Sommario

[**1.Introduzione** 1](#_Toc31016827)

[**1.1Object design trade-offs** 1](#_Toc31016828)

[**1.2 Linee guida per la documentazione delle interfacce** 2](#_Toc31016829)

[1.2.1 Naming Convention 2](#_Toc31016830)

[**2 Package** 3](#_Toc31016831)

[**2.1 Struttura sistema** 4](#_Toc31016832)

[**2.2 Descrizione delle classi** 4](#_Toc31016833)

[2.2.1 Data Layer 4](#_Toc31016834)

[2.2.2 Application Layer 6](#_Toc31016835)

[2.2.3 PresentationLayer 8](#_Toc31016836)

[**3 Interfacce delle classi** 10](#_Toc31016837)

[**3.1 Definizione Contratti** 12](#_Toc31016838)

[3.1.1 UtenteDAO 12](#_Toc31016839)

[3.1.2 RuoloDAO 13](#_Toc31016840)

[3.1.3 IndirizzoDAO 13](#_Toc31016841)

[3.1.4 ProdottoDAO 14](#_Toc31016842)

[3.1.5 OrdineDAO 15](#_Toc31016843)

[3.1.6 ComposizioneDAO 16](#_Toc31016844)

[3.1.7 CarrelloB 16](#_Toc31016845)

[**4.Design Pattern** 18](#_Toc31016846)

[**4.1 MVC** 18](#_Toc31016847)

# **1.Introduzione**

## **1.1Object design trade-offs**

Dopo aver stilato il Requirements Analysis Document e il System Design Document, in cui il sistema è descritto in modo sommario, definendo gli obiettivi ma tralasciando gli aspetti implementativi, si procede ora con lo stilare il documento di Object Design che ha come obiettivo quello di produrre un modello che sia in grado di integrare in modo coerente e preciso tutte le funzionalità individuate nelle fasi precedenti.

Inparticolar modo nell’ODD si definiscono le interfacce delle classi, le operazioni ,i tipi, gli argomenti e le signature dei sottosistemi definiti in fase di System Design, inoltre sono specificati i trade offs e le linee guida.

**Comprensibilità vs Tempo:**

Il codice di sistema deve essere il più comprensibile possibile in modo da facilitare la fase di testing e migliorare la supportabilità del sito per eventuali future modifiche da apportare.

Per rispettare queste linee guida il codice sarà accompagnato da log e commenti volti a semplificarne la comprensione comportando però un aumento del tempo di sviluppo.

**Prestazioni vs Costi**

Poiché il nostro progetto è sprovvisto di budget, per poter mantenere i costi e i tempi di sviluppo verranno utilizzati dei framework open source esterni, in particolare bootstrap.

**Interfaccia vs Usabilità**

L’interfaccia grafica è stata realizzata in modo da essere semplice e minimale così da risultare chiara e concisa.

Verranno utilizzati form e pulsanti con lo scopo di rendere semplice l’utilizzo del sistema da parte dell’utente finale.

**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.

## **1.2 Linee guida per la documentazione delle interfacce**

Gli sviluppatori dovranno seguire determinate linee guida nella stesura del codice.

### 1.2.1 Naming Convention

È buona norma utilizzare nomi:

* Descrittivi
* Pronunciabili
* Non abbreviati
* Utilizzando Camel Case

Utilizzando solo caratteri consentiti

#### **1.2.1.1 Variabili**

In nomi delle variabili devono iniziare con lettera minuscola e le parole successive che le compongono con lettera maiuscola, ogni riga di codice conterrà una sola dichiarazione di variabile e va effettuato l’allineamento per migliorare la leggibilità

In caso di costanti statiche, utilizzare solo caratteri maiuscoli.

In determinati casi, è possibile utilizzare in carattere underscore ”\_” .

#### 1.2.1.2 Metodi

Inomi dei metodi devono iniziare con la lettera minuscola, e le successive che li compongono von la lettera maiuscola. Di solito il nome del metodo è costituito da un verbo che identifica un’azione, seguito dal nome di un oggetto.

I nomi dei metodi per l’accesso e la modifica delle variabili dovranno essere del tipo getNomeVariabile() e setNomeVariabile().

Ai metodi va aggiunta una descrizione che deve essere posizionata prima della dichiarazione del metodo e che deve descriverne lo scopo.

#### 1.2.1.3 Classi e pagine (JSP/HTML)

I nomi delle classi e delle pagine devono iniziare con la lettera maiuscola, e anche le parole successive che all’interno del nome devono iniziare con la lettera maiuscola. I nomi delle classi e delle pagine devono essere evocativi in modo da fornire informazioni sullo scopo di quest’ultime.

Ogni file sorgente \*.java deve contenere una singola classe e deve essere strutturato nel seguente modo:

1. Introduzione alla classe /\*descrizione della classe\*/
2. Dichiarazione della classe, costituita da:
   1. Dichiarazione della classe pubblica
   2. Dichiarazione delle costanti
   3. Dichiarazione delle variabili d’istanza
   4. Dichiarazione delle variabili statiche
   5. Costruttore di default
   6. Dichiarazione di metodi che definiscono il comportamento della classe

# **2 Package**

La struttura del sistema è modellata secondo una divisione in package e sotto-package che raggruppano le classi ed hanno il compito di gestire la logica i base alle richieste dell’utente

## **2.1 Struttura sistema**

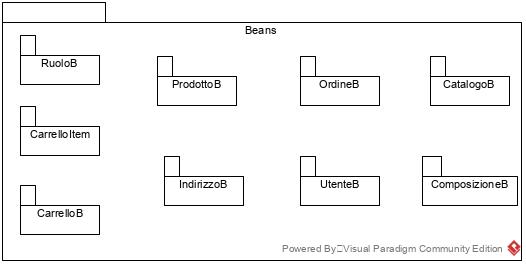


## **2.2 Descrizione delle classi**

### 2.2.1 Data Layer

#### **2.2.1.1 Beans**

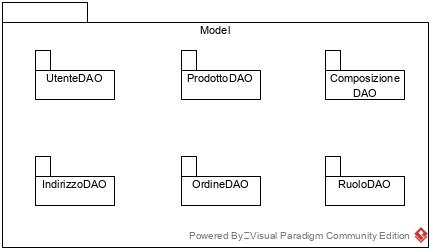
Contiene le classi che modellano le entità del sistema



|  |  |
| --- | --- |
| **Classe** | **Descrizione** |
| UtenteB | Modella un utente registrato |
| RuoloB | Modella il ruolo dell’utente |
| IndirizzoB | Modella un indirizzo dell’utente |
| CatalogoB | Modella il catalogo |
| ProdottoB | Modella un prodotto |
| CarrelloB | Modella il carrello |
| CarrelloItem | Modella un elemento del carrello |
| OrdineB | Modella un ordine |
| ComposizioneB | Modella la composizione di un ordine |

#### **2.2.1.2 Model**

Contiene le classi che modellano la conoscenza sulle entità del sistema

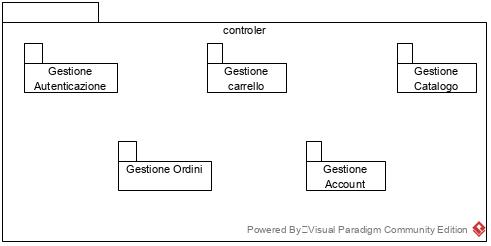


|  |  |
| --- | --- |
| **Classe** | **Descrizione** |
| UtenteDAO | Responsabile della conoscenza dei dari sugli utenti registrati |
| RuoloDAO | Responsabile della conoscenza dei dati sui ruoli degli utenti registrati |
| IndirizzoDAO | Responsabile della conoscenza dei dati sugli indirizzi degli utenti registrati |
| ProdottoDAO | Responsabile della conoscenza dei dati sui prodotti |
| OrdineDAO | Responsabile della conoscenza dei dati sugli ordini |
| ComposizioneDAO | Responsabile della conoscenza dei dati sulle composizioni degli ordini |

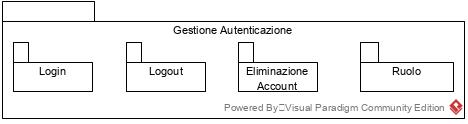
### 2.2.2 Application Layer

#### **2.2.2.1 Controller**

Contiene le Servlet che modellano la logica di business del sistema



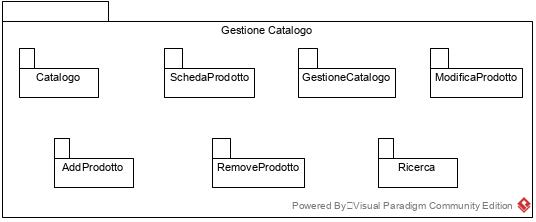
**Gestione Autenticazione**



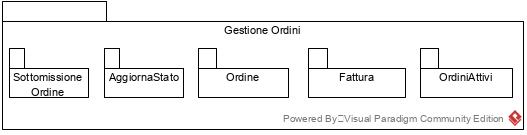
**Gestione Carrello**



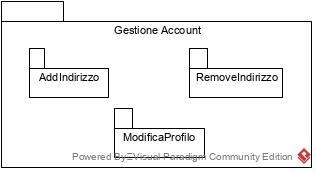
**Gestione Catalogo**



**Gestione Ordini**



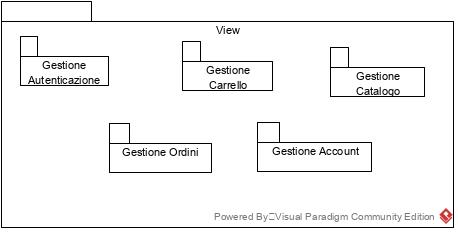
**Gestione Account**



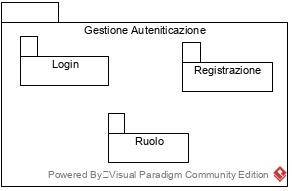
### 2.2.3 PresentationLayer

#### **2.2.3.1 View**

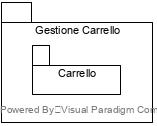
Contiene le pagine JSP e HTML che modellano l’interfaccia grafica del sistema



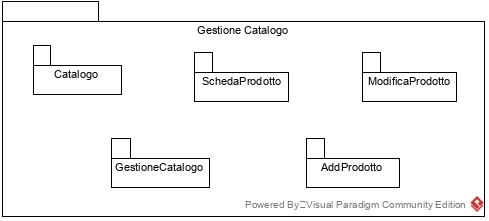
**Gestione Autenticazione**



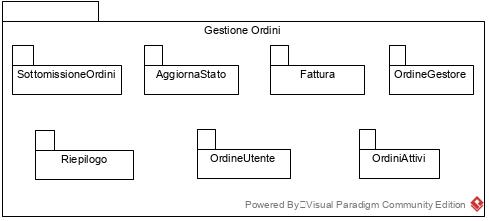
**Gestione Carrello**



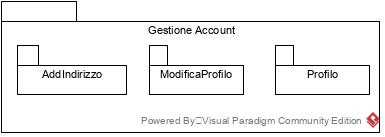
**Gestione Catalogo**



**Gestione Ordini**



**Gestione Account**



# **3 Interfacce delle classi**

**UtenteDAO**

|  |  |
| --- | --- |
| **Servizio** | **Descrizione** |
| public void doSave(UtenteB utente) | Permette di salvare un utente nel DB |
| public UtenteB validate (UtenteB utente) | Permette di verificare che l’username sia presente nel DB e che la password sia quella corrispondente all’account identificato dall’username |
| Public UtenteB doRetrieveByUsername(String Username) | Permette di ottenere un utente dal DB specificando l’username |
| Public boolean doUpdate(UtenteB utente) | Permette di aggiornare I dati di un utente memorizzati nel DB |
| Public Boolean doDelete (UtenteB utente) | Permette di eliminare un utente dal DB |

**RuoloDAO**

|  |  |
| --- | --- |
| **Servizio** | **Descrizione** |
| Public void doSave(RuoloB ruolo) | Permette di salvare un ruolo per un utente nel DB |
| Public Map<String,RuoloB> doRetrieveByUtente(UtenteB utente) | Permette di ottenere I ruoli di un utente dal DB |

**IndirizzoDAO**

|  |  |
| --- | --- |
| **Servizio** | **Descrizione** |
| Public void doSave(IndirizzoB indirizzo) | Permette di salvare un indirizzo per un utente nel DB |
| Public Set<IndirizzoB> doRetrieveByUsername(UtenteB utente) | Permette di ottenere gli indirizzi di un utente dal DB |
| Public boolean doDelete(IndirizzoB indirizzo) | Permette di eliminare un indirizzo nel DB |

**ProdottoDAO**

|  |  |
| --- | --- |
| **Servizio** | **Descrizione** |
| Public void doSave(ProdottoB prodotto) | Permette di salvare un prodotto nel DB |
| Public prodottoB doRetrieveByCodice(String codice) | Permette di ottenere un prodotto dal DB specificandone il codice |
| Public boolean doUpdate(ProdottoB prodotto) | Permette di aggiornare i dati di un prodotto memorizzato nel DB |
| Public boolean doDelete(ProdottoB prodotto) | Permette di eliminare un prodotto dal DB |

**OrdineDAO**

|  |  |
| --- | --- |
| **Servizio** | **Descrizione** |
| Public void doSave(OrdineB ordine) | Permette di salvare un ordine nel DB |
| Public Set<OrdineB> doRetrieveAll() | Permette di ottenere tutti gli ordini dal DB |
| Public set OrdineB doRetrieveByNumero(String numero) | Permette di ottenere un ordine in base al numero |
| Public set<OrdineB> doRetrieveByUtente(UtenteB utente) | Permette di ottenere gli ordini di uno specifico utente |
| publicSet<OrdineB> doRetrieveIfAttivi() | Permette di ottenere tutti gli ordini attivi dal DB |
| Public vpid aggiornaStato(OrdineB ordine) | Permette di aggiornare lo stato di un ordine |
| Public String generatoreSottomissione() | Genera la data di sottomissione dell’ordine |

**ComposizioneDAO**

|  |  |
| --- | --- |
| **Servizio** | **Descrizione** |
| Public void doSave(ComposizioneB composizione) | Permette di salvare un prodotto di un ordine nel DB |
| Public Set<ComposizioneB> doRetrieveByOrdine(OrdineB ordine) | Permette di ottenere dal DB la composizione di un ordine |

**CarrelloB**

|  |  |
| --- | --- |
| **Servizio** | **Descrizione** |
| Public void addProdotto(CarrelloItem carrelloitem) | Permette di aggiungere un prodotto al carrello |
| Public void reAddProdotto(CarrelloItem carrelloitem) | Permette di aggiungere un prodotto al carrello oppure di aumentarne la quantità |
| Public CarrelloItem getProdotto(String codiceProdotto) | Permette di ottenere un prodotto dal carrello specificandone il codice |
| Public void removeProdotto(CarrelloItem carrelloItem) | Permette di rimuovere un prodotto dal carrello |
| Public Set<CarrelloItem> getCarrello() | Permette di ottenere il carrello |
| Public void setCarrello(Set<CarrelloItem> carrello) | Permette di salvare il carrello |
| Public void modificaQt(String codiceProdotto,String action) | Permette di modificare la quantità di un prodotto nel carrello, aumentandola o diminuendola |
| Public Boolean isEmpy(CarrelloItem carrelloItem) | Permette di verificare che il carrello sia vuoto o meno |
| Public Boolean contains(CarrelloItem carrelloItem) | Permette di verificare che un prodotto sia presente nel carrello |
| Public void svuotaCarrello() | Permette di svuotare il carrello |

## **3.1 Definizione Contratti**

### 3.1.1 UtenteDAO

|  |  |
| --- | --- |
| **Classe** | **UtenteDAO** |
| Pre-Condizione | **context** UtenteDAO :: public doSave(UtenteB utente)  **Pre** utente.username!=null && utente.password!=null && utente.nome!=null && utente.cognome!=null && utente.mail!=null && utente!=null  **Context** UtenteDAO :: public validate(UtenteB utente)  **Pre** utente != null && utente.username!=null && utente.password!=null  **Context** UtnteDAO :: public doRetrieveByUsername(String Username)  **Pre** username!= null  **Context** UtenteDAO :: public doUpdate(UtenteB utente)  **Pre** utente.username!=null && utente.password!=null && utente.nome!=null && utente.cognome!=null && utente.mail!=null && doRetrieveByUsername(utente.username) != null && utente!=null  **Context** UtenteDAO :: public doDelete(UtenteB utente)  **Pre** doRetrieveByUsername(utente.username)!=null && utente!=null |
| Post-Condizione | **Context** UtenteDAO :: public doSave(UtenteB utente)  **Post** doRetrieveByUsername(utente.username)!=null  **Context** UtenteDAO :: public validate(UtenteB utente)  **Post** utenteValidato!=null  **Context** UtenteDAO :: public doRetrieveByUsername(String username)  **Post** utenteRestituito!=null  **Context** UtenteDAO :: public doUpadate(UtenteB utente)  **Post** doUpdate==true  **Context** UtenteDAO :: public doDelete(UtenteB utente)  **Post** doDelete==true |
| Invarianti |  |

### 3.1.2 RuoloDAO

|  |  |
| --- | --- |
| **Classe** | **RuoloDAO** |
| Pre-condizioone | **Context** RuoloDAO :: public doSave(RuoloB ruolo)  **Pre** ruolo.username!=null && ruolo.tipo!=null && ruolo!=null  **Context** RuoloDAO :: public doRetrieveByUtente(UtenteB utente)  **Pre** utente.username!=null && utente!=null |
| Post-condizione | **Context** RuoloDAO :: public doSave(RuoloB ruolo)  **Post** ruolo=doRetrieveByUtente(UtenteB utente)  ->Ruolo!=null && ruoli!?empty &&ruoli.contains(ruolo)==true  **Context** RuoloDAO :: public doRetrieveByUtente(UtenteB utente)  **Post** ruoliRestituiti!=null&& ruoliRestituiti!=empty |
| Invarianti |  |

### 3.1.3 IndirizzoDAO

|  |  |
| --- | --- |
| **Classe** | **IndirizzoDAO** |
| Pre-condizioone | **Context** IndirizzoDAO :: public doSave(IndirizzoB indirizzo)  **Pre** indirizzo.username!=null && indirizzo.via!=null && indirizzo.cap!=null && indirizzo.città!=null && indirizzo!=null  **Context** IndirizzoDAO:: public doRetrieveByUtente(UtenteB utente)  **Pre** utente.username!=null && utente!=null  **Context** IndirizzoDAO :: pulblic doDelete(IndirizzoB indirizzo)  **Pre** indirizzi.doRetrieveByUtente(utente)!=null -> indirizzzi!=null && indirizzi!=empty && indirizzi.contains(indirizzo)==true |
| Post-condizione | **Context** IndirizzoDAO:: public doSave(IndirizzoB indirizzo)  **Post** indirizzi=doRetrieveByUtente(UtenteB utente)  -> indirizzo!=null && ruoli!=empty &&indirizzi.contains(indirizzo)==true  **Context** IndirizzoDAO:: public doRetrieveByUtente(UtenteB utente)  **Post** indirizziRestituiti!=null&& indirizziRestituiti!=empty  **Context** IndirizzoDAO :: pulblic doDelete(IndirizzoB indirizzo)  **Post** doDelete=true |
| Invarianti |  |

### 3.1.4 ProdottoDAO

|  |  |
| --- | --- |
| **Classe** | **ProdottoDAO** |
| Pre-condizioone | **Context** ProdottoDAO :: public doSave(ProdottoB prodotto)  **Pre** prodotto.codice!=null && prodotto.prezzo>0 && prodotto.nome!=null && prodotto.descrizione!=null && prodotto!=quantità>0 && prodotto!=null  **Context** ProdottoDAO:: public doRetrieveByCodice(String codice)  **Pre** codice!=null  **Context** ProdottoDAO :: public doUpdate (ProdottoB prodotto)  **Pre** prodotto.codice!=null && prodotto.prezzo>0 && prodotto.nome!=null && prodotto.descrizione!=null && prodotto!=quantità>0 && prodotto!=null  prodtto.doRetrieveByCodice(utente)!=null  **Context** ProdottoDAO:: pulblic doDelete(ProdottoB prodotto)  **Pre** prodtto.doRetrieveByCodice(utente)!=null -> prodotto!=null  **Context** ProdottoDAO :: public Set<ProdottoB> doRetrieveAll()  **pre** |
| Post-condizione | **Context** ProdottoDAO:: public doSave(ProdottoB prodotto)  **Post** indirizzi=doRetrieveByCodice(String codice)!=null  **Context** ProdottoDAO:: public doRetrieveByCodice (String codice)  **Post** prodottoRestituito!=null  **Context** ProdottoDAO :: public doUpdate (ProdottoB prodotto)  **Post** doUpdate==true  **Context** ProdottoDAO:: pulblic doDelete(ProdottoB prodotto)  **Post** doDelete=true  **Context** ProdottoDAO :: public Set<ProdottoB> doRetrieveAll()  **Post** prodottiRestituiti!=null |
| Invarianti |  |

### 3.1.5 OrdineDAO

|  |  |
| --- | --- |
| **Classe** | **OrdineDAO** |
| Pre-condizioone | **Context** OrdineDAO:: public doSave(OrdineB ordine)  **Pre** ordine.n\_fattura!=null && ordine.importo>0 && ordine.username!=null && ordine.stato!=null && ordine.prodotto>0 && ordine!= nulll && composizione!=null  **Context** OrdineDAO:: public doRetrieveByNymero(String numero)  **Pre** numero!=null  **Context** OrdineDAO:: public doRetrieveAll ()  **Pre**  **Context** OrdineDAO:: pulblic doRetrieveByUtente(UtenteB utente)  **Pre** utente.username!=null && utente!=null  **Context** OrdineDAO:: public doRetrieveIfAttivi ()  **Pre**  **Context** OrdineDAO:: public modificaStato (OrdineB ordine)  **Pre** ordine.n\_fattura!=null && ordine.importo>0 && ordine.username!=null && ordine.stato!=null  && doRetrieveByNumero(ordine.n\_fattura) !=null && ordine!=null |
| Post-condizione | **Context** OrdineDAO:: public doSave(OrdineB ordine)  **Post** doRetrieveByNumero(ordine.n\_fattura)!=null  **Context** OrdineDAO:: public doRetrieveByNymero(String numero)  **Post** ordineRestituito!=null  **Context** OrdineDAO:: public doRetrieveAll ()  **Post** ordineRestituito !=null  **Context** OrdineDAO:: pulblic doRetrieveByUtente(UtenteB utente)  **Post** ordineRestituito !=null  **Context** OrdineDAO:: public doRetrieveIfAttivi ()  **Post** ordiniAttivi!=null && ordiniAttivi.stato!=Consegnato  **Context** OrdineDAO:: public modificaStato (OrdineB ordine)  **Post** ordineAggiornato=doRetrieveByNuemro(ordine.n\_fattura) -> ordienAggiornato!=null && ordineAggiornato.stato==ordine.stato |
| Invarianti |  |

### 3.1.6 ComposizioneDAO

|  |  |
| --- | --- |
| **Classe** | **ComposizioneDAO** |
| Pre-condizioone | **Context** ComposizioneDAO:: public doSave(ComposizioneB composizione)  **Pre** composizione.ordine!=null && composizione.prodotto!=null && composizione.nomeprodotto!=null && composizione.quantita>0 composizione.prezzo>0 && composizione!=null  **Context** ComposizioneDAO:: public doRetrieveByOrdine(OrdineB ordine)  **Pre** ordine.numero!=null && ordine!=null |
| Post-condizione | **Context** ComposizioneDAO:: public doSave(ComposizioneB composizione)  **Post** composizione=doRetrieveByOrdine(ordine)!=null ->composizioni!=null &&composizioni!=empty &&composizioni.contains(composizione)==true  **Context** ComposizioneDAO:: public doRetrieveByOrdine(OrdineB ordine)  **Post** composizioni!=null && composizioni!=empty |
| Invarianti |  |

### 3.1.7 CarrelloB

|  |  |
| --- | --- |
| **Classe** | **CarrelloB** |
| Pre-condizioone | **Context** CarrelloB:: public addProdotto(CarrelloItem carrello)  **Pre** carrelloItem.prodotto!=null && carrelloItem.quantità>0 && carrelloItem!=null  **Context** CarrelloB:: public reAddProdotto(CarrelloItem carrello)  **Pre** carrelloItem.prodotto!=null && carrelloItem.quantità>0 && carrelloItem!=null  **Context** CarrelloB :: public getProdotto(String codiceProdotto)  **Pre** codiceProdotto!=null  **Context** CarrelloB :: public removeProdotto(CarelloItem carrelloItem)  **Pre** carrrelloItem.prodotto!=null && carrelloItem.quantità>0 && carrello!=null  **Context** CarrelloB :: public getCarrello()  **Pre**  **Context** CarrelloB :: public setCarrello(Set<CarrelloItem> carrello)  **Pre** carrello!=null && for each carrelloItem in carrello -> carrelloItem.prodotto!=null && carrelloItem.quantita>0 && carrelloItem!=null  **Context** CarrelloB :: public modificaQt(String codiceProdotto, String action)  **Pre** codiceProdotto!=null && action!=null  **Context** CarrelloB :: public contains(CarrelloItem carrelloItem)  **Pre** carrelloItem.prodotto!=null && carrelloItem.quantita>0 && carrelloItem!=null  **Context** CarrelloB :: public svuotaCarrello()  **Pre** |
| Post-condizione | **Context** CarrelloB:: public addProdotto(CarrelloItem carrello)  **Post** getProdotto(CarrelloItem carrelloItem)==carrelloItem  **Context** CarrelloB:: public reAddProdotto(CarrelloItem carrello)  **Post** if carrello.contains(carrelloItem)==true  -> incrementa getProdotto(carrelloItem.prodotto).quantita  Else  ->getProdotto(carrelloItem.prodotto)== carrelloItem  **Context** CarrelloB :: public getProdotto(String codiceProdotto)  **Post** prodottoRestituito!=null  **Context** CarrelloB :: public removeProdotto(CarelloItem carrelloItem)  **Post** carrello.contains(carrelloItem)==false  **Context** CarrelloB :: public getCarrello()  **Post**  **Context** CarrelloB :: public setCarrello(Set<CarrelloItem> carrello)  **Post** carrello.getCarrello()==carrello  **Context** CarrelloB :: public modificaQt(String codiceProdotto, String action)  **Post** if action==plus ->incrementa getProdotto(codiceProdotto).quantita  Else  ->Decrementa getProdotto(codiceProdotto).quantita  **Context** CarrelloB :: public contains(CarrelloItem carrelloItem)  **Post** true if carrelloItem in carrello  False altrimenti  **Context** CarrelloB :: public svuotaCarrello()  **Post** carrello==empty |
| Invarianti |  |

# **4.Design Pattern**

## **4.1 MVC**

Il pattern architetturale model-view-contreoller (MVC) è molto diffuso nello sviluppo di sistemi software e permette la separazione della logica di business dalla logica di presentazione dati.

MVC prevede un’architettura Three-thier:

Model: responsabile della conoscenza del dominio applicativo e implementa lo store principale dei dati

View: responsabile di mostrare gli oggetti del dominio applicativo all’utente

Controller: responsabile dell’interazione con l’utente e di notificare le View dei cambiamenti nel Model. Esso detta esplicitamente il flusso di controllo

