Report JBudget: Francioni Niccolò 105325

Concetti generali

I principali concetti individuati per la realizzazione dell'applicazione di controllo delle spese familiari sono:

Conto: usato per rappresentare ad esempio un conto corrente bancario, una cassa contanti o una carta di credito e le relative operazioni possibili per la gestione delle spese e degli introiti. Divisi in due tipologie conti di disponibilità e conti di debito.

Movimento: usato per rappresentare un'entrata o una uscita da un conto.

Transazione: usata per rappresentare operazioni che possono coinvolgere più movimenti.

Responsabilità delle classi

Per rappresentare questi concetti sono state utilizzate le seguenti classi:

AccountInterface: Interfaccia implementata dalle classi che rappresentano un conto, permette di accedere o modificare le informazioni di un conto. Consente inoltre di accedere ed eventualmente filtrare la lista dei movimenti di un conto e di aggiungerne ulteriori, modificando il saldo attuale.

AccountType: enumerazione usata per descrivere la tipologia di conti.

Account: classe che implementa l'interfaccia AccountInterface , usata per descrivere un conto generico attraverso un ID, un proprietario, un saldo, una descrizione, una tipologia di conto ed una lista dei movimenti.

BankAccount: classe che estende la classe Account, usata per descrivere un conto corrente bancario attraverso l'aggiunta di informazioni come la banca del conto ed il suo codice iban.

CreditCard: classe che estende la classe Account, usata per descrivere una carta di credito attraverso l'aggiunta di una soglia massima di spesa ed un metodo che ogni mese si occupa di azzerare il saldo della carta e di e di sottrarlo al conto corrente associato.

Bancomat: classe che estende la classe CreditCard , con la differenza che ogni movimento viene eseguito sul conto corrente al momento e non a fine mese.

MovementInterface: interfaccia implementata dalle classi che rappresentano un singolo movimento, permette di accedere e di modificare le informazioni relative al movimento, come l'importo, la data o il conto associato.

MovementType: enumerazione usata per descrivere la tipologia del movimento: spesa o entrata.

MovementPredicate: classe utilizzata per definire Predicate usati per filtrare i movimenti all'interno di una lista in base alla data o il tag di appartenenza.

Movement: classe che implementa l'interfaccia MovementInterface , e rappresenta un singolo movimento .

TransactionInterface:interfaccia implementata dalle classi che rappresentato una transazione , permette di accedere e di modificare le informazioni relative alla transazione , ereditate dai movimenti che la compongono, come: lista dei movimenti , data o lista dei tags.

TransactionPredicate: classe utilizzata per definire Predicate usati per filtrare le transazioni all'interno di una lista in base alla data o il tag di appartenenza.

Transaction: classe che implementa l'interfaccia TransactionInterface , con la possibilità di effettuare i movimenti che la compongono.

TagInterface: interfaccia usata per definire una categoria di movimento.

Tag: classe che implementa l'interfaccia TagInterface, costituita da un nome e una descrizione della categoria.

LedgerInterface: interfaccia implementata dalle classi che hanno la responsabilità di gestire tutti i dati dell'applicazione. E' responsabile dell'aggiunta dei conti, dell'aggiunta e cancellazione delle transazioni, della creazione e cancellazione dei tag. Mantiene la lista delle transazioni schedulate. Inoltre si occupa di controllare ed eventualmente eseguire se ci sono transazioni schedulate per la data odierna.

Ledger: classe che implementa l'interfaccia LedgerInterface che contiene le liste di tutti i conti ed i tag creati e di tutte le transazioni effettuate o schedulate.

ReportInterface: Interfaccia implementata dalle classi che realizzano un report delle statistiche di un account.

Report: Classe, che implementa Report Interface, usata per generare un report sulle spese/entrate di un account, in riferimento ai vari tag.

PersistanceManager: interfaccia implementata dalle classi che si occupano della persistenza dei dati.

JsonPersistanceManager: classe che implementa l'interfaccia PersistanceManager e si occupa di salvare i dati dell'applicazione in files JSon.

RuntimeTypeAdapterFactory:classe implementata nelle librerie Google Gson, che estende la classe TypeAdapter, usata per la deserializzazione runtime degli oggetti di tipo generico da formato JSon. Nota: E' stato necessario includere la classe, come indicato, in quanto la classe non è fornita nel package standard.(Deserializzazione lista oggetti polimorfici)

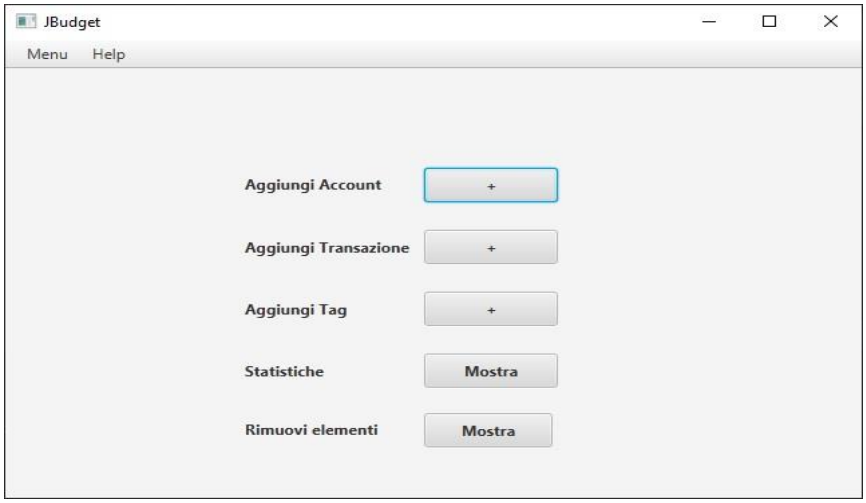
Controller: classe controller dell'applicazione con lo scopo di intermediario tra modello e vista dell'applicazione.

Persistenza:

La persistenza dei dati dell'applicazione è implementata in files di tipo JSon attraverso lo strumento Google Gson.

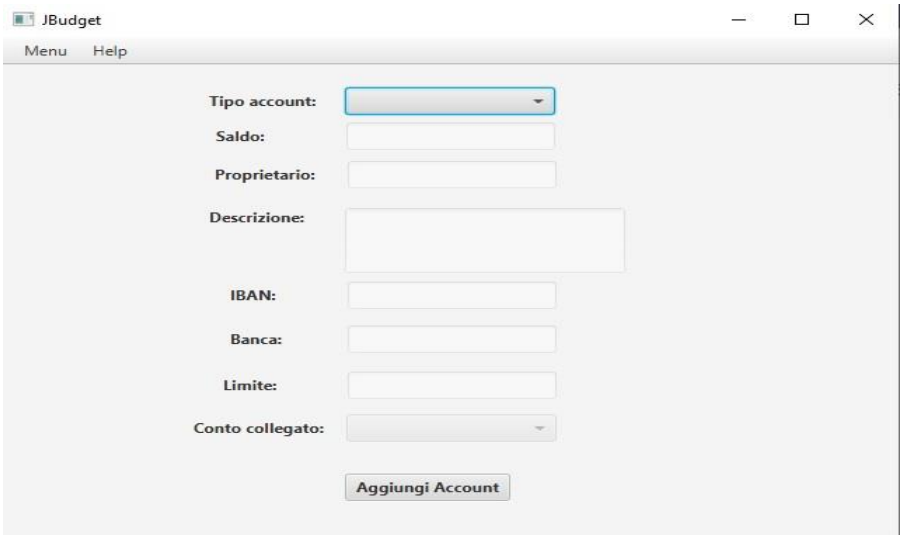
Interfaccia:

L'interfaccia grafica è stata implementata attraverso JavaFx e SceneBuilder e si compone delle seguenti scene:



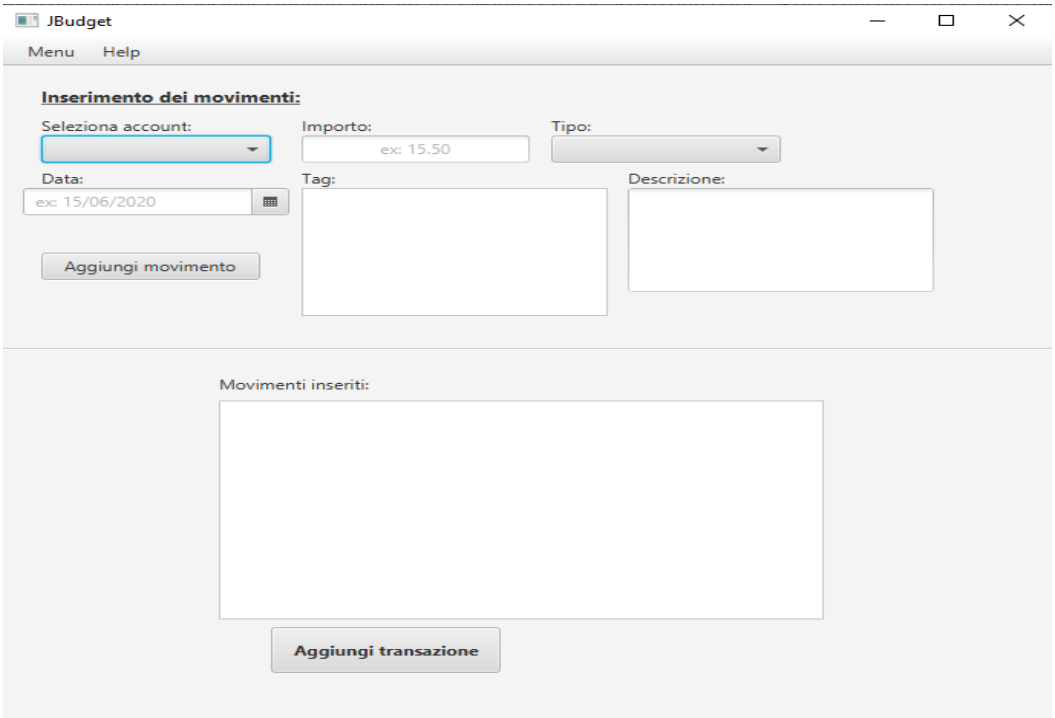
Un menu iniziale che permette di scegliere l'operazione da andare ad eseguire.

La scena è composta dal file fxml "startwindow.fxml" e dal relativo controller "StartWindowController".



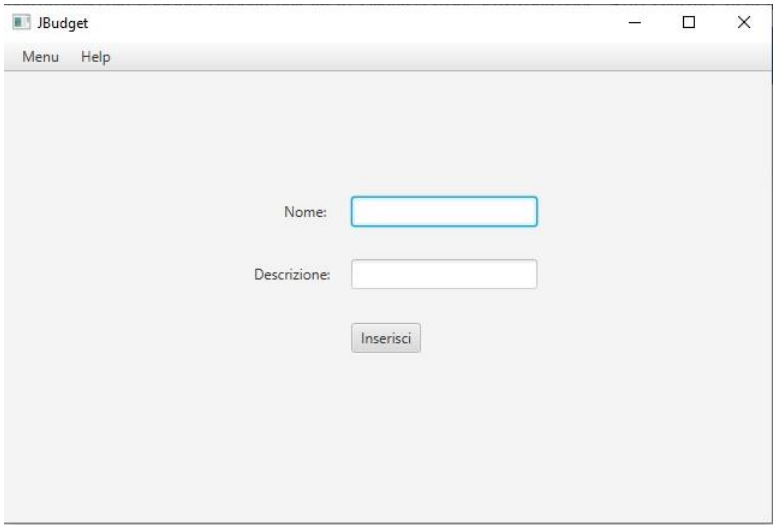
Un form per l'inserimento di un nuovo account.

La scena è composta dal file fxml "accountform.fxml" e dal relativo controller "AccountFormController".



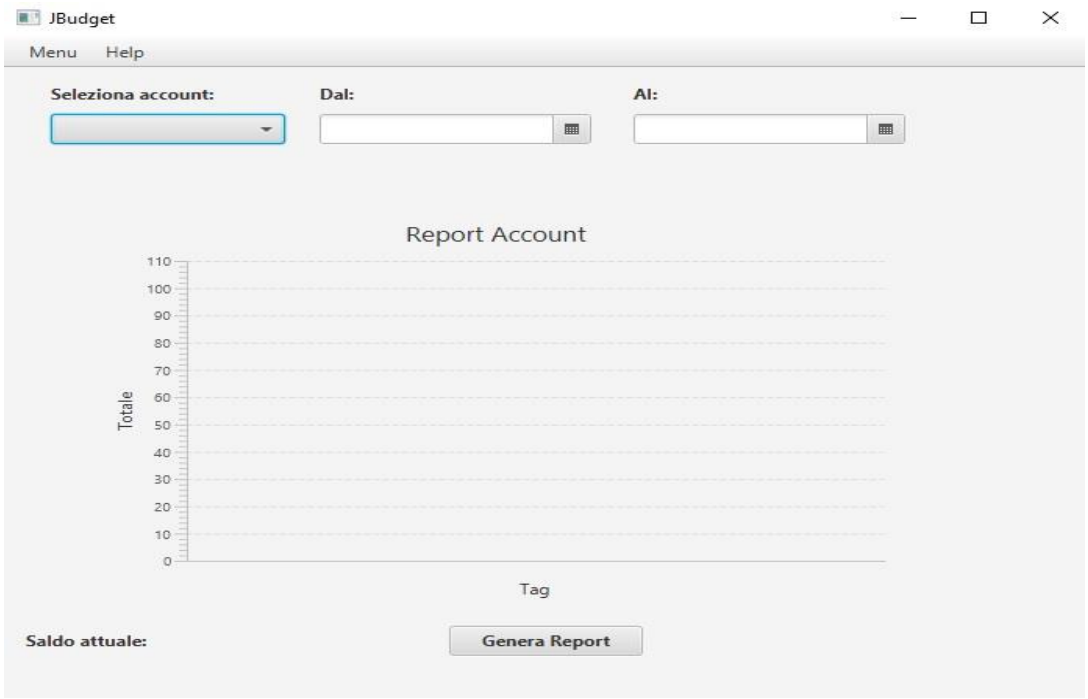
Un form per l'inserimento di una nuova transazione.

La scena è composta dal file fxml "transactionform.fxml" e dal relativo controller "TransactionFormController".



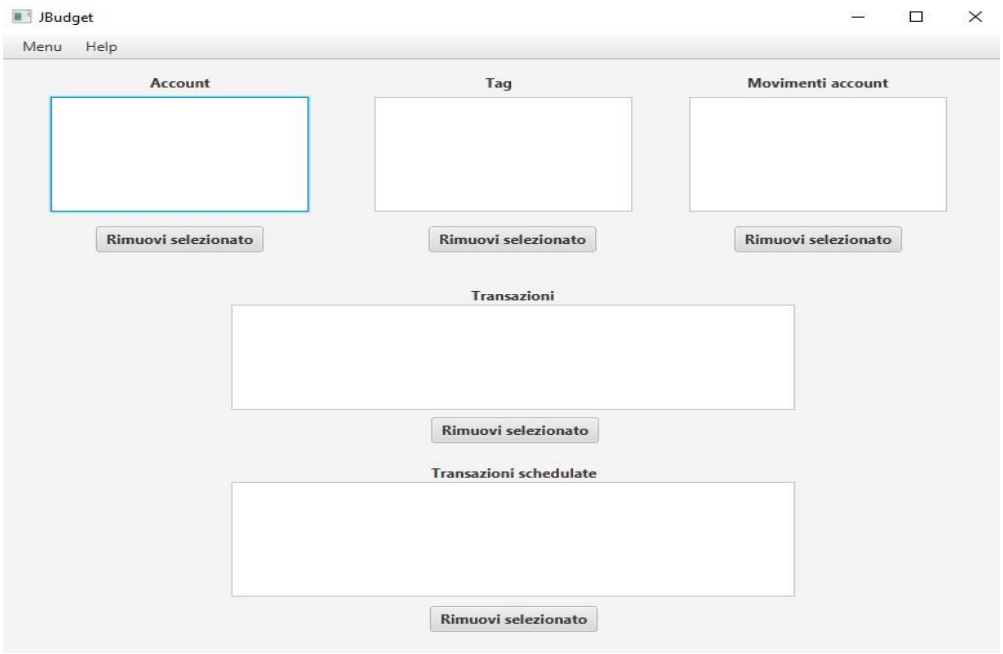
Un form per l'inserimento di un nuovo tag.

La scena è composta dal file fxml "tagform.fxml" e dal relativo controller "TagFormController".



Una finestra per mostrare attraverso un grafico le statistiche di un account.

La scena è composta dal file fxml "report.fxml" e dal relativo controller "ReportController".



Una finestra per la rimozione degli elementi inseriti ed il conseguente ripristino del saldo dei conti relativi.

La scena è composta dal file fxml “removeitems.fxml” e dal relativo controller “RemoveItemsController”.

Funzionalità implementate:

L’applicazione è in grado di creare conti correnti, carte di credito, bancomat e di eseguire movimenti e transazioni su di essi. Inoltre vi è anche la possibilità di inserire transazioni con una data futura che verranno schedulate ed eseguite successivamente, al raggiungimento della data inserita. E’ possibile anche creare ed inserire nuovi tag che descrivono le categorie di movimenti e transazioni, e anche generare report sulle statistiche riferite ad un determinato account. Infine i dati dell’applicazione vengono salvati per garantirne la persistenza.

Conti correnti, carte di credito e bancomat sono implementati dalle classi: AccountInterface, Account, AccountType, BankAccount, CreditCard e Bancomat.

Movimenti e transazioni sono implementati dalle classi: MovementInterface, Movement, MovementType, TransactionInterface e Transaction.

I tag sono implementati dalle classi: Tag e TagInterface.

I report sono implementati dalle classi: Report e ReportInterface.

La persistenza è implementata attraverso le classi: PersistanceManager e JsonPersistanceManager.

Test:

I test sviluppati nelle classi BancomatTest, BankAccountTest, CreditCardTest, LedgerTest e ReportTest servono a verificare che i metodi all'interno delle classi per la creazione, l'inserimento e la gestione dei movimenti, transazioni e account insieme a quello di raccolta dati del report funzionino correttamente, confrontando il risultato ottenuto con quello aspettato.

Javadoc:

I commenti nelle varie classi per la descrizione nei javadoc sono presenti , ma non ho effettuato la generazione in formato HTML di quest'ultimi, in quanto non sono riuscito a risolvere un errore che si presenta quando tento la generazione sia attraverso il task Gradle, sia attraverso il tool di IntelliJ Idea, sia attraverso riga di comando.