Whats2Watch

Panoramica completa

Il progetto Whats2Watch-Kotlin è un'applicazione basata su Kotlin che fornisce una piattaforma completa per gli utenti per scoprire e recensire film. Il progetto è costruito utilizzando un’architettura modulare, con moduli separati per l'interfaccia utente, l'archiviazione dei dati e la comunicazione di rete.

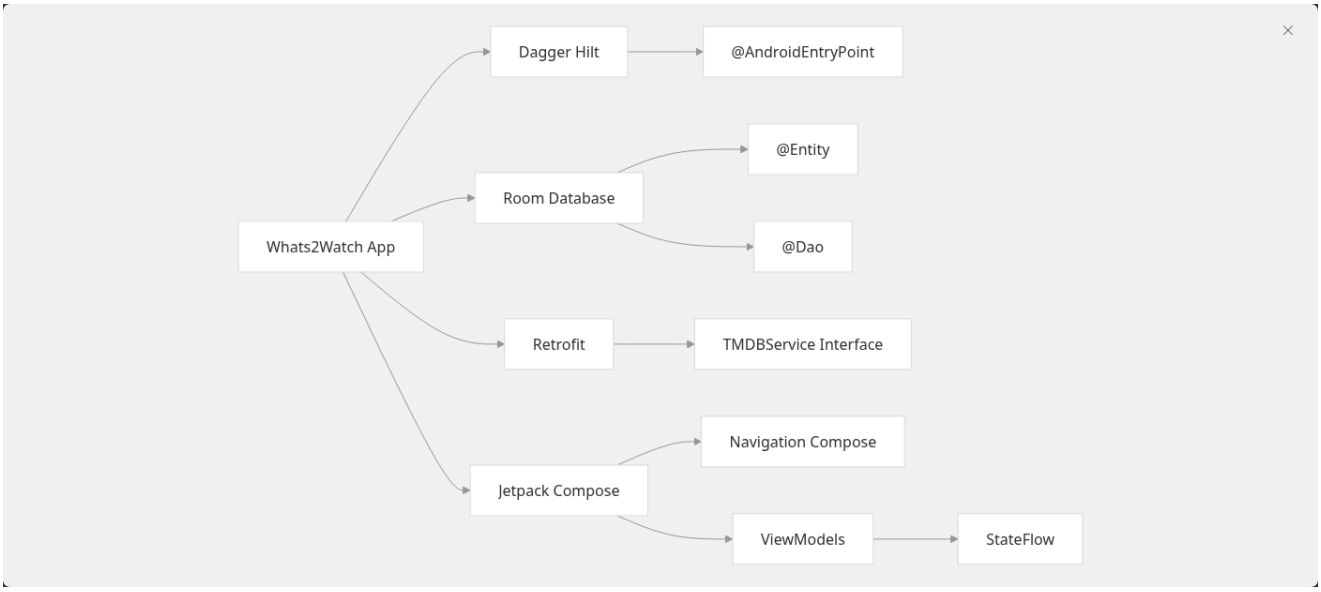
Struttura del progetto

Il progetto è organizzato nei seguenti moduli:

- `app` : Questo modulo contiene l'interfaccia utente e la logica dell'applicazione.
- `di` : Questo modulo contiene la configurazione per l'iniezione delle dipendenze dell'applicazione.
- `model` : Questo modulo contiene i modelli di dati e le entità utilizzate nell'applicazione.
- `repositories` : Questo modulo contiene gli oggetti di accesso ai dati e i repository utilizzati per interagire con l'archiviazione dei dati.
- `ui` : Questo modulo contiene i componenti dell'interfaccia utente e i temi.
- `viewmodels` : Questo modulo contiene i view model utilizzati per gestire lo stato dell'applicazione e la logica di business.

Architettura

L'architettura dell'applicazione si basa sul pattern Model-View-ViewModel (MVVM), con uno strato separato per l'accesso e l'archiviazione dei dati. L'architettura può essere rappresentata utilizzando il seguente diagramma mermaid corretto:



Funzionalità

L'applicazione offre le seguenti funzionalità:

- Registrazione e login degli utenti
- Scoperta e recensione di film
- Creazione e gestione di stanze
- Sistema di valutazione e recensioni

Architettura e progettazione del sistema

Panoramica

Il progetto Whats2Watch-Kotlin è un'applicazione Android costruita utilizzando Kotlin. Il progetto segue il pattern architettonico Model-View-ViewModel (MVVM).

Architettura di alto livello

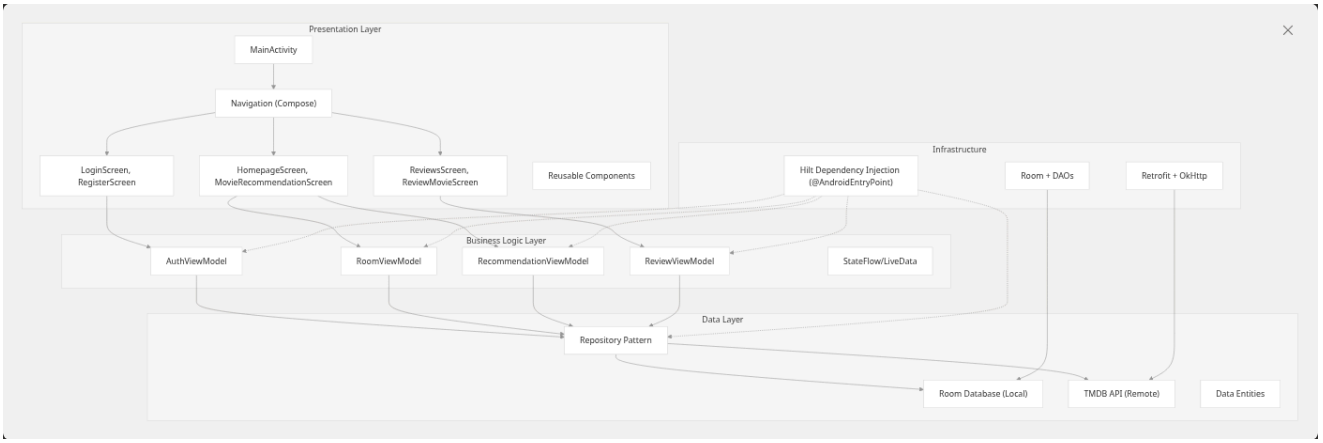
L'applicazione può essere suddivisa nei seguenti livelli:

- **Livello di Presentazione**: Questo livello contiene l'interfaccia utente e la logica dell'esperienza utente. È implementato utilizzando Kotlin e utilizza l'SDK Android.
- **Livello di Logica di Business**: Questo livello contiene la logica di business dell'applicazione. È responsabile di gestire l'input dell'utente, aggiornare il modello di dati e notificare il livello di presentazione delle modifiche.
- **Livello di Accesso ai Dati**: Questo livello è responsabile della gestione dell'archiviazione e del recupero dei dati. Utilizza la libreria Room per interagire con il database locale.

Componenti e interazioni

I seguenti componenti sono utilizzati nell'applicazione:

- **MainActivity**: Questo è il punto di ingresso principale dell'applicazione. È responsabile della configurazione della navigazione e dell'inizializzazione dei componenti necessari.
- **ViewModel**: Queste classi contengono la logica di business dell'applicazione. Interagiscono con il livello di accesso ai dati per recuperare e aggiornare i dati.
- **Repository**: Queste classi incapsulano la logica di accesso ai dati. Forniscono un livello di astrazione tra i view model e il livello di accesso ai dati.
- **Data Model**: Queste classi rappresentano le entità di dati utilizzate nell'applicazione. Vengono utilizzate per memorizzare e recuperare i dati dal database locale.



Decisioni chiave di progettazione

- **Architettura MVVM**: L'applicazione utilizza il pattern architetturale MVVM per separare la logica di presentazione dalla logica di business.
- **Libreria Room**: L'applicazione utilizza la libreria Room per interagire con il database locale.
- **Kotlin Coroutines**: L'applicazione utilizza le coroutine di Kotlin per gestire le operazioni asincrone.

Flussi di lavoro e flussi di dati principali

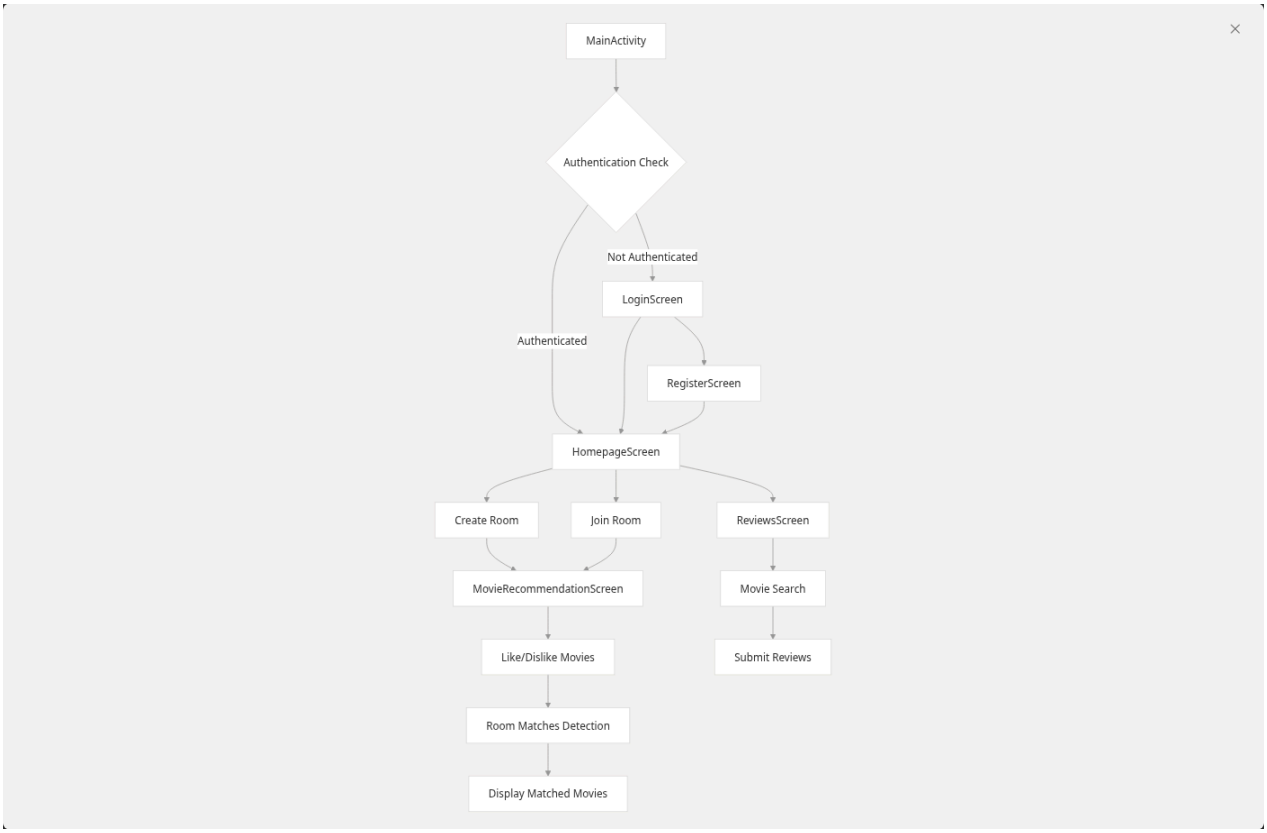
L'applicazione Whats2Watch-Kotlin ha diversi flussi di lavoro e flussi di dati principali, inclusi autenticazione utente, navigazione film, creazione di recensioni e creazione di stanze.

Autenticazione utente

Il flusso di autenticazione utente comprende i seguenti passaggi:

1. L'utente apre l'applicazione e clicca sul pulsante "Login".
2. L'utente viene reindirizzato alla schermata di login, dove inserisce le proprie credenziali.
3. Le credenziali vengono inviate al server per la verifica.

4. Se le credenziali sono valide, l'utente viene reindirizzato alla homepage.



Navigazione film

Il flusso di navigazione film comprende i seguenti passaggi:

1. L'utente apre l'applicazione e clicca sul pulsante "Film".
2. L'utente viene reindirizzato alla schermata di navigazione film, dove può sfogliare un elenco di film.
3. L'utente può filtrare i film per genere, valutazione o data di rilascio.
4. L'utente può cliccare su un film per visualizzarne i dettagli.

Creazione di una recensione

Il flusso di creazione di una recensione comprende i seguenti passaggi:

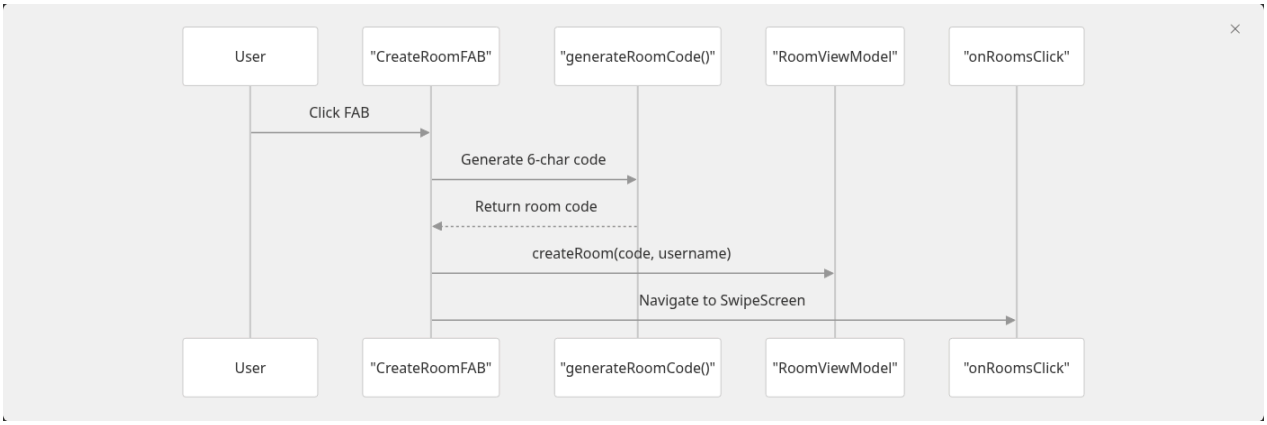
1. L'utente apre l'applicazione e clicca sul pulsante "Recensioni".
2. L'utente viene reindirizzato alla schermata di creazione recensione, dove può inserire la propria recensione.
3. L'utente può valutare il film e aggiungere un commento.
4. La recensione viene inviata al server per l'archiviazione.

Creazione di una stanza

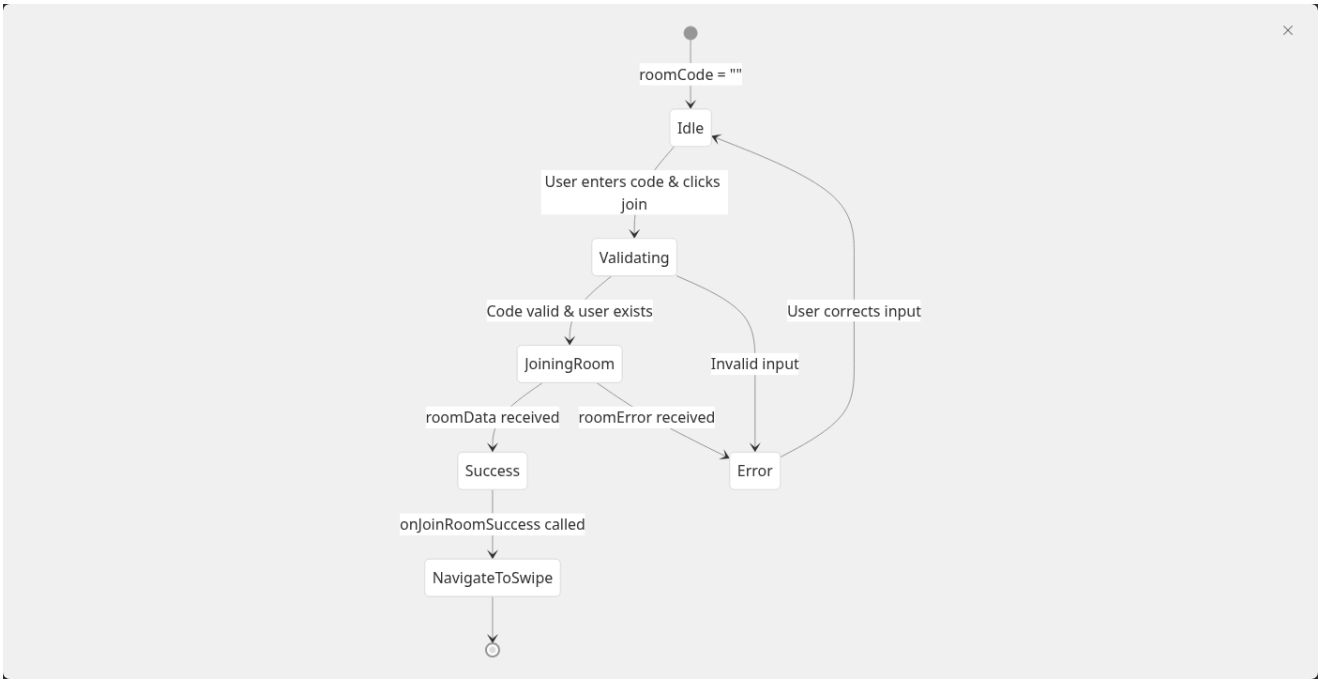
Il flusso di creazione di una stanza comprende i seguenti passaggi:

1. L'utente apre l'applicazione e clicca sul pulsante "Stanze".

2. L'utente viene reindirizzato alla schermata di creazione stanza, dove può inserire i dettagli della stanza.
3. L'utente può aggiungere partecipanti alla stanza.
4. La stanza viene inviata al server per l'archiviazione.



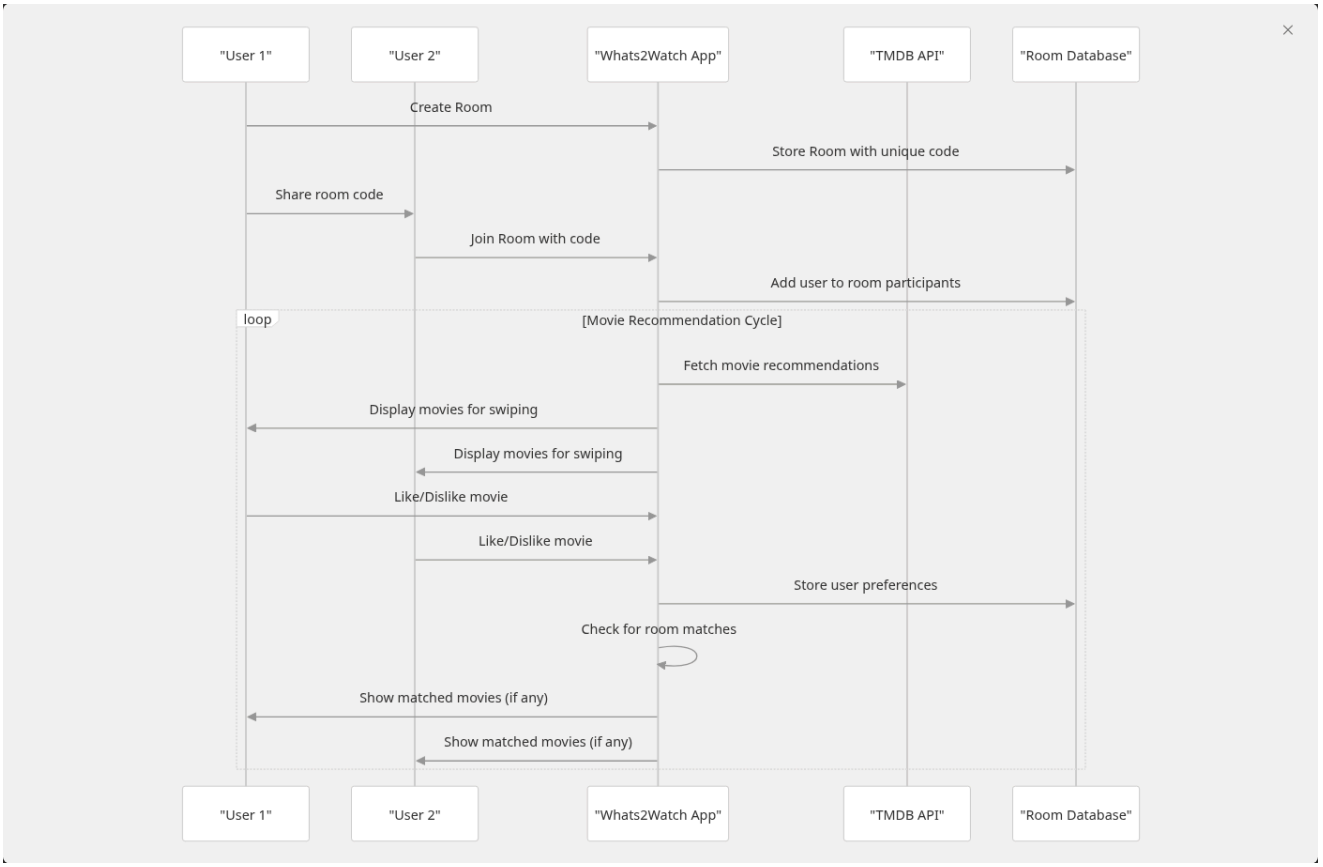
Unirsi ad una stanza esistente



Flussi di dati

I flussi di dati nell'applicazione comprendono i seguenti passaggi:

1. L'utente interagisce con l'applicazione, inviando richieste al server.
2. Il server elabora le richieste e invia risposte all'utente.
3. L'utente riceve le risposte e aggiorna lo stato dell'applicazione.



Interazioni dei componenti di alto livello

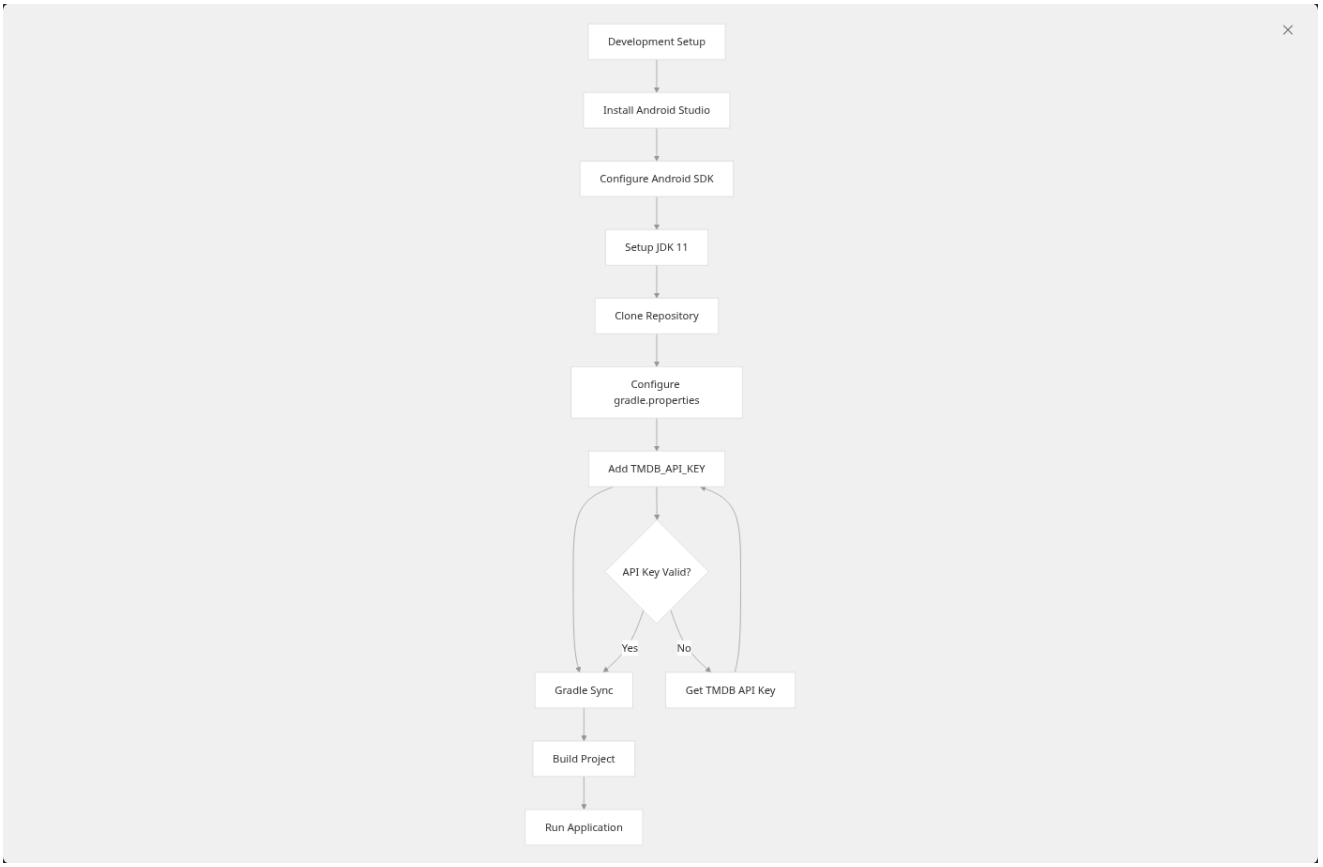
Panoramica

Il progetto Whats2Watch-Kotlin è un sistema complesso che coinvolge più componenti che interagiscono tra loro per fornire un'esperienza utente senza soluzione di continuità. Questo documento descrive le interazioni di alto livello tra questi componenti, inclusi i servizi API, gli oggetti di accesso ai dati, i view model e i componenti dell'interfaccia utente.

Servizi API

I servizi API sono responsabili del recupero dei dati da fonti esterne, come l'API TMDB. La classe `TMDBService` è il punto di ingresso principale per le interazioni con l'API e fornisce

metodi per recuperare dati sui film, dati sui generi e altre informazioni rilevanti.



Oggetti di accesso ai dati

Gli oggetti di accesso ai dati (DAO) sono responsabili della gestione dei dati memorizzati nel database locale. Le classi `MovieDao`, `PreferenceDao`, `ReviewDao`, `RoomDao` e `UserDao` forniscono metodi per inserire, aggiornare ed eliminare dati nelle rispettive tabelle.

View Model

I view model sono responsabili della gestione dei dati e della logica di business per i componenti dell'interfaccia utente. Le classi `AuthViewModel`, `RecommendationViewModel`, `ReviewViewModel` e `RoomViewModel` forniscono metodi per autenticare gli utenti, raccomandare film, inviare recensioni e gestire le stanze.

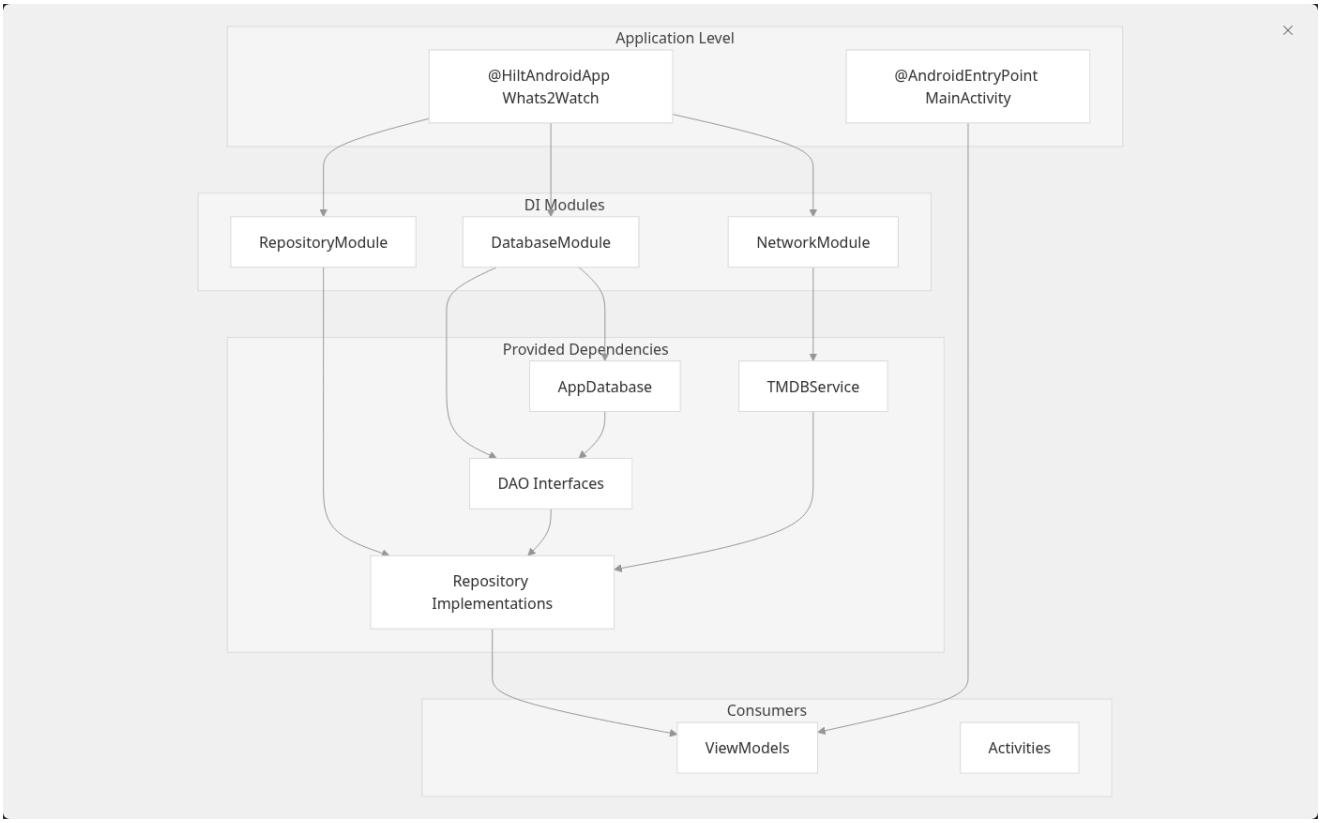
Componenti dell'interfaccia utente

I componenti dell'interfaccia utente sono responsabili della visualizzazione dei dati e forniscono un'interfaccia user-friendly per interagire con il sistema. Le classi `LoginScreen`, `RegisterScreen`, `ReviewScreen`, `RoomsScreen` e `SwipeScreen` forniscono i componenti UI necessari per le rispettive schermate.

Interazioni dei componenti

I componenti interagiscono tra loro attraverso una serie di chiamate a metodi e binding dei dati. Ad esempio, quando un utente effettua il login, `AuthViewModel` chiama `TMDBService` per recuperare le preferenze dell'utente, che vengono poi memorizzate in `PreferenceDao`.

`RecommendationViewModel` può quindi utilizzare queste preferenze per raccomandare film all'utente.



Istruzioni di configurazione

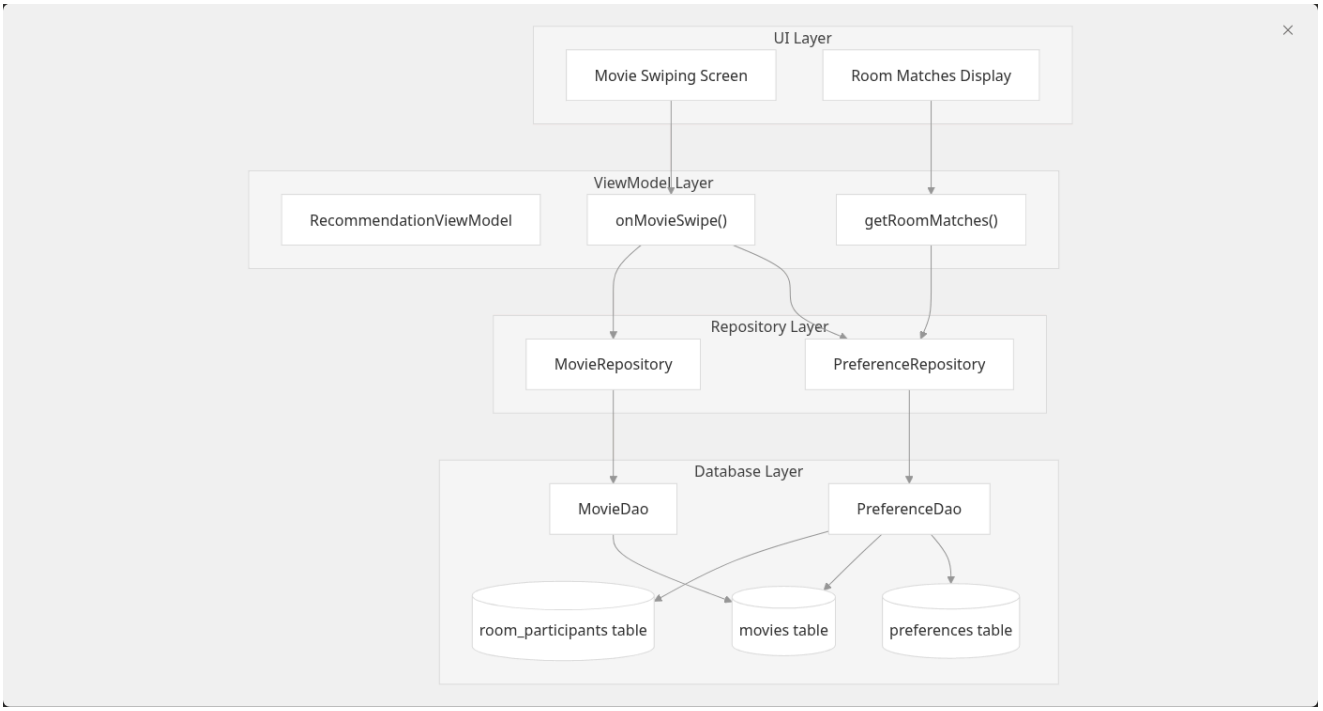
Per configurare il progetto Whats2Watch-Kotlin, seguire questi passaggi:

1. Clonare il repository da GitHub.
2. Aprire il progetto in Android Studio.
3. Compilare il progetto utilizzando il wrapper di Gradle.
4. Eseguire l'applicazione su un emulatore o dispositivo fisico.

Feature chiave - il sistema di swipe

Introduzione

Il **sistema di swipe** rappresenta il cuore dell'esperienza utente di Whats2Watch, permettendo agli utenti di scorrere e valutare film in modo intuitivo attraverso gesti naturali. Questo sistema implementa un meccanismo sofisticato di caricamento asincrono dei contenuti che garantisce un'esperienza fluida e reattiva.



Architettura del Sistema

Componenti Principali

Il sistema di swipe si articola attorno a tre componenti fondamentali che collaborano per fornire un'esperienza utente ottimale:

SwipeScreen: Rappresenta il contenitore principale che orchestra l'intera esperienza di swipe. Questa schermata gestisce lo stato globale dell'applicazione e coordina le interazioni tra i vari componenti, mantenendo traccia del contesto della stanza e dell'utente corrente.

SwipeableMovieCard: Costituisce il cuore dell'interfaccia utente, implementando la logica di riconoscimento dei gesti e la visualizzazione dinamica dei film. Questo componente trasforma i movimenti dell'utente in azioni concrete, fornendo feedback visivo immediato attraverso animazioni fluide.

RecommendationViewModel: Funge da motore intelligente del sistema, gestendo tutta la logica di business relativa al caricamento, alla cache e alla personalizzazione dei suggerimenti cinematografici.

Flusso di Dati

Il flusso di dati segue un pattern reattivo ben definito che garantisce coerenza e performance ottimali. I suggerimenti vengono gestiti attraverso StateFlow, permettendo aggiornamenti reattivi dell'interfaccia utente ogni volta che nuovi contenuti diventano disponibili. Questo approccio elimina la necessità di polling manuale e garantisce che l'utente veda sempre i contenuti più aggiornati.

Sistema di Caricamento Asincrono

Strategia di Caching Intelligente

Il cuore dell'efficienza del sistema risiede nella sua strategia di caching multi-livello. Il sistema mantiene una cache locale (`cachedSuggestions`) che funziona come buffer tra le richieste di rete e l'interfaccia utente. Questa cache viene popolata in modo asincrono e garantisce che l'utente possa continuare a interagire con l'applicazione anche durante il caricamento di nuovi contenuti.

Il sistema implementa un meccanismo di validità temporale della cache (30 secondi) che bilancia perfettamente freschezza dei contenuti e performance. Quando la cache è ancora valida, il sistema può fornire immediatamente i contenuti richiesti senza alcuna latenza di rete.

Meccanismo di Precaricamento Predittivo

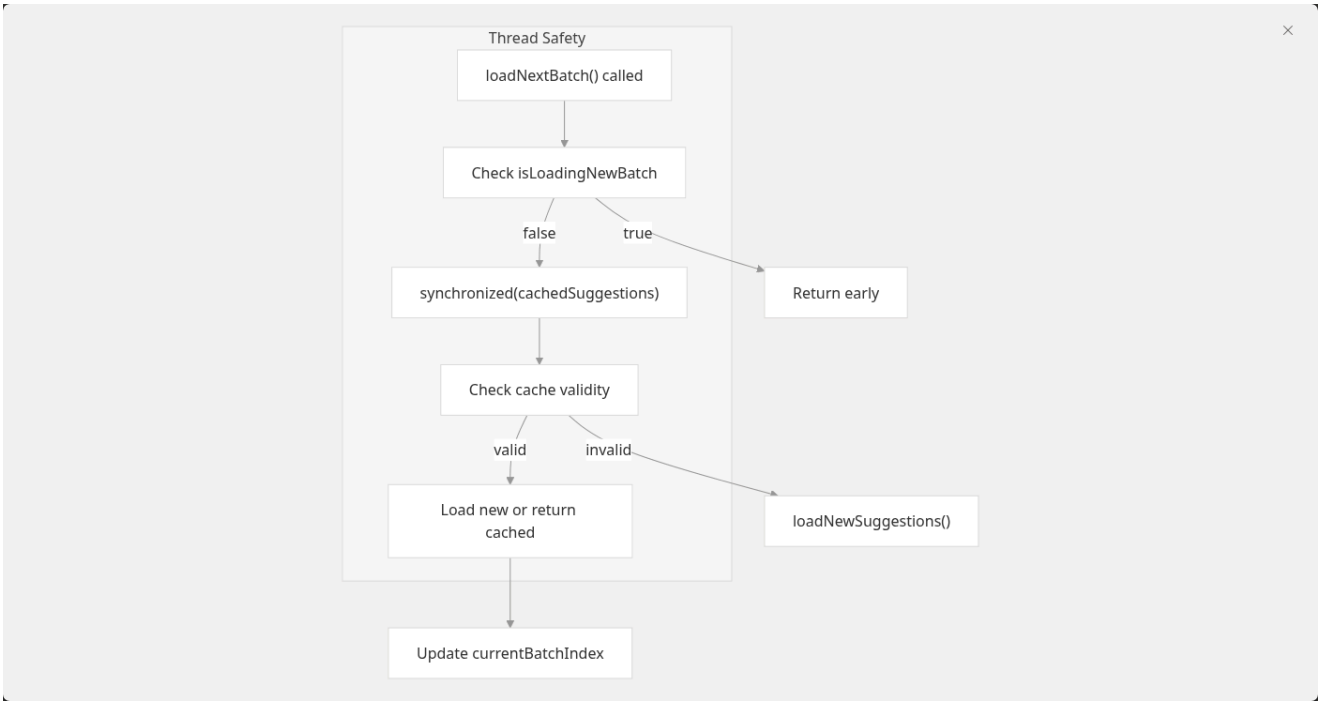
Una delle caratteristiche più sofisticate del sistema è il precaricamento predittivo. Il sistema monitora costantemente il progresso dell'utente attraverso i contenuti e inizia a caricare il prossimo batch quando l'utente si avvicina alla fine di quello corrente. Specificamente, quando l'utente ha visionato tutti i film tranne gli ultimi 10, il sistema inizia automaticamente il caricamento del batch successivo.

```
// Logica di precaricamento predittivo
if (batchCount > suggestions.size - 10) {
    onLoadNextBatch()
    onBatchCountChange(0)
}
```

Gestione della Concorrenza

Per prevenire condizioni di gara e garantire la coerenza dei dati, il sistema implementa un sofisticato meccanismo di controllo della concorrenza. La variabile `isLoadingNewBatch` agisce come semaforo, impedendo l'avvio di operazioni di caricamento multiple simultanee. Questo è particolarmente importante in scenari dove l'utente potrebbe triggerare rapidamente multiple richieste di contenuto.

L'accesso alla cache è protetto da blocchi `synchronized` , garantendo che le operazioni di lettura e scrittura siano atomiche anche in presenza di thread multipli. Questo approccio elimina la possibilità di stati inconsistenti e garantisce l'integrità dei dati.



Algoritmo di Raccomandazione

Strategia Multi-Sorgente

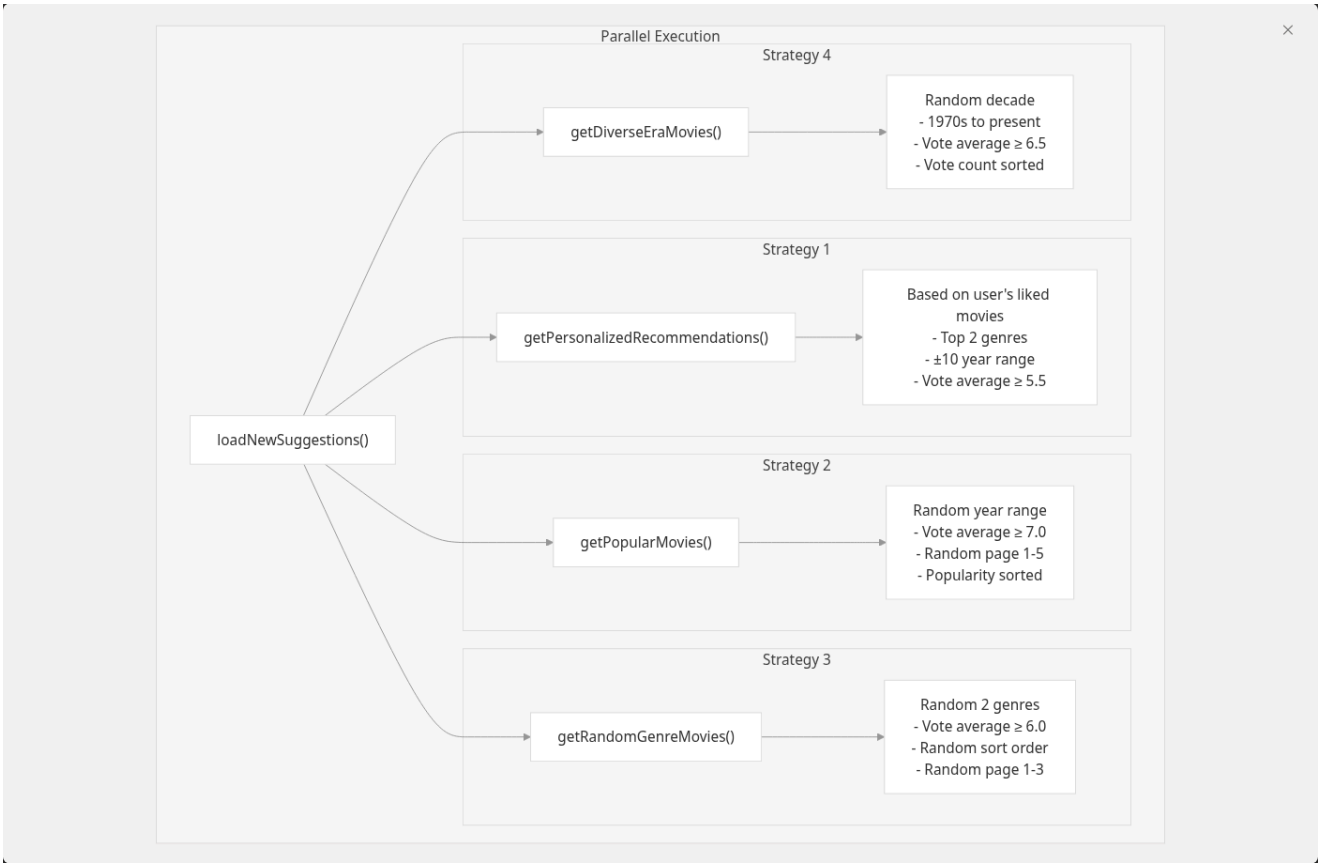
Il sistema di raccomandazione implementa una strategia diversificata che combina multiple sorgenti di contenuto per garantire varietà e rilevanza. Questa approccio multi-dimensionale assicura che gli utenti ricevano sempre contenuti interessanti, indipendentemente dalle loro preferenze pregresse.

Raccomandazioni Personalizzate: Quando l'utente ha espresso preferenze attraverso i "like", il sistema analizza questi dati per identificare pattern comportamentali. L'algoritmo estrae i generi più apprezzati e utilizza l'anno medio dei film graditi per creare un profilo delle preferenze temporali dell'utente.

Contenuti Popolari: Il sistema integra film attualmente popolari, ma con un twist intelligente. Invece di mostrare sempre gli stessi blockbuster, randomizza l'anno di riferimento per garantire diversità temporale e esporre gli utenti a gemme nascoste di diverse epoche.

Esplorazione per Genere: Per stimolare la scoperta di nuovi contenuti, il sistema seleziona casualmente generi cinematografici e presenta i migliori film di quelle categorie, permettendo agli utenti di esplorare territori inesplorati.

Diversità Temporale: Il sistema include deliberatamente film di diverse decadi, dalla Golden Age degli anni '70 ai contenuti più recenti, garantendo un'esperienza cinematografica ricca e variegata.



Parallelizzazione delle Richieste

Per ottimizzare le performance, tutte le strategie di raccomandazione vengono eseguite in parallelo utilizzando coroutine Kotlin. Questo approccio riduce drasticamente i tempi di caricamento, permettendo al sistema di aggregare contenuti da multiple sorgenti simultaneamente.

```
val results: List<List<Movie>> = coroutineScope {
    val deferredPersonal = async{
        getPersonalizedRecommendations(...)
    }
    val deferredPopular = async { getPopularMovies(...) }
    val deferredGenre = async{
        getRandomGenreMovies(...)
    }
    val deferredEra = async { getDiverseEraMovies(...) }

    listOf(deferredPersonal, deferredPopular, deferredGenre,
deferredEra).awaitAll()
}
```

Sistema di Scoring Intelligente

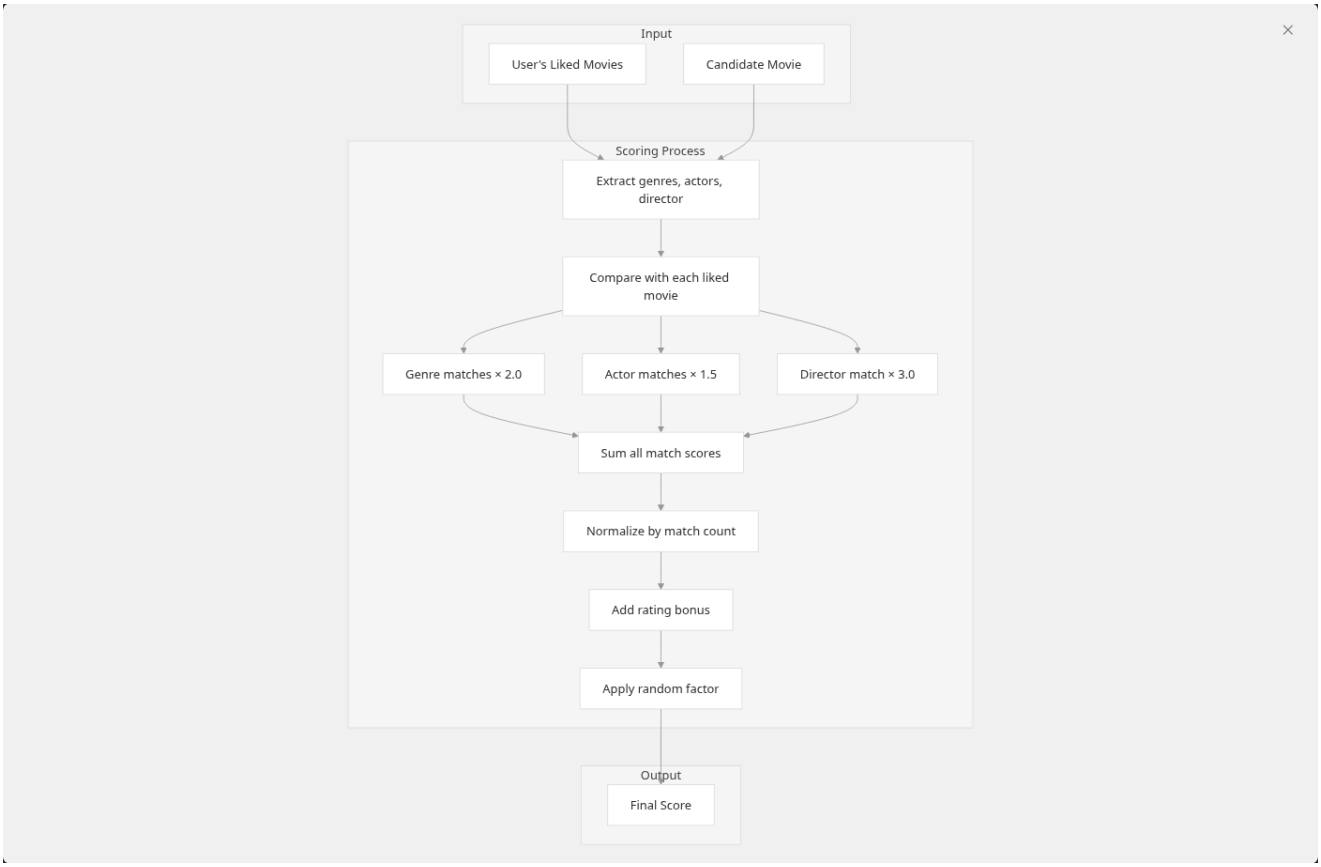
Quando l'utente ha espresso preferenze, il sistema calcola un punteggio personalizzato per ogni candidato utilizzando un algoritmo sofisticato che considera molteplici fattori:

Affinità di Genere: I generi condivisi tra film graditi e candidati ricevono un peso elevato, riconoscendo che le preferenze di genere sono spesso il fattore più importante nelle scelte cinematografiche.

Connessioni Artistiche: Il sistema premia film che condividono attori o registi con contenuti precedentemente apprezzati, riconoscendo che le preferenze artistiche sono spesso coerenti.

Qualità Generale: Il rating IMDb contribuisce al punteggio finale, assicurando che vengano privilegiati contenuti di qualità riconosciuta.

Fattore di Casualità: Un elemento random viene introdotto per prevenire risultati troppo predicibili e mantenere un elemento di sorpresa nelle raccomandazioni.



Interfaccia Utente e Gestione dei Gesti

Riconoscimento dei Gesti di Swipe

Il componente `SwipeableMovieCard` implementa un sofisticato sistema di riconoscimento dei gesti che trasforma i movimenti naturali dell'utente in azioni concrete. Il sistema utilizza la soglia dinamica del 30% della larghezza dello schermo per determinare quando un gesto deve essere interpretato come swipe definitivo.

Durante il trascinamento, il sistema fornisce feedback visivo continuo attraverso trasformazioni grafiche che includono traslazione, rotazione e cambiamenti di opacità.

Questo feedback immediato aiuta l'utente a comprendere l'impatto delle sue azioni prima che vengano finalizzate.

Animazioni Fluide e Responsive

Il sistema implementa animazioni fluide che reagiscono in tempo reale ai gesti dell'utente. La rotazione della card è proporzionale allo spostamento orizzontale, creando un effetto naturale che imita il comportamento di oggetti fisici. L'opacità viene modificata quando l'utente si avvicina alla soglia di swipe, fornendo un feedback visivo chiaro dell'imminente azione.

Gestione dello Stato e Persistenza

Stato dell'Applicazione

Il sistema mantiene uno stato complesso che include i suggerimenti correnti, gli errori eventuali, i match della stanza e i film apprezzati dall'utente. Questo stato viene gestito attraverso StateFlow, garantendo reattività e coerenza in tutta l'applicazione.

La gestione dello stato è progettata per essere resiliente agli errori, con meccanismi di fallback che permettono all'applicazione di continuare a funzionare anche in presenza di problemi di rete o errori temporanei.

Persistenza delle Preferenze

Ogni azione dell'utente viene immediatamente persistita nel database, garantendo che le preferenze non vadano perse anche in caso di interruzioni inaspettate. Il sistema salva sia il film che la preferenza espressa, creando un dataset ricco che può essere utilizzato per migliorare future raccomandazioni.

La persistenza è gestita in modo asincrono per non bloccare l'interfaccia utente, garantendo che l'esperienza rimanga fluida anche durante le operazioni di scrittura sul database.

Ottimizzazioni delle Performance

Gestione della Memoria

Il sistema implementa diverse strategie per ottimizzare l'utilizzo della memoria. La cache ha una dimensione controllata e viene periodicamente pulita per prevenire accumuli eccessivi. Le immagini vengono caricate utilizzando Coil con crossfade per transizioni fluide e gestione automatica della memoria.

Riduzione delle Chiamate di Rete

Attraverso la cache intelligente e il precaricamento predittivo, il sistema minimizza significativamente il numero di chiamate di rete necessarie. La validità temporale della cache

garantisce che i contenuti rimangano freschi senza richiedere aggiornamenti continui.

Thread Safety

Tutte le operazioni critiche sono progettate per essere thread-safe, utilizzando blocchi synchronized dove necessario e garantendo che le operazioni asincrone non creino condizioni di gara. Questo è particolarmente importante nella gestione della cache condivisa e delle operazioni di database.

Gestione degli Errori

Resilienza e Fallback

Il sistema implementa una strategia robusta di gestione degli errori che permette all'applicazione di continuare a funzionare anche quando alcuni servizi non sono disponibili. Se una strategia di raccomandazione fallisce, le altre continuano a funzionare, garantendo che l'utente riceva sempre contenuti, anche se forse non ottimali.

Feedback Utente

Gli errori vengono comunicati all'utente attraverso messaggi chiari e informativi, ma non bloccanti. L'interfaccia rimane utilizzabile anche in presenza di errori, permettendo all'utente di continuare l'esperienza mentre il sistema tenta di risolvere i problemi in background.