UNIVERSITÀ POLITECNICA DELLE MARCHE
FACOLTÀ DI INGEGNERIA
Dipartimento di Ingegneria dell'Informazione
Corso di Laurea in Ingegneria Informatica e dell'Automazione

TESI DI LAUREA

Progettazione e implementazione in Flutter di un'app per la gestione della biblioteca dell'Università Politecnica delle Marche

Design and implementation in Flutter of an app for managing the library of the Polytechnic University of Marche

Relatore

Prof. Domenico Ursino

Candidato

Zaki Mina Youssef Fahmy

ANNO ACCADEMICO 2023-2024

Un uomo saggio impara da una domanda sciocca più di quanto uno sciocco possa imparare da una risposta saggia

Bruce Lee

Sommario

La gestione informativa della biblioteca universitaria comprende la pianificazione, l'erogazione e il controllo dei servizi di prestito, catalogazione e consultazione dei materiali, con l'obiettivo di garantire un accesso rapido e organizzato alle risorse accademiche e di ricerca. In ambito universitario, la biblioteca rappresenta un supporto fondamentale per la didattica e l'attività scientifica, offrendo un ampio patrimonio di risorse, sia fisiche che digitali. L'app Android della biblioteca, ottimizzata per l'utilizzo mobile, si concentra sull'efficienza nella gestione delle collezioni e dei servizi, consentendo agli studenti di prenotare libri e postazioni di studio, e al personale amministrativo di svolgere in modo efficace le operazioni relative ai volumi e alle prenotazioni.

Keyword: Biblioteca, Libri, Prenotazione, Database, Flutter, Firebase, IMGBB, Dart

Indice

Elenco delle figure

Elenco delle tabelle

Introduzione

Nell'era digitale, l'ottimizzazione dei servizi bibliotecari passa sempre più attraverso soluzioni tecnologiche che semplificano l'accesso e la gestione delle risorse. In un contesto universitario, dove studenti e docenti necessitano di strumenti efficienti per la ricerca e lo studio, un'applicazione mobile dedicata alla prenotazione di libri e posti studio può fare la differenza. Questo progetto nasce con l'obiettivo di rispondere a tali esigenze, offrendo un sistema integrato e facile da utilizzare.

L'app per la gestione della biblioteca universitaria non si limita a digitalizzare i processi esistenti, ma introduce nuove funzionalità che migliorano l'interazione tra utenti e struttura. Gli studenti possono prenotare in pochi click i libri desiderati o riservare spazi di studio, mentre il personale amministrativo ha a disposizione strumenti avanzati per monitorare e gestire l'intero sistema. Tutto questo è reso possibile grazie all'integrazione con piattaforme come Firebase, per la gestione dei dati, e IMGBB, per il caricamento delle immagini.

La gestione dei dati raccolti tramite l'app è essenziale per assicurare un funzionamento efficiente e un'esperienza utente ottimale. Attraverso un sistema organizzato, l'app consente di monitorare le prenotazioni, gestire la disponibilità dei libri e degli spazi di studio, e facilitare il lavoro del personale bibliotecario. Questo approccio garantisce un accesso rapido e ordinato alle risorse, offrendo agli utenti un servizio che risponde pienamente alle loro esigenze.

La presente tesi esplora l'intero processo di sviluppo dell'app, dalla progettazione iniziale alla fase di implementazione, soffermandosi sulle scelte tecnologiche e sulle soluzioni adottate. Inoltre, verrà presentata un'analisi SWOT per valutare le potenzialità e i margini di miglioramento del sistema. L'obiettivo finale è quello di offrire uno strumento pratico e innovativo, capace di migliorare l'esperienza degli utenti e di supportare il lavoro del personale bibliotecario.

La presente tesi è composta da sei capitoli strutturati come di seguito specificato:

- Nel Capitolo 1 sarà introdotto Flutter, il framework utilizzato per lo sviluppo dell'applicazione. Verranno, inoltre, presentate le principali caratteristiche e i vantaggi offerti dalla piattaforma.
- Nel Capitolo 2 saranno analizzati i requisiti raccolti e sarà illustrato il diagramma dei casi d'uso, utile per comprendere le funzionalità richieste e le interazioni tra gli attori e il sistema.
- Nel Capitolo 3 sarà descritta la progettazione del database, con particolare attenzione alla struttura e alla definizione delle entità necessarie per supportare le funzionalità dell'applicazione.

- Nel Capitolo 4 verrà approfondita la progettazione della componente applicativa, comprendendo sia la definizione dell'interfaccia utente tramite mockup, sia l'integrazione con i servizi Firebase e IMGBB per la gestione dei dati e delle immagini.
- Nel Capitolo 5 saranno analizzate le fasi di implementazione del progetto. Verranno illustrate le componenti principali dell'app e, successivamente, sarà fornita una guida pratica destinata agli utenti finali.
- Nel Capitolo 6 sarà condotta un'analisi SWOT (Strengths, Weaknesses, Opportunities, Threats) per esaminare i punti di forza, le debolezze, le opportunità di crescita e le minacce dell'applicazione sviluppata.

CAPITOLO 1

Introduzione a Flutter

In questo capitolo introduttivo l'obiettivo è fornire una panoramica di Flutter, il framework utilizzato per sviluppare l'applicazione di gestione della biblioteca. Dopo aver contestualizzato la tecnologia, verranno descritte le sue caratteristiche principali e i concetti fondamentali. Questo processo introduttivo è essenziale per comprendere i vantaggi offerti da Flutter nella realizzazione del progetto. Successivamente, verranno illustrate l'architettura di Flutter e le principali componenti necessarie per costruire un'applicazione reattiva e multi-piattaforma.

1.1 Panoramica di Flutter e della sua storia

Flutter è un framework open-source sviluppato da Google per la creazione di applicazioni cross-platform, consentendo agli sviluppatori di scrivere un'unica base di codice per diverse piattaforme, tra cui Android, iOS, web e desktop.

La storia di Flutter inizia con il rilascio della prima versione stabile, Flutter 1.0, presentata ufficialmente nel dicembre 2018 durante la conferenza Flutter Live. Da allora, Google ha continuato a investire nel framework, ampliando il supporto a nuove piattaforme e migliorando le prestazioni complessive. La versione Flutter 2, lanciata nel 2021, ha introdotto il supporto per il Web e migliorato la compatibilità con il desktop, consolidando Flutter come un vero strumento multipiattaforma. Con il rilascio di Flutter 3 nel 2022, il framework ha completato il passaggio a una piattaforma completamente multipiattaforma, aggiungendo supporto ufficiale per macOS e Linux e ampliando le integrazioni con servizi Google come Firebase.

1.1.1 Flutter come cross-platform

Con il termine "cross-platform" si intende la capacità di un framework di sviluppare applicazioni che possano funzionare su più sistemi operativi (come Android, iOS, Web e Desktop) utilizzando un unico codice base. Questo approccio non solo riduce il tempo di sviluppo, ma anche i costi di manutenzione e aggiornamento dell'applicazione, garantendo, al contempo, una qualità simile a quella delle app native. Questo interesse per il mobile cross-platform è iniziato subito dopo il rilascio del primo SDK (Software Development Kit) per iPhone e la nascita di Android.

Flutter è progettato come un sistema estensibile e stratificato. Esiste come una serie di librerie indipendenti, ciascuna delle quali dipende dal layer sottostante. Nessun layer ha

accesso privilegiato al layer sottostante e ogni parte del livello del framework è progettata per essere opzionale e sostituibile.

L'architettura di Flutter è riassunta graficamente nella Figura 1.1, dove è possibile vedere i tre livelli principali che compongono l'architettura interna e le loro principali responsabilità.

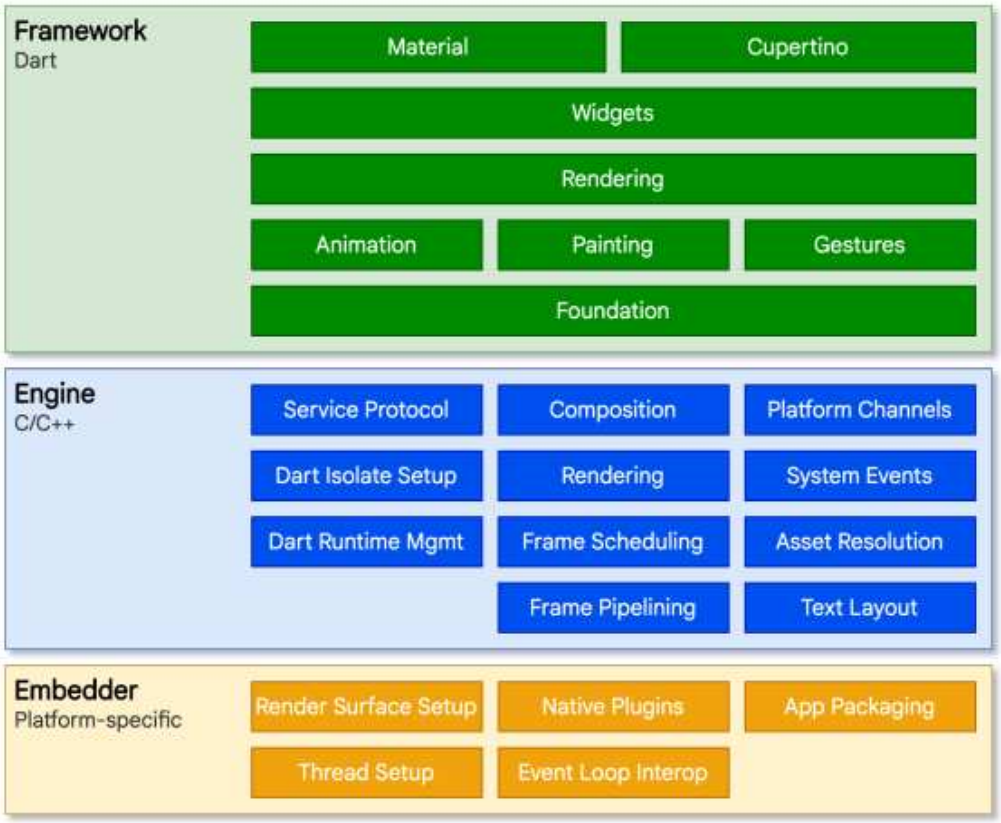

Figura 1.1: Rappresentazione schematica dell'architettura delle applicazioni native

1.1.2 Flutter per il Web e il Desktop

Per gli sviluppatori esperti di app web, Flutter offre un framework rapido ed efficiente, riducendo i tempi di sviluppo e migliorando la stabilità e le prestazioni delle app web. Le app Web Flutter si integrano facilmente con gli strumenti front-end standard, come HTML, CSS e JavaScript, il che significa che i team di sviluppo web che passano alle app Flutter possono mantenere il loro investimento in altri framework di terze parti come, AngularJS e VueJS.

Inoltre, per chi preferisce lavorare con un IDE, è probabile che il proprio ambiente di sviluppo preferito supporti lo sviluppo di applicazioni Flutter, consentendo di evitare i costi legati all'apprendimento di nuovi strumenti. Visual Studio Code, Android Studio e IntelliJ IDEA offrono pieno supporto per lo sviluppo di applicazioni web tramite i plugin Flutter e Dart, mettendo a disposizione degli sviluppatori di applicazioni web una vasta gamma di opzioni. Flutter per il web permette agli sviluppatori di applicare le competenze e le risorse già acquisite nello sviluppo mobile per creare applicazioni web dinamiche e interattive.

1.1.3 Linguaggio moderno e tipizzazione forte

Dart, scelto da Google per Flutter, è stato progettato per creare applicazioni reattive, moderne e ad alte prestazioni. Questo linguaggio si distingue per la sua sintassi intuitiva, la

rapidità di esecuzione e il supporto avanzato alla tipizzazione forte, che lo rendono ideale per applicazioni multipiattaforma. Essendo orientato agli oggetti, è ottimizzato per garantire un'esecuzione efficiente sia su dispositivi mobili che su Web.

Il sistema di tipizzazione forte adottato in Dart richiede la dichiarazione e il rispetto dei tipi di dati per variabili e funzioni. Questa tipizzazione è principalmente statica (verificata in fase di compilazione), ma rimane flessibile grazie all'inferenza dei tipi, che consente di omettere dichiarazioni esplicite in molti casi, pur garantendo il controllo sui tipi di dati utilizzati.

La tipizzazione forte offre diversi vantaggi significativi per lo sviluppo:

- *Sicurezza del codice*: la tipizzazione riduce il rischio di errori di tipo, poiché impedisce che variabili con tipi errati siano assegnate o utilizzate in modo improprio. Questo rende il codice meno incline a bug che potrebbero emergere solo durante l'esecuzione, migliorando, così, la stabilità complessiva del software.
- *Intellisense e autocompletamento migliorati*: la dichiarazione dei tipi consente all'ambiente di sviluppo (IDE) di offrire suggerimenti più accurati e un autocompletamento più efficace durante la scrittura del codice, incrementando la produttività del programmatore e riducendo il rischio di errori.
- *Manutenibilità del codice*: la chiarezza sui tipi facilita la comprensione del codice per altri sviluppatori, rendendo più agevole la manutenzione e l'aggiornamento del progetto nel tempo. La tipizzazione esplicita contribuisce, inoltre, a chiarire il comportamento previsto delle variabili e delle funzioni, supportando, così, una documentazione implicita e migliorando la coerenza del codice.

1.1.4 Progetti famosi realizzati con Flutter

Nella Figura 1.2, sono riportati alcuni dei progetti più noti realizzati con Flutter:

Figura 1.2: Esempi di applicazioni realizzate con Flutter

- L'app Google Ads è uno degli esempi più significativi dell'uso di Flutter. Questa applicazione consente agli utenti di gestire le proprie campagne pubblicitarie direttamente dai dispositivi mobili. Grazie a Flutter, Google Ads ha potuto migliorare l'affidabilità e la funzionalità dell'app, offrendo agli utenti statistiche dettagliate sulle campagne e la possibilità di contattare esperti Google direttamente dall'app. La capacità di Flutter di fornire prestazioni elevate e un'interfaccia utente coerente ha reso questa app un successo tra gli inserzionisti.
- La famosa casa automobilistica BMW ha sviluppato l'app My BMW utilizzando Flutter. Questa applicazione consente ai proprietari di interagire con i propri veicoli, controllando funzioni come il bloccaggio e lo sbloccaggio delle porte, la localizzazione dell'auto

e la navigazione. L'adozione di Flutter ha permesso a BMW di mantenere una coerenza visiva tra le versioni iOS e Android dell'app, riducendo i tempi di sviluppo e migliorando l'esperienza utente complessiva.

- L'app ufficiale del famoso musical Broadway "Hamilton" è stata realizzata con Flutter. Questa applicazione offre contenuti esclusivi, lotterie giornaliere per i biglietti dello spettacolo e informazioni dettagliate sulla produzione. La scelta di utilizzare Flutter ha permesso agli sviluppatori di creare un'app visivamente accattivante e altamente interattiva in tempi rapidi, dimostrando l'efficacia del framework nel settore dell'intrattenimento.

1.2 Confronto con altri framework

Si procederà ad un'analisi di Flutter, confrontandolo con altre soluzioni per lo sviluppo di applicazioni mobili, evidenziando vantaggi e svantaggi dei diversi approcci. Lo sviluppo di applicazioni multipiattaforma è fondamentale per le aziende che desiderano raggiungere utenti su più dispositivi con il minor sforzo possibile.

1.2.1 Confronto con soluzioni native

Le app native comunicano direttamente con il sistema operativo, fornendo le migliori prestazioni e funzionalità. Tuttavia, questo approccio richiede uno sviluppo dedicato per ogni piattaforma (come Android e iOS), rendendo complesso il mantenimento e l'aggiornamento del codice. Ogni modifica deve essere replicata in ciascun codebase separato, aumentando i costi di sviluppo.

Come mostrato in Figura 1.3, l'architettura delle applicazioni native permette un accesso diretto alle risorse del sistema operativo, garantendo alte prestazioni.

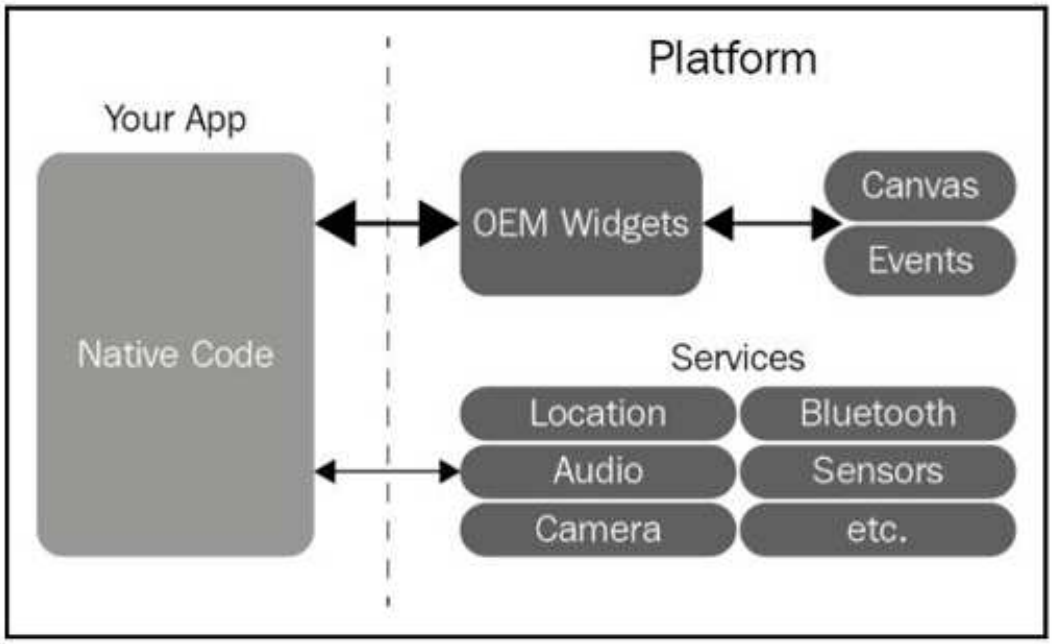

Figura 1.3: Rappresentazione schematica dell'architettura delle applicazioni native

1.2.2 Confronto tra framework cross-platform

Esistono diverse filosofie che possono guidare gli approcci cross-platform, tra cui framework basati su WebView e framework che impiegano il concetto di bridge, come React Native.

Flutter si distingue come alternativa grazie alla sua gestione diretta del rendering grafico e alla compilazione Ahead of Time (AOT).

I framework come Cordova, Ionic e PhoneGap utilizzano una WebView, ovvero una vista del browser, per eseguire codice HTML e JavaScript all'interno di un'app mobile. Questo approccio consente a chi ha esperienza nello sviluppo web di creare applicazioni multipiattaforma rapidamente. Tuttavia, le prestazioni sono generalmente inferiori rispetto a quelle delle app native, poiché l'interfaccia utente viene renderizzata come una pagina web e non accede direttamente alle risorse del sistema operativo.

La Figura 1.4, illustra come funziona l'architettura basata su WebView; il codice HTML viene renderizzato all'interno della WebView, che interagisce con il sistema operativo attraverso un "bridge" JavaScript. Questo "bridge" permette l'accesso a funzionalità di base del dispositivo, ma introduce ritardi che possono compromettere la fluidità dell'applicazione.

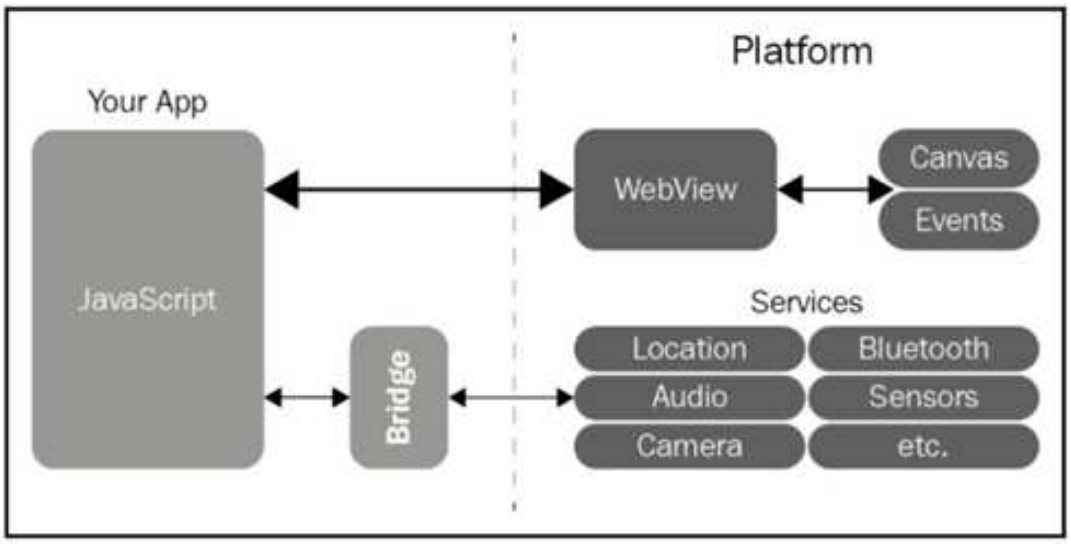

Figura 1.4: Architettura basata su WebView

Questo significa che i framework basati su WebView sono ideali per gli sviluppatori che già lavorano con tecnologie web (HTML, CSS e JavaScript). Grazie a WebView, è possibile creare un'app mobile utilizzando lo stesso codice e le stesse competenze già impiegate per creare siti web. In questo modo, si riduce il lavoro necessario per sviluppare un'app multipiattaforma, poiché il front-end può essere riutilizzato senza dover imparare nuovi linguaggi o strumenti specifici per mobile.

React Native è un esempio di framework basato sul concetto di bridge che consente di sviluppare app utilizzando componenti nativi specifici di ciascuna piattaforma, ma fa uso di un bridge che traduce il codice JavaScript in chiamate native. Questo approccio offre prestazioni migliori rispetto alle soluzioni basate su WebView, ma non raggiunge l'efficienza delle app completamente native, poiché ogni operazione grafica passa attraverso il bridge, che può introdurre rallentamenti, specialmente in animazioni complesse.

La Figura 1.5, mostra l'architettura di React Native, dove il codice JavaScript comunica con il sistema operativo tramite un bridge che gestisce i componenti nativi. Anche se offre un

buon compromesso in termini di prestazioni e riusabilità del codice, la necessità di passare attraverso il bridge può causare ritardi, soprattutto per operazioni grafiche che richiedono il ri-rendering frequente dei componenti.

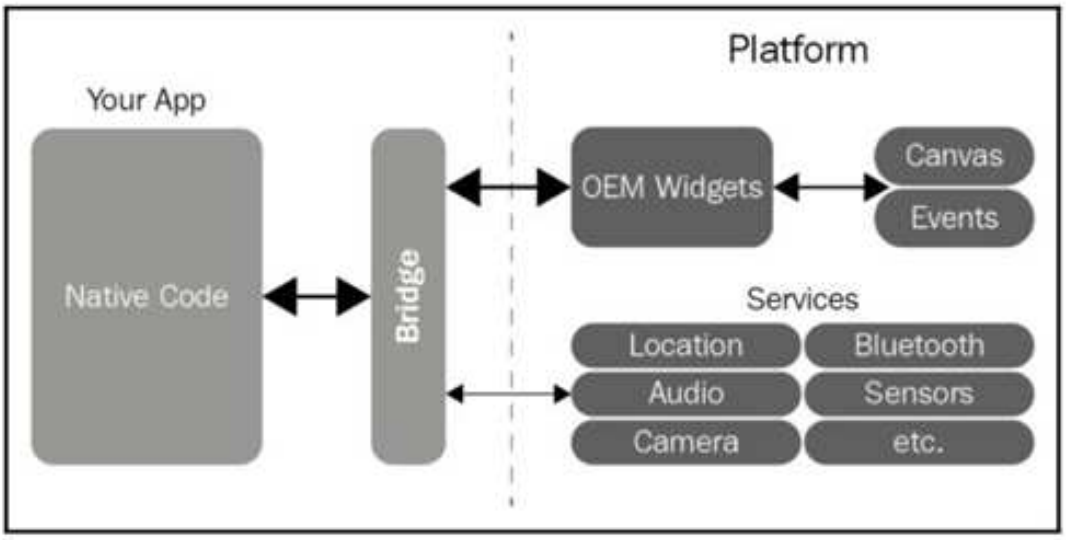

Figura 1.5: Architettura basata su bridge

Flutter si distingue dagli approcci precedenti grazie alla compilazione AOT e all'assenza di WebView o bridge. Quindi il codice viene compilato in anticipo, prima che l'app venga eseguita sul dispositivo. Con AOT, il codice Dart viene trasformato direttamente in codice nativo specifico per il sistema operativo (iOS o Android) già durante la fase di sviluppo.

Flutter non usa un bridge come React Native o altri framework JavaScript per comunicare con le funzionalità del sistema operativo. Invece, Flutter usa tutto direttamente sullo schermo tramite un suo motore di rendering grafico (Figura 1.6). Questo approccio di widget rendering diretto garantisce che l'aspetto e il comportamento dell'app siano gli stessi sia su Android che su iOS, offrendo coerenza visiva e alte prestazioni su entrambe le piattaforme.

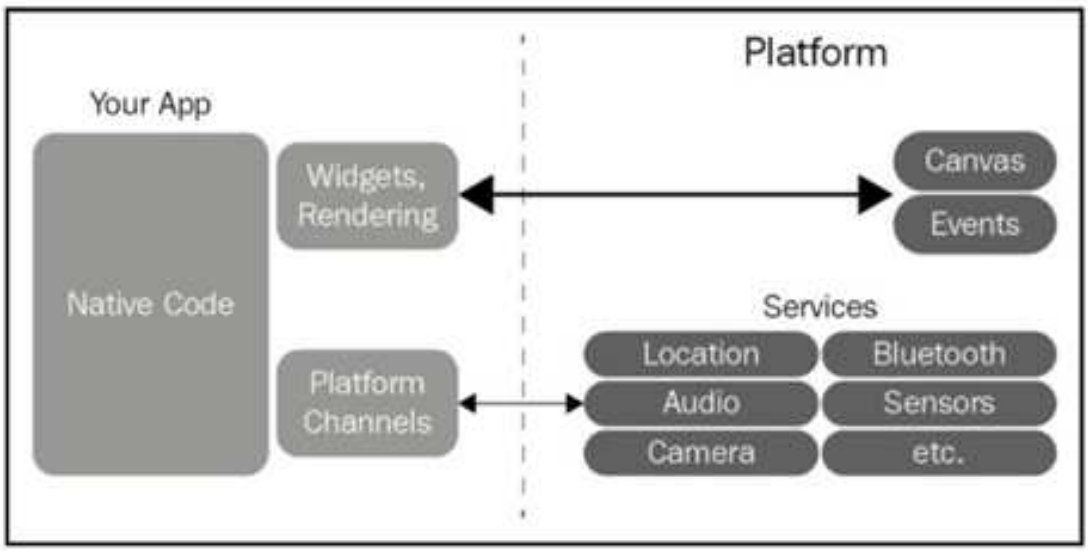

Figura 1.6: Approccio di Flutter

1.3 Rendering e prestazioni

Flutter si distingue come framework cross-platform per le sue prestazioni elevate e l'approccio unico al rendering, che evita i limiti di altre tecnologie multipiattaforma. L'efficienza del rendering in Flutter è il risultato di un'architettura progettata per offrire un'esperienza utente fluida su Android, iOS e altre piattaforme, senza sacrificare la qualità grafica o le prestazioni.

Il rendering in Flutter si basa su una struttura di widget che consente agli sviluppatori di progettare interfacce utente complesse e reattive attraverso la composizione di widget, gli elementi costitutivi dell'interfaccia in Flutter. Ogni parte dell'applicazione è rappresentata da un widget, inclusi layout, animazioni e componenti visivi, i quali vengono combinati per formare strutture visive gerarchiche

Flutter utilizza la compilazione Ahead of Time (AOT) per le sue applicazioni distribuite, differenziandosi dai framework che si affidano alla compilazione Just in Time (JIT), come avviene nei linguaggi basati su JavaScript. La compilazione AOT permette a Flutter di trasformare il codice Dart in codice nativo durante il processo di sviluppo, generando un'app più veloce e reattiva. Questo riduce i tempi di esecuzione e ottimizza l'utilizzo delle risorse del dispositivo, migliorando così la qualità e la fluidità delle animazioni e delle interazioni utente.

In fase di sviluppo, Flutter usa la compilazione JIT, che consente di aggiornare il codice in tempo reale attraverso la funzione Hot Reload. Questo rende l'esperienza di sviluppo estremamente efficiente, poiché le modifiche al codice sono visibili istantaneamente senza bisogno di ricompilare l'intera applicazione.

L'architettura di Flutter non si basa sulle componenti OEM (Original Equipment Manufacturer) della piattaforma ospitante, ma costruisce l'interfaccia con widget indipendenti dalle librerie native di Android o iOS. Flutter crea i propri widget, progettati per rispecchiare i design sia Material (per Android) che Cupertino (per iOS), ma implementati direttamente da Flutter, senza fare affidamento sulle librerie di sistema. Ciò significa che Flutter non solo garantisce una consistenza visiva e funzionale tra le piattaforme, ma evita anche l'uso di un bridge che potrebbe introdurre rallentamenti.

Questo approccio consente di ottenere una performance elevata anche in applicazioni che richiedono un uso intensivo del rendering grafico, come quelle con animazioni avanzate o interfacce interattive. Grazie alla gestione diretta del canvas di rendering, Flutter garantisce che le prestazioni non vengano compromesse da operazioni di traduzione tra linguaggi o da limitazioni nelle librerie native

Flutter offre una serie di strumenti per ottimizzare le prestazioni di animazioni e gesti, elementi critici per la reattività dell'applicazione. Il livello di rendering in Flutter permette di costruire un albero di oggetti renderizzabili, che può essere manipolato in modo dinamico senza interruzioni. Le animazioni possono essere gestite e sincronizzate direttamente tramite la libreria di animazione integrata in Flutter, riducendo il carico sul sistema e migliorando la fluidità dell'interfaccia.

Inoltre, Flutter include widget specifici per la gestione dei gesti, consentendo di creare interazioni complesse e fluide senza dover ricorrere a soluzioni esterne. Ogni gesto viene gestito direttamente all'interno del framework, permettendo una risposta immediata e riducendo al minimo il ritardo tra l'input dell'utente e la risposta dell'applicazione.

1.4 Gestione delle dipendenze

La gestione delle dipendenze è una componente essenziale nello sviluppo di applicazioni moderne, consentendo agli sviluppatori di integrare pacchetti esterni, librerie e funzionalità

già pronte all'uso, riducendo il tempo di sviluppo e aumentando la manutenibilità del codice. Flutter offre un sistema di gestione delle dipendenze avanzato e semplificato attraverso Pub, il gestore di pacchetti per Dart, che permette di aggiungere, aggiornare e gestire librerie e plugin per arricchire le funzionalità di un'applicazione.

Flutter utilizza Pub come gestore di pacchetti, specificamente progettato per gestire le dipendenze del linguaggio Dart, fornendo un sistema centralizzato per accedere a migliaia di librerie e plugin, molti dei quali sviluppati dalla comunità open-source. Pub rende disponibile un repository ufficiale su `pub.dev`, una piattaforma che raccoglie pacchetti verificati e documentati, facilmente integrabili nel progetto.

Gli sviluppatori possono aggiungere dipendenze esterne definendole nel file `pubspec.yaml`, un file di configurazione che specifica i pacchetti richiesti, le versioni, le dipendenze di sviluppo e altre impostazioni. Una volta configurato, Pub gestisce automaticamente il download e l'installazione delle librerie specificate, assicurandosi che siano pronte per l'utilizzo all'interno dell'applicazione.

In Flutter, le dipendenze si classificano in tre categorie principali:

- *Dipendenze dirette*: queste sono librerie che l'applicazione utilizza direttamente. Possono includere pacchetti per interfacce utente, animazioni, rete e database, e sono dichiarate nel file `pubspec.yaml` sotto la sezione `dependencies`.
- *Dipendenze di sviluppo*: sono librerie utilizzate solo durante lo sviluppo e i test, ma non incluse nella versione finale dell'app. Un esempio comune è `flutter_test`, una libreria utilizzata per il testing dell'app. Queste dipendenze vengono specificate nella sezione `dev_dependencies` e aiutano a mantenere il progetto snello, limitando le librerie caricate nell'app finale.
- *Dipendenze transitive*: una libreria può richiedere altre librerie per funzionare, creando dipendenze indirette o transitive. Pub gestisce queste dipendenze automaticamente, assicurando che tutte le librerie richieste siano presenti senza conflitti di versione.

Quando si aggiunge una nuova dipendenza, il comando `flutter pub get` scarica e installa tutti i pacchetti richiesti, configurando l'ambiente affinché possa riconoscere ed eseguire i plugin integrati. Questa procedura è essenziale per garantire che le versioni dei pacchetti siano compatibili tra loro e con Flutter stesso. Pub offre anche il comando `flutter pub upgrade`, che consente di aggiornare tutte le dipendenze all'ultima versione compatibile, aiutando a mantenere il codice aggiornato e a beneficiare delle ultime funzionalità e correzioni di sicurezza.

La piattaforma `pub.dev` ospita migliaia di pacchetti, tra cui plugin ufficiali sviluppati dal team di Flutter e pacchetti creati dalla comunità. Gli sviluppatori possono esplorare la piattaforma per trovare librerie che aggiungono funzionalità specifiche, come la gestione della fotocamera, notifiche push, accesso ai sensori di dispositivo e integrazioni con servizi web. La community di Flutter è attiva nello sviluppo e nel mantenimento di questi pacchetti, fornendo aggiornamenti frequenti e supporto per le versioni più recenti di Flutter e Dart.

Flutter offre accesso a una vasta gamma di plugin nativi, che consentono di interfacciarsi con funzionalità specifiche dell'hardware, come Bluetooth, geolocalizzazione, fotocamera e sensori. Questi plugin sono disponibili sia come soluzioni ufficiali che come pacchetti sviluppati dalla comunità, spesso indicati come "community-driven".

Tuttavia, poiché Flutter è una tecnologia relativamente nuova rispetto a framework come Xamarin o React Native, alcune funzionalità avanzate potrebbero richiedere plugin sviluppati appositamente, soprattutto per dispositivi o API meno comuni. Gli sviluppatori hanno comunque la possibilità di creare i propri plugin nativi quando un pacchetto non è disponibile o se una funzionalità richiede personalizzazioni specifiche.

Pub include strumenti per la gestione delle versioni delle dipendenze, aiutando a evitare conflitti e incompatibilità. Gli sviluppatori possono specificare vincoli di versione per assicurarsi che un'app funzioni correttamente anche in caso di aggiornamenti futuri dei pacchetti. Per esempio, nel file `pubspec.yaml`, è possibile impostare una versione minima e massima per ogni pacchetto, garantendo, così, che la versione caricata sia compatibile con il progetto.

Questo sistema di vincoli assicura stabilità e coerenza nel progetto, prevenendo bug e problemi di compatibilità che potrebbero derivare da cambiamenti non testati o da nuove versioni dei pacchetti.

CAPITOLO 2

Analisi dei requisiti e diagramma dei casi d'uso

In questo capitolo definiremo il contesto, al fine di analizzare le informazioni raccolte. Verranno messe in evidenza le parole chiave in un glossario dedicato. Questo passaggio iniziale è essenziale per lo sviluppo dei requisiti del progetto. Inoltre, saranno presentati l'analisi dei requisiti e i diagrammi dei casi d'uso.

2.1 Raccolta delle informazioni

Per lo sviluppo dell'app Android destinata alla gestione informativa della biblioteca universitaria polo Montedago, si è resa necessaria una raccolta accurata di informazioni sui servizi e le funzionalità attualmente offerti. Questa fase è stata condotta attraverso due principali fonti: il sito web ufficiale della biblioteca, che fornisce una descrizione generale delle risorse disponibili, e un'intervista con il direttore della biblioteca. Dal sito, è emerso che la biblioteca offre il servizio di prestito libri e la possibilità di prenotare un posto singolo per lo studio. Invece, durante l'intervista, il direttore ha richiesto l'integrazione di alcune nuove funzionalità, tra cui la possibilità di prenotare un'aula intera per lo studio di gruppo o per la realizzazione di progetti. Questa opzione, non ancora presente sul sito, consentirebbe agli studenti di avere a disposizione uno spazio dedicato, senza disturbare gli altri studenti.

2.2 Descrizione in linguaggio naturale

Il progetto consiste nella realizzazione di un'app Android informativa per la gestione della biblioteca universitaria. In particolare, siamo interessati a certi aspetti della gestione di questa biblioteca, ovvero:

- Gestione dei libri;
- Gestione dei posti.

La biblioteca è aperta per i servizi in presenza dal lunedì al venerdì dalle 8:30 alle 19:15 e il sabato dalle 8:30 alle 13:00. I servizi disponibili sono riservati esclusivamente agli utenti interni, cioè a coloro formalmente legati all'Ateneo a vario titolo. Gli utenti esterni possono accedere alla biblioteca e consultare i documenti solo in sede.

Gli studenti possono visualizzare i titoli di interesse eseguendo una ricerca nel sistema. Un titolo può essere preso in prestito se sono disponibili più copie; in caso contrario, esso è solo consultabile all'interno della biblioteca. Se il titolo richiesto è già in prestito, l'utente

deve aspettare. Solo i libri possono essere presi in prestito, mentre periodici, tesi e libri di pregio sono disponibili per la sola consultazione. Per usufruire del servizio di prestito, l'utente deve prenotare il libro desiderato tramite l'apposita funzione. È possibile prenotare fino a un massimo di 3 libri, con una durata massima della prenotazione di 3 giorni per ciascun libro. Trascorso tale periodo, se i libri non vengono ritirati, le prenotazioni saranno automaticamente annullate. Durante il periodo di validità della prenotazione, l'utente può recarsi fisicamente in biblioteca per effettuare il ritiro.

Una volta ritirati, lo studente può avere al massimo 3 libri in prestito contemporaneamente. Dal giorno del ritiro, ogni libro deve essere restituito entro 15 giorni; in caso contrario, l'utente sarà sospeso dal servizio per un periodo corrispondente al ritardo accumulato.

Lo studente può prenotare un posto o un'aula all'interno della biblioteca, ma non è consentito avere più di una prenotazione attiva contemporaneamente. La prenotazione di un'aula è possibile solo se tutti i posti dell'aula risultano disponibili al momento della richiesta.

Le prenotazioni sono valide esclusivamente durante l'orario di apertura della biblioteca, che va dalle 8:30 alle 19:15. La biblioteca dispone di 80 posti complessivi, distribuiti in 4 aule: Acquario1, Acquario2, Acquario3 e Salone.

La prenotazione, sia di un posto che di un'aula, deve essere confermata recandosi di persona dal bibliotecario entro 30 minuti dall'orario d'inizio della prenotazione; in caso contrario, la prenotazione sarà automaticamente annullata.

L'amministratore, oltre a monitorare e gestire il sistema, è responsabile delle operazioni di creazione, modifica e eliminazione dei dati relativi ai libri presenti nell'inventario.

2.3 Glossario dei termini

Il glossario dei termini contiene tutte le parole di non immediata comprensione alle quali verrà data una spiegazione accurata. Nella Tabella 2.1 viene mostrato il glossario dei termini relativo al nostro contesto di riferimento.

Termine	Descrizione	Sinonimi
Amministratore	Responsabile dell'organizzazione e del funzionamento della biblioteca e delle risorse ad essa assegnate.	Admin
Utenti interni	Le figure a vario titolo formalmente afferenti all'Ateneo.	Nessuno
Utenti esterni	Tutti gli utenti che non appartengono alla categoria degli Utenti interni.	Utenti non istituzionali
Bibliotecario	Persona che aiuta l'amministratore nella gestione della biblioteca, ma che gode di minori privilegi rispetto ad esso	Staff, Operatore
Riservazione	Un periodo dopo il quale la prenotazione va cancellata per la mancanza dello studente o confermata.	Nessuno
Consultare	Esaminare un documento dentro la biblioteca, senza prenderlo in prestito.	Nessuno
Prendere in prestito	Prelevare un libro dalla biblioteca, ma con l'obbligo di restituirlo entro un determinato tempo.	Prestito librario
Restituire	Riportare un libro preso precedentemente in prestito.	Nessuno
Sanzione	Non permettere l'utilizzo di alcuni servizi forniti dall'applicazione come, per esempio, la prenotazione dei libri.	Sospensione

Tabella 2.1: Glossario dei termini relativo alla descrizione del sistema nel linguaggio naturale

2.4 Analisi dei requisiti

I requisiti dell'applicazione sono stati divisi in requisiti funzionali e requisiti non funzionali, come illustrato nella Figura 2.1. I requisiti funzionali descrivono ciò che l'applicazione deve fare e le azioni che deve compiere per soddisfare le esigenze degli utenti. I requisiti non funzionali, invece, non riguardano direttamente le funzionalità dell'applicazione ma rappresentano le necessità che impongono dei vincoli particolari e non propriamente concreti su alcuni funzionamenti del software.

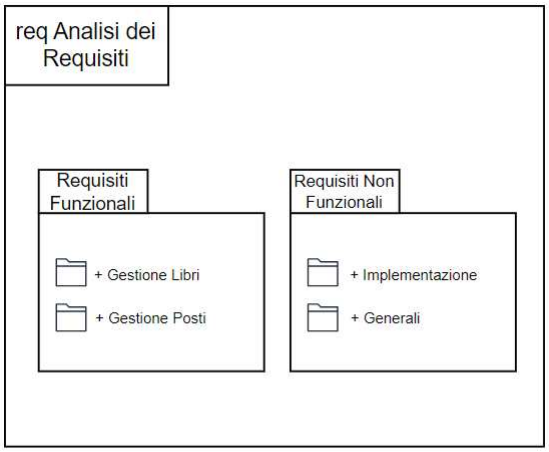

Figura 2.1: Requisiti del sistema e loro classificazione

2.4.1 Requisiti funzionali

Nella Figura 2.2 sono presentati i requisiti funzionali suddivisi nei rispettivi package, seguiti da una breve descrizione di ciascun requisito.

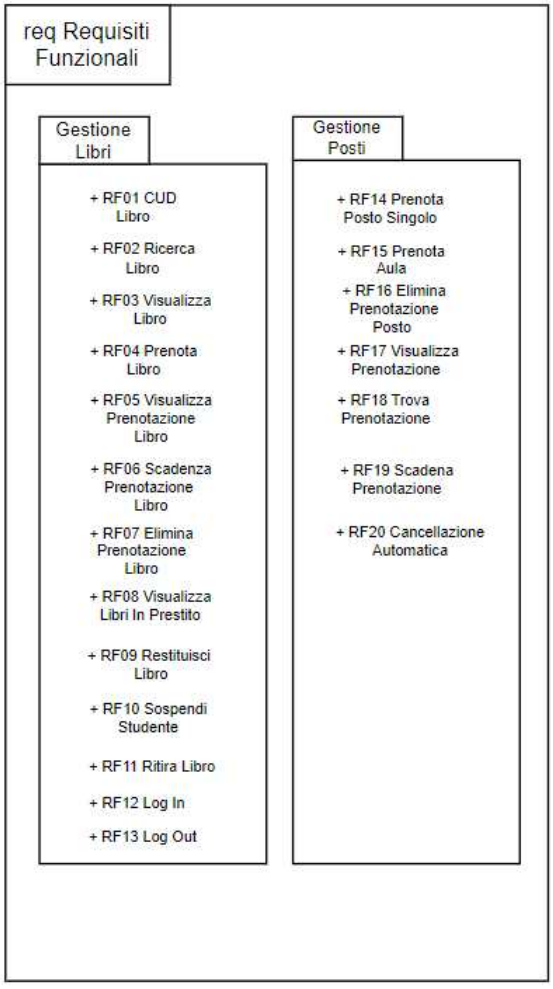

Figura 2.2: Requisiti funzionali e loro classificazione

Requisito Funzionale	**Descrizione**
RF01 - CUD Libro	Il sistema dovrà permettere all'amministratore l'aggiunta, la modifica e la rimozione di un libro.
RF02 - Ricerca Libro	Il sistema dovrà permettere la ricerca di un libro.
RF03 - Visualizza Libro	Il sistema dovrà visualizzare le informazioni relative ad un libro.
RF04 - Prenota Libro	Il sistema dovrà permettere di prenotare un libro.
RF05 - Visualizza Prenotazione Libro	Il sistema dovrà visualizzare le informazioni relative alla prenotazione di un libro.
RF06 - Scadenza Prenotazione Libro	Il sistema dovrà verificare se una determinata prenotazione è già scaduta.
RF07 - Elimina Prenotazione Libro	Il sistema dovrà permettere l'eliminazione di una prenotazione relativa ad un libro.
RF08 - Visualizza Libri In Prestito	Il sistema dovrà visualizzare le informazioni relative al prestito di un libro.
RF09 - Restituisci Libro	Il sistema dovrà permettere allo studente di restituire un libro.
RF10 - Sospendi Studente	Il sistema dovrà permettere di sospendere uno studente in caso di ritardo nella consegna del prestito.
RF11 - Ritira Libro	Il sistema dovrà permettere allo studente di ritirare il libro già prenotato.
RF12 - Log In	Il sistema dovrà gestire l'operazione di accesso all'applicazione.
RF13 - Log Out	Il sistema dovrà gestire l'operazione d'uscita all'applicazione.
RF14 - Prenota Posto Singolo	Il sistema dovrà permettere di prenotare un posto singolo per studiare.
RF15 - Prenota Aula	Il sistema dovrà permettere di prenotare un'aula intera per lo studio in gruppo.
RF16 - Elimina Prenotazione	Il sistema dovrà permettere di eliminare una prenotazione di un posto e/o di un'aula.
RF17 - Visualizza Prenotazione	Il sistema dovrà permettere di visualizzare una prenotazione.
RF18 - Trova Prenotazione	Il sistema dovrà permettere di trovare una prenotazione.
RF19 - Scadena Prenotazione	Il sistema dovrà verificare la scadenza di una prenotazione.
RF20 - Cancellazione Automatica	Il sistema dovrà eliminare automaticamente le prenotazioni scadute.

Tabella 2.2: Requisiti funzionali e loro descrizione

2.4.2 Requisiti non funzionali

Nella Figura 2.3 vengono riportati i requisiti non funzionali, seguiti da una breve descrizione di ciascun requisito.

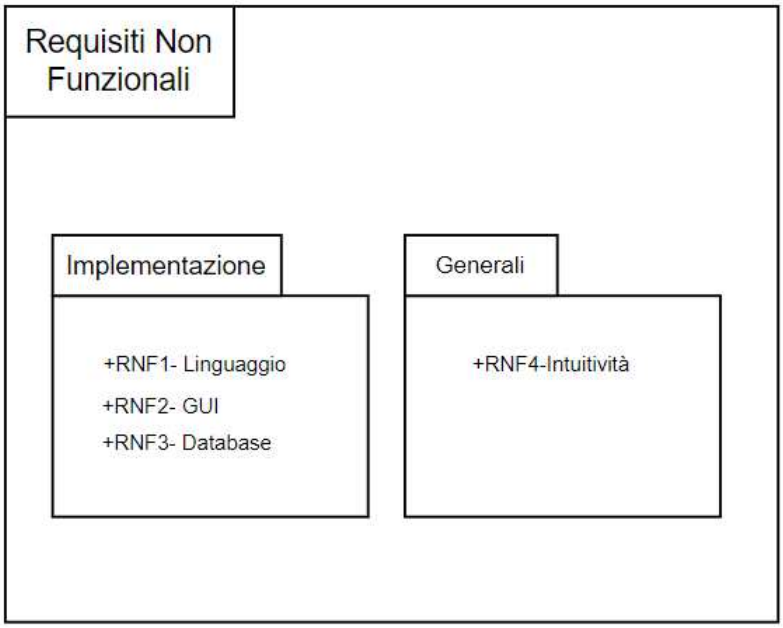

Figura 2.3: Requisiti non funzionali e loro classificazione

Nella Tabella 2.3 vengono riportati i requisiti non funzionali

Requisito Non Funzionale	**Descrizione**
RNF1 - Linguaggio	Il sistema dovrà essere implementato utilizzando Dart con Flutter
RNF2 - GUI	Il sistema dovrà presentare un'interfaccia grafica.
RNF3 - Database	Il sistema dovrà essere dotato di un database per consentire la persistenza dei dati e per gestire l'autenticazione degli utenti.
RNF4 - Intuitività	Il sistema dovrà essere user-friendly e intuitivo per tutti gli utenti.

Tabella 2.3: Requisiti non funzionali e loro descrizione

2.5 Diagramma dei casi d'uso

2.5.1 Gli attori

Nella Figura 2.4 vengono riportati gli attori che interagiscono con il sistema.

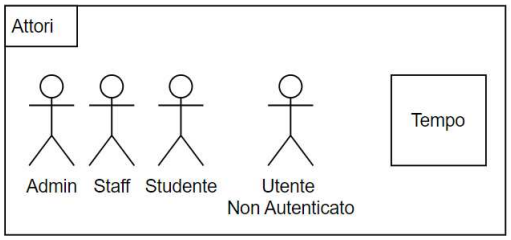

Figura 2.4: Gli attori

2.5.2 Gestione dei libri

Nella Figura 2.5 vengono riportati i casi d'uso riguardanti la gestione dei libri.

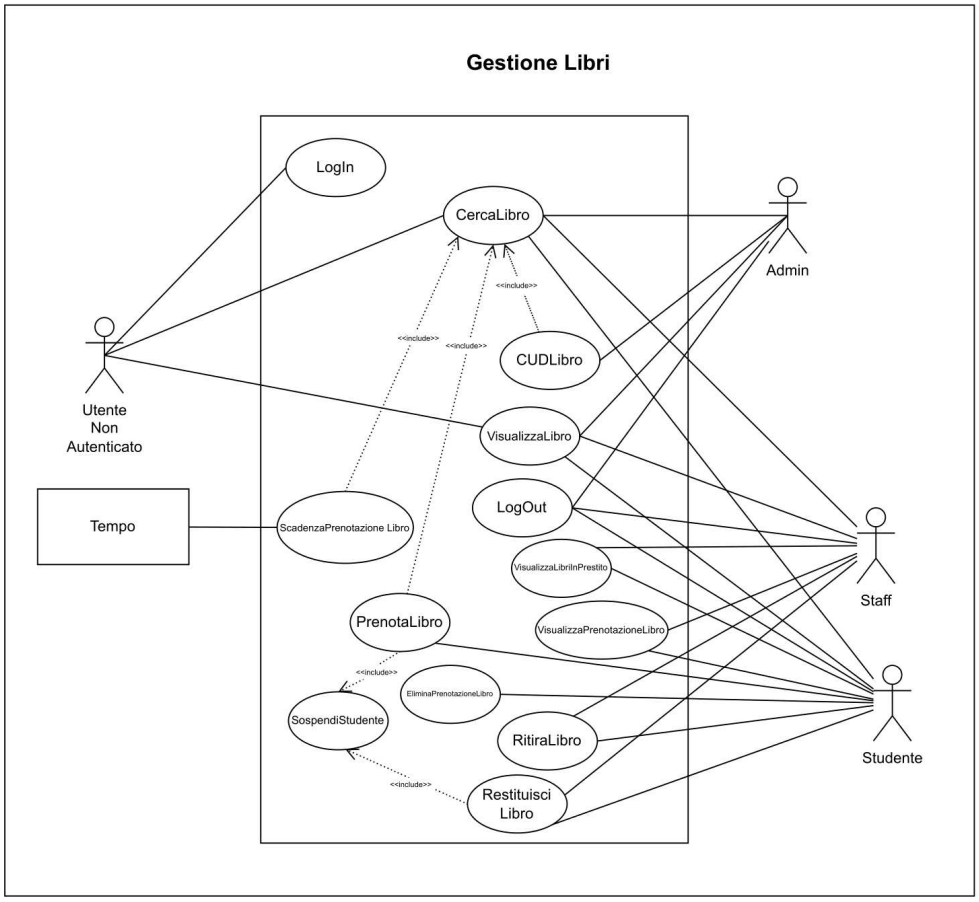

Figura 2.5: Diagramma dei casi d'uso relativo alla gestione dei libri

2.5.3 Gestione dei posti

Nella Figura 2.6 vengono riportati i casi d'uso riguardanti la gestione dei posti.

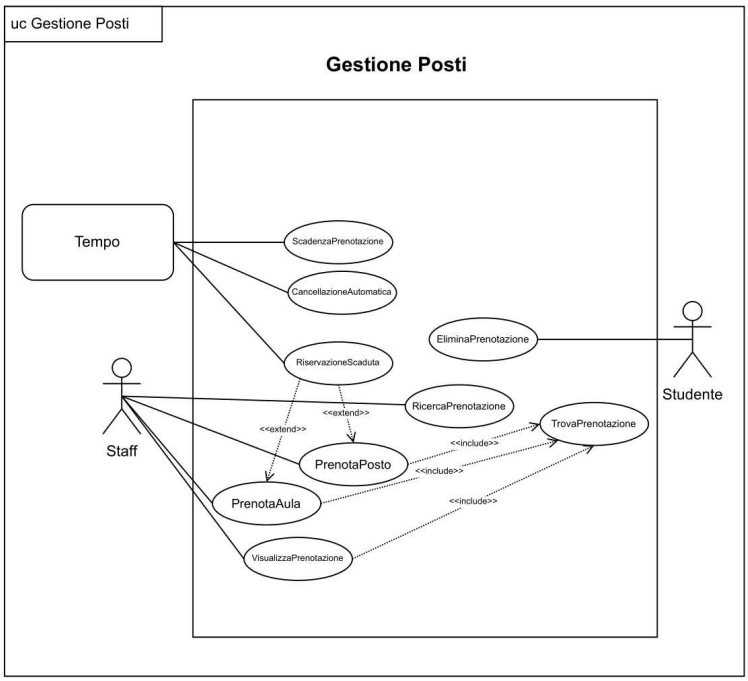

Figura 2.6: Diagramma dei casi d'uso relativo alla gestione dei posti

Nelle Tabelle 2.4 - 2.24 vengono descritti in dettaglio tutti i casi d'uso presenti nei diagrammi precedenti.

Caso d'uso: `CUDLibro`
ID: 01
Breve descrizione: Questo caso d'uso permette di aggiungere, modificare ed eliminare un libro.
Attori primari: Admin
Attori secondari: Nessuno
Precondizioni: 1. L'admin è stato autenticato dal sistema.
Sequenza degli eventi principale: 1. Il caso d'uso inizia quando l'admin vuole effettuare un'operazione di CUD libro. 2. *If* l'admin vuole aggiungere un libro: 2.1. L'admin inserisce i dati del libro richiesti. 2.2. L'admin seleziona Aggiungi Libro. 2.3. *If* il libro è già presente nella lista di libri: 2.3.1. *Include*(RicercaLibro) 2.3.2. Il sistema informa l'admin che il libro è già presente. 2.4. *Else*: 2.4.1. Il sistema aggiunge il nuovo libro alla lista di libri. 3. *If* l'admin vuole modificare i dati di un libro: 3.1. *Include*(RicercaLibro) 3.2. *If* il sistema ha trovato uno o più libri: 3.2.1. L'admin seleziona il libro desiderato dall'elenco dei libri trovati. 3.2.2. L'admin modifica i dati del libro. 3.2.3. L'admin salva le modifiche. 3.2.4. Il sistema aggiorna i dati del libro. 4. *If* l'admin vuole eliminare un libro: 4.1. *Include*(RicercaLibro) 4.2. *If* il sistema ha trovato uno o più libri: 4.2.1. L'admin seleziona il libro desiderato dall'elenco dei libri trovati. 4.2.2. L'admin sceglie di eliminare il libro. 4.2.3. L'admin conferma di voler eliminare il libro. 4.2.4. Il sistema elimina il libro dalla lista dei libri.
Postcondizioni: Il sistema informa l'admin dell'esito dell'operazione.
Sequenza degli eventi alternativa: Nessuna

Tabella 2.4: Descrizione del caso d'uso relativo alle operazioni CUD sui libri.

Caso d'uso: `RicercaLibro`
ID: 02
Breve descrizione: Questo caso d'uso permette di ricercare un libro.
Attori primari: Admin Studente Staff Utente Non Autenticato
Attori secondari: Nessuno
Precondizioni: 1. L'admin è stato autenticato dal sistema.
Sequenza degli eventi principale: 1. Il caso d'uso inizia quando l'attore primario vuole ricercare un libro. 2. L'attore primario inserice alcuni o tutti i parametri di ricerca richiesti, ovvero titolo e/o nome dell'attore. 3. Il sistema ricerca i libri che soddisfano i parametri specificati. 4. *If* il sistema trova uno o più libri: 4.1 Il sistema restituisce l'elenco dei libri trovati. . 5. *Else*: 5.1 Il sistema informa l'attore primario che non ci sono libri che corrispondono ai parametri inseriti.
Postcondizioni: Nessuna
Sequenza degli eventi alternativa: Nessuna

Tabella 2.5: Descrizione del caso d'uso relativo alle operazioni CUD sui libri.

Caso d'uso: `VisualizzaLibro`
ID: 03
Breve descrizione: Questo caso d'uso permette di visualizzare un libro.
Attori primari: Studente Admin Staff Utente Non Autenticato
Attori secondari: Nessuno
Precondizioni: Nessuna
Sequenza degli eventi principale: 1. Il caso d'uso inizia quando l'attore primario seleziona un libro. 2. Il sistema mostra i dettagli del libro.
Postcondizioni: Nessuna
Sequenza degli eventi alternativa: Nessuna

Tabella 2.6: Descrizione del caso d'uso relativo alla visualizzazione di un libro.

Caso d'uso: `PrenotaLibro`
ID: 04
Breve descrizione: Questo caso d'uso permette di prenotare un libro.
Attori primari: Studente
Attori secondari: Nessuno
Precondizioni: 1. Lo studente è stato autenticato dal sistema.
Sequenza degli eventi principale: 1. Il caso d'uso inizia quando lo studente vuole prenotare un libro. 2. *Include* (RicercaLibro) 3. *If* il sistema ha trovato uno o più libri: 3.1. Lo studente seleziona il libro desiderato dall'elenco dei libri trovati. 3.2. Il sistema visualizza i dettagli del libro. 3.3. Lo studente seleziona la data di prenotazione. 3.4. Il sistema verifica le seguenti condizioni: 3.4.1. *If* lo studente è sospeso: 3.4.1.1. *Include* (SospendiStudente). 3.4.1.2. Il sistema informa lo studente che non può effettuare prenotazioni. 3.4.2. *If* lo studente ha già prenotato lo stesso libro: 3.4.2.1. Il sistema informa lo studente che la prenotazione non è consentita. 3.4.3. *If* lo studente ha raggiunto il limite massimo di 3 prenotazioni attive: 3.4.3.1. Il sistema informa lo studente che non può effettuare altre prenotazioni. 3.4.4. *If* il libro non è disponibile 3.4.4.1. Il sistema informa lo studente che il libro non è prenotabile. 3.5. *Else*: 3.5.1. Il sistema chiede la conferma della prenotazione. 3.5.2. Il sistema registra la prenotazione e riduce la quantità del libro 1. 4. *Else*: 4.1. Il sistema informa lo studente che non ci sono libri che corrispondono ai parametri inseriti.
Postcondizioni: Lo studente ha completato la prenotazione del libro o è stato informato delle condizioni che ne impediscono l'esecuzione.
Sequenza degli eventi alternativa: Nessuna

Tabella 2.7: Descrizione del caso d'uso relativo alla Prenotazione di un libro.

Caso d'uso: `VisualizzaPrenotazioneLibro`
ID: 05
Breve descrizione: Questo caso d'uso permette di visualizzare i libri prenotati.
Attori primari: Staff Studente
Attori secondari: Nessuno
Precondizioni: 1. L'utente è autenticato dal sistema.
Sequenza degli eventi principale: 1. Il caso d'uso inizia quando l'utente vuole visualizzare l'elenco dei libri prenotati. 2. Il sistema verifica il ruolo dell'utente: 2.1 *If* l'utente è Staff 2.1.1. Il sistema recupera tutte le prenotazioni registrate. 2.1.2. Il sistema visualizza l'elenco completo delle prenotazioni. 2.2 *If* l'utente è Studente 2.2.1. Il sistema recupera le prenotazioni associate allo studente autenticato. 2.2.2. Il sistema visualizza l'elenco delle prenotazioni personali.
Postcondizioni: L'utente visualizza correttamente l'elenco delle prenotazioni secondo il proprio ruolo.
Sequenza degli eventi alternativa: Nessuna

Tabella 2.8: Descrizione del caso d'uso relativo alla visualizzazione dei libri prenotati.

Caso d'uso: `ScadenzaPrenotazioneLibro`
ID: 06
Breve descrizione: Questo caso d'uso permette di verificare la scadenza delle prenotazioni dei libri.
Attori primari: Tempo
Attori secondari: Nessuno
Precondizioni: Nessuna
Sequenza degli eventi principale: 1. Il caso d'uso inizia ogni giorno dopo le ore 00:00. 2. *For* prenotazione: 2.1. *If* la data attuale è posteriore rispetto alla data massima per ritirare il libro 2.1.1. Il sistema elimina la prenotazione dal database. 2.1.2. *Include* (CercaLibro) 2.1.3. Il sistema aggiorna la quantità disponibile del libro associato, incrementandola di 1.
Postcondizioni: Nessuna.
Sequenza degli eventi alternativa: Nessuna

Tabella 2.9: Descrizione del caso d'uso relativo alla verifica della scadenza Prenotazione Libro.

Caso d'uso: `EliminaPrenotazioneLibro`
ID: 07
Breve descrizione: Questo caso d'uso permette di eliminare una prenotazione di un libro.
Attori primari: Studente
Attori secondari: Nessuno
Precondizioni: 1.Lo studente è autenticato dal sistema. 2.Lo studente ha già almeno una prenotazione attiva.
Sequenza degli eventi principale: 1. Il caso d'uso inizia quando lo studente decide di eliminare una prenotazione di un libro. 2. Il sistema chiede la conferma dell'eliminazione della prenotazione. 3. Il sistema elimina la prenotazione e aumenta la quantità del libro di 1.
Postcondizioni: 1.La prenotazione del libro è stata eliminata.
Sequenza degli eventi alternativa: Nessuna

Tabella 2.10: Descrizione del caso d'uso relativo all'eliminazione di una prenotazione di un libro.

Caso d'uso: `VisualizzaLibriInPrestito`
ID: 08
Breve descrizione: Questo caso d'uso permette di visualizzare l'elenco dei libri presi in prestito.
Attori primari: Studente Staff
Attori secondari: Nessuno
Precondizioni: 1. L'utente è autenticato dal sistema.
Sequenza degli eventi principale: 1. Il caso d'uso inizia quando l'utente vuole visualizzare l'elenco dei libri in prestito. 2. Il sistema controlla il ruolo dell'utente 2.1. *If* l'utente è uno Studente 2.1.1. Il sistema recupera l'elenco dei libri in prestito associati allo studente. 2.2. *If* l'utente è Staff 2.2.1. Il sistema recupera l'elenco di tutte le prenotazioni associate a tutti gli utenti.
Postcondizioni: L'utente visualizza correttamente l'elenco dei libri in prestito. Se necessario, è stato informato della sospensione e della durata della stessa.
Sequenza degli eventi alternativa: Nessuna

Tabella 2.11: Descrizione del caso d'uso relativo alla visualizzazione dei libri in prestito.

Caso d'uso: `RestituisciLibro`
ID: 09
Breve descrizione: Questo caso d'uso permette la restituzione di un libro .
Attori primari: Studente Staff
Attori secondari: Nessuno
Precondizioni: 1. L'utente è autenticato dal sistema. 2. L'utente ha un libro in prestito.
Sequenza degli eventi principale: 1. Il caso d'uso inizia quando lo studente restituisce fisicamente un libro alla biblioteca. 2. Lo staff segna la restituzione nel sistema. 3. Il sistema aggiorna lo stato della prenotazione a "consegnato". 4. Il sistema aggiorna la quantità del libro. 5. *If* il libro è stato restituito oltre la data di scadenza 5.1 *Include* (SospendiStudente) 5.2 Lo staff informa lo studente che è sospeso per il ritardo nella consegna.
Postcondizioni: Lo studente restituisce il libro.
Sequenza degli eventi alternativa: Nessuna

Tabella 2.12: Descrizione del caso d'uso relativo alla restituzione di un libro.

Caso d'uso: `SospendiStudente`
ID: 10
Breve descrizione: Questo caso d'uso permette di sospendere lo studente.
Attori primari: Tempo
Attori secondari: Studente
Precondizioni: 1. L'utente è autenticato dal sistema. 2. L'utente ha uno o più libri in prestito.
Sequenza degli eventi principale: 1. Il caso d'uso inizia ogni giorno dopo le ore 00:00. 2. *for each* prenotazione 2.1. *If* la data attuale è posteriore rispetto alla data massima per consegnare il libro 2.1.1. l sistema determina la durata della sospensione (pari ai giorni di ritardo).
Postcondizioni: Il sistema registra la durata della sospensione.
Sequenza degli eventi alternativa: Nessuna

Tabella 2.13: Descrizione del caso d'uso relativo alla sospensione di uno studente.

Caso d'uso: `RitiraLibro`
ID: 11
Breve descrizione: Questo caso d'uso permette di ritirare di un libro fisicamente dalla biblioteca.
Attori primari: Studente Staff
Attori secondari: Nessuno
Precondizioni: 1. L'utente è autenticato dal sistema. 2. L'utente ha un libro prenotato.
Sequenza degli eventi principale: 1. Il caso d'uso inizia quando lo studente ritira fisicamente un libro dalla biblioteca. 2. Lo staff segna che il libro è stato ritirato nel sistema. 3. Il sistema aggiorna lo stato della prenotazione a "ritirato". 4. Il sistema calcola la data per restituire il libro.
Postcondizioni: Lo studente ritira il libro.
Sequenza degli eventi alternativa: Nessuna

Tabella 2.14: Descrizione del caso d'uso per ritirare un libro.

Caso d'uso: `LogIn`
ID: 12
Breve descrizione: Questo caso d'uso permette agli utenti di accedere all'applicazione.
Attori primari: Utente Non Autenticato
Attori secondari: Nessuno
Precondizioni: 1. L'utente è registrato nella database.
Sequenza degli eventi principale: 1. Il caso d'uso inizia quando l'attore primario accede alla pagina di login. 2. L'attore primario inserisce l'email e la password nei rispettivi campi. 3. Il sistema valida le credenziali inserite. 4. *If* le credenziali sono corrette: 4.1 L'attore primario viene autenticato. 4.2 Il sistema reindirizza l'utente alla home page. 5. *Else*: 5.1 Il sistema informa l'utente che le credenziali non sono valide.
Postcondizioni: L'utente è autenticato oppure viene informato che le credenziali sono errate.
Sequenza degli eventi alternativa: Nessuna

Tabella 2.15: Descrizione del caso d'uso per il login.

Caso d'uso: `LogOut`
ID: 13
Breve descrizione: Questo caso d'uso permette agli utenti di uscire all'applicazione.
Attori primari: Admin Staff Studente
Attori secondari: Nessuno
Precondizioni: 1. L'utente ha effettuato l'accesso.
Sequenza degli eventi principale: 1. Il caso d'uso inizia quando l'utente clicca sul pulsante "Logout". 2. Il sistema interrompe la sessione attiva. 3. Il sistema reindirizza l'utente alla pagina di login.
Postcondizioni: L'utente è autenticato oppure viene informato che le credenziali sono errate.
Sequenza degli eventi alternativa: Nessuna

Tabella 2.16: Descrizione del caso d'uso per il login.

Caso d'uso: `PrenotaPosto`
ID: 14
Breve descrizione: Questo caso d'uso permette di effettuare la prenotazione di un posto singolo per lo studio.
Attori primari: Studente Staff
Attori secondari: Nessuno
Precondizioni: 1. L'utente è autenticato dal sistema.
Sequenza degli eventi principale: 1. Il caso d'uso inizia quando lo studente vuole prenotare un posto singolo. 2. Lo studente primario specifica la data e la durata della prenotazione. 3. *If* i dati inseriti non sono corretti: 3.1 Il sistema restituisce un opportuno messaggio di errore. 4. *Include* (TrovaPrenotazione). 5. *If* lo studente ha già una prenotazione nella fascia oraria indicata: 5.1 Il sistema mostra un messaggio di errore avvisando lo studente che ha già una prenotazione. 6. Il sistema mostra i posti disponibili. 7. Lo studente primario sceglie il posto desiderato. 8. Il sistema procede con la prenotazione generando un ID per la prenotazione. 9. Lo studente si presenta entro 30 minuti dalla data d'inizio della sua prenotazione. punto di estensione: *riservazione scaduta* 10. Lo staff conferma la presenza dello studente attivando la prenotazione.
Postcondizioni: Nessuna.
Sequenza degli eventi alternativa: Nessuna

Tabella 2.17: Descrizione del caso d'uso relativo alla prenotazione di un posto singolo.

Caso d'uso: `PrenotaAula`
ID: 15
Breve descrizione: Questo caso d'uso permette di effettuare la prenotazione di un'intera aula.
Attori primari: Studente Staff
Attori secondari: Nessuno
Precondizioni: 1. L'utente è autenticato dal sistema.
Sequenza degli eventi principale: 1. Il caso d'uso inizia quando lo studente vuole prenotare un'aula. 2. Lo studente specifica la data e la durata della prenotazione. 3. *If* i dati inseriti non sono corretti: 3.1 Il sistema restituisce un opportuno messaggio di errore. 4. *Include* (TrovaPrenotazione). 5. *If* lo studente ha già una prenotazione nella fascia oraria indicata: 5.1 Il sistema mostra un messaggio di errore avvisando lo studente che ha già una prenotazione. 6. Il sistema mostra le aule disponibili. 7. Lo studente primario sceglie l'aula desiderata. 8. Il sistema procede con la prenotazione generando un ID per la prenotazione. 9. Lo studente si presenta entro 30 minuti dalla data d'inizio della sua prenotazione. punto di estensione: *riservazione scaduta* 10. Lo staff conferma la presenza dello studente attivando la prenotazione.
Postcondizioni: Nessuna
Sequenza degli eventi alternativa: Nessuna

Tabella 2.18: Descrizione del caso d'uso relativo alla prenotazione di un'aula.

Caso d'uso di estensione: `RiservazioneScaduta`
ID: 16
Breve descrizione: La prenotazione viene riservata per 30 minuti e dopo viene cancellata.
Attori primari: Tempo
Attori secondari: Nessuno
Precondizioni del segmento1: 1.Lo studente è arrivato in tempo per la prenotazione.
Sequenza degli eventi principale: 1. Il caso d'uso inizia quando lo studente non arriva alla biblioteca entro 30 minuti dal momento dell'inizio della sua prenotazione. 2. Il sistema elimina la prenotazione.
Postcondizioni del segmento1: 1.La prenotazione è stata eliminata.
Sequenza degli eventi alternativa: Nessuna

Tabella 2.19: Descrizione del caso d'uso relativo alla riservazione di una prenotazione di un posto.

Caso d'uso: `EliminaPrenotazionePosto`
ID: 17
Breve descrizione: Questo caso d'uso permette di eliminare una prenotazione di un posto e/o di un'aula.
Attori primari: Studente
Attori secondari: Nessuno
Precondizioni: 1.Lo studente è autenticato dal sistema. 2.Lo studente ha già almeno una prenotazione attiva.
Sequenza degli eventi principale: 1. Il caso d'uso inizia quando lo studente decide di eliminare una prenotazione di un posto e/o di un'aula. 2. Il sistema chiede la conferma dell'eliminazione della prenotazione. 3. Il sistema elimina la prenotazione e mette il posto e/o l'aula come disponibile.
Postcondizioni: 1.La prenotazione è stata eliminata.
Sequenza degli eventi alternativa: Nessuna

Tabella 2.20: Descrizione del caso d'uso relativo all'eliminazione di una prenotazione di un posto.

Caso d'uso: `VisualizzaPrenotazionePosto`
ID: 18
Breve descrizione: Questo caso d'uso permette di visualizzare le informazioni relative ad una prenotazione.
Attori primari: Studente Staff
Attori secondari: Nessuno
Precondizioni: 1.L'attore primario è autenticato dal sistema.
Sequenza degli eventi principale: 1. Il caso d'uso inizia quando l'attore primario vuole visualizzare i dati relativi ad una prenotazione. 2. *Include* (TrovaPrenotazione) 2.1. Il sistema visualizza i dettagli della prenotazione.
Postcondizioni: Nessuna
Sequenza degli eventi alternativa: Nessuna

Tabella 2.21: Descrizione del caso d'uso relativo alla visualizzaione di una prenotazione di un posto.

Caso d'uso: `TrovaPrenotazione`
ID: 19
Breve descrizione: Questo caso d'uso permette di trovare una prenotazione.
Attori primari: Nessuno
Attori secondari: Staff Studente
Precondizioni: Nessuna
Sequenza degli eventi principale: 1. Il caso d'uso inizia quando l'attore primario vuole trovare una prenotazione. 2. *If* il sistema ha trovato la prenotazione 2.1. Il sistema visualizza i dettagli della prenotazione.
Postcondizioni: Nessuna
Sequenza degli eventi alternativa: Nessuna

Tabella 2.22: Descrizione del caso d'uso relativo alla ricerca di una prenotazione

Caso d'uso: `ScadenzaPrenotazione`
ID: 20
Breve descrizione: Questo caso d'uso permette di eliminare tutte le prenotazioni scadute.
Attori primari: Tempo
Attori secondari: Nessuno
Precondizioni: Nessuna.
Sequenza degli eventi principale: 1. Il caso d'uso inizia quando viene effettuata una prenotazione. 2. *for each* prenotazione 2.1. Il sistema controlla la data e l'ora attuali. 2.2. *If* La data e l'ora attuali sono posteriori 2.2.1. Il sistema cancella la prenotazione. 2.2.2. Il sistema mette il posto o l'aula come disponibili.
Postcondizioni: 1.La prenotazione del libro è stata eliminata.
Sequenza degli eventi alternativa: Nessuna

Tabella 2.23: Descrizione del caso d'uso relativo alla scadenza di una prenotazione.

Caso d'uso: `CancellazioneAutomatica`
ID: 21
Breve descrizione: Questo caso d'uso permette di cancellare automaticamente tutte le prenotazioni per la mancanza dello studente.
Attori primari: Tempo
Attori secondari: Nessuno
Precondizioni: 1. Lo studente ha effettuato una prenotazione di un posto.
Sequenza degli eventi principale: 1. Il caso d'uso inizia dopo 30 minuti dall'inizio della prenotazione. 2. Lo studente non si è presentato in tempo. 3. Il sistema cancella la prenotazione.
Postcondizioni: 1.La prenotazione del libro è stata eliminata.
Sequenza degli eventi alternativa: Nessuna

Tabella 2.24: Descrizione del caso d'uso relativo alla cancellazione automatica di una prenotazione.

CAPITOLO 3

Progettazione del database

In questo capitolo definiremo la progettazione del database che rappresenta una fase cruciale nello sviluppo di un sistema informatico, poiché definisce la struttura organizzativa dei dati e ne garantisce l'efficienza e l'affidabilità.

3.1 Progettazione concettuale

Per il progetto della nostra applicazione mobile, dedicata alla gestione della biblioteca universitaria, il database è stato progettato per supportare in modo efficace le principali funzionalità dell'applicazione, ovvero la gestione dei libri e la prenotazione dei posti. La progettazione del database ha seguito un approccio strutturato e modulare, che integra più tecnologie per rispondere alle diverse esigenze del sistema.

Il processo di progettazione è stato condotto utilizzando una strategia *top-down*, che ha permesso di partire da uno schema generale e di perfezionarlo progressivamente per includere tutti i dettagli fondamentali. Questa metodologia ha consentito di garantire un modello solido e funzionale, capace di rispondere ai requisiti del progetto in modo scalabile e sicuro.

3.1.1 Identificazione delle entità principali

Per progettare il database della biblioteca, è stato necessario individuare le entità fondamentali che rappresentano le aree chiave del sistema. Le entità fondamentali sono:

- Libro
- Utente
- Prenotazione
- Posto

Con l'entità "Libro" viene rappresentata ogni volume disponibile nella biblioteca, comprensivo delle sue caratteristiche principali, come titolo, autore, categoria, e quantità disponibile. I libri costituiscono il nucleo centrale dei servizi della biblioteca. Gli studenti accedono al sistema principalmente per cercare, prenotare o ottenere informazioni sui libri disponibili. Senza questa entità, non sarebbe possibile gestire né il catalogo né il sistema di prestiti.

Invece, l'entità "Utente" identifica ogni persona che interagisce con l'applicazione, sia essa uno studente, un membro dello staff o un amministratore. Questa entità consente di distinguere le diverse tipologie di utenti adattando le funzionalità del sistema alle loro esigenze. Inoltre, è essenziale per la gestione di autenticazione e autorizzazioni, nonché per tracciare le attività legate ai prestiti e alle prenotazioni.

L'entità "Prenotazione" rappresenta l'atto di riservare un libro o un posto in biblioteca. Ogni prenotazione include informazioni sullo studente che l'ha effettuata, l'oggetto della prenotazione (libro o posto), e le date di inizio e fine. La prenotazione è una delle funzionalità principali offerte dall'applicazione. Attraverso questa entità, il sistema consente di tracciare quali studenti stanno utilizzando (o intendono utilizzare) specifici libri o posti, migliorando la gestione delle risorse e riducendo il rischio di conflitti o doppie assegnazioni.

L'entità "Posto" rappresenta le postazioni di studio disponibili in biblioteca, siano esse singole sedute o stanze per lavori di gruppo. Ogni posto è caratterizzato da un identificativo e da attributi come la tipologia e la disponibilità. La gestione dei posti di studio è una funzionalità richiesta per migliorare l'esperienza degli studenti e garantire un utilizzo ottimale delle risorse. L'entità "Posto" consente di monitorare la disponibilità delle sedute, permettendo agli studenti di riservare un'area per lo studio personale o per il lavoro di gruppo.

3.1.2 Schema E-R

Nel passaggio al modello logico, abbiamo semplificato la gerarchia dell'entità "Utente" (Figura 3.1), in particolare, abbiamo deciso di accorpare l'entità "Utente" nelle tre figlie "Admin," "Staff" e "Studente," utilizzando un unico attributo discriminante, ovvero Email.

Il sistema identifica il ruolo di ciascun utente analizzando il dominio dell'email. In particolare:

- gli utenti con un'email contenente "@admin" vengono classificati come amministratori;
- gli utenti con un'email contenente "@studente" sono riconosciuti come studenti;
- gli utenti con un'email contenente "@staff" vengono classificati come bibliotecari.

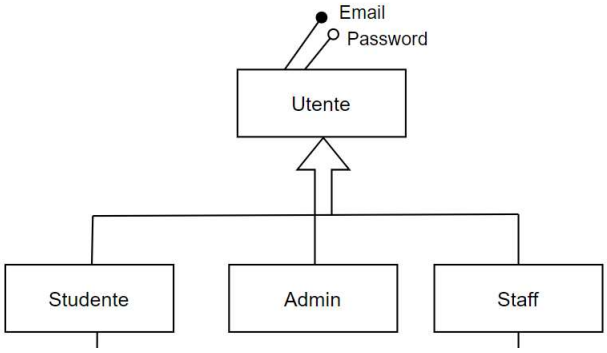

Figura 3.1: Entità "Utente" con i figli

Nella Figura 3.2 viene rappresentato lo schema E-R finale.

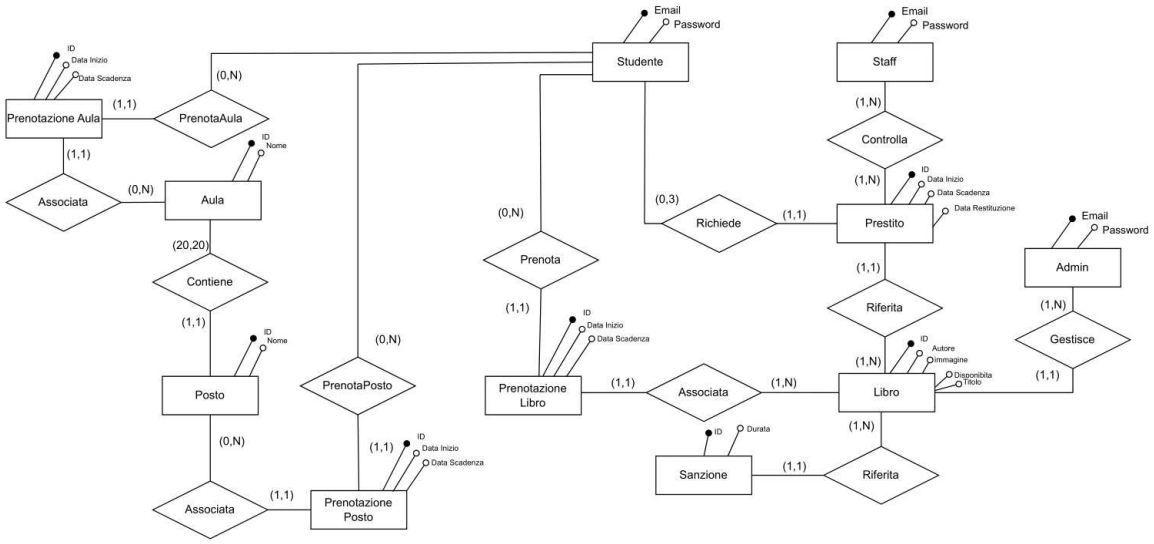

Figura 3.2: Schema E-R finale

3.2 Struttura del database

Il database del progetto è stato progettato seguendo un approccio basato su Firebase Real-time Database, una soluzione scalabile e altamente performante che supporta applicazioni in tempo reale. La struttura del database è organizzata in nodi gerarchici, ciascuno dei quali rappresenta una specifica categoria di informazioni. Questa organizzazione consente un accesso rapido ai dati e una gestione semplice delle relazioni tra diverse entità, pur mantenendo una struttura flessibile.

3.2.1 Relazioni tra nodi

Nella Figura 3.3 è rappresentata la struttura gerarchica del database, composta da nodi principali che corrispondono alle diverse entità fondamentali per il funzionamento dell'applicazione.

Il nodo *Aule* è dedicato alla gestione degli spazi disponibili per lo studio e il lavoro di gruppo. Ogni aula è rappresentata come un sotto-nodo e include venti posti associati. Ciascun posto è identificato da un attributo *ID* univoco e da un attributo *disponibilità*, che indica se il posto è libero o occupato.

Il nodo *Libri* memorizza i dettagli di ogni libro disponibile nella biblioteca. Ogni libro è identificato tramite l'attributo *ID*, che corrisponde al codice ISBN e funge da identificativo univoco. L'attributo *Image* contiene il link all'immagine del libro, salvato su ImgBB. Inoltre, il nodo include l'attributo *Quantità*, che indica il numero di copie disponibili per ogni titolo.

Il nodo *Prenotazioni* contiene le informazioni relative alle prenotazioni dei posti nelle aule. Ogni prenotazione include gli attributi *DataInizio* e *DataFine*, che indicano, rispettivamente, la data di inizio e di fine della prenotazione. L'attributo *TipoPrenotazione* specifica se la prenotazione riguarda un singolo posto o un'intera aula.

Il nodo *PrenotazioniLibro* gestisce le prenotazioni dei libri effettuate dagli utenti. Gli attributi principali includono *Libro*, che identifica il libro prenotato, e *Studente*, che contiene l'identificativo dello studente che ha effettuato la prenotazione. Gli attributi *DataInizio* e *DataFine* specificano il periodo del prestito. Inoltre, l'attributo *Ritirato* indica se il libro è stato effettivamente ritirato. Se il ritiro avviene, il sistema calcola un periodo di quindici giorni per la riconsegna, trascorso il quale lo studente potrebbe essere soggetto a sanzioni.

Infine, il nodo *Sanzione* è responsabile della gestione delle sanzioni disciplinari imposte agli studenti. L'attributo *Studente* identifica lo studente sanzionato, mentre l'attributo *Durata* indica la durata della sanzione in giorni.

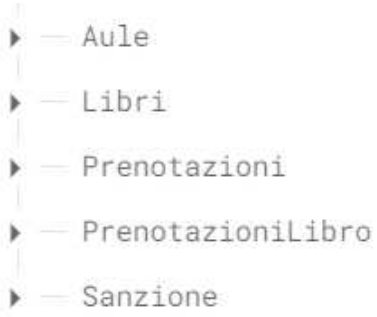

Figura 3.3: Struttura gerarchica del database firebase

3.3 Integrazione con ImgBB

Nel progetto della nostra applicazione mobile per la gestione della biblioteca universitaria, uno degli aspetti fondamentali è stato il caricamento e la gestione delle immagini associate ai libri. Per semplificare questo processo e garantire una soluzione scalabile, è stato scelto il servizio ImgBB, una piattaforma dedicata all'hosting di immagini che fornisce API intuitive per il caricamento e il recupero delle immagini.

3.3.1 Caricamento e gestione delle immagini

Quando un amministratore aggiunge o aggiorna un libro nell'applicazione, l'immagine associata viene caricata su ImgBB attraverso l'API di caricamento. Il file immagine viene inviato come payload della richiesta HTTP POST.

ImgBB restituisce un URL univoco che punta all'immagine caricata. Questo URL viene salvato nel database Firebase Realtime Database o Firestore, associato al record del libro.

le immagini dei libri vengono recuperate dinamicamente tramite gli URL salvati, consentendo una visualizzazione fluida e immediata agli utenti dell'app.

3.4 Gestione dell'autenticazione

Per garantire un accesso sicuro e personalizzato all'applicazione, è stato implementato un sistema di autenticazione basato su Firebase Authentication. Questo servizio offerto da Firebase consente di autenticare gli utenti in modo semplice, sicuro e scalabile, supportando diversi provider di autenticazione, tra cui email e password, social login e altro ancora. Nel nostro progetto, è stata utilizzata la libreria `firebase-auth` per integrare questa funzionalità. Il sistema di autenticazione è stato configurato per supportare tre tipologie di utenti principali:

- *Admin*: responsabili della gestione della biblioteca, con la possibilità di eseguire operazioni di creazione, aggiornamento e cancellazione (CUD) su libri e prenotazioni.
- *Staff*: utenti autorizzati a monitorare e gestire prenotazioni e altre operazioni operative.
- *Studenti*: utenti principali dell'app, che possono prenotare libri e posti studio.

Ogni utente viene identificato attraverso un indirizzo email univoco e associato a un ruolo specifico. La gestione dei ruoli viene eseguita direttamente nella console Firebase Authentication, dove è possibile verificare i dettagli di ciascun utente, come mostrato nella Figura 3.4

Figura 3.4: Utenti salvati in firebase

CAPITOLO 4

Progettazione della componente applicativa

In questo capitolo definiremo la progettazione della componente applicativa, includendo la definizione dell'interfaccia utente tramite mockup, nonché la progettazione della logica di sistema attraverso l'integrazione con Firebase e IMGBB.

4.1 Progettazione dell'interfaccia utente

La progettazione dell'interfaccia utente si concentra sulla creazione di un'esperienza visiva intuitiva e funzionale, che consenta agli utenti di interagire facilmente con l'applicazione. In questa fase vengono definiti i layout, le interazioni e i flussi di navigazione, assicurando che l'interfaccia sia coerente, accessibile e risponda alle esigenze specifiche degli utenti finali.

4.1.1 Mockup dell'interfaccia

I mockup sono modelli visivi utilizzati per illustrare la disposizione degli elementi dell'interfaccia utente e rappresentare nel dettaglio i vari contenuti e le funzionalità di base dell'applicazione. Questi modelli forniscono al cliente un'anteprima tangibile e valutabile prima dell'avvio dello sviluppo vero e proprio.

La Figura 4.1 illustra la home page dedicata agli utenti guest, ovvero coloro che non hanno effettuato l'accesso. In questa schermata è possibile visualizzare l'elenco dei libri disponibili e accedere all'icona per il login. La Figura 4.2 presenta la home page riservata all'amministratore, caratterizzata dalla presenza di un pulsante per l'aggiunta di nuovi libri al sistema. Infine, la Figura 4.3 mostra la home page destinata agli utenti autenticati, sia studenti che membri dello staff, che include una barra di navigazione (navbar) nella parte inferiore dello schermo e l'icona per effettuare il logout. In tutte le versioni della home page è presente, nella parte superiore, un motore di ricerca che consente di filtrare i libri in base al titolo o all'autore, facilitando l'accesso rapido alle informazioni desiderate.

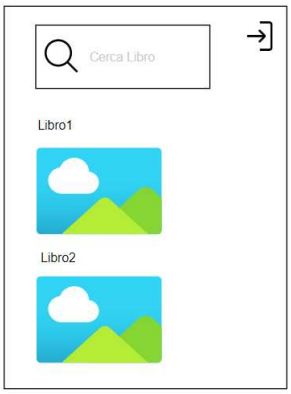

Figura 4.1: Guest

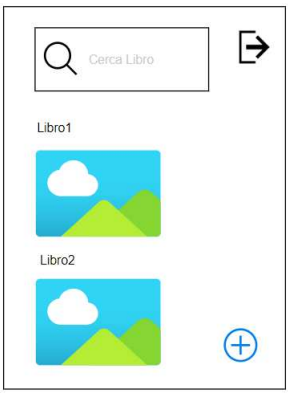

Figura 4.2: Admin

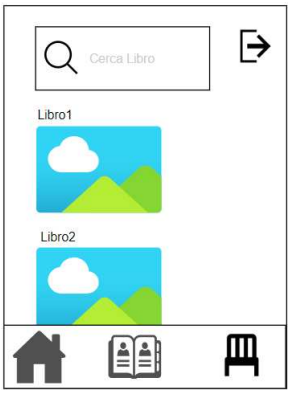

Figura 4.3: Studente

La Figura 4.4 mostra la pagina di dettaglio di un libro dedicata agli utenti guest, nella quale viene visualizzata esclusivamente la descrizione del libro. La Figura 4.5 illustra, invece, la pagina di dettaglio destinata agli studenti, arricchita dalla presenza di un pulsante per effettuare la prenotazione del libro, qualora disponibile. Tuttavia, nel caso in cui sia rimasta un'unica copia del libro, il pulsante per la prenotazione non è visibile; al suo posto, lo studente visualizza un messaggio che informa della temporanea indisponibilità del libro. Infine, la Figura 4.6 rappresenta la pagina di un libro visualizzata da uno studente sospeso, con le relative informazioni.

Figura 4.4: Guest

Figura 4.5: Studente

Figura 4.6: Sospeso

Nella Figura 4.7 è illustrata la pagina delle prenotazioni dedicata agli studenti, in cui viene visualizzato l'elenco delle prenotazioni effettuate. In questa sezione è possibile eliminare una prenotazione, a condizione che il libro non sia stato ancora ritirato presso la biblioteca. La Figura 5.8 mostra la pagina relativa ai libri in prestito, con l'elenco delle prenotazioni e l'indicazione della data limite per la restituzione del libro, oltre la quale lo studente potrebbe essere sospeso. Infine, la Figura 4.9 rappresenta la pagina delle prenotazioni destinata al personale della biblioteca, dove è possibile distinguere lo stato dei libri tra "prenotati" e "in prestito" e gestire il cambio di stato al momento del ritiro o della restituzione fisica dei libri da parte degli studenti.

Figura 4.7: prenotazioni studente

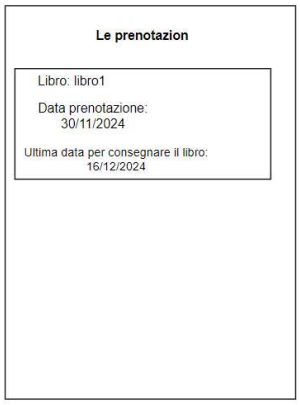

Figura 4.8: prenotazioni staff

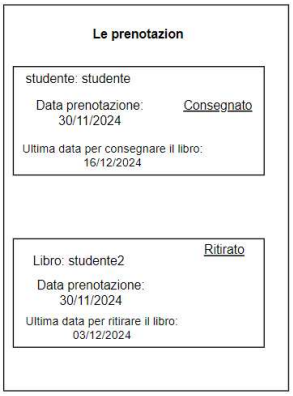

Figura 4.9: prenotazioni staff2

La logica adottata per la prenotazione di posti e aule segue lo stesso principio utilizzato per la prenotazione dei libri. Lo studente, infatti, seleziona il posto o l'aula desiderati, con la differenza che, in questo caso, viene associato un identificativo specifico al posto o all'aula. Il bibliotecario ha la facoltà di confermare la prenotazione al momento dell'arrivo dello studente, a condizione che quest'ultimo si presenti entro 30 minuti dall'orario di inizio della prenotazione. In caso contrario, la prenotazione viene automaticamente annullata dall'applicazione, garantendo, così, l'ottimizzazione della gestione degli spazi.

4.2 Progettazione della logica di sistema

La progettazione della logica di sistema di un'applicazione comprende la definizione e l'implementazione delle componenti principali che ne determinano il comportamento. Una parte cruciale di questa progettazione riguarda l'interazione con i servizi esterni che consentono all'app di funzionare in modo dinamico e sicuro. In particolare, per garantire l'autenticazione degli utenti e la gestione dei dati in tempo reale, l'app Flutter si integra con i servizi di Firebase e IMGBB. Questo permette non solo di gestire gli accessi degli utenti e archiviare e recuperare dati in modo sicuro ed efficiente, ma anche di gestire in modo centralizzato l'upload e la visualizzazione delle immagini tramite link remoti.

4.2.1 Interazione con Firebase

L'integrazione tra l'app Flutter e i servizi di Firebase rappresenta un elemento cruciale per garantire funzionalità essenziali, come l'autenticazione degli utenti e la gestione dei dati in tempo reale. Il processo inizia con la creazione di un progetto sulla piattaforma Firebase Console, dove è necessario assegnare un nome univoco al progetto, che fungerà da identificativo per i servizi associati.

Dopo aver creato il progetto, è indispensabile attivare i servizi necessari, tra cui Firebase Authentication, per la gestione degli accessi, e Firebase Realtime Database, per la memorizzazione e il recupero dei dati. Per collegare l'app Flutter al progetto Firebase, è necessario scaricare i file di configurazione specifici per ciascuna piattaforma supportata. Nel caso di applicazioni Android, il file richiesto è denominato `google-services.json` e deve essere collocato nella directory `android/app` del progetto Flutter. Per le applicazioni iOS, il file di configurazione è `GoogleService-Info.plist` e va posizionato nella directory del progetto associata al file `Runner` in Xcode.

Una volta posizionati i file di configurazione nelle rispettive directory, si procede con l'integrazione del pacchetto `firebase_core`, essenziale per abilitare i servizi Firebase al-

l'interno del progetto Flutter. Questo pacchetto consente di inizializzare Firebase all'avvio dell'applicazione, garantendo che i servizi cloud siano disponibili fin da subito. L'inizializzazione di Firebase è un passaggio fondamentale per assicurare una comunicazione stabile e sicura tra l'app e i servizi cloud, consentendo di sfruttare appieno le funzionalità offerte dalla piattaforma Firebase.

Questa configurazione permette all'app di interagire con Firebase in modo fluido e senza interruzioni, facilitando l'implementazione di funzionalità avanzate come l'autenticazione degli utenti e la gestione dei dati in tempo reale.

Per quanto riguarda la gestione degli accessi, Firebase Authentication permette agli utenti di accedere utilizzando credenziali basate su email e password. È stata posta particolare attenzione alla gestione degli errori di autenticazione tramite l'uso di eccezioni. Questo approccio consente di fornire all'utente messaggi chiari e contestuali in caso di problemi, come l'inserimento di credenziali non valide o l'assenza di un account registrato. Ad esempio, in caso di tentativo di accesso con email non riconosciuta, l'utente viene avvisato con un messaggio specifico, migliorando l'esperienza utente e riducendo possibili frustrazioni.

4.2.2 Interazione con IMGBB

Un altro servizio fondamentale integrato nell'app è IMGBB, che permette la gestione delle immagini. IMGBB fornisce un'API che consente di caricare le immagini sui suoi server e ottenere un URL remoto per ciascun file. Questi URL possono essere successivamente utilizzati all'interno dell'app per visualizzare le immagini senza doverle archiviare localmente, risparmiando così spazio di archiviazione sui dispositivi degli utenti e migliorando le performance.

Per utilizzare l'API di IMGBB è necessario prima creare un account sul suo sito web ufficiale. Dopo la registrazione, l'utente può accedere alla sezione "API" del pannello di controllo del proprio account, dove è possibile generare una chiave API personale. Questa chiave è fondamentale per autenticare le richieste API dall'app Flutter verso i server IMGBB.

Per integrare IMGBB nel progetto Flutter, è stato utilizzato un pacchetto specifico che facilita la comunicazione tra l'app e il servizio IMGBB. Il processo di upload di un'immagine è semplice: l'immagine viene selezionata dall'utente o acquisita tramite la fotocamera, quindi viene inviata tramite una richiesta HTTP POST all'API di IMGBB, includendo la chiave API come parte della richiesta per l'autenticazione. Una volta completato l'upload, il servizio restituisce un link che può essere utilizzato per visualizzare l'immagine nell'app.

Il flusso di lavoro prevede anche il controllo degli errori, garantendo che l'utente riceva un messaggio appropriato in caso di problemi durante l'upload, come nel caso in cui il formato del file non sia supportato o la connessione a Internet sia assente.

L'integrazione con IMGBB consente, quindi, di gestire e visualizzare le immagini in modo semplice ed efficiente, migliorando l'esperienza dell'utente e ottimizzando la gestione dei contenuti visivi nell'app.

CAPITOLO 5

Implementazione e manuale utente

L'implementazione del progetto consente di trasformare l'idea in un'app concreta e funzionante. In questo capitolo verranno analizzate le componenti principali dell'app e, successivamente, sarà fornita una guida destinata all'utente finale.

5.1 Componenti dell'app

L'applicazione per la gestione della biblioteca dell'Università è stata sviluppata con Flutter, che consente di creare interfacce utente interattive. Un'applicazione Flutter è composta da componenti e risorse che collaborano per offrire un'esperienza completa. I componenti, detti widget, rappresentano i blocchi fondamentali dell'interfaccia e consentono di visualizzare contenuti, gestire interazioni e svolgere le principali funzioni dell'app.

5.1.1 Descrizione della navbar

L'applicazione sviluppata per la gestione della biblioteca dell'Università integra una navbar dinamica, realizzata con il framework Flutter e la libreria Google Nav Bar. Questa componente permette agli utenti di navigare agevolmente tra le diverse sezioni dell'applicazione, adattandosi al ruolo dell'utente autenticato (studente, staff o amministratore) e garantendo una user experience coerente ed efficiente.

La navbar è composta da tre tab principali (Figura 5.1), definiti attraverso il widget GNav della libreria `google_nav_bar`, essi sono:

- *Home*: rappresentata dall'icona *Icons.home*, permette agli utenti di tornare alla schermata principale.
- *Prenotazione*: rappresentata dall'icona *Icons.menu_book*, consente di accedere alla sezione per le prenotazioni dei libri.
- *Posti*: rappresentata dall'icona *Icons.event_seat*, indirizza gli utenti alla sezione dedicata alla prenotazione dei posti studio.

Figura 5.1: Navbar

La navbar non si limita a navigare tra i tab, ma attiva diverse funzionalità a seconda del tab selezionato. Quando l'utente seleziona un tab, l'indice "`_selectedIndex`" viene aggiornato, e un controllo condizionale effettua la navigazione alle rispettive pagine (Figura 5.2).

L'accesso alle funzionalità della navbar varia in base al ruolo dell'utente. Questa personalizzazione è gestita mediante il controllo delle e-mail degli utenti. In particolare, la barra di navigazione viene mostrata solo se:

- l'utente non è un amministratore (!isUserAdmin).
- l'utente si è autenticato (user != null).

```
bottomNavigationBar: !(isUserAdmin) && user != null
    ? Container(
  decoration: BoxDecoration(
    color: Colors.white,
    boxShadow: [
      BoxShadow(blurRadius: 20, color: Colors.black.withOpacity(.1)),
    ],
  ),  // BoxDecoration
  child: SafeArea(
    child: Padding(
      padding: const EdgeInsets.symmetric(horizontal: 15.0, vertical: 8.0),
      child: GNav(
        gap: 8,
        activeColor: Colors.white,
        iconSize: 24,
        padding: EdgeInsets.symmetric(horizontal: 20, vertical: 12),
        duration: Duration(milliseconds: 800),
        tabBackgroundColor: Colors.purple,
        tabs: [
          GButton(
            icon: Icons.home,
            text: 'Home',
          ),  // GButton
          GButton(
            icon: Icons.menu_book,
            text: 'Prenotazione',
          ),  // GButton
          GButton(
            icon: Icons.event_seat,
            text: 'Posti',
          ),  // GButton
        ],
        selectedIndex: _selectedIndex,
        onTabChange: (index) {
          setState(() {
            _selectedIndex = index;
            if (_selectedIndex == 1) {
              Navigator.push(
                context,
                MaterialPageRoute(builder: (context) => PrenotazioniLibroPage()),
              );
            } else if (_selectedIndex == 2) {
              Navigator.push(
                context,
                MaterialPageRoute(builder: (context) => PostiScreen()),
              );
```

Figura 5.2: Codice del navbar

5.1.2 Librerie utilizzate

La funzionalità di scelta della data è fondamentale per permettere all'utente di selezionare una data di inizio per la prenotazione di un libro o di un posto.

Il codice illustrato nella Figura 5.3 mostra l'utilizzo della funzione *showDatePicker* per consentire agli utenti di selezionare una data per la prenotazione di un libro. Questa funzione fa parte della libreria *flutter/material.dart*, che è la libreria principale di Flutter per la creazione di interfacce utente. La funzione *showDatePicker* visualizza un dialogo interattivo che permette agli utenti di scegliere una data da un calendario.

```
Future<void> _selectDate(BuildContext context) async {
  final DateTime? picked = await showDatePicker(
    context: context,
    initialDate: DateTime.now(),
    firstDate: DateTime.now(),
    lastDate: DateTime(DateTime.now().year + 1),
  );
  if (picked != null && picked != DateTime.now()) {
    setState(() {
      dataInizio = DateFormat('dd/MM/yyyy').format(picked);
    });
  }
}
```

Figura 5.3: Funzione *showDatePicker*

5.2 Manuale utente

In questa sezione vengono riportate alcune istruzioni necessarie affinché gli utenti possano usufruire completamente di tutte le funzionalità offerte dall'applicazione.

5.2.1 Prenotazione di un libro

Per effettuare una prenotazione di un libro occorre seguire le istruzioni, cioè:

1. selezionare il libro;
2. scegliere la data come nella Figura 5.4;
3. confermare la prenotazione tramite il dialogo di conferma, come mostrato in Figura 5.5.

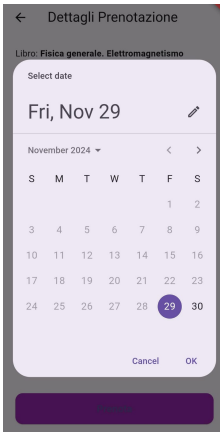

Figura 5.4: Scegliere la data

Figura 5.5: Confermare la prenotazione

Va sottolineato che, nel caso in cui lo studente abbia già prenotato lo stesso libro, verrà mostrato uno SnackBar di avviso, come illustrato in Figura 5.6. Analogamente, se lo studente ha raggiunto il limite massimo di tre prenotazioni, verrà mostrato uno SnackBar di errore, come rappresentato in Figura 5.7.

Figura 5.6: Avviso stesso libro

Figura 5.7: Limite prenotazioni

5.2.2 Ritiro e consegna del libro

La Figura 5.8 mostra la schermata delle prenotazioni accessibile dal personale dello staff. In questa schermata, lo staff può visualizzare tutte le prenotazioni effettuate dagli studenti. È presente un motore di ricerca in cima alla pagina che consente di filtrare le prenotazioni, sia tramite l'email dello studente che attraverso il titolo del libro.

Quando uno studente si presenta per ritirare un libro, lo staff deve semplicemente cliccare sul pulsante "Ritirato" (rappresentato dalla seconda prenotazione nell'immagine). Questa azione avvia il calcolo del tempo per la restituzione del libro.

Al momento della restituzione, se lo studente ha consegnato il libro, lo staff deve cliccare su "Consegnato" (rappresentato dalla prima prenotazione nell'immagine). Il sistema, quindi, verifica se il libro è stato restituito nei tempi stabiliti. Se la restituzione avviene oltre la scadenza, il sistema avvia il calcolo del periodo di sanzione per lo studente.

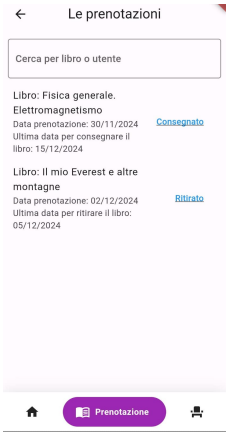

Figura 5.8: Prenotazioni

CAPITOLO 6

Discussione

In questo capitolo verrà effettuata un'analisi SWOT (Strengths, Weaknesses, Opportunities, Threats) per esaminare in modo dettagliato l'applicazione sviluppata per la gestione della biblioteca universitaria. L'analisi ha lo scopo di evidenziare i punti di forza e le debolezze interne del progetto, oltre a identificare le opportunità di crescita e le potenziali minacce esterne.

6.1 Punti di Forza (Strengths)

L'applicazione sviluppata per la gestione della biblioteca universitaria presenta diversi punti di forza che ne migliorano l'efficacia e l'usabilità. Questi aspetti contribuiscono a rendere l'esperienza utente intuitiva e funzionale, garantendo al contempo la flessibilità necessaria per soddisfare le esigenze di diverse tipologie di utenti.

6.1.1 Facilità d'uso e interfaccia intuitiva

Uno dei principali punti di forza è la facilità d'uso garantita da un'interfaccia semplice e intuitiva. La barra di navigazione (navbar) presenta icone chiare e ben organizzate, facilitando l'accesso alle diverse sezioni dell'applicazione. Questa struttura permette agli utenti di orientarsi rapidamente e di comprendere immediatamente le funzioni disponibili. Inoltre, l'utilizzo degli SnackBar come sistema di notifiche interne rappresenta un ulteriore vantaggio: gli studenti ricevono feedback immediato sulle loro azioni, comprendendo facilmente eventuali blocchi o errori, come nel caso di prenotazioni già effettuate o limiti raggiunti. Questo contribuisce a migliorare l'esperienza utente e a ridurre la necessità di supporto esterno.

6.1.2 Versatilità per diversi ruoli

Un altro punto di forza rilevante è la possibilità di utilizzare la stessa applicazione per diversi ruoli, come amministratori, staff e studenti. Questa flessibilità consente una gestione centralizzata e semplificata, evitando la necessità di sviluppare applicazioni separate per ciascun ruolo. Ogni tipologia di utente ha accesso a funzionalità specifiche, adattate alle proprie esigenze, garantendo così un'esperienza personalizzata e ottimizzata per ciascun profilo.

6.2 Punti di Debolezza (Weaknesses)

Nonostante l'applicazione presenti numerosi vantaggi, esistono alcune criticità che potrebbero influire negativamente sull'esperienza utente e sull'efficienza complessiva del sistema. Di seguito vengono analizzati i principali punti di debolezza individuati.

6.2.1 Assenza di modalità offline

Uno dei principali limiti dell'applicazione è la necessità di una connessione Internet per accedere alle funzionalità offerte. L'assenza di una modalità offline impedisce agli studenti di consultare le informazioni o gestire le prenotazioni in assenza di rete. In caso di connessione assente, home page diventa completamente bianca, poiché il sistema non è in grado di recuperare né i dati relativi ai libri né le immagini. Questa criticità potrebbe risultare problematica per gli studenti che si trovano in aree con connessione instabile o assente, limitando l'accessibilità ai servizi della biblioteca.

6.2.2 Mancanza di notifiche agli studenti

Un ulteriore punto di debolezza è l'assenza di un sistema di notifiche integrato. L'applicazione, infatti, non invia promemoria agli studenti in prossimità della scadenza per la restituzione dei libri. La mancanza di tali avvisi potrebbe causare ritardi nella consegna e, di conseguenza, l'applicazione di sanzioni. L'introduzione di notifiche automatizzate, come mostrato nella Figura 6.2, contribuirebbe a migliorare la puntualità degli utenti e a prevenire situazioni di disagio.

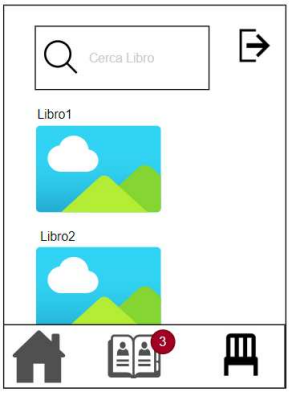

Figura 6.1: L'app con l'icona delle notifiche

6.3 Opportunità (Opportunities)

L'applicazione presenta diverse possibilità di miglioramento e sviluppo che potrebbero incrementarne il valore e l'utilità sia per gli utenti che per l'amministrazione della biblioteca. Di seguito si analizzano alcune delle principali opportunità che potrebbero essere sfruttate per potenziare il sistema.

6.3.1 Introduzione del feedback degli studenti sui libri

Una delle opportunità più significative riguarda l'introduzione di un sistema di feedback da parte degli studenti sui libri presi in prestito. Questa funzionalità permetterebbe agli studenti di recensire i libri e condividere opinioni, favorendo una scelta più consapevole per

altri studenti. Inoltre, le recensioni potrebbero offrire alla biblioteca informazioni preziose sulla qualità e l'utilità dei testi, agevolando le decisioni relative all'acquisto di nuovi volumi, come illustrato nella Figura 6.2.

Figura 6.2: Schermata con la sezione per le recensioni

6.3.2 Statistiche dettagliate per l'amministrazione

Un'altra opportunità consiste nell'implementazione di un sistema di analisi e reportistica per l'amministrazione della biblioteca. Fornire statistiche dettagliate sui libri più prenotati, sui tempi medi di prestito e sulle categorie più richieste consentirebbe agli amministratori di ottimizzare la gestione del catalogo e migliorare l'offerta. Tali informazioni potrebbero, inoltre, supportare decisioni strategiche e allocazioni di risorse più efficaci.

6.4 Minacce (Threats)

L'applicazione per la gestione della biblioteca universitaria deve affrontare alcune potenziali minacce che potrebbero influenzarne il successo e l'efficacia. Questi fattori esterni richiedono una costante attenzione per garantire la sicurezza dei dati e la competitività del servizio offerto.

6.4.1 Concorrenza

Una delle principali minacce è rappresentata dalla concorrenza, in particolare dal sito ufficiale della biblioteca universitaria. Questo portale offre servizi simili, come la prenotazione degli spazi di studio. Poiché l'app fornisce funzionalità comparabili, c'è il rischio che gli utenti preferiscano utilizzare il sito ufficiale, soprattutto se percepito come più affidabile o consolidato. Per affrontare questa minaccia, sarà importante differenziare l'applicazione offrendo funzionalità aggiuntive o miglioramenti nell'usabilità rispetto al sito web.

6.4.2 Normative sulla privacy e sicurezza dei dati

Un'altra minaccia significativa riguarda le normative sulla privacy e la sicurezza dei dati. L'applicazione utilizza Firebase per l'autenticazione, richiedendo agli studenti di accedere tramite la loro email universitaria, che include informazioni sensibili come la matricola. In caso di attacchi informatici o violazioni della sicurezza, potrebbero essere compromesse informazioni personali, con conseguenti rischi legali e reputazionali. Per proteggere la privacy

degli utenti e ridurre i rischi legati alla sicurezza dei dati, è necessario adottare misure di protezione, come la crittografia dei dati e l'uso di sistemi di accesso sicuri. Inoltre, è fondamentale seguire le normative sulla privacy, come il GDPR, per assicurarsi che i dati siano gestiti correttamente, e per tutelare gli utenti.

Conclusione

Nei capitoli precedenti sono stati descritti la progettazione, lo sviluppo e l'implementazione di un'applicazione mobile per la gestione della biblioteca universitaria. L'obiettivo principale era quello di fornire un sistema digitale che automatizzasse il processo di prenotazione, ritiro e restituzione dei libri, migliorando l'esperienza degli utenti e facilitando le operazioni per il personale di biblioteca. Le informazioni necessarie sono state raccolte attraverso un'analisi dei requisiti, che ha coinvolto sia il personale della biblioteca, gli utenti finali, sia i dati provenienti dal sito web della biblioteca.

Durante lo sviluppo, una delle sfide principali è stata garantire una perfetta integrazione tra il front-end, sviluppato con Flutter, e i servizi di backend offerti da Firebase, tra cui Firebase Authentication, per la gestione dell'autenticazione degli utenti, e Firebase Realtime Database, per il salvataggio dei dati in tempo reale. Inoltre, la gestione delle immagini è stata affidata a IMGBB, che consente di caricare e visualizzare immagini dei libri direttamente nell'applicazione.

Il sistema proposto offre funzionalità complete, tra cui la possibilità di prenotare libri, gestire la disponibilità degli spazi e monitorare il ritiro e la restituzione dei libri. Queste funzionalità hanno migliorato significativamente l'efficienza dei processi all'interno della biblioteca, riducendo il rischio di errori manuali e aumentando la disponibilità delle risorse.

In futuro, l'app potrebbe essere arricchita con funzionalità avanzate, come l'invio di notifiche push per ricordare agli utenti le scadenze delle prenotazioni o l'integrazione di un sistema di statistiche per l'analisi delle tendenze di prenotazione e di utilizzo dei libri. Inoltre, l'aggiunta di un sistema di feedback da parte degli utenti potrebbe contribuire al miglioramento continuo del servizio offerto dalla biblioteca.

Bibliografia

BRYANT, C. (2020), *Practical Flutter: Improve your Mobile Development with Google's Latest Open-Source SDK*, O'Reilly Media.

DEBAYO, O. (2021), *Flutter for Beginners: An Introduction to Flutter Development with Dart*, Packt Publishing.

JABARI, H. (2021), *Flutter Projects: Learn to Build Cross-Platform Mobile Applications with Flutter and Dart*, Packt Publishing.

KEMP, J. (2019), *Beginning Flutter: A Hands-On Guide to App Development*, Apress.

MCCARTHY, J. (2021), *Dart: Up and Running: A Hands-On Guide to Programming in Dart*, O'Reilly Media.

ROGERS, K. (2021), *Flutter for Mobile Application Development: Learn how to develop mobile applications with Flutter using Dart*, Independently Published.

SOMMERVILLE, I. (2011), *Software Engineering*, Pearson Education, 9th ed.

WANG, F. (2020), *Building Mobile Apps with Flutter and Dart: A Beginner's Guide to Mobile Development*, Independently Published.

Sitografia

- Flutter Documentation – https://flutter.dev/docs
- Flutter Gallery – https://gallery.flutter.dev/#/
- Dart Programming Language Documentation – https://dart.dev/guides
- Android Studio Official Documentation – https://developer.android.com/studio
- Firebase Realtime Database Documentation – https://firebase.google.com/docs/database
- Firebase Authentication Documentation – https://firebase.google.com/docs/auth
- Firebase Flutter Documentation – https://firebase.flutter.dev/docs/overview
- IMGBB Official Documentation – https://imgbb.com/
- NoSQL Databases Overview – https://www.mongodb.com/nosql-explained
- Firestore Documentation – https://firebase.google.com/docs/firestore

Ringraziamenti

Ringrazio la mia famiglia che, nonostante la distanza dall'Egitto, mi ha sempre sostenuto moralmente durante tutto il percorso universitario.

Ringrazio il Prof. Corradini per i preziosi consigli e il supporto durante l'implementazione del progetto.

Vorrei ringraziare gli amici conosciuti in questi anni di università, che mi hanno aiutato nello studio e con cui ho condiviso tante ore di preparazione agli esami. In particolare, ringrazio Morcos ed Ebram, che, essendo più grandi, mi hanno sempre fornito materiale utile per gli esami e consigli preziosi su come affrontare al meglio lo studio.