INFINITY GAMES

Object Design Document

Membri del Team

Falciano Ciro	0512103596
Schirosa Alessandro	0512104514
Ruggiero Simone	0512106074
De Lucia Antonino	0512105366

REVISION HISTORY

DATA	VERSIONE	DESCRIZIONE	AUTORE
08/12/20	1.0	Prima versione del documento.	C. Falciano, S. Ruggiero, A. Schirosa, A. De Lucia

SOMMARIO

1. INTRODUZIONE

Dopo aver stilato i documenti Requirements Analysis e System Design è necessario porre attenzione sugli aspetti implementativi. Questo documento ha l'obiettivo di produrre un modello che integri in modo coerente tutte le informazioni collezionate nelle fasi precedenti. In particolar modo, in tale documento verranno definite le interfacce delle classi, le operazioni supportate, i tipi dei dati, i parametri delle procedure, i signatures dei sottosistemi definiti nel documento di System Design, i trade-offs e le linee guida.

1.1. Object Design Trade-offs

Comprensibilità vs Tempo:

Il codice del sistema deve essere comprensibile, in modo da facilitare la fase di testing ed eventuali future modifiche da apportare. Al fine di rispettare queste linee guida il codice sarà integrato da commenti volti a migliorarne la leggibilità; tuttavia questo richiederà una maggiore quantità di tempo necessario per lo sviluppo del nostro progetto.

Interfaccia vs Usabilità:

Verrà realizzata un'interfaccia grafica chiara e concisa, usando form e pulsanti predefiniti che hanno lo scopo di rendere semplice l'utilizzo del sistema da parte dell'utente finale.

Sicurezza vs Efficienza:

Il sistema mira ad essere chiuso per garantire la privacy, tuttavia ci limiteremo ad implementare sistemi di sicurezza basati su email e password a causa dei tempi limitati in modo da favorire l'efficienza.

1.2. Linee Guida per la Documentazione delle Interfacce

Naming Convention: alla luce di quanto analizzato, sarà necessario utilizzare nomi:

-Descrittivi
-Pronunciabili
-Di uso comune
-Di lunghezza medio-corta
-Utilizzando solo caratteri consentiti (a-z, A-Z, 0-9)
-Senza prefissi o suffissi

Variabili

-I nomi delle variabili dovranno iniziare con la lettera minuscola, e le parole successive con la lettera maiuscola.

-Le variabili locali non saranno dichiarate all'inizio del blocco che le contiene, ma verranno dichiarate in corrispondenza del punto in cui vengono usate per la prima volta: il tutto è volto alla limitazione del loro scope.

-In ogni riga dovrà esserci un'unica variabile dichiarata, eventualmente allineata con quelle del blocco dichiarativo.

Metodi

-I nomi dei metodi dovranno iniziare con la lettera minuscola e le parole successive con la lettera maiuscola.

-Il nome del metodo sarà costituito da un verbo che ne identifica l'azione seguito da un sostantivo, eventualmente aggettivato.

-Il nome dei metodi accessori e modificatori seguirà, rispettivamente, i pattern getNomeVariabile e setNomeVariabile.

-Nel caso in cui vengano utilizzati costrutti "if", "else", "for", "do" e "while" nei metodi dovranno essere utilizzate le parentesi graffe, anche se essi constano di una sola istruzione.

Classi Java e pagine JSP

-I nomi delle classi e delle pagine dovranno iniziare con la lettera maiuscola, così come le parole successive all'interno del nome.

-I nomi delle classi e delle pagine dovranno corrispondere alle informazioni e le funzioni fornite da quest'ultime.

-Ogni file sorgente .class dovrà contenere una singola classe e dev'essere strutturato come segue.

-Le classi saranno strutturate prevedendo rispettivamente:

1. Dichiarazione della classe pubblica
2. Dichiarazioni di costanti
3. Dichiarazioni di variabili di classe
4. Dichiarazioni di variabili d'istanza
5. Costruttore
6. Commento e dichiarazione metodi e variabili.

-Nel caso in cui una classe avesse costruttori o metodi multipli con lo stesso nome, questi saranno posizionati sequenzialmente senza codice posto fra essi.

Packages:

-Non saranno ammessi caratteri speciali.

-Import statici e non statici saranno specificati in blocchi singoli e separati.

-Gli import statici non saranno usati per le classi annidate ma verranno importate con import tradizionali

1.3. Definizioni, acronimi e abbreviazioni

Acronimi:

RAD: Requirements Analysis Document

SDD: System Design Document

ODD: Object Design Document

Abbreviazioni:

NC: Nessuna Condizione

NA: Nessun Attributo

1.4. Design Pattern

1.4.1 Singleton Pattern

Si è scelto di utilizzare il Singleton Pattern che deve assicurarsi che esista una singola istanza della classe Singleton, che nel nostro caso si chiama *ConPool*, e fornire un accesso globale a tale istanza. L'istanza viene dichiarata come variabile privata statica e creata quando viene inizializzata. Per realizzare il singleton pattern occorre avere:

-una variabile privata statica della classe che rappresenta l'unica istanza creata;

-un metodo pubblico getConnection che restituisce l'istanza.

Il suo scopo è:

-Avere accesso controllato all'unica istanza della classe.

-Centralizzare informazioni e comportamenti in un'unica entità, condivisa dagli utilizzatori.

Il principale vantaggio offerto è:

-Mutua esclusione.

1.5. Riferimenti

- Object-Oriented Software Engineering Using UML, Patterns, and Java, Bernd Bruegge & Allen H. Dutoit.
- RAD_InfinityGames
- SDD_InfinityGames

2. PACKAGES

Il sistema di Infinity Games sarà distribuito in packages:

Il package “controller” conterrà le Java Servlet
Il package “model” conterrà le risorse per l’accesso alle risorse e il relativo formato per la realizzazione dei medesimi.
Il package “front-end” conterrà le pagine css, jsp e js che si occuperanno dell’interazione con l’utente.

2.1. Package Controller

AcquistoServlet.java
AddDesideriServlet.java
AdminCategoriaServlet.java
AdminProdottoServlert.java
AdminUtentiServlet.java
AssistenzaServlet.java
CarrelloServlet.java
CategoriaServlet.java
HomeServlet.java
InitServlet.java
LibreriaServlet.java
ListaDesideriServlet.java
LoginServlet.java
LogoutServlet.java

ModificaProfiloServlet.java
MyServletException.java
OrdiniServlet.java
PopolariServlet.java
ProdottoServlet.java
ProfiloServlet.java
RecentiServlet.java
RegistrazioneFormServlet.java
RegistrazioneServlet.java
RicercaAjaxServlet.java
RicercaServlet.java
ScontoServlet.java
TodoServlet.java
UploadServlet.java
VendutiServlet.java
VerficaEmailServlet.java
VerificaUsernameServlet.java

CLASSE	DESCRIZIONE
AcquistoServlet.java	Controller che permette l'acquisto dei prodotti contenuti nel carrello.
AddDesideriServlet.java	Controller che permette di aggiungere un prodotto alla lista dei desideri.
AdminCategoriaServlet.java	Controller che permette all'amministratore della piattaforma di creare, modificare e rimuovere una categoria di videogiochi.
AdminProdottoServlet.java	Controller che permette all'amministratore della piattaforma di creare, modificare e rimuovere un prodotto.
AdminUtentiServlet.java	Controller che gestisce il reindirizzamento alla pagina dedicata alla gestione degli utenti.
AssistenzaServlet.java	Controller che gestisce il reindirizzamento alla pagina dedicata all'assistenza clienti.
CarrelloServlet.java	Controller che gestisce il carrello.
CategoriaServlet.java	Controller che permette di filtrare i prodotti per categoria.

CLASSE	DESCRIZIONE
HomeServlet.java	Controller che gestisce la visualizzazione della homepage della piattaforma.
InitServlet.java	Controller che si occupa dell'inizializzazione della piattaforma.
LibreriaServlet.java	Controller che gestisce la libreria di prodotti acquistati nello store da parte di un utente.
ListaDesideriServlet.java	Controller che gestisce la lista dei desideri di un utente.
LoginServlet.java	Controller che permette l'autenticazione alla piattaforma.
LogoutServlet.java	Controller che permette il logout dalla piattaforma.
ModificaProfiloServlet.java	Controller che gestisce le modifiche relative alla pagina del profilo dell'utente.
MyServletException.java	Controller che si occupa di mostrare una pagina di errore.
OrdiniServlet.java	Controller che gestisce il reindirizzamento alla pagina degli ordini effettuati da un utente.
PopolariServlet.java	Controller che permette di filtrare i prodotti presenti nello store per popolari.
ProdottoServlet.java	Controller che gestisce il reindirizzamento alla pagina del prodotto.
ProfiloServlet.java	Controller che gestisce il reindirizzamento alla pagina del profilo.
RecentiServlet.java	Controller che permette di filtrare i prodotti presenti nello store per recenti.
RegistrazioneFormServlet.java	Controller che gestisce il reindirizzamento alla pagina del form di registrazione.
RegistrazioneServlet.java	Controller che gestisce la registrazione di un utente alla piattaforma.
RicercaAjaxServlet.java	Controller che gestisce la ricerca di prodotti nello store.
RicercaServlet.java	Controller che gestisce il reindirizzamento alla pagina contenente la lista di prodotti nello store che corrispondo al titolo inserito.
ScontoServlet.java	Controller che permette di filtrare i prodotti presenti nello store per scontati.
TodoServlet.java	Controller che permette la gestione degli utenti della piattaforma da parte dell'amministratore e il reindirizzamento alla pagina della gestione degli utenti.
UploadServlet.java	Controller che permette di aggiornare l'immagine del profilo dell'utente.

CLASSE	DESCRIZIONE
VendutiServlet.java	Controller che permette di filtrare i prodotti presente nello store per più venduti.
VerificaEmailServlet.java	Controller che permette la validazione dell'e-mail durante la modifica del profilo dell'utente.
VerificaUsernameServlet.java	Controller che permette la validazione dell'username durante la modifica del profilo dell'utente.

2.2. Package Model

Carrello.java
CarrelloDAO.java
Categoria.java
CategoriaDAO.java
ConPool.java
Ordini.java
OrdiniDAO.java
Prodotto.java
ProdottoDAO.java
Utente.java
UtenteDAO.java

CLASSE	DESCRIZIONE
Carrello.java	Model che rappresenta le informazioni riguardanti il carrello.
CarrelloDAO.java	Model che permette di eseguire le operazioni sul database per l'entità Carrello.
Categoria.java	Model che rappresenta le informazioni riguardanti le categorie dei prodotti.
CategoriaDAO.java	Model che permette di eseguire le operazioni sul database per l'entità Categoria.
ConPool.java	Model la cui istanza rappresenta la connessione con il database.
Ordini.java	Model che rappresenta le informazioni riguardanti gli ordini effettuati.
OrdiniDAO.java	Model che permette di eseguire le operazioni sul database per l'entità Ordini.
Prodotto.java	Model che rappresenta le informazioni riguardanti il prodotto della piattaforma.
ProdottoDAO.java	Model che permette di eseguire le operazioni sul database per l'entità Prodotto.
Utente.java	Model che rappresenta le informazioni riguardanti tutti gli utenti che possono utilizzare la piattaforma.
UtenteDAO.java	Model che permette di eseguire le operazioni sul database per l'entità Utente.

2.3. Package Front-End

acquistoSuccesso.jsp
admincategoria.jsp
adminprodotto.jsp
adminutenti.jsp
assistenza.jsp
banner.jsp
carrello.jsp
categoria.jsp
desideri.jsp
error.jsp
footer.jsp
index.jsp
libreria.jsp
modificaprofilo.jsp
ordini.jsp
ordiniUtente.jsp
popolari.jsp
prodotto.jsp
recenti.jsp
registrazioneSuccesso.jsp
ricerca.jsp
sconti.jsp
venduti.jsp
contatti.jsp
login.jsp
notizie.jsp
registrazione.jsp

PAGINA	DESCRIZIONE
acquistoSuccesso.jsp	Pagina che notifica l'acquisto avvenuto con successo.
admincategoria.jsp	Pagina che permette la modifica e l'aggiunta di una categoria da parte dell'amministratore.
adminprodotto.jsp	Pagina che permette la modifica e l'aggiunta di un prodotto da parte dell'amministratore.
adminutenti.jsp	Pagina che permette di visualizzare la lista degli utenti della piattaforma.
assistenza.jsp	Pagina che visualizza i link di reindirizzamento per visualizzare i propri ordini, gestire il proprio account e visualizzare il form per contattare l'assistenza clienti.
banner.jsp	Header della piattaforma.
carrello.jsp	Pagina che permette di visualizzare il carrello.
categoria.jsp	Pagina che permette di visualizzare i prodotti dello store per categoria.
desideri.jsp	Pagina che permette di visualizzare la lista dei desideri.
error.jsp	Pagina che visualizza un messaggio di errore.
footer.jsp	Footer della piattaforma.
index.jsp	Pagina che rappresenta l'homepage della piattaforma.
libreria.jsp	Pagina che permette di visualizzare la libreria di videogiochi acquistati.
modificaprofilo.jsp	Pagina che permette di visualizzare il form dedicato alla modifica del profilo.
ordini.jsp	Pagina che permette di visualizzare gli ordini effettuati da un utente.
ordiniUtenti.jsp	Pagina che permette all'amministratore di visualizzare gli ordini effettuati da un singolo utente.
popolari.jsp	Pagina che permette di visualizzare i prodotti più popolari nello store.
prodotto.jsp	Pagina che permette di visualizzare le informazioni relative ad un prodotto dello store.
recenti.jsp	Pagina che permette di visualizzare i prodotti più recenti nello store.
registrazionesuccesso.jsp	Pagina che notifica la registrazione avvenuta con successo.
ricerca.jsp	Pagina che visualizza la lista di prodotti nello store ricercati per titolo.
sconti.jsp	Pagina che permette di visualizzare i prodotti scontati nello store.

PAGINA	DESCRIZIONE
venduti.jsp	Pagina che permette di visualizzare i prodotti più venduti sulla piattaforma.
contatti.jsp	Pagina che visualizza il form per contattare l'assistenza clienti.
login.jsp	Pagina che visualizza il form di autenticazione alla piattaforma.
notizie.jsp	Pagina che visualizza le informazioni relative agli sviluppatori della piattaforma.
registrazione.jsp	Pagina che visualizza il form di registrazione alla piattaforma.

3. CLASS INTERFACE

3.1. Class Interface Controller

NOME CLASSE
AcquistoServlet
METODI
+doGet(HttpServletRequest request, HttpServletResponse response) : void +doPost(HttpServletRequest request, HttpServletResponse response) :void
PRECONDIZIONI
doGet(HttpServletRequest request, HttpServletResponse response)/ doPost(HttpServletRequest request, HttpServletResponse response) : **context** AcquistoServlet:: doGet(request:HttpServletRequest, response: HttpServletResponse) **pre:** request.getSession().getAttribute(“utente”)!=null && request.getParameter(“confermaAcquisto”)!=null **context** AcquistoServlet:: doPost(request:HttpServletRequest, response: HttpServletResponse) -
POSTCONDIZIONI
-

NOME CLASSE
AddDesideriServlet
METODI
+doGet(HttpServletRequest request, HttpServletResponse response) : void +doPost(HttpServletRequest request, HttpServletResponse response) :void
PRECONDIZIONI
doGet(HttpServletRequest request, HttpServletResponse response)/ doPost(HttpServletRequest request, HttpServletResponse response) : **context** AddDesideriServlet:: doGet(request:HttpServletRequest, response: HttpServletResponse) **pre:** request.getSession().getAttribute(“utente”)!=null && request.getParameter(“idProdotto”)!=null && ! request.getParameter(“idProdotto”).equals(“”) **context** AcquistoServlet:: doPost(request:HttpServletRequest, response: HttpServletResponse) -
POSTCONDIZIONI
-