Object Design Document (ODD)

“Gioco Mania”

# 1- Introduzione

# 1.1- Obiettivi Design Trade-offs

Dopo la realizzazione del documento RAD (Requirement Analysis Document) e SDD (System Design Document), abbiamo descritto in linea di massima, quello che sarà il nostro sistema e quindi i nostri obiettivi, tralasciando gli aspetti implementativi.

Prestazioni VS Costi

Considerando il sistema che stiamo realizzando, possiamo dire che il non eccessivo *budget* a nostra disposizione ci ha consentito di realizzare il prodotto utilizzando materiale *open source* partendo da zero minimizzando così i costi e rendendo il sistema più che soddisfacente.

Interfaccia vs. Tempo di risposta

Il tempo di risposta tra server e interfaccia sono più che sufficienti (rapidi) a soddisfare le esigenze dei vari utenti collegati al Sistema.

Ovviamente maggiore sarà la grandezza del database e maggiore sarà il tempo di risposta e ricerca nel database.

Interfaccia vs. Easy-use

L’interfaccia, grazie all’utilizzo delle form e di una impostazione semplice e intuitiva, permette un uso facile (Easy-Use) della gestione del sistema di database prodotti anche considerevolmente grande così da rendere immediata l’attività anche ai meno esperti col computer.

Costi vs. Mantenimento

Grazie a un uso di materiale open source e l’utilizzo del linguaggio javadoc il sistema può essere facilmente modificato, implementato con nuove funzioni o corretto in presenza di errori.

# 1.2- Linee guida per l’interfaccia

Il sistema è multi-utente (può accedervi chiunque, sia un semplice utente(cliente) e/o utente registrato, con il login).

Al semplice utente, il sistema nasconde la logica delle operazioni, fornendogli solamente la consultazione dei giochi e news con informazioni sui punti vendita più vicini al suo domicilio e proponendo un’eventuale iscrizione al sito.

L’utente registrato invece deve avere accesso in ricerca al DataBase totale dei prodotti dell’azienda e solo al suo DataBase storico vendite (esclusi i DataBase storico vendite degli altri utenti registrati).

L’utente registrato riceve i risultati delle richieste effettuate dal sistema, senza preoccuparsi delle operazioni e di come queste sono state effettuate.

Avvenuta l’operazione di vendita di uno o più prodotti il sistema decrementa dal DataBase totale il numero delle quantità vendute del prodotto acquistato (aggiorna quindi il DataBase totale) e riceve poi dal registrato le modalità di pagamento e tipo di consegna da parte del cliente.

Infine il prodotto venduto viene aggiunto al DataBase storico vendite del rispettivo utente.

# 1.3- Definizioni, acronimi e abbreviazioni

Acronimi :

RAD: Requirements Analysis Document

SDD: System Design Document ODD: Object Design Document

DB: DataBase

DBMS: DataBase Management System

Definizioni:

**News:**

Giochi prenotati dal committente del sito e per i quali si attende l’arrivo in magazzino (sede principale) nei prossimi giorni.

Grazie a questa entità l’utente registrato può richiedere una prenotazione di questi giochi.

News (offerte):

Tra le news sono presenti anche delle offerte inserite tra un nuovo giovo ed un altro appositamente per scelta di business: un utente registrato che desidera acquistare un nuovo gioco (uscito nel 2018) sarà interessato anche ad eventuali offerte sullo stesso tipo di video-game ma di una versione retrodatata.

**Video-game (giochi):**

Entità che permette la “virtualizzazione” (sul sistema software) della merce che si intende vendere.

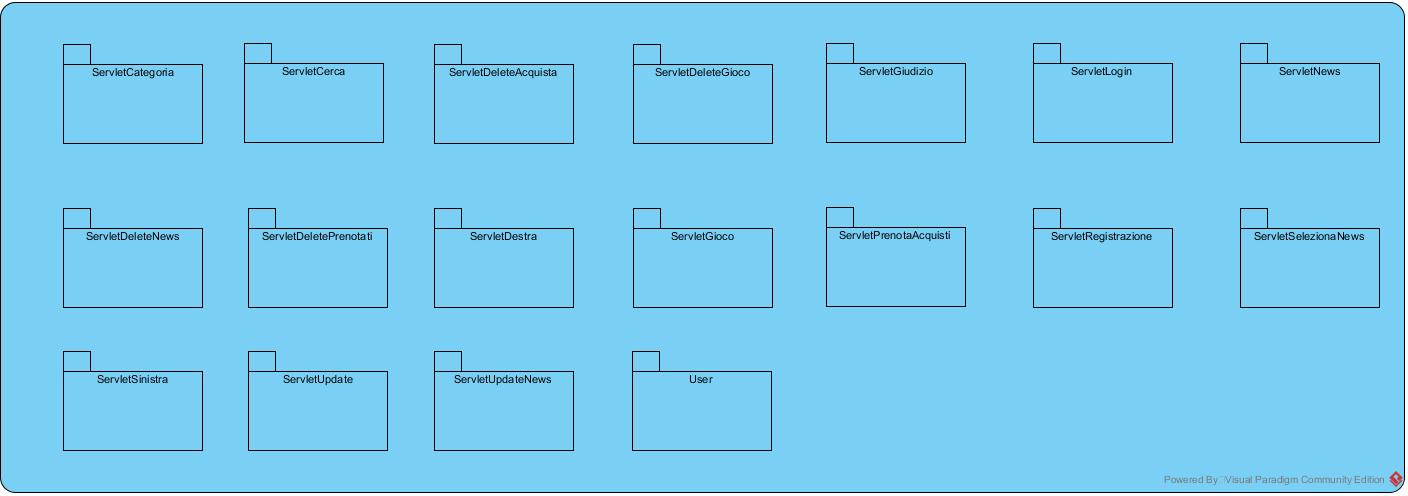
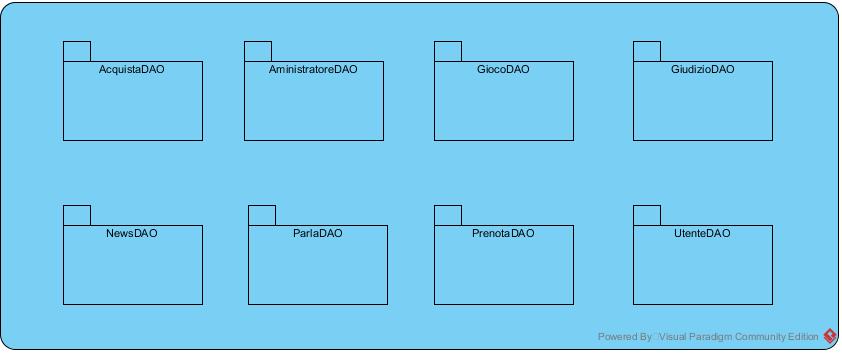
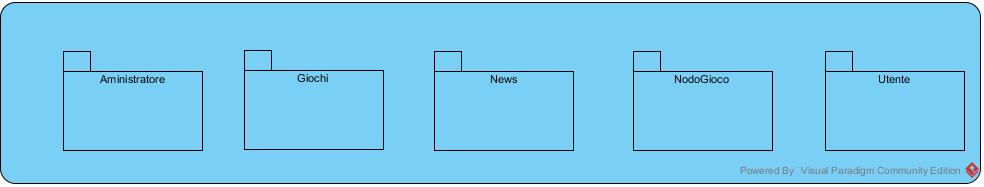
**Utente registrato:**

Persona o entità che concede i suoi dati (ha eseguito la registrazione) ai fini di acquistare, prenotare, commentare e loggarsi in futuro sul sito “GiocoMania”.

# 1.4- Riferimenti

1. Bernd Bruegge & Allen H. Dutoit, Object-Oriented Software Engineering: Using UML, Patterns and Java.
2. -Bernd POO\_Software\_Engineering\_Using\_U

# 2 Package



# Class Interface Glossary

|  |  |
| --- | --- |
| METODO | DEFINIZIONE |
| AcquistaDAO.Insert(Utente x, Gioco y) | Consente di acquisire un acquisto nel d.b. di conseguenza i dati del nostro utente saranno salvati nel carrello e all’occorrenza visualizzati. Ogni volta che il nostro utente fa un nuovo accesso se non ha ancora confermato l’acquisto e svuotato il carrello non perderà i dati. |
| AministratoreDAO.Select(String username, String password) | Consente di riconoscere un amministratore e quindi permette di distinguere questo da un utente registrato.  Ogni volta che l’amministratore si interfaccia al sistema software accede alla piattaforma inserendo username e password come un normale utente già registrato. Sarà poi compito del sistema identificare costui per fargli comparire tutte le funzionalità e le opzioni di un amministratore |
| GiocoDAO.Insert(Int id, Sting nome, String urlVideo, String categoria, String piattaforma, String descrizione, int prezzo) | Consente ad un amministratore di inserire i nuovi giochi |
| GiudizioDAO.Insert(Utente x, Gioco y, String commento) | Consente di inserire un giudizio (in questo caso nel nostro sistema software un commento inerente un particolare gioco scelto dal nostro utente registrato).  Il metodo crea un oggetto persistente Giudizio inserendo l’identificatore dell’utente, l’identificatore del gioco e il rispettivo commento |
| NewsDAO.Insert(String testo, int id) | Consente di inserire una News dove per news nel sistema GiocoMania si intende o una nuova offerta: quindi un gioco scontato; oppure un gioco uscito nel mese in cui avvengono i fatti o addirittura uscirà nei prossimi mesi: in questo caso l’amministratore può decidere di effettuare un ordine e nel mentre può già dare la possibilità ad un utente di prenotare il gioco i cui arrivi si prospettano nei prossimi giorni. |
| ParlaDAO.SelectGiochi(int idNews) | Consente di selezionare i giochi che compaiono nella news a cui fa riferimento l’idNews (identificatore della news sulla quale ha cliccato l’utente)  Il metodo restituisce una lista di giochi |
| UtenteDAO.Insert(String nome, String cognome, String indirizzo, String codFiscale, String username, String password) | Consente di inserire un utente registrato.  Questo metodo è importante per il sistema software in quanto viene chiamato ogni volta che un utente si registra. La registrazione avviene compilando dei campi che permettono al navigatore del sistema di inserire le credenziali che sono passate come parametri al metodo |
| PrenotaDAO.Insert(Utente x, Gioco y) | Consente di acquisire unaprenotazione nel d.b. di conseguenza i dati del nostro utente saranno salvati nel carrello e all’occorrenza visualizzati. Ogni volta che il nostro utente fa un nuovo accesso se non ha ancora confermato l’acquisto e svuotato il carrello non perderà i dati. |
| ServletCategoria.doGet(Response x, Request y) | Consente di selezionare la categoria cliccata dall’utente |
| ServletCerca.doGet(Response x, Request y) | Consente di ricercare i giochi che hanno come nome quello scelto dall’utente. Questo metodo viene invocato ogni volta che un utente usa la barra di ricerca. Può restituire un gioco o una lista di giochi. Ovviamente essendo una servlet il risultato di ritorno sarà presente nel bean |
| ServletDeleteAcquisto.doGet(Response x, Request y) | Consente di cancellare un acquisto di un utente che decide di annullare questa operazione per un qualsiasi motivo |
| ServletDeleteGioco.doGet(Response x, Request y) | Consente ad un amministratore di sistema di cancellare un gioco. Questa operazione si rivela necessaria in caso di esaurimento scorte in magazzino inerenti il gioco che sarà passato come parametro dal nostro bean |
| ServletGiudizio.doGet(Response x, Request y) | Consente di caricare i commenti relativi ad un certo gioco nel momento in cui l’utente clicca sulla pagina di informazioni del gioco stesso |
| ServletLogin.doGet(Response x, Request y) | Consente di fare il login dopo aver inserito username e password |
| ServletNews.doGet(Response x, Request y) | Chiama il metodo News.insert e aggiunge la news al bean |
| ServletDeleteNews(Response x, Request y) | Permette di eliminare una News. Tale operazione può essere svolta soltanto da un amministratore |
| ServletDeletePrenotati(Response x, Request y) | Elimina una prenotazione di un utente rispetto ad un gioco |
| ServletDestra(Response x, Request y) | Permette ad un amministratore di scorrere i giochi o le news tramite delle frecce: destra e sinistra. La classe ServletDestra sposta l’arrey di nodogiochi del bean a destra di uno (esegue uno scift) |
| ServletSinistra(Response x, Request y) | Permette ad un amministratore di scorrere i giochi o le news tramite delle frecce: destra e sinistra. La classe ServletSinistra sposta l’arrey di nodogiochi del bean a sinistra di uno (esegue uno scift) |
| ServletGioco(Response x, Request y) | Rende un gioco persistente (chiama giochi.Insert(nome,urlvideo,id,categoria,piattaforma,descrizione e prezzo) |
| ServletPrenotaAcquista(Response x, Request y) | Genera i voliri persistenti Acquista e prenota distinguendoli a seconda del gioco (se è possibile soltanto prenotare questo gioco inserisce nel database un oggetto prenota. Altrimenti un oggetto acquista. |
| ServletRegistrazione(Response x, Request y) | Chiama UtenteDAO.insert(nome,cognome,indirizzo,codiceFiscale,username,password) e crea poi la sessione dell’utente inserendo l’oggetto utente appena creato nel bean |
| ServletSelezionaNews(Response x, Request y) | Permette di selezionare una News quando l’utente ne fa richiesta di visualizzazione |
| ServletUpdate(Response x, Request y) | Permette ad un utente registrato, ma anche ad un amministratore di modificare le sue credenziali (quelle che ha usato per la registrazione) |
| ServletUpdateNews(Response x, Request y) | Permette di modificare una news. Questa operazione può essere fatta soltanto da un amministratore |
| User | È il nostro bean. |

# Package Manager

# Utente

Utente è una classe manager in quanto senza utente le altre classi non avrebbero senso poiché non potrei generare legami tra entità se questa non esistesse

# UtenteDao

Utente è una classe manager in quanto senza utente le altre classi non avrebbero senso poiché non potrei generare legami tra entità se questa non esistesse.

UtenteDAO è la classe che mi permette di generare l’unità persistente Utente.

# User

User nel package connessioni è di sicuro una classe manager poiché esso è il bean.

Il bean permette di far interagire view e control

# Design Pattern

# Pattern Creazionali

|  |  |
| --- | --- |
| Nome | Classifica News |
| Problema | Devo capire la News che inserisco a che gioco si riferisce. Una News può comprendere più giochi e un gioco può stare in più news |
| Soluzione | Creo l’entità parla e la istanzio ogni volta che creo una news |

# Pattern Comportamentali

|  |  |
| --- | --- |
| Nome | Acquista o Prenota |
| Problema | Per capire o meno se un gioco è una News o no (può essere solo prenotato oppure può essere acquistato) devo fare più letture nel database. Ciò potrebbe aumentare di molto il tempo necesario per un’operazione: nel caso ci sia una grande mole di istanze del d.b. |
| Soluzione | Creo una nuova entità con gli stessi valori della classe giochi ed un parametro in più di tipo booleano |

# Dictionary

|  |  |
| --- | --- |
| PAROLA | DESCRIZIONE |
| Amministratore | Proprietario, gestore ed organizzatore del sistema condominiale virtuale e reale. |
| Autenticazione | Effettuare il login al sistema, attraverso l’inserimento di username e password. |
| Boundary | Rappresentano oggetti che si interfacciano con gli attori del sistema. |
| Design pattern | I design pattern [orientati agli oggetti](https://it.wikipedia.org/wiki/Object_oriented) tipicamente mostrano relazioni ed interazioni tra [classi](https://it.wikipedia.org/wiki/Classe_(informatica)) o [oggetti](https://it.wikipedia.org/wiki/Oggetto_(informatica)), senza specificare le classi applicative finali coinvolte, risiedendo quindi nel dominio dei [moduli](https://it.wikipedia.org/w/index.php?title=Modulo_(software)&action=edit&redlink=1) e delle interconnessioni |
| Pattern creazionali | I pattern creazionali risolvono problematiche inerenti l'istanziazione degli oggetti |
| Pattern strutturali | I pattern strutturali risolvono problematiche inerenti la struttura delle classi e degli oggetti |
| Pattern comportamentali | I pattern comportamentali forniscono soluzione alle più comuni tipologie di interazione tra gli oggetti. |
| Controll | Sono oggetti che mediano tra i Boundary ed Entity, tali gestiscono l’esecuzione dei comandi proveniente dal Boundary. |
| Package | Un [package](https://it.wikipedia.org/wiki/Package_(Java)) è una collezione di [classi](https://it.wikipedia.org/wiki/Classe_(informatica)) e interfacce correlate |
| Database | Base di dati (DB), indica un archivio di dati o un insieme di archivi, in cui le informazioni in esso contenute sono strutturate e collegate tra loro secondo un particolare modello logico e in modo tale da consentire la gestione e/o organizzazione efficiente degli stessi dati e l’interfacciamento con le richieste dell’utente attraverso i cosiddetti “query language”. |
| Download | Scaricare, sul proprio dispositivo, dall’applicativo web la documentazione in formato digitale. |
| Entity | Sono oggetti che rappresentano i dati del sistemi, ovvero i dati presenti all’interno della base di dati che si relaziona con l’applicativo proposto. |
| Piattaforma | Base software e/o hardware sulla quale vengono sviluppate e/o eseguite applicazioni. |
| Vincolo | Tipo di Requisito definito anche “pseudo requisito”, sono condizioni imposte dal cliente o dall’ambiente in cui opera il sistema. |