



***Object Design Document***

Data: 02/11/2020

**Coordinatore del Progetto:**

|  |  |
| --- | --- |
| **Nome** | **Matricola** |
| Francesco Di Palma | 0512104586 |

**Partecipanti:**

|  |  |
| --- | --- |
| **Nome** | **Matricola** |
| Giovanni Di Mauro | 0512104596 |
| Francesco Di Palma | 0512104586 |
| Maria Giuseppina Mosca | 0512106090 |
| Francesco Saviano | 0512104912 |

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Versione** | **Descrizione** | **Autore** |
| 26/12/2020 | 0.1 | Prima stesura documento | Francesco Di Palma  Giovanni Di Mauro  Maria Giuseppina Mosca |
| 27/12/2020 | 0.2 | Class Interface | Francesco Di Palma  Giovanni Di Mauro  Maria Giuseppina Mosca |

SOMMARIO

1 Introduzione.......................................................................................................................................................5

1.1 Object Design Trade-offs...................................................................................................................5 1.2 Linee Guida per la Documentazione delle Interfacce ........................................................................6 1.3 Definizioni, acronimi e abbreviazioni.................................................................................................7

1.4 Riferimenti.........................................................................................................................................7

2 Packages............................................................................................................................................................7

2.1 Package control ................................................................................................................................8

2.2 Package bean..................................................................................................................................10

2.3 Package connectionPool.................................................................................................................11

2.4 Package model................................................................................................................................11

2.5 Amministratore ..............................................................................................................................12

2.6 Utente ............................................................................................................................................13

3 Class Interface .................................................................................................................................................15

3.1 Interface Bean ................................................................................................................................15

3.2 Interface Manager..........................................................................................................................16

4 Design Pattern................................................................................................................................................22

4.1 Object Pool Pattern ........................................................................................................................22

**1.0 INTRODUZIONE**

* 1. **OBJECT DESIGN TRADE-OFFS**

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

* **Comprensibilità vs Tempo:**

Il codice deve essere quanto più comprensibile possibile per facilitare la fase di testing ed eventuali future modifiche. Il codice sarà quindi accompagnato da commenti che ne semplifichino la comprensione. Ovviamente questa caratteristica aggiungerà un incremento di tempo allo sviluppo del nostro progetto. **Interfaccia vs Usabilità:**

L’interfaccia grafica è stata realizzata in modo da essere molto semplice, chiara e concisa, fa uso di form e pulsanti disposti in maniera da rendere semplice l’utilizzo del sistema a quanti più utenti possibili.

* **Sicurezza vs Efficienza:**

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

* **Response Time vs Hardware:**

Il sistema garantisce una certa reattività alle richieste, e quindi essere in grado di poter comunque offrire una contemporaneità di servizi agli utenti. Ovviamente questa caratteristica sarà limitata dall’hardware del sistema.

* **Prestazioni vs Costi**

Il sistema prevede l’utilizzo di template open source esterni per mantenere prestazioni elevate, essendo il progetto sprovvisto di budget.

* 1. **LINEE GUIDA PER LA DOCUMENTAZIONE DELLE INTERFACCE**

Gli sviluppatori dovranno seguire le seguenti convenzioni per la scrittura del codice:

**Naming Convention**

• È buona norma utilizzare nomi:

• Descrittivi

• Di uso comune

• Lunghezza medio-corta

**Variabili**

• I nomi delle variabili devono cominciare con una lettera minuscola, se il nome della variabile è costituito da più parole, solo l’iniziale delle altre parole sarà maiuscola (es: nomeVariabile).

**Metodi**

• I nomi dei metodi devono cominciare con una lettera minuscola, e le parole seguenti con la lettera maiuscola. Il nome del metodo tipicamente consiste di un verbo che identifica una azione, seguito dal nome di un oggetto. • I nomi dei metodi per l’accesso e la modifica delle variabili dovranno essere del tipo getNomeVariabile() e setNomeVariabile().

**Classi e pagine**

• I nomi delle classi e delle pagine devono iniziare con una lettera maiuscola, le parole contenute al suo interno devono cominciare con lettera maiuscola. Il nome deve fornire informazioni utili relative al loro scopo. Ogni file sorgente contiene una singola classe e deve essere strutturato in un determinato modo:

• L’ istruzione package che permette di inserire la classe in un determinato package

• L’ istruzione import che importa le librerie necessarie alla class

• Una piccola descrizione della classe

* 1. **DEFINIZIONI, ACRONIMI E ABBREVIAZIONI**

RAD = Requirement Analysis Document

SDD = System Design Document

ODD = Object Design Document

* 1. **RIFERIMENTI**

Riferimenti a Rad\_IlGocciolatoio e a Sdd\_IlGocciolatoio

**2.0 PACKAGES**

Abbiamo diviso il sistema in:

• Control

• Bean

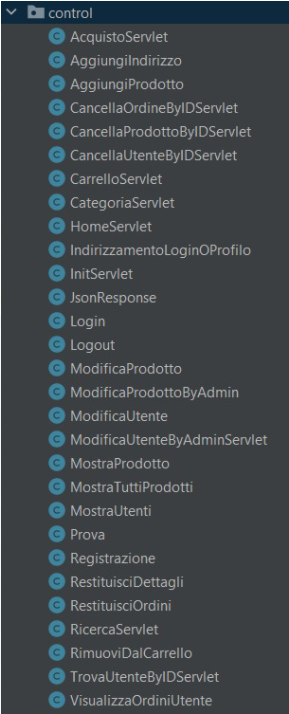
• ConnectionPool

• Model

• View

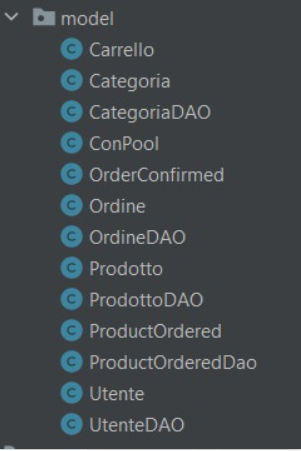
• Test

**2.1 control:**



|  |  |
| --- | --- |
| Classe | Descrizione |
| AcquistoServlet | Servlet che gestisce l’acquisto |
| AggiungiIndirizzo | Servlet che gestisce l’aggiunta di un indirizzo |
| CancellaOrdineByIDServlet | Servlet che gestisce la rimozione di un Ordine tramite id |
| CancellaProdottoByIDServlet | Servlet che gestisce la rimozione di un Prodotto tramite id |
| CancellaUtenteByIDServlet | Servlet che gestisce la rimozione di un Utente tramite id |
| CarrelloServlet | Servlet che gestisce il carrello |
| CategoriaServlet | Servlet che gestisce le categorie dei prodotti |
| HomeServlet | Servlet che gestisce i Prodotti destinati nella HomePage |
| IndirizzamentoLoginOProfilo | Servlet che gestisce il reidirizzamento del Profilo |
| InitServlet | Servlet che gestisce l’avvio della web application |
| JsonResponse | Servlet che gestisce i suggerimenti di ricerca |
| Login | Servlet che gestisce il login al sito |
| Logout | Servlet che gestisce il Logout dal sito |
| ModificaProdotto | Servlet che gestisce la modifica di un Prodotto |
| ModificaProdottoByAdmin | Servlet che gestisce la lista di Prodotti da modificare |
| ModificaUtente | Servlet che gestisce la modifica di un Utente |
| ModificaUtenteByAdminServlet | Servlet che gestisce la modifica di un Utente da parte di un Admin |
| MostraProdotto | Servlet che gestisce la visualizzazione di un Prodotto nel dettaglio |
| MostraTuttiProdotti | Servlet che gestisce la visualizzazione di tutti i Prodotti |
| MostraUtenti | Servlet che gestisce la visualizzazione di tutti gli utenti |
| Registrazione | Servlet che gestisce la registrazione al sito |
| RestituisciDettagli | Servlet che gestisce la visualizzazione dei dettagli di un Ordine |
| RestituisciOrdini | Servlet che gestisce la visualizzazione della lista degli Ordini |
| RicercaServlet | Servlet che gestisce la ricerca di un Prodotto nella barra di ricerca |
| RimuoviDalCarrello | Servlet che gestisce la rimozione di un elemento dal Carrello |
| TrovaUtenteByIDSearvlet | Servlet che gestisce la ricerca di un Utente tramite id |
| VisualizzaOrdiniUtente | Servlet che gestisce la visualizzazione di tutti gli ordini di un utente |

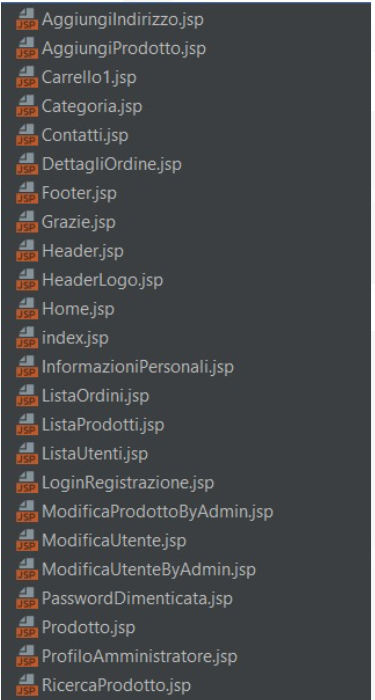
**2.2 Bean, Model e ConnectionPool:**



|  |  |  |
| --- | --- | --- |
| **Classe** | **Tipo** | **Descrizione** |
| Carrello | B | Descrive un Carrello |
| Categoria | B | Descrive una Categoria |
| CategoriaDao | M | Descrive l’interazione con il database per la Categoria |
| ConPool | C | Classe che implementa l’object Pool Pattern, responsabile di fornire connessione con database. |
| OrderConfirmed | M | Descrive l’interazione con il database per l’OrdineConfermato |
| Ordine | B | Descrive un Ordine |
| OrdineDAO | M | Descrive l’interazione con il database per la Ordine |
| Prodotto | B | Descrive un Prodotto |
| ProdottoDao | M | Descrive l’interazione con il database per la Prodotto |
| ProductOrdered | B | Descrive un ProdottoOrdered |
| ProductOrderedDao | M | Descrive l’interazione con il database per la ProdottoOrdinato |
| Utente | B | Descrive un Utente |
| UtenteDao | M | Descrive l’interazione con il database per l’Utente |

**C=connectionPool/M=model/B=Bean**

**2.3: View:**



|  |  |
| --- | --- |
| **View** | **Descrizione** |
| AggiungiIndirizzo.jsp | Pagina che mostra un form per aggiungere un indirizzo |
| AggiungiProdotto.jsp | Pagina che mostra un form per aggiungere un prodotto |
| Carrello1.jsp | Pagina che mostra il Carrello |
| Categoria.jsp | Pagina che mostra la categoria selezionata |
| Contatti.jsp | Pagina che mostra le informazioni personali del sito |
| DettagliOrdine.jsp | Pagina che mostra nel dettaglio un ordine |
| Footer.jsp | Il footer del sito |
| Grazie.jsp | Pagina di ringraziamento di fine acquisto |
| HeaderLogo.jsp | Logo del sito |
| Home.jsp | Pagina principale del sito |
| InformazioniPersonali.jsp | Pagina personale dell’utente |
| ListaOrdini.jsp | Pagina che mostra la lista degli ordini |
| ListaProdotti.jsp | Pagina che mostra la lista dei prodotti |
| ListaUtenti.jsp | Pagina che mostra la lista degli Utenti |
| LoginRegistrazione.jsp | Pagina di login e registrazione |
| ModificaProdottoByAdmin.jsp | Pagina che mostra la lista di prodotti che si vuole modificare |
| Prodotto.jsp | Pagina che mostra il singolo prodotto |
| ProfiloAmministratore.jsp | Pagina che mostra il profilo dell’amministratore |
| RicercaProdotto.jsp | Pagina che mostra la lista di prodotti ricercati nella barra di ricerca |

**3.0 CLASS INTERFACE**

**3.1 ENTITY**

**3.2 MODEL**

**3.3 CONTROL**

|  |  |
| --- | --- |
| Nome classe | CancellaUtenteByIDServlet |
| Descrizione | Questa classe è un control che si occupa di gestire la rimozione di un Utente tramite id |
| Signature dei metodi | + doGet(request: HttpServletRequest, response: HttpServletRequest):void |
| Pre-condizioni | **Context** CancellaUtenteByIDServlet:: doGet(request: HttpServletRequest, response: HttpServletRequest):void  **Pre**: request.getParameter (“id”) |
| Post-condizioni | **Context** CancellaUtenteByIDServlet:: doGet(request: HttpServletRequest, response: HttpServletRequest):void  **post**: uDAO.deleteUser(id) |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | CarrelloServlet |
| Descrizione | Questa classe è un control che si occupa di gestire il carrello |
| Signature dei metodi | + doGet(request: HttpServletRequest, response: HttpServletRequest):void |
| Pre-condizioni | **Context** CarrelloServlet:: doGet(request: HttpServletRequest, response: HttpServletRequest):void  **Pre**: request.getSession() |
| Post-condizioni |  |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | CategoriaServlet |
| Descrizione | Questa classe è un control che si occupa di gestire le categorie dei prodotti |
| Signature dei metodi | + doGet(request: HttpServletRequest, response: HttpServletRequest):void |
| Pre-condizioni | **Context** CategoriaServlet:: doGet(request: HttpServletRequest, response: HttpServletRequest):void  **Pre**: request.getParamter(“category”) |
| Post-condizioni | **Context** CategoriaServlet:: doGet(request: HttpServletRequest, response: HttpServletRequest):void  **post**: proDAO.retriveCategory(c) |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | HomeServlet |
| Descrizione | Questa classe è un control che si occupa di gestire i Prodotti destinati nella HomePage |
| Signature dei metodi | + doGet(request: HttpServletRequest, response: HttpServletRequest):void |
| Pre-condizioni |  |
| Post-condizioni | **Context** HomeServlet:: doGet(request: HttpServletRequest, response: HttpServletRequest):void  **post**: una lista contenente 8 prodotti casuali |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | IndirizzamentoLoginOProfilo |
| Descrizione | Questa classe è un control che si occupa di gestire il reidirizzamento del Profilo |
| Signature dei metodi | + doGet(request: HttpServletRequest, response: HttpServletRequest):void |
| Pre-condizioni | **Context** IndirizzamentoLoginOProfilo:: doGet(request: HttpServletRequest, response: HttpServletRequest):void  **Pre**: request.getSession() AND request.getSession().getAttribute(“utente”) |
| Post-condizioni |  |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | InitServlet |
| Descrizione | Questa classe è un control che si occupa di gestire l’avvio della web application |
| Signature dei metodi | +doGet(request: HttpServletRequest, response: HttpServletRequest):void |
| Pre-condizioni |  |
| Post-condizioni | **Context** InitServlet:: doGet(request: HttpServletRequest, response: HttpServletRequest):void  **post**: catDAO.retriveAll() |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | JsonResponse |
| Descrizione | Questa classe è un control che si occupa di gestire i suggerimenti per la barra di ricerca |
| Signature dei metodi | +doPost(resp: HttpServletRequest, req: HttpServletRequest):void |
| Pre-condizioni |  |
| Post-condizioni | **Context** JsonResponse:: doGet(resp: HttpServletRequest, req: HttpServletRequest)  **post**: prodao.retriveAll() |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | Login |
| Descrizione | Questa classe è un control che si occupa di gestire il Login di un utente nel sito |
| Signature dei metodi | +doGet(request: HttpServletRequest, response: HttpServletRequest):void |
| Pre-condizioni |  |
| Post-condizioni | **Context** Logout:: doGet(request: HttpServletRequest, response: HttpServletRequest)  **post**: request.getSession().removeAttribute(“utente”) |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | Logout |
| Descrizione | Questa classe è un control che si occupa di gestire il Logout di un utente dal sito |
| Signature dei metodi | +doGet(request: HttpServletRequest, response: HttpServletRequest):void |
| Pre-condizioni |  |
| Post-condizioni | **Context** Logout:: doGet(request: HttpServletRequest, response: HttpServletRequest)  **post**: request.getSession().removeAttribute(“utente”) |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | ModificaProdotto |
| Descrizione | Questa classe è un control che si occupa di gestire la modifica di un prodotto |
| Signature dei metodi | +doGet(request: HttpServletRequest, response: HttpServletRequest):void |
| Pre-condizioni | **Context**: ModificaProdottoByAdmin:: doGet(request: HttpServletRequest, response: HttpServletRequest)  **Pre**: request.getParameter(“nome”) AND request.getParamter(“descrizione”) AND request.getParamter(“anno”) AND request.getParamter(“regione”) AND request.getParamter(“gradazione”) AND request.getParamter(“formato”) AND request.getParamter(“quantita\_magazzino”) AND request.getParamter(“tipo”) AND request.getParamter(“prezzo”) AND request.getParamter(“sconto”) |
| Post-condizioni | **Context** ModificaProdottoByAdmin:: doGet(request: HttpServletRequest, response: HttpServletRequest)  **post**: proDAO.doUpdate(p) |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | ModificaProdottoByAdmin |
| Descrizione | Questa classe è un control che si occupa di gestire la selezione di un prodotto che l’admin intende modificare |
| Signature dei metodi | +doGet(request: HttpServletRequest, response: HttpServletRequest):void |
| Pre-condizioni | **Context** ModificaProdottoByAdmin:: doGet(request: HttpServletRequest, response: HttpServletRequest)  **Pre**: request.getParameter(“id”) |
| Post-condizioni | **Context** ModificaProdottoByAdmin:: doGet(request: HttpServletRequest, response: HttpServletRequest)  **post**: uDAO.retriveOne(id) |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | ModificaUtente |
| Descrizione | Questa classe è un control che si occupa di gestire la modifica dei dati personali di un Utente |
| Signature dei metodi | +doGet(request: HttpServletRequest, response: HttpServletRequest):void |
| Pre-condizioni | **Context** MostraUtenti:: doGet(request: HttpServletRequest, response: HttpServletRequest)  **Pre**: request.getParameter(“id”) AND request.getParameter(“username”) AND request.getParameter(“email”) AND request.getParameter(“citta”) AND request.getParameter(“cap”) AND request.getParameter(“via”) AND request.getParameter(“ncivico”) AND request.getParameter(“ruolo”) AND request.getParameter(“nome”) AND request.getParameter(“cognome”) AND request.getParameter(“datanascita”) |
| Post-condizioni | **Context** MostraUtenti:: doGet(request: HttpServletRequest, response: HttpServletRequest)  **post**: userDAO.doUpdate(u) |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | ModificaUtenteByAdminServlet |
| Descrizione | Questa classe è un control che si occupa di gestire la modifica dei dati personali di un Utente da parte di un Admin |
| Signature dei metodi | +doGet(request: HttpServletRequest, response: HttpServletRequest):void |
| Pre-condizioni | **Context** MostraUtenti:: doGet(request: HttpServletRequest, response: HttpServletRequest)  **Pre**: request.getParameter(“id”) AND request.getParameter(“username”) AND request.getParameter(“email”) AND request.getParameter(“citta”) AND request.getParameter(“cap”) AND request.getParameter(“via”) AND request.getParameter(“ncivico”) AND request.getParameter(“ruolo”) AND request.getParameter(“nome”) AND request.getParameter(“cognome”) AND request.getParameter(“datanascita”) |
| Post-condizioni | **Context** MostraUtenti:: doGet(request: HttpServletRequest, response: HttpServletRequest)  **post**: userDAO.doUpdate(u) |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | MostraProdotto |
| Descrizione | Questa classe è un control che si occupa di gestire la visualizzazione di un Prodotto nel dettaglio |
| Signature dei metodi | +doGet(request: HttpServletRequest, response: HttpServletRequest):void |
| Pre-condizioni | **Context** MostraProdotto:: doGet(request: HttpServletRequest, response: HttpServletRequest)  **Pre**: request.getParameter(“product”); |
| Post-condizioni | **Context** MostraProdotto:: doGet(request: HttpServletRequest, response: HttpServletRequest)  **post**: proDAO.retriveOne(Integet.pardeInt(product)) |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | MostraTuttiProdotti |
| Descrizione | Questa classe è un control che si occupa di gestire la visualizzazione di tutti i Prodotti |
| Signature dei metodi | +doGet(request: HttpServletRequest, response: HttpServletRequest):void |
| Pre-condizioni |  |
| Post-condizioni | **Context** MostraTuttiProdotti:: doGet(request: HttpServletRequest, response: HttpServletRequest)  **post**: proDAO.retriveAll() |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | MostraUtenti |
| Descrizione | Questa classe è un control che si occupa di gestire la visualizzazione di tutti gli utenti |
| Signature dei metodi | +doGet(request: HttpServletRequest, response: HttpServletRequest):void |
| Pre-condizioni |  |
| Post-condizioni | **Context** MostraUtenti:: doGet(request: HttpServletRequest, response: HttpServletRequest)  **post**: userDAO.retriveAll() |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | Registrazione |
| Descrizione | Questa classe è un control che si occupa di gestire la registrazione al sito |
| Signature dei metodi | +doGet(request: HttpServletRequest, response: HttpServletRequest):void |
| Pre-condizioni | **Context** Registrazione:: doGet(request: HttpServletRequest, response: HttpServletRequest)  **Pre**: request.getParameter(“username”) AND request.getParameter(“email”) AND  request.getParameter(“pass”) AND  request.getParameter(“dataN”) AND  request.getParameter(“nome”) AND request.getParameter(“Cognome”) |
| Post-condizioni | **Context** Registrazione:: doGet(request: HttpServletRequest, response: HttpServletRequest)  **post**: utenteDAO.doSave(u) |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | RestituisciDettegli |
| Descrizione | Questa classe è un control che si occupa di gestire la visualizzazione dei dettagli di un Ordine |
| Signature dei metodi | +doGet(request: HttpServletRequest, response: HttpServletRequest):void |
| Pre-condizioni | **Context** RestituisciDettegli:: doGet(request: HttpServletRequest, response: HttpServletRequest)  **Pre**: request.getParameter(“idord”) AND request.getParameter(“iduser”) |
| Post-condizioni | **Context** RestituisciDettegli:: doGet(request: HttpServletRequest, response: HttpServletRequest)  **post**: poaDAO.retriveByOrderID(ido) |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | RestituisciOrdini |
| Descrizione | Questa classe è un control che si occupa di gestire la lista di tutti gli ordini |
| Signature dei metodi | +doGet(request: HttpServletRequest, response: HttpServletRequest):void |
| Pre-condizioni |  |
| Post-condizioni | **Context** RestituisciOrdini:: doGet(request: HttpServletRequest, response: HttpServletRequest)  **post**: orderDAO.retriveAll() AND orderDAO.retriveAll() |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | RicercaServlet |
| Descrizione | Questa classe è un control che si occupa di gestire la ricerca di un prodotto |
| Signature dei metodi | +doGet(request: HttpServletRequest, response: HttpServletRequest):void |
| Pre-condizioni | **Context**  RicercaServlet :: doGet(request: HttpServletRequest, response: HttpServletRequest)  **Pre**: request.getParameter(“StringP”) |
| Post-condizioni | **Context** RicercaServlet:: doGet(request: HttpServletRequest, response: HttpServletRequest)  **post**: proDAO.retriveBySearch(s); |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | RimuoviDalCarrello |
| Descrizione | Questa classe è un control che si occupa di gestire la rimozione di un elemento dal carrello |
| Signature dei metodi | +doGet(request: HttpServletRequest, response: HttpServletRequest):void |
| Pre-condizioni | **Context**  RimuoviDalCarrello :: doGet(request: HttpServletRequest, response: HttpServletRequest)  **Pre**: request.getParameter(“id”) |
| Post-condizioni | **Context** RimuoviDalCarrello:: doGet(request: HttpServletRequest, response: HttpServletRequest)  **post**: carrello.remove(id) |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | TrovaUtenteByIDServlet |
| Descrizione | Questa classe è un control che si occupa di gestire la ricerca di un Utente tramite id |
| Signature dei metodi | +doGet(request: HttpServletRequest, response: HttpServletRequest):void |
| Pre-condizioni | **Context**  TrovaUtenteByIDServlet :: doGet(request: HttpServletRequest, response: HttpServletRequest)  **Pre**: request.getParameter(“id”) |
| Post-condizioni |  |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | VisualizzaOrdiniUtente |
| Descrizione | Questa classe è un control che si occupa di gestire gli ordini di un utente. |
| Signature dei metodi | +doGet(request: HttpServletRequest, response: HttpServletRequest):void |
| Pre-condizioni | **Context** VisualizzaOrdiniUtente:: doGet(request: HttpServletRequest, response: HttpServletRequest)  **Pre**: request.getSession().getAttribute(“utente”) != null |
| Post-condizioni |  |
| Invariante |  |