3. Panoramica del sistema

Come definito nel System Design Document, il sistema è stato suddiviso in tre livelli: Model, View e Controller.

I componenti che verranno testati sono i seguenti:

- Gestione Autenticazione
- Gestione Lista
- Gestione Film
- Gestione Sito
- Gestione Utente
- Storage

È stato scelto di non testare il componente “Gestione Suggerimenti Personalizzati” poiché è stato valutato di minore importanza rispetto agli altri componenti.

5. Criteri di successo e fallimento

Il testing ha successo se l’output osservato è diverso da quello atteso, ossia si parla di successo quando il testing rileva una failure. In questo caso, la failure verrà analizzata e corretta nel caso sia causata da un bug. Viceversa, il testing fallisce se non viene rilevata nessuna failure.

6. Approccio

L’approccio scelto per il testing prevede la suddivisione in tre fasi: test di unità, test di integrazione e test di sistema. In questo modo si avrà la possibilità di testare ogni sottosistema e di trovare e correggere eventuali bug rilevati.

6.1 – Test di unità

Questa fase prevede il testing delle singole funzionalità implementate dal sottosistema Storage nel layer Model. Attraverso il framework JUnit verranno testati i DAO, ossia le classi che si occupano di gestire gli oggetti del sistema, i JavaBean.

6.2 – Test di integrazione

Per questa fase è stato scelto di procedere col testing in maniera Bottom-Up, che prevede prima il test dei sottosistemi indipendenti e successivamente verranno testati i sottosistemi che utilizzano i servizi di quelli testati precedentemente.
Nel nostro sistema, il sottosistema Storage rappresenta l’unico sottosistema indipendente, mentre i sottosistemi dipendenti sono costituiti da quelli situati nel layer Controller.
Visto che il sottosistema Storage è stato già testato individualmente dalla fase di test di unità, in questa fase di test di integrazione verranno testate le funzionalità offerte dai sottosistemi presenti nel layer Controller, implementate dalle diverse Servlet, attraverso i framework JUnit e Mockito.

6.3 – Test di sistema

Questa fase prevede il testing dell’intero sistema, attraverso il framework Selenium, sfruttando la tecnica di Black-Box, dividendo gli input di test in classi.