|  |
| --- |
| **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 6](#_Toc504520531)

[2.1 OBJECT DESIGN TRADE-OFF 6](#_Toc504520532)

[2.1.1 MODULARITA’ VS EFFICIENZA 6](#_Toc504520533)

[2.1.2 SICUREZZA VS EFFICIENZA 6](#_Toc504520534)

[2.1.3 PORTABILITA’ VS EFFICIENZA 6](#_Toc504520535)

[2.2 INTERFACCE DOCUMENTATION GUIDELINES 7](#_Toc504520536)

[2.2.1 FILE JAVA 7](#_Toc504520537)

[2.2.2 NAMING 7](#_Toc504520538)

[2.2.3 USO DEI COMMENTI 7](#_Toc504520539)

[2.2.4 ALTRE REGOLE DI STILE 8](#_Toc504520540)

[2.3 DEFINIZIONI, ACRONIMI E ABBREVIAZIONI 8](#_Toc504520541)

[2.4 RIFERIMENTI 8](#_Toc504520542)

[2.5 OVERVIEW 8](#_Toc504520543)

[3 PACKAGES 9](#_Toc504520544)

[3.1 package core 9](#_Toc504520545)

[3.2 package bean 10](#_Toc504520546)

[3.3 package control 11](#_Toc504520547)

[3.3.1 Package Accesso 11](#_Toc504520548)

[3.3.2 Package GestioneProfilo 12](#_Toc504520549)

[3.3.3 Checkout 12](#_Toc504520550)

[3.4 package model 13](#_Toc504520551)

[3.5 package view 14](#_Toc504520552)

[4 INTERFACCE DELLE CLASSI 15](#_Toc504520553)

[4.1 CLASS DIAGRAM 15](#_Toc504520554)

[4.1.1 AccediConrol 15](#_Toc504520555)

[4.1.2 LogoutControl 15](#_Toc504520556)

[4.1.3 AggiungiCartaControl 16](#_Toc504520557)

[4.1.4 RicercaOrdineControl 16](#_Toc504520558)

[4.1.5 VisualizzaOrdiniCliente 17](#_Toc504520559)

[4.1.6 RicercaProdottoControl 17](#_Toc504520560)

[4.1.7 RimuoviProdortoCarrelloControl 18](#_Toc504520561)

[4.1.8 AggiungiProdottoAlCarrelloControl 18](#_Toc504520562)

[4.1.9 CheckoutControl 19](#_Toc504520563)

[4.1.10 ConfermaOrdineControl 19](#_Toc504520564)

[4.1.11 ConfermaModOrdineControl 19](#_Toc504520565)

[4.1.12 ModificaStatoOrdineControl 20](#_Toc504520566)

[4.1.13 VisualizzaCatalogoControl 20](#_Toc504520567)

[4.1.14 AggIndirizzoControl 21](#_Toc504520568)

[4.1.15 CarrelloModel 22](#_Toc504520569)

[4.1.16 ProdottoOrdineModel 23](#_Toc504520570)

[4.1.17 OrdineModel 24](#_Toc504520571)

[4.2 DESCRIZIONE DELLE CLASSI 26](#_Toc504520572)

[4.2.1 UtenteBean 26](#_Toc504520573)

[4.2.2 ProdottoBean 27](#_Toc504520574)

[4.2.3 OrdineBean 28](#_Toc504520575)

[4.2.4 IndirizzoBean 29](#_Toc504520576)

[4.2.5 CartaBean 30](#_Toc504520577)

[4.2.6 CarrelloBean 30](#_Toc504520578)

[4.2.7 ClienteBean 31](#_Toc504520579)

[4.2.8 GestoreOrdiniBean 31](#_Toc504520580)

[4.2.9 ProdottoCatalogo 32](#_Toc504520581)

[4.2.10 ProdottoOrdineBean 32](#_Toc504520582)

[5 GLOSARRIO 33](#_Toc504520583)

# 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  nick.length() >= 5 && nick.length() <=15  password.length()>=8 && password.length()<=20 |
| 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  numCarta.matches("[0-9]{16}") && scadenza.matches(“[0-9]{1,2}[/]{1}[0-9]{2}”) && nomeProprietario.matches(“[A-Za-z]{4,10}[ ]{1}[A-Za-z]{4,10}”) |
| 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 del prodotto da eliminare. |
| Pre-condizione | **Context:**RimuoviProdotto::doGet(request,response)  **Pre:**request.getParameter("CodiceProdotto")!=null |
| Post-condizione | Il prodotto viene rimosso dal 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 è presente nel carrello allora lo aggiunge.  se il prodotto era già presente nel carrello allora viene incrementata di 1 la quantità del 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** : dataConsegna != ni |
| 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** : dataConsegna != null && corriere != null && numeroTracking != null && ordine != null  dataConsegna.matches("[0-9]{2}[/]{1}[0-9]{2}[/]{1}[0-9]{4}") && numeroTracking.matches("[0-9]{10}")&&(corriere.matches("[A-Za-z]{3,20}") |
| 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** : codOrdine != null |
| 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** : nome != null && cognome != null && indirizzo != null && città != null && cap != null && telefono != null  nome.matches("[a-zA-Z]{1,10}[ ]{0,1}[a-zA-Z]{0,10}") && cognome.matches("[a-zA-Z]{1,10}[ ]{0,1}[a-zA-Z]{0,10}") &&città.matches("[a-zA-Z]{1,10}[ ]{0,1}[a-zA-Z]{0,10}")&& indirizzo.matches("[a-zA-Z0-9]{1,10}[ ]{0,1}[a-zA-Z0-9]{0,10}")&& cap.matches("[0-9]{5}") && telefono.matches("[0-9]{10}") |
| Post - condizione | Il cliente ha aggiunto correttamente un nuovo indirizzo. |

### CarrelloModel

|  |  |
| --- | --- |
| Nome della classe | CarrelloModel |
| Descrizione | Questa classe gestisce l’informazione persistente carello nel database |
| Metodi | doSave(List<ProdottoCatalogoBean> list, UtenteBean utente)  generaCodice()  leggi(String nickname)  remove(UtenteBean utente) |

|  |  |
| --- | --- |
| Nome del metodo | + doSave(List<ProdottoCatalogoBean> list, UtenteBean utente): void |
| Descrizione | Questo metodo consente il salvataggio del carrello nel database |
| Pre - condizione | List != null && utente != null |
| Post - condizione | Il sistema memorizza il carrello nel database |

|  |  |
| --- | --- |
| Nome del metodo | +generaCodice(): int |
| Descrizione | Questo metodo genera l’id per l’elemento del database |
| Pre - condizione | // |
| Post - condizione | Il sistema genera l’identificativo per l’elemento nel db |

|  |  |
| --- | --- |
| Nome del metodo | + leggi(String nickname): ArrayList<ProdottoCatalogoBean> |
| Descrizione | Questo metodo restituisce i prodotti del carrello del cliente passato come parametro |
| Pre - condizione | Nickname != null |
| Post - condizione | Il metodo ritorna l’ArrayList di prodotti. |

|  |  |
| --- | --- |
| Nome del metodo | +remove(UtenteBean utente): void |
| Descrizione | Questo metodo rimuove il carrello associato all’utente dal database |
| Pre - condizione | utente != null |
| Post - condizione | Se presente rimuove il carrello dal database |

### ProdottoOrdineModel

|  |  |
| --- | --- |
| Nome della classe | ProdottoOrdineModel |
| Descrizione | Questa classe gestisce l’informazione persistente del prodotto dell’ordine nel database |
| Metodi | doSave(ArrayList<ProdottoCatalogoBean> prodotti)  generaCodice()  prodottiOrdine(**int** codiceOrdine) |

|  |  |
| --- | --- |
| Nome del metodo | +doSave(ArrayList<ProdottoCatalogoBean> prodotti): void |
| Descrizione | Questo metodo memorizza i prodotti dell’ordine nel database |
| Pre - condizione | prodotti != null |
| Post - condizione | Memorizza i prodotti nel database |

|  |  |
| --- | --- |
| Nome del metodo | + generaCodice(): void |
| Descrizione | Questo metodo genera l’id per l’oggetto da memorizzare nel database |
| Pre - condizione | // |
| Post - condizione | Restituisce l’intero che rappresenta l’id |

|  |  |
| --- | --- |
| Nome del metodo | + prodottiOrdine(int codiceOrdine):ArrayList<ProdottoOrdineBean> |
| Descrizione | Questo metodo restituisce i prodotti dell’ordine con codice passato come parametro |
| Pre - condizione | codiceOrdine != null |
| Post - condizione | Restiuisce l’ArrayList di prodotti dell’ordine |

### OrdineModel

|  |  |
| --- | --- |
| Nome della classe | OrdineModel |
| Descrizione | Questa classe gestisce l’informazione persistente ordine nel database |
| Metodi | doRetrieveByKey(**int** code)  doRetrieveByStato(String stato)  doSave(IndirizzoBean indirizzo, CartaBean carta, String stato,String corriere, String tracking, **double** totale, String cliente)  generaCodice()  ordiniUtente(String nickname)  listaOrdini()  aggiorna(**int** cod, String num\_tracking, Date dataConsegna, String corriere)  aggiornaConsegnato(int codice) |

|  |  |
| --- | --- |
| Nome del metodo | + doRetrieveByKey(int code):OrdineBean |
| Descrizione | Questo metodo ritorna l’ordine in base al codice passato come parametro |
| Pre - condizione | code != null |
| Post - condizione | Ritorna ordine != null se trova un ordine corrispondente al codice passato.  Ritorna ordine == null se non trova l’ordine |

|  |  |
| --- | --- |
| Nome del metodo | + doRetrieveByStato(String stato):ArrayList<OrdineBean> |
| Descrizione | Questo metodo ritorna gli ordini in base allo stato passato come parametro |
| Pre - condizione | stato != null |
| Post - condizione | Ritorna ordini.size() > 0 se trova ordini con lo stato passato come parametro  Ritorna ordini.size() == 0 se non trova gli ordini |

|  |  |
| --- | --- |
| Nome del metodo | + doSave(IndirizzoBean indirizzo, CartaBean carta, String stato,String corriere, String tracking, double totale, String cliente) : void |
| Descrizione | Questo metodo salva l’ordine nel database |
| Pre - condizione | Indirizzo != null && carta != null && stato != null && corriere != null && tracking != null && totale != null && cliente != null |
| Post - condizione | Memorizza l’ordine nel database |

|  |  |
| --- | --- |
| Nome del metodo | + generaCodice(): void |
| Descrizione | Questo metodo genera l’id per l’oggetto da memorizzare nel database |
| Pre - condizione | // |
| Post - condizione | Restituisce l’intero che rappresenta l’id |

|  |  |
| --- | --- |
| Nome del metodo | + ordiniUtente(String nickname): ArrayList<OrdineBean> |
| Descrizione | Questo metodo ritorna gli ordini dell’utente passato come parametro |
| Pre - condizione | nickname != null |
| Post - condizione | Ritorna ordini.size() > 0 se trova ordini relativi all’utente passato come parametro  Ritorna ordini.size() == 0 se non trova gli ordini |

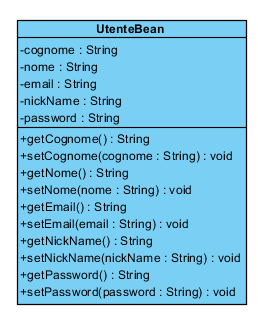
|  |  |
| --- | --- |
| Nome del metodo | + listaOrdini(): ArrayList<OrdineBean> |
| Descrizione | Questo metodo ritorna gli ordini memorizzati nel database |
| Pre - condizione | // |
| Post - condizione | Ritorna ordini.size() > 0 se trova ordini memorizzati nel database  Ritorna ordini.size() == 0 se non trova gli ordini |

|  |  |
| --- | --- |
| Nome del metodo | + aggiorna(int cod, String num\_tracking, Date dataConsegna, String corriere): void |
| Descrizione | Questo metodo aggiorna le informazioni relative all’ordine da in preparazione a spedito |
| Pre - condizione | cod != null && num\_tracking != null && dataConsegna != null && corriere != null |
| Post - condizione | Modifica le informazioni dell’ordine |

|  |  |
| --- | --- |
| Nome del metodo | + aggiornaConsegnato(int cod): void |
| Descrizione | Questo metodo aggiorna le informazioni relative all’ordine da spedito a consegnato |
| Pre - condizione | cod != null |
| Post - condizione | Modifica le informazioni dell’ordine |

## DESCRIZIONE DELLE CLASSI

### UtenteBean



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

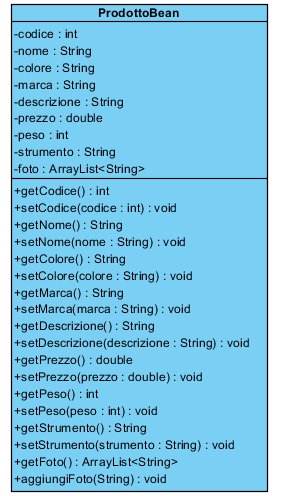
**Password** : la password che utilizza l’utente per l’accesso.

**Nome** : nome dell’utente.

**Cognome** : cognome dell’utente.

**Email** : e-mail dell’utente.

### ProdottoBean

****

**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.

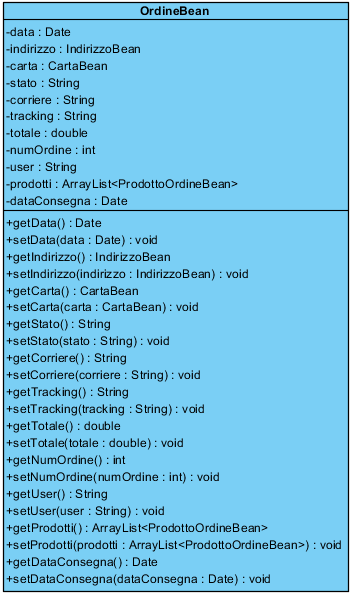
**marca**: Marca del prodotto.

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

**prezzo**: prezzo del prodotto.

**foto:** path delle foto del prodotto.

### OrdineBean



**numOrdine:** 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.

**traking:** 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.

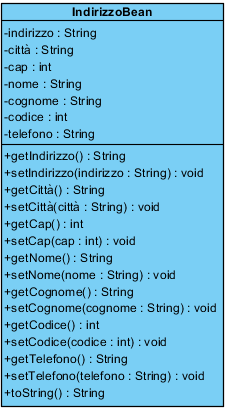
**prodotti:** rappresentano i prodotti di cui è composto l’ordine.

**data:** rappresenta la data in cui è stato effettuato l’ordine.

**user:** rappresenta il nickname dell’utente che ha effettuato l’ordine.

**dataConsegna:** rappresenta la data in cui è prevista la consegna.

### IndirizzoBean



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

**indirizzo**: Indirizzo per la spedizione.

**città**: città per la spedizione

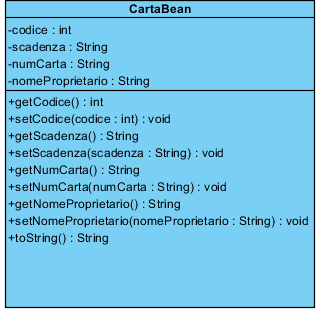
**cap**: Codice di avviamento postale della città dell’indirizzo fornito.

**telefono:** numero di telefono

**nome:** nome del destinatario.

**cognome:** cognome del destinatario.

### CartaBean



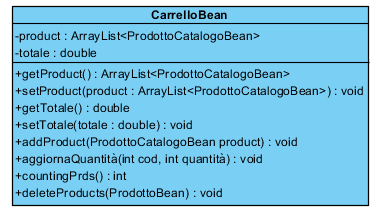
**Codice**: codice che identifica univocamente la carta.

**Scadenza**: scadenza della carta.

**NumeroCarta**: numero della carta prepagata.

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

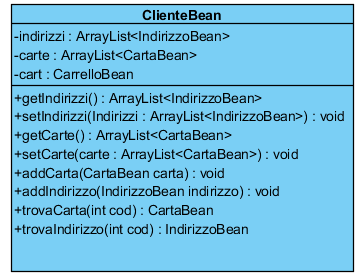
### CarrelloBean



**Prodotti:** prodotti presenti nel carrello.

**Totale:** prezzo totale del carrello.

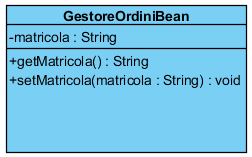
### ClienteBean

  
 **indirizzi:** rappresenta tutti gli indirizzi di spedizione che un cliente ha sul suo profilo.

**carte:** rappresenta tutte le carte di credito che un cliente ha sul suo profilo.

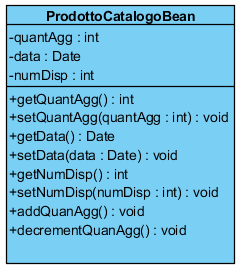
**cart:** rappresenta il carrello di un cliente.

### GestoreOrdiniBean



**matricola:** rappresenta la matricola di un gestore.

### ProdottoCatalogo

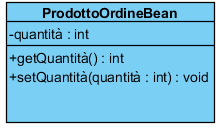


**quantAgg**: rappresenta la quantità di un prodotto nel carrello.

**data**: rappresenta la data di inserimento del prodotto nel catalogo.

**numDisp**: rappresenta la quantità di un prodotto disponibile.

### ProdottoOrdineBean



**quantità**: rappresenta la quantità del prodotto in un ordine.

# GLOSARRIO