Librarium

Documentazione

Indice generale

1. Organizzazione Del Personale

La struttura e stesura del progetto sarà gestita dal team in maniera coesa, tutti i membri del gruppo lavoreranno a stretto contatto per garantire una cooperazione continua e rimanere sempre aggiornati sullo sviluppo del progetto e sulle future modifiche.
Assumeremo l'aspetto di un 'Team Scrum', dove la collocazione, i brevi canali di comunicazione e l'atteggiamento adottato dalla squadra saranno però orientati alle persone piuttosto che alla formalità.
Il team sarà composto da individui con medesime competenze, verrà meno la presenza di membri fortemente specializzati perciò sarà utilizzata una struttura di organizzazione del personale che si scosta da quella gerarchica.
I ruoli di Scrum Master, Product Owner e Development Team saranno ricoperti da tutti i membri del gruppo favorendo un'omogenea ripartizione dei compiti.
Gli aspetti fondamentali saranno gestiti dal team al completo per garantire un avanzamento coordinato del progetto evitando il verificarsi di incongruenze interne, ammorbidendo l'impatto generato da eventuali modifiche.
Mediamente gli sprint saranno fissati tra le 2 e le 4 settimane.
Saranno inoltre pianificati Daily Scrum (di 15-20 minuti) e 1 incontro settimanale di breve durata dove si richiede la presenza di tutti i membri per esprimersi riguardo alle criticità osservate e raccogliere idee.

2. Specifica dei requisiti (Standard IEEE 830)

2.1. Introduzione

2.1.1. Scopo

L'applicazione **Librarium** nasce con lo scopo di incentivare le persone alla lettura, proponendo un catalogo suddiviso per generi attraverso il quale è possibile usufruire dei servizi offerti dalla biblioteca.
Inoltre Librarium garantisce un benefit anche per il bibliotecario aderente, offrendo un pannello di controllo che semplifica la gestione dei libri e dei prestiti nella relativa biblioteca.

2.1.2. Definizioni

- *Account*: si distingue in
  - Account Utente: è l'account usato dai lettori
  - Account Bibliotecario: è l'account usato dal bibliotecario per usufruire dei servizi del pannello di controllo

- *Prenotazione*: è l'operazione che consente al lettore di prenotare un libro presente nel catalogo della biblioteca; la prenotazione è consentita all'utente solo se il suo account è abilitato e il libro è disponibile in biblioteca.
  Passati cinque giorni dalla prenotazione, se l'utente non ha ancora ritirato il libro, esso torna disponibile e la prenotazione viene annullata.

- *Prestito*: è l'operazione che consente al lettore di ottenere un libro in prestito dalla biblioteca dopo aver effettuato una prenotazione.
  Il prestito ha durata mensile.

  - *Sollecito*: alla scadenza del prestito viene inoltrato un messaggio di sollecito destinato alla mail dell'account utente.

Passati cinque giorni dal primo sollecito senza riscontri da parte dell'utente, ogni cinque giorni verrà inoltrato un ulteriore sollecito fino a un totale di tre, oltre al quale si provvederà alla sospensione dell'account utente.

- Riabilitazione:
  A seguito della sospensione di un account utente: se l'utente restituisce tutti i libri presi in prestito, il suo account verrà riabilitato, altrimenti rimarrà sospeso.

2.2. Descrizione Generale

2.2.1. Prospettiva del Prodotto

La prospettiva dell'applicazione è quella di aiutare il bibliotecario a rendere più semplice ed efficace la gestione dei libri e dei prestiti della propria biblioteca attraverso una web app con un'interfaccia grafica intuitiva e ricca di funzionalità.

2.2.2. Funzioni del Prodotto

Funzioni per l'utente:

- Login / Registrazione
- Visione del catalogo
- Visione prenotazioni
- Prenotazione libro
- Modifica dati profilo

Funzioni per il bibliotecario:

- Login
- Gestione del catalogo (aggiunta, rimozione e modifiche da apportare nei libri)
- Gestione dei prestiti/ritiri
- Gestione account utenti

2.2.3. Caratteristiche degli utenti

L'applicazione è facile e intuitiva, perciò non necessita di formazione o alcun tipo di preparazione. Può essere utilizzata da qualsiasi tipo di utente, anche il più inesperto.

2.2.4. Vincoli

Ci sono dei vincoli legati all'uso delle funzionalità:

- L'utente senza aver effettuato la registrazione non può effettuare il login.
- Una mail può corrispondere ad un solo account registrato.
- L'utente senza account non può accedere al servizio di prenotazione ma soltanto sfogliare il catalogo e leggere una breve descrizione sui libri mostrati.
- Concordata la prenotazione ed avvenuto il ritiro presso la biblioteca, l'utente ha un limite di tempo entro il quale deve riportare il libro (la durata è di un mese).
- Se il libro non è stato restituito entro la data stabilita, vengono inoltrati fino a 3 solleciti (uno ogni 5 giorni) nelle settimane consecutive, e se ancora non si ha riscontro dall'utente, il suo account viene sospeso.

2.3. Requisiti Specifici

2.3.1. Interfaccia Esterna

L'interfaccia utente permette di accedere direttamente al catalogo, fornendo il servizio di ricerca per libro, autore, casa editrice e genere. È indispensabile una connessione internet indipendentemente dal dispositivo usato.
Librarium nasce come Web App, l'interfaccia è perciò funzionale sia su PC che su altri dispositivi mobili.

2.3.2. Requisiti Funzionali

2.3.2.1. Autenticazione

- Registrazione: l'utente per effettuare la registrazione deve inserire le credenziali da attribuire all'account (nome, cognome, email, password e password di conferma).
  Può esistere un solo account associato a un dato indirizzo email.
- Login: a seguito di una registrazione effettuata con successo, l'utente può accedere al proprio account utilizzando le credenziali inserite nella Registrazione (email, password).

2.3.2.2. Area Personale

- Catalogo: mostra l'insieme dei libri della biblioteca distinti per genere.
- Prenotazione: operazione che consente all'utente di prenotare un libro presente nel catalogo (se disponibile).
- Prestito: operazione che si genera a seguito della prenotazione di un libro e del relativo ritiro. Il prestito ha una scadenza mensile.
- Modifica dati utente: l'utente ha la possibilità di modificare i suoi dati profilo una volta inseriti.

2.3.3. Requisiti sulle Performance

Non sono richieste risorse computazionali elevate per questo applicativo. È indispensabile una connessione internet indipendentemente dal dispositivo usato in quanto le operazioni vengono eseguite e registrate sui server e database.

3. Architettura del Software

3.1. Stile Architetturale

Con lo scopo di aumentare la qualità del software e rendere più comprensibile e pulito il codice, si è deciso di adottare uno stile architetturale di tipo **Model-View-Controller**'.

3.1.1. Problema

Il modello prevede la suddivisione del codice in tre parti distinte: l'UI (User Interface) che mostra i dati all'utente, il Database contenente l'insieme dei dati e il software che gestisce tutte le operazioni. Ci permetterà di non ripeterci e in secondo luogo, aiuta a creare una solida struttura per la nostra applicazione.

3.1.2. Soluzioni

Svilupperemo la nostra applicazione seguendo questi 3 componenti architetturali.

- Il **Model** gestisce direttamente i dati, comunica con il database ed è fondamentale per fare in modo di registrare tutte le modifiche e di prelevare i dati da rappresentare.
- Il **View** che gestisce l'interfaccia utente che verrà visualizzata dall'utente. Questo componente comunicherà con **controller** per aggiornare i dati da visualizzare.
- Il **Controller** che comprende tutte le classi 'manager' che gestiscono le regole dell'applicazione prendendo i dati dal **model** e spiega come rappresentarli al **view**, serve anche per tutte le operazioni di upgrade/refresh

3.2. Viste architetturali

L'architettura del sistema è stata descritta attraverso 5 diagrammi UML, che descrivono le viste architetturali dell'applicazione.

3.2.1. Class Diagram

Il class diagram contiene le classi implementate con relativi attributi e metodi. Abbiamo ritenuto più comprensibile l'utilizzo di enumerativi per assegnare valori agli stati delle classi.

3.2.2. State Machine Diagram

Descrive la sequenza di stati assunti dalla procedura di registrazione, sino alla scrittura dei dati inseriti nel database. Comprende la verifica di validità degli stessi e l'interruzione in caso di errori seguita dall'invio di un segnale d'errore.

3.2.3. Sequence Diagram

Illustra il processo di prenotazione di un libro da parte di un utente registrato. L'utente invia una richiesta di prenotazione, il sistema verifica la disponibilità del libro e in caso di esito positivo, verifica che l'account non sia sospeso.

3.2.4. Activity diagram

L'Activity Diagram rappresenta il processo di registrazione e quello di accesso, partizionati nei tre interpreti coinvolti: l'utente, il sistema ed il database manager.

3.2.5. Use Case Diagram

Abbiamo utilizzato lo Use Case Diagram per mostrare una visione d'insieme del software, rappresentando le funzionalità che esso fornisce ai vari attori interpretati nel nostro caso da account (che generalizza account utente e bibliotecario) e dal database.

4. Design Pattern

Per come è stata progettata l'architettura del software, abbiamo deciso di adottare principalmente l'utilizzo del design pattern Singleton per la creazione delle classi riguardanti la gestione del database (CatalogManager, AuthenticationManager, PrestitiManager, UsersManager).

Questa scelta di design comporta quindi che per tali classi venga prevista la creazione di un'unica istanza alla quale si può accedere attraverso il metodo pubblico GetInstance(). In questo modo le altre classi possono usufruire dei metodi messi a disposizione da queste, nello specifico:

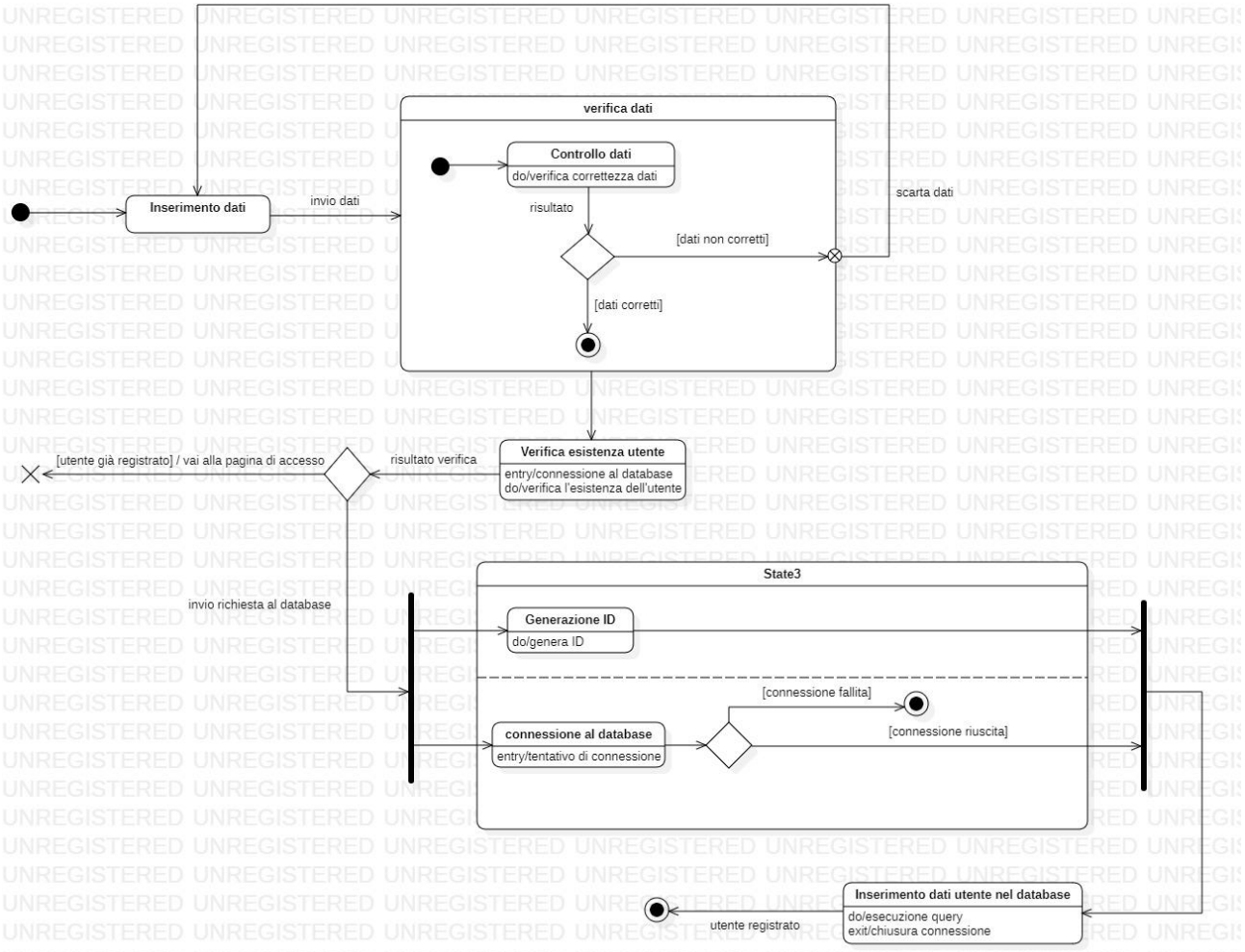
- **CatalogManager**:
  Si occupa dell'accesso al database (attraverso alcune query) per gestire tali operazioni. Fornisce dei metodi per gestire il catalogo e più nello specifico lettura, aggiunta, rimozione e modifica di libri e generi.
- **AuthenticationManager**:
  Si occupa dell'accesso al database (attraverso alcune query) per gestire operazioni relative all'autenticazione. Fornisce metodi per gestire l'accesso e la registrazione di un utente.
- **PrestitiManager**:
  Si occupa dell'accesso al database (attraverso alcune query) per gestire operazioni relative ai prestiti. Fornisce metodi per leggere, creare, aggiornare ed eliminare i prestiti.
- **UsersManager**:
  Si occupa dell'accesso al database (attraverso alcune query) per gestire operazioni relative agli utenti e ai solleciti. Fornisce metodi per leggere, aggiungere e aggiornare i dati relativi agli utenti e gestire i solleciti.

4.1. Diagrammi UML

Le caratteristiche del design dell'applicazione sono inoltre rappresentate da dei diagrammi UML.

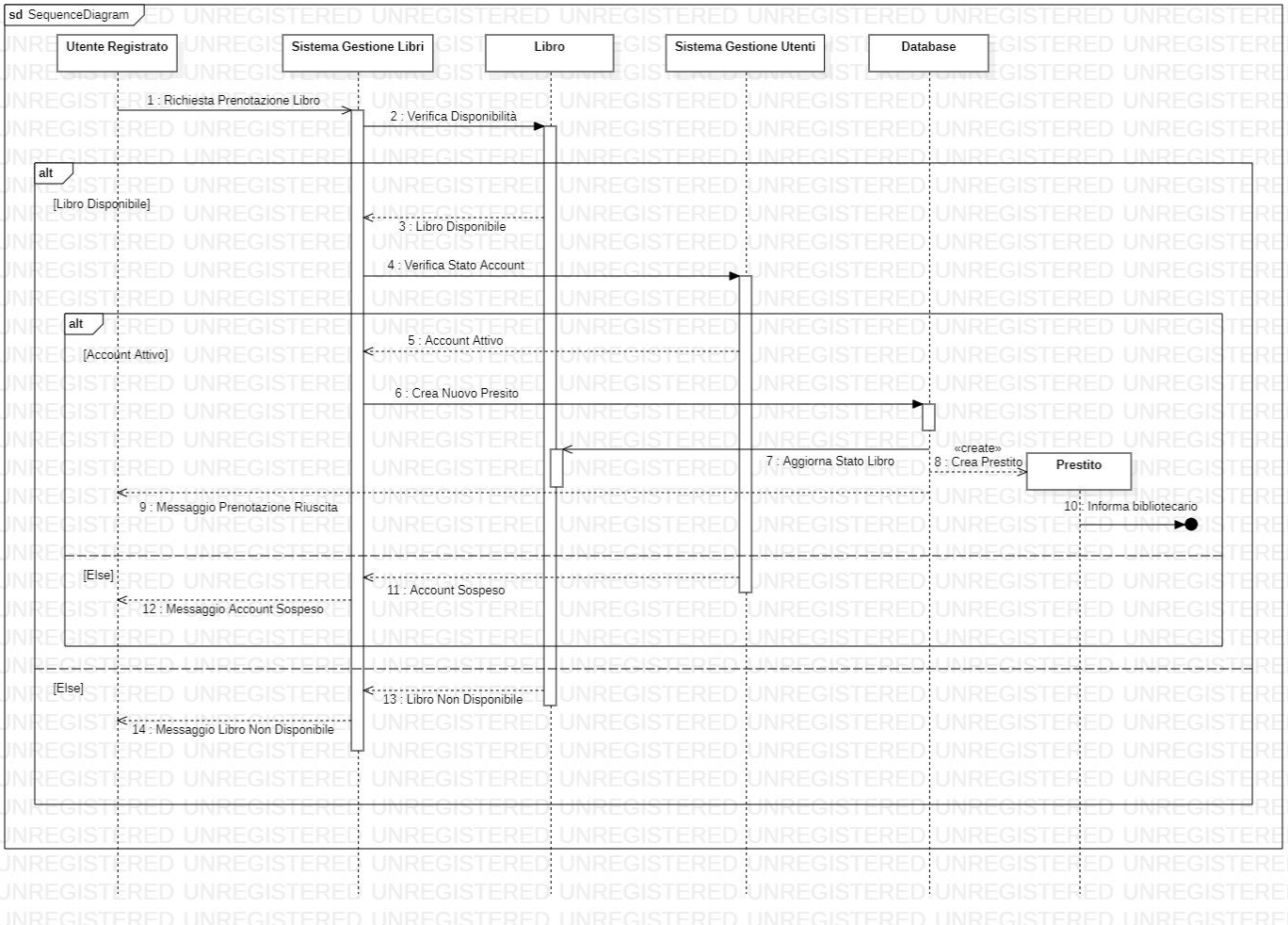
4.1.1. State Machine diagram

Con questo diagramma si rappresenta il processo di sign-in, dalla richiesta di registrazione dell'utente, fino alla creazione della sua identità nel database. Un aspetto importante è che al fine di avere un esito positivo dal processo di registrazione di un nuovo utente è necessario che la connessione con il database avvenga con successo. In caso contrario l'intero procedimento viene interrotto.



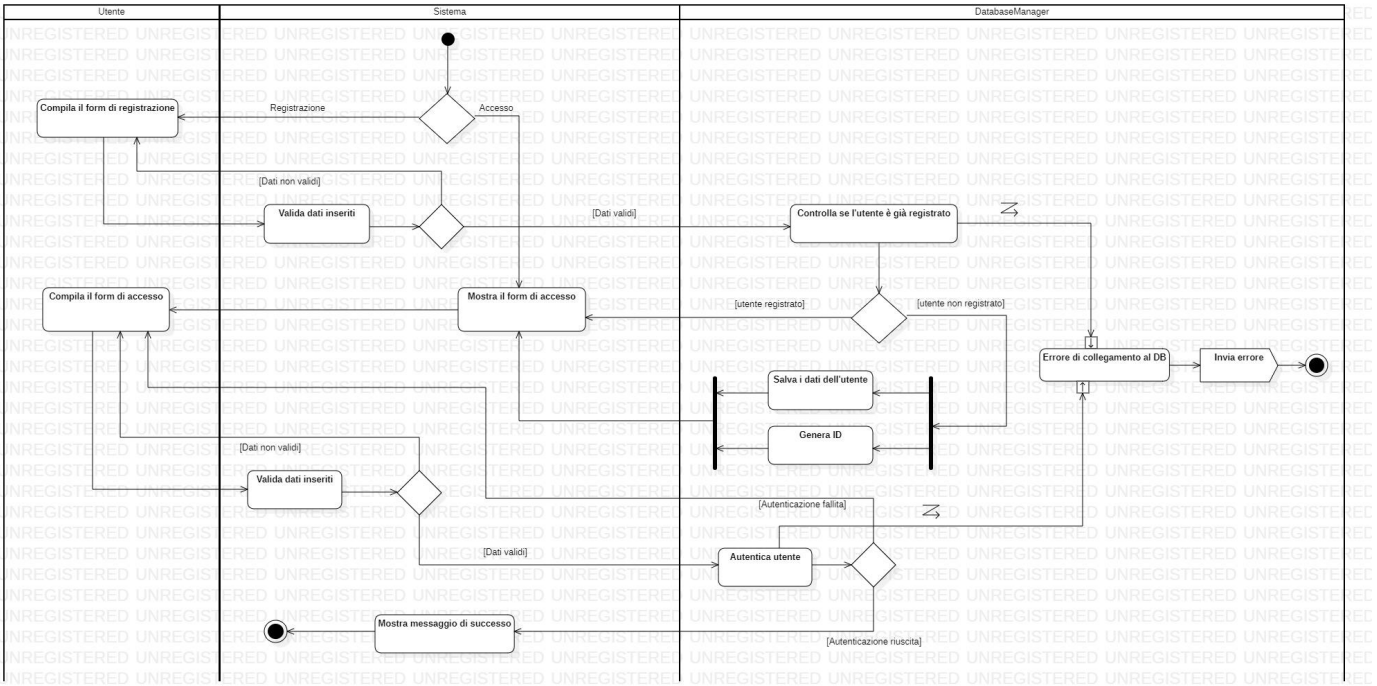
4.1.2. Sequence diagram

Con questo diagramma si sono volute modellare le due attività principali gestite dall'applicazione. In particolare, sono descritti gli scambi di messaggi tra i vari componenti inerentemente ai processi di prenotazione di un libro e di verifica delle disponibilità dell'account. Mostra dettagliatamente come interagiscono i vari componenti e come sono gestiti i processi.



4.1.3. Activity diagram

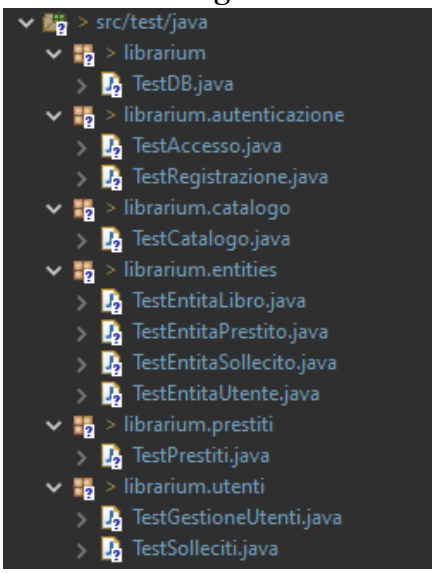
Con questo diagramma si modella il processo di registrazione dell'utente. Con questo diagramma si vuole introdurre un maggior livello di dettaglio per quanto riguarda il flusso delle attività tra i componenti. Infatti, sono messi in evidenza i vari componenti e le rispettive operazioni/attività di loro competenza.



5. Testing

5.1. Organizzazione Test

Abbiamo organizzato e suddiviso i test in classi all'interno dei package come illustrati di seguito:

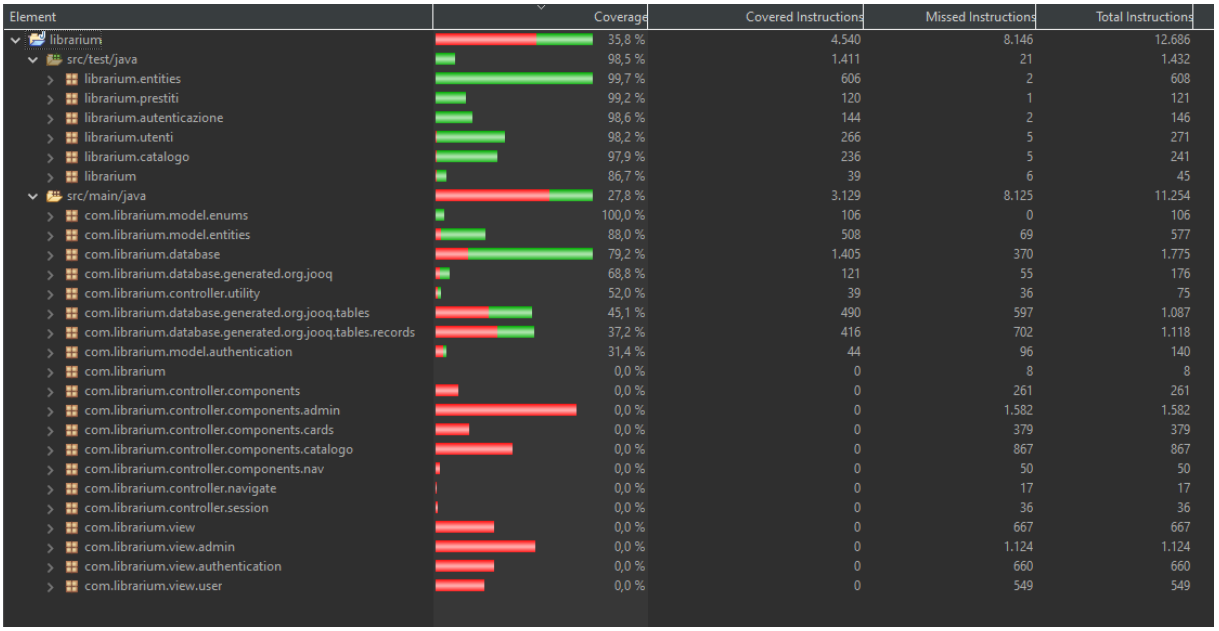


I test effettuati sulla nostra applicazione hanno ricoperto il 35.8% del codice.

La classe "TestDB" è estesa dalle altre classi di test e implementa dei metodi che permettono di eseguire delle operazioni sul Database mantenendone invariato lo stato iniziale, attraverso l'esecuzione di un rollback dei dati alla fine di ogni test.

Per quanto riguarda le funzionalità della Web App abbiamo testato principalmente i metodi delle classi inerenti al database e alle sue entità, dato che costituiscono una sezione fondamentale del codice.

Per quanto riguarda il funzionamento dell'interfaccia e dei suoi componenti abbiamo eseguito dei test direttamente sull'applicazione per verificarne il corretto comportamento.



Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
librarium	35,8 %	4.540	8.146	12.686
src/test/java	98,5 %	1.411	21	1.432
librarium.entities	99,7 %	606	2	608
librarium.prestiti	99,2 %	120	1	121
librarium.autenticazione	98,6 %	144	2	146
librarium.utenti	98,2 %	266	5	271
librarium.catalogo	97,9 %	236	5	241
librarium	86,7 %	39	6	45
src/main/java	27,8 %	3.129	8.125	11.254
com.librarium.model.enums	100,0 %	106	0	106
com.librarium.model.entities	88,0 %	508	69	577
com.librarium.database	79,2 %	1.405	370	1.775
com.librarium.database.generated.org.jooq	68,8 %	121	55	176
com.librarium.controller.utility	52,0 %	39	36	75
com.librarium.database.generated.org.jooq.tables	45,1 %	490	597	1.087
com.librarium.database.generated.org.jooq.tables.records	37,2 %	416	702	1.118
com.librarium.model.authentication	31,4 %	44	96	140
com.librarium	0,0 %	0	8	8
com.librarium.controller.components	0,0 %	0	261	261
com.librarium.controller.components.admin	0,0 %	0	1.582	1.582
com.librarium.controller.components.cards	0,0 %	0	379	379
com.librarium.controller.components.catalogo	0,0 %	0	867	867
com.librarium.controller.components.nav	0,0 %	0	50	50
com.librarium.controller.navigate	0,0 %	0	17	17
com.librarium.controller.session	0,0 %	0	36	36
com.librarium.view	0,0 %	0	667	667
com.librarium.view.admin	0,0 %	0	1.124	1.124
com.librarium.view.authentication	0,0 %	0	660	660
com.librarium.view.user	0,0 %	0	549	549

5.2. Refactoring

Abbiamo apportato modifiche per organizzare package e i nomi delle classi. Abbiamo deciso di aggiungere classi ausiliarie per semplificare parti del codice, aumentandone l'astrazione.

6. Conclusioni

Il punto di forza è stato quello di procedere con la stesura della documentazione parallelamente alla scrittura del codice.

Inizialmente il codice non è stato suddiviso in modo ottimale all'interno dei package e quando ci siamo accorti di questo inconveniente, l'abbiamo riorganizzato ottenendo una struttura più chiara e facilmente comprensibile.

L'implementazione e la gestione del Database hanno rappresentato gran parte degli sforzi iniziali, richiedendo diversi giorni per la scelta della struttura e la definizione delle sue funzionalità.

Durante lo sviluppo del progetto abbiamo dedicato troppo tempo all'implementazione dell'interfaccia grafica, la quale ha richiesto più risorse di quante ne avessimo stimate.

Il tempo richiesto per i casi di test è stato stimato in maniera ottimale, occupando quella piccola parte delle risorse previste. Inoltre, non abbiamo incontrato particolari difficoltà durante la loro implementazione.

Infine, Github è stato fondamentale per la gestione del progetto permettendo al gruppo di lavorare in parallelo su diverse funzionalità.