|  |
| --- |
| A logo with numbers and letters  Description automatically generated  **Università degli Studi di Salerno Corso di Ingegneria del Software** |

**Rated  
System Design Document  
Versione 1.1**

**A black and white logo

Description automatically generated**

Data: 8/1/2025

**Coordinatore del progetto:**

|  |  |
| --- | --- |
| Nome | Matricola |
|  |  |
|  |  |

**Partecipanti:**

|  |  |
| --- | --- |
| Nome | Matricola |
| Francesco Rao | 0512116836 |
| Bruno Nesticò | 0512117268 |
|  |  |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
| **Scritto da:** | Francesco Rao, Bruno Nesticò |

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Data | Versione | Descrizione | Autore |
| 24/11/2024 | 1.0 | Prima stesura completa | Francesco Rao, Bruno Nesticò |
| 08/1/2025 | 1.1 | Modifiche alle sezioni descrittive del testo e ai diagramma | Francesco Rao, Bruno Nesticò |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Indice**

1. Introduzione 4

1.1 Scopo del Sistema 4

1.2 Obiettivi di progettazione 4

1.2.1 Criteri di usabilità 4

1.2.2 Criteri di affidabilità 4

1.3 Riferimenti 4

2. Architettura Software Proposta 5

2.1 Decomposizione in Sottosistemi 5

2.2 Mapping Hardware/Software 6

2.3 Gestione dei dati persistenti 7

2.4 Controllo degli accessi e sicurezza 8

2.5 Controllo Software 9

2.6 Condizioni di confine 10

2.7 Servizi dei Sottosistemi 10

**1. Introduzione**

**1.1. Scopo del sistema**

Lo scopo del sistema è permettere la gestione di una piattaforma web dedicata agli appassionati di cinema. Si intende gestire le recensioni dei film, le valutazioni dei contenuti pubblicati dagli utenti e il sistema reputazionale che favorisce la visibilità dei recensori più attivi e apprezzati. Queste operazioni verranno eseguite rispettivamente dai gestori del catalogo, responsabili dell’aggiornamento dell’offerta di film, e dai moderatori, incaricati di garantire il rispetto delle linee guida e dei contenuti pubblicati. Inoltre, gli utenti avranno la possibilità di registrarsi, accedere, pubblicare recensioni, valutare i contributi della community e interagire con altri membri.

**1.2. Obiettivi di progettazione**

**1.2.1 Criteri di usabilità**

Per garantire un’esperienza utente ottimale, la piattaforma adotterà i seguenti criteri di usabilità:

*Validazione degli input*: saranno implementati meccanismi per prevenire errori durante l’inserimento di dati. Messaggi di errore chiari guideranno l’utente nella correzione dei valori errati.

*Design responsive*: l’interfaccia sarà ottimizzata per adattarsi a diversi dispositivi (PC, tablet, smartphone), rendendo l’esperienza uniforme e accessibile.

*Navigazione intuitiva*: ogni pagina presenterà una barra di navigazione per facilitare l’accesso rapido alle diverse sezioni.

**1.2.2 Criteri di affidabilità**

Il sistema garantirà robustezza e protezione attraverso:

*Controllo avanzato degli input****:*** oltre alla validazione primaria, saranno effettuati ulteriori controlli per gestire scenari non previsti e prevenire errori critici.

*Sicurezza dei dati****:*** saranno adottati protocolli di crittografia per proteggere le informazioni sensibili, come le credenziali degli utenti. Le password dovranno rispettare requisiti di complessità (es. lunghezza minima e inclusione di caratteri speciali).

**1.3 Riferimenti**

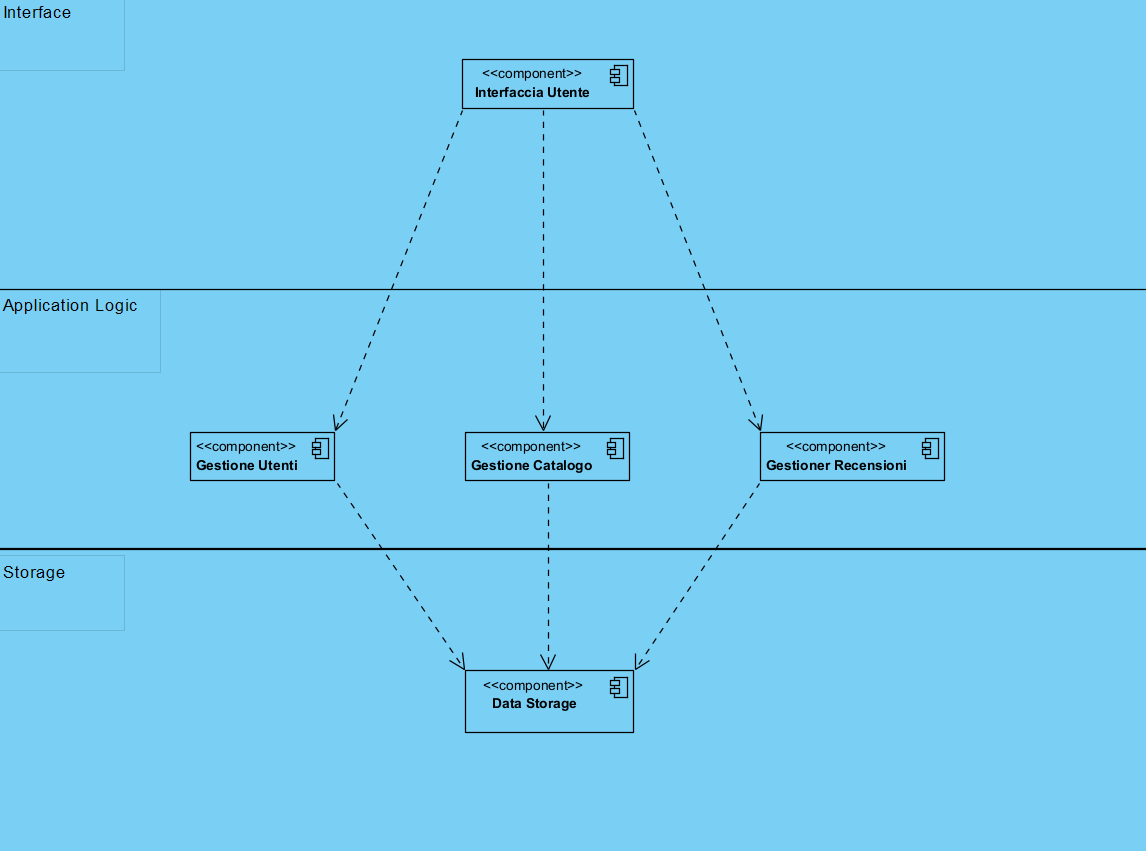
RequirementAnalysisDocument\_Rated

**2. Architettura Software Proposta**

**2.1 Decomposizine in Sottosistemi**

l sistema prodotto sarà caratterizzato da una architettura a tre livelli: Livello Interface, Livello Application Logic e Livello Storage.

*Component Diagram*

****

**Interface**

L’Interfaccia Utente è il sottosistema responsabile dell’interazione con l’utente finale. Fornisce le classi e le operazioni necessarie per acquisire input, visualizzare output e interagire con i servizi della piattaforma. Si occupa inoltre di:

*Coordinare la logica applicativa,* delegando le operazioni principali ai sottosistemi del livello sottostante.

*Validare i dati in ingresso*, assicurando la conformità alle specifiche del sistema e restituendo eventuali messaggi di errore in caso di input errati.

**Application Logic**

La Logica Applicativa è suddivisa in tre sottosistemi principali:

*Gestione Utenti:* si occupa dell’autenticazione, della creazione e dell’eliminazione degli account. Garantisce la protezione delle credenziali attraverso l’adozione di protocolli sicuri per la gestione dei dati sensibili.

*Gestione Catalogo:* consente ai Gestori di inserire, aggiornare e rimuovere film dal catalogo, mantenendo un’offerta costantemente aggiornata e coerente con le preferenze della community.

*Gestione Recensioni:* supporta l’aggiunta, la valutazione e la moderazione delle recensioni. Questo include meccanismi per segnalare contenuti inappropriati e l’assegnazione di punteggi che influenzano il sistema reputazionale.

**Storage**  
Il Data Storage rappresenta il sottosistema responsabile della memorizzazione e gestione degli oggetti persistenti, come:

*Dati utente:* informazioni personali, credenziali e preferenze.

*Dati di catalogo:* elenco dei film disponibili con relative informazioni.

*Dati delle recensioni:* contenuti testuali, valutazioni e metadati associati.

La persistenza dei dati è garantita attraverso l’utilizzo di un database relazionale ottimizzato per supportare operazioni frequenti e simultanee.

Il sistema adotta un’**architettura a strati chiusa**, in cui ogni livello può comunicare solo con lo strato immediatamente inferiore. Questa configurazione favorisce:

* *Alta manutenibilità:* le modifiche a uno strato non influenzano direttamente gli altri, rendendo più semplice la gestione del codice.
* *Flessibilità:* la separazione dei sottosistemi consente di aggiornare o sostituire componenti senza compromettere l’intero sistema.

**2.2 Mapping Hardware/Software**

Il sistema *Rated* è, per sua natura, un sistema distribuito, poiché gli utenti interagiscono con esso da macchine diverse, in remoto. Si distinguono tre tipi principali di componenti:

* Nodo Utente, che fornisce l’interfaccia utente. Questo nodo esegue il sottosistema Interface tramite un web browser.
* Application Server, che gestisce la logica applicativa e i controlli. Esegue il sottosistema Application Logic.
* Database Server, che gestisce la persistenza dei dati e i servizi offerti. Esegue il sottosistema Storage.

Il sistema sarà implementato utilizzando Apache Tomcat in combinazione con Java JDK 23.0. Tomcat è un contenitore di Servlet che consente il deployment e l’esecuzione di applicazioni web. La logica applicativa sarà scritta in Java per sfruttare la sua modularità e robustezza.

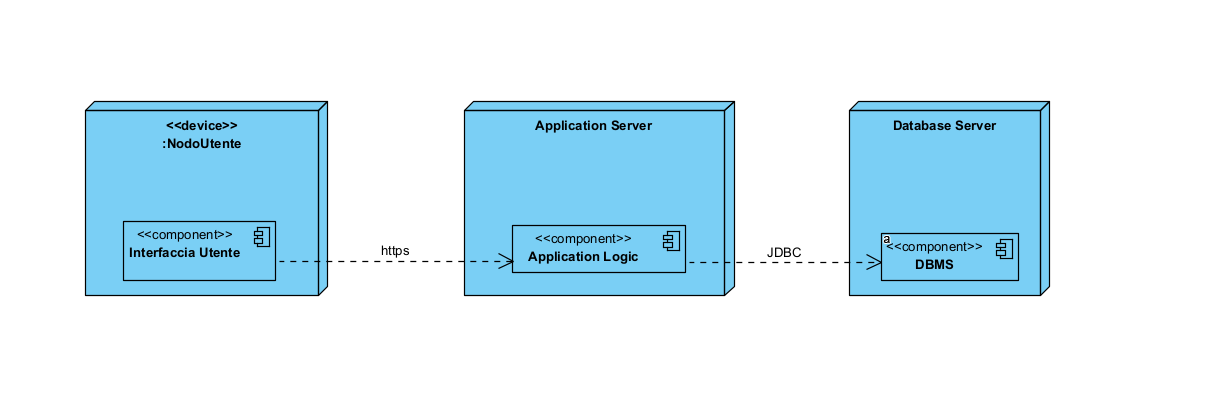
*Tecnologie utilizzate*:

* HTML5, CSS e Bootstrap 5: strumenti utilizzati per la creazione di interfacce utente moderne e responsive, che garantiscono una fruibilità ottimale su dispositivi diversi.
* JavaScript: utilizzato per fornire interattività e validazione dinamica dei dati inseriti dagli utenti.
* JDBC: utilizzato per gestire la comunicazione tra il backend e il database relazionale.
* AJAX: implementato per richieste asincrone, migliorando l’esperienza utente con aggiornamenti dinamici senza ricaricare la pagina.

In particolare, per implementare i sottosistemi eseguiti sul Nodo Server, verranno utilizzate le Servlet Java per gestire le richieste HTTP inviate dal browser e rispondere dinamicamente.

*Deployment Diagram*

Il componente Application Logic racchiude i sottosistemi relativi al layer logico. Il componente Storage, invece, si occupa della gestione dei dati persistenti attraverso MySQL, mentre il componente Interface è responsabile della presentazione e interazione utente.



**2.3 Gestione dei dati persistenti**

*Identificazione dei dati persistenti*

Il sistema gestisce il seguente insieme di oggetti da memorizzare in maniera persistente:

* Utente: informazioni personali, credenziali e preferenze.
* Film: dettagli sui film disponibili nel catalogo (titolo, regista, descrizione, ecc.).
* Recensione: contenuti testuali, votazioni e metadati relativi agli utenti e ai film.

Per memorizzare questi oggetti, verrà utilizzato un database relazionale, MySQL, per le seguenti ragioni:

* Supporto per query concorrenti.
* Meccanismi di transazione per garantire l’integrità e la consistenza dei dati (proprietà ACID).
* Capacità di eseguire backup e ripristino in modo efficiente.

Il sistema dovrà memorizzare i metadati associati ai contenuti degli utenti e ai film. Siccome memorizzare elementi di grandi dimensioni, come immagini di copertina o altri asset, direttamente nel database come BLOB può essere oneroso, questi file saranno salvati in directory dedicate nello spazio di indirizzamento del server. Il database memorizzerà solo i percorsi relativi alle risorse.

*Tecnologie utilizzate*

* MySQL 8.0: DBMS relazionale per la gestione dei dati persistenti.
* JDBC: utilizzato per interfacciarsi tra il backend e il database in modo sicuro ed efficiente.

La logica applicativa interagirà con il database mediante oggetti Java, utilizzando query SQL ottimizzate e verifiche di consistenza per garantire la robustezza del sistema.

**2.4 Controllo degli accessi e sicurezza**

La piattaforma Rated più essere utlizzata da diversi tipi di utenti, ognuno con permessi e privilegi differenti.

*Matrice di controllo degli accessi*

|  |  |  |  |
| --- | --- | --- | --- |
| Attore \ Oggetto | Utente | Film | Recensioni |
| Utente Guest | creaUtente() VisualizzaPaginaUtente() | getFilms() getFilms(nome) | getRecensioni() |
| Utente Registrato | logIn() logOut() VisualizzaPaginaUtente() | getFilms() getFilms(nome) recensisci() | getRecensioni() report() |
| Moderatore | logIn() logOut() VisualizzaPaginaUtente() | getFilms() getFilms(nome) recensisci() | getRecensioni() report() rimuoviRecensione() approvaRecensione() |
| Gestore del catalogo | logIn() logOut() VisualizzaPaginaUtente() | getFilms() getFilms(nome) recensisci() addFilm() removeFilm() | getRecensioni() report() |

**2.5. Controllo Software**

Osservando il *dynamic model* prodotto in fase di analisi dei requisiti, è stato scelto un tipo di controllo centralizzato. I diagrammi di sequenza individuati sono di tipo *fork*, in cui il controllo è affidato a un unico oggetto centrale.

In particolare, *Rated* adotta un flusso di controllo event-driven, basato sul paradigma MVC (*Model – View – Controller*), per garantire modularità e separazione delle responsabilità.

*Funzionamento generale*

Il sistema *Rated* segue un flusso di controllo centralizzato come descritto di seguito:

1. Ricezione della richiesta: Il Web Server attende richieste dal browser degli utenti.
2. Elaborazione della richiesta: Quando il server riceve una richiesta, questa viene elaborata e inoltrata al Controller, che la gestisce in base alla logica applicativa.
3. Thread dedicati: Per ogni richiesta ricevuta, il Web Server alloca un nuovo thread, consentendo la gestione concorrente di più richieste.
4. Risposta al client: Una volta elaborata la richiesta, il Controller restituisce il risultato al Web Server, che lo invia al browser dell’utente.

*Strategie per la gestione della concorrenza*

Per garantire un comportamento corretto in presenza di richieste concorrenti, *Rated* applica le seguenti strategie:

* Oggetti boundary:  
  Gli oggetti boundary, che rappresentano il punto di contatto tra l’interfaccia utente e la logica applicativa, non definiscono campi, ma contengono solo variabili locali temporanee legate alla richiesta corrente. Questo approccio evita problemi di concorrenza poiché tali oggetti non sono condivisi tra thread.
* Oggetti di controllo:  
  Ogni sessione utente gestisce un singolo oggetto di controllo, che non viene condiviso tra più thread. Questo impedisce conflitti tra richieste concorrenti che interessano la stessa sessione.
* Oggetti entità:  
  L’accesso e la modifica dello stato degli oggetti entità avvengono esclusivamente attraverso metodi dedicati, riducendo il rischio di inconsistenza nei dati.

Poiché il sistema è progettato per operare in un contesto enterprise, le transazioni e la gestione della concorrenza sono affidate automaticamente al container del web server (Apache Tomcat). Questo approccio garantisce che tutte le operazioni avvengano in modo sicuro e controllato.

**2.6. Condizioni di confine**

Le condizioni di confine si riferiscono all’inizializzazione, alla terminazione e alla gestione dei fallimenti del sistema.

Essendo una web application, *Rated* non richiede software aggiuntivo per la configurazione da parte dell’amministratore di sistema. L’inizializzazione e la terminazione del sistema sono gestite direttamente tramite l’interfaccia del web container (Apache Tomcat).

**2.7 Servizi dei Sottosistemi**

Considerando i sottosistermi del livello Application Logic: Gestione Utenti, Gestione Film e Gestione Recensioni, si elencano i servizi che offrono.

*Gestione Utenti*

* Servizio di **Autenticazione**, per la registrazione dell’utente e per la verifica delle credenziali di accesso a tempo di Login.

*Gestione Film*

* Servizio di **Aggiunta** Film al Catalogo.
* Servizio di **Rimozione** Film dal Catalogo.
* Servizio di **Modifica** delle informazioni di un Film presente nel Catalogo.

*Gestione Recensioni*

* Servizio di **Aggiunta** di una recensione ad un Film presente nel Catalogo.
* Servizio di **Votazione** di una recensione di un Film presente nel Catalogo.
* Servizio di **Segnalazione** di una recensione di un Film presente nel Catalogo.
* Servizio di **Rimozione** di una recensione segnalata.
* Servizio di **Approvazione** di una recensione segnalata.

