UNIVERSITÀ DEGLI STUDI DI SALERNO
FACOLTÀ DI SCIENZE MM.FF.NN.
CORSO DI LAUREA TRIENNALE IN INFORMATICA
CORSO DI INGEGNERIA DEL SOFTWARE

Gamerland

OBJECT DESIGN DOCUMENT (O.D.D)

VERSIONE 1.2

Anno Accademico 2017/18

Top Manager:

Professore
Prof. De Lucia Andrea
Prof. Francese Rita

Partecipanti:

Nome	Matricola
Capone Natale	0512103586
Stanzione Roberto	0512103763
Iuliano Gianpaolo	0512103514

Revision History:

Data	Versione	Descrizione	Autore
08/02/2018	1.0	Inizio stesura introduzione	Tutto il team
09/02/2018	1.1	Fine stesura introduzione e packages	Tutto il team
18/02/2018	1.2	Aggiunta interfacce delle classi e stesura finale del documento	Tutto il team

Sommario

1. Introduzione

1.1 Object Design Trade-offs

Dopo la realizzazione dei documenti RAD e SDD abbiamo descritto in linea di massima quello che sarà il nostro sistema e quindi i nostri obiettivi, tralasciando gli aspetti implementativi. Il seguente documento ha lo scopo di produrre un modello capace di integrare in modo coerente e preciso tutte le funzionalità individuate nelle fasi precedenti. In particolare definisce le interfacce delle classi, le operazioni, i tipi, gli argomenti e la signature dei sottosistemi definiti nel System Design. Inoltre sono specificati i trade-off e le linee guida. In allegato al documento abbiamo il javadoc delle varie classi del sistema.

Comprensibilità vs Tempo:

Il codice deve essere quanto più comprensibile possibile per facilitare la fase di testing ed eventuali future modifiche. Il codice sarà quindi accompagnato da commenti ai vari metodi delle classi che ne semplifichino la comprensione. I nomi delle variabili utilizzate sono stati scritti in modo tale da essere facilmente riconducibili al loro utilizzo. Ovviamente questa caratteristica aggiungerà un incremento di tempo allo sviluppo del nostro progetto.

Interfaccia vs Easy-use:

L'interfaccia del sistema sarà costruita in modo tale da risultare semplice all'utilizzo dell'utente. Il team di sviluppo, nel momento di implementare l'interfaccia grafica, non solo cercherà di dare alla pagina uno style piacevole alla vista ma fornire una struttura di interazione tale da rendere semplice all'utente la gestione delle sue operazioni. L'interfaccia grafica fa uso di form e pulsanti disposti in maniera da rendere semplice l'utilizzo del sistema da parte dell'utente finale e i vari utilizzatori del sistema.

Sicurezza vs Efficienza:

La sicurezza, come descritto nei requisiti non funzionali del RAD, rappresenta uno degli aspetti importanti del sistema. Tuttavia, dati i tempi di sviluppo molto limitati, ci limiteremo ad implementare sistemi di sicurezza basati su username e password degli utenti.

Comprensibilità vs Costi:

All'interno del team di sviluppo utilizzeremo uno stile di programmazione ben definito tra tutti i componenti del team. In questo modo, nel caso di aggiunta di nuovi membri allo sviluppo, il tempo di training dei nuovi elementi sarà minimizzato, facendo in modo che possano subito iniziare a svolgere task di sviluppo, minimizzando i costi relativi al training, e velocizzando il tempo di sviluppo.

Tempo di risposta vs Spazio di memoria:

il tempo risposta deve essere minimo. Le operazioni che usano più tempo all'interno del sistema sono gli accessi al database, in quanto, oltre all'accesso al disco bisogna effettuare operazioni di unione e controllo sulle tabelle generate dalle query. Dato che le operazioni effettuate dal sistema spesso risultano nella creazione delle stesse tabelle con gli stessi dati al loro interno, abbiamo rilevato che generando le tabelle necessarie in precedenza, in modo tale che quando viene effettuata un'operazione che ha bisogno di tali dati, non si ha la necessità di generare la tabella a runtime, ma semplicemente si effettua una ricerca in una già generata. Questo porta ad un utilizzo di memoria maggiore all'interno del database ma velocizza di molto le operazioni effettuate.

1.2 Linee Guida per la Documentazione delle Interfacce

Gli sviluppatori dovranno seguire alcune linee guida per la scrittura del codice:

Naming Convention

- E' buona norma utilizzare nomi:
    1. Descrittivi
    2. Pronunciabili
    3. Di uso comune
    4. Lunghezza medio-corta
    5. Utilizzando solo caratteri consentiti (a-z, A-Z, 0-9)

Variabili:

- I nomi delle variabili devono cominciare con una lettera minuscola, e le parole seguenti con la lettera maiuscola. Quest' ultime devono essere dichiarate ad inizio blocco, solamente una per riga e devono essere tutte allineate per facilitare la leggibilità.

  Esempio: concatTipo

- E' inoltre possibile, in alcuni casi, utilizzare il carattere **underscore** "_", ad esempio quando utilizziamo delle variabili costanti oppure come prefisso di variabili d'istanza.

Metodi:

- I nomi dei metodi devono cominciare con una lettera minuscola, e le parole seguenti con la lettera maiuscola. Il nome del metodo tipicamente consiste di un verbo che identifica una azione, seguito dal nome di un oggetto.

  Esempio: i nomi dei metodi per l'accesso e la modifica delle variabili dovranno essere del tipo :

  getId(), setId()

- I commenti dei metodi devono essere raggruppati in base alla loro funzionalità, la descrizione dei metodi deve apparire prima di ogni dichiarazione di metodo, e deve descriverne lo scopo. Deve includere anche informazioni sugli argomenti, sul valore di ritorno, e se applicabile, sulle eccezioni.

Classi:

I nomi delle classi devono cominciare con una lettera maiuscola, e anche le parole seguenti all'interno del nome devono cominciare con una lettera maiuscola. I nomi delle Classi devono fornire informazioni sul loro scopo.

- La dichiarazione di classe caratterizzata da:

  1. Dichiarazione della classe pubblica
  2. Dichiarazioni di costanti
  3. Dichiarazioni di variabili di classe
  4. Dichiarazioni di variabili d'istanza
  5. Costruttori
  6. Commento e dichiarazione metodi.

1.3 Definizioni, acronimi e abbreviazioni

Acronimi:

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

Abbreviazioni:

- DB: DataBase

1.4 Riferimenti

- B. Bruegge, A. H. Dutoit, Object Oriented Software Engineering - Using UML, Pattern and Java, Prentice Hall, 3rd edition, 2009
- Documento SDD del progetto Gamerland
- Documento RAD del progetto Gamerland

2. Packages

La gestione del nostro sistema è suddivisa in tre livelli MVC (Model Control View):

- Model layer
- Control layer
- View layer

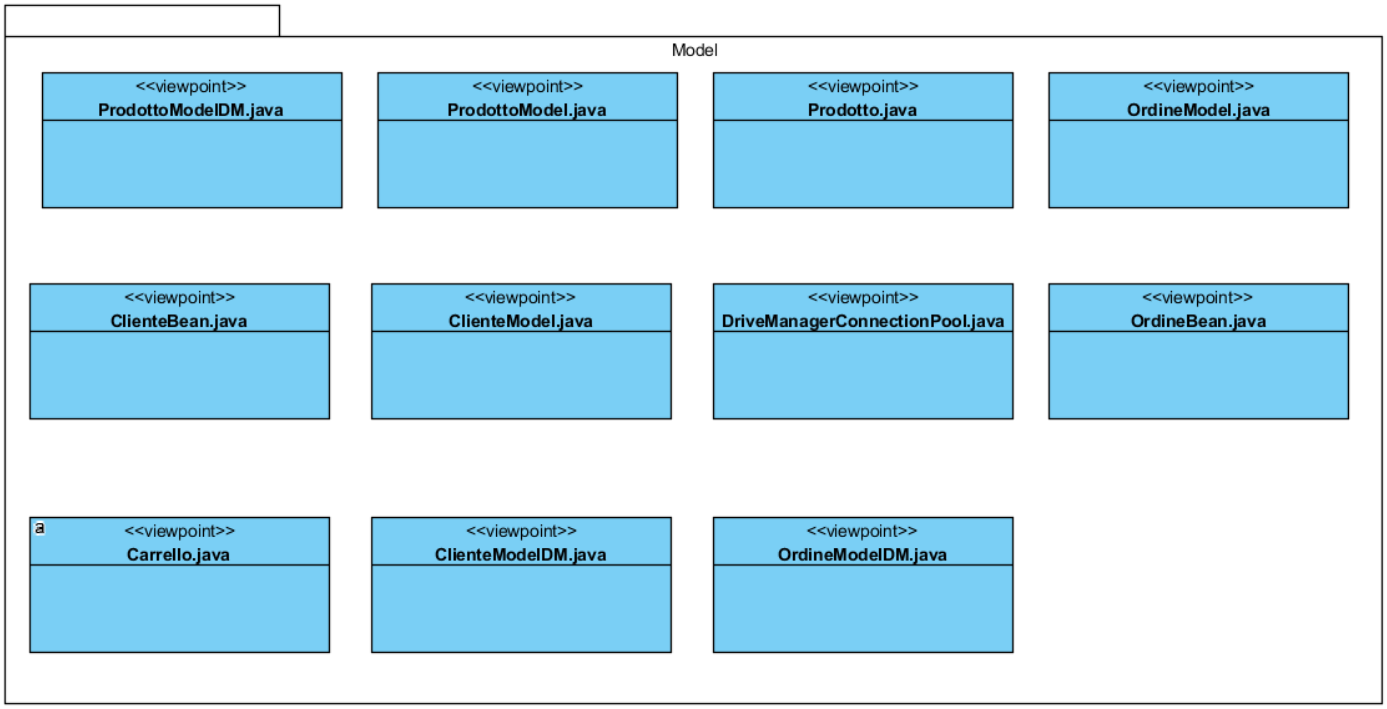
Il package del progetto contiene sottopackage che a loro volta inglobano classi atte alla gestione delle richieste utente. Le classi contenute nel package svolgono il ruolo di gestore logico del sistema.

View layer	Rappresenta l'interfaccia del sistema, ed offre la possibilità all'utente di interagire con quest'ultimo, offrendo sia la possibilità di inviare, in input, che di visualizzare, in output, dati.

Control layer	Ha il compito di elaborare i dati da inviare al client, e spesso grazie a delle richieste fatte al database, tramite il Model Layer, accede ai dati persistenti. Si occupa di varie gestioni quali: **1. Gestione Cataloghi** **2. Gestione Autenticazione** **3. Gestione Magazzino** **4. Gestione Profilo** **5. Gestione Registrazione** **6. Gestione Aggiunta e Acquisto prodotti**

Model layer	Ha il compito di memorizzare i dati sensibili del sistema, utilizzando un DBMS, inoltre riceve le varie richieste dal Control layer inoltrandole al DBMS e restituendo i dati richiesti.

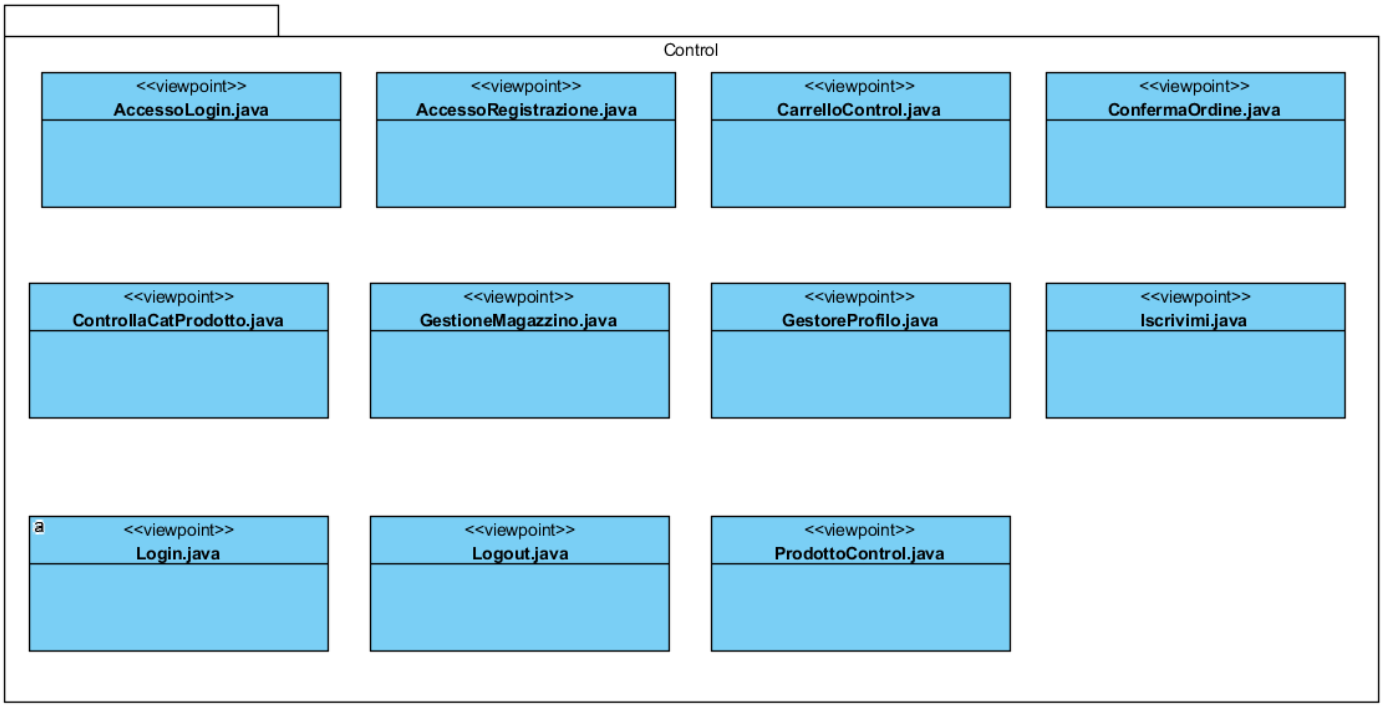
2.1 Package Model



Classe:	**Descrizione:**
ProdottoModelDM.java	Il model che effettua tutte le query riguardanti la gestione dei prodotti nel Database.
ProdottoModel.java	Il model che Descrive L'interfaccia implementata da ProdottoModelDM.java.
Prodotto.java	Descrive un Prodotto.
OrdineModel.java	Il model che descrive L'interfaccia implementata da OrdineModelDM.java.

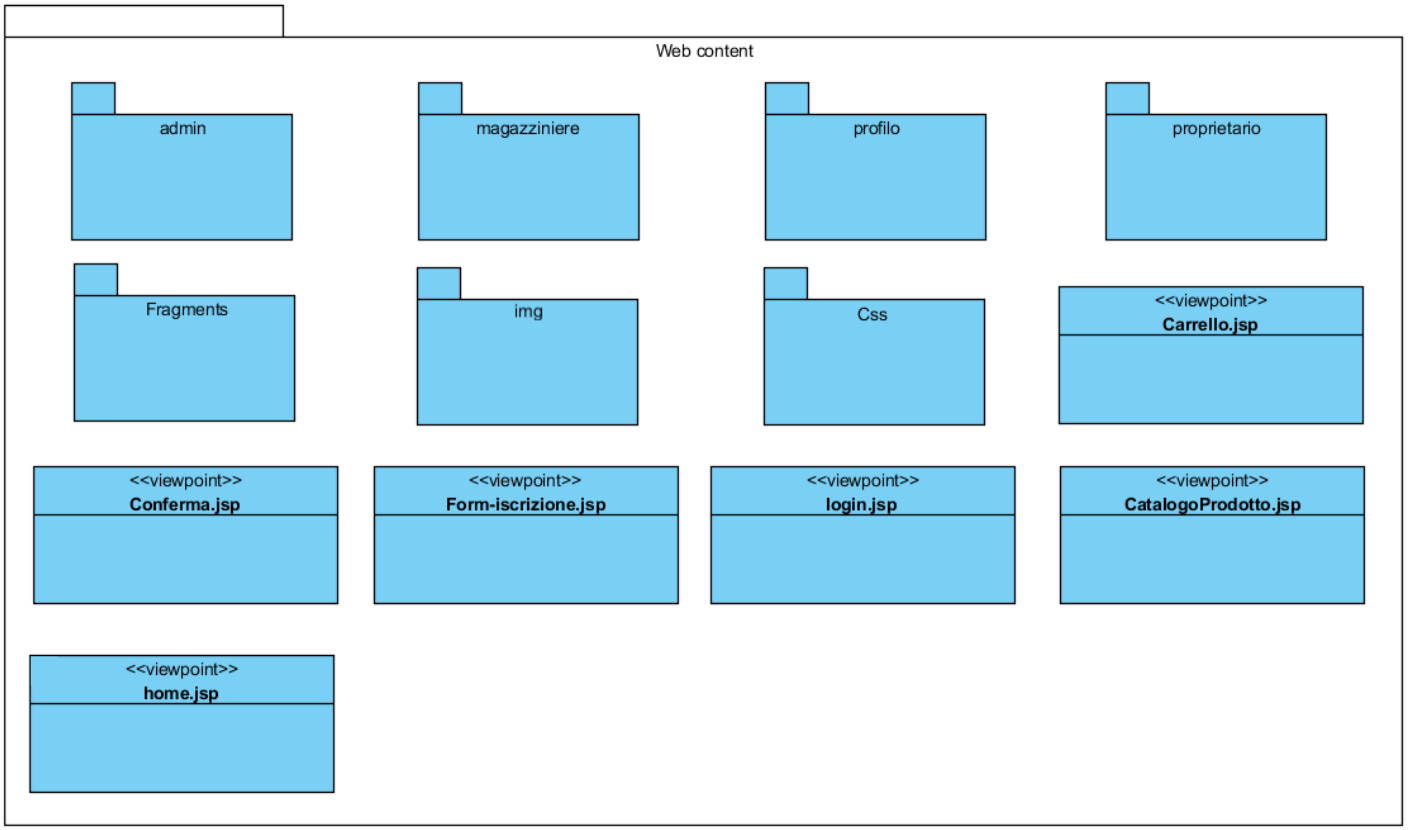
ClienteBean.java	Descrive un Utente del sistema.
ClienteModel.java	Il model che descrive L'interfaccia implementata da ClienteModelDM.java.
ClienteModelDM.java	Il model che effettua tutte le query riguardanti la gestione degli Utenti nel Database.
DriveManagerConnectionPool.java	Il model che si occupa della gestione delle connessioni al database
OrdineBean.java	Descrive un ordine.
Carrello.java	Descrive il carrello.
OrdineModelDM.java	Il model che effettua tutte le query riguardanti la gestione degli ordini nel Database.

2.2 Package Control

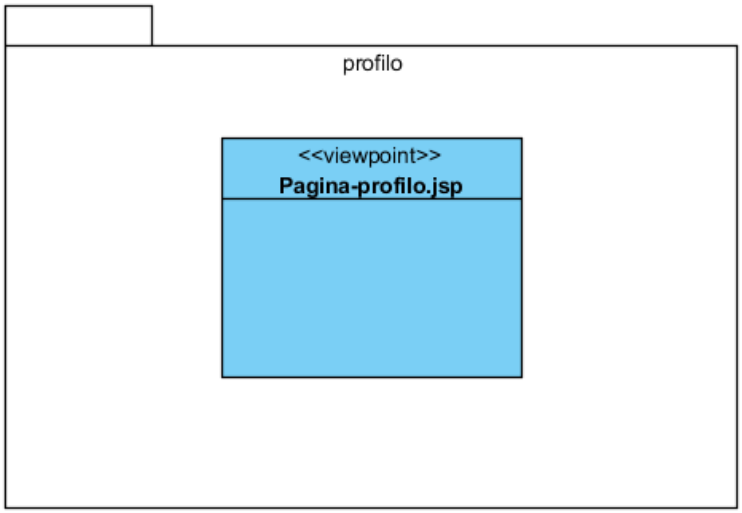


Classe:	**Descrizione:**
AccesoLogin.java	Il controller che gestisce il reindirizzamento alla pagina di login.
AccessoRegistrazione.java	Il controller che gestisce il reindirizzamento alla pagina di iscrizione.
CarrelloControl.java	Il controller che gestisce i prodotti nel carrello e reindirizza alla pagina di conferma ordine.
ConfermaOrdine.java	Il controller che gestisce il calcolo dell'importo dei prodotti nel carrello e la creazione di un ordine.
ControllaCatProdotto.java	Il controller che gestisce l'aggiunta e la rimozione dei prodotti nel catalogo.
GestioneMagazzino.java	Il controller che gestisce gli ordini non spediti.
GestoreProfilo.java	Il controller che gestisce e visualizza gli ordini effettuati dal cliente, modifica password, modifica indirizzo e cancellazione account.
Iscrivimi.java	Il controller che gestisce l'iscrizione di un cliente al sistema.
Login.java	Il controller che gestisce i controlli relativi al login e il reindirizzamento, di un utente, alla propria area di competenza.
Logout.java	Il controller che gestisce il logout di un utente dal sistema.
ProdottoControl.java	Il controller che gestisce l'aggiunta e la rimozione dei prodotti al carrello.

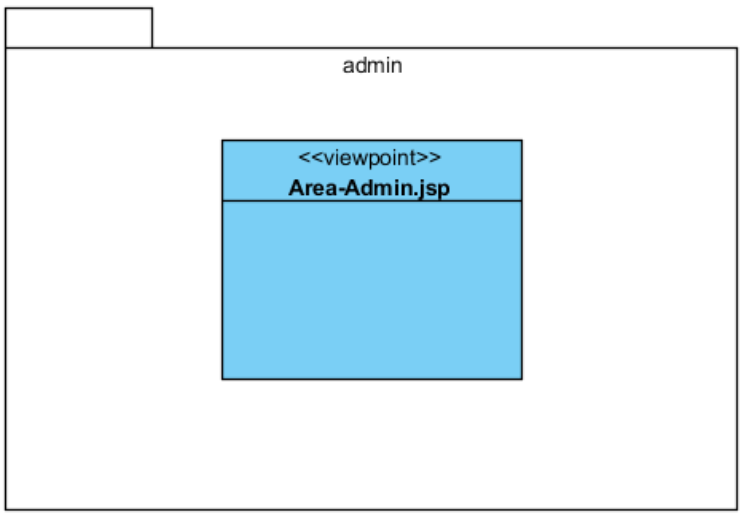
3. Folder View layer



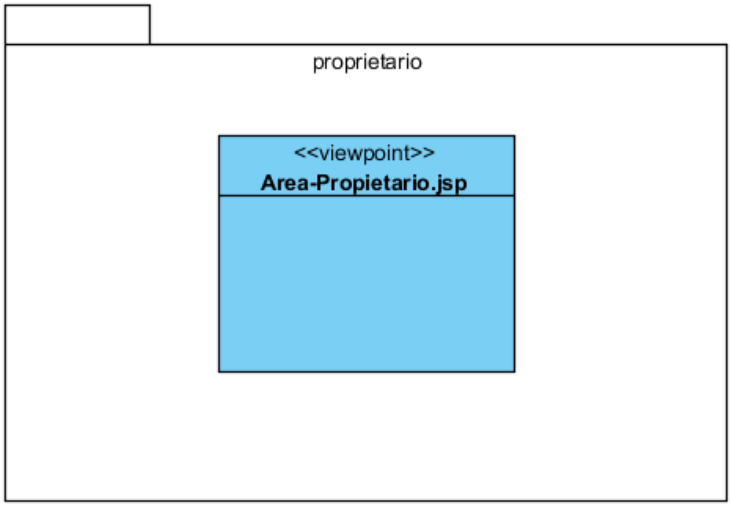
Classe:	**Descrizione:**
Conferma.jsp	La view che consente al cliente di confermare un acquisto.
Form-iscrizione.jsp	La view che consente al cliente di iscriversi al sistema.
Login.jsp	La view che consente all'utente di loggarsi al sistema.
Carrello.jsp	La view che consente al cliente di visualizzare il carrello.
CatalogoProdotto.jsp	La view che consente al cliente di visualizzare i cataloghi con i rispettivi prodotti.
home.jsp	La view che consente all'utente di visualizzare la homepage.



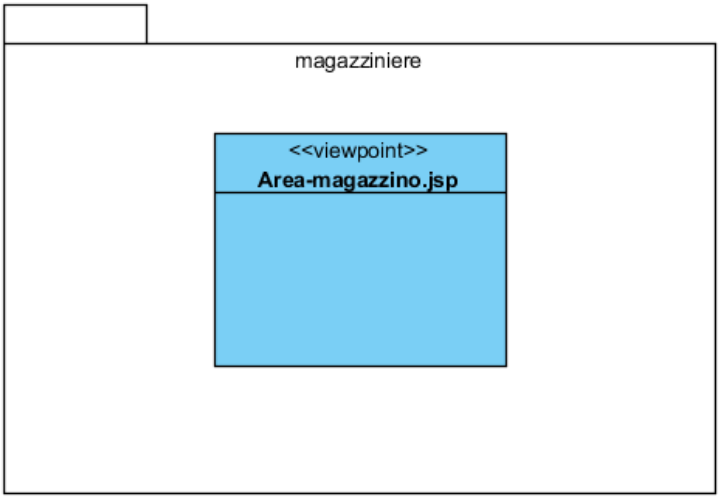
Classe:	**Descrizione:**
Pagina-profilo.jsp	La view che consente al cliente di visualizzare il suo profilo.



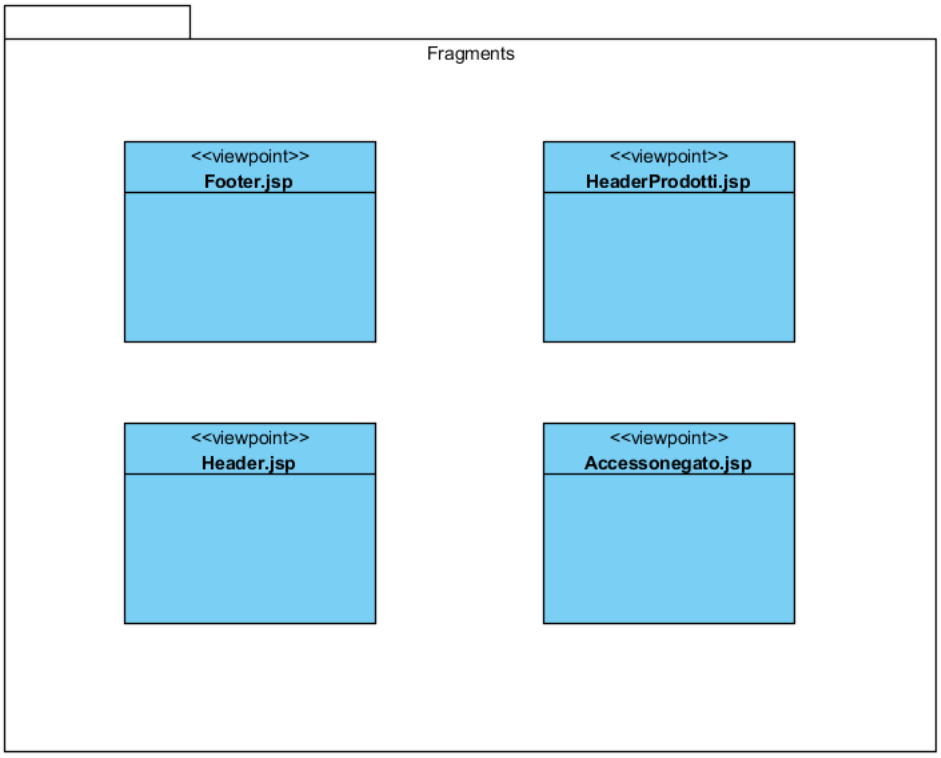
Classe:	**Descrizione:**
Area-Admin.jsp	La view che consente all'amministratore di aggiungere prodotti ai cataloghi.



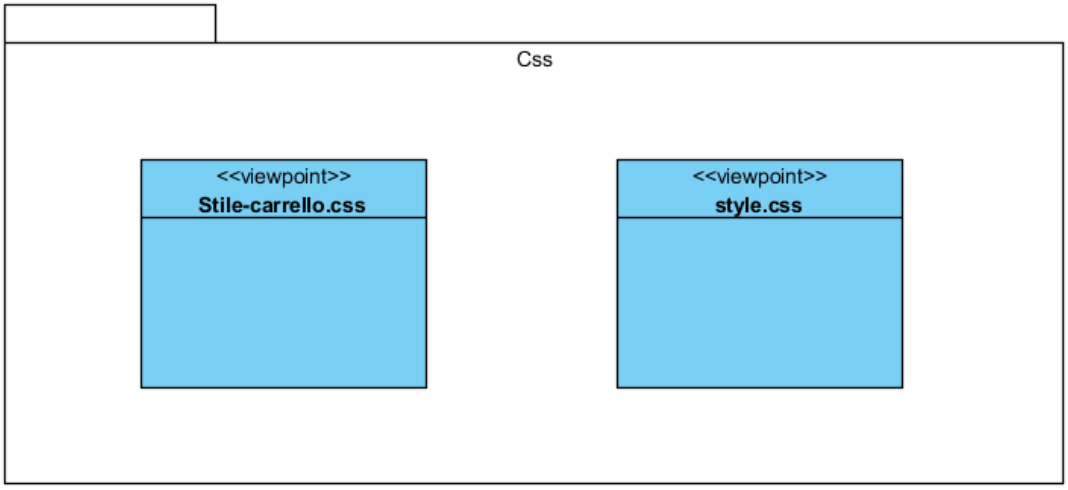
Classe:	**Descrizione:**
Area-Proprietario.jsp	La view che consente al proprietario di visualizzare i ricavi dell'attività.



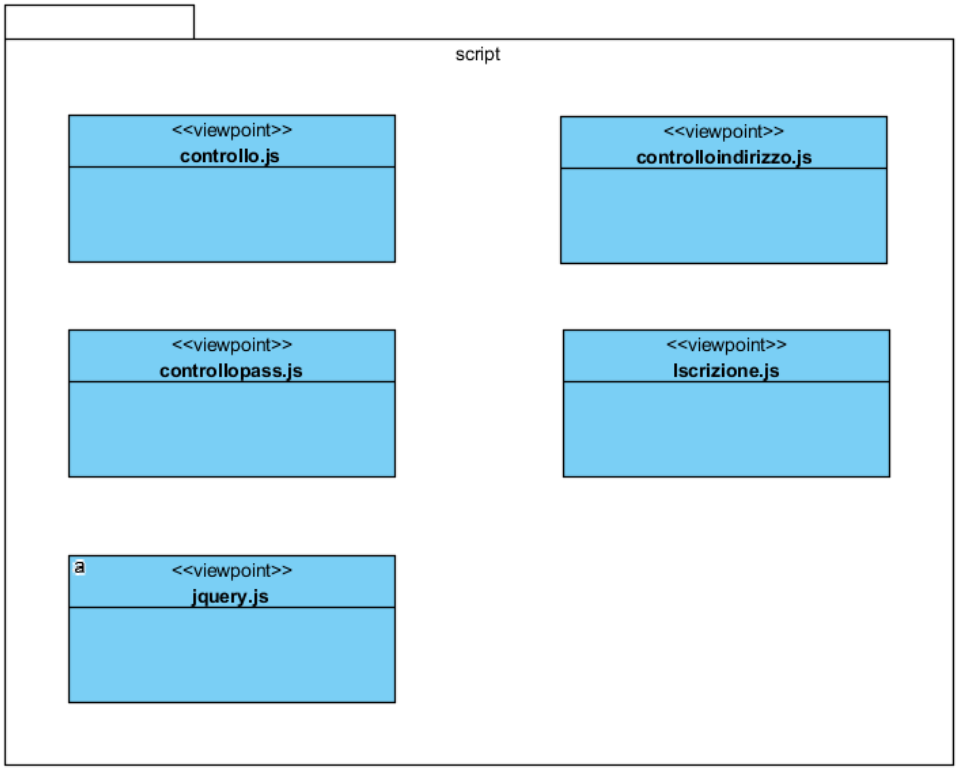
Classe:	**Descrizione:**
Area-magazzino.jsp	La view che consente al magazziniere di gestire gli ordini non ancora spediti dei clienti.



Classe:	**Descrizione:**
Footer.jsp	La view che contiene il Footer delle pagine visualizzate.
HeaderProdotti.jsp	La view che contiene l'Header della pagina CatalogoProdotto.
Header.jsp	La view che contiene l'header delle pagine visualizzate.
Accessonegato.jsp	La view che viene visualizzata quando si tenda di accedere all'area-admin senza autorizzazione.



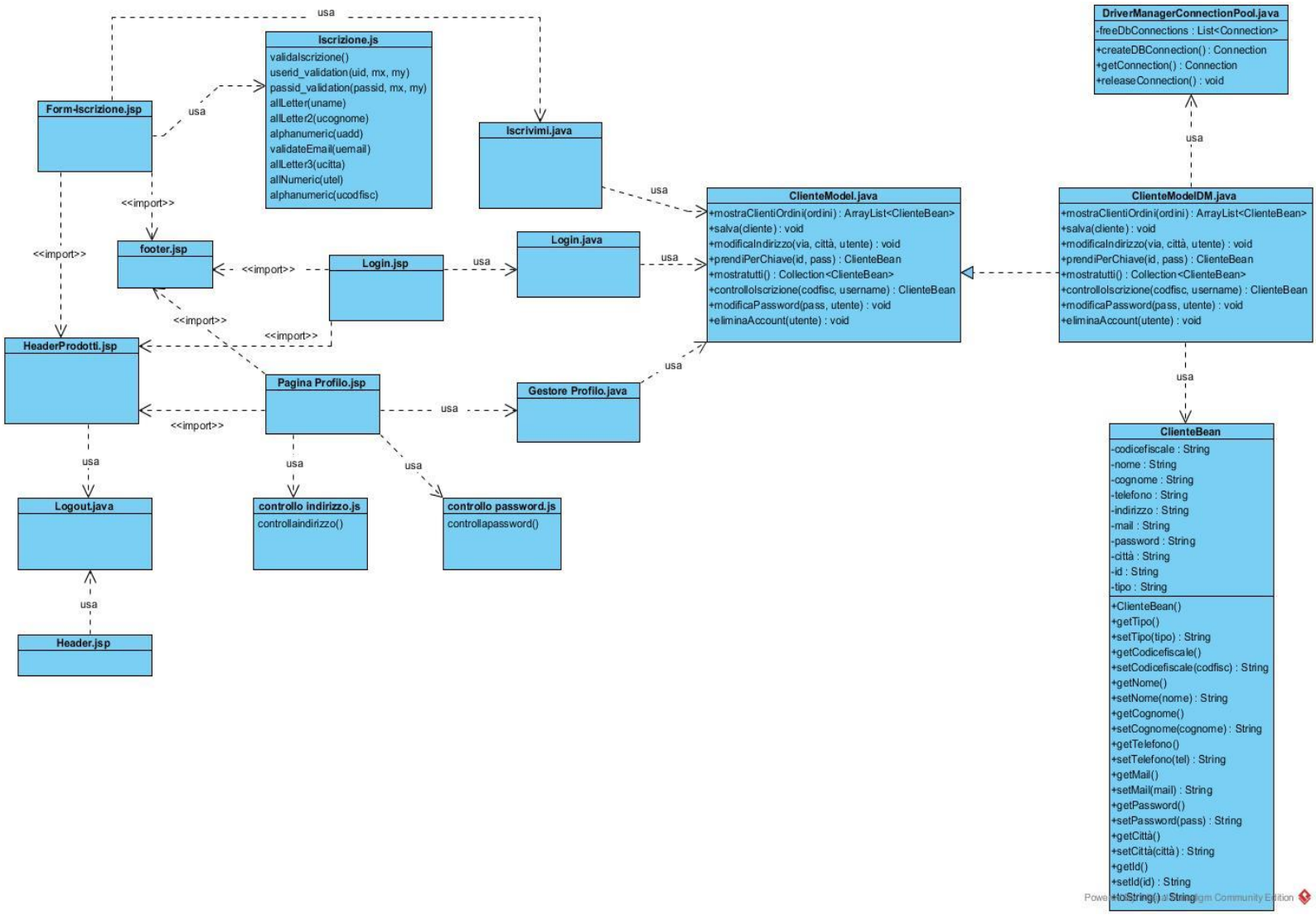
Classe:	**Descrizione:**
style.css	Definisce lo stile della pagina home.
Stile-carrello.css	Definisce lo stile delle pagine visualizzate.



Classe:	**Descrizione:**
controllo.js	Javascript che esegue il controllo sul genere inserito nel form per aggiungere un videogioco al catalogo
ControlloIndirizzo.js	Javascript che esegue i controlli sui campi del form di modifica indirizzo
Controllopass.js	Javascript che esegue i controlli sui campi del form di modifica password
Iscrizione.js	Javascript che esegue i controlli sui campi del form di iscrizione

4. Class Interfaces

4.1 ClienteModelDM



Classe: ClienteBean			
Descrizione	La classe *ClienteBean* rappresenta l'oggetto utente nel database.		
Attributi			
Nome	**Accesso**	**Descrizione**	**Tipo**
codicefiscale	Privato	Codicefiscale dell'utente	String
Nome	privato	Nome dell'utente	String
Cognome	Privato	Cognome dell'utente	String
Telefono	Privato	Numero di telefono dell'utente	String
Indirizzo	Privato	Indirizzo di spedizione dell'utente	String
Mail	Privato	Indirizzo email dell'utente	String
Password	Privato	Password dell'utente	String
Città	Privato	Città dell'utente	String
Id	Privato	Username dell'utente	String
Tipo	Privato	Campo che distingue i vari utenti	String

Metodi			
Nome	**Accesso**	**Descrizione**	**Eccezioni**
ClienteBean	Pubblico	Metodo costruttore che crea un'istanza di Cliente che con i vari attributi vuoti.	
getTipo	Pubblico	Metodo che restituisce il tipo di utente	

setTipo	Pubblico	Metodo che riceve in input la tipologia di utente e setta l'attributo dell'istanza	
getCodicefiscale	Pubblico	Metodo che restituisce il codice fiscale di un utente	
setCodicefiscale	Pubblico	Metodo che riceve in input il codice fiscale di utente e setta l'attributo dell'istanza	
getNome	Pubblico	Metodo che restituisce il nome di un utente	
setNome	Pubblico	Metodo che riceve in input il nome di utente e setta l'attributo dell'istanza	
getCognome	Pubblico	Metodo che restituisce il cognome di un utente	
setCognome	Pubblico	Metodo che riceve in input il cognome di utente e setta l'attributo dell'istanza	
getTelefono	Pubblico	Metodo che restituisce il numero di telefono di un utente	
setTelefono	Pubblico	Metodo che riceve in input il numero di telefono di un utente e setta l'attributo dell'istanza	
getIndirizzo	Pubblico	Metodo che restituisce l'indirizzo di un utente	
setIndirizzo	Pubblico	Metodo che riceve in input l'indirizzo di un utente e setta l'attributo dell'istanza	
getMail	Pubblico	Metodo che restituisce l'email di un utente	

setMail	Pubblico	Metodo che riceve in input l'email di un utente e setta l'attributo dell'istanza	
getPassword	Pubblico	Metodo che la password di un utente	
setPassword	Pubblico	Metodo che riceve in input la password di un utente e setta l'attributo dell'istanza	
getCittà	Pubblico	Metodo che restituisce la città di un utente	
setCittà	Pubblico	Metodo che riceve in input la città di un utente e setta l'attributo dell'istanza	
getId	Pubblico	Metodo che restituisce l'username di un utente	
setId	Pubblico	Metodo che riceve in input l'username di un utente e setta l'attributo dell'istanza	
toString	Pubblico	Metodo che restituisce una stringa con tutti gli attributi di un utente	

Classe: ClienteModel			
Descrizione	La classe *ClienteModel* è l'interfaccia in cui sono dichiarati i metodi implementati nella classe *ClienteModelDM*		
Attributi			
Nome	**Accesso**	**Descrizione**	
Metodi			
Nome	**Accesso**	**Descrizione**	**Eccezioni**
salva	Pubblico	Metodo che prende in input un'istanza di cliente e la inserisce nell'entità "utente" del database	SQLException
prendiPerChiave	Pubblico	Metodo che prende in input un id e una password e cerca nel database, il record contenente l'id e la password uguali ai parametri passati come parametri	SQLException
mostraClientiOrdini	Pubblico	Metodo che prende in input una lista di ordini ed effettua una query per trovare tutte le informazioni del cliente collegate agli ordini della lista. Restituisce una lista di clienti	SQLException
controlloIscrizione	Pubblico	Metodo che prende in input il codice fiscale e l'id del cliente che tenta di registrarsi al sistema e esegue una query che cerca nel database un record avente il codice fiscale o l'id uguale a quelli passati come parametri. Restituisce una istanza di clienteBean.	SQLException
modificaIndirizzo	Pubblico	Metodo che prende in input il nuovo indirizzo, la nuova città e il codice fiscale del cliente ed	SQLException

		esegue una query di aggiornamento dell'indirizzo e della città sul record dell'entità "utente", del database, avente il codice fiscale uguale a quello passato come parametro	
modificaPassword	Pubblico	Metodo che prende in input la nuova password e il codice fiscale del cliente ed esegue una query di aggiornamento della password sul record dell'entità "utente", del database, avente il codice fiscale uguale a quello passato come parametro	SQLException
eliminaAccount	Pubblico	Metodo che prende in input il codice fiscale del cliente e esegue una query di cancellazione del record, nell'entità "utente", avente il codice fiscale uguale a quello passato come parametro	SQLException

Classe: ClienteModelDM			
Descrizione	La classe *ClienteModelDM* implementa i metodi dell'interfaccia *ClienteModel* per eseguire le operazioni sull'entità "utente" del database		
Attributi			
Nome	**Accesso**	**Descrizione**	**Tipo**
Identità	Privato	Nome dell'entità nel database	String
Metodi			
Nome	**Accesso**	**Descrizione**	**Eccezioni**
salva	Pubblico	Metodo che prende in input un'istanza di cliente e la inserisce nell'entità "utente" del database	SQLException
prendiPerChiave	Pubblico	Metodo che prende in input un id e una password e cerca nel database, il record contenente l'id e la password uguali ai parametri passati come parametri	SQLException
mostraClientiOrdini	Pubblico	Metodo che prende in input una lista di ordini ed effettua una query per trovare tutte le informazioni del cliente collegate agli ordini della lista. Restituisce una lista di clienti	SQLException
controlloIscrizione	Pubblico	Metodo che prende in input il codice fiscale e l'id del cliente che tenta di registrarsi al sistema e esegue una query che cerca nel database un record avente il codice fiscale o l'id uguale a quelli passati come parametri. Restituisce una istanza di clienteBean.	SQLException

modificaIndirizzo	Pubblico	Metodo che prende in input il nuovo indirizzo, la nuova città e il codice fiscale del cliente ed esegue una query di aggiornamento dell'indirizzo e della città sul record dell'entità "utente", del database, avente il codice fiscale uguale a quello passato come parametro	SQLException
modificaPassword	Pubblico	Metodo che prende in input la nuova password e il codice fiscale del cliente ed esegue una query di aggiornamento della password sul record dell'entità "utente", del database, avente il codice fiscale uguale a quello passato come parametro	SQLException
eliminaAccount	Pubblico	Metodo che prende in input il codice fiscale del cliente e esegue una query di cancellazione del record, nell'entità "utente", avente il codice fiscale uguale a quello passato come parametro	SQLException

Classe: DriveManagerConnectionPool			
Descrizione	La classe *DriveManagerConnectionPool* consente di eseguire le operazioni di apertura e chiusura di una connessione al database		
Attributi			
Nome	**Accesso**	**Descrizione**	**Tipo**
FreeDbConnections	Privato	Lista di connessioni al database	List<Connection>
Metodi			
Nome	**Accesso**	**Descrizione**	**Eccezioni**
createDbConnection	Pubblico	Metodo che crea e restituisce una connessione al database	SQLException
getConnection	Pubblico	Metodo che restituisce una connessione libera dalla lista delle connessioni	SQLException
ReleaseConnection	Pubblico	Metodo che prende in input una connessione e la aggiunge all'array di connessioni libere	SQLException

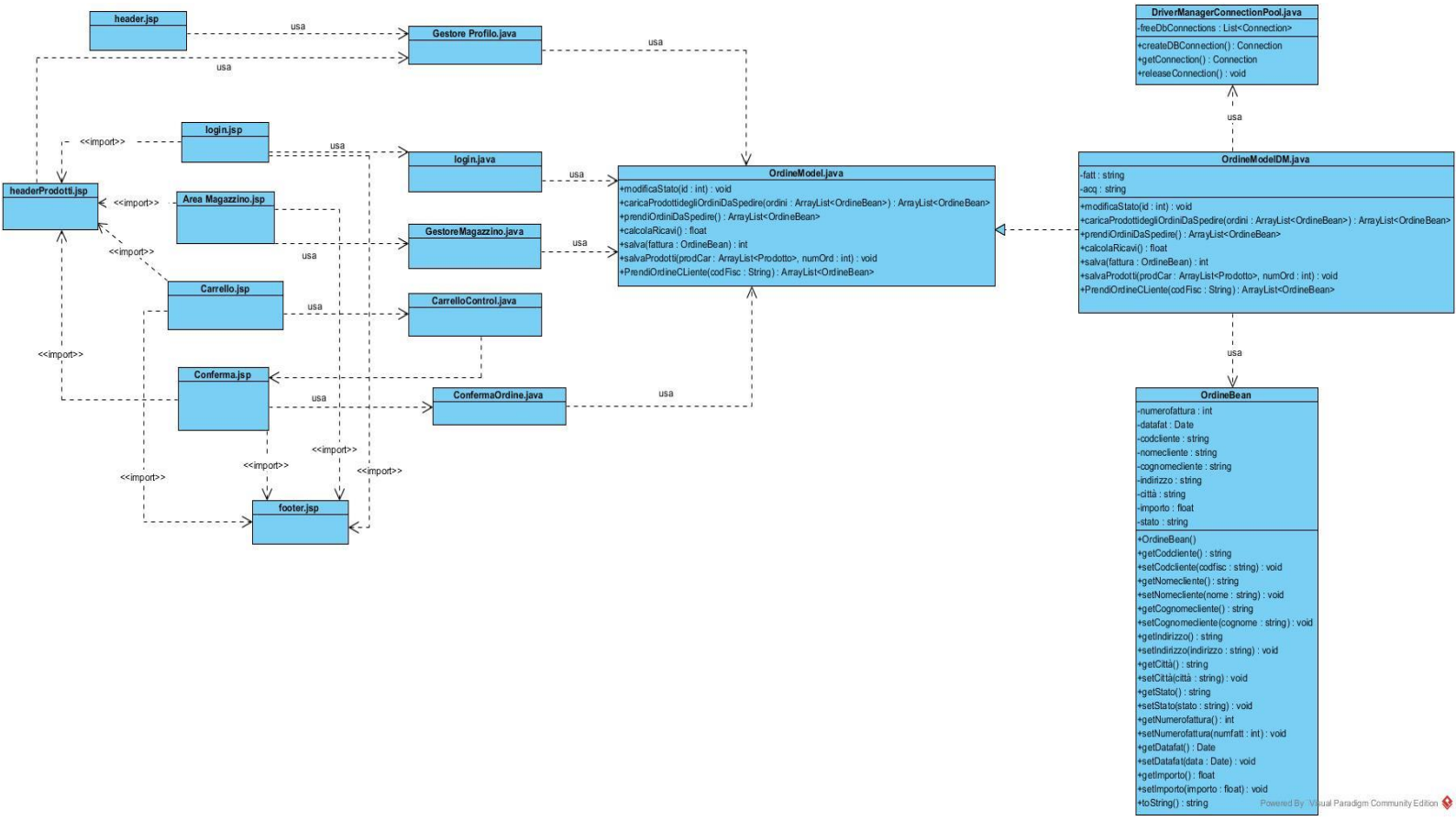
Javascript: controllo indirizzo	
Descrizione	Il file javascript *controllo indirizzo* si occupa dei controlli necessari sul form di modifica indirizzo
funzioni	
Nome	**Descrizione**
Controlla indirizzo	Funzione che esegue i controlli, di lunghezza e formato, sui campi del form

Javascript: controllo password	
Descrizione	Il file javascript *controllo password* si occupa dei controlli necessari sul form di modifica password
funzioni	
Nome	**Descrizione**
Controlla pass	Funzione che esegue i controlli, di lunghezza e formato, sui campi del form

Javascript: iscrizione	
Descrizione	Il file javascript *iscrizione* si occupa dei controlli necessari sul form di registrazione
funzioni	
Nome	**Descrizione**
ValidaIscrizione	Funzione che, chiamando in cascata tutte le altre funzioni descritte sotto, esegue i controlli necessari su tutti i campi del form di iscrizione
Userid_validation	Funzione che esegue i controlli, di lunghezza e formato, sul campo “id utente” del form

Passid_validation	Funzione che esegue i controlli, di lunghezza e formato, sul campo “password” del form
AllLetter	Funzione che esegue i controlli, di lunghezza e formato, sul campo “nome” del form
AllLetter2	Funzione che esegue i controlli, di lunghezza e formato, sul campo “cognome” del form
Alphanumeric	Funzione che esegue i controlli, di lunghezza e formato, sul campo “indirizzo” del form
ValidateEmail	Funzione che esegue i controlli, di lunghezza e formato, sul campo “email” del form
AllLetter3	Funzione che esegue i controlli, di lunghezza e formato, sul campo “città” del form
AllNumeric	Funzione che esegue i controlli, di lunghezza e formato, sul campo “telefono” del form
Alphanumeric	Funzione che esegue i controlli, di lunghezza e formato, sul campo “codice fiscale” del form

4.2 OrdineModelDM



Classe: OrdineBean			
Descrizione	La classe *OrdineBean* rappresenta l'oggetto ordine nel database.		
Attributi			
Nome	**Accesso**	**Descrizione**	**Tipo**
numerofattura	Privato	Numero dell'ordine	Int
Datafat	privato	Data dell'ordine	Date
Codcliente	Privato	Codice fiscale del cliente che effettua l'ordine	String
Nomecliente	Privato	Nome del cliente che effettua l'ordine	String
Cognomecliente	Privato	Cognome del cliente che effettua l'ordine	String
Indirizzo	Privato	Indirizzo di spedizione dell'ordine	String
Città	Privato	Città riferito all'indirizzo dell'ordine	String
Importo	Privato	Importo totale dell'ordine	float
stato	Privato	Stato dell'ordine	String
Metodi			
Nome	**Accesso**	**Descrizione**	**Eccezioni**

OrdineBean	Pubblico	Metodo costruttore che crea un'istanza di OrdineBean con i vari attributi vuoti.	
getCodcliente	Pubblico	Metodo che restituisce il codice fiscale dell'utente che ha effettuato l'ordine	
setCodcliente	Pubblico	Metodo che riceve in input il codice fiscale e setta l'attributo dell'istanza	
getNomecliente	Pubblico	Metodo che restituisce il nome dell'utente che ha effettuato l'ordine	
setNomecliente	Pubblico	Metodo che riceve in input il nome e setta l'attributo dell'istanza	
GetCognomecliente	Pubblico	Metodo che restituisce il cognome dell'utente che ha effettuato l'ordine	
setCognomecliente	Pubblico	Metodo che riceve in input il cognome e setta l'attributo dell'istanza	
GetIndirizzo	Pubblico	Metodo che restituisce l'indirizzo di spedizione dell'ordine	
SetIndirizzo	Pubblico	Metodo che riceve in input l'indirizzo di spedizione e setta l'attributo dell'istanza	
GetCittà	Pubblico	Metodo che restituisce la città relativa all'indirizzo di spedizione dell'ordine	
SetCittà	Pubblico	Metodo che riceve in input la città e setta l'attributo dell'istanza	

getStato	Pubblico	Metodo che restituisce lo stato dell'ordine	
SetStato	Pubblico	Metodo che riceve in input lo stato e setta l'attributo dell'istanza	
getNumerofattura	Pubblico	Metodo che restituisce il numero dell'ordine	
setNumerofattura	Pubblico	Metodo che riceve in input il numero dell'ordine e setta l'attributo dell'istanza	
GetDatafat	Pubblico	Metodo che restituisce la data in cui è stata effettuata l'ordine	
SetDatafat	Pubblico	Metodo che riceve in input la data e setta l'attributo dell'istanza	
GetImporto	Pubblico	Metodo che restituisce l'importo totale dell'ordine	
SetImporto	Pubblico	Metodo che riceve in input l'importo e setta l'attributo dell'istanza	
toString	Pubblico	Metodo che restituisce tutte le informazioni di un ordine	

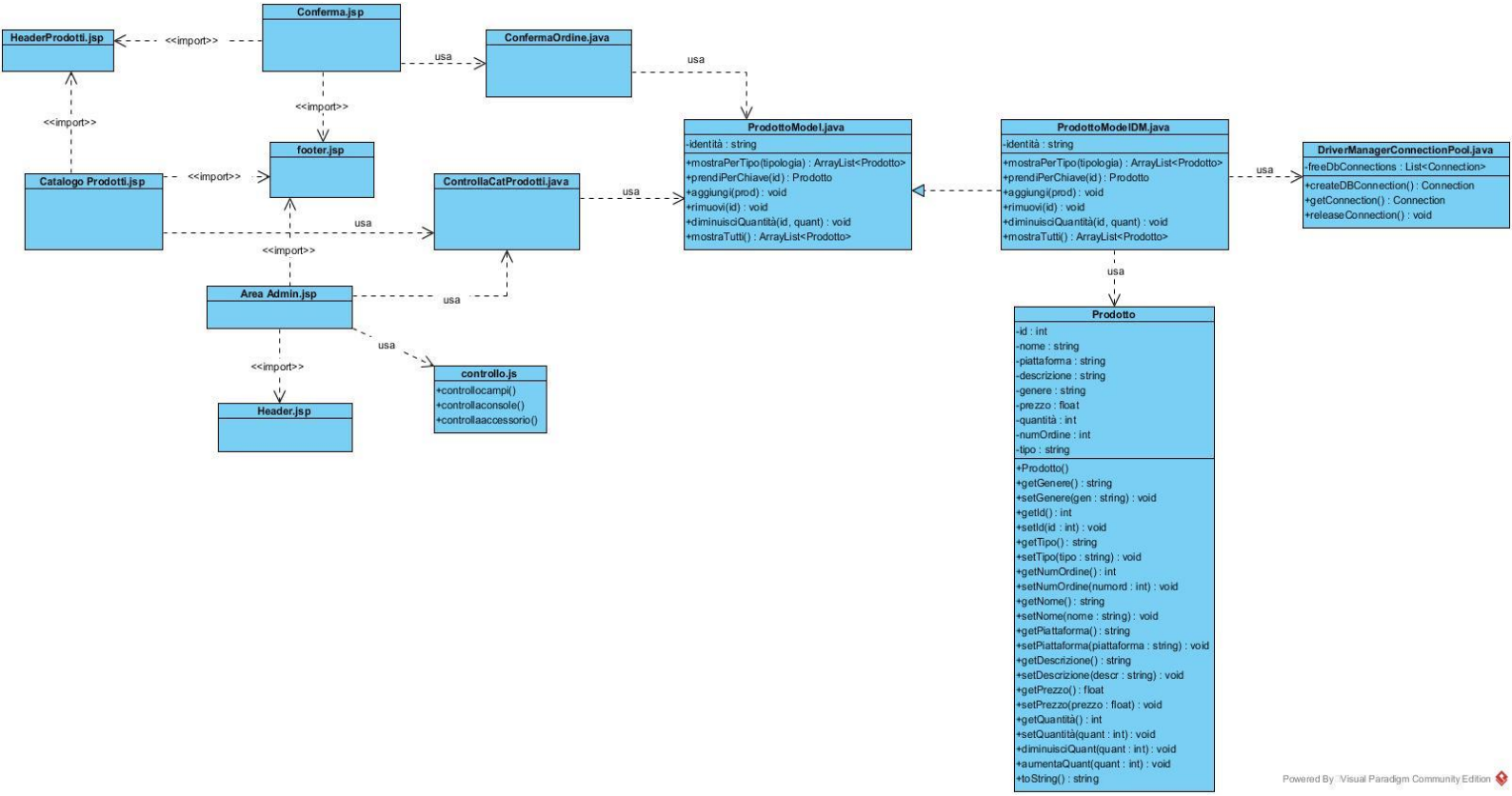
Classe: OrdineModel			
Descrizione	La classe *OrdineModel* è l'interfaccia in cui sono dichiarati i metodi implementati nella classe *OrdineModelDM*		
Attributi			
Nome	**Accesso**	**Descrizione**	
Metodi			
Nome	**Accesso**	**Descrizione**	**Eccezioni**
calcolaRicavi	Pubblico	Metodo che esegue la query di ricerca degli ordini nel database, prende il loro importo e li somma. Restituisce tale somma.	SQLException
salva	Pubblico	Metodo che salva un ordine nel database. Dopo aver salvato l'ordine, esegue una query per ricavare il numero dell'ordine (essendo auto_increment nel database) e restituisce tale numero.	SQLException
salvaProdotti	Pubblico	Metodo che riceve in input la lista di prodotti e il numero dell'ordine a cui si riferiscono. Per ogni prodotto della lista, viene eseguita una query di inserimento nell'entità "acquisto".	SQLException
prendiOrdiniDaSpedire	Pubblico	Metodo che esegue la query di ricerca degli ordini con lo stato = "non spedito". Restituisce tale lista.	SQLException
caricaProdottidegliOrdiniDaSpedire	Pubblico	Metodo che riceve in input una lista di ordini ed esegue, per ogni ordine della lista, la query di ricerca dei prodotti acquistati. Restituisce una lista di prodotti.	SQLException

modificaStato	Pubblico	Metodo che prende in input il numero dell'ordine ed esegue la query di aggiornamento del record nel database con il numero fattura uguale a quello passato come parametro. Nello specifico, la query aggiorna il campo "stato" del record da "non spedito" a "spedito"	SQLException
prendiOrdineCliente	Pubblico	Metodo che riceve in input il codice fiscale di un cliente e esegue la query per trovare tutti gli ordini ad esso collegati. Restituisce tale lista di ordini.	SQLException

Classe: OrdineModelDM			
Descrizione	La classe *OrdineModelDM* implementa i metodi dell'interfaccia *OrdineModel* per eseguire le operazioni sull'entità "ordine" e sull'entità "acquisto" del database		
Attributi			
Nome	**Accesso**	**Descrizione**	**Tipo**
Fatt	Privato	Nome dell'entità "ordine" nel database	String
Acq	Privato	Nome dell'entità "acquisto" nel database	String
Metodi			
Nome	**Accesso**	**Descrizione**	**Eccezioni**
calcolaRicavi	Pubblico	Metodo che esegue la query di ricerca degli ordini nel database, prende il loro importo e li somma. Restituisce tale somma.	SQLException
salva	Pubblico	Metodo che salva un ordine nel database. Dopo aver salvato l'ordine, esegue una query per ricavare il numero dell'ordine (essendo auto_increment nel database) e restituisce tale numero.	SQLException
salvaProdotti	Pubblico	Metodo che riceve in input la lista di prodotti e il numero dell'ordine a cui si riferiscono. Per ogni prodotto della lista, viene eseguita una query di inserimento nell'entità "acquisto".	SQLException
prendiOrdiniDaSpedire	Pubblico	Metodo che esegue la query di ricerca degli ordini con lo	SQLException

		stato= ”non spedito”. Restituisce tale lista.	ParseException
caricaProdottidegliOrdiniDaSpedire	Pubblico	Metodo che riceve in input una lista di ordini ed esegue, per ogni ordine della lista, la query di ricerca dei prodotti acquistati. Restituisce una lista di prodotti.	SQLException
modificaStato	Pubblico	Metodo che prende in input il numero dell’ordine ed esegue la query di aggiornamento del record nel database con il numero fattura uguale a quello passato come parametro. Nello specifico, la query aggiorna il campo “stato” del record da “non spedito” a “spedito”	SQLException
prendiOrdineCliente	Pubblico	Metodo che riceve in input il codice fiscale di un cliente e esegue la query per trovare tutti gli ordini ad esso collegati. Restituisce tale lista di ordini.	SQLException ParseException

4.3 ProdottoModelDM



Classe: Prodotto			
Descrizione	La classe *Prodotto* rappresenta l'oggetto prodotto nel database.		
Attributi			
Nome	**Accesso**	**Descrizione**	**Tipo**
id	Privato	Id del prodotto	Int
Nome	privato	Nome del prodotto	string
Piattaforma	Privato	Tipo di piattaforma del prodotto	String
Descrizione	Privato	Descrizione del prodotto	String
Genere	Privato	Genere del prodotto	String
Prezzo	Privato	Prezzo del prodotto	Float
Quantità	Privato	Quantità disponibile del prodotto	Int
Numordine	Privato	Numero dell'ordine a cui appartiene il prodotto	Int
Tipo	Privato	Tipologia di prodotto	String
Metodi			
Nome	**Accesso**	**Descrizione**	**Eccezioni**

Prodotto	Pubblico	Metodo costruttore che crea un'istanza di Prodotto con i vari attributi vuoti.	
getGenere	Pubblico	Metodo che restituisce il genere del prodotto	
setGenere	Pubblico	Metodo che riceve in input il genere e setta l'attributo dell'istanza	
getId	Pubblico	Metodo che restituisce l'id del prodotto	
setId	Pubblico	Metodo che riceve in input l'id e setta l'attributo dell'istanza	
getTipo	Pubblico	Metodo che restituisce la tipologia di prodotto	
setTipo	Pubblico	Metodo che riceve in input la tipologia e setta l'attributo dell'istanza	
getNumordine	Pubblico	Metodo che restituisce il numero dell'ordine in cui è presente il prodotto	
setNumordine	Pubblico	Metodo che riceve in input il numero dell'ordine e setta l'attributo dell'istanza	
getNome	Pubblico	Metodo che restituisce il nome del prodotto	
setNome	Pubblico	Metodo che riceve in input il nome e setta l'attributo dell'istanza	
getPiattaforma	Pubblico	Metodo che restituisce la piattaforma del prodotto	

setPiattaforma	Pubblico	Metodo che riceve in input la piattaforma e setta l'attributo dell'istanza	
GetDescrizione	Pubblico	Metodo che restituisce la descrizione del prodotto	
SetDescrizione	Pubblico	Metodo che riceve in input la descrizione e setta l'attributo dell'istanza	
getPrezzo	Pubblico	Metodo che restituisce il prezzo del prodotto	
setPrezzo	Pubblico	Metodo che riceve in input il prezzo e setta l'attributo dell'istanza	
getQuantità	Pubblico	Metodo che restituisce la quantità disponibile del prodotto	
setQuantità	Pubblico	Metodo che riceve in input la quantità e setta l'attributo dell'istanza	
diminuisciQuant	Pubblico	Metodo che riceve in input una quantità e la sottrae alla quantità nell'istanza	
aumentaQuant	Pubblico	Metodo che riceve in input una quantità e la aggiunge alla quantità nell'istanza	
toString	Pubblico	Metodo che restituisce tutte le informazioni di un prodotto	

Classe: ProdottoModel			
Descrizione	La classe *ProdottoModel* è l'interfaccia in cui sono dichiarati i metodi implementati nella classe *ProdottoModelDM*		
Attributi			
Nome	**Accesso**	**Descrizione**	
Metodi			
Nome	**Accesso**	**Descrizione**	**Eccezioni**
mostraPerTipo	Pubblico	Metodo che riceve in input una tipologia di prodotto ed esegue la query di ricerca per trovare i prodotti di quel tipo nel database. Restituisce tale lista.	SQLException
prendiPerChiave	Pubblico	Metodo che riceve in input l'id di un prodotto ed esegue la query di ricerca sul database. Restituisce il prodotto	SQLException
aggiungi	Pubblico	Metodo che riceve in input un prodotto ed inserisce quest'ultimo nel database attraverso una query di inserimento	SQLException
rimuovi	Pubblico	Metodo che riceve in input l'id di un prodotto e esegue una query sul database che elimina il prodotto avente l'id uguale a quello passato come parametro.	SQLException
diminuisciQuantità	Pubblico	Metodo che riceve in input l'id di un prodotto e una quantità. Trova il prodotto nel database, sottrae a quest'ultimo la quantità passata come parametro ed effettua una query di aggiornamento sul database.	SQLException

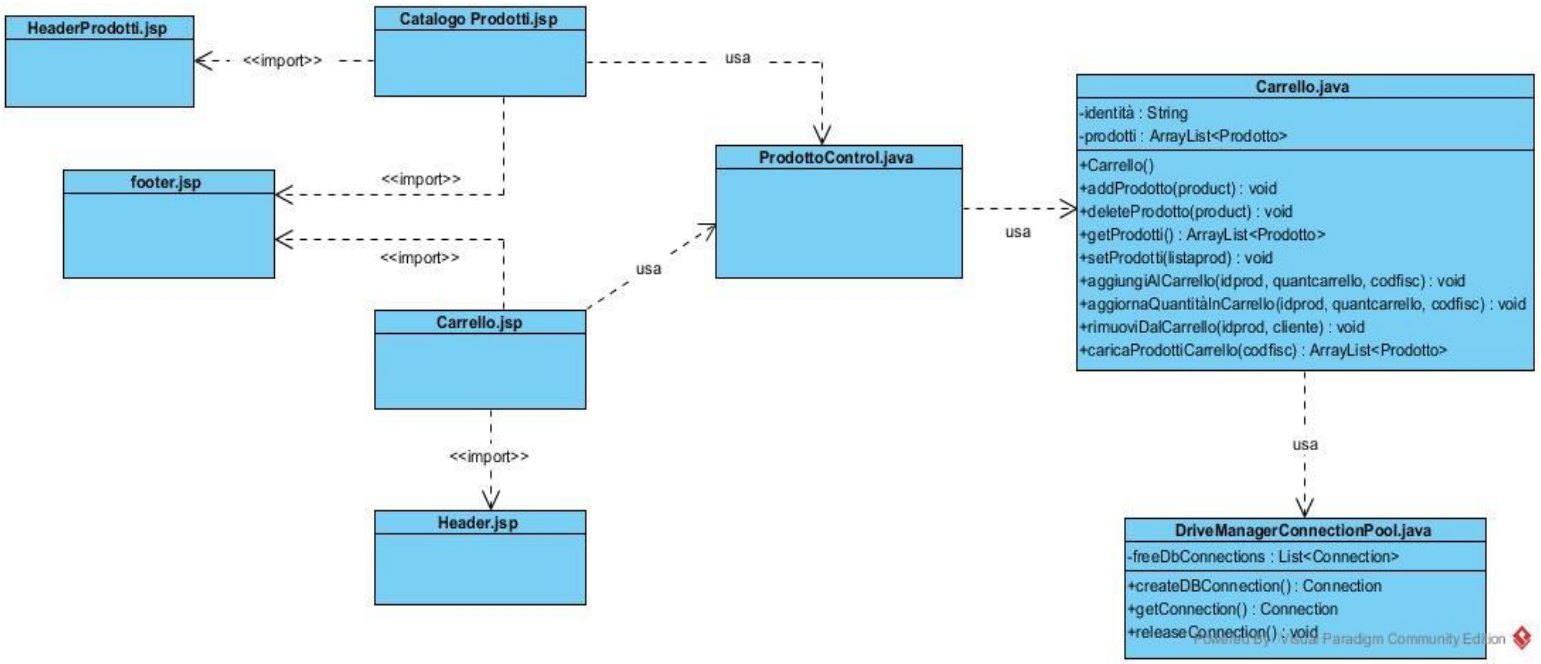
mostraTutti	Pubblico	Metodo che esegue una query di ricerca. Restituisce una lista con tutti i prodotti presenti nel database.	SQLException

Classe: ProdottoModelDM			
Descrizione	La classe *ProdottoModelDM* implementa i metodi dell'interfaccia *ProdottoModel* per eseguire le operazioni sull'entità "prodotto" del database		
Attributi			
Nome	**Accesso**	**Descrizione**	**Tipo**
identità	Privato	Nome dell'entità "prodotto" nel database	String
Metodi			
Nome	**Accesso**	**Descrizione**	**Eccezioni**
mostraPerTipo	Pubblico	Metodo che riceve in input una tipologia di prodotto ed esegue la query di ricerca per trovare i prodotti di quel tipo nel database. Restituisce tale lista.	SQLException
prendiPerChiave	Pubblico	Metodo che riceve in input l'id di un prodotto ed esegue la query di ricerca sul database. Restituisce il prodotto	SQLException
aggiungi	Pubblico	Metodo che riceve in input un prodotto ed inserisce quest'ultimo nel database attraverso una query di inserimento	SQLException
rimuovi	Pubblico	Metodo che riceve in input l'id di un prodotto e esegue una query sul database che elimina il prodotto avente l'id uguale a quello passato come parametro.	SQLException
diminuisciQuantità	Pubblico	Metodo che riceve in input l'id di un prodotto e una quantità. Trova il prodotto nel database, sottrae a quest'ultimo la quantità passata come parametro ed effettua una	SQLException

		query di aggiornamento sul database.	
mostraTutti	Pubblico	Metodo che esegue una query di ricerca. Restituisce una lista con tutti i prodotti presenti nel database.	SQLException

Javascript: controllo	
Descrizione	Il file javascript *controllo* si occupa dei controlli necessari sui campi dei form di aggiunta di un prodotto al catalogo
funzioni	
Nome	**Descrizione**
controllocampi	Funzione che esegue i controlli, di lunghezza e formato, sui campi del form di "aggiunta videogioco al catalogo"
controllaconsole	Funzione che esegue i controlli, di lunghezza e formato, sui campi del form di "aggiunta console al catalogo"
Controllaaccessorio	Funzione che esegue i controlli, di lunghezza e formato, sui campi del form di "aggiunta accessorio al catalogo"

4.4 Carrello


HeaderProdotti.jsp
<<import>>
Catalogo Prodotti.jsp
usa
ProdottoControl.java
Carrello.java
-identità : String
-prodotti : ArrayList<Prodotto>
+Carrello()
+addProdotto(product) : void
+deleteProdotto(product) : void
+getProdotti() : ArrayList<Prodotto>
+setProdotti(listaprod) : void
+aggiungiAlCarrello(idprod, quantcarrello, codfisc) : void
+aggiornaQuantitàInCarrello(idprod, quantcarrello, codfisc) : void
+rimuoviDalCarrello(idprod, cliente) : void
+caricaProdottiCarrello(codfisc) : ArrayList<Prodotto>
footer.jsp
<<import>>
<<import>>
usa
usa
Carrello.jsp
<<import>>
Header.jsp
DriveManagerConnectionPool.java
-freeDbConnections : List<Connection>
+createDBConnection() : Connection
+getConnection() : Connection
+releaseConnection() : void
Paradigm Community Edition

Classe: Carrello			
Descrizione	La classe *Carrello* si rappresenta l'oggetto carrello nel database e offre i metodi per effettuare le query, riferite al carrello, sul database.		
Attributi			
Nome	**Accesso**	**Descrizione**	**Tipo**
identità	Privato	Nome dell'entità nel database	String
Prodotti	privato	Lista dei prodotti nel carrello	ArrayList<Prodotto>
Metodi			
Nome	**Accesso**	**Descrizione**	**Eccezioni**
Carrello	Pubblico	Metodo costruttore che crea un'istanza di Carrello che contiene un array vuoto di prodotti.	
AddProdotto	Pubblico	Metodo che riceve in input un prodotto e lo aggiunge all'array dei prodotti	
deleteProdotto	Pubblico	Metodo che riceve in input un prodotto e lo rimuove dall'array dei prodotti	
getProdotti	Pubblico	Metodo che restituisce l'array di prodotti nell'istanza di carrello	
setProdotti	Pubblico	Metodo che permette di settare l'array di prodotti all'interno dell'istanza con l'array passato come parametro	
aggiungiAlCarrello	pubblico	Metodo che riceve in input l'id del prodotto, la quantità di prodotto nel carrello e il codice del cliente. Salva il prodotto nell'entità carrello del database	SQLException

aggiornaQuantitàInCarrello	Pubblico	Metodo che riceve in input l'id del prodotto, la quantità di prodotto nel carrello e il codice del cliente. Esegue la query per aggiornare le quantità del relativo prodotto salvato nell'entità carrello del database	SQLException
rimuoviDalCarrello	Pubblico	Metodo che riceve in input l'id del prodotto e il codice del cliente. Esegue la query per eliminare il record dell'entità Carrello nel database	SQLException
caricaProdottiCarrello	Pubblico	Metodo che riceve in input il codice del cliente ed esegue la query per caricare tutti i prodotti, collegati al codice preso in input, dall'entità carrello nel database	SQLException