Focus

Barberini Elisa - Frattarola Marco
Mastrilli Alice - Puce Edoardo - Siroli Alex

24 aprile 2021

Indice

Capitolo 1

Analisi

1.1 Requisiti

Focus ha come obbiettivo quello di mettere insieme alcune funzioni utili alla gestione della vita quotidiana, in particolare gestendo l'aspetto economico e gli impegni giornalieri. L'idea é quella di coniugare le funzionalità di un'agenda elettronica con quelle della classica gestione finanziaria di conti.

Requisiti funzionali

- La sezione delle finanze si occupa di creare conti su cui è possibile effettuare transazioni in entrata e in uscita, a seconda della categoria ed altri parametri.
- La sezione del calendario permette di annotare e visualizzare i propri impegni.
- La sezione del diario offre un To Do List, un blocco note e la possibilità di salvare l'umore giornaliero.
- L'applicazione dispone, inoltre, di un timer e cronometro per registrare sessioni di una determinata attività .
- Focus deve inoltre poter mostrare le statistiche relative agli eventi e alle transazioni.
- L'applicazione permette anche di registrare delle transazioni di gruppo e di gestire i relativi debiti/crediti.

Requisiti non funzionali

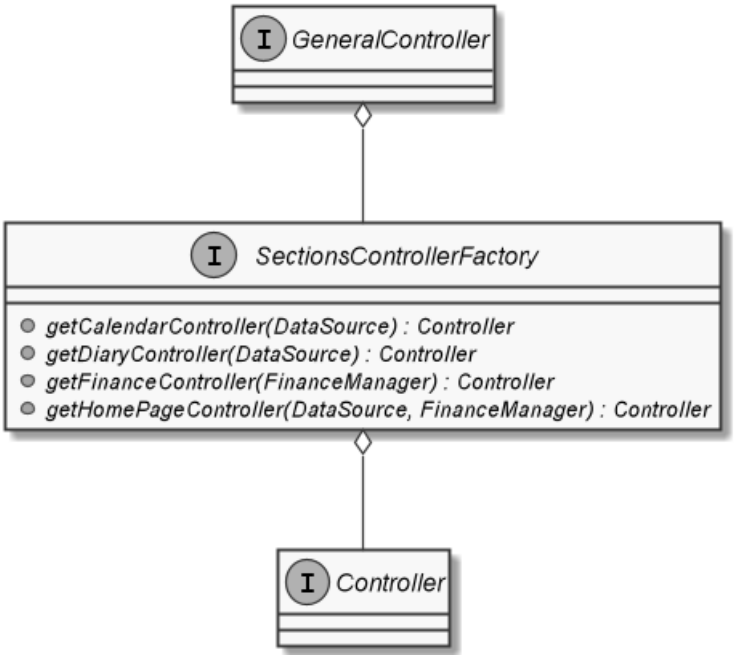
- L'applicazione dispone di un DataBase per il salvataggio dei dati.
- Utilizzo di un algoritmo ottimale per il calcolo della soluzione dei debiti in un gruppo.
- Organizzazione di una grafica intuitiva, che permetta il resize dei suoi elementi e che si aggiorni automaticamente alla modifica dei dati.

1.2 Analisi e modello del dominio

Il software dovrà essere in grado di modellare dati che riguardano le sezioni di calendario, finanze e diario.

- Per il contesto delle finanze verranno gestiti dei conti su cui sarà possibile eseguire delle transazioni ognuna delle quali farà riferimento ad una categoria. Sarà possibile anche gestire degli abbonamenti fornendo una ripetizione ad una transazione. Verranno gestite anche delle transazioni relative ad un gruppo di persone.
- Per il calendario verrà data la possibilità di salvare eventi con una possibile frequenza (giornaliero, settimanale, ...). Sarà possibile anche assegnare con quali persone viene svolto un determinato evento.

- Per il diario verrà data la possibilità di annotare le proprie idee come fosse una pagina di diario. Ci sarà anche la possibilità di inserire una lista delle cose da fare o effettuare delle sessioni di una determinata attività.
- Ci sarà la possibilità di memorizzare delle transazioni rapide o eventi rapidi per facilitarne l'inserimento.



Capitolo 2

Design

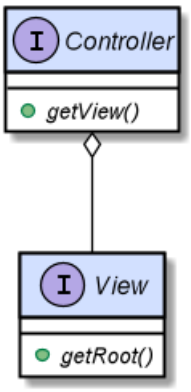
2.1 Architettura

Il pattern architetturale utilizzato dall'applicativo è MVC(Model-View-Controller), al fine di rendere le varie parti indipendenti tra loro e più semplice un eventuale modifica futura.

Il Model si occupa della parte di modellazione dei dati dell'applicazione, definendo le caratteristiche degli oggetti principali(eventi, giorni, transazioni,...). Questa sezione è indipendente da Controller e View e non ci sono riferimenti ad esse.

Il Controller si occupa di mettere in comunicazione la view con il modello, manipolando le informazioni dell'una per fornirla all'altra. Nello specifico è stato utilizzato un'interfaccia Controller che fa riferimento ad un'unica View. Questa scelta permette ad un Controller più complesso di delegare le sue varie responsabilità ad altri Controllers più specifici e, sfruttando l'interfaccia del Controller, di organizzare, in un'unica View, le sotto-view a cui essi fanno riferimento. In questo modo è possibile transitare da una libreria grafica ad un'altra solamente modificando l'interfaccia della View e delle sue implementazioni. Allo stesso modo un eventuale cambiamento del Controller non richiederà una sostanziale modifica della View.

La View rappresenta l'insieme delle schermate che verranno visualizzate. Esse offrono anche la possibilità all'utente di interagire, affidando ai controllers la gestione delle azioni da compiere in seguito ad un'interazione. Alcune View fanno riferimento a delle componenti del model ma solamente per leggerne lo stato e fornirne una rappresentazione grafica, e non per eventuali modifiche del loro stato. Questa parte è stata realizzata cercando di renderla il più possibile indipendente da Model e Controller, così che delle eventuali modifiche non intaccherebbero l'intera applicazione.



2.2 Design dettagliato

Elisa Barberini

In questa sezione andrò a porre particolare attenzione all'implementazione della parte relativa alla gestione degli eventi, delle persone e delle parentele, dell'homepage relativa al calendario e la parte del calendario riguardante la settimana.

Gestione eventi

Un evento è rappresentato da un'interfaccia Event implementata dalla classe EventImpl, mentre a livello applicativo è contemplato che un evento si ripeta oppure no.

La ripetizione di un evento può essere stoppata in qualsiasi momento utilizzando il campo isRepetead, di tipo booleano, che nel caso di un evento ripetuto verrà settato a true, in caso contrario a false.

Se si volesse stoppare la ripetizione in una data x, dovrà essere chiamato il metodo stopRepeat() cosicchè verranno salvati tutti gli eventi generati fino a tale data, in base al tipo di ripetizione, e successivamente il campo isRepeated verrà posto a false.

Vi è una distinzione tra gli eventi, quelli con durata superiore alle 24 ore, verranno etichettati come giornalieri, mentre quelli con durata inferiore come eventi normali.

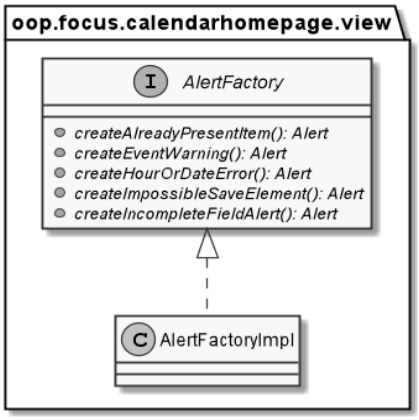
Per la gestione degli eventi ho implementato un gestore tramite l'interfaccia EventManager che viene implementata dalla classe EventManagerImpl.

È importante sottolineare che anche i tasti rapidi vengono salvati come eventi, con data e ora di inizio corrispondenti a data e ora della fine per cui ho provveduto a creare nel manager anche una sezione per la gestione dei suddetti dati ed il loro salvataggio.

Ogni qualvolta si voglia inserire un evento, nel caso di errori o sovrapposizioni si aprirà una finestra con un messaggio in grado di segnalare anche il tipo di errore commesso.

La finestra in cui viene segnalato l'errore, anche nel caso dell'eliminazione, sarà di tipo Alert e verrà generata tramite una factory, ovvero l'AlertFactory, interfaccia implementata dalla classe AlertFactoryImpl.

ALERT_FACTORY

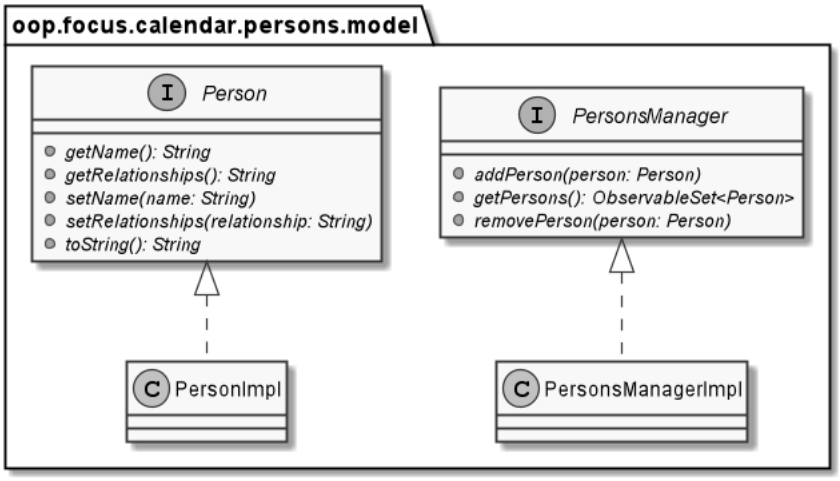


Il controller garantisce, la corretta comunicazione tra model e view, permettendo il corretto aggiornamento della suddetta in base a ciò che è memorizzato effettivamente nel database. I controller utilizzati estendono l'interfaccia Controller(/common/Controller).

Gestione persona

Una persona è rappresentata da un nome e una parentela, mentre la sua gestione è effettuata dal PersonsManager.

PERSONS



La finestra dedicata alla visualizzazione delle persone salvate è caricata in formato .FXML e contiene una tabella con tutti i loro dati.

La parte del controller si occupa di creare una perfetta collaborazione tra PersonsManager e PersonsView, in modo tale da permettere il corretto aggiornamento delle varie finestre in base a ciò che viene memorizzato nel database.

Il controller è stato realizzato estendendo l'interfaccia Controller(/common/Controller).

La parentela di una persona, viene selezionata tra quelle già salvate che possono essere addizionate utilizzando l'apposita finestra.

Homepage del calendario

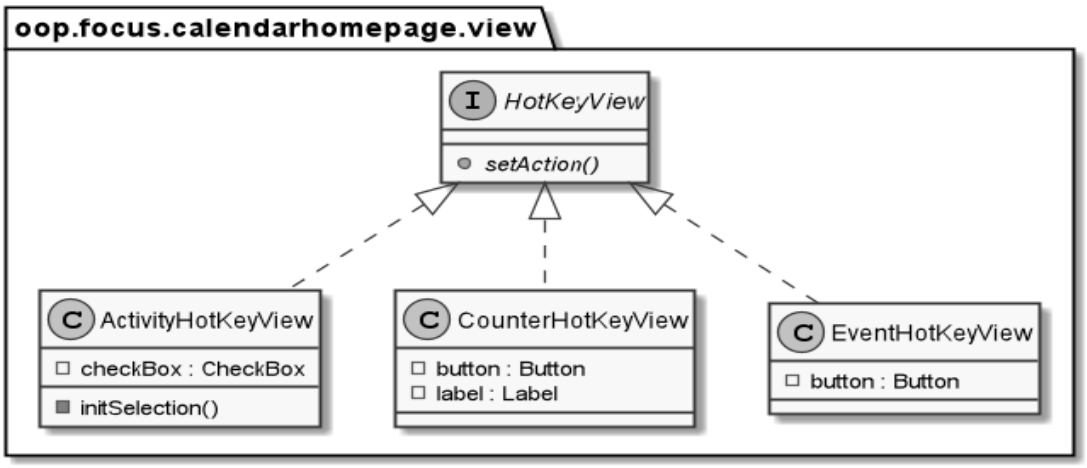
L'homepage del calendario presenta tre principali componenti, ovvero: il calendario in forma ridotta per la visualizzazione delle giornate, la giornata odierna con gli eventi in programma e la visualizzazione dei vari tasti rapidi.

Un tasto rapido è rappresentato da un nome e una categoria., quest'ultima è implementata tramite una enumerazione contenente tre differenti tipologie.

Le tre tipologie sono: contatore, attività ed evento.

Gli hot key sono rappresentati da tre differenti view che implementano l'interfaccia HotKeyView, la quale possiede un metodo per settare l'azione corrispondente al click di un determinato tasto.

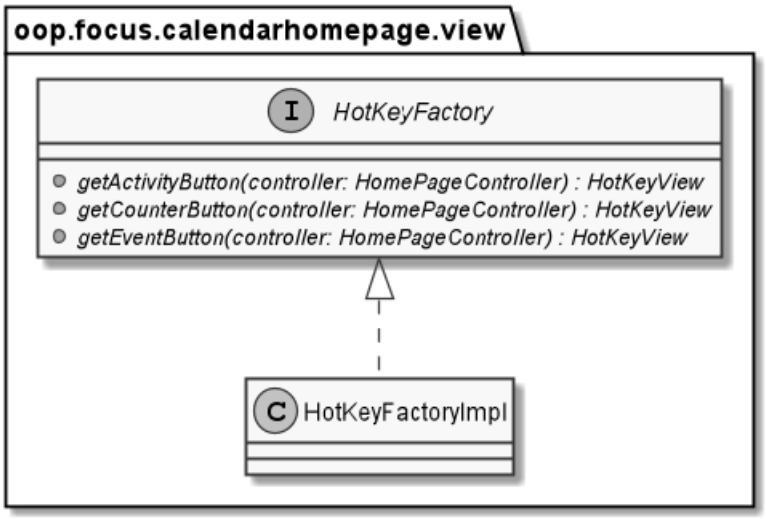
HOTKEYVIEW



I tasti rapidi, in base alla tipologia di appartenenza, vengono generati utlizzando il pattern factory.

L'entità HotKeyFactory presenta tre differenti metodi responsabili della creazione di altrettanti tasti.

HOT_KEY_FACTORY



La componente Controller permette la comunicazione tra Model e View ed ha il compito di manipolare le informazioni fornite dal Model e comunicarle alla View, così che questa venga riaggiornata.

Settimana

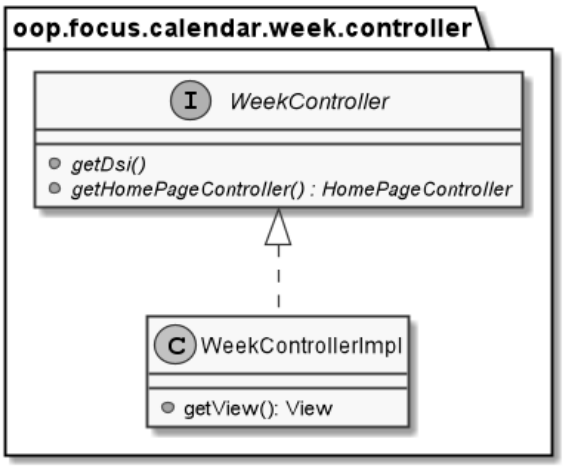
Per la creazione della sottosezione "settimana" del calendario, ho realizzato una finestra dinamica, in cui è possibile scorrere tra le varie settimane.

Ogni giornata permette di visualizzare gli eventi in programma tra cui quelli giornalieri.

La schermata principale viene caricata tramite un file in formato .FXML(/layout/...).

Ovviamente la settimana si aggiorna ogni qualvolta viene aggiunto o eliminato un evento.

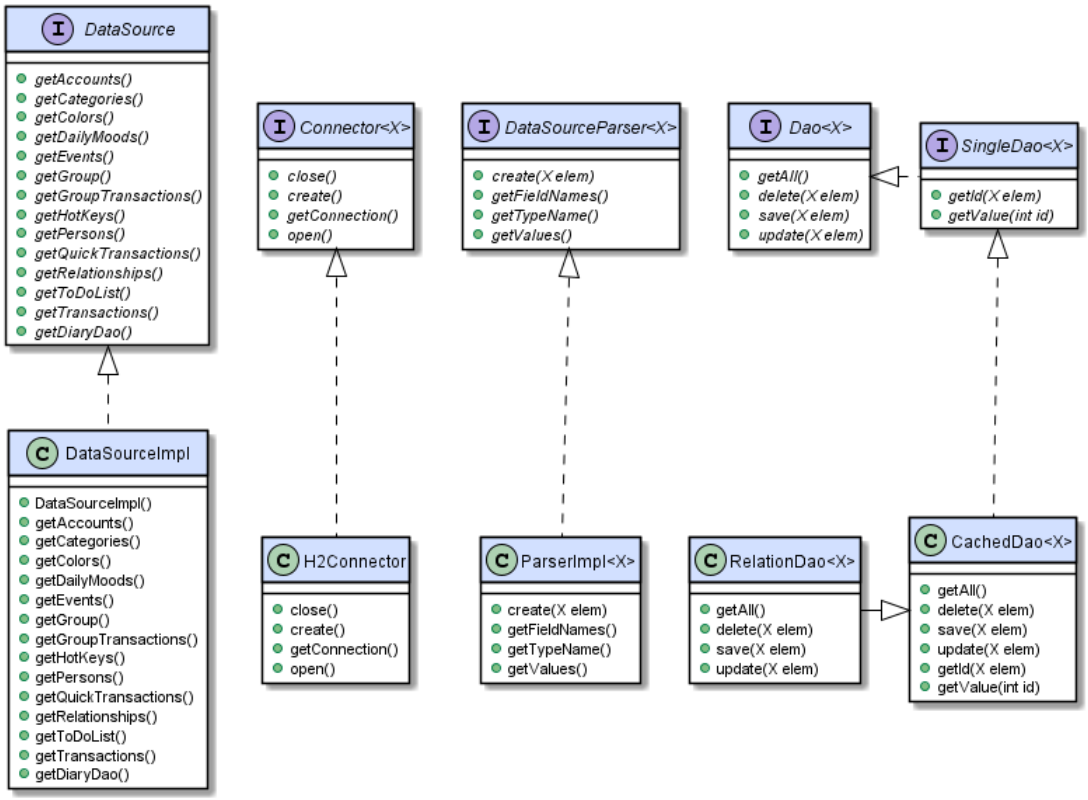
WEEK



Marco Frattarola

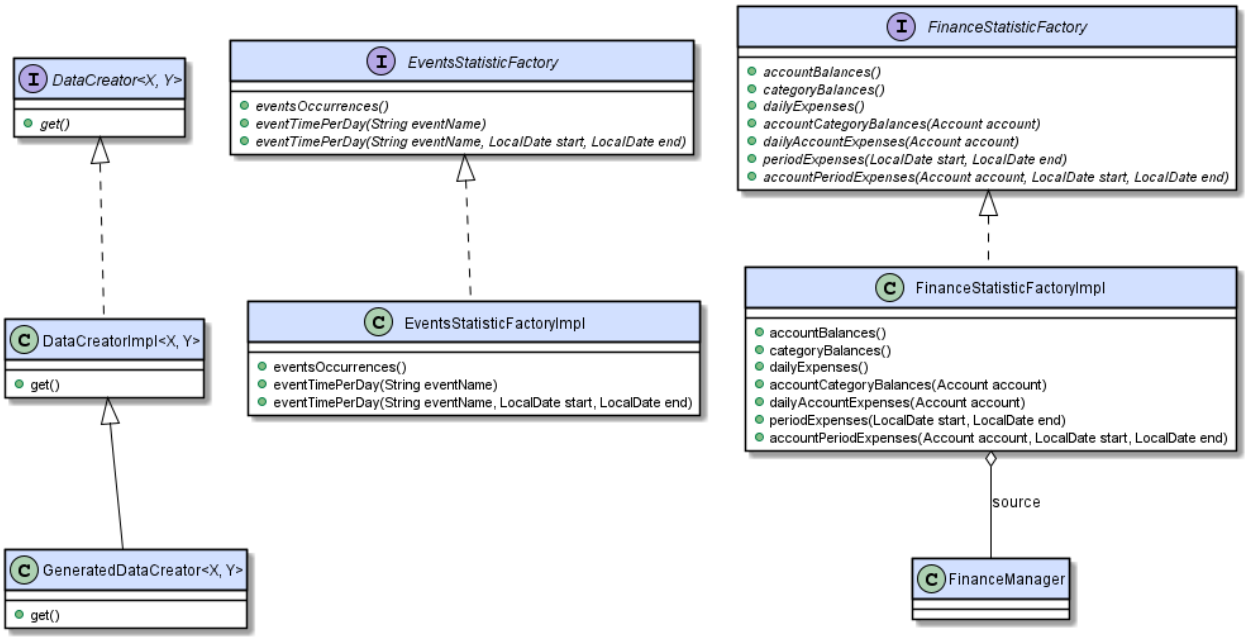
Persistenza dei dati

Per consentire le operazioni base di CRUD (create, read, update, delete) su una base di dati all'interno del software, é stato utilizzato il pattern DAO (Data Access Object), in modo da isolare la logica dell'applicazione dalle meccaniche di persistenza dei dati e fornire un API per interagire con essi. É stata quindi creata un interfaccia generica Dao per poter permettere implementazioni su tipi diversi di dato mantenendo invariate le operazioni eseguibili. Per evitare un forte accoppiamento tra classi della View e del Model é stato utilizzato il pattern Observer, permettendo ai client di essere observer sui dati a cui il dao fa riferimento (ad esempio l'aggiunta di un evento). Il dominio dei dati di cui viene effettuata le persistenza é stato definito attraverso l'interfaccia DataSource che rappresenta una sorgente dati astratta da cui é possibile recuperare interfacce di accesso a diversi tipi di dato (Eventi, Transazioni, ...). É stata utilizzata un'interfaccia generica Connector che permette di astrarre dallo specifico tipo di file su cui verranno salvati i dati e rendere più facile un eventuale transito ad un altro tipo di file. Essa permette infatti di aprire, chiudere o richiedere l'istanza della connessione con un qualsiasi tipo di dato su cui é possibile salvare o eliminare dati. Sono inoltre state utilizzate le eccezioni DaoException e ConnectionExcpetion per nascondere eventuali eccezioni, specifiche di una determinata implementazione, lanciate rispettivamente durante un operazione di lettura o scrittura, oppure durante una connessione effettuata tramite uno specifico Connector.

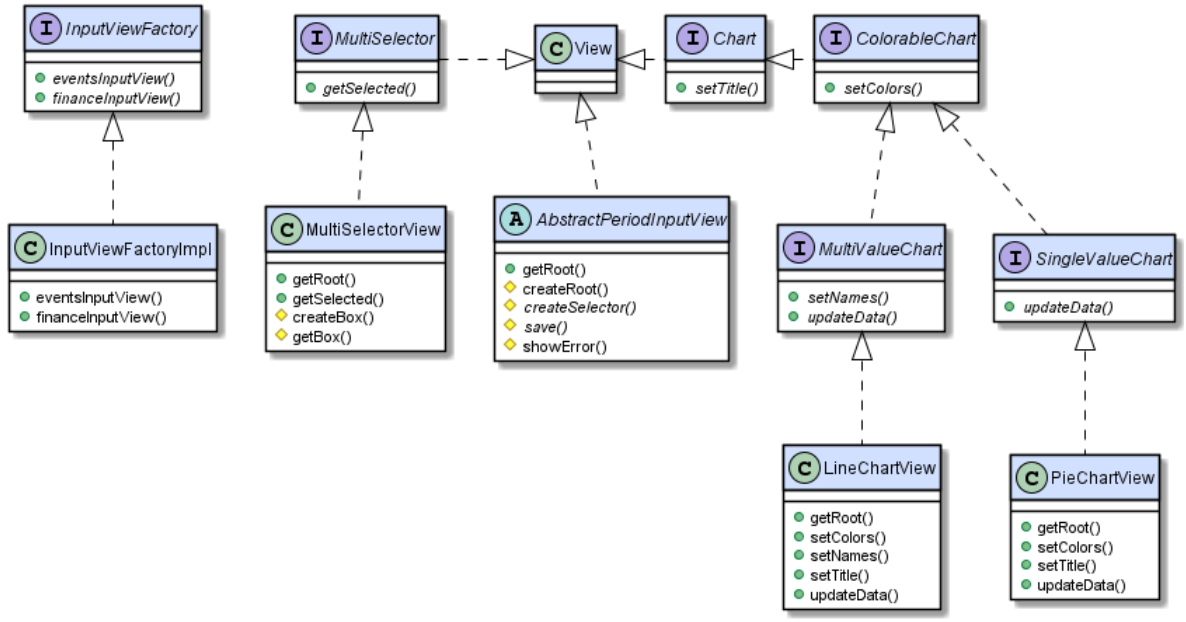


Statistiche

Una funzionalità del software é quella di calcolare e visualizzare statistiche riguardo i vari eventi del calendario e riguardo i dati relativi alla sezione delle finanze. Al fine di seguire il pattern architetturale MVC, questa sezione é stata divisa in tre sotto sezioni. La sezione relativa generazione dei dati si occupa della fase di calcolo delle statistiche. La creazione di un dataset viene eseguita attraverso classi che implementano l'interfaccia generica DataCreator, ovvero un oggetto in grado di mappare un'insieme di elementi di tipo A ad un insieme di tipo B (Ad esempio da eventi alla loro durata). Al fine costruire molteplici tipi di DataCreator con differenti logiche di creazione é stato utilizzato il pattern Factory method. Per differenziare i DataCreator relativi alla sezione degli eventi e delle finanze sono state utilizzate due factory diverse. In questo modo è possibile aggiungere eventuali nuove logiche di creazione più facilmente aggiungendo nuovi metodi nelle rispettive factory.



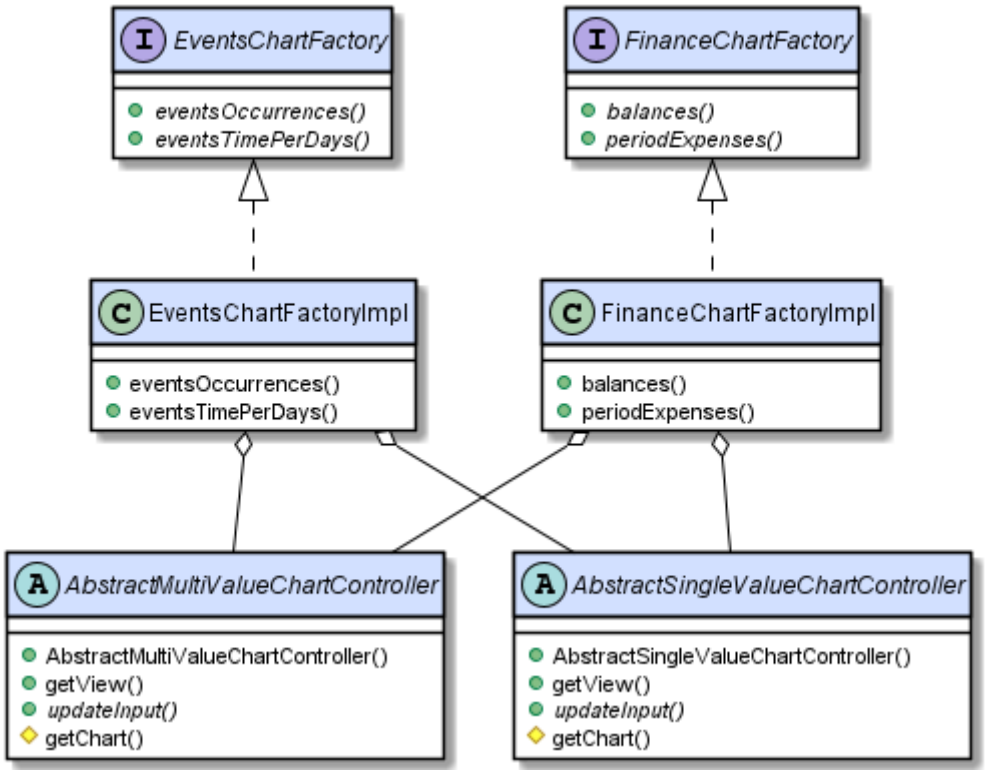
Per quanto riguarda la visualizzazione delle statistiche è stata utilizzata un interfaccia Chart come estensione di View che modella un grafico banale senza particolari funzionalità. Successivamente sono state definite due interfacce che estendono il concetto di Chart per rappresentare grafici che necessitano di un solo dato per ogni elemento (ideogrammi, grafici a torta, ...) e grafici che necessitano due dati per ogni elemento (diagrammi cartesiani, grafici a barre, ...). Entrambe le interfacce permettono ai client di aggiornare lo stato del grafico in modo dinamico attraverso il metodo updateData e nascondono interamente l'implementazione dei grafici e il modo in cui essi verranno visualizzati.



Per mettere il relazione i dati generati dal modello e la loro rappresentazione grafica sono stati introdotti una serie di controllers che si interpongono tra le due sezioni. Nonostante la loro similitudine al livello implementativo le sezioni di calendario e statistiche, indipendenti tra loro, sono state separate in due diversi Controller, per rendere più flessibili eventuali modifiche future. Ogni sezione si compone di un Controller che si occupa di permettere all'utente di inserire dati in input per interagire con i grafici e di una sezione che si occupa di comporre più grafici all'interno di un unica View. Per permettere al controller dell'input di comunicare un update al controller relativo ai grafici è stata utilizzata un interfaccia generica UpdatableController, che rappresenta un controller che offre la possibilità di essere aggiornato mediante un metodo. É risultato utile rappresentare l'input inserito dall'utente come un unico tipo di dato, utilizzando un'interfaccia

generica, TimePeriodInput, che verrà poi utilizzata all'interno dei controller e delle View relative alla gestione dell'input utilizzando il pattern Builder.

Infine é stato utilizzato il pattern Factory method per la creazione dei diversi controller dell'input relativi alla sezione degli eventi e alla sezione delle finanze che. La factory al suo interno fa uso di una classe astratta che segue il pattern Template Method. Il metodo save() influenza il comportamento che verrá assunto dalla view quando l'utente inserirà dei nuovi dati. Diverse implementazioni del metodo potrebbero ad esempio effettuare controlli diversi sull'input prima di notificare il controller. Lo stesso meccanismo é stato utilizzato per la creazione dei controller relativi ai grafici. Nel dettaglio sono state utilizzate due classi astratte, che fanno riferimento ai due tipi di chart diversi presenti (SingleValueChart e MultiValueChart). Il metodo astratto updateInput() determina il tipo di dati che verranno consegnati alla View e visualizzati nel grafico all'arrivo di un nuovo input. Sono state poi create due factory (distinguendo le sezioni Calendario e finanze) che permettono di creare diversi controller relativi ai grafici implementando in modo diverso i metodi astratti. Questi ultimi sono coloro che assemblano i dati prodotti dai generatori di dati del model (DataCreator) e le loro rappresentazioni grafiche, attraverso le precedenti factory di grafici.



Eseguendo l'override dei metodi di supporto delle due classi astratte é inoltre possibile cambiare il tipo di implementazione del grafico che verrá visualizzato (Ad esempio da grafico a torta a ideogramma). Grazie all'utilizzo di questi pattern, é possibile un eventuale futura modifica o aggiunta dei grafici disponibili, solamente utilizzando diversi DataCreator, diverse Implementazioni grafiche dei grafici e in seguito assemblando entrambi i componenti in un apposito controller, servendosi delle factory disponibili per i tre elementi. Ogni sezione é in questo modo indipendente dalle altre.

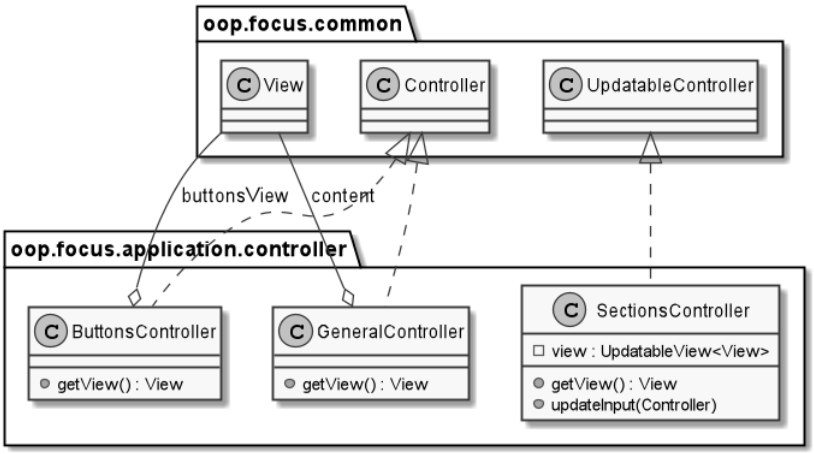
Alice Mastrilli

La parte che ho svolto riguarda l'implementazione di Controller e View dell'applicazione e la sezione diario.

Applicazione

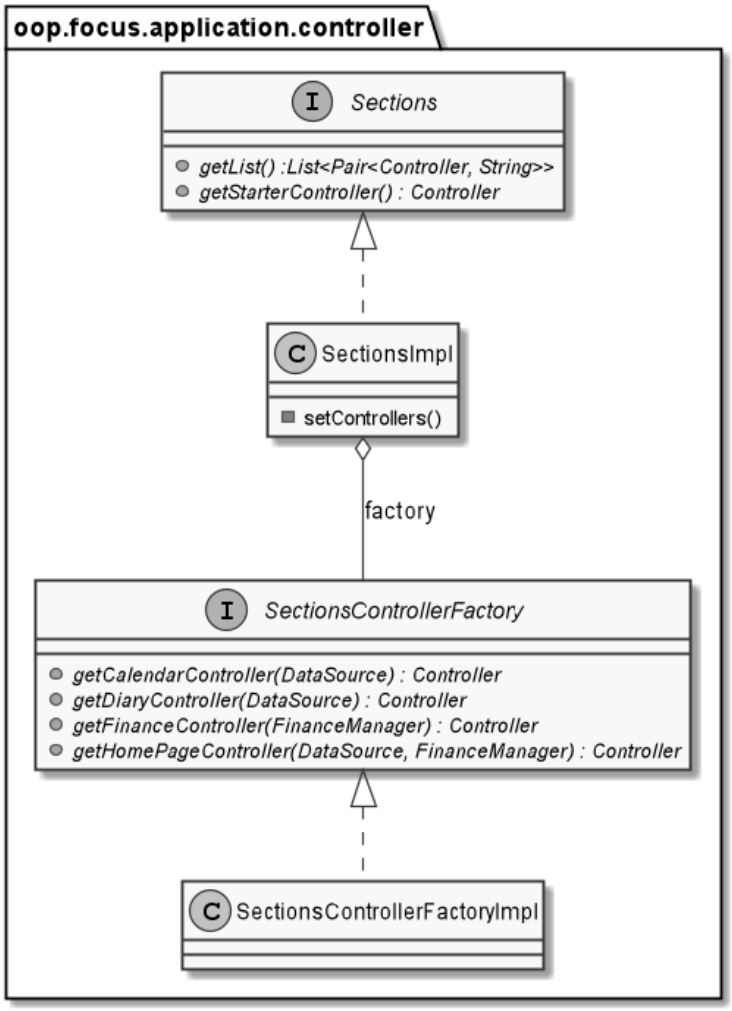
Per la gestione dell'applicazione ho utilizzato due Controller : ButtonsController e SectionsController, coordinati dal GeneralController, il quale ne unisce le View.

APPLICATION



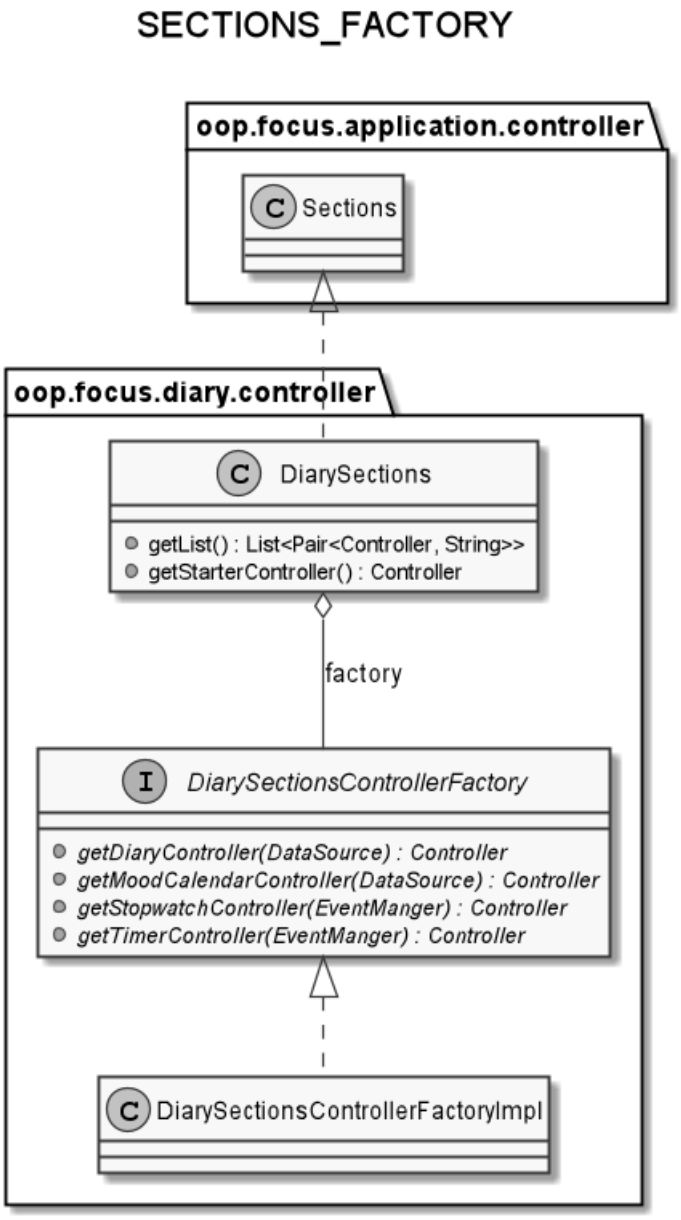
Per la creazione dei Controller delle quattro sezioni principali(quindi la relativa View da mostrare premendo un bottone), è stato utilizzato il pattern Factory Method (la cui interfaccia è SectionsControllerFactory).

FACTORY_APP



Diario

Lo schema di questa sezione è analogo a quello dell'applicazione : SectionsController e ButtonsDiaryController sono coordinati da GeneralDiaryController e la DiarySectionsControllerFactory è la classe che utilizza il pattern factory method per la creazione dei Controller delle quattro sotto-sezioni di diario.

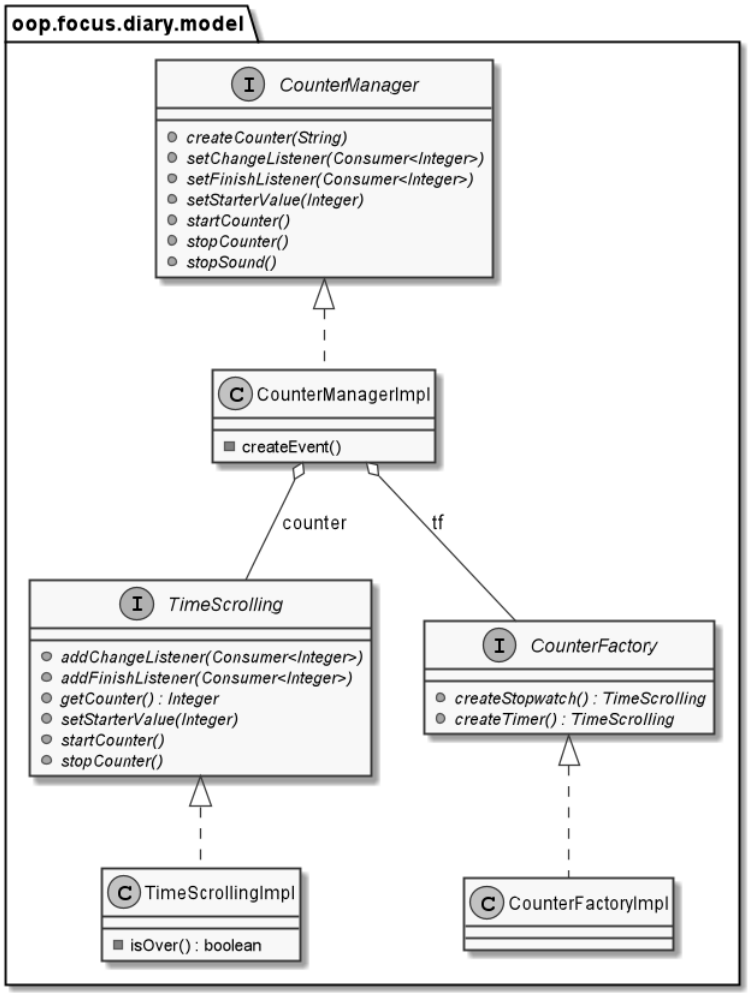


Per il caricamento delle icone nella sezione del mood giornaliero, ho utilizzato due View : dailyMoodSection e dailyMoodImpl. DailyMoodSection rappresenta l'intera sezione del DailyMood e inizializza una griglia di bottoni: l'emoji rappresentata su ognuno di questi è data da DailyMoodView, che ritorna la rappresentazione grafica di una singola icona. La scelta di avere queste due classi permette di astrarre la rappresentazione grafica di un singolo daily mood come View, nascondendo, a chi lo usa, la sua implementazione. Inoltre, qualora si volesse modificare la grafica del bottone, non sarà difficile: bisognerà ritoccare soltanto dailyMoodSection mentre DailyMoodImpl rimarrà invariata.

Contatori

Per le sezioni dei contatori, rilevante è la classe TimeScrollingImpl(che implementa TimeScrolling) si occupa dello "scorrere del tempo" vero e proprio: prende in input una funzione, che stabilisce come modificare il valore del contatore ad ogni secondo, ed un predicato, che permette di stabilire la durata del contatore stesso. In questo modo, nel caso del timer, la funzione, ad ogni secondo,

diminuirà il valore di 1, nel caso del cronometro lo aumenterà di uno. Seguendo questa logica, non è un problema realizzare un contatore che modifichi diversamente il valore ad ogni secondo, ad esempio in modo quadratico. Per la creazione dei diversi tipi di contatori ho utilizzato una factory(CounterFactory), che inizializza i vari counter passandogli gli appositi parametri. Per "monitorare" la fine del timer o il cambiamento del valore ad ogni secondo, è stato utilizzato il pattern Observer. Tramite la notifica della fine del timer è stato molto semplice impostare il suono del timer(che si avvia quando il timer è a zero) e, nella View, la notifica di cambio del valore ha permesso di aggiornare molto semplicemente il display del contatore.



Edoardo Puce

In questa sezione l'attenzione è focalizzata sull'implementazione della sezione del calendario, il mese, il giorno e la loro logica. Al fine di seguire il pattern architetturale MVC, le varie classi sono state suddivise in tre sotto sezioni (Model, View e Controller).

Day - CalendarLogic

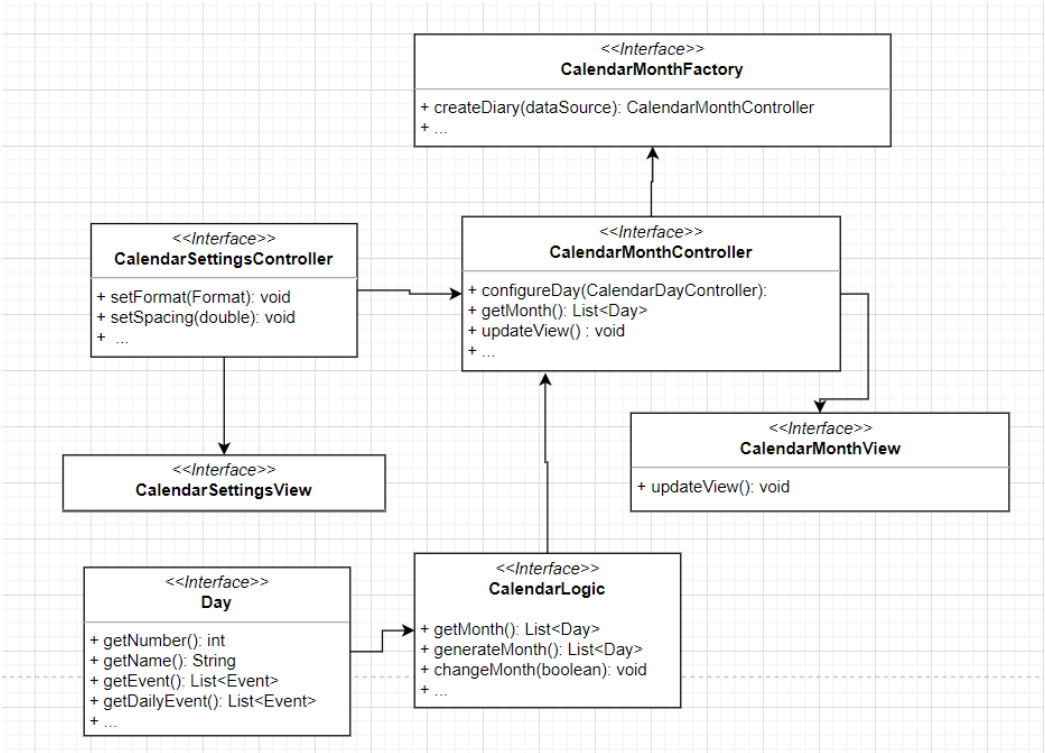
la classe DayImpl (implementazione dell'interfaccia Day) è l'oggetto alla base del calendario, il quale oltre a contenere i metodi base per la descrizione di un giorno (es. giorno, mese, anno...) ha presente metodi (getEvent(), getDailyEvent()) per poter ottenere gli eventi di quella giornata.
la classe CalendarLogicImpl (implementazione dell'interfaccia CalendarLogic) è la classe che usando l'oggetto Day crea delle liste che corrispondono alla settimana, il mese e l'anno (per esempio generateMonth()). Possiede anche dei metodi per poter cambiare il periodo delle liste, per esempio changeMonth(true) permette di passare dal mese corrente a quello precedente, invece changeMonth(false) permette il passaggio al mese successivo.

Month

Per costruire i vari tipi di mesi, è stato utilizzato il pattern Factory method per la creazione dei diversi controller che generano una View diversa secondo il metodo chiamato (es. CalendarMonthFactory.createDiary() crea il mese del calendario dedicato al diario).
CalendarMonthController è l'interfaccia della classe che collega il model con la view, i metodi principali passano i dati alla CalendarMonthView, come getMonth() che passa una lista dei giorni del mese o come il metodo configureDay() che invece permette di cambiare la spaziatura o il formato dei CalendarDayView presenti nella CalendarMonthView.
CalendarMonthView è l'interfaccia della classe che permette la generazione dell'interfaccia utente del mese, tre tipi di calendario, quello normale (con pulsanti che permettono di aver maggiori informazioni sul giorno cliccato), quello dell'homepage (con label che fanno vedere i giorni del mese e se sono presenti eventi, quet'ultime vengono colorate) e quello del diario (che ad ogni giorno fa vedere un'icona che rappresenta l'umore della giornata), un altro metodo importante è updateView() che viene chiamato dal Controller per potersi aggiornare.

Settings

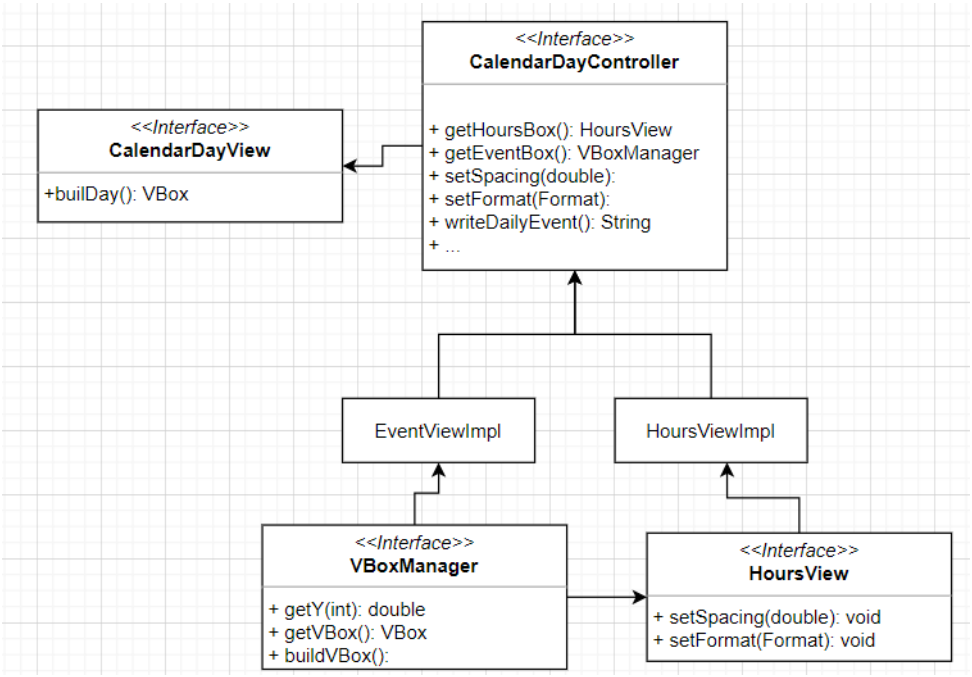
la classe CalendarSettingsController è l'interfaccia della classe che passa al CalendarMonthController la spaziatura o il formato se modificati dall'utente attraverso l'interfaccia CalendarSettingsView.



Day

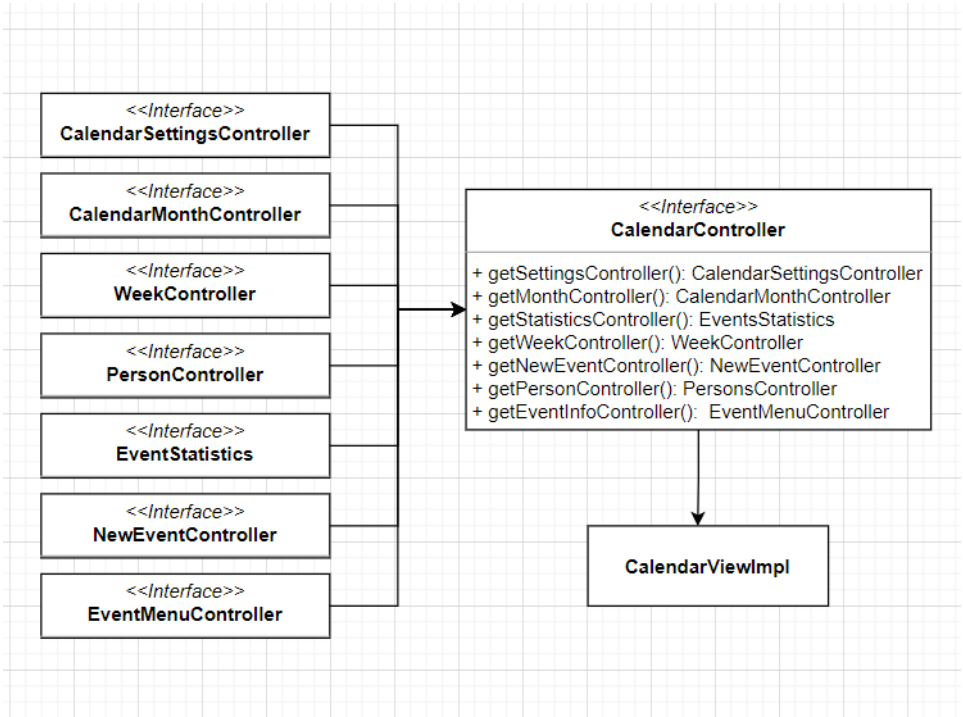
CalendarDayController è l'interfaccia della classe che si occupa di passare e/o modificare i dati alla CalendarDayView. Piu precisamente si occupa di passare HoursViewImpl (box delle ore) e EventViewImpl (box degli eventi) alla CalendarDayView. Vengono passate anche la spaziatura (spazio che c'è tra un ora e un altra) e il formato (visione degli eventi nella giornata divisi in ora per ora oppure mezz'ora per mezz'ora) delle ore.
La classe CalendarDayViewImpl (implementazione dell'interfaccia CalendarDayView) definisce il giorno grafico, composto da tre sezioni (partendo dall'alto): il box con il nome del giorno (es. domenica) e il numero del giorno (es. 18); il secondo box contiene gli eventi considerati giornalieri e il terzo contiene gli eventi della giornata disposti a seconda dell'orario. Il terzo box è stato creato

tramite l'implementazione delle classi HoursViewImpl e EventViewImpl, passategli dal CalendarDayController, che si occupano di allineare gli eventi agli orari della giornata.



Calendar

la classe CalendarController è l'interfaccia della classe che permette di passare i controller dei componenti che formano la sezione del calendario alla CalendarView.
la classe CalendarView è l'interfaccia della classe che permette la generazione della GUI della pagina generale del calendario,la classe semplicemente permette di visualizzare la pagina richiesta cliccando sull'apposito bottone (esempio: cliccando su "Mese" fara vedere il Mese del calendario)

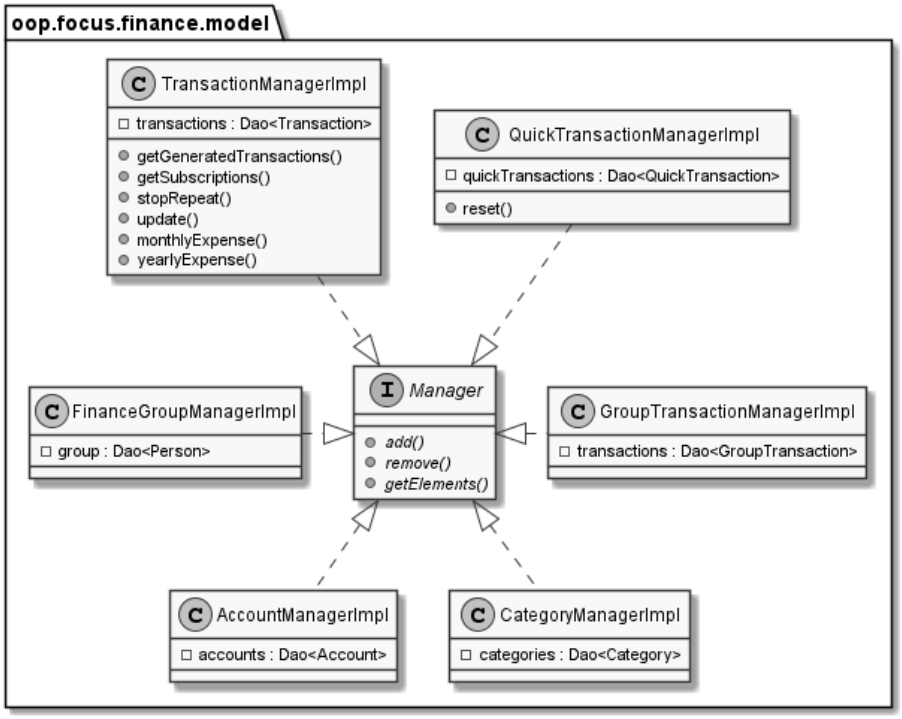


Alex Siroli

In questa sezione l'attenzione è focalizzata sulla gestione dell'area relativa alla finanza.

Model

Questa area mi sono occupato di tutto ciò che riguarda la logica relativa alla finanza, quindi gestire conti, transazioni, categorie, transazioni di gruppo, le persone presenti nel gruppo e le transazioni rapide. Per la gestione di tutti gli elementi citati ci si avvale dell'utilizzo di diversi manager, che implementano l'interfaccia generica Manager. Questi manager si occupano anche del salvataggio di questi elementi nel database.



Per la gestione di tutti i manager, tramite la composizione, è stato creato anche il FinanceManager che compone tutti i manager sopracitati ed effettua le operazioni che riguardano più manager contemporaneamente.

Controller

La sezione controller di finanza fornisce tutti i metodi necessari per raggiungere qualsiasi tipo di dato che potrebbe richiedere la view. Per permettere un cambiamento della view in blocco in maniera semplice non è presente alcun elemento di una qualsiasi libreria grafica nelle implementazioni dei controller.

View

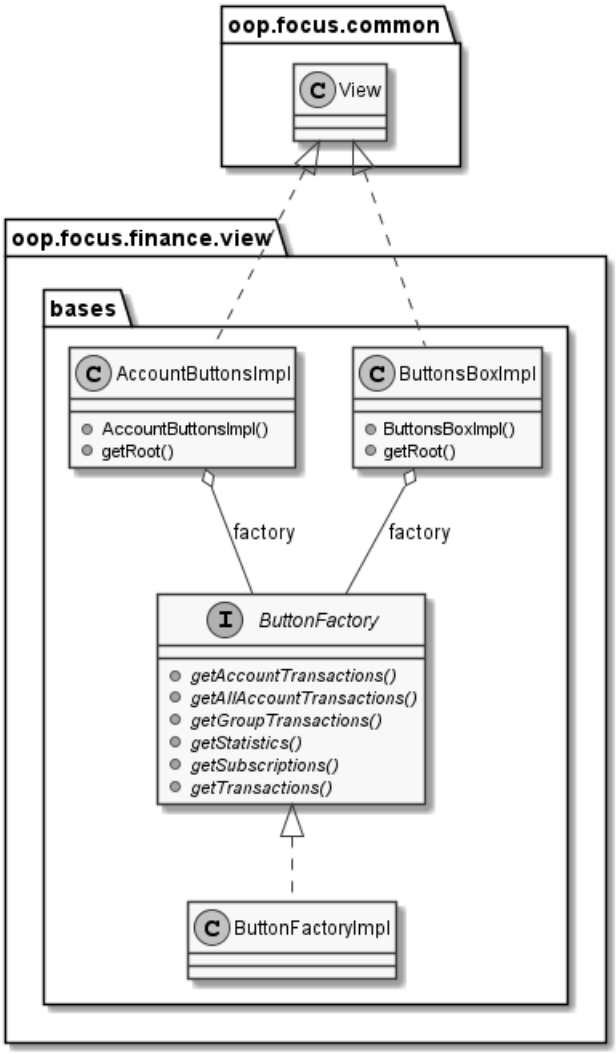
La sezione della view è stata sviluppata in modo che la visualizzazione sia semplice ed intuitiva, ma non banale. Per questo motivo sono stati creati diversi file FXML che racchiudessero tutte le parti statiche della grafica, mentre da codice ci si occupa di tutte le parti dinamiche. Le view sono divise tra:

- bases, cioè tutte le schermate principali che mostrano le informazioni di maggiore importanzai;
- tiles, per quanto riguarda la visualizzazione del singolo oggetto (una transazione, un conto o altro);
- windows, cioè tutte le schermate visualizzate a pop-up (aggiunta di un elemento o altro).

Di seguito vengono descritte alcune parti della view degne di nota.

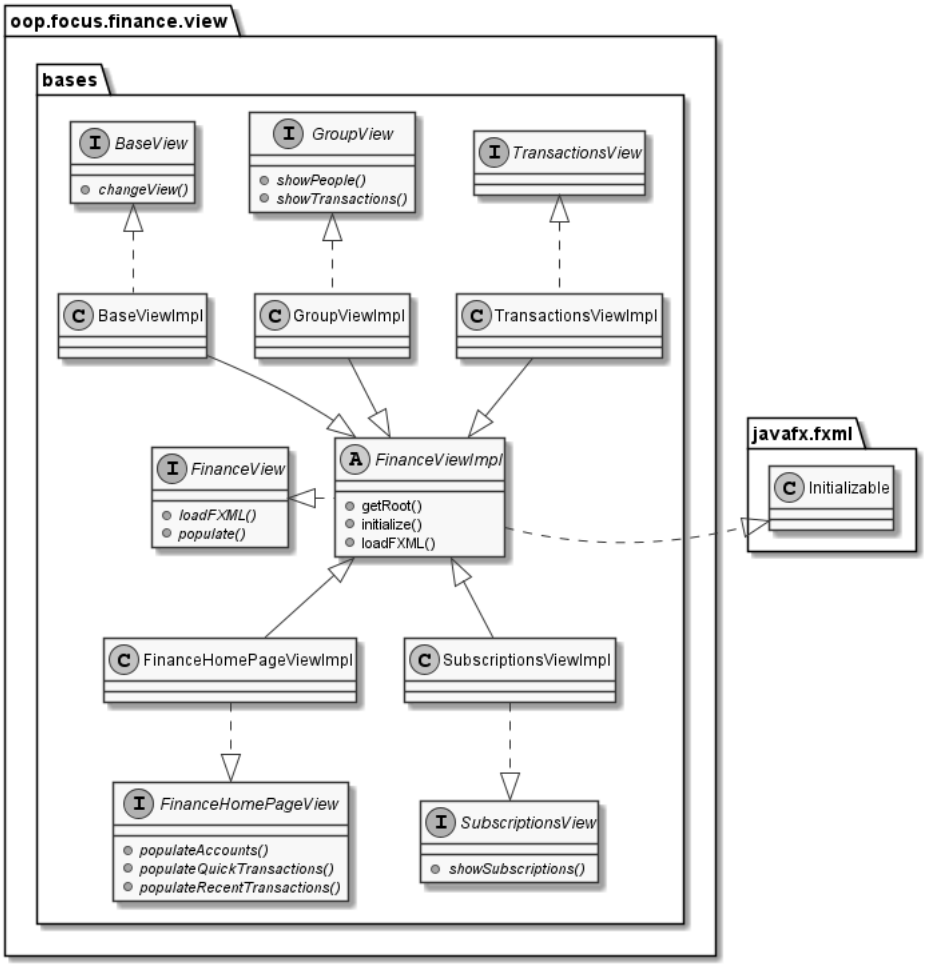
Factory di button

Per quanto riguarda la creazione di pulsanti è stata implementata una factory (ButtonFactory). La factory crea sia i button di navigazione tra le aree di finanza sia i bottoni relativi alla navigazione fra conti. Grazie alla factory sarà più semplice, ad esempio, aggiungere una nuova area di finanza oltre a quelle già implementate.

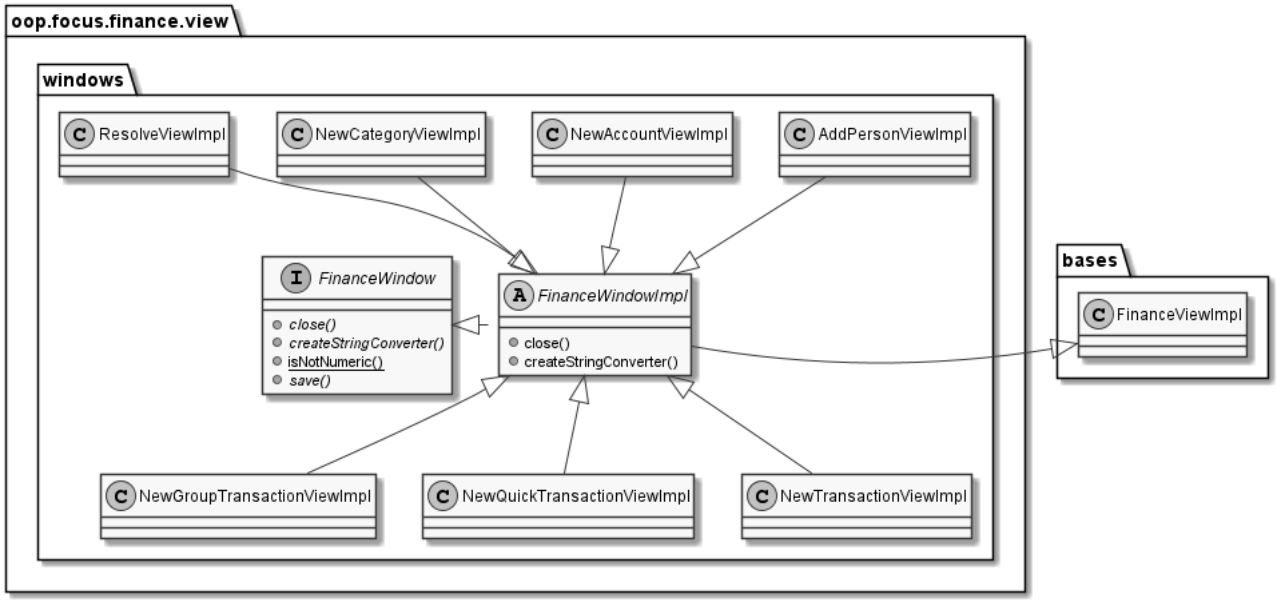


FinanceView e FinanceWindow

Dal momento in cui per tutte le schermate della view sono stati utilizzati file FXML, tutte le classi richiedevano una buona parte del codice in comune. Per questo motivo è stata implementata una classe astratta FinanceView, che implementa una sola volta i metodi in comune. In questa maniera le classi che utilizzano un file FXML possono estendere la classe astratta e implementare solo il metodo astratto relativo all'inizializzazione delle informazioni visualizzate(populate).



Per quanto riguarda le view relative a finestre (aggiunta di una transazione, visualizzazione dettagli di un abbonamento e altro) vi erano ulteriori parti di codice in comune. Per questo motivo è stata creata una nuova classe astratta FinanceWindow che estende FinanceView e integra le parti di codice comuni nelle finestre.



Questi accorgimenti hanno permesso di rendere il codice più semplice e meno ripetitivo.

Capitolo 3

Sviluppo

3.1 Testing automatizzato

Elisa Barberini

- EventTest: verifica la correttezza nell'aggiunta e nella rimozione dei vari eventi e la correttezza dell'operazione di ricerca di questi in base ad una specifica data.
- HotKeyTest: verifica la correttezza nell'aggiunta, nella rimozione, e nell'assegnamento della catgeoria dei vari tasti rapidi.
- PersonAndDegreeTest: verifica la correttezza nell'aggiunta delle parentele.
- TimePropertyTest: verifica la correttezza nella verifica del calcolo della durata di un evento, per classificarlo e per verificare che rispetti la minima durata che deve avere. Questo test è anche utilizzato per verificare che un evento non si sovrapponga ad altri.

Marco Frattarola

persistenza dei dati:

- Creazione e connessione al database
- É stato creato un test per ogni tipo di dato che l'applicazione aveva bisogno di memorizzare. In particolare sono state testate le funzionalità di lettura, scrittura, modifica e cancellazione.
- La possibilità di essere Observer di un Dao.

Per la sezione delle statistiche tutti i test automatizzati sono stati effettuati verificando che dati dei precisi valori di input, vengano creati dei dataset corretti per un eventuale visualizzazione in un grafico:

- dataset sulla la durata giornaliera per evento.
- dataset sul numero di occorrenze per evento.
- dataset sul saldo per ogni conto.
- dataset sul saldo per ogni categoria.
- dataset sui movimenti giornalieri per ogni conto.

Alice Mastrilli

- ComputeTotalTimeOfEventTest: verifica che il tempo totale calcolato per svolgere un evento sia corretto.
- DailyMoodManagerTest: verifica la correttezza nell'aggiunta, rimozione o modifica del mood giornaliero.
- DiaryTest: verifica la correttezza nell'aggiunta, rimozione o modifica delle pagine di diario.
- ToDoListManagerTest: verifica la correttezza nell'aggiunta, rimozione o modifica delle To Do Actions.

Edoardo Puce

- **CalendarDayTest:** test automatizzato per controllare che la classe generi correttamente l'oggetto Day, controlla che le informazioni e gli eventi siano del giorno generato.
- **CalendarLogicTest:** test automatizzato per controllare che la classe generi le liste (mese, settimana o anno) in modo corretto e che cambiando il periodo (esempio, mese successivo) venga generata la giusta lista di giorni.
- **CalendarAppTest:** test non del tutto automatizzato, avviare una demo della sezione Calendario tramite la classe Test CalendarAppTest. (CalendarPageViewTest è usato dall AppTest per creare una finestra con cui interagire)

Alex Siroli

- **FinanceTest:** test completamente automatizzato per controllare che tutte le parti di logica relative alla finanza siano funzionanti. In particolare si controlla il corretto funzionamento della gestione dei conti, delle categorie, delle transazioni, degli abbonamenti, delle transazioni rapide, delle persone presenti nel gruppo di finanza e le relative transazioni di gruppo.

3.2 Metodologia di lavoro

Per lo sviluppo del progetto, abbiamo cercato di mantenere la suddivisione stabilita dei ruoli nel momento della presentazione del progetto, assegnando ad ognuno compiti distinti, in modo da poter effettuare delle scelte di design in autonomia. Per coordinare il nostro lavoro abbiamo usato Git come DVCS. In particolare, abbiamo usato un unico branch principale, cercando di non modificare porzioni di codice di altri componenti del gruppo per evitare eventuali complicazioni. Le parti in comune sono state raccolte nel package common in modo da renderle piú accessibili a tutti.

- **Elisa Barberini:** sezione del calendario nell'homepage, intera gestione di eventi (aggiunta, eliminazione. informazioni), persone (aggiunta,cancellazione), visualizzazione settimana del calendario.
- **Marco Frattarola:** gestione persistenza dei dati, intera gestione delle statistiche relative agli eventi e alla sezione delle finanze. Alcune classi e interfacce di utilizzo comune.
- **Alice Mastrilli:** intera gestione delle sezioni del diario, ad eccezione delle statistiche umori. Intera gestione dell'applicazione.
- **Edoardo Puce:** grafica generale del calendario, logica e visualizzazione giornaliera e mensile del calendario.
- **Alex Siroli:** intera gestione delle sezione della finanza dell'homepage e intera gestione della finanza (transazioni, abbonamenti, conti, categorie, transazioni di gruppo e transazioni rapide) compresa di grafica generale.

3.3 Note di sviluppo

É stato fatto uso di Gradle come gestore di dipendenze per il progetto. Di seguito un resoconto più dettagliato:

Elisa Barberini

Nell'ambito del mio sviluppo di codice sottolineo :

- l'utilizzo di Stream e ForEach per rendere più leggibile il codice;
- l'uso di lambda expression;
- lo sviluppo di alcuni algoritmi non banali, come ad esempio quello per verificare, quando si vuole aggiungere un evento, che questo non si sovrapponga con quelli già salvati;
- l'uso dell'Optional;
- l'utilizzo della libreria JavaFx, e della libreria joda.org.time.

Marco Frattarola

- Costruzione di classi generiche per la rappresentazione di generatori di dati e per l'interfaccia Dao.
- Uso di lambda expressions per rednere il codice piú compatto e leggibile
- Uso di Stream per operare piú facilmente su collezzioni di dati.
- Utilizzo della libreria JavaFx per la realizzazione delle interfacce grafiche e per la visualizzazione dei grafici.
- Utilizzo di Joda time per una maggiore facilitá di utilizzo delle date rispetto alla libreria Time di java.util.Date.
- Alcune parti di codice, relative alla personalizzazione dell'aspetto grafico dei grafici sono state estratte e leggermente riadattate da una fonte online.
  - https://stackoverflow.com/questions/15219334/javafx-change-piechart-color
  - https://stackoverflow.com/questions/12114302/change-color-and-line-type-of-linechart-symbols

Edoardo Puce

- **Utilizzo della librearia JavaFX:** ha permesso di creare la GUI, quindi tutti quei compenenti adibiti all'interazione con l'utente
- **Utilizzo delle Lambda expression e Stream:** usate per gli EventHandler dei pulsanti e per lavorare con le liste, hanno aiutato il codice ad essere un po' piu pulito.
- **Utilizzo del pattern Factory:** usato per la creazione di diversi controller per i diversi tipi di mese (HomePage,Normal,Diary)
- **Utilizzo di JUnit4:** per la creazione di test automatizzati.
- **Utilizzo di Joda-Time:** un libreria per la gestione del tempo, utile nella creazione dell'oggetto Day e nella logica del calendario.

Alice Mastrilli

- **libreria JavaFX:** per la rappresentazione della grafica
- ogni pagina di diario è salvata come un file : a tale scopo ho utilzzaro la libreria base java.IO per la creazione di fileReader/fileWriter(di conseguenza buffer reader/writer).
- **classe generica:** (WindowRemoveAnnotation): mi ha permesso di utilizzare la stessa finestra per rimuovere note sia dal diario sia dal toDoList.
- **libreria Joda-Time:** utilizzata nell'ambito dei contatori.
- utilizzo di stream e lambda expression
- **utilizzo dei file .FXML:** utilizzati per la creazione delle parti del diario e di alcune finestre

Alex Siroli

- **libreria JavaFX** per la rappresentazione della grafica
- **uso di stream e lambda expressions** in molte parti del codice
- **utilizzo di interfacce/classi generiche**, come l'interfaccia Manager o la classe FinanceButton.
- **libreria joda-time:** utilizzata per la gestione delle date
- **utilizzo dei file .FXML:** utilizzati per la creazione delle parti statiche della grafica
- **Utilizzo di JUnit4:** per la creazione di test automatizzati.
- **Sviluppo di algoritmi interessanti,** come l'algoritmo per generare le transazioni ripetute automaticamente in base alla data odierna e l'algoritmo per calcolare le transazioni di gruppo minime per saldare tutti i debiti.

Capitolo 4

Commenti finali

4.1 Autovalutazione e lavori futuri

Elisa Barberini

Lo sviluppo di questo progetto è stato importante ed interessante per la mia persona, essendo stata la prima vera e propria applicazione che ho sviluppato.

Ho avuto la possibilità di mettermi alla prova, testando tutto ciò che è stato assimilato durante il corso, arricchendomi inoltre di nuove conoscenze, grazie alla ricerca e allo studio, ma anche all'apporto dato dai miei compagni di gruppo.

Il lavoro in team mi ha aiutata molto sia per riuscire ad arrivare a soluzioni ottimali risolvendo i problemi riscontrati, che per comprendere gli errori fatti.

Ho cercato di implementare gli Eventi, i tasti rapidi e la parte di homepage relativa al calendario nel modo migliore possibile, dando sempre il massimo di quello che potevo e cercando di trovare soluzioni adeguate ed efficienti.

Abbiamo sempre lavorato da casa, data la situazione, e devo dire che questo è stato inizialmente l'ostacolo più grande.

Siamo però comunque riusciti ad avere un'ottima collaborazione attraverso l'utilizzo di un gruppo Whatsapp dedicato e di Discord che ci ha permesso di effettuare numerose riunioni per poterci confrontare sul lavoro svolto.

Nonostante le numerose difficoltà riscontrate durante lo sviluppo della nostra applicazione, penso il nostro gruppo sia riuscito a consegnare un progetto valido e funzionante, che spero potrà essere rivisto in futuro per apportare eventuali migliorie.

Ringrazio ovviamente i miei compagni di team che sono stati sempre in grado di fare critiche costruttive utili a migliorare il mio codice, e per la collaborazione che ha permesso scambi di informazioni e punti di vista.

Marco Frattarola

Personalmente credo che la realizzazione di Focus sia stata un'esperienza molto formativa, sia per quanto riguarda la coordinazione con il gruppo, sia per la fase di sviluppo. Essendo stata la prima esperienza di gruppo quasi per tutti inizialmente non é stato facile capire come organizzarsi per un corretto utilizzo del DVCS. Sono piuttosto soddisfatto del mio lavoro svolto anche se non é stato facile gestire le ore di lavoro. Sono consapevole che le metodologie utilizzate per il salvataggio dei dati non siano state delle migliori e che la relativa sezione di codice potrebbe risultare opaca e poco leggibile. Ho cercato nel tempo disponibile di trovare la soluzione piú facilmente realizzabile, cercando di dare piú importanza alla sezione delle statistiche, di cui sono sicuramente piú soddisfatto e in cui ho cercato di rendere il codice il piú possibile estendibile e riusabile. É stato interessante cimentarsi nell'utilizzo del pattern MVC, cercando di rendere le varie sezioni indipendenti tra loro. Una difficoltá che ho riscontrato in questa fase é stata inizialmente quella di comprendere in che modo utilizzare la libreria JavaFX in un contesto MVC, visto che introduce il concetto di Controller, il quale mi é sembrato molto piú un componente della sezione di View del MVC. É stata anche molto interessante la realizzazione della relazione che ha aiutato a rendere ancora piú chiara l'architettura e il design dell'applicazione.

Alice Mastrilli

Il progetto è stato molto impegnativo ma sono soddisfatta del risultato ottenuto: è stato il primo progetto a cui ho preso parte e, prima del corso, non avevo conoscenza della programmazione ad oggetti. Su diverse cose ho avuto dubbi ma i miei compagni di gruppo sono sempre stati molto disponibili ad aiutarmi. Grazie a questa esperienza sento di aver imparato molto(forse non tutto) sul mondo di Java e non solo, come i file .FXML e css. Come gruppo non abbiamo avuto grosse difficoltà: chiaramente non ci siamo mai potuti incontarre personalmente, ma ci siamo organizzati con incontri settimanali dove ci aggiornavamo su quanto fatto. Nello stesso tempo non è stato semplice far concordare cinque "teste" ma, quando discordanti, abbiamo adottato delle soluzioni che soddisfacessero il più possibile tutti quanti. Un progetto futuro che spero verrà portato a termine è la realizzazione di questo stesso software anche per dispositivi mobili, così da rendere Focus più portabile e sfruttarlo al meglio.

Edoardo Puce

Essendo stato questo il mio primo progetto informatico di gruppo, lo sviluppo è risultato abbastanza impegnativo, sopratutto nelle fasi iniziali. Essenziale è stato il supporto dei colleghi di progetto, i quali sono sempre stati disponibili per delucidazioni.

Per quanto riguarda lo svolgimento del lavoro di gruppo, questo è stato eseguito tramite incontri da remoto infrasettimanali, utilizzando poi il fine settimana per fare il punto della situazione e valutare i progressi fatti. Lo studio e la stesura del programma si sono rilevati altamente stimolanti. Infatti, alcune sessioni di lavoro si sono concluse in tarda serata, tra il piacere di programmare e la mia lentezza nell'eseguire il debug.

Concludendo, non posso che ritenermi soddisfatto dei miei progressi nel mondo della programmazione ad oggetti. Ad oggi sento di avere le idee più chiare per quanto riguarda l'organizzazzione di un progetto nel suo complesso e tali conoscenze acquisite mi torneranno sicuramente utili per portare a termine una precedente applicazione software in C sharp a cui ho dato inizio l'anno scorso senza avere alcuna base.

Alex Siroli

L'impegno in questo progetto non è mancato, da parte mia e da parte di tutti. Nonostante il lavoro da svolgere fosse tanto, nel gruppo c'è sempre stata disponibilità, sia per consigli che per dubbi implementativi.

Il mio principale ruolo nel gruppo penso sia stato quello di tester del software, sempre alla ricerca di qualche bug di cui nessuno si era reso conto. Contributo sicuramente positivo per quanto riguarda lo sviluppo dell'applicazione ma che può essere altrettanto fastidioso per i colleghi, che però hanno sempre accolto i miei consigli con piacere.

Il progetto ha migliorato molto le mie skill informatiche, ma anche quelle sociali e di collaborazione dato che questo è stato il mio primo vero progetto di spessore.

In conclusione posso ritenermi soddisfatto di quanto fatto e di quanto imparato, nonostante l'esperienza sia stata molto impegnativa.

4.2 Difficoltà incontrate e commenti per i docenti

Edoardo Puce

I problemi riscontrati sono stati essenzialmente due. Il primo è relativo al pattern MVC che, inizialmente, non è stato implementato in maniera ottimale. Solo dopo diverso tempo ho compreso come seguirlo correttamente (sperando di averlo fatto).

Il secondo è legato alla divisione dei compiti, rispetto al monte ore massimo richiesto. Infatti, non avendo mai svolto progetti informatici di gruppo, L'iniziale stima delle ore non si è rilevata congrua con la realtà e il tempo massimo è stato sforato, soprattutto a causa di difficoltà incontrate durante le fasi di debug.

Alex Siroli

L'unico vero e proprio problema riscontrato nell'esecuzione del progetto è stata la necessità di dover imparare molto in autonomia. Le lezioni svolte non permettono a mio parere un'implementazione di un software decente quindi siamo tenuti noi studenti a impiegare tanto tempo solo per avere tutte le facoltà necessarie. Mi rendo conto che l'obbiettivo del progetto sia proprio questo, ma a

questo punto questa parte di esame è veramente molto più impegnativa rispetto alla prima parte in laboratorio, considerando che bisogna addirittura svolgere una sezione aggiuntiva in c sharp che attualmente reputo eccessiva.

Appendice A

Guida utente

HomePage

L'homepage presenta due sottosezioni, una relativa al calendario ed una alle finanze.

Sottosezione calendario

Questa parte permette all'utente di visualizzare tutti i tasti rapidi salvati, e, cliccando su uno di questi, di effettuare una determinata azione definita in base al tipo:

- se si tratta di un contatore ne verrà incrementato il valore;
- se si tratta di un'attività, il tasto verrà disabilitato per il resto della giornata;
- se si tratta di un evento si aprirà una nuova finestra dalla quale sarà possibile aggiungere un nuovo evento per la giornata odierna.

Nell'ultimo caso, sarà possibile selezionare più partecipanti tenendo premuto il tasto ctrl.

Selezionando il tasto modifica si aprirà una finestra contenente una tabella nella quale saranno contenute tutte le informazioni dei tasti rapidi.

Dalla suddetta finestra l'utente potrà, selezionando la riga e cliccando il tasto "elimina l'elemento selezionato", eliminare un elemento e aggiungerne uno nuovo cliccando il tasto "aggiungi".

Sottosezione finanze

Questa parte si divide in tre sezioni principali:

- **sezione conti:** è la sezione in alto a destra e premette una visualizzazione rapida di tutti i conti salvati con i relativi saldi aggiornati in tempo reale. I conti sono visualizzati in ordine alfabetico.
- **sezione transazioni recenti:** è la sezione in basso a destra che permette una visualizzazione rapida delle transazioni svolte nella giornata odierna aggiornate in tempo reale. E' anche possibile cliccare su una transazione per visualizzarla nel dettaglio. Le transazioni sono visualizzate dalle più recenti alle meno recenti.
- **sezione transazioni rapide:** è la sezione a sinistra e rappresenta tutte le transazioni rapide salvate in precedenza. Al click su una transazione rapida questa viene eseguita, con data e ora del momento in cui si è stato effettuato il click. Le transazioni rapide sono visualizzate in ordine alfabetico.

In basso sono presenti tre pulsanti:

- **Nuova transazione:** permette di creare una transazione in base ai dati inseriti. Non è consentito eseguire una transazione in data e ora future e per permettere un corretto inserimento è necessario che tutti i campi siano compilati correttamente.
- **Nuova transazione Rapida:** permette di creare una nuova transazione rapida, come per l'aggiunta di una transazione tutti i campi devono essere compilati correttamente.
- **Elimina Transazioni Rapide:** elimina tutte le transazioni rapide salvate.

Calendario

La sezione del calendario consente all'utente di visualizzare, aggiungere, eliminare e stoppare la ripetizione dei vari eventi salvati e visualizzarli nella sezione del mese o nella settimana. Nella sezione mensile del calendario l'utente può visualizzare gli eventi nella giornata cliccando sull'apposito giorno. Cliccando sul tasto "Impostazioni" l'utente può cambiare due proprietà che caratterizzano la visualizzazione del giorno: lo spazio tra un ora e quella successiva (lo spazio minimo è 30) e il formato che permette la visualizzazione degli eventi o da ora in ora o mezz'ora a mezz'ora. L'utente può aggiungere un evento in qualsiasi momento tramite il tasto "aggiungi evento" e potrà poi visualizzarlo nella settimana o nel mese.

Per selezionare più partecipanti per lo stesso evento basterà che l'utente tenga premuto il tasto ctrl e selezioni le persone desiderate.

Scegliendo il tasto "informazioni eventi" si aprirà una nuova finestra contenente una tabella dalla quale l'utente potrà selezionando un elemento della tabella, premere il tasto "elimina l'elemento selezionato" per eliminarlo o premere il tasto "informazioni", per visualizzare le suddette e per, eventualmente, stopparne una ripetizione.

Sempre all'interno della sezione calendario l'utente potrà gestire l'aggiunta e l'eliminazione di persone e parentele, cliccando sull'apposito tasto "persone".

Finanza

La sezione della finanza si divide in sei sottosezioni:

Tutte, uscite e entrate

In queste prime tre sezioni è possibile visualizzare le transazioni tramite dei filtri. In TUTTE troveremo quindi tutte le transazioni effettuate ordinate in base alla data di esecuzione, in USCITE solo le transazioni in uscita e in ENTRATE solo le transazioni in entrata. Eseguendo un click su una transazione verranno visualizzati tutti i dettagli di essa e sarà possibile anche eliminarla. A sinistra è presente anche un elenco di pulsanti rappresentanti i conti salvati, cliccando su uno di essi saranno visualizzare solo le transazioni relative a quel conto. I conti sono visualizzati in ordine alfabetico. Esempio: se mi trovo nella sezione USCITE e faccio click sul conto "Portafoglio", si visualizzeranno le transazioni in uscita del solo conto "Portafoglio". In queste tre sezioni possiamo trovare in basso tre pulsanti:

- **Crea conto:** permette di creare un nuovo conto. Per permettere l'inserimento devono essere compilati tutti i campi correttamente.
- **Elimina *nome conto/i*:** elimina il conto visualizzato a schermo e tutte le transazioni relative.
- **Nuova Transazione:** permette di creare una transazione in base ai dati inseriti. Non è consentito eseguire una transazione in data e ora future e per permettere un corretto inserimento è necessario che tutti i campi siano compilati correttamente.

Statistiche

In questa sezione è possibile visualizzare le statistiche relative alla finanza potendo considerare i conti e il lasso di tempo desiderati.

Abbonamenti

In questa sezione è possibile visualizzare le transazioni ripetute ancora attive, ordinate dalla meno frequente alla più frequente. Facendo click su un abbonamento verranno mostrati a schermo tutti i dettagli relativi e sarà anche possibile interrompere la ripetizione automatica. In basso sono mostrate la spesa totale media in abbonamenti nell'arco di un mese e nell'arco di un anno.

Gruppo

In questa sezione è possibile gestire un gruppo di persone e tutte le transazioni di gruppo relative. In "Gruppo" sono mostrate le persone aggiunte al gruppo in ordine alfabetico con il relativo debito/credito, facendo click su una persone è possibile visualizzarla e/o rimuoverla dal gruppo. Non è possibile eliminare una persona appartenente al gruppo se questa ha debiti o crediti. In basso a sinistra il pulsante "Aggiungi persona al gruppo" permette di aggiungere persone esistenti o di

crearne di nuove. In "Transazioni di gruppo" sono invece mostrate le transazioni di gruppo ordinate dalla più recente alla meno recente. Facendo click su una di esse verranno mostrati a schermo i dettagli di ognuna e la possibilità di eliminarle. In basso a sinistra in questa sezione è presente il pulsante "Reset" che permette di eliminare tutte le persone e le relative transazioni di gruppo ma solo se tutti hanno saldato prima i propri debiti. Per saldare i debiti in maniera automatica è presente il pulsante "Risolvi crediti/debiti" che mostra a schermo le transazioni da eseguire per permettere una veloce risoluzione dei debiti. Una volta eseguiti sarà necessario premere su "Salva" e le transazioni generate verranno eseguite automaticamente. Per creare invece una transazione di gruppo è necessario premere il pulsane in basso a destra "Crea transazione". Ovviamente i campi dovranno essere compilati correttamente e la data non può essere futura.

Diario

In questa sezione sono presenti: un toDoList, una sezione per aggiornare il mood giornaliero e il diario vero e proprio. Il diario si compone di due pulsanti per l'aggiunta o rimozione delle pagine di diario stesse, caratterizzate da un titolo e da un contenuto. Le pagine salvate possono essere modificate: basterà cliccare sul contenuto e sarà possibile cambiarne il testo. Il pulsante "modifica" in basso salverà le modifiche effettuate. Anche la sezione To Do List contiene due bottoni per il salvataggio o rimozione di una To Do Action: in questo caso, per creare una nuova nota, sarà necessario inserire solamente la stringa rappresentante l'attività da fare. Inoltre, spuntando un'annotazione, questa vedrà cambiarsi il suo stato: se non era stata selezionata, verrà segnata come fatta, e viceversa. Infine, nel daily mood, cliccando su un'icona questa verrà salvata come mood relativo alla giornata odierna. Se, invece, si vuole cambiare scelta, sarà possibile cliccare su "modifica" in basso e il mood precedentemente salvato verrà eliminato. I mood scelti nei giorni potranno essere visualizzati nella sezione "statistiche umore".

Cronometro

Qui è possibile far partire un cronometro e assegnarne una stringa rappresentante l'evento che si vuole conteggiare. Sarà possibile scegliere, nell'apposito menù a tendina, eventi già salvati, mentre, se si vuole aggiungerne di nuovi, basterà cliccare sul pulsante vicino "+". Appena scelto un evento, in "Tempo totale" verrà visualizzato il tempo totale finora trascorso a svolgere quell'evento. Premendo il pulsante "Start", il cronometro partirà e con "Stop" si fermerà. Allo stop, il tempo totale si aggiornerà automaticamente.

Timer

Il funzionamento del timer è pressoché lo stesso del cronometro.L'unica funzione aggiuntiva è la possibilità di scegliere il tempo del timer: a tal scopo sono presenti quattro pulsanti. Tre di questi hanno un tempo preimpostato, il quarto, "Scegli", permette all'utente di scegliere il tempo, inserendo, in ordine, ore, minuti e secondi del timer.

Appendice B

Esercitazioni di laboratorio

Alice Mastrilli

- Laboratorio 04: `https://virtuale.unibo.it/mod/forum/discuss.php?d=62685#p108212`
- Laboratorio 05: `https://virtuale.unibo.it/mod/forum/discuss.php?d=62684#p108213`
- Laboratorio 06: `https://virtuale.unibo.it/mod/forum/discuss.php?d=62579#p108215`
- Laboratorio 08: `https://virtuale.unibo.it/mod/forum/discuss.php?d=63865#p108216`
- Laboratorio 09: `https://virtuale.unibo.it/mod/forum/discuss.php?d=64639#p108252`
- Laboratorio 10: `https://virtuale.unibo.it/mod/forum/discuss.php?d=66753#p108284`