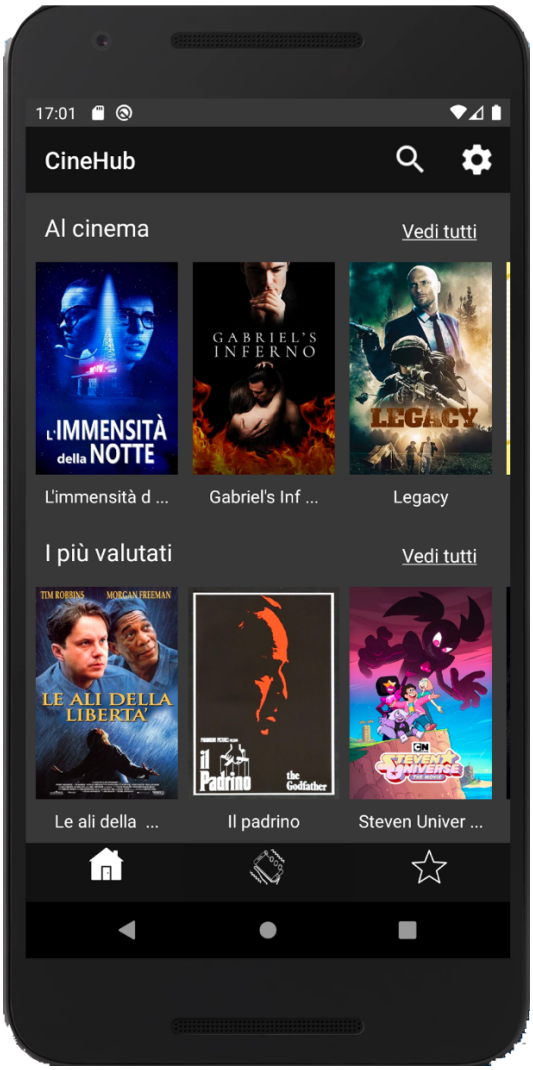
CINEHUB

Documento di design



All'attenzione della Prof.ssa Micucci e del Prof. Ginelli
Gruppo LOREAL
Lorenzo Gallorini 816390
Alessandro Albi 817769

SOMMARIO

INTRODUZIONE

Questa relazione ha lo scopo di ripercorrere quelle che sono state le tappe principali che ci hanno portato alla realizzazione dell'applicazione CineHub.

Analizzeremo brevemente quelle che sono state le funzionalità pensate e l'architettura sviluppatasi per esse, arrivando infine a prevedere come poter inserire l'applicazione nel mercato odierno.

FUNZIONALITÀ

In questa sezione andremo a descrivere le funzionalità che abbiamo implementato nell'applicazione.

DESCRIZIONE DELLE FUNZIONALITÀ

L'applicazione fornisce informazioni sui film utilizzando l'API di TMDB e permette all'utente di essere aggiornato sulle uscite più imminenti.

Tra le funzionalità dell'applicazione è possibile trovare quella di ricerca con filtri selezionati dall'utente stesso, per ricercare sia film che persone.

Nella scheda del film è possibile visualizzare il trailer, se disponibile.

L'utente, oltre alle informazioni del film, potrà visualizzare i dati relativi ai registi e agli attori e potrà aggiungere i film che preferisce nella sezione "I TUOI PREFERITI".

Inoltre, è stata pensata da noi un'ulteriore funzionalità, lo "SHAKE": questa suggerisce all'utente, tramite lo shake del telefono, un film. La selezione del film da consigliare non sarà del tutto casuale, infatti si tiene conto di quali siano i gusti dell'utente prendendo spunto dai film presenti nella sezione "I TUOI PREFERITI".

CASI D'USO

Andiamo ad analizzare i casi d'uso in maniera descrittiva pensando all'applicazione come una black box, analizzeremo quindi solamente

il comportamento dell'applicazione lato utente, senza soffermarci su come funziona.

Per ogni caso d'uso abbiamo creato un diagramma di interazione, per spiegare il funzionamento interno dell'applicazione.

CASO D'USO UC1: CERCA UN FILM TRAMITE LA RICERCA

Portata: Applicazione CineHub

Livello: Obbiettivo Utente

Attore Primario: Utente

Parti interessate e interessi:

L'utente: vuole cercare informazioni riguardo un film tramite l'applicazione.

Scenario principale di successo:

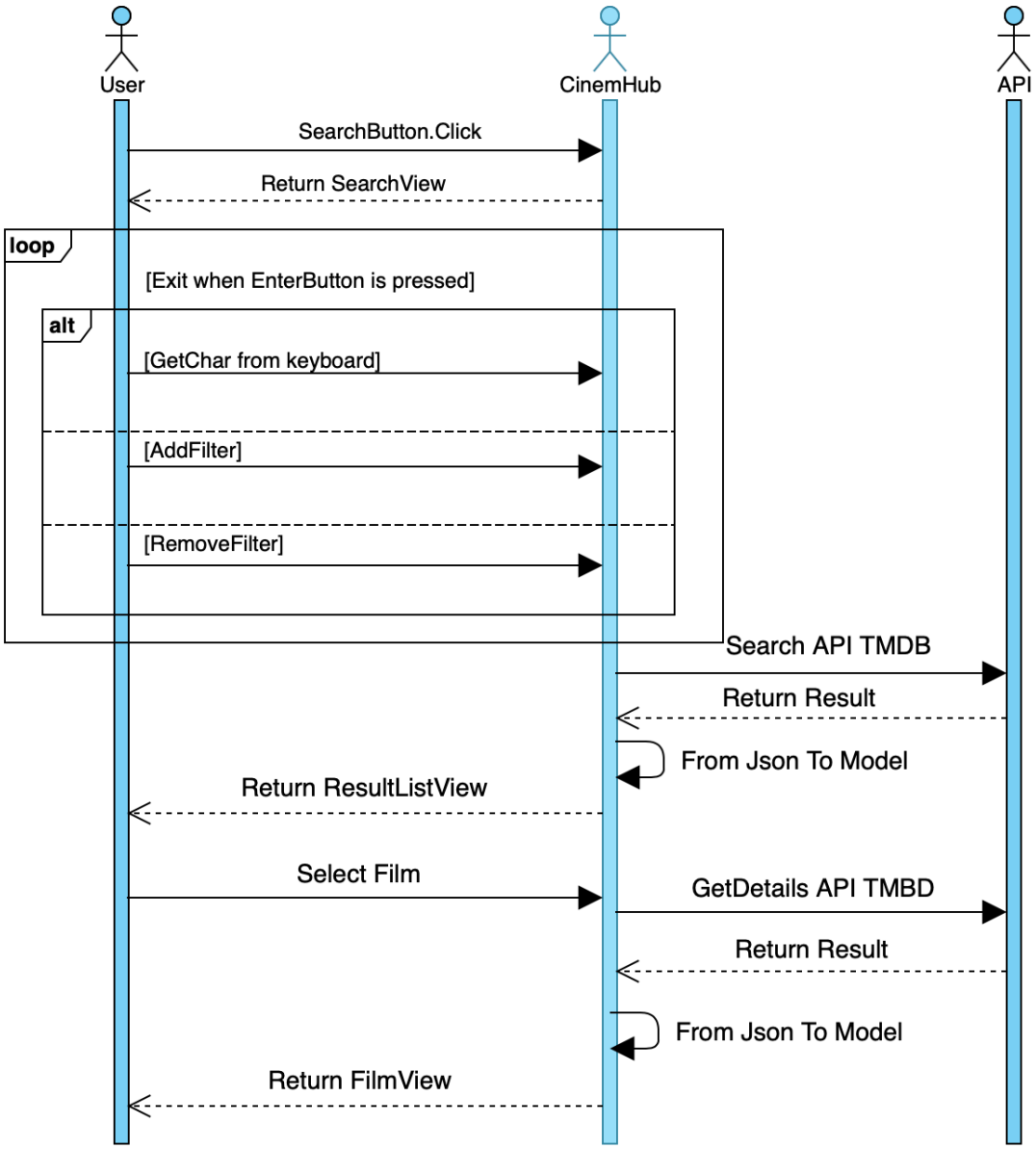
1. L'utente apre l'applicazione;
2. Clicca sul tasto ricerca;
3. Digita il nome del film che vuole cercare;
4. Visualizza i risultati e clicca sul film che stava cercando;
5. L'utente legge le informazioni che gli interessavano di quel film;

Estensioni

4A. L'utente non trova il film che stava cercando, o sta cercando il film tramite la parola chiave errata, oppure quel film non si trova sul database.

Requisiti speciali

L'utente per poter cercare un film deve disporre di una connessione a internet.


User
CinemHub
API
SearchButton.Click
Return SearchView
loop
[Exit when EnterButton is pressed]
alt
[GetChar from keyboard]
[AddFilter]
[RemoveFilter]
Search API TMDB
Return Result
From Json To Model
Return ResultListView
Select Film
GetDetails API TMBD
Return Result
From Json To Model
Return FilmView

CASO D'USO UC2: AGGIUNGERE E RECUPERARE UN FILM DAI PREFERITI

Portata: Applicazione CineHub

Livello: Obbiettivo Utente

Attore Primario: Utente

Parti interessate e interessi:

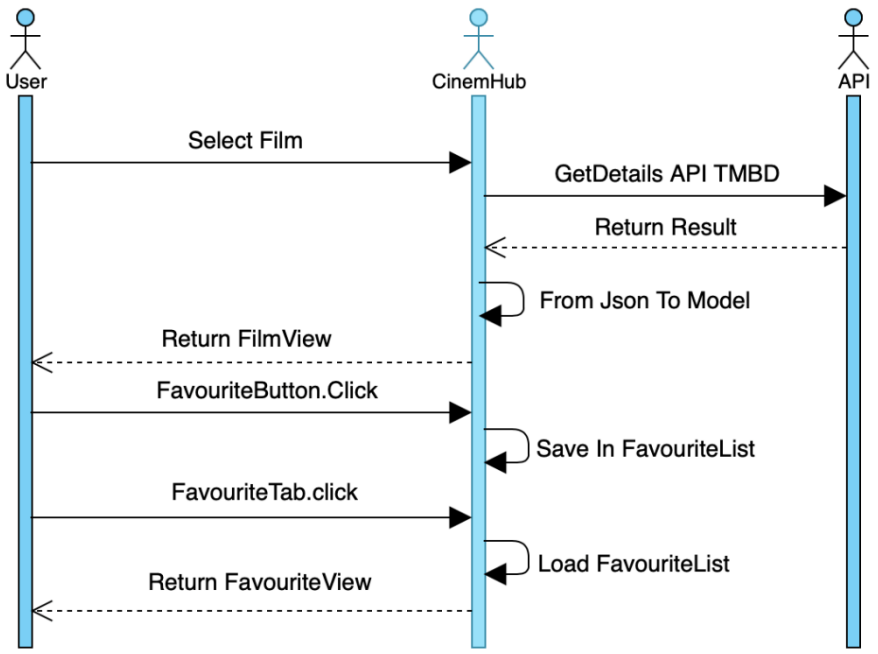
L'utente: Vuole aggiungere un film ai preferiti per poi poterlo visualizzarlo nella pagina ad hoc.

Scenario principale di successo:

1. L'utente apre l'applicazione;
2. Apre la scheda dettaglio di un film qualsiasi;
3. Clicca sul bottone per aggiungere il film ai preferiti (stella);
4. Apre la sezione dell'applicazione per visualizzare i preferiti;
5. L'utente visualizza i film che ha aggiunto ai preferiti;

Requisiti speciali

L'utente per poter visualizzare la scheda di un film deve disporre di una connessione a internet.



CASO D'USO UC3: FARSI CONSIGLIARE UN FILM DALL'APPLICAZIONE

Portata: Applicazione CineHub

Livello: Obbiettivo Utente

Attore Primario: Utente

Parti interessate e interessi:

L'utente: Vuole che l'applicazione gli consigli un film da vedere.

Scenario principale di successo:

1. L'utente apre l'applicazione;

2. Apre la sezione dell'applicazione per visualizzare la pagina dello shake;

3. Come indicato dalle istruzioni shakera il telefono;

4. L'applicazione propone un film;

5. L'utente visualizza il film che l'applicazione gli ha suggerito;

Estensioni

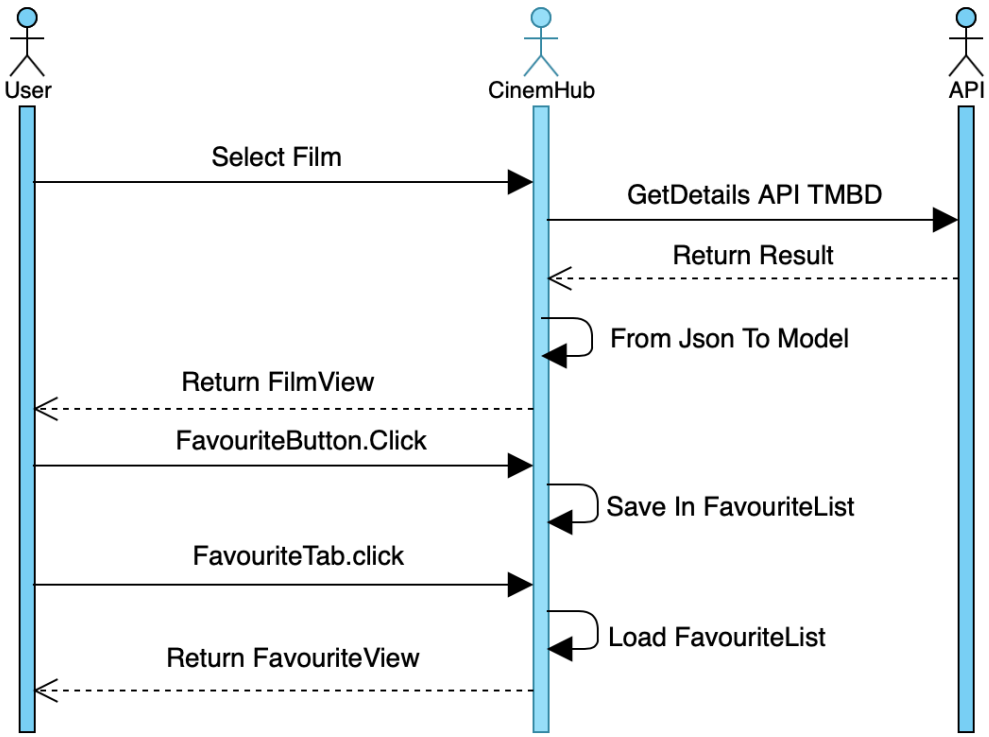
4A. L'utente non gradisce il film consigliato;

4B. L'utente può farsi consigliare un altro film shakerando nuovamente;

4C. Continua questo iter finché l'applicazione non gli consiglia un film da lui gradito;

Requisiti speciali

L'utente per poter usufruire di questa funzione deve avere una connessione internet.


User
CinemHub
API
Select Film
GetDetails API TMBD
Return Result
From Json To Model
Return FilmView
FavouriteButton.Click
Save In FavouriteList
FavouriteTab.click
Load FavouriteList
Return FavouriteView

PREPARAZIONE ALL'IMPLEMENTAZIONE

Prima di iniziare a implementare l'applicazione, abbiamo creato la documentazione necessaria per il progetto.

Di seguito andremo a spiegare brevemente le attività che abbiamo svolto, linkando ogni documento che abbiamo creato e condiviso su GitHub.

CREAZIONE DEL GAANT DI PROGETTO

Per prima cosa abbiamo pianificato le nostre attività utilizzando un software chiamato GaantProject, con il quale abbiamo organizzato le attività da svolgere prima di iniziare la fase di progettazione e successivamente, dopo aver analizzato il progetto, abbiamo pianificato le nostre attività di programmazione (Documento delle attività pianificate).

PROGETTAZIONE GRAFICA DELL'APPLICAZIONE

Siamo partiti facendo degli scratch su foglio per avere una idea generale, successivamente abbiamo creato il wireframe per capire quali interazioni creare tra le varie schermate.

In ultimo abbiamo creato il mockUp e partendo da esso il relativo prototipo.

Il prototipo è stato utilizzato per somministrare dei test agli utenti assieme ad un UX Researcher, il Dott. Parente Loris. Tale attività è stata svolta al fine di migliorare l'interfaccia dell'applicazione e, di conseguenza, l'usabilità.

Nel prossimo capitolo verrà spiegato meglio cosa è stato fatto.

PIANIFICAZIONE DELLE ATTIVITÀ

Abbiamo deciso di utilizzare un metodo agile (SCRUM) per lo sviluppo dell'applicazione.

Nel Documento delle attività pianificate si possono vedere gli sprint. Questi avevano la durata di 4 giorni e il venerdì veniva utilizzato per fare il punto della situazione e pianificare la settimana successiva.

Abbiamo fatto dei test incrementali delle funzionalità somministrandoli ad amici e parenti.

CONSULENZA DELL'UX RESEARCHER

Per rendere l'applicazione il più semplice ed usabile possibile abbiamo deciso di dare un'importanza specifica alla UI ed alla UX. Per fare ciò ci siamo avvalsi dell'aiuto di un esperto in materia: l'UX Researcher Parente Loris.

Considerando il tempo e le risorse a disposizione, abbiamo deciso di optare per un test di tipo qualitativo per valutare gli aspetti di usabilità e della User Experience. Data l'emergenza sanitaria il test è stato somministrato da remoto.

Abbiamo collaborato con l'esperto per la creazione del protocollo di ricerca, finalizzato ad individuare eventuali problemi di struttura, grafici e di interazione con l'interfaccia dell'applicazione.

Il Dott. Parente ha provveduto a somministrare il test assumendo il ruolo di moderatore dell'intervista.

Gli utenti sono stati scelti in base a dei criteri ben stabiliti dall'esperto per la validità del test.

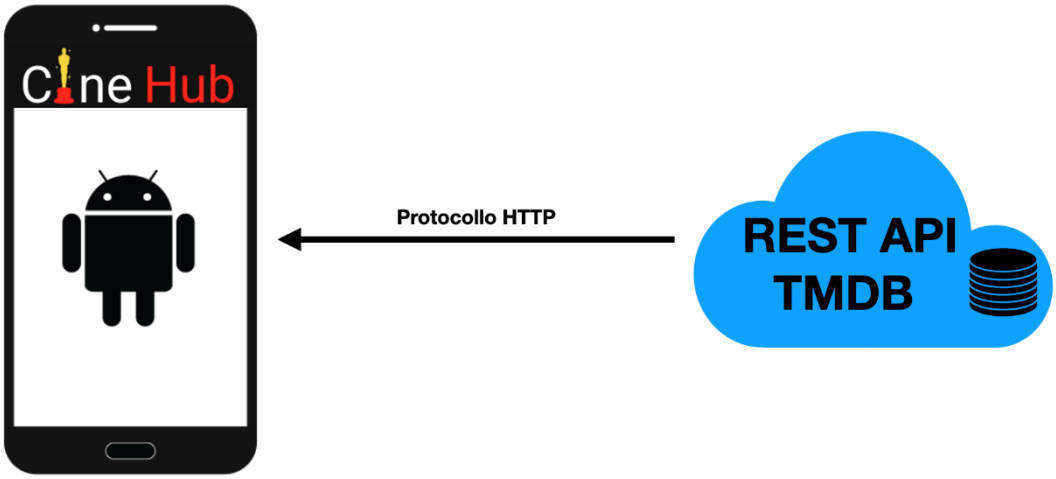
Presa visione di quelli che erano gli aspetti dell'applicazione da migliorare, abbiamo cambiato la struttura dell'applicazione, tenendo conto dei suggerimenti presenti sul report fornitoci dell'esperto.

Alla fine di questo documento troverete il Report completo e dettagliato scritto dal nostro UX Researcher.

ARCHITETTURA

Andiamo ad analizzare l'architettura del sistema partendo da come interagisce l'applicazione con il mondo esterno e, successivamente, analizzando come interagiscono le varie componenti interne tra loro.

ARCHITETTURA ESTERNA DELL'APP



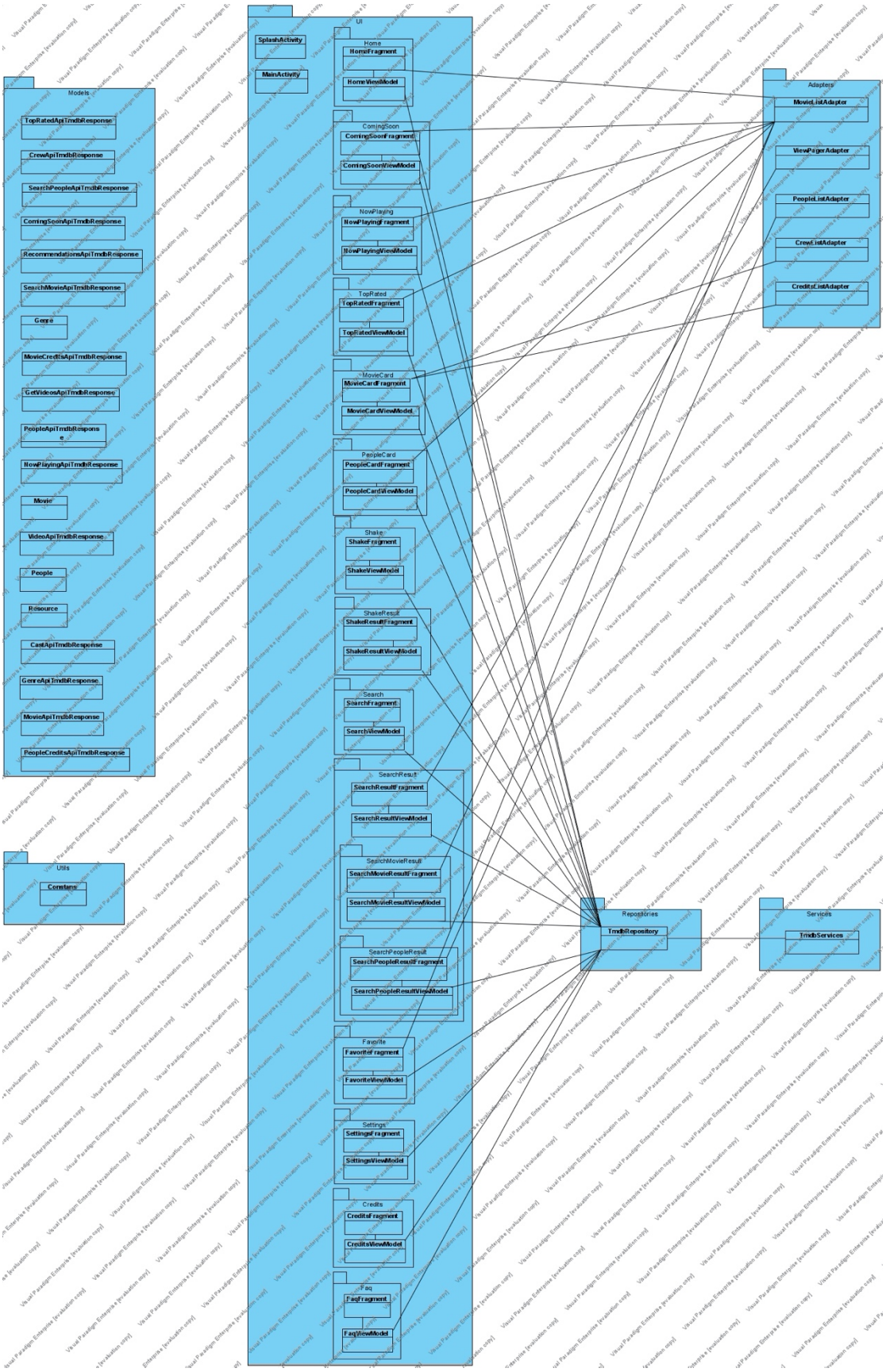
L'applicazione non dispone di un DataBase interno, ma utilizza solamente delle API REST messe a disposizione dal sito TMDB.

Ci interfacciamo con l'API tramite delle chiamate HTTP utilizzando una libreria JAVA chiamata Retrofit.

Nel prossimo paragrafo spiegheremo nel dettaglio il funzionamento dei componenti interni.

ARCHITETTURA INTERNA DELL'APP

Per quanto riguarda l'architettura interna dell'applicazione abbiamo cercato di mantenere un basso accoppiamento, dando delle precise responsabilità ad ogni nostro componente.



Quello qui sopra è il nostro UML semplificato che rappresenta tutte le componenti che abbiamo realizzato.

La nostra applicazione ha due sole Activity: la prima è lo splash screen e viene visualizzata durante il caricamento iniziale, mentre la seconda è la MainActivity, la quale contiene la Bottom Navigation Menu e un contenitore dove gestiamo i vari Fragment.

I Fragment presentano tutti delle strutture simili e hanno la responsabilità di visualizzare i contenuti nel layout e di settare i vari componenti di cui abbiamo bisogno.

Ogni Fragment è affiancato da un View Model, il quale ha la responsabilità di recuperare e mantenere i dati da visualizzare.

Il View Model fa una chiamata al Repository per poter ottenere i dati necessari.

Il Repository ha la responsabilità di gestire le risposte delle chiamate sia in caso di successo che di insuccesso.

A sua volta il Repository si appoggia al Service, nel quale sono definite tutte le possibili chiamate che possiamo fare.

Le chiamate avvengono tutte in sicurezza passando il Bareer Token nell'header.

Andiamo a vedere il funzionamento dei Fragment dove sono state utilizzate delle implementazioni degne di nota.

Nel HomeFragment possiamo visualizzare i film presenti momentaneamente in sala, i film con la valutazione migliore ed i film che prossimamente sbarcheranno al cinema.

Per effettuare le chiamate abbiamo bisogno di tre parametri fondamentali, il Parental Control, la lingua ed il Paese.

Il Parental Control ed il Paese vengono settati nel SettingsFragment e salvati nelle Shared Preferences, per poter permettere all'HomeFragment di recuperarli al momento opportuno.

La lingua verrà impostata in base a quella selezionata nelle impostazioni del dispositivo dell'utente.

Questi parametri vengono utilizzati per quasi tutte le chiamate.

Cliccando sulla locandina di un film verremo indirizzati nel Fragment chiamato Scheda Film nel quale visualizzeremo le informazioni di quella pellicola.

All'interno della Scheda Film abbiamo implementato una funzionalità che permette all'utente di aggiungere un film alla sezione "I TUOI PREFERITI". Per gestire questa funzione abbiamo utilizzato le Shared Preferences, salvandoci l'ID del film, il titolo ed il path della locandina.

La lista dei preferiti varia in base alla lingua. Abbiamo optato per questa soluzione poiché la nostra API non ci permette di fare una chiamata passandogli una lista di film.

Un'altra funzionalità che troviamo all'interno di questa pagina è la visualizzazione del trailer, se presente. Per fare ciò abbiamo sfruttato una libreria creata da Google, YouTubePlayerSupportFragment. Questa libreria ci permette di visualizzare i video di YouTube sfruttando le informazioni precedentemente recuperate tramite le API.

La peculiarità di questa applicazione è la funzione "SHAKE", che suggerisce un film all'utente. Per fare questo abbiamo sfruttato la libreria ShakeDetector. Abbiamo riscontrato delle problematiche nell'utilizzo di questa libreria su alcuni dispositivi, pertanto abbiamo pensato di rimpiazzarla con una libreria implementata da noi. A causa delle tempistiche abbiamo rimandato questa miglioria a data da destinarsi.

Per suggerire un film scegliamo in maniera casuale l'ID di un film presente nella lista dei preferiti nella Shared Preferences, nel caso non ci siano film in questa lista prendiamo sempre in maniera casuale l'ID di un film presente nei film "PIÙ VALUTATI". Successivamente usiamo questo ID per fare la chiamata getRecommendations, la quale ci restituirà una lista di film raccomandati e fra questi ne scegliamo uno in maniera casuale facendolo visualizzare all'utente.

L'applicazione prevede anche la possibilità di cercare direttamente un film o una persona. Per effettuare la ricerca si deve necessariamente

inserire una parola chiave, poiché l'API non permette di fare una ricerca a vuoto. Inoltre, l'utente ha la possibilità di selezionare dei filtri di ricerca. Si possono utilizzare due filtri diversi, uno per anno e l'altro per categoria. Il filtro per anno è un parametro passabile all'API così che il risultato della chiamata ci arriverà già filtrato. Invece, dato che il metodo dell'API che utilizziamo per la ricerca non prevede la possibilità di filtrare i generi, abbiamo implementato un nostro sistema di filtraggio che viene applicato ai risultati della chiamata.

Per visualizzare i risultati abbiamo deciso di utilizzare una TabView, la quale contiene un ViewPager utilizzato per gestire i due Fragment contenenti i risultati della ricerca. I Fragment sono due poiché uno serve per visualizzare i risultati inerenti ai film e l'altro per i risultati inerenti alle persone.

Per gestire il ViewPager abbiamo dovuto creare un ViewPagerAdapter, necessario per la gestione dei Fragment al suo interno.

Il ViewPagerAdapter fa parte del package adapters, il quale contiene gli altri adapters da noi utilizzati. Gli altri adapters sono utilizzati per popolare le varie RecyclerView presenti nei vari Fragment. Si è reso necessario creare più adapters per gestire oggetti differenti e per implementare il Lazy Loading dove necessario.

Infine, abbiamo il package dei models. Abbiamo cercato di utilizzare nelle classi più vicine alla UI, ad esempio nei ViewModel e nel Repository, dei modelli standard per evitare di servirci direttamente dei modelli che utilizziamo per effettuare le chiamate nel Service.

Questa scelta è stata fatta per facilitare l'integrazione dell'applicazione con nuove API. Questo pattern implementativo non è stato seguito nelle sezioni fortemente legate al funzionamento dell'API su cui si appoggia.

CONCLUSIONI

Abbiamo analizzato il mercato delle applicazioni di questo tipo, sia per prendere spunto sia per cercare eventuali mancanze.

Una mancanza che abbiamo individuato nel mercato è la possibilità di farsi suggerire un film. Per questo abbiamo implementato lo "SHAKE".

I nostri obbiettivi futuri sono quelli di aggiungere le funzionalità necessarie a rendere l'applicazione competitiva sul mercato.

Una funzionalità che vorremo aggiungere riguarda la visione dei film. Infatti, vorremmo inserire una sezione che dice all'utente dove poter visualizzare il film di suo interesse, sia che si trovi su una piattaforma di streaming, sia che si trovi al cinema. Inoltre, questa funzionalità faciliterà l'acquisto o la visione del film all'utente reindirizzandolo direttamente al sito scelto.

Un'altra funzionalità riguarda la possibilità di poter condividere la scheda di un film o di una persona dell'applicazione sui vari social (WhatsApp, Telegram, Facebook, …). Questa funzionalità permetterebbe agli utenti di condividere i propri interessi ed inoltre accrescerebbe la visibilità dell'applicazione.