Documento Progetto Ingegneria del Software

You

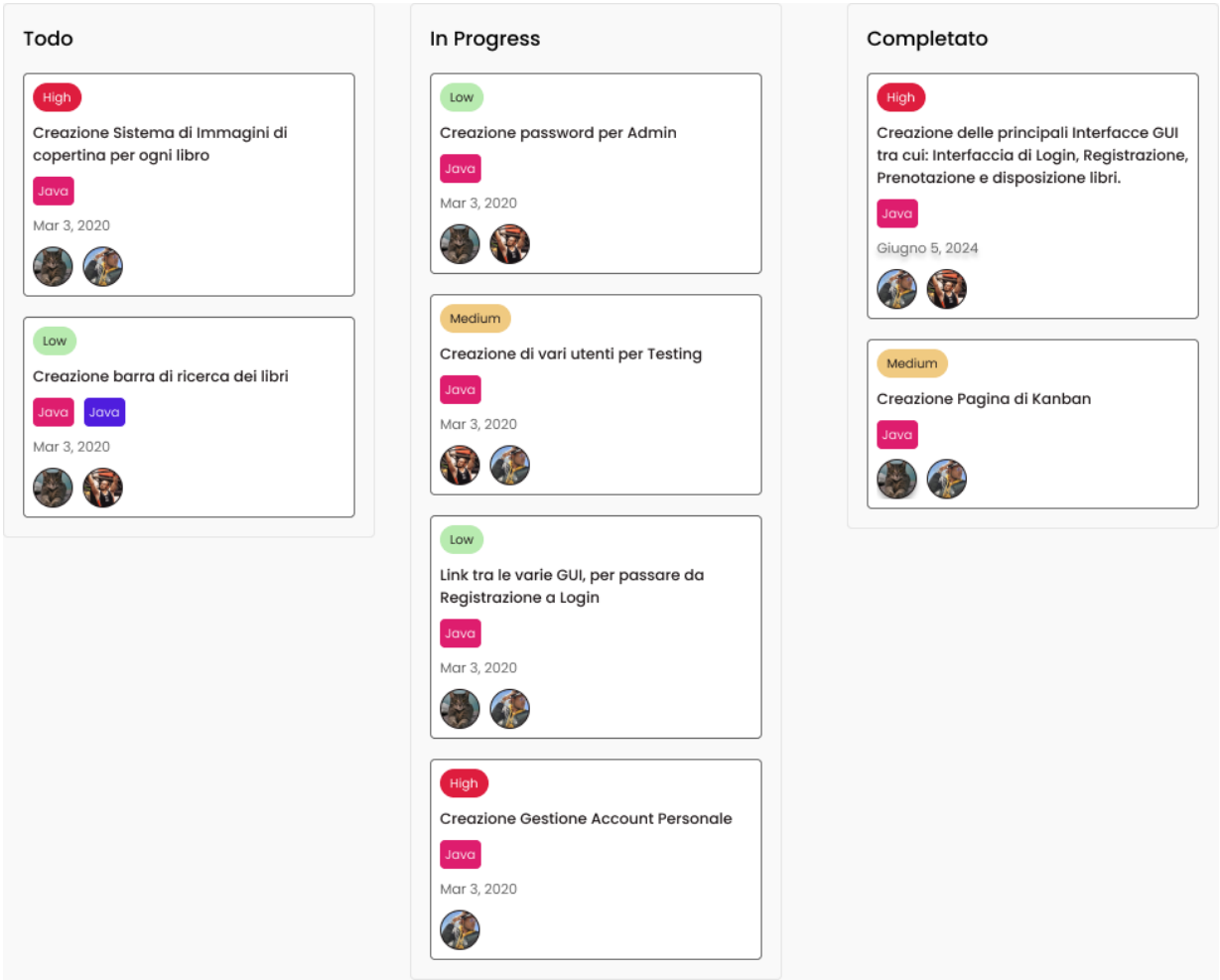
October 6, 2024

Contents

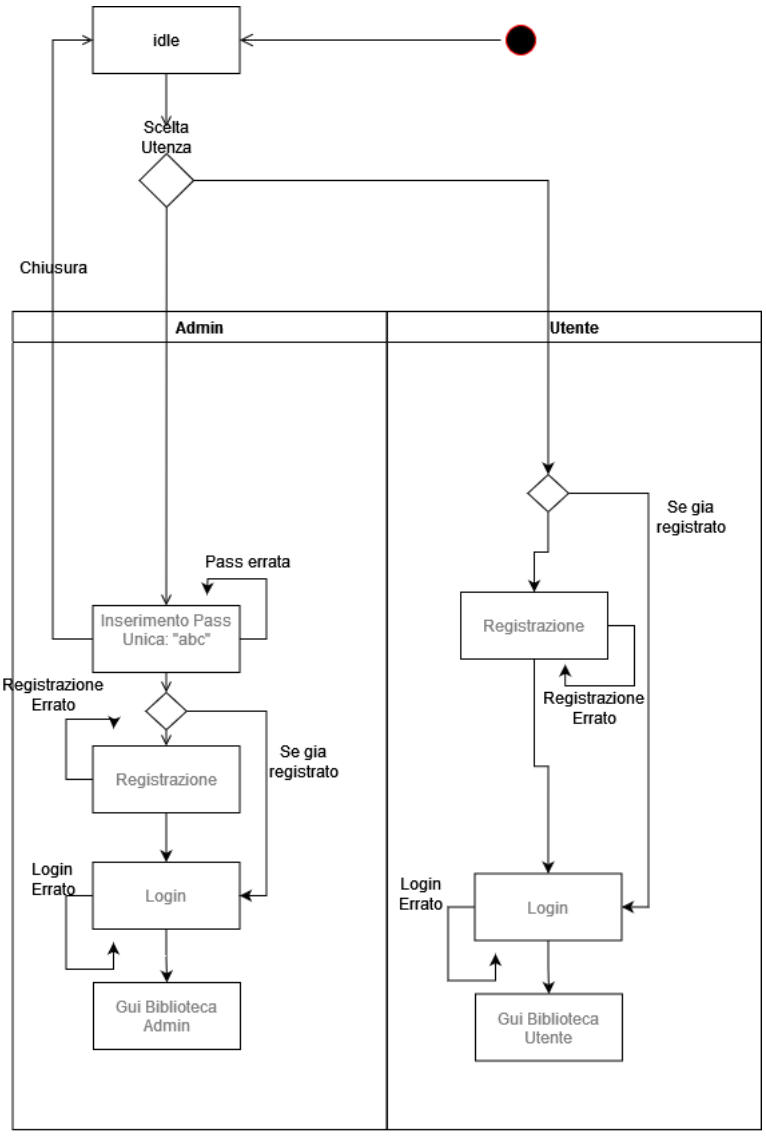
1 Introduzione

2 Software Life Cycle

Nel nostro progetto di biblioteca virtuale, abbiamo seguito il processo di sviluppo Agile con Kanban, tramite il software Figma.



Questa metodologia ci ha permesso di gestire in modo efficace lo sviluppo delle funzionalità divise tra utenti e amministratori. Ad esempio, le funzionalità di ricerca e prenotazione per gli utenti e quelle di gestione della bacheca per gli amministratori sono state sviluppate in fasi incrementali, consentendo un rapido feedback e adattamenti continui basati sulle esigenze del progetto. Abbiamo formalizzato una parte del ciclo di sviluppo, come la gestione delle interfacce utente. La rete di Petri è stata utilizzata per rappresentare le transizioni di stato quando un utente/amministratore deve fare il login.


idle
Scelta
Utenza
Chiusura
Admin
Utente
Pass errata
Inserimento Pass
Unica: "abc"
Registrazione
Errato
Registrazione
Se gia
registrato
Login
Errato
Login
Gui Biblioteca
Admin
Se gia
registrato
Registrazione
Registrazione
Errato
Login
Errato
Login
Gui Biblioteca
Utente

3 Configuration Management

Nel nostro progetto della biblioteca virtuale, abbiamo utilizzato GitHub come strumento di configuration management, per garantire una gestione efficiente del codice sorgente e della collaborazione tra i membri del team. Il link per accedere al nostro repository è il seguente: https://github.com/mathieuteti2002/Progetto_ING_SW/tree/main Durante il lavoro su GitHub, abbiamo utilizzato issues, branches, pull requests, e code reviews

4 People Management and Team Organization

Nel progetto della biblioteca virtuale, abbiamo adottato un approccio di gestione delle persone basato su un team di tipo **SWAT**. Questo tipo di team è caratterizzato da un gruppo ristretto di membri altamente specializzati, selezionati per le loro competenze specifiche e capacità di risolvere problemi complessi in tempi rapidi. Struttura del Team

Il nostro team SWAT era composto da:

- **Project Manager**: Responsabile del coordinamento generale, definizione delle priorità, e gestione delle comunicazioni tra i membri del team e con gli stakeholder. Si è occupato anche di monitorare l'avanzamento del progetto, garantendo il rispetto delle scadenze. Il seguente ruolo è stato svolto da Mathieu e Selmani.
- **Lead Developer**: Ha guidato lo sviluppo tecnico, prendendo decisioni architetturali e supervisionando l'implementazione delle funzionalità principali, come la gestione delle prenotazioni e l'integrazione del database SQLlite. Il seguente ruolo è stato svolto da Riccardo.
- **UI/UX Designer**: Ha curato l'aspetto grafico e l'esperienza utente dell'interfaccia della biblioteca, assicurandosi che la GUI fosse intuitiva sia per gli utenti che per gli admin. Il seguente ruolo è stato svolto da Mathieu.
- **Database Administrator**: Responsabile della progettazione e manutenzione del database, gestendo la struttura delle tabelle, le query e l'integrità dei dati. Il seguente ruolo è stato svolto da Riccardo
- **Quality Assurance (QA) Specialist**: Ha garantito che tutte le funzionalità fossero testate accuratamente, utilizzando test manuali e automatizzati per individuare e correggere eventuali bug prima del rilascio. Il seguente ruolo è stato svolto da Selmani.

4.1 Organizzazione del Lavoro

Seguendo l'approccio SWAT, il lavoro è stato suddiviso in missioni specifiche, ognuna affidata al membro del team con le competenze più adatte. Per esempio:

- La missione di sviluppo della funzione di ricerca dei libri è stata assegnata al Lead Developer, con il supporto del QA Specialist per i test.
- La missione di progettazione dell'interfaccia utente è stata affidata al UI/UX Designer, con feedback continuo dal Project Manager e dal Lead Developer per assicurare che la UI fosse coerente con i requisiti funzionali e tecnici.

Il team SWAT ha permesso una gestione efficiente del progetto, grazie alla chiara definizione dei ruoli e all'alta specializzazione dei membri. La collaborazione stretta e l'approccio orientato ai risultati hanno garantito il successo dello sviluppo della biblioteca virtuale, rispondendo in modo agile e rapido alle sfide incontrate durante il processo.

5 Software Quality

Nel progetto della biblioteca virtuale, abbiamo seguito i principi definiti dallo standard ISO/IEC 9126 per garantire un software di alta qualità. Questo standard identifica diverse caratteristiche di qualità che abbiamo considerato fondamentali durante lo sviluppo. Ecco una lista delle principali qualità, contestualizzate al nostro progetto:

- **Funzionalità**: La nostra priorità era garantire che tutte le funzionalità richieste fossero implementate correttamente. Abbiamo progettato il sistema in modo che gli utenti potessero cercare e prenotare libri facilmente, mentre gli admin potevano aggiungere e gestire le informazioni sui libri. La funzionalità è stata continuamente testata per assicurare che il sistema rispondesse esattamente ai requisiti definiti.

- **Affidabilità**: Abbiamo concentrato sforzi significativi per assicurare che il sistema fosse stabile e affidabile. Ad esempio, la gestione delle prenotazioni e l'accesso al database dovevano funzionare correttamente anche in caso di carico elevato. Per raggiungere questo obiettivo, sono stati implementati meccanismi di recupero e gestione degli errori.

- **Usabilità**: Poiché la biblioteca virtuale è destinata a un pubblico ampio, era essenziale che l'interfaccia utente fosse intuitiva e facile da usare. Il UI/UX Designer ha lavorato per creare un'esperienza utente chiara e accessibile, minimizzando la curva di apprendimento per i nuovi utenti e garantendo che le operazioni comuni come la ricerca e la prenotazione dei libri fossero semplici e rapide.

- **Efficienza**: L'efficienza del sistema è stata garantita attraverso un'ottimizzazione del codice e del database. Abbiamo implementato query SQL ottimizzate per ridurre i tempi di risposta durante le ricerche e gestito in modo efficace le risorse di sistema per evitare rallentamenti, specialmente durante l'accesso concorrente da parte di più utenti.

- **Manutenibilità**: La struttura del codice è stata progettata per essere facilmente comprensibile e modificabile. Abbiamo utilizzato pratiche di code review e documentazione per assicurare che il sistema potesse essere facilmente mantenuto e aggiornato in futuro. L'uso di un approccio modulare ha permesso di isolare le varie funzionalità, facilitando l'implementazione di nuove caratteristiche o la correzione di bug.

- **Portabilità**: Abbiamo scelto di sviluppare la biblioteca virtuale con tecnologie che permettono una facile migrazione su diverse piattaforme, come il database MySQL e il linguaggio Java in Eclipse. Questo garantisce che il sistema possa essere distribuito su diversi ambienti senza richiedere modifiche significative al codice.

Queste qualità hanno guidato ogni fase dello sviluppo del progetto, assicurando che il prodotto finale fosse non solo funzionale, ma anche robusto, facile da usare e da mantenere, rispondendo alle esigenze di tutti i tipi di utenti.

6 Requirement Engineering

Nel contesto del progetto della **biblioteca virtuale**, la gestione dei requisiti è stata fondamentale per garantire che il sistema sviluppato rispondesse pienamente alle esigenze degli utenti e degli amministratori. Per specificare i requisiti, abbiamo seguito le linee guida dello standard **IEEE 830**, che definisce un modello strutturato per la redazione della Specifica dei Requisiti del Software (SRS). Di seguito presentiamo la specifica dei requisiti per il nostro progetto, suddivisa nelle principali sezioni:

6.1 Introduzione

- **Scopo**: Lo scopo del sistema della biblioteca virtuale è fornire una piattaforma digitale per la gestione e la prenotazione di libri. Gli utenti possono cercare, prenotare e visualizzare informazioni sui libri disponibili, mentre gli admin possono aggiungere, aggiornare e rimuovere i libri dalla bacheca.
- **Ambito**: Il sistema sarà accessibile via web attraverso una GUI dedicata e dovrà supportare la gestione simultanea di utenti e admin, con funzionalità distinte per ciascuna categoria.
- **Definizioni, acronimi e abbreviazioni**: Ad esempio, "GUI" si riferisce all'interfaccia utente grafica, "DBMS" indica il sistema di gestione del database (in questo caso, SQLlite).

6.2 Descrizione Generale

- **Prodotto**: La biblioteca virtuale è un'applicazione web-based che permette agli utenti di gestire la loro interazione con i libri disponibili attraverso un'interfaccia intuitiva.
- **Utenti del sistema**: Gli utenti principali sono:
  - **Utenti finali**: Persone che accedono al sistema per cercare e prenotare libri.
  - **Admin**: Personale autorizzato che gestisce l'inventario di libri e altre operazioni amministrative.
- **Vincoli**: Il sistema deve essere compatibile con i browser web più comuni e deve utilizzare SQLlite per la gestione dei dati.

6.3 Requisiti Funzionali

- **RF1: Autenticazione degli utenti**
  - **Descrizione**: Gli utenti e gli admin devono potersi autenticare nel sistema utilizzando un nome utente e una password.
  - **Priorità**: Alta
  - **Input**: Credenziali di accesso (username e password)
  - **Output**: Accesso al sistema con autorizzazioni appropriate.
- **RF2: Ricerca dei libri**
  - **Descrizione**: Gli utenti devono poter cercare libri tramite titolo, autore o genere.
  - **Priorità**: Alta
  - **Input**: Parola chiave di ricerca
  - **Output**: Elenco di libri che corrispondono ai criteri di ricerca.
- **RF3: Prenotazione dei libri**
  - **Descrizione**: Gli utenti devono poter prenotare un libro disponibile e ricevere una conferma della prenotazione.
  - **Priorità**: Alta
  - **Input**: Selezione del libro

  - **Output**: Conferma della prenotazione.

- **RF4: Gestione dell'inventario da parte degli admin**
  - **Descrizione**: Gli admin devono poter aggiungere, modificare o rimuovere libri dal sistema.
  - **Priorità**: Alta
  - **Input**: Dati del libro (titolo, autore, descrizione, ecc.)
  - **Output**: Aggiornamento dell'inventario della biblioteca.

6.4 Requisiti Non Funzionali

- **RNF1: Usabilità**
  - **Descrizione**: L'interfaccia utente deve essere semplice e intuitiva, consentendo agli utenti di eseguire operazioni comuni con facilità.
  - **Priorità**: Media

- **RNF2: Prestazioni**
  - **Descrizione**: Il sistema deve rispondere alle richieste degli utenti entro 3 secondi durante la ricerca e la prenotazione dei libri.
  - **Priorità**: Alta

- **RNF3: Sicurezza**
  - **Descrizione**: Le credenziali degli utenti devono essere criptate, e il sistema deve prevenire accessi non autorizzati.
  - **Priorità**: Alta

6.5 Attributi di Qualità

- **Affidabilità**: Il sistema deve essere operativo il 99,9% del tempo, con ridondanza nei componenti critici.

- **Manutenibilità**: Il codice deve essere modulare e ben documentato, facilitando gli aggiornamenti e le correzioni future.

6.6 Altri Requisiti

- **Compatibilità**: Il sistema deve essere compatibile con i principali browser (Chrome, Firefox, Safari) e con dispositivi mobili.

- **Scalabilità**: Il sistema deve poter gestire un aumento del numero di utenti senza degradare le prestazioni.

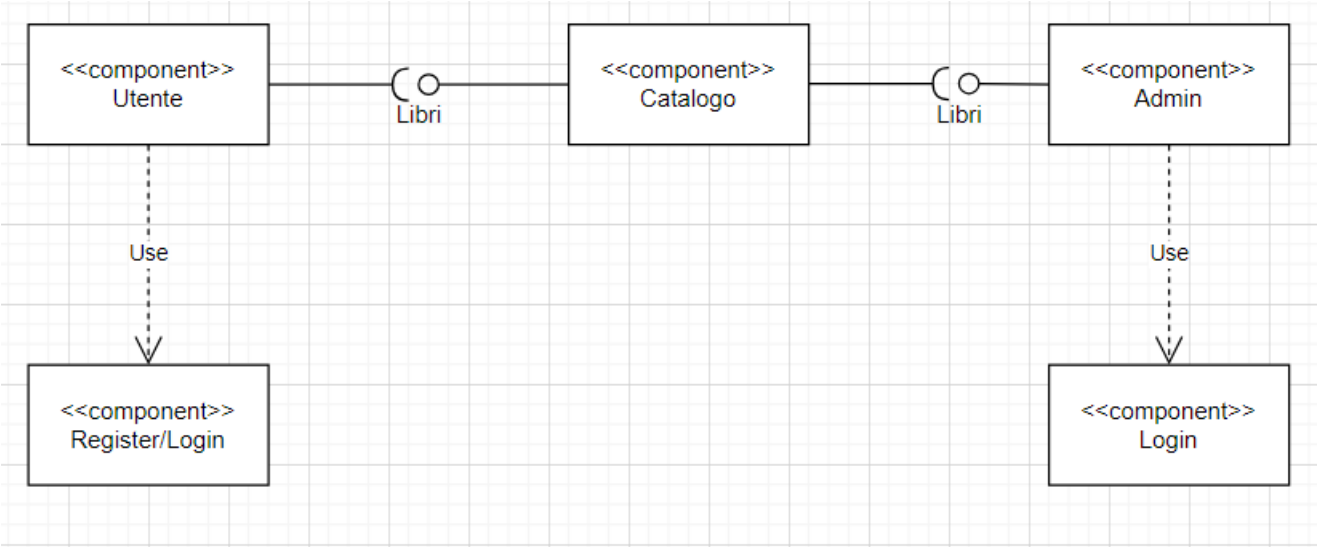
Questa specifica dei requisiti ha guidato lo sviluppo del sistema, garantendo che tutte le funzionalità necessarie fossero correttamente implementate e che il software risultasse robusto e adatto all'uso previsto.

Per essere il piu precisi possibile e creare il giusto applicativo richiesto dal nostro cliente, è sembrato opportuno svolgere un'intervista al cliente per poter raccogliere più informazioni possibili su come e quando il progetto doveva essere svolto.

6.7 Intervista al Cliente

- **Intervistatore**: Buongiorno, grazie per averci dedicato il suo tempo. Per iniziare, potrebbe descrivere brevemente quali sono le principali funzionalità che desidera nel sistema della biblioteca virtuale?
- **Cliente**: Certamente. Vorrei che il sistema permettesse agli utenti di cercare e prenotare i libri online. Inoltre, ci deve essere una sezione dedicata agli amministratori, dove possano gestire l'inventario, aggiungere nuovi libri e aggiornare le informazioni esistenti.
- **Intervistatore**: Capisco. E per quanto riguarda l'accesso al sistema, ha delle preferenze specifiche su come dovrebbero avvenire l'autenticazione e la gestione degli utenti?
- **Cliente**: Sì, mi piacerebbe che gli utenti si registrassero con un nome utente e una password. Inoltre, è importante che ci sia una distinzione chiara tra i permessi di un utente normale e quelli di un amministratore.
- **Intervistatore**: Perfetto. Riguardo l'interfaccia utente, ci sono particolari caratteristiche che vorrebbe vedere implementate?
- **Cliente**: Vorrei che l'interfaccia fosse semplice e intuitiva, in modo che anche chi non è molto esperto di tecnologia possa utilizzarla senza problemi. In particolare, la ricerca dei libri dovrebbe essere facile e veloce, con opzioni di filtro.
- **Intervistatore**: Questo è molto utile. Ha pensato a come vorrebbe che fossero gestiti gli aggiornamenti e la manutenzione del sistema?
- **Cliente**: Sì, mi piacerebbe che fosse facile aggiungere nuovi libri o modificare quelli esistenti senza bisogno di conoscenze tecniche particolari. Inoltre, il sistema dovrebbe essere robusto e affidabile, con tempi di inattività ridotti al minimo.
- **Intervistatore**: E per quanto riguarda la sicurezza? Ci sono delle specifiche misure di sicurezza che vorrebbe implementare?
- **Cliente**: La sicurezza è una priorità. I dati degli utenti e le loro credenziali devono essere protetti, e il sistema deve prevenire accessi non autorizzati.
- **Intervistatore**: Ottimo, grazie per le informazioni dettagliate. Ci sono altre caratteristiche o requisiti che vorrebbe aggiungere?
- **Cliente**: Per ora penso di aver coperto tutto. Mi piacerebbe solo che il sistema fosse scalabile, in modo da poter supportare un numero crescente di utenti man mano che la biblioteca si espande.
- **Intervistatore**: Perfetto, grazie mille per il suo tempo e per tutte le informazioni fornite. Questo ci aiuterà molto nella progettazione del sistema.

7 Modeling



8 Software Architecture

8.1 Vista Logica (Logical View)

La vista logica si concentra sulla struttura e sull'organizzazione delle funzionalità del sistema, descrivendo i componenti principali e le loro interazioni. Nel contesto della biblioteca virtuale, i principali componenti sono:

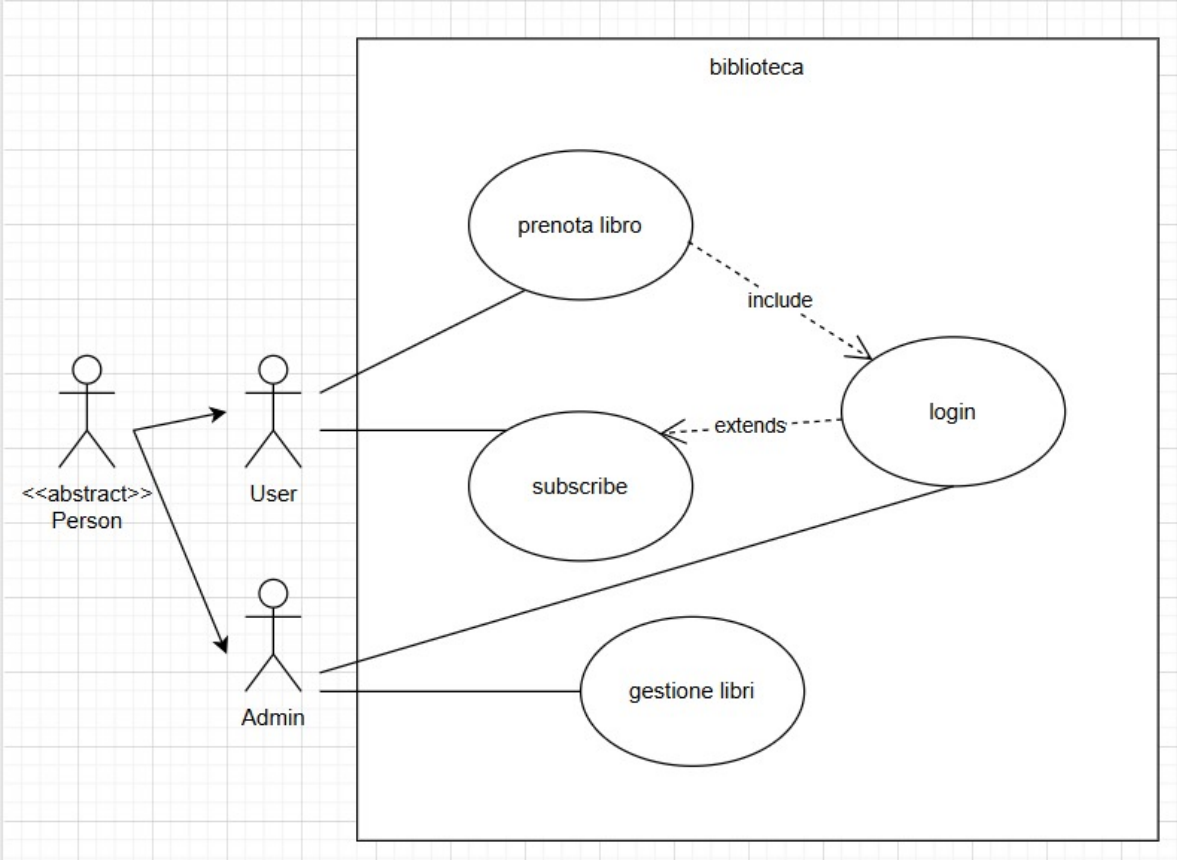
- **User Interface (GUI)**: L'interfaccia utente è composta da finestre per il login, la registrazione, la ricerca e la prenotazione di libri, nonché per la gestione dei libri da parte dell'amministratore. La GUI è responsabile della comunicazione tra l'utente e il sistema.
- **Gestione Utenti**: Modulo che si occupa dell'autenticazione e della gestione delle sessioni utente. Gestisce l'accesso differenziato tra utenti regolari e amministratori.
- **Gestione dei Libri**: Modulo che si occupa delle operazioni sui libri (aggiunta, aggiornamento, rimozione, prenotazione, ricerca). Gli utenti possono visualizzare i libri disponibili e prenotarli, mentre gli amministratori possono gestire i titoli o aggiungerne di nuovi.
- **Database**: Il database contiene tutte le informazioni sui libri, gli utenti, le prenotazioni e le sessioni. Si interfaccia con i moduli di gestione per fornire e aggiornare i dati.

8.2 Vista Viewpoint View (User View)

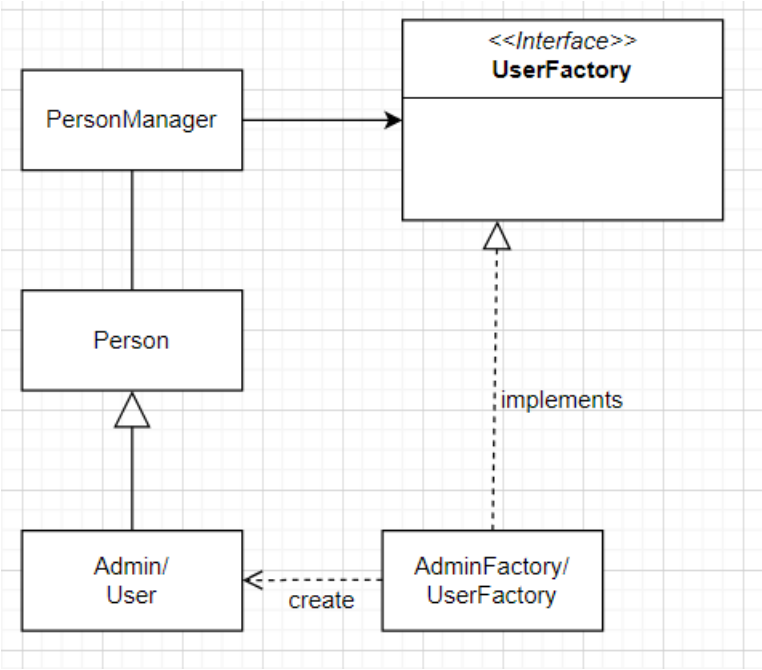
La vista viewpoint view descrive le necessità e le aspettative degli utenti che interagiscono con il sistema.

- **Utente Regolare**:
  - **Obiettivi**: Cercare e prenotare libri nella bacheca. Gestire le prenotazioni e visualizzare lo stato dei libri prenotati.
  - **Funzionalità**:
    - **Ricerca Libri**: L'utente può cercare libri per titolo.
    - **Prenotazione**: L'utente può prenotare un libro se disponibile.
    - **Gestione Prenotazioni**: L'utente può visualizzare e annullare le prenotazioni dalla propria area riservata.
- **Amministratore**:

– **Obiettivi**: Gestire il catalogo dei libri, aggiungere nuovi libri, aggiornare informazioni e rimuovere titoli non più disponibili.
– **Funzionalità**:
  * **Aggiunta Libri**: L'amministratore può aggiungere nuovi libri, specificando titolo, autore, genere e disponibilità.
  * **Modifica Informazioni**: È possibile aggiornare i dettagli dei libri esistenti.
  * **Rimozione Libri**: L'amministratore può rimuovere libri non più disponibili.



9 Software Design



- **UserManager**: Questo rappresenta il gestore degli utenti, che si occuperà di gestire la creazione di diversi tipi di utenti.
- **UserFactory**: Questa interfaccia sarà usata per creare utenti differenti, come **RegularUser** o **Admin**.
- **AdminFactory** / **RegularUserFactory**: Questi sono i factory concreti che creano i diversi tipi di utenti.
- **Admin** o **RegularUser**: Questi rappresentano i tipi concreti di utenti che verranno creati.
- **User**: Questo rappresenta la classe base o l'interfaccia comune a tutti i tipi di utenti.

Quindi, il diagramma potrebbe diventare qualcosa del genere:

- **UserManager**: gestisce la creazione e gestione degli utenti.
- **UserFactory** (interfaccia): definisce il metodo per creare utenti.
- **AdminFactory** e **RegularUserFactory**: implementano **UserFactory** e creano istanze di **Admin** e **RegularUser**.
- **Admin** e **RegularUser**: sono i tipi concreti di utenti creati.

10 Software Testing

10.1 Test per il metodo `ricercaPassword(String utente, String password)`

Il metodo `ricercaPassword(String utente, String password)` controlla se un utente è di livello 0 (admin) o 1 (utente). I seguenti test coprono i diversi casi:

1. **testUtenteNormale:** Verifica che il metodo restituisca `true` quando viene fornito un nome utente e una password validi per un utente normale (livello 1).
2. **testUtenteAdmin:** Verifica che il metodo restituisca `false` quando viene fornito un nome utente e una password validi per un amministratore (livello 0).

3. **testCredenzialiErrate:** Verifica che il metodo restituisca `false` quando vengono fornite credenziali non valide (utente o password errati).

4. **testUtenteNonEsistente:** Verifica che il metodo restituisca `false` quando si cerca di autenticare un utente che non esiste nel database.

10.2 Test per il Metodo `doppi(String email, String utente)`

Il metodo `doppi` controlla se esistono duplicati nel database per email e nome utente. I seguenti test coprono i diversi casi:

1. **testNessunDuplicato:** Verifica il caso in cui non ci sono duplicati. Si aspetta che il metodo restituisca `true` quando viene fornita una nuova email e un nuovo nome utente.

2. **testEmailDuplicata:** Verifica se il metodo restituisce `false` quando viene fornita un'email già esistente nel database.

3. **testUtenteDuplicato:** Verifica se il metodo restituisce `false` quando viene fornito un nome utente già esistente nel database.

4. **testEmailEUtenteDuplicati:** Verifica se il metodo restituisce `false` quando sia l'email che il nome utente esistono già nel database.

10.3 Test per il Metodo `trovaloggato()`

Il metodo `trovaloggato` cerca nel database un utente con l'attributo `isLogged` pari a 1. I seguenti test coprono i diversi casi:

1. **testUtenteLoggato:** Verifica che il metodo restituisca il nome dell'utente attualmente loggato (ad esempio, `"utente1"`).

2. **testUltimoUtenteLoggato:** Verifica che il metodo restituisca il nome dell'ultimo utente loggato (ad esempio, `"ultimoUtenteLoggato"`) quando nessun utente è attualmente loggato.

3. **testNessunUtenteLoggato:** Verifica che il metodo restituisca una stringa vuota quando non ci sono utenti loggati e non ci sono ultimi loggati disponibili.

11 Software Maintenance

Abbiamo seguito le seguenti tipologie di refactoring: interventi per evitare la ridondanza del codice scritto tra i collaboratori, semplificazione del codice, infine cerchiamo di attuare della manutenzione preventiva, per esempio attraverso una buona documentazione e dei commenti regolari nel codice. Il tutto è semplificato grazie all'utilizzo di Github.