

Movie Survey

Titolo del progetto

Movie Survey

Alunno/a

Jan Rezzonico

Classe

Info 4 AA

Anno scolastico

2023/2024

Superiore professionale

Maurizio Di Florio

1 Indice

1	Indice	2
2	Introduzione.....	4
2.1	Informazioni sul progetto.....	4
2.2	Abstract	4
2.3	Scopo	4
3	Analisi	5
3.1	Analisi del dominio	5
3.2	Analisi e specifica dei requisiti	5
3.3	Use case	7
3.4	Pianificazione	7
3.5	Analisi dei mezzi.....	9
3.5.1	Software	9
3.5.2	Hardware.....	9
4	Progettazione	10
4.1	Design dell'architettura del sistema	10
4.2	Design dei dati e database.....	12
4.3	Design delle interfacce	13
4.4	Design procedurale	17
5	Implementazione	19
5.1	Gestione delle versioni.....	19
5.2	Convenzioni codice	19
5.3	Getting started.....	20
5.3.1	Backend	20
5.3.2	Frontend.....	20
5.3.3	Build e deploy	21
5.4	Backend	22
5.4.1	Creazione progetto	22
5.4.2	Struttura cartelle.....	22
5.4.3	Versioni database	23
5.4.4	Listener	25
5.4.5	Controller.....	25
5.4.6	Emitter.....	26
5.4.7	Codice particolare	26
5.5	Frontend	29
5.5.1	Struttura cartelle.....	29
5.5.2	Socket	29
5.5.3	Salvataggio locale	30
5.5.4	Gestione lingua	30
5.5.5	Costanti	30
5.5.6	Font	30
5.5.7	Blocco portrait	31
5.5.8	Toast	31
5.5.9	Componenti particolari	32
5.5.10	Risultati interfacce.....	35
6	Test.....	39
6.1	Protocollo di test.....	39
6.2	Risultati test	44
6.3	Mancanze/limitazioni conosciute.....	50
6.3.1	Bug voti troppo rapidi	50
6.3.2	Disconnessione socket con app in background.....	50
6.3.3	Blocco portrait	50
7	Consuntivo.....	51
8	Conclusioni	52
8.1	Sviluppi futuri.....	52
8.1.1	Modalità aggiuntive	52

8.1.2	Selezione manuale di film	53
8.1.3	Collegamento e filtri con servizi di streaming	53
8.1.4	Supporto multilingua sui film	53
8.1.5	Deploy	53
8.2	Considerazioni personali	54
9	Bibliografia	55
9.1	Sitografia	55
10	Glossario	56
11	Indice delle figure	57
12	Allegati	58

2 Introduzione

2.1 Informazioni sul progetto

Allievo: Jan Rezzonico

Superiore Professionale: Maurizio Di Florio

Perito 1: Gionata Genazzi

Perito 2: Andrea Gazzi

Scuola: CPT Trevano, SAMT I4AA

Data inizio: 25.04.2024

Data fine: 23.05.2024

2.2 Abstract

It's probably happened to everyone at least once: you are with your friends or family and want to watch a movie, however either no one has any idea on what to watch or everyone has a different idea. Movie Survey is an application designed to fix this very problem, by offering to the user various lists of movies to then vote in order to find the perfect compromise. Movie Survey is available on both Android and iOS devices thanks to React Native. The movie information is obtained from TheMovieDB API, which offers plenty of movies and suggestions that can fit anyone's taste.

2.3 Scopo

Questo progetto ha come obiettivo operativo la creazione di un'applicazione mobile che permetta agli utenti di organizzare sondaggi tra amici e familiari per raggiungere un compromesso su quale film guardare. L'applicazione offre diverse liste di film tra cui scegliere e, una volta creata una lobby e invitati tutti i partecipanti, questi possono votare i film indicando se desiderano vederli o meno. Al termine della votazione, viene visualizzata una classifica dei film più votati positivamente. Da questo punto, è possibile avviare una nuova iterazione di votazione considerando solo i film meglio votati precedentemente.

Gli obiettivi didattici di questo progetto sono migliorare le mie conoscenze personali di React Native, Expo, MongoDB, Mongoose e, in aggiunta, imparare ad utilizzare i web socket tramite Socket.io. Oltre a queste tecnologie, il progetto mira a dimostrare e migliorare le mie capacità di documentazione, gestione del tempo e pianificazione di un progetto nell'ambito dell'informatica.

3 Analisi

3.1 Analisi del dominio

Il prodotto dovrà funzionare su dispositivi mobile e l'utente potrebbe essere chiunque abbia uno smartphone e desidera utilizzare l'applicazione per essere aiutato a prendere una decisione. Gli utenti possono provenire da qualsiasi tipo di background a patto che sappiano come utilizzare un'applicazione sul proprio smartphone. Al momento non sembrerebbe esistere un'applicazione apposita sul mercato, ma alcuni servizi di streaming offrono già una funzionalità simile all'interno della loro applicazione. Per operare in questo dominio bisogna avere delle conoscenze di sviluppo applicazioni e conoscenze di sviluppo backend.

3.2 Analisi e specifica dei requisiti

Req-001	
Nome	Creazione lobby
Priorità	1
Versione	1.0
Note	Un utente può creare una lobby e condividerla con il gruppo

Req-002	
Nome	Entrata in una lobby
Priorità	1
Versione	1.0
Note	Un utente può entrare in una lobby creata da un altro utente
Sotto requisiti	
SubReq_1	Feature opzionale: condivisione tramite codice QR
SubReq_2	Gli utenti si possono unire alla lobby anche dopo che l'amministratore l'ha avviata (Req-004)

Req-003	
Nome	Uscita da una lobby
Priorità	1
Versione	1.0
Note	Un utente può in qualsiasi momento uscire dalla lobby. Il comportamento è però differente in base al tipo di utente che esce.
Sotto requisiti	
SubReq_1	I film a disposizione devono essere ottenuti da una API pubblica

Req-004

Nome	Scelta dei film da votare
Priorità	1
Versione	1.0
Note	L'utente che ha creato la lobby deve avere a disposizione una o più liste di film da selezionare per mandarle in votazione
Sotto requisiti	
SubReq_1	I film a disposizione devono essere ottenuti da una API pubblica

Req-005

Nome	Avviare sessione di votazione
Priorità	1
Versione	1.0
Note	L'utente che ha creato la lobby può far partire la votazione. Una volta inviato questo comando a tutti gli utenti viene presentata la pagina per votare i film.

Req-006

Nome	Votazione film
Priorità	1
Versione	1.0
Note	Ogni film può essere votato sia in modo positivo che in modo negativo.

Req-007

Nome	Classifica film
Priorità	1
Versione	1.0
Note	Una volta terminata la votazione dei film è mostrata la classifica dei film in ordine dal più votato positivo a quello meno positivamente votato

Req-008

Nome	Trailer film
Priorità	2
Versione	1.0
Note	È possibile lanciare il trailer del film direttamente dall'app senza uscire

Req-009

Nome	Multiple iterazioni di votazione
Priorità	1
Versione	1.0
Note	È possibile lanciare, dalla schermata della classifica, una nuova iterazione di votazioni per dare possibilità di dare di nuovo voti tra i film più votati nell'iterazione precedente.

3.3 Use case

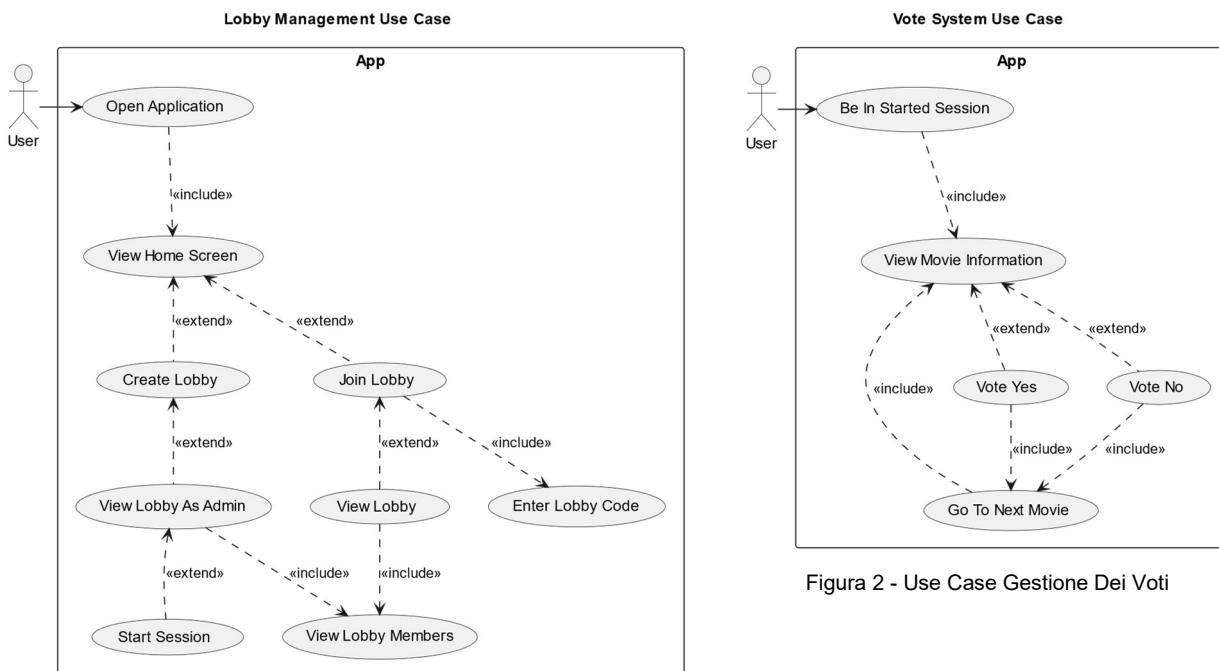


Figura 1 - Use Case Gestione Lobby

Queste due immagini sono Use Case dell'applicazione, ovvero diagrammi usati per descrivere i requisiti funzionali di un sistema o sottosistema. In questo caso questi diagrammi rappresentano la gestione delle lobby (a sinistra) e la gestione dei voti (a destra). Questi diagrammi sono creati per rappresentare in una semplice immagine le funzionalità incluse rispettivamente nella pagina Home e nella pagina di votazione dei film.

3.4 Pianificazione

Per pianificare il progetto è stata utilizzata la metodologia Waterfall, ovvero una metodologia che prevede una gestione dei progetti in modo sequenziale, dove il progetto viene suddiviso in fasi distinte e una fase può iniziare solo dopo che quella precedente è stata completata. Utilizzare questa metodologia porta ad avere una pianificazione dei tempi molto più dettagliata ma può anche causare problemi in quanto ha una flessibilità nettamente inferiore alle metodologie Agile. Nella prossima pagina è possibile vedere il diagramma di Gantt preventivo, ovvero il diagramma utilizzato per pianificare le attività sul tempo per lo svolgimento dei progetti utilizzanti la metodologia Waterfall.

Movie Survey

Il diagramma di Gantt preventivo contiene quattro categorie principali: Pianificazione, Progettazione, Implementazione e Test. Rispetto al classico stile Waterfall, mancano due categorie: Analisi e Manutenzione. La prima è inclusa nella Pianificazione, mentre la seconda è omessa poiché il progetto non prevede attività di manutenzione. La sezione Implementazione è suddivisa in due sottosezioni: Backend e Frontend. Il modello Waterfall impone un ordine ben definito delle attività, ma in questo progetto il backend è stato modificato più volte durante lo sviluppo del frontend, a causa della natura bidirezionale dei socket. La sezione Test include solo l'esecuzione dei test case, anche se alcuni test sono stati eseguiti durante le fasi di sviluppo. La documentazione si svolge parallelamente a tutto il progetto e ogni attività include sia il tempo di attuazione che quello di documentazione. Ad esempio, l'attività di creazione dei listener/emitter è stata pianificata per durare due ore; tuttavia, questo tempo comprende sia l'implementazione che la documentazione.

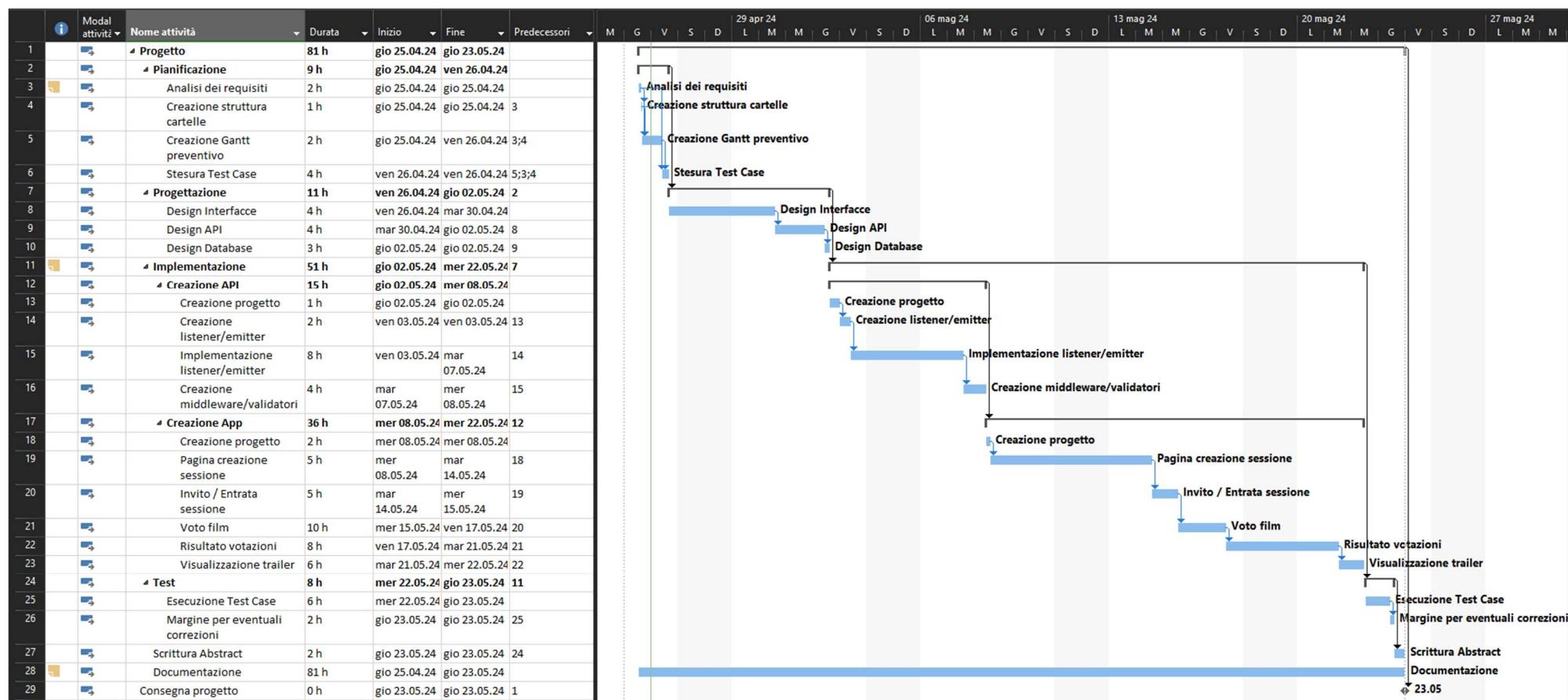


Figura 3 - Gantt Preventivo

3.5 Analisi dei mezzi

3.5.1 Software

Nome	Versione
Visual Studio Code	1.86.2
git	2.40.0.windows.1
Microsoft Word	Professional 2019
Windows 10	Enterprise 22H2
Node.js	v18.14.2
npm	9.5.0
PlantUML by jebbs	v2.17.5
Moon Modeler	7.3.0
mongod	7.0.1
java	openjdk 11.0.16.1 2022-08-12 LTS
bundletool.jar	1.15.6

3.5.2 Hardware

Il progetto è stato sviluppato su un pc fornito dalla scuola con le seguenti caratteristiche:

- Sistema Operativo: Windows 10 Enterprise
- Versione: 22H2
- Processore: Intel(R) Core (TM) i7-9700 CPU @ 3.00GHz
- RAM: 32.0 GB
- GPU: NVIDIA GeForce RTX 2060
- Base: 64 bit, processore su x64

Inoltre sono anche stati utilizzati i seguenti dispositivi mobile per testare l'applicazione:

- Galaxy Tab A8
 - OS Versione 11
- iPad Pro (10,5")
 - OS Versione 15.5
- iPhone XR
 - OS Versione 15.5

4 Progettazione

4.1 Design dell'architettura del sistema

L'idea fondamentale è di avere una comunicazione tramite socket. Il client socket si deve connettere al server socket in modo da ricevere ed emettere degli eventi. Il server socket è anche connesso ad un database MongoDB, utilizzato per salvare i dati. Il database per come è stata sviluppata l'applicazione contiene sia dati persistenti che non. I dati persistenti sono le liste di film, mentre quelli non persistenti sono quelli delle lobby, che vengono eliminati ad ogni distruzione di lobby. Il database è accessibile solo localmente sulla porta 27017, mentre il server socket è idealmente raggiungibile tramite internet sulla porta 3000.

Qui voglio aprire una piccola parentesi su Socket.io. Socket.io internamente utilizza il protocollo WebSocket che è differente dal protocollo HTTP. HTTP, infatti, è un protocollo di comunicazione stateless half-duplex basato su richiesta/risposta ed è principalmente orientato alle richieste del client. WebSocket, invece, è un protocollo di comunicazione bidirezionale full-duplex e stateful che fornisce un canale di comunicazione persistente e interattivo tra client e server su una singola connessione TCP/IP. Mentre tutti e due di default utilizzano le porte 80 e 443, sono due protocolli di natura differente. HTTP non è full-duplex, ovvero, mentre consente al client di inviare richieste al server e al server di rispondere al client, questa comunicazione avviene in un flusso di scambio unidirezionale: il client infatti invia una richiesta e il server risponde a quella specifica richiesta, non c'è un flusso continuo di comunicazione bidirezionale, ma solo una richiesta per ogni canale di connessione. I WebSocket, d'altro canto, permettono una comunicazione anche simultanea tra client e server senza necessità di aprire nuove connessioni per ogni scambio di dati in parallelo. La connessione WebSocket tipicamente utilizza il protocollo HTTP/S per eseguire un handshake, ma successivamente esegue un upgrade e comincia a comunicare in full-duplex tramite il protocollo WebSocket. Tutto questo è gestito internamente da Socket.io.

È stato scelto di utilizzare i socket rispetto a una REST API perché l'applicazione richiede numerose operazioni bidirezionali, in cui il server deve inviare messaggi al client. Anche se questo è possibile tramite polling nel caso di una REST API, l'uso dei socket è molto più appropriato poiché sono stati ideati proprio per gestire comunicazioni bidirezionali in tempo reale in modo efficiente.



Figura 4 - Schema di rete

Movie Survey

Socket Logic API Design

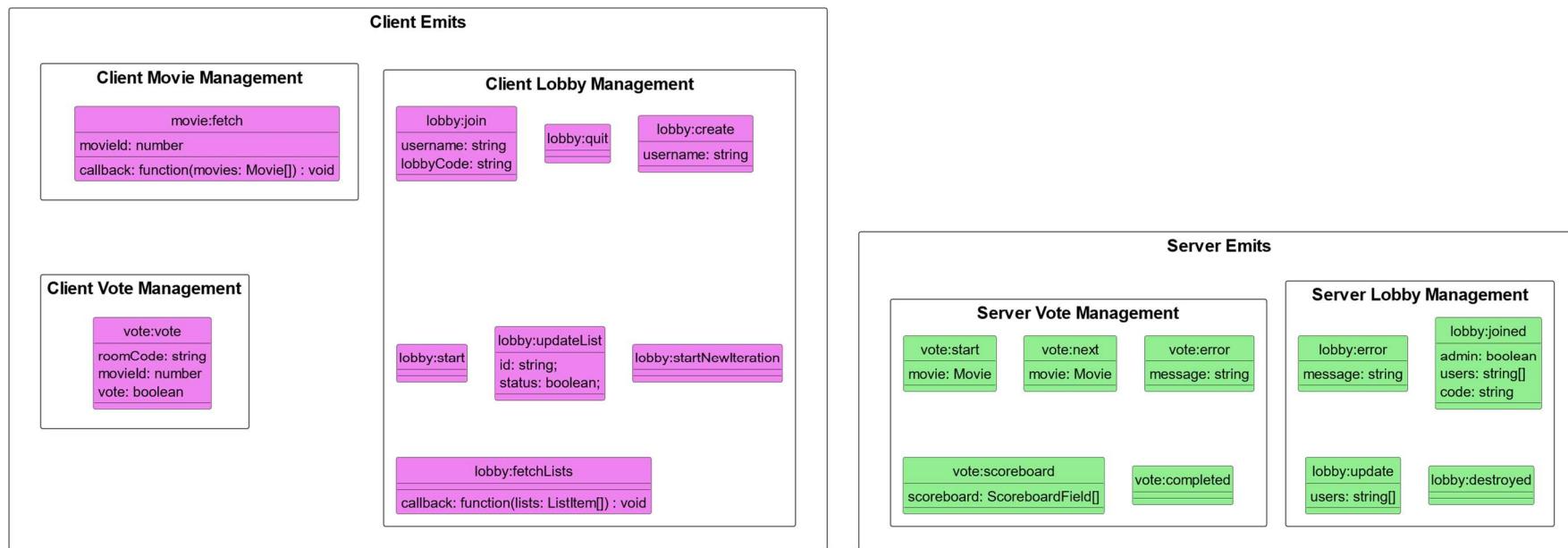
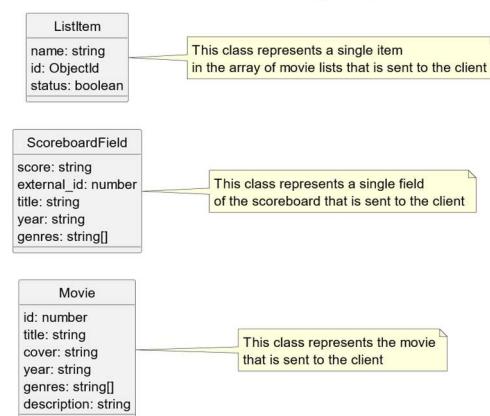


Figura 5 - Logica socket

Nell'immagine sopra si può vedere la logica dei socket. Il rettangolo a sinistra contiene gli emit effettuati dal server, mentre quello a destra quelli effettuati dal client. Le comunicazioni possono essere suddivise in tre gruppi: legate alla lobby, legate al voto o legate al film. Per standardizzare queste comunicazioni ho deciso di creare la convenzione nomeContesto:nomeFunzione, ad esempio, quando un utente vuole entrare in una lobby, emette il messaggio lobby:join. Questo mi ha aiutato a tenere la codebase in ordine. Nell'immagine a destra possiamo vedere i dettagli delle classi specificate all'interno del diagramma sopra.

Figura 6 - Classi
diagramma logica
socket



4.2 Design dei dati e database

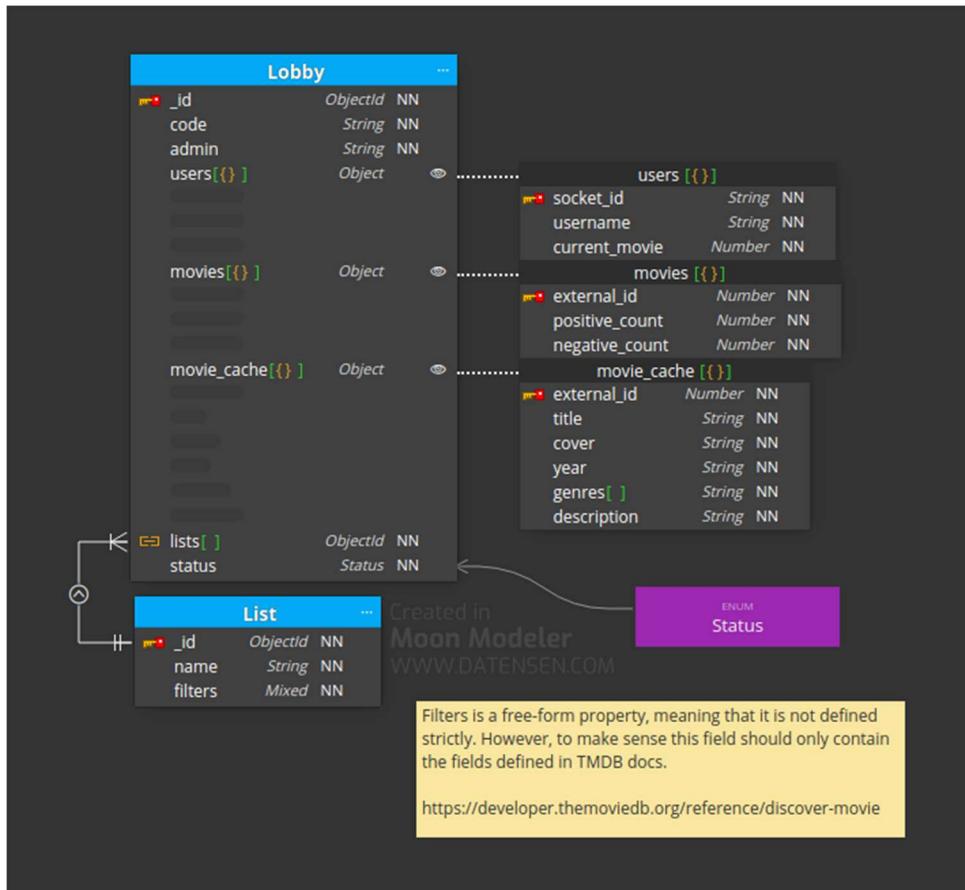


Figura 7 - Diagramma database

Nell'immagine sopra possiamo vedere il diagramma del database. Come database è stato utilizzato MongoDB assieme all'ODM Mongoose, quindi ci troviamo davanti ad un database non relazionale. In questo database sono presenti due collezioni, ovvero `Lobby` e `List`. `Lobby` rappresenta, come intuibile dal nome, una lobby dell'applicazione.

Una lobby contiene il proprio codice d'accesso, il socket id dell'amministratore, gli utenti, un array per il salvataggio delle votazioni, uno per fare da cache dei film, le liste sui quali si basa la scelta di film ed uno stato. Gli utenti vengono salvati tramite il proprio socket id e username, inoltre viene anche tenuto conto del film che stanno votando in questo momento, inizialmente 0.

Le votazioni vengono salvate all'interno di `movie`, dove viene salvato l'id esterno del film (id di TMDB), conteggio di voti positivi e conteggio di voti negativi. All'interno di `movie_cache` vengono salvati i dati dei film. Questo per evitare di dover fare numerose richieste ogni volta che un utente richiede un film. Vengono salvati solo i dati che devono essere mandati all'utente. La proprietà `status` è un enumerabile, dove i suoi possibili valori sono `waiting`, `started` e `scoreboard`. Questo stato indica in che momento la lobby si trova. La proprietà `lists` è un array di `ObjectId` che fanno riferimento alla collezione `List`.

La collezione `List` ha solo due campi, ovvero il nome della lista che verrà mostrato all'amministratore e i filtri che questa lista va ad applicare. La proprietà dei filtri è `Mixed`, ovvero un oggetto libero. Questo vuol dire che si può inserire in questa proprietà qualsiasi sotto proprietà di qualsiasi tipo. Questo è stato fatto perché altrimenti si sarebbe dovuto specificare tutti i possibili filtri offerti da TMDB. Logicamente, però, alla creazione di un'istanza di `List` si vuole far ben attenzione che l'oggetto sia conforme alle regole impostate dall'API.

4.3 Design delle interfacce

Per mantenere un design consistente ho deciso di creare una color palette con i colori principali dell'applicazione. L'applicazione prevede solo un tema scuro e nessun tema chiaro, anche se sarebbe relativamente semplice in futuro implementare diversi temi. Inoltre è stato progettato un design delle interfacce che è stato seguito quasi interamente durante la fase di implementazione. L'applicazione è progettata per essere utilizzata da dispositivi mobile, principalmente telefoni ma anche tablet. L'interfaccia è progettata per funzionare in portrait e non in landscape, quindi nella soluzione finale non sarà possibile cambiare l'orientazione dello schermo. Durante la fase di sviluppo, è emersa una differenza tra le interfacce progettate inizialmente e quelle effettivamente implementate. Questo perché alcune delle interfacce previste sono state ritenute non necessarie o migliorabili.

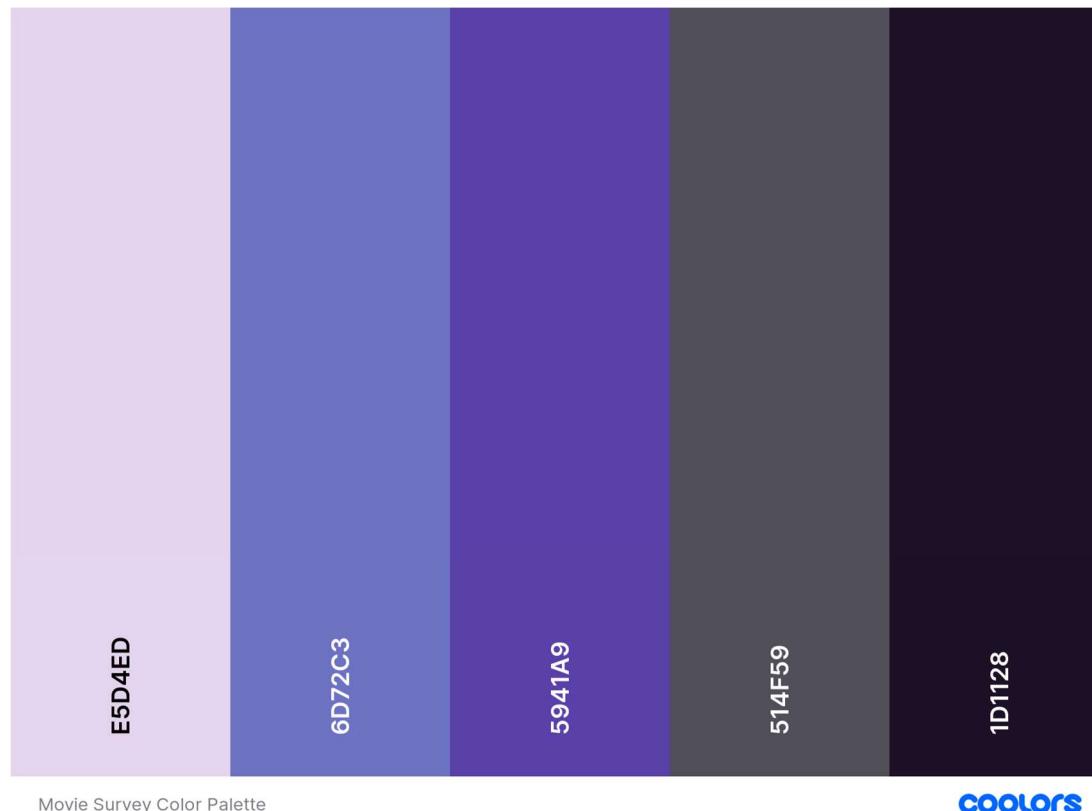


Figura 8 - Color Palette

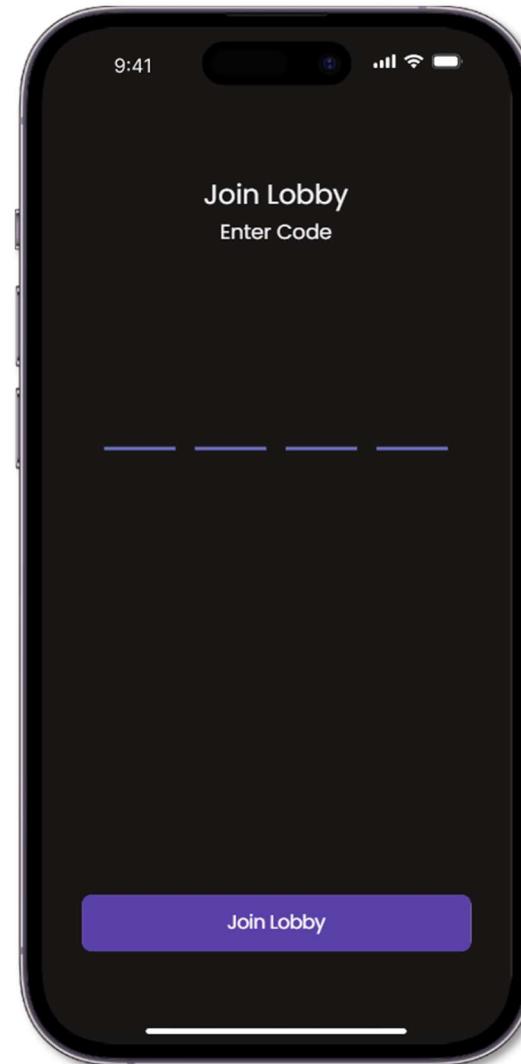
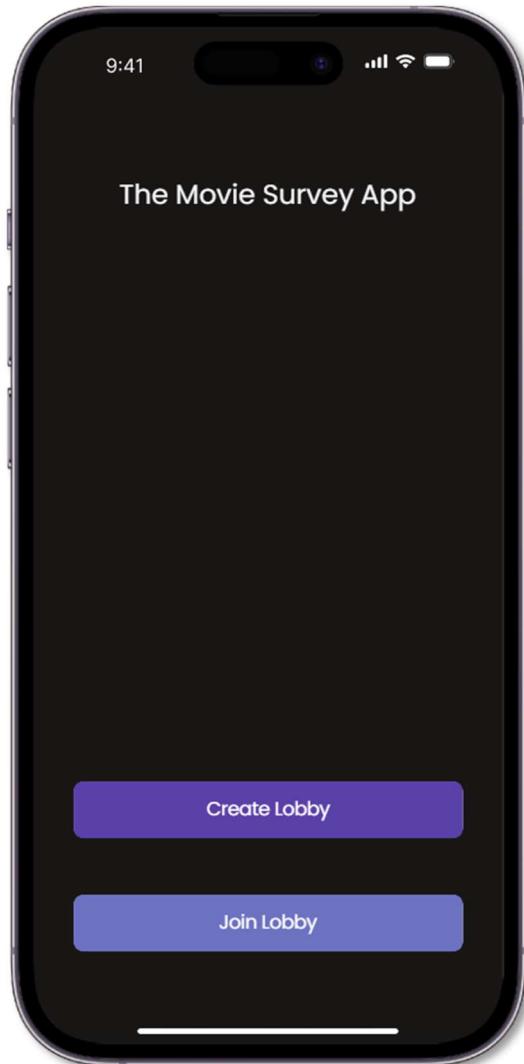


Figura 9 - Pagina iniziale

Figura 10 - Pagina creazione lobby

Figura 11 - Pagina join lobby

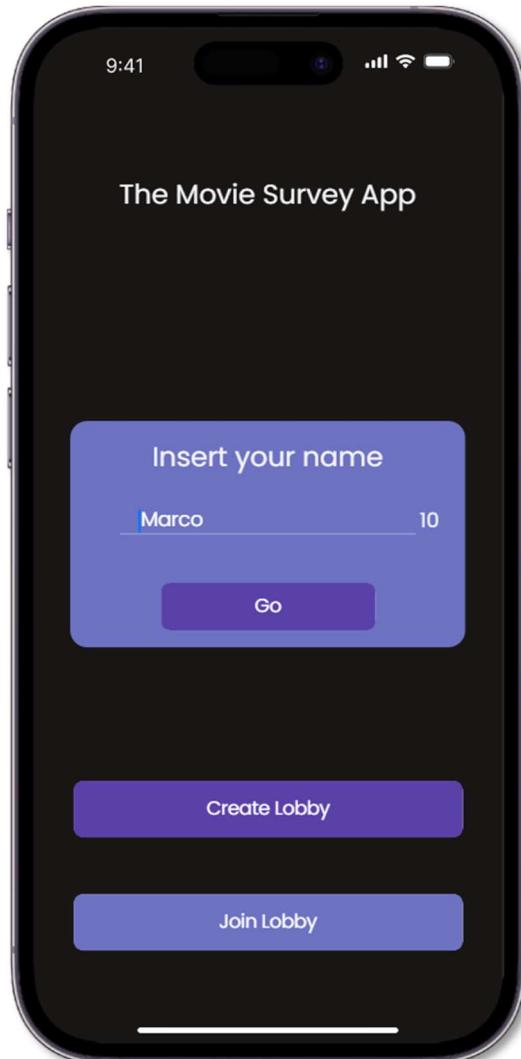


Figura 12 - Modal inserimento username

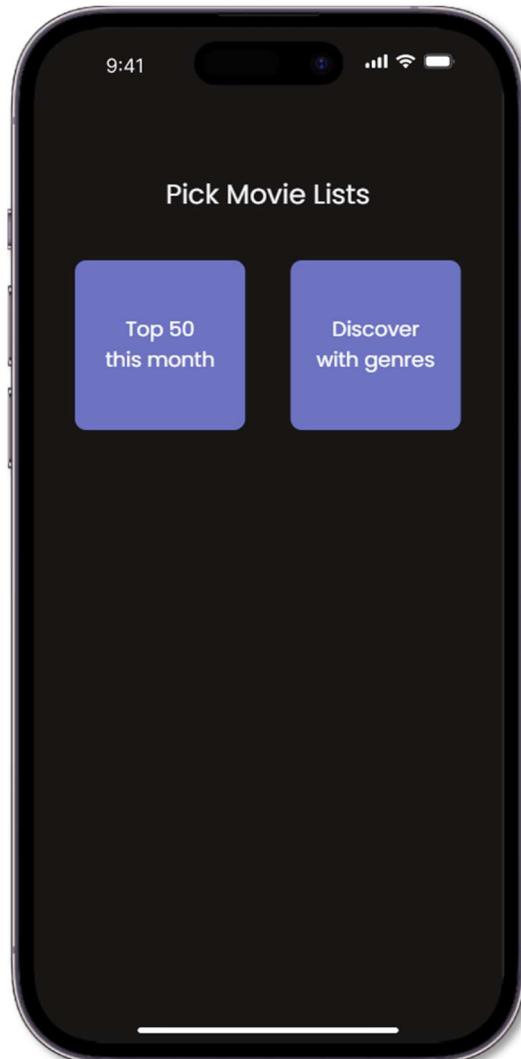


Figura 13 - Pagina scelta list di film

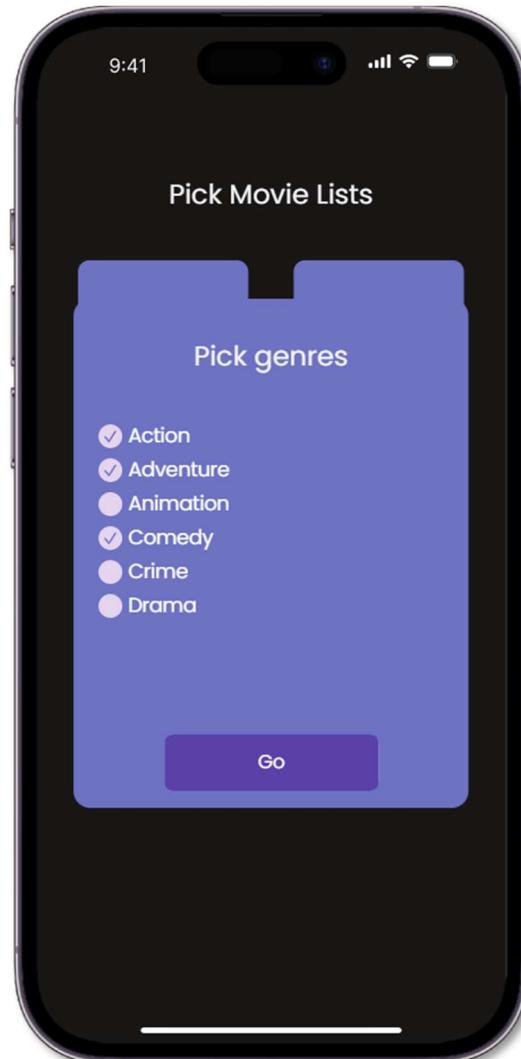


Figura 14 - Pagina scelta generi

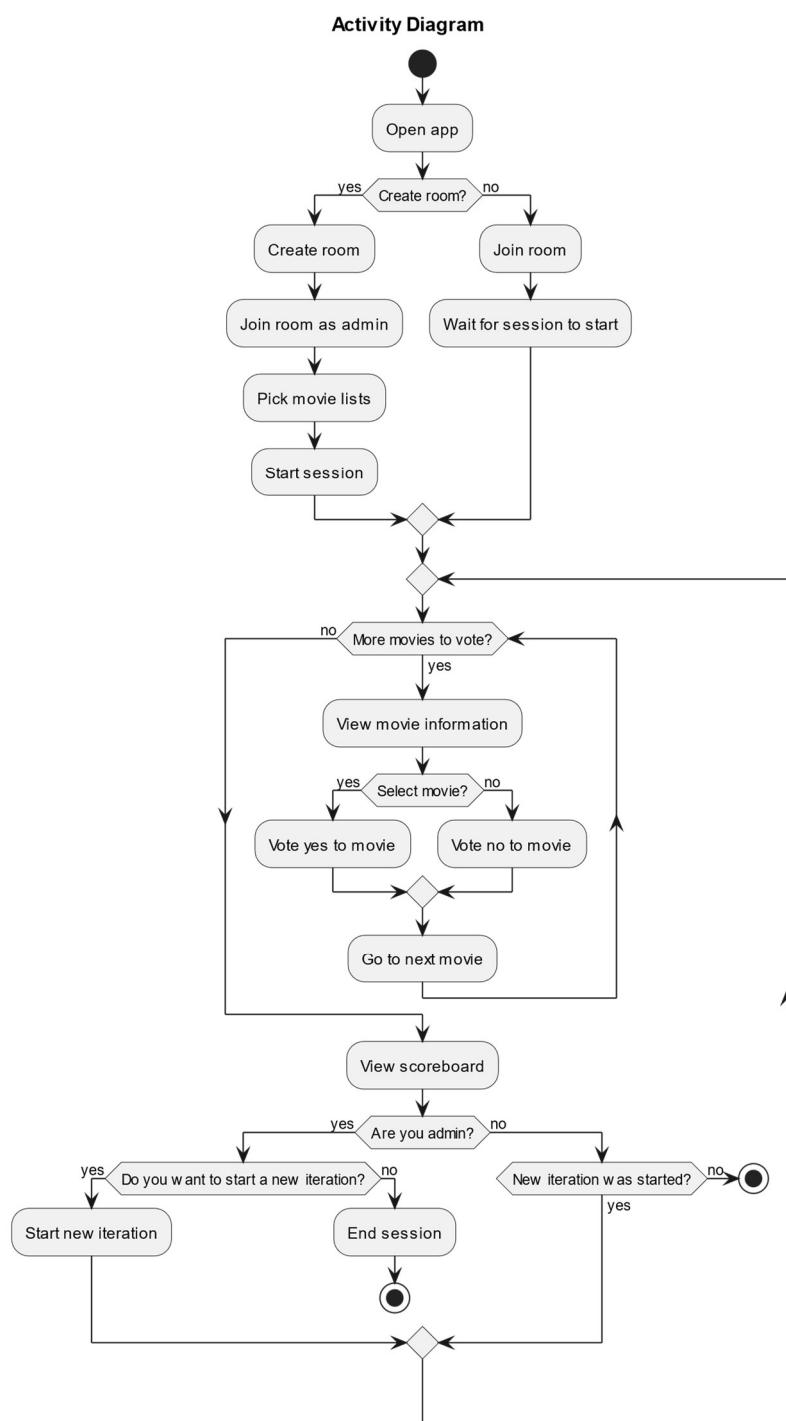


Figura 15 - Pagina votazione film



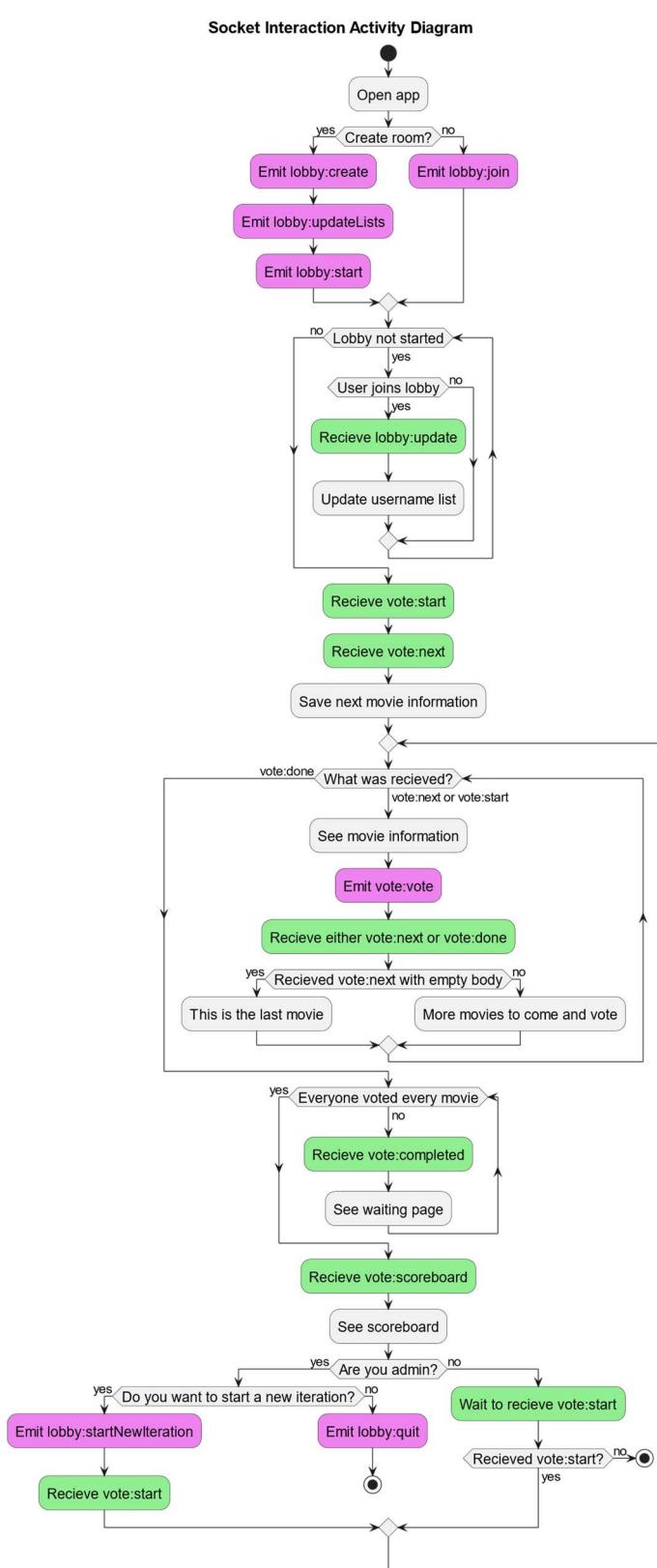
Figura 16 - Pagina scoreboard

4.4 Design procedurale



L'immagine sulla sinistra è l'Activity Diagram rappresentante il flusso logico generale dell'applicazione. Si può vedere come all'inizio è presente la logica di creazione o entrata di una lobby, successivamente la logica di votazione e dopo ancora la logica per far partire una nuova iterazione. Questo diagramma rappresenta soltanto il flusso generale e non è molto tecnico. Per un diagramma tecnico guardare quello della prossima pagina.

Figura 17 - Activity Diagram Generale



L'immagine sulla sinistra è l'Activity Diagram rappresentante il flusso logico nel contesto dei socket. Possiamo vedere come è molto simile a quello della pagina precedente, che è logico in quanto il flusso rimane più o meno lo stesso. Le attività sono suddivise in 3 categorie, ben visibili tramite i colori. In viola si vedono gli emit da parte del client, in verde gli emit da parte del server e in grigio le attività non strettamente contenenti un'azione sui socket.

Figura 18 - Activity Diagram Socket

5 Implementazione

5.1 Gestione delle versioni

Per assicurare una gestione efficiente delle versioni del progetto e del suo codice, ho adottato il sistema di controllo delle versioni Git. Il repository Git è ospitato sul servizio GitLab, accessibile all'indirizzo http://gitsam.cpt.local/2023_2024_lpi/movie-survey.

Durante la fase iniziale, i miei superiori hanno creato il repository per ospitare il progetto, il quale ho clonato per consentirmi di avviare il lavoro. Questa procedura ha garantito una solida base per la collaborazione e la gestione condivisa del codice tra me ed i miei superiori. Ad ogni cambiamento significativo, ho eseguito un commit, caratterizzato da un nome descrittivo, al fine di agevolare il processo di versioning e facilitare eventuali operazioni di recovery. Questa pratica mi ha consentito di mantenere un registro accurato delle modifiche apportate al codice e alla documentazione nel corso dello sviluppo. All'interno del progetto sono stati utilizzati dei file .gitignore. Questi file servono per escludere dei file o cartelle dai commit a causa della loro sensibilità o dimensione.

All'interno delle API sono presenti questi file per escludere la cartella node_modules, a causa della sua dimensione e .env, in quanto contiene informazioni sensibili.

5.2 Convenzioni codice

Generalmente, le convenzioni di codice applicate seguono gli standard dei rispettivi linguaggi o contesti. Per fare un rapido riepilogo:

Contesto	Nome	Convenzione	Esempio
Database	Collezione	Singolare, Capitalized	List
Database	Campo collezione	snake_case	positive_count
Backend - Database	Interfaccia Modello	UpperCamelCase Interface notation	IList
Frontend	Componente	UpperCamelCase	LoadingIndicator
Socket	Evento	nomecontesto:nomeFunzione	lobby:updateList
Generale	Interfaccia	UpperCamelCase	ScoreboardField
Generale	Funzione	lowerCamelCase	addVote
Generale	Costante	SCREAMING snake_case	MAX_MOVIES
Generale	Variabile normale	lowerCamelCase	standardizedMovies
Generale	Proprietà di oggetti	lowerCamelCase	socketId

Voglio anche ricordare che in JavaScript e TypeScript si usa spesso la keyword const anche per variabili normali. Inoltre, gli oggetti dichiarati con const sono comunque internamente mutabili. Di conseguenza, queste variabili seguono la convenzione delle variabili normali e non delle costanti. La notazione delle costanti è utilizzata per variabili realmente costanti, come le configurazioni.

5.3 Getting started

Sia backend che frontend necessitano di un piccolo processo di setup per cominciare a sviluppare o eseguire un deploy. In questo capitolo saranno mostrati i passaggi necessari da seguire

5.3.1 Backend

Prima di tutto è necessario avere sulla propria macchina Node.js, npm e MongoDB. MongoDB è opzionale se si vuole utilizzare un altro server o servizio di hosting apposito, come MongoDB. Una volta fatto il clone della repository è necessario entrare nella cartella 5_Sito_o_applicativo/api e creare il file .env. Questo file contiene le configurazioni d'ambiente ed è essenziale per il funzionamento del backend. Per il momento (maggio 2024) .env richiede 8 variabili:

- PORT
 - Porta sulla quale offrire il server socket
- MONGO_URI
 - Indirizzo di connessione al database MongoDB
- TMDB_BASE_URL
 - Indirizzo di base dell'API TMDB **con** / finale (<https://api.themoviedb.org/3/>)
- TMDB_API_KEY
 - Chiave per l'API di TMDB, ottenibile qui: <https://www.themoviedb.org/signup>
- TMDB_IMAGE_URL
 - Indirizzo di base per le immagini di TMDB **senza** / finale (<https://image.tmdb.org/t/p/w500>)
- MAX_MOVIES
 - Numero massimo di film per votazione
- PAGES_REQUESTED
 - Numero di pagine richieste per lista
- CODE_LENGTH
 - Lunghezza del codice della lobby

Ora installare le dipendenze con il comando `npm i`. Una volta installate le dipendenze, per utilizzare il server in fase di sviluppo eseguire `npm run dev`, altrimenti per ambiente produttivo utilizzare `npm run build` e `npm run start`. Per un ambiente produttivo, è importante considerare l'utilizzo di strumenti come pm2 o simili per gestire in modo ottimale i processi. Questi strumenti consentono il monitoraggio dei processi, il load balancing e garantiscono che il server rimanga sempre disponibile e accessibile agli utenti.

5.3.2 Frontend

Per utilizzare il frontend per lo sviluppo, dopo aver clonato la repository, è necessario entrare nella cartella 5_Sito_o_applicativo/movie-survey. Anche qui è necessario creare il file .env contenente solo una variabile, ovvero l'URL del server socket associato alla chiave EXPO_PUBLIC_SOCKET_SERVER_URI. Ora installare le dipendenze con il comando `npm i`. Una volta installate le dipendenze, per utilizzare il client in fase di sviluppo eseguire `npm start`. Altrimenti, per eseguire la build, seguire il prossimo capitolo.

5.3.3 Build e deploy

Una volta che l'app è pronta, è necessario fare il deploy per inserirla all'interno degli store dei dispositivi. Io ho fatto solo un passaggio in meno, ovvero ho fatto il deploy dell'applicazione per ottenere il file apk per installare l'applicazione nativamente su dispositivi Android. Per fare questo andare nel terminale dell'applicazione ed eseguire il comando `npx eas login` ed eseguire il login. Successivamente eseguire `npx eas build:configure`. Qui selezionare Android e questo andrà a creare il file `eas.json`, che andiamo a modificare. Dentro l'oggetto build andiamo ad inserire questi oggetti, rimpiazzando i tre puntini con l'effettivo indirizzo pubblico del server socket:

```
"production": {  
  "env": {  
    "EXPO_PUBLIC_SOCKET_SERVER_URI": "..."  
  }  
},  
"test": {  
  "env": {  
    "EXPO_PUBLIC_SOCKET_SERVER_URI": "..."  
  }  
}
```

Questo serve a definire la variabile definita precedente nel file `.env` dell'app, anche nell'applicativo buildato. Una volta fatto questo eseguire il comando `npx eas build` e anche qui selezionare Android. Ora è necessario aspettare, quindi nel mentre aprire il link dato a terminale per controllare il progresso della build.

Una volta completata la build, scaricare il file. Sarà scaricato un file `.aab` (Android App Bundle) quindi ci vuole ancora qualche passaggio per ottenere l'APK. Portare il file in `7_Allegati/bundletool` e dopo aver scaricato il bundletool dal collegamento nella cartella, aprire un terminale in questa posizione. Ora eseguire il comando

```
java -jar bundletool.jar build-apks --bundle=<nomefile>.aab --output=moviesurvey.apks -  
-mode=universal
```

Questo procedimento genera un file chiamato `moviesurvey.apks`, che è un archivio. Per estrarre il contenuto, utilizziamo il nostro tool preferito per l'estrazione dei file e apriamo `moviesurvey.apks`. Questo creerà una cartella denominata `moviesurvey`, all'interno della quale troveremo `universal.apk`, il file APK. A questo punto, è possibile rinominarlo a piacimento e trasferirlo sul dispositivo Android di test per l'installazione. Una volta scaricato il file APK, va aperto, consentita l'installazione dell'app e sarà quindi trovabile l'applicazione installata sul dispositivo.

5.4 Backend

Il backend di questo applicativo è composto da un database MongoDB e un server socket scritto con Node.js e il framework Socket.io. Il backend è scritto in TypeScript.

5.4.1 Creazione progetto

Per creare un progetto Node.js ho eseguito il comando `npm init` e ho risposto alle varie domande per creare il corretto `package.json`.

Una volta creato `package.json` ho creato il file `tsconfig.json`, il file di configurazione di TypeScript. Successivamente ho creato la cartella `src` e al suo interno ho creato il file root del server, `index.ts`.

Per installare una dipendenza ho utilizzato il comando `npm install <nomepacchetto>`.

Successivamente sono andato a modificare il file `package.json` per utilizzare `nodemon` durante lo sviluppo e adattare il pacchetto a TypeScript.

5.4.2 Struttura cartelle

Al fine di mantenere una struttura pulita, ogni parte dell'applicazione è separata in cartelle e sottocartelle nominate in modo semantico.

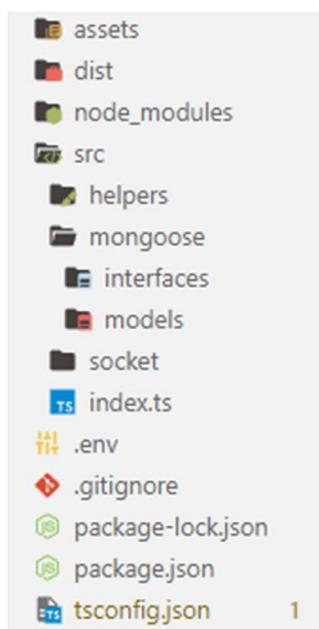


Figura 19 - Struttura cartelle backend

- **assets:** Contiene gli asset utilizzati, nel mio caso il file JSON contenente i modelli di lista
- **dist:** Contiene i file traspilati da TS a JS
- **node_modules:**
 - Dipendenze del progetto, generate automaticamente da npm
- **src:**
 - **helpers:** Contiene funzioni di utilità
 - **mongoose:** Contiene modelli e logica legati al database
 - **interfaces:** Interfacce TS dei modelli
 - **models:** Gli effettivi modelli
 - **socket:** Contiene logica e funzioni legate ai socket
 - **index.ts:** file root
- **.env:**
 - Contiene variabili d'ambiente, tipicamente sensibili
- **.gitignore:**
 - Specifica file e cartelle ignorati da Git
- **package.json e package-lock.json:**
 - Contengono informazioni sul progetto e le dipendenze, con `package-lock.json` che registra versioni precise delle dipendenze
- **tsconfig.json:**
 - Contiene le configurazioni di typescript.

5.4.3 Versioni database

Ci sono state 3 versioni principali del database. In questo capitolo saranno analizzate queste versioni e spiegato il motivo dei cambiamenti.

5.4.3.1 Versione 1

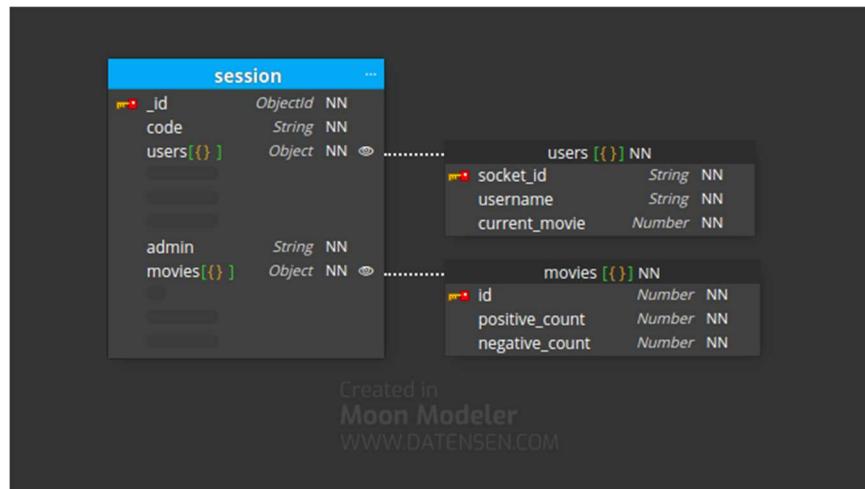


Figura 20 - DB Versione 1

La prima versione del database contiene solo una collezione ed è stata progettata con l'anticipazione di futuri cambiamenti. Questa versione ha gettato comunque una base stabile di quella che sarà la collezione Lobby.

5.4.3.2 Versione 2

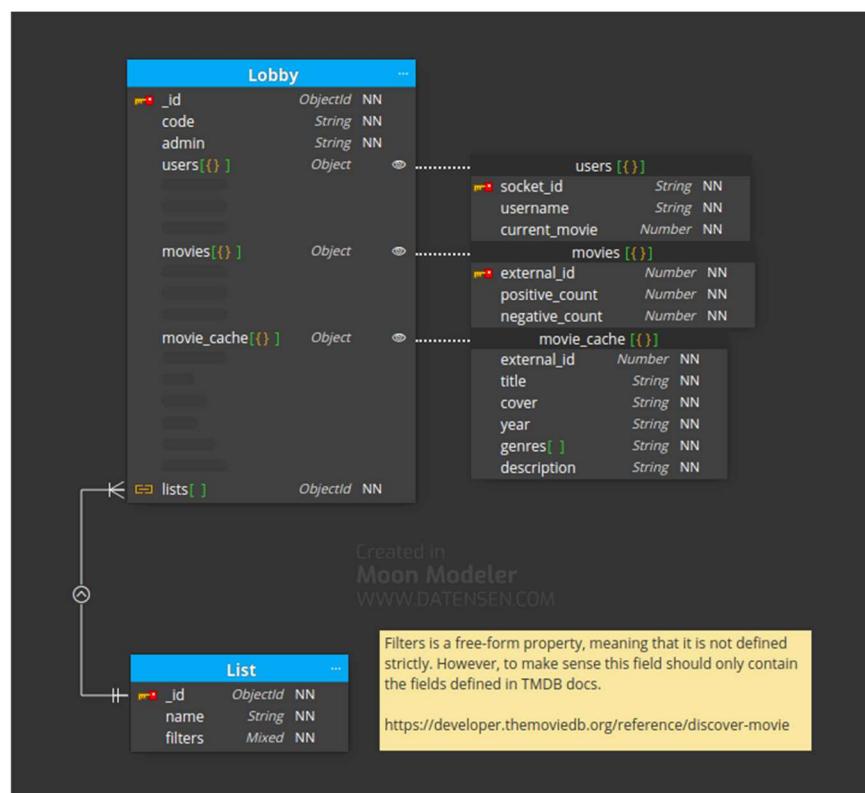


Figura 21 - DB Versione 2

La versione 2 del database porta molti cambiamenti necessari alla versione 1. La collezione principale viene rinominata a Lobby e vede l'aggiunta di due campi, ovvero movie_cache e lists. La proprietà movie_cache ha la funzione di, come descrive il nome, storage a modi cache per le informazioni dei film, al fine di evitare numerose richieste all'API pubblica. La proprietà lists invece contiene gli ObjectId delle liste selezionate dall'amministratore. Un altro grande cambiamento all'interno del database è infatti l'aggiunta della collezione List, contenente le liste messe a disposizione all'amministratore. List contiene al suo interno solo il suo nome e i filtri da applicare alla richiesta da mandare all'API pubblica. La lista contiene solo i filtri invece che i film di per sé per rendere più dinamica e pertinente la selezione dei film. La proprietà filters della lista è Mixed, ovvero un oggetto libero. Questo vuol dire che è possibile mettere qualsiasi tipo di proprietà al suo interno. Questo è stato fatto perché altrimenti si sarebbe dovuto specificare tutti i possibili filtri offerti da TMDB. Logicamente, però, alla creazione di un'istanza di List si vuole far ben attenzione che l'oggetto sia conforme alle regole impostate dall'API.

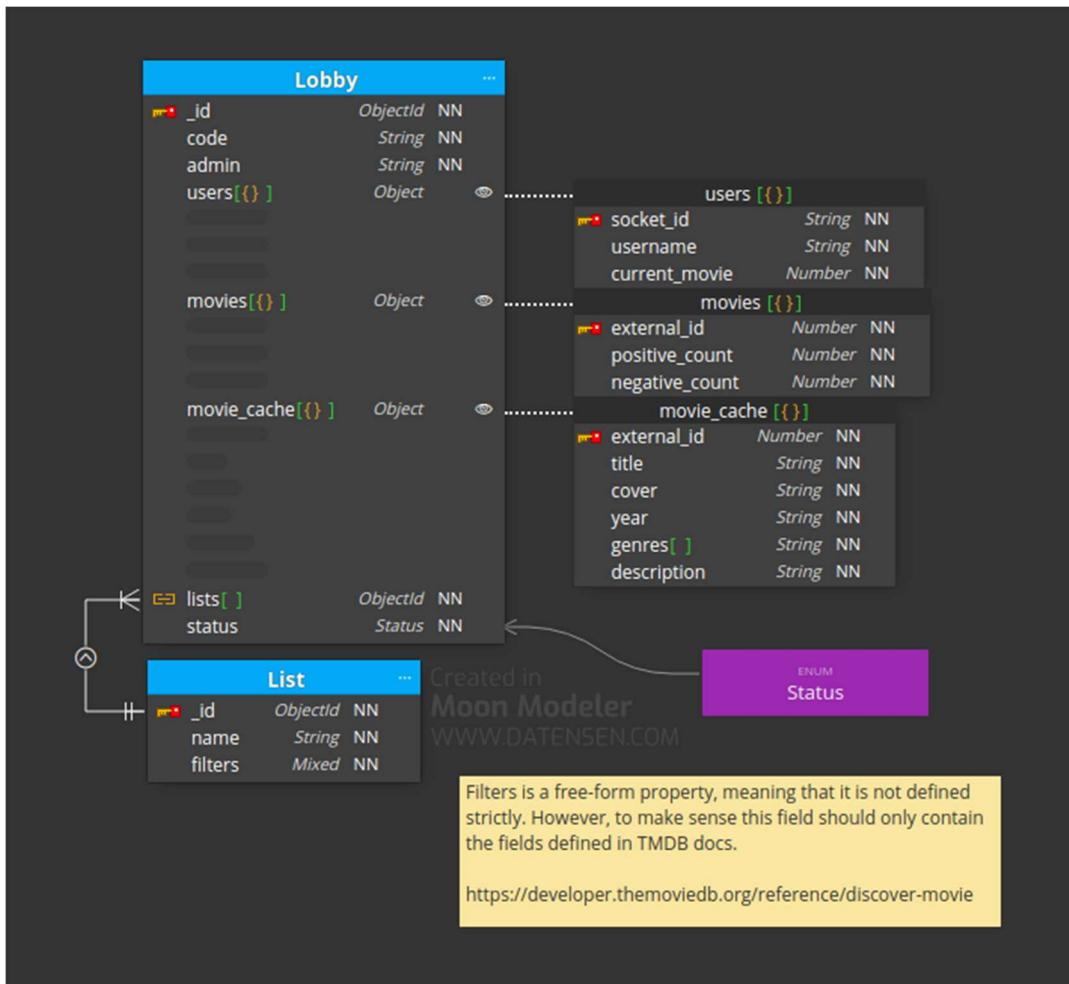


Figura 22 - DB Versione 3

La versione 3 del DB non vede tanti cambiamenti rispetto alla versione 2. I cambiamenti essenzialmente sono 2: l'aggiunta di status e il ruolo di identificatore dato a List.movie_cache.external_id. La proprietà status indica lo stato nella quale la lobby si trova ed è definito da un enum con i seguenti possibili valori:

- **waiting**
 - Stato iniziale
- **started**
 - Stato impostato una volta iniziata la votazione e mantenuto fino al completamento da parte di tutti gli utenti di questa
- **scoreboard**
 - Stato impostato una volta completata la votazione da parte di tutti gli utenti

5.4.4 Listener

La logica dei socket si basa su listener ed emitter, ovvero ascoltatori ed emettitori. L'ascoltatore di più alto livello è connection, indicante che è in arrivo una connessione. All'interno dell'handler di questa connessione vengono dichiarati gli altri listener. Per mantenere la struttura ordinata ho deciso di applicare una separazione simile a quella tipica tra router e controller.

```
io.on("connection", (socket: Socket) => {
    console.log("Connected", socket.id);
    socket.on("lobby:join", lobby.join(socket));
    socket.on("lobby:quit", lobby.quit(socket));
    socket.on("lobby:create", lobby.create(socket));
    socket.on("lobby:start", lobby.start(socket));
    socket.on("lobby:updateList", lobby.updateList(socket));
    socket.on("lobby:fetchLists", lobby.fetchLists(socket));
    socket.on("lobby:startNewIteration", lobby.startNewIteration(socket));

    socket.on("vote:vote", vote.vote(socket));

    socket.on("movie:trailer", movie.trailer(socket));

    socket.on("disconnect", () => {
        lobby.quit(socket)();
        console.log("Disconnected", socket.id);
    });
});
```

Qui è possibile vedere la dichiarazione dei listener pulita, senza nessuna implementazione vera e propria, dato che questa viene delegata ai controller. Oltre ai listener documentati nel capitolo 4.1, possiamo vedere che è presente anche un listener all'evento disconnect. Questo evento è mandato in automatico in caso di disconnessione del client (perdita di connessione, chiusura dell'applicazione, ...) e io ho deciso di gestirlo tramite il controller lobby.quit, per far in modo che in caso di chiusura della connessione, le azioni eseguite siano le stesse di quelle applicate quando si esce da una lobby.

5.4.5 Controller

Le librerie di controller sono 2: lobby e vote. Ogni libreria offre funzioni per gestire un determinato evento ascoltato. Andando a vedere più nel dettaglio una funzione:

```
create: (socket: Socket) => async ({ username }: CreateParams) => {
    ...
},
```

Andando ad analizzare maggiormente questa funzione possiamo vedere che si tratta di una funzione che prende un socket come parametro, che ritorna a sua volta una funzione che prende come parametro un oggetto contenente la proprietà username. Il richiamo di lobby.create(socket) va infatti semplicemente a ritornare una funzione a sua volta, che gestisce l'evento prendendo l'oggetto con username come parametro. Grazie a questa matroska di funzioni è possibile tenere pulita la codebase e allo stesso momento avere anche il socket all'interno del gestore, che è essenziale.

5.4.6 Emitter

Nel contesto di questo backend ho utilizzato l'emit in due modi differenti. Il primo, più intuitivo, è l'emit al socket che ha interpellato il listener. Questo si può fare semplicemente tramite la funzione emit del socket ottenuto come parametro. Il secondo è invece l'emit verso una room. Socket.io mette a disposizione delle room, ovvero dei sistemi per gestire degli utenti in bulk, mandando direttamente un messaggio in broadcast verso tutti i membri di quella stanza. Questo mi risulta molto utile nel contesto della lobby dove, a volte, un messaggio va emesso verso tutti i membri della lobby. Questo è fattibile in due passaggi:

Inserire il socket all'interno della room, identificata in questo caso dal codice della lobby.

```
socket.join(lobbyCode);
```

Successivamente, quando si vuole mandare la richiesta all'intera room si può fare nel seguente modo.

```
io.to(room).emit("<nome>", info);
```

È da notare che le stanze sono un concetto strettamente server-side, quindi un client non può mai sapere, a meno che non gli sia esplicitato, di che room faccia parte.

Un socket può far parte di tante room allo stesso momento, ma per l'implementazione di questo progetto questa funzionalità non è stata utilizzata.

5.4.7 Codice particolare

5.4.7.1 ScoreboardStorage

Come documentato in modo dettagliato nel diario del 03.05.2024, c'è stato un processo decisionale prima di decidere di adottare uno storage basato sulla RAM per la scoreboard. In questo capitolo spiegherò come questo storage è stato implementato.

Lo storage è essenzialmente molto semplice, si tratta di una mappa con, come chiave, l'ObjectId della lobby sotto forma di stringa e come valore la scoreboard. Vengono messi a disposizione 3 funzioni per interagirci, ovvero getScoreboard, setScoreboard e deleteScoreboard.

```
const scoreboardMap = new Map<string, Array<ScoreboardField>>();
```

La mappa viene salvata nello spazio riservato ai moduli di Node.js. Nel dettaglio la mappa di per sé risiede nell'heap della memoria, mentre il suo riferimento all'interno dello stack.

5.4.7.2 LobbyCodeGenerator

Ogni lobby ha un proprio codice a 4 cifre da utilizzare per far entrare gli utenti. Questo codice deve essere univoco e allo stesso tempo anche semplice. L'idea iniziale era un codice numerico di 4 cifre offre 9000 combinazioni possibili (1000-9999), che sono abbastanza. Il codice al momento è generato in un modo molto banale, ovvero si genera un codice randomico e si controlla che non sia già esistente all'interno del database. Per questo motivo ho deciso di implementare già un cambiamento da numerico ad alfanumerico, mantenendo sempre 4 cifre, passando così a $(26+10)^4 = 1'679'616$ possibili combinazioni senza tanti problemi, dato che il campo è già trattato come stringa.

```
const CODE_LENGTH = parseInt(process.env.CODE_LENGTH ?? "4");
const randomChar = () => {
    let char = Math.floor(Math.random() * 36) + 48;
    if (char > 57) {
        char += 7;
    }
    return String.fromCharCode(char);
}

const generateLobbyCode = async () => {
    let code = '';
    do {
        for (let i = 0; i < CODE_LENGTH; i++) { code += randomChar(); }
    } while (!!(await (Lobby.findOne({ code }))))
    return code;
};
```

Se i codici non dovessero bastare o ci fossero dei problemi di concorrenza, sarebbe possibile comunque facilmente implementare un codice più lungo tramite la variabile, oppure generare un UUID e dare la possibilità di invitare tramite link o codice QR con un deep link.

5.4.7.3 MovieFetcher

L'ottenimento dei dati dei film da parte del backend verso l'API pubblica è una parte essenziale di questo progetto. Per questo motivo è stato implementato un helper apposta per questa funzione. La libreria helper situata nel file movieFetcher.ts offre la funzione addMoviesToCache che prende come parametro soltanto l'ObjectId della lobby e si occupa di inserire all'interno del database tutte le informazioni necessarie dei film nel formato corretto. Internamente le funzioni sono 4, ovvero: addMoviesToCache, fetchMovies, fetchGenres e getGenres.

La funzione fetchMovies si occupa di fare l'effettiva richiesta dei film verso l'API pubblica utilizzando Axios. La funzione getGenres, invece, si occupa di, partendo dall'array di id dei generi restituita dall'API, ritornare un array di stringhe contenente l'effettivo nome del genere. Questa funzione utilizza al suo interno fetchGenres, che si occupa di eseguire la richiesta all'API per ottenere la mappatura tra id e nome del genere. Queste funzioni sono quindi utilizzate all'interno di addMoviesToCache per ottenere i dati dei film necessari. Le richieste per ottenere i film sono eseguite tramite la route /discover/movie dell'API, con i filtri inseriti come parametri. Questo permette di ottenere sempre una lista aggiornata.

La funzione fetchMovies contiene in realtà al suo interno la dichiarazione di un'altra funzione, ovvero fetchPage. Essenzialmente, l'API di TMDB funziona con un sistema a pagine e ogni pagina contiene 20 film per pagina. Per questo motivo ho deciso di implementare una logica per fare un fetch di N pagine, al fine di avere un numero appropriato di film da votare. Di default questo valore è 2 ed è definito tramite la variabile dotenv PAGES_REQUESTED, quindi viene fatto un fetch di 40 film per filtro selezionato. Ci sono però delle eccezioni dove dei film sono scartati, questi film sono quelli senza una copertina, senza dei generi definiti, senza una data di rilascio definita o senza una descrizione. Per questo motivo per ogni lista non si ottengono esattamente 40 film. Inoltre viene applicata anche una logica aggiuntiva di mescolamento e limitazione dei

film al fine di limitare i film votabili in una volta sola a un numero che non sia eccessivo. Di default questo valore è 50 ed è definito tramite la variabile dotenv MAX_MOVIES. Un altro criterio di filtraggio dei film, questa volta però eseguibile direttamente sulla richiesta all'API, è che il film sia uscito nel passato. Questo filtro è necessario in quanto l'API contiene anche dei film che devono ancora essere rilasciati ed hanno già una data di rilascio fissa, che è però nel futuro. Questo è fattibile tramite il seguente parametro nella richiesta:

```
'primary_release_date.lte': new Date().toISOString().split('T')[0]
```

La funzione fetchGenres contiene due richieste verso l'API, una verso /genre/movie/list e una verso /genre/tv/list, successivamente i risultati di queste richieste, che sono entrambi essenzialmente delle mappe id-nome vengono combinati in una singola mappa e ritornata. Il risultato ritornato viene quindi tenuto in memoria per velocizzare il processo per le prossime volte dove sarà necessario fare un lookup del nome del genere basandosi sul suo id.

5.4.7.4 Comunicazione vote:start e vote:next

Come descritto nei diagrammi, il server invia un evento vote:start quando la votazione viene avviata. Inizialmente, l'evento vote:start conteneva le informazioni di un film e, una volta completata la votazione, venivano inviati i vote:next. Tuttavia, per motivi di preloading, come descritto nel diario del 15 maggio 2024, questa logica è stata modificata. Ora, l'evento vote:start include le informazioni di un film e immediatamente dopo viene inviato anche un vote:next per pre-caricare l'immagine. In questo modo, mentre si vota un film, l'immagine del film successivo viene già caricata, migliorando l'esperienza utente anche con una connessione internet lenta.

Questa modifica ha comportato due correzioni aggiuntive:

1. Invio dell'ultimo vote:next: Applicando la nuova logica, dopo aver votato il penultimo film, il client si aspetta ancora un vote:next per aggiornare le informazioni del film. Tuttavia, non ci sono più film da inviare. Per risolvere questo problema, ho deciso di non inviare niente al posto del film quando si tratta dell'ultimo film, permettendo al client di intercettare questa eccezione gestire correttamente il passaggio delle informazioni senza problemi.
2. Delay necessario tra vote:start e vote:next: Dopo un vote:start, il server invia immediatamente anche un vote:next. Tuttavia, nella pratica è stato aggiunto un delay di 1 secondo tra questi due messaggi. Questo perché, una volta inviato il vote:start, a livello di applicazione viene effettuato un redirect alla pagina di votazione. Il vote:next, invece, non è gestito a livello di applicazione ma a livello di pagina. Pertanto, se vote:start e vote:next vengono inviati senza delay, la pagina di votazione potrebbe non essere ancora caricata completamente e l'evento potrebbe non essere catturato, poiché il relativo listener non è ancora stato dichiarato.

Queste correzioni garantiscono un funzionamento fluido e una migliore esperienza utente, anche in condizioni di rete meno ottimali.¹

```
else if (moviesLeftToVote === 1) {
  socket.emit("vote:next");
} else {
  //+1 means the next movie, but we want the one after that
  const movie = lobby.movie_cache[index + 2];
  socket.emit("vote:next", { movie });
}
```

```
socket.emit("vote:start", { movie: lobby.movie_cache[0] });
await new Promise((resolve) => setTimeout(resolve, 1000));
socket.emit("vote:next", { movie: lobby.movie_cache[1] });
```

¹ Questo testo è stato parzialmente generato da un'IA partendo da un testo scritto a mano e successivamente revisionato e corretto manualmente.

5.5 Frontend

Il frontend di questo applicativo è creato con React Native ed Expo per permettere uno sviluppo semplice anche delle funzionalità native. Per il routing dell'applicazione ho utilizzato Expo Router, che semplifica molto la navigazione all'interno dell'applicazione. Expo Router è una libreria di routing open-source per applicazioni React Native costruite con Expo. Expo Router è un router basato su file che permette di gestire la navigazione tra le schermate nella tua app, consentendo agli sviluppatori di programmare la navigazione tra pagine più facilmente. Quando un file viene aggiunto alla directory dell'app, il file diventa automaticamente un percorso nel tuo sistema di navigazione.

5.5.1 Struttura cartelle

Sono presenti diverse cartelle all'interno del progetto, quindi saranno spiegate solo quelle più essenziali:

- **app:** Cartella contenente tutte le view
 - **(components):** Componenti riutilizzati all'interno del progetto
 - **home:** Pagina home e relative sotto pagine e componenti
 - **lobby:** Pagina della lobby e relative sotto pagine e componenti
 - **vote:** Pagina del voto e relative sotto pagine e componenti
- **assets:** Asset come immagini salvate localmente
- **constants:** Moduli per costanti
- **language:** Moduli legati al supporto multilingua
 - **packs:** Pacchetti di lingua
- **socket:** Moduli per gestione dei socket
 - **listeners:** Listener dei socket a livello dell'intero applicativo
- **storage:** Moduli per salvataggio RAM/Disco di dati a livello dell'intero applicativo

5.5.2 Socket

Per interagire con il server socket, è necessario effettuare una connessione verso il server. Questa operazione è fatta all'interno di `socket/index.ts`:

```
const socket = io(process.env.EXPO_PUBLIC_SOCKET_SERVER_URI || "");
```

Una volta effettuata la connessione è quindi possibile aggiungere all'oggetto `socket` dei listener. Nello stesso file ho quindi aggiunto dei listener che necessitano di essere in ascolto su tutta l'applicazione e non solo su una pagina specifica. Un listener nativo in più è quello per l'evento `connect_error`, un evento che viene simulato dal client nel caso in cui non sia possibile raggiungere il server. Ho deciso, al fine di mantenere una codebase più pulita, di separare i listener e i "controller" degli eventi, quindi possiamo trovare i gestori degli eventi dentro la cartella `listeners`. Queste funzioni, generalmente, si occupano di mostrare messaggi ed effettuare dei redirect sul client. Per esempio il gestore dell'evento `lobby:joined`, quando questo evento arriva, si occupa di mandare l'utente sulla pagina della lobby nella quale ha mandato la richiesta di entrare, assicurandosi che abbia anche la corretta lista di utenti presenti nella lobby e altre informazioni essenziali.

Gli eventi possono avere più funzioni associate, come nel caso di `lobby:update`, che è sia ascoltato a livello dell'intera applicazione (per gestire il toast) che a livello della pagina della lobby (per aggiornare la lista di utenti). Come il server, anche il client emette degli eventi. Questi eventi sono però dichiarati soltanto all'interno delle pagine e non a livello dell'intera applicazione. La gran parte degli emit eseguiti dal client sono classici, ovvero viene emesso un evento con eventuali dati e non si attende nulla in risposta. L'unica eccezione a questo caso è l'evento `lobby:fetchLists` che funziona tramite un callback. Essenzialmente il client emette l'evento e passa come parametro una funzione richiedente un parametro, così che il server quando riceve questa richiesta possa passare indietro il parametro necessario per eseguire la funzione. A livello pratico è molto più semplice da comprendere: `lobby:fetchList` serve al client per richiedere le liste di film disponibili, quindi il server gli ritorna direttamente la lista in modo da eseguire il callback con i dati. Questa procedura essenzialmente simula una richiesta HTTP GET classica tramite i socket.

5.5.3 Salvataggio locale

I moduli di storage sono 4, ovvero: isAdmin, roomCode, username e users e sono differenziabili in due categorie: storage temporaneo in memoria e storage persistente sul disco. I primi 2 moduli fungono essenzialmente da variabili globali dove i dati vengono salvati nel modulo in modo da essere raggiungibili da qualsiasi componente all'interno dell'app. I dati sono tenuti in memoria tramite delle semplici variabili e di conseguenza sono accessibili in modo sincrono. Il modulo username fa invece parte della seconda categoria in quanto salva i dati nel disco tramite AsyncStorage invece che mantenerli in memoria. Il modulo users invece internamente salva in memoria gli utenti presenti nella lobby ma li espone direttamente; espone infatti solo la funzione getUserDifference e la funzione deleteUsers, utilizzate dal controller di lobby:update per visualizzare un toast con un messaggio indicante <username> left o <username> joined ed eliminare questa lista una volta distrutta la lobby.

5.5.4 Gestione lingua

Nonostante non fosse richiesta esplicitamente, per tenere il codice più pulito e facilitare eventuali modifiche nel futuro, ho già implementato un sistema per la gestione delle stringhe tramite la libreria i18n. Al momento è presente soltanto la lingua inglese ma grazie a questa struttura basterebbe solo aggiungere un file JSON in un'altra lingua e fare delle piccole modifiche alla configurazione di i18n per aggiungere nuove lingue. Nell'applicazione, quando si vuole utilizzare del testo si importa t da useTranslation e si richiama passando la chiave del testo, che deve essere associata ad un attributo all'interno del file JSON della lingua. Se la richiesta della stringa è fatta all'esterno di un componente dove non è permesso l'utilizzo di hook, si usa la stessa funzione, richiamata però direttamente dall'oggetto i18n, quindi: i18n.t.

```
const { t } = useTranslation();
...
<Text>{t('LOBBY_DESTROYED')}</Text>
...

const text = i18n.t('LOBBY_DESTROYED');

{
  "LOBBY_DESTROYED": "The lobby has been destroyed",
  ...
}
```

5.5.5 Costanti

Per centralizzare la gestione delle costanti, ho creato la cartella "constants" contenente diversi file dedicati a gestire costanti in vari contesti, ovvero:

- Colors.ts: Responsabile per la gestione dei colori
- FontSize.ts Responsabile per la gestione della dimensione dei font

Questa organizzazione consente una chiara suddivisione delle costanti in base al loro contesto d'uso, facilitando così la manutenzione e comprensione del codice.

5.5.6 Font

I font utilizzati all'interno dell'applicativo sono 2: Poppins e FiraMono. Poppins è utilizzato essenzialmente per tutto, tranne per la visualizzazione della lobby, dove ho ritenuto fosse appropriato un font monospace simile a Poppins, quindi FiraMono. I font sono caricati da @expo-google-fonts e caricati nel RootLayout tramite Font.loadAsync di expo-font, in modo da essere disponibili in qualsiasi parte dell'applicazione.

5.5.7 Blocco portrait

L'applicazione è stata progettata e sviluppata per essere utilizzata esclusivamente in modalità portrait, ovvero con il dispositivo in posizione verticale. Per garantire che gli utenti rispettino questa regola e evitare eventuali bug grafici, sono state adottate le seguenti misure:

All'interno del componente RootLayout, è stata aggiunta un'operazione all'interno di un useEffect:

```
ScreenOrientation.lockAsync(ScreenOrientation.OrientationLock.PORTRAIT_UP);
```

Questo assicura che anche se l'utente tenta di ruotare lo schermo, sarà bloccato nell'utilizzo della modalità PORTRAIT_UP, ossia con il dispositivo in posizione verticale, con la fotocamera in alto e i controlli in basso.

Inoltre, nel file app.json, è stata specificata la proprietà expo.orientation con valore "portrait", per indicare che l'applicazione deve essere visualizzata esclusivamente in modalità portrait.

È stato anche utilizzato il plugin expo-screen-orientation per garantire che, quando l'applicazione viene avviata, sia sempre aperta in modalità PORTRAIT_UP.

Queste misure assicurano una coerente e corretta esperienza utente, evitando problemi grafici e garantendo che l'applicazione venga utilizzata secondo le specifiche di progettazione.

5.5.8 Toast

L'applicazione si basa completamente sul socket per la gestione delle sessioni e delle votazioni. Per questo motivo in caso di problemi di connessione o in caso di errori mandati dal server, è necessario informare l'utente. Per implementare questa funzionalità ho utilizzato la libreria react-native-toast-message. Questa libreria permette di far comparire un messaggio temporaneo sullo schermo dell'utente in qualsiasi momento o pagina sia necessario. La libreria offre 3 tipi di toast, ovvero error, info e success. I toast di errore sono mostrati quando si riceve un evento lobby:error o vote:error, mentre quelli di tipo info sono soltanto due che vengono mostrati quando una lobby viene distrutta o quando un utente esce o entra in una lobby. Quello di successo è soltanto uno e viene mostrato in caso di riconnessione al server socket dopo un problema di rete che aveva fatto perdere la connessione. Al momento i toast non applicano completamente il supporto multilingua, dato che in caso di lobby:error o vote:error viene mostrato direttamente il messaggio (in inglese) inviato dal server. Questo sarebbe facilmente aggiornabile facendo in modo che il server invii al client una chiave presente nei file di lingua invece che il vero e proprio testo, oppure richiedendo la lingua del client quando viene connesso e fare una traduzione server-side.



Figura 23 - Toast Errore

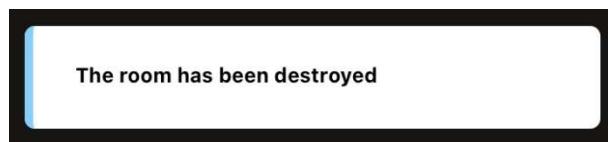


Figura 24 - Toast Informazione



Figura 25 - Toast Successo

5.5.9 Componenti particolari

5.5.9.1 LoadingIndicator

Quando una pagina è in fase di caricamento, è fondamentale evitare di lasciare l'utente con una schermata statica, e a questo scopo entra in gioco il `LoadingIndicator`. Il componente in sé non è complesso; è semplicemente una `MainView` che incorpora un `ActivityIndicator` (un cerchio che indica il caricamento). `LoadingIndicator` viene reso al posto della pagina di destinazione mentre si attende che tutti i dati siano caricati. Ecco un esempio semplificato di utilizzo.

```
const RootLayout = () => {
  const [fontsLoaded, setFontsLoaded] = useState(false);
  useEffect(() => {
    const loadFonts = async () => {
      await Font.loadAsync(...);
      setFontsLoaded(true);
    }
    loadFonts();
  }, []);
  if (!fontsLoaded) {
    return <LoadingIndicator />
  }
  return (
    // Loaded page
  );
};
```

Essenzialmente, quando la variabile `fontsLoaded` è impostata su `false`, mostriamo il `LoadingIndicator`; quando, invece, è impostato su `true`, sappiamo che i font sono stati caricati e quindi la pagina effettiva con i font caricati prende il suo posto.

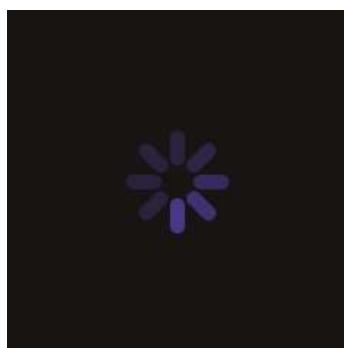


Figura 26 - LoadingIndicator iOS

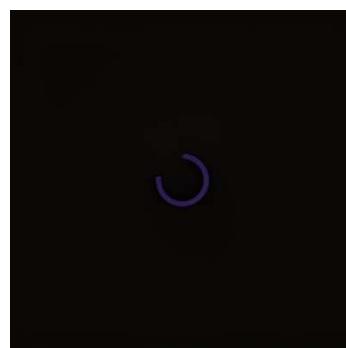


Figura 27 - LoadingIndicator Android

5.5.9.2 SText

Il testo viene mostrato in diverse occasioni nell'applicazione e, avendo un font personalizzato e un tema scuro, è risultato estremamente comodo avere un componente di testo personalizzato. Questo componente è veramente molto semplice, si tratta infatti di un semplice Text con degli stili già impostati, ovvero il font, il colore e la dimensione del font. Per garantire la flessibilità di questo componente, tutto è sovrascrivibile ed è possibile utilizzare tutte le proprietà native di Text.

5.5.9.3 Button

Nell'applicazione, i pulsanti sono utilizzati molto spesso, quindi ha senso creare un componente che abbia questo ruolo ed applichi automaticamente tutti gli stili necessari, senza dover riscrivere lo stesso codice in diverse parti. Il componente Button è molto semplice, si tratta infatti soltanto di un Pressable con al suo interno un SText e un elemento per permettere l'animazione di feedback. Ho creato questo componente con l'idea di mantenerlo in più semplice e intuitivo possibile, quindi richiede soltanto 5 parametri, tutti opzionali:

- **type**: Tipo di bottone, che può essere primary, secondary o other. Questo determina il colore del bottone, gestito internamente. Di default il valore è primary.
- **text**: Il testo da inserire all'interno del bottone.
- **color**: Un eventuale colore in caso si volesse sovrascrivere quello definito da type.
- **onPress**: La funzione eseguita quando il pulsante viene premuto.
- **enabled**: Variabile booleana indicante se il pulsante è abilitato o meno. Nel caso in cui il bottone sia disabilitato, gli viene dato un altro colore e viene tolta la possibilità di premerlo.

Per l'animazione di feedback ho utilizzato la libreria Animated di react-native, che permette di eseguire animazioni personalizzate utilizzando i driver nativi del dispositivo.

5.5.9.4 SModal

Come per SText, anche SModal nasce dalla necessità di personalizzare un componente nativo. SModal è infatti la versione personalizzata del componente Modal. I cambiamenti dal componente nativo sono sia estetici che funzionali. A livello estetico viene impostata di default il tipo di animazione, il colore e le dimensioni. A livello funzionale, invece, è stata aggiunta la possibilità di chiudere il modal premendo al suo esterno, un'opzione non implementata nel componente nativo ma molto comoda. Questo componente ha solo 3 parametri, ovvero:

- **show**: Variabile booleana indicante se mostrare il modal o meno
- **setShow**: Funzione che modifica show, utilizzata per chiudere il modal
- **children**: I componenti che vanno mostrati all'interno del modal

5.5.9.5 STextInput

STextInput è una versione avanzata di TextInput, progettata per offrire un'esperienza utente migliore e uno stile coerente con l'applicazione. Oltre alle modifiche di stile, include un sistema di conteggio dei caratteri rimanenti, attivato quando è impostato un limite massimo di caratteri. Questo migliora l'UX, dato che l'utente può facilmente vedere quanti caratteri può ancora inserire e capire perché non può aggiungerne altri. Le proprietà di questo componente sono identiche a quelle del classico TextInput, ma vengono intercettati alcuni valori per implementare queste funzionalità aggiuntive.

5.5.9.6 MainView

Per un design consistente si vuole avere, oltre agli stessi componenti, anche lo stesso contenitore generale, ed è questo il ruolo di MainView. MainView si occupa infatti di offrire un contenitore per una pagina con già delle regole impostate in modo da avere un comportamento ed interfacce uniformi. Nello specifico questo componente impone lo sfondo del colore corretto e impone un padding minimo necessario, oltre ad essere costruito tramite SafeAreaView, un componente che si assicura di lasciare un margine appropriato nei dispositivi (per esempio, per dispositivi con una notch evita che del contenuto sia posizionato sotto questa).

5.5.9.7 CodeInput

Il componente CodeInput è progettato per inserire e visualizzare un codice della lobby ed è molto semplice da utilizzare grazie a tre proprietà principali: code, setCode ed enabled. Internamente, però, la logica è più complessa e interessante.

Per visualizzare il codice, viene utilizzata una View per ogni carattere, che contiene a sua volta un elemento SText per il carattere e un'altra View per il trattino sotto il carattere. Devono sempre essere mostrati 4 trattini, quindi ho implementato un sistema che aggiunge spazi vuoti al codice, garantendo la presenza di 4 caratteri e 4 trattini. Questo sistema funziona bene per la visualizzazione del codice, ma non offre funzionalità di inserimento.

Per risolvere questo problema, ho implementato un sistema che utilizza Pressable e riferimenti (refs) per simulare un TextInput. Ecco come funziona:

1. Il blocco che mostra il codice è avvolto in un Pressable, un componente trasparente che esegue una funzione quando viene premuto.
2. Un TextInput invisibile posizionato fuori dallo schermo è presente nella pagina.
3. Quando si preme il Pressable, una funzione mette a fuoco il TextInput, aprendo così la tastiera del dispositivo.
4. Il TextInput ha una funzione onChangeText che sanitizza l'input, rimuovendo tutti i caratteri che non siano alfabetici o numerici e convertendo il testo in maiuscolo. Questo testo viene quindi aggiornato tramite il setter setCode.
5. Un useEffect, con code come dipendenza, aggiorna filledCode, il codice con gli spazi aggiunti, che viene poi mostrato nella vista del codice.

Questo trucco non funziona su web, quindi, nonostante l'applicativo non sia destinato al web, ho aggiunto un TextInput visibile per l'ambiente web, permettendo così l'uso durante lo sviluppo, anche se con uno stile meno accattivante.

In poche parole viene simulato un TextInput utilizzando il Pressable trasparente e un effettivo TextInput fuori dallo schermo.²

5.5.9.8 Card

Il componente Card è stato creato per gestire il contenuto di una carta utilizzata nel sistema di voto dei film, implementato con la libreria react-tinder-card. La carta contiene un'immagine di copertina del film, che occupa lo spazio della carta, e include anche le informazioni relative al film.

Il componente richiede cinque parametri:

- movie: Rappresenta il film da visualizzare.
- trimmedDescription: È una descrizione abbreviata del film da visualizzare nella schermata. La descrizione completa è inclusa nel parametro 'movie' ed è mostrata tramite un modal quando si preme sulla descrizione abbreviata.
- onSwipe: È la funzione da eseguire quando si esegue uno swipe sulla carta, in qualsiasi direzione.
- setShowDescriptionModal: È il setter per il valore booleano che indica se il modal contenente l'intera descrizione del film è aperto. Viene utilizzato per aprire il modal quando necessario.
- cardRef: È il riferimento associato alla carta, che consente al chiamante di interagire direttamente con la carta.

In breve, questo componente offre la carta pronta per essere utilizzata per votare i film.³

² Questo testo è stato parzialmente generato da un'IA partendo da un testo scritto a mano e successivamente revisionato e corretto manualmente.

³ Questo testo è stato parzialmente generato da un'IA e successivamente revisionato e corretto manualmente.

5.5.9.9 CardButton

Il componente CardButton è stato creato per gestire i due pulsanti all'interno di una carta nel sistema di voto dei film. I suoi parametri sono:

- **onPress**: È la funzione da eseguire quando il pulsante viene premuto.
- **direction**: Indica la direzione del pulsante, che può essere left o right.

Il pulsante è rappresentato da un'icona, una X in caso di voto negativo (sinistra) e un cuore in caso di voto positivo (destra). Quando viene premuto, il pulsante esegue un'animazione simile a quella di Button per dare un feedback visivo all'utente. In breve, CardButton è il componente che definisce i pulsanti all'interno delle carte del sistema di voto dei film, consentendo agli utenti di votare tramite dei bottoni oltre che tramite lo swipe della carta.

5.5.10 Risultati interfacce

Come già menzionato nel capitolo 4.3, le interfacce hanno subito alcuni cambiamenti rispetto al loro design originale. In questo capitolo, verranno presentate le interfacce dell'applicazione sia per smartphone che per tablet.

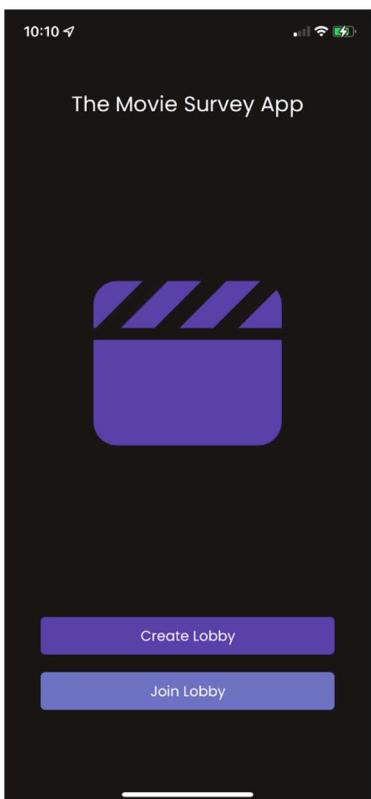


Figura 28 - Schermata Home Smartphone

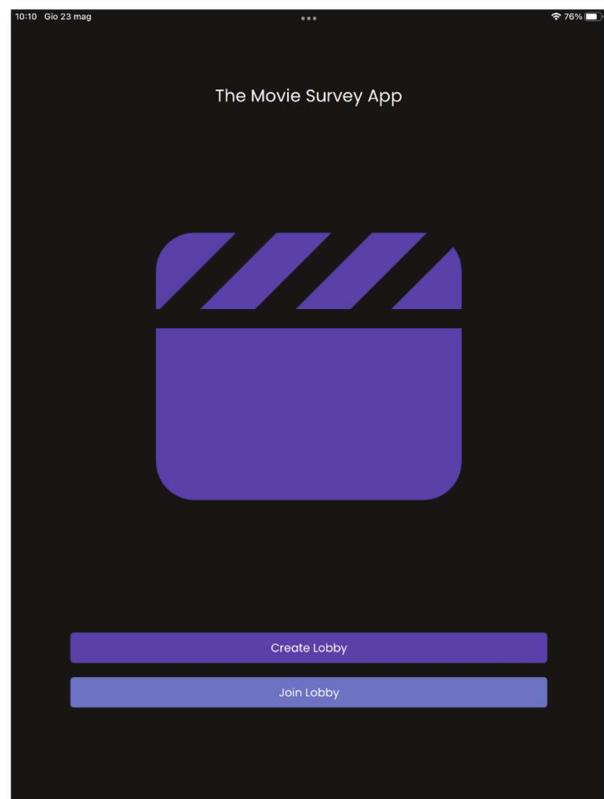


Figura 29 - Schermata Home iPad



Figura 30 - Modal Nome Smartphone

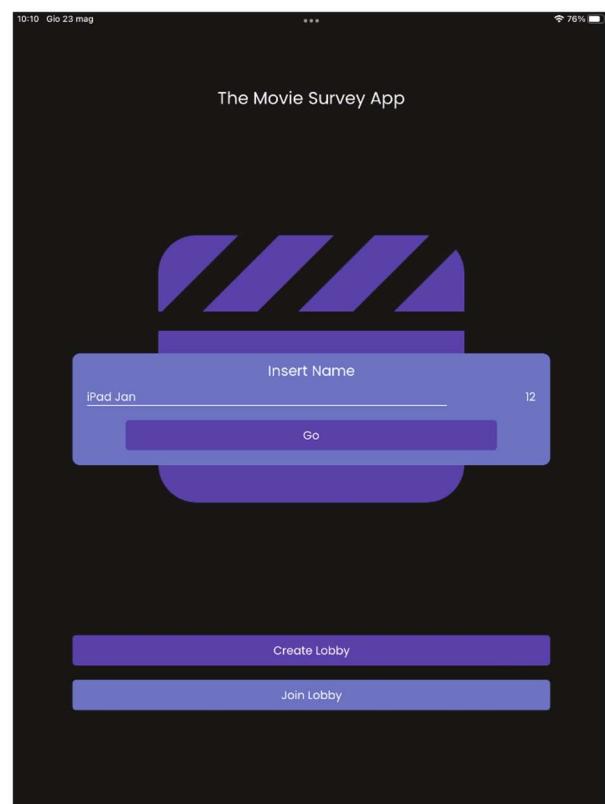


Figura 31 - Modal Nome iPad



Figura 32 - Schermata Lobby Smartphone



Figura 33 - Schermata Lobby iPad

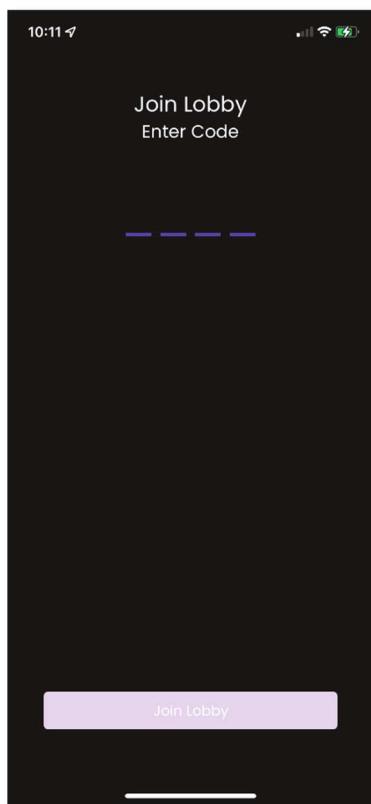


Figura 34 - Schermata Inserimento Codice Smartphone

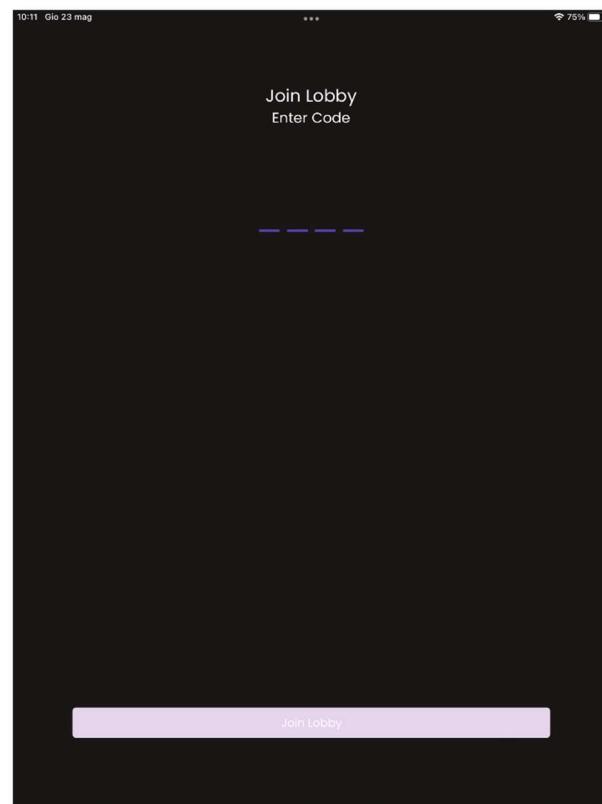


Figura 35 - Schermata Inserimento Codice iPad

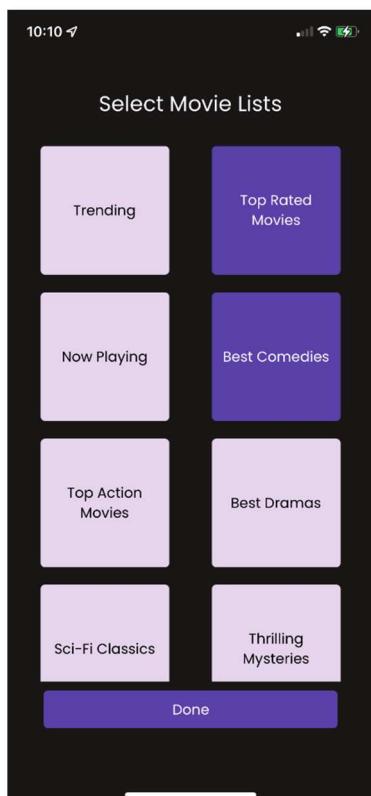


Figura 36 - Schermata Selezione Liste Smartphone

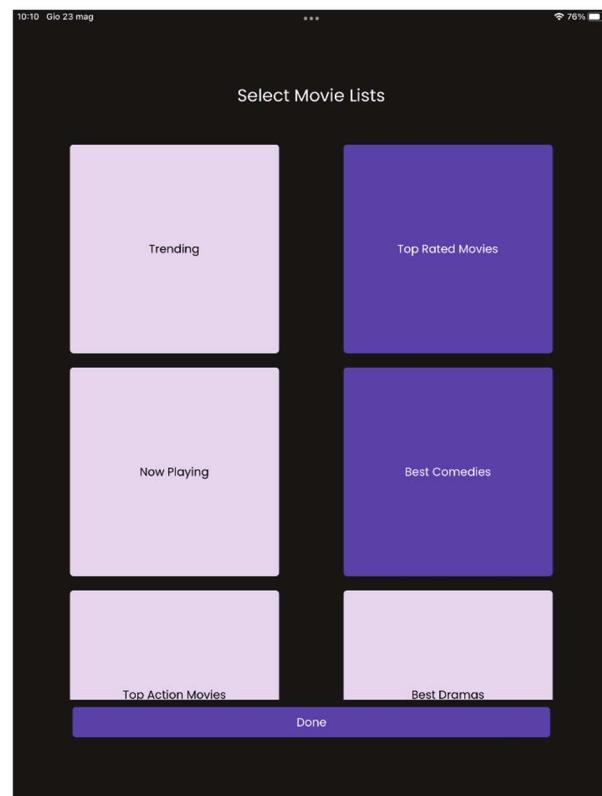


Figura 37 - Schermata Selezione Liste iPad



Figura 38 - Schermata Votazione Film Smartphone

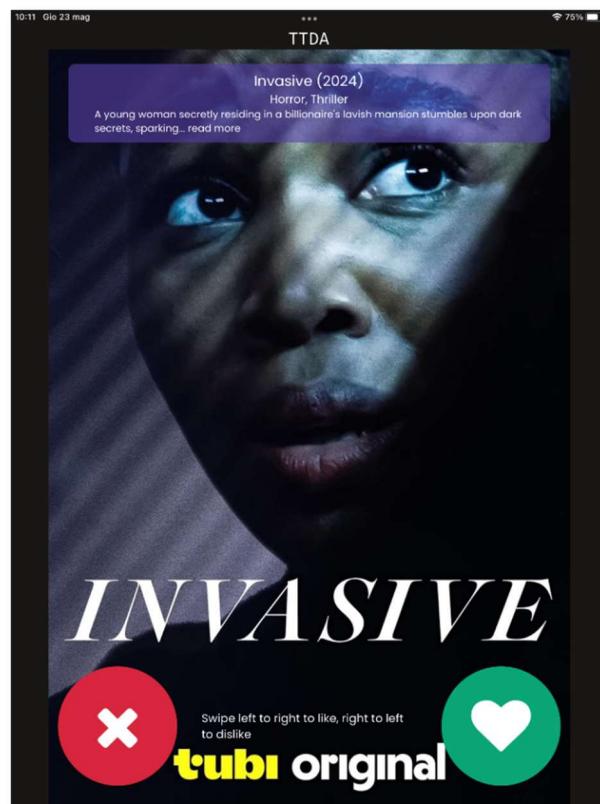


Figura 39 - Schermata Votazione Film iPad



Figura 40 - Schermata Classifica Smartphone



Figura 41 - Schermata Classifica iPad

6 Test

6.1 Protocollo di test

TC-001			
Nome	Errore di connessione	Riferimento	Connessione frontend-backend
Descrizione	In caso di errore di connessione, l'utente viene informato correttamente		
Prerequisiti	Server socket spento		
Procedura	<ol style="list-style-type: none"> 1. Aprire l'applicazione 2. Visualizzare messaggio indicante l'errore di connessione 		
Risultati attesi	Viene mostrato un messaggio di errore indicante l'errore di connessione		

TC-002			
Nome	Creazione lobby	Riferimento	Req-001
Descrizione	L'utente può creare una lobby		
Prerequisiti			
Procedura	<ol style="list-style-type: none"> 1. Aprire l'applicazione 2. Premere su "Create Lobby" 3. Inserire il proprio nome 4. Inviare 5. Controllare di essere nella pagina della lobby 6. Controllare che il proprio nome sia presente nella lista degli utenti nella lobby 		
Risultati attesi	Una volta creata la lobby ci si trova al suo interno e si visualizza il proprio nome nella lista di utenti nella lobby.		

TC-003			
Nome	Entrata in una lobby in attesa	Riferimento	Req-002
Descrizione	L'utente può unirsi ad una lobby già esistente		
Prerequisiti	<ol style="list-style-type: none"> 1. Avere una lobby già esistente 		
Procedura	<ol style="list-style-type: none"> 1. Aprire l'applicazione 2. Premere su "Join Lobby" 3. Inserire il proprio nome 4. Inviare 5. Inserire il codice della lobby 6. Inviare 7. Controllare di essere nella pagina della lobby 8. Controllare che il proprio nome sia presente nella lista degli utenti nella lobby 		
Risultati attesi	Una volta uniti alla lobby ci si trova al suo interno e si visualizza il proprio nome nella lista di utenti nella lobby.		

TC-004			
Nome	Entrata in una lobby non esistente	Riferimento	Req-002
Descrizione	L'utente non può unirsi ad una lobby non esistente		
Prerequisiti	1. Avere una lobby già esistente		
Procedura	<ol style="list-style-type: none"> 1. Aprire l'applicazione 2. Premere su "Join Lobby" 3. Inserire il proprio nome 4. Inviare 5. Inserire un codice non esistente 6. Controllare di essere ancora nella pagina di join 7. Controllare di ricevere un messaggio indicante che la lobby non esiste 		
Risultati attesi	Non è possibile entrare in una lobby non esistente e in caso ci si provi viene dato un messaggio appropriato.		

TC-005			
Nome	Entrata in una lobby già avviata	Riferimento	Req-002 SubReq_2
Descrizione	L'utente può entrare in una lobby già avviata		
Prerequisiti	1. Avere una lobby già esistente che sia anche già stata avviata		
Procedura	<ol style="list-style-type: none"> 1. Aprire l'applicazione 2. Premere su "Join Lobby" 3. Inserire il proprio nome 4. Inviare 5. Inserire il codice 6. Inviare 7. Controllare di essere direttamente nella pagina di votazione dei film 		
Risultati attesi	Una volta uniti la lobby ci si trova al suo interno e si può votare normalmente.		

TC-006			
Nome	Entrata in una lobby in stato di scoreboard	Riferimento	Req-002 SubReq_2
Descrizione	L'utente può entrare in una lobby già avviata che ha anche già finito di votare		
Prerequisiti	1. Avere una lobby già esistente in stato scoreboard		
Procedura	<ol style="list-style-type: none"> 1. Aprire l'applicazione 2. Premere su "Join Lobby" 3. Inserire il proprio nome 4. Inviare 5. Inserire il codice 6. Inviare 7. Controllare di essere direttamente nella pagina di votazione dei film 		
Risultati attesi	Una volta uniti la lobby ci si trova al suo interno e si può visualizzare la scoreboard normalmente.		

TC-007			
Nome	Stato utente dopo entrata in una lobby in stato di scoreboard	Riferimento	Req-002 SubReq_2
Descrizione	L'utente può entrare in una lobby già avviata che ha anche già finito di votare. Quando si fa partire una seconda iterazione l'utente viene preso in considerazione e può votare.		
Prerequisiti	1. Avere una lobby già esistente in stato scoreboard		
Procedura	<ol style="list-style-type: none"> 1. Aprire l'applicazione 2. Premere su "Join Lobby" 3. Inserire il proprio nome 4. Inviare 5. Inserire il codice 6. Inviare 7. Controllare di essere direttamente nella pagina di votazione dei film 8. Far avviare una nuova iterazione all'amministratore 9. Controllare di essere nella pagina di votazione 		
Risultati attesi	Una volta uniti alla lobby ci si trova al suo interno e si può visualizzare la scoreboard normalmente. Successivamente in caso di partenza di una nuova iterazione, si può votare.		

TC-008			
Nome	Uscita da una lobby come utente normale (stato waiting)	Riferimento	Req-003
Descrizione	L'utente non admin può uscire dalla lobby in qualsiasi momento		
Prerequisiti	<ol style="list-style-type: none"> 1. Avere una lobby già esistente 2. Essere un utente non admin 		
Procedura	<ol style="list-style-type: none"> 1. Uscire dalla lobby 2. Controllare di essere nella pagina home 3. Controllare che la lobby esista ancora 4. Controllare che la lobby ancora esistente non abbia più al suo interno l'utente che è uscito 		
Risultati attesi	Una volta usciti dalla lobby ci si trova nella pagina home e la lobby viene aggiornata correttamente.		

TC-009			
Nome	Uscita da una lobby come utente admin	Riferimento	Req-003
Descrizione	L'utente admin può uscire dalla lobby ma questo causa la distruzione della stessa		
Prerequisiti	<ol style="list-style-type: none"> 1. Avere una lobby già esistente 2. Essere un utente admin 3. Avere all'interno della lobby almeno un altro utente 		
Procedura	<ol style="list-style-type: none"> 1. Uscire dalla lobby 2. Controllare di essere nella pagina home 3. Controllare che la lobby non esista più 4. Controllare che tutti gli utenti presenti nella lobby siano stati cacciati e rimandati alla pagina home 		
Risultati attesi	Una volta usciti dalla lobby ci si trova nella pagina home e la lobby viene eliminata correttamente.		

TC-010

Nome	Uscita da una lobby che lascia l'admin da solo	Riferimento	Req-003
Descrizione	Se la lobby contiene solo due utenti e l'utente non admin esce, l'operazione è permessa ma questo causa la distruzione della lobby. Questo succede solo in stato di votazione o scoreboard, in caso di attesa iniziale la lobby non viene distrutta.		
Prerequisiti	<ol style="list-style-type: none"> 1. Avere una lobby già esistente 2. Non essere l'utente admin 3. Avere all'interno della lobby solo un altro utente (l'admin) 4. La lobby non si deve trovare nello stato waiting (quindi deve essere o in stato di votazione o in stato di scoreboard) 		
Procedura	<ol style="list-style-type: none"> 1. Uscire dalla lobby 2. Controllare di essere nella pagina home 3. Controllare che la lobby non esista più 4. Controllare che tutti gli utenti presenti nella lobby siano stati cacciati e rimandati alla pagina home 		
Risultati attesi	Una volta usciti dalla lobby ci si trova nella pagina home e la lobby viene eliminata correttamente.		

TC-011

Nome	Scelta della lista di film	Riferimento	Req-004
Descrizione	L'utente admin può scegliere diverse liste di film da far passare in votazione		
Prerequisiti	<ol style="list-style-type: none"> 1. Avere una lobby già esistente 2. Essere un utente admin 3. Avere all'interno della lobby almeno un altro utente 		
Procedura	<ol style="list-style-type: none"> 1. Aprire la pagina per selezionare liste di film 2. Selezionare una categoria 3. Avviare la sessione 4. Controllare che i film presentati siano quelli della categoria scelta 		
Risultati attesi	I film scelti sono effettivamente quelli che sono presentati nel sondaggio.		

TC-012

Nome	Avviare sessione senza utenti	Riferimento	Req-005
Descrizione	Se l'utente admin tenta di avviare la sessione senza nessun altro all'interno della lobby questa operazione viene bloccata.		
Prerequisiti	<ol style="list-style-type: none"> 1. Avere una lobby già esistente 2. Essere un utente admin 3. Avere all'interno della lobby solo l'utente admin 		
Procedura	<ol style="list-style-type: none"> 1. Tentare di avviare la sessione 2. Controllare che la sessione non sia effettivamente avviata 		
Risultati attesi	La sessione non può essere avviata se l'unico utente nella lobby è l'admin.		

TC-013			
Nome	Avviare sessione senza aver scelto film	Riferimento	Req-005
Descrizione	Se l'utente admin non sceglie manualmente nessuna lista di film, viene automaticamente scelta la lista dei film di tendenza.		
Prerequisiti	<ol style="list-style-type: none"> 1. Avere una lobby già esistente 2. Essere un utente admin 3. Avere all'interno della lobby almeno un altro utente 		
Procedura	<ol style="list-style-type: none"> 3. Avviare la sessione 4. Controllare che i film presentati siano quelli della lista dei film di tendenza 		
Risultati attesi	I film presentati sono effettivamente quelli della lista dei film di tendenza.		

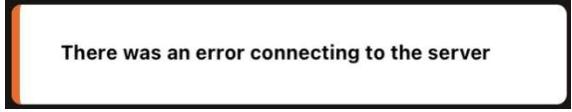
TC-014			
Nome	Votazione film	Riferimento	Req-006
Descrizione	L'utente deve poter votare ogni film indicando una reazione positiva o negativa.		
Prerequisiti	<ol style="list-style-type: none"> 1. Avere una lobby già esistente e già avviata 		
Procedura	<ol style="list-style-type: none"> 1. Votare positivamente un film 2. Votare negativamente un film 3. Completare votazioni 4. Controllare che nella classifica i punti siano stati assegnati correttamente 		
Risultati attesi	I voti dati ai film sono salvati e mostrati correttamente.		

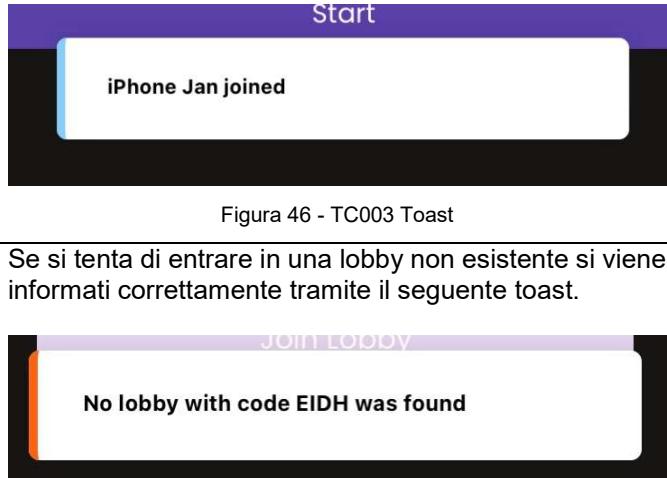
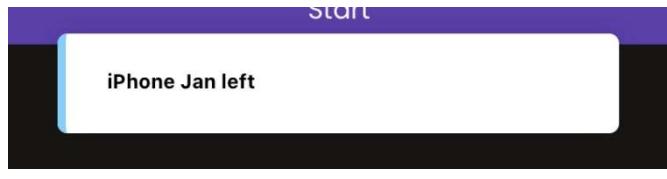
TC-015			
Nome	Classifica film	Riferimento	Req-007
Descrizione	L'utente dopo aver votato tutti i film visualizza una classifica dei film votati in ordine decrescente in base al numero di voti positivi ottenuti.		
Prerequisiti	<ol style="list-style-type: none"> 1. Avere una lobby già esistente e già avviata 		
Procedura	<ol style="list-style-type: none"> 1. Votare tutti i film 2. Aspettare nella pagina di attesa che tutti gli altri utenti finiscano di votare 3. Controllare che nella classifica i punti oltre ad essere assegnati correttamente sono in ordine decrescente. 		
Risultati attesi	I voti dati ai film sono salvati e mostrati nell'ordine corretto.		

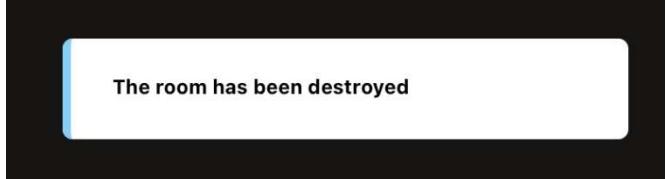
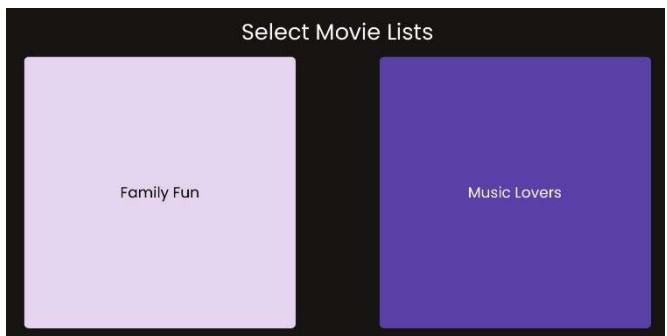
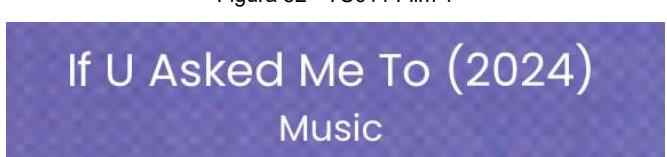
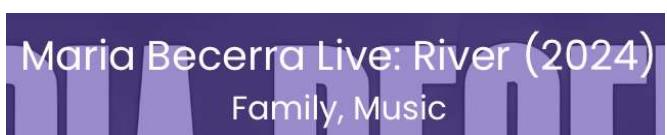
TC-016			
Nome	Visualizzazione trailer film	Riferimento	Req-008
Descrizione	L'utente nella pagina della classifica ha la possibilità di riprodurre direttamente il trailer dei film.		
Prerequisiti	<ol style="list-style-type: none"> 1. Avere una lobby già esistente e già avviata 2. La lobby deve aver già raggiunto la fase di classifica 		
Procedura	<ol style="list-style-type: none"> 1. Nella pagina della classifica premere il pulsante per riprodurre il trailer di un film 2. Controllare di star visualizzando il trailer del film corretto 		
Risultati attesi	Premere il pulsante per riprodurre il trailer riproduce effettivamente il trailer.		

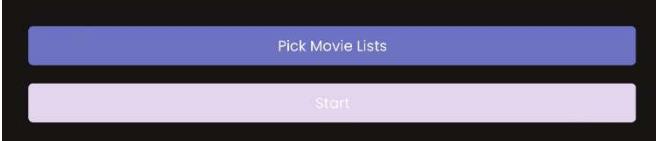
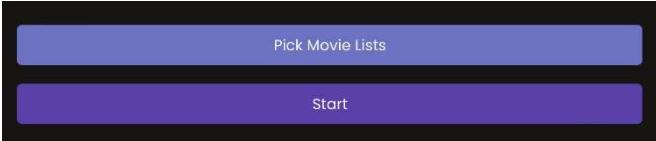
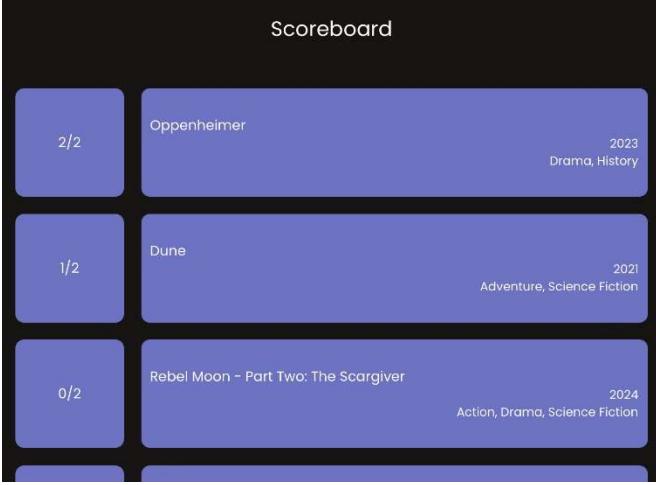
TC-017				
Nome	Far partire una seconda iterazione di votazione	Riferimento	Req-009	
Descrizione	L'utente admin nella pagina della classifica ha la possibilità di far partire una seconda iterazione di votazione con i film che hanno ottenuto il punteggio più alto.			
Prerequisiti	<ol style="list-style-type: none"> Avere una lobby già esistente e già avviata La lobby deve aver già raggiunto la fase di classifica Essere l'utente admin 			
Procedura	<ol style="list-style-type: none"> Far partire la seconda iterazione Controllare di star votando solo i film con il punteggio migliore Controllare che tutti gli utenti presenti nella lobby stiano facendo la stessa cosa 			
Risultati attesi	La seconda iterazione prende in considerazione solo i film meglio votati dell'iterazione precedente e tutti gli utenti nella lobby partecipano al secondo round di votazioni.			

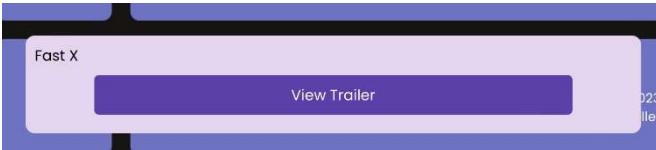
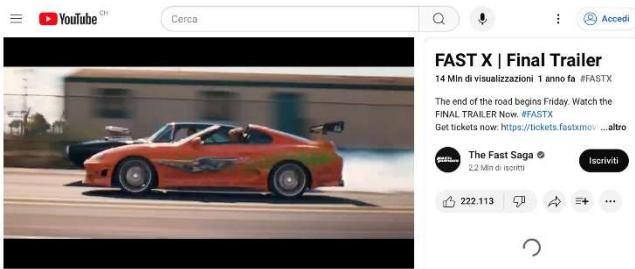
6.2 Risultati test

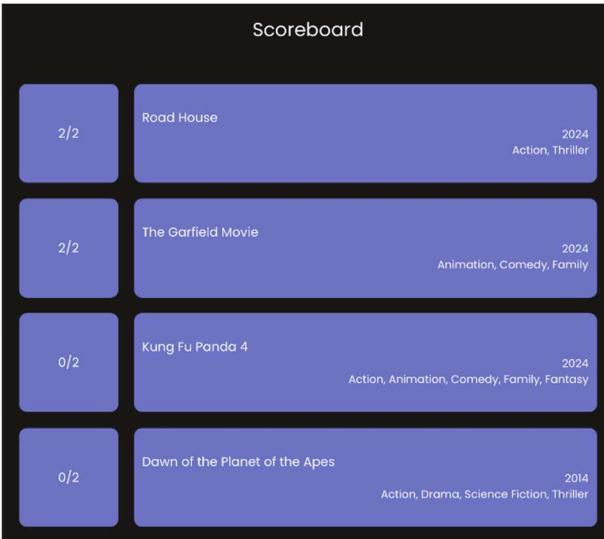
Test Case	Data	Risultato ottenuto	Stato iOS	Stato Android
TC-001	22.05.2024	<p>Come atteso, se il server è spento, una volta finita in timeout la richiesta di connessione viene mostrato un messaggio. Questo messaggio rimane visibile fino a quando la connessione viene ristabilita e viene quindi mostrato un messaggio indicante questa informazione.</p>  <p>Figura 42 - TC001 Toast Errore</p>  <p>Figura 43 - TC001 Toast Riconnessione</p>	Passato	Passato
TC-002	21.05.2024	<p>L'utente può creare senza problemi una lobby e viene indirizzato automaticamente alla pagina dove può vedere il codice e il proprio username.</p>  <p>Figura 44 - TC002 Lobby</p>	Passato	Passato

		L'utente può unirsi alla lobby e viene indirizzato automaticamente alla pagina corretta con i dati corretti. Gli altri utenti ottengono un aggiornamento della lista degli utenti e in più ricevono un messaggio sotto forma di toast indicando quale utente sia appena entrato.			
TC-003	21.05.2024	 <p>Share this code 2 X 4 D iPad Jan Tablet iPhone Jan</p> <p>Figura 45 - TC003 Lobby</p>	Passato	Passato	
TC-004	21.05.2024	<p>Se si tenta di entrare in una lobby non esistente si viene informati correttamente tramite il seguente toast.</p>  <p>JOIN LOBBY No lobby with code EIDH was found</p> <p>Figura 47 - TC004 Toast</p>	Passato	Passato	
TC-005	21.05.2024	È possibile entrare in una lobby già avviata. Si viene indirizzati direttamente alla pagina di votazione.	Passato	Passato	
TC-006	21.05.2024	È possibile entrare in una lobby in stato di scoreboard. Si viene indirizzati direttamente alla pagina di scoreboard.	Passato	Passato	
TC-007	21.05.2024	Una volta entrati si visualizza la scoreboard. Quando viene avviata la nuova iterazione si viene indirizzati alla pagina di votazione assieme agli altri utenti.	Passato	Passato	
TC-008	21.05.2024	Quando un utente lascia una lobby, gli altri utenti vengono informati tramite toast.	 <p>Start iPhone Jan left</p> <p>Figura 48 - TC008 Toast</p>	Passato	Passato

TC-009	21.05.2024	<p>Quando un admin esce dalla lobby, tutti gli utenti vengono indirizzati alla pagina home. Viene inoltre visualizzato questo toast.</p>  <p>Figura 49 - TC009 Toast</p>	Passato	Passato
TC-010	22.05.2024	<p>Quando le condizioni di distruzione della lobby sono raggiunte, l'utente rimanente viene indirizzato automaticamente alla pagina home e viene informato della distruzione della lobby con il seguente toast.</p>  <p>Figura 50 - TC010 Toast</p>	Passato	Passato
TC-011	21.05.2024	<p>Durante la fase di test ho selezionato la lista di film musicali. Come visibile tramite le immagini, tutti i film proposti hanno il genere Music.</p>  <p>Figura 51 - TC011 Scelta lista</p>  <p>Figura 52 - TC011 Film 1</p>  <p>Figura 53 - TC011 Film 2</p>  <p>Figura 54 - TC011 Film 3</p>	Passato	Passato

TC-012	21.05.2024	<p>Il pulsante per far partire la votazione è disabilitato se meno di due utenti sono nella lobby. Se un utente entra ed esce, il pulsante è abilitato solo mentre l'utente è nella lobby e, quando esce, viene disabilitato di nuovo.</p>  <p>Figura 55 - TC012 Pulsante Disabilitato</p>  <p>Figura 56 - TC012 Pulsante Abilitato</p>	Passato	Passato
TC-013	21.05.2024	<p>Se la sessione viene avviata mentre non ci sono liste selezionate, viene utilizzata correttamente la lista dei film di tendenza.</p>	Passato	Passato
TC-014	21.05.2024	<p>Per questo test ho deciso di votare positivamente con 2 utenti Oppenheimer, con 1 utente Dune e tutti gli altri voti negativamente. L'immagine della scoreboard riflette queste decisioni correttamente.</p>  <p>Figura 57 - TC014 Scoreboard</p>	Passato	Passato

TC-015	21.05.2024	<p>Test parallelo a TC-014. La classifica viene visualizzata correttamente con i dati giusti dei voti dati dagli utenti.</p>	Passato	Passato
TC-016	21.05.2024	<p>Nella pagina della scoreboard premendo su un qualsiasi film viene aperto un modal con al suo interno il nome del film e un pulsante per visualizzare il trailer. Se questo pulsante viene premuto, si viene reindirizzati su YouTube sul trailer del film.</p>  <p>Figura 58 - TC016 Modal</p>  <p>Figura 59 - TC016 Trailer</p>	Passato	Passato

TC-017	21.05.2024	<p>Come atteso, nella seconda iterazione sono utilizzati solo i film votati più positivamente nell'iterazione precedente.</p>  <table border="1"> <thead> <tr> <th colspan="3">Scoreboard</th> </tr> </thead> <tbody> <tr> <td>2/2</td> <td>Road House</td> <td>2024 Action, Thriller</td> </tr> <tr> <td>2/2</td> <td>Kung Fu Panda 4</td> <td>2024 Action, Animation, Comedy, Family, Fantasy</td> </tr> <tr> <td>2/2</td> <td>Rise of the Planet of the Apes</td> <td>2011 Action, Drama, Science Fiction, Thriller</td> </tr> <tr> <td>2/2</td> <td>The Garfield Movie</td> <td>2024 Animation, Comedy, Family</td> </tr> <tr> <td>2/2</td> <td>Dawn of the Planet of the Apes</td> <td>2014 Action, Drama, Science Fiction, Thriller</td> </tr> </tbody> </table> <p>Figura 60 - TC017 Scoreboard iterazione 1</p>  <table border="1"> <thead> <tr> <th colspan="3">Scoreboard</th> </tr> </thead> <tbody> <tr> <td>2/2</td> <td>Road House</td> <td>2024 Action, Thriller</td> </tr> <tr> <td>2/2</td> <td>The Garfield Movie</td> <td>2024 Animation, Comedy, Family</td> </tr> <tr> <td>0/2</td> <td>Kung Fu Panda 4</td> <td>2024 Action, Animation, Comedy, Family, Fantasy</td> </tr> <tr> <td>0/2</td> <td>Dawn of the Planet of the Apes</td> <td>2014 Action, Drama, Science Fiction, Thriller</td> </tr> </tbody> </table> <p>Figura 61 - TC017 Scoreboard iterazione 2</p>	Scoreboard			2/2	Road House	2024 Action, Thriller	2/2	Kung Fu Panda 4	2024 Action, Animation, Comedy, Family, Fantasy	2/2	Rise of the Planet of the Apes	2011 Action, Drama, Science Fiction, Thriller	2/2	The Garfield Movie	2024 Animation, Comedy, Family	2/2	Dawn of the Planet of the Apes	2014 Action, Drama, Science Fiction, Thriller	Scoreboard			2/2	Road House	2024 Action, Thriller	2/2	The Garfield Movie	2024 Animation, Comedy, Family	0/2	Kung Fu Panda 4	2024 Action, Animation, Comedy, Family, Fantasy	0/2	Dawn of the Planet of the Apes	2014 Action, Drama, Science Fiction, Thriller	Passato	Passato
Scoreboard																																					
2/2	Road House	2024 Action, Thriller																																			
2/2	Kung Fu Panda 4	2024 Action, Animation, Comedy, Family, Fantasy																																			
2/2	Rise of the Planet of the Apes	2011 Action, Drama, Science Fiction, Thriller																																			
2/2	The Garfield Movie	2024 Animation, Comedy, Family																																			
2/2	Dawn of the Planet of the Apes	2014 Action, Drama, Science Fiction, Thriller																																			
Scoreboard																																					
2/2	Road House	2024 Action, Thriller																																			
2/2	The Garfield Movie	2024 Animation, Comedy, Family																																			
0/2	Kung Fu Panda 4	2024 Action, Animation, Comedy, Family, Fantasy																																			
0/2	Dawn of the Planet of the Apes	2014 Action, Drama, Science Fiction, Thriller																																			

6.3 Mancanze/limitazioni conosciute

6.3.1 Bug voti troppo rapidi

È stato notato che se si vota troppo rapidamente alcuni film potrebbero essere saltati o votati due volte dallo stesso utente e occasionalmente ci sono dei bug grafici. Questo, mentre probabilmente causato anche dal componente di terze parti TinderCard, sarebbe da debuggare per capire la vera natura del problema. Come soluzione temporanea si potrebbe disabilitare i pulsanti e lo swipe per votare subito dopo un voto, ma questa sarebbe solo una soluzione temporanea e non risolverebbe realmente il problema.

6.3.2 Disconnessione socket con app in background

Durante la fase di test, è stato notato che se si apre il trailer su YouTube, mettendo così l'applicazione in background, dopo un po' di tempo viene mandato l'evento di disconnessione. Questo è probabilmente causato da un ottimizzazione dei sistemi operativi che chiude le connessioni di app rimaste in background per un po' di tempo. Per risolvere questo problema si potrebbe controllare sia la documentazione di Socket.io che quella di Expo, per trovare una configurazione che permetta di sovrascrivere questo comportamento. Questo problema è stato notato mentre testavo l'app utilizzando Expo Go, è possibile che il problema non si verifichi nella build o si verifichi in modo leggermente differente.

6.3.3 Blocco portrait

Durante le fasi di test preliminari, è emerso un problema relativo al blocco in modalità portrait che sembra non funzionare correttamente sugli iPad. Questa problematica è stata riconosciuta e documentata su GitHub (<https://github.com/expo/expo/issues/5188>) e potrebbe essere legata semplicemente all'app Expo Go anziché alle configurazioni specifiche. Tuttavia, sarebbe opportuno condurre ulteriori test in caso di distribuzione sull'App Store.

Movie Survey

7 Consuntivo

Il diagramma di Gantt consuntivo evidenzia le differenze tra la pianificazione e l'effettivo utilizzo del tempo. In arancione è visibile il tempo effettivo, mentre in blu è indicato il tempo pianificato. Sono presenti diverse discrepanze evidenti: alcune attività sono risultate più rapide del previsto, mentre altre hanno richiesto più tempo. Tuttavia, nel complesso, tutto si è bilanciato e, anzi, è rimasto un po' di tempo in eccesso per delle migliorie finali. Riconosco che la pianificazione non è stata molto precisa, ma considerando che non avevo mai utilizzato i socket in un progetto di questa complessità, ritengo che un margine di errore di questa portata sia accettabile e comprensibile.

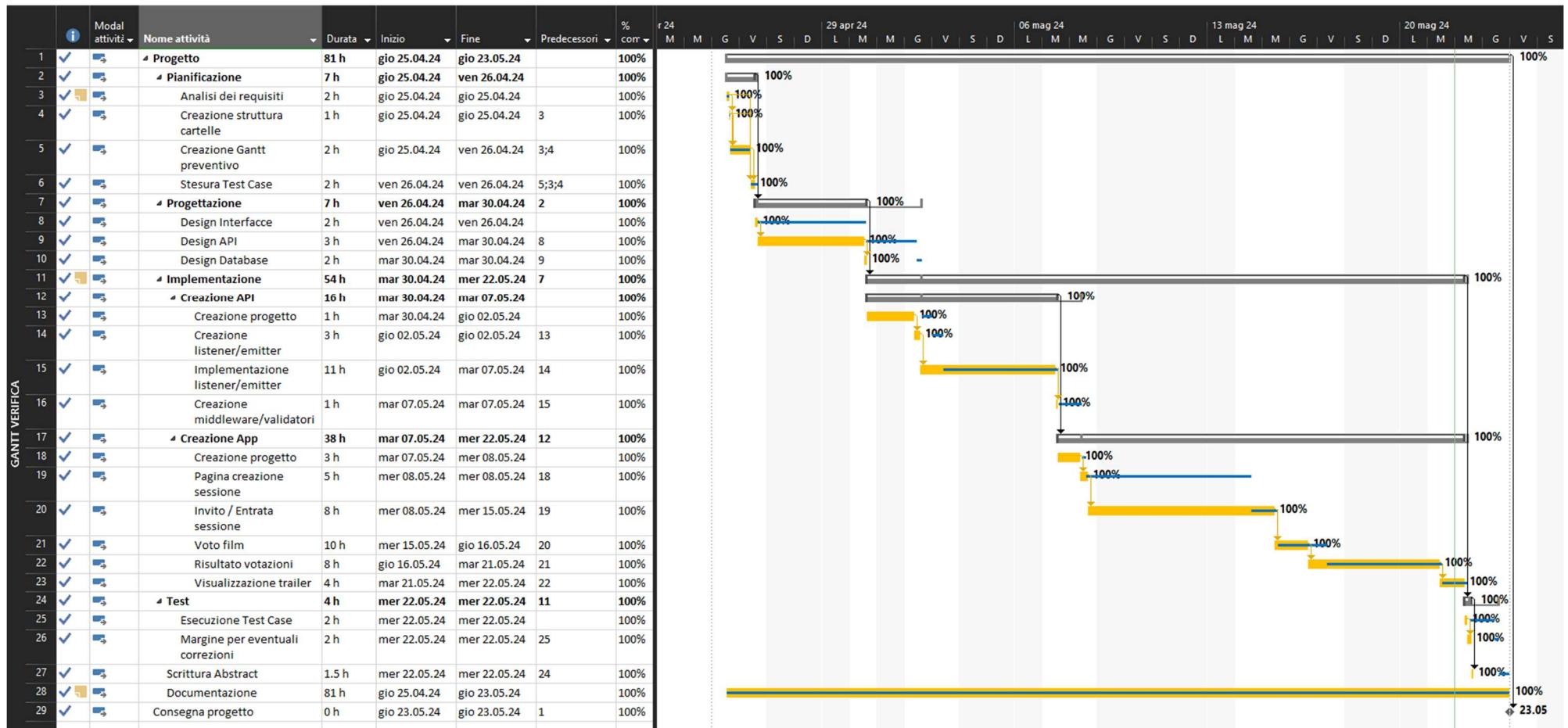


Figura 62 - Gantt consuntivo

8 Conclusioni

La soluzione ottenuta rappresenta una base solida per un'applicazione che ha il potenziale per eccellere nel suo settore. Attualmente, tutte le funzionalità principali sono state implementate, anche se ci sono ancora alcuni miglioramenti possibili che non sono stati realizzati a causa della scarsità di tempo. Nonostante ciò, sono convinto che il prodotto sia già pubblicabile per una fase di testing su un pubblico più ampio. Dopo aver risolto eventuali problemi emersi durante questa fase, l'applicazione potrebbe essere facilmente mantenuta e ampliata con nuove funzionalità. Grazie al lavoro svolto finora, il percorso per il perfezionamento futuro dell'applicazione sarà molto più agevole.

8.1 Sviluppi futuri⁴

8.1.1 Modalità aggiuntive

Sarebbe interessante aggiungere delle modalità di votazione alternative per migliorare l'esperienza utente. Alcune proposte includono:

- Modalità di Votazione Controllata dall'Admin
 - L'amministratore della lobby può terminare la votazione in qualsiasi momento, garantendo maggiore controllo e flessibilità nel processo decisionale.
- Modalità Lotteria
 - Un film viene selezionato casualmente tra quelli disponibili, offrendo un elemento di sorpresa e rendendo la scelta del film più divertente.
- Modalità a Tempo
 - Impostare un timer per ciascuna fase di votazione. Gli utenti devono votare entro un tempo prestabilito, rendendo il processo più dinamico e rapido.
- Modalità Preferenze Ponderate
 - Gli utenti possono assegnare un punteggio di preferenza a ciascun film. I film con i punteggi complessivi più alti saranno selezionati.
- Modalità Eliminazione Diretta
 - In ogni round, i film con meno voti vengono eliminati, fino a quando ne rimane solo uno.

Queste modalità aggiuntive renderebbero l'applicazione più versatile e personalizzabile, aumentando il coinvolgimento degli utenti e adattandosi meglio alle diverse esigenze dei gruppi di utenti.

⁴ Alcune di queste idee sono state suggerite da un'IA

8.1.2 Selezione manuale di film

Un'altra funzionalità interessante da aggiungere sarebbe la possibilità di selezionare manualmente i film da includere nella votazione. Attualmente, l'applicazione offre delle liste predefinite da cui gli utenti possono scegliere, ma consentire una selezione manuale potrebbe offrire maggiore flessibilità e personalizzazione. Alcune proposte per migliorare questa funzionalità sarebbero:

- Ricerca Personalizzata
 - Gli utenti possono cercare e aggiungere specifici film attraverso una barra di ricerca integrata, utilizzando l'API di TMDB per trovare i film desiderati.
- Liste Personalizzate
 - Gli utenti possono creare e salvare liste personalizzate di film, da utilizzare in successive sessioni di votazione.
- Filtri Avanzati
 - Aggiungere filtri per anno, genere, regista, attori principali, e altro, per facilitare la selezione manuale dei film.
- Selezione Multi-Utente
 - Consentire a tutti i partecipanti della lobby di aggiungere i propri film alla lista di votazione, rendendo il processo più collaborativo.

Implementare la selezione manuale di film aumenterebbe la personalizzazione e il controllo che gli utenti hanno sulla votazione, migliorando l'esperienza complessiva dell'applicazione.

8.1.3 Collegamento e filtri con servizi di streaming

Una buona aggiunta sarebbe la possibilità di applicare filtri ai film, permettendo agli utenti di visualizzare solo quelli disponibili sulle piattaforme di streaming da loro selezionate. Successivamente, sarebbe utile integrare un link nella classifica dei film per aprire direttamente il film sull'app o sul sito del servizio di streaming corrispondente. Queste funzionalità migliorerebbero ulteriormente l'esperienza utente, rendendo più facile e veloce l'accesso ai film scelti.

8.1.4 Supporto multilingua sui film

Una buona aggiunta sarebbe la possibilità di avere un supporto multilingua anche a livello di film. Oltre alle interfacce nella lingua dell'utente, verrebbero offerti solo i film che dispongono di doppiaggio nella lingua selezionata. Questo sarebbe possibile mantenendo l'API di TMDB, che già offre opzioni di questo tipo. Questa funzionalità migliorerebbe notevolmente l'esperienza utente, garantendo che i film proposti siano disponibili nella lingua preferita dagli utenti.

8.1.5 Deploy

Ultimo, ma non meno importante, è il deploy dell'applicazione e del backend su internet. Per il backend in un ambiente produttivo, sarebbe ottimale utilizzare un servizio come PM2 per consentire una gestione più efficiente del servizio. Per quanto riguarda l'applicazione, questa è essenzialmente pronta e richiede solo la modifica della configurazione dell'indirizzo del server. In generale, il progetto è quasi pronto per essere pubblicato; sono necessari solo alcuni piccoli ritocchi per renderlo completamente funzionale per il pubblico.

8.2 Considerazioni personali

Grazie a questo progetto, ho avuto l'opportunità di approfondire le mie competenze in React Native ed Expo, integrando le ultime novità della tecnologia. Inoltre, ho acquisito familiarità con lo sviluppo tramite socket, ampliando le mie conoscenze in questo ambito. Durante il processo, ho notato miglioramenti nelle mie capacità di design delle interfacce per dispositivi mobili, un'area che mi ha presentato sfide in passato ma che sto progressivamente migliorando con la pratica costante.

Una delle lezioni più importanti è stata l'importanza di scrivere test case immediatamente dopo la definizione dei requisiti, garantendo così una copertura completa degli scenari possibili. Nonostante fossi alle prime armi con i socket, sono riuscito ad apprendere le basi durante lo sviluppo senza troppi intoppi.

Durante il percorso, ho incontrato diversi ostacoli e complicazioni, ma sono riuscito a superarli grazie alla determinazione e alla risoluzione creativa dei problemi. Ho anche avuto l'opportunità di ottimizzare il codice precedentemente scritto, dimostrando la mia capacità di individuare e risolvere possibili bottleneck in modo efficiente.

Complessivamente, sono soddisfatto del risultato ottenuto, che riflette accuratamente le mie conoscenze e capacità nel campo dello sviluppo e dell'informatica. Anche se avrei trovato interessante integrare una gestione degli utenti con account e autenticazione, comprendo le limitazioni temporali e la necessità di mantenere il progetto focalizzato e gestibile.

9 Bibliografia

9.1 Sitografia

1. <https://chat.openai.com/>, Intera durata del progetto, occasionalmente usato per automatizzare task monotone e fare correzioni linguistiche, l'utilizzo è documentato più nel dettaglio all'interno dei diari, sotto forma di commento o con note a piè pagina
2. <https://www.conventionalcommits.org/en/v1.0.0/>, 25.04.2024
3. <https://developer.themoviedb.org/>, Intera durata del progetto
4. <https://coolors.co/>, 26.04.2024
5. <https://stackoverflow.com/>, Intera durata del progetto
6. <https://docs.expo.dev/>, Intera durata del progetto
7. <https://icons.expo.fyi/>, Fase di sviluppo del progetto
8. <https://www.npmjs.com/>, Fase di sviluppo del progetto
9. <https://socket.io/docs/v3>, Intera durata del progetto
10. <https://lucid.app/>, Fase di design del progetto
11. <https://www.swisstransfer.com/>, Intera durata del progetto

10 Glossario⁵

Termine	Significato
Axios	Axios è un framework JavaScript leggero e flessibile utilizzato per la gestione delle richieste HTTP nel frontend e nel backend delle applicazioni web. Grazie alla sua sintassi semplice e intuitiva, Axios semplifica il processo di comunicazione con i server, consentendo agli sviluppatori di scrivere codice più pulito ed efficiente.
Bottleneck (informatica)	Un bottleneck (dall'inglese: collo di bottiglia) nell'ambito dell'informatica si riferisce a un punto critico o a una limitazione che rallenta l'efficienza complessiva di un sistema. È una parte del sistema in cui la capacità è inferiore rispetto ad altre parti, creando un collo di bottiglia che limita le prestazioni globali. Identificare e risolvere i bottleneck è fondamentale per ottimizzare le prestazioni dei sistemi informatici.
Expo	Expo è una piattaforma e insieme di strumenti per lo sviluppo rapido di applicazioni mobili con React Native. Semplifica il processo di sviluppo, offrendo funzionalità come l'anteprima in tempo reale, il debugging semplificato e l'accesso a una serie di API predefinite.
Hook (React)	Gli hook sono funzioni speciali introdotte in React che permettono di utilizzare lo stato e altre funzionalità di React nei componenti funzionali (funzioni). Gli hook semplificano il codice dei componenti e promuovono il riutilizzo della logica di stato e di effetti in diversi componenti.
HTTP (Hypertext Transfer Protocol)	HTTP, o Hypertext Transfer Protocol, è il protocollo di comunicazione utilizzato nel World Wide Web per trasferire risorse, come pagine web e immagini, tra client (come browser web) e server.
JS (JavaScript)	JavaScript (JS) è un linguaggio di scripting ampiamente utilizzato per aggiungere interattività alle pagine web. È flessibile e supporta paradigmi di programmazione imperativa, orientata agli oggetti e funzionale. JavaScript può essere eseguito sia lato client che lato server ed è essenziale per lo sviluppo web, giochi e app mobile.
Node.js	Node.js è un ambiente di runtime open-source basato su JavaScript, progettato per eseguire codice lato server. Consente agli sviluppatori di utilizzare JavaScript per scrivere applicazioni server-side, facilitando la creazione di server web altamente scalabili e efficienti.
npm	npm, acronimo di "Node Package Manager", un gestore di pacchetti per l'ambiente di sviluppo JavaScript Node.js. Consente agli sviluppatori di JavaScript di installare, condividere e gestire facilmente le dipendenze dei progetti JavaScript.
React Native	React Native è un framework di sviluppo cross-platform creato da Facebook per la creazione di applicazioni mobili native utilizzando React. Consente agli sviluppatori di scrivere codice in React e utilizzare le stesse componenti per creare app per iOS e Android, migliorando l'efficienza dello sviluppo.
Socket (comunicazione tramite socket):	Un socket è un punto terminale per lo scambio di dati tra due programmi su una rete. Identificato da un indirizzo IP e un numero di porta, può essere sia un client che un server. Consente la comunicazione bidirezionale in tempo reale, indipendentemente dalla piattaforma o dal linguaggio di programmazione.
TS (TypeScript)	TypeScript (TS) è un linguaggio di programmazione sviluppato da Microsoft che estende le funzionalità di JavaScript introducendo il concetto di tipizzazione statica. Con l'aggiunta dei tipi, TypeScript offre un controllo più rigoroso sugli errori durante lo sviluppo e migliora la manutenibilità del codice.

⁵ Diversi significati all'interno di questo glossario sono stati parzialmente generati da un'IA e controllati prima di essere inseriti.

11 Indice delle figure

Figura 1 - Use Case Gestione Lobby	7
Figura 2 - Use Case Gestione Dei Voti	7
Figura 3 - Gantt Preventivo	8
Figura 4 - Schema di rete	10
Figura 5 - Logica socket	11
Figura 6 - Classi diagramma logica socket..	11
Figura 7 - Diagramma database	12
Figura 8 - Color Palette.....	13
Figura 9 - Pagina iniziale	14
Figura 10 - Pagina creazione lobby.....	14
Figura 11 - Pagina join lobby	14
Figura 12 - Modal inserimento username.....	15
Figura 13 - Pagina scelta list di film.....	15
Figura 14 - Pagina scelta generi.....	15
Figura 15 - Pagina votazione film	16
Figura 16 - Pagina scoreboard	16
Figura 17 - Activity Diagram Generale	17
Figura 18 - Activity Diagram Socket	18
Figura 19 - Struttura cartelle backend	22
Figura 20 - DB Versione 1	23
Figura 21 - DB Versione 2	23
Figura 22 - DB Versione 3	24
Figura 23 - Toast Errore	31
Figura 24 - Toast Informazione	31
Figura 25 - Toast Successo.....	31
Figura 26 - LoadingIndicator iOS.....	32
Figura 27 - LoadingIndicator Android	32
Figura 28 - Schermata Home Smartphone.....	35
Figura 29 - Schermata Home iPad	35
Figura 30 - Modal Nome Smartphone	36
Figura 31 - Modal Nome iPad.....	36
Figura 32 - Schermata Lobby Smartphone	36
Figura 33 - Schermata Lobby iPad	36
Figura 34 - Schermata Inserimento Codice Smartphone	37
Figura 35 - Schermata Inserimento Codice iPad	37
Figura 36 - Schermata Selezione Liste Smartphone.....	37
Figura 37 - Schermata Selezione Liste iPad	37
Figura 38 - Schermata Votazione Film Smartphone	38
Figura 39 - Schermata Votazione Film iPad.....	38
Figura 40 - Schermata Classifica Smartphone	38
Figura 41 - Schermata Classifica iPad	38
Figura 42 - TC001 Toast Errore	44
Figura 43 - TC001 Toast Riconnessione	44
Figura 44 - TC002 Lobby	44
Figura 45 - TC003 Lobby.....	45
Figura 46 - TC003 Toast.....	45
Figura 47 - TC004 Toast.....	45
Figura 48 - TC008 Toast.....	45
Figura 49 - TC009 Toast.....	46
Figura 50 - TC010 Toast.....	46
Figura 51 - TC011 Scelta lista	46
Figura 52 - TC011 Film 1	46
Figura 53 - TC011 Film 2	46
Figura 54 - TC011 Film 3	46
Figura 55 - TC012 Pulsante Disabilitato	47

Figura 56 - TC012 Pulsante Abilitato.....	47
Figura 57 - TC014 Scoreboard	47
Figura 58 - TC016 Modal.....	48
Figura 59 - TC016 Trailer	48
Figura 60 - TC017 Scoreboard iterazione 1	49
Figura 61 - TC017 Scoreboard iterazione 2	49
Figura 62 - Gantt consuntivo	51

12 Allegati

QdC	1_QdC
Abstract	2_Abstract/Abstract.pdf
Diari	4_Diari/pdf