|  |
| --- |
| **Università degli Studi di Salerno Corso di Ingegneria del Software** |

**MusicParadise  
Object Design Documentation  
Versione 0.1**

**LOGO PROGETTO**

****

Data: 19/12/2017

**Coordinatore del progetto:**

|  |  |
| --- | --- |
| Nome | Matricola |
|  |  |
|  |  |

**Partecipanti:**

|  |  |
| --- | --- |
| Nome | Matricola |
| **Antonio Spera** | 0512103538 |
| **Alessandro De Riso** | 0512103802 |
| **Domenico Pannone** | 0512103730 |
| **Vincenzo Pandolfo** | 0512103694 |
|  |  |
|  |  |

|  |  |
| --- | --- |
| **Scritto da:** |  |

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Data | Versione | Descrizione | Autore |
| 22/12/2017 | 0.1 | Aggiunta descrizione classe Ordine | Antonio Spera |
| 22/12/2017 | 0.1 | Aggiunta descrizione classe Prodotto | Alessandro De Riso |
| 22/12/2017 | 0.1 | Aggiunta descrizione classe Indirizzo | Domenico Pannone |
| 22/12/2017 | 0.1 | Aggiunta descrizione classe Utente | Vincenzo Pandolfo |

Sommario

[2 INTRODUZIONE 5](#_Toc502306558)

[2.1 OBJECT DESIGN TRADE-OFF 5](#_Toc502306559)

[2.1.1 MODULARITA’ VS EFFICIENZA 5](#_Toc502306560)

[2.1.2 SICUREZZA VS EFFICIENZA 5](#_Toc502306561)

[2.1.3 PORTABILITA’ VS EFFICIENZA 5](#_Toc502306562)

[2.2 INTERFACCE DOCUMENTATION GUIDELINES 6](#_Toc502306563)

[2.2.1 FILE JAVA 6](#_Toc502306564)

[2.2.2 NAMING 6](#_Toc502306565)

[2.2.3 USO DEI COMMENTI 6](#_Toc502306566)

[2.2.4 ALTRE REGOLE DI STILE 7](#_Toc502306567)

[2.3 DEFINIZIONI, ACRONIMI E ABBREVIAZIONI 7](#_Toc502306568)

[2.4 RIFERIMENTI 7](#_Toc502306569)

[2.5 OVERVIEW 7](#_Toc502306570)

[3 PACKAGES 8](#_Toc502306571)

[4 INTERFACCE DELLE CLASSI 9](#_Toc502306572)

[4.1 CLASS DIAGRAM 9](#_Toc502306573)

[4.1.1 Accesso 9](#_Toc502306574)

[4.1.2 Ricerca prodotto 10](#_Toc502306575)

[4.2 DESCRIZIONE DELLE CLASSI 11](#_Toc502306576)

[4.2.1 Utente 11](#_Toc502306577)

[4.2.2 Prodotto 11](#_Toc502306578)

[4.2.3 Ordine 13](#_Toc502306579)

[4.2.4 Indirizzo 13](#_Toc502306580)

[5 GLOSARRIO 14](#_Toc502306581)

# INTRODUZIONE

## OBJECT DESIGN TRADE-OFF

### MODULARITA’ VS EFFICIENZA

### SICUREZZA VS EFFICIENZA

Nel nostro sistema ogni richiesta viene validata attraverso l’uso della sessione e un controllo a livello di utenza. Questo ci permette di impedire un accesso non autorizzato. Per fare ciò ogni pagina servlet o jsp deve autenticare l’utente oppure verificare che l’utente sia stato autenticato in precedenza. Questa caratteristica però potrebbe far alzare il tempo di risposta del sistema soprattutto con carichi di lavoro molto alti. Eliminando i controlli per l’esistenza e validità della sessione e introdurre la precondizione di sessione esistente, può portare a rischi di sicurezza molto elevati, in quanto utenti maliziosi potrebbero richiamare servizi non consentiti provocando danni al sistema. Per questo motivo anche se a discapito dell’efficienza optiamo per questi controlli.

### PORTABILITA’ VS EFFICIENZA

## INTERFACCE DOCUMENTATION GUIDELINES

### FILE JAVA

Ogni file deve contenere:

* Commenti per comprendere al meglio il codice
* Dichiarazione dei package
* Sezione import
* Dichiarazione di classe o interfaccia
  + Attributi pubblici
  + Attributi privati
  + Attributi protetti
  + Costruttori
  + Metodi getter e setter
  + Altri metodi
* Classi interne

### NAMING

Vengono utilizzate convenzioni su nomi per rendere il programma più leggibile e comprensibile.

**Classi e interfacce**

I nomi delle classi possono essere composti da più parole (ad esempio **ProdottoBean),** ogni parola che compone il nome della classe ha l’iniziale maiuscola. I nomi devono essere semplici e descrittivi in modo da capire il ruolo della classe. Evitare abbreviazioni.

**Metodi**

I **metodi** devono essere verbi (composti anche da più parole) con iniziale maiuscola. Ad esempio: addProdotto().

**Costanti**

Rispettando le convenzioni suggerite da Sun, i nomi delle **costanti** vengono indicati da nomi con tutti i caratteri maiuscoli. Le parole vengono separate da **“\_”.** Ad esempio: **static final MAX\_COUNT = 10**

### USO DEI COMMENTI

E’ possibile utilizzare due diversi tipi di commenti:

* Commenti **Javadoc** (testo compreso tra /\*\* e \*/)
* Commenti per singole righe (delimitate da //)

L’utilizzo dei commenti **Javadoc** è suggerito prima della dichiarazione di:

* Classi e interfacce
* Costruttori
* Metodi
* Variabili di classe

I commenti compresi tra /\*\* e \*/ devono essere semplici e chiari, in modo da rendere leggibile la documentazione.

### ALTRE REGOLE DI STILE

Ulteriori regole sono le seguenti:

* Nomi di package, classi e metodi devono essere nomi descrittivi
* Si consiglia l’utilizzo di parti standard dei nomi in casi come:
  + Classi astratte (es: AbstractClass)
  + Design pattern (es: ProdottoModel)
  + Eccezioni (es: ProdottoNonTrovatoException)
* è consigliato l’uso di suffissi standard come “get”, “set”, “is”,”has”

## DEFINIZIONI, ACRONIMI E ABBREVIAZIONI

|  |  |
| --- | --- |
|  |  |
| RAD | Documento di analisi dei requisiti |
| SDD | System Design Document |
| MVC | Model Control View |

## RIFERIMENTI

RAD-MusicParadise.com

SDD-MusicParadise.com

## OVERVIEW

# PACKAGES

Le componenti base che costituiscono il sistema sono raggruppati in 3 livelli:

* Interface layer
* Application Logic layer
* Storage layer

## package core

Immagine che contiene cielo

Descrizione generata con affidabilità molto elevata

## package bean

Immagine che contiene screenshot

Descrizione generata con affidabilità molto elevata

|  |  |
| --- | --- |
|  |  |
| UtenteBean | Reppresenta l’ogetto utente |
| ProdottoBean | Rappresenta l’ogetto prodotto |
| OrdineBean | Rappresenta l’ogetto ordine |
| IndirizzoBean | Rappresenta l’ogetto indirizzo |
| CartaBean | Rappresenta l’ogetto carta |
| CarrelloBean | Rappresenta l’ogetto carrello |

## package control

Immagine che contiene screenshot, cielo

Descrizione generata con affidabilità molto elevata

### Package Accesso

Immagine che contiene screenshot

Descrizione generata con affidabilità molto elevata

|  |  |
| --- | --- |
|  |  |
| AccediControl | Permette di effettuare il login al sistema |
| LogoutControl | Permette di effettuare il logout dal sistema |

### Package GestioneProfilo

Immagine che contiene screenshot

Descrizione generata con affidabilità molto elevata

|  |  |
| --- | --- |
|  |  |
| AggIndirizzoControl | Permette di aggiungere un nuovo indirizzo all’account |
| AggiungiCartaControl | Permette di aggiungere una nuova carta collegata all’account |
| visualizzaOrdiniCliente | Permette di ottenere gli ordini eseguiti dal cliente |

### Checkout

Immagine che contiene screenshot, cielo, donna

Descrizione generata con affidabilità molto elevata

|  |  |
| --- | --- |
|  |  |
| AggiungiProdottoAlCarrelloControl | Permette di aggiungere un prodotto al carrello |
| RimuovoProdottoCarrelloControl | Permette di rimuovere un prodotto dal carrello |
| RicercaProdottoControl | Permette di effettuare una ricerca di un prodotto |
| RicercaOrdineControl | Permette di effettuare la ricerca degli ordini |
| CheckoutControl | Permette di accedere all’area checkout |
| ConfermaOrdineControl | Permette di confermare l’ordine |

## package model

Immagine che contiene screenshot, cielo

Descrizione generata con affidabilità molto elevata

|  |  |
| --- | --- |
|  |  |
| UtenteModel | Permette di effettuare le query riguardanti all’utente |
| ProdottoModel | Permette di effettuare le query riguardanti al prodotto |
| FotoModel | Permette di effettuare le query riguardanti alle foto dei prodotti |
| OrdineModel | Permette di effettuare le query riguardanti all’ordine |
| IndirizzoModel | Permette di effettuare le query riguardanti all’indirizzo |
| CartaModel | Permette di effettuare le query riguardanti alla carta |
| ComposizioneModel | Permette di effettuare le query riguardanti alla composizione degli ordini |
| CarrelloModel | Permette di effettuare le query riguardanti al salvataggio persistente del carrello |

## package view

Immagine che contiene screenshot

Descrizione generata con affidabilità molto elevata

# INTERFACCE DELLE CLASSI

## CLASS DIAGRAM

### AccediConrol

|  |  |
| --- | --- |
| Nome della classe | AccediControl |
| Descrizione | Questa classe è una servelt che si occupa del login dell’utente. |
| Metodi | +doGet(HttpServletRequest request,HttpServletResponse response): void  +doPost(HttpServletRequest request, HttpServletResponse response): void |

|  |  |
| --- | --- |
| Metodo | +doPost(HttpServletRequest request,HttpServletResponse response): void |
| Descrizione | Questo metodo si occupa di effettuare il login da parte dell’utente utilizzando la coppia di stringhe user/password inserite dall’utente negli appositi campi di testo |
| Pre-Condizione | **Context:** AccediControl : doPost(request,response)  **Pre:** request.getParameter(“nick”) !=null && request.getParameter(“numPassword”) !=null && request.getParameter(“action”) !=null && request.getParameter(“i”) !=null |
| Post-Condizione | If(cliente!=null || gestoreOrdini!=null)  Then loginOK |

### LogoutControl

|  |  |
| --- | --- |
| Nome della classe | LogoutControl |
| Descrizione | Questa classe è una servelt che si occupa del logout dell’utente. |
| Metodi | +doGet(HttpServletRequest request,HttpServletResponse response): void  +doPost(HttpServletRequest request, HttpServletResponse response): void |

|  |  |
| --- | --- |
| Metodo | +doGet(HttpServletRequest request,HttpServletResponse response): void |
| Descrizione | Questo metodo permette il logout da parte dell’utente. |
| Pre-Condizione | **Context:** LogoutControl: doGet(request,response)  **Pre:** request.getParameter("action") !=null |
| Post-Condizione | il cliente ha effettuato correttamente il logout dal sistema |

### AggiungiCartaControl

|  |  |
| --- | --- |
| Nome della classe | AggiungiCartaControl |
| Descrizione | Questa classe è una servelt che si occupa di poter aggiungere una carta all’account dell’utente. |
| Metodi | +doGet(HttpServletRequest request,HttpServletResponse response): void  +doPost(HttpServletRequest request, HttpServletResponse response): void |

|  |  |
| --- | --- |
| Metodo | +doPost(HttpServletRequest request,HttpServletResponse response): void |
| Descrizione | Questo metodo permette l’inserimento di una carta di credito utilizzando un form dove vengono specificati: proprietario, numero carte, data di scadenza e codice cv. |
| Pre-Condizione | **Context:** AggiungiCartaControl : doPost(request,response)  **Pre**: request.getParameter(“numCarta”) !=null && request.getParameter(“mese”) !=null && request.getParameter(“anno”) !=null && request.getParameter(“nomProrietario”) !=null |
| Post-Condizione | Il cliente ha aggiunto correttamente una nuova carta di credito. |

### RicercaOrdineControl

|  |  |
| --- | --- |
| Nome della classe | RicercaOrdineControl |
| Descrizione | Questa classe è una servlet che si occupa di gestire la ricerca degli ordini |
| Metodi | +doGet(HttpServletRequest request, HttpServletResponse response): void  +doPost(HttpServletRequest request, HttpServletResponse response): void |

|  |  |
| --- | --- |
| Nome Metodo | +doGet(HttpServletRequest request, HttpServletResponse response): void |
| Descrizione | Questo metodo si occupa di effettuare la ricerca degli ordini all'interno del database usando come riferimento il codice dell'ordine da ricercare. |
| Pre-condizione | **Context:**RicercaOrdine::doGet(request,response)  **Pre:**request.getParameter("CodiceOrdine")!=null |
| Post-condizione | Il cliente visualizza tutti gli ordini da lui effettuati |

### VisualizzaOrdiniCliente

|  |  |
| --- | --- |
| Nome della classe | VisualizzaOrdiniControl |
| Descrizione | Questa classe è una servelt che si occupa di far visualizzare gli ordini da parte dell’utente. |
| Metodi | +doGet(HttpServletRequest request,HttpServletResponse response): void  +doPost(HttpServletRequest request, HttpServletResponse response): void |

|  |  |
| --- | --- |
| Metodo | +doGet(HttpServletRequest request,HttpServletResponse response): void |
| Descrizione | Questo metodo permette la visualizzazione degli ordini effettuati dal cliente. |
| Pre-Condizione | **Context:** VisualizzaOrdiniControl: doGet(request,response)  **Pre:** request.getParameter("nickname") != null |
| Post-Condizione | Il cliente visualizza tutti gli ordini da lui effettuati |

### RicercaProdottoControl

|  |  |
| --- | --- |
| Nome della classe | RicercaProdottoControl |
| Descrizione | Questa classe è una servlet che si occupa di gestire la ricerca di prodotti dal catalogo. |
| Metodi | +doGet(HttpServletRequest request, HttpServletResponse response): void  +doPost(HttpServletRequest request, HttpServletResponse response): void |

|  |  |
| --- | --- |
| Nome Metodo | +doGet(HttpServletRequest request, HttpServletResponse response): void |
| Descrizione | Questo metodo si occupa di effettuare la ricerca di prodotti all'interno del catalogo utilizzando la stringa inserita dall'utente nell'apposita barra di ricerca. |
| Pre-condizione | **Context:**RicercaProdottoControl :doGet(request,response)  **Pre:** request.getParameter("stringaRicerca")!=null |
| Post-condizione | il cliente visualizza tutti i prodotti riguardanti la chiave di ricerca inserita |

### RimuoviProdortoCarrelloControl

|  |  |
| --- | --- |
| Nome della classe | RimuoviProdottoControl |
| Descrizione | Questa classe è una servlet che permette di rimuovere un prodotto dal database |
| Metodi | +doGet(HttpServletRequest request, HttpServletResponse response): void  +doPost(HttpServletRequest request, HttpServletResponse response): void |

|  |  |
| --- | --- |
| Nome Metodo | +doGet(HttpServletRequest request, HttpServletResponse response): void |
| Descrizione | Questo metodo si occupa della cancellazione del prodotto passando come parametro il codice prodotto da cancellare. |
| Pre-condizione | **Context:**RimuoviProdotto::doGet(request,response)  **Pre:**request.getParameter("CodiceProdotto")!=null |
| Post-condizione | Il prodotto non è più visualizzato all’interno del carrello. |

### AggiungiProdottoAlCarrelloControl

|  |  |
| --- | --- |
| Nome della classe | AggiungiProdottoAlCarrelloControl |
| Descrizione | Questa classe è una servlet che permette l’aggiunta di un prodotto al carrello. |
| Metodi | +doGet (HttpServletRequest request, HttpServletResponse response): void +doPost (HttpServletRequest request, HttpServletResponse response) : void |

|  |  |
| --- | --- |
| Nome del metodo | +doGet ( HttpServletRequest request, HttpServletResponse response) : void |
| Descrizione | Questo metodo si occupa di aggiungere un prodotto nel carrello del cliente. |
| Pre - condizione | **Context**: AggiungiProdottoAlCarrelloControl : doGet ( request, response )  **Pre** : request.getParameter(“azione”) !=null && request.getParameter(“codice”) != null |
| Post - condizione | Se il prodotto non era presente nel carrello allora lo aggiunge, se il prodotto era già presente nel carrello allora viene incrementata la quantità di quel prodotto |

### CheckoutControl

|  |  |
| --- | --- |
| Nome della classe | CheckoutControl |
| Descrizione | Questa classe salva le informazioni dell’ordine nel database. |
| Metodi | +doGet (HttpServletRequest request, HttpServletResponse response): void  +doPost (HttpServletRequest request, HttpServletResponse response) : void |

|  |  |
| --- | --- |
| Nome del metodo | +doGet ( HttpServletRequest request, HttpServletResponse response) : void |
| Descrizione | Questo metodo consente il salvataggio dei dati nel database. |
| Pre - condizione | **Context**: CheckoutControl :: doGet ( request, response )  **Pre** : |
| Post - condizione | Il cliente visualizzerà la form per inserire metodo di pagamento e indirizzo di spedizione. |

### ConfermaOrdineControl

|  |  |
| --- | --- |
| Nome della classe | ConfermaOrdineControl |
| Descrizione | Questa classe permette di confermare l’ordine. |
| Metodi | +doGet (HttpServletRequest request, HttpServletResponse response): void  +doPost (HttpServletRequest request, HttpServletResponse response) : void |

|  |  |
| --- | --- |
| Nome del metodo | +doPost ( HttpServletRequest request, HttpServletResponse response) : void |
| Descrizione | Questo metodo consente la conferma dell’ordine effettuato. |
| Pre - condizione | **Context**: ConfermaOrdineControl :: doPost ( request, response )  **Pre** : |
| Post - condizione | Il cliente ha effettuato l’ordine, il quale apparirà nella pagina personale “i miei ordini” |

### ConfermaModOrdineControl

|  |  |
| --- | --- |
| Nome della classe | ConfermaModOrdineControl |
| Descrizione | Questa classe permette la conferma della modifica dello stato di un ordine. |
| Metodi | +doGet (HttpServletRequest request, HttpServletResponse response): void  +doPost (HttpServletRequest request, HttpServletResponse response) : void |

|  |  |
| --- | --- |
| Nome del metodo | +doGet ( HttpServletRequest request, HttpServletResponse response) : void |
| Descrizione | Questo metodo consente la conferma della modifica dello stato dell’ordine effettuato. |
| Pre - condizione | **Context**: ConfermaModOrdineControl :: doGet( request, response )  **Pre** : |
| Post - condizione | Lo stato di un ordine è passato da “In preparazione” a “Spedito” |

ModificaStatoOrdineControl

|  |  |
| --- | --- |
| Nome della classe | ModificaStatoOrdineControl |
| Descrizione | Questa classe permette la modifica dello stato di un ordine. |
| Metodi | +doGet (HttpServletRequest request, HttpServletResponse response): void  +doPost (HttpServletRequest request, HttpServletResponse response) : void |

|  |  |
| --- | --- |
| Nome del metodo | +doGet ( HttpServletRequest request, HttpServletResponse response) : void |
| Descrizione | Questo metodo consente la dello stato di un ordine. |
| Pre - condizione | **Context**: ModificaStatoOrdineControl :: doGet( request, response )  **Pre** : |
| Post - condizione | Il gestore visualizzerà a video la form per modificare lo stato di un ordine. |

VisualizzaCatalogoControl

|  |  |
| --- | --- |
| Nome della classe | VisualizzaCatalogoControl |
| Descrizione | Questa classe è una servlet che permette di caricare tutti i prodotti presenti nel catalogo |
| Metodi | +doGet (HttpServletRequest request, HttpServletResponse response): void  +doPost (HttpServletRequest request, HttpServletResponse response) : void |

|  |  |
| --- | --- |
| Nome del metodo | +doGet ( HttpServletRequest request, HttpServletResponse response) : void |
| Descrizione | Questo metodo carica tutti i prodotti presenti nel catalogo. |
| Pre - condizione | **Context**: VisualizzaCatalogoControl :: doGet( request, response )  **Pre** : |
| Post - condizione | Il cliente visualizzarà tutti i prodotti presenti nel catalogo. |

AggIndirizzoControl

|  |  |
| --- | --- |
| Nome della classe | AggIndirizzoControl |
| Descrizione | Questa classe è una servlet che gestisce l’inserimento di un nuovo indirizzo da parte del cliente |
| Metodi | +doGet (HttpServletRequest request, HttpServletResponse response): void  +doPost (HttpServletRequest request, HttpServletResponse response) : void |

|  |  |
| --- | --- |
| Nome del metodo | +doGet ( HttpServletRequest request, HttpServletResponse response) : void |
| Descrizione | Questo metodo gestisce l’inserimento di un nuovo indirizzo da parte di un cliente. |
| Pre - condizione | **Context**: AggIndirizzoControl:: doGet( request, response )  **Pre** : |
| Post - condizione | Il cliente ha aggiunto correttamente una nuova carta di credito. |

## DESCRIZIONE DELLE CLASSI

### UtenteBean

Immagine che contiene screenshot

Descrizione generata con affidabilità molto elevata

**Nickname** : nickname dell’utente per effettuare l’accesso

**Password** : la password che utilizza l’utente per l’accesso

**Nome** : nome dell’utente

**Cognome** : cognome dell’utente

**Email** : e-mail dell’utente

**Matricola** : matricola che identifica l’utente gestore

**Tipo** : tipo di utente

**Indirizzi:** indirizzi che possiede il cliente

**Carte:** carte collegate all’account del cliente

### ProdottoBean

**Immagine che contiene testo

Descrizione generata con affidabilità elevata**

**codice** : codice che identifica univocamente il prodotto nel sistema;

**nome**: Nome del prodotto;

**descrizione**: Descrizione dettagliata del prodotto;

**colore**: Colorazione in cui è disponibile il prodotto;

**peso**: Peso del prodotto;

**marca**: Marca del prodotto;

**dataInserimento**: Data in cui il prodotto è stato inserito nel catalogo;

**numDisponibilita**: Quantità disponibile delle unità di prodotto;

**strumento**: Famiglia degli strumenti di cui fa parte il prodotto;

**prezzo**: prezzo del prodotto;

**quantità:** quantità del prodotto inserita nel carrello

**foto:** path delle foto del prodotto

### OrdineBean

Immagine che contiene screenshot

Descrizione generata con affidabilità molto elevata

**NumeroOrdine:** identifica univocamente l’ordine

**Stato:** rappresenta lo stato dell’ordine che può trovarsi “in preparazione” o “spedito”

**Corriere:** rappresenta l’informazione relativa al corriere per la spedizione

**NumTraking:** numero che permette di rintracciare l’ordine durante la spedizione

**Totale:** rappresenta il totale dell’ordine

**Indirizzo:** rappresenta l’indirizzo dove l’ordine viene spedito

**Carta:** rappresenta la carta con cui è stato pagato l’ordine

**Prodotti:** rappresentano i prodotti di cui è composto l’ordine.

### IndirizzoBean

**Immagine che contiene screenshot

Descrizione generata con affidabilità elevata**

**IdIndirizzo**: Id dell’indirizzo identifica univocamente un indirizzo nel database.

**Indirizzo**: Indirizzo per la spedizione

**Città**: città per la spedizione

**CAP**: Codice di avviamento postale della città dell’indirizzo fornito

**Telefono:** numero di telefono

**Nome:** nome del destinatario

**Cognome:**  cognome del destinatario

### CartaBean

Immagine che contiene screenshot

Descrizione generata con affidabilità molto elevata

**Codice**: codice che identifica univocamente la carta

**Scadenza**: scadenza della carta

**NumeroCarta**: numero della carta prepagata

**NomeProrpeitario**: nome e cognome dell’intestatario della carta

### CarrelloBean

Immagine che contiene screenshot

Descrizione generata con affidabilità molto elevata

**Prodotti:** prodotti presenti nel carrello

**Totale:** prezzo totale del carrello

# GLOSARRIO