Immagine che contiene testo, logo, emblema, simbolo

Descrizione generata automaticamente

Corso di Laurea in Informatica

Progetto Ingegneria del Software

OBJECT DESIGN

VERSIONE 1.2

Immagine che contiene cerchio, Elementi grafici, simbolo, Policromia

Descrizione generata automaticamente

**GameChanger**

# Partecipanti

Isidoro D’Arco – 0512113779

Ruben Petina - 0512114733

Sommario

[Partecipanti 1](#_Toc186821673)

[1. Introduzione 3](#_Toc186821674)

[1.1 - Object design trade-off 3](#_Toc186821675)

[1.2 - Linee guida per la documentazione delle interfacce 3](#_Toc186821676)

[1.3 - Definizione, acronimi e abbreviazioni 4](#_Toc186821677)

[1.4 – Riferimenti 4](#_Toc186821678)

[2. Packages 4](#_Toc186821679)

[2.1 – Back-end Package 5](#_Toc186821680)

[2.2 – Front-end Package 8](#_Toc186821681)

[3. Class Interface 11](#_Toc186821682)

[3.1 AmministratoreDao 11](#_Toc186821683)

[3.2 CarrelloDao 11](#_Toc186821684)

[3.3 OrdineDao 12](#_Toc186821685)

[3.4 ProdottoDao 12](#_Toc186821686)

[3.5 UserDao 13](#_Toc186821687)

## Introduzione

Il documento di Analisi dei Requisiti e il System Design Document hanno fornito una visione complessiva del sistema e dei suoi obiettivi, evitando di entrare nei dettagli tecnici legati all'implementazione. Questo documento ha l’obiettivo di sviluppare un modello capace di integrare tutte le funzionalità precedentemente individuate in modo preciso e coerente.

### - Object design trade-off

* **Interfaccia vs Usabilità**  
  La progettazione della GUI si è focalizzata sul garantire un’esperienza utente semplice e intuitiva. Per raggiungere questo obiettivo, sono stati adottati colori e icone facilmente riconoscibili, insieme a un design che semplifica le interazioni con il sistema. Queste scelte hanno permesso di ottenere un'interfaccia chiara, efficace e accessibile a un ampio pubblico.
* **Sicurezza vs Efficienza**  
  Pur essendo la sicurezza un elemento prioritario, i tempi limitati di sviluppo hanno portato a concentrarsi su interventi mirati. In particolare, sono stati introdotti controlli avanzati sui campi di testo per garantire l'integrità del sistema, impedendo l'inserimento di dati potenzialmente dannosi per il codice. Questa misura riduce i rischi legati a vulnerabilità come l’injection e migliora complessivamente la protezione del sistema.
* **Prestazioni vs Costi**  
  A causa delle ristrettezze di budget, si è deciso di evitare l’utilizzo di librerie e template a pagamento. Il progetto si è quindi basato sull’adozione di soluzioni open source e sull’utilizzo di risorse hardware già disponibili.

### 1.2 - Linee guida per la documentazione delle interfacce

Gli sviluppatori devono rispettare le seguenti convenzioni per la scrittura del codice:

#### Naming Convention

La nomenclatura deve essere:

* Descrittiva
* Pronunciabile
* Di lunghezza medio-corta
* Di uso comune
* Composta solo da caratteri consentiti (a-z, A-Z, 0-9)

#### Variabili

* I nomi delle variabili devono iniziare con una lettera minuscola
* Se il nome è composto da più parole, utilizzare la Lower Camel Notation (es: nomeVariabile) oppure l’underscore (\_) come separatore
* Ogni riga deve contenere la dichiarazione di una sola variabile

#### Metodi

* I nomi dei metodi devono iniziare con una lettera minuscola, con eventuali parole successive in maiuscolo (Lower Camel Notation)
* Il nome del metodo deve iniziare con un verbo che descrive l’azione, seguito da un sostantivo
* I metodi accessori e modificatori devono seguire i pattern getNomeVariabile e setNomeVariabile

#### Classi

* I nomi delle classi devono essere scritti in Pascal Case Notation, iniziando con una lettera maiuscola e con ogni parola successiva in maiuscolo (es: ProdottoMagazzino)
* I nomi delle classi devono riflettere le informazioni e le funzionalità fornite dalla classe stessa
* La struttura delle classe deve seguire questo ordine:
  + Dichiarazione della classe pubblica
  + Dichiarazione di costanti
  + Dichiarazioni di variabili di classe
  + Dichiarazioni di variabili di istanza
  + Costruttore
  + Metodi

#### Packages

* I nomi dei packages dovranno essere scritti in minuscolo concatenando insieme diversi sostantivi o sigle, separate dal carattere “.”

### - Definizione, acronimi e abbreviazioni

**RAD** sta per Requirement Analysis Document

**SDD** sta per System Design Document

**ODD** sta per Object Design Document

**JSP** sta per Java Servlet Page

### – Riferimenti

Riferimento al RAD e SDD

## Immagine che contiene testo, schermata, Carattere Descrizione generata automaticamentePackages

Il back-end è implementato in pacchetti presenti nella directory **/java/cn/gamechanger.**

Immagine che contiene testo, schermata, menu

Descrizione generata automaticamenteIl front-end è implementato in pacchetti presenti nella directory **/WebContent/.**

### 2.1 – Back-end Package

#### 2.1.1 Servlet

|  |  |
| --- | --- |
| **Classe** | **Descrizione** |
| **AggiungiAlCarrelloServlet** | Gestisce l'aggiunta di un prodotto al carrello di un utente autenticato, reindirizzando alla pagina di login se non autenticato. |
| **CheckoutServlet** | Gestisce il processo di checkout in più fasi: verifica dati utente, salvataggio indirizzo, pagamento PayPal, e conferma ordine con svuotamento carrello. |
| **EliminaProdottoSerlvet** | Elimina un prodotto dal database utilizzando il suo codice e reindirizza l'utente alla pagina di conferma della cancellazione |
| **GetAdminServlet** | Recupera i dati di un amministratore autenticato tramite il suo ID di sessione e li visualizza sulla pagina dedicata all'amministratore. |
| **InserisciProdottoServlet** | Gestisce l'inserimento di un nuovo prodotto nel database e il caricamento di fino a tre immagini associate, salvandole in una directory specifica. |
| **LoginServlet** | Gestisce il processo di login per amministratori e utenti. Autentica le credenziali, crea sessioni, e reindirizza alla pagina amministratore o homepage utente. |
| **LogoutServlet** | Gestisce il logout degli utenti. Invalida la sessione corrente e reindirizza alla pagina di login. |
| **ModDatiServlet** | Gestisce la modifica dei dati del profilo utente, inclusi username, password, email, immagine del profilo, e altre informazioni personali, aggiornandoli nel database e reindirizzando alla pagina del profilo. |
| **IdentificaProdottoServlet** | Gestisce l’identificazione di un prodotto. Recupera il prodotto dal database tramite il codice e lo inoltra alla pagina "ModificaProdotto.jsp" per l'aggiornamento. |
| **ModProdottoServlet** | Gestisce la modifica delle informazioni di un prodotto nel database. Riceve i dettagli del prodotto, aggiorna il prodotto nel database e reindirizza l'utente alla pagina "prodottoModificato.jsp" per confermare l'aggiornamento. |
| **ModProfiloServlet** | Gestisce la richiesta di modifica del profilo utente. A seconda della categoria selezionata (ad esempio, username, password, immagine), recupera i dati dell'utente dal database e reindirizza l'utente alla pagina "modifica.jsp" per effettuare la modifica specifica. |
| **ModQuantitàCarrelloServlet** | Gestisce la modifica della quantità di un prodotto nel carrello dell'utente. Se l'utente è autenticato, aggiorna la quantità del prodotto nel carrello e reindirizza l'utente alla pagina "carrello.jsp" con la lista aggiornata degli articoli. Se l'utente non è autenticato, lo redirige alla pagina di login. |
| **MostraCarrelloServlet** | Recupera i prodotti nel carrello dell'utente autenticato e li visualizza nella pagina "carrello.jsp". |
| **MostraCatalogoServlet** | Visualizza tutti i prodotti disponibili nel catalogo, filtrati per categoria. |
| **MostraProdPrezzoServlet** | Filtra i prodotti per categoria (videogiochi, computer, accessori, console) in base al prezzo, visualizzandoli nella pagina del catalogo. |
| **MostraProdSingoloServlet** | Recupera un prodotto specifico tramite il suo codice e lo mostra nella pagina dedicata del prodotto. |
| **ProdottiAcquistatiServlet** | Recupera e mostra gli ordini e i relativi prodotti acquistati da un utente, visualizzandoli nella pagina "prodottiacquistati.jsp". |
| **RimuoviDalCarrelloServlet** | Rimuove un prodotto dal carrello di un utente e aggiorna la visualizzazione del carrello nella pagina "carrello.jsp". |
| **RecuperaProfiloServlet** | Recupera i dati del profilo utente dal database e li visualizza nella pagina "profilo.jsp". |
| **VerificaEsistenzaProdServlet** | Verifica se un prodotto esiste nel database in base al codice fornito e restituisce un valore booleano. |
| **VerificaPasswordServlet** | Verifica se la password inserita dall'utente corrisponde alla password memorizzata nel database, restituendo un valore booleano. |
| **VerificaUsernameServlet** | Controlla se un nome utente esiste già nel database, restituendo un valore booleano che indica se il nome utente è disponibile. |
| **VisualOrdiniPerDataServlet** | Recupera gli ordini effettuati in un intervallo di date specificato e li mostra, insieme ai prodotti associati, nella pagina "prodottiacquistati.jsp". |
| **VisualOrdiniPerUserServlet** | Recupera gli ordini effettuati da un utente specificato e li mostra, insieme ai prodotti associati, nella pagina "prodottiacquistati.jsp". |
| **VisualTuttiOrdiniServlet** | Recupera tutti gli ordini nel sistema, insieme ai prodotti associati, e li mostra nella pagina "prodottiacquistati.jsp". |

Immagine che contiene testo, menu, schermata

Descrizione generata automaticamente

#### Dao

|  |  |
| --- | --- |
| **Classe** | **Descrizione** |
| **AmministratoreDao** | Gestisce l'autenticazione e il recupero dei dati dell'amministratore dal database, eseguendo query per il login e il recupero delle informazioni. |
| **CarrelloDao** | Gestisce le operazioni sul carrello dell'utente, inclusi l'aggiunta, la rimozione e l'aggiornamento della quantità dei prodotti, e la visualizzazione dei contenuti. |
| **OrdineDao** | Gestisce le operazioni sugli ordini, inclusi l'aggiunta di un nuovo ordine e dei relativi prodotti, e la lettura degli ordini di un utente o di tutti gli ordini in un determinato intervallo di date. |
| **ProdottoDao** | Gestisce le operazioni sui prodotti nel database, inclusi filtraggio per categoria e prezzo, inserimento, aggiornamento e cancellazione. |
| **UserDao** | Gestisce le operazioni di accesso ai dati dell'utente, come login, registrazione, aggiornamento e recupero informazioni dal database. |

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

#### Connection

|  |  |
| --- | --- |
| **Classe** | **Descrizione** |
| **DbCon** | Classe che permette l’accesso al database attraverso i driver JDBC. |



#### Bean

|  |  |
| --- | --- |
| **Classe** | **Descrizione** |
| **Amministratore** | Classe che rappresenta un amministratore con attributi come ID, nome, cognome, password ed email, e fornisce metodi per gestirne i dati. |
| **Carrello** | Classe che rappresenta un carrello con attributi come ID, nome prodotto, quantità, prezzo e username dell'utente, con metodi per gestirne i dati. |
| **Ordine** | Classe che rappresenta un ordine con ID, username, stato e data dell'ordine, con metodi per gestire e visualizzare i dati dell'ordine. |
| **Prodotto** | Classe che rappresenta un prodotto con codice, prezzo, nome, marca, descrizione, data di uscita e categoria, con metodi per gestire e visualizzare i dati. |
| **ProdottoOrdine** | Classe che rappresenta i prodotti in un ordine, con informazioni su ID ordine, codice prodotto, nome, prezzo e quantità. |
| **User** | Classe che rappresenta un utente con dati personali, contatti, indirizzo, stato, informazioni di accesso, e un'immagine. |

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

### – Front-end Package

#### 2.2.1 WebContent

|  |  |
| --- | --- |
| **JSP** | **Descrizione** |
| **carrello.jsp** | Pagina JSP che mostra i prodotti nel carrello dell'utente, calcola il totale e consente di modificare le quantità o rimuovere articoli prima di procedere al checkout. |
| **checkout.jsp** | Pagina JSP che mostra il primo step del checkout, raccogliendo le informazioni di spedizione dell'utente. |
| **checkout2.jsp** | Pagina JSP per il secondo step del checkout, dove l'utente inserisce l'e-mail PayPal per il pagamento. |
| **checkout3.jsp** | Pagina JSP per il terzo step del checkout, che mostra il riepilogo dei prodotti nel carrello con il calcolo del totale e le informazioni di spedizione e pagamento. |
| **checkout4.jsp** | Pagina JSP che conferma l'avvenuto pagamento, mostrando un messaggio di successo con un'immagine e un link per tornare alla homepage. |
| **faq.jsp** | Pagina JSP che mostra una sezione FAQ con risposte alle domande più comuni su pagamenti, spedizioni, disponibilità prodotti, resi, garanzie e contatti del servizio clienti. |
| **homepage.jsp** | Pagina JSP per la homepage di GameChanger, che include un carosello interattivo con prodotti in evidenza, una sezione "Best Seller", nuove uscite e un'anteprima video di un gioco in arrivo. |
| **login.jsp** | Pagina JSP per il login, che include un form con campi per username e password. Inoltre, la pagina include un link per la registrazione. |
| **modificaProdotto.jsp** | La pagina JSP permette di modificare un prodotto. I dati del prodotto sono precompilati nel form per consentire all'amministratore di aggiornarli. |
| **modificaUser.jsp** | La pagina JSP consente all'utente di modificare diverse informazioni del profilo (username, password, email, ecc.). In base al tipo di modifica richiesta, mostra il relativo modulo con il valore corrente precompilato e un form per aggiornare i dati. |
| **operazioniAmministratore.jsp** | Gestisce l'inserimento, modifica, eliminazione dei prodotti e la visualizzazione degli ordini per data o utente. |
| **ordiniAdmin.jsp** | Visualizza gli ordini effettuati dall'utente con dettagli su prodotto, quantità, prezzo e stato dell'ordine. Mostra un messaggio se non ci sono ordini. |
| **paginaAmministratore.jsp** | Pagina di gestione per l'amministratore, visualizza informazioni personali e fornisce link per operazioni su prodotti e ordini. |
| **paginaProdotto.jsp** | Pagina del prodotto singolo, visualizza immagini, dettagli, prezzo, descrizione e permette l'aggiunta al carrello con opzione di selezione quantità. |
| **privacy.jsp** | Pagina informativa sulla privacy con dettagli su politiche e pratiche, presentata in un formato di domanda e risposta. |
| **prodottiAcquistati.jsp** | Pagina che visualizza gli ordini e i prodotti acquistati da un utente, con dettagli come ID ordine, quantità, prezzo e stato dell'ordine. |
| **prodottoAggiunto.jsp** | Pagina di conferma che informa l’amministratore dell'aggiunta di un prodotto al database, con un link per tornare alla pagina dell'amministratore. |
| **prodottoCancellato.jsp** | Pagina di conferma che informa l’amministratore dell'eliminazione di un prodotto al database, con un link per tornare alla pagina dell'amministratore. |
| **prodottoModificato.jsp** | Pagina di conferma che informa l’amministratore della modifica di un prodotto nel database, con un link per tornare alla pagina dell'amministratore. |
| **profilo.jsp** | Pagina del profilo utente che visualizza le informazioni personali, con opzioni per modificare dettagli come nome, username, password, immagine del profilo, e altre informazioni di contatto. |
| **register.jsp** | Pagina di registrazione utente con un modulo che raccoglie username, password, nome, cognome, data di nascita, e-mail, e consente l'accettazione dei termini e della privacy. |
| **sfogliaProdotti.jsp** | Pagina che mostra un catalogo di prodotti, con filtri per categoria e prezzo. Include una funzione di ricerca dinamica e consente di visualizzare la pagina di un prodotto cliccando su di esso. |
| **termini.jsp** | Pagina che presenta i "Termini e condizioni" del sito. |

Immagine che contiene testo, schermata, menu, libro

Descrizione generata automaticamente

#### 2.2.2 WebContent/includes

|  |  |
| --- | --- |
| **Classe** | **Descrizione** |
| **footer.jsp** | Footer della pagina con informazioni aziendali, link rapidi (Home, FAQ, Privacy, Termini e condizioni), indirizzo, contatti telefonici ed email. |
| **head.jsp** | Inclusione di fogli di stile e script necessari per il funzionamento di Slick Carousel, icona del sito, e l'uso delle librerie Bootstrap e Font Awesome. |
| **navbar.jsp** | Creazione di una barra di navigazione interattiva con jQuery, icone di categoria (console, videogiochi, computer, gadget), e una funzione di ricerca. |
| **topbar.jsp** | Creazione di una barra superiore con il logo e icone dinamiche per il login, il profilo utente, il carrello e il logout. La visibilità delle icone cambia in base allo stato di sessione (utente o admin). |

Immagine che contiene testo, Carattere, schermata, numero

Descrizione generata automaticamente

## Class Interface

### AmministratoreDao

|  |  |
| --- | --- |
| **Nome classe** | **AmministratoreDao** |
|  |  |
| **precondizione** | context AmministratoreDao::**adminLogin**(email:String, password:String): Admin  **pre**: email!=null, password!=null  context AmministratoreDao::**getAdmin**(id:int): Admin  **pre**: email!=null, password!=null |
| **post-condizione** |  |
| **invariante** |  |

### CarrelloDao

|  |  |
| --- | --- |
| **Nome classe** | **CarrelloDao** |
|  |  |
| **precondizione** | context CarrelloDao::**getCarrelloByUsername**(username:String): List<carrelloList>  **pre**: username!=null  context CarrelloDao::**aggiungiProdottoAlCarrello**(username:String, codiceProdotto:int, quantita:int): void  **pre**: username!=null, codiceProdotto!=null, quantita!=null  context CarrelloDao::**cambiaQuantitaProdotto**(username:String, codiceProdotto:int, nuovaQuantita:int): void  **pre**: username!=null, codiceProdotto!=null, nuovaQuantita!=null  context CarrelloDao::**rimuoviProdottoDalCarrello**(username:String, codiceProdotto:int): void  **pre**: username!=null, codiceProdotto!=null  context CarrelloDao::**cancellaCarrello**(username:String): void  **pre**: username!=null |
| **post-condizione** |  |
| **invariante** |  |

### OrdineDao

|  |  |
| --- | --- |
| **Nome classe** | **OrdineDao** |
|  |  |
| **precondizione** | context OrdineDao::**aggiungiOrdine(**ordine: Ordine, prodottiOrdine: List<Carrello>): void  **pre**: ordine!=null, prodottiOrdine!=null  context OrdineDao::**getOrdiniUtente**(username:String): Map<Ordine, List<ProdottoOrdine>>  **pre**: username!=null  context OrdineDao::**getOrdiniByDateRange**(startDate:String, endDate:String): Map<Ordine, List<ProdottoOrdine>>  **pre**: startDate!=null, endDate!=null |
| **post-condizione** | context OrdineDao::**getAllOrdini**(): Map<Ordine, List<ProdottoOrdine>> map  **post**: map!= null |
| **invariante** |  |

### ProdottoDao

|  |  |
| --- | --- |
| **Nome classe** | **ProdottoDao** |
|  |  |
| **precondizione** | context ProdottoDao::**getProdottoByCodice**(codice:int): Prodotto  **pre**: codice!=null  context ProdottoDao::**getVideogameByPrezzo**(min:int, max:int): List<Prodotto>  **pre**: min!=null, max!=null  context ProdottoDao::**getComputerByPrezzo**(min:int, max:int): List<Prodotto>  **pre**: min!=null, max!=null  context ProdottoDao::**getAccessorioByPrezzo**(min:int, max:int): List<Prodotto>  **pre**: min!=null, max!=null  context ProdottoDao::**getConsoleByPrezzo**(min:int, max:int): List<Prodotto>  **pre**: min!=null, max!=null  context ProdottoDao::**insertProdotto** (prezzo:float, nome:String, marca:String, descrizione:String, data\_usc:String, categoria:String): void  **pre**: prezzo!=null, nome!=null, marca!=null, descrizione!=null, data\_usc!=null, categoria!=null  context ProdottoDao::**updateProdotto** (codice:int, prezzo:float, nome:String, marca:String, descrizione:String, data\_usc:String, categoria:String): void  **pre**: codice!=null, prezzo!=null, nome!=null, marca!=null, descrizione!=null, data\_usc!=null, categoria!=null  context ProdottoDao::**deleteProdotto**(codice:String): boolean  **pre**: codice!=null  context ProdottoDao::**controllaEsistenza**(codice:String): boolean  **pre**: codice!=null |
| **post-condizione** | context ProdottoDao::**getAllProdotto**(): List<Prodotto> list  **post**: list!=null  context ProdottoDao::**getAllVideogame** (): List<Prodotto> list  **post**: list!=null  context ProdottoDao::**getAllConsole**(): List<Prodotto> list  **post**: list!=null  context ProdottoDao::**getAllComputer**(): List<Prodotto> list  **post**: list!=null  context ProdottoDao::**getAllAccessori**(): List<Prodotto> list  **post**: list!=null  context ProdottoDao::**getUltimoProdotto**(): Prodotto p  **post**: p!=null |
| **invariante** |  |

### UserDao

|  |  |
| --- | --- |
| **Nome classe** | **UserDao** |
|  |  |
| **precondizione** | context UserDao::**userLogin**(username:String, password:String): User  **pre**: username!=null, password!=null  context UserDao::**registerUser**(user:User): boolean  **pre**: user!=null  context UserDao::**getUserProfile**(username:String): User  **pre**: username!=null  context UserDao::**updateUser**(user:User): void  **pre**: user!=null  context UserDao::**updateUserPaypal**(user:User): void  **pre**: user!=null  context UserDao::**updateUsername**(vecchioUsername:String, nuovoUsername:String): void  **pre**: vecchioUsername!=null, nuovoUsername!=null  context UserDao::**checkUsername**(nuovoUsername:String): boolean  **pre**: nuovoUsername!=null  context UserDao::**updatePassword**(username:String, nuovaPassword:String): void  **pre**: username!=null, nuovaPassword!=null  context UserDao::**updateImmagine**(username:String, nuovaImmagine:String): void  **pre**: username!=null, nuovaImmagine!=null  context UserDao::**updateEmail**(username:String, nuovaEmail:String): void  **pre**: username!=null, nuovaEmail!=null  context UserDao::**updatePassword**(username:String, nuovaPassword:String): void  **pre**: username!=null, nuovaPassword!=null  context UserDao::**updatePaypal**(username:String, nuovaPaypal:String): void  **pre**: username!=null, nuovPaypal!=null  context UserDao::**updateDataNascita**(username:String, nuovaDataNascita:String): void  **pre**: username!=null, nuovaDataNascita!=null  context UserDao::**updateNome**(username:String, nuovoNome:String): void  **pre**: username!=null, nuovoNome!=null  context UserDao::**updateCognome**(username:String, nuovoCognome:String): void  **pre**: username!=null, nuovoCognome!=null  context UserDao::**updateNumero**(username:String, nuovoNumero:String): void  **pre**: username!=null, nuovoNumero!=null  context UserDao::**updateIndirizzo**(username:String,indirizzo:String, stato:String, citta:String, codPostale: int): void  **pre**: username!=null, indirizzo!=null, stato!=null, citta!=null, codPostale!=null  context UserDao::**updateCodiceFiscale**(username:String, nuovoCodiceFiscale:String): void  **pre**: username!=null, nuovoCodiceFiscale!=null  context UserDao::**getCheckoutUse**(username:String): User  **pre**: user!=null |
| **post-condizione** |  |
| **invariante** |  |