Università degli Studi di Verona

Online Library
Documentazione al prototipo

Magdalena M. Solitro

21 settembre 2020

Indice

1 Introduzione

L'obiettivo di questo progetto era quello di sviluppare un software che simulasse un'applicazione online di e-commerce per una libreria.
Questa relazione costituisce la documentazione del prototipo, in cui spiegheremo l'utilizzo del software e descriveremo le scelte progettuali e implementative adottate.

2 Requisiti

I requisiti del progetto sono stati specificati nella consegna, che riportiamo integralmente di seguito:

> *Si vuole progettare un sistema informatico per gestire gli acquisti on-line di una libreria.*
>
> *Per ogni libro si memorizza il titolo, l'autore o gli autori, la casa editrice, l'anno di pubblicazione, il codice ISBN (identificativo), il genere, il 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 cha 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.

3 UML

3.1 Use Cases

In questa sezione verranno presentate alcune schede dei casi d'uso principali. I casi d'uso riguardano uno dei tre attori che possono utilizzare il software:

- il responsabile di libreria (employee)
- l'utente registrato
- l'utente non registrato

Di seguito riportiamo gli *use case diagrams* di ogni attore.
Successivamente forniamo le schede di ogni specifico caso d'uso.

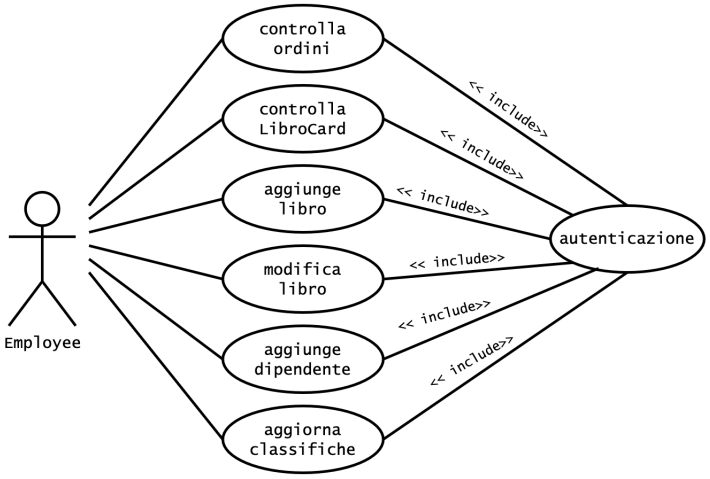


Figura 1: Use case diagram per il responsabile di libreria (employee)

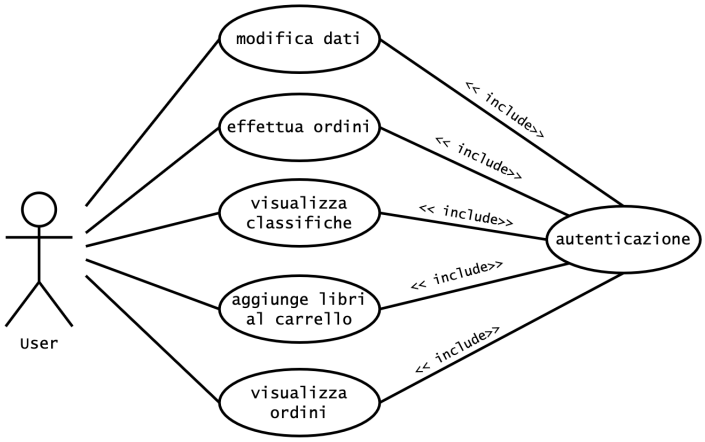


Figura 2: Use case diagram per l'utente registrato

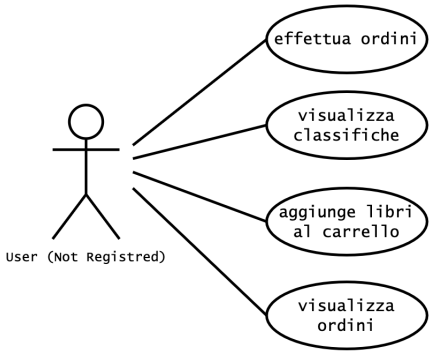


Figura 3: Use case diagram per l'utente non registrato

ID	UC1: Controllo ordini
Attori	Responsabile di libreria
Precondizioni	Il responsabile di libreria deve aver effettuato il log in
Sequenza	1) Dalla ChoiceBox, il responsabile filtra gli ordini in base allo stato 1.2) Se non esiste alcun ordine che si trova nello stato selezionato, viene presentato un avviso
Postcondizione	Viene visualizzata la LibroCard ricercata, se presente.

ID	UC2: Controllo LibroCard
Attori	Responsabile di libreria
Precondizioni	Il responsabile di libreria deve aver effettuato il log in
Sequenza	1) Dalla `MainPage` il responsabile clicca sul pulsante `Check LibroCards` 2) Dalla pagina di controllo delle LibroCard, il responsabile può decidere come effettuare la ricerca della specifica LibroCard: 2.1) Se decide di effettuare la ricerca in base all'e-mail dell'utente: 2.1.1) Allora seleziona il bottone `e-mail` 2.1.2) Inserisce l'e-mail dell'utente nel campo apposito 2.1.3 Clicca il pulsante `Search` 2.2) Se decide di effettuare la ricerca in base al codice univoco della carta (*cardID*): 2.2.1) Allora seleziona il bottone `cardID` 2.2.2) Inserisce la cardID nel campo apposito 2.2.3) Clicca il pulsante `Search`
Caso d'eccezione	Se i parametri di ricerca non corrispondono ad alcuna LibroCard registrata, si presenta un messaggio d'errore
Postcondizione	Viene visualizzata la LibroCard ricercata, se presente.

ID	UC3: Inserimento di un nuovo libro
Attori	Responsabile di libreria
Precondizioni	Il responsabile di libreria deve aver effettuato il log in
Sequenza	1) Dalla `MainPage` il responsabile clicca sul pulsante `Add a new book` 2) Il responsabile inserisce tutti i dati richiesti negli appositi campi 3) Il responsabile clicca sul pulsante `Add book` 4) Si apre un messaggio che informa la corretta aggiunta del libro
Caso d'eccezione	1) Se il responsabile non inserisce tutti i dati richiesti, viene presentato un messaggio d'errore e l'inserimento del libro viene bloccato 2) Se l'ISBN specificato corrisponde a quello di un libro già registrato nel database, viene presentato un messaggio d'errore e l'inserimento del libro viene bloccato
Postcondizione	Viene visualizzata la LibroCard ricercata, se presente.

ID	UC4: Modifica di un libro
Attori	Responsabile di libreria
Precondizioni	Il responsabile di libreria deve aver effettuato il log in
Sequenza	1) Dalla `MainPage` il responsabile clicca sul pulsante `Modify books` 2) Il responsabile inserisce l'ISBN del libro da modificare 3) Il responsabile inserisce solo i campi che desidera modificare 4) Se il responsabile clicca sul pulsante `Confirm changes`
Caso d'eccezione	Se l'ISBN non corrisponde a un libro registrato, viene presentato un messaggio d'errore
Postcondizione	Viene visualizzata la LibroCard ricercata, se presente.

ID	UC5: Inserimento di un nuovo dipendente
Attori	Responsabile di libreria
Precondizioni	1) Il responsabile di libreria deve aver effettuato il log in 2) Il responsabile deve avere ruolo di `General Director` oppure `Local Director`
Sequenza	1) Dalla `MainPage`, il responsabile clicca sul pulsante `Add a new Employee` 2) Il responsabile inserisce tutti i dati richiesti relativi al nuovo dipendente negli appositi campi 3) Il responsabile clicca sul pulsante `Register` 4) Se i dati inseriti sono tutti validi, si apre un messaggio che informa la corretta aggiunta del nuovo dipendente nel database
Caso d'eccezione	1) Se il responsabile non ha il ruolo di `General Director` o `Local Director`, viene presentato un messaggio d'errore e l'accesso alla funzione viene negato 2) Se il responsabile non ha riempito tutti i campi richiesti, viene presentato un messaggio d'errore e l'inserimento non viene effettuato
Postcondizione	Il nuovo dipendente viene aggiunto al database.

ID	UC6: Aggiornamento classifiche
Attori	Responsabile di libreria
Precondizioni	Il responsabile di libreria deve aver effettuato il log in
Sequenza	1) Dalla `MainPage`, il responsabile di libreria clicca il pulsante `Update Rankings` 2) Dalla pagina di aggiornamento classifiche, il responsabile clicca il pulsante `Update`
Postcondizione	Le classifiche sono state aggiornate

ID	UC7: Modifica dei dati personali
Attori	Utente registrato
Precondizioni	L'utente deve aver effettuato il log in
Sequenza	1) Dalla pagina del profilo, l'utente clicca il pulsante `Modify your data`, che porta alla pagina di modifica dei dati 2) L'utente riempie esclusivamente i campi che desidera modificare 3) Se l'utente clicca sul pulsante `Confirm changes`, si apre un messaggio che richiede la conferma della modifica: 3.1) Se l'utente clicca sul pulsante `OK` del dialog, i dati nel database vengono modificati e l'utente viene riportato nella pagina del profilo 3.2) Se l'utente clicca sul pulsante `Cancel`, il dialog si chiude e si resta nella pagina di modifica dei dati 4) Se l'utente clicca sul pulsante `Cancel` della pagina, si apre un messaggio che richiede la conferma dell'azione: 4.1) Se l'utente clicca il pulsante `OK` del dialog, si viene riportati nella pagina del profilo e i dati restano immutati 4.2) Se l'utente clicca il pulsante `Cancel` del dialog, si resta nella pagina di modifica dei dati
Postcondizioni	I dati dell'utente vengono modificati

ID	UC8: Creazione di un ordine
Attori	Utente registrato
Precondizioni	1. L'utente deve aver effettuato il log in 2. Il carrello dell'utente deve avere almeno un articolo
Sequenza	1) Dalla pagina del carrello, l'utente clicca il pulsante `Check Out` 2) L'utente seleziona un metodo di pagamento 3) L'utente sceglie l'indirizzo di spedizione, che può essere quello di domicilio o un altro indirizzo 3.a) Se l'utente sceglie un indirizzo diverso da quello del domicilio, allora deve specificare l'indirizzo nel campo apposito 4) L'utente clicca sul pulsante `Check Out`
Caso d'eccezione	1) Se il carrello è vuoto, viene presentato un messaggio d'errore che impedisce di procedere all'acquisto 2) Se l'utente non ha selezionato un metodo di pagamento o non ha specificato l'indirizzo alternativo, viene presentato un messaggio di errore che richiede l'inserimento dei dati mancanti
Postcondizioni	L'ordine viene confermato

ID	UC9: Visualizzazione classifiche
Attori	Utente generico
Precondizioni	
Sequenza	1) Dalla `MainPage`, l'utente clicca il pulsante `Rankings` 2) Dalla pagina delle classifiche, l'utente può selezionare un genere di cui visualizzare la relativa classifica
Postcondizione	

ID	UC10: Aggiunta di un libro al carrello
Attori	Utente generico
Precondizioni	
Sequenza	1) Dalla `MainPage`, l'utente seleziona un libro 2) L'utente viene portato nella pagina contenente tutte le informazioni relative al libro 3) Se l'utente desidera più di una copia di quel libro, può selezionare fino a dieci unità tramite lo Spinner a destra 4) L'utente clicca sul pulsante `Add To Cart` per aggiungere il libro
Caso d'eccezione	1) Se non ci sono più copie di quel libro, viene presentato un messaggio d'errore e il libro non viene aggiunto 2) Se il libro è già presente nel carrello, viene presentato un messaggio di errore e il libro non viene aggiunto
Postcondizione	Il libro è stato aggiunto al carrello

ID	UC11: Visualizzazione ordini
Attori	Utente registrato
Precondizioni	L'utente deve essere registrato
Sequenza	1) Se l'utente si trova nella `MainPage`, clicca sulla scritta `MY ORDERS` 2) Se l'utente si trova sulla pagina del profilo, clicca sul pulsante `MY ORDERS`
Postcondizioni	

ID	UC12: Ricerca di un ordine
Attori	Utente generico
Precondizioni	L'utente non deve aver effettuato il log in
Sequenza	1) Dalla `MainPage`, l'utente clicca sulla scritta `MY ORDERS` 2) L'utente inserisce il codice dell'ordine nel campo apposito 3) L'utente clicca sul pulsante `Search`
Caso d'eccezione	Se il codice inserito non corrisponde ad alcun ordine registrato, viene presentato un messaggio di errore
Postcondizioni	L'utente visualizza le informazioni relative al suo ordine, se questo è presente

ID	UC13: Creazione di un ordine
Attori	Utente non registrato
Precondizioni	Il carrello dell'utente deve avere almeno un articolo
Sequenza	1) Dalla pagina del carrello, l'utente clicca il pulsante `Check Out` 2) L'utente deve specificare: nome, cognome, metodo di pagamento, un indirizzo di spedizione e le informazioni di contatto nei campi appositi 3) L'utente clicca sul pulsante `Check Out`
Caso d'eccezione	1) Se il carrello è vuoto, viene presentato un messaggio d'errore che impedisce di procedere all'acquisto 2) Se l'utente non ha inserito tutti i campi richiesti, viene presentato un messaggio d'errore che impedisce di completare l'acquisto
Postcondizioni	L'ordine viene confermato

ID	UC15: Ricerca di un ordine
Attori	Utente non registrato
Precondizioni	
Sequenza	1) Dalla `MainPage`, l'utente clicca sulla scritta `MY ORDERS` in alto a sinistra 2) L'utente viene portato nella pagina di relativa al tracciamento degli ordini 3) L'utente inserisce nel campo apposito il codice univoco dell'ordine che desidera visualizzare 4) L'utente clicca sul pulsante di ricerca `Search`
Caso d'eccezione	Se il codice non corrisponde ad alcun ordine presente nel database, si apre un messaggio d'errore
Postcondizione	L'utente visualizza le informazioni relative all'ordine se questo è presente nel database.

3.2 Activity Diagram

Di seguito si riportano gli activity diagrams delle operazioni principali messe a disposizione dal programma. I diagrammi rappresentano il flusso di esecuzione che porta al compimento di una determinata funzione.

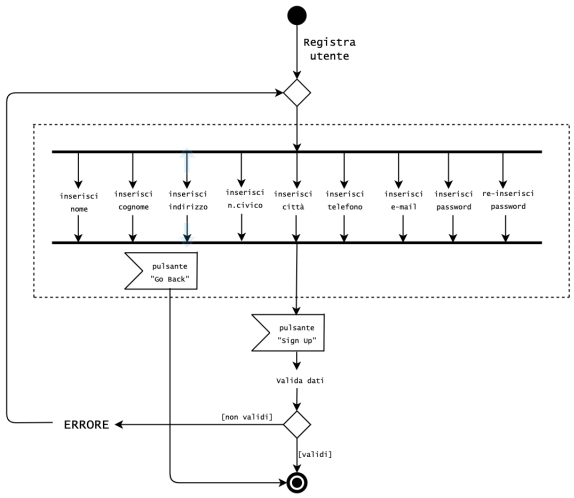


Figura 4: Activity diagram per la **registrazione di un utente**

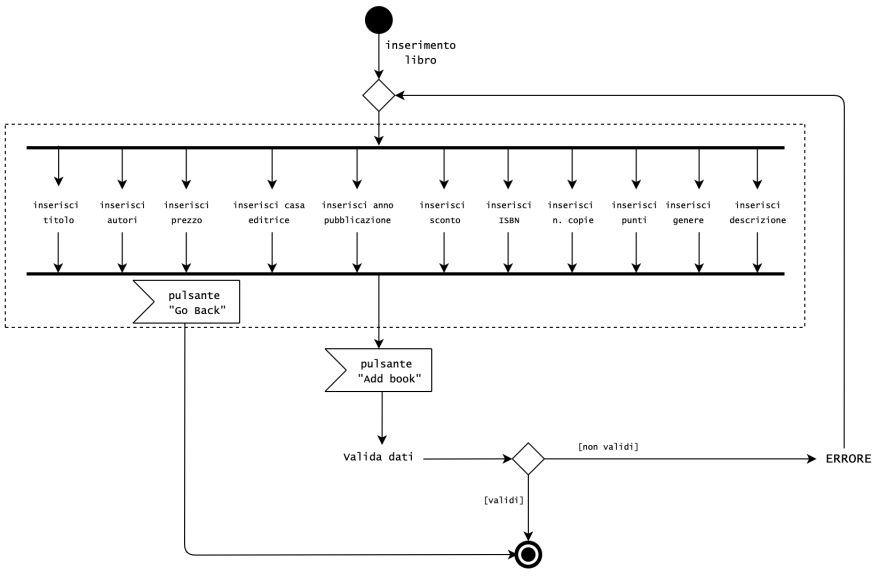


Figura 5: Activity diagram per l'**inserimento di un libro**

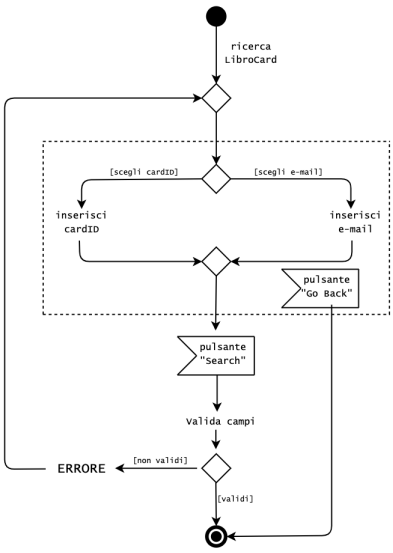


Figura 6: Activity diagram per la **ricerca di una LibroCard**

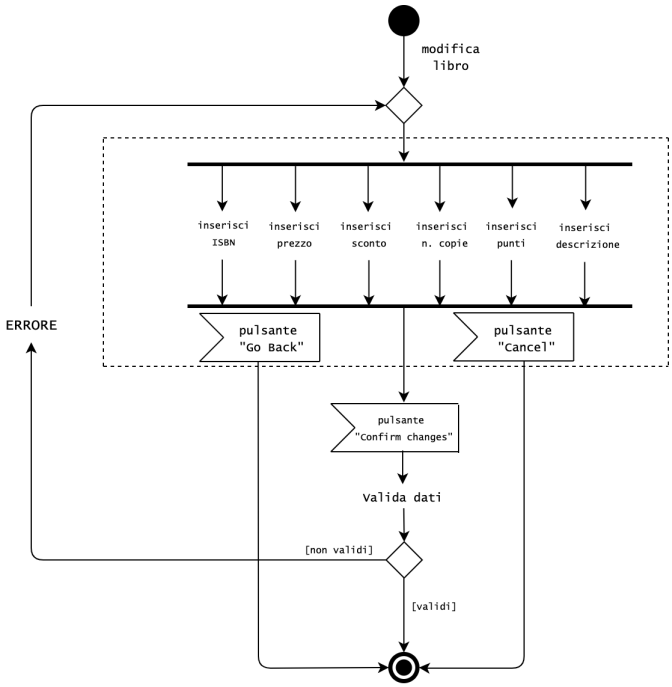


Figura 7: Activity diagram per la **modifica di un libro**

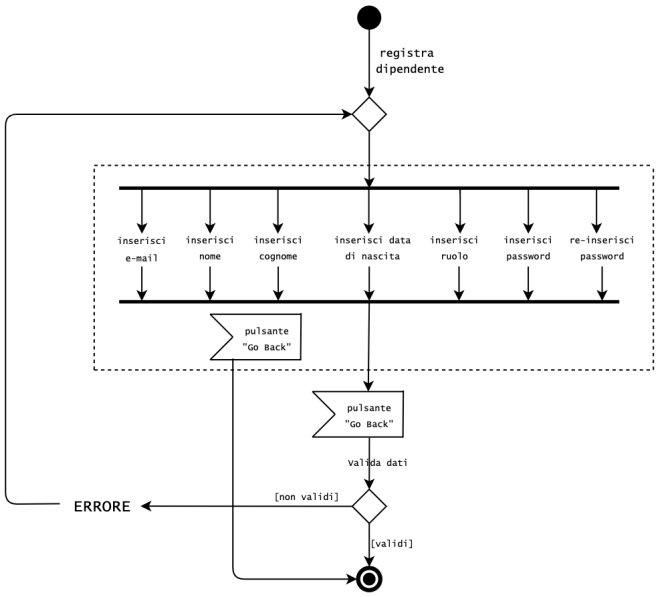


Figura 8: Activity diagram per l'**inserimento di un nuovo dipendente**

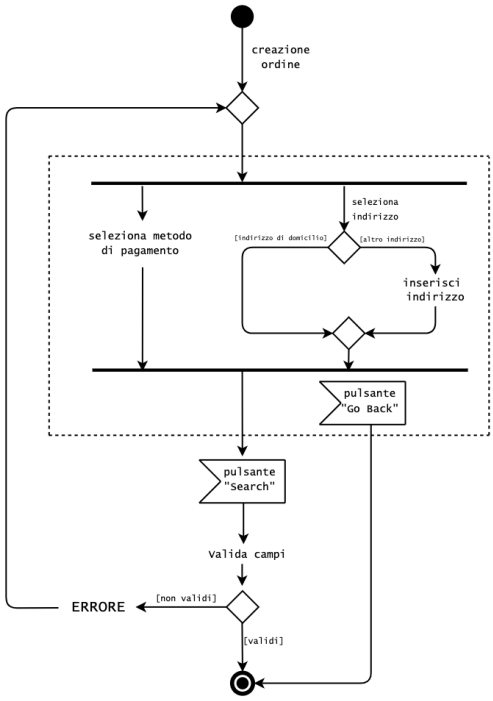


Figura 9: Activity diagram per la**creazione di un ordine** (utente)

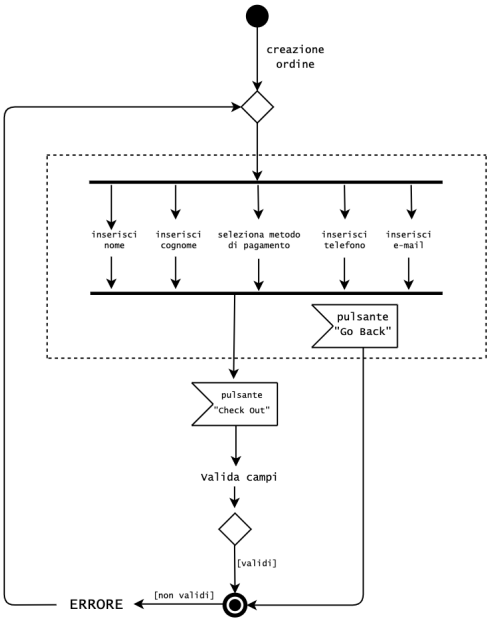


Figura 10: Activity diagram per la **creazione di un ordine** (utente non registrato)

3.3 Class Diagrams

In questa sezione si riportano i class diagrams, che rappresentano l'architettura del programma e aiutano a comprendere chiaramente come le varie classi interagiscono tra loro.

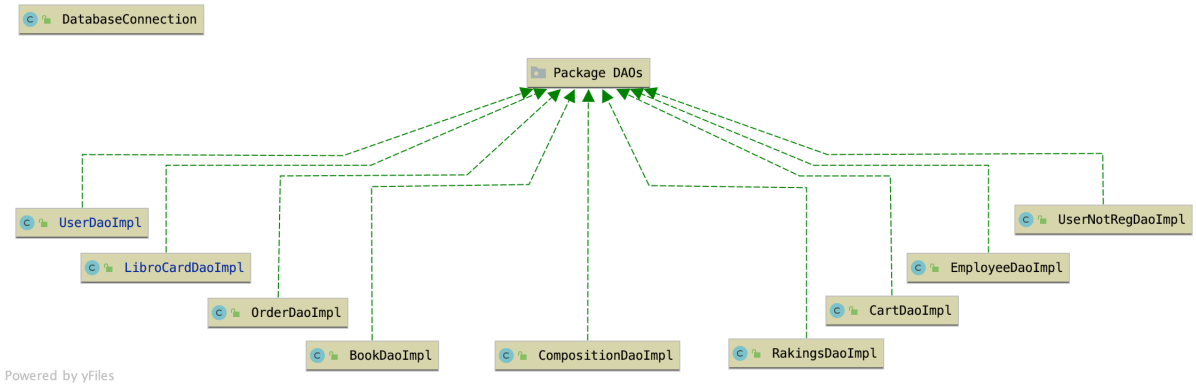


Figura 11: Class Diagram per il package Utils

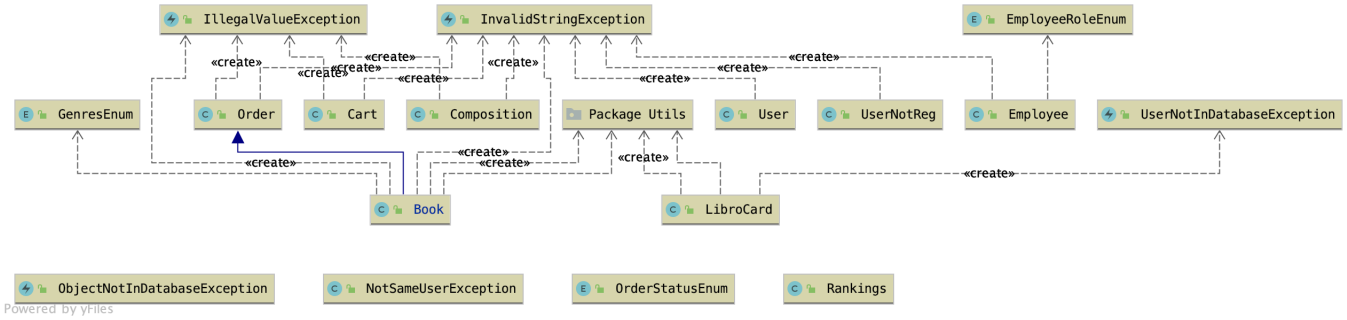


Figura 12: Class Diagram per il package Model

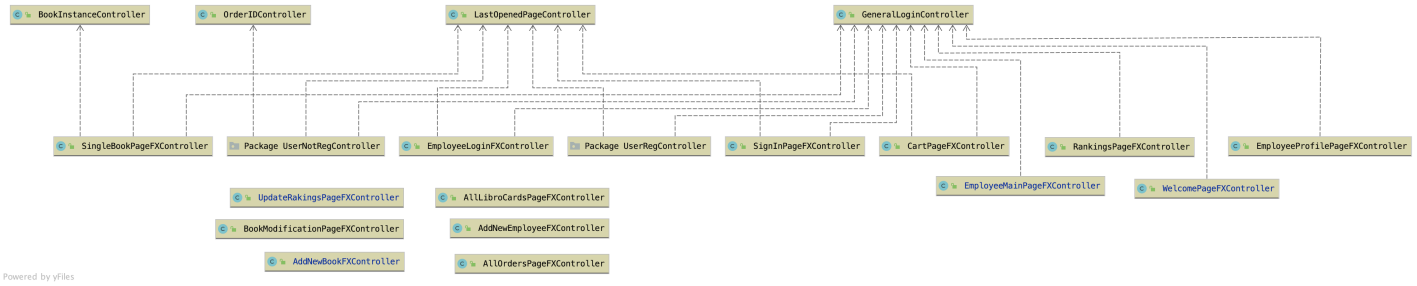


Figura 13: Class Diagram per il package Controller

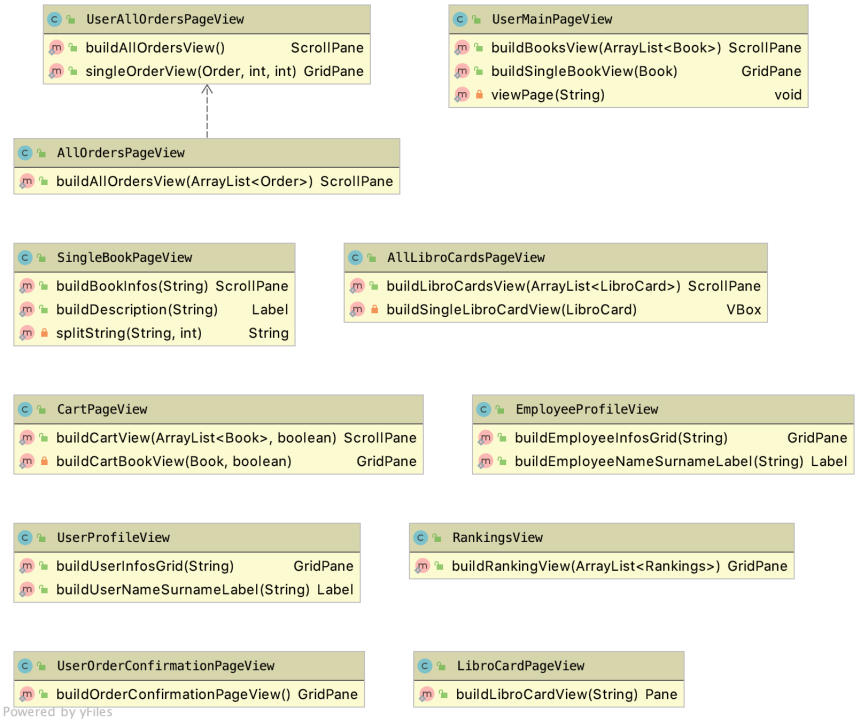


Figura 14: Class Diagram per il package View

3.4 Sequence Diagrams

In questa sezione riportiamo i sequence diagrams, che visualizzano lo scambio di informazioni tra le varie entità nel tempo: il loro scopo è mostrare come un certo comportamento viene realizzato dalla collaborazione delle entità in gioco.

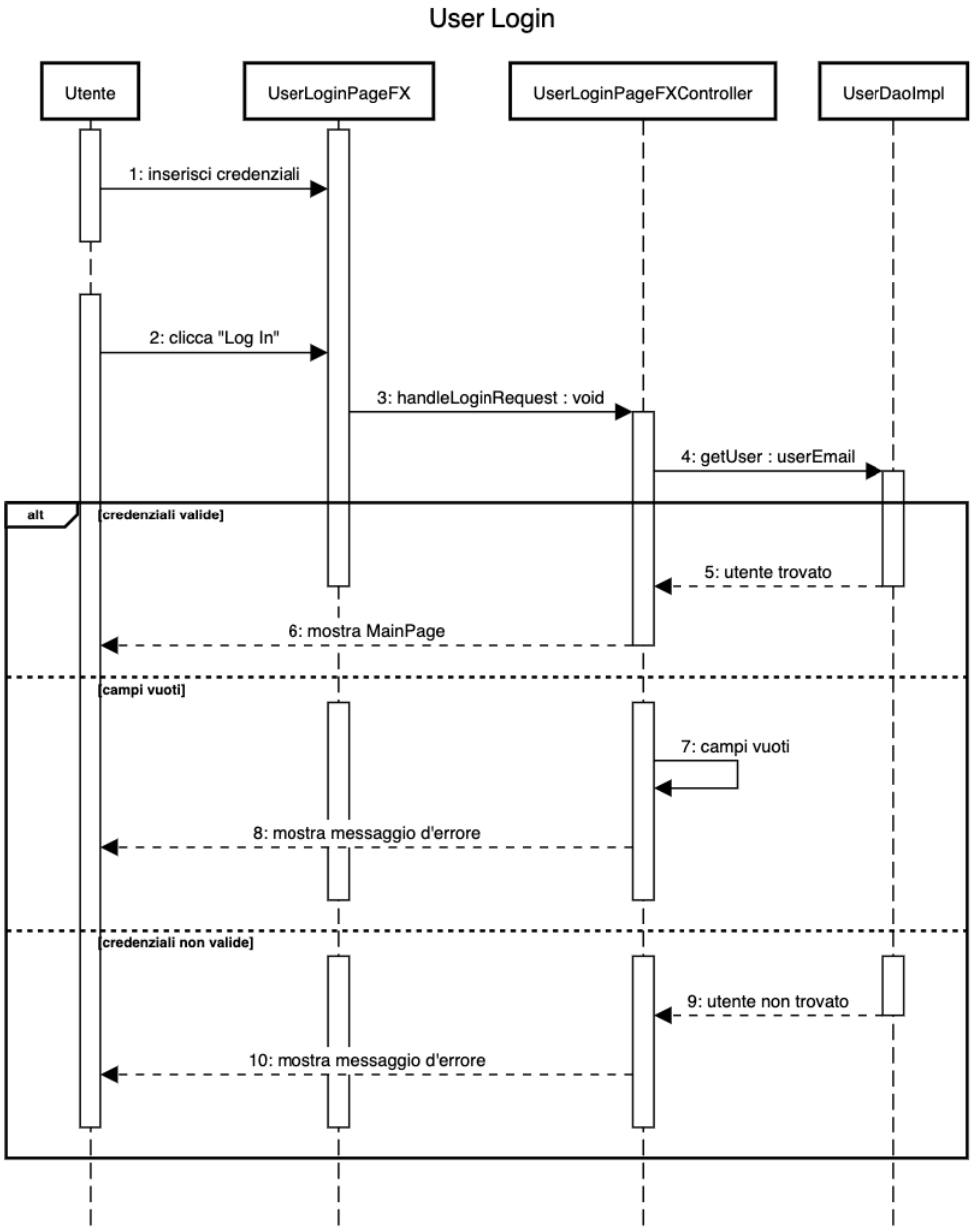


Figura 15: Sequence diagram per il **log in dell'utente**

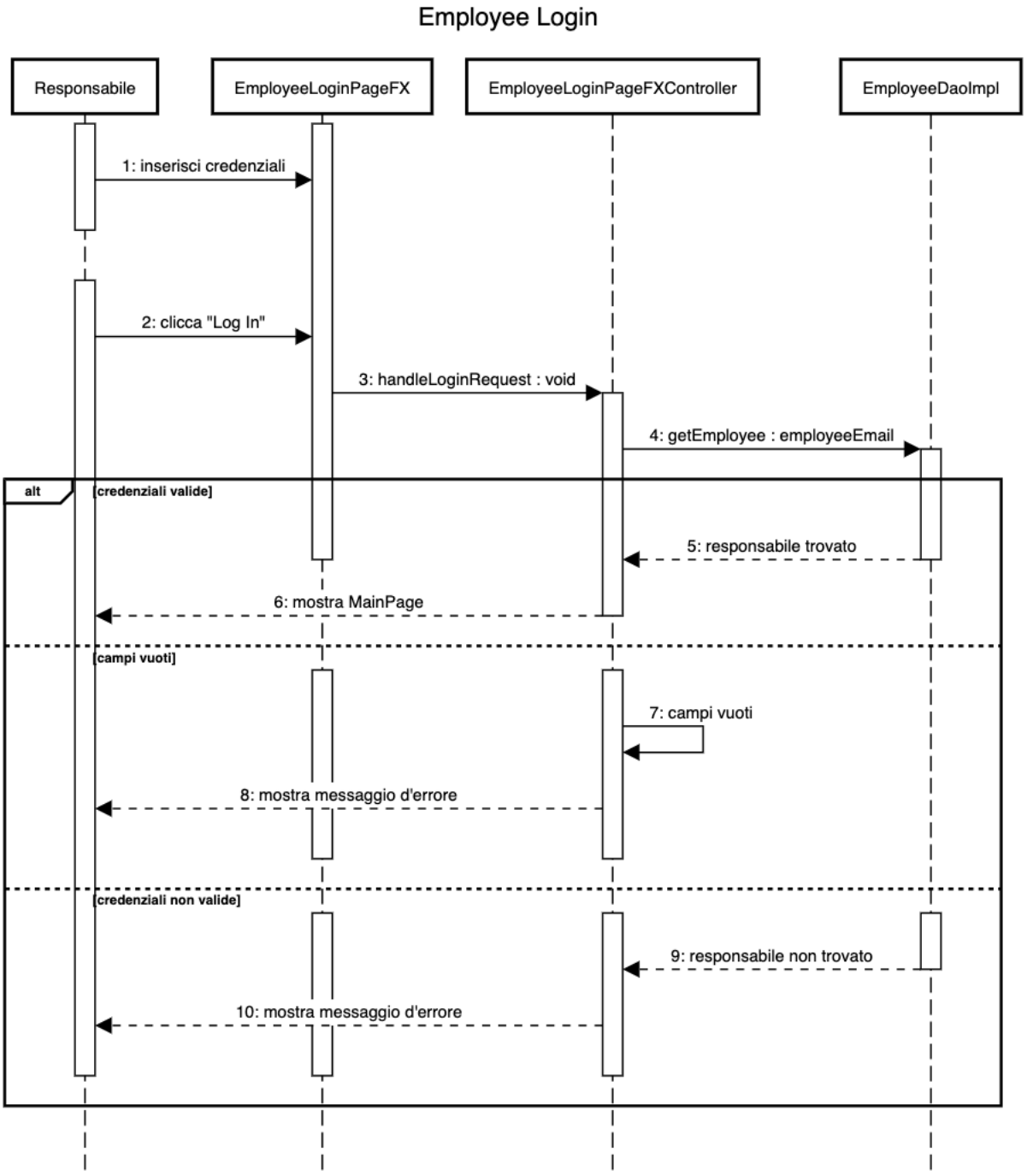


Figura 16: Sequence diagram per il **log in del responsabile**

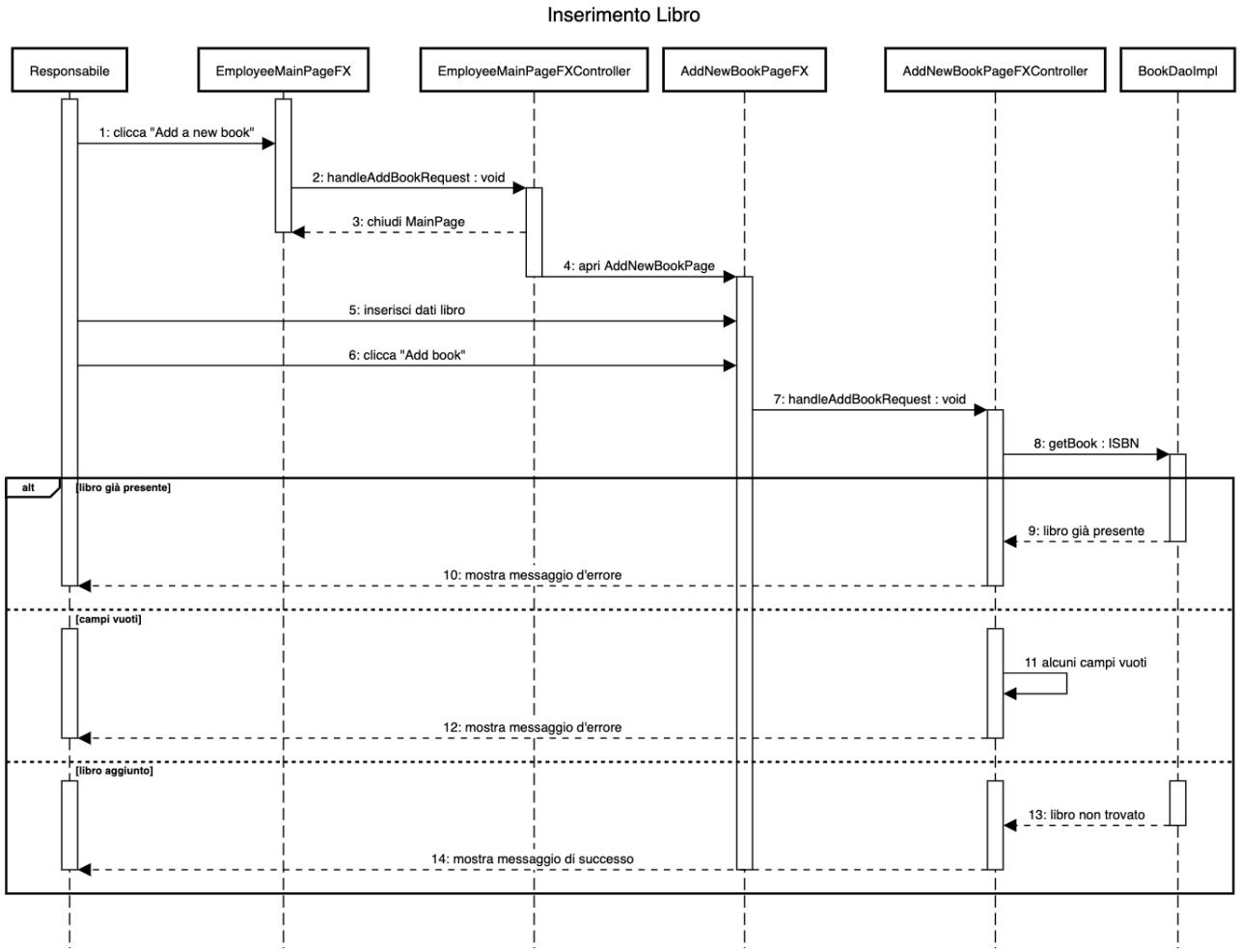


Figura 17: Sequence diagram per l'**inserimento di un nuovo libro**

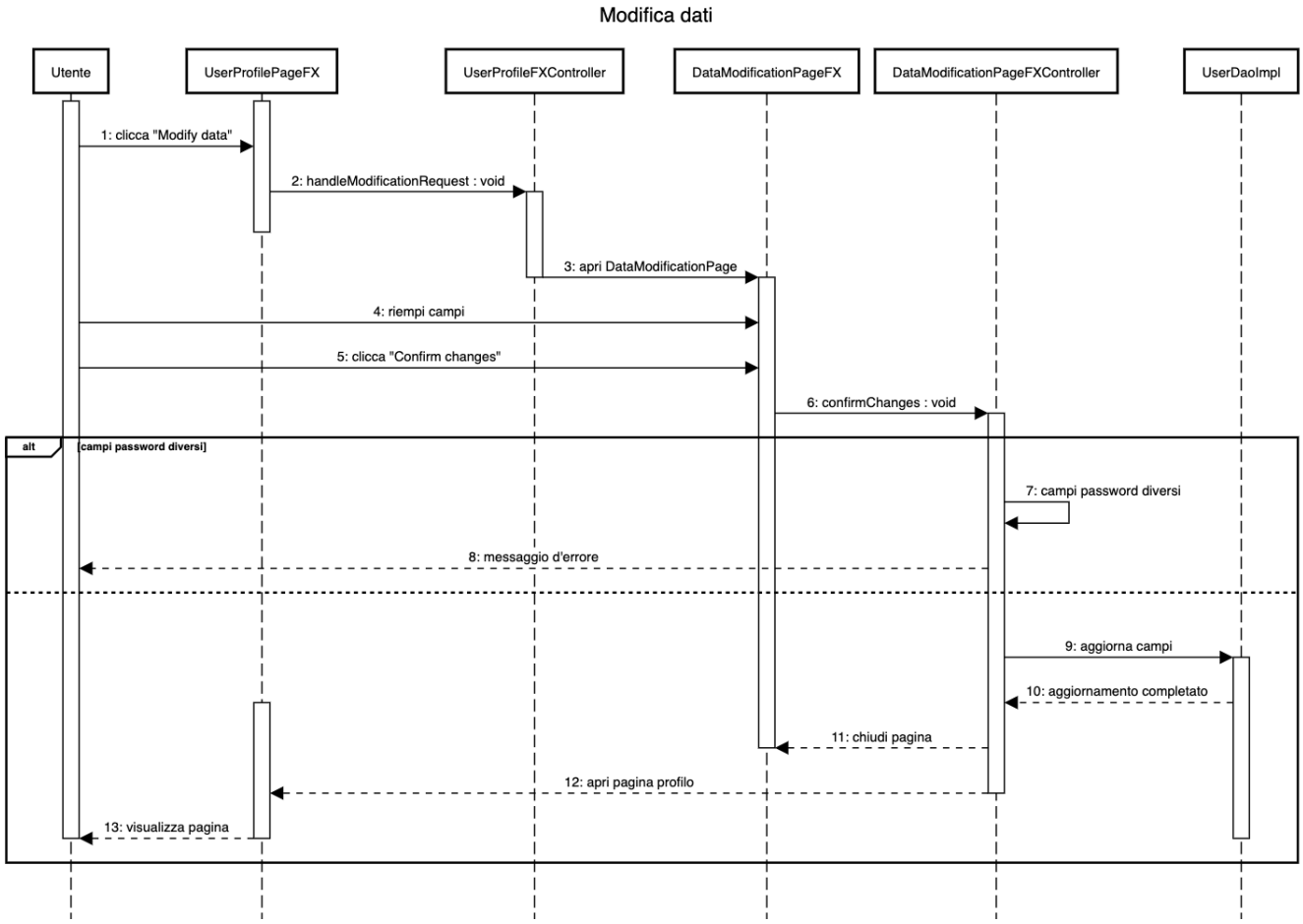


Figura 18: Sequence diagram per la **modifica dei dati** da parte di un utente

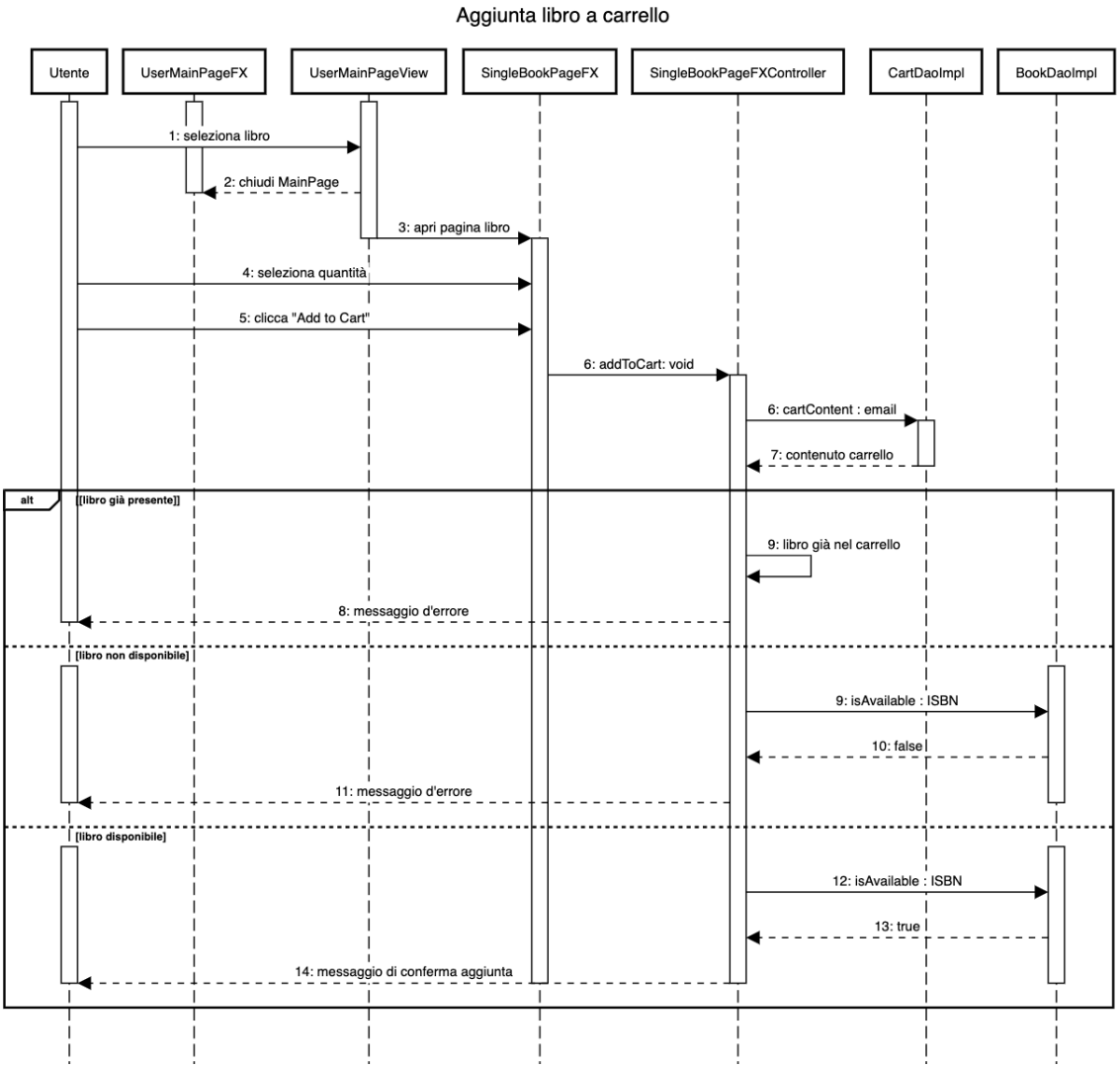


Figura 19: Sequence diagram per la **aggiunta di un libro** al carrello

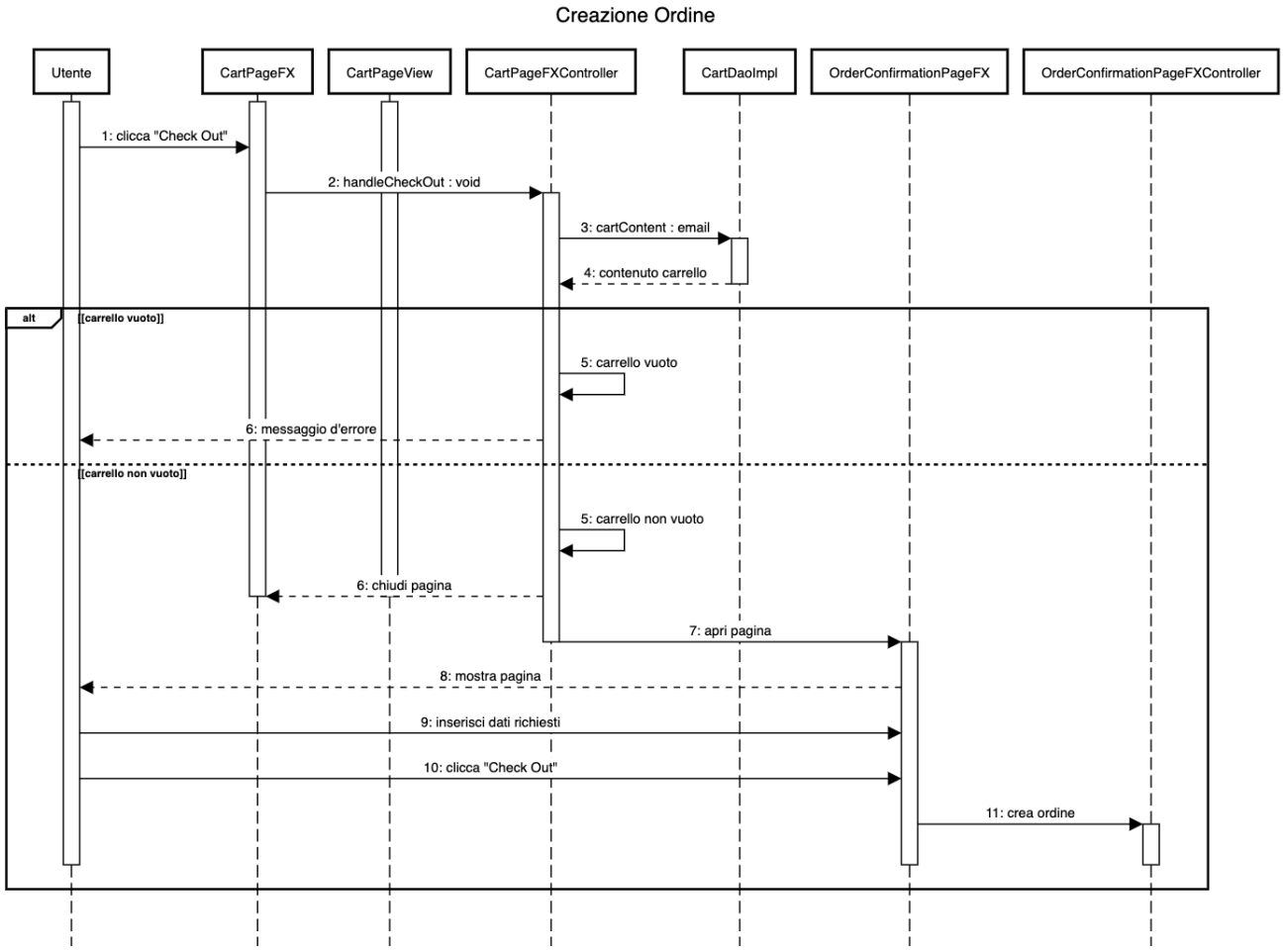


Figura 20: Sequence diagram per la **creazione di un ordine** da parte di un utente

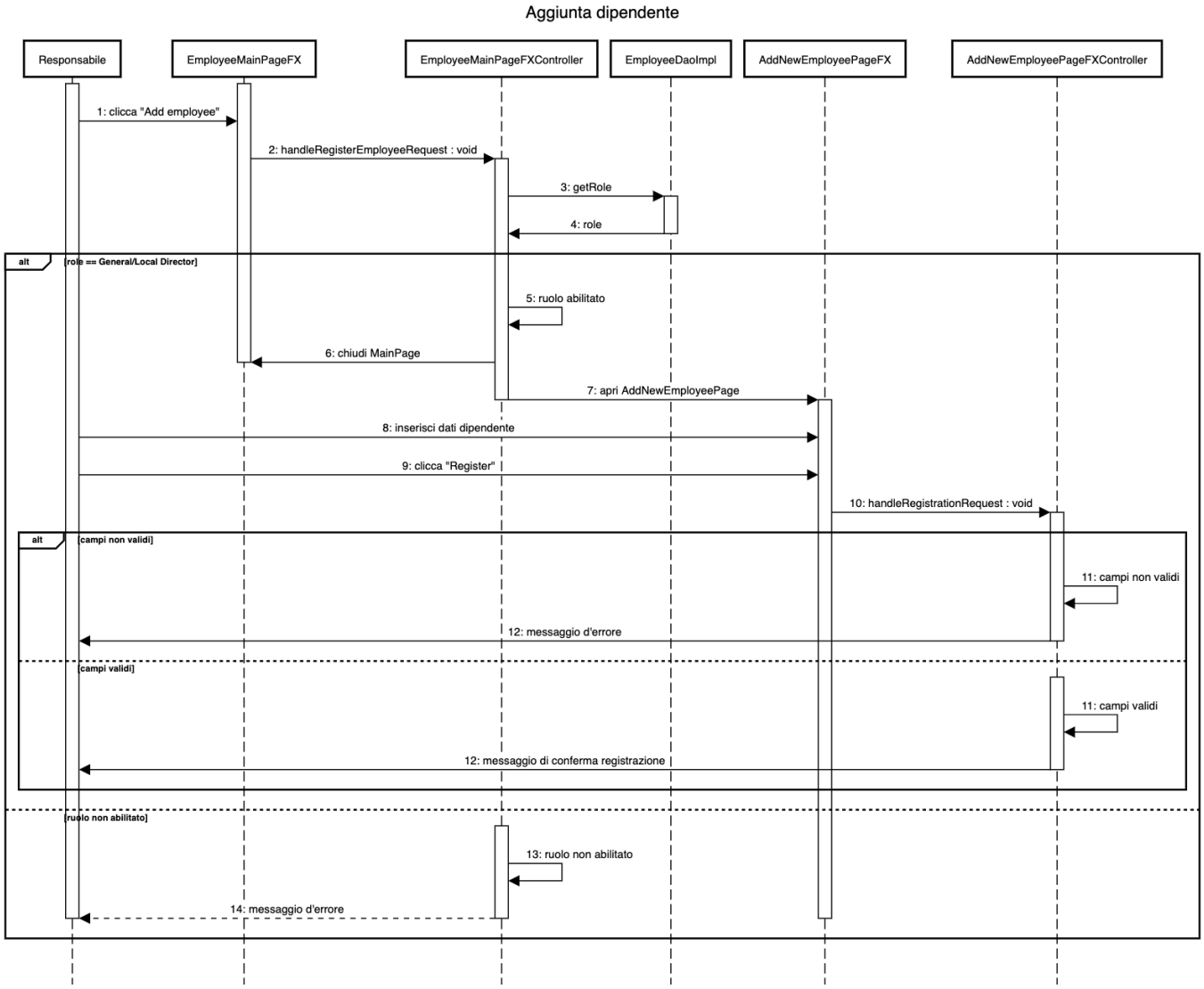


Figura 21: Sequence diagram per la **registrazione di un dipendente** da parte di un responsabile

4 Scelte progettuali

4.1 Sviluppo

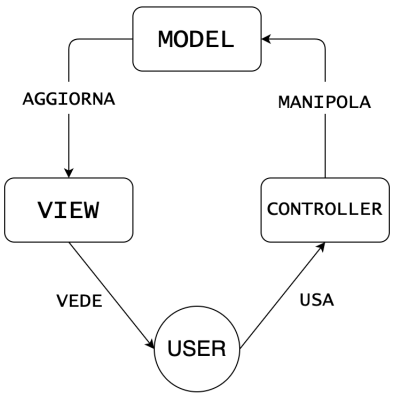
Questo progetto è stato sviluppato principalmente Java, che costituisce un linguaggio estremamente vario e flessibile, e ancora oggi risulta essere molto diffuso nel mercato.
Per l'interfaccia grafica ho deciso di utilizzare due strumenti: JavaFX, una libreria nativa di Java, e FXML, un linguaggio di mark-up basato su XML.
JavaFX rappresenta un'alternativa sicuramente più ricca e versatile rispetto a Swing, che è stato invece il tool proposto a lezione: essa permette infatti di realizzare GUI molto più elaborate, grazie alla presenza di animazioni, oggetti geometrici 2D e 3D, grafici, contenuti multimediali, e altro.
FXML, invece, è stato scelto perché permette di utilizzare un designer di interfacce grafice, Scene Builder, che rende la progettazione sicuramente più semplice e veloce. Inoltre, FXML è particolarmente adatto a interagire con JavaFX, visto che la struttura gerarchica del file rispecchia esattamente la struttura del *scene graph* di JavaFX.
Per stilizzare ulteriormente gli elementi dell'interfaccia grafica si è utilizzato anche CSS, che è completamente supportato sia da JavaFX, sia da FXML.

4.2 Database

Fin dall'inizio, risultava evidente la necessità di mantenere i dati nel tempo: per questo motivo ho deciso di far uso di un database, sfruttando così anche le conoscenze apprese durante il corso di Database frequentato durante lo sviluppo del progetto.
Dato che la gestione e l'utilizzo del database non rientravano negli obiettivi del progetto, ho scelto di utilizzare SQLite, che costituisce un RDBMS semplice, leggero e open-source.
SQLite è un database engine *"serverless"*: questo significa che i dati vengono memorizzati interamente in locale e non c'è bisogno di stabilire una connessione con un server esterno per lavorare con i dati. In un caso reale, un'applicazione di e-commerce farà chiaramente uso di un'architettura client-server per memorizzare il suo database, ma ricordiamo che il progetto sviluppato rappresenta solo una simulazione di un'applicazione web.
Un altro vantaggio di SQLite è il fatto di essere scritto interamente in C, il che lo rende particolarmente veloce.

4.3 MVC Pattern

Il funzionamento del progetto è basato sul MVC pattern, che prevede la suddivisione logica delle classi in tre componenti:
Il **Model** contiene una classe per ogni entità presente nel programma e fornisce i metodi per interagire con esse. Ogni entità contiene dati da memorizzare, infatti all'interno di questo componente troviamo anche le classi che forniscono i metodi per interagire con il database.



Il **Controller** risponde alle interazioni dell'utente con l'interfaccia grafica attivando determinate procedure ed è il componente che agisce sulle entità del Model.
Il **View** è la parte di programma che modella l'interfaccia grafica, quindi gestisce il modo in cui i dati vengono presentati all'utente.
I tre componenti del pattern MVC corrispondono concretamente ai tre package principali del programma.

4.4 Singleton Pattern

Il *Singleton Pattern* è stato utilizzato per limitare l'instanziazione di un oggetto a uno solo. In termini pratici, l'applicazione può essere usata da un solo utente alla volta, e quindi non è possibile essere loggati durante la stessa sessione come due utenti diversi. Nonostante l'applicazione simuli idealmente un sito di e-commerce, quindi un'applicazione web a cui possono accedere più utenti contemporaneamente, nella pratica è stata sviluppata come un'applicazione desktop: di conseguenza il software non è utilizzabile da più persone contemporaneamente, e questo giustifica l'utilizzo di questo design pattern. L'implementazione di questo pattern è stata sviluppata nella classe `GeneralLoginController`: la classe contiene l'attributo `loginInstance`, inizialmente settato a `null`, la cui funzione è memorizzare una stringa che identifica univocamente l'utente che ha eseguito il login.
Quando l'utente esegue il logout, il valore della stringa torna ad essere `null`.

4.5 Observer Pattern

L'*Observer Patter* prevede che un oggetto possa impostare una lista di "ascoltatori" (tecnicamente chiamati *Observers* o *Listeners*) che vengono notificati automaticamente ogni qualvolta lo stato dell'oggetto cambia. Questo pattern è stato utilizzato nell'interfaccia grafica, per ottenere dei cambiamenti dinamici in risposta alle interazioni dell'utente con determinati elementi. Più precisamente, è stato impiegato nei seguenti contesti:

- nella `MainPage` dei clienti (registrati e non), per visualizzare i libri appartenenti a un determinato genere;
- nelle `RankingsPage` dei clienti e dei responsabili, per visualizzare le classifiche relative a un determinato genere;
- in `AllLibroCardsPage`, ovvero la pagina che consente ai responsabili della libreria di controllare le LibroCard degli utenti. In questo caso, l'interfaccia cambiava a seconda che si volesse effettaure la ricerca della LibroCard con l'e-mail dell'utente o con la cardID;
- in `AllOrdersPage`, ovvero la pagina che permette ai responsabili di controllare lo stato degli ordini. In questo caso, l'insieme degli ordini visualizzato cambiava in base allo stato che veniva selezionato.

4.6 Data Access Object Pattern

Per la gestione del database ho deciso di adottare il *Data Access Object Pattern*, che permette di separare l'insieme delle azioni che l'applicazione ha bisogno di eseguire sul database dal modo in cui esse sono concretamente implementate.
Per ogni classe del `Model` è stata quindi definita un'interfaccia `DAO`, contenente la dichiarazione dei metodi tramite cui l'oggetto poteva interagire con il database. Ad ogni interfaccia corrisponde una classe `DaoImpl` che la implementa, sfruttando l'API di Java per l'interazione con i DBMS, JDBC, e semplici query SQL.

5 Validazione

Per la validazione dei dati inseriti ho utilizzato ampiamente di oggetti di tipo `Alert`, messi a disposizione da JavaFX. In questo modo, ho potuto programmare la comparsa di vari tipi di alert a seconda che le condizioni desiderate si verificassero o meno.
Per garantire una corretta esecuzione del programma anche in presenza di errori, ho utilizzato i blocchi `try-catch`: qualora si verificassero eventi indesiderati che avessero come conseguenza il sollevamento di eccezioni, l'esecuzione dei blocchi `catch` permettono al programma di segnalare la situazione d'errore senza crashare. Per quanto riguarda la GUI non mi è stato possibile testarla automaticamente, ho quindi provveduto ad eseguire un test manuale delle funzionalità. Mi sono inoltre appoggiata a colleghi ed amici al fine di raccogliere feedback sulle prestazioni del prototipo.

6 Conclusioni

Obiettivo del progetto non era quello di sviluppare un prodotto finito e completo ma di realizzare un prototipo rispettando le linee guida. Di conseguenza, alcune funzioni sono state realizzate

in maniera più semplice rispetto al modo in cui andrebbero implementate nella realtà, e alcuni controlli sono stati resi più "tolleranti".
Nel complesso, il software sviluppato riesce ad eseguire correttamente tutte le funzioni richieste.