|  |
| --- |
| UNISA | Home  **Università degli Studi di Salerno Corso di Ingegneria del Software** |

**Mockbuster  
System Design Document**

**Versione 1.1**

****

Data: 17/01/2025

**Coordinatore del progetto:**

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

**Partecipanti:**

|  |  |
| --- | --- |
| Nome | Matricola |
| Roberto Ambrosino | 0512117886 |

|  |  |
| --- | --- |
| **Scritto da:** | Roberto Ambrosino |

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Data | Versione | Descrizione | Autore |
| 18/11/2024 | 0.1 | Prima stesura del documento | Roberto Ambrosino |
| 23/11/2024 | 1.0 | Scrittura introduzione, architettura software corrente, architettura software proposta, glossario servizi offerti dai sotto-sistemi | Roberto Ambrosino |
| 17/11/2025 | 1.1 | Aggiornamento class diagram e glossario servizi sotto-sistemi | Roberto Ambrosino |

|  |  |
| --- | --- |
| **1 - Introduzione** | 4 |
| **2 - Architettura software corrente** | 5 |
| **3 - Architettura software proposta** | 6 |
| **4 - Glossario dei servizi offerti dai sotto-sistemi** | 10 |

1 – Introduzione

**1.1 – Scopo del sistema**

Mockbuster è una piattaforma digitale che permette agli utenti di noleggiare e acquistare film online, fornendo una vasta gamma di titoli accessibili comodamente da casa. Il sistema integra la gestione del catalogo, l’elaborazione dei pagamenti, la gestione delle licenze e offre un'interfaccia intuitiva per migliorare l’esperienza utente.

**1.2 – Obiettivi di design**

* Garantire un’architettura modulare e scalabile.
* Fornire elevata usabilità tramite interfacce grafiche intuitive e responsive.
* Assicurare sicurezza e affidabilità, includendo crittografia delle password e comunicazioni HTTPS.
* Supportare fino a 10.000 utenti simultanei con uptime del 99.9%.

**1.3 – Definizioni, acronimi e abbreviazioni**

* HTTPS: HyperText Transfer Protocol Secure.
* JDBC: Java DataBase Connectivity

**1.4 – Riferimenti**

* Documento di problem statement: PS\_Mockbuster
* Documento di analisi dei requisiti: RAD\_Mockbuster

**1.5 – Panoramica**

Il documento descrive l'architettura corrente e quella proposta per Mockbuster, fornendo dettagli sui sottosistemi, il mapping hardware/software, e i meccanismi di sicurezza e persistenza.

2 – Architettura software corrente

Il sistema corrente utilizza:

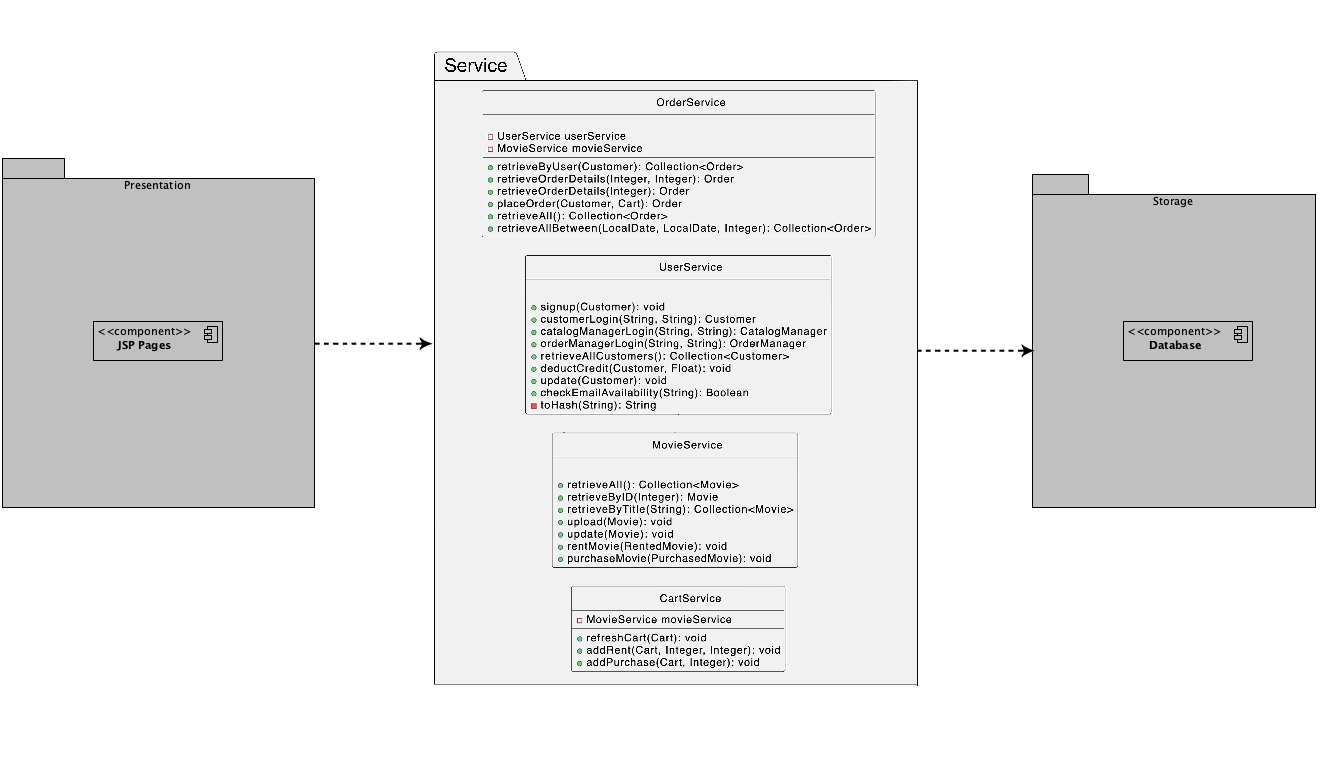
* **Componenti**: Basati su Java Servlet e JSP.
* **Gestione persistenza**: DBMS MySQL per memorizzare utenti, film in catalogo e ordini.
* **Mapping persistenza**: utilizzo di JPA (Java Persistence Api)
* **Funzionalità**:
  + Registrazione e autenticazione degli utenti.
  + Navigazione e ricerca nel catalogo.
  + Gestione del carrello per noleggi/acquisti.
  + Finalizzazione degli ordini
  + Gestione catalogo

3 – Architettura software proposta

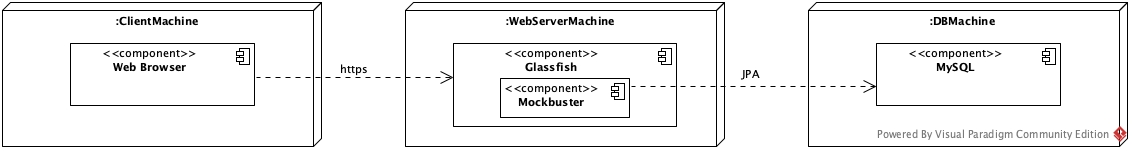
**3.1 – Panoramica**

L’architettura proposta adotta un modello Client-Server scalabile, con un’interfaccia utente web responsive e una gestione centralizzata dei dati tramite un database relazionale.

**3.2 – Decomposizione sotto-sistemi**

****

**3.3 – Mapping hardware/software**

****

**3.4 – Gestione persistenza dei dati**

Il sistema utilizza un database relazionale gestito tramite DBMS MySQL per archiviare i seguenti dati:

• La lista dei film presenti in catalogo

• Le informazioni degli utenti registrati

• La lista degli ordini eseguiti dai clienti con i relativi dettagli

Le entità principali del sistema sono mappate tramite JPA e gestite dal livello di persistenza. Il sistema inoltre implementa un sistema di caching per evitare frequenti interazioni con il database.

**3.5 – Controllo degli accessi e sicurezza**

Di seguito viene elencata la Global Access Table, che descrive, per ogni cella della tabella, quali operazioni può effettuare un attore su un oggetto:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Account** | **Film** | **Carrello** | **Ordine** |
| **Utente non registrato** | * Registrazione |  |  |  |
| **Utente registrato (cliente)** | * Login * Logout * Modifica profilo |  | * Inserire film in carrello * Modificare quantità presenti nel carrello * Svuotare il carrello |  |
| **Gestore catalogo** | * Login * Logout | * Caricare nuovo film * Rimuovere film dal catalogo * Modificare informazioni |  |  |
| **Gestore ordini** | * Login * Logout |  |  | * Ricercare ordine * Visualizzare tutti i dettagli di un ordine |

**3.6 – Controllo software globale**

Il sistema è implementato come una Web Application, seguendo l’architettura

client/server.

Componenti JSP forniscono al browser degli utenti le pagine della piattaforma con le quali interagire.

Il controllo del flusso globale viene gestito da filtri e servlet Java, i quali hanno il compito di gestire le richieste inoltrate dai client.

**3.7 – Boundary Conditions**

Il server sarà sempre attivo permettendo l’utilizzo della piattaforma in qualsiasi orario, e in presenza di un malfunzionamento, il sistema mostrerà pagine di manutenzione, non permettendo le normali operazioni previste.

**3.7.1 - Start-up**

Per il primo avvio della piattaforma, è necessario l’avvio di un web server che fornisca un servizio di JPA verso un database relazionale per la gestione dei dati persistenti.

**3.7.2 - Shutdown**

Nel caso di shutdown inteso come spegnimento del server, le sessioni verranno

terminate e tutti i cambiamenti apportati che non riguardano dati persistenti

verranno cancellati.

**3.7.3 – Comportamento in caso di errori**

In caso di errori imprevisti, il sistema mostrerà all’utente una pagina d’errore senza che vengano resi pubblici i log del web server.  
In caso l’utente inserisca dati in un formato sbagliato, il sistema mostrerà all’utente un messaggio di errore in corrispondenza del campo compilato in modo errato e fornirà indicazioni su come compilarlo in modo corretto.

Tutti i controlli sui campi sono effettuati sia lato client che lato server.

4 – Glossario dei servizi offerti dai sotto-sistemi

**Responsabilità sotto-sistemi:**

|  |  |
| --- | --- |
| **OrderService: Gestione degli ordini** | |
| **Responsabilità** | * Recuperare gli ordini di uno specifico cliente. * Recuperare i dettagli di un ordine specifico. * Inserire un nuovo ordine e gestire le transazioni associate (come film acquistati o noleggiati). * Filtrare gli ordini per intervallo temporale. |
| **Collaborazioni** | * Interagisce con UserService per gestire gli utenti che hanno effettuato gli ordini. * Interagisce con MovieService per aggiornare lo stato dei film ordinati. |

|  |  |
| --- | --- |
| **UserService: Gestione degli utenti** | |
| **Responsabilità** | * Registrazione di nuovi clienti (con hash della password). * Login per clienti, gestori del catalogo e gestori degli ordini. * Recupero di tutti i clienti registrati. * Aggiornamento dei dati di un cliente. * Verifica della disponibilità di un'email per la registrazione. |
| **Funzionalità interna** | * Funzione privata per generare hash sicuri delle password (toHash) |

|  |  |
| --- | --- |
| **MovieService: Gestione dei film** | |
| **Responsabilità** | * Recupero di tutti i film disponibili o di uno specifico film tramite ID o titolo. * Caricamento e aggiornamento di informazioni relative ai film. * Gestione del noleggio e dell'acquisto dei film, aggiornando la disponibilità delle licenze. |

|  |  |
| --- | --- |
| **CartService: Gestione del carrello** | |
| **Responsabilità** | * Sincronizzazione dello stato del carrello con la disponibilità dei film. * Aggiunta di film al carrello per noleggio o acquisto, convalidando la disponibilità e i criteri (es. giorni di noleggio). |
| **Collaborazioni** | * Utilizza MovieService per verificare e aggiornare i dettagli sui film disponibili. |