TB Bagnolini Tommaso
Mat.: 0001071116
tommaso.bagnolini@studio.unibo.it

Tecnologie dei Sistemi Informatici
Progetto per Programmazione di Sistemi Mobile

MovieHunter

Scopo del Progetto

L'obiettivo del progetto "*MovieHunter*" è sviluppare un'applicazione mobile tramite linguaggio Kotlin per piattaforma Android che consenta agli utenti di cercare informazioni su film, visualizzare dettagli correlati e interagire con contenuti multimediali associati.

L'applicazione mira a fornire un'interfaccia intuitiva e funzionalità più avanzate per migliorare l'esperienza dell'utente nella scoperta e nell'esplorazione del mondo cinematografico.

Mockup e Design UI

Prima di sviluppare l'applicazione, è stato realizzato un **mockup** dell'interfaccia utente con **Figma**, per progettare un layout intuitivo e accattivante. Il design segue i principi di material design e include:

- Schermata home con barra di ricerca per filtrare i contenuti.
- Lista di film con immagini e titoli.
- Pagina di login utente con possibilità di scattarsi una foto.
- Pagina dettagli film con informazioni sulla trama.
- Visualizzazione di GIF correlate per un'esperienza più interattiva.
- Pagina per inserimento dati di spedizione (indirizzo)
- Pagina di conferma ordine

(Il link per il mockup è recuperabile nella sezione "Description" della repository GitHub)

Struttura

La struttura del progetto è organizzata secondo il principio della "*separazione dei principi*" (Separation of Concerns), garantendo una chiara distinzione tra le diverse componenti dell'applicazione.

Di seguito è riportata la struttura principale delle cartelle e dei file:

- ***app/*** → Contiene il codice sorgente principale dell'applicazione.
- ***src/*** → Include i file sorgente organizzati in base ai package.
- ***main/*** → Directory principale per il codice dell'applicazione.
- ***java/com/moviehunter/*** → Contiene il codice Kotlin dell'applicazione.
- **ui/** → contiene le attività e i fragment dell'interfaccia utente.
- **data/** → gestione dei dati.
- **network/** → gestione delle API.
- **viewmodel/** → gestione dello stato dell'app.
- ***res/*** → Include le risorse dell'applicazione come layout XML, immagini, stringhe e valori in generale.
- ***AndroidManifest.xml*** → File di manifest dell'applicazione che definisce componenti essenziali e permessi richiesti per garantirne il funzionamento.
- ***assets/*** → Contiene risorse grafiche, principalmente screenshots generati durante la fase di utilizzo dell'app, con lo scopo di essere inseriti all'interno di tale documento
- ***gradle/*** → Include gli script di build e le configurazioni di Gradle.
- ***build.gradle.kts*** → Script di build principale per la configurazione del progetto.
- ***gradle.properties*** → File di configurazione per le proprietà di Gradle.
- **settings.gradle.kts** → Definisce le impostazioni del progetto e include i moduli necessari.

Punti di Forza

Tale struttura presenta alcuni punti di forza, come ad esempio:

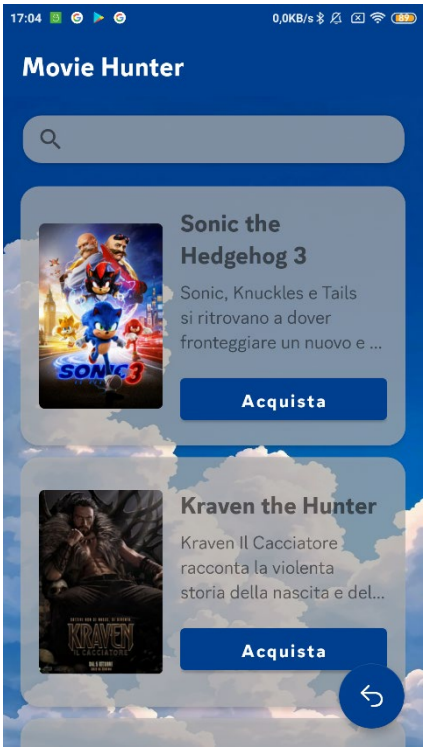
- ***Architettura Modulare***: L'applicazione è strutturata in moduli distinti, ciascuno responsabile di una specifica funzionalità, facilitando la manutenzione e l'estensibilità del codice.
- ***Utilizzo dei Componenti di Lifecycle***: L'implementazione di ViewModel e LiveData garantisce una gestione efficiente del ciclo di vita delle componenti UI, migliorando le prestazioni e la reattività dell'applicazione.
- ***Gestione delle Coroutines***: L'uso delle coroutines di Kotlin permette una gestione asincrona efficiente, separando le operazioni sul Main Thread da quelle in Background, migliorando la responsività dell'applicazione.
- ***Integrazione con API Remote***: L'applicazione effettua chiamate a più API remote per il recupero dinamico dei dati da visualizzare

TB Bagnolini Tommaso
Mat.: 0001071116
tommaso.bagnolini@studio.unibo.it

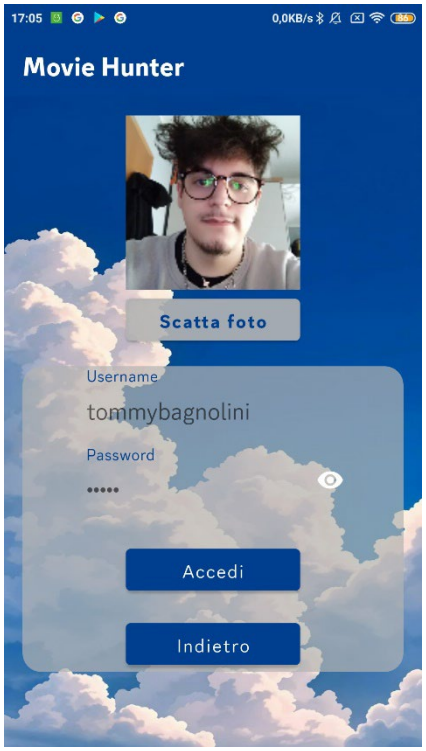
Pagine

Di seguito riportate le pagine (Activities) di **MovieHuter**:

HomePage

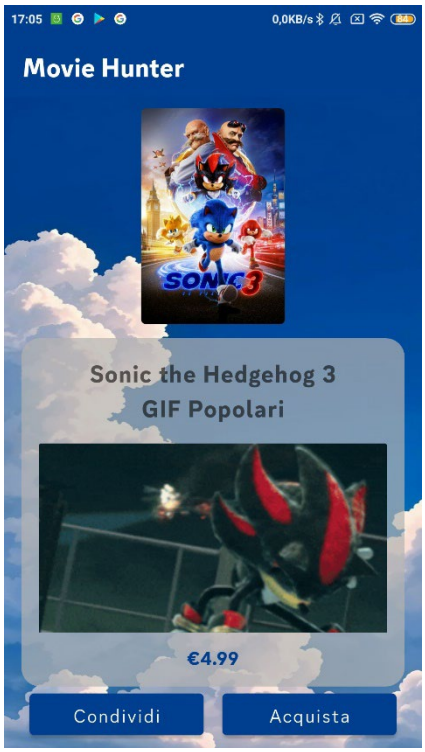


Login w/ Credentials

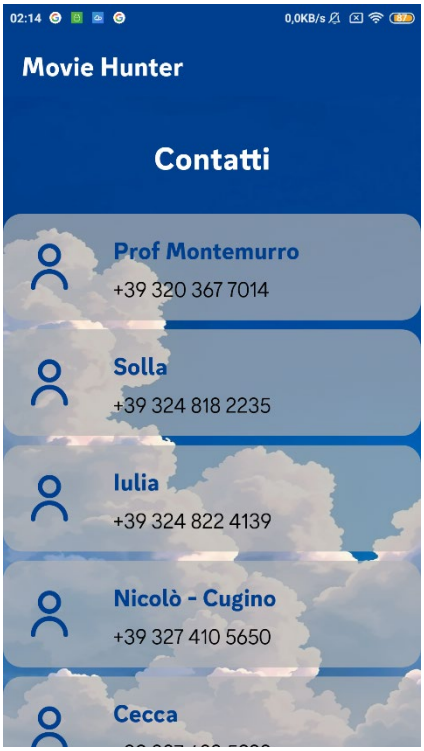


Overview

Overview w/ GIFs

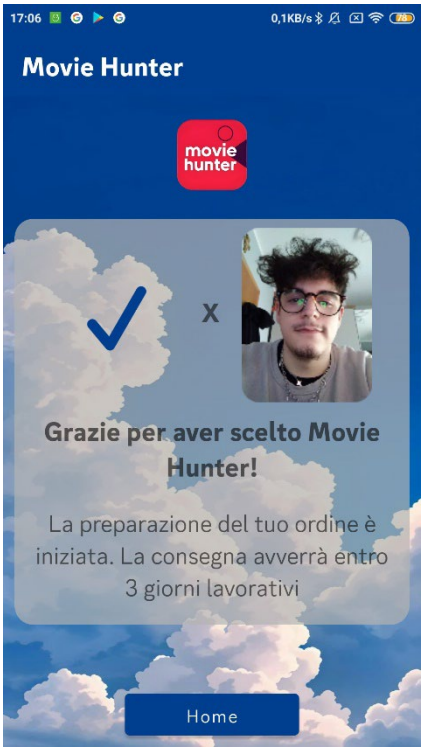


Contacts



Shipping Page

Ringraziamenti



In sintesi

Ecco un quadro generale delle funzionalità supportate da MovieHunter:

Funzionalità di Base:

✅ Visualizzazione dettagli film → informazioni sulla trama.
✅ Navigazione tra schermate → grazie al Navigation Component.
✅ Gestione del ciclo di vita → ViewModel per gestire dati senza perdere stato.
✅ Chiamate API a The Movie Database → per recuperare info sui film.
✅ Chiamate API a endpoint SEARCH → per filtrare i film in base al titolo.

Funzionalità Avanzate:

🚀 LiveData e ViewModel → gestione dati reattiva ed efficiente.
🚀 Gestione delle immagini con Glide → caricamento ottimizzato dei banner dei film.
🚀 Uso di Retrofit e Gson → chiamate API strutturate e parsing JSON efficiente.
🚀 RecyclerView per liste di film → scorrimento fluido e ottimizzato.
🚀 Integrazione con API Giphy → mostra GIF correlate ai film.
🚀 Contatti → possibilità di condividere titoli con i propri contatti

Possibili Upgrade Futuri:

🛠️ Cache locale con Room → per accesso offline ai film già cercati.
🛠️ Notifiche personalizzate → per avvisare su nuovi film in uscita.
🛠️ Salvataggio film preferiti → con database locale.