezionata. È possibile passare alla schermata di gestione delle attività anche effettuando un clic su una delle celle visualizzate sul calendario.

2.1.4 Schermata Attività



Questa schermata permette di gestire le attività per una certa data. L'utente potrà inserire una nuova attività per la data selezionata riempiendo almeno il campo Topic e successivamente premendo il bottone Add. Nel caso in cui l'utente prema il bottone senza aver riempito i campi verrà mostrato un messaggio di errore di colore rosso.

In basso tramite la TableView è possibile visualizzare la lista delle attività memorizzate per la data selezionata; inoltre tramite un menù a scomparsa sarà possibile eliminarle.

Premendo il bottone Calendar l'utente viene riportato alla schermata Home.

2.2. Note tecniche sul lato client

Per poter trasferire le informazioni relative all'utente e alla data selezionata tra le varie schermate dell'applicazione è stata creata una classe SessionData che utilizza il Singleton Pattern.
Considerando che le interazioni con il server potrebbero richiedere tempi non indifferenti, le richieste sono state inserite all'interno di Task e Platform.runLater in

modo che le interfacce non rimangano bloccate durante le attese. Questa operazione non è stata effettuata nei casi di login e registrazione di un nuovo utente.

2.3. Unit Test

L'applicazione client è dotata di un Unit Test che verifica durante la fase di compilazione se il server è attivo, inviando una richiesta e verificando che la risposta sia corretta.

3. Lato Server

3.1. Lista API

- **/inizializza** **GET**: permette di popolare il database con alcuni dati su richiesta del client a condizione che le tabelle siano vuote.
- **/login** **POST**: restituisce i dati corrispondenti ad un utente ricercandoli all'interno del database tramite username e password ricevuti dal client.
- **/signUp** **POST**: riceve dal client i dati necessari per definire un nuovo utente e dopo aver controllato che l'username ricevuto non esista già viene aggiunto il nuovo utente.
- **/attivitaMese** **POST**: riceve dal client: username, mese, anno; utilizzando questi dati restituisce un elenco dove ogni elemento corrisponde ad una data ed il numero di attività memorizzate per quella data.
- **/attivitaGiorno** **POST**: riceve dal client: username, yyyy-mm-dd; utilizzando questi dati restituisce l'elenco delle attività memorizzate dall'utente per la data richiesta.
- **/inserisci** **POST**: riceve dal client i dati necessari a definire una nuova attività e successivamente la memorizza nel database.
- **/elimina** **DELETE**: elimina dal database l'attività che è stata inviata dal client.
- **/test** **GET**: permette al client di effettuare un test di connessione con il server.

3.2. Note tecniche sul lato server

Prima di mandare in esecuzione l'applicazione Spring, il server verifica tramite una classe di inizializzazione che il database esista aprendo una connessione con MySQL e cercando il nome del database nei metadati ricevuti. Nel caso in cui il database non esista viene eseguito lo script contenuto nel file "creazione.sql" che si occupa di creare lo schema, la tabella utente e quella attività. Su richiesta del client sarà poi possibile popolare il database con alcuni dati di prova che verranno inseriti eseguendo lo script contenuto nel file "popolamento.sql".

Un utente è rappresentato all'interno del database da:

- username (chiave primaria),
- nome,
- cognome,
- password.

Un'attività è rappresentata all'interno del database da:

- id (chiave primaria gestita in modalità auto increment),
- topic,
- descrizione,
- data,
- utente (chiave esterna).

Le query sono state definite all'interno di due repository: UtenteRepository ed AttivitaRepository; dove quando stato possibile si è sfruttato il meccanismo delle keywords JPA altrimenti si sono definite delle query con l'utilizzo dell'annotazione @Query.

4. Prompt ChatGPT

- How to perform serialization and deserialization with TypeAdapter of LocalDate using Gson in java.
- Colors to use for application interface (con allegato screenshot schermata applicazione).
- Come fare in modo che non ci sia spazio inutilizzato sulle colonne di una TableView in JavaFx.
- Remove all elements that are inside a GridPane.