Università degli studi di Salerno

***Corso di Laurea in Informatica***

******

***PROGETTO DI INGEGNERIA DEL SOFTWARE – System Design Document***

***“BUY & SEE”***

**Docente:**

Andrea De Lucia

# Studenti:

##### Nome Matricola

## Fabio Curci 0512103296

## Francesca Di Mauro 05121

## Francesco Ciampa 05121

## Anna Santoro 05121

**Indice**

1. Introduzione

1.1. Scopo del sistema

1.2. Obiettivi di design

1.2.1 Criteri di Performance

1.2.2 Criteri di Affidabilità

1.2.3 Criteri di Costi

1.2.4 Criteri di Manutenzione

1.2.5 Criteri per l’Utente Finale

1.3 Definizioni, acronimi e abbreviazioni

1.4 Riferimenti

1.5 Panoramica

1. Architettura del software proposto

2.1. Panoramica

2.2.Decomposizione del sistema

2.3.Mapping Hardware/Software

2.4.Gestione dei dati persistenti

2.5.Controllo degli accessi e della sicurezza

2.6.Controllo del software globale

2.7.Condizioni Boundary

2.7.1Avvio del sistema

2.7.2Terminazione del sistema

2.7.3Fallimento del sistema

1. Servizi dei sottosistemi

3.1 Gestione

3.2 Gestione

3.3 Gestione

3.4 Gestione

4 Glossario

**1****. Introduzione**

**1.1 Scopo del sistema**

L’avvento del world wide web, ha semplificato molto la possibilità di vedere un film. Infatti solitamente, chiunque volesse vedere un film, era costretto a scendere di casa, recarsi al cinema e sostenere il costo del biglietto di un film. Se per alcuni non c’è nessun problema fare questo, per altri può risultare noioso doversi spostare e fare km per raggiungere il cinema nonchè poco conveniente economicamente pagare un prezzo abbastanza elevato per vedere un film. Inoltre le persone disabili sono maggiormente colpite dal problema di doversi spostare, in quanto a volte è necessario che abbiano un accompagnatore che li porti fino al cinema. Infine ci sono tutti quelli che per per problemi lavorativi sono sempre in viaggio e non hanno tempo di andare al cinema. Il nostro sito “Buy & See”, ha lo scopo di eliminare le problematiche relative a spostamenti, tempo e ridurre i costi per il prezzo di un film, consentendo agli utenti che sono in viaggio o comodamente a casa di guardare film gratis o a prezzi modici

**1.2 Obiettivi di Design**

Il sistema Buy & See deve poter essere il più efficiente e intuitivo possibile. Tale efficienza sarà costruita attraverso rapidi tempi di risposta ad ogni genere di input ma anche tramite differenti politiche di tolleranza all’errore. In più si punterà ad avere una buona manutenibiltà attraverso il facile inserimento di nuove funzionalità. Per ottenere gli obiettivi finali, vanno seguiti dei criteri di progettazione tenendo presente: performance, affidabilità, costi, manutenzione e utente finale.

**1.2.1 Criteri di Performance**

|  |  |
| --- | --- |
| **Tempo di risposta** | Il sistema deve essere reattivo per tutte le operazioni più immediate. Per operazioni più elaborate, il sistema deve garantire tempi di risposta ragionevoli nell’ordine delle decine di secondi, ma, essendo un sistema web, molto dipenderà dalla qualità della connessione e dalla congestione della rete sul sistema online. |
| **Throughput** | I picchi di carico, fino a circa 500 utenti simultaneamente collegati, devono essere gestiti dal sistema senza rallentamenti, garantendo fluidità e una latenza molto bassa. Il sistema deve garantire consistenza in tutte le operazioni che includono chiamate al database, specialmente nei momenti di maggiore carico. |
| **Memoria** | Il sistema utilizza un database relazionale per memorizzare tutti i dati. La mole dei dati non rappresenterà un problema di performance del sistema. Vengono adottate strategie di caching in modo da favorire le prestazioni e ridurre il carico sul sistema. |

**1.2.2 Criteri di Affidabilità**

|  |  |
| --- | --- |
| **Robustezza** | I componenti devono essere affidabili ed essere in grado di poter mantenere i propri dati anche in caso di guasti come problemi elettrici, guasti dell'hardware, attacchi informatici, problemi legati al browser. |
| **Disponibilità** | Il nostro sito deve essere sempre disponibile all’uso, 24 ore su 24, da parte degli utenti, grazie ad un server sempre attivo |
| **Tolleranza all’errore** | Il sistema deve essere capace di operare durante condizioni d’errore. Ciò sarà reso possibile tra tutte quelle sotto-parti del sistema che hanno un grado di accoppiamento basso, in modo tale che l’errore in un sottosistema non intacchi le funzionalità di un altro. Inoltre, deve essere in grado di pianificare dei backup periodici dell’intero sistema. |
| **Sicurezza** | Il sistema, tramite username e password, riuscirà ad individuare il tipo di utente e gli permetterà di effettuare solo le operazioni appartenenti alla sua categoria. |

**1.2.3 Criteri di Costo**

|  |  |
| --- | --- |
| **Sviluppo** | Prima dell’inizio dello sviluppo della piattaforma verranno valutati i costi di sviluppo del sistema, costi di manutenzione e costi di amministrazione. |
| **Deployment** | Non sono previsti costi di deployment |

**1.2.4 Criteri di Manutenzione**

|  |  |
| --- | --- |
| **Estensibilità** | La progettazione del sistema sarà condotta in modo da agevolare la facile introduzione di nuove funzionalità utilizzando il linguaggio di markup HTML5, i fogli di stile CSS3 e JQuery. |
| **Modificabilità** | Deve essere possibile intervenire sul codice esistente per correggere eventuali bugs o implementare nuove funzionalità. Bisogna garantire che il codice sia leggibile per rendere agevole la modifica. |
| **Portabilità** | Il nostro sito sarà portabile in quanto sarà accessibile da qualsiasi dispositivo che abbia una connessione ad internet |
| **Leggibilità** | Il codice sarà ben strutturato per semplificare eventuali interventi su di esso. |
| **Tracciabilità dei requisiti** | Grazie alla tracciabilità dei requisiti, sarà possibile effettuare le modifiche necessarie al corretto funzionamento del sistema, valutando correttamente i costi e i rischi che le modifiche porteranno. |

**1.2.5 Criteri per l’Utente Finale**

|  |  |
| --- | --- |
| **Usabilità** | Il sistema deve essere facilmente apprendibile (l’utente deve essere in grado di interagire con il sistema e padroneggiare le funzionalità in modo rapido), deve essere flessibile e robusto (l’utente deve essere in grado di capire quando ha successo nel perseguire i suoi obiettivi nel sistema o quando sta sbagliando qualcosa per poter intervenire). |

**1.3 Definizioni, acronimi e abbreviazioni**

**1.4 Riferimenti**

**1.5 Panoramica**

Il documento si compone di una prima parte in cui vengono introdotti gli obiettivi di design. Verrà poi brevemente spiegata l’architettura scelta per lo sviluppo del sistema ma prima di addentrarci nel discutere l’architettura, è doveroso fare un excursus alle attività di system design che compongono le basi per l’architettura software del sistema.

* **Decomposizione del sistema**: il sistema viene decomposto in sottosistemi allo scopo di poter assegnare ad un team parti di software semplici da sviluppare. Un sottosistema è una collezione di classi, associazioni, operazioni, eventi e vincoli che sono in relazione tra di loro. Un insieme di operazioni correlate forma un servizio. Ogni sottosistema quindi sarà caratterizzato dai servizi che offre ad altri sottosistemi; l’insieme dei servizi che un sottosistema espone sarà denominato Interfaccia(API).
* **Mapping Hardware/Software**: descrive come i sottosistemi vengono assegnati all’hardware e alle componenti. Elenca anche le problematiche introdotte da nodi multipli e dal riuso del software.
* **Gestione dati persistenti**: descrive i dati persistenti memorizzati dal sistema e l’infrastruttura di gestione richiesta per essi.
* **Politiche di accesso e sicurezza**: descrive il modello utente del sistema in termini di una matrice degli accessi, stabilendo in modo più preciso le operazioni e le informazioni effettuabili da ogni singolo attore e come questi si autenticano al sistema.
* **Flusso di controllo globale**: descrive quali operazioni eseguire ed in che ordine, per garantire il corretto flusso di controllo del sistema.
* **Condizioni Limite**: descrive lo start-up, lo shutdown e i comportamenti errati del sistema.

**2. Architettura del software proposto**

**2.1 Panoramica**

Il sistema proposto è un software web-based per la gestione e visione di film. Le tipologie di utenti si dividono in: gestore account, gestore catalogo e spettatore. L’architettura scelta per il sistema è l’architettura MVC che è strutturata in tre componenti, ognuno con un compito diverso all’interno del sistema. I componenti sono i seguenti:

* Model: rappresenta lo strato di businness dell’applicazione e fornisce i metodi per accedere ai dati dell’applicazione
* View: visualizza i dati contenuti nel Model e si occupa dell’interazione con l’utente
* Controller: gestisce la logica dell’applicazione, riceve i comandi dell'utente e li attua modificando lo stato degli altri due componenti.

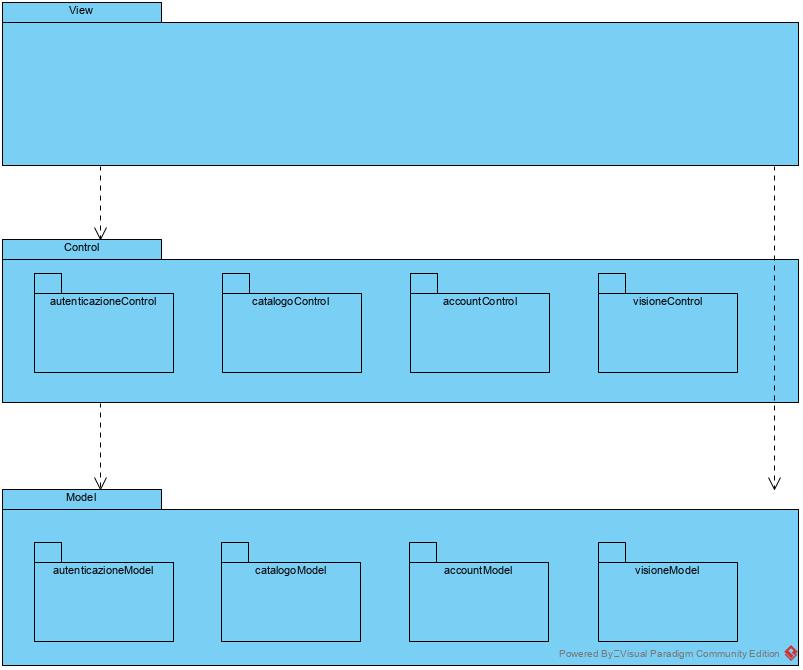
L’architettura MVC è molto utilizzata per sistemi web-based ed è stato scelto poichè tra i suoi vantaggi permette l’indipendenza fra le varie componenti in modo da poter suddividere il lavoro nei casi in cui, al sistema, ci debbano lavorare più persone con competenze diverse. Il pattern MVC consente anche una netta separazione tra la logica di presentazione e la logica di businness. Inoltre è un modello che si adatta perfettamente all’utilizo delle tecnologie Java, Servlet e JSP, che utilizzeremo per lo sviluppo del sistema. Il sistema, poi, verrà decomposto in sottosistemi per renderlo sia più facilmente progettabile, sia più facilmente modificabile per l’aggiunta di nuove funzionalità. Infine, per decomporre il sistema in sottosistemi, si deve rendere minima la dipendenza fra i vari sottosistemi e rendere massima la coesione all’interno degli stessi.

**2.2 Decomposizione del sistema**

Abbiamo decomposto il sistema nei seguenti sottosistemi:

* Autenticazione
* Catalogo
* Account
* Visione Contenuti

Di seguito ecco il diagramma che mostra le relazioni di dipendenza tra i package di ogni sottosistema e tra i diversi sottosistemi



**2.2.1 Autenticazione**

Questo sottosistema si occupa della gestione dell’autenticazione ed è suddiviso in:

* AutenticazioneControl
* AutenticazioneModel

AutenticazioneControl si occupa della logica di controllo relativa al sottosistema di Autenticazione. Il suo compito è quello di gestire i dati relativi all’autenticazione forniti dall’utente e di modificare il Model.

AutenticazioneModel, invece, si occupa della logica applicativa relativa al sottosistema di Autenticazione.

Questo sottosistema fornisce le seguenti funzionalità:

* Registrazione: è l’operazione che consente la registrazione al sito
* Login: è l’operazione che consente l’accesso al sito
* Logout: è l’operazione che consente la disconessione dal sito
* Recupero Password: è l’operazione che consente il recupero della password

**2.2.2 Catalogo**

Questo sottosistema si occupa della gestione del catalogo ed è suddiviso in:

* CatalogoControl
* CatalogoModel

CatalogoControl si occupa della logica di controllo relativa al sottosistema Catalogo. Il suo compito è quello di gestire l’interazione dell’utente con il catalogo e di modificare il Model.

CatalogoModel, invece, si occupa della logica applicativa relativa al sottosistema Catalogo.

Questo sottosistema fornisce le seguenti funzionalità:

* Ricerca film
* Sfoglia catalogo per genere
* Sfoglia catalogo per anno
* Sfoglia catalogo per tutti i film
* Sfoglia catalogo per sezione novità
* Rimozione film dal catalogo
* Aggiorna sezione novità
* Richiesta aggiunta film
* Modifica catalogo

**2.2.3 Account**

Questo sottosistema si occupa della gestione dell’account ed è suddiviso in:

* AccountControl
* AccountModel

AccountControl si occupa della logica di controllo relativa al sottosistema Account. Il suo compito è quello di gestire l’interazione dell’utente con le operazioni relative all’account e di modificare il Model.

AccountModel, invece, si occupa della logica applicativa relativa al sottosistema Account.

Questo sottosistema fornisce le seguenti funzionalità:

* Visualizza dati personali
* Modifica dati personali
* Modifica password
* Invia segnalazione
* Visualizza segnalazione
* Visualizza film acquistati

**2.2.4 Visione**

Questo sottosistema si occupa della gestione della visione dei contenuti ed è suddiviso in:

* VisioneControl
* VisioneModel

VisioneControl si occupa della logica di controllo relativa al sottosistema Visione. Il suo compito è quello di gestire l’interazione dell’utente con le operazioni relative alla visione dei contenuti e di modificare il Model.

VisioneModel, invece, si occupa della logica applicativa relativa al sottosistema Visione.

Questo sottosistema fornisce le seguenti funzionalità:

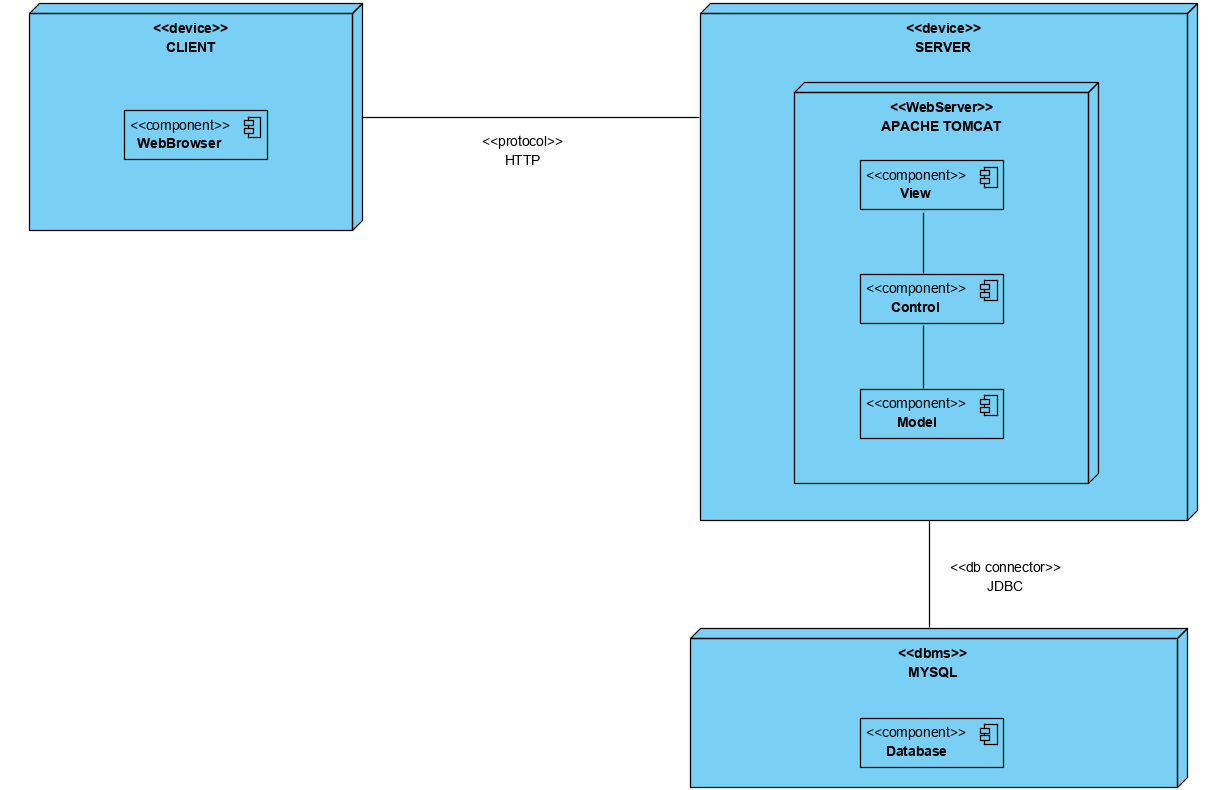
* Visione trailer
* Visione film
* Acquisto film
* Rilascio recensione
* Rilascio valutazione
* Modera recensioni
* Banna utenti

Il sistema è stato decomposto nei sottosistemi appena illustrati per permettere progettazione più semplice e la possibilità di poter aggiungere funzionalità in modo più facile possibile. La decomposizione del sottosistema è stata effettuata cercando di mantenere massima la coesione all’interno degli stessi sottosistemi e minima l’accoppiamento fra i vari sottosistemi. Infatti nella nostra decomposizione, ogni sottosistema ha funzionalità che sono altamente coese fra loro e non hanno un forte accoppiamento con gli altri sottosistemi. Cosi facendo, riduciamo la possibilità che errori o cambiamenti all’interno di un sottosistema, vadano a toccare altri sottosistemi.

Il pattern architetturale scelto è MVC.

L’architettura MVC è molto utilizzata per sistemi web-based ed è stato scelto poichè tra i suoi vantaggi permette l’indipendenza fra le varie componenti in modo da poter suddividere il lavoro nei casi in cui, al sistema, ci debbano lavorare più persone con competenze diverse. Il pattern MVC consente anche una netta separazione tra la logica di presentazione e la logica di businness. Inoltre è un modello che si adatta perfettamente all’utilizo delle tecnologie Java, Servlet e JSP, che utilizzeremo per lo sviluppo del sistema. Il sistema, poi, verrà decomposto in sottosistemi per renderlo sia più facilmente progettabile, sia più facilmente modificabile per l’aggiunta di nuove funzionalità. Infine, per decomporre il sistema in sottosistemi, si deve rendere minima la dipendenza fra i vari sottosistemi e rendere massima la coesione all’interno degli stessi.

**2.3 Mapping Hardware/Software**



Il sistema utilizza un’architettura client/server, in cui un server fornisce servizi a più client. Su una macchina client è eseguito un browser web che consente all’utente di interagire a livello View (lato server) per inoltrare richieste e visualizzare le risposte ricevute. La macchina server gestisce la logica applicativa e i dati persistenti. La comunicazione tra client e server avviene tramite protocollo HTTP. Questo protocollo permette di trasferire ipertesti, consentendo a due macchine, client e server, di interagire attraverso un meccanismo di richiesta e risposta. Il client inoltra una richiesta al server che verrà soddisfatta con la risposta di quest’ultimo. Per il client, le specifiche hardware sono una qualsiasi macchina dotata di connessione a internet, mentre per il software, un sistema operativo con un web browser installato. Per il server, invece, le specifiche hardware consistono di una macchina connessa a internet capace di immagazzinare dati a sufficienza. Le specifiche software necessarie comprendono un Database Management System (MySQL) per la gestione dei dati persistenti, un Web Server (Apache Tomcat) per la gestione della logica applicativa e della comunicazione con più client.

**2.4 Gestione dei dati persistenti**

Il nostro sistema memorizza i dati adottando un database di tipo relazionale che garantisce:

* Accesso efficiente ai dati
* Tempistiche di risposta basse
* Ampio spazio di archiviazione
* Accesso concorrente ai dati
* Affidabilità dei dati
* Privatezza dei dati: i componenti devono avere consistenza, sicurezza e affidabilità ed essere in grado di poter mantenere i propri dati anche in caso di attacchi informatici e guasti dovuti a eventi esterni, di tipo hardware e software

Il tipo di database utilizzato è relazionale MySql.

**2.5 Controllo degli accessi e della sicurezza**

**Matrice degli accessi per ideentificare**

**2.6 Controllo del software globale**

**2.7 Condizioni Boundary**

**2.7.1 Avvio del sistema**

Per il primo avvio del sistema è necessario l'avvio di un web server che fornisca il servizio di un Database MySQL per la gestione dei dati persistenti e l’interpretazione ed esecuzione del codice lato server. In seguito, verrà mostrata l’home page della piattaforma dalla quale è possibile entrare nell’area della Login, dove sarà possibile effettuare l’accesso all’interno del sistema.  
Una volta effettuato l'accesso, sarà possibile effettuare tutte le operazioni consentite, che saranno diverse in base al tipo dell’utente

**2.7.2 Terminazione del sistema**

Al momento della chiusura della pagina web, si ha la terminazione del sistema come un regolare logout

**2.7.3 Fallimento del sistema**

Nel caso si verifichi un errore dovuto all’hardware o al software si cercherà di ripristinare una configurazione del sistema precedente allo stato d’errore. Poiché i dati sono gestiti dal DBMS non c’è alcun rischio di perderli. Tuttavia, non è da escludere la perdita dei dati, se si verifica un guasto al supporto di memorizzazione dei dati nel database-server. Per minimizzare questo rischio, si eseguiranno periodicamente dei backup del database del sistema e periodicamente il sistema hardware verrà sottoposto a controlli.

**3. Servizi dei sottosistemi**

**3.1 Autenticazione**

**3.2 Catalogo**

**3.3 Account**

**3.4 Visione**

**4. Glossario**