MoneyMind

*Documentazione del progetto **MoneyMind***

Prota Raffaele, Matr. 635650

UNIVERSITÀ DI PISA

A.A. 2024/25

Indice

Capitolo 1

Introduzione

1.1 Caso d'uso

L'idea alla base di questa applicazione è la realizzazione di un servizio per la gestione delle finanze personali.
L'utente può accedere allo storico delle proprie spese, aggiungerne di nuove e, all'occorrenza, rimuoverle.
I dati gestiti all'interno dell'applicazione sono prelevati da un database opportunamente progettato, a cui il servizio accede tramite la comunicazione con un server.

1.2 Struttura dell'applicazione

L'applicazione è divisa in due parti: un client e un server.
Il client comprende tutta la parte di UI e tutte le strutture dati necessarie alla comunicazione con il server.
Il server si occupa di gestire le richieste inviate dal client e interagire con il database.

Capitolo 2

Client

2.1 Struttura interna

La struttura interna del client e' cosi' sviluppata:

```
- MoneyMindApp
|-- src
|   |- main
|      |- java
|         |- com
|            |- prota
|               |- moneymindapp
|                  |- common
|                     |- Categoria.java
|                     |- DataSession.java
|                     |- SpesaCategoria.java
|                     |- SpesaDiary.java
|                     |- SpesaRicorrente.java
|                  |- exceptions
|                     |- BadCredentialsException.java
|                     |- NotFoundException.java
|                  |- AddCostPage.java
|                  |- App.java
|                  |- DiaryPage.java
|                  |- ErrorType.java
|                  |- HomePage.java
|                  |- HttpManager.java
|                  |- LogInPage.java
|                  |- RecurrentsPage.java
|                  |- SignUpPage.java
|         |- module-info.java
|      |- resources
|         |- com
|            |- prota
|               |- moneymind
|                  |- addCostPage.fxml
|                  |- diaryPage.fxml
```

```
|                   |- homePage.fxml
|                   |- logInPage.fxml
|                   |- recurrentsPage.fxml
|                   |- signUpPage.fxml
|          |- css
|             |- style.css
|          |- images
|-- pom.xml
```

Le classi principali che gestiscono la parte client dell'applicazione sono quelle contenute all'interno della cartella *moneymindapp*.
Tali classi "sfuse", escluse quelle all'interno delle cartelle common ed exceptions e la classe HttpManager.java svolgono il ruolo di controller delle varie pagine dell'applicazione.
La classe HttpManager mette a disposizione una struttura standard per l'invio delle richieste Http al server serializzando i dati nella maniera opportuna e selezionando i percorsi adeguati ad ogni richiesta. Si occupa anche di gestire le risposte ricevute dal server.

Nella cartella *common* sono racchiuse tutte le classi utili per la gestione in locale delle strutture dati ricevute dal server o da visualizzare all'interno dell'interfaccia (tabelle e grafici).

La cartella *exceptions* contiene le classi che implementano nuovi tipi di eccezioni specifiche per il caso d'uso.

2.2 Schermate

2.2.1 Log in

- Appare all'avvio dell'applicazione;
- Presenta un pulsante **DB initializer** che permette di popolare il database in automatico al primo avvio;
- Pulsante **registrati** che rimanda alla pagina della registrazione;
- L'account per il test e': Username='test' e Password='test'.

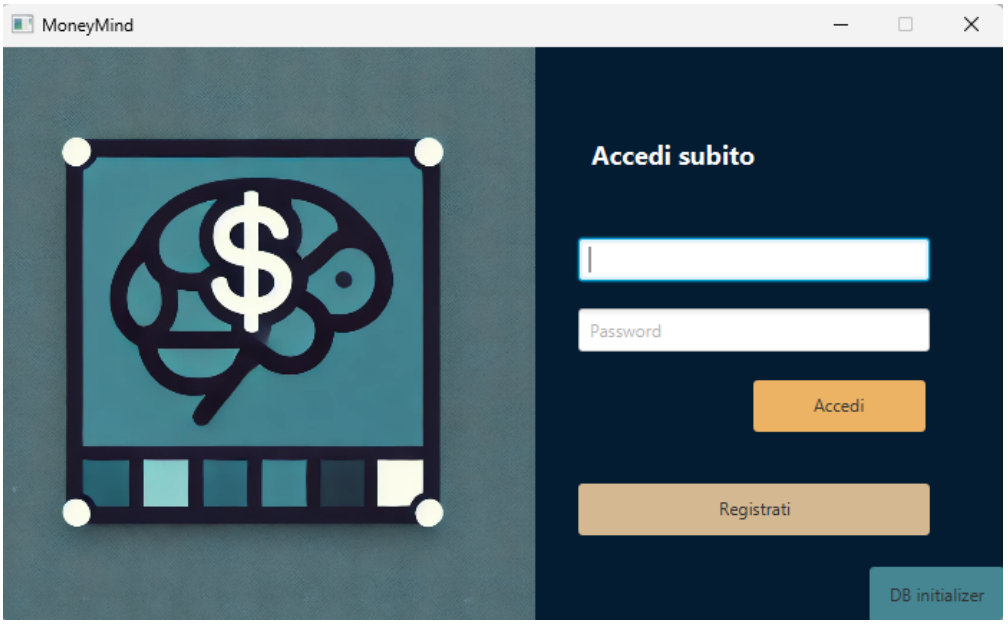

FIGURA 2.1: Pagina di login

2.2.2 Sign up

- Permette di registrarsi;
- Rimanda alla pagina di login in caso di registrazione completata con successo.

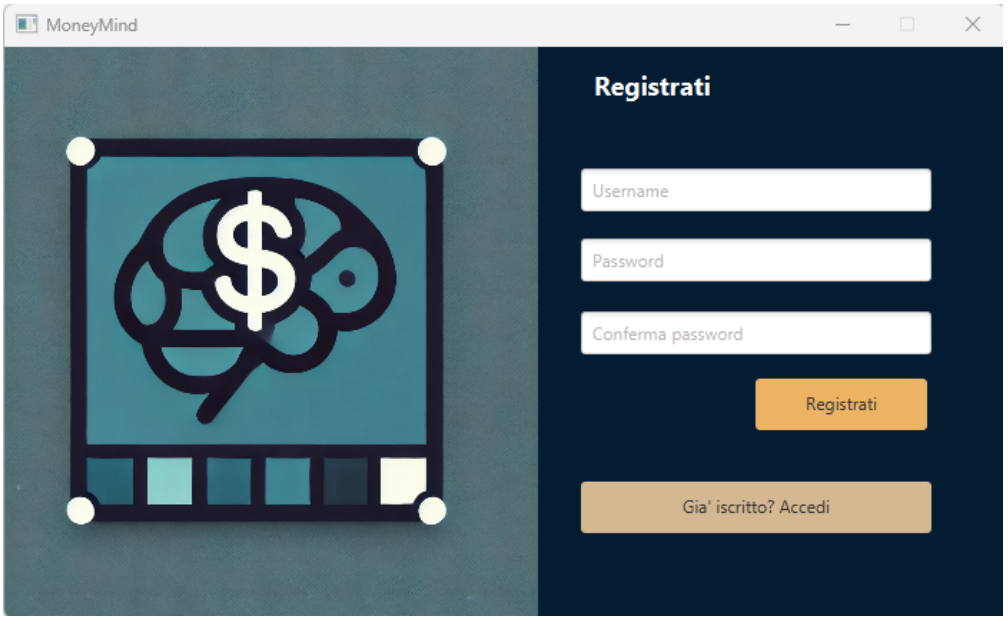

FIGURA 2.2: Pagina di registrazione

2.2.3 Home page

- Presenta un menu verticale con i collegamenti alle altre pagine dell'applicazione (il pulsante **Esci** riporta l'applicazione alla pagina di login);
- Visualizza lo stato delle spese del mese in corso fino al giorno corrente;
- Visualizza un grafico a torta con la ripartizione delle spese nel mese in corso fino al giorno corrente.

FIGURA 2.3: Pagina principale

2.2.4 Diario

- Presenta una tabella con lo storico di tutte le spese dell'utente. Ogni spesa ha un flag che indica se ricorrente[1] o meno.
- Presenta un bottone per l'aggiornamento della tabella;
- Presenta un bottone per l'aggiunta di una nuova spesa[2].
- Offre la possibilità' di eliminare una spesa precedentemente inserita tramite click con tasto destro sulla spesa che si vuole eliminare.

[1] Una spesa ricorrente e' una spesa che viene sostenuta periodicamente che quindi ricorre nel tempo.
[2] Apre una nuova finestra descritta al paragrafo **2.2.5**

FIGURA 2.4: Pagina riassuntiva delle spese

FIGURA 2.5: Rimozione di una spesa

2.2.5 Nuova spesa

- Presenta un form sulla sinistra dove inserire i dati relativi alla spesa da inserire;

- Quando viene scelta una spesa come ricorrente, vengono abilitati i due *ChoiceBox* che permettono di specificare il periodo di ricorrenza(ogni quanto si ripete);
- Sulla destra e' visualizzata una tabella contenente le categorie tra cui scegliere.

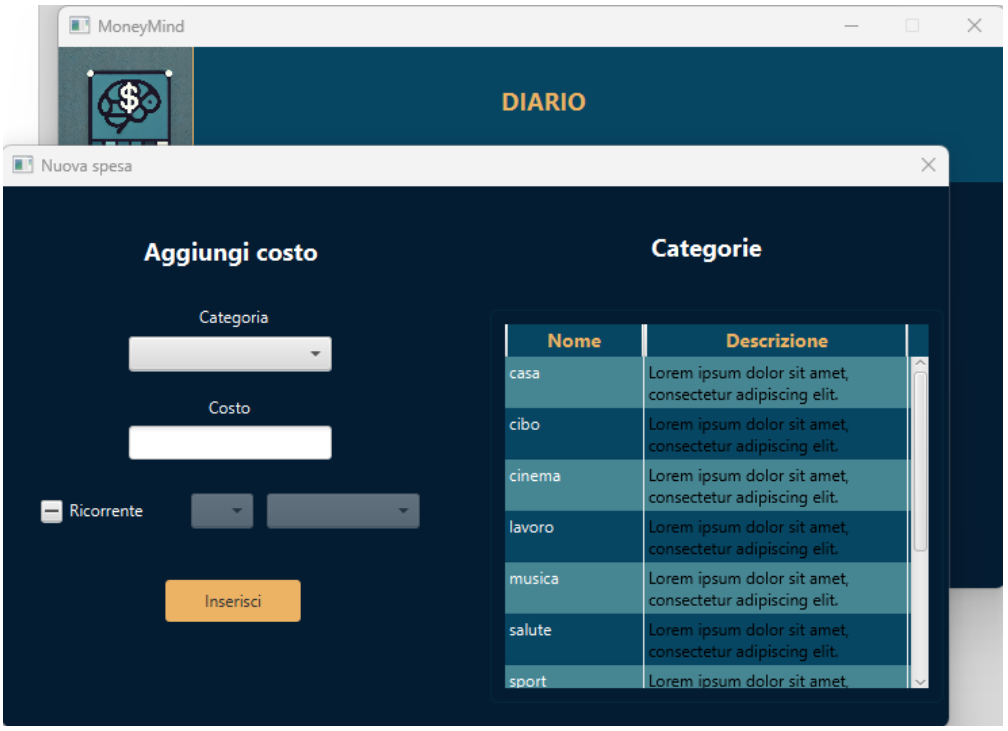

FIGURA 2.6: Inserimento di una nuova spesa

2.2.6 Ricorrenti

- Presenta una tabella con lo storico di tutte le spese ricorrenti e le relative date dei prossimi pagamenti.
- Offre, come nella pagina **Diario**, la possibilità' di eliminare le spese.

FIGURA 2.7: Pagina riassuntiva delle spese

2.3 Interazione con il server

I dati vengono inviati al server dopo essere stati serializzati in formato json.
All'interno delle classi controller, viene creato il corpo della richiesta con i necessari parametri. Viene anche aggiunto un parametro "speciale", **query type**: tale parametro server alla classe HttpManager per individuare il corretto percorso per inoltrare la richiesta al server.
La classe HttpManager si occupa, quindi, di prelevare il corpo della richiesta dalla classe controller e inviare la richiesta al server con tutte le specifiche necessarie relative al protocollo Http. Ha anche il compito di ricevere le risposte del server, trasformarle in un formato standard e successivamente restituirle alle classi controller.

Le richieste Http sono gestite con l'utilizzo delle classi *HttpClient, HttpRequest e HttpResponse* appartenenti al package *java.net.http*

Capitolo 3

Server

3.1 Struttura interna

La struttura interna del server e' cosi' sviluppata:

```
- MoneyMindServer
|-- src
|   |- main
|      |- java
|         |- com
|            |- prota
|               |- MoneyMindServer
|                  |- Common
|                     |- SpesaCategoria.java
|                     |- SpesaUpdater.java
|                  |- DBentity
|                     |- Categoria.java
|                     |- Spesa.java
|                     |- User.java
|                  |- DBrepository
|                     |- CategoriaRepository.java
|                     |- SpesaRepository.java
|                     |- UserRepository.java
|                  |- CategoriaController.java
|                  |- DBController.java
|                  |- MoneyMindServerApplication.java
|                  |- SpesaController.java
|                  |- UserController.java
|      |- resources
|         |- DBInsert.sql
|         |- application.properties
|   |- test
|      |- java
|         |- com
|            |- prota
|               |- MoneyMindServer
|                  |- MoneyMindServerApplicationTests.java
```

```
|                   |- SpesaControllerTest.java
|-- pom.xml
```

Il server e' composto da:

- Classi entità che mappano le relazioni del database tramite il costrutto *@Entities* nel contesto Spring JPA;
- Classi controller che contengono gli endpoint delle chiamate http effettuate dal client;
- Classi repository che tramite l'interfaccia CrudRepository e l'uso di query native gestiscono l'interazione con il database;
- Classi di utilità' (SpesaCategoria: usata come classe wrapper per contenere i risultati di una interrogazione al database; SpesaUpdater: utilizzata per implementare un metodo ricorrente per l'aggiornamento del campo *prossimopagamento* nel database).