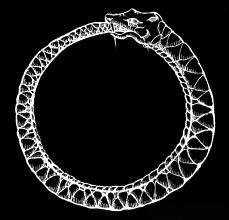
|  |
| --- |
| **Università degli Studi di Salerno Corso di Ingegneria del Software** |

**Uroboros  
Object Design Document  
Versione 1.0**



Data: 27/10/2020

**Coordinatore del progetto:**

|  |  |
| --- | --- |
| Nome | Matricola |
| Andrea Amorosini | 005714 |
|  |  |

**Partecipanti:**

|  |  |
| --- | --- |
| Nome | Matricola |
| Andrea Amorosini | 005714 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
| **Scritto da:** | Andrea Amorosini |

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Data | Versione | Descrizione | Autore |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Indice

1. INTRODUZIONE 5

1.1. Guideline per la documentazione dell’Interfaccia 5

1.2. Definizioni, acronimi e abbreviazioni 6

1.3. References 6

2. PACKAGES 6

2.1. ControlloAccount 7

2.1.1. Class Login 7

2.1.2. Class Logout 8

2.1.3. Class ModificaDatiAccount 8

2.1.4. Class Registrazione 9

2.1.5. Class ShowAccountDetail 10

2.1.6. Class GestioneAccount 10

2.1.7. Class VerificaUsername 11

2.1.8. Class DeleteAccount 11

2.1.9. Class ShowAccount 12

2.2. controlloProdotti 13

2.2.1. Prodotto 13

2.2.2. Catalogo 14

2.2.3. Search 14

2.2.4. GestioneProdotti 15

2.2.5. AggiungiProdotto 16

2.2.6. CancellaProdotto 16

2.2.7. ModificaDatiProdotto 17

2.2.8. ModificaProdottoPage 18

2.3. controlloCategorie 18

2.3.1. Categoria 19

2.3.2. GestioneCategorie 19

2.3.3. AggiungiCategoria 20

2.3.4. DeleteCategoria 21

2.3.5. ShowCategoriaDetails 21

2.3.6. AggiornaCategoria 22

2.4. controlloOrdini 23

2.4.1. Storico 23

2.4.2. ProdottiOrdine 24

2.4.3. ShowStoricoAdmin 24

2.5. ControlloAcquisto 25

2.5.1. Carrello 25

2.5.2. CarrelloRemove 26

2.5.3. Acquisto 27

2.6. HomePage 27

2.6.1. Home 28

2.6.2. BackToHome 28

2.6.3. CambioPagina 29

2.6.4. MyServletException 31

2.7. JavaBeans 31

2.7.1. Account 32

2.7.2. Carrello 33

2.7.2.1. ProdottoQuantità 34

2.7.3. Carta 35

2.7.4. Categoria 35

2.7.5. Contact 36

2.7.6. Indirizzo 37

2.7.7. Prodotto 38

2.7.8. Transazione 39

2.8. Connection 40

2.8.1. ConPool 41

2.8.2. AccountDAO 41

2.8.3. CartaDAO 42

2.8.4. CategoriaDAO 43

2.8.5. ContactDAO 44

2.8.6. IndirizzoDAO 45

2.8.7. ProdottoDAO 46

2.8.8. TransazioneDAO 48

3.1. ControlloHome 49

3.2. ControlloAccount 50

3.3. ControlloCategorie 50

3.4. ControlloAcquisto 50

3.5. ControlloProdotti 51

3.6. ControlloOrdini 52

3.7. Connection 52

3.8. JavaBeans 52

1. INTRODUZIONE

## Guideline per la documentazione dell’Interfaccia

Questa sezione darà un insight sulle convenzioni per i nomi per rendere il programma più comprensibile rendendolo più semplice da leggere.

Come prima cosa si spiegano le convenzioni sui nomi per ogni tipo di identificatore:

**Tipo di identificatore: Packages**

**Regole per i nomi:**

Il prefisso di un nome di un package unico è sempre scritto con la prima lettera in maiuscolo e minuscolo per le restanti lettere

**Esempio:**

package AccountController

**Tipo di identificatore: Classi**

**Regole per i nomi:**

I nomi delle classi devono sempre iniziare con una lettera maiuscola con le iniziali delle successive parole in maiuscolo, il nome della classe deve essere semplice e descrittivo

**Esempio:**

class LoginServlet

**Tipo di identificatore: Interfacce**

**Regole per i nomi:**

I nomi delle interfacce devono seguire le stesse regole dei nomi delle classi

**Esempio:**

interface DataManager

**Tipo di identificatore: Metodi**

**Regole per i nomi:**

I nomi dei metodi dovrebbero essere verbi con la prima lettera in minuscolo e con le iniziali delle successive parole in maiuscolo

**Esempio:**

doPost ()

**Tipo di identificatore: Variabili**

**Regole per i nomi:**

I nomi delle variabili non possono iniziare con un underscore (\_) o con il segno del dollaro ($), i nomi devono essere brevi ma comprensibili

**Esempio:**

int accountID

**Tipo di identificatore: Costanti**

**Regole per i nomi:**

Seguono le stesse regole per le variabili

## Definizioni, acronimi e abbreviazioni

|  |  |
| --- | --- |
| ID | Identificativo |
|  |  |
|  |  |
|  |  |

## References

[Problem Statement Uroboros](https://unisalerno-my.sharepoint.com/:w:/g/personal/a_amorosini_studenti_unisa_it/ETbMpLhJQuRNqHQZsA-SRnkBODYbOzYvP1KNBq350o1oYA)

[Requirement Analysis Document Uroboros](https://unisalerno-my.sharepoint.com/:w:/g/personal/a_amorosini_studenti_unisa_it/EblNRhfh6qtMjielr3oy0VcBhMvqCg__FZ9E682tBvgK9w)

[SystemDesignDocument](https://unisalerno-my.sharepoint.com/:w:/g/personal/a_amorosini_studenti_unisa_it/ESMM6QiVxX9Eng6Kd9xqMycB87suPLJNXg0LmtbkBVsV2g)

1. PACKAGES

* **PagineAccount:** questo package contiene le classi necessarie per il caricamento e per l’interazione con le pagine dedicate all’account utente
* **PagineProdotti:** questo package contiene le classi necessarie al caricamento e l’interazione con le pagine dedicate ai prodotti
* **PagineCategorie:** questo package contiene le classi necessarie al caricamento e l’interazione con le pagine dedicate alle categorie
* **PagineOrdini:** questo package contiene le classi necessarie al caricamento e l’interazione con le pagine dedicate agli ordini
* **PagineAcquisto:** questo package contiene le classi necessarie al caricamento e l’interazione con le pagine dedicate alle procedure di acquisto
* **ControlloAccount:** questo package contiene le classi per il recupero, la gestione e l’interazione con i dati riguardanti gli account
* **ControlloProdotti:** questo package contiene le classi per il recupero, la gestione e l’interazione con i dati riguardanti i prodotti
* **ControlloCategorie:** questo package contiene le classi per il recupero, la gestione e l’interazione con i dati riguardanti le categorie
* **ControlloOrdini:** questo package contiene le classi per il recupero, la gestione e l’interazione con i dati riguardanti gli ordini
* **ControlloAcquisto:** questo package contiene le classi per il recupero, la gestione e l’interazione con i dati riguardanti gli acquisti
* **JavaBeans:** questo package contiene tutti i javabeans usati per la gestione dei dati provenienti da database
* **Connection:** questo package contiene le classi usate per la connessione con il database e per il recupero di dati riguardanti gli oggetti

## ControlloAccount

Questo package contiene le classi per le operazioni logiche che riguardano il login e logout di un utente, la visualizzazione dei propri dati personali, la registrazione di un nuovo utente, la gestione degli account da parte del gestore

|  |  |
| --- | --- |
| Class Summary | |
| Login | Questa classe gestisce la procedura di login dell’utente |
| Logout | Questa classe gestisce la procedura di logout dell’utente |
| ModificaDatiAccount | Questa classe gestisce la procedura di modifica dei propri dati |
| Registrazione | Questa classe gestisce la procedura di registrazione al sito |
| ShowAccountDetail | Questa classe gestisce la procedura per mostrare i dettagli di un account |
| GestioneAccount | Questa classe gestisce la procedura per recuperare tutti gli account da mostrare nella pagina di gestione degli account |
| VerificaUsername | Questa classe serve a verificare se lo username inserito in fase di registrazione è già presente nel database, e se è valido |
| DeleteAccount | Questa classe gestisce la procedura per l’eliminazione del profilo di un utente selezionato dal gestore |
| ShowAccountPage | Questa classe serve a caricare la pagina di dettagli dell’account o la pagina di login a seconda se un account abbia già effettuato l’accesso o meno |

### Class Login

Public class Login extends HttpServlet

Metodi:

* protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo POST
  + PRECONDIZIONI
    - Request!=null &&
    - Response != null
  + POSTCONDIZIONI
* protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo GET
  + PRECONDIZIONI
    - Request!=null &&
    - Response !=null
  + POSTCONDIZIONI
* public void login(HttpServletRequest request, HttpServletResponse response) throws IOException, ServletException
  + questo metodo implementa la procedura di login di un utente verificando eventuali corrispondenze nel database per la combinazione di username e password inserita dall’utente
  + PRECONDIZIONI
    - Request != null &&
    - Response != null &&
    - Parameter(“username”) != null &&
    - Parameter(“password”) != null
  + POSTCONDIZIONI
    - L’utente ha effettuato l’accesso al sito e si ritrova nella pagina in cui stava in precedenza

### Class Logout

Public class Logout extends HttpServlet

Metodi:

* protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo POST
  + PRECONDIZIONI
    - Request!=null &&
    - Response != null
  + POSTCONDIZIONI
* protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo GET
  + PRECONDIZIONI
    - Request!=null &&
    - Response !=null
  + POSTCONDIZIONI
* public void logout(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + questo metodo implementa la procedura di logout svuotando dalla sessione corrente l’account
  + PRECONDIZIONI
    - Request != null &&
    - Response != null
  + POSTCONDIZIONI
    - L’utente ha effettuato il logout e si ritrova nella pagina di login

### Class ModificaDatiAccount

Public class ModificaDatiAccount extends HttpServlet

Metodi:

* protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo POST
  + PRECONDIZIONI
    - Request!=null &&
    - Response != null
  + POSTCONDIZIONI
* protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo GET
  + PRECONDIZIONI
    - Request!=null &&
    - Response !=null
  + POSTCONDIZIONI
* public void modificaDati(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Questo metodo implementa la procedura per la modifica dei propri dati usando i dati inseriti dall’utente nei form appositi
  + PRECONDIZIONI
    - Request != null &&
    - Response != null
  + POSTCONDIZIONI
    - Vengono aggiornati i dati dell’account e l’utente si ritrova nella pagina di visualizzazione dettagli dell’account

### Class Registrazione

Public class Registrazione extends HttpServlet

Metodi:

* protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo POST
  + PRECONDIZIONI
    - Request!=null &&
    - Response != null
  + POSTCONDIZIONI
* protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo GET
  + PRECONDIZIONI
    - Request!=null &&
    - Response !=null
  + POSTCONDIZIONI
* public void Registrazione(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + questo metodo implementa la procedura di registrazione di un nuovo utente utilizzando i dati inseriti dall’utente negli appositi form
  + PRECONDIZIONI
    - Request != null &&
    - Response != null
  + POSTCONDIZIONI
    - Viene creato un nuovo account e l’utente si ritrova in una pagina che comunica il risultato della registrazione

### Class ShowAccountDetail

Public class ShowAccountDetail extends HttpServlet

Metodi:

* protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo POST
  + PRECONDIZIONI
    - Request!=null &&
    - Response != null
  + POSTCONDIZIONI
* protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo GET
  + PRECONDIZIONI
    - Request!=null &&
    - Response !=null
  + POSTCONDIZIONI
* public void mostraDettagliAccountAdmin(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Questo metodo implementa il recupero di dettagli limitati di un account da mostrare all’admin nella pagina di gestione degli account
  + PRECONDIZIONI
    - Request != null &&
    - Response != null
  + POSTCONDIZIONI
    - Vengono recuperati dei dati limitati dell’account selezionato dal gestore e questo si ritrova in una pagina che presenti questi dettagli

### Class GestioneAccount

Public class GestioneAccount extends HttpServlet

Metodi:

* protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo POST
  + PRECONDIZIONI
    - Request!=null &&
    - Response != null
  + POSTCONDIZIONI
* protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo GET
  + PRECONDIZIONI
    - Request!=null &&
    - Response !=null
  + POSTCONDIZIONI
* public void mostraGestioneAccount(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Questo metodo recupera tutti gli account per poterli mostrare nella pagina di gestione degli account al gestore
  + PRECONDIZIONI
    - Request != null &&
    - Response != null
  + POSTCONDIZIONI
    - Vengono recuperati tutti gli account per poter essere visionati dal gestore nella pagina di gestione degli account

### Class VerificaUsername

Public class VerificaUsername extends HttpServlet

Metodi:

* protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo POST
  + PRECONDIZIONI
    - Request!=null &&
    - Response != null
  + POSTCONDIZIONI
* protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo GET
  + PRECONDIZIONI
    - Request!=null &&
    - Response !=null
  + POSTCONDIZIONI
* public void verificaUsername(HttpServletRequest request, HttpServletResponse response) throws IOException
  + Questo metodo verifica che l’username inserito dall’utente nella procedura di registrazione rispetti le linee guida per gli username e che non sia già presente nel database un username uguale
  + PRECONDIZIONI
    - Request != null &&
    - Response != null
  + POSTCONDIZIONI
    - Viene mandato un segnale positivo in caso l’username rispetti le condizioni, o uno negativo in caso non le rispetti

### Class DeleteAccount

Public class DeleteAccount extends HttpServlet

Metodi:

* protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo POST
  + PRECONDIZIONI
    - Request!=null &&
    - Response != null
  + POSTCONDIZIONI
* protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo GET
  + PRECONDIZIONI
    - Request!=null &&
    - Response !=null
  + POSTCONDIZIONI
* public void cancellaAccount(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Questo metodo elimina dal database tutti i riferimenti all’account selezionato dal gestore nella pagina di gestione degli account per poi ricaricare la pagina
  + PRECONDIZIONI
    - Request != null &&
    - Response != null
  + POSTCONDIZIONI
    - Tutti i riferimenti all’account selezionato vengono eliminati e la pagina di gestione degli account viene ricaricata

### Class ShowAccount

Public class ShowAccount extends HttpServlet

Metodi:

* protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo POST
  + PRECONDIZIONI
    - Request!=null &&
    - Response != null
  + POSTCONDIZIONI
* protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo GET
  + PRECONDIZIONI
    - Request!=null &&
    - Response !=null
  + POSTCONDIZIONI
* public void mostraPagina(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Questo metodo decide se caricare la pagina di visualizzazione dei dettagli dell’account utente o la pagina di login in base alla presenza o meno di un account loggato
  + PRECONDIZIONI
    - Request != null &&
    - Response != null
  + POSTCONDIZIONI
    - L’utente si ritrova o nella pagina di login o nella pagina di visualizzazione dei dettagli account

## controlloProdotti

Questo package contiene le classi per le operazioni di visualizzazione , ricerca e gestione dei prodotti

|  |  |
| --- | --- |
| Class Summary | |
| Prodotto | Questa classe recupera i dati di un prodotto da mostrare |
| Catalogo | Questa classe recupera dati su tutti i prodotti da mostrare nel catalogo |
| Search | Questa classe implementa la ricerca di un prodotto per nome |
| GestioneProdotti | Questa classe implementa recupera i dati su tutti i prodotti per poterli mostrare nella pagina di gestione prodotti |
| AggiungiProdotto | Questa classe implementa la procedura per aggiungere un prodotto al database |
| CancellaProdotto | Questa classe implementa la procedura per rimuovere un prodotto dal database |
| ModificaDatiProdotto | Questa classe implementa la procedura per modificare i dati di un prodotto già esistente |
| ModificaProdottoPage | Questa classe recupera i dati richiesti per poter caricare la pagina di modifica del prodotto selezionato dal gestore |

### Prodotto

Public class Prodotto extends HttpServlet

Metodi:

* protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo POST
  + PRECONDIZIONI
    - Request!=null &&
    - Response != null
  + POSTCONDIZIONI
* protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo GET
  + PRECONDIZIONI
    - Request!=null &&
    - Response !=null
  + POSTCONDIZIONI
* public void mostraPaginaProdotto(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Questo metodo implementa il recupero di dati di un prodotto per poterli mostrare nella pagina del prodotto
  + PRECONDIZIONI
    - Request != null &&
    - Response != null
  + POSTCONDIZIONI
    - Vengono recuperati i dettagli del prodotto selezionato e l’utente si ritrova sulla pagina di visualizzazione del prodotto

### Catalogo

Public class Catalogo extends HttpServlet

Metodi:

* protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo POST
  + PRECONDIZIONI
    - Request!=null &&
    - Response != null
  + POSTCONDIZIONI
* protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo GET
  + PRECONDIZIONI
    - Request!=null &&
    - Response !=null
  + POSTCONDIZIONI
* public void visualizzaCatalogo(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Questo metodo implementa il recupero di tutti i prodotti da poi mostrare nella pagina di visualizzazione catalogo
  + PRECONDIZIONI
    - Request != null &&
    - Response != null
  + POSTCONDIZIONI
    - Vengono recuperati vari prodotti da mostrare e l’utente si ritrova nella pagina di visualizzazione del catalogo

### Search

Public class Search extends HttpServlet

Metodi:

* protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo POST
  + PRECONDIZIONI
    - Request!=null &&
    - Response != null
  + POSTCONDIZIONI
* protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo GET
  + PRECONDIZIONI
    - Request!=null &&
    - Response !=null
  + POSTCONDIZIONI
* public void cercaProdotto(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Questo metodo implementa la procedura di ricerca di un prodotto per somiglianza al termine di ricerca inserito dall’utente nell’apposito form
  + PRECONDIZIONI
    - Request != null &&
    - Response != null
  + POSTCONDIZIONI
    - Si recuperano i prodotti con un nome simile al termine di ricerca e l’utente si ritrova nella pagian di visualizzazione dei risultati di ricerca

### GestioneProdotti

Public class GestioneProdotti extends HttpServlet

Metodi:

* protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo POST
  + PRECONDIZIONI
    - Request!=null &&
    - Response != null
  + POSTCONDIZIONI
* protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo GET
  + PRECONDIZIONI
    - Request!=null &&
    - Response !=null
  + POSTCONDIZIONI
* public void mostraGestioneProdotto(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Questo metodo implementa il recupero dei dati di tutti i prodotti per poterli mostrare al gestore nella pagina di gestione dei prodotti
  + PRECONDIZIONI
    - Request != null &&
    - Response != null
  + POSTCONDIZIONI
    - Vengono recuperati tutti i prodotti da mostrare al gestore nella pagina di gestioen dei prodotti sulla quale si ritrova

### AggiungiProdotto

Public class AggiungiProdotto extends HttpServlet

Metodi:

* protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo POST
  + PRECONDIZIONI
    - Request!=null &&
    - Response != null
  + POSTCONDIZIONI
* protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo GET
  + PRECONDIZIONI
    - Request!=null &&
    - Response !=null
  + POSTCONDIZIONI
* public void aggiungiProdotto(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Questo metodo implementa la procedura di aggiunta di un nuovo prodotto al database
  + PRECONDIZIONI
    - Request != null &&
    - Response != null
  + POSTCONDIZIONI
    - Viene creato un nuovo prodotto e il gestore si ritrova nella pagina di gestione dei prodotti

### CancellaProdotto

Public class CancellaProdotto extends HttpServlet

Metodi:

* protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo POST
  + PRECONDIZIONI
    - Request!=null &&
    - Response != null
  + POSTCONDIZIONI
* protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo GET
  + PRECONDIZIONI
    - Request!=null &&
    - Response !=null
  + POSTCONDIZIONI
* public void cancellaProdotto(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Questo metodo implementa la procedura di rimozione di un prodotto dal database
  + PRECONDIZIONI
    - Request != null &&
    - Response != null
  + POSTCONDIZIONI
    - Viene eliminato il prodotto selezionato dal database e il gestore si ritrova nella pagina di gestione dei prodotti

### ModificaDatiProdotto

Public class ModificaDatiProdotto extends HttpServlet

Metodi:

* protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo POST
  + PRECONDIZIONI
    - Request!=null &&
    - Response != null
  + POSTCONDIZIONI
* protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo GET
  + PRECONDIZIONI
    - Request!=null &&
    - Response !=null
  + POSTCONDIZIONI
* public void (HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Questo metodo implementa la procedura di modifica dei dati di un prodotto usando i dati inseriti dal gestore negli appositi form
  + PRECONDIZIONI
    - Request != null &&
    - Response != null
  + POSTCONDIZIONI
    - I dati del prodotto selezionato vengono aggiornati e il gestore si ritrova nella pagina di gestione dei prodotti

### ModificaProdottoPage

Public class ModificaProdottoPage extends HttpServlet

Metodi:

* protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo POST
  + PRECONDIZIONI
    - Request!=null &&
    - Response != null
  + POSTCONDIZIONI
* protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo GET
  + PRECONDIZIONI
    - Request!=null &&
    - Response !=null
  + POSTCONDIZIONI
* public void mostraPaginaModificaProdotto(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Questo metodo carica i dati del prodotto selezionato per poterli mostrare nella pagina di modifica dati del prodotto al gestore
  + PRECONDIZIONI
    - Request != null &&
    - Response != null
  + POSTCONDIZIONI
    - Vengono caricati i dati del prodotto selezionato e il gestore si ritrova nella pagina di modifica dei dati del prodotto

## controlloCategorie

Questo package contiene le classi per la visualizzazione e gestione delle categorie

|  |  |
| --- | --- |
| Class Summary | |
| Categoria | Questa classe recupera i dati di una categoria da mostrare |
| GestioneCategorie | Questa classe recupera i dati di tutte le categorie da mostrare al gestore |
| AggiungiCategoria | Questa classe implementa la procedura di aggiunta di una categoria al database |
| DeleteCategoria | Questa classe implementa la procedura di rimozione di una categoria dal database |
| ShowCategoriaDetails | Questa classe recupera i dati della categoria selezionata dal gestore |
| AggiornaCategoria | Questa classe implementa la procedura per modificare i dati di una categoria già esistente |

### Categoria

Public class Categoria extends HttpServlet

Metodi:

* protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo POST
  + PRECONDIZIONI
    - Request!=null &&
    - Response != null
  + POSTCONDIZIONI
* protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo GET
  + PRECONDIZIONI
    - Request!=null &&
    - Response !=null
  + POSTCONDIZIONI
* public void mostraCategoria(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Questo metodo recupera i dati di una categoria per poterli mostrare nella pagina della categoria
  + PRECONDIZIONI
    - Request != null &&
    - Response != null
  + POSTCONDIZIONI
    - Viene recuperata la categoria desiderata e l’utente si ritrova nella pagina di visualizzazione della categoria

### GestioneCategorie

Public class GestioneCategorie extends HttpServlet

Metodi:

* protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo POST
  + PRECONDIZIONI
    - Request!=null &&
    - Response != null
  + POSTCONDIZIONI
* protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo GET
  + PRECONDIZIONI
    - Request!=null &&
    - Response !=null
  + POSTCONDIZIONI
* public void mostraGestioneCategorie(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Questo metodo recupera i dati di tutte le categorie così da mostrarle al gestore nella pagina di gestione delle categorie
  + PRECONDIZIONI
    - Request != null &&
    - Response != null
  + POSTCONDIZIONI
    - Vengono recuperate tutte le categorie per poter essere mostrate nella pagina di gestione delle categorie dove viene mandato il gestore

### AggiungiCategoria

Public class AggiungiCategoria extends HttpServlet

Metodi:

* protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo POST
  + PRECONDIZIONI
    - Request!=null &&
    - Response != null
  + POSTCONDIZIONI
* protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo GET
  + PRECONDIZIONI
    - Request!=null &&
    - Response !=null
  + POSTCONDIZIONI
* public void aggiungiCategoria(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Questo metodo implementa la procedura per aggiungere una categoria al database usando i dati inseriti dal gestore
  + PRECONDIZIONI
    - Request != null &&
    - Response != null
  + POSTCONDIZIONI
    - Viene creata una nuova categoria e il gestore si ritrova nella pagina di gestione delle categorie

### DeleteCategoria

Public class DeleteCategoria extends HttpServlet

Metodi:

* protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo POST
  + PRECONDIZIONI
    - Request!=null &&
    - Response != null
  + POSTCONDIZIONI
* protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo GET
  + PRECONDIZIONI
    - Request!=null &&
    - Response !=null
  + POSTCONDIZIONI
* public void cancellaCategoria(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Questo metodo implementa la procedura per eliminare dal database la categoria selezionata dal gestore
  + PRECONDIZIONI
    - Request != null &&
    - Response != null
  + POSTCONDIZIONI
    - La categoria selezionata viene eliminata e il gestore si ritrova nella pagina di gestione delle categorie

### ShowCategoriaDetails

Public class ShowCategoriaDetails extends HttpServlet

Metodi:

* protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo POST
  + PRECONDIZIONI
    - Request!=null &&
    - Response != null
  + POSTCONDIZIONI
* protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo GET
  + PRECONDIZIONI
    - Request!=null &&
    - Response !=null
  + POSTCONDIZIONI
* public void mostraDettagliCategoria(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Questo metodo recupera i dati di una categoria per poterli mostrare al gestore in una pagina di dettagli della categoria
  + PRECONDIZIONI
    - request != null &&
    - response != null
  + POSTCONDIZIONI
    - Vengono recuperati i dettagli della categoria selezionata e il gestore di ritrova nella pagina di visualizzazione dei dettagli della categoria

### AggiornaCategoria

Public class AggiornaCategoria extends HttpServlet

Metodi:

* protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo POST
  + PRECONDIZIONI
    - Request!=null &&
    - Response != null
  + POSTCONDIZIONI
* protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo GET
  + PRECONDIZIONI
    - Request!=null &&
    - Response !=null
  + POSTCONDIZIONI
* public void aggiornaCategoria(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Questo metodo implementa la procedura di modifica dei dati di una categoria usando i dati inseriti dal gestore negli appositi form
  + PRECONDIZIONI
    - request != null &&
    - response != null
  + POSTCONDIZIONI
    - La categoria viene aggiornata con nuovi dati e il gestore si ritrova sulla pagina di gestione delel categorie

## controlloOrdini

Questo package contiene le classi per la visualizzazione dello storico degli ordini di un utente

|  |  |
| --- | --- |
| Class Summary | |
| Storico | Questa classe recupera le varie transazioni fatte da un utente per presentargliele nel proprio storico |
| ProdottiOrdine | Questa classe recupera i prodotti associati alla transazione selezionata |
| ShowStoricoAdmin | Questa classe recupera lo storico ordini dell’utente selezionato dal gestore |

### Storico

Public class Storico extends HttpServlet

Metodi:

* protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo POST
  + PRECONDIZIONI
    - Request!=null &&
    - Response != null
  + POSTCONDIZIONI
* protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo GET
  + PRECONDIZIONI
    - Request!=null &&
    - Response !=null
  + POSTCONDIZIONI
* public void mostraStorico(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Questo metodo recupera i dati di tutte le transazioni effettuate dall’utente per mostragliele nel proprio storico degli ordini
  + PRECONDIZIONI
    - request != null &&
    - response != null
  + POSTCONDIZIONI
    - Vengono recuperate tutte le transazioni dell’utente attuale e quest’ultimo si ritrova nella pagina di visualizzazione dello storico degli ordini

### ProdottiOrdine

Public class ProdottiOrdine extends HttpServlet

Metodi:

* protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo POST
  + PRECONDIZIONI
    - Request!=null &&
    - Response != null
  + POSTCONDIZIONI
* protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo GET
  + PRECONDIZIONI
    - Request!=null &&
    - Response !=null
  + POSTCONDIZIONI
* public void recuperaProdottiTransazione(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Questo metodo recupera i prodotti associati alla transazione selezionato e li mostra nella pagina di dettagli della transazione
  + PRECONDIZIONI
    - request != null &&
    - response != null
  + POSTCONDIZIONI
    - Vengono recuperati i prodotti associati alla transazione selezionata e l’utente si ritrova nella pagina di visualizzazione dei dettagli della transazione

### ShowStoricoAdmin

Public class ShowStoricoAdmin extends HttpServlet

Metodi:

* protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo POST
  + PRECONDIZIONI
    - Request!=null &&
    - Response != null
  + POSTCONDIZIONI
* protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo GET
  + PRECONDIZIONI
    - Request!=null &&
    - Response !=null
  + POSTCONDIZIONI
* public void mostraStoricoAdmin(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Recupera tutte le transazioni effettuate da un utente selezionato dal gestore nella pagina di gestione degli account
  + PRECONDIZIONI
    - request != null &&
    - response != null
  + POSTCONDIZIONI
    - Vengono recuperate le transazioni effettuate dall’account selezionato dal gestore e quest’ultimo si ritrova nella pagina di visualizzazione transazioni

## ControlloAcquisto

Questo package contiene classi per visualizzare il carrello, aggiungerci e rimuoverci prodotti e finalizzare l’acquisto

|  |  |
| --- | --- |
| Class Summary | |
| Carrello | Questa classe implementa l’aggiunta di un prodotto al carrello e la sua visualizzazione |
| CarrelloRemove | Questa classe implementa la procedura per rimuovere un prodotto dal carrello |
| Acquisto | Questa classe implementa la procedura di acquisto |

### Carrello

Public class Carrello extends HttpServlet

Metodi:

* protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo POST
  + PRECONDIZIONI
    - Request!=null &&
    - Response != null
  + POSTCONDIZIONI
* protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo GET
  + PRECONDIZIONI
    - Request!=null &&
    - Response !=null
  + POSTCONDIZIONI
* public void aggiungiAlCarrello(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Questo metodo implementa l’aggiunta di un prodotto al carrello e la visualizzazione del carrello stesso
  + PRECONDIZIONI
    - request != null &&
    - response != null
  + POSTCONDIZIONI
    - È presente un oggetto carrello nella sessione, il prodotto selezionato è stato aggiunto al carrello e l’utente si ritrova nella pagina di visualizzazione del carrello

### CarrelloRemove

Public class CarrelloRemove extends HttpServlet

Metodi:

* protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo POST
  + PRECONDIZIONI
    - Request!=null &&
    - Response != null
  + POSTCONDIZIONI
* protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo GET
  + PRECONDIZIONI
    - Request!=null &&
    - Response !=null
  + POSTCONDIZIONI
* public void rimuoviDalCarrello(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Questo metodo implementa la procedura di rimozione di un prodotto dal carrello
  + PRECONDIZIONI
    - request != null &&
    - response != null
  + POSTCONDIZIONI
    - Il prodotto selezionato viene rimosso dal carrello e viene ricaricata la pagina

### Acquisto

Public class Acquisto extends HttpServlet

Metodi:

* protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo POST
  + PRECONDIZIONI
    - Request!=null &&
    - Response != null
  + POSTCONDIZIONI
* protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo GET
  + PRECONDIZIONI
    - Request!=null &&
    - Response !=null
  + POSTCONDIZIONI
* public void acquisto(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Questo metodo implementa la procedura di finalizzazione di un acquisto e la creazione di una nuova transazione da inserire nello storico ordini
  + PRECONDIZIONI
    - request != null &&
    - response != null
  + POSTCONDIZIONI
    - Viene creato un oggetto transazione e l’utente si ritrova nella pgian di conferma dell’acquisto

## HomePage

Questo package contiene classi per la visualizzazione iniziale della home e per la visualizzazione di pagine che non necessitano di operazioni complesse prima di caricare

|  |  |
| --- | --- |
| Class Summary | |
| Home | Questa classe implementa il primo caricamento della pagina Home all’apertura del sito |
| BackToHome | Questa classe implementa la procedura per ritornare direttamente alla home |
| CambioPagina | Questa classe implementa vari metodi per il cambio di pagina |
| ServletException | Questa classe implementa un eccezione personalizzata per il sito |

### Home

Public class Home extends HttpServlet

Metodi:

* protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo POST
  + PRECONDIZIONI
    - Request!=null &&
    - Response != null
  + POSTCONDIZIONI
* protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo GET
  + PRECONDIZIONI
    - Request!=null &&
    - Response !=null
  + POSTCONDIZIONI
* public void mostraPaginaIniziale(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Questo metodo recupera i prodotti da mostrare nella home e la carica
  + PRECONDIZIONI
    - request != null &&
    - response != null
  + POSTCONDIZIONI
    - Vengono recuperati una serie di prodotti e viene caricata la pagina iniziale del sito
* Public void init() throws ServletException
  + Questo metodo viene chiamato all’apertura del sito ed inizializza le categorie nel context

### BackToHome

public class BackToHome extends HttpServlet

Metodi:

* protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo POST
  + PRECONDIZIONI
    - Request!=null &&
    - Response != null
  + POSTCONDIZIONI
* protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo GET
  + PRECONDIZIONI
    - Request!=null &&
    - Response !=null
  + POSTCONDIZIONI
* public void tornaAllaHome(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Questo metodo recupera nuovamente i prodotti e ricarica la Home Page
  + PRECONDIZIONI
    - request != null &&
    - response != null
  + POSTCONDIZIONI
    - Vengono recuperati tutti i prodotti e viene caricata la pagina iniziale del sito

### CambioPagina

Public class CambioPagina extends HttpServlet

Metodi:

* protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo POST
  + PRECONDIZIONI
    - Request!=null &&
    - Response != null
  + POSTCONDIZIONI
* protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Metodo di servizio di HttpServlet per la comunicazione con il server con metodo GET
  + PRECONDIZIONI
    - Request!=null &&
    - Response !=null
  + POSTCONDIZIONI
* public void mostraPaginaAboutUs(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Questo metodo reindirizza alla pagina about us
  + PRECONDIZIONI
    - request != null &&
    - response != null
  + POSTCONDIZIONI
    - L’utente si ritrova sulla pagina about us
* public void mostraPaginaAggiungiCategoria(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Questo metodo reindirizza alla pagina per l’aggiunta di una categoria
  + PRECONDIZIONI
    - request != null &&
    - response != null
  + POSTCONDIZIONI
    - Il gestore si ritrova nella pagina di aggiunta di una nuova categoria
* public void mostraPaginaAggiungiProdotto(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Questo metodo reindirizza alla pagina per l’aggiunta di un prodotto
  + PRECONDIZIONI
    - request != null &&
    - response != null
  + POSTCONDIZIONI
    - Il gestore si ritrova nella pagina di aggiunta di un nuovo prodotto
* public void mostraPaginaContact(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Questo metodo reindirizza alla pagina Contact
  + PRECONDIZIONI
    - request != null &&
    - response != null
  + POSTCONDIZIONI
    - L’utente si ritrova nella pagina dei contact del sito
* public void mostraDashBoardGestore(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Questo metodo reindirizza alla pagina della DashBoard di gestione del sito
  + PRECONDIZIONI
    - request != null &&
    - response != null
  + POSTCONDIZIONI
    - Il gestore si ritrova nella DashBoard di gestione del sito
* public void mostraPaginaModificaAccount(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Questo metodo reindirizza alla pagina per la modifica del proprio account
  + PRECONDIZIONI
    - request != null &&
    - response != null
  + POSTCONDIZIONI
    - L’utente si ritrova nella pagina per la modifica dei dati del proprio account
* public void mostraPaginaRegistrazione(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  + Questo metodo reindirizza alla pagina di registrazione di un nuovo utente
  + PRECONDIZIONI
    - request != null &&
    - response != null
  + POSTCONDIZIONI
    - L’Utente si ritrova nella pagina di registrazione di un nuovo account

### MyServletException

Public class MyServletException extends ServletException

Costruttore:

* Public MyServletException

Metodi:

* Public MyServletException(String message)
  + Crea una ServletException utilizzando il messaggio passato al metodo
  + PRECONDIZIONI
    - Message!=null
  + POSTCONDIZIONI
    - Viene lanciata una ServletException con il messaggio passato al metodo
* Public MyServletException(String message, Throwable rootCause)
  + Crea una ServletException utilizzando il messaggio e la causa mandata al metodo
  + PRECONDIZIONI
    - Message!=null &&
    - rootCause!=null
  + POSTCONDIZIONI
    - Viene lanciata una ServletException con il messaggio e la causa passati al metodo
* Public MyServletException(Throwable rootCause)
  + Crea una ServleteException utilizzando la causa passata al metodo
  + PRECONDIZIONI
    - rootCause!=null
  + POSTCONDIZIONI
    - Viene lanciata una ServletException con la causa passata al metodo

## JavaBeans

Questo package contiene le classi per la gestione di oggetti di vario tipo da associare ai dati provenienti dall’database

|  |  |
| --- | --- |
| Class Summary | |
| Account | Questa classe implementa metodi per la gestione dei dati di un oggetto di tipo Account |
| Carrello | Questa classe implementa metodi per la gestione dei dati di un oggetto di tipo Carrello |
| Carta | Questa classe implementa metodi per la gestione dei dati di un oggetto di tipo Carta |
| Categoria | Questa classe implementa metodi per la gestione dei dati di un oggetto di tipo Categoria |
| Contact | Questa classe implementa metodi per la gestione dei dati di un oggetto di tipo Contact |
| Indirizzo | Questa classe implementa metodi per la gestione dei dati di un oggetto di tipo Indirizzo |
| Prodotto | Questa classe implementa metodi per la gestione dei dati di un oggetto di tipo Prodotto |
| Transazione | Questa classe implementa metodi per la gestione dei dati di un oggetto di tipo Transazione |

### Account

Public class Account

Attributi:

* private String username;
* private String password;
* private int CF;
* private String nomeAzienda;
* private int numCarta;
* private int IDTransazione;
* private String indirizzo;
* private boolean admin;

Metodi:

* Public void setInfoString(String info)
  + Questo metodo imposta uno degli attributi di tipo String con quella passata al metodo
  + PRECONDIZIONI
    - Info!=null
  + POSTCONDIZIONI
    - Viene assegnata una stringa all’attributo richiesto
* Public void setInfoInt(int info)
  + Questo metodo imposta uno degli attributi di tipo int con quello passato al metodo
  + PRECONDIZIONI
    - Info!=null
  + POSTCONDIZIONI
    - Viene assegnato un intero all’attributo richiesto
* Public void setAdmin(boolean Admin)
  + Questo metodo imposta l’attributo Admin sul valore passato al metodo
  + PRECONDIZIONI
    - Admin!=null
  + POSTCONDIZIONI
    - L’attributo admin viene impostato su vero o falso
* Public String getInfoString()
  + Questo metodo recupera i dati contenuti in uno degli attributi di tipo String
  + PRECONDIZIONI
  + POSTCONDIZIONI
    - Viene mandato il contenuto dell’attributo richiesto al chiamante sotto forma di stringa
* Public int getInfoInt()
  + Questo metodo recupera i dati contenuti in uno degli attributi di tipo int
  + PRECONDIZIONI
  + POSTCONDIZIONI
    - Viene mandato il contenuto dell’attributo richiesto al chiamante sotto forma di intero
* Public boolean isAdmin()
  + Questo metodo restituisce se l’utente attuale è o non è admin
  + PRECONDIZIONI
  + POSTCONDIZIONI
    - Viene restituito il valore booleano su cui è settato admin
* Public boolean equals(Object o)
  + Questo metodo controlla se l’oggetto account attuale è corrisponde all’oggetto passato al metodo
  + PRECONDIZIONI
    - O!=null
  + POSTCONDIZIONI
    - Viene restituito true se gli oggetti corrispondono , false altrimenti
* Public int hashCode()
  + Trasforma tutti i valori degli attributi in un codice hash

### Carrello

public class Carrello

Attributi:

* Private LinkedHashMap<Integer, ProdottoQuantità> prodotti

Metodi:

* Public Collection<ProdottoQuantita> getProdotti()
  + Recupera tutti gli oggetti di tipo ProdottoQuantita presenti per il carrello attuale
  + PRECONDIZIONI
  + POSTCONDIZIONI
    - Ritorna una Collection di oggetti di tipo ProdottoQuantita contententi i prodotti presenti nel carrello e le loro quantita
* Public ProdottoQuantita get(int prodID)
  + Recupera un oggetto di tipo ProdottoQuantita contenente il prodotto identificato dall’ID passato al metodo
  + PRECONDIZIONI
    - prodID!=null
  + POSTCONDIZIONI
    - Ritorna un oggetto di tipo ProdottoQuantita contenente il prodotto specificato dall’ID passato al metodo
* Public void put(Prodotto prodotto, int quantita)
  + Inserisce un nuovo oggetto di tipo ProdottoQuantita all’interno del carrello
  + PRECONDIZIONI
    - Prodotto!=null &&
    - Quantita!=null
  + POSTCONDIZIONI
    - Viene creato un nuovo oggetto di tipo ProdottoQuantita e lo inserisce all’interno della collection del carrello
* Public ProdottoQuantita remove(int prodID)
  + Rimuove l’oggetto ProdottoQuantita contenente il prodotto identificato dall’ID passato al metodo
  + PRECONDIZIONI
    - prodID!=null
  + POSTCONDIZIONI
    - L’oggetto ProdottoQuantita contente il prodotto specificato dall’ID passato al metodo viene rimosso dal carrello
* Public String getPrezzoTot()
  + Recupera la somma dei prezzi dei prodotti contenuti all’interno del carrello
  + PRECONDIZIONI
  + POSTCONDIZIONI
    - Ritorna una stringa contenente il prezzo totale di tutti i prodotti nel carrello

### ProdottoQuantità

Public static class ProdottoQuantita

Sottoclasse di carrello per la gestione della coppia prodotto acquistato e quantità acquistata

Attributi:

* + Private Prodotto prodotto
  + Private int quantita

Metodo Costruttore:

* + Public ProdottoQuantita(Prodotto prodotto, int quantità)

Metodi:

* + Public int getQuantita()
    - Recupera l’attributo quantità dell’oggetto ProdottoQuantita
    - PRECONDIZIONI
    - POSTCONDIZIONI
      * Ritorna il valore dell’attributo quantita dell’oggetto ProdottoQuantita
  + Public void setQuantita(int quantita)
    - Imposta l’attributo quantità sull’intero passato al metodo
    - PRECONDIZIONI
      * Quantita!=null
    - POSTCONDIZIONI
      * l’attributo quantita viene settato sull’intero passato al metodo
  + Public Prodotto getProdotto()
    - Recupera l’attributo Prodotto dell’oggetto ProdottoQuantita
    - PRECONDIZIONI
    - POSTCONDIZIONI
      * Ritorna il valore dell’attributo prodotto dell’oggetto ProdottoQuantita
  + Public double getPrezzotot()
    - Recupera il prezzo totale della quantità del prodotto nell’oggetto ProdottoQuantita
    - PRECONDIZIONI
    - POSTCONDIZIONI
      * Ritorna il prezzo totale delle quantità del prodotto nell’oggetto ProdottoQuantita

### Carta

public class Carta

Attributi:

* private int numCarta;
* private String dataDiScadenza;
* private int CSV;
* private String nomeTitolare;
* private String cognomeTitolare;

Metodi:

* Public void setInfoString(String info)
  + Questo metodo imposta uno degli attributi di tipo String con quella passata al metodo
  + PRECONDIZIONI
    - Info!=null
  + POSTCONDIZIONI
    - Viene assegnata una stringa all’attributo richiesto
* Public void setInfoInt(int info)
  + Questo metodo imposta uno degli attributi di tipo int con quello passato al metodo
  + PRECONDIZIONI
    - Info!=null
  + POSTCONDIZIONI
    - Viene assegnato un intero all’attributo richiesto
* Public String getInfoString()
  + Questo metodo recupera i dati contenuti in uno degli attributi di tipo String
  + PRECONDIZIONI
  + POSTCONDIZIONI
    - Viene mandato il contenuto dell’attributo richiesto al chiamante sotto forma di stringa
* Public int getInfoInt()
  + Questo metodo recupera i dati contenuti in uno degli attributi di tipo int
  + PRECONDIZIONI
  + POSTCONDIZIONI
    - Viene mandato il contenuto dell’attributo richiesto al chiamante sotto forma di intero

### Categoria

Public class Categoria

Attributi:

* private int IDCategoria;
* private String nomeCategoria;
* private String descrizioneCategoria;
* private List<Prodotto> prodotti;

Metodi:

* Public void setInfoString(String info)
  + Questo metodo imposta uno degli attributi di tipo String con quella passata al metodo
  + PRECONDIZIONI
    - Info!=null
  + POSTCONDIZIONI
    - Viene assegnata una stringa all’attributo richiesto
* Public void setInfoInt(int info)
  + Questo metodo imposta uno degli attributi di tipo int con quello passato al metodo
  + PRECONDIZIONI
    - Info!=null
  + POSTCONDIZIONI
    - Viene assegnato un intero all’attributo richiesto
* Public String getInfoString()
  + Questo metodo recupera i dati contenuti in uno degli attributi di tipo String
  + PRECONDIZIONI
  + POSTCONDIZIONI
    - Viene mandato il contenuto dell’attributo richiesto al chiamante sotto forma di stringa
* Public int getInfoInt()
  + Questo metodo recupera i dati contenuti in uno degli attributi di tipo int
  + PRECONDIZIONI
  + POSTCONDIZIONI
    - Viene mandato il contenuto dell’attributo richiesto al chiamante sotto forma di intero
* Public List<Prodotto> getProdotti()
  + Questo metodo recupera tutti i prodotti associati alla categoria corrente
  + PRECONDIZIONI
  + POSTCONDIZIONI
    - Viene mandata la lista di prodotti al chiamante
* Public void setProdotti(List<Prodotto> prodotti)
  + Questo metodo imposta una lista di prodotti come associati alla categoria corrente
  + PRECONDIZIONI
    - Prodotti!=null
  + POSTCONDIZIONI
    - Viene assegnata la lista all’attributo prodotti di categoria

### Contact

Public class Contact

Attributi:

* private String nome;
* private String cognome;
* private int CF;
* private String telefono;
* private String dataDiNascita;
* private String email;

Metodi:

* Public void setInfoString(String info)
  + Questo metodo imposta uno degli attributi di tipo String con quella passata al metodo
  + PRECONDIZIONI
    - Info!=null
  + POSTCONDIZIONI
    - Viene assegnata una stringa all’attributo richiesto
* Public void setInfoInt(int info)
  + Questo metodo imposta uno degli attributi di tipo int con quello passato al metodo
  + PRECONDIZIONI
    - Info!=null
  + POSTCONDIZIONI
    - Viene assegnato un intero all’attributo richiesto
* Public String getInfoString()
  + Questo metodo recupera i dati contenuti in uno degli attributi di tipo String
  + PRECONDIZIONI
  + POSTCONDIZIONI
    - Viene mandato il contenuto dell’attributo richiesto al chiamante sotto forma di stringa
* Public int getInfoInt()
  + Questo metodo recupera i dati contenuti in uno degli attributi di tipo int
  + PRECONDIZIONI
  + POSTCONDIZIONI
    - Viene mandato il contenuto dell’attributo richiesto al chiamante sotto forma di intero

### Indirizzo

Public class Indirizzo

Attributi:

* private String via;
* private String via2;
* private int CAP;
* private String nazione;
* private String regione;
* private String citta;

Metodi:

* Public void setInfoString(String info)
  + Questo metodo imposta uno degli attributi di tipo String con quella passata al metodo
  + PRECONDIZIONI
    - Info!=null
  + POSTCONDIZIONI
    - Viene assegnata una stringa all’attributo richiesto
* Public String getInfoString()
  + Questo metodo recupera i dati contenuti in uno degli attributi di tipo String
  + PRECONDIZIONI
  + POSTCONDIZIONI
    - Viene mandato il contenuto dell’attributo richiesto al chiamante sotto forma di stringa

### Prodotto

Public class Prodotto

Attributi:

* private int IDprodotto;
* private int quantita;
* private String nomeProdotto;
* private String descrizioneProdotto;
* private double prezzoProdotto;
* private List<Categoria> categorieProdotto;

Metodi:

* Public void setInfoString(String info)
  + Questo metodo imposta uno degli attributi di tipo String con quella passata al metodo
  + PRECONDIZIONI
    - Info!=null
  + POSTCONDIZIONI
    - Viene assegnata una stringa all’attributo richiesto
* Public void setInfoInt(int info)
  + Questo metodo imposta uno degli attributi di tipo int con quello passato al metodo
  + PRECONDIZIONI
    - Info!=null
  + POSTCONDIZIONI
    - Viene assegnato un intero all’attributo richiesto
* Public String getInfoString()
  + Questo metodo recupera i dati contenuti in uno degli attributi di tipo String
  + PRECONDIZIONI
  + POSTCONDIZIONI
    - Viene mandato il contenuto dell’attributo richiesto al chiamante sotto forma di stringa
* Public int getInfoInt()
  + Questo metodo recupera i dati contenuti in uno degli attributi di tipo int
  + PRECONDIZIONI
  + POSTCONDIZIONI
    - Viene mandato il contenuto dell’attributo richiesto al chiamante sotto forma di intero
* Public List<Prodotto> getCategorie()
  + Questo metodo recupera tutti i prodotti presenti nel database
  + PRECONDIZIONI
  + POSTCONDIZIONI
    - Viene mandata la lista di categorie al chaimante
* Public void setCategorieProdotto(List<categoria> categorie)
  + Questo metodo imposta una lista di categorie per il prodotto
  + PRECONDIZIONI
    - Categorie!=null
  + POSTCONDIZIONI
    - Viene assegnata la lista all’attributo categorieProdotto di prodotto
* Public void setPrezzoProdotto(double prezzoProdotto)
  + Questo metodo imposta il prezzo del prodotto
  + PRECONDIZIONI
    - prezzoProdotto!=null
  + POSTCONDIZIONI
    - Viene assegnato un valore all’attributo prezzoProdotto

### Transazione

Public class Transazione

Attributi:

* public int idTransazione;
* public String username;
* public Date data;
* public double spesa;

Metodo:

* Public void setInfoString(String info)
  + Questo metodo imposta uno degli attributi di tipo String con quella passata al metodo
  + PRECONDIZIONI
    - Info!=null
  + POSTCONDIZIONI
    - Viene assegnata una stringa all’attributo richiesto
* Public void setInfoInt(int info)
  + Questo metodo imposta uno degli attributi di tipo int con quello passato al metodo
  + PRECONDIZIONI
    - Info!=null
  + POSTCONDIZIONI
    - Viene assegnato un intero all’attributo richiesto
* Public String getInfoString()
  + Questo metodo recupera i dati contenuti in uno degli attributi di tipo String
  + PRECONDIZIONI
  + POSTCONDIZIONI
    - Viene mandato il contenuto dell’attributo richiesto al chiamante sotto forma di stringa
* Public int getInfoInt()
  + Questo metodo recupera i dati contenuti in uno degli attributi di tipo int
  + PRECONDIZIONI
  + POSTCONDIZIONI
    - Viene mandato il contenuto dell’attributo richiesto al chiamante sotto forma di intero
* Public void setData(Date data)
  + Questo metodo imposta l’attributo data
  + PRECONDIZIONI
    - Data!=null
  + POSTCONDIZIONI
    - L’attributo data viene impostato
* Public Date getData()
  + Questo metodo recupera l’attributo data
  + PRECONDIZIONI
  + POSTCONDIZIONI
    - Viene mandato al chiamante il contenuto dell’attrbuto data
* Public void setSpesa(double spesa)
  + Questo metodo imposta l’attributo spesa
  + PRECONDIZIONI
    - Spesa!=null
  + POSTCONDIZIONI
    - Viene assegnato un valore all’attributo spesa
* Public double getSpesa()
  + Questo metodo recupera l’attributo spesa
  + PRECONDIZIONI
  + POSTCONDIZIONI
    - Viene passato il contenuto dell’attributo spesa al chiamante

## Connection

Questo package contiene le classi che si occupano del collegamento col database e del recupero dei dati da esso

|  |  |
| --- | --- |
| Class Summary | |
| ConPool | Classe con lo scopo di creare una connessione con il database |
| AccountDAO | Classe con lo scopo di gestire la comunicazione con il database per quanto riguarda gli oggetti di tipo Account |
| CartaDAO | Classe con lo scopo di gestire la comunicazione con il database per quanto riguarda gli oggetti di tipo Carta |
| CategoriaDAO | Classe con lo scopo di gestire la comunicazione con il database per quanto riguarda gli oggetti di tipo Categoria |
| ContactDAO | Classe con lo scopo di gestire la comunicazione con il database per quanto riguarda gli oggetti di tipo Contact |
| IndirizzoDAO | Classe con lo scopo di gestire la comunicazione con il database per quanto riguarda gli oggetti di tipo Indirizzo |
| ProdottoDAO | Classe con lo scopo di gestire la comunicazione con il database per quanto riguarda gli oggetti di tipo Prodotto |
| TransazioneDAO | Classe con lo scopo di gestire la comunicazione con il database per quanto riguarda gli oggetti di tipo Transazione |

### ConPool

Public class ConPool

Attributi:

* private static DataSource datasource;

Metodi:

* public static Connection getConnection()
  + Recupera un oggetto Connection contenete la connesione stabilita con il database
  + PRECONDIZIONI
    - Nessuna connessione già in corso con il database
  + POSTCONDIZIONI
    - Viene mandato al chiamante un riferimento alla connessione sotto forma di Connection

### AccountDAO

Public class AccountDAO

Metodi:

* Public Account doRetrieveByUsernameAndPassword(String username, String password)
  + Questo metodo recupera l’oggetto di tipo account identificato dalla combinazione di username e password
  + PRECONDIZIONI
    - Username != null &&
    - Password != null
  + POSTCONDIZIONI
    - Viene passato al chiamante un oggetto di tipo account contenente i dati dell’account desiderato
* Public Account doRetrieveByUsername(String username)
  + Questo metodo recupera l’oggetto di tipo account identificato dallo username passato al metodo per motrarne i dettagli al gestore
  + PRECONDIZIONI
    - Username!=null
  + POSTCONDIZIONI
    - Viene passato al chiamante un oggetto di tipo account contenenti i dati desiderati
* Public void doSave(Account account)
  + Questo metodo salva il contenuto dell’oggetto di tipo Account all’interno del database
  + PRECONDIZIONI
    - Account!=null
  + POSTCONDIZIONI
    - I dati contenuti nell’oggetto di tipo Account passato al metodo vengono salvati correttamente nel database
* Public ArrayList<Account> doRetrieveAll()
  + Questo metodo recupera tutti gli account presenti nel database
  + PRECONDIZIONI
  + POSTCONDIZIONI
    - L’ArrayList contenente i vari oggetti di tipo account viene passata al chiamante del metodo
* Public void doUpdate(String userOriginal, String Username, String Password, int CF, int NumCarta, String Indirizzo)
  + Questo metodo aggiorna l’account attuale con i dati passati al metodo
  + PRECONDIZIONI
    - userOriginal != null &&
    - Username != null &&
    - Password != null &&
    - CF!= null &&
    - NumCarta != null &&
    - Indirizzo != null
  + POSTCONDIZIONI
    - I dati dell’account sono stati aggiornati all’interno del database con successo
* Public void doDelete(String Username)
  + Questo metodo elimina l’account identificato dallo username passato al metodo
  + PRECONDIZIONI
    - Username!=null
  + POSTCONDIZIONI
    - I dati identificati dallo username passato al metodo sono stati eliminati dal database

### CartaDAO

Public class CartaDAO

Metodi:

* Public carta doRetrieveByNumCarta(int numCarta)
  + Questo metodo recupera l’oggetto di tipo Carta dal database identificato dal numero di carta passato al metodo
  + PRECONDIZIONI
    - numCarta!=null
  + POSTCONDIZIONI
    - Viene passato al chiamante l’oggetto di tipo carta appena creato
* Public void doSave(Carta carta)
  + Questo metodo salva i dati contenuti all’interno dell’oggetto carta passato al metodo all’interno di una nuova entry del database
  + PRECONDIZIONI
    - Carta!=null
  + POSTCONDIZIONI
    - I dati contenuti nell’oggetto di tipo carta passato al metodo vengono salvati correttamente nel database
* Public ArrayList<Carta> doRetrieveAll()
  + Questo metodo recupera tutti gli oggetti di tipo carta
  + PRECONDIZIONI
  + POSTCONDIZIONI
    - L’ArrayList contenente i vari oggetti di tipo carta viene passata al chiamante del metodo
* Public void doUpdate(int originalNumCarta, int numCarta, String dataDiScadenza, int CSV, String nomeTitolare, String cognomeTitolare)
  + Questo metodo aggiorna la carta identificata da originalNumCarta con i restanti dati passati al metodo
  + PRECONDIZIONI
    - originalNumCarta != null &&
    - numCarta != null &&
    - dataDiScadenza != null &&
    - CSV!= null &&
    - nomeTitolare != null &&
    - cognomeTitolare != null
  + POSTCONDIZIONI
    - I dati della carta sono stati aggiornati all’interno del database con successo
* Public void doDelete(int numCarta)
  + Questo metodo elimina la entry dell carta identificata dal numero di carta passato al metodo dal database
  + PRECONDIZIONI
    - numCarta!=null
  + POSTCONDIZIONI
    - I dati identificati dal numero di carta passato al metodo sono stati eliminati dal database

### CategoriaDAO

Public class CategoriaDAO

Metodi:

* Public categoria doRetrieveByID(int IDCategoria)
  + Questo metodo recupera la categoria identificata dall’ID della categoria passato al metodo dal database
  + PRECONDIZIONI
    - IDCategoria != null
  + POSTCONDIZIONI
    - Viene passato al chiamante un oggetto di tipo categoria contenente i dati della categoria corrispondente all’ID passato al metodo
* Private static List<prodotto> getProdotti(Connection con, int ID)
  + Questo metodo recupera i prodotti associati con la categoria identificata dall’ID passato al metodo
  + PRECONDIZIONI
    - Con!=null
    - ID != null
  + POSTCONDIZIONI
    - Viene passata una lista contenente i prodotti associati alla categoria specificata dall’ID passato al metodo
* Public void doSave(categoria Categoria)
  + Questo metodo salva i dati contenuti nell’oggetto categoria passato al metodo all’interno di una nuova entry nel databse
  + PRECONDIZIONI
    - Categoria!=null
  + POSTCONDIZIONI
    - I dati contenuti nell’oggetto di tipo categoria passato al metodo vengono salvati correttamente nel database
* Public ArrayList<Categoria> doRetrieveAll()
  + Questo metodo recupera tutte le categorie dal database
  + PRECONDIZIONI
  + POSTCONDIZIONI
    - L’ArrayList contenente i vari oggetti di tipo categoria viene passato al chiamante del metodo
* Public void doUpdate(int IDCategoria, String nomeCategoria, String descrizioneCategoria)
  + Questo metodo aggiorna la categoria identificata dall’IDCategoria passato al metodo con i restanti dati passati al metodo
  + PRECONDIZIONI
    - IDCategoria != null &&
    - nomeCategoria != null &&
    - descrizioneCategoria != null
  + POSTCONDIZIONI
    - I dati della categoria sono stati aggiornati all’interno del database con successo
* Public void doDelete(int IDCategoria)
  + Questo metodo elimina la entry identificata dall’ID passato al metodo dal database
  + PRECONDIZIONI
    - IDCategoria != null
  + POSTCONDIZIONI
    - I dati identificati dall’ID passato al metodo sono stati eliminati dal database

### ContactDAO

Public class ContactDAO

Metodi:

* Public contact doRetrieveByCF(int CF)
  + Questo metodo recupera il contact identificato dal CF passato al metodo dal database
  + PRECONDIZIONI
    - CF!=null
  + POSTCONDIZIONI
    - Viene passato al chiamante un oggetto di tipo contact contenente i dati personali del contact specificato dal Codice fiscale passato al metodo
* Public void doSave(Contact contact)
  + Questo metodo salva i dati contenuti nell’oggetto di tipo Contact passato al metodo all’interno di una nuova entry nel database
  + PRECONDIZIONI
    - Contact != null
  + POSTCONDIZIONI
    - I dati contenuti nell’oggetto di tipo contact passato al metodo vengono salvati correttamente nel database
* Public ArrayList<Contact> doRetrieveAll()
  + Questo metodo recupera tutti i contact dal database
  + PRECONDIZIONI
  + POSTCONDIZIONI
    - L’ArrayList contenente i vari oggetti di tipo contact viene passata al chiamante del metodo
* Public void doUpdate(int originalCF, int CF, String nome, String cognome, String dataDiNascita, String telefono, String mail)
  + Questo metodo aggiorna i dati nelle entry identificate da originalCF con i restanti dati passati al metodo
  + PRECONDIZIONI
    - originalCF != null &&
    - CF != null &&
    - Nome != null &&
    - Cognome != null &&
    - dataDiNascita != null &&
    - telefono != null &&
    - mail != null
  + POSTCONDIZIONI
    - I dati della carta sono stati aggiornati all’interno del database con successo
* Public void doDelete(int CF)
  + Questo metodo elimina la entry identifcata dal CF passato al metodo dal database
  + PRECONDIZIONI
    - CF!=null
  + POSTCONDIZIONI
    - I dati identificati dal codice fiscale passato al metodo sono stati eliminati dal database

### IndirizzoDAO

Public class IndirizzoDAO

Metodi:

* Public indirizzo doRetrieveByVia(String via)
  + Questo metodo recupera l’indirizzo identificato dalla via passata al metodo dal database
  + PRECONDIZIONI
    - Via!=null
  + POSTCONDIZIONI
    - Viene passato al chiamante un oggetto di tipo indirizzo identificato dalla via passata al metodo
* Public void doSave(Indirizzo indirizzo)
  + Questo metodo salva i dati contenuti nell’oggetto di tipo indirizzo passato al metodo all’interno di una nuova entry nel database
  + PRECONDIZIONI
    - Indirizzo!=null
  + POSTCONDIZIONI
    - I dati contenuti nell’oggetto di tipo indirizzo passato al metodo vengono salvati correttamente nel database
* Public ArrayList<Indirizzo> doRetrieveAll()
  + Questo metodo recupera tutti gli indirizzi nel database
  + PRECONDIZIONI
  + POSTCONDIZIONI
    - L’ArrayList contenente i vari oggetti di tipo indirizzo viene passato al chiamante del metodo
* Public void doUpdate(String originalVia, String via, int CAP, String via2, String nazione, String regione, String città)
  + Questo metodo aggiorna l’indirizzo identificato da originalVia con i restanti dati passati al metodo
  + PRECONDIZIONI
    - OriginalVia != null &&
    - via != null &&
    - CAP != null &&
    - Via2 != null &&
    - Nazione != null &&
    - Regione != null &&
    - Città != null
  + POSTCONDIZIONI
    - I dati dell’indirizzo sono stati aggiornati all’interno del database con successo
* Public void doDelete(String via)
  + Questo metodo elimina la entry identificata dalla via passata al metodo
  + PRECONDIZIONI
    - Via != null
  + POSTCONDIZIONI
    - I dati identificati dalla via passata al metodo sono stati eliminati dal database

### ProdottoDAO

Public class ProdottoDAO

Metodi:

* Public prodotto doRetrieveByID(int IDProdotto)
  + Questo metodo recupera il prodotto identificato dall’ID del prodotto passato al metodo dal database
  + PRECONDIZIONI
    - IDProdotto != null
  + POSTCONDIZIONI
    - Viene passato al chiamante un oggetto di tipo prodotto contenente i dati del prodotto specificato dall’ID passato al metodo
* Public void doSaveCategoria(int prodID, int catID)
  + Questo metodo esegue l’associazione tra la categoria identificata da catID e un prodotto identificato da prodID
  + PRECONDIZIONI
    - prodID!=null &&
    - catID!=null
  + POSTCONDIZIONI
    - Vengono associati i dati del prodotto desiderato con quelli della categoria desiderata
* Static List<Categoria> getCategorie(Connection con, int IDProdotto)
  + Questo metodo recupera tutte le categorie associate al prodotto nel database
  + PRECONDIZIONI
    - Con!=null &&
    - IDProdotto!=null
  + POSTCONDIZIONI
    - Viene passato al chiamante una lista di categorie associate al prodotto specificato dall’ID passato al metodo
* Public List<prodotto> getProdottiByNameLikeness(String searchTerm)
  + Questo metodo recupera tutti i prodotti con un nome simile a searchTerm
  + PRECONDIZIONI
    - searchTerm!=null
  + POSTCONDIZIONI
    - Viene passata al chiamante una lista di prodotti col nome somigliante al termine di ricerca passato al metodo
* Public void doSave(Prodotto prodotto)
  + Questo metodo salva i dati contenuti all’intenro dell’oggetto di tipo prodotto passato al metodo all’interno di una nuova entry nel database
  + PRECONDIZIONI
    - Prodotto!=null
  + POSTCONDIZIONI
    - I dati contenuti nell’oggetto di tipo prodotto passato al metodo vengono salvati correttamente nel database
* Public ArrayList<prodotto> doRetrieveAll(int offset, int limit)
  + Questo metodo recupera tutti i prodotti che partono da offset dal database fino a limit
  + PRECONDIZIONI
    - Offset!=null
    - Limit!=null
  + POSTCONDIZIONI
    - L’ArrayList contenente i vari oggetti di tipo prodotto viene passato al chiamante del metodo
* Public void doUpdate(int prodID, String nome, String descrizione, Double prezzo, int quantità)
  + Questo metodo aggiorna i dati del prodottto identificato da prodID con i restanti dati passati al metodo
  + PRECONDIZIONI
    - prodID != null &&
    - nome != null &&
    - descrizione != null &&
    - prezzo != null &&
    - quantità != null
  + POSTCONDIZIONI
    - I dati del prodotto sono stati aggiornati all’interno del database con successo
* Public void doDelete(int prodID)
  + Questo metodo elimina il prodotto identificato dall’ID del prodotto passato al metodo
  + PRECONDIZIONI
    - prodID!=null
  + POSTCONDIZIONI
    - I dati identificati dall’ID del prodotto passato al metodo sono stati eliminati dal database

### TransazioneDAO

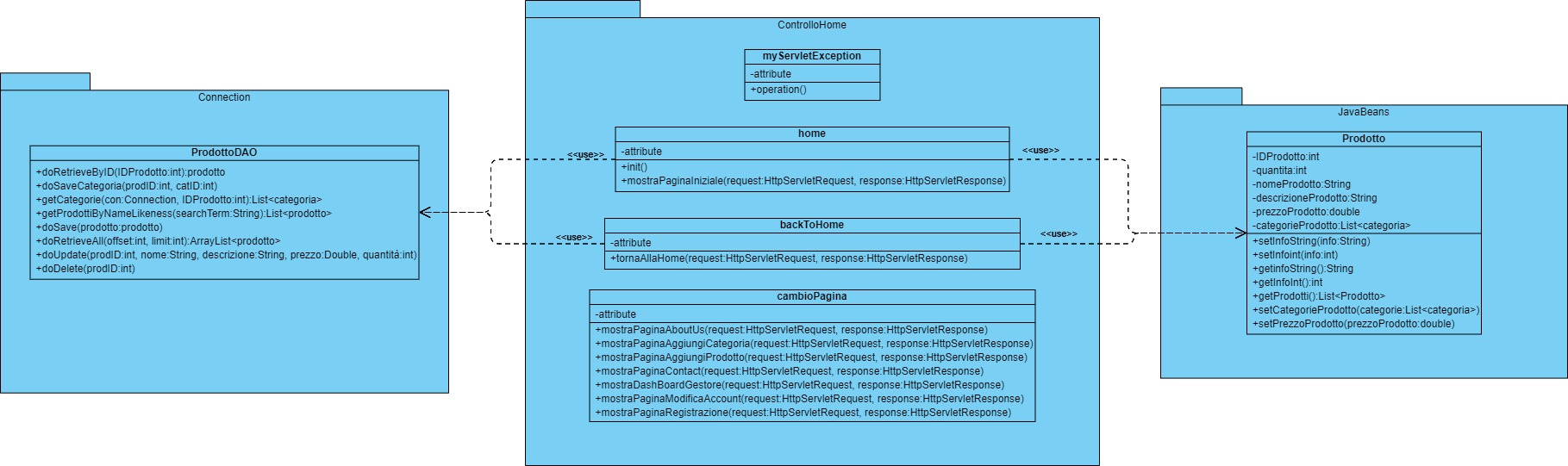
Public class TransazioneDAO

Metodi:

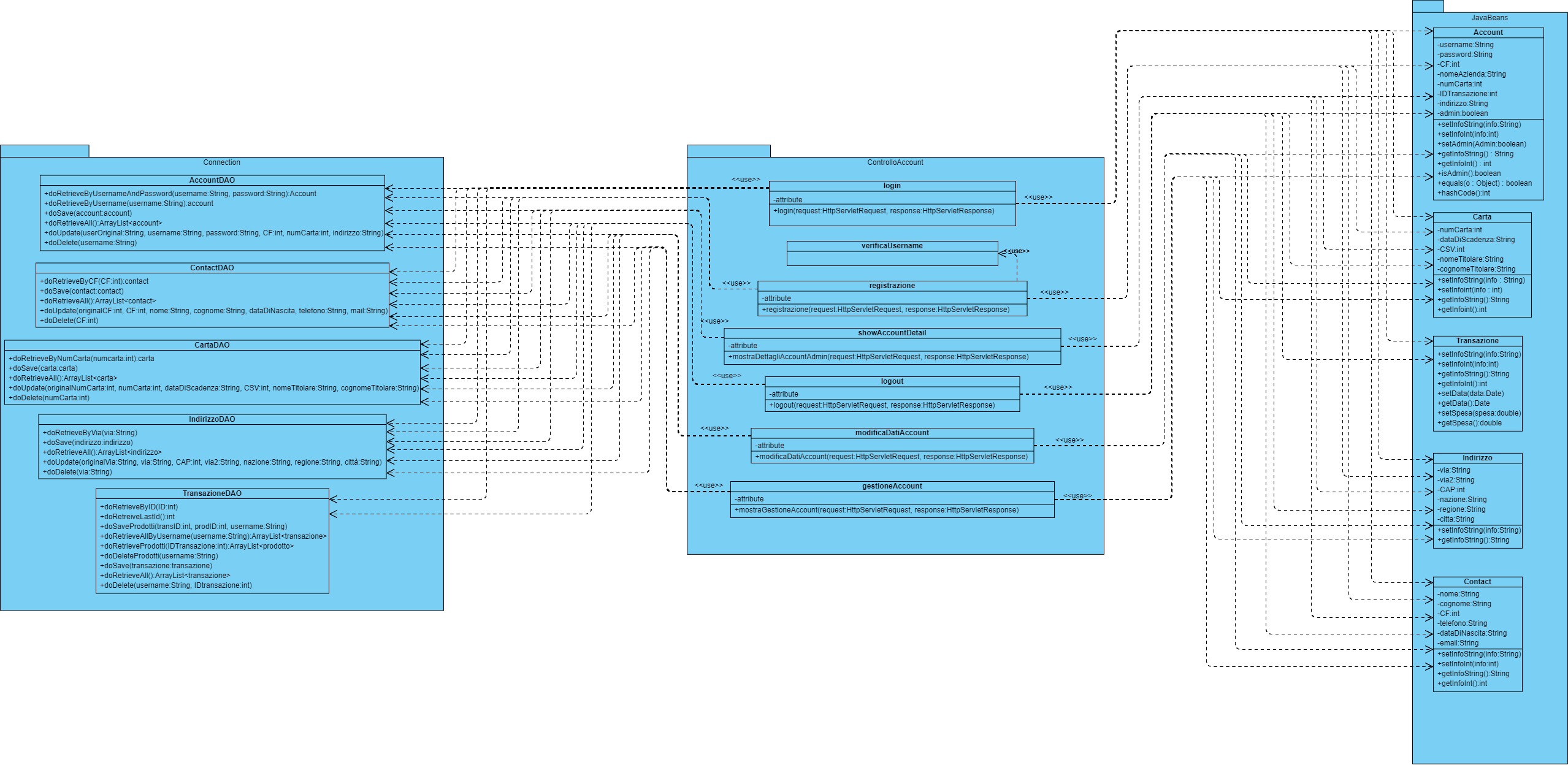
* Public Transazione doRetrieveByID(int ID)
  + Questo metodo recupera la transazione identificata dall’ID passato al metodo dal database
  + PRECONDIZIONI
    - ID!=null
  + POSTCONDIZIONI
    - Viene passato al chiamante un oggetto di tipo transazione contenente i dati della transazione specificata dall’ID passato al metodo
* Public int doRetrieveLastId()
  + Questo metodo recupera l’ID più grande associato alle transazioni dal database
  + PRECONDIZIONI
  + POSTCONDIZIONI
    - Viene passato l’ID massimo trovato nel database
* Public void doSaveProdotti(int transID, int prodID, String username)
  + Questo metodo crea un associazione tra la transazione identificata da transID e il prodotto identificato da prodID così come con l’account identificato dallo username
  + PRECONDIZIONI
    - transID!=null &&
    - prodID!=null &&
    - username!=null
  + POSTCONDIZIONI
    - Viene creata l’associazione tra prodotto e transazione
* Public ArrayList<transazione> doRetrieveAllByUsername(String username)
  + Questo metodo recupera tutte le transazione associate all’account identificato dallo username passato al metodo
  + PRECONDIZIONI
    - Username!=null
  + POSTCONDIZIONI
    - Viene passato al chiamante un ArrayList contenente tutte le transazioni effettuate dall’utente specificato dallo username passato al metodo
* Public ArrayList<Prodotto> doRetrieveProdotti(int IDTransazione)
  + Questo metodo recupera tutti i prodotti associati alla transazione identificata dall’ID della transazione passato al metodo
  + PRECONDIZIONI
    - IDTransazione != null
  + POSTCONDIZIONI
    - Viene passato al chiamante un ArrayList contenente tutti i prodotti associati alla transazione specificata dall’ID passato al metodo
* Public void doDeleteProdotti(String username)
  + Elimina le associazioni dei prodotto con le transazioni , dell’utente specificato dallo username passato al metodo
  + PRECONDIZIONI
    - Username!=null
  + POSTCONDIZIONI
    - Vengono eliminate le associazioni tra prodotto e transazione associate all’account specificato dallo username passato al metodo
* Public void doSave(Transazione transazione)
  + Questo metodo salva i dati contenuti nell’oggetto transazione passato al metodo in una nuova entry nel database
  + PRECONDIZIONI
    - Transazione != null
  + POSTCONDIZIONI
    - I dati contenuti nell’oggetto di tipo transazione passato al metodo vengono salvati correttamente nel database
* Public ArrayList<Transazione> doRetrieveAll()
  + Questo metodo recupera tutte le transazioni nel database
  + PRECONDIZIONI
  + POSTCONDIZIONI
    - L’ArrayList contenente i vari oggetti di tipo transazione viene passata al chiamante del metodo
* Public void doDelete(String username, int IDtransazione)
  + Questo metodo elimina la transazione identificata dalla combinazione di username e IDtransazione
  + PRECONDIZIONI
    - Username!=null &&
    - IDTransazione!=null
  + POSTCONDIZIONI
    - I dati identificati dallo username e dall’ID della transazione passati al metodo sono stati eliminati dal database

#### CLASS DIAGRAMS

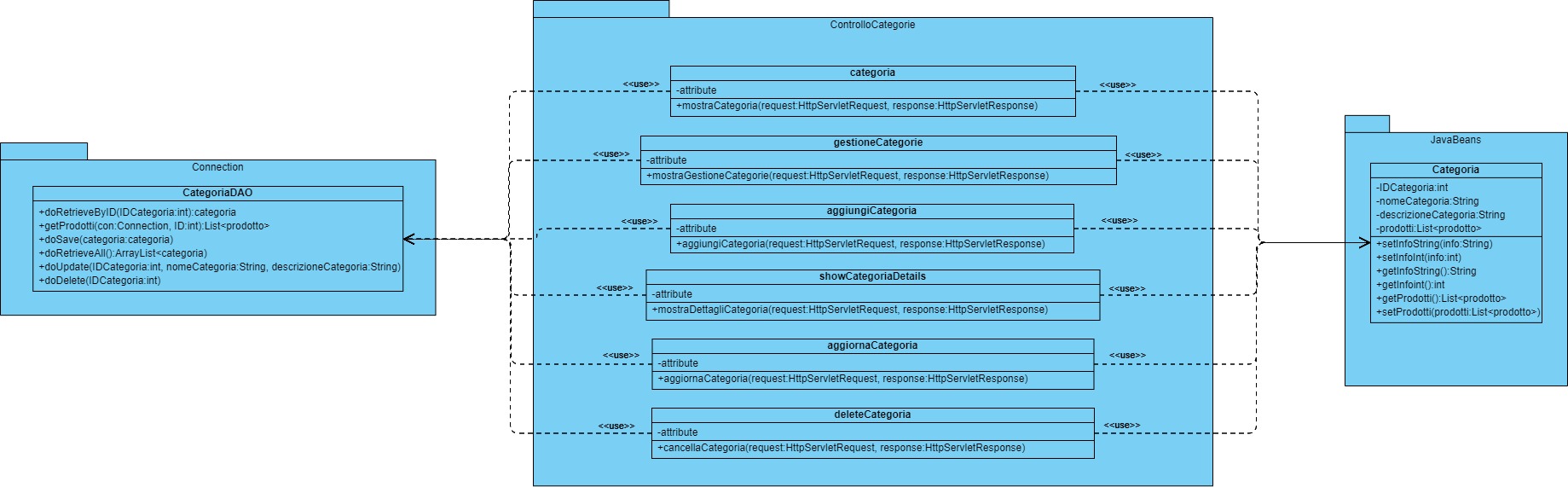
## ControlloHome



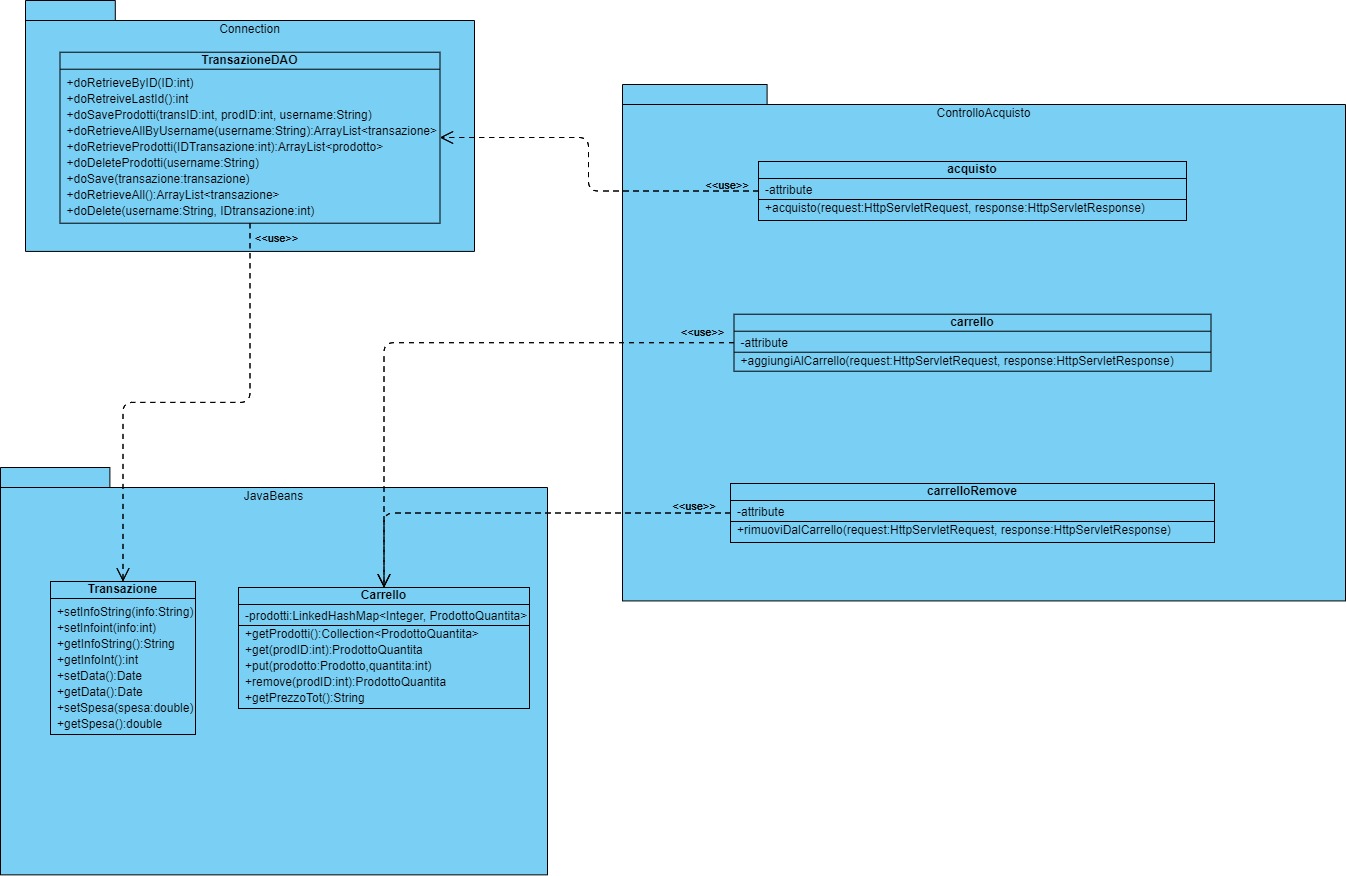
## ControlloAccount



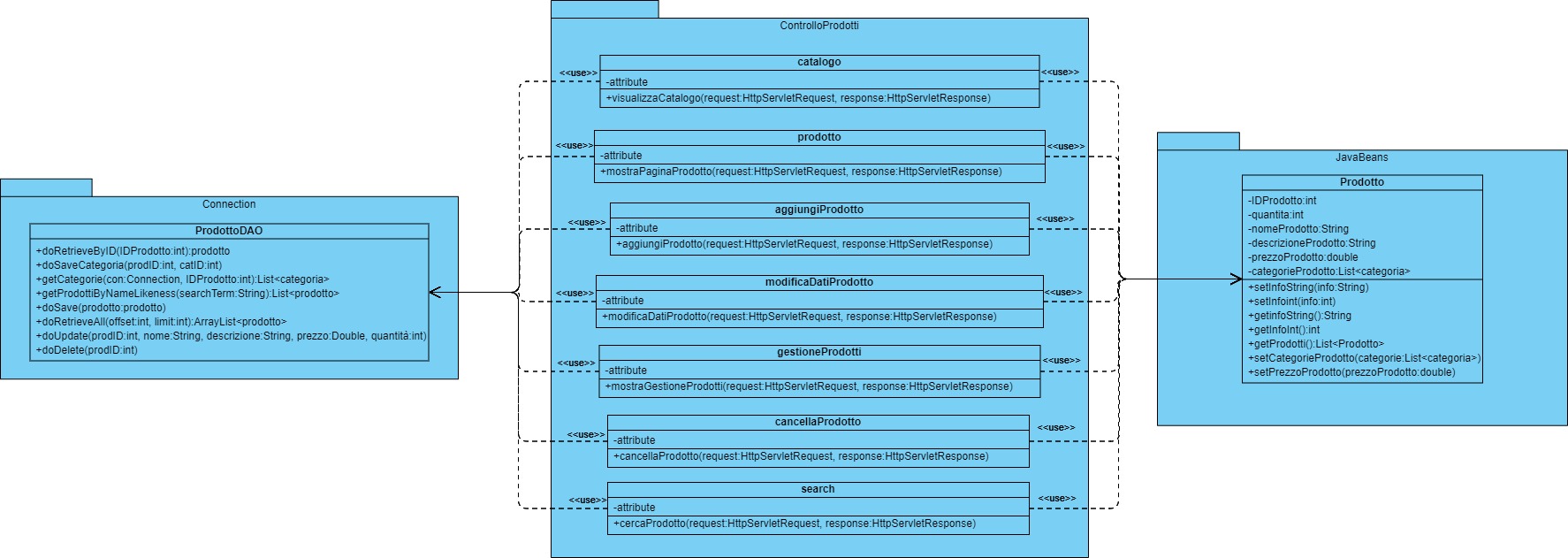
## ControlloCategorie



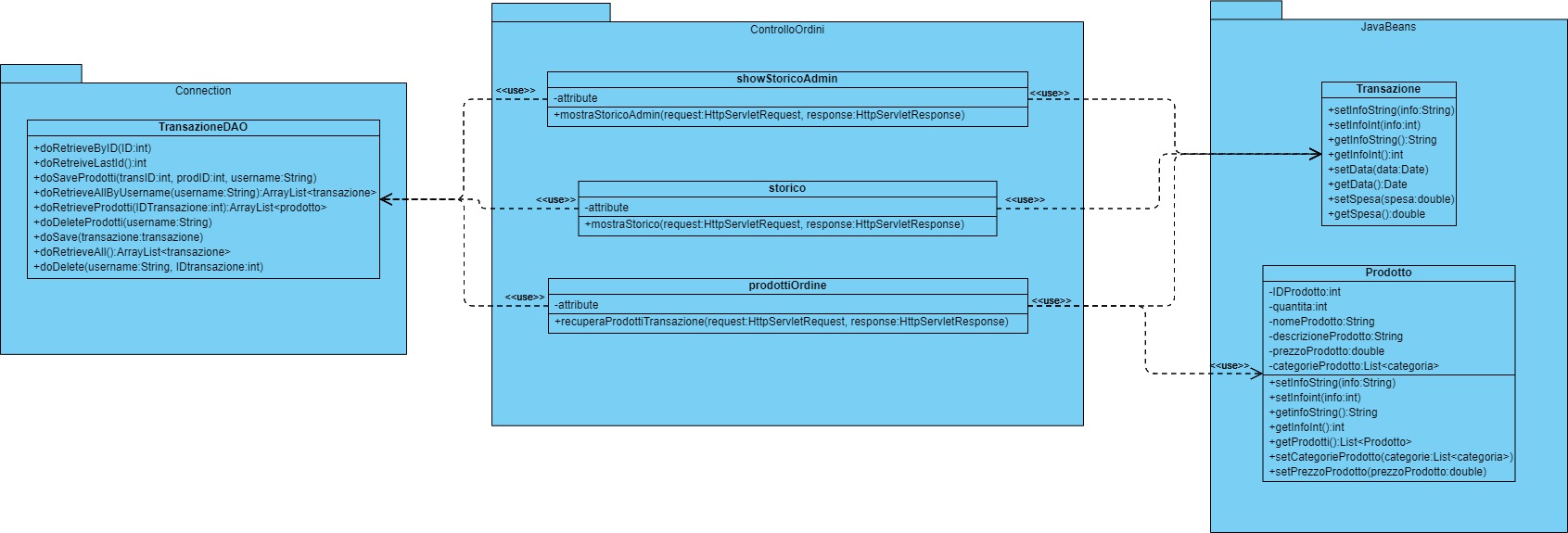
## ControlloAcquisto



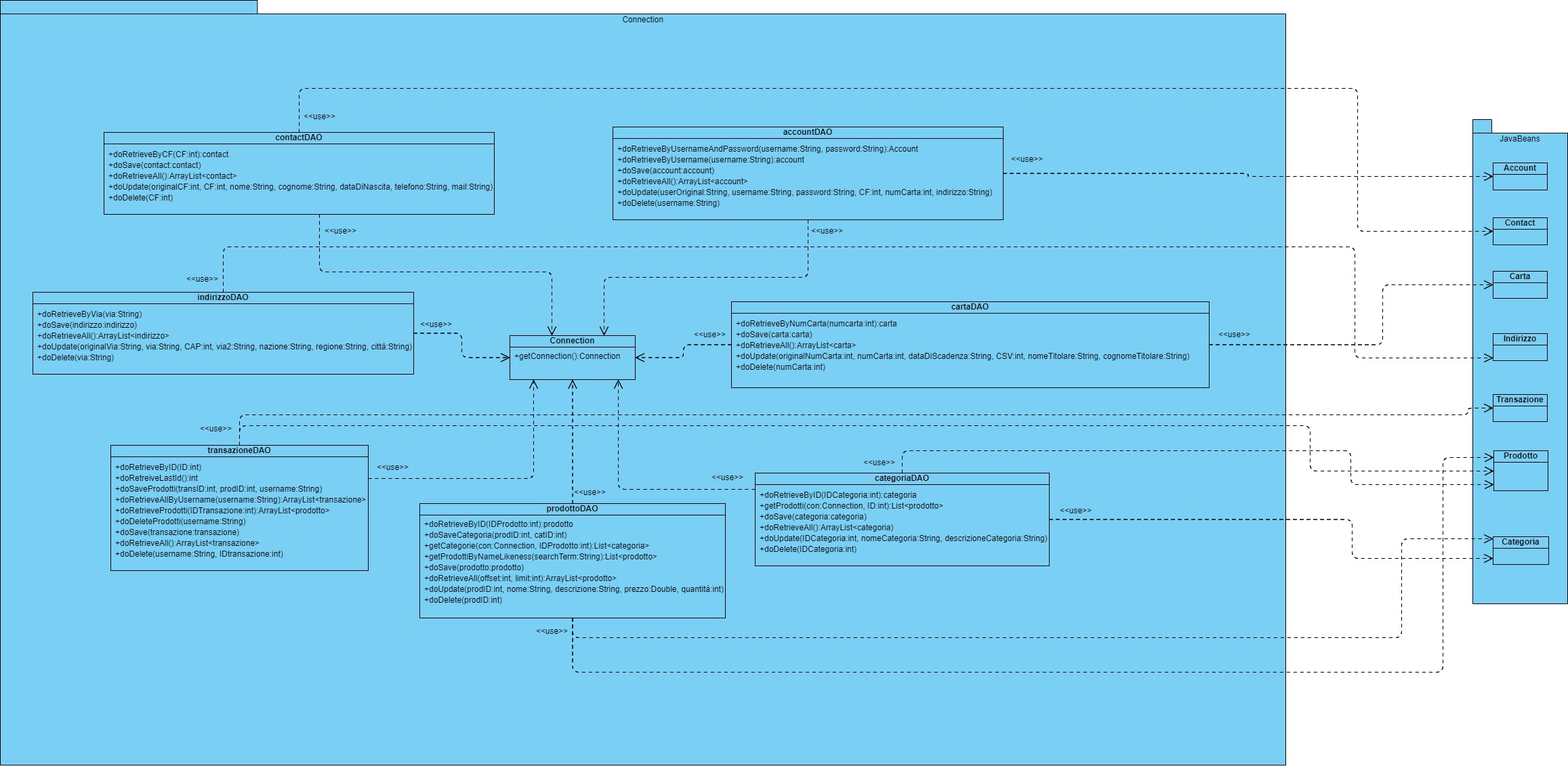
## ControlloProdotti



## ControlloOrdini



## Connection



## JavaBeans

