

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 |
| 29/12/20 | 1.1 | Aggiornamento manager | 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. Façade Pattern

Si è scelto di utilizzare il Façade Pattern che deve assicurarsi di fornire un'interfaccia unificata semplificata, per l'accesso a sottosistemi che espongono interfacce complesse e diverse fra loro.

L'utilizzo del Façade Pattern permette di nascondere la complessità delle operazioni, fornendo un'interfaccia di alto livello per rendere più facile l'utilizzo del sottosistema.

Nel nostro caso, il Façade Pattern è realizzato attraverso le interfacce manager.

### 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

## Manager Registrazione

| DESCRIZIONE MANAGER | | | |
|---|---|---|---|
| Questo manager si occupa di fornire un'interfaccia di controllo per la registrazione dell'utente. | | | |
| **Firma del Metodo** | **Descrizione Metodo** | **Precondizione** | **Postcondizione** |
| public Utente doRetriveByUsername Email(String username, String password, String email) | Metodo incaricato di validare i dati inviati dal form di registrazione e di verificare se l'utente è già registrato o meno alla piattaforma. | email != null && password != null && username != null | Se l'utente non è già registrato alla piattaforma return null, altrimenti return Utente |
| public void addUser(Utente) | Metodo incaricato di registrare un nuovo utente alla piattaforma. | doRetriveByUsername Email()==null | L'utente viene aggiunto alla piattaforma. |

## Manager Login

| DESCRIZIONE MANAGER | | | |
|---|---|---|---|
| Questo manager si occupa di fornire un'interfaccia di controllo per l'autenticazione dell'utente registrato. | | | |
| **Firma del Metodo** | **Descrizione Metodo** | **Precondizione** | **Postcondizione** |
| public void validateInfoLogin (String username, String password) | Metodo incaricato di validare i dati inviati dal form di autenticazione e il loro formato. | username != null && password != null | La richiesta di autenticazione è consentita se i dati inseriti dall'utente rispettano il formato. |
| public Utente doRetriveByUsername Password(String username, String password) | Metodo incaricato di autenticare un utente registrato alla piattaforma tramite i dati inseriti. | doRetriveByUsernam e Password() == null | return Utente else return null |

## Manager Modifica Profilo

| DESCRIZIONE MANAGER | | | |
|---|---|---|---|
| Questo manager si occupa di fornire un'interfaccia di controllo per la modifica del profilo di un utente registrato. | | | |
| **Firma del Metodo** | **Descrizione Metodo** | **Precondizione** | **Postcondizione** |
| public String doRetriveInfoUtente(int IdUtente) | Metodo incaricato di recuperare le informazioni inerenti all' area personale di un utente | IdUtente!=null | Ritorna l'username, immagine del profilo, email. |
| public void validateFormatPassword(String passwordPrecedente, String nuovaPassword) | Metodo incaricato di validare i dati inviati dal form di modifica password e il loro formato. | passwordPrecedente != null && nuovaPassword != null | La richiesta di modifica password è consentita se i dati inseriti rispettano il formato. |
| public boolean checkOldPassword (int idUtente, String passwordPrecedente) | Metodo incaricato di verificare che la password inserita sia uguale a passwordPrecedente | idUtente != null && passwordPrecedente != null | Return true se passwordPrecedente è uguale alla password che si vuole cambiare, return false altrimenti. |
| public void doSavePassword(bool ean flag, int idUtente, String nuovaPassword) | Metodo incaricato di effettuare la modifica della password da parte di un idUtente | idUtente != null && nuovaPassword != null && flag == true | nuovaPassword viene settata come password corrente. |
| public boolean validateRicarica(String metodoPagamento, double importo, int idUtente) | Metodo incaricato di verificare se è possibile ricaricare sul proprio portafogli virtuale l'importo selezionato tramite uno specifico metodo di pagamento esterno al sistema. | Importo != null && Importo > 0 && metodoPagamento != null | Return true se è possibile continuare con la ricarica del portafogli, altrimenti false |
| public void RicaricaPortafogli(double importo, double portafogliUtente) | Metodo incaricato di effettuare la ricarica di un portafogli utente dato l'importo selezionato | Importo != null && Importo > 0 && portafogliUtente != null | portafogliUtente+=importo |

## Manager Ordine

| DESCRIZIONE MANAGER | | | |
|---|---|---|---|
| Questo manager si occupa di fornire un'interfaccia di controllo per gli acquisti sulla piattaforma. | | | |
| **Firma del Metodo** | **Descrizione Metodo** | **Precondizione** | **Postcondizione** |
| public boolean checkTransaction (double portafogliUtente, List<Prodotto>, int idUtente) | Metodo incaricato di verificare il formato e la validità dei dati dell'utente che vuole effettuare un ordine. | List<Prodotto> != null && idUtente != null && portafogliUtente != null | Return true se il totale dei prezzi dei prodotti inseriti nell'ordine <= portafogliUtente, altrimenti return false |
| public void addOrdineDoTransacti on( boolean flag, Ordine ordine, double portafogliUtente) | Metodo incaricato di aggiungere l'ordine allo storico degli acquisti dell'utente ed effettuare la transazione. | flag == true && ordine != null && portafogliUtente != null | Viene scalato il totale di ordine da portafogliUtente |
| public List <Ordine> doRetriveOrdiniByIdUtente(int idUtente) | Metodo incaricato di ritornare la lista di ordine effettuato dall' utente | idUtente != null | Return List<Ordine> !=null altrimenti return List<Ordine>==null |
| public String getInfoOrdine(int IdUtente, int IdOrdine) | Metodo incaricato di recuperare informazioni di un singolo ordine effettuato dall'utente | idUtente != null && idOrdine != null && List <Ordine> doRetriveOrdiniByIdUtente() != null | Return String ordine di quel IdOrdine generato dal quel idUtente |

## Manager Prodotto

| DESCRIZIONE MANAGER | | | |
|---|---|---|---|
| Questo manager si occupa di fornire un'interfaccia di controllo per la gestione del prodotto della piattaforma. | | | |
| **Firma del Metodo** | **Descrizione Metodo** | **Precondizione** | **Postcondizione** |
| public Prodotto getProduct (String titoloProdotto) | Metodo incaricato di controllare che il titolo di un prodotto che si vuole aggiungere non sia già presente nella piattaforma. | titoloProdotto != null | Return null se il prodotto non è già presente, return Prodotto altrimenti |
| public void doSaveProdotto (Prodotto prodotto) | Metodo incaricato di aggiungere un prodotto all'interno della piattaforma. | prodotto != null | prodotto è aggiunto alla piattaforma. |
| public void removeProdotto (int idProdotto) | Metodo incaricato di rimuovere un prodotto all'interno della piattaforma | idProdotto != null | idProdotto è stato rimosso dalla piattaforma. |
| public void doUpdateProdottoValidation ( int Idcategoria, int idProdotto, String descrizione, double prezzo, int sconto, String immagine, String video) | Metodo incaricato di modificare un prodotto nella piattafroma | Idcategoria !=null && idProdotto !=null && descrizione !=null && ! descrizione.equals("") && prezzo != null && prezzo>0 && sconto != null && immagine != null && !immagine.equals("") && video != null && !video.equals("") | prodotto.idProdotto è stato aggiornato con le nuove informazioni |
| public List <Prodotto> doRetrivebyLibreria(int IdUtente) | Metodo incaricato di restituire la lista di prodotti acquistati dall'utente | idUtente !=null | Return List<Prodotto> ! =null altrimenti return List<Prodotto>==null |
| public List <Prodotto> getDesideri(int IdUtente) | Metodo incaricato di restituire la lista di prodotti aggiunti alla lista dei desideri | idUtente !=null | Return List<Prodotto> ! =null altrimenti return List<Prodotto>==null |

| | | | |
|---|---|---|---|
| public boolean checkDesideriById Prodotto(int IdProdotto, int IdUtente) | Metodo incaricato di verificare se un prodotto è stato già aggiunto alla lista dei desideri | idUtente !=null && idProdotto!= null | Return true se il prodotto è stato già aggiunto alla lista; false altrimenti |
| public void addProdottoDesid eri(int IdUtente, int idProdotto) | Metodo incaricato di aggiungere un prodotto ai desideri | checkDesideriByIdPro dotto()==false && idUtente != null && idProdotto != null | Aggiunge quel idProdotto nella lista dei desideri di quell' idUtente |
| public void removeDesideri(int IdUtente, int IdProdotto) | Metodo incaricato di rimuovere un prodotto dai desideri | checkDesideriByIdPro dotto()==true && idUtente != null && idProdotto != null | rimuove quel idProdotto dalla lista dei desideri di quell' idUtente |
| public List<Prodotto> search(String titoloProdotto) | Metodo incaricato di ricercare i prodotti tramite il titolo inserito dall' utente | titoloProdotto != null && ! titoloProdotto.equals ("") | Return List<Prodotto> ! =null altrimenti return List<Prodotto>==nul l |
| public List<Prodotto> doRetriveByIdCate goria(int IdCategoria) | Metodo incaricato di restituire la lista di prodotti filtrati per la categoria | idCategoria != null | Return List<Prodotto> ! =null altrimenti return List<Prodotto>==nul l |
| public Prodotto doRetriveByIdProd otto(int IdProdotto) | Metodo incaricato di restituire la pagina del prodotto | idProdotto != null | Return Prodotto associato a quell' idProdotto |

## Manager Categoria

| DESCRIZIONE MANAGER | | | |
|---|---|---|---|
| Questo manager si occupa di fornire un'interfaccia di controllo per la gestione delle categorie di prodotti della piattaforma. | | | |
| **Firma del Metodo** | **Descrizione Metodo** | **Precondizione** | **Postcondizione** |
| public Categoria getGategoria (String nomeCategoria) | Metodo incaricato di controllare che una categoria di prodotti che si vuole aggiungere non sia già presente nella piattaforma. | nomeCategoria != null | Return null se nomeCategoria non è già presente, return Categoria altrimenti. |
| public void doSaveCategoria(Cat e goria categoria) | Metodo incaricato di aggiungere una categoria all'interno della piattaforma. | categoria != null | categoria è aggiunta alla piattaforma. |
| public void removeCategoria(int idCategoria) | Metodo incaricato di rimuovere una categoria dalla piattaforma | idCategoria != null | categoria.idCateg oria è rimossa dalla piattaforma |
| public void doUpdateCategoria(in t idCategoria, String descrizioneCategoria) | Metodo incaricato di modificare la descrizione di una categoria nella piattaforma | idCategoria != null && descrizione != null && ! descrizione.equals("") | categoria.idCateg oria è aggiornata con descrizioneCatego ria |

## Manager Carrello

| DESCRIZIONE MANAGER | | | |
|---|---|---|---|
| Questo manager si occupa di fornire un'interfaccia di controllo per la gestione del carrello di un utente della piattaforma. | | | |
| **Firma del Metodo** | **Descrizione Metodo** | **Precondizione** | **Postcondizione** |
| public Carrello doRetriveCartById Utente (int IdUtente) | Metodo incaricato di recuperare il carrello di un utente specifico | idUtente != null | Return carrello se quell' idUtente è autenticato alla piattaforma, null altrimenti . |
| public void addCart(int idProdotto, int quantitàProdotto) | Metodo incaricato di aggiungere un prodotto con la relativa quantità selezionata | doRetriveCartByIdUte nte() != null<br>&&<br>quantitàProdotto != null &&<br>quantitàProdotto >=1 &&<br>idProdotto != null | L' idProdotto viene aggiunto al carrello, altrimenti viene lanciato un messaggio di errore |

## Manager Amministratore

| DESCRIZIONE MANAGER | | | |
|---|---|---|---|
| Questo manager si occupa di fornire un'interfaccia di controllo per la gestione degli Utenti da parte di un amministratore. | | | |
| **Firma del Metodo** | **Descrizione Metodo** | **Precondizione** | **Postcondizione** |
| public void addAdmin(Utente utente) | Metodo incaricato di promuovere un utente come amministratore | utente != null | utente.admin=true |
| public void downgradeAdmin(int idUtente) | Metodo incaricato di degradare un amministratore aggiuntivo della piattaforma | idUtente != null | Il valore booleano admin di quel idUtente viene settato a false |
| public void removeUtente(int idUtente) | Metodo incaricato di rimuovere un utente dalla piattaforma | idUtente != null | L' utente con quel idUtente viene rimosso dalla piattaforma |

# 4. GLOSSARIO

**CSS:** Cascading Style Sheet è un linguaggio di stile per pagine html.

**DBMS:** sistema software per la creazione, manipolazione e interrogazione efficiente di database.

**Design Pattern:** si tratta di una descrizione o modello logico da applicare per la risoluzione di un problema che può presentarsi in diverse situazioni durante le fasi di progettazione e sviluppo software.

**HTML:** Html è un linguaggio di mark-up per pagine web.

**JSON:** Json è un formato adatto per lo scambio dei dati in applicazioni Client-Server

**Jsp:** Una pagina jsp è un documento di testo, scritto con una sintassi specifica, che rappresenta una pagina Web di contenuto parzialmente o totalmente dinamico.

**Package:** Un package è un meccanismo per organizzare classi all'interno di sottogruppi ordinati.

**Piattaforma:** Definisce l'insieme delle funzionalità fornite dal sistema attraverso l'applicazione web.

**RAD (Requirement Analysis Document):** documento contenente informazioni inerenti al sistema da realizzare raccolte durante la fase di Requirement Analysis e Requirement Elicitation.

**SDD (System Design Document):** Documento formalizzato alla definizione di obiettivi di progettazione del sistema, decomposizione del sistema in sottosistemi più piccoli e scelta di architettura software più adatta al sistema.

**Servlet:** oggetti Java all'interno del server web che permettono di creare web applications in combinazione con JSP.

**Sistema:** Definisce la totalità delle componenti dal punto di vista strutturale/ implementativo.

**Web Browser:** applicazione software installata sul client che permette di visualizzare e navigare le risorse del web.