Università degli Studi di Salerno
Corso di Ingegneria del Software

RogueGames
System Design Document
Versione 0.1

Data: 24/11/2024

Coordinatore del progetto:

Nome	Matricola

Partecipanti:

Nome	Matricola
Marco Battaglia	0512116374
Giovanni Lavorato	0512116371
di Caprio Domenico	0512118543
De Vita Antonio	0512116944

Scritto da:	Giovanni Lavorato Battaglia Marco di Caprio Domenico De Vita Antonio

Revision History

Data	Versione	Descrizione	Autore
24/11/2024	0.1	Stesura del System Design Document (SDD)	Giovanni Lavorato Battaglia Marco di Caprio Domenico De Vita Antonio

Indice

1.Introduzione

1.1 Purpose of the system

Lo obiettivo del sistema è di creare una piattaforma user-friendly che permette a tutti gli appassionati di videogiochi di poter trovare i titoli che desiderano per le loro piattaforme preferite.
Si tratta di una e-commerce online basato sulla vendita di videogiochi e di tutti i prodotti inerenti a quel mondo.
Il sito offre agli utenti una serie di funzionalità per poter cercare i loro videogiochi in modo rapido e veloce, mettendo a loro disposizione una serie di metodi per poter trovare i loro titoli.
Inoltre, gli verrà permesso di visualizzare tutti i dettagli di un gioco e di poterlo acquistare con tanta semplicità, solo co n previo accesso al loro account del sito.
Agli utenti loggati viene offerto anche la possibilità di gestire e visualizzare il proprio profilo, di poter usufruire di sconti e di poter accedere al proprio storico degli ordini.
Infine, garantisce un elevata sicurezza per le transazioni online.
Per poter raggiungere il sito, basterà loro avere un qualsiasi dispositivo con connessione ad internet e con un qualsiasi motore di ricerca. Basterà digitare il nome del sito e si verrà direttamente indirizzati sulla homepage.
Ai gestori, invece, gli verranno forniti una serie di interfacce per potere: gestire il catalogo, consentendo di poter aggiungere, modificare o eliminare prodotti da esso; gestire e visualizzare lo storico degli ordini effettuati da tutti gli utenti del sito.

1.2 Design goals

1.2.1 Criteri di End User

1.2.1.1 Usabilità

L'interfaccia utente deve essere intuitiva, seguendo convenzioni standard, con guide online e documentazione per facilitare la navigazione e l'uso del sito da parte di utenti di ogni livello.

1.2.2 Criteri di Affidabilità

1.2.2.1 Disponibilità

Il sito deve essere attivo 24 ore su 24, 7 giorni su 7, garantendo un'alta disponibilità per gli utenti.

1.2.2.2 Robustezza

Il sito deve garantire che il sistema funzioni anche se l’utente inserisce input sbagliati

1.2.2.3 Tolleranza ai fault

Il sito deve gestire correttamente tutte le eccezioni, mantenendo la stabilità del servizio anche in caso di errori imprevisti.

1.2.2.4 Protezione

Il sito deve garantire la capacità di resistere ed evitare attacchi malevoli.

1.2.3 Criteri di Performance

1.2.3.1 Throughput

Il sito deve consentire a più utenti di collegarsi simultaneamente senza problemi di sovraccarico, garantendo la scalabilità e robustezza del sistema.

1.2.3.2 Tempo di risposta

Il sito deve rispondere alle richieste dell'utente entro due secondi, garantendo tempi di risposta rapidi per una migliore esperienza di navigazione.

1.2.4 Criteri di Supportabilità

1.2.4.1 Mantenibilità

Il sito deve essere facilmente aggiornabile e mantenibile per permettere future modifiche o aggiustamenti con minimo impatto sull'operatività.

1.2.4.2 Estensibilità

Il sito deve poter permettere di aggiungere nuove funzionalità in modo facile.

1.2.4.3 Portabilità

Il sito deve poter essere visualizzabile su qualsiasi dispositivo, sia esso un computer o dispositivo mobile, che abbia una connessione ad internet e che faccia uso di un qualsiasi motore di ricerca.

1.3 Definition, acronyms and abbreviations

RAD: Requirements Analysis Documentation

DBMS: DataBase Managment System

1.4 References

RAD_RogueGames

2. Current software architecture

L'architettura di RogueGames è progettata con un approccio modulare e orientato ai servizi per garantire scalabilità e facilità di manutenzione. Il sistema si basa su un'architettura three-tier con pattern MVC che include componenti per l'interfaccia utente, la logica del gioco e la gestione dei dati persistenti.

3. Proposed software architecture

3.1 Overview

Il sistema si basa sul modello Three-tier caratterizzato dalla separazione di tre livelli, ciascuno con responsabilità distinte:

· Presentation Layer:

Questo livello è responsabile di presentare i dati all'utente in un formato comprensibile, e interagisce con l'application layer per inviare richieste e ricevere dati.

· Application Layer:

Questo livello si occupa di gestire la logica di business. È responsabile della manipolazione dei dati, dell'esecuzione delle operazioni richieste dagli utenti.

· Data Access Layer:

Questo livello si occupa dell'accesso ai dati e della persistenza.

3.2 Subsystem decomposition

Presentation Layer

-GestoreGUI: Interfacce disponibili esclusivamente al gestore.

-UtenteProdottiGUI: Interfacce del profilo, modifiche di dati personali, ordini effettuati, login, registrazione, logout e Interfacce del carrello, preferiti, catalogo e dettagli prodotti.

Application Layer

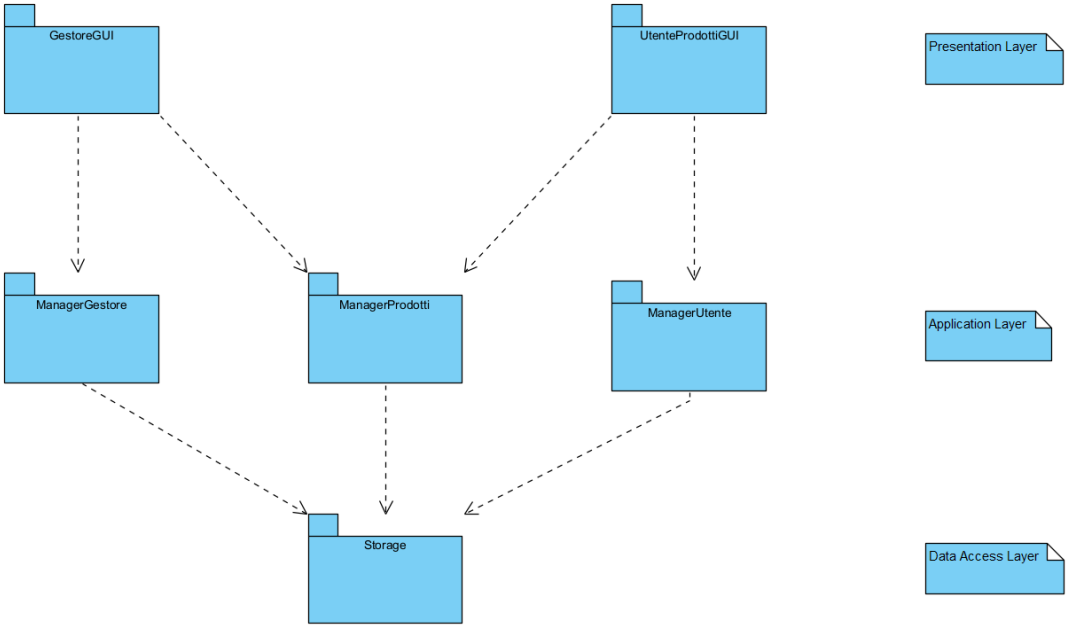
-ManagerUtente: Coordina la logica dei dati, operazioni e stati dell'utente.

-ManagerGestore: Coordina la logica per le operazioni effettuabili dal gestore.

-ManagerProdotti: Coordina la logica per le operazioni effettuabili sui prodotti e i loro stati

Storage

-Storage: Si occupa di conservare i dati degli user, gestori e dei prodotti.



3.2.1 Design Pattern

I Design Pattern utilizzati sono: MVC

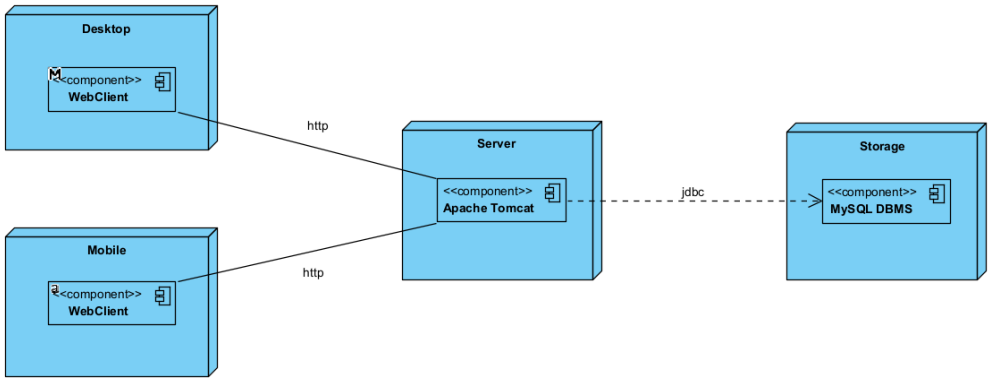
Il sistema si basa sul modello MVC caratterizzato dalla separazione di 3 livelli, ciascuno con responsabilità distinte:

Model: Gestisce i dati e la logica dell'applicazione. È responsabile di interagire con il database e aggiornare la View quando i dati cambiano.

View: Si occupa della presentazione grafica e dell'interfaccia utente. Mostra i dati ricevuti dal Model e invia input dell'utente al Controller.

Controller: Funge da intermediario tra Model e View. Gestisce le richieste dell'utente, elabora la logica, e aggiorna il Model o la View di conseguenza.

3.3 Mapping hardware/software



Il sistema è suddiviso in tre parti abbiamo:

1. **Utenti/Desktop e Mobile**: Gli utenti interagiscono con il sistema attraverso un WebClient. Questo avviene grazie all'utilizzo del protocollo HTTP.

2. **Server**: Sistema ha come suo centro il server, che ospita il Web Server Apache Tomcat. Questa applicazione gestisce le richieste degli utenti e si interfaccia con lo storage del sistema.

3. **Storage**: Lo storage del sistema contiene un DBMS MySQL. Il server accede al DBMS usando il protocollo JDBC per recuperare o memorizzare i dati.

3.4 Persistent data management

I dati persistenti sono gestiti tramite DBMS relazionali. Per incapsulare l'accesso ai dati si utilizza un ORM (object-Relational Mapping). Questo approccio ci permette di semplificare la gestione dei dati e astrarre la complessità del database sottostante.
L'ORM semplifica le operazioni di accesso ai dati e consente di rappresentare gli oggetti dell'applicazione in modo più diretto nella base di dati relazionale. Inoltre, agisce come uno strato intermedio tra le entità del sistema e la base di dati, semplificando le operazioni di creatura, lettura, scrittura e modifica (Create, Read, Update, Delete)..

3.5 Access control and security

Per l'autenticazione al sistema l'utente inserirà email e la password, di quest'ultima verrà eseguita una funzione di hash dopodiché verrà nel Database.
Di seguito riportiamo la matrice degli accessi.

3.5.1 Matrice degli accessi ManagerUtente

ATTORI	CLASSE	FUNZIONI
Ospite	Guest	Singnin()
Utente	User	Login() Logout() modificaProfilo() aggiungiCartaCredito() aggiungiIndirizzoSpedizione() rimuovereCartaCredito() rimuoviIndirizzoSpedizione() VisualizzoOrdini()

3.5.2 Matrice degli accessi ManagerProdotti

ATTORI	CLASSE	FUNZIONI
Utente	User	aggiungialCarrello() aggiungiaiPreferiti() rimuoviprodottoPreferiti() rimuoviprodottoCarrello() modificaQnt() ordinamentoData() filtraggioPiattaforma()

3.5.3 Matrice degli accessi ManagerGestori

ATTORI	CLASSE	FUNZIONI
Gestore	Manager	Login() AggiungiProdotto() ModificaProdotto() RimuoviProdotto() FiltraggioOrdiniPerData() FiltraggioOrdiniPerEmail()

3.6 Global software control

Il nostro sistema avrà un controllo esplicito di tipo centralizzato, il controllo risiede in un dispatcher. Le richieste sono gestite attraverso il Web Server, che funge da intermediario per il sistema. Inizialmente, vengono sottoposte a filtri dedicati per operazioni di autenticazione e gestione delle sessioni. Successivamente le richieste vengono smistate ai rispettivi controller. Questi controller interagiscono con i servizi che contengono la logica di business del sistema. Una volta completate le elaborazioni, le risposte vengono inviate dal controller al client.

3.7 Boundary conditions

3.7.1 Start-up

Per l'inizializzazione del sistema bisogna avviare il DBMS, ovvero MySQL, così da poter accedere ai dati persistenti. L'apertura della connessione al database MySQL viene gestita attraverso JDBC. Dopodiché, bisogna avviare il Web Server, nonché Tomcat dove avviene il deployment dell'applicazione, la quale espone le funzionalità del sistema in modo trasparente agli utenti.

3.7.2 Shutdown

In questa fase avviene lo spegnimento del DBMS e del Web Server, prima però l'applicazione si assicura che tutte le attività in corso siano completate prima della terminazione

3.7.3 Error Behavior of the system

In caso di problemi di connessione al database, il sistema ritenta la connessione.
In caso di problemi di alimentazione viene effettuato il ripristino all'ultimo stato persistente.

4. Subsystems services

4.1 Servizi della Manager Utente

Servizio	Descrizione
singin()	Servizio che consente ad un utente di potersi registrare
login()	Servizio che consente ad un utente registrato di potersi autenticare
logout()	Servizio che consente ad un utente autenticato di poter uscire dal sistema
modificaProfilo()	Servizio che consente ad un utente autenticato poter modificare tutti i suoi dati anagrafaci
aggiungiCartaCredito()	Servizio che consente ad un utente autenticato di poter aggiungere un metodo di pagamento
aggiungiIndirizzoSpedizione()	Servizio che consente ad un utente autenticato di poter aggiungere l'indirizzo in cui verranno consegnati i prodotti
rimuovereCartaCredito()	Servizio che consente ad un utente autenticato di poter rimuovere un metodo di pagamento
rimuoviIndirizzoSpedizione()	Servizio che consente ad un utente autenticato di poter rimuovere un indirizzo di spedizione

VisualizzoOrdini()	Servizio che consente ad un utente autenticato di poter visualizzare il suo personale elenco di ordini effettuati

4.2 Servizi della Manager Prodotti

Servizio	Descrizione
aggiungialCarrello()	Servizio che consente di poter aggiungere un prodotto al carrello
aggiungiaiPreferiti()	Servizio che consente di poter aggiungere un prodotto dai preferiti
rimuoviprodottoCarrello()	Servizio che consente di poter rimuovere un prodotto al carrello
rimuoviprodottoPreferiti()	Servizio che consente di poter rimuovere un prodotto dai preferiti
modificaQnt()	Servizio che consente di poter modificare la quantità di un prodotto presente nel carrello che si desidera acquistare
ordinamentoData()	Servizio che consente di poter ordinare i prodotti in base alla loro data di rilascio
filtraggioPiattaforma()	Servizio che consente di poter filtrare i prodotti in base alla loro piattaforma di appartenenza

4.3 Servizi della Manager Gestori

Servizio	Descrizione
login()	Servizio che consente ad un gestore di potersi autenticare
logout()	Servizio che consente ad un gestore autenticato di poter uscire dal sistema
aggiungiProdotto()	Servizio che consente ad un gestore di poter aggiungere un prodotto al catalogo
modificaProdotto()	Servizio che consente ad un gestore di poter modificare le informazioni di un prodotto presente nel catalogo
rimuoviProdotto()	Servizio che consente ad un gestore di poter rimuovere un prodotto presente nel catalogo
filtraggioOrdiniPerData()	Servizio che consente ad un gestore di poter filtrare tutti gli ordini effettuati dagli utenti in base alla data di acquisto
filtraggioOrdiniPerEmail()	Servizio che consente ad un gestore di poter filtrare tutti gli ordini effettuati dagli utenti in base alla loro email