2023

*Gestore delle Spese tra Coinquilini*

Antonio Hu

937652

# Indice

* Pagina 3 ---------------------------------------------------------- introduzione
* Pagina 4 ----------------------------------------- elence delle caratteristiche
* Pagina 12 ----------------------------------------------- descrizione database
* Pagina 13 --------------------------------------------- dipendenze e versioni
* Pagina 14 ------------ esecuzione del pacchetto jar ed altre informazioni

# Introduzione

* **Obiettivo:**

L'obiettivo principale dell'applicazione è quello di semplificare la gestione delle spese di un appartamento con più coinquilini, fornendo una piattaforma centralizzata per l'inserimento e la ripartizione delle spese. Grazie al sistema distribuito, ogni coinquilino può utilizzare il proprio client per inserire le spese sostenute e la loro ripartizione, e per visualizzare il saldo corrente.

* **Connessione Client - Server:**

Il progetto consiste in un sistema distribuito per la gestione delle spese di un appartamento abitato da più coinquilini. Il sistema è composto da un Server e da Client, e utilizza una connessione Socket per permettere agli utenti di interagire con il Server.

* **Funzionalità:**

L'applicazione offre una serie di funzionalità per la gestione delle spese, come la possibilità di inserire una spesa con l'importo, la data e la descrizione, di visualizzare tutti i coinquilini in una tabella, di pagare una spesa inserendo il codice della spesa, di consultare la lista delle spese da pagare, delle spese pagate e la tabella del bilancio.

* **Autenticazione:**

Il sistema prevede anche un sistema di autenticazione, che richiede il login degli utenti e permette di distinguere tra utenti Admin e User. Gli utenti Admin hanno accesso a ulteriori funzionalità rispetto agli utenti User, come la possibilità di cancellare tutte le informazioni degli utenti e delle spese.

* **Interfaccia Grafica:**

L'obiettivo dell'applicazione è quello di semplificare la gestione delle spese, riducendo il rischio di errori nella ripartizione delle spese e offrendo una visione chiara del bilancio delle spese. Grazie al sistema distribuito e al supporto di JavaFX, l'applicazione offre un'interfaccia user-friendly e intuitiva per gli utenti.

# Elenco delle caratteristiche del progetto

L'applicazione è caratterizzata da diverse funzionalità, che permettono agli utenti di gestire in modo efficiente le spese di un appartamento condiviso. Di seguito sono elencate le principali caratteristiche dell'applicazione:

1. **Sistema distribuito:** l'applicazione è composta da un Server e da Client, e utilizza una connessione Socket per permettere agli utenti di interagire con il Server.
   1. Nel caso di prova: host = “localhost” port = “8080”

Per sviluppatore-> pacchetto “NetWork”:

* + 1. **Classe “Client”:** inizializza le due classi:
       1. “CommandHandler” -> per scegliere il tipo di messaggio da inviare.
       2. “MessageHandler”-> invia, riceve e analizza il messaggio ricevuto, quindi contiene gli oggetti “output” e “input” del socket Client.
    2. **Classe “Server”:** inizializza le due classi:
       1. “ClientHandler”-> un singolo thread tenuto per la comunicazione con il Client, elabora il messaggio ricevuto dal Client poi richiama i metodi dei controller per svolgere le operazioni richieste su databse.
       2. “ServerMessageHandler”-> invia,riceve e analizza il messaggio recevuto,quindi contiene gli oggetti “output” e “input” del socket Server.
    3. **Classe “Message”:** contiene tutti i messaggi usati per la comunicazione:
       1. “Message”-> classe padre di tutti i messaggi da inviare, ha un attributo come MessageType
       2. “MessageType”-> classe enum che contiene tutti i tipi dei messaggi usati
       3. Tutti i messaggi vengono classificati in: (request) e (response):
          1. Request: inviato da Client al Server
          2. Response: inviato da Server al Client

1. **Login, Register e Logout:** l'applicazione prevede un sistema di autenticazione per gli utenti, che richiede il login per accedere alle funzionalità dell'applicazione. Inoltre, è possibile effettuare la registrazione e il logout dalla propria sessione.
   1. I dati relativi agli utenti vengono registrati nella database, nel caso di prova è database MySQL
   2. Interfaccia di login:
      1. Immagine che contiene testo

         Descrizione generata automaticamente
   3. Interfaccia di register:
      1. Immagine che contiene testo

         Descrizione generata automaticamente

Per sviluppatore->

1. **Credenziali per accedere al database:**
   * + 1. Username database: root
       2. Password: 825310894
       3. Porta: 3306
       4. Ip: localhost
       5. url: jdbc:mysql//localhost:3306/test
          1. “test” o “database\_for\_test”
     1. **Tabella utilizzata: “users”**
        1. Un esempio di screenshot:
           1. Immagine che contiene tavolo

              Descrizione generata automaticamente
2. **Database “test”:**

destinato per il funzionamento dell’applicazione

1. **Database “database\_for\_test”:**

destinato solo per le classi di test Junit

1. **Utenti Admin e User:** l'applicazione prevede due tipi di utenti, gli Admin e gli User. Gli utenti Admin hanno accesso a funzionalità aggiuntive rispetto agli User.
   1. **Gli utenti nella tabella “uesrs” hanno un campo (colonna) chiamatoo “role” che è il ruolo del utente**
2. **Interfaccia diversa prima e dopo il login:** l'interfaccia dell'applicazione cambia a seconda del fatto che l'utente abbia effettuato o meno il login.
   1. **Prima di login -**>
      1. Login (accedere con le credenziali registrate)
      2. Register (creare un account nuovo scegliendo anche il ruolo)
      3. Exit (uscire/ chiudere dal/ il programma)

Immagine che contiene testo, montagna, natura, aria aperta

Descrizione generata automaticamente

* 1. **Dopo login ->**
     1. **Visualizzazione il proprio id identificativo**



* + 1. Logout

Immagine che contiene testo

Descrizione generata automaticamente

* + 1. Inserire una spesa

Immagine che contiene testo

Descrizione generata automaticamente

* + 1. Visualizzare tutti i coinquilini

Immagine che contiene testo

Descrizione generata automaticamente

* + 1. Pagare una spesa

Immagine che contiene testo

Descrizione generata automaticamente

* + 1. Consultazione lsita delle spese da pagare

Immagine che contiene testo

Descrizione generata automaticamente

* + 1. Consultazione lista delle spese pagate

Immagine che contiene testo

Descrizione generata automaticamente

* + 1. Consultazione tutta la cronologia

Immagine che contiene testo

Descrizione generata automaticamente

* + 1. Consultazione la tabellabilancio

Immagine che contiene testo

Descrizione generata automaticamente

------------ solo per Admin ---------------------

* + 1. Cancellare tutte le informazioni (di tabella users)

Immagine che contiene testo

Descrizione generata automaticamente

* + 1. Cancellare le spese e i conti (di tabella “expense” + “balance”)
       1. Immagine che contiene testo

          Descrizione generata automaticamente

Immagine che contiene testo, natura

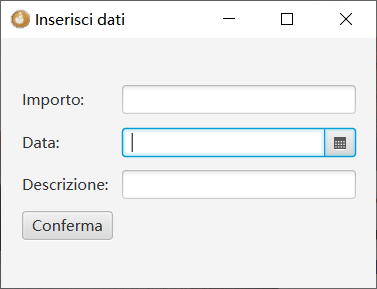
Descrizione generata automaticamente

--- L’immagine illustrato è un esempio dell’interfaccia di Admin

1. **Interfaccia per Admin e User diversa:** l'interfaccia dell'applicazione cambia a seconda del tipo di utente che ha effettuato il login. Gli utenti Admin hanno accesso a funzionalità aggiuntive rispetto agli User.
   1. **User:**
      1. Immagine che contiene testo, natura

         Descrizione generata automaticamente
   2. **Admin:**
      1. Immagine che contiene testo, natura

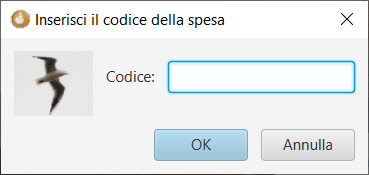
         Descrizione generata automaticamente
2. **Inserimento di una spesa:** l'utente può inserire una nuova spesa specificando l'importo, la data e una descrizione.
   1. **Interfaccia:**



* + - 1. Importo: importo della spesa con 2 cifre dopo virgola sentite
      2. Dat: possibilità di scegliere una data dal calendario con il mouse senza input sulla tastiera
      3. Descrizione: dettaglio della spesa

1. **Visualizzazione di tutti i coinquilini:** l'utente può visualizzare una tabella contenente i nomi di tutti gli utenti registrati nell'applicazione.
   1. **Interfaccia:**
   2. Immagine che contiene tavolo

      Descrizione generata automaticamente
      1. Id = codice identificativo dell’utente
      2. Username = informazione per individuare il nome proprio
2. **Pagamento di una spesa:** l'utente può pagare una spesa inserendo il codice della spesa.
   1. **Interfaccia:**



1. Codice = codice univoco di una spesa, l’utente può consultare le tabelle di spese per ottenerlo
2. **Consultazione della lista delle spese da pagare:** l'utente può consultare la lista delle spese che devono ancora essere pagate.
   1. **Interfaccia:**

Immagine che contiene testo

Descrizione generata automaticamente

1. **Consultazione della lista delle spese pagate:** l'utente può consultare la lista delle spese che sono già state pagate.
   1. **Interfaccia:**

Immagine che contiene tavolo

Descrizione generata automaticamente

1. **Consultazione della cronologia delle spese:** l'utente può consultare la cronologia delle spese inserite nell'applicazione.
   1. **Interfaccia**



1. **Consultazione della tabella del bilancio:** l'utente può consultare la tabella del bilancio, che mostra l'importo da pagare, l'importo pagato e il saldo corrente.
   1. **Interfaccia**

Immagine che contiene tavolo

Descrizione generata automaticamente

1. **Cancellazione delle informazioni degli utenti:** gli utenti Admin hanno la possibilità di cancellare tutte le informazioni degli utenti registrati nell'applicazione.
2. **Cancellazione delle spese e della tabella del bilancio:** gli utenti Admin hanno la possibilità di cancellare tutte le spese e la tabella del bilancio.

Grazie alla combinazione di queste funzionalità, l'applicazione offre una piattaforma centralizzata per la gestione delle spese di un appartamento condiviso, semplificando la ripartizione delle spese e offrendo una visione chiara del bilancio.

# Descrizione di database

La gestione delle informazioni degli utenti e delle spese avviene tramite l'utilizzo del database MySQL, mentre MySQL Workbench8.0CE è il tool utilizzato per la modellazione del database e la sua gestione:

Applicazione per database: MySQL,MySQL Workbench 8.0 CE

Database utilizzate:

Test: utilizzo principale dell’applicazione.

Database\_for\_test: per test Junit.

Esempio di connessione:

/\*\*  
 \* Instantiates a new Expense controller.  
 \*  
 \* **@param** url "jdbc:mysql//localhost:3306/test" , url address for Database connection  
 \* **@param** username "root"  
 \* **@param** password "825310894"  
 \* **@throws** SQLException the sql exception  
 \*/  
public ExpenseController(String url, String username, String password) throws SQLException {  
 dbConnection = DriverManager.getConnection(url, username, password);  
  
}

# Dipendenze e versioni

Pacchetti jar introdotti per implementazione del progetto

* Java SDK 17
* Javafx – dk – 19.0.2.1
* Junit 5

# Esecuzione del pacchetto jar e altre informazioni

* Pagina github: <https://github.com/ANTONIOHU10/ProgettoSoftware2023>

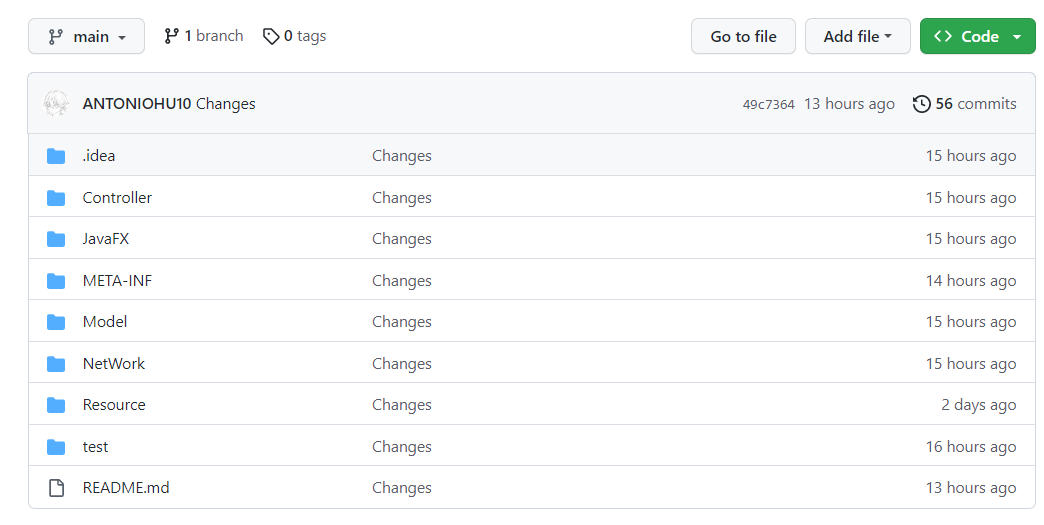


Immagine che contiene testo

Descrizione generata automaticamente

* Esecuzione jar:

### Client:

From root

E:-> ***cd :\ingegneria\_software\_progetto\_finale\ProgettoSoftware2023\out\artifacts\Client\_jar***

path del pacchetto jar

poi per esecuzione:

***java --module-path "E:\javaFX\javafx-sdk-19.0.2.1\lib" --add-modules javafx.controls,javafx.fxml -jar Client.jar***

### Server:

From root

E:-> ***E:\ingegneria\_software\_progetto\_finale\ProgettoSoftware2023\out\artifacts\Server\_jar***

path del pacchetto jar

poi per esecuzione:

***java -jar Server.jar***