 **Università Degli Studi Di Salerno**



Progetto di Ingegneria del software 2018/2019

System Design Document

Immagine che contiene oggetto

Descrizione generata automaticamente

Sommario

[Componenti del team di progetto 2](#_Toc534037084)

[1.Introduzione 2](#_Toc534037085)

[1.1 Scopo del sistema 2](#_Toc534037086)

[1.2 Design goals 2](#_Toc534037087)

[1.2.1 DG\_1: Performance Criteria 3](#_Toc534037088)

[1.2.2 DG\_2: Dependability criteria 3](#_Toc534037089)

[1.2.3 DG\_3: Maintenance criteria 3](#_Toc534037090)

[1.2.4 DG\_4: End-user criteria 4](#_Toc534037091)

[1.3 Definizioni, acronimi e abbreviazioni 4](#_Toc534037092)

[1.4 Riferimenti 4](#_Toc534037093)

[1.5 Overview 4](#_Toc534037094)

[2. Architettura Software Corrente 5](#_Toc534037095)

[3. Architettura Software Proposta 5](#_Toc534037096)

[3.1 Overview 6](#_Toc534037097)

[3.2 Decomposizione in sottosistemi 7](#_Toc534037098)

[3.3 Mapping Hardware\Software 7](#_Toc534037099)

[3.4 Gestione dei dati persistenti 7](#_Toc534037100)

[3.5 Controllo degli accessi e sicurezza 8](#_Toc534037101)

[3.5.1 Matrice di accesso 9](#_Toc534037102)

[3.5.2 Sicurezza 10](#_Toc534037103)

[3.6 Flusso di controllo globale 11](#_Toc534037104)

[3.7 Boundary conditions 11](#_Toc534037105)

[4. Servizi dei sottosistemi 11](#_Toc534037106)

[Glossario 11](#_Toc534037107)

# Componenti del team di progetto

|  |  |
| --- | --- |
| Partecipanti | Matricola |
| Cosimo Bacco | **0512104516** |
| Michele Castellaneta | **0512104804** |
| Domenico Trotta | **0512104882** |

# 1.Introduzione

Questo documento descrive gli obiettivi di design del progetto. Illustra la scomposizione del sistema in sottosistemi e definisce le strategie adottate per il loro sviluppo.

## 1.1 Scopo del sistema

L'obiettivo del progetto è quello di realizzare un portale web che permetta ai clienti di acquistare videogiochi per diverse piattaforme (PS4, PS3, Xbox One, Nintendo Switch, …). Lo sviluppo di un sito web di questo tipo permetterebbe di ottimizzare i tempi di servizio e di aggiornare i clienti sullo stato degli ordini da essi effettuati. Tale sito web dovrebbe consentire una comunicazione più immediata ed efficace tra gli altri attori del sistema (visitatore, gestore degli ordini, gestore del catalogo, cliente). ~~In definitiva GamesHub nasce per ottimizzare il lavoro del personale e velocizzare le preferenze dei clienti attraverso un portale web efficace ed interattivo.~~

Il sistema deve permettere:

1. la gestione degli account
2. la gestione degli ordini
3. la gestione del carrello
4. la gestione del catalogo

## 1.2 Design goals

Il sistema si avvale di una struttura grafica chiara e completa, con pulsanti, finestre di dialogo, icone, form per l'immissione dei dati e finestre scorrevoli. Le informazioni presentate sullo schermo saranno in grado di indirizzare l’utente verso le funzionalità a cui desidera accedere, cercando di volta in volta di isolare soltanto le informazioni necessarie per l’esecuzione della funzione

richiesta. L’utente non dovrà necessariamente effettuare operazioni che richiedono una discreta conoscenza dell’applicazione, quindi l’utilizzo del sistema sarà guidato dall’interfaccia semplice e intuitiva. Il sistema GamesHub ha come struttura centrale un database, il quale sarà periodicamente aggiornato per garantire il corretto funzionamento del sistema stesso. Il sistema proposto cerca di rispettare tutti i criteri di design sotto elencati.

### 1.2.1 DG\_1: Performance Criteria

Il sistema dovrebbe essere usabile e leggero, cosicché, nel caso di utilizzo in contemporanea da parte di più utenti, il sistema non risulti rallentato.

GamesHub si propone di rispettare i seguenti requisiti di qualità (rispetto alle prestazioni):

• ***DG\_1.1 Tempo di risposta*** : le risposte dovrebbero essere date in un tempoaccettabile a seguito dell’elaborazione dell’input.

• ***DG\_1.2 Throughput*** : il sistema dovrebbe completare il maggior numero dioperazioni nel minor tempo possibile, per garantire una maggiore interattivitàcon i vari utenti connessi.

• ***DG\_1.3 Memoria*** : il sistema necessita di una quantità di memoriadipendente da tutti i dati che saranno memorizzati all’interno dellaWeb‐Application realizzata.

### 1.2.2 DG\_2: Dependability criteria

Il sistema dovrebbe garantire il corretto svolgimento delle proprie funzioni, gestendo i vari errori logici (quelli derivanti da una negligenza da parte dell’utente), che potranno verificarsi durante l’utilizzo, ed eventuali attacchi alla sicurezza.

GamesHub si propone, quindi, di rispettare i seguenti requisiti di qualità, relativi all’affidabilità:

● ***DG\_2.1 Robustezza***: GamesHub dovrebbe offrire un buon grado dirobustezza agli input invalidi forniti dagli utenti. Non verranno alterati idati contenuti nel database: nel caso in cui l’utente sottometta dati erratial sistema, questo lancerà un messaggio d’errore per avvisare lo stessoutente che i dati inseriti sono invalidi;

● ***DG\_2.2 Affidabilità*** : GamesHub dovrebbe garantire il correttosvolgimento delle proprie funzionalità, producendo unicamente l’outputatteso;

● ***DG\_2.3 Disponibilità*** : il software dovrebbe essere sempre disponibile efunzionante, tranne in eventuali periodi di manutenzione;

● ***DG\_2.4 Sicurezza*** : Ogni utente potrà accedere con una login e passwordpersonale; l’accesso al sistema sarà controllato da un apposito sistema diautenticazione, che permetterà ad ogni categoria di utente di eseguire ilproprio lavoro senza intaccare o modificare quello altrui.

### 1.2.3 DG\_3: Maintenance criteria

Il sistema dovrebbe garantire un’alta manutenibilità.

GamesHub si propone, quindi, di rispettare i seguenti requisiti di qualità:

• ***DG\_3.1 Estendibilità*** : il sistema dovrebbe essere realizzato in modo da poter garantirel’inserimento di nuove funzionalità in maniera semplice, senza doverne modificare altre.

• ***DG\_3.2 Modificabilità*** : il sistema dovrebbe essere realizzato in modo da potergarantire la modifica di funzionalità già presenti all’interno del sistema, senzadoverne apportare altre, quindi il grado di entropia del sistema deve esseremantenuto il più basso possibile.

• ***DG\_3.3 Tracciabilità dei requisiti*** : tramite una buona documentazione saràpossibile risalire ai corrispettivi requisiti funzionali, cui faranno riferimento levarie classi e i metodi.

### 1.2.4 DG\_4: End-user criteria

Per quanto riguarda gli utenti, GamesHub si propone di garantire i seguenti requisiti di qualità:

• ***DG\_4.1 Utilità*** : grazie ai requisiti funzionali raccolti, GamesHub dovrebbe supportare in pieno le esigenze delle varie tipologie di utenti.

• ***DG\_4.2 Usabilità*** : il sistema dovrebbe essere semplice ed intuitivo e dopo unbreve utilizzo dovrà consentire all’utente di compiere le operazioni nel minortempo possibile. Inoltre, dovrebbero essere rispettati i requisiti non funzionali di usabilità del sistema (NFR\_5, NFR\_6 di GamesHub\_RAD)

## 1.3 Definizioni, acronimi e abbreviazioni

Vengono di seguito esplicitati definizioni, acronimi e abbreviazioni che verranno

incontrati all’interno del documento.

|  |  |
| --- | --- |
| Acronimo | Descrizione |
| RAD | Requirement Analysis Document |
| GUI | Graphical User Interface |
| SW | Software |
| HW | Hardware |
| SQL | Structured Query Language |
| SDD | System Design Document |
| DBMS | Database Management System |

## 1.4 Riferimenti

1. GamesHub\_RAD

## 1.5 Overview

Per garantire una migliore leggibilità, il System Design Document è stato diviso in tre sezioni:

1. **Introduzione**: viene riporta una descrizione del sistema specificando il motivo per cui è stato sviluppato, le sue caratteristiche e un accenno sull’utilizzo delle sue funzionalità.
2. **Architettura Software proposta**: viene descritta l’architettura usata nel sistema, ed in particolare:

* **Decomposizione in sottosistemi**, nella quale è descritta la suddivisione del sistema in vari sottosistemi.
* **Hardware/software mapping**, in cui sono prese alcune decisioni sulle piattaforme hardware e software su cui il sistema dovrà girare e il mapping delle componenti su di esse.
* **Gestione dei dati persistenti**, in cui sono identificati gli oggetti persistenti e viene scelto il tipo di infrastruttura da usare per memorizzarli.
* **Controllo degli accessi e sicurezza**, che descrive il modello utente del sistema in termini di matrici di accesso e i problemi di sicurezza, come la scelta di un meccanismo di autenticazione.
* **Flusso di controllo globale**, esterno (tra client e server) ed interno, che descrive come il controllo globale del software è implementato, come le procedure di richiesta sono avviate e come si sincronizzano i sottosistemi.
* **Boundary Condition**, in cui sono descritte le condizioni limite del sistema.

1. **Servizi dei Sottosistemi**, dove viene riportata una descrizione dei sottosistemi individuati e i servizi offerti da ogni sottosistema in termini di operazioni.

# 2. Architettura Software Corrente

Il sistema proposto non sostituirà nessun sistema già esistente.

# 3. Architettura Software Proposta

L’architettura scelta per il sistema GamesHub è quella Three‐Tier. L'espressione architettura Three‐Tier ("a tre strati") indica una particolare architettura software di tipo multi‐tier per l'esecuzione di un'applicazione web. Essa prevede la suddivisione dell'applicazione in tre diversi moduli o strati

dedicati rispettivamente all’interfaccia utente, alla logica funzionale e alla gestione dei dati persistenti. Tale architettura va tipicamente a mappare a livello fisico‐infrastrutturale quella del sistema informatico ospitante l'applicazione da eseguire. Tali moduli interagiscono fra loro secondo le linee generali del paradigma Client-Server e utilizzando interfacce ben definite. In questo modo, ciascuno dei tre moduli può essere modificato o sostituito indipendentemente dagli altri, conferendo scalabilità e manutenibilità all'applicazione.



## 3.1 Overview

Il sistema che s’intende realizzare è, quindi, un sistema distribuito e gli utenti di GamesHub potranno interagire con esso tramite i propri terminali (client). I client, per realizzare le funzionalità richieste, dovranno comunicare con l’application server (che si occupa della logica di controllo), il quale, a sua volta, comunica con il database server, dove sono contenute tutte le informazioni.

Sul database server, risiede un DBMS che si occupa di recuperare, memorizzare ed interrogare i dati presenti nel database, elaborando, quindi, la richiesta degli utenti, inviata sottoforma di query da parte dell’application server. L’aspetto della concorrenza di accessi multipli al database sarà pertanto gestito dal DBMS stesso. Tale architettura conferisce all’intero sistema una maggiore manutenibilità e permette di gestire il problema della concorrenza degli accessi ai dati in

maniera semplice ed efficace. Inoltre, grazie alla natura atomica delle transazioni col database, non si possono verificare problemi d’incoerenza nei dati (a meno di guasti ai supporti di memorizzazione secondari).

## 3.2 Decomposizione in sottosistemi

Per rendere il sistema più facile da progettare e aumentare il requisito di manutenibilità è stato deciso di decomporlo in sottosistemi, al fine di rendere la progettazione il più semplice possibile. L’architettura Three‐Tier organizza i sottosistemi in tre strati: Interface Layer, ovvero tutti gli oggetti Boundary che interagiscono con l’utente. Application Logic Layer, ovvero tutti gli oggetti che riguardano il controllo e le entità del sistema, e Storage Layer, la quale funzione è quella di memorizzare, recuperare ed interrogare gli oggetti persistenti (i dati che sono presenti nei DB).

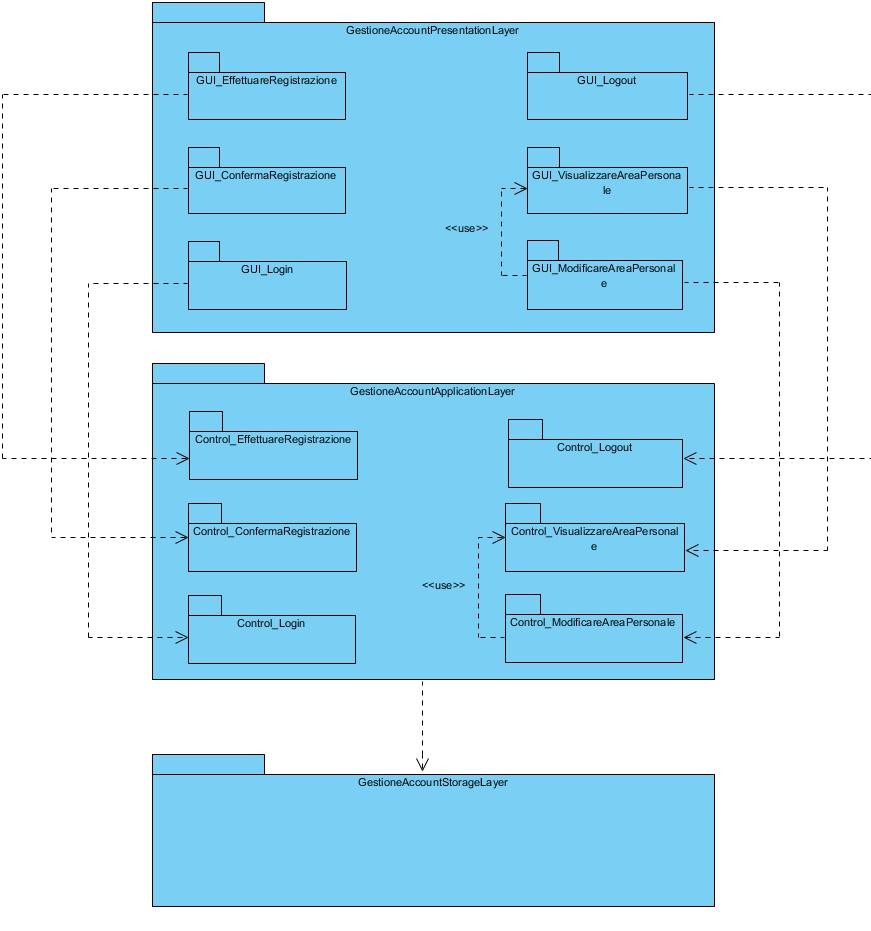
I tre livelli relativi all’architettura adottata per Sandwich on web, quindi, saranno:

• GamesHubPresentationLayer

• GamesHubApplicationLayer

• GamesHubStorageLayer

### Sottosistema 0: Gestione Utente



GestioneAccountPresentationLayer

Include tutte le componenti dell’interfaccia grafica del sistema che offrono le funzionalità per la gestione dell’account. È riportata di seguito una breve descrizione di ognuna di esse:

* *GUI\_EffettuareRegistrazione*: è un’interfaccia tramite la quale il visitatore si registra nel sistema.
* *GUI\_ConfermaRegistrazione*: è un’interfaccia tramite la quale il visitatore conferma la registrazione.
* *GUI\_Login*: è un’interfaccia tramite la quale visitatore effettua l’accesso nel sistema.
* *GUI\_Logout*: è un’interfaccia tramite la quale l’utente registrato esce dal sistema.
* *GUI\_VisualizzareAreaPersonale*: è un’interfaccia tramite la quale il cliente accede all’area personale.
* *GUI\_ModificareAreaPersonale: è un’interfaccia tramite la quale il cliente modifica i propri dati nell’area personale.*

GestioneAccountApplicationLayer

Comprende tutte le componenti logiche, responsabili del corretto funzionamento del sistema, e fa da tramite con il database per aggiornare i dati relativi alla gestione account.

È riportata di seguito una breve descrizione di ognuno di esse:

* Control\_Effettuare*Registrazione*: si occupa di eseguire le operazioni necessarie alla registrazione di un visitatore del sistema.
* *Control\_ConfermareRegistrazione*: si occupa di eseguire le operazione necessarie registrare il visitatore nel sistema.
* *Control\_Login*: si occupa di eseguire le operazione necessarie all’autenticazione degli attori del sistema.
* *Control\_Logout:* si occupa di eseguire le operazione necessarie al logout degli attori del sistema.
* *Control\_VisualizzareAreaPersonale:* si occupa di eseguire le operazioni necessarie per far accedere l’utente registrato alla sua area personale.
* *Control\_ModificareAreaPersonale*: si occupa di eseguire le operazione necessarie alla modifica dei dati di un utente registrato del sistema.

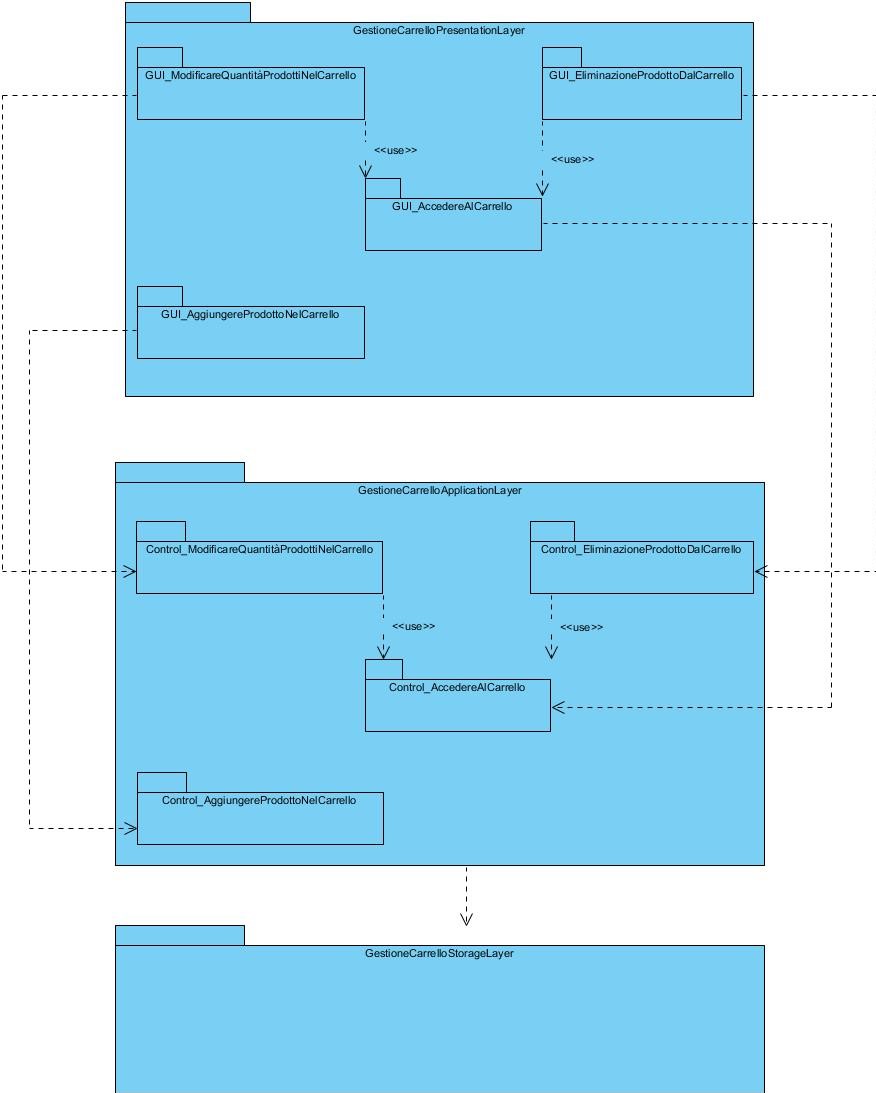
GestioneAccountDataLayer

Sottosistema che si occupa di rendere reperibili i dati, presenti all’interno del

database, relativi alla gestione dell’account. Questo livello consente agli attori

del sistema di effettuare l’autenticazione.

### Sottosistema 1: Gestione Carrello



GestioneCarrelloPresentationLayer

Include tutte le componenti dell’interfaccia grafica del sistema che offrono le

funzionalità per la gestione del carrello.

È riportata di seguito una breve descrizione di ognuna di esse:

* *GUI\_AccedereAlCarrello*: è un’interfaccia tramite la quale il visitatore del sistema può accedere a tutte le operazioni legate alla visualizzazione del carrello.
* *GUI\_AggiungereProdottoNelCarrello*: è un’interfaccia tramite la quale il visitatore può inserire un prodotto nel carrello.
* *GUI\_EliminazioneProdottoDalCarrello*: è un’interfaccia tramite la quale visitatore del sistema può accedere a tutte le operazioni legate alla rimozione di un prodotto dal carrello.
* GUI\_ModificareQuantitàProdottoNelCarrello: è un’interfaccia tramite la quale il visitatore del sistema può modificare la quantità di un prodotto dal carrello.

GestioneCarrelloApplicationLayer

Comprende tutte le componenti logiche, responsabili del corretto

funzionamento del sistema, e fa da tramite con il database per aggiornare i dati

relativi alla gestione del carrello.

È riportata di seguito una breve descrizione di ognuno di esse:

* *Control\_AccedereAlCarrello*: occupa di eseguire le operazioni necessarie alla

visualizzazione del carrello.

* *Control\_AggiungereProdottoNelCarrello*: si occupa di eseguire le operazioni necessarie all’aggiunta di un prodotto nel carrello.
* *Control\_EliminazioneProdottoDalCarrello*: si occupa di eseguire le operazioni necessarie

alla rimozione di un prodotto dal carrello.

* *Control\_ModificareQuantitàProdottoNelCarrello*: si occupa di eseguire le operazioni necessarie per modificare la quantità di un prodotto nel carrello.

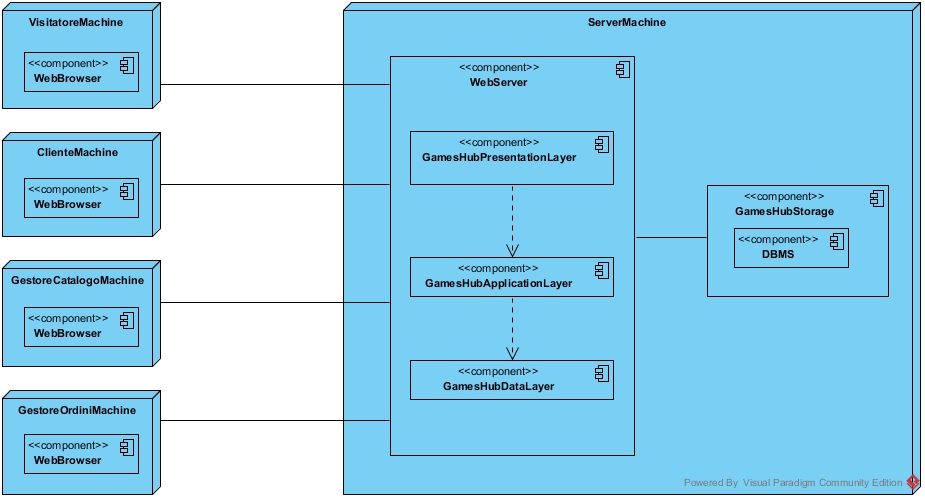
GestioneCarrelloStorageLayer

Sottosistema che si occupa di rendere reperibili i dati, presenti all’interno del

database, relativi alla gestione del carrello.

## 3.3 Mapping Hardware\Software

Per quanto riguarda il flusso d’informazioni, l’architettura è stata scomposta in due livelli, un livello CLIENT ed un livello SERVER. Rappresentiamo nel seguito la distribuzione delle componenti Hardware e Software sui due nodi.



## 3.4 Gestione dei dati persistenti

Il problema della persistenza dei dati nasce dalla necessità di rendere permanenti alcune informazioni all’interno di un sistema anche quando questo è spento. Molte volte capita di avere dei programmi con un gran numero di dati da gestire, dati che non possono essere reinseriti dopo un blackout o dopo lo spegnimento della macchina; si pensi ad un archivio di mille utenti contenente tutti i dati anagrafici. Sarebbe impensabile il reinserimento degli stessi ogni volta che la macchina sul quale risiedono i dati si spegne. Bisogna, quindi, trovare il modo per poter tener traccia dei dati utili anche quando l’applicazione che li ha utilizzati e/o creati ha

smesso di funzionare. I dati persistenti sono sempre lì e vengono cancellati solo se lo si richiede espressamente. Nel caso specifico, in previsione del gran numero di accessi al database e alla necessità di gestire anche più query contemporaneamente, si è optati per un DBMS relazionale, MySQL. La sigla SQL presente all'interno del nome sta a indicare che si tratta di un DBMS in grado di interpretare le istruzioni del linguaggio SQL (utile per le query). MySQL permette la creazione di “database relazionali” ossia consente la conservazione dei dati in tabelle separate anziché in un'unica grande entità. Questa sua particolare caratteristica consente di raggiungere una buona flessibilità e velocità di accesso ai dati ed una maggior modellazione delle basi dati. Semplicità d'uso, robustezza e velocità di esecuzione sono le caratteristiche principali di questo DBMS.

La sua struttura multi‐thread consta di un thread fisso che controlla le connessioni in ingresso e un thread attivo per ogni connessione. I vari client non devono aspettare che le query di altri siano soddisfatte, possono lavorare simultaneamente. Il thread che controlla le connessioni

impedisce che due thread scrivano sulla stessa tabella nello stesso momento. In ogni tabella all’interno del nostro database sarà possibile accedere ai dati per compiere operazioni quali inserimento, modifica, cancellazione o semplicemente consultazione.

## 3.5 Controllo degli accessi e sicurezza

GamesHub sarà realizzato in modo da interagire con diverse tipologie di utenti: visitatore, cliente, gestore degli ordini, gestore del catalogo.

Ognuno di essi potrà accedere esclusivamente ad un determinato tipo di funzionalità e dati, ovviamente previo autenticazione. I dati per effettuare quest’ultima saranno concordati nella fase di creazione dell’account, per tutti gli utenti tranne che per gli admin (Gestore degli ordini, Gestore del catalogo), i cui dati saranno impostati di default all’interno del database, non appena creato il sistema.

● Il visitatore può creare il suo account, visualizzare la lista dei giochi, ricercare un gioco, visualizzare un gioco, visualizzare il suo carrello, rimuovere un gioco dal carrello, svuotare il carrello.

● Il cliente può effettuare il login e il logout e modificare i dati del proprio account, visualizzare la lista dei giochi , ricercare un gioco, visualizzare un gioco, acquistare giochi, visualizzare il carrello, rimuovere un gioco dal carrello, svuotare il carrello, visualizzare l’ordine e annullare l’ordine, visualizzare i dettagli dell’ordine ,ricercare un ordine.

● Il Gestore degli ordini può effettuare il login e il logout del proprio account, visualizzare gli ordini e modificare lo stato dell’ordine in accettato, in preparazione o spedito, aggiungere un track-id.

● Il Gestore del catalogo può effettuare il login e il logout del proprio account, visualizzare la lista dei giochi e rimuovere un gioco dalla lista, modificare i giochi presenti nella lista e inserire un nuovo gioco.

Per rappresentare in modo schematico e permettere una lettura immediata delle operazioni consentite agli attori sulle diverse entità del nostro sistema, utilizzeremo la matrice degli accessi, qui di seguito presentata.

### 3.5.1 Matrice di accesso

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attore | Oggetti | | | | |
| Utente | Ordine | Carrello | Carta | Gioco |
| **Visitatore** | registraAccount() |  | visualizzaCarrello()  aggiungiGioco()  rimuoviGioco() |  | visualizzaGioco()  ricercaGioco()  visualizzaCatalogo() |
| **Cliente** | login()  logout()  visualizzaDati()  modificaDati() | creaOrdine()  annullaOrdine()  visualizzaOrdine() | visualizzaCarrello()  aggiungiGioco()  rimuoviGioco() | aggiungiCarta()  eliminaCarta() | visualizzaGioco()  ricercaGioco()  visualizzaCatalogo() |
| **Gestore catalogo** | login()  logout() |  |  |  | visualizzaCatalogo()  aggiungiGioco()  modificaGioco()  rimuoviGioco()  ricercaGioco() |
| **Gestore ordini** | login()  logout() | visulizzaOrdini()  modificaOrdini()  ricercaOrdine() |  |  |  |

### 3.5.2 Sicurezza

La sicurezza del sistema è garantita dall’obbligo, di tutte le utenze che lo utilizzano, di autenticarsi, infatti GamesHub chiederà come forma di autenticazione l’immissione di un’username e di una password. Tali credenziali dovranno esser inserite ogni volta che si desidera utilizzare il sistema; tale sessione terminerà quando l’utente richiederà esplicitamente di effettuare logout. Nel caso l’accesso al sistema non abbia successo, GamesHub mostrerà una schermata in cui sarà specificato se c’è stato un errore di inserimento della password, o se l’username non esiste all’interno del database, consentendo, successivamente, all’utente di inserire le credenziali per effettuare un nuovo tentativo. Nello specifico la password deve avere una lunghezza minima di 6 caratteri

alfanumerici. In seguito all’autenticazione il GamesHub mostrerà diverse “viste” dello stesso sistema a seconda dell’utente, ovvero sarà mostrata una schermata contenente le sole funzionalità a cui una determinata tipologia di può accedere.

Per quanto riguarda la registrazione di un utente al sistema, GamesHub prevede un verifica in due passaggi. Dopo che i dati inseriti sono stati validati dal sistema, l’utente riceverà una e-mail contenente un link che gli permetterà di completare la sua registrazione a GamesHub.

Inoltre, il sistema cripterà username e password di ogni utente e anche eventuali numeri delle carte inserite.

## 3.6 Flusso di controllo globale

## 3.7 Boundary conditions

# 4. Servizi dei sottosistemi

# Glossario

|  |  |
| --- | --- |
| Termini | Descrizione |
| DBMS | **Database Management System** |
| DG | **Design-goals** |
|  |  |
|  |  |