RELAZIONE FINALE “JBUDGET”

Vittorio Rinaldi 100763

Premessa:

L’ applicazione è pubblicata dall’autore come “Open Source senza scopo di lucro” (licenza GPL v3).

Pattern utilizzati:

* MVC (Model-View-Controller, architetturale)
* Singleton (creazionale).
* Facade (tramite il Ledger) nell’iterazione tra l’utente ed il resto del model (strutturale)

Sviluppato utilizzando le seguenti versioni:

* Gradle 6.5
* Java 11
* JavaFX 14

È presente documentazione JavaDoc!

Note dello sviluppatore

Unicamente per questioni legate alle tempistiche di consegna, l’interfaccia utente implementata è in fase di lavorazione: con questo si intende che la filosofia scelta dallo sviluppatore è “Qualità prima della quantità”.

In altre parole, a costo di un minor ritmo, ma con ben maggiore usabilità, *si è scelto* di NON intraprendere una programmazione imprudente ed acerba, ma una che prevedesse la cura di ogni funzionalità prima della loro pubblicazione.

Attualmente, tramite interfaccia grafica è possibile gestire perfettamente la creazione di nuovi Account e Tag nell’applicazione, la loro rimozione, nonché di visualizzarne le caratteristiche.

Si sono risolti i problemi legati alla precedente consegna e la parte di interfaccia grafica accessibile all’utente in questo momento è garantita come perfettamente funzionante (anche al netto di imprevisti); si fa notare come lato back end moltissime altre funzionalità siano curate e già altrettanto pronte, necessitanti unicamente di essere collegate all’utente finale tramite GUI (si parla di: gestione dei movimenti, gestione transazioni, la schedulazione di transazioni, accantonamento di spese per Tag specifici (budget) e report consuntivi che rappresentano il saldo tra entrate ed uscite). 🡪 Saranno disponibili a breve.

Anche nel lungo termine è in programma un ampliamento del programma:

* Lettura e scrittura dei dati utilizzando file Json, per garantire la sincronizzazione tra dispositivi, magari tramite cloud direttamente dall’applicazione.

I file FXML sono stati separati dalla directory del FxCcontroller per una questione di sintassi e pulizia;

sono locati nella cartella Resources.

Per iniziare…

L’applicazione può essere avviata tramite qualsiasi IDE importando un progetto Gradle, o ancora più facilmente tramite Console tramite il comando “gradle run” nella directory.

Package it.unicam.cs.pa.jbudget100763

Qui sono contenute tutte le interfacce e classi del Progetto.

**Main:** Classe principale che permette di avviare l’applicazione e la GUI scelta, attualmente JavaFX.

Package it.unicam.cs.pa.jbudget100763.controller

include la classe che ha il compito di ricevere i comandi dell'utente (in genere attraverso il View) e di attuarli modificando lo stato degli altri due componenti del MVC.

**Controller:** ha la responsabilità di ricevere i comandi dell'utente tramite la View e di attuarli modificando lo stato del Model. Gestisce il Ledger, il Budget ed il TagBudgetReport.

Package it.unicam.cs.pa.jbudget100763.view

stabilisce i comportamenti ritenuti fondamentali per visualizzare i dati contenuti nel model ed occuparsi dell'interazione con gli utenti.

**View:** Interfaccia, ha la responsabilità di indicare le direttive principali riguardo l'interazione dell'utente con l'applicazione.

Package it.unicam.cs.pa.jbudget100763.view.javafx

include il percorso di implementazione della GUI tramite il framework JavaFX, seguendo le direttive della View.

**App:** Applicazione principale dell'implementazione tramite javaFX.

**FxController:** classe, gestisce le schermate da avviare e raccoglie tutte le interazioni dell'utente tramite la GUI, inoltrandole al controller dell'applicazione principale.

**AccountController:** classe, gestisce la finestra per la creazione dei conti.

**TagController:** classe, che gestisce la finestra per la creazione dei Tag.

Package it.unicam.cs.pa.jbudget100763.model

ha il compito di gestire i dati dell’applicazione, fornisce i metodi per accedere ad essi.

**Account:** interfaccia, è implementata dalle classi che hanno la responsabilità di gestire un conto.

**Budget:** interfaccia, ha la responsabilità di rappresentare e gestire un particolare budget, ovvero la previsione di spesa/guadagno per uno o più Tag

**Ledger:** interfaccia, è implementata dalle classi che hanno la responsabilità di gestire tutti i dati dell'applicazione.

**Movement:** interfaccia, è implementata dalle classi che hanno la responsabilità di gestire un singolo movimento.

**ScheduledTransaction:** interfaccia, indica una transazione o una serie di transazioni schedulate (previste) ad una certa data.

**Tag:** interfaccia, è implementata dalle classi che hanno la responsabilità di definire una categoria di spesa/guadagno.

**TagBudgetReport:** interfaccia, è implementata dalle classi che hanno la responsabilità di gestire un budget previsto per uno o più tag.

**Transaction:** interfaccia, è implementata dalle classi che hanno la responsabilità di gestire una transazione.

**AccountImpl:** classe, ha la responsabilità di gestire un conto.

**BudgetImpl:** classe, ogni budget associa ad ogni tag un importo che indica l'ammontare di spesa/guadagno previsto per il particolare tag.

**LedgerImpl:** classe, ha la responsabilità di gestire tutti i dati dell'applicazione.

**MovementImpl:** questa classe implementa la responsabilità di gestire un singolo movimento

**RegistryImpl<T>:** DISCLAIMER: Classe embrionale in via di sviluppo per la persistenza dei dati dell'applicazione.

**ScheduledTransactionImpl:** indica una transazione o una serie di transazioni schedulate (previste) ad una certa data.

**TagImpl:** ha la responsabilità di definire una categoria di spesa/guadagno.

**TagBudgetReportImpl:** ha la responsabilità di mostrare il saldo di positivo/negativo di uno o più Tag.

**TransactionImpl:** ha la responsabilità di gestire una transazione.

**AccountType:** enumerazione che rappresenta le tipologie di account

**MovementType:** enumerazione che rappresenta la direzione di un movimento su un conto.

