



WhatsToWatch

10 FEBBRAIO

Ingegneria del Software e Progettazione Web
Cecili Christian 0328245
A.A. 2024/2025

Indice

- [Software Requirement Specification.....3](#)
- [User Stories.....4](#)
- [Requisiti Funzionali.....5](#)
- [Casi d'uso.....5](#)
- [Use Case Diagram.....6](#)
- [Internal Steps.....6](#)
- [Storyboards.....7](#)
- [Class Diagram.....12](#)
- [VOPC Analysis.....12](#)
- [Design Level Diagram.....13](#)
- [Design Patterns.....13](#)
- [Activity Diagram.....15](#)
- [Sequence Diagram.....16](#)
- [State Diagram.....17](#)
- [Testing.....18](#)
- [Considerazioni Finali.....21](#)

Software Requirement Specification

Introduzione

Scopo del Documento

Il presente documento ha l'obiettivo di descrivere in modo dettagliato il sistema Whats2Watch (W2W), fornendo una panoramica della sua architettura, delle funzionalità principali e dei requisiti necessari per la sua implementazione. Viene inoltre analizzato il processo di progettazione del suddetto, evidenziandone i punti di forza e le criticità, al fine di contestualizzare il progetto e delinearne il valore aggiunto.

Panoramica del Sistema

Whats2Watch (W2W) è un'applicazione progettata per facilitare la scelta di contenuti multimediali attraverso un approccio interattivo e collaborativo. Il sistema consente agli utenti di creare stanze virtuali nelle quali possono esprimere preferenze su film e serie TV mediante un meccanismo di selezione simile a quello di un'app di incontri, quindi facendo 'swipe' sul contenuto proposto. In base alle preferenze aggregate dei partecipanti, l'applicazione fornisce suggerimenti ottimizzati sfruttando un algoritmo di raccomandazione basato su criteri dei contenuti ai quali hanno reagito positivamente quali genere, attori, popolarità, periodo di uscita, regista, aziende produttrici e fornitori di tali contenuti. Quando tutti i partecipanti della stanza virtuale reagiscono positivamente allo stesso contenuto, l'applicazione notificherà le persone coinvolte avvisandole del possibile candidato da poter vedere in tv o su piattaforme streaming, quali Netflix, Prime Video, etc... Il sistema integra inoltre aggiornamenti periodici sulle valutazioni e sulla popolarità dei contenuti, garantendo suggerimenti sempre aggiornati e pertinenti.

Requisiti Hardware e Software

L'implementazione di W2W richiede una piattaforma hardware e software adeguata a garantire prestazioni ottimali e scalabilità.

Requisiti hardware:

- Server con capacità di elaborazione adeguata a gestire le richieste degli utenti e i processi di raccomandazione.
- Memoria sufficiente per archiviare e processare i dati relativi agli utenti e ai contenuti multimediali.
- Connessione di rete stabile per garantire la sincronizzazione in tempo reale delle stanze virtuali.

Requisiti software:

- Linguaggio di programmazione: Java per il backend, con l'utilizzo di librerie adeguate per la gestione delle API verso la base di dati contenente i contenuti audiovisivi e dell'elaborazione dati.
- Database relazionale per la gestione delle informazioni sugli utenti e sui contenuti.
- API di terze parti, come TMDb, per il recupero di metadati aggiornati sui film e le serie TV.

- JavaFX per la realizzazione di un'interfaccia utente reattiva e intuitiva.

Sistemi Correlati

Per comprendere meglio il contesto in cui si inserisce Whats2Watch, vengono analizzati tre sistemi esistenti che offrono funzionalità simili:

1. Teleparty

Pro:

- Sincronizzazione della riproduzione tra più utenti in tempo reale.
- Collaborazione con Netflix per la visione del contenuto scelto.
- Chat integrata per interazioni durante la visione.

Contro:

- Non fornisce suggerimenti basati sulle preferenze condivise del gruppo, è quindi limitato alla visione congiunta senza meccanismi avanzati di raccomandazione. Per cui gli utenti si recheranno su tale applicativo solo per vedere il contenuto, non per un aiuto su cosa scegliere da vedere

2. TasteDive

Pro:

- Algoritmo di raccomandazione basato sulle preferenze degli utenti del momento.
- Possibilità di scoprire contenuti in base alla propria cronologia di preferenze.

Contro:

- Mancanza di un'interazione diretta tra più utenti per selezionare contenuti in modo collaborativo. Tale sistema si limita a fornirti una lista di film in base a determinati criteri scelti.
- Banca dati non aggiornata.

3. MovieMatch

Pro:

- Sincronizzazione della riproduzione tra più utenti in tempo reale.
- Interazione diretta tra più utenti per selezionare contenuti in modo collaborativo.

Contro:

- Le stanze virtuali possono essere formate da massimo 2 persone.
- La banca dati è completa per quanto riguarda film prodotti dal mondo orientale, ma completamente scoperta su film prodotti in occidente (Europa e America).

User Stories

1. Come utente, voglio poter creare un profilo personale e selezionare i miei film preferiti, in modo da ricevere film affini ai miei gusti fin da subito.
2. Come utente, voglio poter essere notificato quando dei film hanno ricevuto una reazione positiva da tutti gli utenti nella stessa stanza, così da poter decidere quale vedere facilmente.
3. Come utente, voglio ricevere raccomandazioni di film sempre più affini ai contenuti a cui si sta reagendo positivamente nella stanza, così da poter scoprire nuove opzioni che potrebbero piacermi.

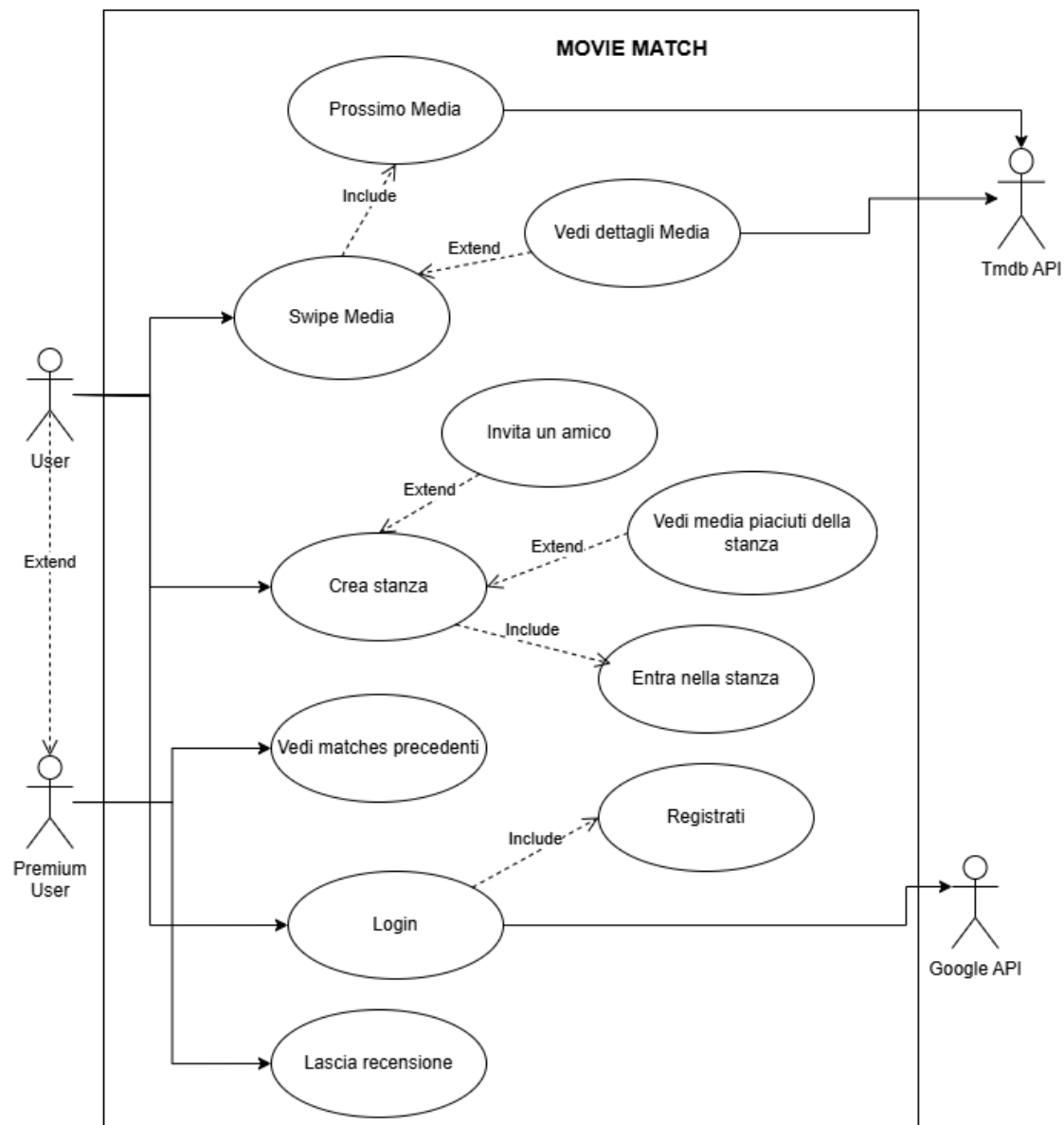
Requisiti Funzionali

1. Il sistema deve consentire agli utenti di registrarsi e creare un profilo, specificando almeno 5 film preferiti e selezionando le proprie preferenze di genere.
2. Il sistema deve consentire la creazione di stanze private in cui solo gli utenti invitati possano partecipare, assicurando che gli abbinamenti siano basati su interessi comuni relativi ai film selezionati.
3. Il sistema deve implementare un algoritmo di raccomandazione che suggerisca film sempre più affini ai gusti degli utenti all'interno di ciascuna stanza, utilizzando i film preferiti e le interazioni degli utenti nella stanza per migliorare le *raccomandazioni**.

Raccomandazione*: Una raccomandazione non è altro che un contenuto audiovisivo che si suggerisce all'utente che si basa su attributi di altri film a cui l'utente ha messo mi piace. Tali attributi sono: Genere, Regista, Anno di Uscita, Cast, Compagnie Produttrici e Fornitori Streaming di contenuti

Casi d'uso

Use Case Diagram



Internal Steps

Si è scelto di scrivere gli steps interni del caso d'uso: *'Swipe Media'*.

Passaggi interni:

1. L'utente passa alla sezione di scorrimento.
2. Il sistema visualizza un suggerimento di film con i dettagli di base (ad esempio, titolo e locandina).
3. L'utente può visualizzare maggiori dettagli sul film selezionato.
4. L'utente scorre rapidamente verso sinistra per rifiutare e verso destra per mettere mi piace al film.
5. Il sistema registra l'azione dell'utente.
6. Il sistema recupera il suggerimento del film successivo.
7. Il sistema aggiorna periodicamente il profilo delle preferenze dell'utente in base agli scorrimenti.
8. Se il film è piaciuto, il sistema controlla se è piaciuto anche agli altri utenti nella stanza.

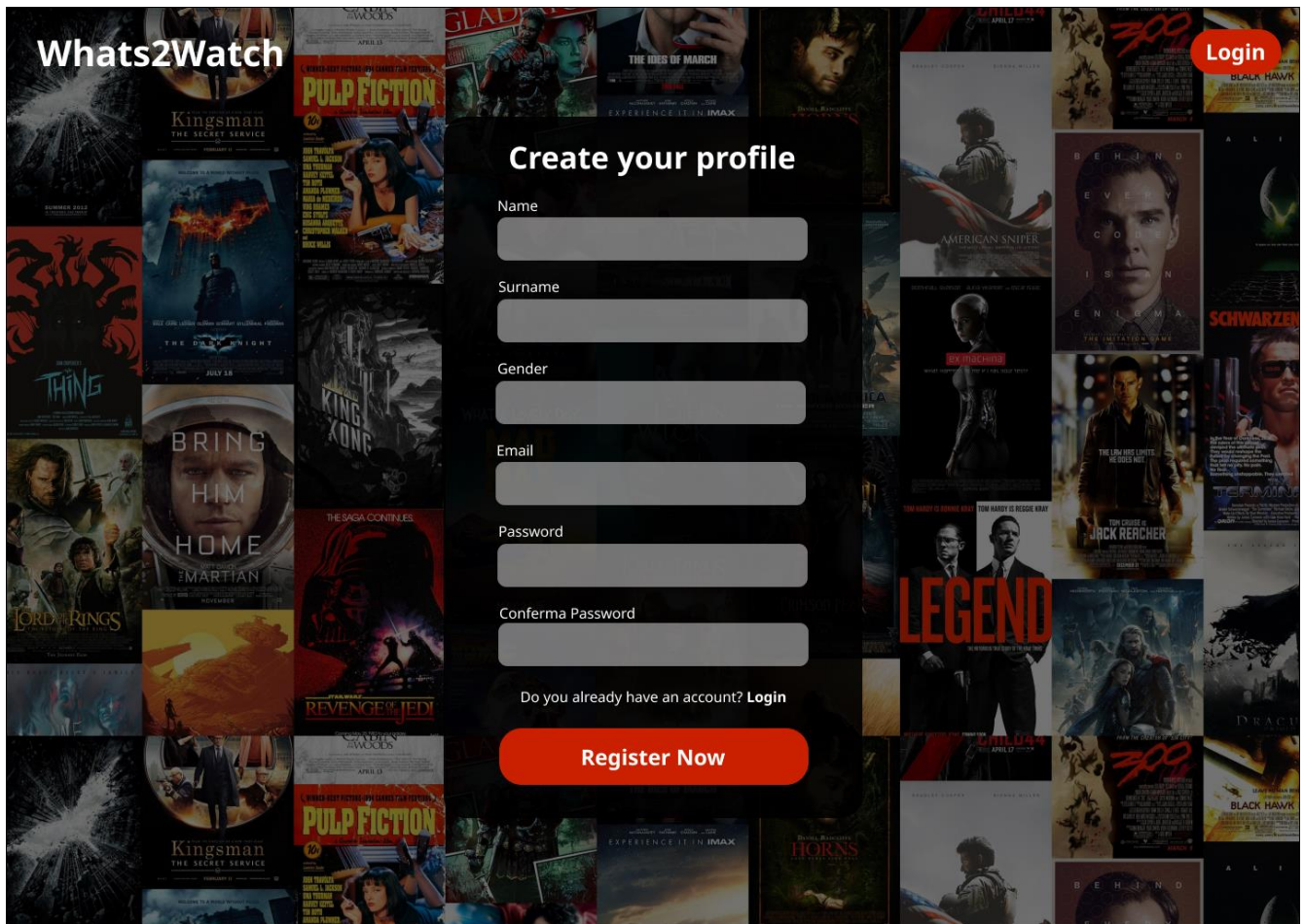
Estensioni:

- 5a. Indicatore di caricamento per scorrimento veloce - Se l'utente scorre troppo velocemente, viene visualizzato un indicatore di caricamento.
- 6a. Messaggio di fine film disponibili - Se non sono disponibili altri film, viene mostrato il messaggio "Niente più film".
- 6b. Indicatore di caricamento per ritardo nel recupero - Se si verifica un ritardo nel recupero di nuovi film, viene visualizzato un indicatore di caricamento.
- 8a. Notifica di corrispondenza - Se il film è piaciuto anche ad altri utenti nella stanza, il sistema invia una notifica a entrambi gli utenti.

Storyboards

La prima storyboard rappresenta la pagina di registrazione al sistema. Per il design è stata scelta una palette di colori ispirata a una nota piattaforma di streaming, con l'obiettivo di rendere Whats2Watch immediatamente familiare all'utente. La pagina di registrazione richiede solo le informazioni essenziali, così da ridurre al minimo i tempi necessari per completare il processo.

Una volta registrato, l'utente può accedere tramite la Storyboard di login, che mantiene gli stessi asset grafici della registrazione. Questa scelta garantisce un'interfaccia uniforme, riducendo il rischio di errori e rendendo l'esperienza più intuitiva, poiché l'utente interagisce con elementi visivi già riconosciuti.



The registration form is centered on a dark background with a collage of movie posters. The 'Whats2Watch' logo is in the top left. A 'Login' button is in the top right. The form fields are: Name, Surname, Gender, Email, Password, and Confirma Password. Below the fields is a link 'Do you already have an account? Login'. A large orange 'Register Now' button is at the bottom.

Whats2Watch

Login

Create your profile

Name

Surname

Gender

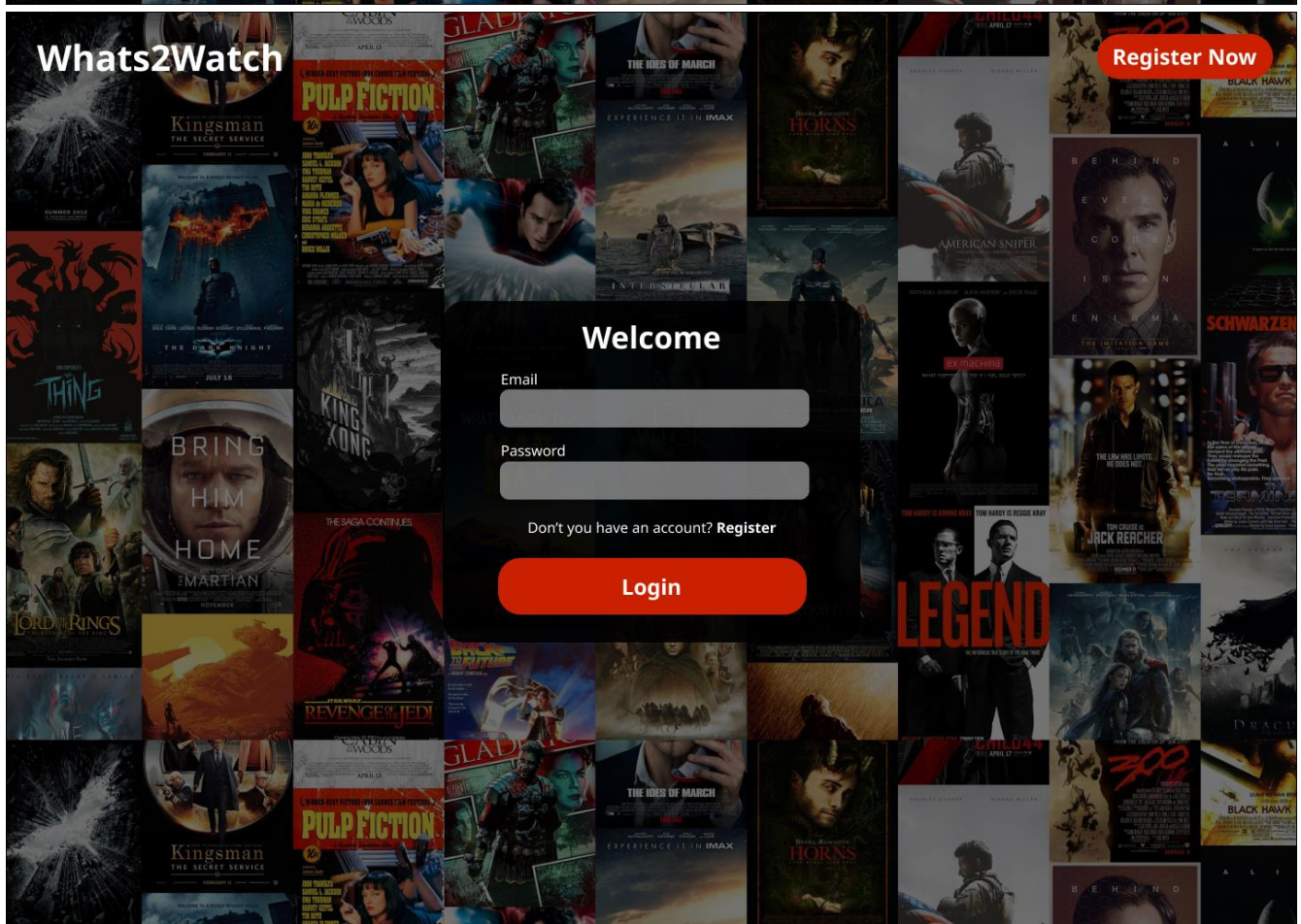
Email

Password

Confirma Password

Do you already have an account? [Login](#)

[Register Now](#)



The login form is centered on the same movie poster background. The 'Whats2Watch' logo is in the top left. A 'Register Now' button is in the top right. The form fields are: Email and Password. Below the fields is a link 'Don't you have an account? Register'. A large orange 'Login' button is at the bottom.

Whats2Watch

Register Now

Welcome

Email

Password

Don't you have an account? [Register](#)

[Login](#)

La terza Storyboard rappresenta l'homepage del sistema: da qui l'utente può arrivare a fare tutto ciò che è lo scopo principale dell'applicazione stessa. La palette di colori scelta richiama ancora una nota piattaforma di streaming, con l'obiettivo di rendere l'interfaccia più familiare e facilmente riconoscibile. In primis si trova in alto le due modalità di W2W: Film o Serie TV; dopodiché l'utente può navigare tra le stanze recenti, accedendo rapidamente a quelle attive in base ai generi e alle preferenze di visione. Inoltre, ha la possibilità di creare nuove stanze tematiche con un semplice click, scegliendo tra diverse categorie cinematografiche. La homepage include anche una sezione dedicata ai film d'azione più popolari, con classifiche dettagliate e descrizioni sintetiche per facilitare la scelta. Un'altra sezione evidenzia i film di tendenza, permettendo agli utenti di scoprire rapidamente i titoli più discussi del momento. Se un film non è presente nel catalogo, l'utente può suggerirne l'aggiunta attraverso l'apposito pulsante. La barra di navigazione, posizionata nella parte inferiore dello schermo, garantisce un accesso rapido alle principali sezioni dell'applicazione, tra cui la home, la gestione delle stanze e il profilo utente. L'interfaccia, coerente con il design dell'intera applicazione, è stata studiata per garantire una navigazione fluida e un'interazione immediata, riducendo al minimo il rischio di errori e migliorando l'esperienza complessiva dell'utente.

Cliccando sul pulsante centrale della barra di navigazione, Gestione Stanze, si arriva nella quarta Storyboard, dove si può entrare in una stanza avendo il codice di accesso oppure la si può creare. Nel processo di creazione si possono selezionare vari criteri iniziali se si ha già un'idea di che tipo di film o serie tv si vuole vedere:

- Si può creare una stanza per Genere per avere proposte ad esempio solamente Horror;
- Si può creare una stanza per Anno di Uscita per avere proposte di contenuti usciti in un intorno di quell'anno (stessa decade);
- Si può creare una stanza per Fornitore per avere proposte di contenuti che sono disponibili solo su una piattaforma streaming;
- Si può creare una stanza per Compagnia Produttrice per avere proposte solo di contenuti della Fox o Disney;
- Infine il tipo di media (unico campo obbligatorio della Storyboard oltre al nome della stanza) indica quale tipo di media si vuole vedere, se Film o Serie TV.

Whats2Watch

Logout



Recent Rooms



Create Room per Genre

Horror Room +

Fantasy Room +

Mystery Room +

Action Room +

Children Room +



Leaderboard Action Movies

#1 - Mad Max: Fury Road

Set in a post-apocalyptic world, this visually stunning film follows Max and Fur...



#2 - The Dark Knight

A crime thriller where Batman faces his greatest enemy, the Joker, in a dark...



#3 - John Wick

An action-packed revenge story of a former assassin drawn back into the...



#4 - Gladiator

Set in ancient Rome, this film follows the journey of a betrayed general fightl...



#5 - Die Hard

The iconic story of NYPD officer John McClane as he takes on terrorists...



Leaderboard Action Movies

#1 - Mad Max: Fury Road

Set in a post-apocalyptic world, this visually stunning film follows Max and Fur...



#2 - The Dark Knight

A crime thriller where Batman faces his greatest enemy, the Joker, in a dark...



#3 - John Wick

An action-packed revenge story of a former assassin drawn back into the...



#4 - Gladiator

Set in ancient Rome, this film follows the journey of a betrayed general fightl...



#5 - Die Hard

The iconic story of NYPD officer John McClane as he takes on terrorists...



Leaderboard Action Movies

#1 - Mad Max: Fury Road

Set in a post-apocalyptic world, this visually stunning film follows Max and Fur...



#2 - The Dark Knight

A crime thriller where Batman faces his greatest enemy, the Joker, in a dark...



#3 - John Wick

An action-packed revenge story of a former assassin drawn back into the...



#4 - Gladiator

Set in ancient Rome, this film follows the journey of a betrayed general fightl...



#5 - Die Hard

The iconic story of NYPD officer John McClane as he takes on terrorists...



Trending Movies



Venom: The Last Dance



Terrifier 3



The Wild Robot



Deadpool & Wolverine



Gladiator II



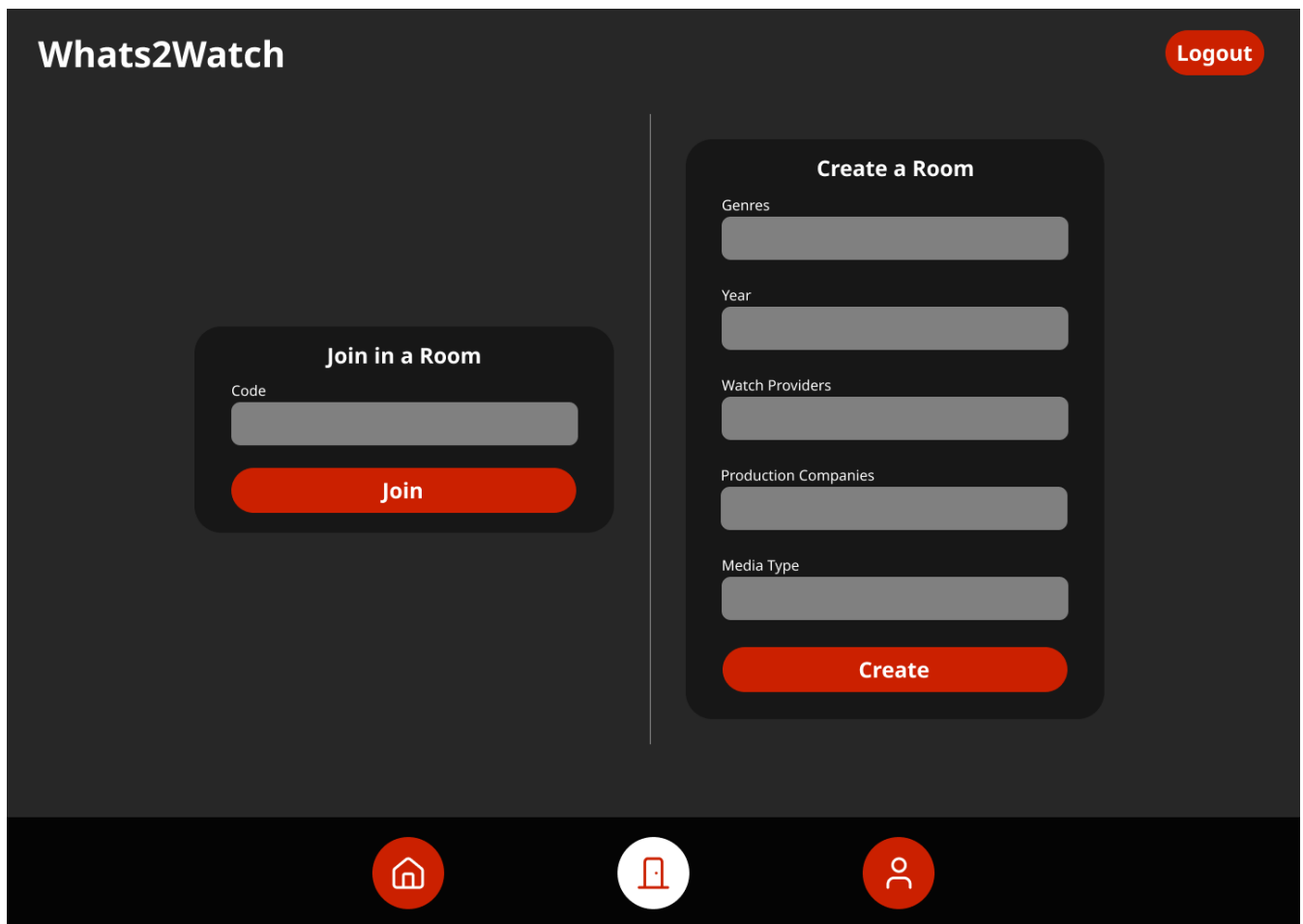
Despicable Me 4



Do you think a film is missing? Request it now by clicking the button right here!

Request Movie


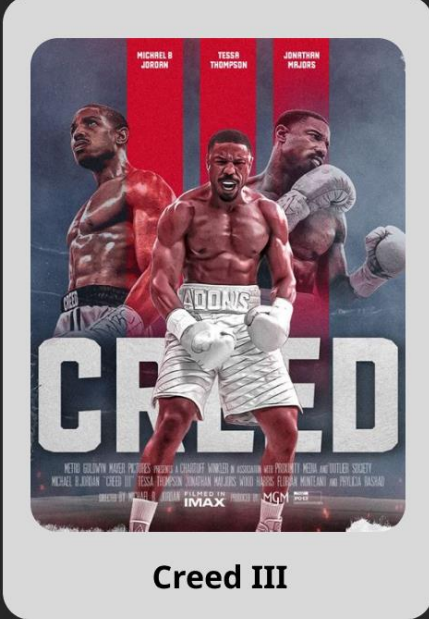






Una volta creata la stanza oppure essere entrato tramite il codice che viene inviato al creatore della stanza vi è la Storyboard dello Swipe, dove finalmente l'utente può reagire ai media proposti grazie all'algoritmo di raccomandazione che è fulcro dell'applicativo. Inoltre, cliccando sulla copertina del film avremo tutte le informazioni necessario per conoscerlo al meglio, come trama, cast, anno di uscita, dove lo si può vedere e trailer. Se si clicca su 'Matches' si potranno vedere i film a cui si è reagito positivamente e quelli a cui tutta la stanza ha reagito positivamente. Infine, uscendo dalla stanza, se si clicca sul pulsante di destra della barra di navigazione si va sul proprio profilo, dove si possono selezionare le proprie preferenze (associate all'utente e non alla singola stanza).

Whats2Watch

Logout




Creed III

SwipeMatches


Whats2Watch

Logout


Room Matches




Venom: The Last Dance




Terrifier 3




The Wild Robot



Deadpool & Wolverine




Gladiator II




Despicable Me 4


Your Likes




Venom: The Last Dance




Terrifier 3




The Wild Robot



Deadpool & Wolverine

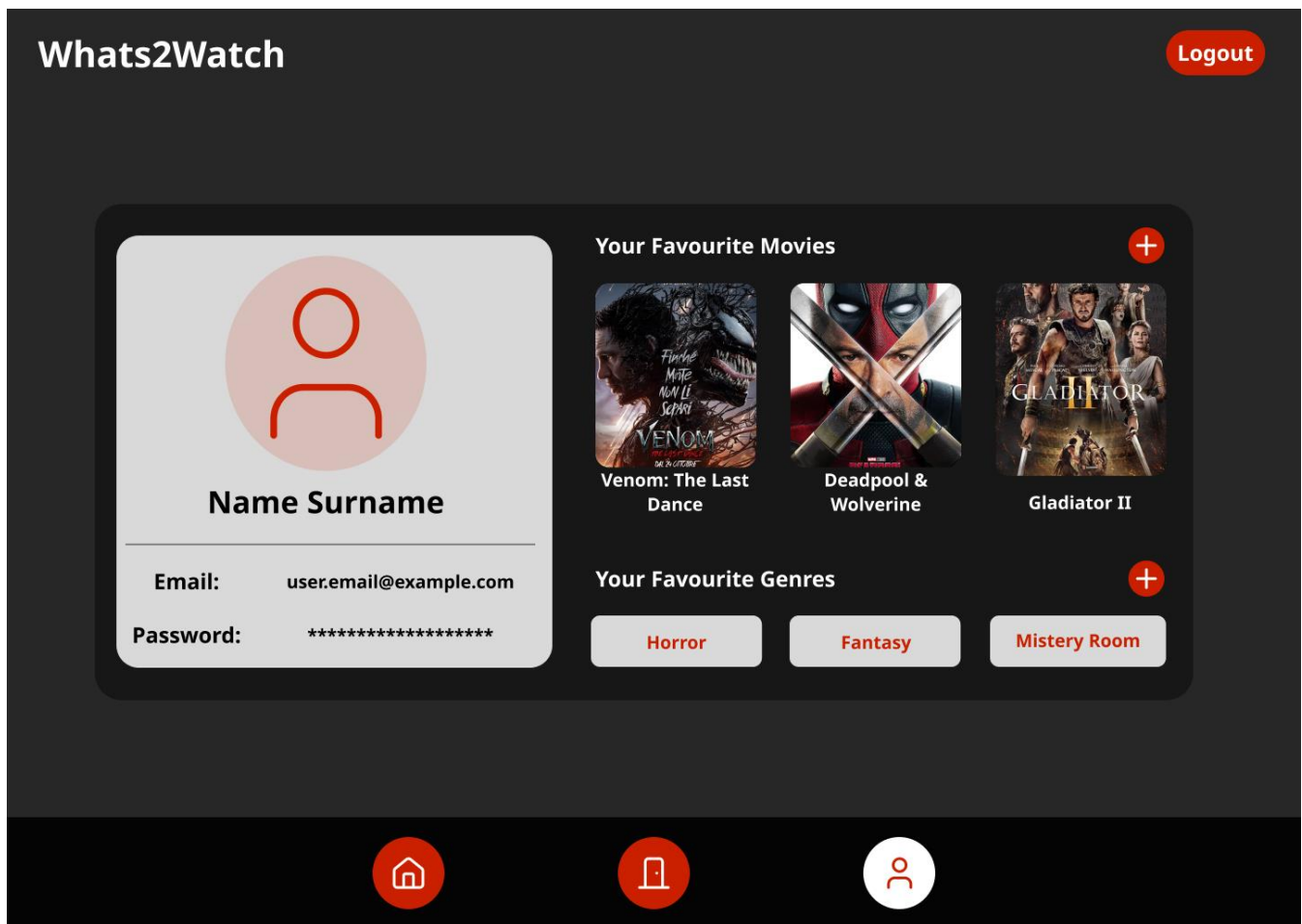


Gladiator II



Despicable Me 4

SwipeMatches



Design

Class Diagram

VOPC Analysis

Dato lo *Use Case Diagram* precedente si nota come un *Class Diagram* a livello del problema deve avere tre controller poiché ci sono tre casi d'uso principali e tre boundaries poiché ci sono tre interconnessioni tra attori coinvolti e casi d'uso. In realtà vi è anche una quarta boundaries (invisibile all'utente) che si interfaccia con le API esterne. Il class diagram a livello di problema è presente al seguente [link](#) (se tale link non dovesse funzionare, il suddetto è stato caricato anche in formato PDF seguendo il percorso './CDFallback/W2W-PL.pdf'). Tale diagramma rappresenta l'architettura del sistema W2W secondo il pattern BCE (Boundary-Control-Entity), che suddivide le responsabilità tra classi di interfaccia utente (Boundary), logica applicativa (Control) e dati persistenti (Entity). Le classi Entity, evidenziate in giallo, modellano i dati principali del sistema, mentre le classi Control, in rosso, gestiscono la logica dell'applicazione. Le classi Boundary, rappresentate in blu, fungono da interfaccia tra l'utente e il sistema.

Nel cuore del diagramma troviamo la classe Media, che funge da superclasse per Movie e TVSeries, contenendo attributi comuni come titolo, anno di uscita, valutazione e cast. Movie e TVSeries specializzano Media, aggiungendo rispettivamente informazioni specifiche sui film e sulle serie TV, con TVSeries che include dettagli sugli episodi e i personaggi principali. Actor e Character rappresentano gli attori e i ruoli interpretati nei vari media, mentre Genre, ProductionCompanies e WatchProvider

arricchiscono le informazioni note sui Media. Room rappresenta le stanze di discussione, includendo attributi come membri, genere e storico delle scelte. Per quanto riguarda la logica applicativa, le classi di controllo orchestrano le operazioni sulle entità. SwipeController implementa il sistema di suggerimenti basandosi sulle preferenze degli utenti e sui dati delle stanze. RoomController gestisce la creazione e il funzionamento delle stanze, mentre il RegisterController orchestra le modalità di accesso all'applicativo. Le classi boundary fungono da ponte tra il sistema e l'utente, ricevendo input ed eseguendo chiamate ai controller. RegisterBoundary, RoomBoundary e MatchesBoundary si occupano dell'interazione con l'utente per la visualizzazione e gestione dei rispettivi dati. HomePageBoundary rappresenta l'interfaccia principale dell'applicazione, fornendo accesso ai contenuti principali. In conclusione, il diagramma segue il pattern BCE, separando nettamente i dati, la logica e l'interfaccia. La struttura risulta ben organizzata, con una forte modularità che facilita l'espandibilità e la manutenzione del sistema.

Design Level Diagram

Tale diagramma è disponibile al seguente [link](#) (se tale link non dovesse funzionare, il suddetto è stato caricato anche in formato PDF seguendo il percorso `./CDFallback/W2W-DL.pdf`). Il nuovo diagramma delle classi introduce ulteriori livelli di astrazione e separazione delle responsabilità rispetto alla versione precedente, riflettendo un design più strutturato e manutenibile, cercando di seguire i principi SOLID.

Le classi Bean fungono da intermediari tra le classi Boundary e Controller, migliorando l'incapsulamento dei dati. Queste permettono di trasferire dati tra i livelli dell'applicazione, riducendo il legame diretto tra interfaccia utente e logica di business, e forniscono un'interfaccia standard per manipolare entità come UserBean e RoomBean. L'introduzione delle DAO (Data Access Object) separa la logica di accesso ai dati dalla logica applicativa. Ogni entità principale ha il proprio DAO (MovieDAO, TVSeriesDAO, RoomDAO, UserDAO), il che garantisce un'architettura più scalabile e aderente ai principi SOLID, in particolare il Single Responsibility Principle. I DAO interagiscono direttamente con l'architettura di persistenza sottostante e vengono chiamati dai Controller per eseguire operazioni CRUD. Il pattern BCE (Boundary-Control-Entity) è ora più raffinato con l'inclusione di diversi componenti: i Boundary gestiscono l'interazione con l'utente, i Bean trasferiscono dati tra boundary e controller, i Controller contengono la logica applicativa, i DAO accedono al database, e le Entity rappresentano i dati persistenti.

Design Patterns

Nel progetto Whats2Watch, sono stati implementati diversi pattern progettuali al fine di garantire una gestione scalabile e ben strutturata delle entità e della loro persistenza. In particolare, sono stati utilizzati i pattern Builder, Factory e Abstract Factory per rispondere a specifiche esigenze di flessibilità, estensibilità e separazione dei concetti. Questi pattern sono stati scelti non solo per la loro capacità di risolvere problemi progettuali concreti, ma anche per i benefici che portano in termini di GRASP (General Responsibility Assignment Software Patterns), in particolare per quanto riguarda la cohesion, la low coupling, e la flexibility del sistema.

Il pattern Builder è stato utilizzato per costruire oggetti complessi come Movie e TVSeries, che sono specializzazioni della classe astratta Media. Questi oggetti richiedono l'aggregazione di diverse

proprietà, come titolo, cast, regista, durata, episodi, valutazioni, etc... L'uso di un builder permette di costruire questi oggetti passo per passo, mantenendo il codice chiaro e facilmente leggibile, evitando costruttori troppo complessi con un numero elevato di parametri ([Telescoping Constructor](#)).

In combinazione con il pattern Factory, il processo di istanziamento degli oggetti Media (e delle sue specializzazioni, Movie e TVSeries) è stato centralizzato e uniformato. La Factory ha il compito di creare le istanze delle classi Media e delle sue specializzazioni in modo coerente, facilitando l'estendibilità del sistema. Quando si aggiungeranno nuove tipologie di media, sarà sufficiente estendere la factory senza modificare altre parti del sistema, riducendo il rischio di coppie strette e aumentando la manutenibilità. L'utilizzo combinato di questi pattern comporta:

- **Cohesion:** Ogni classe è focalizzata su un compito specifico, il builder è responsabile solo della costruzione di oggetti complessi, mentre la factory si occupa della loro creazione.
- **Low Coupling:** I client non devono preoccuparsi di come vengono creati gli oggetti Media; basta chiedere alla factory di fornirli, riducendo il dipendimento diretto dalle classi concrete.
- **Flexibility:** L'aggiunta di nuove tipologie di Media (ad esempio, Documentary) è facilitata dalla struttura modulare e dal pattern Factory. Allo stesso modo, il pattern Builder consente di modificare facilmente il processo di costruzione degli oggetti senza alterare altre parti del codice.

Nel contesto della persistenza dei dati, è stato utilizzato il pattern Abstract Factory per gestire la creazione dei DAO (Data Access Object). L'Abstract Factory permette di creare una serie di oggetti DAO che si specializzano nella persistenza di diverse entità, come Movie e TVSeries, ma che devono anche poter operare in diverse modalità di persistenza: Database, File System, e In RAM per la versione Demo. Il pattern Abstract Factory consente di separare la logica di creazione dei DAO dalla logica di utilizzo. Ogni tipo di DAO può essere creato dalla Factory corrispondente, a seconda della modalità di persistenza desiderata. La separazione tra le tipologie di DAO e le modalità di persistenza rende il sistema altamente flessibile, permettendo di aggiungere nuove modalità di persistenza senza modificare il codice dei client che utilizzano i DAO. Ciò comporta:

- **Cohesion:** La classe Abstract Factory è focalizzata sulla creazione dei DAO e non si preoccupa della loro implementazione specifica, delegando le istanze concrete alle classi specializzate.
- **Low Coupling:** I client non devono preoccuparsi di quale modalità di persistenza viene utilizzata; la logica di creazione è centralizzata nella factory.
- **Flexibility:** L'introduzione di una nuova modalità di persistenza (ad esempio, l'uso di un nuovo tipo di database) può avvenire facilmente con l'aggiunta di una nuova factory concreta, senza influire sul resto del sistema.

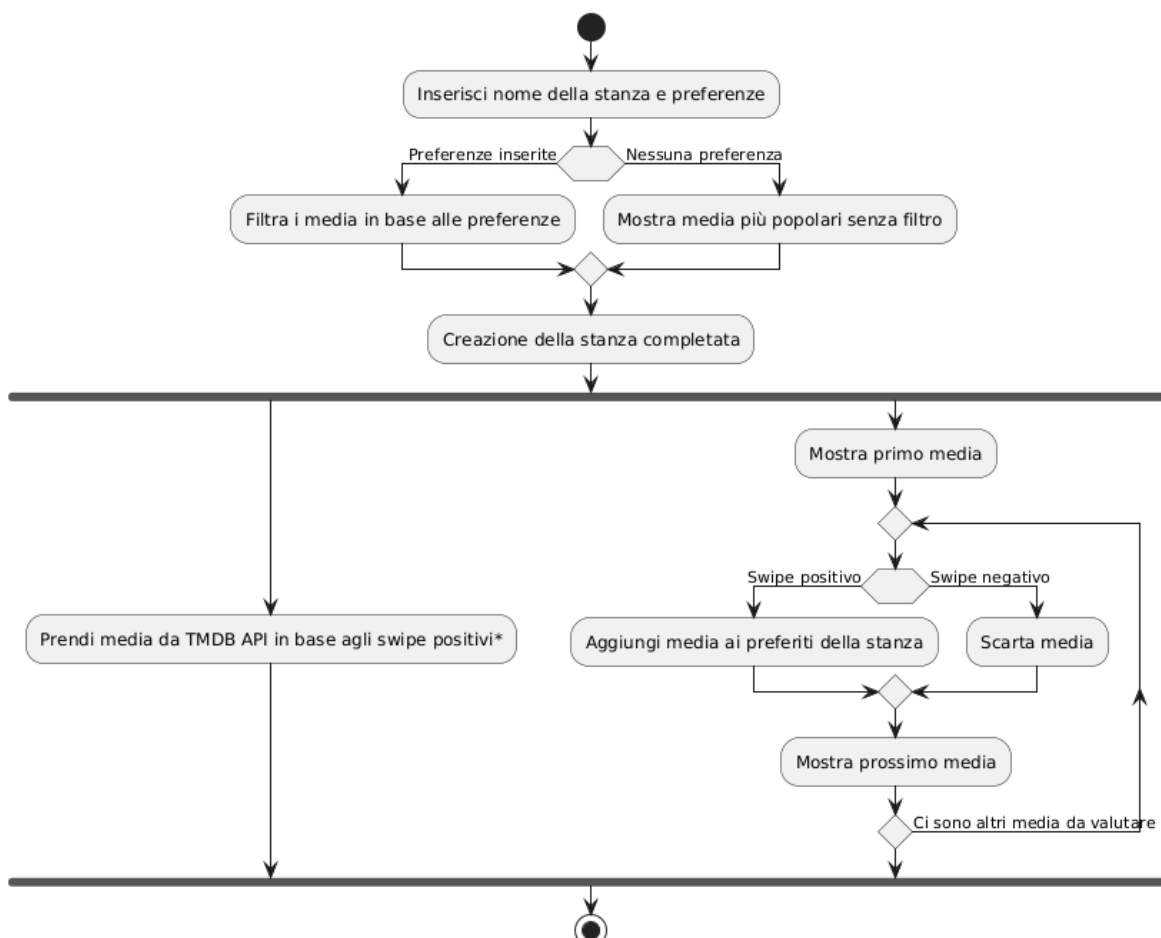
Oltre ai benefici pratici derivanti dall'uso dei pattern progettuali, l'adozione dei principi *GRASP* ha contribuito significativamente alla robustezza dell'architettura. I principali principi che sono stati seguiti includono:

- **Information Expert:** La responsabilità di creazione degli oggetti Media è affidata ai builder e alle factory, che sono esperti nell'assemblaggio e nell'istanziamento delle rispettive classi. Allo stesso modo, la responsabilità della persistenza dei dati è affidata ai DAO, che sono esperti nella gestione delle varie modalità di salvataggio.

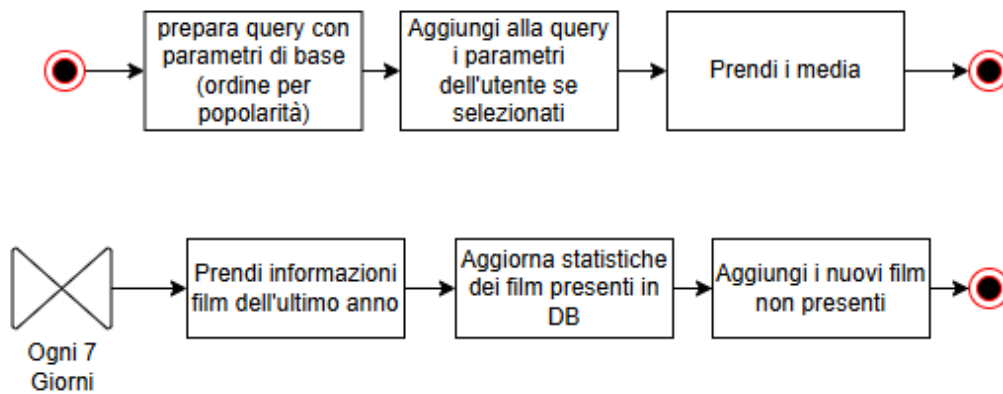
- **Creator:** Le factory si occupano della creazione degli oggetti, rispettando il principio di separazione delle preoccupazioni. Inoltre, grazie al pattern Abstract Factory, le modalità di persistenza sono gestite da una gerarchia di factory che facilita la manutenzione e l'estensione del codice.
- **Controller:** La gestione della logica di persistenza e creazione degli oggetti è delegata ai DAO e alle Factory, riducendo il carico di responsabilità sulle classi di livello superiore, come quelle che gestiscono la logica di business dell'applicazione.
- **Low Coupling:** Come accennato, l'utilizzo di pattern come Factory e Abstract Factory riduce la dipendenza diretta tra i vari componenti del sistema, migliorando la modularità e rendendo il codice più riutilizzabile e testabile.
- **High Cohesion:** Ogni classe ha una responsabilità ben definita, contribuendo a mantenere il codice ordinato e focalizzato su un singolo compito, facilitando così la manutenibilità.

Activity Diagram

I diversi diagrammi UML seguenti tratteranno lo stesso caso d'uso ('*Swipe Media*' nonché quello sviluppato nell'applicativo) in modo da poter vedere il suddetto da diversi punti di vista e capire quindi al meglio le potenzialità ed i dettagli che i diversi diagrammi hanno da offrire. L'activity diagram per esempio è un diagramma di flusso che mostra come un'attività porta ad un'altra attraverso una sequenza di operazione svolte sequenzialmente o in parallelo.



Prendi Film da IMDB API

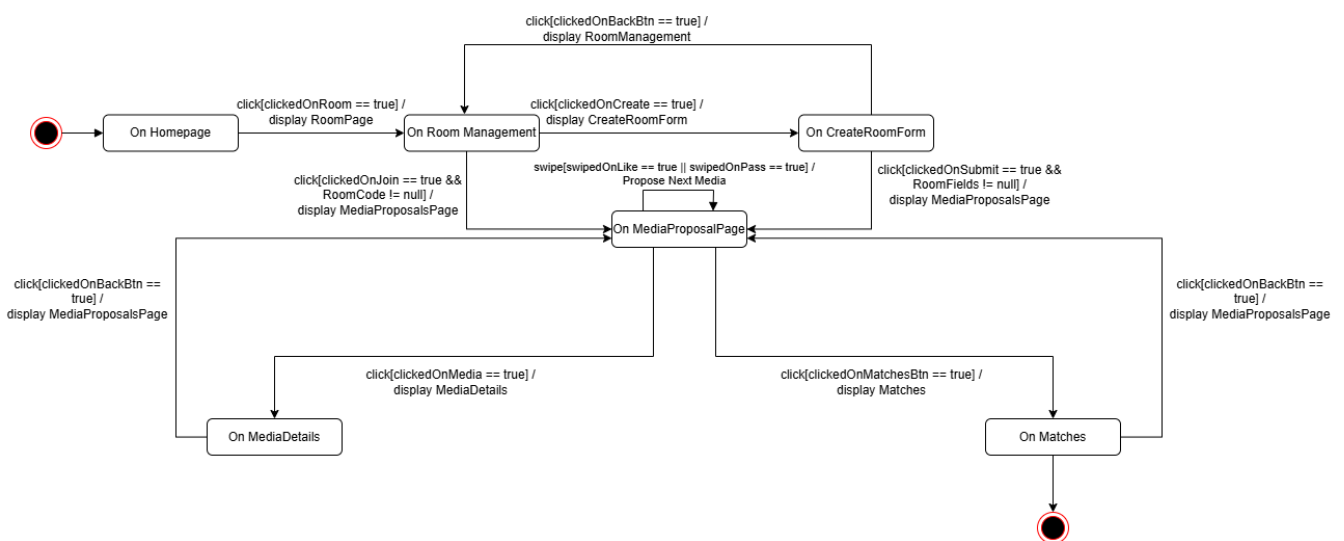


Sequence Diagram

Nel Sequence Diagram invece si cercano di modellare le interazioni degli elementi in un singolo caso d'uso. Dimostrano le interazioni che si verificano quando viene eseguito uno specifico caso d'uso e l'ordine in cui i vari componenti del sistema interagiscono per eseguire una funzione. Un diagramma di sequenza, quindi, dimostra come i vari componenti di un sistema interagiscono tra loro per completare un compito. Il diagramma si trova a pagina seguente.

State Diagram

Infine in un diagramma di stato UML si visualizzano gli stati di un procedimento finito, dunque di un modello di comportamento costituito da azioni e stati o transizioni di stato. Inoltre il diagramma prevede per ogni oggetto del modello uno stato iniziale, uno stato finale e almeno uno stato intermedio. Il diagramma di stato consente di raffigurare l'intero ciclo di vita di un caso d'uso.



State Diagram



Sequence Diagram

Testing

In questo capitolo si analizza in dettaglio la parte di testing dell'applicativo Whats2Watch. L'obiettivo dei test è verificare il corretto funzionamento delle logiche applicative e garantire le performance attese, adottando una strategia di test unitari che copre sia casi di normalità che scenari limite.

1. Test del RoomController

La classe di test *'RoomMatcherTest'* si occupa di verificare l'operazione *getRoomMatches* del controller, responsabile della restituzione dei contenuti multimediali comuni (match) tra i membri di una stanza. In particolare, vengono testati diversi scenari:

a. Caso di stanza senza membri

```
@Test
void getRoomMatchesEmptyRoomMembersCase() {
    Room room = RoomFactory.createRoomInstance().build();
    List<Media> result = RoomController.getRoomMatches(room);
    assertTrue(result.isEmpty());
}
```

- **Obiettivo:** Verificare che, in assenza di membri, il metodo restituisca una lista vuota.
- **Approccio:** Viene creata una stanza senza membri e si effettua l'asserzione sul fatto che il risultato sia vuoto.

b. Caso di un singolo membro

```
@Test
void getRoomMatchesSingleMemberCase() {
    RoomMember member = new RoomMember(new User("", "", null, "", ""));
    member.getLikedMedia().addAll(Set.of(mediaA, mediaB, mediaC));

    Room room =
RoomFactory.createRoomInstance().roomMembers(Set.of(member)).build();

    List<Media> result = RoomController.getRoomMatches(room);

    assertEquals(List.of(mediaC, mediaA, mediaB), result);
}
```

- **Obiettivo:** Quando la stanza contiene un solo membro, tutti i media da lui preferiti devono essere restituiti come risultato.
- **Approccio:** Dopo aver popolato il set dei media preferiti con tre istanze, il test verifica che il risultato contenga esattamente gli stessi elementi, eventualmente ordinati in base ad un criterio di popolarità o logica interna (in questo esempio, si attende l'ordine [mediaC, mediaA, mediaB]).

c. Caso di due membri con media comuni**@Test**

```
void getRoomMatchesTwoMembersWithCommonMediaCase() {
    RoomMember m1 = new RoomMember(new User("", "", null, "u1", ""));
    m1.getLikedMedia().addAll(Set.of(mediaA, mediaB));
    RoomMember m2 = new RoomMember(new User("", "", null, "u2", ""));
    m2.getLikedMedia().addAll(Set.of(mediaC, mediaB));
    Room room = RoomFactory.createRoomInstance().roomMembers(Set.of(m1, m2)).build();
    List<Media> result = RoomController.getRoomMatches(room);
    assertEquals(List.of(mediaB), result);
}
```

- **Obiettivo:** Testare la logica che estrae i media preferiti comuni a tutti i membri. Nel caso specifico, solo mediaB è presente in entrambi i set.
- **Approccio:** Vengono creati due membri con insiemi di media differenti, in cui solo uno è condiviso. L'asserzione verifica che il risultato contenga esclusivamente il media comune.

d. Caso di membri senza media comuni**@Test**

```
void getRoomMatchesMembersWithNoCommonMediaCase() {
    RoomMember m1 = new RoomMember(new User("", "", null, "u1", ""));
    m1.getLikedMedia().add(mediaA);
    RoomMember m2 = new RoomMember(new User("", "", null, "u2", ""));
    m2.getLikedMedia().addAll(Set.of(mediaC, mediaB));
    Room room = RoomFactory.createRoomInstance().roomMembers(Set.of(m1, m2)).build();
    List<Media> result = RoomController.getRoomMatches(room);
    assertTrue(result.isEmpty());
}
```

- **Obiettivo:** Garantire che, se non vi è alcun elemento in comune tra i membri, il metodo restituisca una lista vuota.
- **Approccio:** Vengono utilizzati insiemi di media completamente disgiunti, con la conseguente verifica dell'assenza di match.

e. Gestione dei casi in cui un membro ha un set di media vuoto**@Test**

```
void getRoomMatches_OneMemberWithEmptyLikedMedia_ReturnsEmptyList() {
    RoomMember m1 = new RoomMember(new User("", "", null, "", ""));
    m1.getLikedMedia().addAll(Collections.emptySet());
    Room room = RoomFactory.createRoomInstance().roomMembers(Set.of(m1)).build();
    List<Media> result = RoomController.getRoomMatches(room);
    assertTrue(result.isEmpty());
}
```

E

@Test

```
void getRoomMatches_MemberWithEmptyAndNonEmptyLikedMedia_ReturnsEmptyList() {
    RoomMember m1 = new RoomMember(new User("", "", null, "u1", ""));
    m1.getLikedMedia().addAll(Collections.emptySet());
    RoomMember m2 = new RoomMember(new User("", "", null, "u2", ""));
    m2.getLikedMedia().addAll(Set.of(mediaC, mediaB));
    Room room = RoomFactory.createRoomInstance().roomMembers(Set.of(m1, m2)).build();
    List<Media> result = RoomController.getRoomMatches(room);
    assertTrue(result.isEmpty());
}
```

- **Obiettivo:** Verificare che la presenza di un membro con lista di media vuota porti a non avere alcun match, anche se altri membri hanno media preferiti.
- **Approccio:** Questi test evidenziano la robustezza del metodo, assicurando che l'algoritmo di matching richieda la presenza del media in **tutti** i membri della stanza per essere considerato valido.

2. Test del MediaDAO

La classe MediaDAOTest si concentra sulla verifica delle performance della query per il recupero dei dati multimediali. In particolare, viene testato il metodo findAll del DAO che utilizza la persistenza tramite file system.

a. Misurazione della durata della query

@Test

```
void testFindAllMediaQueryDuration() throws DAOException {
    // Measure the time taken by the findAll method
    long startTime = System.currentTimeMillis();
    PersistenceFactory.createDAO(PersistenceType.FILESYSTEM).createMovieDAO()
        .findAll().stream()
        .map(movie -> (Media) movie);
    long endTime = System.currentTimeMillis();
    long duration = endTime - startTime;
    // Assert that the query duration is less than 3 seconds
    assertTrue(duration < 3000, "The query took longer than 3 seconds");
}
```

- **Obiettivo:** Garantire che l'operazione di recupero di tutti i media dal file system avvenga in un tempo accettabile (inferiore a 3 secondi).
- **Approccio:**
 - **Misurazione del tempo:** Utilizzo delle chiamate System.currentTimeMillis() per determinare la durata dell'operazione.

- **Verifica delle performance:** L'asserzione finale garantisce che il tempo impiegato non superi il limite prestabilito, contribuendo così a monitorare e mantenere le performance dell'applicazione.

3. Considerazioni Tecniche e Best Practice

- **Test Unitari e Coverage:** I test coprono sia casi di uso 'comuni' (scenari in cui i dati sono presenti e validi) sia casi limite (assenza di membri o liste vuote), garantendo una copertura completa della logica applicativa. Questo è fondamentale per prevenire bug e regressioni, soprattutto in sistemi che si basano su regole di aggregazione e filtraggio dei dati.
- **Misurazioni delle Performance:** L'integrazione di test di performance, come nel caso del `MediaDAOTest`, dimostra l'attenzione verso non solo la correttezza funzionale ma anche verso l'efficienza dell'implementazione. Questo tipo di test è particolarmente utile per identificare eventuali colli di bottiglia e per mantenere standard di qualità nelle operazioni di I/O e accesso ai dati.
- **Utilizzo di JUnit:** La suite di test sfrutta JUnit 5, evidenziando l'adozione di framework moderni e consolidati per il testing in Java. L'uso di asserzioni (`assertTrue`, `assertEquals`) consente di definire chiaramente le aspettative per ogni scenario testato.

Considerazioni Finali

Lo sviluppo applicativo di *Whats2Watch* si è concentrato principalmente sul caso d'uso '*Swipe Media*' e sul motore di raccomandazione associato, che rappresentano il nucleo funzionale dell'applicativo. La gestione delle preferenze degli utenti, l'elaborazione delle interazioni e la selezione dei contenuti suggeriti hanno richiesto un'architettura progettuale scalabile e modulare, in grado di garantire performance ottimali anche con un numero elevato di utenti concorrenti.

Dal punto di vista architetturale, l'adozione di un pattern *Abstract Factory* per la gestione dei DAO ha permesso di rendere il sistema più flessibile, delegando la scelta del livello di persistenza alla configurazione iniziale dell'applicativo. Inoltre, il *Builder Pattern* è stato utilizzato per costruire dinamicamente gli oggetti *Media* e *Room*, assicurando una maggiore modularità nella creazione delle istanze senza dover gestire costruttori con numerosi parametri.

Il sistema *Swipe Media*, oltre a richiedere un'infrastruttura di persistenza efficiente, ha introdotto sfide legate alla latenza delle operazioni di raccomandazione. Queste ultime sono state progettate seguendo un approccio ibrido, combinando un sistema '*content-based filtering*' semplificato con un modello di aggregazione delle preferenze in tempo reale. Questo ha comportato la necessità di ottimizzare l'accesso ai dati, bilanciando il numero di operazioni di lettura e scrittura eseguite sui diversi livelli di persistenza. Per mitigare ciò, si è adottata una strategia di *caching* per i contenuti più rilevanti e un aggiornamento asincrono delle preferenze utente, riducendo così il carico computazionale sulle operazioni di elaborazione in tempo reale.

Infine, invece di far eseguire l'applicativo con un livello di persistenza multiplo (anche solo per alcune entità), si è preferito adottare l'utilizzo di una singola tipologia di persistenza alla volta (Database, File System o Memoria Centrale) a favore di un comportamento coerente e prevedibile. Questa decisione mette in evidenza uno degli elementi fondamentali del corso: cambiando solamente un flag e utilizzando le stesse firme delle operazioni che coinvolgono la persistenza di dati, è possibile passare

da un livello di persistenza all'altro. In questo modo, si evidenzia in maniera ancora più marcata il concetto di modularità e flessibilità del sistema.

In sintesi, il design tecnico di *Whats2Watch* è stato guidato da scelte architettureali che privilegiano modularità, performance e manutenibilità. Il riutilizzo della stessa logica applicativa per le diverse UI e per i diversi livelli di persistenza, unita a una gestione ottimizzata dell'accesso ai dati, garantisce un funzionamento stabile ed efficiente del motore di raccomandazione, ponendo le basi per eventuali future estensioni del sistema.