|  |
| --- |
| **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](#_Toc504604744)

[2.1 OBJECT DESIGN TRADE-OFF 5](#_Toc504604745)

[2.1.1 MODULARITA’ VS EFFICIENZA 5](#_Toc504604746)

[2.1.2 SICUREZZA VS EFFICIENZA 5](#_Toc504604747)

[2.1.3 PORTABILITA’ VS EFFICIENZA 5](#_Toc504604748)

[2.2 INTERFACCE DOCUMENTATION GUIDELINES 6](#_Toc504604749)

[2.2.1 FILE JAVA 6](#_Toc504604750)

[2.2.2 NAMING 6](#_Toc504604751)

[2.2.3 USO DEI COMMENTI 6](#_Toc504604752)

[2.2.4 ALTRE REGOLE DI STILE 7](#_Toc504604753)

[2.3 DEFINIZIONI, ACRONIMI E ABBREVIAZIONI 7](#_Toc504604754)

[2.4 RIFERIMENTI 7](#_Toc504604755)

[2.5 OVERVIEW 7](#_Toc504604756)

[3 PACKAGES 8](#_Toc504604757)

[3.1 package core 8](#_Toc504604758)

[3.2 package bean 9](#_Toc504604759)

[3.3 package control 10](#_Toc504604760)

[3.3.1 Package Accesso 10](#_Toc504604761)

[3.3.2 Package GestioneProfilo 11](#_Toc504604762)

[3.3.3 Checkout 11](#_Toc504604763)

[3.4 package model 12](#_Toc504604764)

[3.5 package view 13](#_Toc504604765)

[4 INTERFACCE DELLE CLASSI 14](#_Toc504604766)

[4.1 CLASS DIAGRAM 14](#_Toc504604767)

[4.1.1 AccediConrol 14](#_Toc504604768)

[4.1.2 LogoutControl 14](#_Toc504604769)

[4.1.3 AggiungiCartaControl 15](#_Toc504604770)

[4.1.4 RicercaOrdineControl 15](#_Toc504604771)

[4.1.5 VisualizzaOrdiniCliente 16](#_Toc504604772)

[4.1.6 RicercaProdottoControl 16](#_Toc504604773)

[4.1.7 RimuoviProdortoCarrelloControl 17](#_Toc504604774)

[4.1.8 AggiungiProdottoAlCarrelloControl 17](#_Toc504604775)

[4.1.9 CheckoutControl 17](#_Toc504604776)

[4.1.10 ConfermaOrdineControl 18](#_Toc504604777)

[4.1.11 ConfermaModOrdineControl 18](#_Toc504604778)

[4.1.12 ModificaStatoOrdineControl 19](#_Toc504604779)

[4.1.13 VisualizzaCatalogoControl 19](#_Toc504604780)

[4.1.14 AggIndirizzoControl 20](#_Toc504604781)

[4.1.15 CarrelloModel 20](#_Toc504604782)

[4.1.16 ProdottoOrdineModel 22](#_Toc504604783)

[4.1.17 OrdineModel 23](#_Toc504604784)

[4.1.18 IndirizzoModel 25](#_Toc504604785)

[4.1.19 ProdottoModel 26](#_Toc504604786)

[4.1.20 CartaModel 29](#_Toc504604787)

[4.1.21 COMPOSIZIONE MODEL 30](#_Toc504604788)

[4.1.22 FOTO MODEL 31](#_Toc504604789)

[4.1.22 ClienteModel 32](#_Toc504604790)

[4.1.23 GestoreOrdineModel 33](#_Toc504604791)

[4.2 DESCRIZIONE DELLE CLASSI 34](#_Toc504604792)

[4.2.1 UtenteBean 34](#_Toc504604793)

[4.2.2 ProdottoBean 35](#_Toc504604794)

[4.2.3 OrdineBean 36](#_Toc504604795)

[4.2.4 IndirizzoBean 37](#_Toc504604796)

[4.2.5 CartaBean 38](#_Toc504604797)

[4.2.6 CarrelloBean 38](#_Toc504604798)

[4.2.7 ClienteBean 39](#_Toc504604799)

[4.2.8 GestoreOrdiniBean 39](#_Toc504604800)

[4.2.9 ProdottoCatalogo 40](#_Toc504604801)

[4.2.10 ProdottoOrdineBean 40](#_Toc504604802)

[5 GLOSARRIO 41](#_Toc504604803)

# 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

La portabilità del sistema MusicParadise.com è garantita dalla scelta del linguaggio di programmazione Java. Lo svantaggio dato da questa scelta è nella perdita di efficienza introdotta dal meccanismo della macchina virtuale Java. Tale compromesso è accettabile per i numerosi supporti forniti dal linguaggio Java.

## 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:

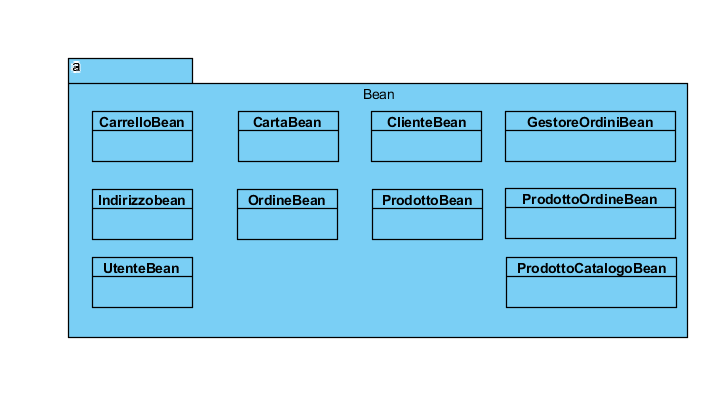
* Interface layer
* Application Logic layer
* Storage layer

## package core

Immagine che contiene cielo

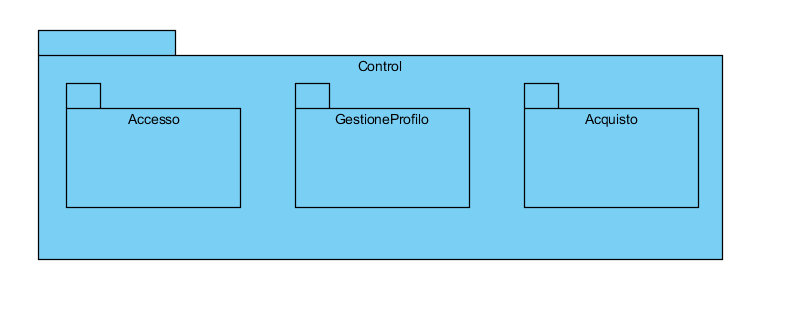
Descrizione generata con affidabilità molto elevata

## package bean



|  |  |
| --- | --- |
|  |  |
| UtenteBean | Reppresenta l’oggetto utente |
| ProdottoBean | Rappresenta l’oggetto prodotto |
| OrdineBean | Rappresenta l’oggetto ordine |
| IndirizzoBean | Rappresenta l’oggetto indirizzo |
| CartaBean | Rappresenta l’oggetto carta |
| CarrelloBean | Rappresenta l’oggetto carrello |
| ProdottoCatalogo | Rappresenta l’oggetto prodotto presente nel catalogo |
| ProdottoOrdineBean | Rappresenta l’oggetto prodotto presente nell’ordine |
| ClienteBean | Rappresenta l’oggetto cliente |
| GestoreOrdiniBean | Rappresenta l’oggetto gestore ordini |

## package control



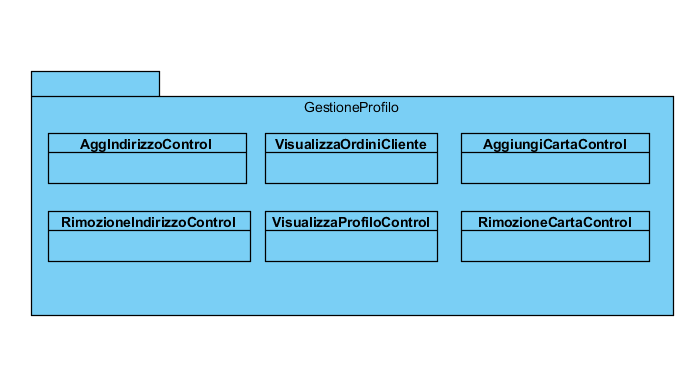
### Package Accesso

Immagine che contiene screenshot

Descrizione generata con affidabilità molto elevata

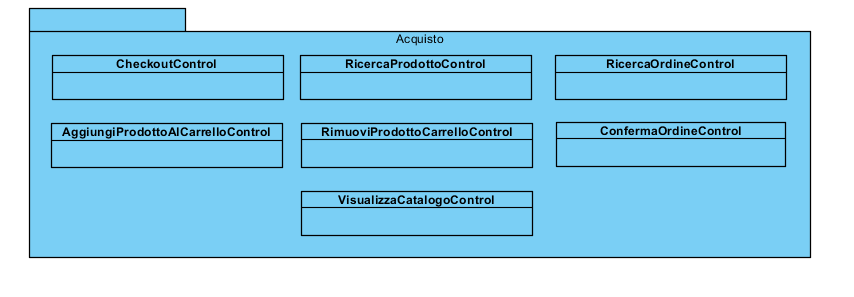
|  |  |
| --- | --- |
|  |  |
| AccediControl | Permette di effettuare il login al sistema |
| LogoutControl | Permette di effettuare il logout dal sistema |

### Package GestioneProfilo



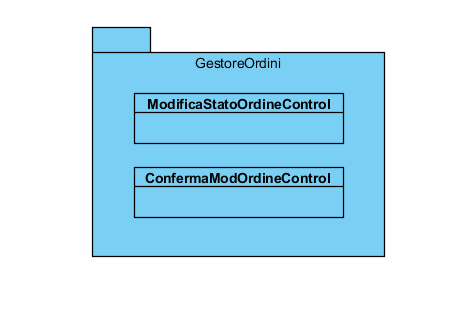
|  |  |
| --- | --- |
|  |  |
| AggIndirizzoControl | Permette di aggiungere un nuovo indirizzo all’account |
| AggiungiCartaControl | Permette di aggiungere una nuova carta collegata all’account |
| VisualizzaOrdiniCliente | Permette di ottenere gli ordini eseguiti dal cliente |
| RimozioneIndirizzoControl | Permette la rimozione di un indirizzo |
| RimozioneCartaControl | Permette la rimozione di una carta |
| VisualizzaProfiloControl | Permette di visualizzare il profilo utente |

### Acquisto



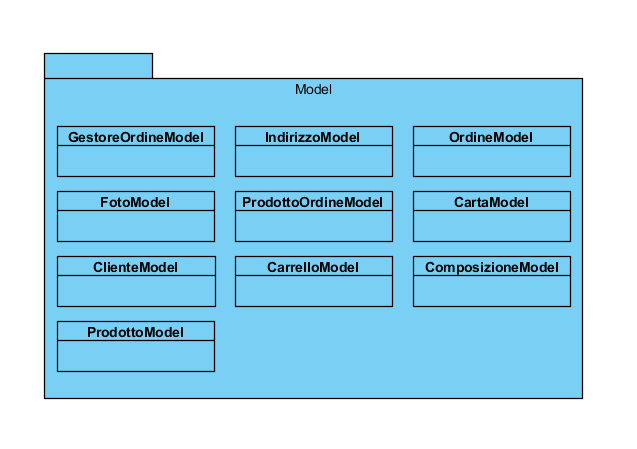
|  |  |
| --- | --- |
|  |  |
| AggiungiProdottoAlCarrelloControl | Permette di aggiungere un prodotto al carrello |
| RimuovoProdottoCarrelloControl | Permette di rimuovere un prodotto dal carrello |
| RicercaProdottoControl | Permette di effettuare una ricerca di un prodotto |
| RicercaOrdineControl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

### GestoreOrdini



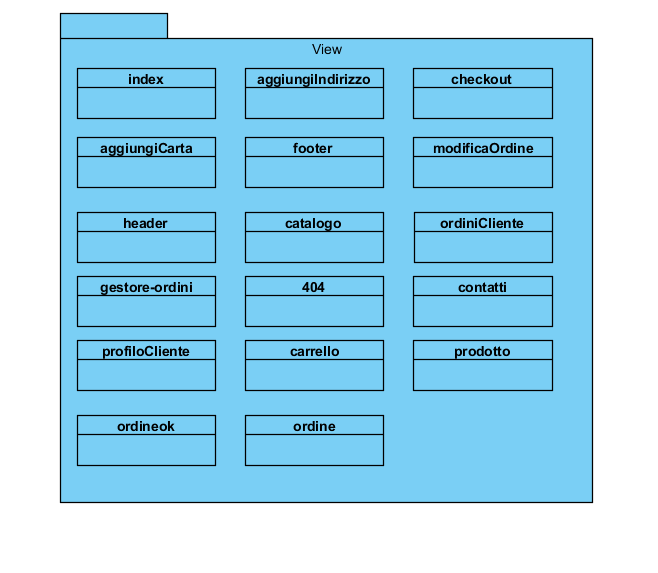
|  |  |
| --- | --- |
|  |  |
| ModificaStatoOrdineControl | Permette la modifica dello stato di un ordine |
| ConfermaModOrdineControl | Permette la conferma della modifica dello stato di un ordine |

## package model



|  |  |
| --- | --- |
|  |  |
| ClienteModel | Permette di effettuare le query riguardanti al cliente |
| 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 |
| GestoreOrdineModel | Permette di effettuare le query riguardanti il GesotoreOrdine |
| ProdottoOrdineModel | Permette di effettuare le query riguardanti il ProdottoOrdine |
| 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



# 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. |

### RimuoviIndirizzoControl

|  |  |
| --- | --- |
| Nome della classe | RimuoviIndirizzoControl |
| Descrizione | Questa classe è una servlet che gestisce la rimozione di un indirizzo da parte del cliente |
| 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 che gestisce la rimozione di un indirizzo da parte del cliente |
| Pre-condizione | **Context**: RimuoviIndirizzoControl:: doPost( request, response )  **Pre:** cliente!=null && codice!=null |
| Post-condizione | Il cliente ha rimosso correttamente l’indirizzo |

### RimuoviCartaControl

|  |  |
| --- | --- |
| Nome della classe | RimuoviCartaControl |
| Descrizione | Questa classe è una servlet che gestisce la rimozione di una carta da parte del cliente |
| 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 che gestisce la rimozione di una carta da parte del cliente |
| Pre-Condizione | **Context**: RimuoviCartaControl:: doPost( request, response )  **Pre:** cliente!=null && codice!=null |
| Post-Condizione | Il cliente ha rimosso correttamente la carta |

### CarrelloModel

|  |  |
| --- | --- |
| Nome della classe | CarrelloModel |
| Descrizione | Questa classe gestisce l’informazione persistente carello nel database |
| Metodi | doSave(List<ProdottoCatalogoBean> list, UtenteBean utente)  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 | + 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)  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 | + 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)  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 | + 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 da in preparazione a spedito |

|  |  |
| --- | --- |
| 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 da spedito a consegnato |

### IndirizzoModel

|  |  |
| --- | --- |
| Nome della classe | IndirizzoModel |
| Descrizione | Questa classe gestisce l’informazione persistente indirizzo nel database |
| Metodi | doSave(IndirizzoBean indirizzo, String nickname)  leggi(String nickname) |

|  |  |
| --- | --- |
| Nome del metodo | + doSave(IndirizzoBean indirizzo, String nickname): void |
| Descrizione | Questo metodo salva l’indirizzo nel database |
| Pre - condizione | Indirizzo!=null && nickname!=null |
| Post - condizione | Memorizza l’indirizzo nel database |

|  |  |
| --- | --- |
| Nome del metodo | + leggi(String nickname) : ArrayList<IndirizzoBean> |
| Descrizione | Restituisce gli indirizzi dell’utente passato come parametro. |
| Pre - condizione | Nickname != null |
| Post - condizione | Ritorna indirizzi.size() > 0 se il cliente ha degli indirizzi memorizzati || Ritorna indirizzi.size() == 0 se non ci sono indirizzi memorizzati per il cliente |

### Prodotto

### ProdottoModel

|  |  |
| --- | --- |
| Nome del metodo | + doSave(int codice, int numDisp, String nome, String colore, String marca, String descrizione, int peso, double Prezzo, java.sql.Date data, String strumento) : void |
| Descrizione | Salva i prodotti nel database |
| Pre - condizione | codice!=null && numDisp>0 && nome!=null && colore!=null && marca!=null && descrizione!=null && peso>0, prezzo>0, data!=null && strumento!=null |
| Post - condizione | Memorizza i prodotti nel database |

|  |  |
| --- | --- |
| Nome della classe | ProdottoModel |
| Descrizione | Questa classe gestisce l’informazione persistente prodotto nel database |
| Metodi | doSave( int codice, int numDisp, String nome, String colore, String marca, String descrizione, int peso, double Prezzo, java.sql.Date data, String strumento )  generaCodice()  doDelete(int codice)  doRetriveAll()  doRetriveByInstruments(String strumento)  doRetriveByMarca(String marca)  doRetriveByKey(int codice)  doRetriveByName(String nome) |

|  |  |
| --- | --- |
| Nome del metodo | + generaCodice(): int |
| 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 | + aggiorna(ArrayList<ProdottoCatalogo prodotti) : void |
| Descrizione | Metodo che aggiorna i prodotti nel database |
| Pre - condizione | prodotti.size()>0 |
| Post - condizione | I prodotti sono stati aggiornati. |

|  |  |
| --- | --- |
| Nome del metodo | + doDelete(int codice) : void |
| Descrizione | Elimina un prodotto dal database |
| Pre - condizione | Codice != null |
| Post - condizione | Il prodotto è stato eliminato |

|  |  |
| --- | --- |
| Nome del metodo | + doRestriveAll() : ArrayList<ProdottoCatalogoBean> |
| Descrizione | Metodo che restituisce tutti i prodotti presenti nel database nella tabella prodottoCatalogo |
| Pre - condizione | // |
| Post - condizione | Prodotti.size() > 0 se ci sono presenti dei prodotti memorizzati nel database || Prodotti.size() == 0 se non ci sono dei prodotti memorizzati nel database |

|  |  |
| --- | --- |
| Nome del metodo | + doRestriveByInstruments(String strumento) : ArrayList<ProdottoCatalogoBean> |
| Descrizione | Metodo che restituisce i prodotti in base al tipo di strumento |
| Pre - condizione | strumento!=null |
| Post - condizione | Prodotti.size() > 0 se ci sono presenti dei prodotti memorizzati nel database del tipo passato come parametro || Prodotti.size() == 0 se non ci sono dei prodotti memorizzati nel database del tipo passato come parametro |

|  |  |
| --- | --- |
| Nome del metodo | + doRestriveByKey(int codice) : ProdottoCatalogoBean |
| Descrizione | Metodo che restituisce il prodotto in base al codice passato come parametro |
| Pre - condizione | Codice != null |
| Post - condizione | Prodotto != null se è prodotto.getCodice() == codice || Prodotto == null se nel database non ci sono prodotti con quel codice |

|  |  |
| --- | --- |
| Nome del metodo | + doRestriveByMarca(String marca) : ArrayList<ProdottoCatalogoBean> |
| Descrizione | Metodo che restituisce i prodotti in base alla marca passato come parametro |
| Pre - condizione | marca!=null |
| Post - condizione | Prodotti.size() > 0 se ci sono presenti dei prodotti memorizzati nel database della marca passato come parametro || Prodotti.size() == 0 se non ci sono dei prodotti memorizzati nel database della marca passato come parametro |

|  |  |
| --- | --- |
| Nome del metodo | + doRestriveByName(String nome) : ArrayList<ProdottoCatalogoBean> |
| Descrizione | Metodo che restituisce i prodotti in base al nome passato come parametro |
| Pre - condizione | nome!=null |
| Post - condizione | Prodotti.size() > 0 se ci sono presenti dei prodotti memorizzati nel database con il nome passato come parametro || Prodotti.size() == 0 se non ci sono dei prodotti memorizzati nel database con il nome passato come parametro |

### CartaModel

|  |  |
| --- | --- |
| Nome della classe | CartaModel |
| Descrizione | Questa classe gestisce l’informazione persistente carta di credito nel database |
| Metodi | doSave(CartaBean carta, ClienteBean cliente)  leggi(String nickname) |

|  |  |
| --- | --- |
| Nome del metodo | + doSave(CartaBean carte, ClienteBean cliente) : void |
| Descrizione | Questo metodo salva la carta di credito nel database |
| Pre - condizione | carta!=null && cliente!=null |
| Post - condizione | Memorizza la carta di credito nel database |

|  |  |
| --- | --- |
| Nome del metodo | + leggi(String nickname) : ArrayList<CartaBean> |
| Descrizione | Metodo che legge le carte di credito di un determinato utente nel database. |
| Pre - condizione | nickname!=null |
| Post - condizione | carte.size() > 0 se il cliente ha associato delle carte || carte.size() == 0 se non ci sono associate delle carte al cliente |

### COMPOSIZIONE MODEL

|  |  |
| --- | --- |
| Nome della classe | ComposizioneModel |
| Descrizione | Questa classe permette il salvataggio della coppia di chiavi ordine, prodottoOrdine all'interno del database. |
| Metodi | doSave(ArrayList <ProdottoCatalogoBean> prodotti, int codOrdine) |

|  |  |
| --- | --- |
| Nome del metodo | + doSave(ArrayList <ProdottoCatalogoBean> prodotti): int codOrdine |
| Descrizione | Questo metodo permette il salvataggio della coppia di chiavi ordine, prodottoOrdine all'interno del database. |
| Pre - condizione | prodotti!=null && codOrdine!=null |
| Post - condizione | Il sistema memorizza la coppia ordine e prodottoOrdine nel database. |

### 4.1.22 FOTO MODEL

|  |  |
| --- | --- |
| Nome della classe | FotoModel |
| Descrizione | Questa classe permette il salvataggio di un’immagine nel database. |
| Metodi | doSave(String url, int cod)  generaCodice() |

|  |  |
| --- | --- |
| Nome del metodo | + doSave(String url, int cod) |
| Descrizione | Questo metodo permette il salvataggio di un immagine nel database. |
| Pre - condizione | url!=null && cod!=null |
| Post - condizione | Il sistema memorizza il path dell’immagine nel database. |

|  |  |
| --- | --- |
| Nome del metodo | + generaCodice() |
| Descrizione | Questo metodo genera l’id dell’immagine da memorizzare nel database. |
| Pre - condizione | // |
| Post - condizione | Restituisce l’ID. |

### ClienteModel

|  |  |
| --- | --- |
| Nome della classe | ClienteModel |
| Descrizione | Questa classe gestisce l'informazione persistente cliente nel database. |
| Metodi | doSave(ClienteBean cliente)  leggi(String nickname,String pwd) |

|  |  |
| --- | --- |
| Nome del metodo | + doSave(ClienteBean cliente) : void |
| Descrizione | Questo metodo salva il cliente nel database. |
| Pre - condizione | Cliente != null |
| Post - condizione | Memorizza il cliente nel database |

|  |  |
| --- | --- |
| Nome del metodo | + leggi(String nickname, String pwd) : cliente |
| Descrizione | Metodo che restituisce il cliente se presente nel database |
| Pre - condizione | nickname!=null && pwd!=null |
| Post - condizione | cliente != null se è presente nel database || cliente == null se non è presente nel database |

### GestoreOrdineModel

|  |  |
| --- | --- |
| Nome del metodo | + leggi(String nickname,String pwd) : gestoreOrdine |
| Descrizione | Metodo che restituisce il gestore se presente nel database |
| Pre - condizione | Nickname != null && pwd !=null |
| Post - condizione | gestore != null se è presente nel database || gestore == null se non è presente nel database |

|  |  |
| --- | --- |
| Nome della classe | gestoreOrdineModel |
| Descrizione | Questa classe gestisce l'informazione persistente gestore dell'ordine nel database. |
| Metodi | leggi(String nickname,String pwd) |

### ClienteModel

|  |  |
| --- | --- |
| Nome della classe | ClienteModel |
| Descrizione | Questa classe gestisce l'informazione persistente cliente nel database. |
| Metodi | doSave(ClienteBean cliente, String nickname, String e-mail, String nome, String cognome, String pwd)  generaCodice()  leggi(String nickname,String pwd) |

|  |  |
| --- | --- |
| Nome del metodo | + doSave(ClienteBean cliente, String nickname, String e-mail, String nome, String cognome, String pwd) : void |
| Descrizione | Questo metodo salva il cliente nel database. |
| Pre - condizione | nickname!=null && e-mail!=null && nome!=null && cognome!=null && pwd!=null |
| Post - condizione | Memorizza il cliente nel database |

|  |  |
| --- | --- |
| Nome del metodo | + generaCodice () : int |
| 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 | + leggi(String nickname, String pwd) : cliente |
| Descrizione | Metodo che legge il nickname e la password di un determinato cliente nel database |
| Pre - condizione | nickname!=null && pwd!=null |
| Post - condizione | Restituisce l'oggetto cliente |

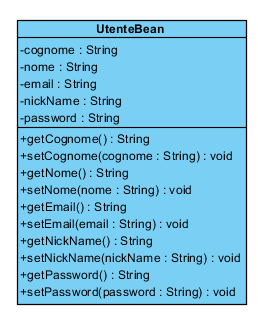
### GestoreOrdineModel

|  |  |
| --- | --- |
| Nome della classe | GestoreOrdineModel |
| Descrizione | Questa classe gestisce l'informazione persistente gestore dell'ordine nel database. |
| Metodi | leggi(String nickname,String pwd) |

|  |  |
| --- | --- |
| Nome del metodo | + leggi(String nickname,String pwd) : gestoreOrdine |
| Descrizione | Metodo che legge nickname e pwd di un determinato gestoreOrdine nel database |
| Pre - condizione | Nickname != null && pwd !=null |
| Post - condizione | Restituisce l'oggetto gestore ordine |

## DESCRIZIONE DELLE CLASSI

### UtenteBean



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

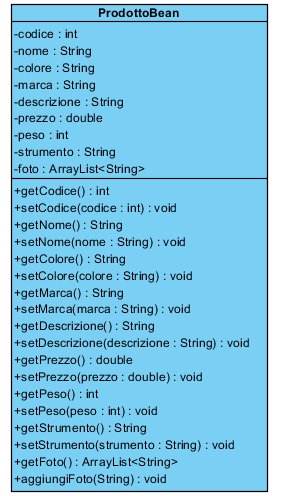
**Password** : la password che utilizza l’utente per l’accesso.

**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.

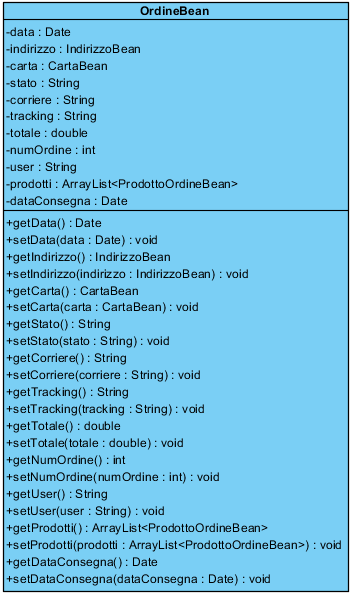
**marca**: Marca del prodotto.

**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.

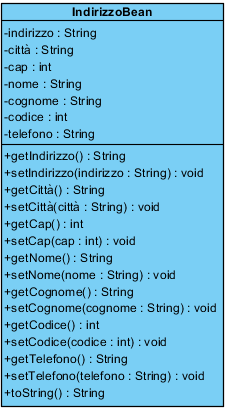
**prodotti:** rappresentano i prodotti di cui è composto l’ordine.

**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

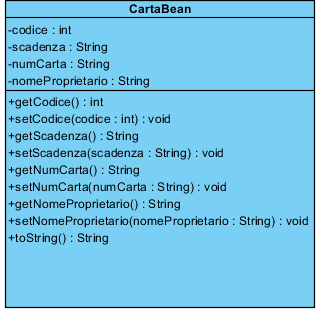
**cap**: Codice di avviamento postale della città dell’indirizzo fornito.

**telefono:** numero di telefono

**nome:** nome del destinatario.

**cognome:** cognome del destinatario.

### CartaBean



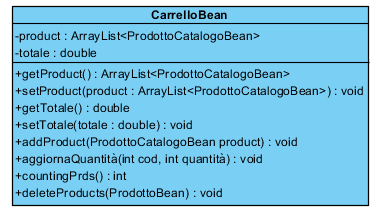
**Codice**: codice che identifica univocamente la carta.

**Scadenza**: scadenza della carta.

**NumeroCarta**: numero della carta prepagata.

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

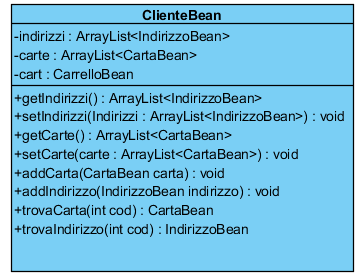
### CarrelloBean



**Prodotti:** prodotti presenti nel carrello.

**Totale:** prezzo totale del carrello.

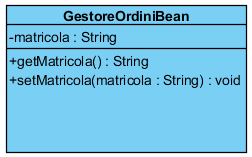
### ClienteBean

  
 **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.

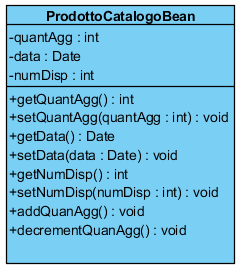
**cart:** rappresenta il carrello di un cliente.

### GestoreOrdiniBean



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

### ProdottoCatalogo

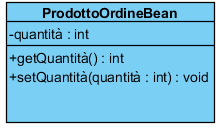


**quantAgg**: rappresenta la quantità di un prodotto nel carrello.

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

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

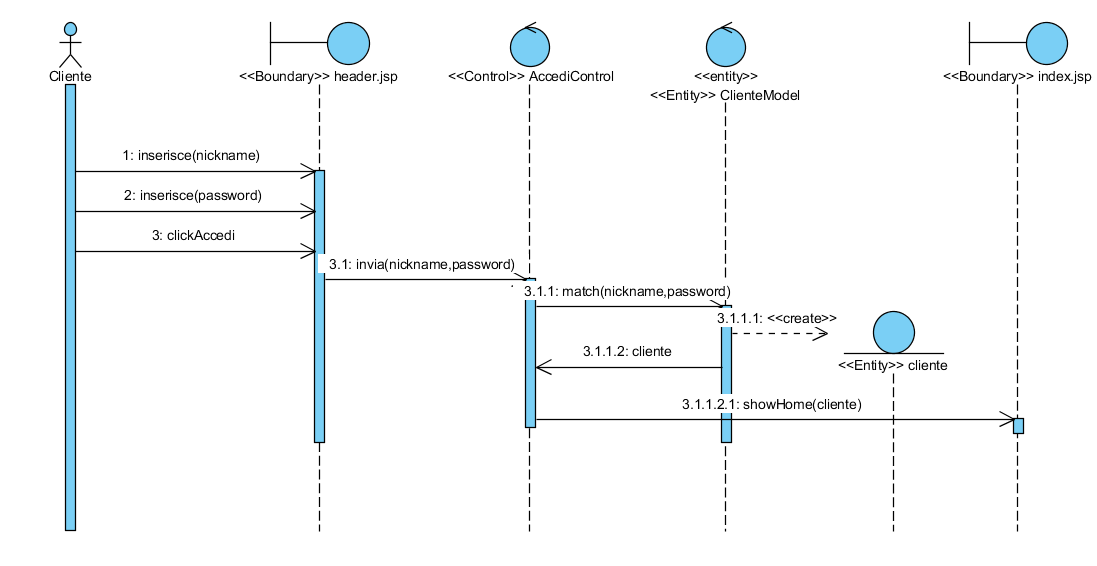
### ProdottoOrdineBean



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

## Sequence Diagram

### Login cliente



**Attore**:

* **Cliente,** che inizia il caso d’uso

**Boundary object**:

* **Header.jsp**, permette di inserire le credenziali per accedere
* **Index.jsp,** è la pagina che permette di visualizzare l’home del sistema.

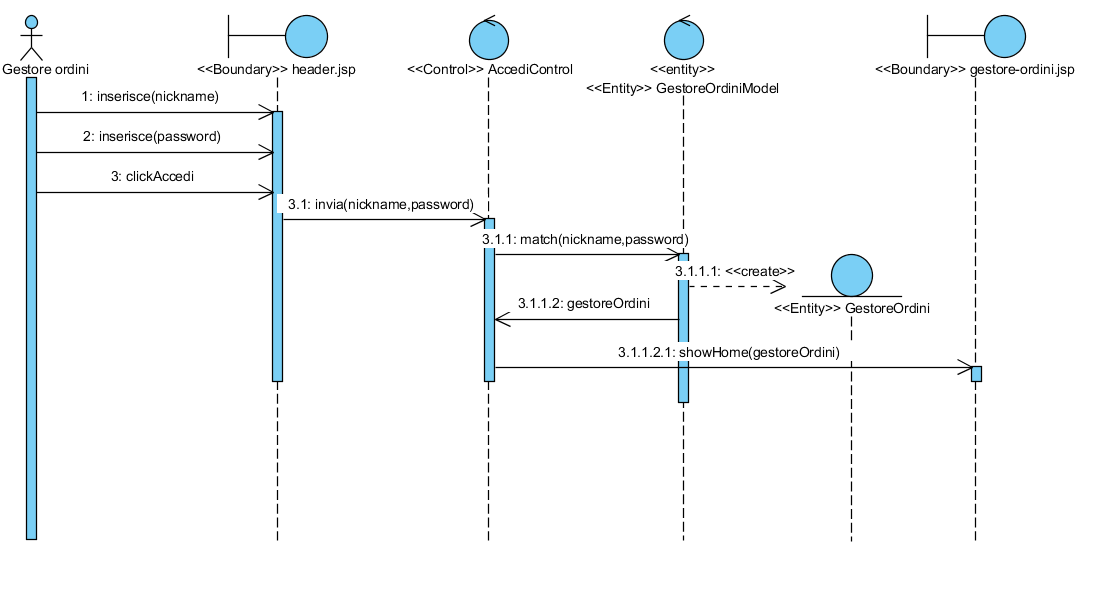
**Control object:**

* **accediControl,** raccoglie dal <<Boundary>> header.jsp e passa le informazioni all’entity ClienteModel, per controllare che le credenziali sono corrette.

**Entity object:**

* **ClienteModel** controlla se il cliente è registrato al sistema.
* **cliente,** rappresenta l’informazione nel sistema relative al cliente

### Login Gestore



**Attore**:

* **Gestore,** che inizia il caso d’uso

**Boundary object**:

* **Header.jsp**, permette di inserire le credenziali per accedere
* **Gestore-ordini.jsp,** è la pagina che permette di visualizzare l’home del gestore ordini.

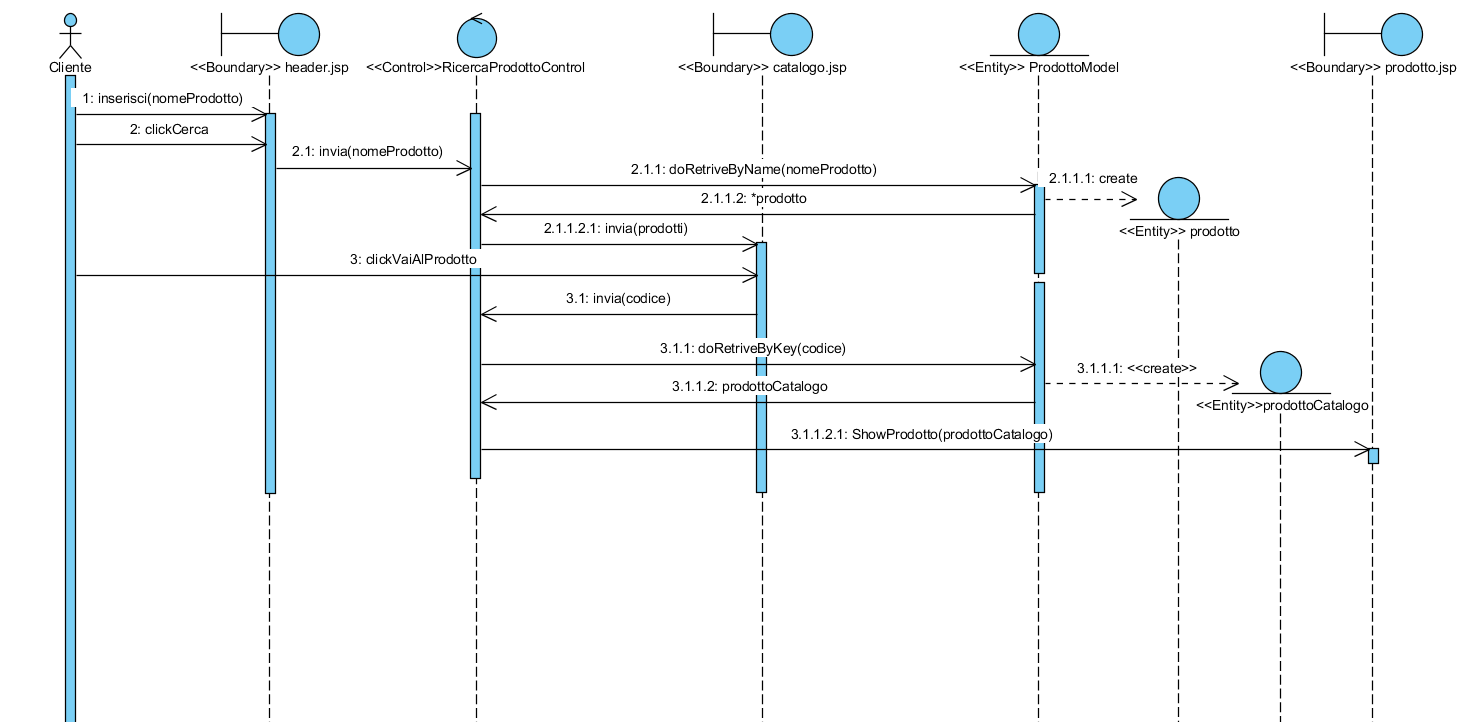
**Control object:**

* **accediControl,** raccoglie dal <<Boundary>> header.jsp e passa le informazioni all’entity GestoreOrdiniModel, per controllare che le credenziali sono corrette.

**Entity object:**

* **GestoreOrdiniModel** controlla se il gestore ordini è registrato al sistema.
* **GestoreOrdini,** rappresenta l’informazione nel sistema relative al gestore ordini

### Ricerca Prodotto



**Attore**:

* **Cliente** che inizia il caso d’uso

**Boundary object**:

* **header.jsp**, permette di inserire il nome del prodotto da ricercare
* **catalogo.jsp**, è la pagina che permette di visualizzare tutti i prodotti risultati dalla ricerca
* **prodotto.jsp**, contiene le informazioni del prodotto selezionato

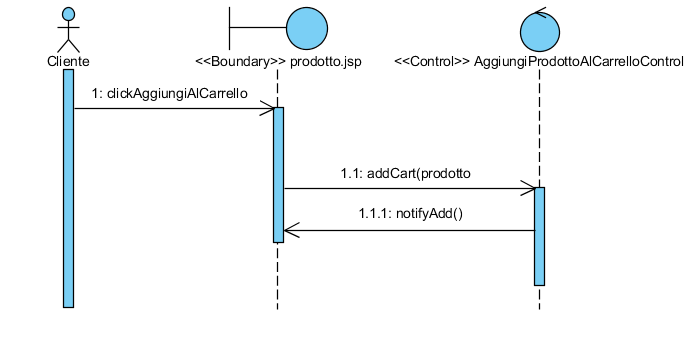
**Control object:**

* **RicercaProdottoControl,** raccoglie dal <<Boundary>> barra delle ricerche il nome del prodotto e passa le informazioni all’entity, da cui riceve la lista dei prodotti. Passa la lista dei prodotti al <<boundary>> catalogo.jsp.

**Entity object:**

* **ProdottoModel** ricerca il prodotto all’interno del database e lo passa al control
* **ProdottoCatalogo,** creata dal ProdottoModel, rappresenta un singolo prodotto del catalogo.

### Aggiungi prodotto al carrello



**Attore**

* **Cliente,** che inizia il caso d’uso

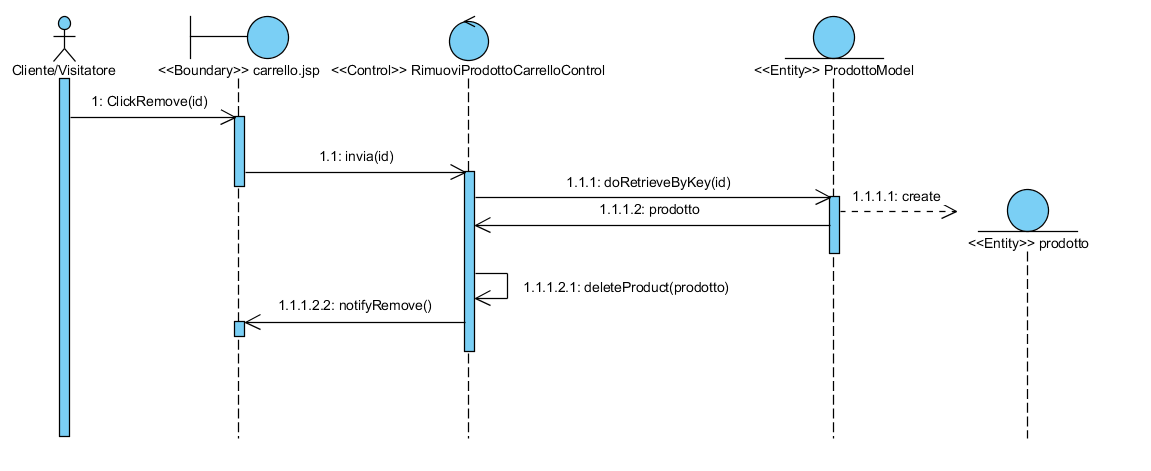
**Boundary object**

* **Prodotto.jsp,** contiente le informazioni del prodotto selezionato e da aggiungere al carrello

**Control object**

* **AggiungiProdottoAlCarrelloControl,** riceve l’oggetto prodotto e lo aggiunge al carrello.

### Rimuovi prodotto dal carrello

****

**Attore**:

* **Cliente** che inizia il caso d’uso

**Boundary object**:

* **Carrello.jsp**, pagina usata per visualizzare i prodotti nel carrello

**Control object:**

* **RimuoviProdottoCarrelloControl,** riceve l’id del prodotto da eliminare dal <<Boundary>> carrello.jsp, lo invia al ProdottoModel, il quale invierà la risposta con il prodotto da eliminare, il control eliminerà il prodotto e notificherà l’avvenuta rimozione

**Entity object:**

* **ProdottoModel** ricerca il prodotto nel database tramite l’id ricevuto dal control
* **prodotto,** rappresenta l’informazione all’interno del sistema relative al prodotto

# GLOSARRIO