Immagine che contiene logo, simbolo, Blu elettrico, Elementi grafici

Descrizione generata automaticamente

ODD: Object Design Document

SerieAShop

|  |  |
| --- | --- |
| Riferimento |  |
| Versione | 1.0 |
| Data | 03/01/2024 |
| Destinatario | Gravino Carmine |
| Presentato da | Arcamone Domenico, Ruta Antonio, Guerriero Michele, Nevola Mattia |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| RevisionHistory   |  |  |  |  | | --- | --- | --- | --- | | **Data** | **Versione** | **Descrizione** | **Autori** | | 22/12/2023 | 0.1 | Inserimento componenti off-the-shelf | Tutti | | 23/12/2023 | 0.2 | Inserimento dei Design trade-offs | Tutti | | 27/12/2023 | 0.3 | Inserimento Packages | Tutti | | 28/12/2023 | 0.4 | Inserimento dei paragrafi 1.3.2, 1.3.3, 1.3.4, 1.3.5, 1.3.6 | Tutti | | 29/12/2023 | 0.5 | Completamento paragrafo 1 | Tutti | | 30/12/2023 | 0.6 | Inserimento Class Interfaces | Tutti | | 2/01/2024 | 0.7 | Inserimento Class Diagram | Tutti | | 3/01/2024 | 1.0 | Revisione e conclusione ODD | Tutti | |

Sommario

1. Introduzione

**1.1 Object Design trade-offs**

**1.2 Componenti off-the-shelf**

**1.3 Interface Documentation guidelines**

1.3.1 Classi e interfacce Java

1.3.2 Pagine lato Server (JSP)

1.3.3 Pagine HTML

1.3.4 Script Javascript

1.3.5 Fogli di stile CSS

1.3.6 Database SQL

**1.4 Design Pattern**

1.4.1 Singleton Pattern

**1.5 Definizioni, Acronimi e abbreviazioni**

2. Packages

**2.1 Interface**

**2.2 View**

**2.3 Model**

**2.4 Controller**

3. Class interfaces

4. Class diagram

5. Glossario

# 1. Introduzione

## 1.1 Object Design trade-offs

Nella fase dell’Object Design sorgono diversi compromessi di progettazione ed è necessario stabilire quali punti rispettare e quali rendere opzionali. Per quanto riguarda la realizzazione del sistema sono stati individuati i seguenti trade-off:

**Memoria vs Estensibilità:** Il sistema deve permettere l’estensibilità a discapito della memoria utilizzata così da dare la possibilità al cliente di richiedere lo sviluppo di nuove funzionalità, dando meno importanza alla memoria utilizzata da queste ultime.

**Tempo di risposta vs Affidabilità:** Il sistema sarà implementato in modo tale da preferire l’affidabilità al tempo di risposta, garantendo un controllo più accurato dei dati in input a discapito del tempo di risposta del sistema.

**Disponibilità vs Tolleranza ai guasti:** Il sistema dovrà sempre essere disponibile all’utente in caso di errore durante l’uso di una funzionalità, anche al costo di rendere non disponibile quest’ultima per un lasso di tempo.

**Criteri di manutenzione vs Criteri di performance:** Il sistema sarà implementato preferendo la manutenibilità alla performance in modo da facilitare gli sviluppatori nel processo di aggiornamento del software a discapito delle performance del sistema.

**Comprare vs Costruire:** Il sistema sarà strutturato per la maggior parte da elementi creati dagli sviluppatori in modo tale da aumentare l’accessibilità del sistema.

## 1.2 Componenti off-the-shelf

Per ottenere alcuni effetti visuali nonché velocizzare il sistema utilizzeremo inoltre JQuery e Javascript che permetteranno all’interfaccia di rispondere alle azioni dell’utente agilmente.

Inoltre, verrà utilizzato per gestire le connessioni, un sistema di connection pool, fornito da Tomcat, riferimenti: https://tomcat.apache.org/tomcat-10.0-doc/jdbc-pool.html. Risulta senza alcun’ombra di dubbio, più efficiente e veloce utilizzare una componente preesistente e fornita da un ente riconosciuto.

## 1.3 Interface Documentation guidelines

Nell’implementazione del sistema, i programmatori dovranno attenersi alle linee guida di seguito definite.

### 1.3.1 Classi e interfacce Java

Nella scrittura di codice per le classi Java ci si atterrà allo standard Google Java (<http://google.github.io/styleguide/javaguide.html#s4.6-whitespace> ) nella sua interezza. Tale standard fornisce delle regole da seguire ad esempio ogni metodo ed ogni file possono non essere preceduti da un commento. Potranno esserci, inoltre, commenti e giustificazioni in merito a particolari decisioni o calcoli. La convenzione utilizzata dai team member per quanto riguarda i nomi delle variabili, è la nota lowerCamelCase, che consiste nello scrivere parole composte o frasi unendo tutte le parole tra loro. Quando si codificano classi e interfacce Java, si dovrebbero rispettare le seguenti regole di formattazione:

1. Non inserire spazi tra il nome del metodo e la parentesi tonda “(” che apre la lista dei parametri.
2. La parentesi graffa aperta “{“si trova alla fine della stessa linea dell’istruzione di dichiarazione.
3. La parentesi graffa chiusa “}” inizia su una nuova riga vuota allo stesso livello di indentazione del nome della classe o dell’interfaccia.

Nel caso di istruzioni semplici, ogni linea deve contenere al massimo una sola istruzione. Mentre nel caso di istruzioni composte vanno rispettate le seguenti regole:

1. Le istruzioni racchiuse all’interno di un blocco (esempio: for), devono essere indentate di un’unità all’interno dell’istruzione composta.
2. La parentesi di apertura del blocco deve trovarsi alla fine della riga dell’istruzione composta.
3. La parentesi di chiusura del blocco deve trovarsi allo stesso livello di indentazione dell’istruzione composta
4. Le istruzioni composte formate da un’unica istruzione devono essere racchiuse da parentesi.

I nomi di classe devono essere sostantivi, con lettere minuscole e, sia la prima lettera del nome della classe sia la prima lettera di ogni parola interna, deve essere maiuscola. I nomi delle classi dovrebbero essere semplici, descrittivi e che rispettino il dominio applicativo. Non dovrebbero essere usati underscore per legare nomi. I nomi dei metodi iniziano con una lettera minuscola (non sono consentiti caratteri speciali) e seguono la notazione a cammello. Dovranno essere semplici, descrittivi e che rispettino il dominio applicativo.

### 1.3.2 Pagine lato Server (JSP)

Le pagine JSP devono, quando eseguite, produrre un documento conforme allo standard HTML 5. Il codice Java delle pagine deve aderire alle convenzioni per la codifica in Java, con le seguenti puntualizzazioni:

1. Il tag di apertura (<%) è seguito immediatamente dalla fine della riga;
2. Il tag di chiusura (%>) si trova all'inizio di una riga;
3. È possibile evitare le due regole precedenti, se il corpo del codice Java consiste in una singola istruzione (<%= %>):

### 1.3.3 Pagine HTML

Le pagine HTML, sia in forma statica che dinamica, devono essere conformi allo standard HTML 5. Inoltre, il codice HTML statico deve utilizzare l'indentazione, per facilitare la lettura, secondo le seguenti regole:

1. Un'indentazione consiste in una tabulazione;
2. Ogni tag deve avere un'indentazione maggiore del tag che lo contiene;
3. Ogni tag di chiusura deve avere lo stesso livello di indentazione del corrispondente tag di apertura;
4. I tag di commento devono seguire le stesse regole che si applicano ai tag normali.

### 1.3.4 Script Javascript

Gli Script in Javascript devono rispettare le seguenti convenzioni:

1. Gli script che svolgono funzioni distinte dal mero rendering della pagina dovrebbero essere collocati in file dedicati.
2. Il codice Javascript deve seguire le stesse convenzioni per il layout e i nomi del codice Java.
3. I documenti Javascript devono essere iniziati da un commento analogo a quello presente nei file Java.
4. Le funzioni Javascript devono essere documentate in modo analogo ai metodi Java.

### 

### 1.3.5 Fogli di stile CSS

I fogli di stile (CSS) devono seguire le seguenti convenzioni:

Tutti gli stili non inline devono essere collocati in fogli di stile separati.

Ogni regola CSS deve essere formattata come segue:

1. I selettori della regola si trovano a livello 0 di indentazione, uno per riga;
2. L'ultimo selettore della regola è seguito da parentesi graffa aperta ({);
3. Le proprietà che costituiscono la regola sono listate una per riga e sono indentate rispetto ai selettori;
4. La regola è terminata da una parentesi graffa chiusa (}), collocata da sola su una riga;

### 1.3.6 Database SQL

I nomi delle tabelle devono seguire le seguenti regole:

1. Devono essere costituiti dalla prima lettera maiuscola, il restante in minuscolo;
2. Il nome deve essere un sostantivo singolare tratto dal dominio del problema ed esplicativo del contenuto.

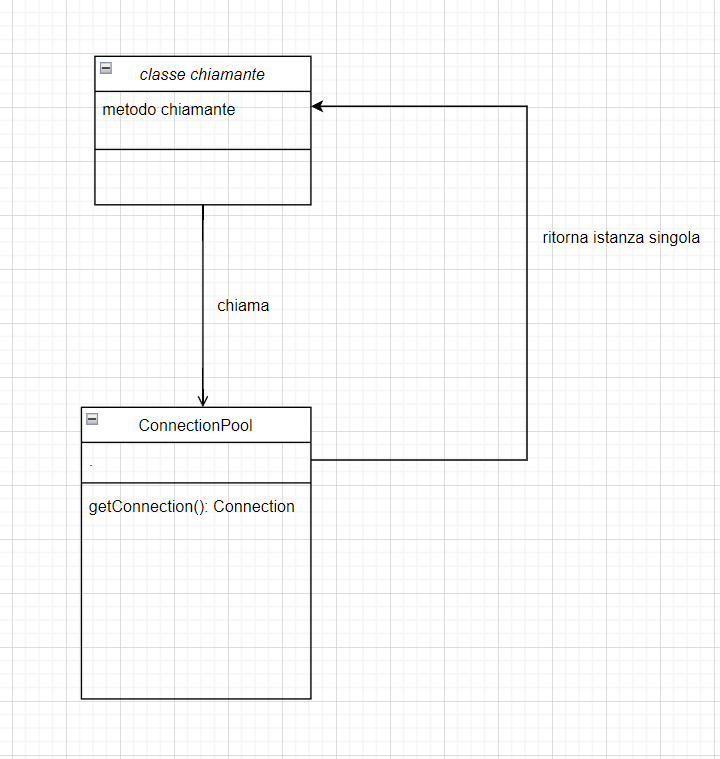
I nomi dei campi devono seguire le seguenti regole:

1. Devono essere costituiti dalla prima lettera maiuscola, il restante in minuscolo;
2. Se il nome è costituito da più parole, è previsto l’uso di lettere maiuscole per evidenziare l’unione di parole;
3. Il nome deve essere un sostantivo singolare tratto dal dominio del problema ed esplicativo del contenuto.

## 1.4 Design Pattern

### 1.4.1 Singleton Pattern

Il Pattern Singleton è uno dei design pattern creazionali. Questo pattern è composto di una sola classe, che è responsabile del fatto che ci sia, in ogni momento, solo una istanza della stessa. Questa classe permette l’accesso a questa istanza senza che sia necessario istanziare oggetti. In SerieAShop, utilizziamo questo pattern per ConnectionPool, rendendo questa classe a singola istanza e facilmente accessibile.



## 1.5 Definizioni, Acronimi e abbreviazioni

**JSP:** acronimo diJava Scripting Preprocessor, è una tecnologia di programmazione web injava per lo sviluppo della logica di presentazione (tipicamente secondo il pattern MVC) di applicazioni web.

**MVC:** acronimo di Model-view-controller, è un pattern architetturale molto diffuso nello sviluppo di sistemi software**.**

**Off-The-Shelf:** Servizi esterni al sistema di cui viene fatto utilizzo..

**HTML:** Linguaggio di programmazione utilizzato per lo sviluppo di pagine Web.

**CSS:** acronimo di Cascading Style Sheetsè un linguaggio usato per definire la formattazione delle pagine Web.

**JavaScript:** JavaScript è un linguaggio di scripting orientato agli oggetti e agli eventi, comunemente utilizzato nella programmazione Web lato client per la creazione di effetti dinamici interattivi.

**JQuery:** JQuery è una libreria JavaScript per applicazioni web.

**lowerCamelCase:** Il lowerCamelCase è una tecnica di naming delle variabili adottata dallo standard Google Java. Essa consiste nello scrivere più parole insieme delimitando la fine e l’inizio di una nuova parola con una lettera maiuscola.

**Servlet:** i servlet sono oggetti scritti in linguaggio Java che operano all'interno di un server web.

**Tomcat:** Apache Tomcat è un web server open source. Implementa le specifiche JavaServer Pages (JSP) e servlet, fornendo quindi una piattaforma software per l'esecuzione di applicazioni Web sviluppate in linguaggio Java.

# 2. Packages

In questa sezione viene mostrata la suddivisione del sistema in vari package. Tale struttura ricalca la suddivisione in sottosistemi effettuata nel documento di SDD ed è motivata dalle scelte architetturali prese in precedenza. A queste si aggiungono file di configurazione o classi di utilità. La struttura di directory è definita da Maven.

* **.idea**
* **src:** contiene tutti i file sorgente.

**– main**

**∗ java**: contiene le classi Java relative alle componenti Control, Model e Test.

**∗ webapp**: contiene le jsp, file CSS ,immagini di Sistema e Script.

* **target:** contiene tutti i file prodotti dal sistema di build di Maven.

## 2.1 Package SerieAShop

Nella presente sezione si mostra la struttura del package principale di SerieAShop, contenente tutte le classi Java (non verrà descritta la divisione delle jsp in diverse directory).

La struttura generale è stata ottenuta creando due packages per dividere le classi relative al Controller da quelle del Model;

## 

## 

Immagine che contiene testo, schermata, Carattere, diagramma

Descrizione generata automaticamente

### 2.1.1 Package Controller

### 2.1.2 Package Model

# 

# 3. Class interfaces

### 3.1 Controller

| **Nome classe** | **AddProdotto** |
| --- | --- |
| **descrizione** | Questa classe viene utilizzata per aggiungere un prodotto al catalogo |
| **metodi** | +doGet(HttpServletRequest request, HttpServletResponse response):void  +extractFileNamePart(Part part):string |
|  |  |

| **Nome metodo** | +doGet(HttpServletRequest request, HttpServletResponse response):void |
| --- | --- |
| **descrizione** | Questo metodo permette di caricare in database un nuovo prodotto |
| **pre-condizione** | Dati corretti |
| **post-condizione** | Carica il prodotto sul databse |

| **Nome metodo** | +extractFileNamePart(Part part):string |
| --- | --- |
| **descrizione** | Questo metodo permette di prendere il “file name” del file caricato |
| **pre-condizione** | Nessuna |
| **post-condizione** | Restituisce una stringa contenente il File Name |

| **Nome classe** | **AggiungiAlCarrello** |
| --- | --- |
| **descrizione** | Questa classe aggiunge al carrello personale un prodotto |
| **metodi** | +doGet(HttpServletRequest request, HttpServletResponse response):void |

| **Nome metodo** | +doGet(HttpServletRequest request, HttpServletResponse response):void |
| --- | --- |
| **descrizione** | Questa metodo permette di aggiungere al carrello personale in sessione un prodotto selezionato |
| **pre-condizione** | l’utente è in sessione, l’id del prodotto deve trovarsi nella request |
| **post-condizione** | esiste l’associazione utente in sessione-prodotto selezionato |

| **Nome classe** | **CarrelloServlet** |
| --- | --- |
| **descrizione** | Questa Classe gestisce il carrello con l’aggiunta prodotti e il controllo del totale |
| **metodi** | +doGet(HttpServletRequest request, HttpServletResponse response):void |

| **Nome metodo** | +doGet(HttpServletRequest request, HttpServletResponse response):void |
| --- | --- |
| **descrizione** | Questa metodo permette di visualizzare i prodotti aggiunti al carrello, e modificarne la quantità |
| **pre-condizione** | l’utente è in sessione |
| **post-condizione** | la request contiene l’arraylist con i prodotti nel carrello Utente |

| **Nome classe** | **CreaOrdine** |
| --- | --- |
| **descrizione** | Questa classe crea un ordine dopo aver effettuato il check-out |
| **metodi** | +doGet(HttpServletRequest request, HttpServletResponse response):void  +isValid…(String):Boolean |

| **Nome metodo** | +doGet(HttpServletRequest request, HttpServletResponse response):void |
| --- | --- |
| **descrizione** | Questa metodo permette di inserire all’interno del database l’ordine creato |
| **pre-condizione** | l’utente è in sessione |
| **post-condizione** | Viene caricato sul databse i dati riguardanti l’ordine dell’utente |

| **Nome metodo** | +isValid…(String):Boolean |
| --- | --- |
| **descrizione** | Questa metodo permette di controllare se i campi inseriti sono validi |
| **pre-condizione** | l’utente è in sessione |
| **post-condizione** | Restituisce True/False nel caso in cui i campi sono compilati correttamente o meno |

| **Nome classe** | **DeleteOrdineServlet** |
| --- | --- |
| **descrizione** | Questa classe elimina gli ordini creati |
| **metodi** | +doGet(HttpServletRequest request, HttpServletResponse response):void |

| **Nome metodo** | +doGet(HttpServletRequest request, HttpServletResponse response):void |
| --- | --- |
| **descrizione** | Questa metodo permette di eliminare un ordine creato da database |
| **pre-condizione** | Utente Admin Loggato |
| **post-condizione** | il database elimina i dati relativi a un determinato ordine |

| **Nome classe** | **DeleteProdotto** |
| --- | --- |
| **descrizione** | Questa classe elimina un prodotto dal database |
| **metodi** | +doGet(HttpServletRequest request, HttpServletResponse response):void |

| **Nome metodo** | +doGet(HttpServletRequest request, HttpServletResponse response):void |
| --- | --- |
| **descrizione** | Questa metodo permette di eliminare un prodotto dal database |
| **pre-condizione** | Utente Admin Loggato e ID prodotto esistente nella request |
| **post-condizione** | L’Id Prodotto se esiste viene elimanto dal Database |

### 3.1.4 Package GestioneProfilo

| **Nome classe** | **DeleteUser** |
| --- | --- |
| **descrizione** | Questa classe elimina da Database un Utente esistente |
| **metodi** | +doGet(HttpServletRequest request, HttpServletResponse response):void |

| **Nome metodo** | +doGet(HttpServletRequest request, HttpServletResponse response):void |
| --- | --- |
| **descrizione** | Questa metodo permette di eliminare un utente esistente nel database |
| **pre-condizione** | l’id dell’utente si trova nella request |
| **post-condizione** | L’utente viene eliminato dal database |

| **Nome classe** | **DirectServlet** |
| --- | --- |
| **descrizione** | Questa classe serve per dividere la pagina Admin da quella Utente |
| **metodi** | +doGet(HttpServletRequest request, HttpServletResponse response):void |

| **Nome metodo** | +doGet(HttpServletRequest request, HttpServletResponse response):void |
| --- | --- |
| **descrizione** | Questo metodo permette di reindirizzare l’utente alle pagine appropriate |
| **pre-condizione** | nessuna |
| **post-condizione** | Viene reindirizzato alla pagina richiesta, Admin nel caso di Admin, Utente nel caso Utente |

| **Nome classe** | **LeggiProdotti** |
| --- | --- |
| **descrizione** | Questa classe permette di caricare il catalogo prodotti |
| **metodi** | +doGet(HttpServletRequest request, HttpServletResponse response):void |

| **Nome metodo** | +doPost(HttpServletRequest request, HttpServletResponse response):void |
| --- | --- |
| **descrizione** | Questo metodo permette di reindirizzare al catalogo Prodotti |
| **pre-condizione** | nessuna |
| **post-condizione** | la response contiene un arraylist con i prodotti presi dal Database |

| **Nome classe** | **Login** |
| --- | --- |
| **descrizione** | Questa classe permette all’utente di effettuare il login per accedere alla pagina |
| **metodi** | +doGet(HttpServletRequest request, HttpServletResponse response):void |

| **Nome metodo** | +doGet(HttpServletRequest request, HttpServletResponse response):void |
| --- | --- |
| **descrizione** | Questa metodo permette all’utente di effettuare il login tramite un controllo sulle credenziali |
| **pre-condizione** | i campi di login devono essere presenti nella richiesta |
| **post-condizione** | se le credenziali sono corrente l’utente è loggato |

| **Nome classe** | **Logout** |
| --- | --- |
| **descrizione** | Questa classe permette all’utente di effettuare il logout |
| **metodi** | +doGet(HttpServletRequest request, HttpServletResponse response):void |

| **Nome metodo** | +doGet(HttpServletRequest request, HttpServletResponse response):void |
| --- | --- |
| **descrizione** | Questa metodo permette di invalidare la sessione corrente rendendola Nulla |
| **pre-condizione** | Utente Loggato |
| **post-condizione** | L’utente viene disconnesso dalla sessione corrente |

| **Nome classe** | **ManageProdottoCarrello** |
| --- | --- |
| **descrizione** | Questa classe ci permette di aumentare o diminuire la quantità dei prodotti presenti nel carrello, inoltre ci permette anche di rimuoverli dal carrello. |
| **metodi** | +doPost(HttpServletRequest request, HttpServletResponse response):void |

| **Nome metodo** | +doPost(HttpServletRequest request, HttpServletResponse response):void |
| --- | --- |
| **descrizione** | Questo metodo ci permette di interagire con i prodotti presenti nel carrello, consente di aumentare o diminuire la quantità del prodotto, inoltre ci permette di rimuoverlo dal carrello |
| **pre-condizione** | L’utente deve essere in sessione, e il prodotto deve essere nel carrello |
| **post-condizione** | La quantità del prodotto viene modificata, o viene eliminato dal carrello |

| **Nome classe** | **NewUser** |
| --- | --- |
| **descrizione** | Questa classe ci permette di creare un nuovo utente all’interno del sistema |
| **metodi** | +doGet(HttpServletRequest request, HttpServletResponse response):void  +isValid…(String):boolean |

| **Nome metodo** | +doGet(HttpServletRequest request, HttpServletResponse response):void |
| --- | --- |
| **descrizione** | Questo metodo ci permette di creare un nuovo utente all’interno del sistema e salvarlo nel database. |
| **pre-condizione** | L’utente non deve essere presente all’interno del database e i dati inseriti devo rispettare le specifiche. |
| **post-condizione** | L’utente viene creato e i dati vengono inseriti all’interno del database, e l’utente viene messo in sessione |

| **Nome metodo** | +isValid…(String):boolean |
| --- | --- |
| **descrizione** | Questo metodo ci permette di verificare se i dati inseriti dall’utente, rispettino le specifiche. |
| **pre-condizione** | Nessuna |
| **post-condizione** | Se i risultati rispettano le specifiche viene restitutito true, altrimenti false |

| **Nome classe** | **ProdottoSingolo** |
| --- | --- |
| **descrizione** | Questa classe ci permette di passare i dati del prodotto selezionato, alla pagina di prodotto singolo, dove verrà mostrata la card del prodotto |
| **metodi** | +doGet(HttpServletRequest request, HttpServletResponse response):void |

| **Nome metodo** | +doGet(HttpServletRequest request, HttpServletResponse response):void |
| --- | --- |
| **descrizione** | Questo metodo ci permette di passare i dati del prodotto selezionato, alla pagina di prodotto singolo, dove verrà mostrata la card del prodotto |
| **pre-condizione** | L’id del prodotto deve essere presente nella request |
| **post-condizione** | I dati vengono passati alla pagina |

| **Nome classe** | **SearchServlet** |
| --- | --- |
| **descrizione** | Questa classe ci permette di ricercare determinati prodotti attraverso query |
| **metodi** | +doGet(HttpServletRequest request, HttpServletResponse response):void  +performSearch(String):List<Prodotto>  +getProductsFromDatabase():List<Prodotto>  +convertToJson(List<Prodotto>):String |

| **Nome metodo** | +doGet(HttpServletRequest request, HttpServletResponse response):void |
| --- | --- |
| **descrizione** | Questo metodo ci permette di fare ricerca all’interno del database attraverso i parametri inseriti |
| **pre-condizione** | Deve esserci una stringa |
| **post-condizione** | Risultati di ricerca |

| **Nome metodo** | +performSearch(String):void |
| --- | --- |
| **descrizione** | Questo metodo effettua la ricerca all’interno del database |
| **pre-condizione** | Che la lista allproducts sia piena |
| **post-condizione** | Risultati di ricerca |

| **Nome metodo** | +getProductsFromDatabase():List<Prodotto> |
| --- | --- |
| **descrizione** | Questo metodo ci permette di salvare tutti i prodotti presenti nel database in una lista |
| **pre-condizione** | Devono esserci prodotti all’interno del db |
| **post-condizione** | Viene restituita la lista dei prodotti |

| **Nome metodo** | +convertToJson(List<Prodotto>):String |
| --- | --- |
| **descrizione** | Questo metodo ci permette di convertire una lista in una stringa Json |
| **pre-condizione** | nessuna |
| **post-condizione** | Viene restituita una stringa Json |

| **Nome classe** | **UpdatePrezzo** |
| --- | --- |
| **descrizione** | Questa classe ci permette di modificare il prezzo di un prodotto presente nel nostro catalogo. |
| **metodi** | +doGet(HttpServletRequest request, HttpServletResponse response):void |

| **Nome metodo** | +doGet(HttpServletRequest request, HttpServletResponse response):void |
| --- | --- |
| **descrizione** | Questo metodo ci permette di modificare il prezzo di un prodotto presente nel nostro catalogo. |
| **pre-condizione** | L’id del prodotto da modificare e il nuovo prezzo deve essere presente nella request, e i dati devono rispettare le specifiche |
| **post-condizione** | Il prodotto viene aggiornato |

| **Nome classe** | **UpdateProdotto** |
| --- | --- |
| **descrizione** | Questa classe ci permette di modificare i dati di un prodotto presente nel nostro catalogo. |
| **metodi** | +doGet(HttpServletRequest request, HttpServletResponse response):void |

| **Nome metodo** | +doGet(HttpServletRequest request, HttpServletResponse response):void |
| --- | --- |
| **descrizione** | Questo metodo ci permette di modificare i dati un prodotto presente nel nostro catalogo. |
| **pre-condizione** | L’id del prodotto da modificare e i dati aggiornati deve essere presente nella request, e i dati devono rispettare le specifiche |
| **post-condizione** | Il prodotto viene aggiornato |

### 3.1.4 Model

| **Nome classe** | **Carrelo** |
| --- | --- |
| **descrizione** | Questa classe descrive un oggetto carrello |
| **metodi** | +getIdCarrello():int  +setIdCarrello(int):void  +getUsername():String  +setUsername(String):void  +getListaCarrello():List<Prodotto>  +setListaCarrello(List<Prodotto>):void  +getTotale():double  +setTotale(double):void  +isEmpty():boolean  +Empty():void  +contains(Prodotto):Boolean  +getCarrello():List<Prodotto>  +setCarrello(List<Prodotto>):void  +addProdotto(Prodotto):void  +sumPrezzi():double  +removeProdotto(int):Prodotto  +abbassaQuantitaProdotto(int):void  +alzaQuantitaProdotto(int):void |

| **Nome metodo** | +getIdCarrello():int |
| --- | --- |
| **descrizione** | Ritorna l’ID del carrello |
| **pre-condizione** | NULL |
| **post-condizione** | Ritorna l’ID del carrello |

| **Nome metodo** | +setIdCarrello(int):void |
| --- | --- |
| **descrizione** | Setta l’ID del carrello |
| **pre-condizione** | Deve passare un intero |
| **post-condizione** | Setta l’ID del carrello |

| **Nome metodo** | +getUsername():String |
| --- | --- |
| **descrizione** | Ritorna l’Username |
| **pre-condizione** | NULL |
| **post-condizione** | Ritorna l’Username |

| **Nome metodo** | +setUsername(String):void |
| --- | --- |
| **descrizione** | Setta l’username con una stringa |
| **pre-condizione** | Passare una stringa |
| **post-condizione** | Setta l’username con una stringa |

| **Nome metodo** | +getTotale():double |
| --- | --- |
| **descrizione** | Questo metodo permette di settare un nuovo corso di laurea del materiale didattico |
| **pre-condizione** | la stringa non deve essere nulla |
| **post-condizione** | nel database risulta il nuovo nome del corso di laurea |

| **Nome metodo** | +setTotale(Double):void |
| --- | --- |
| **descrizione** | Setta il totale con un double |
| **pre-condizione** | Passare un double |
| **post-condizione** | Setta il totale con un double |

| **Nome metodo** | +getCarrello():List<Prodotto> |
| --- | --- |
| **descrizione** | Ritorna una lista di prodotti |
| **pre-condizione** | NULL |
| **post-condizione** | Ritorna una lista di prodotti |

| **Nome metodo** | +setCarrello(List<Prodotto>):void |
| --- | --- |
| **descrizione** | Setta il carrello con una lista di prodotti |
| **pre-condizione** | Passare una lista di prodotti |
| **post-condizione** | Setta il carrello con una lista di prodotti |

| **Nome metodo** | +addProdotto(Prodotto):void |
| --- | --- |
| **descrizione** | Aggiungi i prodotti all’interno della lista |
| **pre-condizione** | Passare un Prodotto |
| **post-condizione** | Aggiunta Prodotto |

| **Nome metodo** | +removeProdotto(int):Prodotto |
| --- | --- |
| **descrizione** | Toglie un prodotto all’interno della lista |
| **pre-condizione** | Passare un int |
| **post-condizione** | Ritorna il prodotto tolto |

| **Nome classe** | **CarrelloDAO** |
| --- | --- |
| **descrizione** | Questa classe DAO implementa un’interfaccia per l’interrogazione al database per i metodi maggiormente usati di Carrello |
| **metodi** | +doCreateCarrello(Carrello):void  +doRetrieveAll():List<Carrello>  + doRetrieveByUsername(String):Carrello |

| **Nome metodo** | + doCreateCarrello(Carrello):void |
| --- | --- |
| **descrizione** | Crea un carrello associato all’utente |
| **pre-condizione** | nessuna |
| **post-condizione** | restituisce una lista |

| **Nome metodo** | + doRetrieveAll():List<Carrello> |
| --- | --- |
| **descrizione** | Restituisce tutti i carrelli che ci sono |
| **pre-condizione** | NULL |
| **post-condizione** | restituisce una lista |

| **Nome metodo** | + doRetrieveByUsername(String):Carrello |
| --- | --- |
| **descrizione** | Questo metodo restituisce un Carrello tramite un username |
| **pre-condizione** | Passare una Stringa |
| **post-condizione** | restituisce un Carrello |

| **Nome classe** | **Ordine** |
| --- | --- |
| **descrizione** | La classe modella oggetti che rappresentano gli Ordini |
| **metodi** | + getId():int  +setId(int):void  +getTotate():int  +setTotale(int):void  +getDataOrd():Date  +setDataOrd(Date):void  +getPagamento():String  +setPagamento(String):void  +getSpedizione():String  +setSpedizione(String):void  +getIdCarrello():int  +setIdCarrello(int):void  +getIdCliente():String  +setIdCliente(String):void  +equals(Object):Boolean  +getProdotti():String  +setProdotti(String):void |

| **Nome metodo** | + getId():int |
| --- | --- |
| **descrizione** | Ritorna un intero |
| **pre-condizione** | nessuna |
| **post-condizione** | Ritorna un intero |

| **Nome metodo** | +setId(int):void |
| --- | --- |
| **descrizione** | Questo metodo permette di settare l’Id |
| **pre-condizione** | Devo passare un intero |
| **post-condizione** | Setta l’id |

| **Nome metodo** | + getTotale():int |
| --- | --- |
| **descrizione** | Ritorna un intero |
| **pre-condizione** | nessuna |
| **post-condizione** | Ritorna un intero |

| **Nome metodo** | +setTotale(int):void |
| --- | --- |
| **descrizione** | Questo metodo permette di settare il totale |
| **pre-condizione** | Devo passare un intero |
| **post-condizione** | Setta il totale |

| **Nome metodo** | + getDataOrd():Date |
| --- | --- |
| **descrizione** | Ritorna una data |
| **pre-condizione** | nessuna |
| **post-condizione** | Ritorna una data |

| **Nome metodo** | +setDataOrd(Date):void |
| --- | --- |
| **descrizione** | Questo metodo permette di settare la data dell’ordine |
| **pre-condizione** | Devo passare una data |
| **post-condizione** | Setta la data |

| **Nome metodo** | + getPagamento():String |
| --- | --- |
| **descrizione** | Ritorna una Stringa |
| **pre-condizione** | nessuna |
| **post-condizione** | Ritorna una Stringa |

| **Nome metodo** | +setPagamento(String):void |
| --- | --- |
| **descrizione** | Questo metodo permette di settare il pagamento |
| **pre-condizione** | Devo passare una Stringa |
| **post-condizione** | Setta il Pagamento |

| **Nome metodo** | + getSpedizione():String |
| --- | --- |
| **descrizione** | Ritorna una Stringa |
| **pre-condizione** | nessuna |
| **post-condizione** | Ritorna una Stringa |

| **Nome metodo** | +setSpedizione(String):void |
| --- | --- |
| **descrizione** | Questo metodo permette di settare la spedizione |
| **pre-condizione** | Devo passare una stringa |
| **post-condizione** | Setta la spedizione |

| **Nome metodo** | + getIdCarrello():int |
| --- | --- |
| **descrizione** | Ritorna un intero |
| **pre-condizione** | nessuna |
| **post-condizione** | Ritorna un intero |

| **Nome metodo** | +setIdCarrello(int):void |
| --- | --- |
| **descrizione** | Questo metodo permette di settare l’Id del Carrello |
| **pre-condizione** | Devo passare un intero |
| **post-condizione** | Setta l’id del Carrello |

| **Nome metodo** | + getIdCliente():String |
| --- | --- |
| **descrizione** | Ritorna una Stringa |
| **pre-condizione** | nessuna |
| **post-condizione** | Ritorna una Stringa |

| **Nome metodo** | +setIdCliente(String):void |
| --- | --- |
| **descrizione** | Questo metodo permette di settare l’Id del Cliente |
| **pre-condizione** | Devo passare una Stringa |
| **post-condizione** | Setta l’id del cliente |

| **Nome metodo** | + equals(Object):boolean |
| --- | --- |
| **descrizione** | Confronta due oggetti e ritorna un boolean differente rispetto al risultato |
| **pre-condizione** | Passare un object |
| **post-condizione** | Ritorna un boolean |

| **Nome metodo** | + getProdotti():String |
| --- | --- |
| **descrizione** | Ritorna una Stringa |
| **pre-condizione** | nessuna |
| **post-condizione** | Ritorna una Stringa |

| **Nome metodo** | +setProdotti(String):void |
| --- | --- |
| **descrizione** | Questo metodo permette di settare il Prodotto |
| **pre-condizione** | Devo passare una Stringa |
| **post-condizione** | Setta il Prodotto |

| **Nome classe** | **OrdineDAO** |
| --- | --- |
| **descrizione** | Questa classe DAO implementa un’interfaccia per l’interrogazione al database per i metodi maggiormente usati di Ordine |
| **metodi** | +doSave(Ordine):void  +doRetriveAll():List<Ordine> |

| **Nome metodo** | +doSave(Ordine):void |
| --- | --- |
| **descrizione** | Salva i dati dell’ordine nel database |
| **pre-condizione** | Passare un ordine |
| **post-condizione** | NULL |

| **Nome metodo** | +doRetriveAll():List<Ordine> |
| --- | --- |
| **descrizione** | Restituisce tutti gli ordini |
| **pre-condizione** | NULL |
| **post-condizione** | restituisce una lista di ordini |

| **Nome classe** | **ProdottiCarrello** |
| --- | --- |
| **descrizione** | La classe modella oggetti che rappresentano ProdottiCarrello |
| **metodi** | + getIdProdCarr():int  +setIdProdCarr(int):void  +getQuantita():int  +setQuantita(int): void  +getIdCarrelo():int  +setIdCarrello(int):void  +getIdProdotto():int  +setIdProdotto(int):void  +getTaglia():String  +setTaglia(String):void |

| **Nome metodo** | + getIdProdCarr():int |
| --- | --- |
| **descrizione** | Questo metodo ritorna l’id del prodotto presente nel carrello |
| **pre-condizione** | nessuna |
| **post-condizione** | restituisce l’id del prodotto presente nel carrello |

| **Nome metodo** | + setIdProdCarr(int):void |
| --- | --- |
| **descrizione** | Questo metodo permette di settare l’id del prodotto presente nel carrello |
| **pre-condizione** | Devo passare un int |
| **post-condizione** | Setto l’Id del prodotto presente nel carrello |

| **Nome metodo** | + getQuantita():int |
| --- | --- |
| **descrizione** | Questo metodo ritorna la quantita |
| **pre-condizione** | nessuna |
| **post-condizione** | restituisce la quantita |

| **Nome metodo** | + setQuantita(int):void |
| --- | --- |
| **descrizione** | Questo metodo permette di settare la quantita |
| **pre-condizione** | Devo passare un int |
| **post-condizione** | Setto la quantita |

| **Nome metodo** | +getIdCarrello():int |
| --- | --- |
| **descrizione** | Questo metodo visiona l’id del carrello |
| **pre-condizione** | nessuna |
| **post-condizione** | restituisce l’id del carrello |

| **Nome metodo** | +setIdCarrello(int):void |
| --- | --- |
| **descrizione** | Questo metodo permette di settare l’id del carrello |
| **pre-condizione** | Passare un intero |
| **post-condizione** | Setta l’Id del carrello |

| **Nome metodo** | +getIdProdotto():int |
| --- | --- |
| **descrizione** | Questo metodo visiona l’id del prodotto |
| **pre-condizione** | nessuna |
| **post-condizione** | restituisce l’id del prodotto |

| **Nome metodo** | +setIdProdotto(int):void |
| --- | --- |
| **descrizione** | Questo metodo permette di settare l’id del Prodotto |
| **pre-condizione** | Passare un intero |
| **post-condizione** | Setta l’Id del prodotto |

| **Nome metodo** | +getTaglia():String |
| --- | --- |
| **descrizione** | Questo metodo ritorna la taglia |
| **pre-condizione** | nessuna |
| **post-condizione** | restituisce la taglia |

| **Nome metodo** | +setTaglia(String):void |
| --- | --- |
| **descrizione** | Questo metodo permette di settare la taglia |
| **pre-condizione** | Passare una Stringa |
| **post-condizione** | Setta la taglia |

| **Nome classe** | **ProdottiCarrelloDAO** |
| --- | --- |
| **descrizione** | Questa classe DAO implementa un’interfaccia per l’interrogazione al database per i metodi maggiormente usati di ProdottiCarrello |
| **metodi** | +doSaveProdottoCarrello(ProdottiCarrello):void  +doRetrivebyId(int):ProdottiCarrello  +doRetriveByCarrelloAndProdotto(int,int):ProdottiCarrello  +doRetriveAllById(int):List<ProdottiCarrello>  +deleteProdottoCarrello(int,int):ProdottiCarrello  +deleteProdottoCarrelloByCarrello(int):ProdottiCarrello  +doRetriveByCarrello(int):List<ProdottiCarrelo>  +doUpdateProdottiQuantita(ProdottiCarrello):void  +doRetriveAll():List<ProdottiCarrello> |

| **Nome metodo** | +doSaveProdottoCarrello(ProdottiCarrello):void |
| --- | --- |
| **descrizione** | Questo metodo permette di salvare i prodotti del carrello |
| **pre-condizione** | Passiamo un ProdottiCarrello |
| **post-condizione** | Salva i prodotti |

| **Nome metodo** | +doRetrivebyId(int):ProdottiCarrello |
| --- | --- |
| **descrizione** | Questo metodo permette di cercare un prodotto in base all’id |
| **pre-condizione** | Passo un intero |
| **post-condizione** | Restituisce un ProdottiCarrello |

| **Nome metodo** | +doRetriveByCarrelloAndProdotto(int,int):ProdottiCarrello |
| --- | --- |
| **descrizione** | Questo metodo permette di cercare un prodotto in base al prodotto e al carrello |
| **pre-condizione** | Passo 2 interi |
| **post-condizione** | Restituisco un prodotto |

| **Nome metodo** | +doRetriveAllById(int):List<ProdottiCarrello> |
| --- | --- |
| **descrizione** | Questo metodo permette di cercare tutto tramite id |
| **pre-condizione** | Passo un int |
| **post-condizione** | Stampo una lista di ProdottiCarrello |

| **Nome metodo** | +deleteProdottoCarrello(int,int):ProdottiCarrello |
| --- | --- |
| **descrizione** | Questo metodo permette di eliminare un prodotto dal carrello |
| **pre-condizione** | Passo due interi |
| **post-condizione** | Stampo il prodotto eliminato |

| **Nome metodo** | +deleteProdottoCarrelloByCarrello(int):ProdottiCarrello |
| --- | --- |
| **descrizione** | Questo metodo permette di eliminare un prodotto dal carrello |
| **pre-condizione** | Passo un intero |
| **post-condizione** | Stampo il prodotto eliminato |

| **Nome metodo** | +doRetriveByCarrello(int):List<ProdottiCarrelo> |
| --- | --- |
| **descrizione** | Questo metodo permette di cercare tramite il carrello |
| **pre-condizione** | Passiamo un intero |
| **post-condizione** | Ritorna una lista di ProdottiCarrello |

| **Nome metodo** | +doUpdateProdottiQuantita(ProdottiCarrello):void |
| --- | --- |
| **descrizione** | Questo metodo permette di modificare la quantità dei prodotti |
| **pre-condizione** |  |
| **post-condizione** | Modifica la quantità dei prodotti nel carrello |

| **Nome metodo** | +doRetriveAll():List<ProdottiCarrello> |
| --- | --- |
| **descrizione** | Questo metodo permette di cercare i prodotti |
| **pre-condizione** | NULL |
| **post-condizione** | Stampa una lista di ProdottiCarrelo |

| **Nome classe** | **SquadraDAO** |
| --- | --- |
| **descrizione** | Questa classe DAO implementa un’interfaccia per l’interrogazione al database per i metodi maggiormente usati di Squadra |
| **metodi** | +doSave():List<Squadra> |

| **Nome metodo** | +doSave():List<Squadra> |
| --- | --- |
| **descrizione** | Questo metodo permette di salvare una squadra |
| **pre-condizione** |  |
| **post-condizione** | Stampa una lista di Squadra |

| **Nome classe** | **User** |
| --- | --- |
| **descrizione** | La classe modella oggetti che rappresentano User |
| **metodi** | +getUsername():String  +setUsername(String):void  +getPassword():String  +getTelefono():String  +getNome():String  +getCognome():String  +getEmail():String  +getIndirizzo():String  +getId():int  +setNome(String):void  +setCognome(String):void  +setIndirizzo(String):void  +setEmail(String):void  +setTelefono(String):void  +setPassword(String):void  +isadmin():boolean  +setAtmin(boolean):void |

| **Nome metodo** | +getUsername():String |
| --- | --- |
| **descrizione** | Questo metodo permette di ritornare l’Username |
| **pre-condizione** |  |
| **post-condizione** | Ritorna una stringa |

| **Nome metodo** | +setUsername(String):void |
| --- | --- |
| **descrizione** | Questo metodo permette di settare l’username |
| **pre-condizione** |  |
| **post-condizione** | Setta l’Username |

| **Nome metodo** | +getPassword():String |
| --- | --- |
| **descrizione** | Questo metodo permette di ritornare la Password |
| **pre-condizione** |  |
| **post-condizione** | Ritorna una stringa |

| **Nome metodo** | +getTelefono():String |
| --- | --- |
| **descrizione** | Questo metodo permette di ritornare il Telefono |
| **pre-condizione** |  |
| **post-condizione** | Ritorna una stringa |

| **Nome metodo** | +getNome():String |
| --- | --- |
| **descrizione** | Questo metodo permette di ritornare il nome |
| **pre-condizione** |  |
| **post-condizione** | Ritorna una stringa |

| **Nome metodo** | +getCognome():String |
| --- | --- |
| **descrizione** | Questo metodo permette di ritornare il Cognome |
| **pre-condizione** |  |
| **post-condizione** | Ritorna una stringa |

| **Nome metodo** | +getEmail():String |
| --- | --- |
| **descrizione** | Questo metodo permette di ritornare l’Email |
| **pre-condizione** |  |
| **post-condizione** | Ritorna una stringa |

| **Nome metodo** | +getIndirizzo():String |
| --- | --- |
| **descrizione** | Questo metodo permette di ritornare l’Indirizzo |
| **pre-condizione** |  |
| **post-condizione** | Ritorna una stringa |

| **Nome metodo** | +getId():int |
| --- | --- |
| **descrizione** | Questo metodo permette di ritornare l’Id |
| **pre-condizione** |  |
| **post-condizione** | Ritorna un intero |

| **Nome metodo** | +setNome(String):void |
| --- | --- |
| **descrizione** | Questo metodo permette di settare il nome |
| **pre-condizione** |  |
| **post-condizione** | Setta il nome |

| **Nome metodo** | +setCognome(String):void |
| --- | --- |
| **descrizione** | Questo metodo permette di settare il Cognome |
| **pre-condizione** |  |
| **post-condizione** | Setta il cognome |

| **Nome metodo** | +setIndirizzo(String):void |
| --- | --- |
| **descrizione** | Questo metodo permette di settare l’indirizzo |
| **pre-condizione** |  |
| **post-condizione** | Setta l’indirizzo |

| **Nome metodo** | +setId(String):void |
| --- | --- |
| **descrizione** | Questo metodo permette di settare l’Id |
| **pre-condizione** |  |
| **post-condizione** | Setta l’Id |

| **Nome metodo** | +setEmail(String):void |
| --- | --- |
| **descrizione** | Questo metodo permette di settare l’Email |
| **pre-condizione** |  |
| **post-condizione** | Setta l’email |

| **Nome metodo** | +setTelefono(String):void |
| --- | --- |
| **descrizione** | Questo metodo permette di settare il telefono |
| **pre-condizione** |  |
| **post-condizione** | Setta il telefono |

| **Nome metodo** | +setPassword(String):void |
| --- | --- |
| **descrizione** | Questo metodo permette di settare la Password |
| **pre-condizione** |  |
| **post-condizione** | Setta la Password |

| **Nome metodo** | +isAdmin():boolean |
| --- | --- |
| **descrizione** | Questo metodo permette di capire se uno user è un admin |
| **pre-condizione** |  |
| **post-condizione** | Ritorna se è un admin o no |

| **Nome metodo** | +setAdmin(boolean):void |
| --- | --- |
| **descrizione** | Questo metodo permette di settare l’Admin |
| **pre-condizione** |  |
| **post-condizione** | Setta l’Admin |

| **Nome classe** | **UserDAO** |
| --- | --- |
| **descrizione** | Questa classe DAO implementa un’interfaccia per l’interrogazione al database per i metodi maggiormente usati di User |
| **metodi** | +doSave(User):void  +deleteUser(String):void  +doRetriveAll():List<User>  +doRetriveByUsername(String):User |

| **Nome metodo** | +doSave(User):void |
| --- | --- |
| **descrizione** | Questo metodo permette di salvare uno user |
| **pre-condizione** |  |
| **post-condizione** | Salva l’user |

| **Nome metodo** | +deleteUser(String):void |
| --- | --- |
| **descrizione** | Questo metodo permette di cancellare uno user |
| **pre-condizione** |  |
| **post-condizione** | Cancella User |

| **Nome metodo** | +doRetriveAll():List<User> |
| --- | --- |
| **descrizione** | Questo metodo permette di ricercare gli utenti |
| **pre-condizione** |  |
| **post-condizione** | Ritorna una lista di utenti |

| **Nome metodo** | +doRetriveByUsername(String):User |
| --- | --- |
| **descrizione** | Questo metodo permette di ricercare gli utenti tramite username |
| **pre-condizione** |  |
| **post-condizione** | Ritorna uno user |

| **Nome classe** | **Prodotto** |
| --- | --- |
| **descrizione** | La classe modella oggetti che rappresentano prodotti |
| **metodi** | + getId():int  + setId(int id):void  + getNome():String  + setNome(String nome):void  + getPrezzo():double  + setPrezzo(double Prezzo):void  + getDescrizione():String  + setDescrizione(String descrizione):void  + getQuantita():int  + setQuantita(int quantita):void  + getIdSquadra():String  + setIdSquadra(String idSquadra):void  + getCategoria():String  + setCategoria(String categoria): void |

| **Nome metodo** | + getId():int |
| --- | --- |
| **descrizione** | Questo metodo visiona l’id del prodotto |
| **pre-condizione** | nessuna |
| **post-condizione** | Restituisce l’id del |

| **Nome metodo** | + setId(int id):void |
| --- | --- |
| **descrizione** | Questo metodo permette di settare l’id del prodotto |
| **pre-condizione** | L’id non deve essere vuoto |
| **post-condizione** | nel database risulta l’id dell’utente |

| **Nome metodo** | +getNome():String |
| --- | --- |
| **descrizione** | Questo metodo visiona il nome del prodotto |
| **pre-condizione** | nessuna |
| **post-condizione** | restituisce il nome del prodotto |

| **Nome metodo** | +setNome(String nome):void |
| --- | --- |
| **descrizione** | Questo metodo permette di settare il nome del prodotto |
| **pre-condizione** | la stringa non deve essere nulla |
| **post-condizione** | nel database risulta il nome del prodotto |

| **Nome metodo** | +getPrezzo():double |
| --- | --- |
| **descrizione** | Questo metodo visiona il prezzo del prodotto |
| **pre-condizione** | nessuna |
| **post-condizione** | Restituisce il prezzo del prodotto |

| **Nome metodo** | +setPrezzo(double prezzo):void |
| --- | --- |
| **descrizione** | Questo metodo permette di settare il prezzo del prodotto |
| **pre-condizione** | la stringa non deve essere nulla |
| **post-condizione** | nel database risulta il prezzo del prodotto |

| **Nome metodo** | +getQuantita():int |
| --- | --- |
| **descrizione** | Questo metodo visiona la quantità del prodotto |
| **pre-condizione** | nessuna |
| **post-condizione** | Restituisce la quantità del prodotto |

| **Nome metodo** | +setQuantita(int quantita):void |
| --- | --- |
| **descrizione** | Questo metodo permette di settare la quantità del prodotto |
| **pre-condizione** | la stringa non deve essere nulla |
| **post-condizione** | nel database risulta la quantità del prodotto |

| **Nome metodo** | +getIdSquadra():String |
| --- | --- |
| **descrizione** | Questo metodo visiona l’id della squadra del prodotto |
| **pre-condizione** | nessuna |
| **post-condizione** | Restituisce l’id della squadra |

| **Nome metodo** | +setIdSquadra(String idSquadra):void |
| --- | --- |
| **descrizione** | Questo metodo permette di settare l’id della squadra del prodotto |
| **pre-condizione** | la stringa non deve essere nulla |
| **post-condizione** | nel database risulta l’id della squadra |

| **Nome metodo** | +getCategoria():String |
| --- | --- |
| **descrizione** | Questo metodo visiona la categoria del prodotto |
| **pre-condizione** | nessuna |
| **post-condizione** | Restituisce la categoria del prodotto |

| **Nome metodo** | +setCategoria(String categoria):void |
| --- | --- |
| **descrizione** | Questo metodo permette di settare la categoria del prodotto |
| **pre-condizione** | la stringa non deve essere nulla |
| **post-condizione** | nel database risulta la categoria del prodotto |

| **Nome classe** | **ProdottoDAO** |
| --- | --- |
| **descrizione** | Questa classe DAO implementa un’interfaccia per l’interrogazione al database per i metodi maggiormente usati di Prodotto |
| **metodi** | + doRetrieveById(int id):Prodotto  + doSave(Prodotto p):void  + doRetriveAll():List<Prodotto>  + doUpdateProdotto(Prodotto p):Prodotto  + deleteProdotto(int id):Prodotto  + doUpdatePrezzo(int id, double prezzo):Prodotto |

| **Nome metodo** | +doRetrieveById(int id):Prodotto |
| --- | --- |
| **descrizione** | Questo metodo permette di creare un prodotto con un determinato id, cercando nel database |
| **pre-condizione** | La variabile id non deve essere nulla |
| **post-condizione** | restituisce un prodotto, con quell’id |

| **Nome metodo** | +doSave(Prodotto p):void |
| --- | --- |
| **descrizione** | Questo metodo permette di aggiugere un prodotto al database |
| **pre-condizione** | I prodotti non devono essere nulli |
| **post-condizione** | Il prodotto viene aggiunto al database |

| **Nome metodo** | +doRetrieveAll():List<Prodotto> |
| --- | --- |
| **descrizione** | Questo metodo crea una lista con tutti i prodotti presenti nel database |
| **pre-condizione** | nessuna |
| **post-condizione** | Restituisce la lista di prodotti |

| **Nome metodo** | +doUpdateProdotto(Prodotto p): Prodotto |
| --- | --- |
| **descrizione** | Questo metodo permette di modificare un prodotto presente nel database |
| **pre-condizione** | Il prodotto deve essere presente nel database, e i dati non devono essere nulli |
| **post-condizione** | Il prodotto viene aggiornato |

| **Nome metodo** | +deleteProdotto(int id):Prodotto |
| --- | --- |
| **descrizione** | Questo metodo permette di eliminare un prodotto dal database |
| **pre-condizione** | Il prodotto deve essere presente nel database, e l’id non deve essere nullo |
| **post-condizione** | Il prodotto viene eliminato dal database |

| **Nome metodo** | +doUpdatePrezzo(int id, double prezzo):Prodotto |
| --- | --- |
| **descrizione** | Questo metodo permette di aggiornare il prezzo di un prodotto esistente nel database. |
| **pre-condizione** | Il prodotto deve essere presente nel database, l’id e il prezzo non devono essere nulli |
| **post-condizione** | Il prezzo del prodotto viene aggiornato nel database |

| **Nome classe** | **Squadra** |
| --- | --- |
| **descrizione** | La classe modella oggetti che rappresentano squadra |
| **metodi** | + getPathLogo():String  + setPathLogo(String p):void  +getNomeSquadra():String  +setNomeSquadra(String nomeSquadra):void |

| **Nome metodo** | +getPathLogo():String |
| --- | --- |
| **descrizione** | Questo metodo ci mostra la path del logo della squadra |
| **pre-condizione** | nessuna |
| **post-condizione** | Viene restituita la path |

| **Nome metodo** | +setPathLogo(String path):void |
| --- | --- |
| **descrizione** | Questo metodo setta la pathLogo della squadra |
| **pre-condizione** | nessuna |
| **post-condizione** | Viene aggiornato la pathLogo |

| **Nome metodo** | +getNomeSquadra():String |
| --- | --- |
| **descrizione** | Questo metodo ci mostra il nome della squadra |
| **pre-condizione** | nessuna |
| **post-condizione** | Viene restituito il nome della squadra |

| **Nome metodo** | +setNomeSquadra(String nome):void |
| --- | --- |
| **descrizione** | Questo metodo setta il nome della squadra |
| **pre-condizione** | nessuna |
| **post-condizione** | Viene aggiornato il nome della squadra |

### 3.3 Package Utils

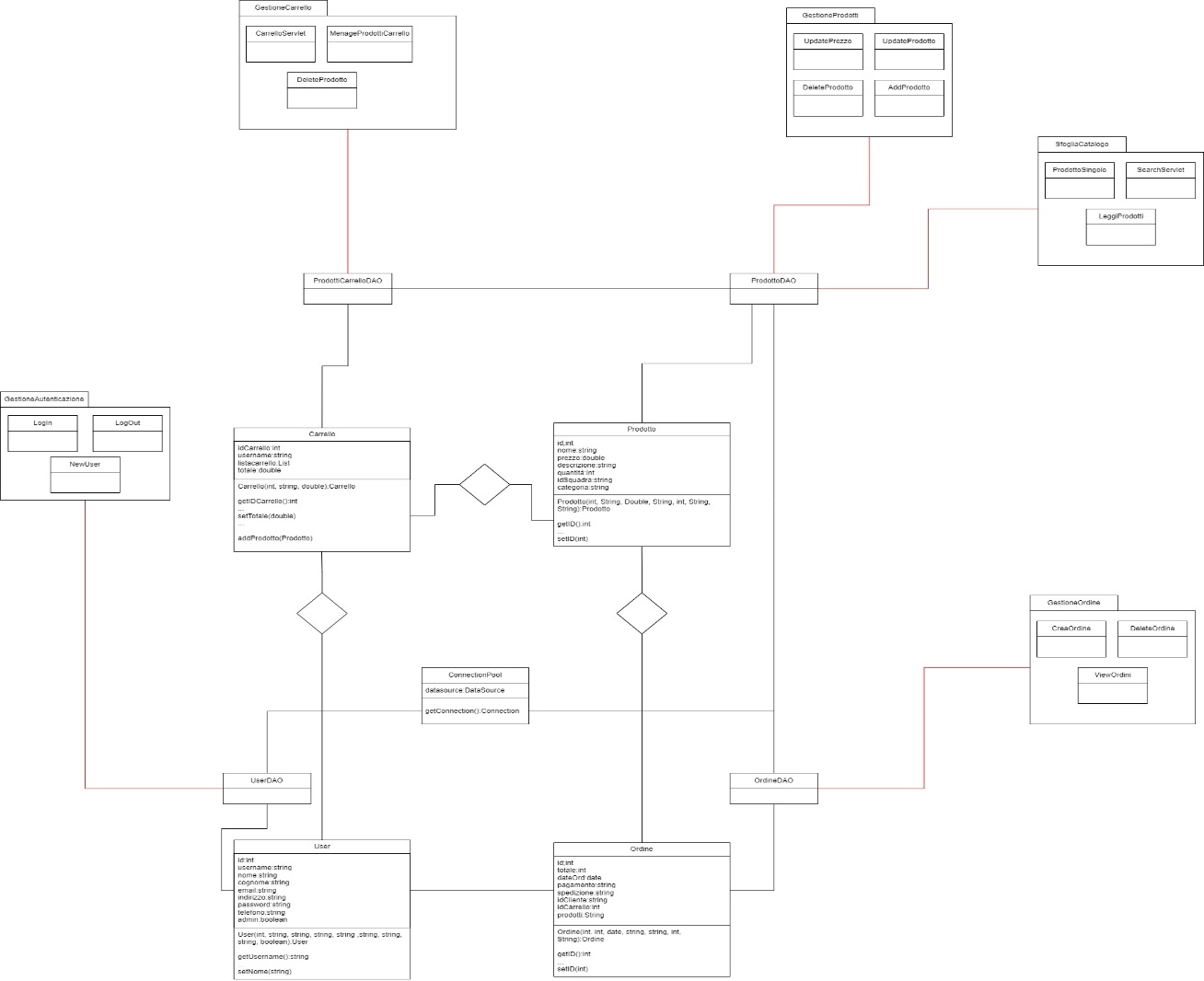
| **Nome classe** | **ConnectionPool** |
| --- | --- |
| **descrizione** | Questa classe di Servizio collega il Sistema al DataBase |
| **metodi** | + getConnection() : Connection |

| **Nome metodo** | + getConnection() : Connection |
| --- | --- |
| **descrizione** | Questo metodo restituisce la connessione al database |
| **pre-condizione** | la connessione esiste e che il server esiste |
| **post-condizione** | restituisce la connessione con il database |

### 

### 

4. Class diagram



# 4. Glossario

**JSP:** acronimo diJava Scripting Preprocessor, è una tecnologia di programmazione web injava per lo sviluppo della logica di presentazione (tipicamente secondo il pattern MVC) di applicazioni web.

**MVC:** acronimo di Model-view-controller, è un pattern architetturale molto diffuso nello sviluppo di sistemi software**.**

**Off-The-Shelf:** Servizi esterni al sistema di cui viene fatto utilizzo.

**HTML:** Linguaggio di programmazione utilizzato per lo sviluppo di pagine Web.

**CSS:** acronimo di Cascading Style Sheetsè un linguaggio usato per definire la formattazione delle pagine Web.

**JavaScript:** JavaScript è un linguaggio di scripting orientato agli oggetti e agli eventi, comunemente utilizzato nella programmazione Web lato client per la creazione di effetti dinamici interattivi.

**JQuery:** JQuery è una libreria JavaScript per applicazioni web.

**lowerCamelCase:** Il lowerCamelCase è una tecnica di naming delle variabili adottata dallo standard Google Java. Essa consiste nello scrivere più parole insieme delimitando la fine e l’inizio di una nuova parola con una lettera maiuscola.

**Servlet:** i servlet sono oggetti scritti in linguaggio Java che operano all'interno di un server web.

**Tomcat:** Apache Tomcat è un web server open source. Implementa le specifiche JavaServer Pages (JSP) e servlet, fornendo quindi una piattaforma software per l'esecuzione di applicazioni Web sviluppate in linguaggio Java.