book-depot

Cecchetti Giulia
Colombo Andrea
Croccolino Lorenzo
Feroce Marcello

29 Febbraio 2016

Indice

Capitolo 1 - Analisi

1.1 Requisiti

Il software mira alla realizzazione di un programma adibito alla gestione di trasferimenti di libri dai magazzini di una casa editrice verso enti esterni come librerie, privati e stamperie; e viceversa. Book-Depot è nato dall'esigenza di uno dei componenti del gruppo di tenere traccia degli spostamenti dei libri della casa editrice di famiglia, per cui sarà destinato ad un reale utilizzo. Una casa editrice ha un elenco di titoli da essa prodotti e uno o più magazzini dove questi, una volta stampati verranno depositati; ha inoltre un elenco di clienti, che possono essere privati o librerie. L'esigenza da soddisfare è quella di tenere traccia di tutti questi enti, ma soprattutto di memorizzare i trasferimenti a fini fiscali.

Requisiti concordati:

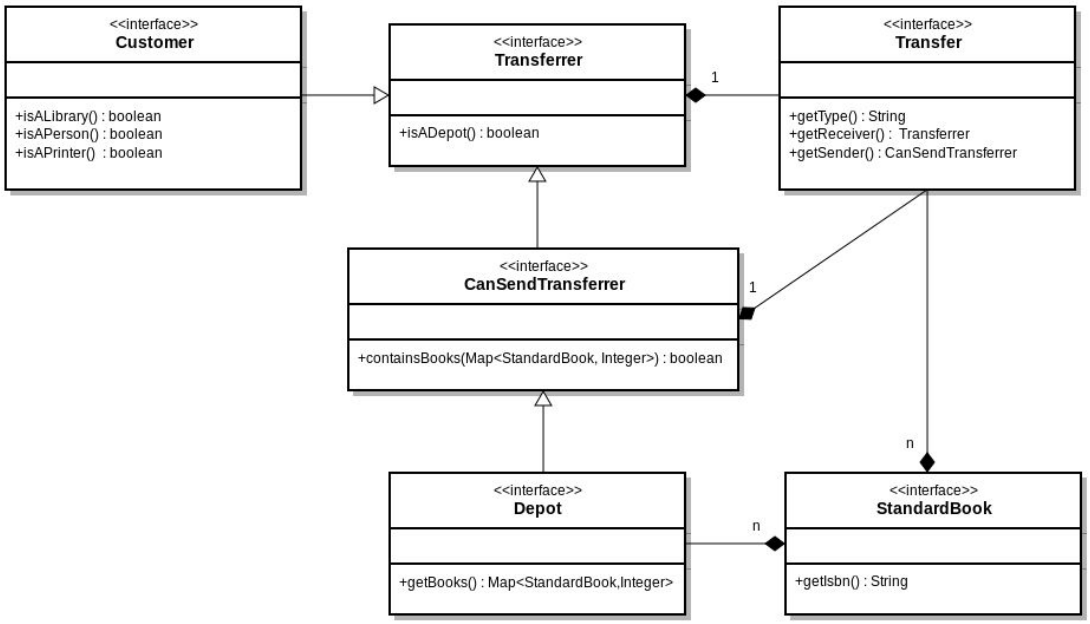
- Il programma dovrà occuparsi della gestione completa di tutte le attività di trasferimento. Dovrà perciò rendere possibile la creazione di uno o più magazzini, l'inserimento di titoli e la registrazione di clienti e/o fornitori che diventeranno mittenti e destinatari dei movimenti registrati.
- I movimenti potranno essere di tre tipi: "venduto" se un magazzino invia libri a un cliente, "reso" se una libreria invia libri a un magazzino, o "rifornimento" se c'è un movimento interno fra magazzini o proveniente da stamperia.
- Il software dovrà inoltre elaborare dei file PDF contenenti i resoconti stampabili dei movimenti inseriti e dovrà permetterne l'invio verso tramite email

1.2 Analisi e modello del dominio

Book-Depot dovrà essere in grado di accedere e controllare uno o più magazzini tenendo traccia dei movimenti interni ed esterni. Dovrà perciò elaborare informazioni al fine di effettuare corretti trasferimenti verso diversi tipi di entità: alcune abilitate a ricevere o inviare libri, altre in grado unicamente di riceverne. Il software dovrà inoltre gestire i trasferimenti registrando una data, un numero di tracciamento e il tipo e numero di libri trasferiti. In questo modo, determinerà un

cambiamento di valori all'interno di un magazzino, aumentando o diminuendo le quantità in esso presenti. Calcolerà e registrerà inoltre il prezzo totale del trasferimento, per tenere traccia dei movimenti fiscali.

Una difficoltà sarà perciò quella di salvare le informazioni su file, in modo che riavviata l'applicazione tutte le informazioni relative ai trasferimenti, ai depositi e alle quantità di libri in essa presenti non vadano perse.
Il programma si soffermerà inoltre sulla limitazione degli errori da parte dell'utente nella fase di inserimento di un movimento. Cercherà perciò di suggerire combinazioni corrette filtrando e suggerendo un destinatario possibile a seconda del mittente ed altri parametri. Gli elementi costitutivi il dominio sono sintetizzati nella figura sottostante.



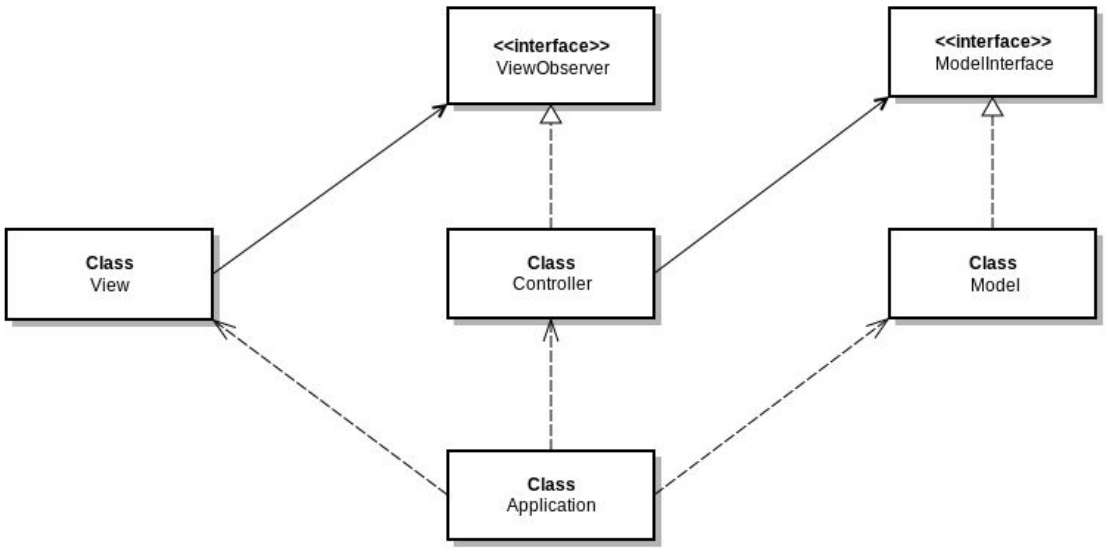
Capitolo 2 - Design

2.1 Architettura

Per la realizzazione del progetto abbiamo sviluppato il pattern MVC. Per questo motivo, ci siamo concentrati su una totale separazione tra Model e View, utilizzando un Controller come unico tramite fra essi. Abbiamo perciò studiato la struttura del primo in modo tale da creare entità ben strutturate ed istanziabili dal controller, che una volta elaborate e registrate venissero restituite alla view al fine di un'iterazione completa e ben progettata con l'utente.

L’ interfaccia grafica è stata connessa al controller attraverso più interfacce ViewObserver in modo da garantire una relazione pulita ed indipendente. Questa, infatti, una volta raccolte informazioni dall’utente effettua chiamate a metodi del controller che, oltre a garantire la correttezza delle informazioni, fanno uso delle interfacce del model al fine di modificare o istanziare nuovi oggetti.

In questo modo, il gruppo ha ritenuto di separare in maniera efficiente la parte grafica da ogni relazione col model e di renderla immune a danni causati da una sostituzione in blocco della view. I metodi del controller sono stati infatti strutturati per ricevere dai componenti grafici informazioni e dati base da elaborare, o confrontare con quelli già inseriti. La GUI ha il solo compito di ricevere gli input dell’utente e trasmettere le elaborazioni del controller come output, ma è completamente immune da ogni tipo di modifica dei dati o delle strutture.



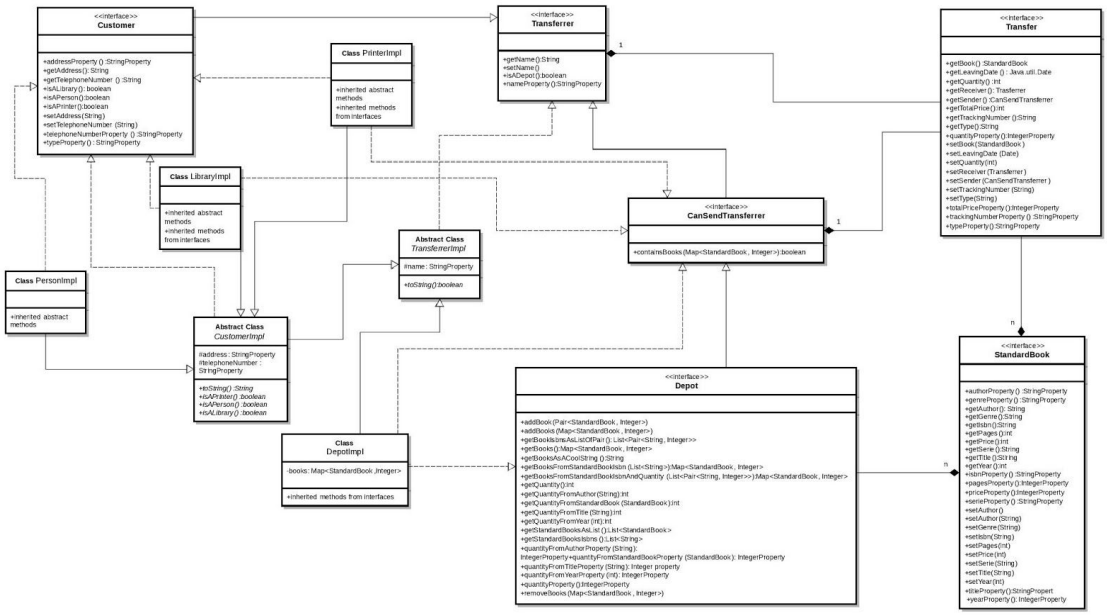
2.2 Design dettagliato

2.2.1 Model

Feroce Marcello

Quando si lavora con i trasferimenti, necessariamente si dovrà lavorare con entità trasferenti come persone private, magazzini(depot), librerie, stamperie. Tutti gli oggetti di questo tipo saranno di un tipo quindi che estende una classe astratta TransferrerImpl, la quale a sua volta implementa l'interfaccia Transferrer. Il

vantaggio di avere TransferrerImpl astratta è che si può sfruttare il fatto che ogni entità trasferente possiede una proprietà comune, il nome. I metodi relativi al nome infatti sono gestiti in TransferrerImpl. Ho quindi sfruttato qui il pattern Strutturale Template Method. A sua volta possiamo suddividere le entità trasferenti in due gruppi, i magazzini(depot), e i "clienti"(customer, scusate la poca fantasia). Per i depot ho creato una classe a parte che implementa la sua interfaccia Depot, e estende TransferrerImpl,così da poter implementare i modo diverso da altre classi i metodi astratti della classe padre. Ho fatto lo stesso lavoro usando lo stesso pattern per la classe astratta CustomerImpl, che a sua volta estende TransferrerImpl. Ho poi usato il pattern strutturale Strategy, per creare classi che implementino Customer e estendino CustomerImpl; è il caso delle Classi PrinterImpl, LibraryImpl(indica una libreria), e PersonImpl.
Inoltre ho stabilito che ogni oggetto di tipo Transfer deve possedere necessariamente un campo di un tipo particolare, di tipo CanSendTransferrer. Questa interfaccia è implementata solo da classi che identificano entità che possono inviare libri. Questa interfaccia contiene infatti metodi necessari solo da queste classi. Questa moltitudine di classi astratte e interfacce mi consente di fare poche modifiche al codice se voglio cambiare la struttura di ogni oggetto, andando a modificare solo la classe più generale. Nel model ho avuto la necessità di creare due diversi packages, uno per le interfacce ed uno per le classi.



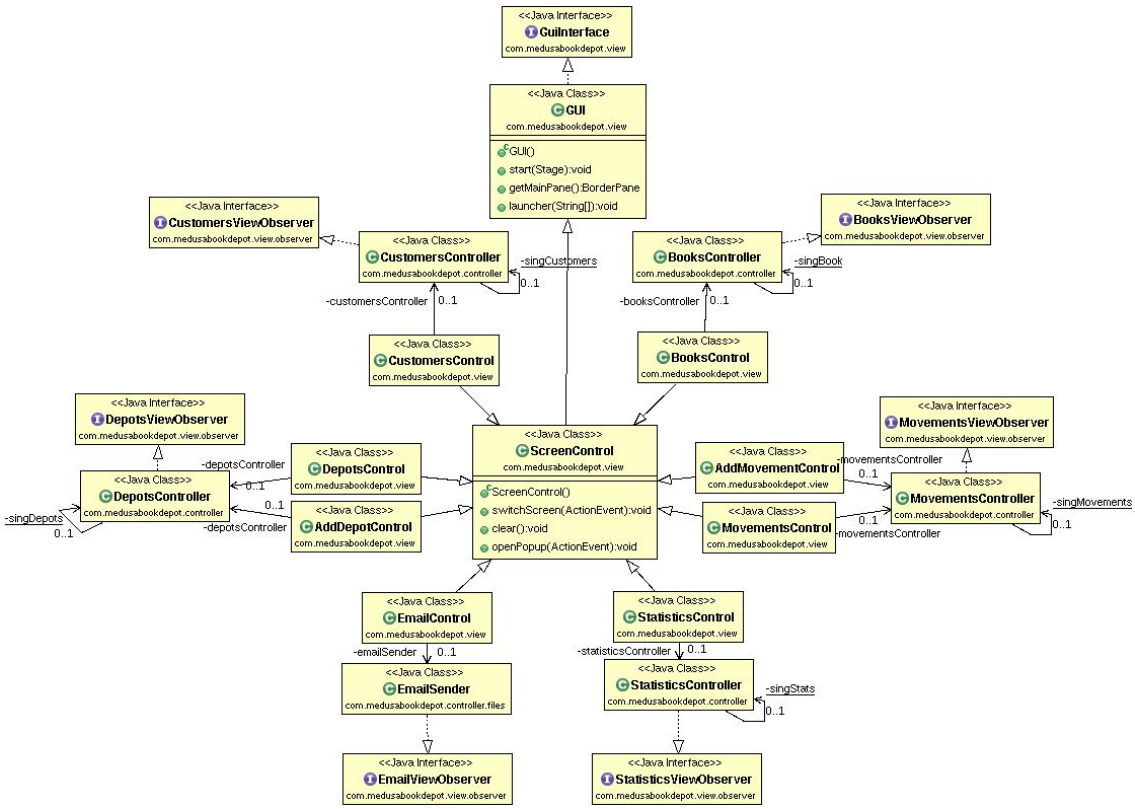
2.2.2 View

Per realizzare l'interfaccia grafica del software è stata utilizzata la libreria JavaFX, e in particolare il linguaggio FXML per definire il layout delle schermate. È stato applicato inoltre un foglio di stile css.

La struttura e la libreria per lo sviluppo della GUI sono stati concordati in fase iniziale a seguito di due incontri di gruppo fra Giulia Cecchetti e Andrea Colombo, dove abbiamo analizzato tutte le necessità di tipo grafico: pagine necessarie, gestione del menu, librerie da utilizzare e abbiamo lavorato su una prima versione funzionante.

In particolare, è stato gestito un menù attraverso una classe ScreenControl che, alla pressione di uno dei bottoni della barra, carica al centro del pannello principale un diverso file FXML a seconda dell'ID ricevuto direttamente dal bottone che è stato premuto. Successivamente, ogni classe control specifica ha esteso la classe ScreenControl per ricevere questa funzionalità. Ognuna di queste classi è stata inoltre dotata di un metodo initialize(), per inizializzare al caricamento della pagina i componenti quali tabelle o bottoni secondo la necessità. Ogni richiamo nel codice java agli elementi presenti nei file FXML è reso possibile da un id del componente e un'inizializzazione con notazione @FXML.
Si è utilizzata inoltre, per la gestione di ogni pagina, un'istanza di uno specifico Controller, incaricato di creare e apportare le modifiche richieste ai file. Le classi della View sono state strutturate per passare e/o ricevere i dati dal Controlller sotto forma di stringa e per visualizzarli come tali all'interno di tabelle. L'utilizzo di TableView popolate con gli oggetti delle ObservableList ha reso semplice e dinamico l'aggiunta di elementi e l'aggiornamento dei dati presenti.

Abbiamo scelto di partire dalla schermata Books per avere una prima schermata funzionante, ci siamo perciò divisi i compiti tra il layout dello screen con rispettivo controllo, e la parte riguardante la tabella, con gestione delle relative azioni. Dopodichè abbiamo usato il lavoro fatto per completare il resto delle schermate.



In figura, l'UML della view e la sua relazione con il controller.

Cecchetti Giulia

Per iniziare mi sono occupata del layout dello screen Books, la rispettiva classe BooksControl a livello base, e del succesivo screen AddBooks. Dopodichè mi sono soffermata sulla costruzione del layout di ogni altra pagina, e della realizzazione del relativo Control per tutte le altre schermate, tra le quali Depots e AddMovements tralasciando alcuni funzioni aggiuntive. Le classi Control generalmente si occupano di relazionare la GUI, quindi i suoi bottoni, i suoi campi e le sue tabelle con il controller, oltre che instanziare e rendere reattivi i componenti creati tramite layout FXML. Ho realizzato la schermata e la gestione delle Statistiche, per farlo ho utilizzato gli appositi componenti delle librerie JavaFX. Ho lavorato alla ricerca / filtraggio all'interno delle TableView, in primo momento utilizzando classi apposite di JavaFX, poi volendo migliorare il mio codice ho scelto di sfruttare direttamente i metodi scritti dal controller in precedenza. Ho inoltre gestito il package "Alerts" e ottimizzato delle ripetizioni di codice sfruttando un ciclo "for" per lo scorrimento delle colonne di una tabella. Infine ho realizzato la view per la pagina Email.

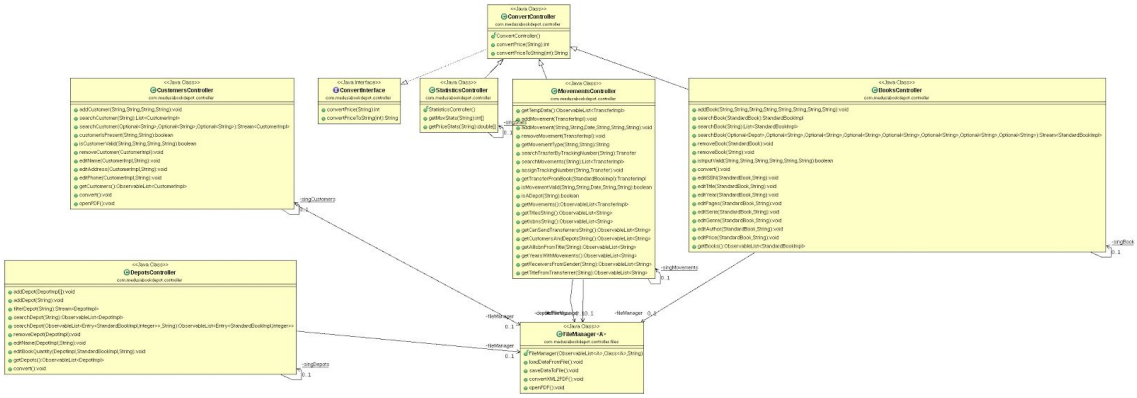
Colombo Andrea

Dopo gli incontri di gruppo, mi sono concentrato sulla parte della tabella, il controllo di essa e delle sue funzioni, riguardanti la schermata Books, per rendere la schermata funzionante. Mi sono occupato poi di rendere la schermata Depot filtrabile a seconda del depot selezionato premendo il rispettivo bottone, rendendo la parte dei bottoni dinamica a seconda della lista dei depot. Mi sono soffermato sulla schermata Add Movement in cui ho implementato la pulizia dei campi a inserimento avvenuto, e ho realizzato delle ComboBox con autocompletamento. Di queste, alcune usano metodi del controller per filtrare le scelte possibili a seconda di precedenti selezioni. Infine ho realizzato un popup in cui si visualizza il nome degli autori, premendo il pulsante più a destra nel menu. Mi sono anche occupato di tutte le modifiche relative al CSS.

2.2.3 Controller

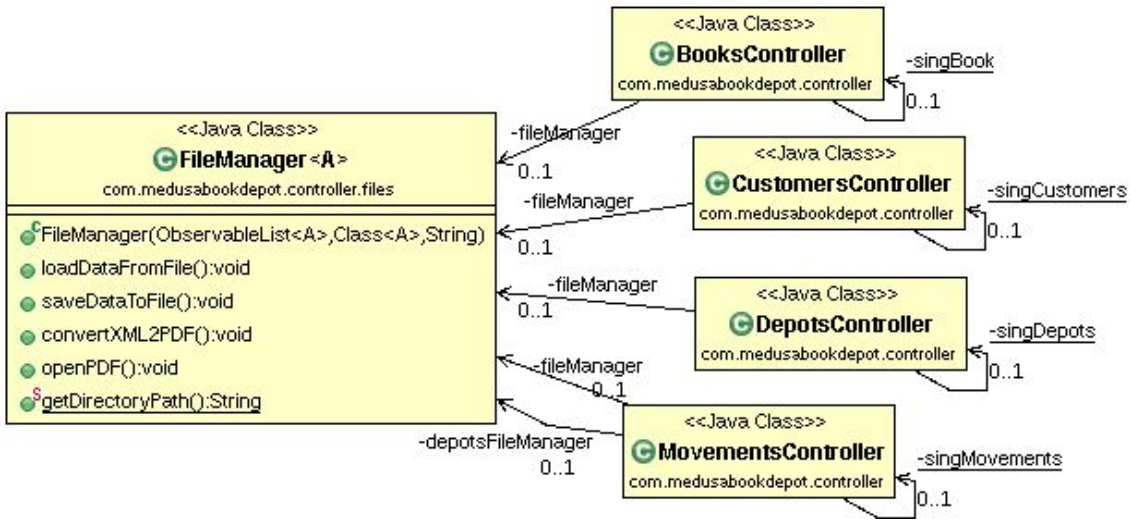
Croccolino Lorenzo

L'intero controller è stato scritto rispettando e seguendo passo passo tutte le funzionalità descritte nella presentazione iniziale del software. La logica seguita è stata quella di una rappresentazione delle classi come se fossero reparti, o graficamente parlando, pagine di una interfaccia grafica.
La maggior parte dei metodi prendono in input String anziché oggetti già formati o altri tipi di dati come Integer, questo per facilitare il lavoro alla View, cosicché quest'ultima non debba eseguire nessun controllo sull'input, demandandoli, appunto, al controller.
Il pattern di programmazione utilizzato principalmente nelle classi è stato quello del Singleton, utilizzato per mantenere le informazioni in un oggetto unico caricabile ad ogni richiamo. Questo ha facilitato non poco l'interazione fra le varie classi e il recupero dei dati all'interno senza perdita di informazioni.
Ho scelto all'interno di ogni metodo di modifica delle informazioni, di eseguire sempre un salvataggio su file dopo ogni operazione. Questa strategia per aumentare la sicurezza e diminuire al minimo il rischio di perdita di dati per mancati salvataggi dovuti ad eventi improvvisi o a dimenticanze.
Proprio per questa caratteristica ho preso ispirazione da tool come Google Doc e altri, oltre che da brutte esperienze personali.
Si precisa che al cambiare di view, ad esempio da una GUI a una UI a linea di comando, il controller non presenterà alcuna problematica poichè le interazioni del controller sono solamente in direzione del model e non della view.



Colombo Andrea

Contemporaneamente al lavoro sulla GUI, mi sono concentrato sulla scrittura su file, utilizzando la libreria JAXB presente nel JDK. Inizialmente ho implementato il salvataggio solo per gli oggetti di tipo StandardBook, attraverso un wrapper della classe StandardBookImpl. Dopodichè ho reso generico il File Manager e il relativo Wrapper, per cui è stato reso possibile salvare ogni tipo di oggetto del model, e la relativa lettura. Ho dovuto per cui aggiungere al model diverse annotazioni JAXB dedicate, per strutturare correttamente i file XML risultanti. In seguito attraverso la libreria Apache FOP e il codice di esempio incluso, ho creato un metodo per convertire a PDF nella classe FileManager. Per la conversione è necessario un file di tipo XSL-FO che ho dovuto creare appositamente per ogni oggetto da convertire. Non è per cui possibile la conversione da XML a PDF senza template di conversione. Mi sono occupato della classe per mandare email ed allegati, che manda email da un account gmail harcoded (soluzione che ho provato ad evitare implementando login a Gmail tramite le rispettive API, mi sono arreso ai vari protocolli necessari che non conosco).



In figura, l'UML del FileManager e la sua relazione con il controller, che lo utilizza per salvare oggetti provenienti dal model.

Capitolo 3 - Sviluppo

3.1 Testing automatizzato

Il programma è stato sottoposto a testing automatizzato. Il testing riguarda le entità principali del dominio, ovvero le classi che descrivono oggetti di tipo trasferimento, deposito, libro, persona, libreria, stampa.

Per il testing è stata utilizzata la libreria Junit.

3.2 Metodologia di lavoro

Ognuno dei componenti del gruppo si è concentrato maggiormente su una parte del progetto. Nello specifico, facendo uso del pattern MVC, la suddivisione è stata questa:

Feroce Marcello, Model
Croccolino Lorenzo, Controller
Cecchetti Giulia, View
ColomboAndrea, View + ricerca API e relativo sviluppo

La divisione del lavoro è stata equa, e il lavoro di ognuno è stato fondamentale per il corretto funzionamento del programma.
La realizzazione dell'architettura dei principali elementi del dominio è stato un lavoro corale. Nonostante questo, in un caso particolare, è stato necessario un processo a spirale per revisionare l'architettura, a causa di una dimenticanza in fase iniziale. Il campo "tipo" del trasferimento è stato infatti gestito dal controller piuttosto che dal model, mentre sarebbe stato più opportuno il contrario, se solo ci fossimo accorti dell'esigenza di questo campo in fase di progettazione.
Il gruppo ha utilizzato il DVCS Mercurial; è stato impiegato un repository online Bitbucket, e si è lavorato su un singolo branch.

3.3 Note di sviluppo

Nel model è stata adoperata la classe Pair realizzata da Danilo Pianini e Mirko Viroli.

Per l'apprendimento e l'approfondimento riguardo alle librerie JavaFX, il salvataggio dei dati su file XML, e la realizzazione della schermata delle statistiche abbiamo seguito le guide del blog di Marko Jacob. Ogni spunto è comunque stato preso solamente come idea, quindi studiato e rielaborato.
Il CSS utilizzato per la parte grafica è disponibile in questo repository di Agix, certi elementi sono stati modificati e alcune righe di codice sono state aggiunte per particolari necessità. Per l'utilizzo di una barra menu, posizionata in alto, e condivisa da ogni schermata, è stata sviluppata l'idea del blog Java Programming Tutorials, rendendola dinamica secondo le nostre esigenze. Per la conversione da XML a PDF sono state utilizzate le librerie e il codice di esempio Apache FOP. Per l'invio di email con allegati è stata seguita questa risposta, mentre per l'autocompletamento delle comboBox quest'altra, entrambe su Stack Overflow. Per il PersistentButtonToggleGroup la risposta di JSmith sulla community ufficiale Oracle.

Capitolo 4 - Commenti finali

4.1 Autovalutazione e lavori futuri

Il punto di forza dell'applicazione è l'intuitività e l'attenzione alla relazione con l'utente. Infatti, è stato sviluppato un controllo che non solo esamina l'input dell'utente ma lo guida ad effettuare inserimenti sempre corretti. Inoltre, è stata molto curata l'interfaccia grafica secondo le guidelines del Material Design per offrire all'utilizzatore un'esperienza visiva gradevole, semplice e funzionale. Oltretutto, l'utente ha poche probabilità di perdere i dati registrati, in quanto l'applicazione effettua salvataggi istantanei su file ad ogni modifica.

Un punto di debolezza è la specificità del software, realizzato per le particolari esigenze di questa casa editrice. Per questo motivo adattare l'applicazione ad altre case editrici potrebbe non essere immediato.

> *Feroce Marcello*: Penso di aver lavorato abbastanza bene evitando il più possibile ripetizioni di codice, e costruendo le interfacce necessarie per una comprensione del dominio.
>
> *Croccolino Lorenzo*: Penso di aver eseguito un buon lavoro in termini di funzionalità, cercando di rendere sempre più semplice il lavoro alla view ed eseguendo controlli il più possibile accurati e precisi. Ho utilizzato stream quando possibile preferendoli ai classici cicli ritenendoli migliori. Diverse cose nel codice forse sarebbero migliorabili sia in termini

prestazionali sia in termine di ripetizioni del codice, anche se comunque ho sempre cercato di evitarle.

Cecchetti Giulia: Ritengo di aver svolto un buon lavoro sia per quanto riguarda la strutturazione dell'interfaccia grafica, intuitiva e studiata, sia per quanto riguarda la progettazione del codice. Ho cercato di separare il piu possibile ogni dipendenza della view dal model e di gestire le relazioni col controller in modo chiaro e coerente. Sono molto fiera del risultato ottenuto, soprattutto perchè ho sperimentando un nuovo linguaggio come l'FXML e il suo relativo funzionamento. Una difficoltà che ho riscontrato è stata soprattutto nella suddivisione del lavoro, non è stato facile spartirsi i compiti soprattutto per la rapidità con cui il progetto si evolveva e con cui si presentavano delle necessarie modifiche, ma ad ogni modo ci siamo impegnati a cooperare in modo efficiente e trovo che, nonostante una poco netta divisione, si stato possibile un lavoro armonico ed equilibrato.

Colombo Andrea: Per la mia parte relativa al controller (del package controller.files), sono molto contento di come ho reso il file manager, e il relativo wrapper di elementi, generici, in quanto ogni oggetto creato dal model può essere salvato su file XML passando attraverso il wrapper. Invece per quanto riguarda la view sono contento di come il Menu e il cambio di schermata siano generici, ma credo che si sarebbe potuta evitare ripetizione di codice se in fase architetturale le relazioni col Controller fossero state strutturate diversamente, soprattutto per i metodi comuni fra una schermata e l'altra, ad esempio quelli per i bottoni add, edit e delete. Credo anche che avrei potuto migliorare il mio codice nel metodo update() nella classe AddMovementControl, che ho dovuto fare poco prima della consegna.

Il programma potrebbe essere migliorato ulteriormente, sviluppando funzionalità aggiuntive, tra cui il tracciamento di un pacco attraverso un Tracking Number, ma crediamo che per l'ammontare di ore dedicate sia stato raggiunto un buon livello e ci sentiamo di dire che complessivamente siamo soddisfatti del lavoro svolto.

4.2 Difficoltà incontrate e commenti per i docenti

A seguito di alcuni problemi con le annotazioni delle librerie JAXB, abbiamo avuto difficoltà nella scrittura su file di oggetti di tipo Depot, a loro volta composti da una mappa di StandardBook e Integer. Il problema è quello che JAXB non accetta il salvataggio di interfacce, se non con relative annotazioni in esse. Siamo riusciti a risolvere il problema in parte, come per le interfacce

Customer e per altre classi estese, ma data la complessità della struttura Depot dentro alla quale è annidata un interfaccia StandardBook, abbiamo scelto di utilizzare sempre oggetti di tipo DepotImpl e StandardBookImpl. Nonostante ciò, nei diagrammi UML riferiti al modello, non abbiamo rappresentato queste modifiche ma abbiamo scelto di lasciare intatta la progettazione architetturale iniziale.

> *Colombo Andrea*: Non ho avuto problemi di nota ad apprendere librerie non trattate nel corso, come JavaFX o FOP, ma ho provato a relazionarmi con API di Google per login e autorizzazione a Gmail, ed ho avuto problemi a trattare i vari protocolli web a me sconosciuti. Stessa cosa quando ho provato ad utilizzare le API del corriere FedEx, in fase di progettazione avevamo pensato si potesse usare una libreria che leggesse un feed RSS per ricevere lo stato della spedizione, ma ciò non è stato possibile per cui la situazione si è rilevata più difficile.
> Credo quindi sarebbe interessante probabilmente dedicare un laboratorio, verso la fine del corso, a scrivere una piccola applicazione usando API di aziende importanti, giusto per dare idea agli studenti di cosa si possono aspettare e per magari dare loro un piccolo esempio di cosa potranno fare nel progetto.

Appendice

Guida utente

Aprendo il programma ci si trova nella pagina movimenti, nella tabella sono visualizzati i trasferimenti registrati. Attraverso i bottoni in alto a destra è possibile aggiungerne di nuovi, convertire l'elenco a PDF o, dopo aver selezionato uno in tabella, eliminarlo. Entrambe le azioni provocheranno dei cambiamenti nella schermata depots. Quando si aggiunge un movimento la scelta è interamente guidata: dopo aver selezionato il sender, per gli altri campi ci sarà una scelta limitata dalla selezione, e così via per i campi successivamente selezionati.

Le altre schermate funzionano similarmente, un'eccezione è la schermata Depots che ha una lista di bottoni rappresentante i depot registrati, cliccando i quali sarà visibile il relativo elenco di libri contenuti. Inoltre, le schermate Customers e Books permettono la modifica degli elementi direttamente da un campo a piacere della tabella. Per confermare la modifica è necessario premere invio.

In fondo alla maggioranza delle schermate di ricerca per filtrare il contenuto della tabella secondo le necessità.

I bottoni in alto a destra, permettono di raggiungere la pagina delle statistiche, quella per l'invio delle email e permettono di visualizzare gli autori del programma. La pagina delle statistiche ha un funzionamento simile a quella dei depot, è possibile scegliere la statistica da visualizzare attraverso i bottoni blu e l'anno interessato attraverso il menu a tendina in alto a destra. La pagina delle email è stata pensata per inviare i PDF convertiti in precedenza ad un eventuale commercialista, per cui bisogna selezionare un allegato dalla lista e riempire i campi.

Il programma non necessita di salvare tramite icone o collegamenti in quanto il salvataggio avviene automaticamente come in Google Documenti.