*Object Design*

*Document*

*Progetto*

*OctoPlus*



**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Data | Versione | Descrizione | Autore |
| 06/12/2023 | **0.1** | **Prima stesura** | **Tutto il team** |
| 07/12/2023 | **0.2** | **Aggiunta sezione package + revisione documento** | **Donnarumma Salvatore** |
| 08/12/2023 | **0.3** | **Aggiunti interfaccia admin e prodotto** | **Tomeo Orlando** |
| 09/12/2023 | **0.4** | **Aggiunta interfaccia Ordini** | **Donnarumma Salvatore** |
| 20/01/2023 | **0.5** | **Modifica e correzione dei Dao** | **Donnarumma Salvatore** |
| 22/01/2024 | **0.6** | **Revisione** | **Tutto il team** |
| 26/01/2024 | **0.7** | **Aggiunto Layer Storage** | **Orlando Tomeo** |
| 27/01/2024 | **0.8** | **Modificati tutti i grafici** | **Donnarumma Salvatore** |
| 28/01/2024 | **0.9** | **Modifica linguaggio OCL** | **Donnarumma Salvatore** |
| 04/01/2024 | **1.0** | **Revisione** | **Tutto il team** |

Sommario

[1. Introduzione 3](#_Toc157806823)

[1.1. Object design trade-offs 3](#_Toc157806824)

[1.1.1. Robustezza vs Tempo 3](#_Toc157806825)

[1.1.2. Sicurezza vs Tempo 4](#_Toc157806826)

[1.2. Linee guida 4](#_Toc157806827)

[1.3. Referenze 4](#_Toc157806828)

[2. Packages 4](#_Toc157806829)

[2.1. Presentation Layer Sito 4](#_Toc157806830)

[2.2. Presentation Layer Utente 5](#_Toc157806831)

[2.3. Presentation Layer Gestore 5](#_Toc157806832)

[2.4. Presentation Layer Prodotti 5](#_Toc157806833)

[2.5. Presentation Layer Carrello 6](#_Toc157806834)

[2.6. Presentation Layer Ordini 6](#_Toc157806835)

[2.7. Presentation Layer Carta 6](#_Toc157806836)

[2.8. Gestione Utenti 6](#_Toc157806837)

[2.9. Gestione Prodotti 7](#_Toc157806838)

[2.10. Gestione Ordini 7](#_Toc157806839)

[2.11. Gestione Carrello 8](#_Toc157806840)

[2.12. Gestione Carta 8](#_Toc157806841)

[3. Interfacce di classe 9](#_Toc157806842)

[3.1. UtenteDaoDataSource 9](#_Toc157806843)

[3.2. GestoreDaoDataSource 11](#_Toc157806844)

[3.3. OrdiniDaoDataSource 12](#_Toc157806845)

[3.4. Prodotto Model IDS 13](#_Toc157806846)

[3.5. Carrello Model IDS 15](#_Toc157806847)

[3.6. Carta Model IDS 15](#_Toc157806848)

# 1. Introduzione

## Object design trade-offs

### Robustezza vs Tempo

Nei primi tempi non sono previsti picchi di utenza elevati, pertanto si è deciso di ritardare l’implementazione di funzionalità avanzate che renderebbero il sistema più robusto e capace di gestire uno stress maggiore, questo per velocizzare il lancio della prima versione.

### Sicurezza vs Tempo

Si ritiene che la sicurezza dei dati degli utenti registrati e l’attendibilità della piattaforma siano caratteristiche necessarie che la piattaforma dovrebbe avere fin dalla prima versione al fine di tutelare i clienti. Il tutto, naturalmente, richiede tempo per lo sviluppo il quale sarà sottratto alla robustezza.

## Linee guida

Qui di seguito sono riportate alcune linee guide per la stesura del codice:

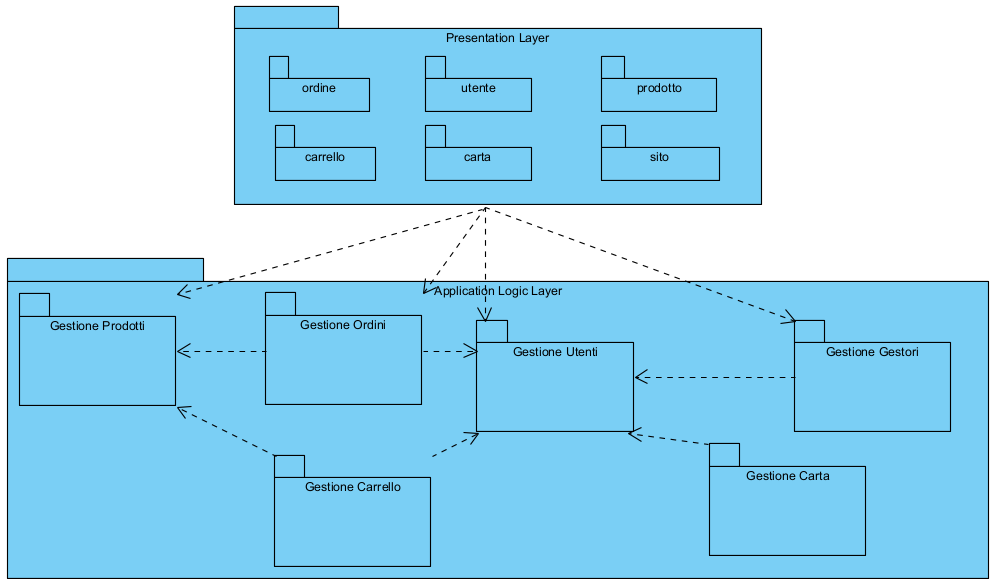
* Gli oggetti Dao dovranno avere il suffisso DaoDataSource nel nome.
* Le classi che identificano le entità devono essere chiamate con nomi singolari.
* I nomi dei file jsp devono essere totalmente in minuscolo mentre i nomi delle Servlet e classi Bean, così come i DAO dovranno iniziare con la prima lettera in maiuscolo.

## Referenze

* **R.A.D.**
* **S.D.D.**

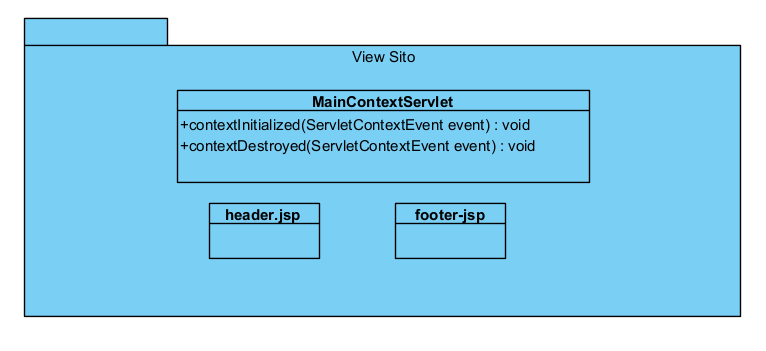
# Packages

Nella sezione presente vengono mostrati nel dettaglio i packages implementati nel nostro sistema. Il pacchetto Presentation Layer contiene tutti i boundary object del sistema, il pacchetto Application Logic Layer conterrà entity objects, control objects e DAO del sistema. È stata scelta questa soluzione poiché i DAO verranno implementati sulla componente server e non database.



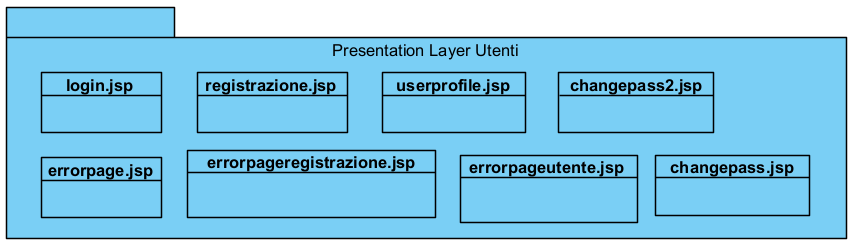
## Presentation Layer Sito

Questo sottopacchetto è composto dai file jsp adibite alle funzioni generali del sito come footer, header, etc.



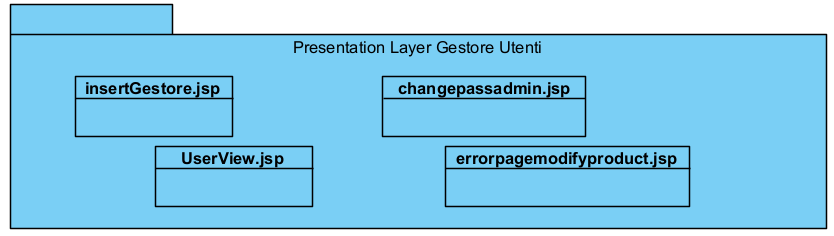
## Presentation Layer Utente

Questo pacchetto è composto dai file jsp relativi alle funzionalità per la gestione degli utenti, come autenticazione, registrazione, cambio password, etc.



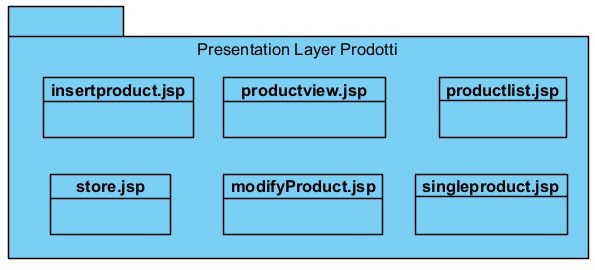
## Presentation Layer Gestore

Questo pacchetto è composto dai file jsp relativi alle funzionalità per la gestione degli utenti da parte del Gestore Utente come creazione di nuovi gestori, cambio password degli altri gestori, ordinamento degli utenti etc.



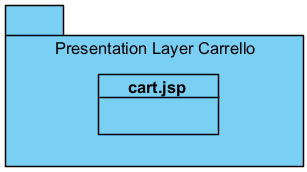
## Presentation Layer Prodotti

Questo pacchetto è composto dai file jsp relativi alle funzionalità per la gestione dei prodotti come rimozione degli stessi dal catalogo, aggiunta, modifica, etc.



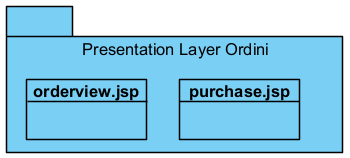
## Presentation Layer Carrello

Questo pacchetto è composto dai file jsp relativi alle funzionalità per la gestione del carrello come rimozione dei prodotti da esso, aggiunta e visualizzazione.



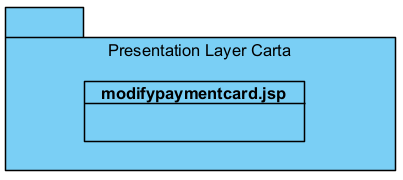
## Presentation Layer Ordini

Questo pacchetto è composto dai file jsp relativi alle funzionalità per la gestione degli ordini come rimozione degli ordini, creazione dell’ordine (dunque acquisto di prodotti), visualizzazione, etc.



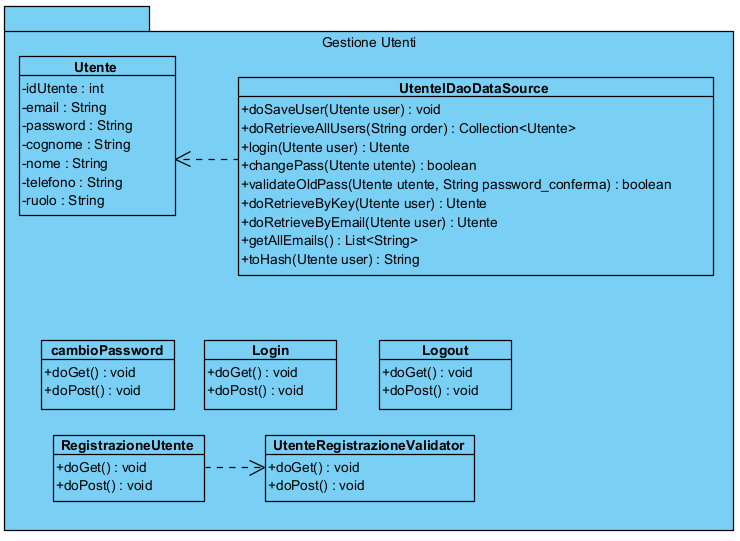
## Presentation Layer Carta

Questo pacchetto è composto dai jsp relativi alle funzionalità per la gestione della carta come il suo salvataggio o rimozione etc.

****

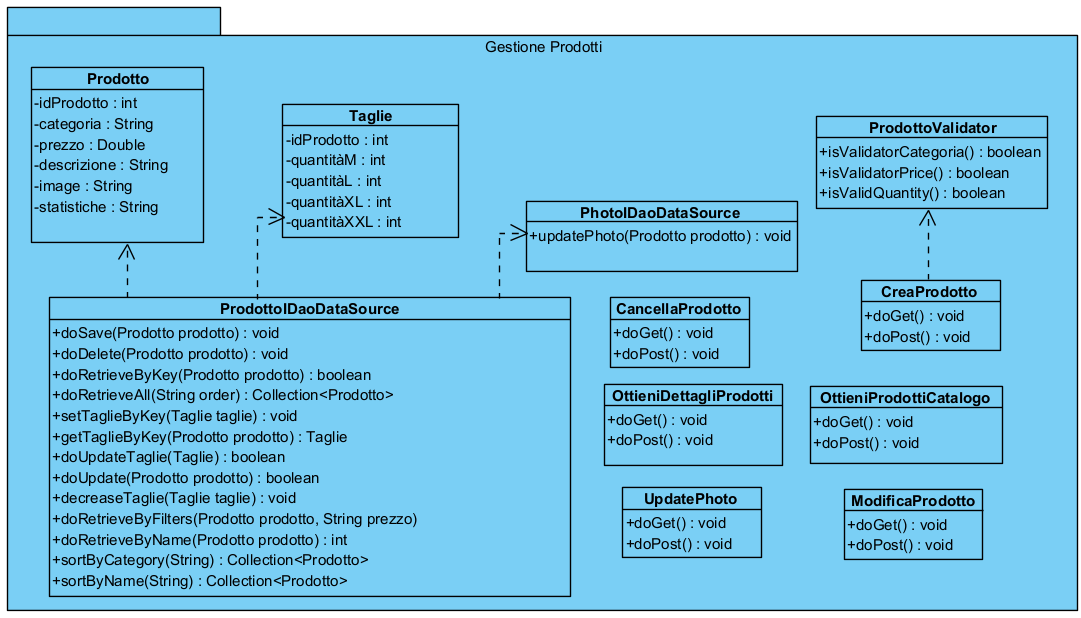
## Gestione Utenti

Questo pacchetto contiene le classi Java (Control Objects, Entity Objects e DAO) dedicati alle funzionalità per la gestione degli utenti e informazioni relative ad essi.



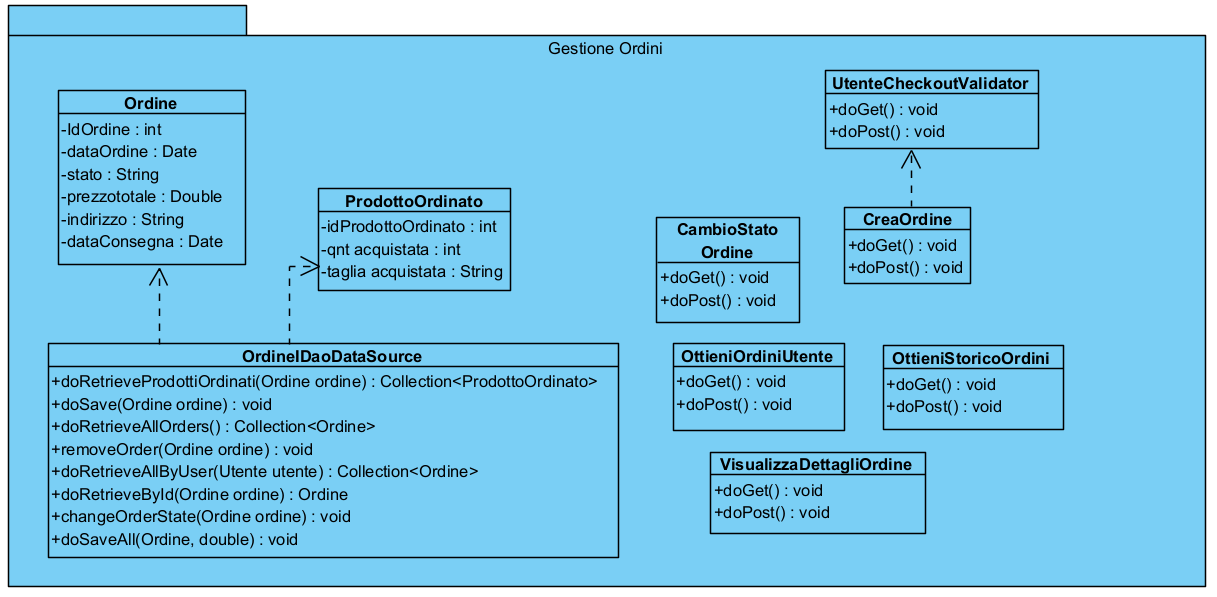
## Gestione Prodotti

Questo pacchetto contiene le classi Java (Control Objects, Entity Objects e DAO) dedicati alle funzionalità per la gestione dei prodotti e informazioni relative ad essi.

****

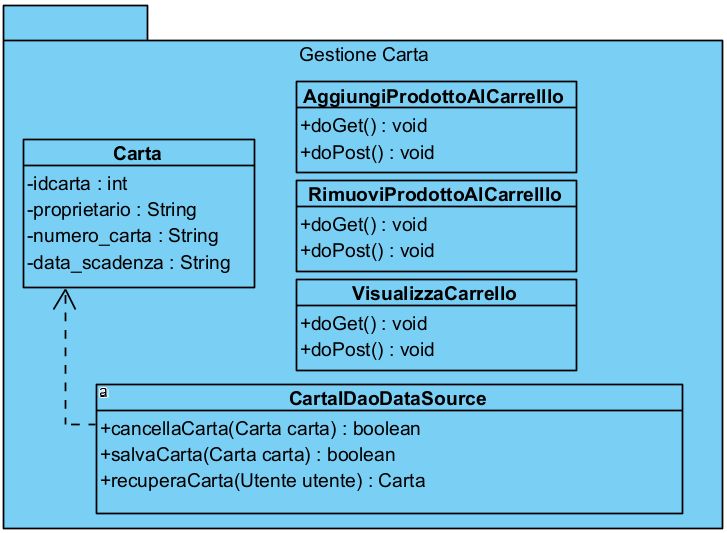
## Gestione Ordini

Questo pacchetto contiene le classi Java (Control Objects, Entity Objects e DAO) dedicati alle funzionalità per la gestione degli ordini e informazioni relative ad essi.

****

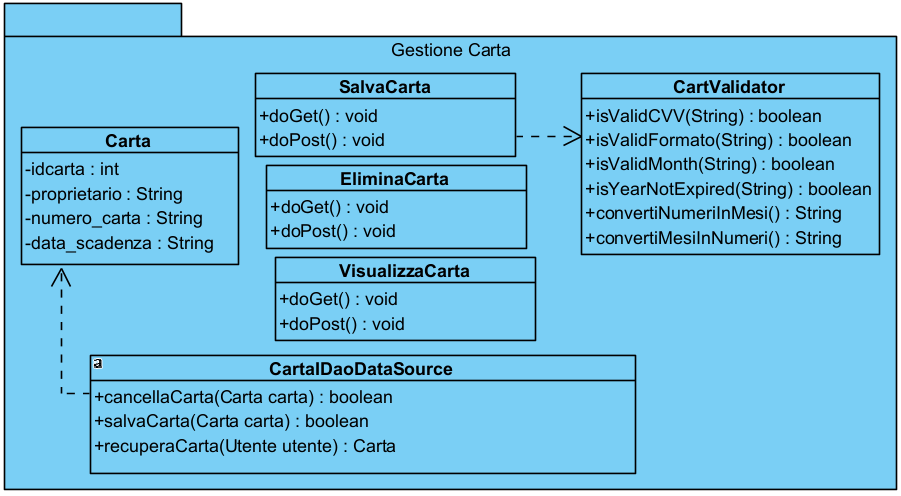
## Gestione Carrello

Questo pacchetto contiene le classi Java (Control Objects, Entity Objects e DAO) dedicati alle funzionalità per la gestione degli ordini e informazioni relative ad essi.

****

## Gestione Carta

Questo pacchetto contiene le classi Java (Control Objects, Entity Objects e DAO) dedicati alle funzionalità per la gestione della carta.



# Interfacce di classe

## UtenteDaoDataSource

|  |  |
| --- | --- |
| UtenteDaoDataSource | Questa classe permette di interfacciarci al DBMS relazione modificando e interrogando l’entità Utente. |
| Precondizione | **PostCondizione** |
| context *UtenteDaoDataSource::doRetrieveByKey(utente : Utente)*  pre: utente.email <> null and utente.email <> “” | **context** UtenteDaoDataSource::doRetrieveByKey(utente: Utente)  **post:**  if DB.users->exists(u | u.email = utente.email)  then  result <> null  and result instanceof Utente  and result.email = utente.email  and result.nome = utente.nome  and result.cognome = utente.cognome  and result.numerotelefono = utente.numerotelefono  else  result = null  endif |
| context UtenteDaoDataSource::doSaveUser(utente: Utente)  pre:  utente.email <> null and utente.email <> ""  and UtenteValidator::isValidEmail(utente.email)  and UtenteValidator::isValidPassword(utente.password) and utente.password = utente.passwordConferma  and utente.nome <> null and utente.nome <> ""  and utente.cognome <> null and utente.cognome <> ""  and UtenteValidator::isValidTelefono(utente.numerotelefono)  and utente.ruolo <> null and utente.ruolo <> “”  -- Funzione per verificare il formato dell'email  def: isValidEmail(email: String): Boolean =  email.matches('^[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}$')  -- Funzione per verificare il formato della password (almeno 12 caratteri)  def: isValidPassword(password: String): Boolean =  password.size() >= 12  -- Funzione per verificare il formato del numero di telefono  def: isValidPhoneNumber(numerotelefono: String): Boolean = numerotelefono.matches("^\\d{3}-\\d{7}$"); | **context** UtenteDaoDataSource::doSaveUser(utente: Utente)  **post:**  DB.users->exists(u | u.email = utente.email  u.password = utente.password  and u.nome = utente.nome  and u.cognome = utente.cognome  and u.numerotelefono = utente.numerotelefono  and u.ruolo = utente.ruolo  ) |
| context UtenteDaoDataSource::doRetrieveAllUsers(order: String)  pre:  order = null or order = "Email" or order = "Cognome" | **context** UtenteDaoDataSource::doRetrieveAllUsers(order: String)  **post:**  result <> null  and result->forAll(u | u instanceof Utente) |
| context UtenteDaoDataSource::changePass(utente: Utente)  pre:  utente.email <> null and utente.email <> ""  and utente.password <> null and utente.password <> ""  and UtenteValidator.isValidPassword(utente.password) | **context** UtenteDaoDataSource::changePass(utente: Utente)  **post:**  result =  DB.users->exists(u | u.email = utente.email and u.password = utente.password)  implies  (  DB.users->select(u | u.email = utente.email)->forAll(u | u.password = utente.nuovaPassword)  ) |
| context UtenteDaoDataSource::validateOldPassword(old\_pass: Utente, new\_pass: Utente)  pre:  old\_pass.password <> null and old\_pass.password <> ""  and new\_pass.password <> null and new\_pass.password <> "" | **context** UtenteDaoDataSource::validateOldPassword(old\_pass: Utente, new\_pass: Utente)  **post:**  result =  DB.users->exists(u | u.email = old\_pass.email and u.password = old\_pass.password) |

## GestoreDaoDataSource

|  |  |
| --- | --- |
| GestoreDaoDataSource | Questa classe permette di interfacciarci al DBMS relazione modificando e interrogando l’entità Utente. |
| Precondizione | **PostCondizione** |
| context *GestoreDaoDataSource::doSaveUser(utente: Utente)*  pre:  utente.email <> null and utente.email <> ""  and UtenteValidator::isValidEmail(utente.email)  and UtenteValidator::isValidPassword(utente.password) and utente.password = utente.passwordConferma  and utente.nome <> null and utente.nome <> ""  and utente.cognome <> null and utente.cognome <> ""  and UtenteValidator::isValidTelefono(utente.numerotelefono)  and utente.ruolo <> null and utente.ruolo <> “” | **context** *GestoreDaoDataSource::doSaveGestore(Utente utente)*  **post:**  DB.users->exists(u | u.email = utente.email  u.password = utente.password  and u.nome = utente.nome  and u.cognome = utente.cognome  and u.numerotelefono = utente.numerotelefono  and u.ruolo = utente.ruolo  ) |
| context *GestoreDaoDataSource::doRetrieveUtentiSorted(String order)*  pre: order == null or order == “Email” or order == ”Cognome” | **context** *GestoreDaoDataSource::doRetrieveUtentiSorted(String order)*  **post:**  result <> null  and result->forAll(u | u instanceof Utente)  and (  (order = "Email" and result = result->sortedBy(e | e.email)) or  (order = "Cognome" and result = result->sortedBy(e | e.cognome))  or result = result->sortedBy(e | e.idutente)  ) |
| context *GestoreDaoDataSource::changePassGestore(Utente utente)*  pre:  utente.email <> null and utente.email <> ""  and utente.password <> null and utente.password <> ""  and UtenteValidator.isValidPassword(utente.password) | **context** GestoreDaoDataSource::changePass(utente: Utente)  **post:**  result =  if DB.gestori->exists(g | g.email = utente.email and g.password = utente.oldPassword)  then  DB.gestori->select(g | g.email = utente.email)->forAll(g | g.password = utente.newPassword)  else  false  endif |
| context *GestoreDaoDataSource::validateOldPassword(Utente old\_pass, Utente new\_pass)*  pre: old\_pass.password <>null and old.password <>“” and new\_pass.password <>null and new\_pass.password <>“” | ***context*** *GestoreDaoDataSource::validaOldPassword(utente: Utente)*  **post:**  result =  if DB.gestori->exists(g | g.email = utente.email and g.password = utente.oldPassword)  then  DB.gestori->select(g | g.email = utente.email)->forAll(g | g.password = utente.newPassword)  else  false  endif |
| context *GestoreDaoDataSource::doDeleteGestore(Utente utente)*  pre: utente.email <>null and utente.email <>“” | **context** GestoreDaoDataSource::doDeleteGestore(utente: Utente)  **post:**  not DB.gestori->exists(g | g.email = utente.email) |

## OrdiniDaoDataSource

|  |  |
| --- | --- |
| OrdineDaoDataSource | Questa classe permette di interfacciarci al DBMS relazione modificando e interrogando l’entità Ordine. |
| Precondizione | **PostCondizione** |
| context *OrdineDaoDataSource::doRetrieveByKeyO(ordine : Ordine)*  pre: ordine <> null and ordine.id <> null and ordine.id <> “” | **context** OrdineDaoDataSource::doRetrieveByKeyO(ordine: Ordine)  **post:**  result <> null  and result->forAll(po | po instanceof ProdottoOrdinato and po.id\_ordine = ordine.id\_ordine) |
| context OrdiniDaoDataSource::doSave(ordine: Ordine)  pre:  ordine.id <> null and ordine.id <> ""  and ordine.dataOrdine <> null and ordine.dataOrdine <> ""  and ordine.stato <> null and ordine.stato <> ""  and ordine.prezzototale <> null and ordine.prezzototale <> ""  and ordine.indirizzo <> null and ordine.indirizzo <> ""  and ordine.dataConsegna <> null and ordine.dataConsegna <> "" | **context** OrdiniDaoDataSource::doSave(ordine: Ordine)  **post:**  DB.ordini->exists(o | o.id = ordine.id  and o.dataOrdine = ordine.dataOrdine  and o.stato = ordine.stato  and o.prezzototale = ordine.prezzototale  and o.indirizzo = ordine.indirizzo  and o.dataConsegna = ordine.dataConsegna)  and DB.prodottiOrdinati->exists(po | po.id\_ordine = ordine.id and po.id\_prodotto = ordine.prodotto.id and po.quantita = ordine.prodotto.quantita) |
| context O*rdiniDaoDataSource::doSaveAll(ordine : Ordine, totp : Double)*  pre:  ordine.id <> null and ordine.id <> ""  and ordine.dataOrdine <> null and ordine.dataOrdine <> ""  and ordine.stato <> null and ordine.stato <> ""  and ordine.prezzototale <> null and ordine.prezzototale <> ""  and ordine.indirizzo <> null and ordine.indirizzo <> ""  and ordine.dataConsegna <> null and ordine.dataConsegna <> "" and totp <> null | **context** OrdiniDaoDataSource::doSaveAll(ordine: Ordine, totp : Double)  **post:**  DB.ordini->exists(o | o.id = ordine.id  and o.dataOrdine = ordine.dataOrdine  and o.stato = ordine.stato  and o.prezzototale = totp  and o.indirizzo = ordine.indirizzo  and o.dataConsegna = ordine.dataConsegna)  and DB.prodottiOrdinati->exists(po | po.id\_ordine = ordine.id and po.id\_prodotto = ordine.prodotto.id and po.quantita = ordine.prodotto.quantita) |
| context *OrdiniDaoDataSource::doRetrieveAllOrders()*  pre: | **context** OrdiniDaoDataSource::doRetrieveAllOrders()  **post:**  result <> null and result->forAll(o | o instanceof Ordine)  or (result = null and not DB.ordini->exists()) |
| context *OrdiniDaoDataSource::changeOrderState(Ordine ordine)*  pre: ordine <> null and ordine.id <>null and ordine.id = “” and ordine.stato <>null and ordine.stato <>“” | **context** OrdiniDaoDataSource::changeOrderState(ordine: Ordine)  **post:**  DB.ordini->exists(o | o.id = ordine.idOrdine)  and DB.ordini->forAll(o | (o.id = ordine.idOrdine) implies (o.stato = ordine.nuovoStato)) |

## Prodotto Model IDS

|  |  |
| --- | --- |
| ProdottoDaoDataSource | Questa classe permette di interfacciarci al DBMS relazione modificando e interrogando l’entità Prodotto. |
| Precondizione | **PostCondizione** |
| context *ProdottoDaoDataSource::doSave(Prodotto prodotto)*  pre: prodotto <> null and prodotto.categoria <> null and prodotto.categoria <> “” and prodotto.nome <> null and prodotto.nome <> “” and prodotto.prezzo>=0 and prodotto.descrizione <> “” and prodotto.descrizione <> nul  and prodotto.imagePath <> null and prodotto.statistiche <> “” and prodotto.statistiche <> “” | **context** ProdottoDaoDataSource::doSave(prodotto: Prodotto)  **post:**  DB.prodotti->exists(p | p.id = prodotto.id  and p.nome = prodotto.nome  and p.categoria = prodotto.categoria  and p.prezzo = prodotto.prezzo  and p.descrizione = prodotto.descrizione  and p.imagePath = prodotto.imagePath  and p.statistiche = prodotto.statistiche) |
| context Prodotto*DaoDataSource::doDelete(Prodotto*  *pre: prodotto <> null and DB.prodotti->exists(p | p.id = idProdotto)* | **context** Prodotto*DaoDataSource::doDelete(Prodotto prodotto)*  **post:**  not DB.prodotti->exists(p | p.id = prodotto.id) |
| context Prodotto*DaoDataSource::doRetrieveAll(String order)*  pre: order == null or order == “Nome” or order == ”Categoria” | **context** ProdottoDaoDataSource::doRetrieveAll(order: String)  **post:**  result <> null and result->forAll(p | p instanceof Prodotto)  or (result = null and not DB.prodotti->exists()) |
| context Prodotto*DaoDataSource::doRetrieveByName(Prodotto prodotto)*  pre: prodotto <> null prodotto.nome <>null and prodotto.nome <>“” | **context** ProdottoDaoDataSource::doRetrieveByName(prodotto: Prodotto)  **post:**  result <> null implies (result instanceof Integer or result = null)  and (result = null implies not DB.prodotti->exists(p | p.nome = prodotto.nome)) |
| context Prodotto*DaoDataSource::doRetrieveByKey(Prodotto prodotto)*  pre: prodotto <> null and prodotto.id <>null and prodotto.id <> “” | **context** ProdottoDaoDataSource::doRetrieveByKey(prodotto: Prodotto)  **post:**  result <> null  and result->forAll(p | p instanceof Prodotto and p.id = prodotto.id) |
| context ProdottoDaoDataSource::setTaglieByKey(prodotto: Prodotto, taglie: Taglie)  pre:  prodotto.id <> null and taglie.id <> null and DB.prodotti->exists(p | p.id = prodotto.id)  and prodotto.id <> "" and taglie.id <> "" | **context** ProdottoDaoDataSource::setTaglieByKey(prodotto: Prodotto, taglie: Taglie)  **post:**  DB.taglie->forAll(t | t.id\_prodotto = prodotto.id implies t = taglie) |
| context Prodotto*DaoDataSource::getTaglieByKey(Prodotto prodotto)*  pre: prodotto <> null and prodotto.id <>null and prodotto.id <> “” | **context** ProdottoDaoDataSource::getTaglieByKey(prodotto: Prodotto)  **post:**  result <> null and result instanceof Taglie  and DB.taglie->exists(t | t.id\_prodotto = prodotto.id and t = result) |
| context ProdottoDaoDataSource::doUpdate(prodotto: Prodotto)  pre:  prodotto.id <> null and prodotto <> “” and DB.prodotti->exists(p | p.id = prodotto.id and p = prodotto) | **context** ProdottoDaoDataSource::doUpdate(prodotto: Prodotto)  **post:**  DB.prodotti->exists(p | p.id = prodotto.id and p = prodotto)  implies (DB.prodotti->forAll(p | (p.id = prodotto.id and p = prodotto))  and result = Boolean::TRUE)  and (not DB.prodotti->exists(p | p.id = prodotto.id and p = prodotto)  implies result = Boolean::FALSE) |
| context ProdottoDaoDataSource::doUpdateTaglie(taglie: Taglie)  pre:  DB.prodotti->exists(p | p.id = taglie.id\_prodotto)  and DB.taglie->exists(t | t.id = taglie.id)  and taglie.id\_prodotto <> null  and taglie.id <> null | **context** ProdottoDaoDataSource::doUpdateTaglie(taglie: Taglie)  **post:**  DB.taglie->forAll(t | (t.id\_prodotto = taglie.id\_prodotto and t = taglie)) |

## Carrello Model IDS

|  |  |
| --- | --- |
| CarrelloDaoDataSource | Questa classe permette di interfacciarci al DBMS relazione modificando e interrogando l’entità Carrello. |
| Precondizione | **PostCondizione** |
| context *CarrelloDaoDataSource::eliminaCarrello(Utente utente)*  pre: utente <> null and utente.id <>“” and utente.id <>null | **context** CarrelloDaoDataSource::eliminaCarrello(utente: Utente)  **post:**  DB.prodottiCarrello->forAll(pc | pc.id\_utente = utente.id implies DB.prodottiCarrello->excludes(pc)) |
| context *CarrelloDaoDataSource::recuperaCarrello(Utente utente)*  pre: utente<> null and utente.id <>“” and utente.id <>null | **context** CarrelloDaoDataSource::recuperaCarrello(utente: Utente)  **post:**  result <> null and result->forAll(p | DB.prodottiCarrello->exists(pc | pc.id\_utente = utente.id and pc.prodotto = p)) |
| context *CarrelloDaoDataSource::salvaCarrello(Carrello carrello)*  pre: carrello <> null and carrello.id <> null and carrello.id <> “” | **context** CarrelloDaoDataSource::salvaCarrello(carrello: Carrello)  **post:**  DB.prodottiCarrello->forAll(pc | pc.id\_utente = carrello.id\_utente and carrello.prodotti->includes(pc.prodotto)) |

## Carta Model IDS

|  |  |
| --- | --- |
| CartaDaoDataSource | Questa classe permette di interfacciarci al DBMS relazione modificando e interrogando l’entità Carta. |
| Precondizione | **PostCondizione** |
| context *CarrelloDaoDataSource::cancellaCarta(Carta carta*  pre: carta <> null and carta.idcarta <>“” and carta.idcarta <>null | **context** CarrelloDaoDataSource::cancellaCarta(carta: Carta)  **post:**  if DB.carte->exists(c | c.idCarta = carta.idCarta) then  result = Boolean::TRUE  else  result = Boolean::FALSE  endif |
| context *CarrelloDaoDataSource::salvaCarta(Carta carta)*  pre: carta <> null and carta.idcarta <>“” and carta.idcarta <>null and carta.numero\_carta <>“” and carta.numero\_carta <>null and carta.data\_scadenza <>null and carta.data\_scadenza <>“” and carta.proprietario <>null and carta.proprietario <>“” | **context** CarrelloDaoDataSource::salvaCarta(carta: Carta)  **post:**  result = DB.carte->exists(c | c.idCarta = carta.idCarta and c.numeroCarta = carta.numeroCarta and c.utente = carta.utente and c.dataScadenza = carta.dataScadenza) |
| context *CarrelloDaoDataSource::recuperaCarta(Utente utente)*  pre: utente <> null and utente.idutente <>“” and utente.idutente <>null | **context** CarrelloDaoDataSource::recuperaCarta(utente: Utente)  **post:**  result = DB.carte->any(c | c.utente = utente and c.idCarta = utente.idUtente) |