20/08/2016

Library Management

Applicazione gestionale di una libreria

Mattia Rovinelli
Lorenzo Ceccarelli
Erik Maraldi

Sommario

Capitolo 1

1 Analisi

L'obbiettivo del nostro progetto è la realizzazione di un software sviluppato in linguaggio Java che prevede come utilizzatori le librerie (Mondadori ,Feltrinelli) .Il software è dunque un gestionale che permette di gestire tutte le principali azioni che si svolgono in una libreria partendo dalla registrazione di un nuovo libro appena uscito, fino a svolgere il ciclo completo di vendita di un libro ,comprese tutte le azioni che gli susseguono (decrementazione delle quantità nello shop, implementazione della tessera se il cliente è registrato).

Di seguito analizzeremo in modo sempre più dettagliato ogni aspetto logico ed implementativo del nostro software, servendoci di diagrammi specifici, in modo da rendere più visibile ed immediata la comprensione delle relazioni che legano gli elementi portanti del nostro progetto.

1.1 Requisiti

Il gestionale Library Management ha diverse funzionalità , per sopperire al basso livello di difficoltà e alla mancanza di librerie esterne. Le funzionalità del gestionale sono varie: trattare il ciclo di vendita di un libro, registrare un nuovo dipendete, acquistare le nuove uscite di libri e altro. Tutte queste azioni memorizzano dati, la memorizzazione avviene attraverso l'utilizzo di file.dat

1.2 Elementi Negativi

Library Management non è un software particolarmente complesso, ma è un software carico di funzionalità con un intreccio abbastanza contorto, dove è facile sbagliare il richiamo e la collocazione delle funzioni di una classe. I problemi che ci aspettiamo di affrontare sono vari, ma principalmente sono due: la lavorazione dei nostri dati sulle nostre liste e sui nostri file, e la costruzione delle funzionalità del nostro progetto. Questi 2 casi che consideriamo le principali difficoltà vengono affrontate attraverso vari controlli.

1.3 Analisi e modello del dominio

Library management dovrà essere in grado di svolgere le principali operazioni per una libreria. In primis loggare e registrare un dipendete per poter accedere alle funzionalità del software. Successivamente comprare le nuove uscite di un libro, quindi l'aggiunta di un nuovo libro nel magazzino e nello shop. Sarà poi possibile gestire la vendita di un libro, o più libri, compresa l'incrementazione/ decrementazione dallo shop e dal magazzino, la possibilità di registrare un abbonamento , e la registrazione di tutte le emissioni fiscali per un resoconto mensile o annuale degli incassi.

Capitolo 2

2 Architettura

Dopo aver svolto una prima analisi del software e definito i requisiti che dovrà implementare, ci concentriamo sulla sua architettura. Per architettura intendiamo la disposizione di elementi quali classi e interfacce che mediante la loro interazione, gestiscono le varie parti del programma. Per la realizzazione di Library management abbiamo sfruttato ampiamente il pattern MVC, consentendoci di separare i diversi aspetti del Model, della View e del Controller, puntando così ai 4 principi della **programmazione ad oggetti**(Incapsulamento, Ereditarietà ,Polimorfismo ,Delegazione) e ai principi di progettazione (anticipazione dei cambiamenti, separazione degli argomenti, modularità, astrazione, generalità).

Ogni Componente del gruppo si è dedicato in modo indipendente allo sviluppo di una delle parti del software quali *model, view e controller.*

- Model
  - Nel model troviamo le classi che rappresentano gli oggetti.
  - Le classi create vengono utilizzate in tutto il software e compongono la struttura dei dati utilizzati.
  - Nel model del programma sono state create 2 tipi di classi una esterna a core(che rappresentano gli oggetti con tutte le sue caratteristiche e la classe stream per leggere e scrivere) ed interna a core(che contiene tutte le classi che

comprendono una mappa di oggetti ,quindi una sorta di cuore delle informazioni).
    - Le classi nel model hanno delle caratteristiche come un oggetto vero e proprio (persona ha i vari dati personali come nome cognome ecce ecc).

❖ View
- Nel folder **view** si trovano tutte le interfacce e le relative implementazioni delle view del programma, la view principale è quella MainViewImpl, che imlementa l'interfaccia MainView, tutte le altre view seguono lo stesso schema logico, interfacca-implementazione-interfacciaObserver.
- La creazione delle interfacce è stata scelta per il fatto che in futuro sarebbe stato più semplice aggiungere eventuali modifiche.
- Per ogni view sono state create anche le intefacce per gli Observer contenuti nella cartella **observer** contenuta all'interno della cartella **view,** Le interfacce create serviranno poi per implementare il "collegamento" tra le **view** ed il **controller** in quanto le **view** foniscono una piattaforma di input ed output delle informazioni elaborate nel controller.

❖ Controller
- Il Controller rappresenta la struttura fondamentale dell'applicazione, al suo interno troviamo le classi che legano view e model.
- Ogni controller implementa la relativa classe observer contenuta all'interno della view, permettendo così alla view di controllare se sono avvenuti cambiamenti ai dati attraverso il controller.

2.1 Design Dettagliato

Il nostro software Library Management è suddiviso in 6 package, ognuno dei quali contiene file relativi ad una specifica parte di progetto.

- Controller: package che contiene la parte del controller
- Main: Contiene il **Main** che gestisce l'avvio del programma
- Model: package che contiene le classi che rappresentano gli oggetti di una libreria
- Model.core: contiene le classi contenenti mappe e liste degli oggetti utilizzati
- View: contiene le interfacce e le loro implementazioni
- View.observer: contiene tutte le interfacce degli observer relativi alle view

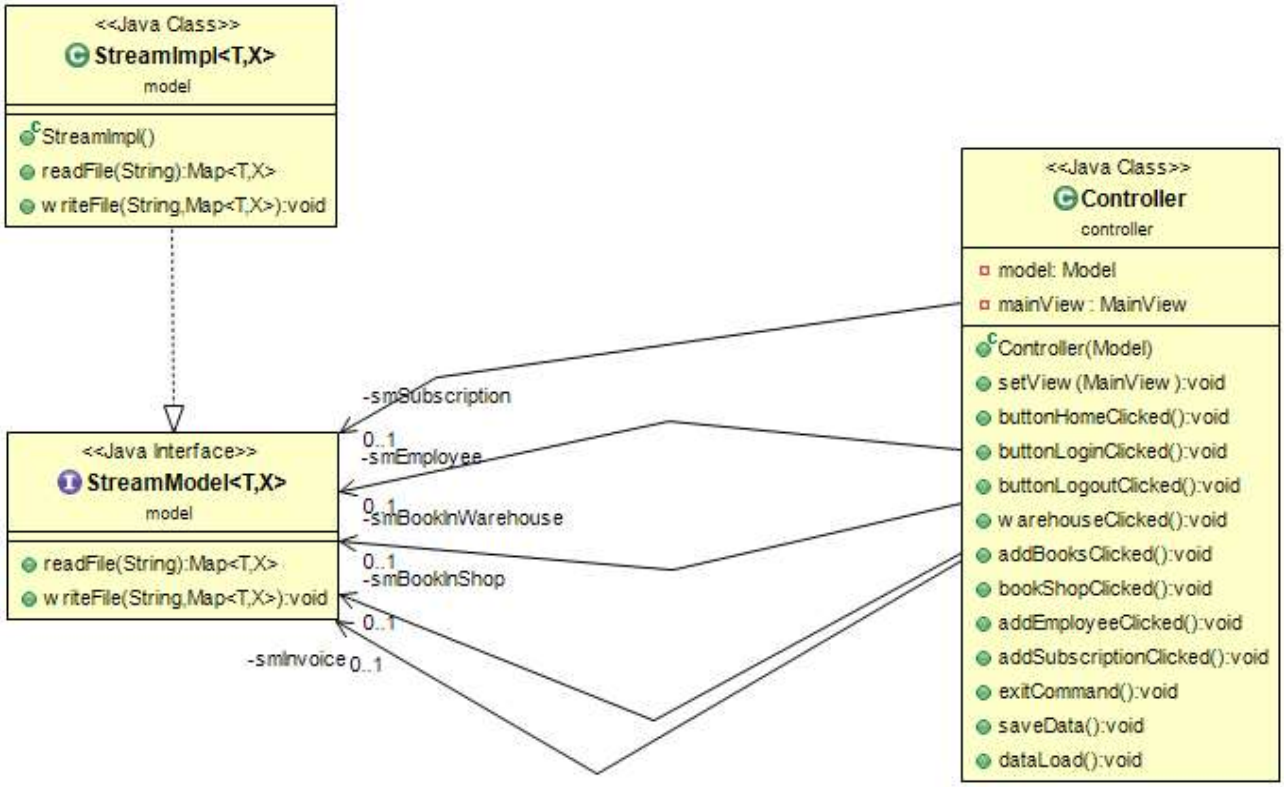
Al fine di mostrare dei diagrammi UML più chiari e leggibili abbiamo
preferito procedere illustrando prima le funzionalità del software senza de-

scriverle nei minimi dettagli, in modo da mostrare le relazioni tra il model, la view e il controller.

2.2 Azioni Maraldi Erik

2.2.1 Lettura e scrittura file

Le operazioni di lettura e scrittura su file vengono effettuate attraverso la classe StreamImpl. All'apertura del programma vengono caricati, quindi letti, i file contenenti i dati dell'applicazione, mentre alla chiusura vengono salvati con le relative modifiche effettuate durante l'esecuzione dell'applicazione. Sarà poi affidato al controller il compito di gestire la scrittura e lettura su file.



2.2.2 Inserimento

L'inserimento permette di aggiunger e salvare su file (una volta chiuso il programma) un nuovo dipendente (immagine 2.1), un nuovo libro (immagine 2.2), o un abbonamento (immagine 2.3). Tutto ciò avviene compilando gli appositi campi dei relativi pannelli AddEmployeePanelImpl, AddBookPanelImpl e AddSubscriptionPanelImpl. Prima di eseguire il salvataggio su file, attraverso la classe StreamImpl, vengono eseguiti gli opportuni controlli sugli input.

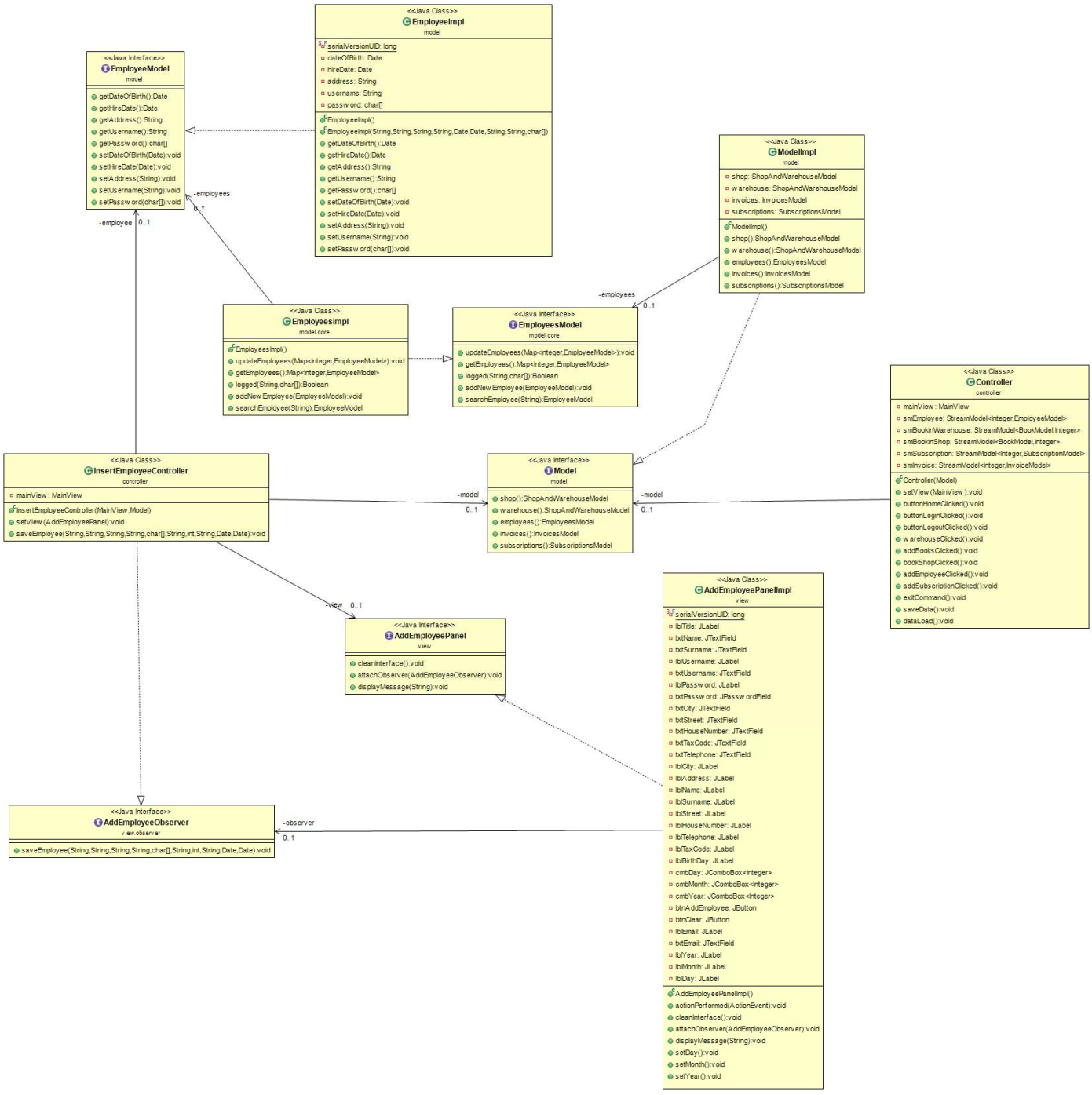


Figura 2.1: UML che rappresenta le relazioni tra le classi e le interfacce che permettono di aggiungere un nuovo dipendente.

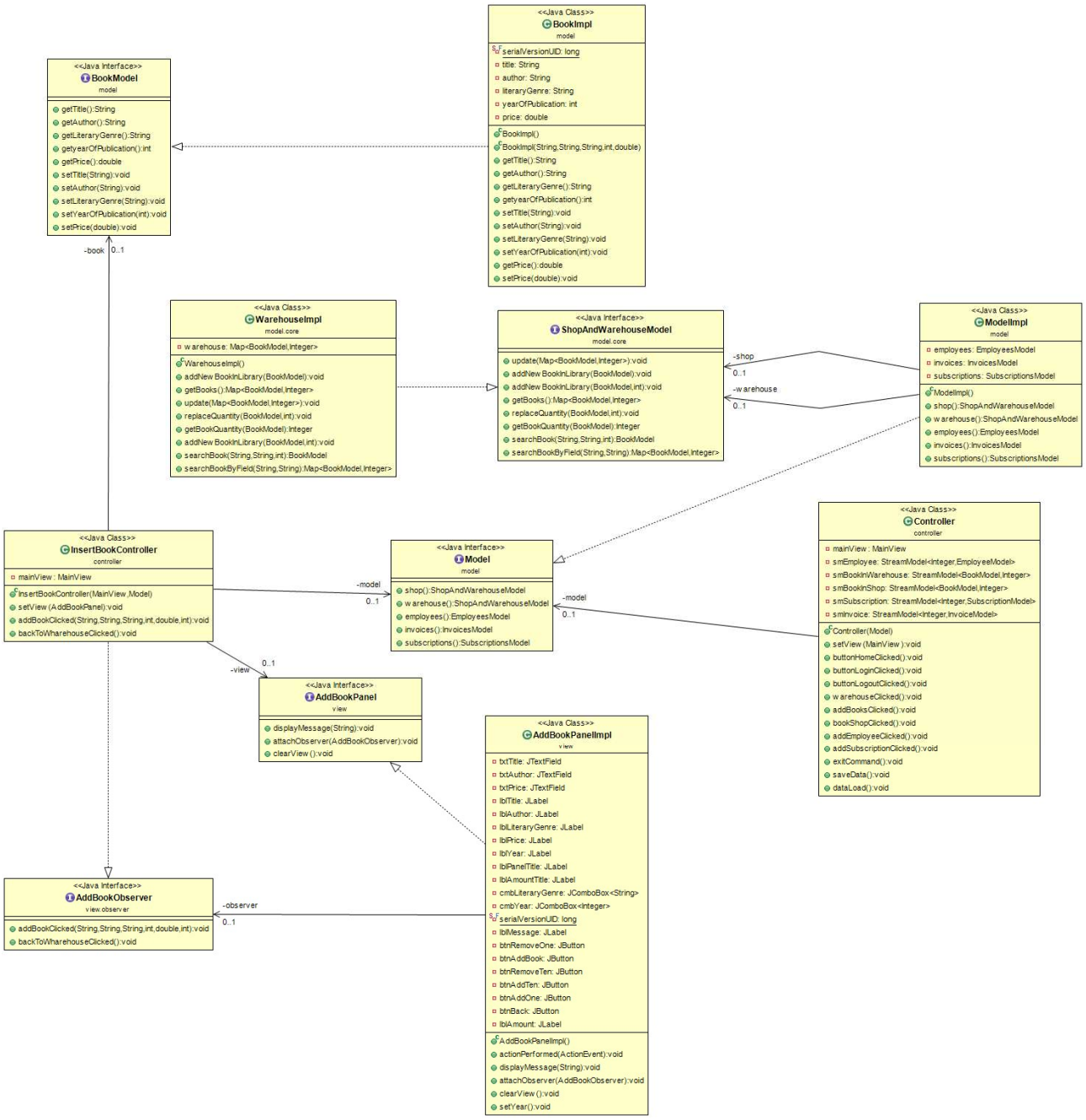


Figura 2.2: UML che rappresenta le relazioni tra le classi e le interfacce che permettono di aggiungere libro.

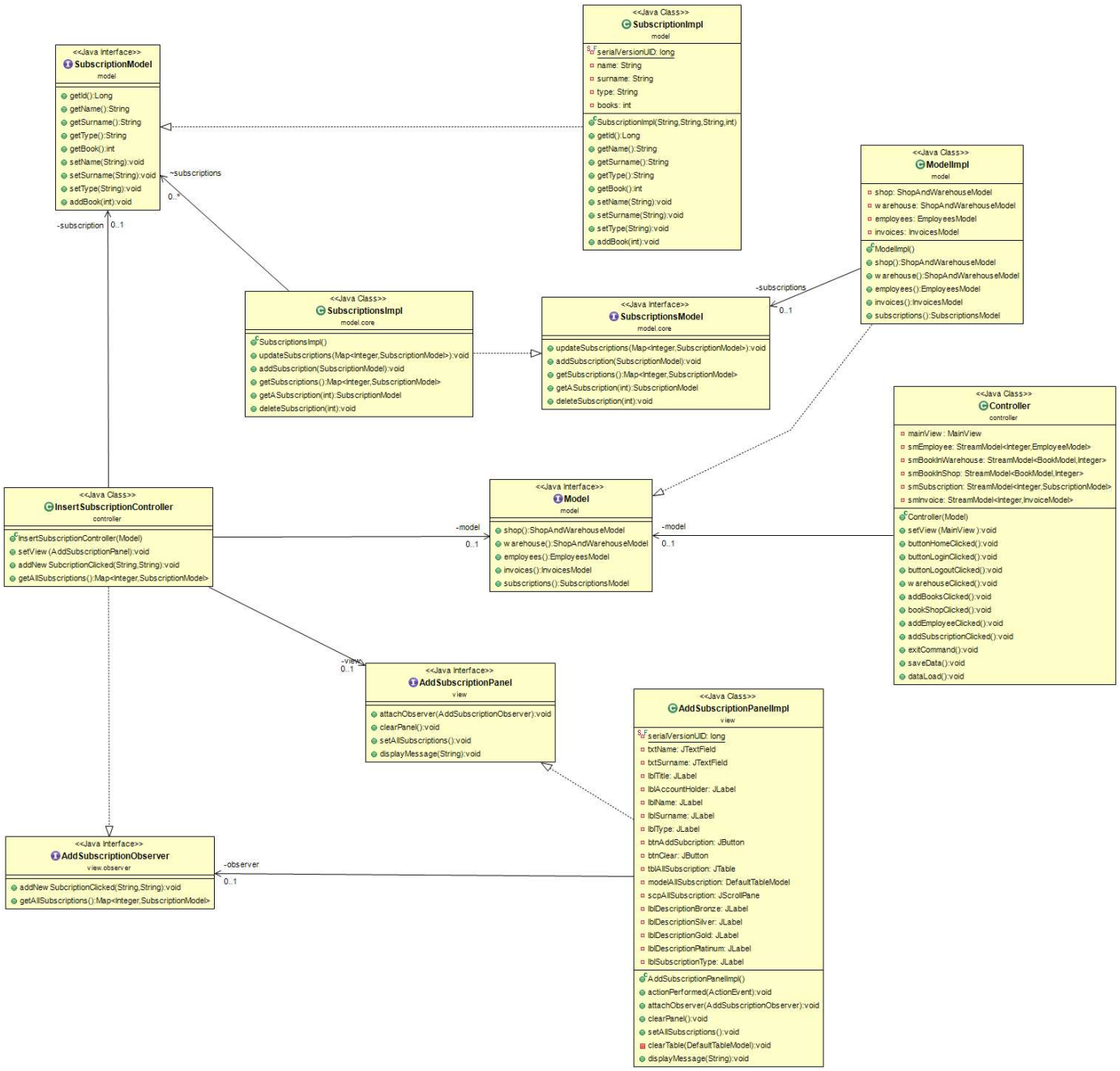


Figura 2.3: UML che rappresenta le relazioni tra le classi e le interfacce che permettono di aggiungere un abbonamento.

2.2.3 Gestione libreria

La gestione della libreria permette di ricercare un libro e di venderlo a un cliente (immagine 2.4) attraverso il pannello BookshopPanelImpl. La ricerca può essere effettuata per titolo, autore o anno, permettendo così al dipendente di avere una ricerca dinamica del prodotto richiesto dal cliente.

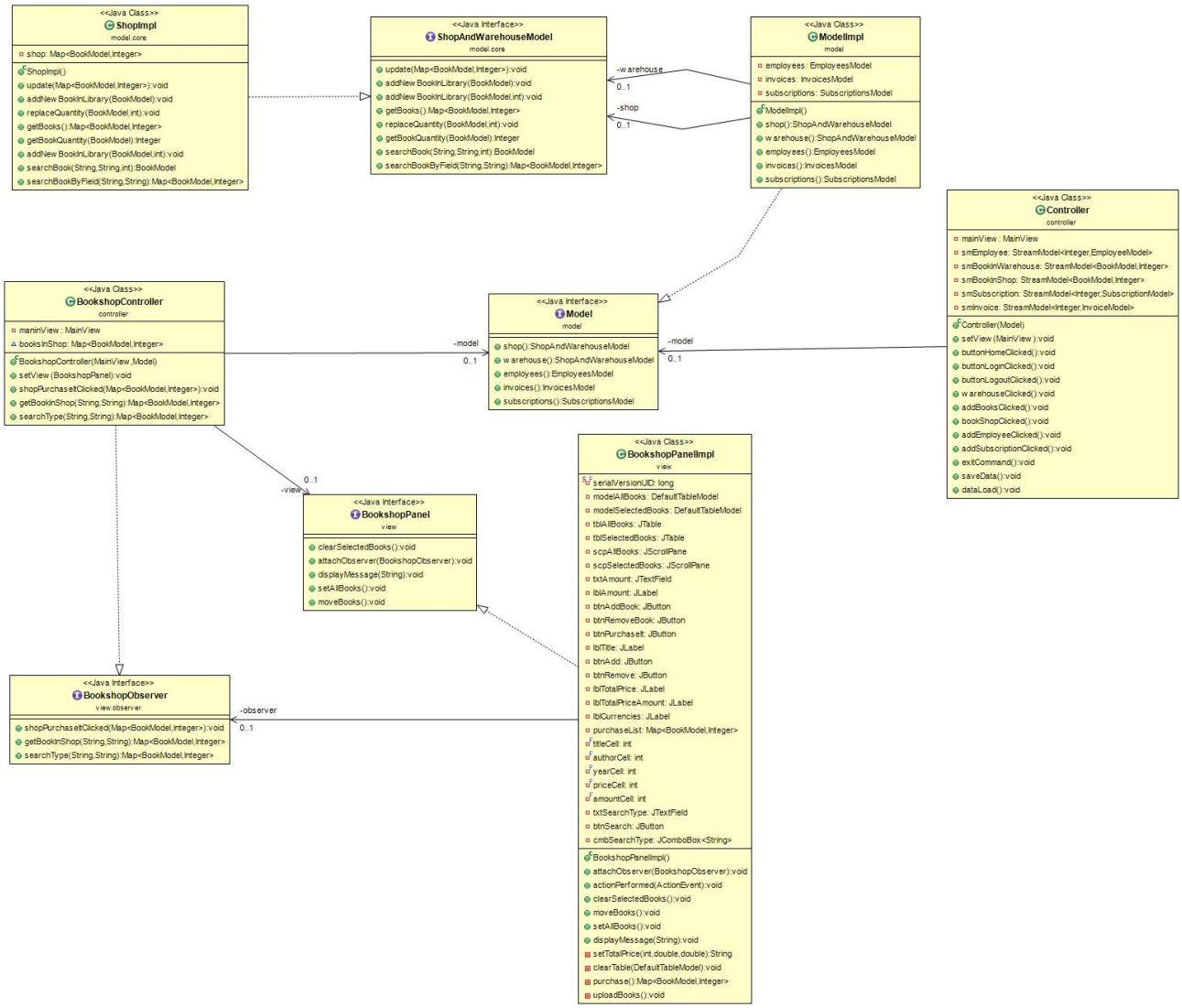


Figura 2.4: UML che rappresenta le relazioni tra le classi e le interfacce che permettono di gestire la libreria.

2.2.4 Salvataggio scontrino

Il salvataggio dello scontrino è il riepilogo dei libri acquistati da un cliente con la relativa quantità. Dopo aver selezionato i libri da vendere (da parte del dipendente) nella gestione della libreria (vedi sopra), potrà procedere alla conferma dell'acquisto sancendo così la fine della vendita (immagine 2.5). Il pannello che si occupa della conferma acquisto (con il relativo salvataggio su file) è ReceiptPanelImpl.

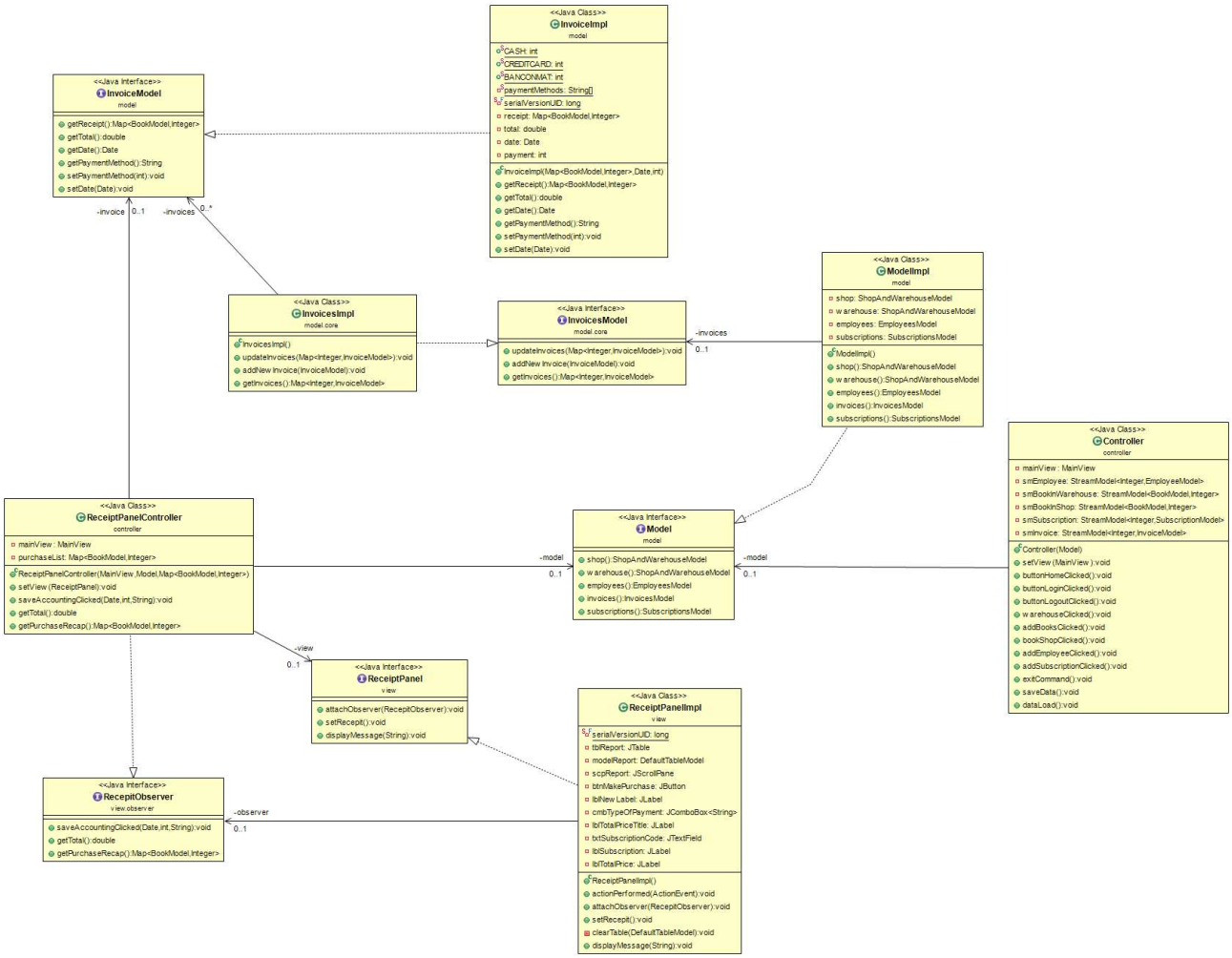


Figura 2.5: UML che rappresenta le relazioni tra le classi e le interfacce che permettono di salvare uno scontrino (riepilogo di quello che è stato acquistato da un cliente).

2.2.5 Gestione magazzino

La gestione del magazzino permette di controllare quanti libri sono ancora disponibili e di trasferirli in negozio qualora ne sia necessario, il tutto viene effettuato attraverso il panello WarehousePanelImpl (immagine 2.6). Se un libro è esaurito o comunque le scorte sono scarse, il dipendete potrà aggiungere la quantità comprata da un ipotetico fornitore senza aggiungere un nuovo libro da zero.

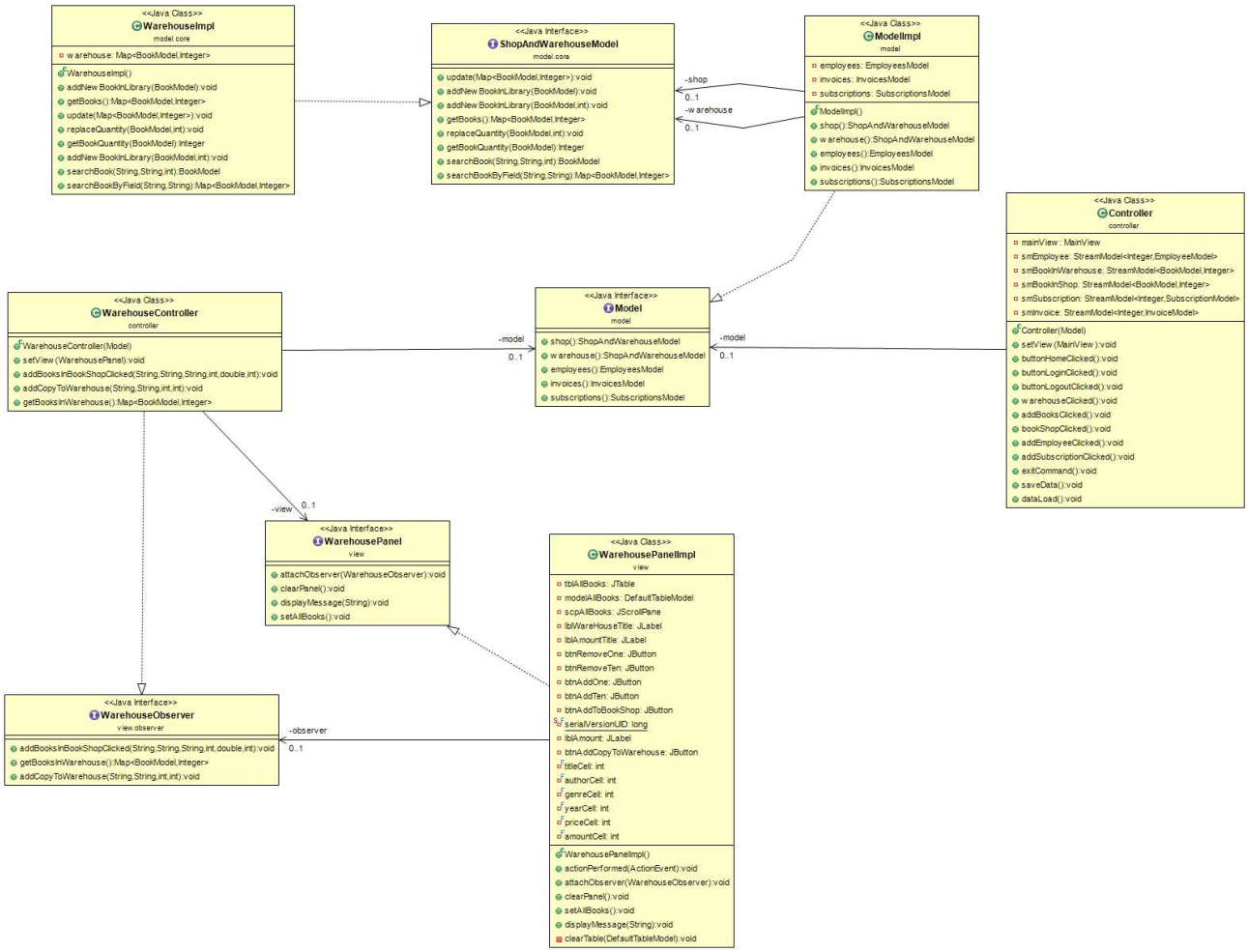


Figura 2.6: UML che rappresenta le relazioni tra le classi e le interfacce che permettono gestire il magazzino.

2.2.6 Login

Il login permette al dipendente di entrare nell'applicazione, per farlo dovrà fornire al programma username e password, in caso di avvenuto successo si avrà accesso all'applicazione (immagine 2.7).

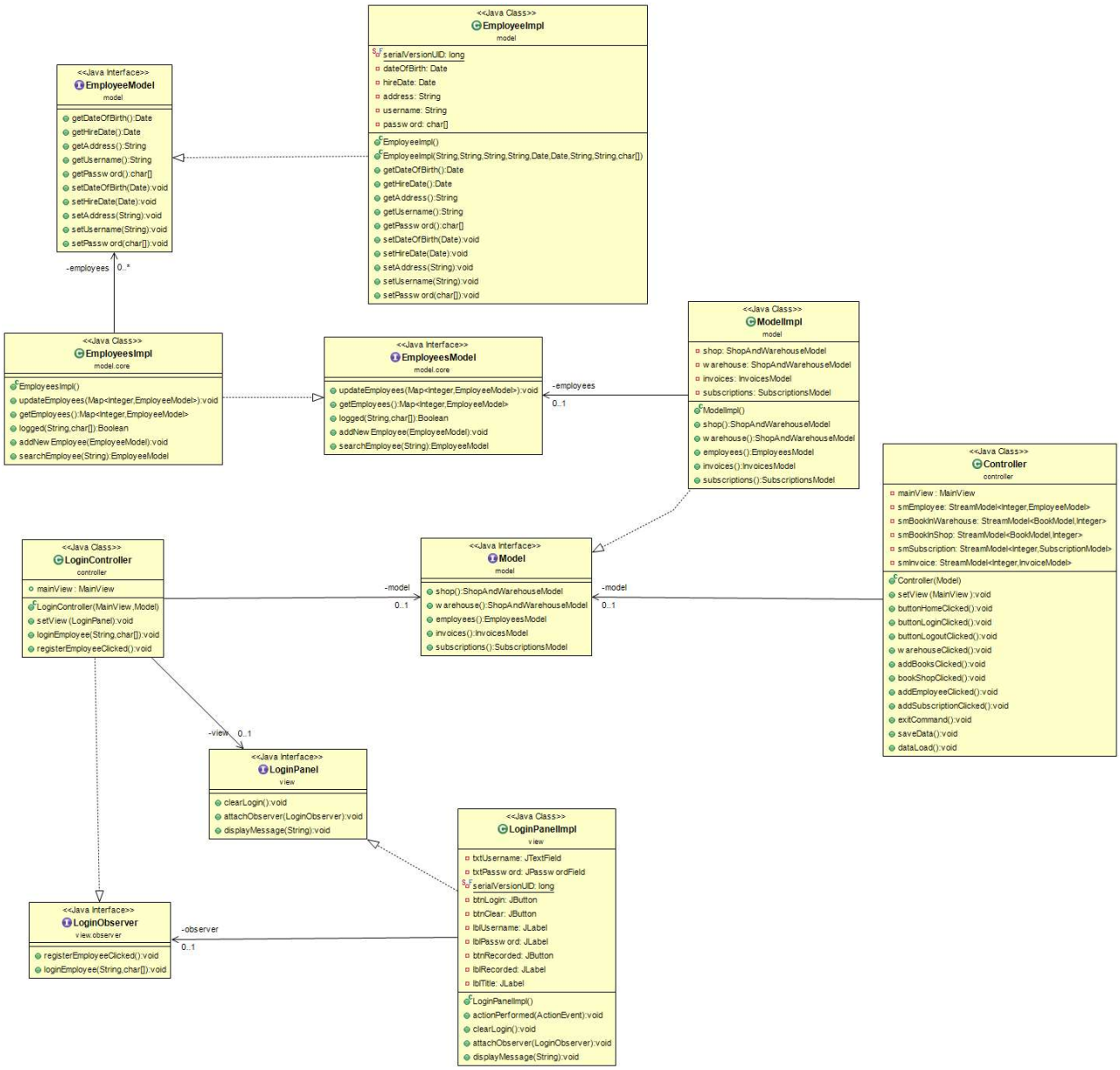


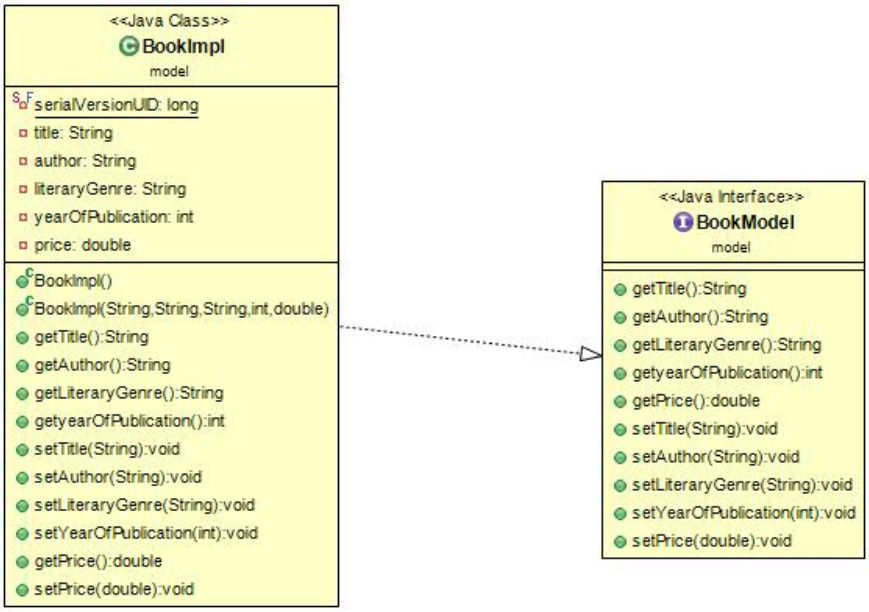
Figura 2.7: UML che rappresenta le relazioni tra le classi e le interfacce che controllano e permettono al dipendente di effettuare il login nell'applicazione.

2.3 Design dettagliato: Model Mattia Rovinelli

In questo package vengono implementate tutte le classi che vengono utilizzate nel programma, le classi sono di due tipi: quelle interne al core e quelle all'esterno del core.

2.3.1 Classi rappresentati un oggetto

Esterne al Model.core ci sono tutte le classi che rappresentano un oggetto reale come lo scontrino, il dipendente o il libro. Tutte le classi esterne al core, tranne lo stream e il model hanno la stessa morfologia, cioè sono strutturate nello stesso modo.

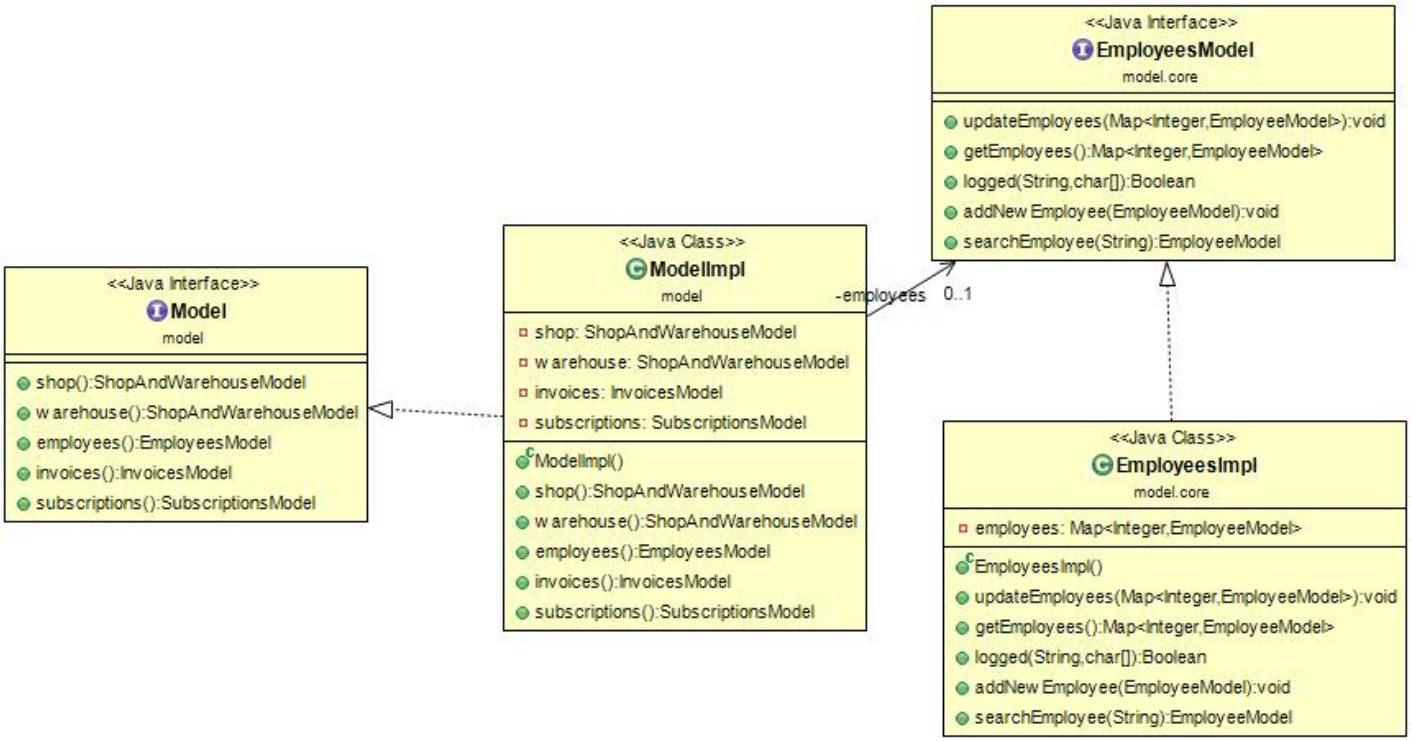


L'immagine sopra rappresenta la classe BookImpl con la sua interfaccia e rappresenta l'oggetto reale di un libro. Ogni suo campo come title, author rappresentano i dati di un libro mentre in verde abbiamo i metodi che il controller e la view possono utilizzare per estrapolare o inserire dati .

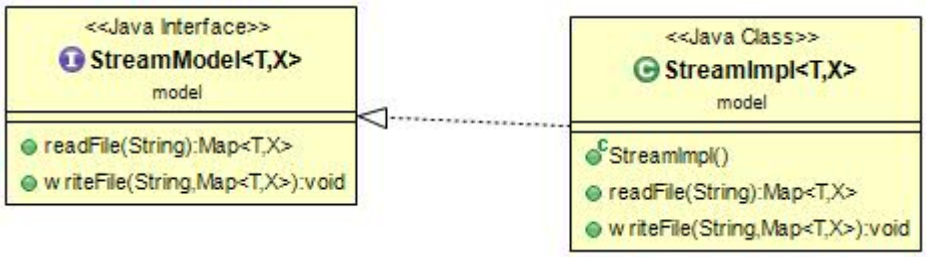
2.3.2 Classi non rappresentati un oggetto

Le uniche 2 classi che non rappresentano un oggetto reale, che sono esterne al Model.core sono il model e lo stream. Queste due classi sono classi utilizzate per elaborare dati. La classe model serve per collegare tutte le classi del core, quindi per elaborare tutte le mappe contenute nelle classi, mentre lo stream serve per scrivere/leggere dati, che nel nostro caso sono file.dat

L’immagine qui sottostante mostra la classe modelImpl collegata ad una delle classi del model.core. Tutte le classi del model.core sono collegate alla stessa maniera al model, infatti il controller per utilizzare i dati delle classi nel core, richiama le classi in egual maniera.



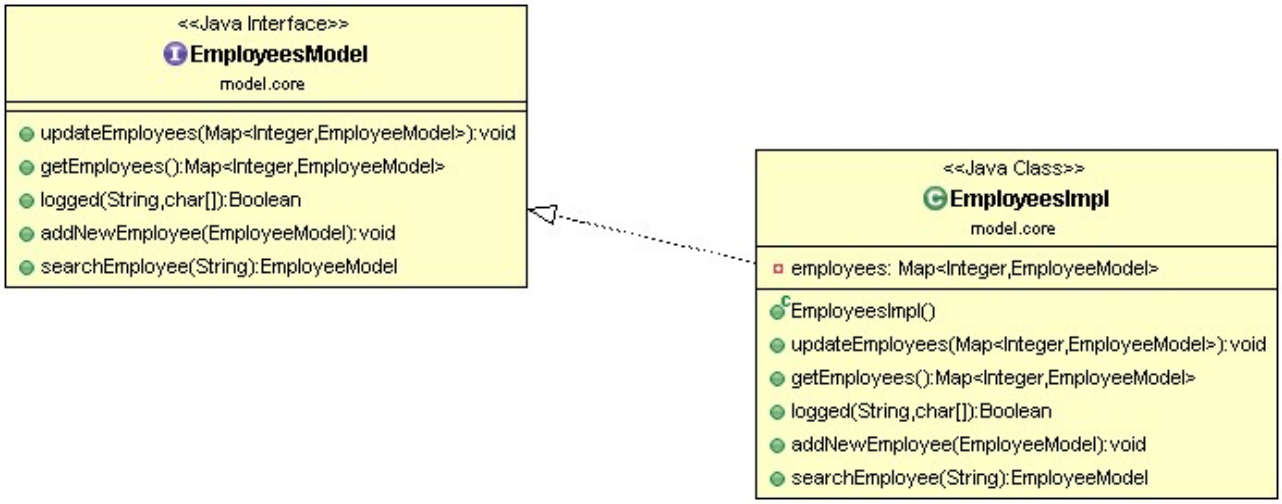
Un’altra classe che non rappresenta un oggetto e che viene utilizzata per scrivere e leggere dei file e streamImpl rappresentata qui sotto.



2.3.3 Classe core il cuore del Model

Nella classe core sono contenute le classi che vengono utilizzate per qualsiasi cosa, dal salvataggio dei dati, fino alla lettura di una mappa completa e la ricerca di uno specifico dato. Qui sotto è visualizzata la classe contenente una mappa di dipendenti, e relative funzioni strutturate apposta per il nostro progetto a richiesta del controller

e della view, e ogni classe contenuta nel core è strutturata in egual maniera.



2.4 Design dettagliato: View Ceccarelli Lorenzo

In questo package vengono implementate tutte le classi e interfacce relative

al layout di tutti i componenti grafici del software. La base di tutto il software realizzato è composta da un JFrame 900x700 a cui è collegato un JPanel nel locazione top che rimarrà visibile durante tutto il funzionamento del programma e che permette di tornare alla schermata di home e di effettuare login e logout. Sia il JFrame che il JPanel (chiamati rispettivamente MainViewImpl e NorthPanelImpl) implementano le loro interfacce (del resto come tutti gli altri JPanel presenti).

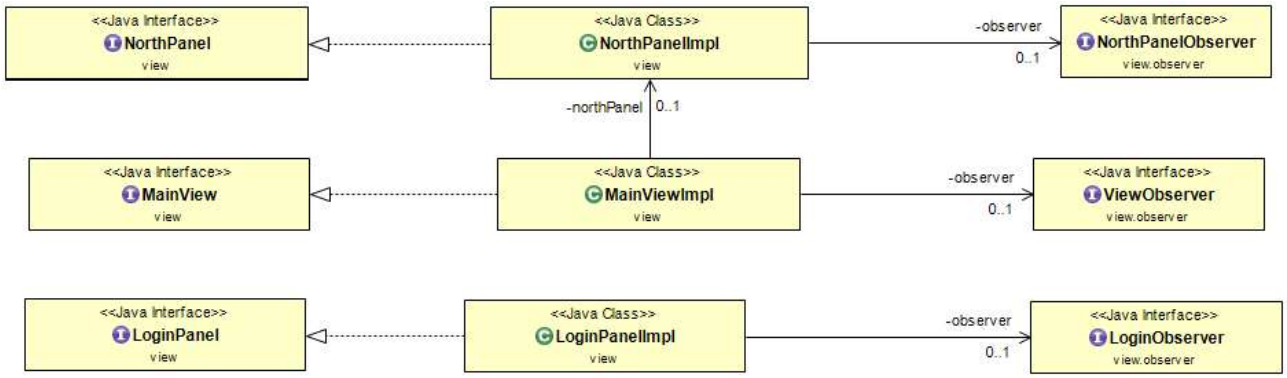
Tramite il metodo replaceMainPanel all'interno del MainViewImpl che permette di cambiare i vari JPanel in base alle richieste dell'utente. Le schermate di cui si compone il software sono rispettivamente:

base: come descritto sopra è la base di tutto il programma in alto contiene un pannello e al centro ci sta un altro pannello adibito a contenere una immagine per fare lo sfondo, quest'ultimo pannello si chiama imagePanel ed è l'unico JPanel ha non avere una propria interfacci per il semplice fatto che non deve implementare nessuna funzionalità deve solamente contenere una immagine (MainViewImpl)

pannello del login/logout: questo JPanel è l'unico che rimane fisso e non cambia mai permette come già detto di effettuare login e logout ed il ritorno alla schermata home, cioè il negozio (NorthPanelImpl)

la login: dove un utente può effettuare l'accesso tramite le proprie credenziali (LoginPanelImpl)

In questo diagramma viene illustrato il primo approccio da parte di un utente subito dopo l'avvio,



l'utilizzatore si troverà davanti il Main (JFrame) con al suo interno il northPanel nel quale potrà compiere solo l'azione di accedere alla view del login LoginPanelImpl (JPanel)

il negozio: dove si possono visualizzare tutti i libri pronti alla vendita anche tramite ricerche e selezionare i libri che effettivamente saranno venduti (BookShopPanelImpl)

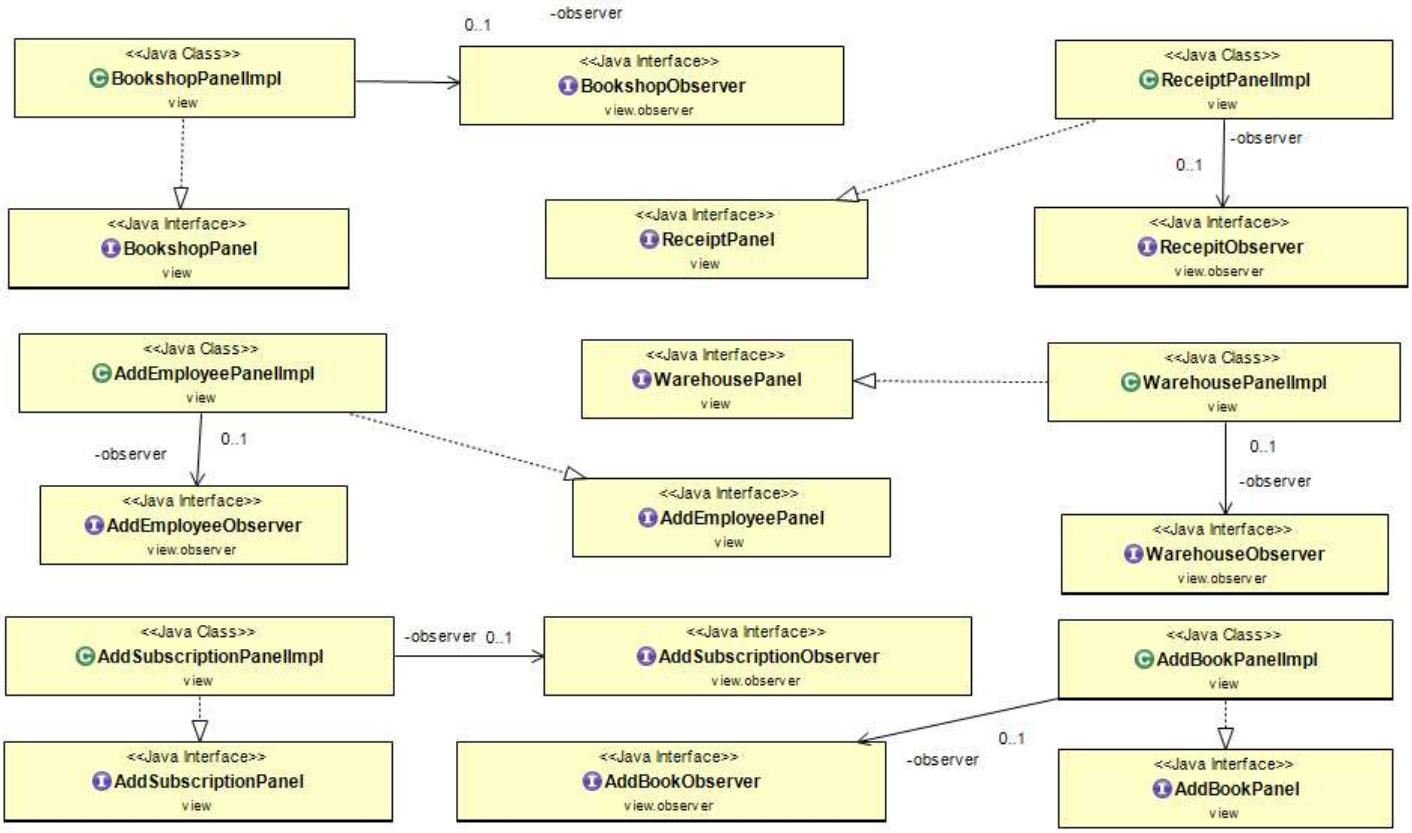
lo scontrino: dove effettivamente avviene la vendita dei libri selezionati nel negozio (ReceiptPanelImpl)

il magazzino: dove c'è l'elenco di tutti i libri presenti nel suddetto magazzino appunti e dove si possono ordinare scorte dei libri presenti e trasferirli nel negozio(WarehousePanelImpl)

l'aggiunta di un dipendete: si può aggiungere un nuovo dipendente capace di effettuare la login e quindi di accedere alle funzionalità del software (AddEmployeeImpl)

la gestione degli abbonamenti: in questa schermata è possibili accedere all'elenco di tutti gli abbonamenti presenti e si ha la possibilità di aggiungerne di nuovi, funzionalità da cui prende il nome il JPanel (AddSubscriptionPanelImpl)

aggiunta di un nuovo libro: questa schermata permette di aggiungere un libro non presente nel magazzino abbiamo optato per una Jpanel a parte per il fatto che richiedeva troppo spazio aggiungere tutte nel magazzino cosi abbiamo aggiunto una schermata a sé stante (AddBookPanelImpl).



Nel diagramma che segue sono rappresentate tutte le altre view collegate con le relative intefacce, sia quelle della view stessa che degli observer.

Tutte le interfacce presenti hanno lo stesso nome delle classi che le implementano con la differenza che non hanno nella parte finale del nome **Impl,** mentre per quello che riguarda gli observer hanno un nome molto simile alle view che li utilizzano e con la parte finale del proprio nome che termina sempre con **Observer**

2.5 Design dettagliato: Controller Maraldi Erik

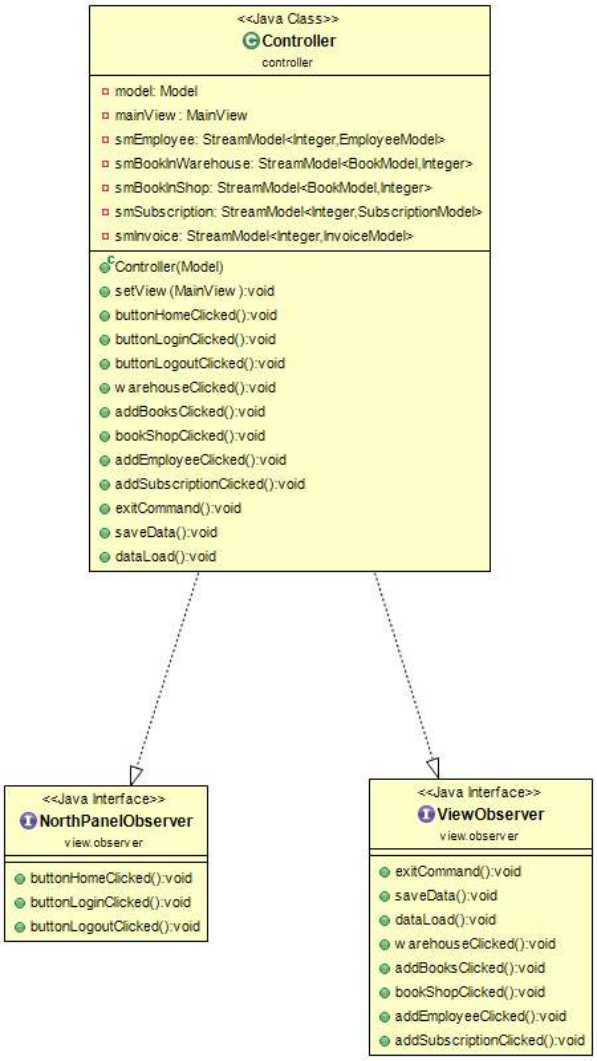
Nel package controller, sviluppato da Erik Maraldi, tutte le classi svolgono un ruolo fondamentale per il corretto funzionamento del programma in quanto si occupa di gestire le operazioni dell'applicazione.

Pattern Observer

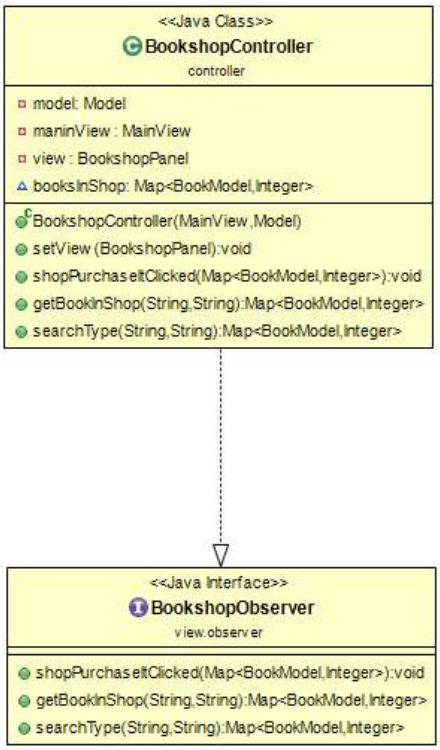
Il pattern observer viene utilizzato quando oggetti diversi devono conoscere i cambiamenti di stato di un particolare oggetto. Nel caso del pattern MVC i soggetti sono: il Model, la View e il Controller. L'observer della view (interfaccia observer)

osserverà se avvengono cambiamenti ai dati da parte degli altri soggetti e potrà interrogare il soggetto, ricavandone le informazioni.

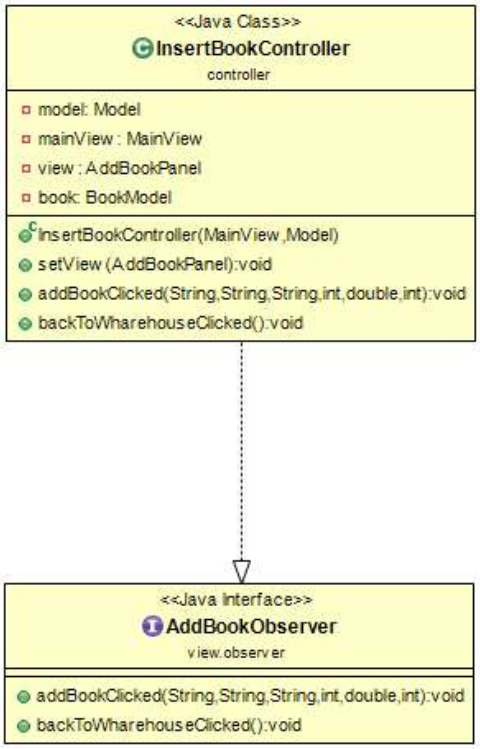
Entrando nel dettaglio abbiamo le classi Controller, BookshopController, InserBookController, InsertEmployeeController, InsertSubscriptionController, LoginController, ReceiptController, WarehouseController.



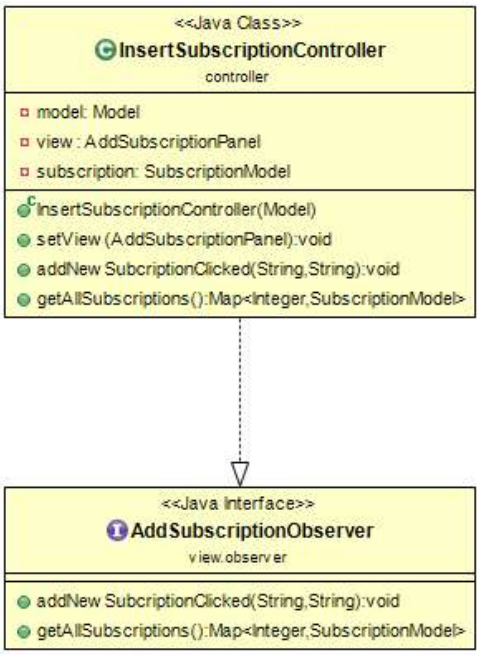
La classe Controller è la classe fondamentale, è quella che permette all'utente di interagire con l'applicazione.



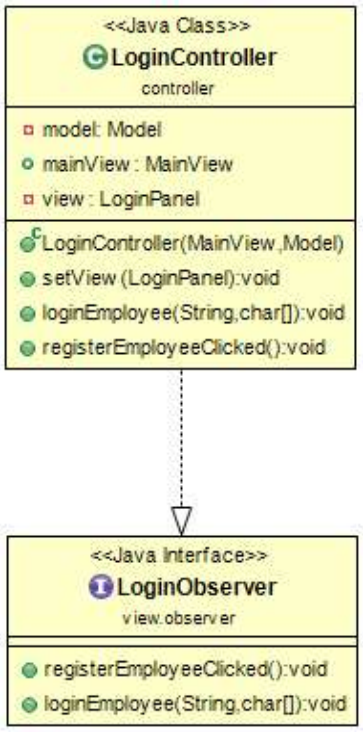
La classe BookshopController gestisce tutto ciò che riguarda la libreria, come la ricerca di un libro e la vendita del medesimo a un cliente.



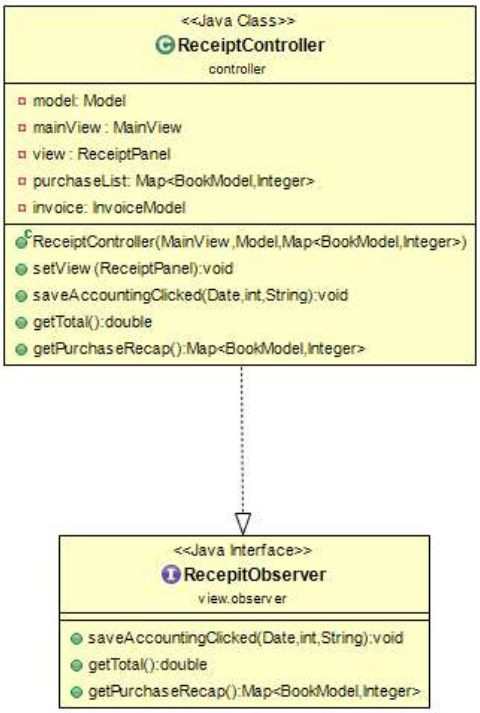
InsertBookController gestisce l'aggiunta di un libro nel magazzino.



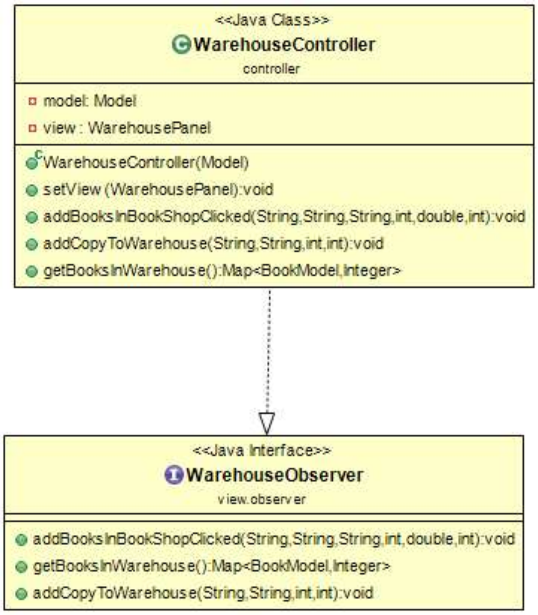
InsertSubscriptionController gestisce l'aggiunta di un abbonamento



LoginController gestisce tutto ciò che riguarda il login del dipendente



ReceiptController gestisce l'aggiunta di uno scontrino, attraverso i metodi presenti



WarehouseController gestisce il magazzino, permettendo l'aggiunta di libri in negozio.

Capitolo 3: sviluppo

3.1 Impostare il lavoro

Prima di buttarsi a capofitto nella programmazione a oggetti, abbiamo prima di tutto stilato un piano di lavoro, strutturato su vari schemi e tabelle di lavoro. Prima di tutto abbiamo creato uno schema dei casi d'uso (mostrato nel capito 5), che ci ha aiutato a capire come dovevamo strutturare le view, e soprattutto a capire che classi utilizzare e che come collegarle al controller. Stimate le azioni che il programma dovesse fare, abbiamo incominciato a costruire a strati il nostro programma, suddividendo ogni azione, e analizzando ogni suo comportamento. Così facendo abbiamo strutturato e suddiviso ogni problema che si è riscontrato entrando nella logica della programmazione ad oggetti.

3.2 Ciclo di vita del software

- **Definizione strategica**: definita inizialmente con schema dei casi d'uso e delle classi
- **Pianificazione:** definiti gli obbiettivi del nostro progetto (azioni svolte dal programma) pianificazione dei tempi(100 ore da strutturare dal 27/07/2016 al 28/07/2016) nessun costo del programma essendo noi i creatori di esso.
- **Controllo Qualità:** abbiamo predisposto che ognuno dei componenti del gruppo, supervisionasse il lavoro degli altri a livello di qualità di codice, ogni volta che si andava a creare una classe e la relativa interfaccia.
- **Realizzazione e collaudo :** realizzazione del nostro progetto e collaudo fatto in pochi giorni(relativamente uno) ,per via di un collaudo in massa di 3 persone e beta rilasciato ad altre persone per una prova reale ed effettiva del nostro progetto. È stato constato che la grafica è vecchia, che il programma è molto statico, con possibile implementazione di funzionalità nuove. Nel nostro caso non si avrebbe nessun problema a implementare per via della struttura che si è creata. Pregi: fluido, una schermata molto semplice e procedurale, impossibile sbagliare azione(vendere un libro, registrare un abbonamento). Difetti: schermata vecchia, in assenza di tempo e non avendo considerato il

design grafico ci siamo scordati di fare una grafica al passo con i tempi, molte funzionalità ma se dovesse essere implementato in una libreria mancano alcune funzionalità che potrebbero essere utili.

- **Installazione:** per una installazione più fluida nel computer, e a richiesta nei requisiti di consegna, abbiamo generato una Jar del programma.
- **Diagnosi Collaudo e Manutenzione:** come detto nel punto precedente abbiamo dato a 3 conoscenti (esterni alla scuola) il nostro programma per un test di qualche ora. Finito il test le richieste fatte da loro, per modificare o implementare nuove funzionalità non ci sono sembrate difficili da implementare, anzi con la struttura MVC abbiamo riscontrato che è molto semplice sia la ricerca dell'errore, che in caso futuro l'implementazione di funzionalità.

3.3 Test

Per i test abbiamo utilizzato 2 tipologie di test per via delle verifiche fatte. Abbiamo iniziato con la tecnica di verifica Statica(analisi), dove siamo andati ad analizzare la struttura del codice per vedere di sintetizzare il codice e per una maggiore fluidità. Successivamente abbiamo iniziato a usare le tecniche di verifica dinamiche(testing) dove controlliamo il nostro software nel suo comportamento.

Nelle tecniche di verifica dinamiche abbiamo utilizzato il **Testing in the large**, considerando il software come una scatola nera, ha lo scopo di verificare dell'output con specifici input forniti, l'insieme dei test è selezionato sulla base delle specifiche di progetto. Successivamente, nel caso di alcuni errori, si prende la funzione che si vuole far eseguire correttamente e gli si applica il **Testing in the small**, cioè test che hanno lo scopo di verificare singoli moduli o specifiche parti rilevanti per complessità o importanza.

Capitolo 4 Commenti finali

Il processo di sviluppo del nostro software è stato strutturato in moltissime parti seguendo i criteri e i principi dell'ingegneria software. Prima di incominciare a scrivere codice, per vari giorni ci siamo consultati moltissimo e scontrati altrettante volte per ideali e formalità. In principio lo scheletro del programma è stato creato da uno schema delle classi e dal diagramma dei casi d'uso, che ci ha aiutato a creare le View

da utilizzare, e le classi utilizzate nel nostro programma. Da quel punto in poi , il creare funzionalità è divenuta un' azione spontanea, ma inizialmente ,stenderla e ricercare la componentistica è stata meccanica quasi faticosa per poi divenire naturale. La suddivisione in MVC è stata ottima ,per quanto riguarda il lavoro di gruppo, per via che nessuno "montava sui piedi agli altri" , ma sempre in parallelo.

Per quanto riguarda la parte del Model fatta da Mattia Rovinelli, le uniche difficoltà riscontrate sono state alquanto dipendenti dal controller e dalla View, per via del fatto che bisognava creare le classi, e le relative funzioni, come le voleva il controller e la view, e quando controller e view erano in disaccordo su come doveva essere passato un dato, il model era fermo nella progettazione.

Durante la scrittura della parte della grafica , sviluppata da Lorenzo Ceccarelli, le maggiori difficoltà sono state riscontrate durante l'implementazione del BookShopPanelImpl in questo si tratta della parte grafica contente il maggior numero di controlli, la disabilitazione delle JTable in quanto si deve poter selezionare un singolo elemento ma non modificarlo mentre per il resto del programma non sono state riscontrate difficoltà insormontabili. Per lo sviluppo della grafica si è cercato di creare qualcosa di gradevole alla vista con colori che non stanchino gli occhi durante un lungo utilizzo, motivo per il quale si è optato per un colore azzurro, ad una interfaccia abbastanza intuitiva che permetta a un qualsiasi utente di riuscire ad orientarsi all'interno del software.

Durante la scrittura della parte relativa al controller, sviluppata da Erik Maraldi, le maggiori difficoltà sono state riscontrate nell'implementazione del BookshopController e WarehouseController, in quanto bisognava gestire le quantità e aggiornare il tutto una volta eseguite determinate azioni da parte del dipendente. Si è cercato di interagire il più possibile con gli altri due componenti del gruppo, dato che il controller "collega" la view e il model, quindi avevo bisogno di tenermi costantemente aggiornato su quello che veniva fatto (nella view e nel model) di mano in mano. Del resto non sono state riscontrate particolari difficoltà. Durante lo sviluppo del codice ho cercato di rendere il codice scritto il più pulito possibile.

Appendice A

Guida all'uso del programma e alle sue funzionalità.

A.1 Introduzione

Library management è un gestionale di una libreria che svolge operazioni di vario tipo, dalla sequenza di vendita di un libro, alla messa in vendita di un nuovo libro, fino alla registrazione di un nuovo dipendente. La sua interfaccia è molto semplice e intuitiva, fatta apposta per gli utenti che devono interagire con più clienti o una fila di persone indispettita dalla coda alla cassa.

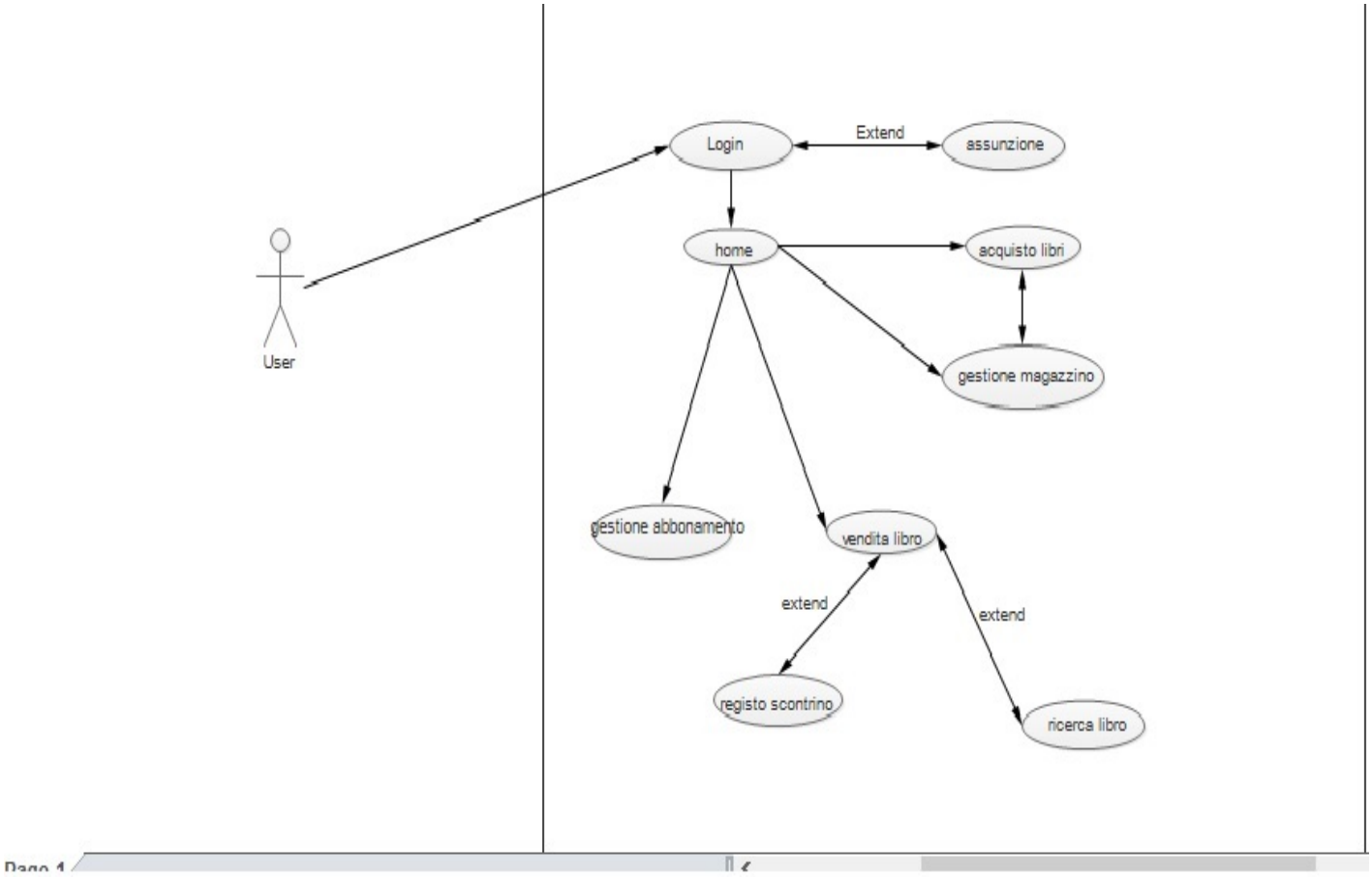
Come ogni programma che si rispetti, anche il nostro gestionale ha una guida all' uso schematizzata e commentata.

PS: Per effettuare l'accesso all'applicazione utilizzando la configurazione fornita, le credenziali sono: admin, admin.

A.2 Schema caso d'uso

Il caso d'uso rappresentato qui in figura è la rappresentazione della gerarchia delle nostre azione in modo generico e non specifico, infatti fa vedere alcune azioni

specifiche che estendono una interfaccia, ma le rimanenti sono interfacce generiche.



A.3 utilizzo di Library management

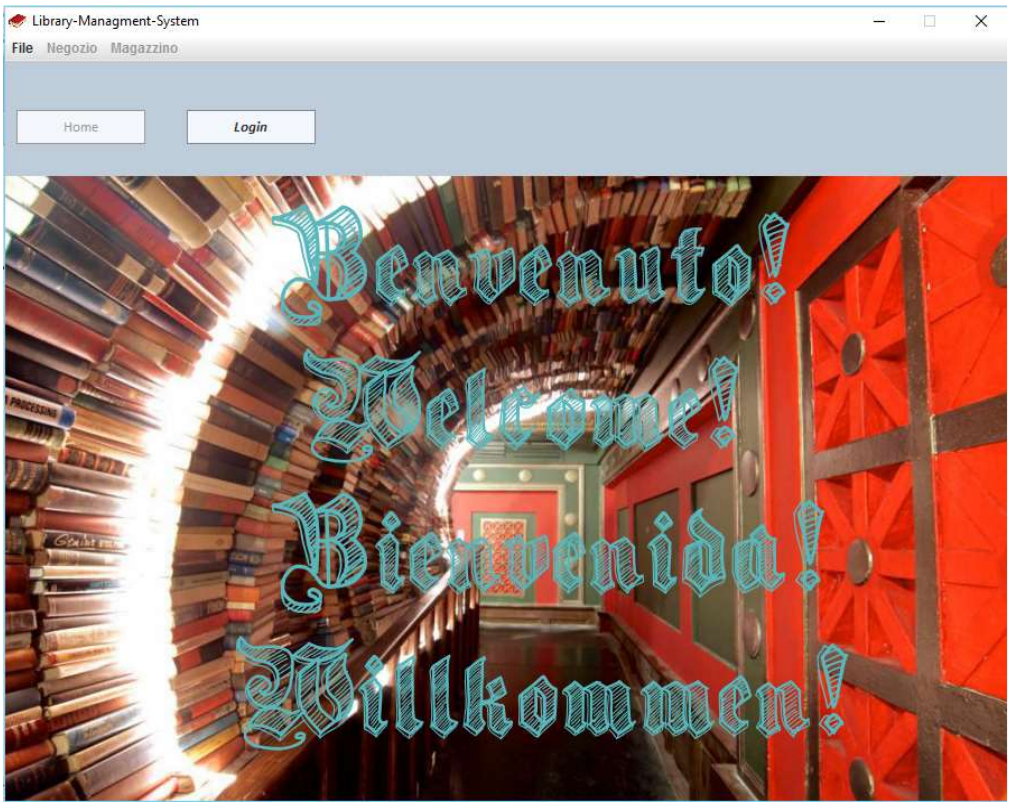
Descrizione:

Per l'utilizzo di questo software è possibile utilizzare una nostra configurazione iniziale con i 5 file .dat (file_configurazione.zip) che da la possibilità di essere utilizzato e testato in poco tempo, con utenti e libri già registrati.

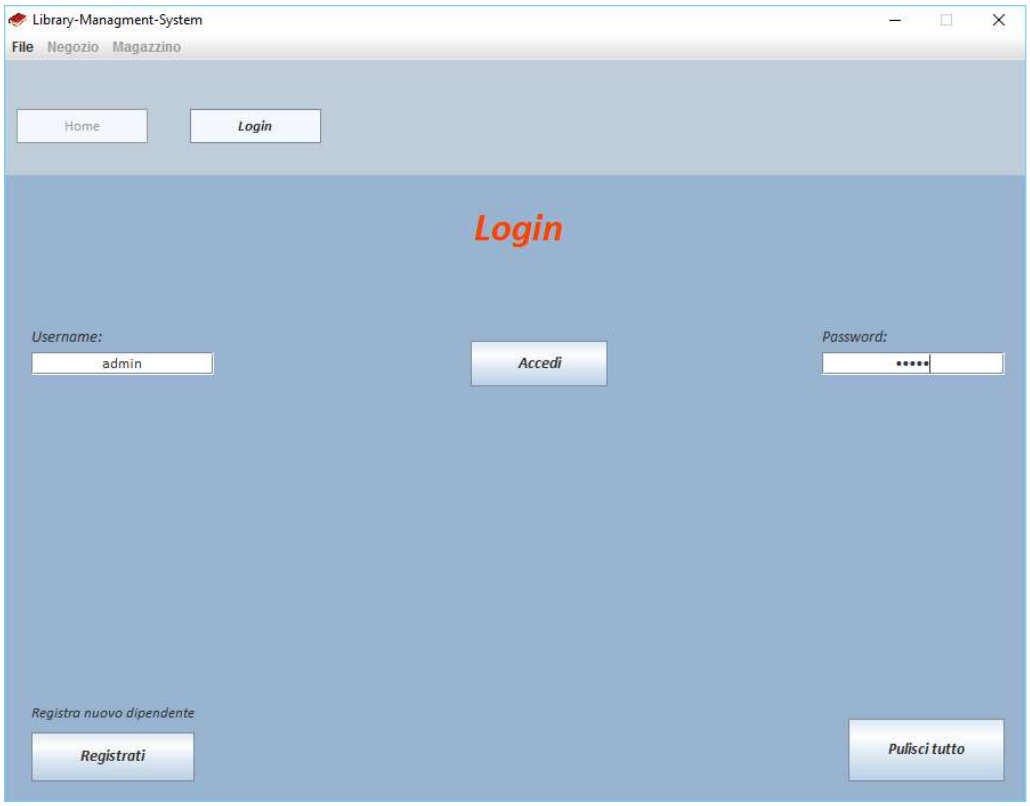
Nel caso in cui non si volesse utilizzare la configurazione già esistente il software è in grado di ricreare l'ambiente in cui operare, ricreando i file alla chiusura dell'applicazione.

Utilizzo:

Per inserire la configurazione scaricare il file .zip indicato sopra, estrarlo e copiare i file .dat all'interno di una cartella contenente il .jar.

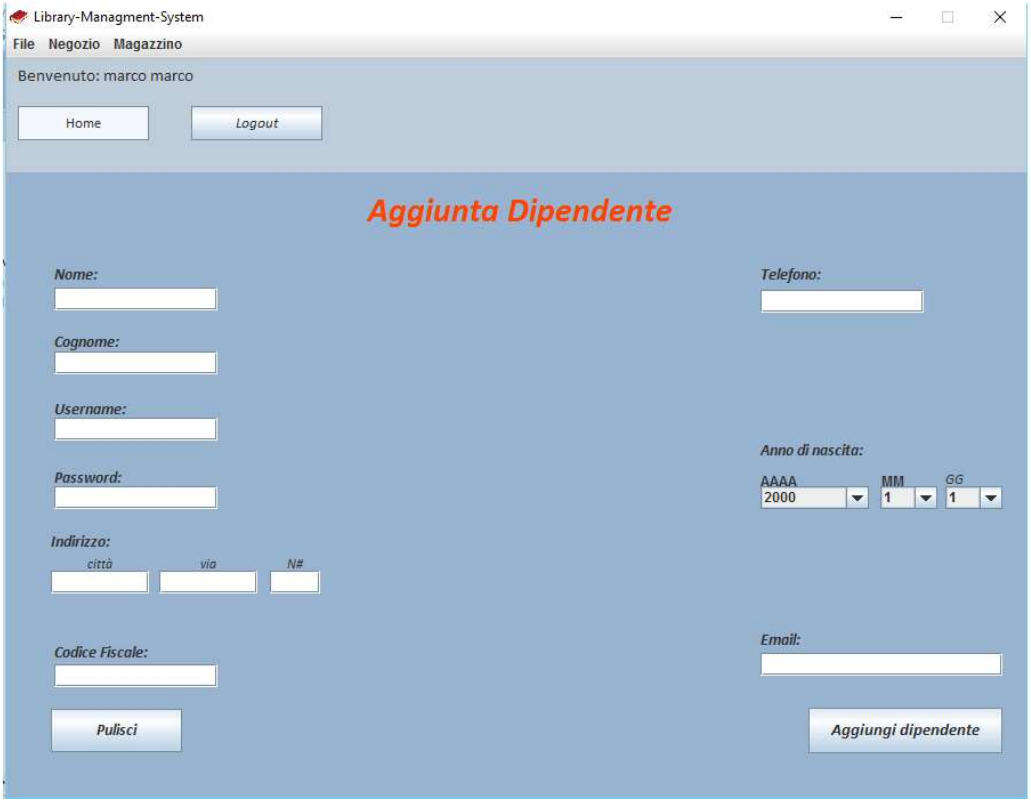


Nella prima schermata che si presenta ad un utente si può accedere solamente alla login di un dipendente attraverso l'apposito bottone situato in alto a sinistra

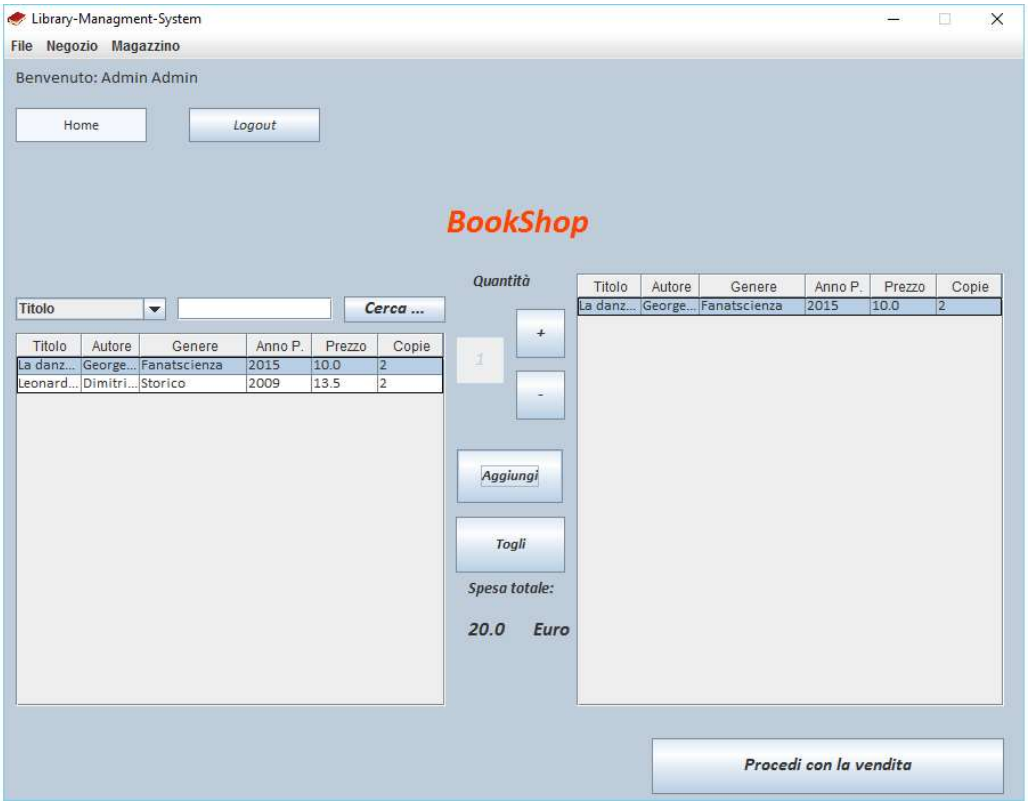


PS: Per effettuare l'accesso all'applicazione utilizzando la configurazione fornita, le credenziali sono: admin, admin.

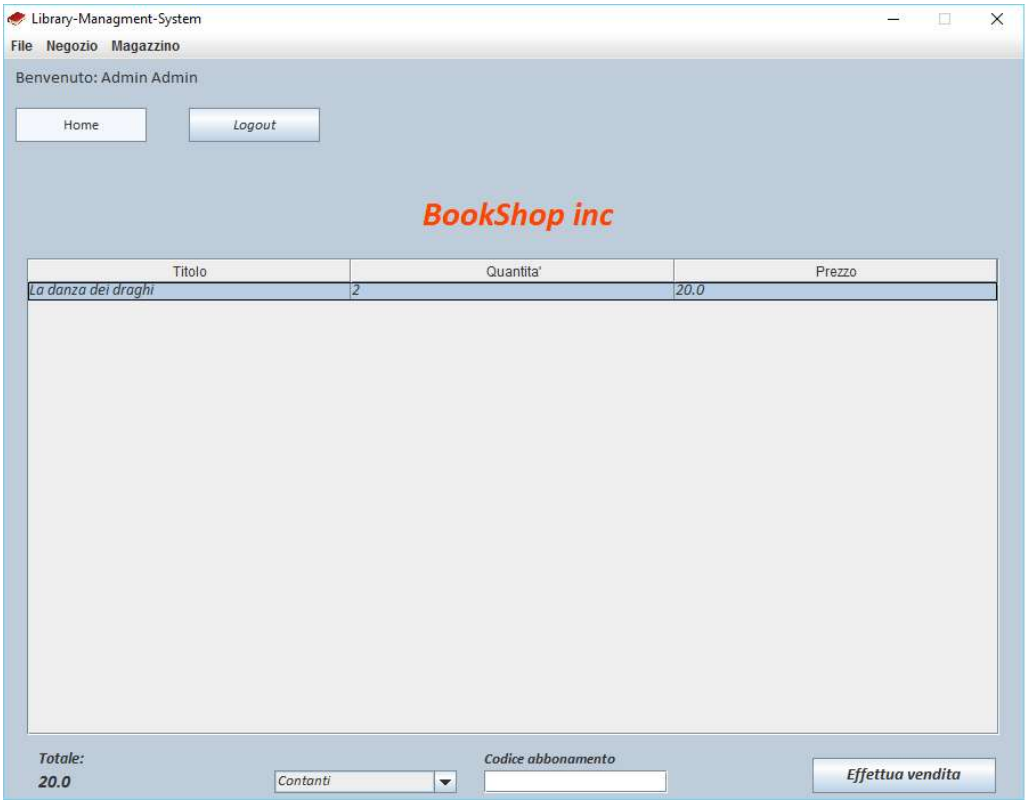
Successivamente si presenta la possibilità di effettuare la login riempiendo appositamente i campi Username e Password, e cliccando il bottone "Accedi" situato al centro della schermata, se non si è registrati è possibile effettuare una registrazione tramite il bottone "Registrati". Il bottone "Pulisci tutto" serve semplicemente a pulire i campi dello Username e della Password



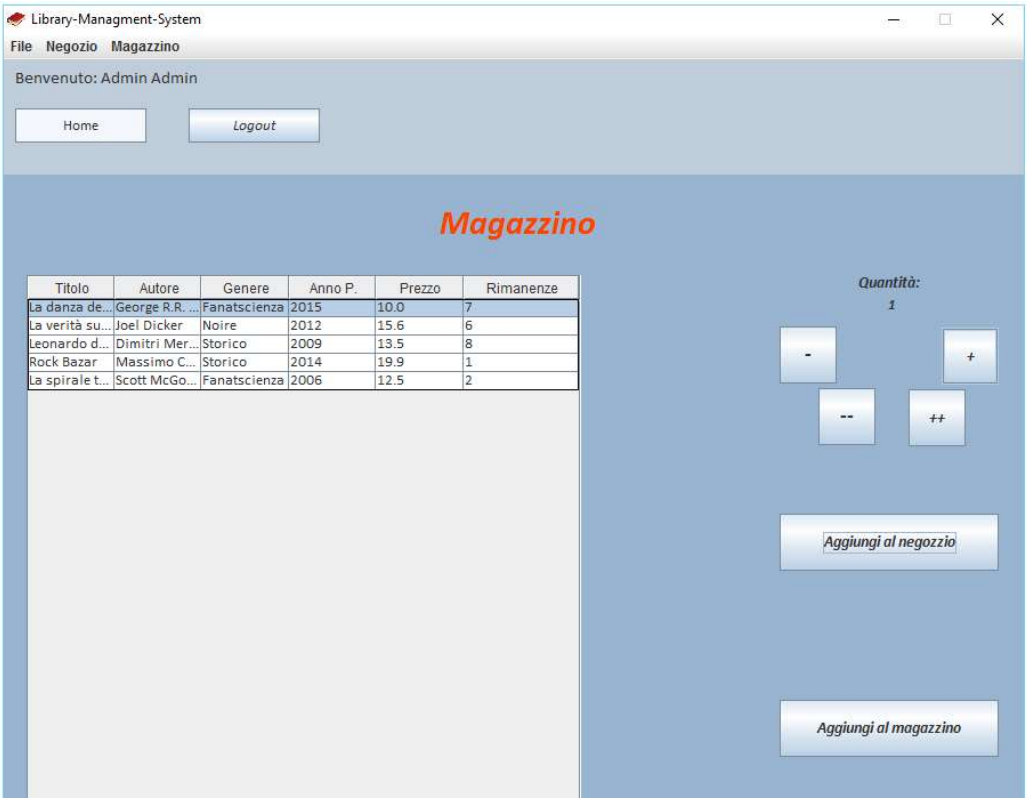
In questa schermata è possibile registrare un nuovo dipendete inserendo adeguatamente tutti i campi richiesti dopo di che bisogna cliccare su “Aggiungi dipendete”, il bottone “Pulisci” servirà a resettare tutti i campi presenti



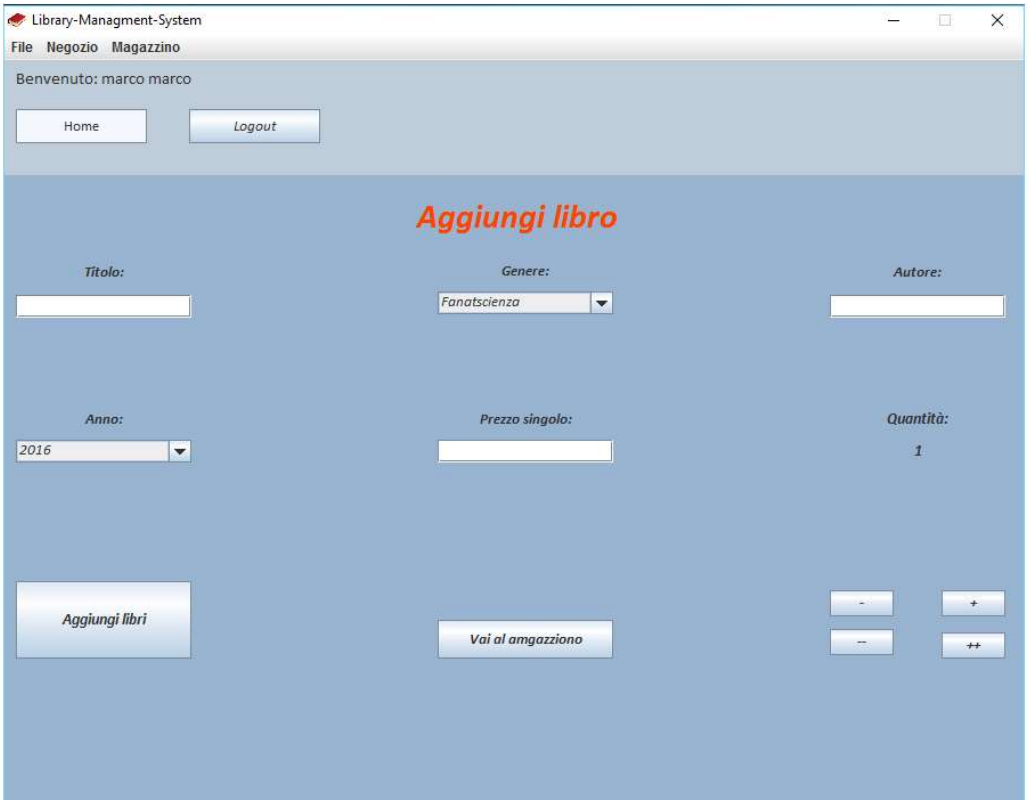
Una volta effettuata la login si può notare che nella barra in alto si sono abilitati dei bottoni ed il menù a tendina e si è effettuato l'accesso a quella che è la home del programma cioè il negozio in cui sulla sinistra c'è l'elenco di tutti i libri pronti per la vendita, sulla destra tutti quelli selezionati. I bottoni “+” e “-” servono ad aumentare la quantità di libri selezionati, senza sforare la quantità residua, il bottone “Aggiungi” permette di spostare i libri con la quantità selezionata dalla tabella di sinistra a quella di destra, il bottone “Togli” permette di togliere dalla tabella dei libri selezionati l'elemento selezionato dall'utente con tutte le copie. Quando la lista di destra avrà almeno un campo al suo interno risulterà possibile accedere alla schermata dello scontrino



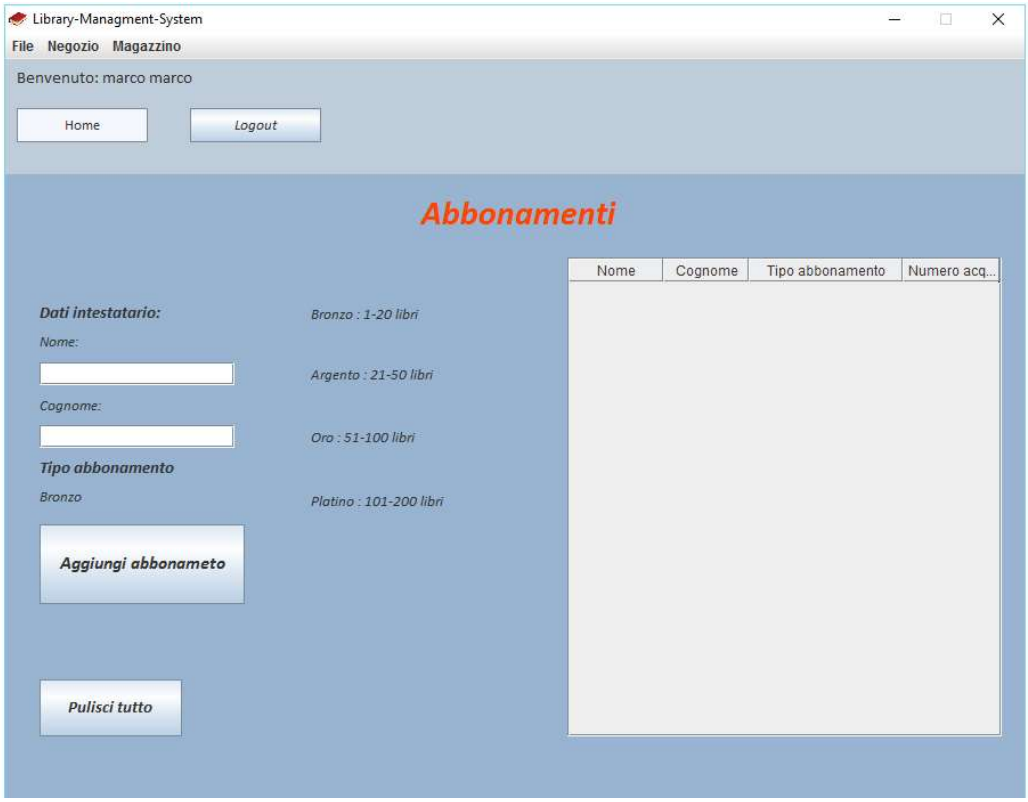
In questa schermata si effettuerà l'acquisto vero e proprio della merce richiesta, nella tabella centrale apparirà l'elenco di tutti i libri precedentemente selezionati, in basso a sinistra sarà riportato il totale da spendere, in basso al centro si dovrà scegliere il tipo di pagamento con accanto lo spazio per inserire il codice di un possibile abbonamento del cliente. Il bottone “Effettua vendita” concluderà la transazione.



Una volta effettuato l'accesso al magazzino tramite la barra in alto si potranno trasferire i libri da dove si trovano a negozio selezionando le quantità con i bottoni "+" "++" "-" "--" che rispettivamente aggiungeranno uno o dieci toglieranno uno o dieci alla quantità, per trasferire i libri al negozio una volta selezionata la quantità desiderata si premerà il bottone "Aggiungi al negozio", mentre per aumentare le scorte del magazzino basterà cliccare "Aggiungi al magazzino" che incrementerà le scorte con le quantità selezionate



Per aggiungere al magazzino un nuovo libro non presente bisognerà accedere alla schermata qui sopra riportata tramite il menu in alto sotto la voce “Magazzino”, qui una volta riempiti tutti i campi con i dati di un certo libro che si vuole aggiungere bisognerà premere il bottone “Aggiungi Libri” per aggiungere il libro, invece se si vuole accedere al magazzino basterà cliccare il bottone “Vai al magazzino” per accedervi, il bottoni “+” ”++” ”-” ”--” servono per settare le quantità.



In questa schermata è possibile aggiungere un nuovo abbonamento, basta inserire nelle apposite caselle le informazioni richieste e premere “Aggiungi abbonamento” dopo di che nella tabella di destra apparirà l'intestatario di un abbonamento con il relativo codice che sarà utilizzato durante l'acquisto di dei libri. Il bottone “Pulisci tutto” serve a resettare i campi in cui si inseriranno i dati.