0512104198 di Lillo Gianluca
0512104720 Gelsomino Mariano
0512104546 Russo Mattia
0512103592 Somma Alfonso

Object Design Document (ODD)

1. Introduzione

1.1 Object design trade-off

1.1.1 Modularità VS Efficienza

L'ampia modularità descritta nel documento *"Kloudy_SDD"* si scontra con un'efficienza necessaria nelle elaborazioni lato server. Ciò facilita la creazione e la manutenzione del programma (principio del divide et impera) ed, inoltre, aumenta la possibilità di riutilizzare lo stesso codice per altre applicazioni. La scelta di preferire la modularità avvale il sistema di una notevole indipendenza nelle differenti gestioni delle caratteristiche, offrendo una buona manutenibilità pur riducendo l'efficienza dei tempi di risposta dei moduli che si occupano di determinati servizi.

1.1.2 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.

1.1.3 Portabilità VS Efficienza

La portabilità del sistema *Kloudy* 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.

1.2 Interface documentation guidelines

1.2.1 File Java

Ogni file deve contenere:

- Dichiarazione dei package
- Sezione import
- Classi Interne
- Dichiarazione di classe o interfaccia
    - Attributi pubblici
    - Attributi privati
    - Attributi protetti
    - Costruttori
    - Metodi getter e setter
    - Altri metodi

1.2.2 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 minuscola. 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

1.2.3 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)
- E' consigliato l'uso di suffissi standard come "get", "set", "is", "has"

1.3 Definizioni, acronimi e abbreviazioni

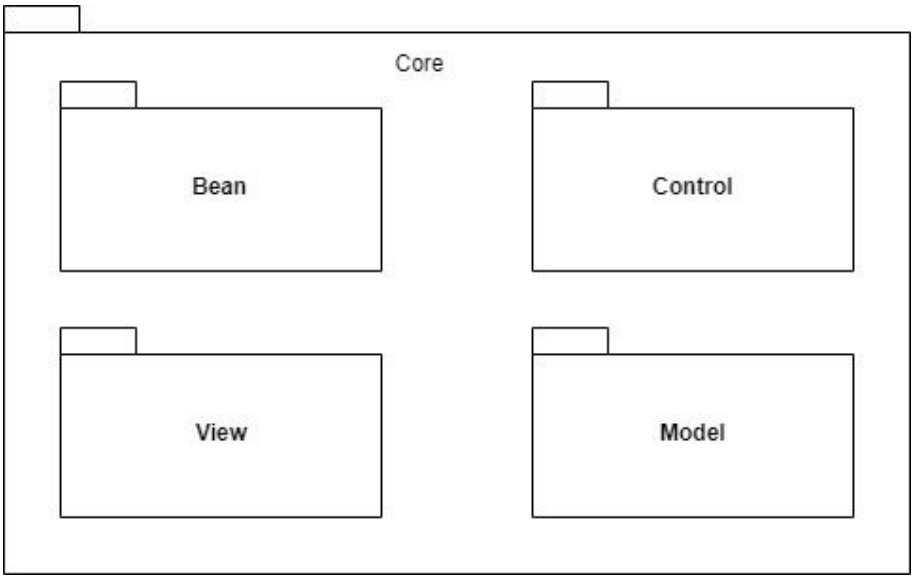
RAD: Requirements Analysis Document.
SDD: System Design Document.
ODD: Object Design Document.

2. Packages

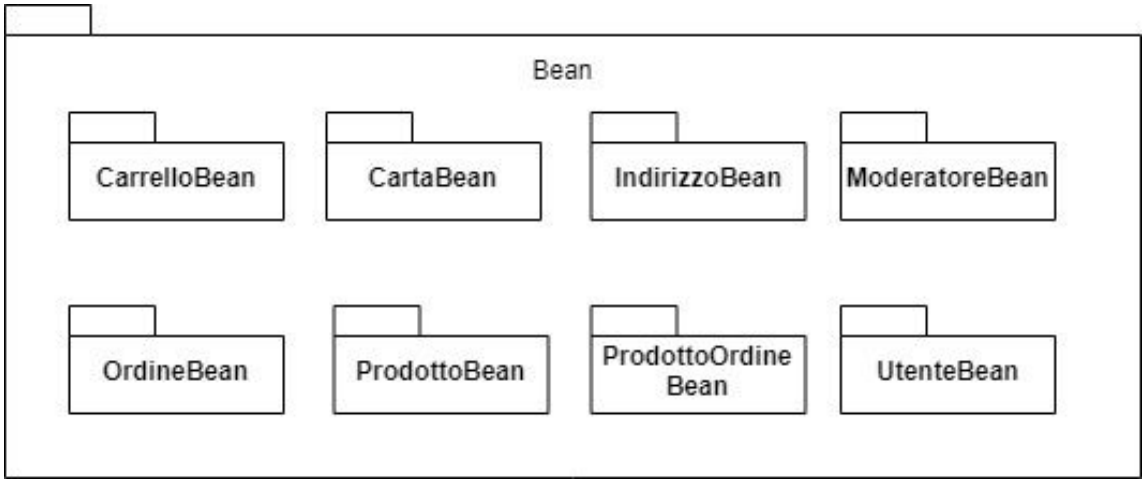
Le componenti base che costituiscono il sistema sono raggruppati in 3 livelli:

- Interface layer
- Application Logic layer
- Storage layer

2.1 Package Core

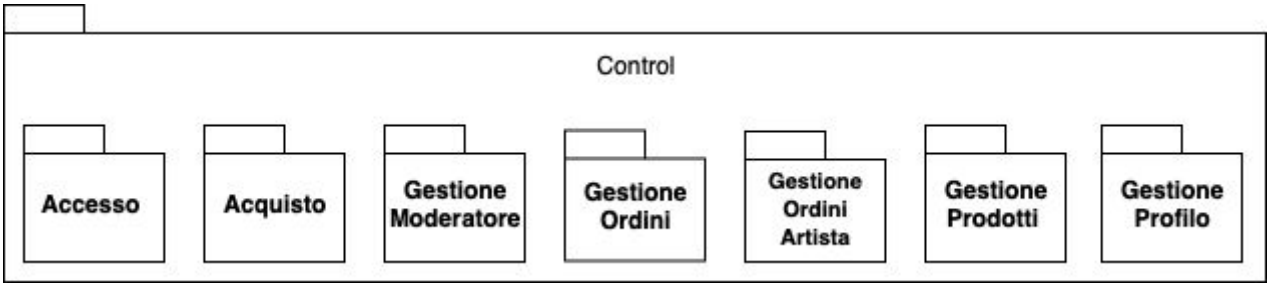


2.2 Package Bean

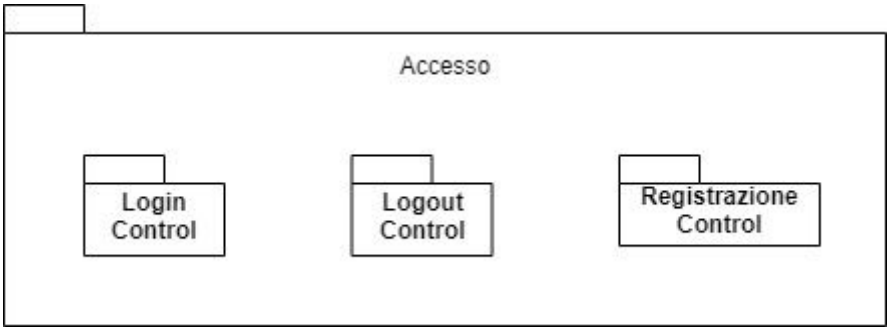


CarrelloBean	**Rappresenta l'oggetto carrello**
CartaBean	**Rappresenta l'oggetto carta**
IndirizzoBean	**Rappresenta l'oggetto indirizzo**
ModeratoreBean	**Rappresenta l'oggetto moderatore**
OrdineBean	**Rappresenta l'oggetto ordine**
ProdottoBean	**Rappresenta l'oggetto prodotto**
ProdottoOrdineBean	**Rappresenta l'oggetto prodotto dell'ordine**
UtenteBean	**Rappresenta l'oggetto utente**

2.3 Package Control

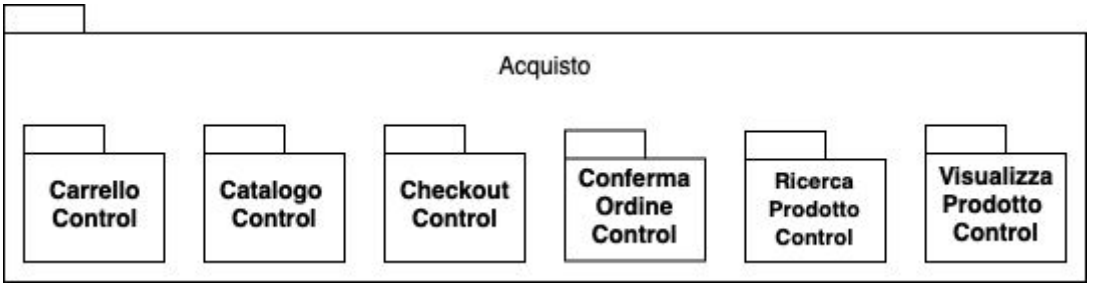


2.3.1 Package Accesso



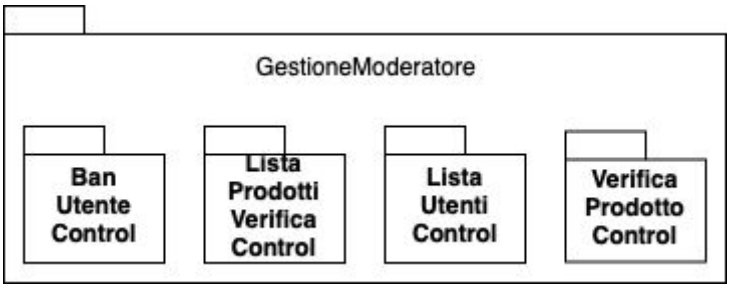
LoginControl	**Permette di effettuare il login al sistema**
LogoutControl	**Permette di effettuare il logout al sistema**
RegistrazioneControl	**Permette di effettuare la registrazione al sistema**

2.3.2 Package Acquisto



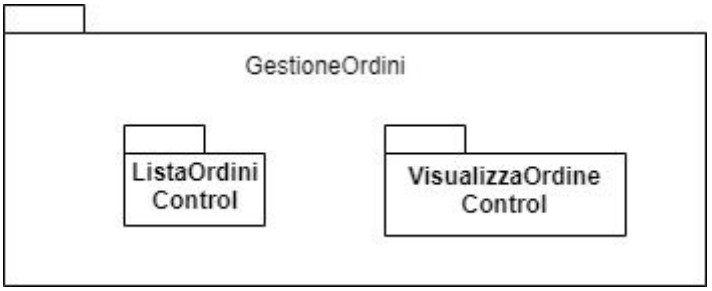
CarrelloControl	**Permette di aggiungere un prodotto al carrello**
CatalogoControl	**Permette di visualizzare il catalogo**
CheckoutControl	**Permette di accedere alla pagina di checkout**
ConfermaOrdineControl	**Permette di conferma l'ordine**
RicercaProdottoControl	**Permette di effettuare la ricerca**
VisualizzaProdottoControl	**Permette di visualizzare i dettagli di un prodotto**

2.3.3 Package Gestione Moderatore



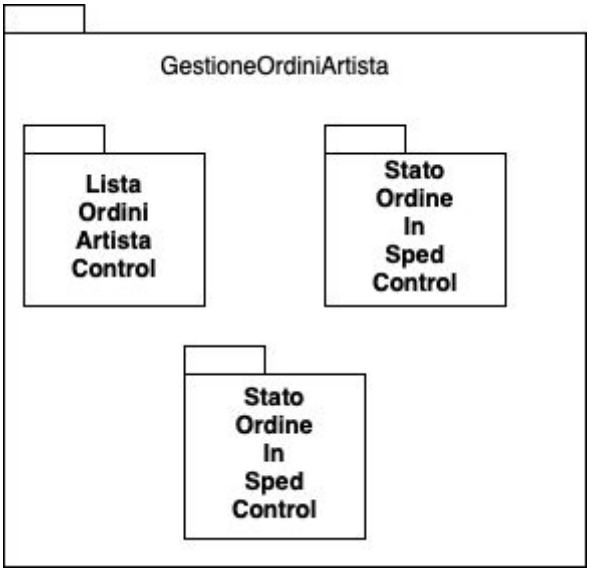
BanUtenteControl	**Permette di bannare un utente**
ListaProdottiVerificaControl	**Permette di visualizzare i prodotti da verificare**
ListaUtentiControl	**Permette di visualizzare la lista degli utenti**
VerificaProdottoControl	**Permette di verificare di confermare o rifiutare un prodotto**

2.3.4 Package Gestione Ordini



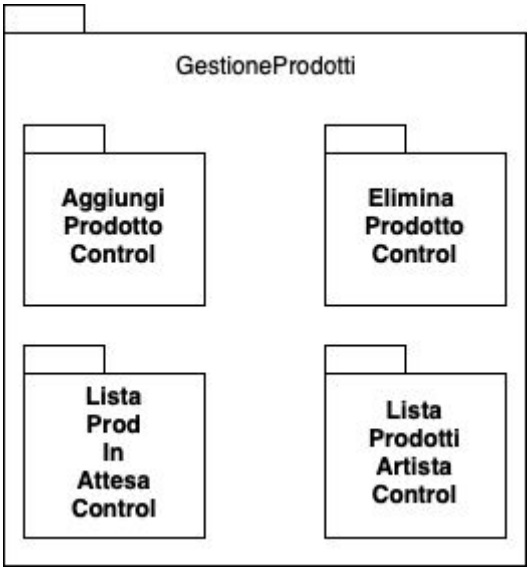
ListaOrdiniControl	**Permette di visualizzare la lista degli ordini**
VisualizzaOrdineControl	**Permette di visualizzare i dettagli di un ordine**

2.3.5 Package Gestione Ordini Artista



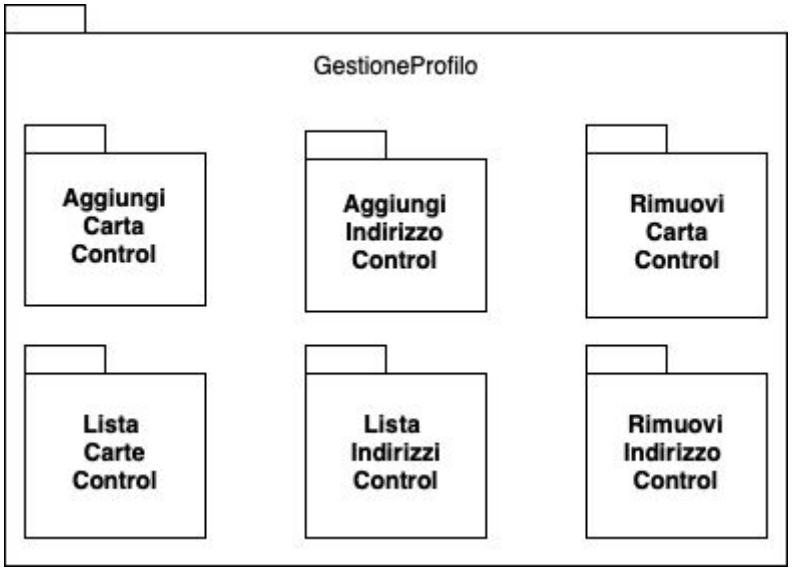
ListaOrdiniArtistaControl	**Permette di visualizzare la lista degli ordini ricevuti di un artista**
StatoOrdineInSpedControl	**Permette di settare lo stato in spedizione ed inserire corriere e tracking**
StatoOrdineSpeditoControl	**Permette di settare lo stato dell'ordine spedito**

2.3.6 Package Gestione Prodotti



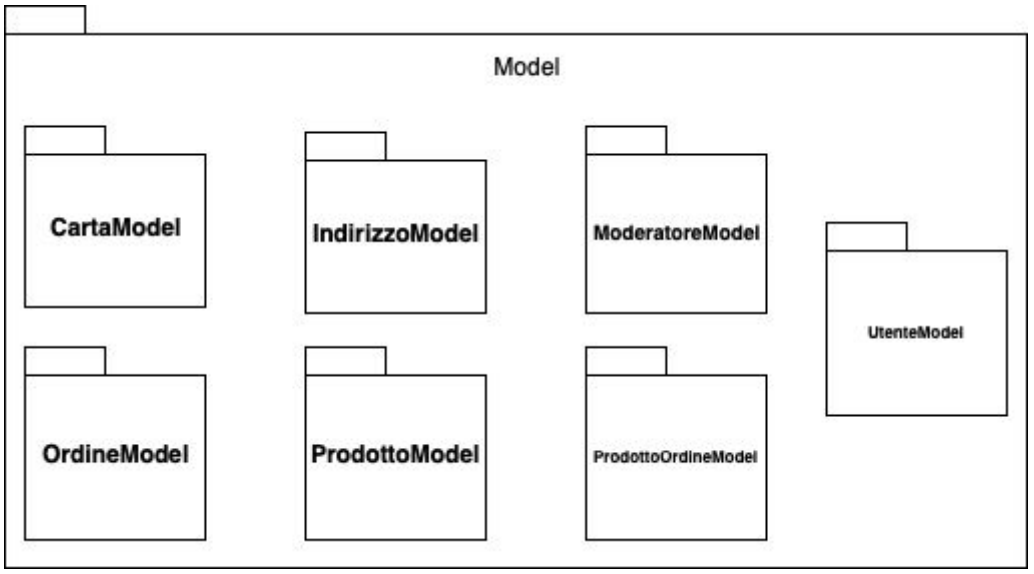
AggiungiProdottoControl	**Permette di aggiungere un prodotto**
ListaProdInAttesaControl	**Permette di visualizzare i prodotti in attesa di verifica**
ListaProdottiArtistaControl	**Permette di visualizzare i prodotti in vendita**
EliminaProdottoControl	**Permette di eliminare un prodotto**

2.3.7 Package Gestione Profilo



AggiungiCartaControl	**Permette di aggiungere una carta**
ListaCarteControl	**Permette di visualizzare le carte**
ListaIndirizziControl	**Permette di visualizzare gli indirizzi**
AggiungiIndirizzoControl	**Permette di modificare le informazioni del profilo**
RimuoviCartaControl	**Permette di rimuovere una carta**
RimuoviIndirizzoControl	**Permette di rimuovere un indirizzo**

2.4 Package Model



ProdottoModel	**Permette di effettuare le query riguardanti il prodotto**
CartaModel	**Permette di effettuare le query riguardanti le carte**
IndirizzoModel	**Permette di effettuare le query riguardanti gli indirizzi**
UtenteModel	**Permette di effettuare le query riguardanti l'utente**
OrdineModel	**Permette di effettuare le query riguardanti gli ordini**
ModeratoreModel	**Permette di effettuare le query riguardanti al moderatore**
UtenteModel	**Permette di effettuare le query riguardanti dell'utente**

3. Interfacce delle classi

3.1.1 - LoginControl

Nome della classe	**LoginControl**
Descrizione	Servlet che si occupa del login dell'utente
Lista 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 (tra cui artista, appassionato e moderatore) utilizzando la coppia di stringhe user/password inserite negli appositi campi di testo.
Pre-Condizioni	request.getParameter("uname") != null && request.getParameter("psw") != null
Post-Condizioni	if(user != null || moderatore != null) l'utente è loggato con successo

3.1.2 - LogoutControl

Nome della classe	**LogoutControl**
Descrizione	Servlet che si occupa del logout dell'utente
Lista Metodi	+doGet(HttpServletRequest request,HttpServletResponse response): void +doPost(HttpServletRequest request, HttpServletResponse response): void
Metodo	**+doGet(HttpServletRequest request, HttpServletResponse response): void**
Descrizione	Questo metodo si occupa di effettuare il logout da parte dell'utente.

Pre-Condizioni	
Post-Condizioni	L'utente/moderatore ha effettuato il logout dal sistema

3.1.3 - ListaOrdiniControl

Nome della classe	**ListaOrdiniControl**
Descrizione	Servlet che si occupa della visualizzazione della lista di ordini effettuati da parte dell'utente
Lista 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 dall'utente
Pre-Condizioni	request.getParameter("loggedIn") != null
Post-Condizioni	L'utente visualizza tutti gli ordini effettuati

3.1.4 - VisualizzaOrdineControl

Nome della classe	**VisualizzaOrdineControl**
Descrizione	Servlet che si occupa della visualizzazione dei dettagli di un ordine e la lista dei prodotti dell'ordine
Lista 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 dei dettagli di un

	ordine e della lista dei prodotti dell'ordine
Pre-Condizioni	request.getParameter("loggedIn") != null && request.getParameter("id") != null
Post-Condizioni	L'utente visualizza i dettagli dell'ordine

3.1.5 - CarrelloControl

Nome della classe	**CarrelloControl**
Descrizione	Servlet che si occupa dell'aggiunta al carrello di un prodotto o della sua rimozione, e della visualizzazione del carrello
Lista Metodi	+doGet(HttpServletRequest request,HttpServletResponse response): void +doPost(HttpServletRequest request, HttpServletResponse response): void
Metodo	**+doGet(HttpServletRequest request, HttpServletResponse response): void**
Descrizione	Questo metodo permette di aggiungere una quantità specifica di un prodotto al carrello o di rimuoverla
Pre-Condizioni	request.getParameter("action") != null && request.getParameter("id") != null && request.getParameter("quantity") != null

Post-Condizioni	if(action.equals("addToCart")
	Se il prodotto è già nel carrello, aumenta la sua quantità, altrimenti aggiunge il nuovo prodotto al carrello.
	if(action.equals("deleteFromCart")
	Se la quantità da eliminare è uguale a quella nel carrello, elimina il prodotto altrimenti diminuisci la quantità del prodotto nel carrello.
	Visualizza i prodotti presenti nel carrello

3.1.6 - CatalogoControl

Nome della classe	**CatalogoControl**

Descrizione	Servlet che si occupa della visualizzazione del catalogo dei prodotti per categoria
Lista 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 del catalogo dei prodotti per categoria
Pre-Condizioni	
Post-Condizioni	L'utente visualizza il catalogo

3.1.7 - CheckoutControl

Nome della classe	**CheckoutControl**
Descrizione	Servlet che si occupa della visualizzazione della pagina di checkout in cui si chiede la selezione della carta e dell'indirizzo per effettuare l'ordine
Lista 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 della pagina di checkout
Pre-Condizioni	user != null && cart != null
Post-Condizioni	L'utente visualizza la pagina di checkout

3.1.8 - ConfermaOrdineControl

Nome della classe	**ConfermaOrdineControl**
Descrizione	Servlet che si occupa di salvare l'ordine nel database
Lista Metodi	+doGet(HttpServletRequest request,HttpServletResponse response): void +doPost(HttpServletRequest request, HttpServletResponse response): void
Metodo	**+doGet(HttpServletRequest request, HttpServletResponse response): void**
Descrizione	Questo metodo permette di salvare l'ordine nel database
Pre-Condizioni	user != null && cart != null && request.getParameter("carta") != null && request.getParameter("indirizzo") != null
Post-Condizioni	L'utente conferma l'ordine e viene reindirizzato alla sezione "I miei ordini"

3.1.9 - RicercaProdottoControl

Nome della classe	**RicercaProdottoControl**
Descrizione	Servlet che si occupa di cercare un prodotto del catalogo per nome dell'artista
Lista Metodi	+doGet(HttpServletRequest request,HttpServletResponse response): void +doPost(HttpServletRequest request, HttpServletResponse response): void
Metodo	**+doGet(HttpServletRequest request, HttpServletResponse response): void**
Descrizione	Questo metodo permette di cercare un prodotto nel catalogo per nome dell'artista

Pre-Condizioni	request.getParameter(“artist”) != null
Post-Condizioni	L’utente visualizza i risultati della ricerca

3.1.10 - VisualizzaProdottoControl

Nome della classe	**VisualizzaProdottoControl**
Descrizione	Servlet che si occupa di visualizzare i dettagli di un singolo prodotto
Lista Metodi	+doGet(HttpServletRequest request,HttpServletResponse response): void +doPost(HttpServletRequest request, HttpServletResponse response): void
Metodo	**+doGet(HttpServletRequest request, HttpServletResponse response): void**
Descrizione	Questo metodo permette di visualizzare i dettagli di un singolo prodotto
Pre-Condizioni	request.getParameter(“id”) != null
Post-Condizioni	L’utente visualizza i dettagli di un singolo prodotto

3.1.11 - ListaProdottiVerificaControl

Nome della classe	**ListaProdottiVerificaControl**
Descrizione	Servlet che si occupa di visualizzare al moderatore la lista di prodotti in attesa della sua verifica
Lista Metodi	+doGet(HttpServletRequest request,HttpServletResponse response): void +doPost(HttpServletRequest request, HttpServletResponse response): void
Metodo	**+doGet(HttpServletRequest request, HttpServletResponse response): void**
Descrizione	Questo metodo permette di visualizzare la lista dei prodotti in

	attesa di verifica
Pre-Condizioni	utente != null && isMod != null
Post-Condizioni	Il moderatore visualizza la lista di prodotti in attesa di verifica

3.1.12 - VerificaProdottoControl

Nome della classe	**VerificaProdottoControl**
Descrizione	Servlet che si occupa di gestire lo stato di verifica di un prodotto
Lista Metodi	+doGet(HttpServletRequest request,HttpServletResponse response): void +doPost(HttpServletRequest request, HttpServletResponse response): void
Metodo	**+doGet(HttpServletRequest request, HttpServletResponse response): void**
Descrizione	Questo metodo permette di gestire lo stato di verifica di un prodotto
Pre-Condizioni	utente != null && isMod != null && request.getParameter(“action”) != null && request.getParameter(“id”) != null
Post-Condizioni	Lo stato del prodotto diventa confermato oppure viene aggiunta la motivazione della non accettazione.

3.1.13 - ListaProdInAttesaControl

Nome della classe	**ListaProdInAttesaControl**
Descrizione	Servlet che si occupa di visualizzare la lista di prodotti in attesa di verifica del moderatore
Lista Metodi	+doGet(HttpServletRequest request,HttpServletResponse response): void +doPost(HttpServletRequest request, HttpServletResponse response): void

Metodo	+doGet(HttpServletRequest request, HttpServletResponse response): void
Descrizione	Questo metodo permette di visualizzare la lista dei prodotti di un artista in attesa di verifica
Pre-Condizioni	user != null && isArtista != null
Post-Condizioni	L'artista visualizza la lista di prodotti in attesa di verifica

3.1.14 - ListaProdottiArtistaControl

Nome della classe	ListaProdottiArtistaControl
Descrizione	Servlet che si occupa di visualizzare all'artista la lista dei suoi prodotti in vendita
Lista Metodi	+doGet(HttpServletRequest request,HttpServletResponse response): void +doPost(HttpServletRequest request, HttpServletResponse response): void
Metodo	**+doGet(HttpServletRequest request, HttpServletResponse response): void**
Descrizione	Questo metodo permette di visualizzare la lista dei prodotti di un artista
Pre-Condizioni	user != null && isArtista != null
Post-Condizioni	L'artista visualizza la lista di prodotti in attesa di verifica

3.1.15 - ListaCarteControl

Nome della classe	ListaCarteControl
Descrizione	Servlet che si occupa di visualizzare la lista delle carte di un utente
Lista Metodi	+doGet(HttpServletRequest request,HttpServletResponse response): void +doPost(HttpServletRequest request, HttpServletResponse

	response): void
Metodo	**+doGet(HttpServletRequest request, HttpServletResponse response): void**
Descrizione	Questo metodo permette di visualizzare la lista delle carte di un utente
Pre-Condizioni	user != null
Post-Condizioni	L'utente visualizza la lista delle sue carte

3.1.16 - ListaIndirizziControl

Nome della classe	**ListaIndirizziControl**
Descrizione	Servlet che si occupa di visualizzare la lista degli indirizzi di un utente
Lista Metodi	+doGet(HttpServletRequest request,HttpServletResponse response): void +doPost(HttpServletRequest request, HttpServletResponse response): void
Metodo	**+doGet(HttpServletRequest request, HttpServletResponse response): void**
Descrizione	Questo metodo permette di visualizzare la lista degli indirizzi di un utente
Pre-Condizioni	user != null
Post-Condizioni	L'utente visualizza la lista dei suoi indirizzi

3.1.17 - RimuoviCartaControl

Nome della classe	**RimuoviCartaControl**
Descrizione	Servlet che si occupa della rimozione di una carta dal database
Lista Metodi	+doGet(HttpServletRequest request,HttpServletResponse response): void

	+doPost(HttpServletRequest request, HttpServletResponse response): void
Metodo	**+doGet(HttpServletRequest request, HttpServletResponse response): void**
Descrizione	Questo metodo permette di eliminare una carta dal database
Pre-Condizioni	user != null && request.getParameter(“id”) != null
Post-Condizioni	La carta è eliminata dal database

3.1.18 - AggiungiCartaControl

Nome della classe	**AggiungiCartaControl**
Descrizione	Servlet che si occupa del login dell’utente
Lista 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 salvare la nuova carta nel database con i dati ottenuti dal rispettivo form
Pre-Condizioni	user != null && request.getParameter("cardnumber") != null && request.getParameter("cardholder") != null && request.getParameter("mesescadenza") != null && request.getParameter("annoscadenza") != null && numeroCarta.matches("[0-9]{16}") && scadenza.matches("[0-9]{1,2}[/]{1}[0-9]{2}") && cvv.matches("[0-9]{3}") && nomeProprietario.matches(“[A-Za-z]{4,35}”)
Post-Condizioni	Carta aggiunta al database

3.1.19 - AggiungiIndirizzoControl

Nome della classe	AggiungiIndirizzoControl
Descrizione	Servlet che si occupa di aggiungere un nuovo indirizzo al database
Lista 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 salvare la nuova carta nel database con i dati ottenuti dal rispettivo form
Pre-Condizioni	user != null && request.getParameter("nome") != null && request.getParameter("cognome") != null && request.getParameter("indirizzo") != null && request.getParameter("telefono") != null && request.getParameter("cap") != null && request.getParameter("citta") != null && request.getParameter("provincia") != null && nome.matches("[A-Za-z]{3,20}") && cognome.matches("[A-Za-z]{3,20}") && indirizzo.matches("[A-Za-z 0-9]{4,40}") && telefono.matches("[0-9]{10}") && cap.matches("[0-9]{5}") && citta.matches("[A-Za-z]{4,40}") && provincia.matches("[A-Za-z]{2,30}"))
Post-Condizioni	Indirizzo aggiunto al database

3.1.20 - RimuoviIndirizzoControl

Nome della classe	RimuoviIndirizzoControl
Descrizione	Servlet che si occupa della rimozione di un indirizzo dal database
Lista Metodi	+doGet(HttpServletRequest request,HttpServletResponse response): void +doPost(HttpServletRequest request, HttpServletResponse

	response): void
Metodo	**+doGet(HttpServletRequest request, HttpServletResponse response): void**
Descrizione	Questo metodo permette di eliminare un indirizzo dal database
Pre-Condizioni	user != null && request.getParameter("id") != null
Post-Condizioni	L'indirizzo è eliminato dal database

3.1.21 - ListaUtentiControl

Nome della classe	**ListaUtentiControl**
Descrizione	Servlet che si occupa della visualizzazione della lista utenti per il moderatore
Lista Metodi	+doGet(HttpServletRequest request,HttpServletResponse response): void +doPost(HttpServletRequest request, HttpServletResponse response): void
Metodo	**+doGet(HttpServletRequest request, HttpServletResponse response): void**
Descrizione	Questo metodo permette di visualizzare la lista utenti
Pre-Condizioni	user != null && isMod != null
Post-Condizioni	La lista degli utenti è visualizzata al moderatore

3.1.22 - BanUtenteControl

Nome della classe	**BanUtenteControl**
Descrizione	Servlet che si occupa della rimozione di un utente dal database
Lista Metodi	+doGet(HttpServletRequest request,HttpServletResponse response): void +doPost(HttpServletRequest request, HttpServletResponse response): void

Metodo	+doGet(HttpServletRequest request, HttpServletResponse response): void
Descrizione	Questo metodo permette di rimuovere un utente dal database
Pre-Condizioni	user != null && isMod != null && request.getParameter(“username”) != null
Post-Condizioni	L’utente con l’username selezionato è rimosso dal database

3.1.23 - StatoOrdineInSpedControl

Nome della classe	StatoOrdineInSpedControl
Descrizione	Servlet che si occupa di cambiare lo stato dell’ordine “In spedizione”
Lista Metodi	+doGet(HttpServletRequest request,HttpServletResponse response): void +doPost(HttpServletRequest request, HttpServletResponse response): void
Metodo	**+doGet(HttpServletRequest request, HttpServletResponse response): void**
Descrizione	Questo metodo permette di cambiare lo stato dell’ordine “In spedizione”
Pre-Condizioni	user != null && isArtista!=null && request.getParameter("id") != null && request.getParameter("corriere") != null && request.getParameter("tracking") != null && corriere.matches("[A-Za-z]{3,30}") && tracking.matches("[A-Z]{4,30}")
Post-Condizioni	L’ordine ha stato “In spedizione” e corriere e tracking inseriti

3.1.24 - StatoOrdineSpeditoControl

Nome della classe	StatoOrdineSpeditoControl
Descrizione	Servlet che si occupa di cambiare lo stato dell’ordine “Spedito”

Lista Metodi	+doGet(HttpServletRequest request,HttpServletResponse response): void +doPost(HttpServletRequest request, HttpServletResponse response): void
Metodo	**+doGet(HttpServletRequest request, HttpServletResponse response): void**
Descrizione	Questo metodo permette di cambiare lo stato dell'ordine "Spedito"
Pre-Condizioni	user != null && isArtista!=null && request.getParameter("id") != null
Post-Condizioni	L'ordine ha stato "Spedito"

3.1.25 - CartaModel

Nome della classe	**CartaModel**
Descrizione	Classe che gestisce l'informazione persistente "carta" nel database
Lista Metodi	+doSave(CartaBean carta): void +doRetrieveAllByUsername(String nickname): ArrayList<CartaBean> +doDeleteByKey(int codice): void
Metodo	**+doSave(CartaBean carta): void**
Descrizione	Questo metodo salva la carta nel database
Pre-Condizioni	carta != null
Post-Condizioni	La carta è salvata nel database

Metodo	**+doRetrieveAllByUsername(String nickname): ArrayList<CartaBean>**
Descrizione	Questo metodo ritorna la lista delle carte di uno specifico utente
Pre-Condizioni	username != null

Post-Condizioni	Ritorna l'ArrayList carte con carte.size()>0 se l'utente ha delle carte memorizzate, carte.size()==0 altrimenti

Metodo	**+doDeleteByKey(int codice): void**
Descrizione	Questo metodo rimuove una carta dal database
Pre-Condizioni	codice != null && codice >= 0
Post-Condizioni	La carta col codice corrispondente è eliminata dal database

3.1.26 - IndirizzoModel

Nome della classe	**IndirizzoModel**
Descrizione	Classe che gestisce l'informazione persistente "indirizzo" nel database
Lista Metodi	+doSave(IndirizzoBean indirizzo): void +doRetrieveAllByUsername(String nickname): ArrayList<IndirizzoBean> +rimuoviIndirizzo(int codice): void
Metodo	**+doSave(IndirizzoBean indirizzo): void**
Descrizione	Questo metodo salva l'indirizzo nel database
Pre-Condizioni	indirizzo != null
Post-Condizioni	L'indirizzo è salvato nel database

Metodo	**+doRetrieveAllByUsername(String nickname): ArrayList<IndirizzoBean>**
Descrizione	Questo metodo ritorna la lista degli indirizzi di uno specifico utente
Pre-Condizioni	username != null
Post-Condizioni	Ritorna l'ArrayList indirizzi con indirizzi.size()>0 se l'utente ha

	degli indirizzi memorizzati, indirizzi..size()==0 altrimenti

Metodo	**+rimuoviIndirizzo(int codice): void**
Descrizione	Questo metodo rimuove un indirizzo dal database
Pre-Condizioni	codice >= 0
Post-Condizioni	L'indirizzo col codice corrispondente è eliminata dal database

3.1.27 - ModeratoreModel

Nome della classe	**ModeratoreModel**
Descrizione	Classe che gestisce l'informazione persistente "moderatore" nel database
Lista Metodi	+doRetrieveByKey(String nickname): ModeratoreBean
Metodo	**+doRetrieveByKey(String nickname): ModeratoreBean**
Descrizione	Questo metodo ritorna un oggetto ModeratoreBean
Pre-Condizioni	nickname != null
Post-Condizioni	Ritorna un oggetto di tipo ModeratoreBean

3.1.28 OrdineModel

Nome della classe	**OrdineModel**
Descrizione	Classe che gestisce l'informazione persistente "ordine" nel database

Lista Metodi	+doSave(Ordinebean order): int +doRetrieveAll(String cliente, String ordinamento): ArrayList<OrdineBean> +doRetrieveByArtist(String artist): ArrayList<OrdineBean> +setInSpedizione(int codice): void +setSpedito (int codice): void
Metodo	**+doSave(Ordinebean order): int**
Descrizione	Questo metodo salva un ordine nel database
Pre-Condizioni	ordine =! null
Post-Condizioni	Ritorna un intero “id”

Metodo	**+doRetrieveAll(String cliente, String ordinamento): ArrayList<OrdineBean>**
Descrizione	Questo metodo estrae gli ordini di un determinato cliente
Pre-Condizioni	cliente != null && ordinamento != null
Post-Condizioni	Ritorna un array con gli ordini del cliente

Metodo	**+doRetrieveByArtist(String artist): ArrayList<OrdineBean>**
Descrizione	Questo metodo estrae gli ordini ricevuti di un artista
Pre-Condizioni	artist != null
Post-Condizioni	Ritorna un array con gli ordini ricevuti di un artista

Metodo	**+setInSpedizione(int ordine, String corriere, String tracking): void**
Descrizione	Questo metodo setta lo stato di un ordine “In Spedizione”
Pre-Condizioni	ordine >=0 && corriere != null && tracking != null
Post-Condizioni	L’ordine ha settato lo stato “In spedizione”, il tracking e il corriere

Metodo	**+setSpedito(int ordine): void**

Descrizione	Questo metodo setta lo stato di un ordine “Spedito”
Pre-Condizioni	ordine >=0
Post-Condizioni	L’ordine ha settato lo stato “Spedito”

3.1.29 ProdottoModel

Nome della classe	**ProdottoModel**
Descrizione	Classe che gestisce l’informazione persistente “prodotto” nel database
Lista Metodi	+doSave(ProdottoBean product):void +doRetrieveByKey(int code):ProdottoBean +doRetrieveByVerificato():ArrayList<ProdottoBean> +doDelete(int code):boolean +doRetrieveAllByType(String type, String order):ArrayList<ProdottoBean> +doRetrieveByUser(String user, String order):ArrayList<ProdottoBean> +searchByArtist(String user, String order):ArrayList<ProdottoBean> +confermaProdotto(int code):void +rifiutaProdotto(int code, String motivazione):void +doRetrieveByArtistVerificato(String user):ArrayList<ProdottoBean>
Metodo	**+doSave(ProdottoBean product):void**
Descrizione	Questo metodo salva le informazioni di un prodotto
Pre-Condizioni	product != null
Post-Condizioni	Salva il prodotto nel database

Metodo	**+doRetrieveByKey(int code):ProdottoBean**
Descrizione	Estrae i prodotti con un determinato codice
Pre-Condizioni	code>=0

Post-Condizioni	Restituisce un ProdottoBean

Metodo	**+doRetrieveByVerificato():ArrayList<ProdottoBean>**
Descrizione	Estrae i prodotti non verificati
Pre-Condizioni	
Post-Condizioni	Restituisce un ArrayList di prodotti in attesa di verifica dal moderatore

Metodo	**+doDelete(int code):void**
Descrizione	Elimina un prodotto con un determinato codice
Pre-Condizioni	code>=0
Post-Condizioni	Elimina il prodotto dal database

Metodo	**+doRetrieveAllByType(String type, String order):ArrayList<ProdottoBean>**
Descrizione	Estrae i prodotti che sono verificati di un determinata categoria
Pre-Condizioni	type !=null && order != null
Post-Condizioni	Restituisce un arrayList di prodottoBean

Metodo	**+doRetrieveByUser(String user, String order):ArrayList<ProdottoBean>**
Descrizione	Estrae i prodotti di un determinato autore
Pre-Condizioni	user !=null && order != null
Post-Condizioni	Restituisce un arrayList di prodottoBean

Metodo	+SearchByArtist(String user, String order):ArrayList<ProdottoBean>
Descrizione	Estrae i prodotti verificati di un determinato autore
Pre-Condizioni	user !=null && order != null
Post-Condizioni	Restituisce un arrayList di prodottoBean

Metodo	+confermaProdotto(int code):void
Descrizione	Imposta verificato un prodotto con un determinato codice
Pre-Condizioni	code>=0
Post-Condizioni	Conferma il prodotto

Metodo	+rifiutaProdotto(int code, String motivazione):void
Descrizione	Imposta non verificato un prodotto con un determinato codice per un determinato motivo
Pre-Condizioni	code>=0 && motivazione !=null
Post-Condizioni	Rifiuta il prodotto

Metodo	+doRetrieveByArtistVerificato(String user):ArrayList<ProdottoBean>
Descrizione	Estrae i prodotti in attesa di verifica di un determinato artista
Pre-Condizioni	user != null
Post-Condizioni	Restituisce un arrayList di prodottoBean

3.1.30 Utente Model

Nome della classe	UtenteModel

Descrizione	Classe che gestisce l'informazione persistente "ordine" nel database
Lista Metodi	+doSave(UtenteBean utente, IndirizzoBean indirizzo): void +doRetrieveByKey(String uname) : UtenteBean +countMail(UtenteBean user): boolean +editPsw(String username, String psw): void + doRetrieveAll(): ArrayList<UtenteBean> +doDelete(String nickname): void
Metodo	**+doSave(UtenteBean utente, IndirizzoBean indirizzo): void**
Descrizione	Salva un utente e un indirizzo correlato
Pre-Condizioni	utente != null && indirizzo != null
Post-Condizioni	L'utente e il rispettivo indirizzo sono stati aggiunti al database

Metodo	**+doRetrieveByKey(String uname) : UtenteBean**
Descrizione	Preleva un utente dato il nickname
Pre-Condizioni	uname != null
Post-Condizioni	L'utente è mostrato a video

Metodo	**+countMail(UtenteBean user): boolean**
Descrizione	Non permette più registrazioni da una stessa mail
Pre-Condizioni	user != null
Post-Condizioni	Restituisce un valore booleano che verifica se l'email è o no presente nel sistema

Metodo	**+ doRetrieveAll(): ArrayList<UtenteBean>**
Descrizione	Permette di prelevare tutti gli utenti
Pre-Condizioni	
Post-Condizioni	Restituisce un array di utenti

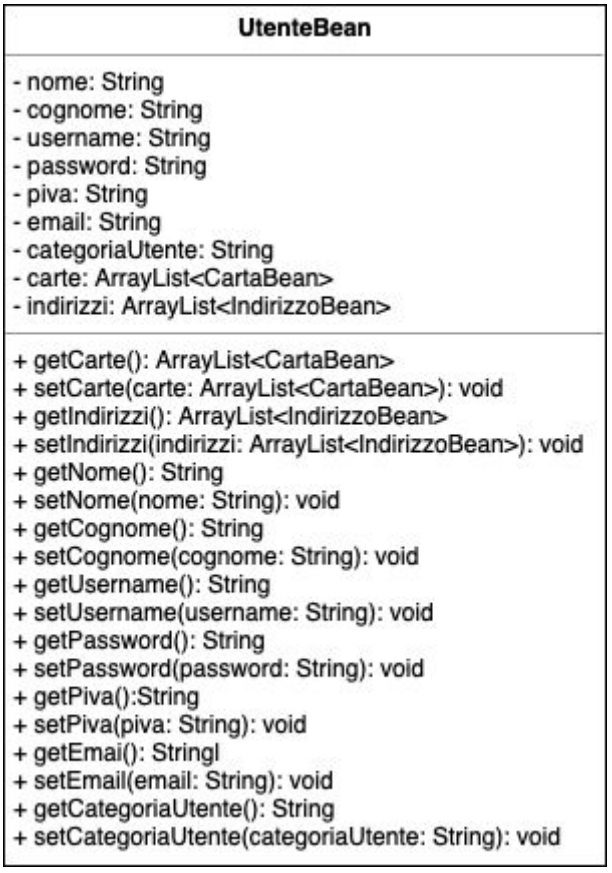
Metodo	+doDelete(String nickname): void
Descrizione	Permette la cancellazione (ban) di un utente
Pre-Condizioni	nickname != null
Post-Condizioni	L'utente è stato cancellato

3.1.31 ProdottoOrdineModel

Nome della classe	ProdottoOrdineModel
Descrizione	Classe che gestisce l'informazione persistente "prodottoordine" nel database
Lista Metodi	+doSave(ProdottoOrdineBean bean, int ordine): void + doRetrieveByOrdine(int ordine): ArrayList<UtenteBean>
Metodo	**+doSave(ProdottoOrdineBean bean, int ordine): void**
Descrizione	Salva un prodotto nel database di uno specifico ordine
Pre-Condizioni	bean != null && ordine >=0
Post-Condizioni	Il prodotto dell'ordine è aggiunto al database

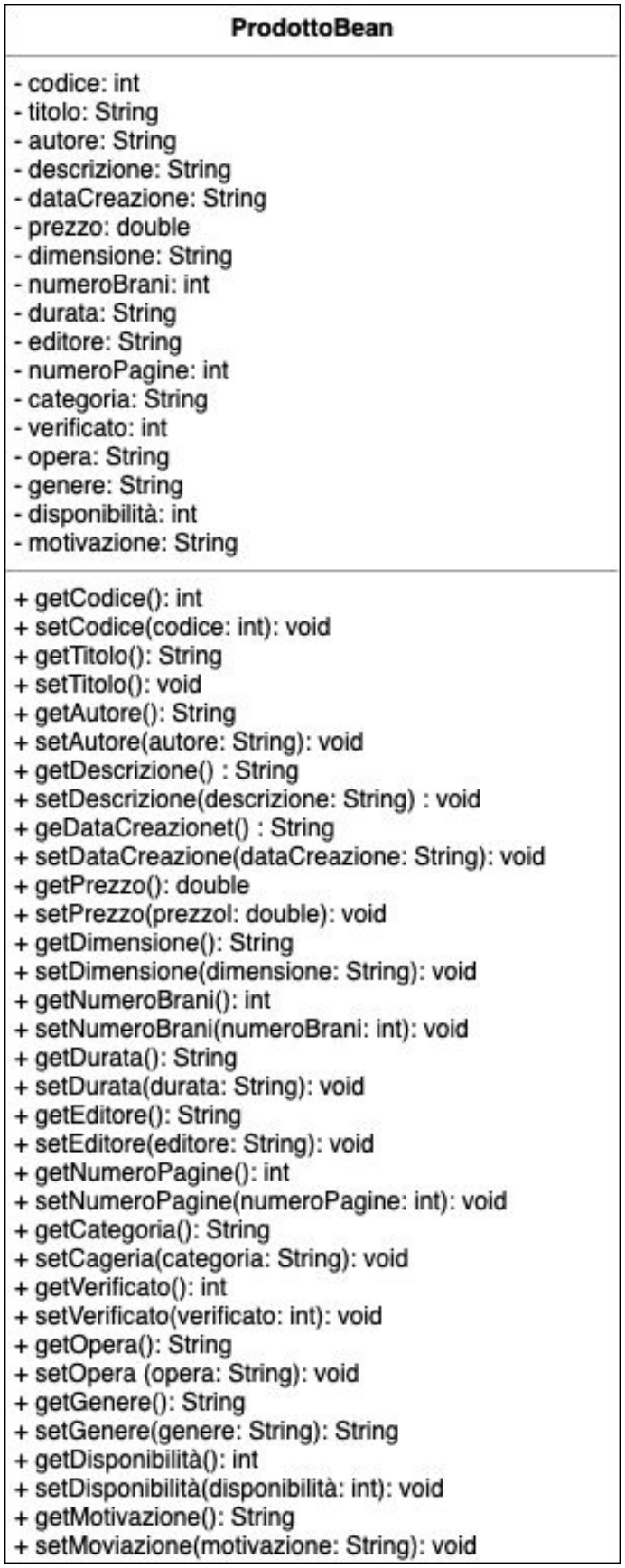
Metodo	+doRetrieveByOrdine(int ordine): ArrayList<ProdottoOrdineBean>
Descrizione	Preleva tutti i prodotti correlati a uno specifico ordine
Pre-Condizioni	ordine >=0
Post-Condizioni	Restituisce una list di ProdottoOrdineBean

4. Descrizione delle classi
4.1 UtenteBean



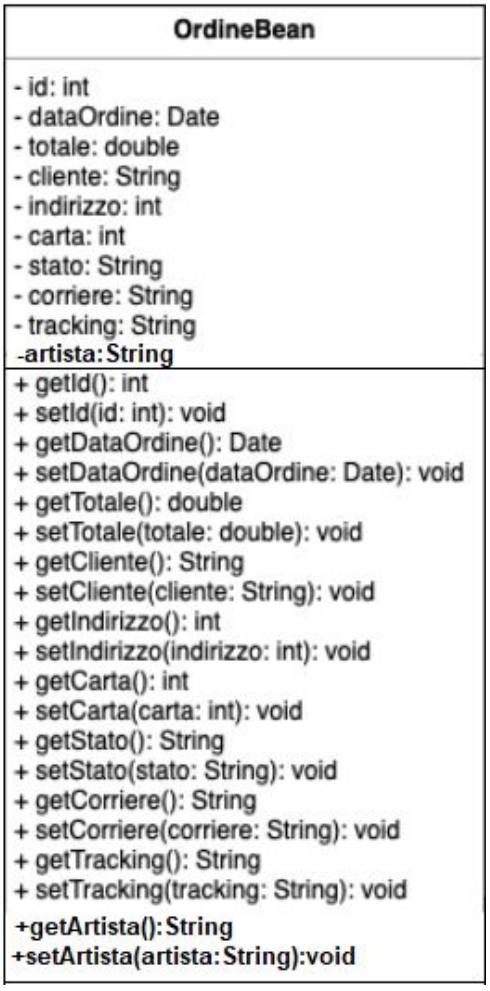
nome: nome dell'utente
cognome: cognome dell'utente
username: username dell'utente
password: password dell'utente
piva: partita iva dell'utente
email: email dell'utente
categoriaUtente: categoria di appartenenza dell'utente
carte: lista delle carte dell'utente
indirizzi: lista degli indirizzi dell'utente

4.2 ProdottoBean



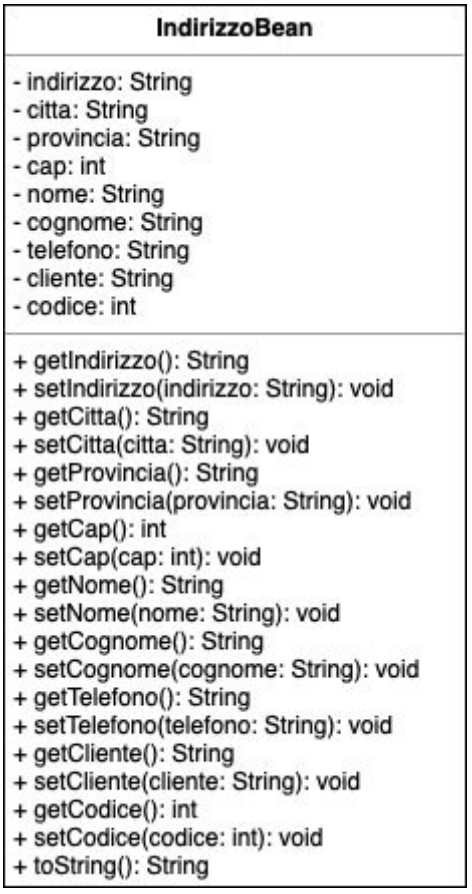
codice: codice identificativo del prodotto
titolo: nome del prodotto
autore: autore del prodotto
descrizione: descrizione del prodotto
dataCreazione: data di creazione del prodotto
prezzo: prezzo del prodotto
dimensione: dimensione (mb per audio e testi e WxH per digital art) del prodotto
numeroBrani: numero brani del prodotto audio
durata: durata del prodotto audio
editore: editore del prodotto testo
numeroPagine: numero pagine del prodotto testo
categoria: categoria del prodotto testo
verificato: prodotto verificato o non verificato dal moderatore
opera: file del prodotto
genere: genere del prodotto testo
disponibilità: disponibilità del prodotto
motivazione: motivazione dal moderatore ottenuta al momento del rifiuto

4.3 OrdineBean



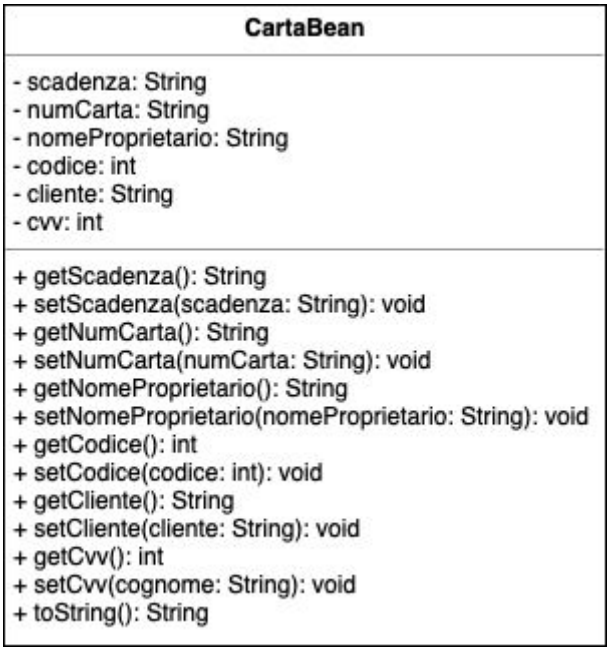
id: id associato all'ordine
dataOrdine: data dell'ordine
totale: importo totale dell'ordine
cliente: cliente che ha effettuato l'ordine
indirizzo: indirizzo di recapito dell'ordine
carta: carta utilizzata per effettuare il pagamento dell'ordine
stato: stato dell'ordine (esempio: "in preparazione","in transito","in consegna")
corriere: corriere delegato per la spedizione dell'ordine
tracking: numero di tracking dell'ordine (per individuare dove esso si trovi)
artista: artista che deve gestire l'ordine

4.4 IndirizzoBean



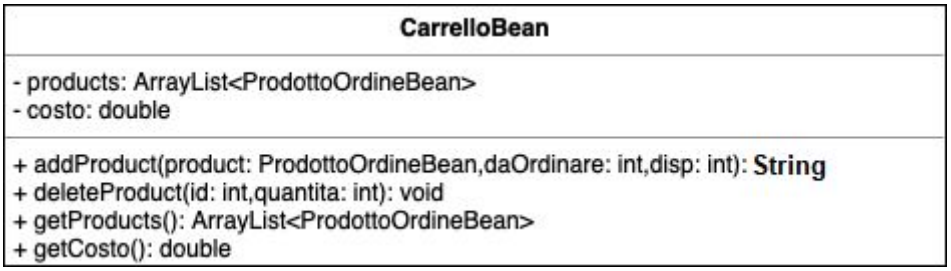
indirizzo: indirizzo previsto per la spedizione
citta: città di recapito della spedizione
provincia: provincia di appartenenza della città di recapito della spedizione
cap: codice di avviamento postale della città dell'indirizzo fornito
nome: nome destinatario
cognome: cognome destinatario
telefono: numero di telefono scelto in caso di eventuale contatto
cliente: utente a cui appartiene l'indirizzo
codice: Id dell'indirizzo, identifica univocamente un indirizzo nel database

4.5 CartaBean



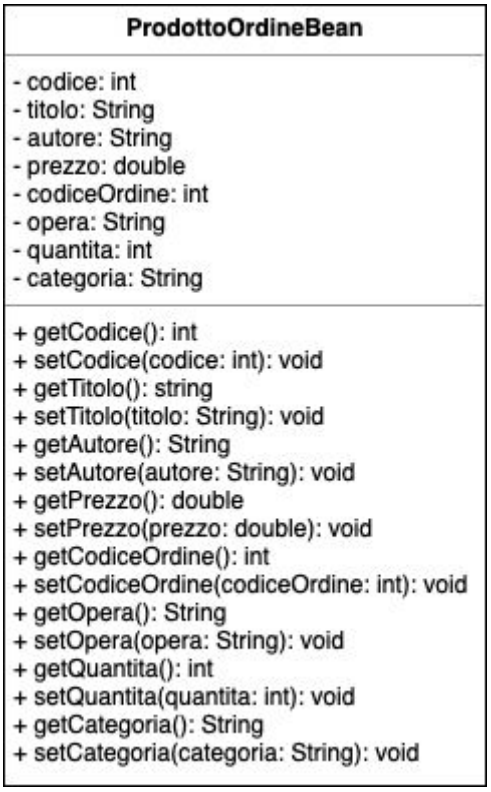
scadenza: mese e giorno indicanti la scadenza della carta
numCarta: numero della carta di credito/prepagata
nomeProprietario: nome e cognome dell'intestatario della carta
codice: codice univoco, identificativo della carta
cliente: utente a cui è associata la carta
cvv: codice di sicurezza impiegato nelle carte di pagamento per mitigarne i rischi di uso fraudolento

4.6 CarrelloBean



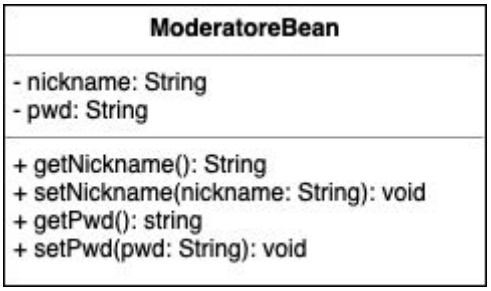
products: lista di prodotti presenti nel carrello
costo: prezzo totale del carrello

4.7 ProdottoOrdineBean



codice: codice identificativo prodotto
titolo: titolo del prodotto
autore: autore del prodotto
prezzo: prezzo del prodotto
codiceOrdine: codice d'ordine del prodotto
opera: file del prodotto
quantita: quantità del prodotto
categoria: categoria di appartenenza del prodotto

4.8 ModeratoreBean



nickname: nickname moderatore
pwd: password moderatore

5. Class Diagram