Università degli Studi di Salerno
Corso di Ingegneria del Software

RAAF-GAMING
System Design Document
Versione 1.0

Data: 11/11/2021

Partecipanti:

Nome	Matricola
Rocco Iuliano	0512106804
Francesco Peluso	0512107194
Antonio De Lucia	0512109225
Antonio Maddaloni	0512107890

Revision History

Data	Versione	Descrizione	Autore
11/11/2021	1.0	Prima stesura del documento	Membri del team

Indice

1. Introduzione

1.1. Obiettivo del sistema

Lo sviluppo del web ha reso popolare la vendita e l'acquisto di oggetti tramite negozi online, il cosiddetto e-commerce, che durante gli ultimi anni ha cambiato radicalmente il modo in cui le persone fanno shopping. Questo sviluppo ha coinvolto anche il mondo videoludico e ha fatto sì che la richiesta d'acquisto di prodotti videoludici online crescesse in modo esponenziale. Per questo RAAF-GAMING ha l'obiettivo di proporsi in questo settore come un e-commerce di riferimento per la vendita di questi prodotti a prezzo scontato. Inoltre, al giorno d'oggi c'è una grande affluenza sui siti web che offrono la possibilità di acquistare noti videogiochi a prezzi scontati ma è anche molto richiesto una piattaforma web per la vendita di console e abbonamenti ad un prezzo ragionevole, proprio per questo RAAF-GAMING offre la possibilità di acquistare oltre ai videogiochi, le console e gli abbonamenti a prezzi scontati.

1.2. Design Goals

Affidabilità:

- *Messaggi di errore:*
  Il sistema deve rispondere attraverso messaggi di errore in presenza di input non validi inseriti dall'utente. Questo verrà mostrato tramite degli allert sulla pagina contente l'errore e evidenziare graficamente i campi del form che sono errati. **Priorità=** alta.
- *Sicurezza della memorizzazione dati:*
  Il sistema deve far sì che la password del cliente sia criptata. Questo verrà implementato tramite il metodo di crittografia MD5.
  Priorità= alta.
- *Sicurezza nell'estrapolazione dei dati:*
  Il sistema deve estrapolare i dati persistenti in modo sicuro. Ovvero deve poter prevenire iniezioni malevole esterne sui dati persistenti. Questo viene implementato tramite le query parametriche che permettono di prevenire l'SQL injection. **Priorità=** alta.

Performance:

- *Carico di lavoro:*
  Il sistema deve essere in grado di fornire servizio contemporaneamente ad almeno 100 utenti. Ciò si potrà ottenere tramite la scalabilità, ovvero attraverso l'aggiunta di più server dedicati al servizio in questione. **Priorità=** bassa.
- *Tempi di risposta:*

Il sistema deve fornire tempi di risposta minimizzati ad almeno 2 secondi per una maggiore fluidità. Si potrà ottenere tramite un compressore di dati (Gzip) e ci permette di inviare i dati compressi al client così da poterli scaricare entro i 2 secondi prestabiliti. **Priorità=** bassa.

Supportabilità:

- *Architettura del sistema:*
  Il sistema deve avere un'architettura a 3 livelli per facilitare la manutenibilità. Questo è possibile perché ci basiamo sul modello MVC, infatti, nel nostro sistema abbiamo i DAO(che rappresentano la logica di business) e i DTO e permettono di accedere e modificare i dati persistenti, le servlet che ci permettono di gestire il flusso dei dati e le JSP che vengono utilizzate per creare pagine dinamiche. **Priorità=** alta.

Usabilità:

- *Graphic User Interface:*
  Il sistema deve fornire un'interfaccia grafica. Viene implementata tramite le jsp che sono pagine dinamiche create dalle servlet. **Priorità=** alta.
- *Menu contestuale:*
  Il sistema deve fornire un menu contestuale che permette di non smarrirsi all'interno del sito, cioè di sapere dove si trova in qualsiasi momento. Verrà implementato un oggetto grafico all'interno della navbar che in base al ruolo dell'utente conterrà un menù diverso. **Priorità=** alta.

- *Nav-bar:*
  Il sistema deve fornire una nav-bar grafica con una componente grafica che dia la possibilità di poter raggiungere la homepage ed effettuare varie ricerche all'interno del sito. Verrà posta in alto nella pagina e conterrà: a sinistra il logo del sito che è un'immagine cliccabile che ci permette di tornale alla homepage, al centro è presente una piccola barra di ricerca, a destra ci sarà il menu contestuale e il carrello. Al di sotto del logo verranno posti degli oggetti grafici che ci permettono di visualizzare diverse categorie di prodotti. **Priorità=** alta.

- *Sistema Responsive:*
  Il sito deve essere responsive, cioè adattarsi a vari dispositivi, ovvero: Smartphone, Tablet, Personal computer. Questo è implementato grazie a Bootstrap. **Priorità=** alta.

1.3. Riferimenti

Riferimento al R.A.D., SDD_dataManagement

2. Architettura del Sistema Corrente

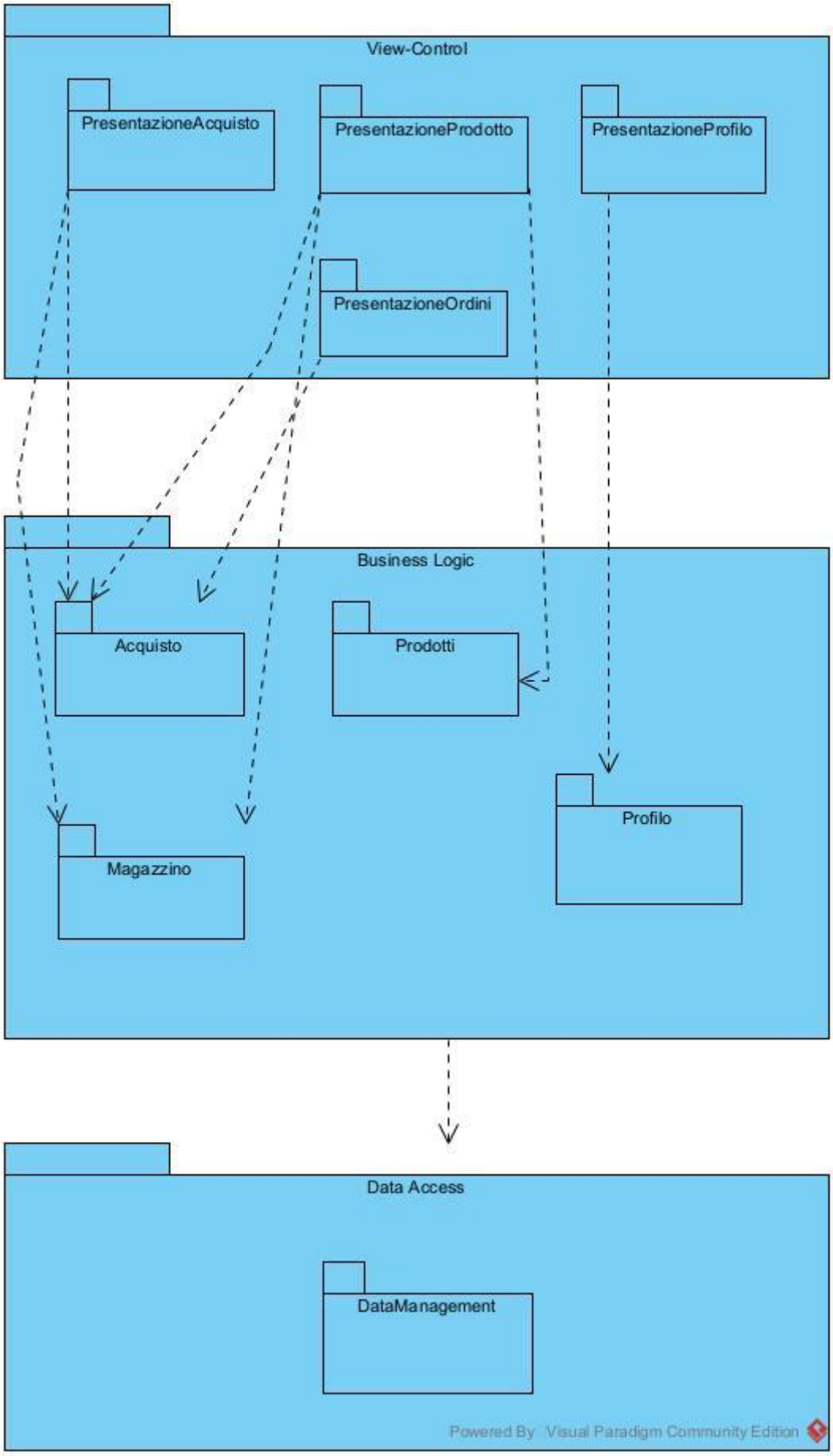
Non avendo un sistema esistente, non abbiamo nessun riferimento ad un'architettura corrente.

3. Architettura del Sistema Proposto

3.1. Decomposizione in Sottosistemi

3.1.1. Layering & Partitioning

Decomponiamo il nostro sistema in 3 layers, che si occupano di gestire aspetti e funzionalità differenti. La decomposizione è basata su un particolare pattern per l'architettura software chiamato MVC (MODEL-VIEW-CONTROL).


View-Control
PresentazioneAcquisto
PresentazioneProdotto
PresentazioneProfilo
PresentazioneOrdini
Business Logic
Acquisto
Prodotti
Profilo
Magazzino
Data Access
DataManagement
Powered By Visual Paradigm Community Edition

View-Control

PresentazioneAcquisto = visualizzazione e gestione del carrello poter procedere all'acquisto e per poter eliminare un prodotto dal carrello.

PresentazioneProdotto = visualizzazione e gestione: ricerca, recensione, aggiunta prodotto al carrello, visualizzare pagina informativa.

PresentazioneProfilo = gestione e visualizzazione di: pagina di autenticazione, pagina di registrazione, modifica dati personali.

PresentazioneOrdini = gestione e visualizzazione degli ordini.

Business Logic

Acquisto = Offre servizi per: fare un ordine, ottenere tutti gli ordini e gestire la consegna.

Prodotti = Offre servizi per: aggiungere prodotto al carrello, rifornire prodotto, aggiungere recensione, ottenere tutti i prodotti per una ricerca, vedere la disponibilità dei prodotti.

Profilo = Offre servizi per: autenticazione utente, registrazione, modifica dati personali.

Magazzino = Offre servizi per controllare se un prodotto è disponibile e decrementare o aumentare la quantità di un prodotto.

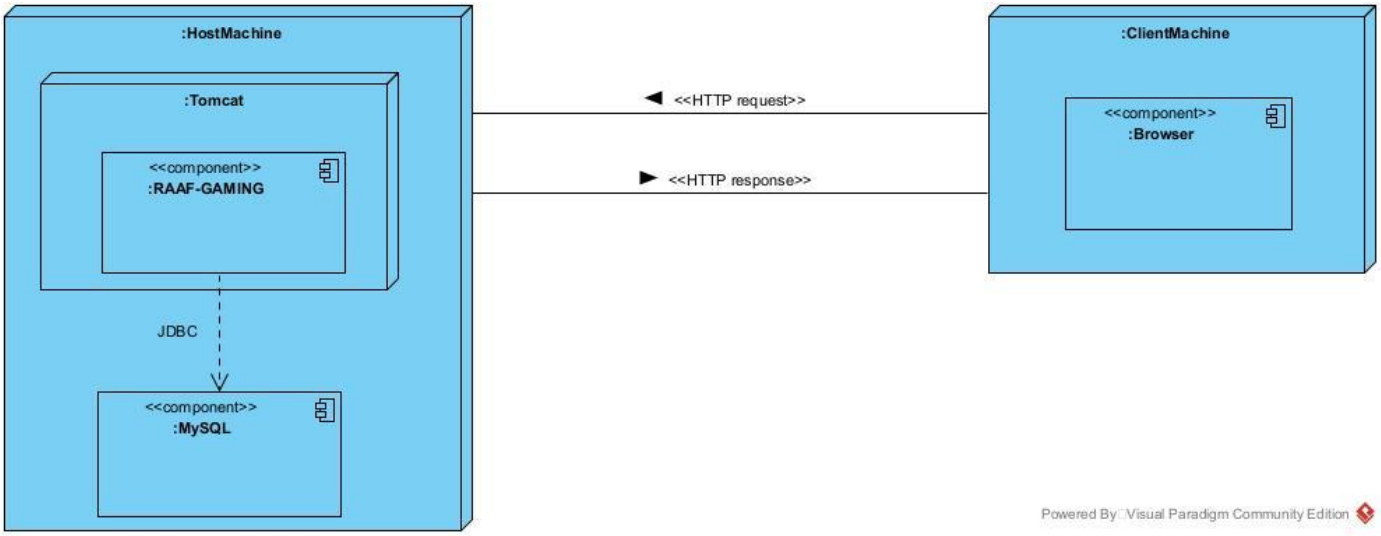
Data Access

DataManagement = Si occupa di transazione, garantire le proprietà ACID e per fare ciò delegherà il DBMS relazionale MySQL.

3.2. Hardware/Software Mapping

RAAF-GAMING utilizza un'architettura Client-Server. Il Web Server è realizzato da Apache Tomcat ed è situato su una singola macchina, l'interfaccia utente è realizzata utilizzando pagine Java Server Page (JSP) e dalle Servlet che fanno da control. La logica di Business è implementata tramite DAO (Data Access

Object) che permettono di accedere e modificare i dati del DB (si trova sulla stessa macchina server) tramite il driver JDBC (Java Database Connectivity) mentre i Bean ci permettono di manipolare i dati ottenuti dal DB tramite i DAO. Il Client è rappresentato dal Web Browser utilizzato dall'utente, che raggiunge il sito tramite un URL utilizzando un protocollo http, il Web Server accetta le richieste dei client reindirizzandoli alla homepage del sito.



3.3. Persistent and Data Managment

La gestione dei dati persistenti è descritta nel documento SDD_dataManagment.

3.4. Access Control and Security

Ogni tipo di utente: utente autenticato, utente visitatore, gestore magazzino e gestore degli ordini avranno a disposizione diverse interfacce grafiche (ad eccezione per l'utente visitatore e autenticato che hanno alcune interfacce in comune) in modo che ogni utente possa accedere solo alle rispettive funzionalità che rientrano nella sua categoria di utenza.

Il gestore del magazzino e il gestore degli ordini per poter interagire con

gli oggetti del sistema devono necessariamente essere autenticati nel sistema. L'utente visitatore ha un'interazione parziale con gli oggetti riguardanti il cliente, per poter usufruire di funzionalità aggiuntive è necessaria l'autenticazione.

Per poter autenticare i vari utenti è necessaria una mail e una password.

Per l'autenticazione dei gestori e dei clienti utilizziamo due pagine differenti.

Password e dati della carta di credito vengono criptati tramite una funzione SQL MD5.

La matrice degli accessi è descritta nel documento MatriceDegliAccessiSDD

3.5. Global Software Control

Il flusso di controllo del nostro sistema è un controllo centralizzato. Più nel dettaglio è un event-driven control in quanto la nostra applicazione in attesa di dati da parte del client. Alla ricezione dei dati il flusso di controllo all'interno del nostro sistema è procedure-driven control in quanto gli oggetti tra loro per comunicare si cedono il controllo.

3.6. Boundary Conditions

Non abbiamo Boundary Condition a causa dell'istallazione di un web container come Tomcat quindi per far partire e stoppare il server abbiamo bisogno solo di premere un tasto di avvio/spegnimento.