

UNIVERSITÀ DEGLI STUDI DI VERONA
DIPARTIMENTO DI INFORMATICA

**Documentazione progetto
Ingegneria del Software**

A.A 2018/2019

Autori:

Luca Marzari VR421483
Deborah Pintani VR422805

Professore:

CARLO COMBI

Indice

1 Requisiti di sistema e casi d'uso	3
1.1 Specifiche progetto	3
1.2 Casi d'uso	4
1.2.1 Casi d'uso utente registrato	6
1.2.2 Casi d'uso utente non registrato	8
1.2.3 Casi d'uso utente responsabile	10
2 Schede di specifica dei casi d'uso	11
3 Diagramma delle sequenze	17
4 Diagrammi delle attività	35
4.1 Diagramma attività modifica dati anagrafici utente registrato	35
4.2 Diagramma attività registrazione utente	36
4.3 Diagramma attività aggiunta di un libro al carrello	36
4.4 Diagramma attività creazione di un ordine	37
4.5 Diagramma attività modifica libro catalogo	38
4.6 Diagramma attività aggiunta libro al catalogo	38
4.7 Diagramma attività aggiornamento classifiche	39
5 Diagrammi delle classi	40
5.1 Diagramma delle classi del <i>Main</i>	40
5.2 Diagramma delle classi del <i>package Controller</i>	41
5.3 Diagramma delle classi del <i>package Model</i>	42
5.4 Diagramma delle classi del <i>package View</i>	43
5.5 Diagramma delle classi del <i>package Data</i>	44
5.6 Diagramma delle classi del <i>package Utils</i>	45
6 Struttura del database	46
6.1 Scelte progettuali	46
6.2 Schema Logico	48

6.3	Schema Relazionale	48
7	Scelte progettuali	51
7.1	Sviluppo	51
7.2	Metodologia di sviluppo	52
7.3	Design Pattern utilizzati	54
7.3.1	MVC pattern	54
7.3.2	Facade pattern	55
7.3.3	DAO pattern	57
7.3.4	Singleton pattern	59
7.3.5	Observer pattern	59
7.3.6	Iterator pattern	59
8	Validazione e Test	60
8.1	Esempio caso d'uso - Ordine utente registrato	60

Capitolo 1

Requisiti di sistema e casi d'uso

1.1 Specifiche progetto

Il progetto presentato consiste nella realizzazione di un sistema informatico per gestire gli acquisti on-line di una libreria.

Il sistema contiene un catalogo di libri disponibili, ognuno dei quali è identificato da un codice ISBN e possiede un titolo, un autore o degli autori, una casa editrice, l'anno di pubblicazione, un genere, un prezzo ed una breve descrizione.

Gli utenti possono visualizzare le classifiche di vendita che sono organizzate per genere (novità, narrativa, ragazzi, ...) e vengono aggiornate ogni settimana. Per ogni posizione della classifica si indica da quante settimana il libro è in quella posizione.

Il sistema memorizza gli ordini degli utenti. Gli utenti possono essere registrati o meno. Per gli utenti si memorizzano nome, cognome, indirizzo, CAP, città, numero di telefono ed email.

Ogni utente registrato accede con email e password ed ha associata una LibroCard per la raccolta punti. Ogni LibroCard ha un numero identificativo, una data di emissione e il totale dei punti raccolti. Gli utenti registrati possono specificare uno o più indirizzi di spedizione diversi da quello di residenza. Ogni libro ha associato il numero di punti che vengono caricati sulle LibroCard in caso di acquisto da parte di utenti registrati.

Per ogni ordine si memorizzano il codice (univoco), la data, i libri che lo compongono, l’utente che lo ha effettuato, il costo totale, il tipo di pagamento (carta di credito, paypal o contrassegno) e il saldo punti se l’utente è registrato. Il sistema deve permettere agli utenti registrati di accedere al loro profilo, modificare i dati anagrafici, verificare il saldo punti e lo stato dei loro ordini. Ogni utente registrato può vedere tutti gli ordini che ha effettuato nel tempo con il totale dei punti accumulati per ogni ordine. Gli utenti non registrati possono accedere agli ordini che hanno effettuato tramite il codice dell’ordine.

I responsabili della libreria devono poter verificare lo stato degli ordini, e il saldo punti delle LibroCard degli utenti registrati. Inoltre, i responsabili della libreria sono responsabili dell’inserimento dei dati relativi ai libri che si possono ordinare e dell’aggiornamento delle classifiche. Tutti gli utenti sono opportunamente autenticati dal sistema per poter accedere alle funzionalità di loro competenza.

1.2 Casi d’uso

Il sistema informatico proposto permette l’utilizzo da parte di qualsiasi utente.

Un singolo utente appartiene ad una di queste tre categorie:

- Utente non registrato.
- Utente registrato.
- Utente responsabile.

Qualsiasi utente ha la possibilità di consultare sia il catalogo dei libri disponibili sul sistema, sia le classifiche che sono divise per genere e categorie.

Per quanto riguarda le ultime due categorie di utenti, entrambi hanno a disposizione delle credenziali con cui hanno la possibilità di effettuare l’autenticazione e accedere alla propria area personale.

L’utente non registrato ha la possibilità di inserire i libri all’interno del carrello, ma affinché l’ordine sia completato, vi è la necessità di specificare se si ha l’intenzione di registrarsi oppure procedere come utente non registrato.

Inoltre, l’utente non registrato ha la possibilità di verificare lo stato di un suo ordine previa autenticazione tramite mail e tracking code.

Il sistema, come da specifiche, prevede che l'utente registrato una volta effettuato l'accesso, abbia la possibilità di visionare i propri dati anagrafici, effettuare eventualmente delle modifiche, controllare lo stato degli ordini e infine il saldo dei punti della relativa LibroCard.

L'utente registrato ha inoltre la possibilità di effettuare degli ordini dopo essersi opportunamente autenticato e aver scelto indirizzo di spedizione e modalità di pagamento.

Per quanto riguarda invece l'utente responsabile, quest'ultimo ha la possibilità, previa autenticazione, di aggiornare i dati relativi ai libri del catalogo, aggiornare classifiche, visionare e/o aggiornare lo stato degli ordini degli utenti e infine visionare il saldo dei punti delle LibroCard relative agli utenti registrati.

Viene presentato di seguito lo schema dei casi d'uso generale per tutti gli utenti (prossima pagina):

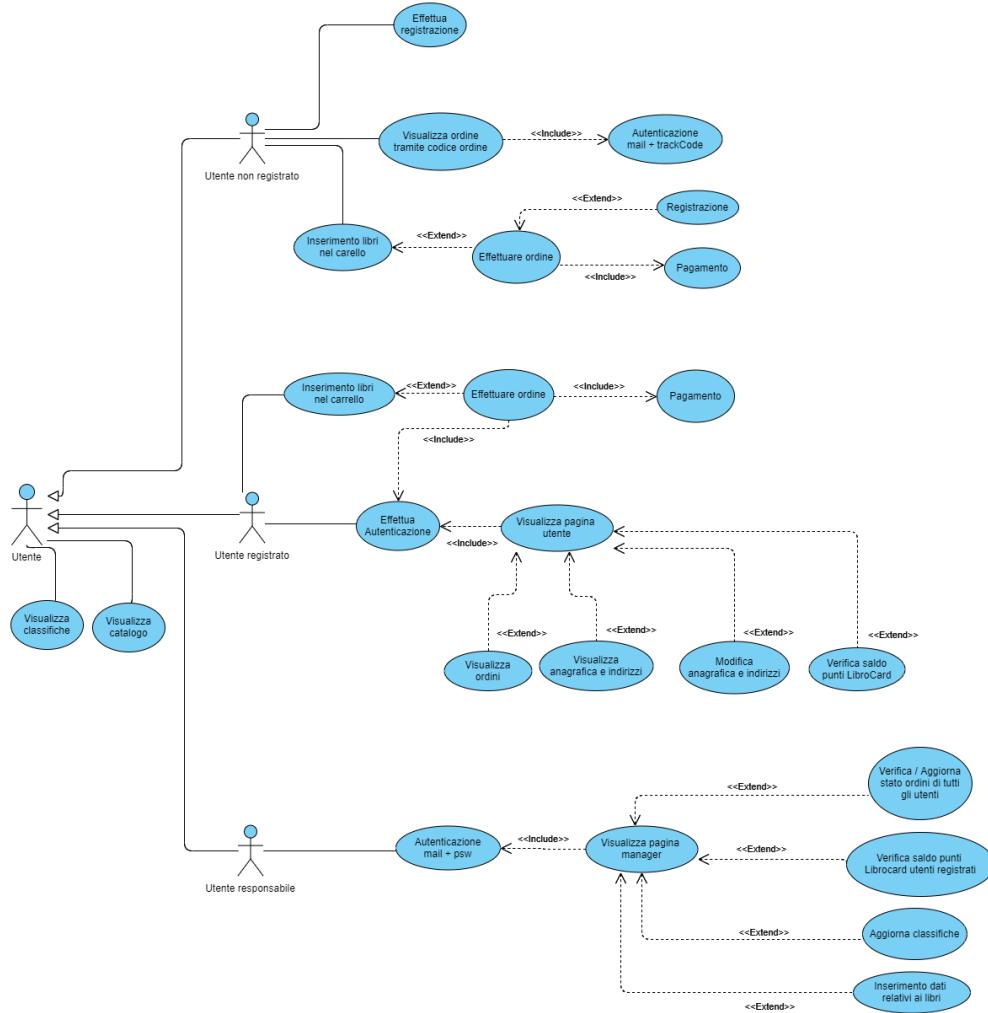


Figura 1.1: Casi d'uso generale

1.2.1 Casi d'uso utente registrato

Al fine di comprendere al meglio i casi d'uso di ogni singolo utente, viene di seguito riportata una descrizione maggiormente dettagliata circa le possibili interazioni con il sistema da parte di un utente registrato:

Partendo dalle azioni che possono svolgere tutti gli utenti notiamo che anche l'utente registrato ha la possibilità di visionare le classifiche filtrandole, qualora lo volesse, per genere e/o categoria.

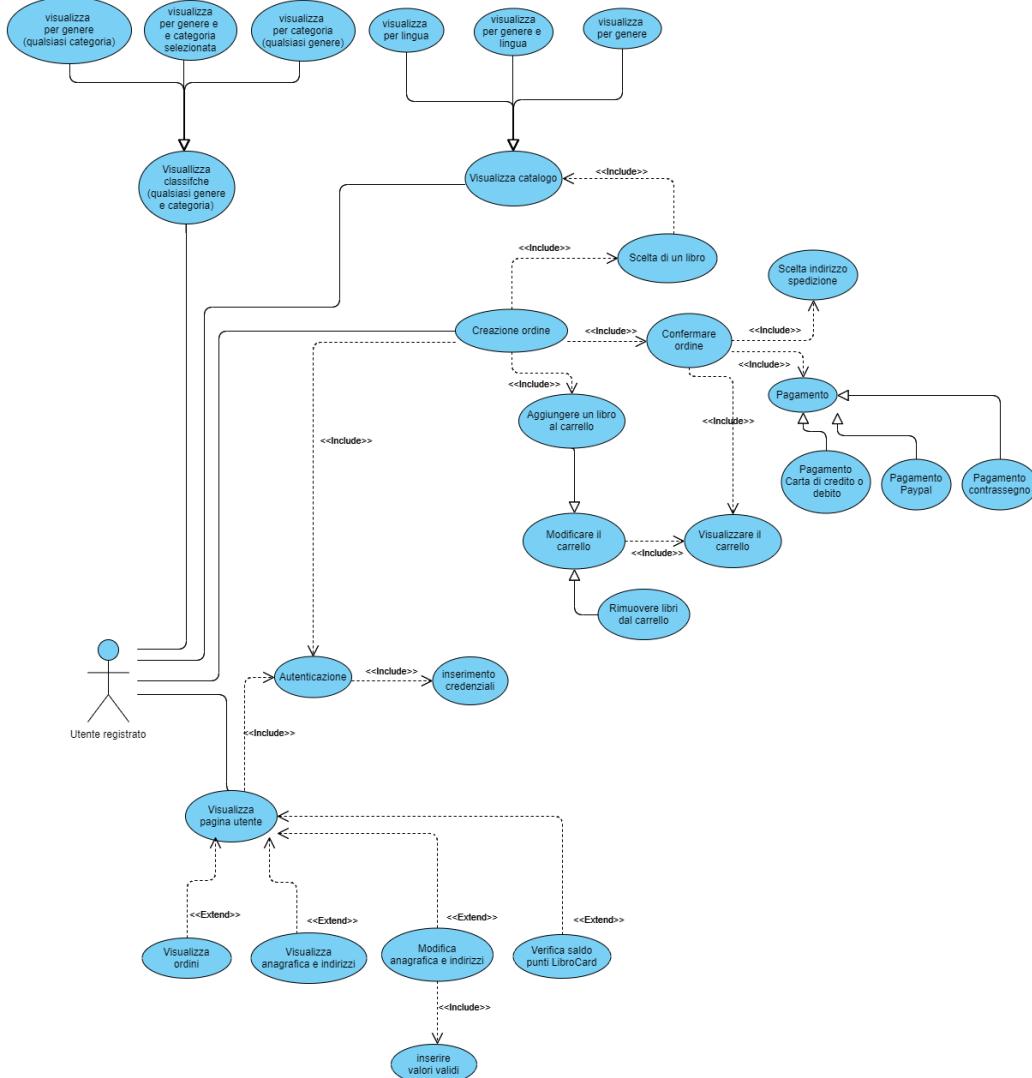


Figura 1.2: Casi d'uso utente registrato

L'utente registrato ha la possibilità di visionare il catalogo, anch'esso potenzialmente filtrato per genere e/o lingua.

Come presentato in precedenza notiamo inoltre che, dopo aver aggiunto libri al carrello e affinché l'utente registrato concluda un ordine, vi è la necessità che l'utente sia opportunamente autenticato dal sistema e successivamente che specifichi un indirizzo di spedizione e una modalità di pagamento valida. L'utente registrato può inoltre accedere alla propria area personale previa autenticazione mediante mail e password.

Una volta effettuato l'accesso può visualizzare le proprie informazioni personali, modificare i propri dati anagrafici, visualizzare lo stato degli ordini e/o il saldo totale dei punti della LibroCard.

1.2.2 Casi d'uso utente non registrato

L'utente non registrato ha la possibilità di visionare il catalogo e le classifiche, potenzialmente filtrate qualora lo desiderasse.

Può inoltre creare un ordine e decidere se registrarsi o meno al sistema; in caso si registri deve necessariamente inserire tutte le informazioni necessarie al registrazione quali dati anagrafici, mail univoca e password.

In caso invece voglia continuare come utente non registrato, viene chiesto comunque all'utente di inserire le proprio informazioni anagrafiche, una mail univoca e un indirizzo di spedizione valido.

L'utente non registrato può controllare lo stato degli ordini inserendo mail e tracking code dell'ordine.

Si rimanda alla prossima pagina per lo *use case* completo:

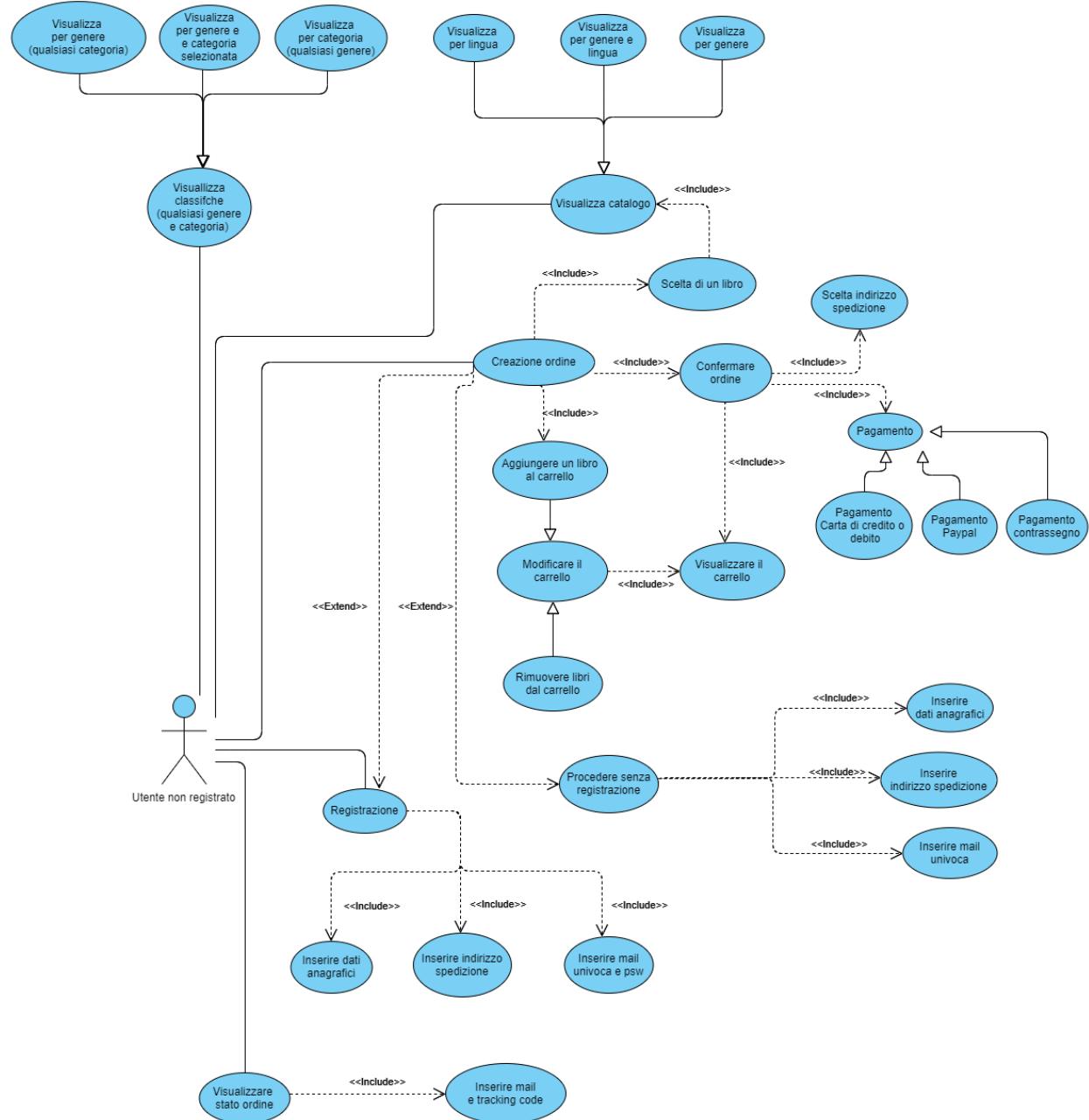


Figura 1.3: Casi d'uso utente non registrato

1.2.3 Casi d'uso utente responsabile

L'utente responsabile può visualizzare classifiche e catalogo opportunamente filtrato qualora lo desiderasse; non può però effettuare acquisti.

Può accedere alla sua area personale previa autenticazione tramite mail e password e una volta effettuato l'accesso ha la possibilità di visualizzare lo stato degli ordini degli utenti registrati e non registrati, può aggiungere nuovi libri al catalogo o modificare libri già esistenti.

Infine l'utente responsabile può visualizzare il saldo relativo alle LibroCard degli utenti registrati.

L'utente responsabile inoltre può aggiornare le classifiche dei libri, scelte per categoria o genere o entrambi.

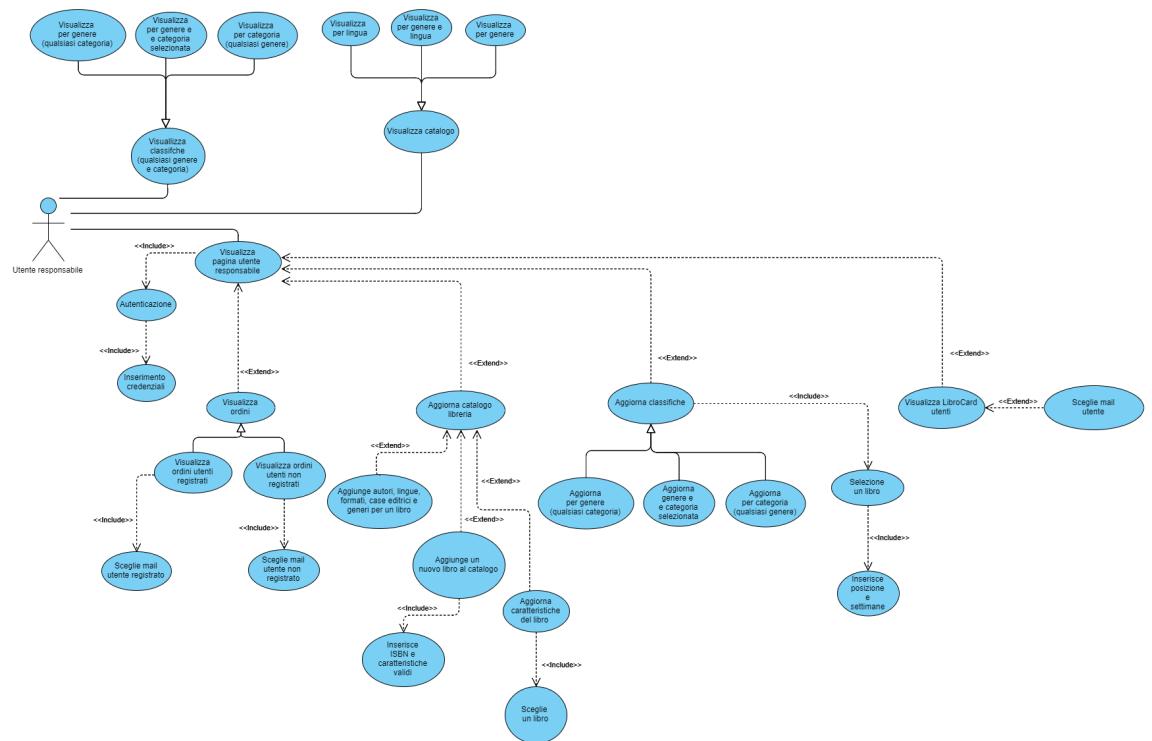


Figura 1.4: Casi d'uso utente responsabile

Capitolo 2

Schede di specifica dei casi d'uso

Seguono ora una serie di schede di specifiche dei casi d'uso più dettagliati rispetto agli schemi sopra riportati:

ID: Caso d'uso utente generico: Visualizzazione classifiche
ATTORI: Utente generico
PRECONDIZIONI:
SEQUENZA: <ol style="list-style-type: none">1. L'utente cliccando sul bottone "Charts", ha la possibilità di vedere visionare le classifiche (generali di default).<ol style="list-style-type: none">1.1 L'utente può filtrare le classifiche scegliendo il genere dall'apposito filtro.1.2 L'utente ha la possibilità di filtrare le classifiche per categoria con l'apposito filtro.1.3 L'utente ha la possibilità di filtrare le classifiche per genere e categoria selezionando entrambi i filtri dagli appositi menù a tendina.
POSTCONDIZIONI: <ol style="list-style-type: none">1. La schermata delle classifiche viene aggiornata in base ai filtri cliccati, se non si filtra rimane quella di default.

<p>ID: Caso d'uso utente registrato: Pagina personale utente registrato e modifica indirizzi di spedizione</p>
<p>ATTORI: Utente registrato</p>
<p>PRECONDIZIONI:</p> <ol style="list-style-type: none">1. L'Utente deve essersi autenticato con le proprie credenziali.
<p>SEQUENZA:</p> <ol style="list-style-type: none">1. L'utente cliccando sul bottone "My Orders", ha la possibilità di vedere lo stato di tutti i suoi ordini.2. L'utente cliccando sul bottone "My LibroCard", ha la possibilità di verificare il saldo attuale dei punti della sua LibroCard
<p>POSTCONDIZIONI:</p> <ol style="list-style-type: none">1. In caso l'utente modifichi i suoi dati anagrafici, i valori vengono raccolti dal sistema e viene aggiornato il DB con le nuove informazioni.

<p>ID: Caso d'uso generale: Creazione Ordine</p>
<p>ATTORI: Utente registrato o Utente non registrato</p>
<p>PRECONDIZIONI:</p> <ol style="list-style-type: none">1. L'Utente ha consultato il catalogo, eventualmente filtrandolo per genere e/o lingua.2. L'utente ha inserito nel carrello almeno un libro.
<p>SEQUENZA:</p> <ol style="list-style-type: none">1. L'utente clicca sul bottone del carrello e viene indirizzato alla pagine del carrello contenente i libri scelti.2. Quando l'utente clicca sul bottone "Proceed to CheckOut":<ol style="list-style-type: none">a. Se l'utente è registrato e ha già effettuato l'autenticazione, il sistema lo porta alla schermata di pagamento, dove dopo avere selezionato un indirizzo di spedizione e un metodo di pagamento valido, l'ordine si conclude.b. Se l'utente è registrato ma non ha ancora effettuato l'accesso, il sistema lo porta alla schermata di login, dove dopo aver inserito le proprie credenziali, prosegue come al punto [a].c. Se l'utente non è registrato, allora il sistema lo porta alla schermata di Login-SignUp, dove l'utente può decidere o di registrarsi o di proseguire come utente non registrato:<ol style="list-style-type: none">c.1 se l'utente decide di registrarsi, viene portato alla schermata di registrazione e dopo avere inserito i dati per la registrazione si prosegue con [a].c.2 se l'utente decide di proseguire senza registrazione viene condotto ad una pagina dove inserire le sue informazioni personali, indirizzo di spedizione e una mail univoca per associare l'ordine, se i campi inseriti sono validi allora il sistema lo aggiunge al DB e poi si prosegue con [a].
<p>POSTCONDIZIONI:</p> <ol style="list-style-type: none">1. L'ordine viene creato e inserito nel DB associato a quel determinato utente.

<p>ID: Caso d'uso utente non registrato: Registrazione</p>
<p>ATTORI: Utente non registrato</p>
<p>PRECONDIZIONI:</p> <ol style="list-style-type: none">1. L'Utente non deve essere registrato e deve usare per la registrazione una mail diversa da quella usata per un eventuale precedente acquisto come non registrato.
<p>SEQUENZA:</p> <ol style="list-style-type: none">1. L'utente cliccando sul bottone "Login-SignUp", viene portato dal sistema alla pagina di accesso o registrazione.2. L'utente cliccando sul bottone "SignUp", viene condotto alla schermata di registrazione.3. Qui inserisce tutti i dati anagrafici, un indirizzo valido e un mail univoca e password.4. L'utente poi clicca sul bottone "SignUp":<ol style="list-style-type: none">3.1 Il sistema dopo che l'utente ha inserito i dati anagrafici e mail, verifica che i valori siano validi, ovvero che i campi rispettino le richieste e che la mail sia univoca.3.2 In caso non siano validi il sistema avvisa l'utente e si ritorna al punto [3.1].5. Se tutto è corretto l'utente cliccando sul bottone "SignUp" si registra al sistema.
<p>POSTCONDIZIONI:</p> <ol style="list-style-type: none">1. L'utente viene inserito nel DB.

ID: Caso d'uso utente responsabile: Visualizzazione ordine e inserimento o modifica di un libro
ATTORI: Utente responsabile
PRECONDIZIONI: <ol style="list-style-type: none"> L'Utente responsabile deve essersi autenticato con le proprie credenziali.
SEQUENZA: <ol style="list-style-type: none"> L'utente responsabile cliccando sul bottone "View Order Status" ha la possibilità di vedere lo stato degli ordini di tutti gli utenti (default): <ol style="list-style-type: none"> Ha la possibilità inoltre di filtrare gli ordini per utenti registrati e non registrati da un apposito filtro L'utente responsabile ha la possibilità di modificare lo stato degli ordini tramite un apposito menù a tendina. Qualora il responsabile modificasse lo stato di un ordine compare un messaggio che lo informa che stato dell'ordine è stato cambiato. L'utente responsabile cliccando sul bottone "Add/Edit book", ha la possibilità di aggiungere autori, generi, formati, lingue e case editrici ognuna di queste in un apposita schermata, oppure aggiungere direttamente un nuovo libro cliccando su "Add new book" che porta in una schermata apposita. L'utente può anche modificare un libro già esistente, andando a modificare i singoli campi che compongono il libro, cliccando sul bottone "Edit Book". <ol style="list-style-type: none"> Dopo che l'utente ha inserito i dati relativi ai libri, sia che sia un nuovo libro o una modifica e dopo aver cliccato rispettivamente sul bottone "Add new book" o "Edit book", il sistema verifica che i valori siano validi, ovvero che i campi rispettino le richieste. In caso non siano validi il sistema avvisa l'utente e si ritorna al punto [3.1].
POSTCONDIZIONI: <ol style="list-style-type: none"> In caso di modifica dello stato dell'ordine viene aggiornata l'informazione nel DB. Il libro viene aggiornato nel DB (in caso di Edit). Il libro viene inserito nel DB (in caso di Add).

<p>ID: Caso d'uso utente responsabile: Aggiornamento delle classifiche</p>
<p>ATTORI: Utente responsabile</p>
<p>PRECONDIZIONI:</p> <ol style="list-style-type: none">1. L'Utente responsabile deve essersi autenticato con le proprie credenziali.
<p>SEQUENZA:</p> <ol style="list-style-type: none">1. L'utente responsabile cliccando sul bottone "Update Charts" ha la possibilità di aggiornare le classifiche:<ol style="list-style-type: none">1.1 Ha la possibilità di filtrare le classifiche da aggiornare per genere dei libri e/o categoria.1.2 Oppure può aggiornare la classifica inerente a qualsiasi genere e categoria, una sorta di classifica assoluta.<ul style="list-style-type: none">• Per inserire un libro in classifica è necessario indicare la posizione dove si vuole inserire e da quante settimane è in classifica.1.3 Cliccando sul bottone "Insert/Edit book in Charts" l'utente può inserire o aggiornare un libro nella classifica selezionata.<ul style="list-style-type: none">• Il sistema dopo che l'utente ha cliccato sul bottone verifica che i campi (posizione e settimane) siano riempiti correttamente e in caso contrario avvisa l'utente.1.4 Cliccando sul bottone "Delete book from Charts" l'utente può rimuovere un libro dalla classifica selezionata.<ul style="list-style-type: none">• Il sistema dopo che l'utente ha cliccato sul bottone verifica che i campi (posizione e settimane) siano riempiti correttamente e in caso contrario avvisa l'utente.
<p>POSTCONDIZIONI:</p> <ol style="list-style-type: none">1. La classifica viene aggiornata anche sul DB.

Capitolo 3

Diagramma delle sequenze

Vengono presentati di seguito i principali diagrammi delle sequenze dei casi d'uso presentati al paragrafo 2.

Si precisa che alcuni di questi diagrammi di sequenza qui presentati sono stati semplificati per un discorso di chiarezza e leggerezza; si considerano quindi le singole sequenze contenti le chiamate ai metodi principali.

Partiamo con il diagramma di sequenza inherente alla visualizzazione del catalogo da parte di un utente generico:

LD Books

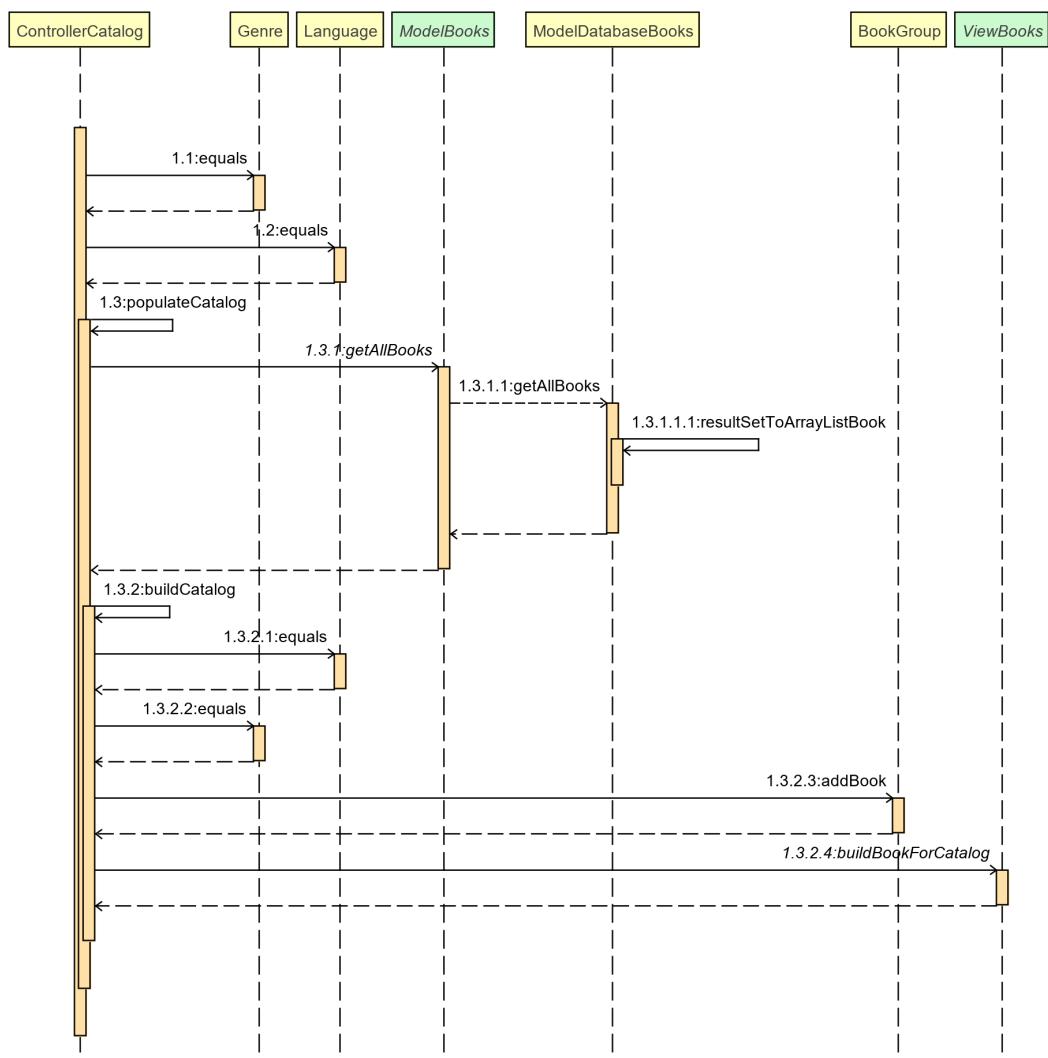


Figura 3.1: Diagramma sequenza visualizza catalogo utente generico

Viene di seguito riportato il diagramma di sequenza per la visione delle classifiche per l'utente generico:

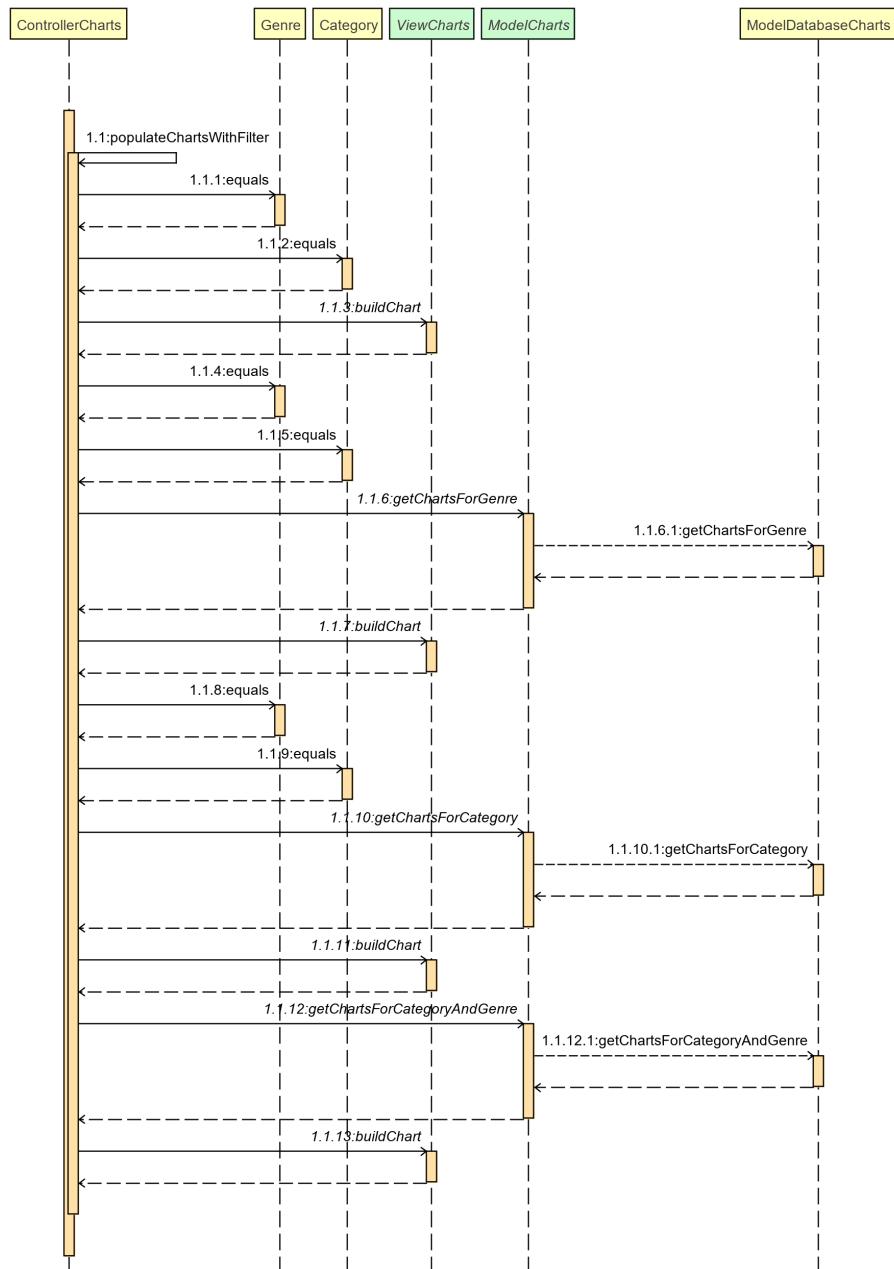


Figura 3.2: Diagramma sequenza visualizza classifiche utente generico

Iniziamo a vedere il diagramma delle sequenze inerente all'utente registrato e in particolare la **modifica dei dati anagrafici**:

Per quanto riguarda la spiegazione di classi, interfacce e pattern utilizzati si rimanda al capitolo inerente alle scelte progettuali e allo sviluppo.

Definiamo qui di seguito però alcune caratteristiche importanti implementante al fine di rispettare i requisiti. L'utente dopo essersi opportunamente autentico può accedere tramite il bottone "Edit Profile" alla pagine per modificare i propri dati anagrafici.

Qui l'utente ha la possibilità di modificare i propri dati anagrafici, aggiungere e/o rimuovere indirizzi di spedizione.

Nel diagramma di sequenza, presentato nella prossima pagina, siamo voluti andare ad un livello di dettaglio maggiore al fine di mostrare la comunicazione tra sistema e DB nel caso di aggiornamento di indirizzi da parte dell'utente registrato.

Lo schema è stato inserito nella prossima pagina per questioni di spazio.

LD Books

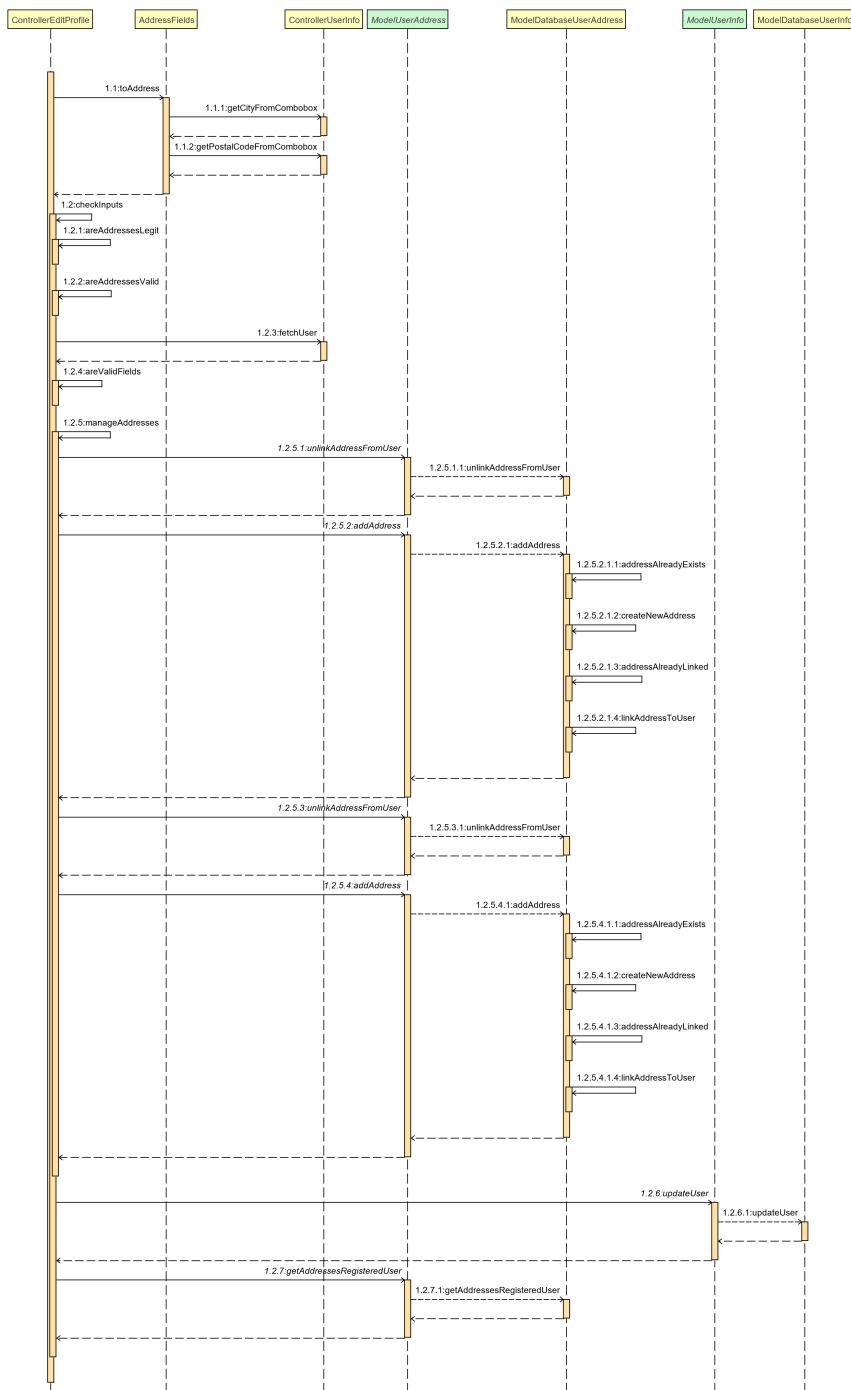


Figura 3.3: Diagramma sequenza modifica dati anagrafici utente registrato

I prossimi diagrammi delle sequenze che presentiamo sono sempre inerenti all'utente registrato e in ordine riguardano:

- Visione del profilo personale
- Visione di tutti gli ordini effettuati
- Visione del saldo punti inerenti alla LibroCard

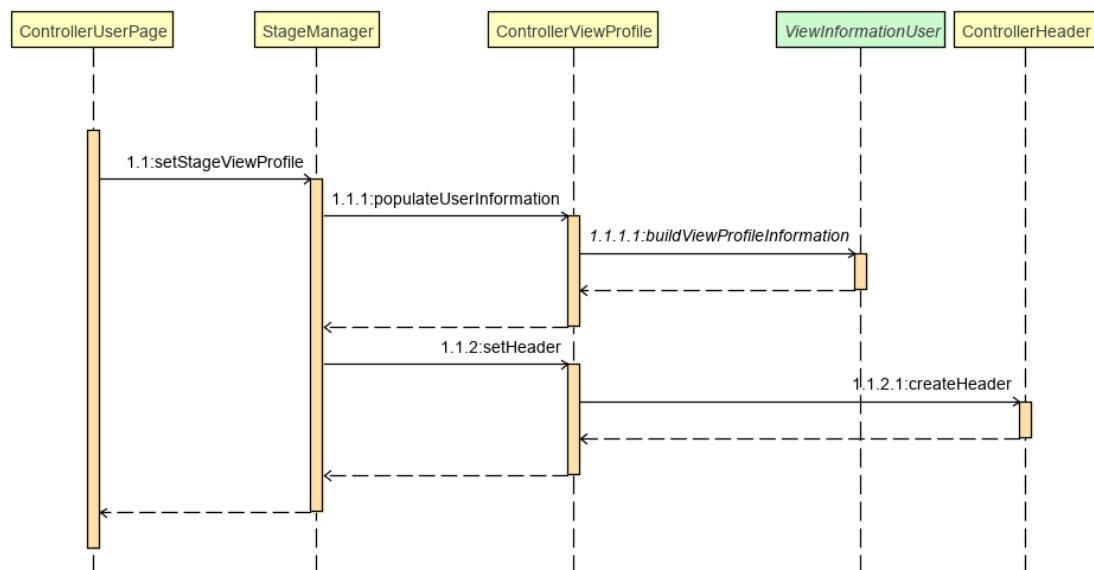


Figura 3.4: Diagramma sequenza visualizza profilo utente registrato

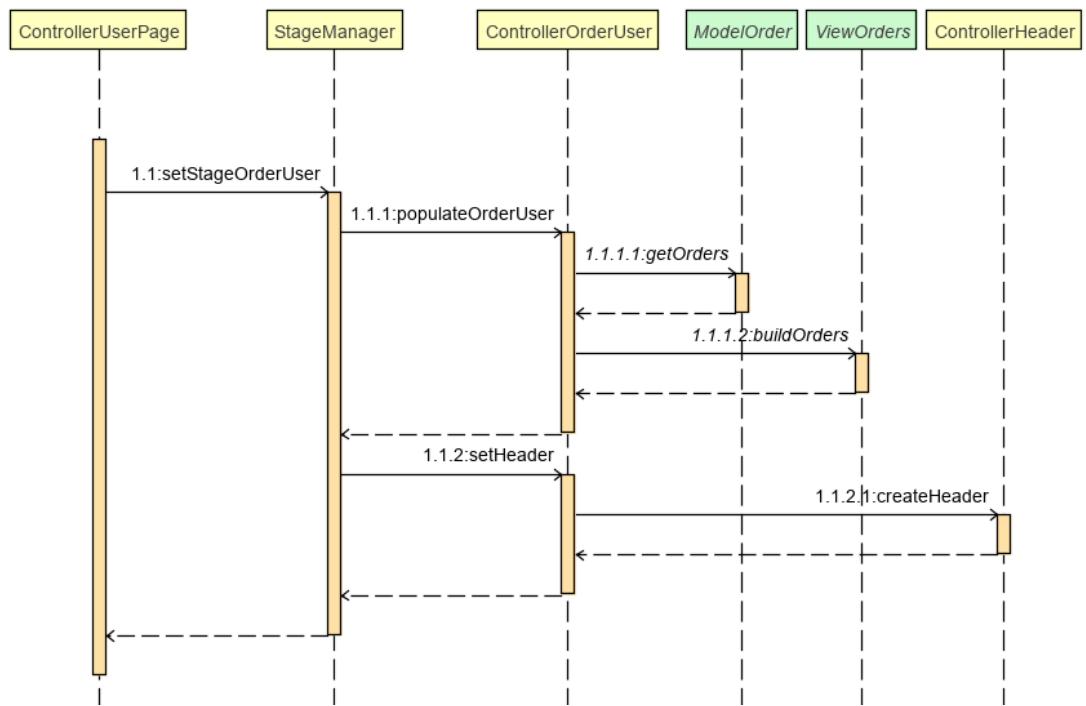


Figura 3.5: Diagramma sequenza visualizza ordini effettuati utente registrato

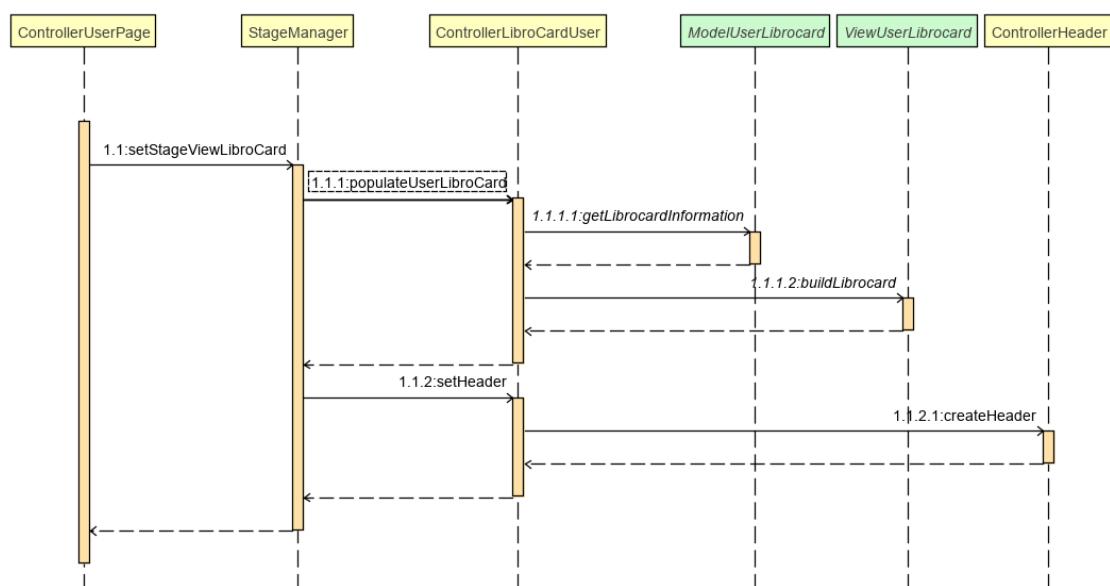


Figura 3.6: Diagramma sequenza visualizza LibroCard utente registrato

I prossimi diagrammi delle sequenze riguardano invece la **creazione dell'ordine** da parte di utenti registrati o utenti non registrati.

Abbiamo detto in precedenza (nello use case per la creazione dell'ordine) che è necessario prima cliccare sul carrello e visualizzare il riepilogo dei libri scelti. Presentiamo quindi di seguito cosa succede quando l'utente clicca sull'immagine del carrello:

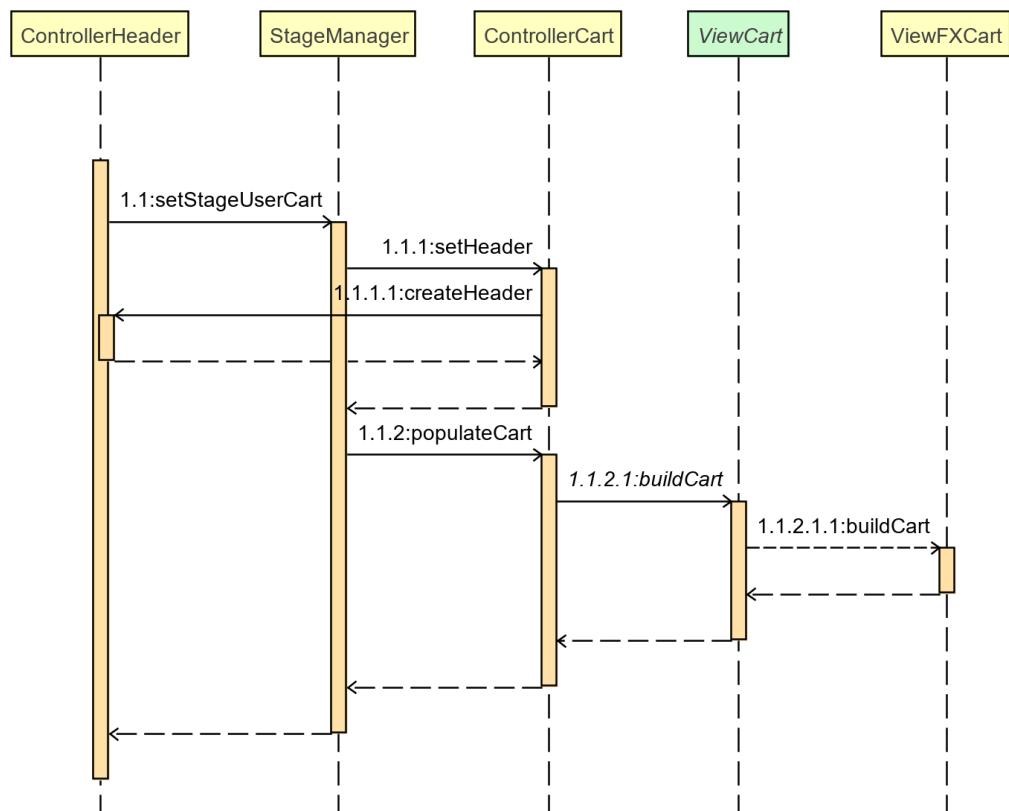


Figura 3.7: Diagramma sequenza visualizza carrello utente registrato - rato

Dopo aver cliccato sul carrello l'utente ha la possibilità di modificare il carrello rimuovendo alcuni libri inseriti oppure può procedere al pagamento. Se l'utente clicca su "Remove book from cart", la struttura dati utilizzata per il carrello, in questo caso una mappa, viene aggiornata e si ricostruisce la schermata, dunque il diagramma di sequenza è analogo a quello presentato sopra 3.8.

Presentiamo ora invece il diagramma delle sequenze di cosa succede se l'utente dalla schermata del carrello preme sul pulsante "Proceed to CheckOut":

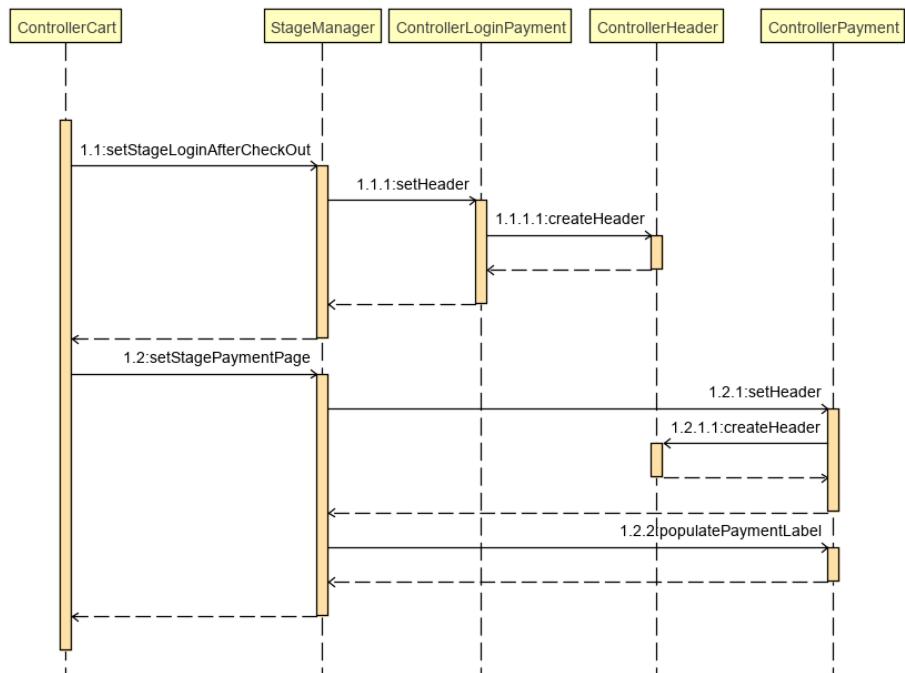


Figura 3.8: Diagramma sequenza pagamento utente registrato - utente non registrato

Notiamo che la prima parte del diagramma di sequenza ovvero la chiamata al metodo "setStageLoginAfterCheckOut" è inherente alla possibilità che l'utente registrato non si sia ancora autenticato oppure alla possibilità che l'utente sia non registrato.

Presentiamo quindi i diagrammi di sequenza per:

- Autenticazione
- Registrazione
- Procedere senza registrazione

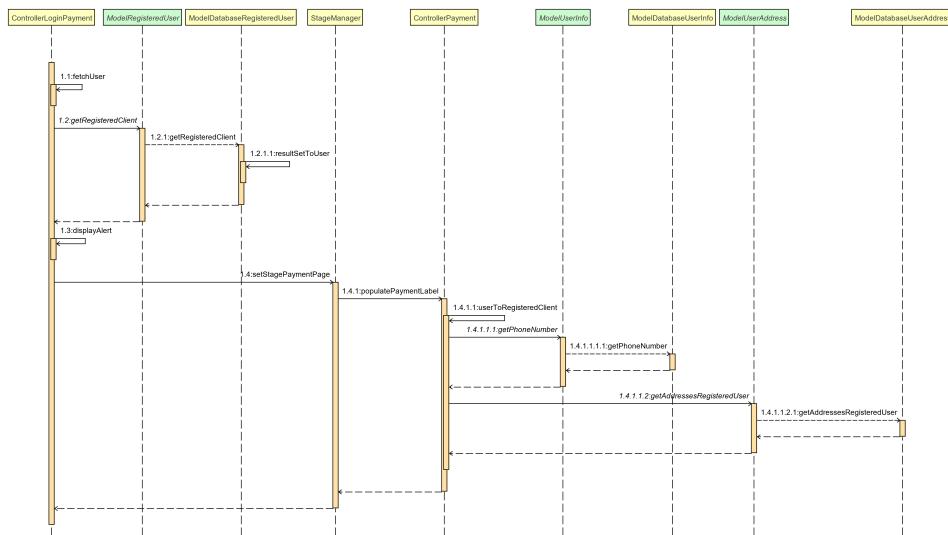


Figura 3.9: Diagramma sequenza autenticazione utente registrato

In questo diagramma sequenza e nei successivi abbiamo inserito qualche dettaglio in più rispetto ai precedenti al fine di mostrare l'utilizzo del Database per la gestione delle varie tipologie di utenti.

La struttura e l'utilizzo del database nel dettaglio vengono presentati, come detto in precedenza, al capitolo inerente alle scelte progettuali e allo sviluppo del sistema.

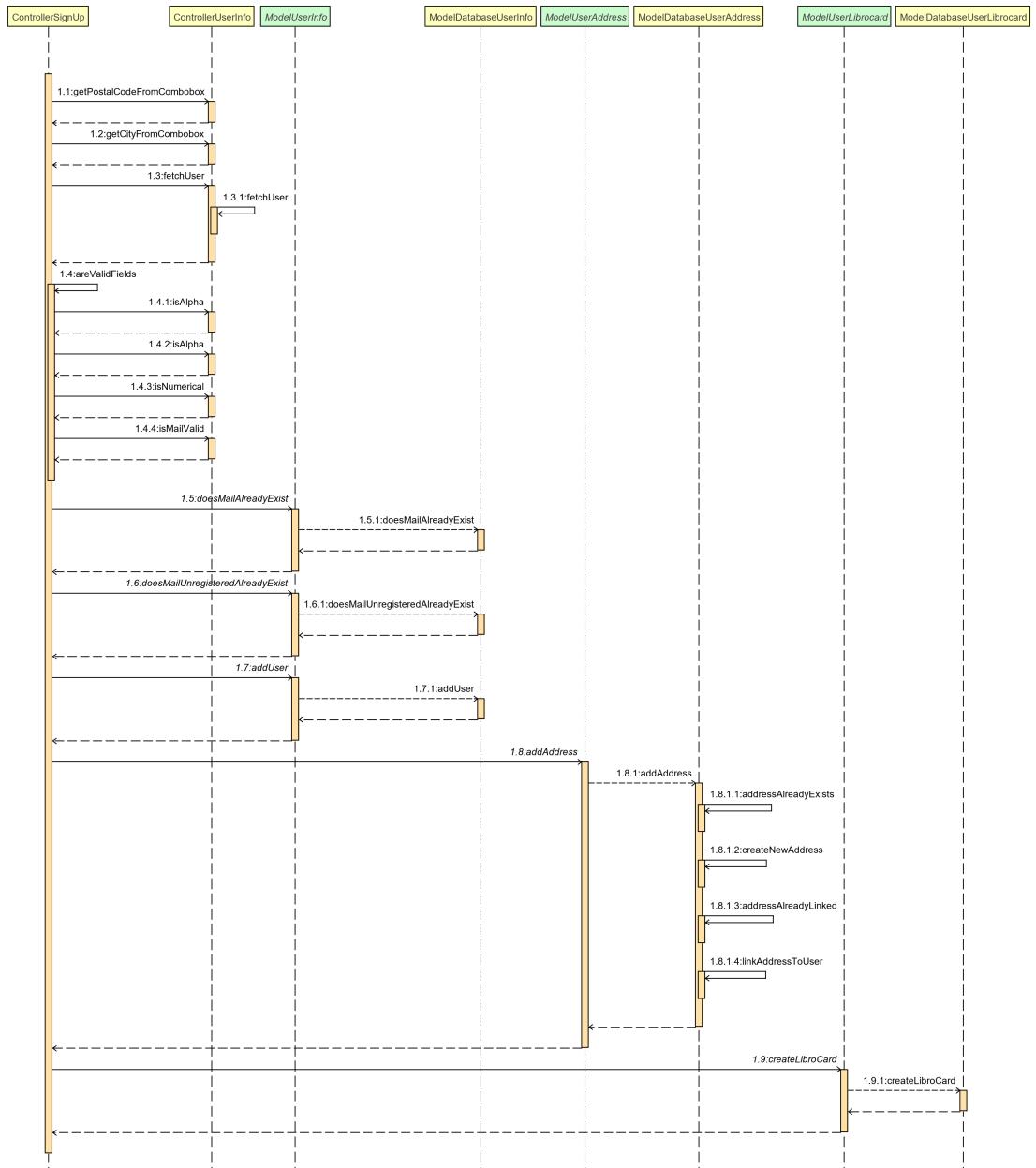


Figura 3.10: Diagramma sequenza registrazione utente non registrato

Per la registrazione di un utente si recuperano tutte le informazioni, successivamente si verifica nel DB che la mail sia univoca. A questo punto, se i dati inseriti sono corretti, si aggiunge l'utente e si collega all'indirizzo di spedizione scelto.

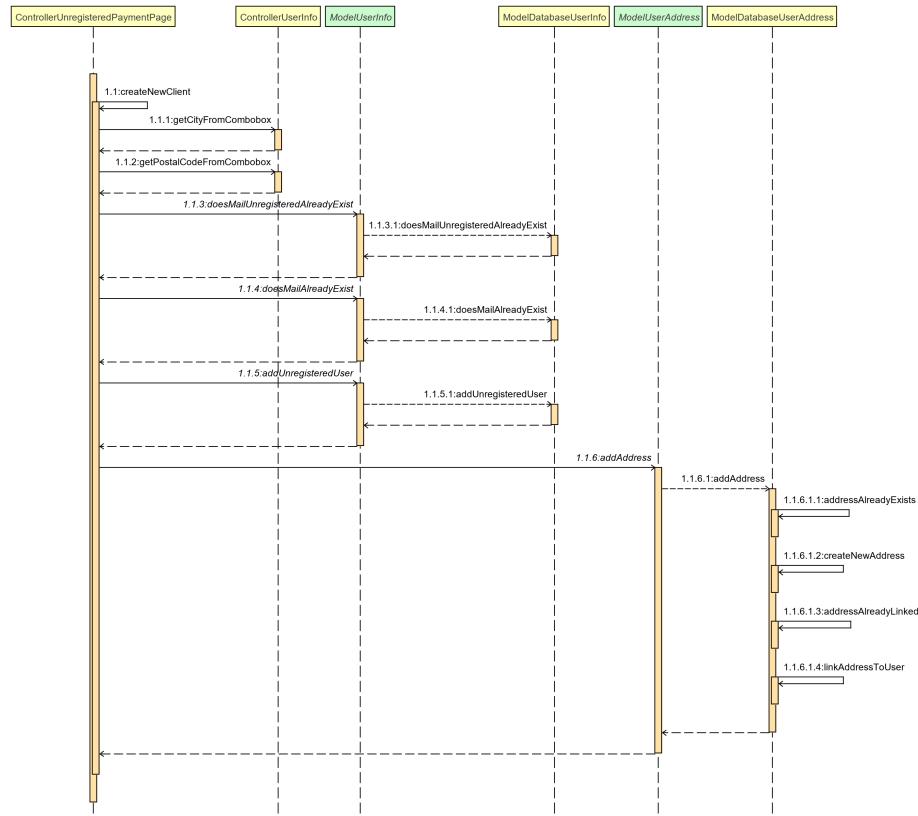


Figura 3.11: Diagramma sequenza utente continua come non registrato

Se l'utente decide di proseguire alla creazione dell'ordine senza registrarsi, il sistema comunque gli richiede tutte le informazioni al fine di poterlo identificare come utente non registrato.

Quindi vengono richiesti i suoi dati anagrafici, una mail che deve essere univoca e un indirizzo di spedizione: se i dati forniti sono validi l'utente viene aggiunto al DB (come non registrato) e condotto alla pagina di pagamento.

Viene qui di seguito riportato il diagramma delle sequenze dell'utente responsabile per **visualizzare ordini e aggiornare lo stato dell'ordine**:

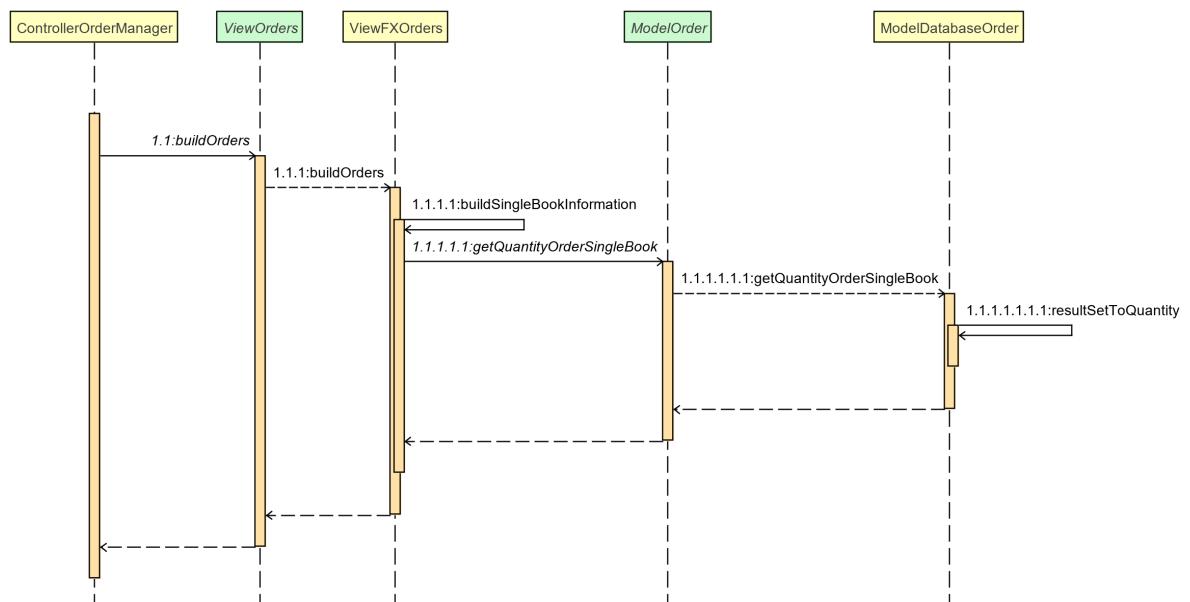


Figura 3.12: Diagramma sequenza utente responsabile visualizza ordine utenti

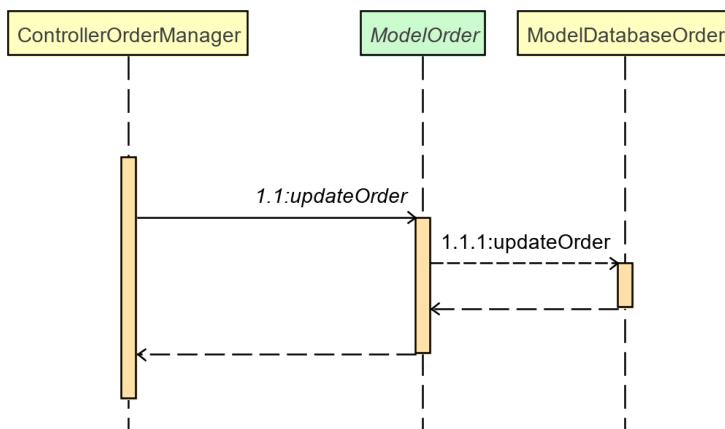


Figura 3.13: Diagramma sequenza utente responsabile aggiorna stato ordine utenti

Viene di seguito riportato il diagramma delle sequenze dell'utente responsabile per aggiunta di un nuovo libro al catalogo e aggiornamento libro esistente:

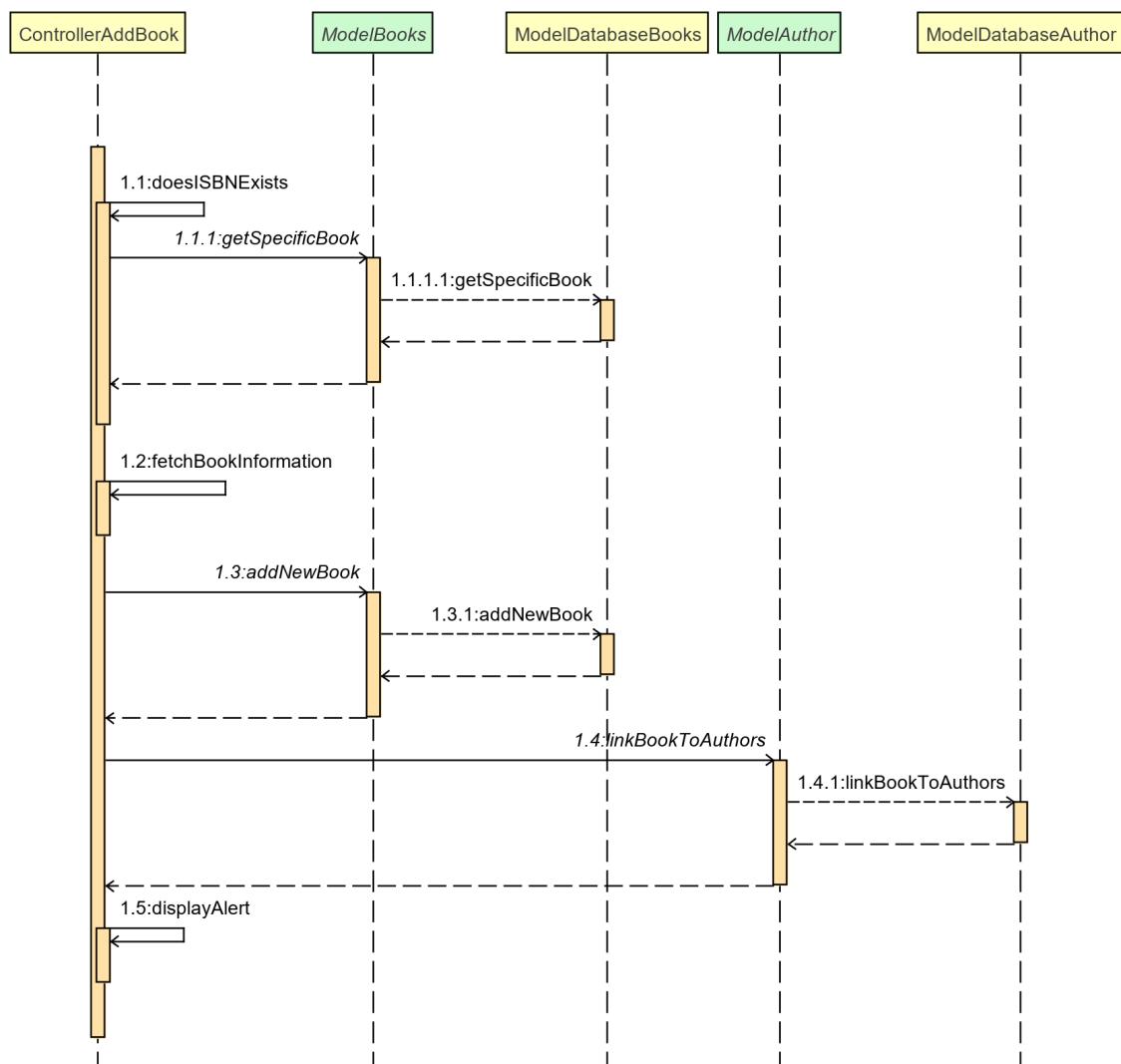


Figura 3.14: Diagramma sequenza utente responsabile aggiunta nuovo libro al catalogo

LD Books

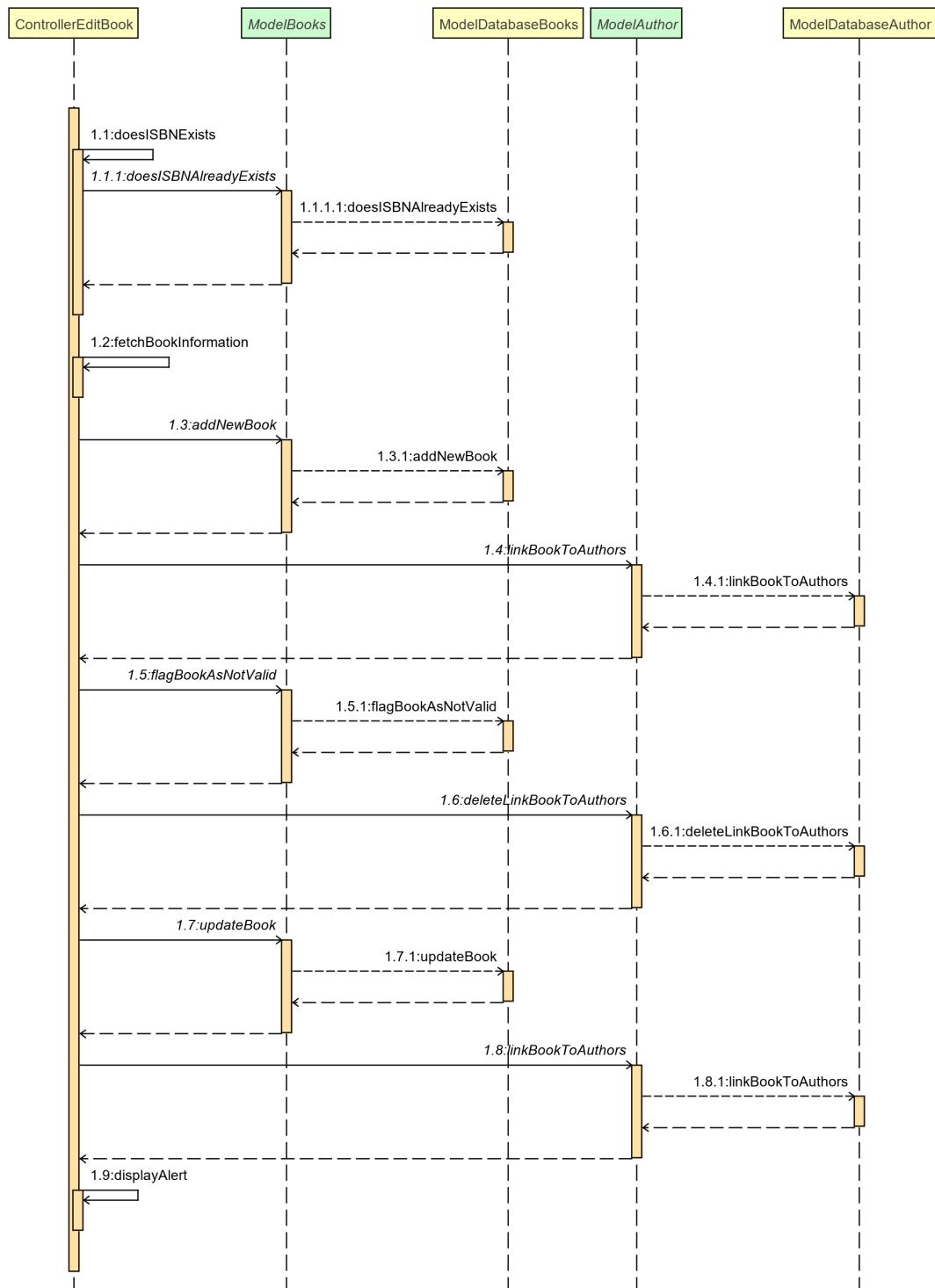


Figura 3.15: Diagramma sequenza utente responsabile aggiorna informazioni libro esistente

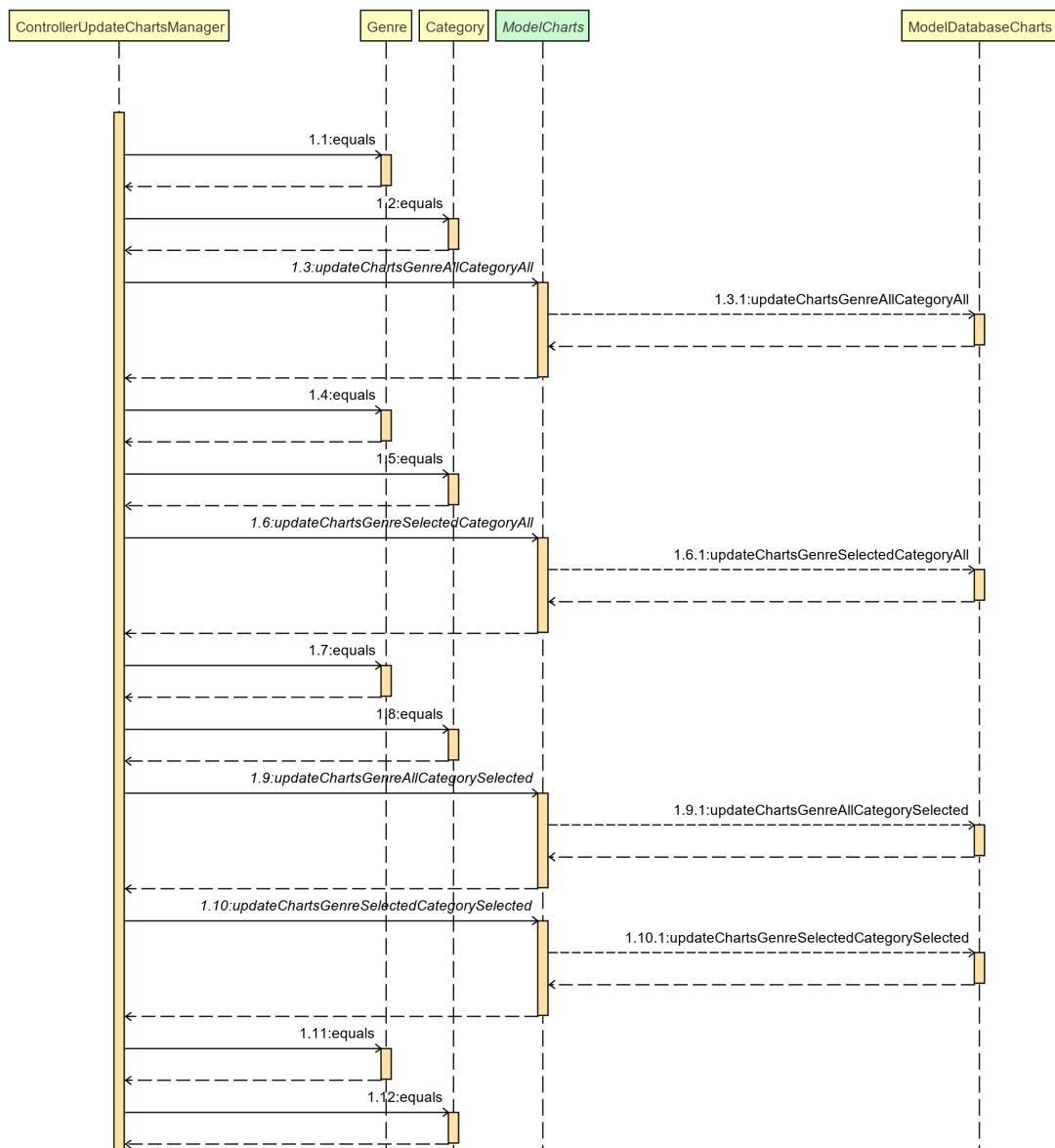


Figura 3.16: Diagramma sequenza utente responsabile aggiorna classifiche libro

Continua nella seguente pagina

LD Books

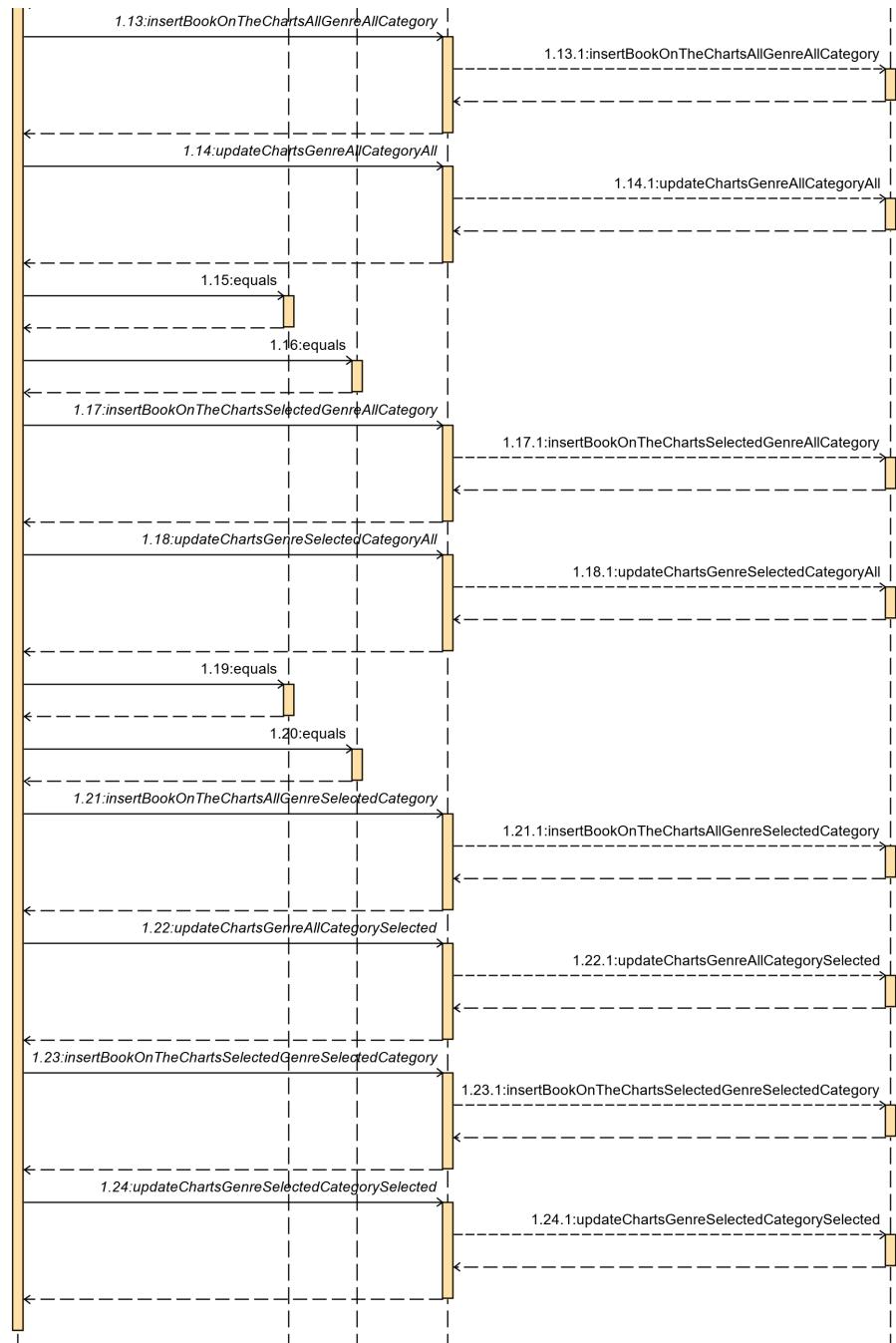


Figura 3.17: Diagramma sequenza utente responsabile aggiorna classifiche libro

Capitolo 4

Diagrammi delle attività

Vengono ora presentati i principali diagrammi delle attività:

4.1 Diagramma attività modifica dati anagrafici utente registrato

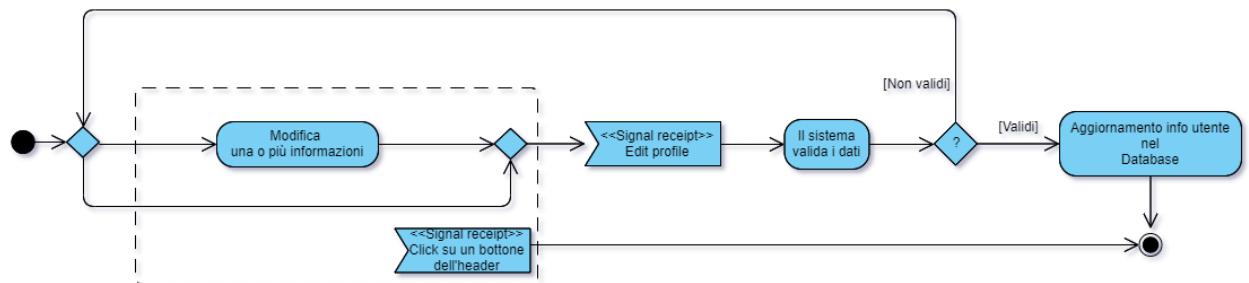


Figura 4.1: Diagramma attività modifica dati anagrafici utente registrato

4.2 Diagramma attività registrazione utente

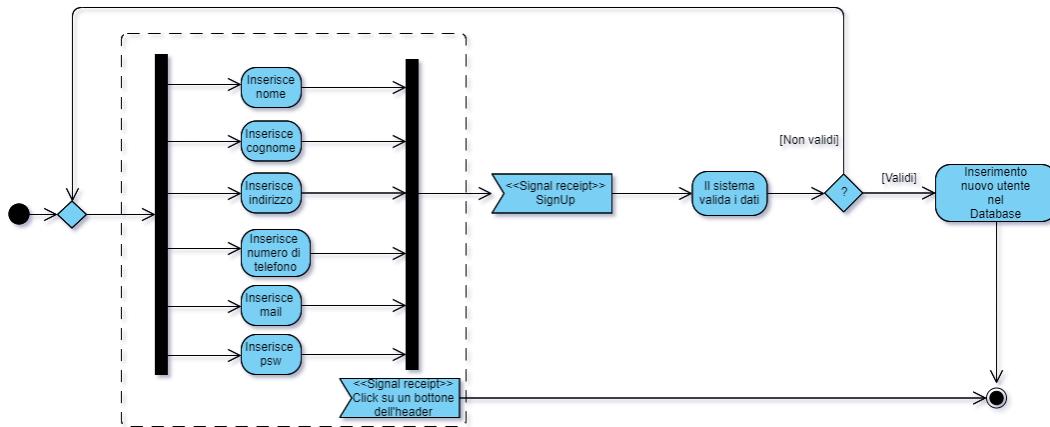


Figura 4.2: Diagramma attività registrazione utente

4.3 Diagramma attività aggiunta di un libro al carrello

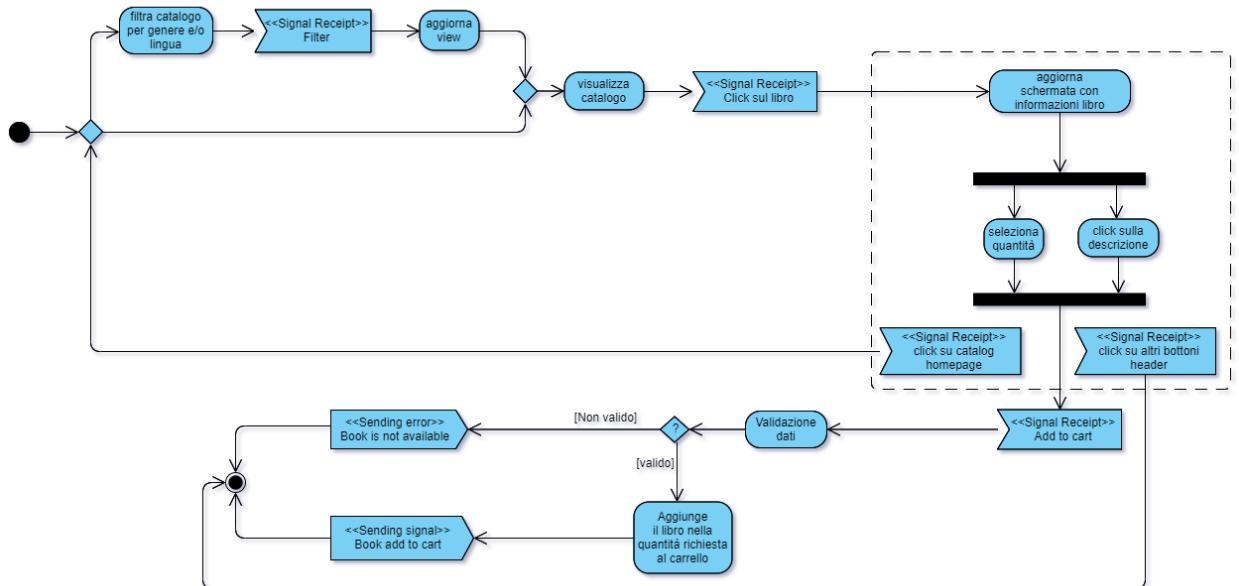


Figura 4.3: Diagramma attività aggiunta libro al carrello

4.4 Diagramma attività creazione di un ordine

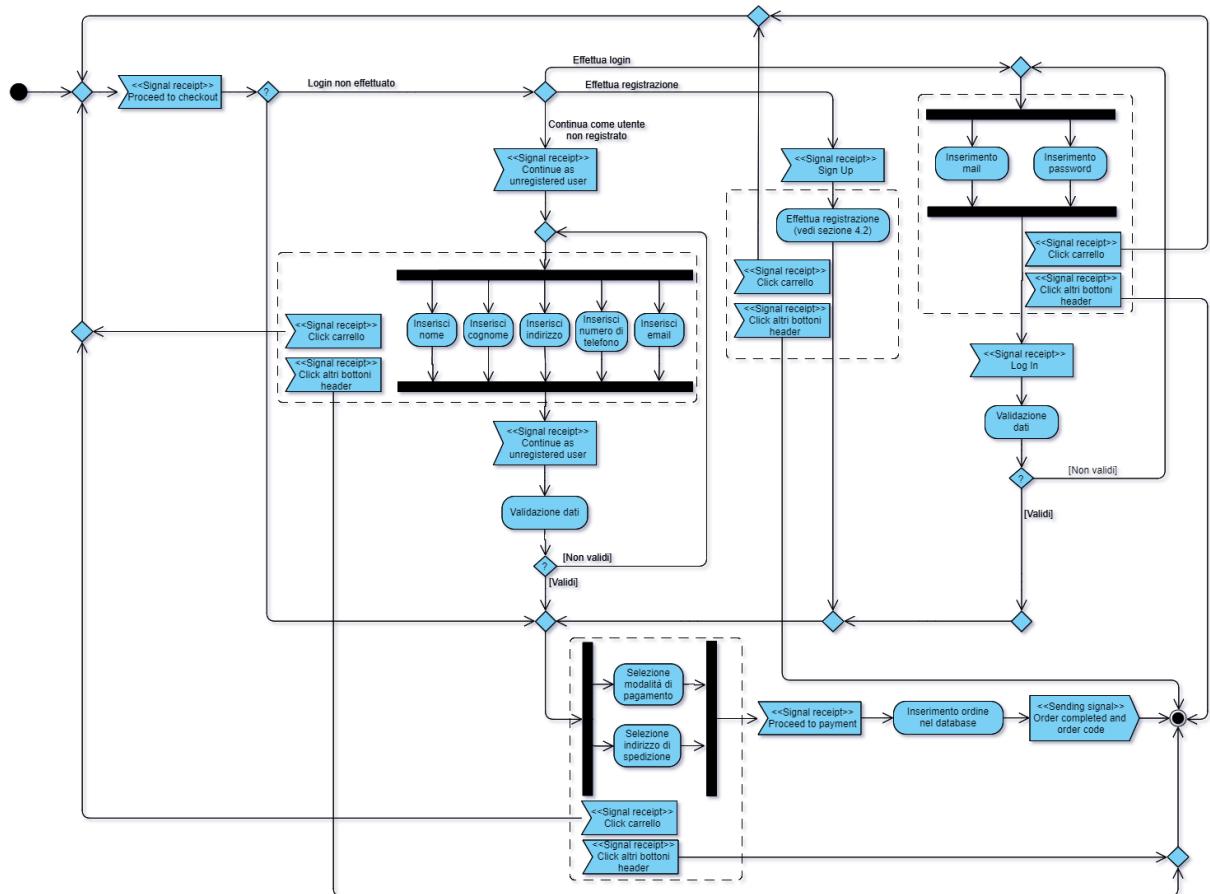


Figura 4.4: Diagramma attività creazione ordine

4.5 Diagramma attività modifica libro catalogo

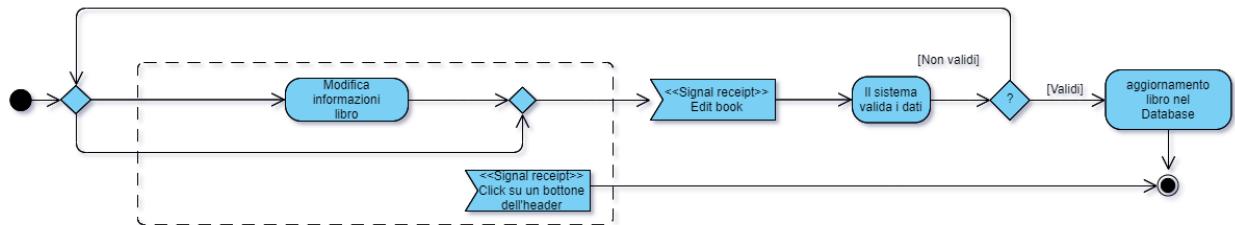


Figura 4.5: Diagramma attività modifica libro catalogo

4.6 Diagramma attività aggiunta libro al catalogo

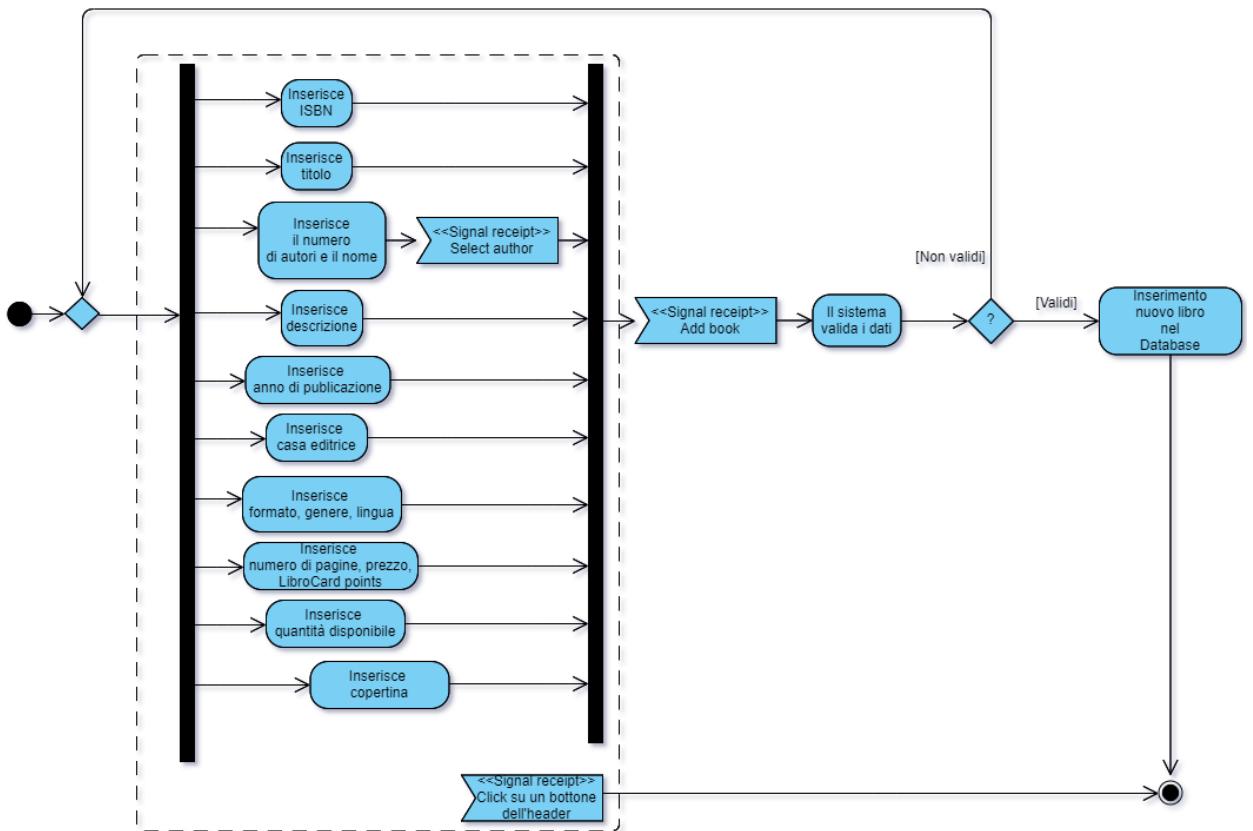


Figura 4.6: Diagramma attività aggiunta libro al catalogo

4.7 Diagramma attività aggiornamento classifiche

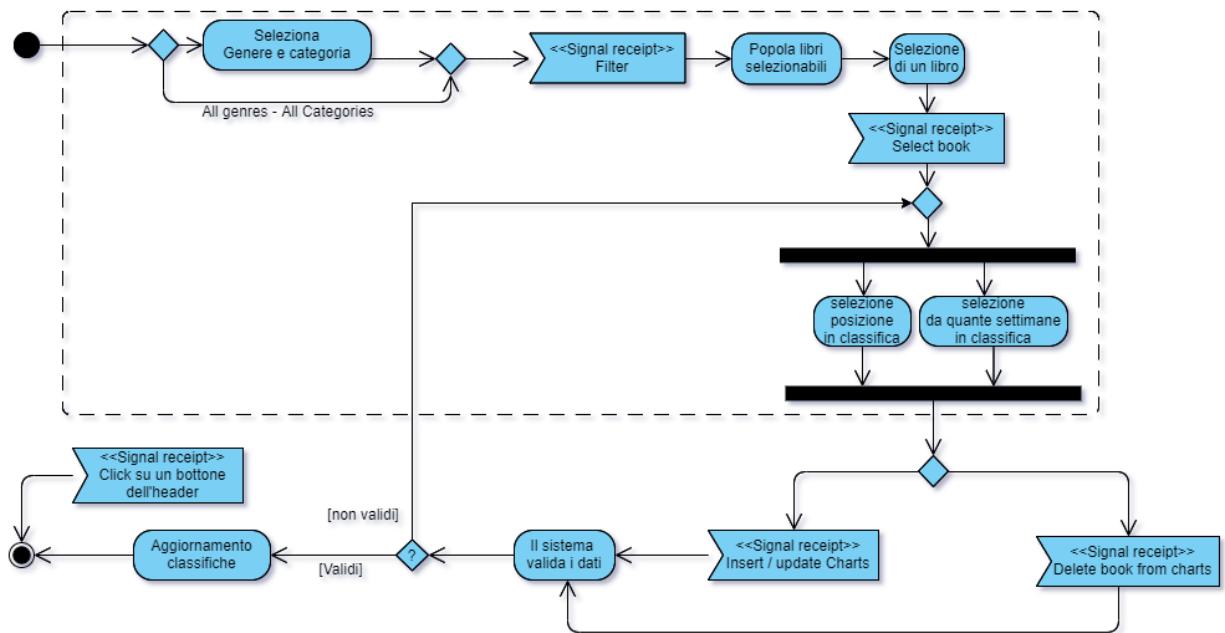


Figura 4.7: Diagramma attività aggiornamento classifiche

Capitolo 5

Diagrammi delle classi

Di seguito i diagrammi delle classi.
Sono stati divisi per *package* per facilitare la comprensione.

5.1 Diagramma delle classi del *Main*

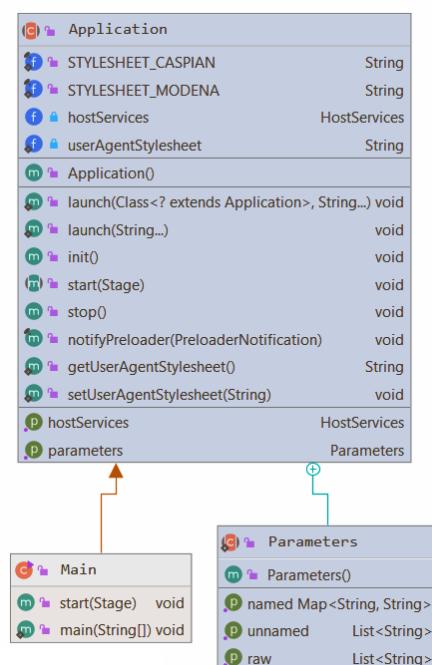


Figura 5.1: Diagramma classe Main

5.2 Diagramma delle classi del package *Controller*

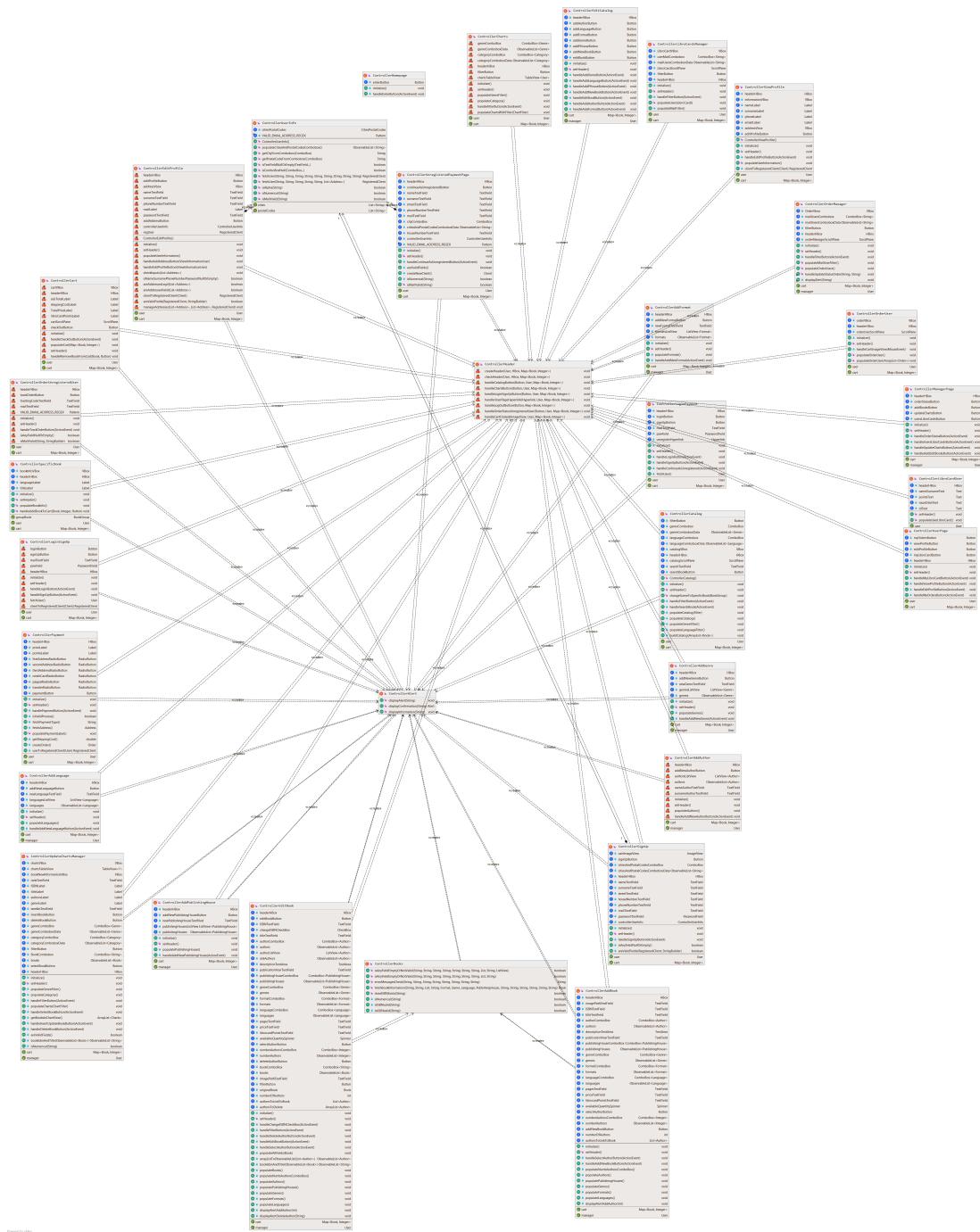


Figura 5.2: Diagramma classi Controller

5.3 Diagramma delle classi del package Model

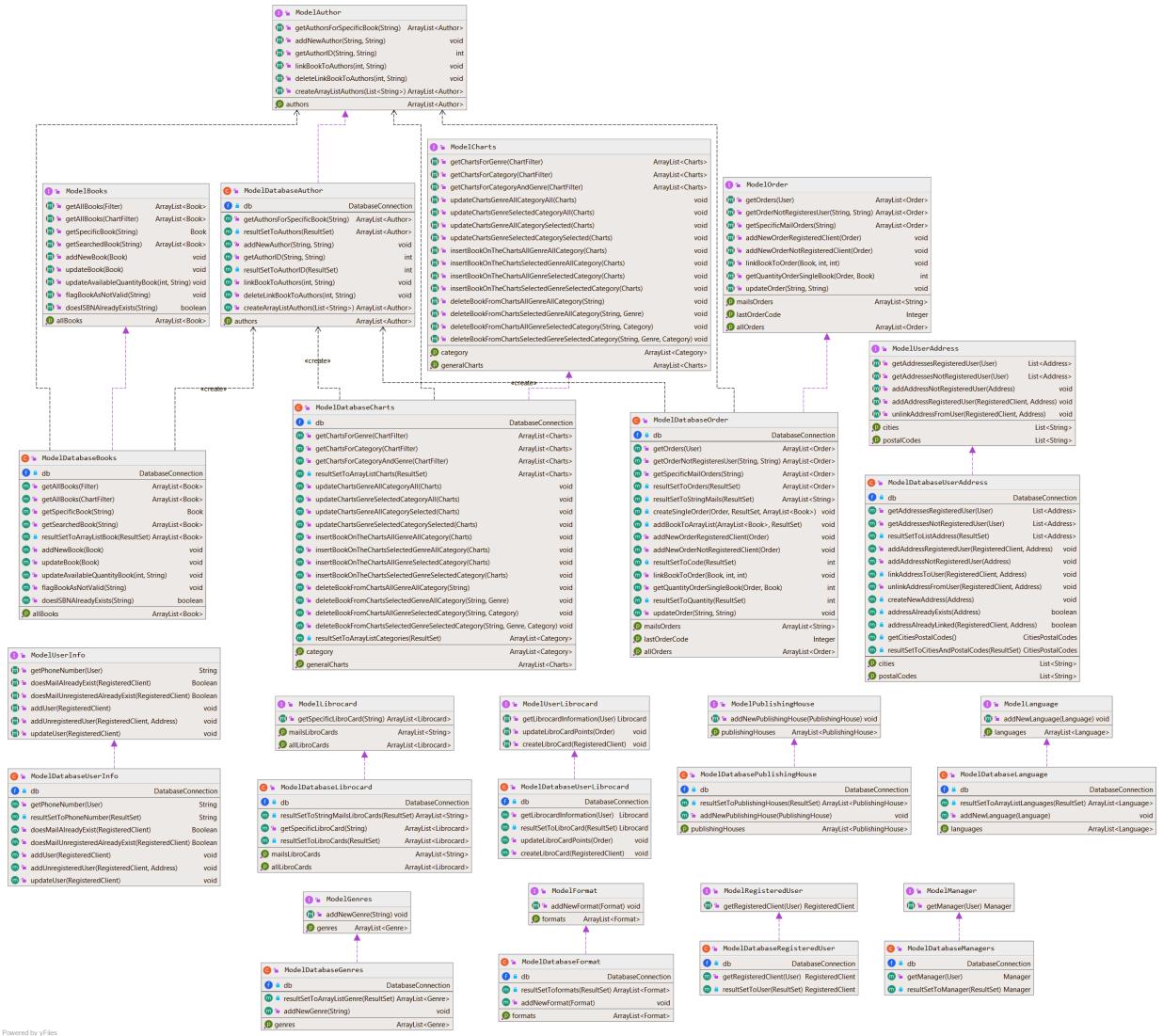


Figura 5.3: Diagramma classi Model

5.4 Diagramma delle classi del package View

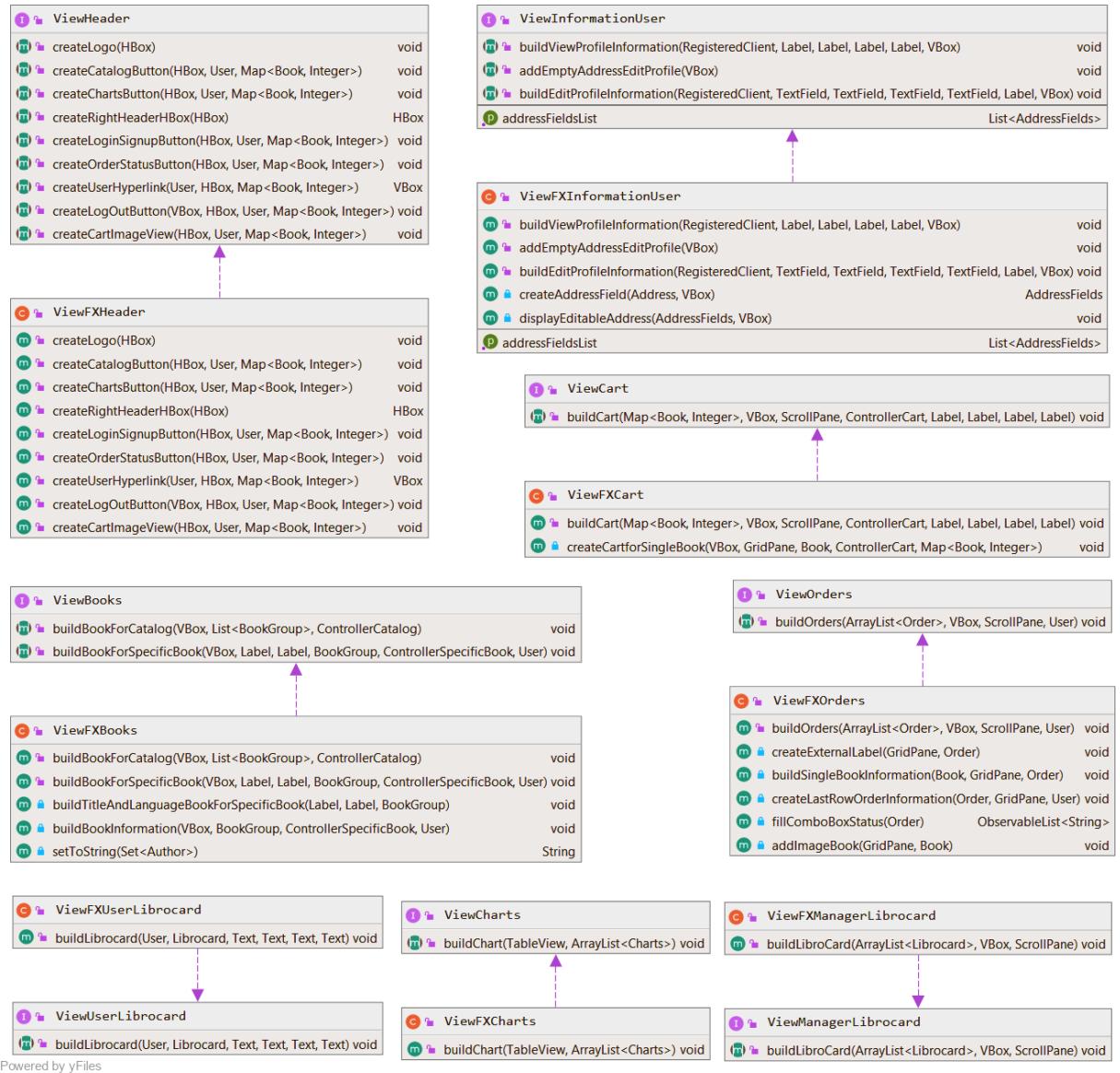


Figura 5.4: Diagramma classi View

5.5 Diagramma delle classi del package Data

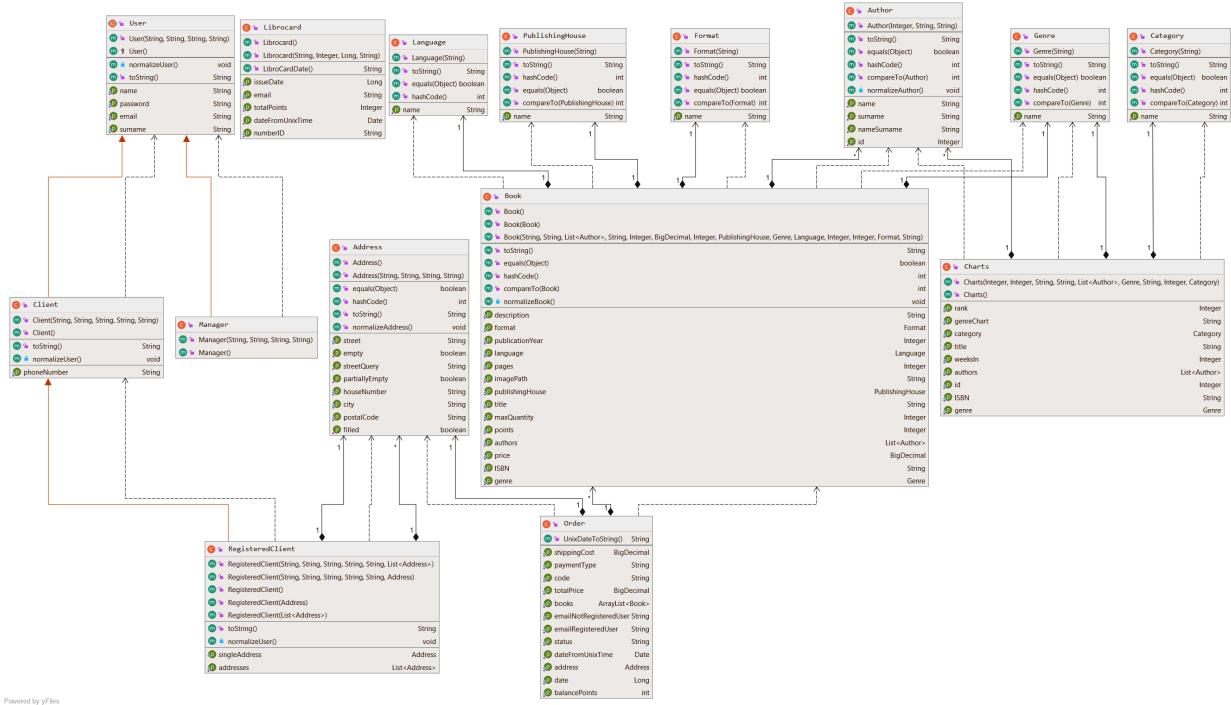


Figura 5.5: Diagramma classi Data

5.6 Diagramma delle classi del package *Utils*

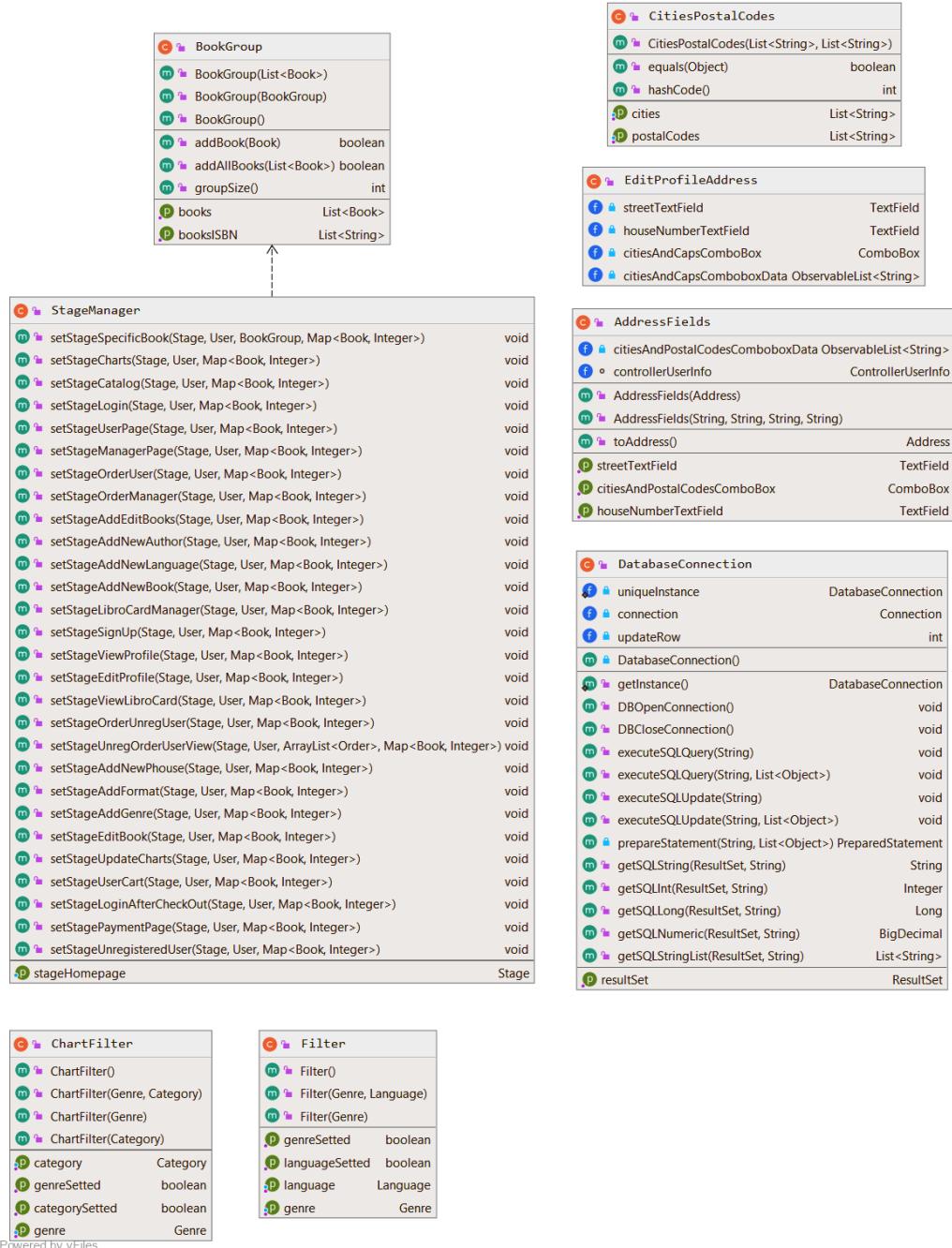


Figura 5.6: Diagramma classi Utils

Capitolo 6

Struttura del database

6.1 Scelte progettuali

La memorizzazione delle informazioni nell'applicativo avviene utilizzando una base di dati. Si è deciso di utilizzare **SQLite** come DBMS, in quanto presenta diversi vantaggi: l'applicativo è abbastanza piccolo e l'utilizzo di SQLite è efficace, in quanto presenta funzionalità sufficienti, senza bisogno di un DBMS che invece ne implementerebbe di aggiuntive; non vi è bisogno di un collegamento remoto alla base, in quanto questa è presente come file in locale; è più facile in fase di sviluppo apportare modifiche al database se si sta lavorando in gruppo, visto che ad ogni modifica è sufficiente ricondividere il file stesso.

Di seguito sono riportate le scelte progettuali:

- *Generi, lingue, case editrici, formati*: si è scelto di dividere tali attributi di un libro in entità separate, in quanto è facile aggiungerne di nuovi o cambiarli per tutti i libri. Inoltre, tutti i libri utilizzano queste entità, perciò era conveniente separarle dagli altri attributi.
- *Generi*: si è deciso che ogni libro potesse avere al più un genere, in quanto questo semplificava il filtraggio dei libri nel catalogo e la gestione delle classifiche. Di conseguenza, un genere può appartenere a più libri, ma un libro ha solamente un genere (cardinalità 1-N).
- *Autori*: gli autori sono memorizzati in un'entità *authors* separata, che ha come identificatore un attributo *idAuthor*: infatti non è stato possibile utilizzare *name* e *surname* come identificatori, visto che vi potrebbe essere omonimia tra autori.

Il collegamento tra *books* e *authors* è dato dall'entità *write*, la quale unisce l'autore che scrive uno specifico libro. Notiamo che è possibile che un libro sia scritto da più autori, e che un autore possa scrivere più libri. Questo è possibile grazie all'utilizzo dell'entità *write* (cardinalità N-N).

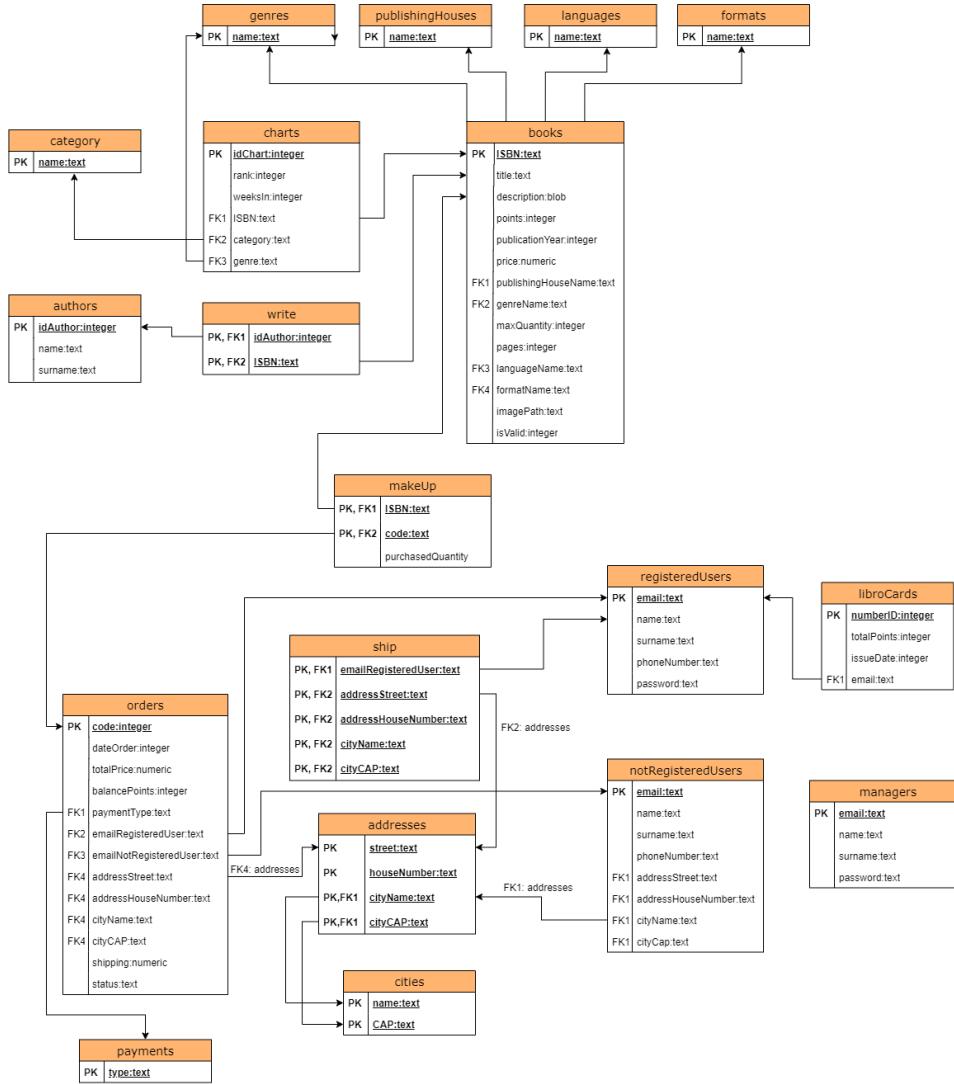
- *Utenti*: le specifiche del progetto distinguevano l'utente in tre ruoli: utente non registrato, utente registrato e responsabili. Si è deciso di risolvere la gerarchia di utenti come accorpamento degli attributi nei figli, ottenendo perciò tre entità: *notRegisteredUsers*, *registeredUsers*, *managers*, dove gli attributi in comune sono: *email*, *name*, *surname*. Gli attributi che *notRegisteredUsers* aggiunge sono: *phoneNumber*, *addressStreet*, *addressHouseNumber*, *cityName*, *cityCAP*.

Gli attributi che *registeredUsers* aggiunge sono: *phoneNumber* e *password*.

L'attributo che *manager* aggiunge è: *password*.

- *LibroCards*: la relazione che unisce *libroCards* e *registeredUsers* è di cardinalità 1-1. Nella risoluzione dello schema è stato quindi deciso di assegnare a *libroCards* la mail dell'utente registrato come vincolo d'integrità.
- *Indirizzi*: gli indirizzi sono contenuti nell'entità *addresses*. Si è deciso di separare questa entità dalle altre in quanto molte entità si riferivano agli indirizzi (come per esempio *orders* o *notRegisteredUser*). L'entità *ship* mantiene il collegamento tra indirizzi e utenti registrati, in modo che un utente registrato possa avere più indirizzi come da specifica, e che un indirizzo possa appartenere a più utenti (cardinalità N-N)

6.2 Schema Logico



6.3 Schema Relazionale

I vincoli di chiave primaria sono sottolineati.

- Addresses (street, houseNumber, cityName, cityCap)
- Authors (idAuthor, name, surname)

- Books (ISBN, title, description, points, publicationYear, price, publishingHouseName, genreName, maxQuantity, pages, languageName, formatName, imagePath, isValid)
- Category (name)
- Charts (idChart, rank, weeksIn, ISBN, category, genre)
- Cities (name, CAP)
- Formats (name)
- Genres (name)
- Languages (name)
- LibroCards (numberID, totalPoints, issueDate, email)
- MakeUp (ISBN, code, purchasedQuantity)
- Managers (email, name, surname, password)
- NotRegisteredUsers (email, name, surname, phoneNumber, addressStreet, addressHouseNumber, cityName, cityCAP)
- Orders (code, dateOrder, totalPrice, balancePoints, paymentType, emailNotRegisteredUser, emailRegisteredUser, addressStreet, addressHouseNumber, cityName, cityCAP, shipping, status)
- Payments (type)
- PublishingHouses (name)
- RegisteredUsers (email, name, surname, phoneNumber, password)
- Ship (emailRegisteredUser, addressStreet, addressHouseNumber, cityName, cityCAP)
- Write (idAuthor, ISBN)

I vincoli di integrità sono:

- Addresses.cityName, Addresses.cityCAP → Cities.name, Cities.CAP
- Books.formatName → Formats.name
- Books.languageName → Languages.name
- Books.genreName → Genres.name

- Books.publishingHouseName → PublishingHouses.name
- Books.genreName → Genres.name
- Charts.ISBN → Books.ISBN
- Charts.category → Category.name
- Charts.genre → Genres.name
- LibroCards.email → RegisteredUsers.email
- MakeUp.ISBN → Books.ISBN
- MakeUp.code → Orders.code
- NotRegisteredUsers.addressStreet,
NotRegisteredUsers.addressHouseNumber,
NotRegisteredUsers.cityName, NotRegisteredUsers.cityCAP →
Addresses.street, Addresses.houseNumber, Addresses.cityName,
Addresses.cityCAP
- Orders.emailRegisteredUser → RegisteredUsers.email
- Orders.emailNotRegisteredUser → NotRegisteredUsers.email
- Orders.addressStreet, Orders.addressHouseNumber,
Orders.cityName, Orders.cityCAP → Addresses.street,
Addresses.houseNumber, Addresses.cityName, Addresses.cityCAP
- Orders.paymentType → Payments.type
- Ship.emailRegisteredUser → RegisteredUsers.email
- Ship.addressStreet, Ship.addressHouseNumber, Ship.cityName,
Ship.cityCAP → Addresses.street, Addresses.houseNumber,
Addresses.cityName, Addresses.cityCAP
- Write.idAuthor → Authors.idAuthor
- Write.ISBN → Books.ISBN

Capitolo 7

Scelte progettuali

In questo capitolo presentiamo le scelte progettuali e i principali design pattern utilizzati al fine della realizzazione del sistema informativo richiesto.

7.1 Sviluppo

Abbiamo scelto di sviluppare il sistema informativo utilizzando JavaFX basato su Java in quanto è una libreria grafica relativamente recente, supportata ufficialmente e cross platform: se un domani volessimo trasferire il sistema (che in questo caso è stato sviluppato per desktop), su dispositivo mobile, potremmo farlo senza problemi in quanto l'interfaccia si adatterebbe senza incorrere in eventuali errori di visualizzazione.

JavaFX presenta poi ulteriori funzionalità aggiuntive rispetto a Swing, come ad esempio la possibilità di personalizzazione tramite CSS.

Abbiamo trovato inoltre molto utile l'utilizzo del software SceneBuilder, un designer di interfacce grafiche che è perfettamente compatibile con JavaFX. Tutte le interfacce grafiche risiedono in una cartella "fxml" e sono opportunamente caricate ed elaborate in JavaFX tramite apposite classi.

7.2 Metodologia di sviluppo

Il sistema è stato interamente sviluppato usando Git (<https://git-scm.com/>) come sistema di versionamento del codice e GitHub (<https://github.com/>) per l'hosting della repository.

Il sistema è stato sviluppato utilizzando una metodologia **Agile**: dopo aver analizzato i requisiti abbiamo deciso di suddividere la progettazione in macro aree:

- una prima area inerente allo memorizzazione dei dati, quindi ci siamo concentrati sullo sviluppo del database presentato al capitolo 6
- successivamente abbiamo analizzato con attenzione la parte logica e user interface, al fine di rispettare al meglio i requisiti.

Quest'ultimo punto è stato a sua volta suddiviso in task brevi che rispecchiassero le singole schermate sulle quale l'utente poi andava ad interagire. Avendo diviso il lavoro in task relativamente brevi, è risultato semplice testare e verificare immediatamente il corretto funzionamento di quanto implementato.

I design pattern, presentati nel prossimo paragrafo 7.3, sono stati decisi in fase di sviluppo del progetto e adattati al meglio alle caratteristiche richieste dal software.

In alcune occasioni è stato necessario un refactoring del codice al fine di migliorare la pulizia, la correttezza e le prestazioni del software.

Quest'ultimo punto è stato semplificato grazie all'utilizzo di diverse branch su GitHub, che ci hanno permesso di mantenere una cronologia delle modifiche eseguite e eventualmente scartare quelle ritenute non corrette.

 LM095	Added implementation of filter Button and fixed some bugs
 DebbyX3	Edited bookChart.fxml
 LM095	Added partial implementation of Login button
 DebbyX3	Added Model implementation of BookCharts and Genre Combobox
 DebbyX3	Added data in DB
 LM095	Try query login
 LM095	finished login button and started userPage

Figura 7.1: esempio interazione GitHub

7.3 Design Pattern utilizzati

Vengono presentati in questa sezione i principali pattern utilizzati.

Sono inoltre presentate le varie difficoltà incontrate e soluzioni adottate per la realizzazione degli stessi.

7.3.1 MVC pattern

Lo schema architetturale scelto per l'applicazione è quello del **MVC** (Model-View-Controller).

Come presentato nel Diagramma delle Sequenze e delle Classi, le tre componenti di questo pattern comunicano tra loro.

La View genera eventi che vengono mandati al Controller corrispondente, il quale contiene la logica applicativa: questo decide, infatti, se interfacciarsi con il Model, per richiedere o salvare dati, oppure se comandare altre azioni alla View.

Schematicamente possiamo vedere il flusso dell'esecuzione come segue:

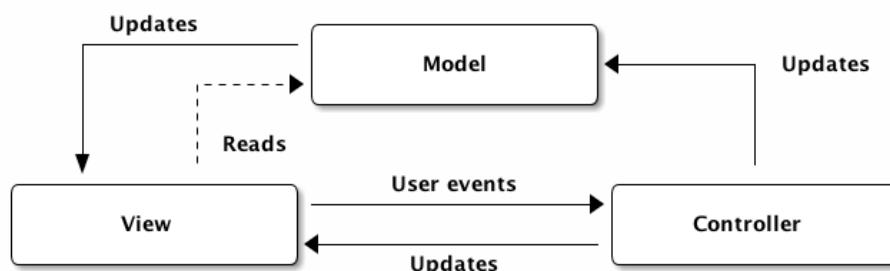


Figura 7.2: MVC design pattern

Possiamo notare che, in alcuni casi, la View si interfaccia direttamente con il Model per recuperare delle informazioni al fine di aggiornare la View, in tutti gli altri casi però, è il Controller a fare da "ponte" tra Model e View. Nel nostro progetto abbiamo tutti questi possibili casi implementati.

7.3.2 Facade pattern

Al fine di nascondere la complessità computazionale del DB, inizialmente avevamo deciso di utilizzare il **Facade** come pattern per la gestione dei Model.

Tuttavia non è stato possibile utilizzare questo pattern in quanto, avendo parecchi metodi e classi Model specifici per ogni esigenza, avremmo avuto il seguente problema: creando un'unica interfaccia Model con all'interno tutti i metodi non implementati, ogni volta che si creava una classe concreta "ModelSpecifico", quest'ultima era costretta, per definizione di implementazione di interfaccia, ad implementare, tramite Override, tutti i metodi dell'interfaccia Model, anche se quei metodi non la interessavano.

La soluzione allora sarebbe stata quella di non fare l'Override dei metodi non interessati e lasciare quindi il metodo di "default" dell'interfaccia. Quest'ultima soluzione però risulta, in primis poco corretta e successivamente soggetta a potenziali generazioni di errori nel codice.

Viene riportato un esemplificazione del problema riscontrato.

Seguendo la struttura standard del facade pattern avremmo dovuto avere una struttura analoga:

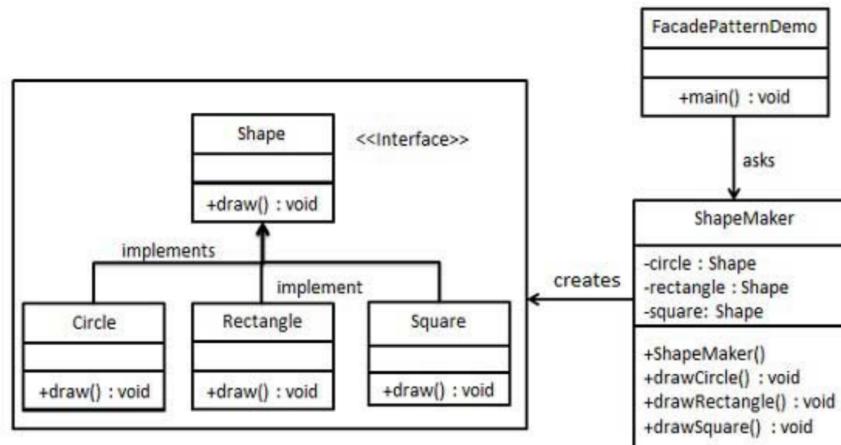


Figura 7.3: Struttura facade

Dove sostanzialmente la nostra interfaccia Model era:

```
public interface Model{
    public ArrayList<Book> getAllBooks();
    public ArrayList<Charts> getGeneralCharts();
    public void addNewLanguage(Language language);
    public Boolean doesMailAlreadyExist(RegisteredClient user);
    .....
    .....
    .....
}
```

Figura 7.4: esempio possibile interfaccia progetto

e le varie classi concrete erano:

```
public class ModelBooks implements Model {
    @Override
    public ArrayList<Book> getAllBooks(){
        .....
    }

    @Override
    public Boolean doesMailAlreadyExist(RegisteredClient user){
        .....
    }

    .....
    .....
    .....
}
```

Figura 7.5: esempio possibile interfaccia progetto

Il problema sopra menzionato è ben visibile nell'immagine qui sopra riportata: avere un'unica interfaccia con tutti i metodi inerenti all'utilizzo del database non era possibile. Infatti, le varie classi concrete avrebbero dovuto implementare, tramite Override, anche metodi non specifici per quella

determinata classe, come ad esempio il metodo "doesMailAlreadyExist" nell'immagine sopra, che riguarda ModelUser piuttosto che ModelBook.

Era possibile non implementare il metodo che non interessa direttamente la classe, ma questo risulta essere, come già detto in precedenza, poco corretto (visto che si sarebbe dovuto scrivere un metodo *default* vuoto nell'interfaccia) e soggetto a possibili errori a runtime.

La soluzione a questo problema quindi è l'implementazione del pattern DAO, a cui si rimanda.

7.3.3 DAO pattern

I Model seguono l'implementazione base del pattern **DAO** (Data Access Object).

Il pattern DAO risulta particolarmente adatto, in quanto è stato pensato appositamente per accedere a dati persistenti. Visto che l'accesso alle informazioni dipende dalla sorgente nella quale sono conservate, e dal tipo di memorizzazione (file, database relazionali oppure object-oriented), il pattern DAO ne astrae e incapsula l'accesso, in modo che le altre classi non sappiano con che tecnologia sono memorizzate. Inoltre, gestisce la connessione con la sorgente per ottenere e memorizzare i dati.

Le classi utilizzate per implementare il pattern sono di tre tipi:

- una classe di tipo *entity*, che contiene tutte le informazioni del dato da prelevare o memorizzare;
- una classe di tipo *Model interface*, la quale funge da interfaccia verso l'esterno. Non importa che tecnologia si sta usando per prelevare l'informazione, in quanto questa è nascosta dall'interfaccia stessa, che fornisce solo i metodi per il recupero o la memorizzazione del dato;
- una classe concreta che implementa l'interfaccia appena descritta, di tipo *Model*. È qui che si specifica come collegarsi alla sorgente dati e come accedere ad essa per eseguire le operazioni. Sono implementati tutti quei metodi che l'interfaccia corrispondente richiede.

Facciamo un esempio utilizzando la LibroCard.

Tutte le operazioni che riguardano le LibroCard passano per una classe *LibroCard* che contiene email, punti, data di creazione e via dicendo. Questa

è la classe di tipo *entity*, ovvero la prima.

Successivamente vi è la classe di interfaccia per le LibroCard, che chiameremo *ModelLibroCard*, e conterrà metodi *getAllLibroCards()* o *addLibroCard()* ancora da implementare. Questa è la seconda vista nell'elenco, la *Model interface*.

Infine, la classe concreta *ModelDatabaseLibroCard* implementa i metodi dell'interfaccia, specificando come collegarsi alla base di dati e le query da effettuare. Corrisponde alla terza, il *Model*.

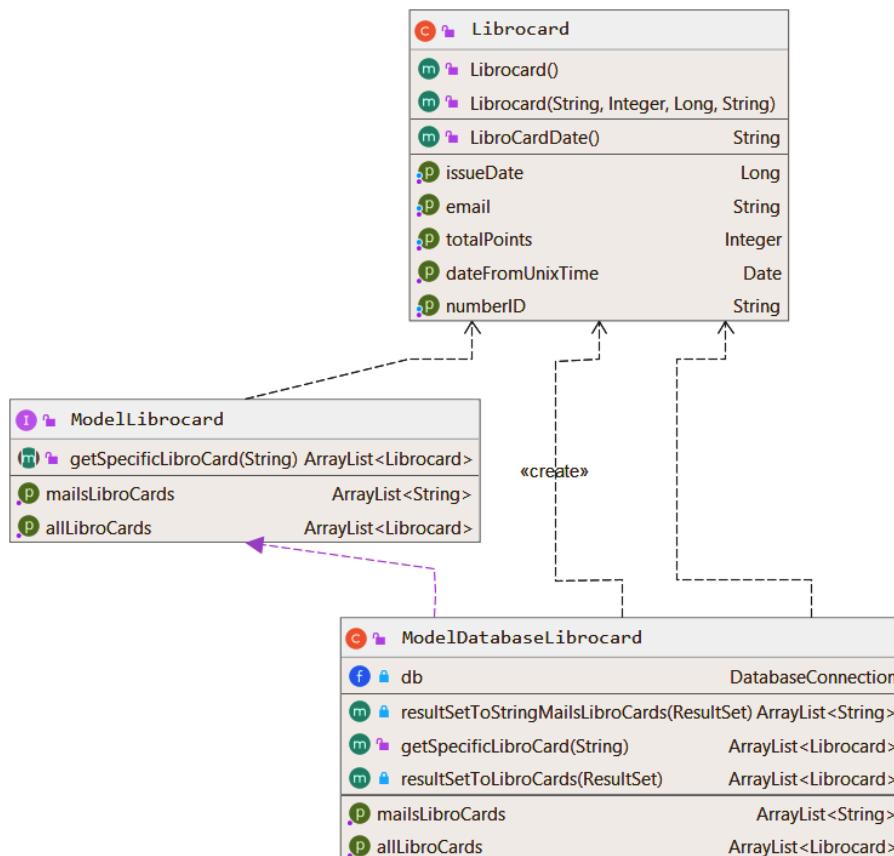


Figura 7.6: esempio implementazione DAO

La classe concreta *ModelDatabaseLibroCard* implementa l'interfaccia "dedicata" *ModelLibroCard*, la quale contiene solo i metodi inerenti alle Libro-

Cards.

Notiamo che, se avessimo utilizzato i file come supporto di memorizzazione, allora la classe concreta in questione avrebbe dovuto utilizzare i metodi corretti per accedere al file, mantenendo però invariata la struttura delle prime due classi, quella *entity* e quella *Model interface*. È questo il principale **vantaggio** dell'utilizzo del pattern DAO: se si modifica il supporto di memorizzazione è necessario soltanto implementare una nuova classe concreta (per esempio *ModelFileLibroCard*) che contiene i metodi per accedere alla nuova sorgente, senza però che le classi che si appoggiano ad essa cambino, o sappiano che tecnologia viene utilizzata.

7.3.4 Singleton pattern

Al fine di assicurare che la classe "DatabaseConnection" avesse una sola istanza ed un unico punto di accesso globale, questa è stata realizzata utilizzando il pattern singleton.

L'implementazione adottata è "lazy" ovvero l'istanza della classe viene creata solamente alla prima richiesta di utilizzo.

7.3.5 Observer pattern

Non abbiamo implementato un Observer da zero, ma abbiamo utilizzato quello messo a disposizione dalle librerie di JavaFX inerente ai listener dei buttoni, al fine di permettere la comunicazione tra View e Controller.

Abbiamo quindi i pulsanti delle varie View che sono gli observer, che stanno in ascolto e aspettano un determinato evento, in questo caso un click, e all'avvenimento dell'evento, una funzione di handler specifica per quel determinato bottone, va in esecuzione.

7.3.6 Iterator pattern

Anche in questo caso non abbiamo implementato un Iterator da zero ma abbiamo utilizzato quello messo a disposizione dalle librerie di Java.

In particolare è stato utilizzato per iterare su tutte le Collection utilizzate e per scorrere in maniera sequenziale i vari "result set" delle query effettuate al Database.

Capitolo 8

Validazione e Test

In questa parte ci siamo concentrati su due aspetti dettati dal test software: la verifica se l'applicativo raggiunge i risultati pianificati e la scoperta dei difetti. Per quanto riguarda l'ultimo caso, abbiamo forzato l'applicativo nei casi limite.

L'*ispezione del software* è stata svolta in modo statico, ovvero senza l'avvio diretto del programma, e riguarda i requisiti utente discussi precedentemente, oltre alla pianificazione della base di dati. È stato dedicato più tempo al *test software* dinamico, che comprende test di sviluppo, di rilascio e verso l'utente.

Nel **test di sviluppo** abbiamo effettuato test dei componenti e del sistema. In particolare, nel test dei componenti abbiamo testato le parti di codice che l'altra persona nel gruppo aveva scritto, e riguardava l'utilizzo delle interfacce, o il passaggio dei parametri. Il test di sistema è stato eseguito su tutto il programma da entrambi.

Il **test di rilascio** è stato eseguito da persone esterne, ma non ancora utenti finali, ovvero colleghi universitari tecnici, ma che non conoscevano com'era stato programmato il software.

Infine, nel **test utente** si sono coinvolti utenti finali, che nel nostro caso sono stati familiari e amici. È stato più precisamente un α -test, in quanto è stato svolto sulle nostre macchine.

8.1 Esempio caso d'uso - Ordine utente registrato

Di seguito mostriamo un caso d'uso da parte di un utente registrato.
All'avvio del programma si presenta la schermata iniziale.

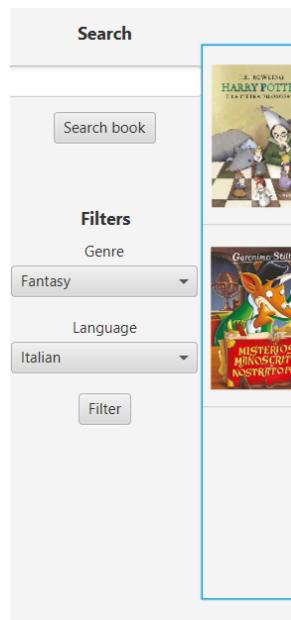


Appena si clicca il bottone, si viene reindirizzati al catalogo dei libri. Qui è possibile filtrare per genere e/o lingua, oppure ricercare il titolo di un libro.

A screenshot of the LD Books catalog page. The window title is "Catalog - LD Books". The interface includes a top navigation bar with icons for home, catalog, charts, login/signup, and order status. On the left, there is a sidebar with a search bar ("Search book"), filters for genre (set to "All") and language (set to "All"), and a "Filter" button. The main area is titled "Catalog" and shows three book entries in a table format:

Book Title	Author	Price
1984	by George Orwell	11.48€ - Paperback
Alan Turing. Storia di un enigma	by Andrew Hodges	15.30€ - Paperback
Ciclo delle Fondazioni	by Isaac Asimov	13.60€ - Paperback

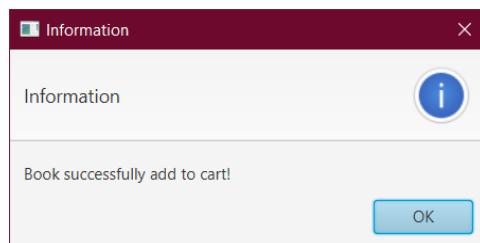
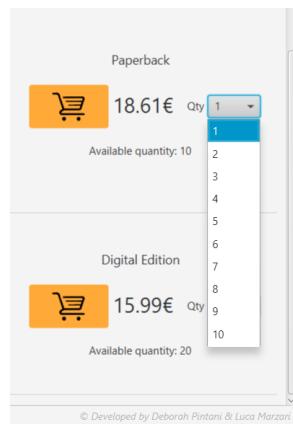
Each entry includes the book cover thumbnail, author, price, and genre information (Language and Genre). At the bottom right of the catalog area, there is a small copyright notice: "© Developed by Deborah Pintani & Luca Marzari".



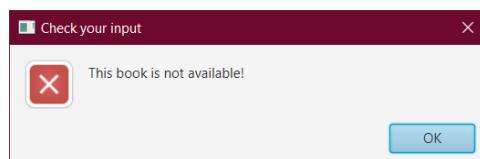
In qualsiasi caso, dopo aver filtrato o ricercato un libro, non appena si clicca su di esso è possibile visualizzare tutte le sue informazioni, comprese le varie tipologie di formati (copertina flessibile, rigida...) i quali sono raggruppati per libro.

The screenshot shows a product detail page for 'L'istituto' by Stephen King. At the top, there are navigation links: Catalog, Charts, Login - Sign Up, Order Status for unregistered user, and a shopping cart icon. The main title is 'L'istituto' and the language is listed as 'Italian'. The book cover is shown on the left. Key details listed include: ISBN: 978-8820068288, Authors: Stephen King, Genre: Mystery & Thriller, Publication Year: 2019, Publishing House: Sperling & Kupfer, Pages: 576, Librcards Point: 200, and Format: Paperback. To the right, there is a price of 18.61€ with a quantity selector set to 1. Below this, there is a 'Description' section with a note about a mysterious group of people introducing themselves to Luke Ellis. Another section for a 'Digital Edition' is also shown with a price of 15.99€ and a quantity selector set to 1. The footer of the page includes the text '© Developed by Deborah Pintani & Luca Marzari'.

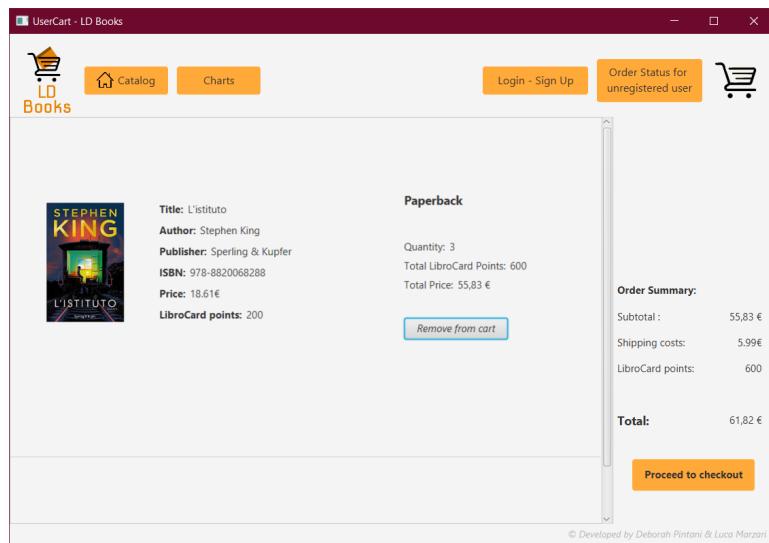
Da questa pagina è possibile scegliere quante copie del libro acquistare dal menu a tendina, e successivamente metterle nel carrello premendo il pulsante accanto.



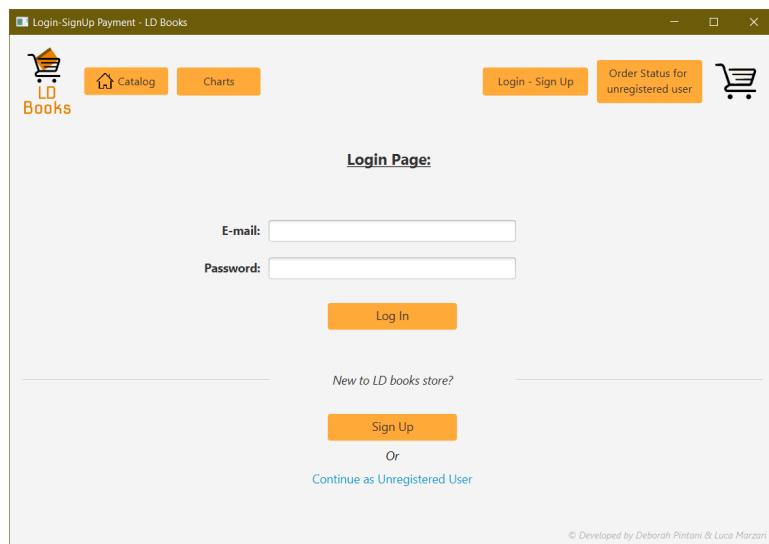
Se il libro non è disponibile perché non ci sono copie in magazzino, è mostrato un messaggio di errore.



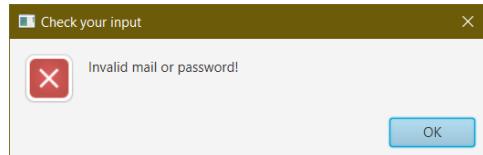
Cliccando sul pulsante del carrello in alto a destra di ogni schermata, si accede al proprio carrello. Sono mostrati i dati riepilogativi dei libri e dell'eventuale ordine, con il prezzo totale, i punti LibroCard e la spedizione.



È possibile rimuovere un libro dal carrello cliccando il pulsante "Remove from cart". Cliccando sul bottone "Proceed to checkout" si arriva alla schermata di Login nel caso questo non fosse stato effettuato precedentemente.

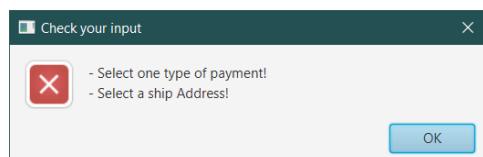


L'utente può quindi autenticarsi se possiede un account, registrarsi se non lo possiede oppure continuare come utente non registrato. Stiamo discutendo un caso d'uso di un utente già registrato, perciò si esegue il login. Se i dati inseriti non sono corretti, è mostrato un messaggio di errore.



Una volta effettuato l'accesso, si rimanda alla schermata di pagamento. Notiamo come anche l'header della schermata ci informi che siamo autenticati. È mostrato un riepilogo e si sceglie il tipo di pagamento e l'indirizzo al quale si desidera spedire.

Visto che entrambe le informazioni sono necessarie, se non vengono selezionate è mostrato un messaggio di errore.



Se l'ordine è andato a buon fine, allora è mostrato il seguente messaggio, con il codice dell'ordine.

