Università degli Studi di Salerno

Corso di Ingegneria Del Software

Eat & Reorder

Object Design

Document

**Docente:**

Prof. Andrea De Lucia

**Studenti:**

Abate Francesco

Dello Buono Marco

De Martino Vincenzo

Gagliardi Rosario

Sommario

[1. Introduzione 2](#_Toc30262414)

[1.1 Trade-offs 2](#_Toc30262415)

[1.2 Linee guida per la Documentazione delle Interfacce 3](#_Toc30262416)

[1.3 Definizioni, acronimi e abbreviazioni 4](#_Toc30262417)

[1.4 Riferimenti 4](#_Toc30262418)

[1.5 Design Pattern 4](#_Toc30262419)

[2. Packages 5](#_Toc30262420)

[2.1 Servlets package 6](#_Toc30262421)

[2.2 Model package 7](#_Toc30262422)

[2.3 Bean package 7](#_Toc30262423)

[2.4 Dao package 7](#_Toc30262424)

[2.5 Interfaces package 7](#_Toc30262425)

[2.6 WebContent package 7](#_Toc30262426)

[3. Interfacce delle classi 8](#_Toc30262427)

[Object Constrain Language 12](#_Toc30262428)

[SERVLETS 18](#_Toc30262429)

[4. Class Diagram 30](#_Toc30262430)

[5.Runtime Deployement 31](#_Toc30262431)

1. Introduzione

Dopo aver stilato il documento di Requirements Analysis e il documento di System Design, occorre concentrarsi sugli aspetti implementativi. L’Object Design Document (ODD) ha come obiettivo quello di produrre un modello che sia in grado di unire tutte le informazioni accumulate nelle fasi precedenti. In tale documento verranno definite le interfacce delle classi, le operazioni supportate, i tipi dei dati, i parametri delle procedure, i signatures dei sottosistemi definiti nel documento di System Design, i trade-offs e le linee guida, per evitare compromessi di progettazione.

* 1. Trade-offs

**Prestazioni VS Costi**

Considerando il sistema che da realizzare, possiamo dire che il non eccessivo budget a nostra disposizione ci consentirà di realizzare il prodotto utilizzando materiali reperibili in maniera gratuita, come un server open source e un ambiente di sviluppo gratuito, partendo da zero minimizzando così i costi e rendendo il sistema più che soddisfacente.

**Perfomance VS Usabilità**

L’interfaccia, grazie all’utilizzo delle form e di un cammino semplice e intuitivo, permette un uso facilitato del sistema per tutti i tipi di utenti all’interno della nostra piattaforma. E’ stata preferita l’usabilità per rendere l’esperienza d’utilizzo il più agevole possibile rendendo l’interfaccia chiara e pulita. Il sistema deve garantire tempi di risposta brevi per ogni funzionalità.

**Interfaccia VS Tempo di risposta**

Il tempo di risposta tra server e interfaccia sono rapidi per soddisfare le esigenze di tutti i tipi di utenti collegati al sistema, negli input degli utenti verrà fatto un controllo più meticoloso dei dati andando a specificare i parametri errati di caso di insuccesso.

* 1. Linee guida per la Documentazione delle Interfacce

Per l’implementazione del sistema saranno utilizzate le seguenti convenzioni.

1. **Naming Convention:**

È buona norma utilizzare nomi:

* Descrittivi
* Pronunciabili
* Di uso comune
* Di lunghezza medio-corta
* Non abbreviati
* Evitando la notazione ungherese
* Utilizzando solo caratteri consentiti (a-z, A-Z, 0-9)

1. **Variabili:**

* I nomi delle variabili devono iniziare con la lettera minuscola, e le parole successive con la lettera maiuscola. La dichiarazione delle variabili deve essere effettuata ad inizio blocco; in ogni riga vi deve essere una sola dichiarazione di variabile e va effettuato l’allineamento per migliorare la leggibilità.
* In determinati casi, è possibile utilizzare il carattere underscore “\_”, ad esempio quando si fa uso di variabili costanti oppure quando si fa uso di proprietà statiche.

1. **Metodi:**

* I nomi dei metodi devono iniziare con la lettera minuscola, e le parole successive con la lettera maiuscola. Di solito il nome del metodo è costituito da un verbo che identifica un’azione, seguito dal nome di un oggetto. I nomi dei metodi per l’accesso e la modifica delle variabili dovranno essere del tipo getNomeVariabile() e setNomeVariabile(). Se viene dichiarata una variabile all’interno di un metodo quest’ultima deve essere dichiarata appena prima del suo utilizzo e deve servire per un solo scopo, per facilitare la leggibilità. Esempio: getId(), setId()
* Ai metodi va aggiunta una descrizione, la quale deve essere posizionata prima della dichiarazione del metodo, e deve descriverne lo scopo. La descrizione del metodo deve includere anche informazioni riguardanti gli argomenti, il valore di ritorno, le eccezioni.

1. **Classi e pagine:**

* I nomi delle classi e delle pagine devono iniziare con la lettera maiuscola, e anche le parole successive all’interno del nome devono iniziare con la lettera maiuscola. I nomi delle classi e delle pagine devono essere evocativi, in modo da fornire informazioni sullo scopo di quest’ultime. Ogni file sorgente .java contiene una singola classe e dev’essere strutturato in un determinato modo:
* Una breve introduzione alla classe che ne specifichi lo scopo

/\*\*

**sommario dello scopo della classe.**

\*\*/

* L’istruzione import che permette di importare all’interno della classe gli altri oggetti che la classe utilizza.
* La dichiarazione di una classe è caratterizzata da:

1. Dichiarazione della classe pubblica
2. Dichiarazioni di costanti
3. Dichiarazioni di variabili di classe
4. Dichiarazioni di variabili d’istanza
5. Costruttore
6. Commento e dichiarazione metodi e variabili
   1. Definizioni, acronimi e abbreviazioni

**Acronimi:**

* **RAD**: Requirements Analysis Document
* **SDD**: System Design Document
* **ODD**: Object Design Document

**Abbreviazioni:**

* **DB**: DataBase
  1. Riferimenti
* B. Bruegge, A. H. Dutoit, Object Oriented Software Engineering - Using UML, Pattern and Java, Prentice Hall, 3rd edition, 2009
* Documento SDD Eat & Reorder
* Documento RAD Eat & Reorder
  1. Design Pattern

**Data Access Object (DAO)**

È un pattern architetturale che consente di separare i servizi della logica di business

dalle operazioni per la gestione dei dati persistenti. La funzionalità di questo pattern è

nascondere dall’applicazione tutte le complessità coinvolte nell’esecuzione dell’operazioni che interagiscono con la sorgente dei dati in modo da permettere ad entrambi i livelli di evolversi separatamente senza sapere nulla dell’altro.

Il **DAO** implementa il meccanismo di accesso richiesto per lavorare con la sorgente

dei dati. I moduli della logica di business utilizzano l’interfaccia esposta dal DAO, tale

interfaccia definisce le operazioni standard da eseguire. La classe DAO concreta che

implementerà l’interfaccia è responsabile dell’archivazione e restituzione dei dati. Per

archiviare i dati recuperati dalla classe DAO si utilizzano gli oggetti Bean che contengono imetodi get/set per memorizzare i dati recuperati.

Il design pattern sarà usato per la gestione della persistenza dei dati. L’oggetto del

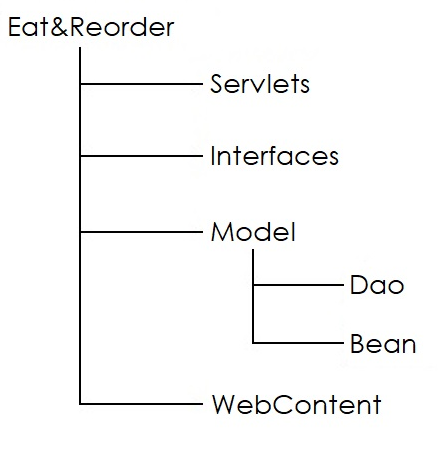
controller comunicherà con l’interfaccia DAO che verrà implementata dalle classi dao specifiche.

**Bridge Pattern.**

E’ un pattern strutturale che consente di separare una astrazione dalla sua implementazione tramite l’utilizzo di interfacce. Nel nostro sistema utilizzeremo tale pattern per implementare il gestore delle mail. Tale scelta si è resa necessaria per garantire la possibilità di modificare la gestione delle mail sia dal punto di vista di provider utilizzato, sia per la gestione della creazione grafica della mail stessa.

# Packages

La decomposizione dei sottosistemi in packages è la seguente:



Il **Package Servlets** contiene al suo interno tutte le servlet che rappresentano i servizi offerti dai sottosistemi Gestione Utente e Gestione Ordini individuati nel SDD;

Il **Package Model** contiene al suo interno tutti i bean e tutti i dao, i quali rappresentano i servizi offerti dai sottosistemi. Ovviamente, i bean e i dao sono raccolti in appositi package contenuti nel package model;

Il **Package Interfaces** contiene al suo interno tutte le interfacce che saranno utilizzate dai rispettivi dao e dal gestore mail.

Il **Package WebContent** contiene tutte le componenti che raffigurino la presentazione del sistema, quindi contiene HTML e JSP.

La suddivisione dei package in servlets, model e webcontent è motivata dall’utilizzo dell’architettura **MVC**, la quale permette di separare la logica di business (**servlets**) dalla presentazione (**webcontent**) e dalla gestione dei dati (**model**). Il package model è a sua volta suddiviso in bean e dao. Le servlet nel package servlets si occuperanno dell’elaborazione dei dati.

## Servlets package

Nel **package servlets** sono presenti tutte le servlet che implementano la logica di business e che tramite le interfacce DAO, contenute nell’apposito package, comunicheranno con le jsp per la gestione dei contenuti nei bean, presenti nell’apposito package.

È bene precisare che tutte le classi appartenenti a tale package dovranno iniziare il proprio nome con “Do”.

Di seguito vediamo tutte le classi contenute in tale package:

Sistema Gestione Utente:

* DoBannaAzienda.java
* DoRegistrazioneCliente.java
* DoRegistrazioneAzienda.java
* DoRegistrazioneFattorino.java
* DoLogin.java
* DoModificaProfiloCliente.java
* DoModificaProfiloAzienda.java
* DoModificaProfiloFattorino.java
* DoCreaSegnalazione.java
* DoLogout.java
* DoRicercaAzienda.java
* DoInserimentoProdotto.java
* DoVaiModificaProdotto.java
* DoModificaProdotto.java
* DoEliminaProdotto.java

Sistema Gestione Ordine:

* DoAggiungiAlCarrello.java
* DoVisualizzaListinoAzienda.java
* DoModificaQuantita.java
* DoRimuoviDalCarrello.java
* DoVisualizzaProdotto.java
* DoOrdinazione.java
* DoVisualizzaConsegne.java
* DoVaiVisualizzaOrdine.java
* DoSetRitirato.java
* DoSerConsegnato.java

## Model package

Nel **package model** sono presenti tutte le classi che rappresentino servizi per il sistema. Tale package, inoltre, è suddiviso a sua volta in bean e dao.

Di seguito vediamo tutte le classi contenute in tale package:

* Carrello.java
* CheckFormato.java
* DBConnectionPool.java
* GestoreMail.java
* ProdottoQuantita.java

## Bean package

Nel **package bean** sono presenti tutti gli oggetti entity che definiscono gli oggetti di dominio e i quali saranno utilizzati come bean dalle servlet presenti nel package servlets.

È bene precisare che tutte le classi appartenenti a tale package dovranno concludere il proprio nome con “\_Bean”.

Di seguito vediamo tutte le classi contenute in tale package:

Sistema Gestione Utente:

* AccountAzienda\_Bean.java
* AccountCliente\_Bean.java
* AccountFattorino\_Bean.java
* AccountModeratore\_Bean.java
* AccountUtenteRegistrato\_Bean.java

Sistema Gestione Ordine

* Ordine\_Bean.java
* Prodotto\_Bean.java

## Dao package

Nel **package dao** sono presenti tutte le classi che permettono di gestire il database.

Di seguito vediamo tutte le classi contenute in tale package:

* GestoreOrdineDaoImpl.java
* GestoreUtenteDaoImpl.java

## Interfaces package

Nel **package interfaces** sono presenti tutte le interfacce implementabili dai vari dao presenti nel package dao.

Di seguito vediamo tutte le classi contenute in tale package:

* GestoreMail\_Interface.java
* GestoreOrdineDao.java
* GestoreUtenteDao.java

## WebContent package

Nel **package webcontent** sono presenti JSP e sorgenti HTML che raffigurino la presentazione del sistema.

Di seguito vediamo le JSP ed i sorgenti HTML presenti in tale package:

Sistema Gestore Utente

* AziendaScelta.jsp
* DettagliProdotto.jsp
* Homepage.jsp
* HomepageAzienda.jsp
* HomepageCliente.jsp
* HomepageModeratore.jsp
* HomepageFattorino.jsp
* Listino.jsp
* Login.jsp
* ModificaProdotto.jsp
* ModificaProfiloAzienda.jsp
* ModificaProfiloCliente.jsp
* ModificaProfiloFattorino.jsp
* NuovoProdotto.jsp
* Registrazione.jsp
* RegistrazioneAzienda.jsp
* RegistrazioneCliente.jsp
* RegistrazioneFattorino.jsp
* Ricerca.jsp
* Segnalazione.jsp
* VisualizzaProfilo.jsp

Sistema Gestore Ordine

* Carrello.jsp
* Consegne.jsp
* DettagliOrdineAzienda.jsp
* DettagliOrdineCliente.jsp
* Ordinazione.jsp
* OrdiniAzienda.jsp

# Interfacce delle classi

**Di seguito saranno elencate le interfacce delle classi dao utilizzate nel sistema.**

|  |  |
| --- | --- |
| **GestoreUtenteDaoImpl** | |
| **Metodo** | **Descrizione** |
| public boolean controlloEsistenzaMail(String email) | Permette di controllare se l’email passata come stringa appartiene a qualche utente presente nel DB. |
| Public void registrazioneCliente(AccountCliente\_Bean cliente) | Permette la creazione di un account utente con le informazioni presenti nell’istanza cliente all’interno del DB |
| Public void registrazioneAzienda(AccountAzienda\_Bean azienda) | Permette la creazione di un account azienda con le informazioni presenti nell’istanza azienda all’interno del DB |
| Public void registrazioneFattorino(AccountFattorino\_Bean fattorino) | Permette la creazione di un account fattorino con le informazioni presenti nell’istanza fattorino all’interno del DB |
| Public boolean controllaBan(String email) | Permette di controllare se l’account associato all’email è bannato dal sistema |
| Public boolean controllaEsistenzaAccount(String user, String password) | Permette di controllare se esiste un account con l’user e la password forniti |
| Public AccountUtenteRegistrato\_Bean dammiUtente(String email) | Permette di recuperare l’account utente associato alla mail specificata |
| Public void aggiornaCliente(AccountCliente\_Bean cliente) | Permette di aggiornare le informazioni di un account cliente presente a DB |
| Public void aggiornaAzienda(AccountAzienda\_Bean azienda) | Permette di aggiornare le informazioni di un account azienda presente a DB |
| Public void aggiornaFattorino(AccountFattorino\_Bean cliente) | Permette di aggiornare le informazioni di un account fattorino presente a DB |
| Public void dammiAziendaConOrdine(Long idOrdine) | Permette di recuperare le informazioni di un account azienda associato all’ordine con l’id fornito |
| Public List<AccountAzienda\_Bean> dammiListaAziende(String citta) | Permette di recuperare la lista di aziende che hanno sede nella città indicata. |
| Public void banAzienda(AccountAzienda\_Bean azienda) | Permette di bannare l’account azienda indicato |
| Public void aggiungiAlListino(AccountAzienda\_Bean azienda, Prodotto\_Bean prodotto) | Permette di inserire il prodotto all’interno del profilo dell’azienda |
| Public void aggiornaProdotto(AccountAzienda\_Bean azienda, Prodotto\_Bean prodotto) | Permette di aggiornare un prodotto all’interno del profilo dell’azienda |
| Public void rimuoviProdotto(AccountAzienda\_Bean azienda, Prodotto\_Bean prodotto) | Permette di eliminare il prodotto dall’interno del profilo dell’azienda |

|  |  |
| --- | --- |
| **GestoreOrdineDaoImpl** | |
| **Metodo** | **Descrizione** |
| Public void creaOrdine(Ordine\_Bean ordine) | Permette di creare un ordine all’interno del DB |
| Public boolean controlloEsistenzaOrdine(Long idOrdine) | Permette di verificare l’esistenza di un ordine con il codice specificato |
| Public List<Ordine\_Bean> dammiOrdiniPreparazione(AccountAzienda\_Bean azienda) | Permette di ricavare la lista di ordini in preparazione dell’azienda fornita |
| Public Ordine\_Bean dammiOrdine(Long idOrdine) | Permette di ricavare l’ordine associato al codice fornito. |
| Public List<Ordine\_Bean> dammiConsegne(AccountFattorino\_Bean fattorino) | Permette di ricavare la lista di ordini associate al fattorino |
| Public void ordineSetRitirato(Long idOrdine) | Permette di settare lo stato dell’ordine con il codice fornito a “Ritirato” |
| Public void ordineSetConsegnato(Long idOrdine) | Permette di settare lo stato dell’ordine con il codice fornito a “Ritirato” |

|  |  |
| --- | --- |
| **GestoreMail** | |
| **Metodo** | **Descrizione** |
| Public void inviaMailModeratore(Long idOrder,String description) | Permette di inviare una mail di segnalazione al moderatore per l’ordine associato al codice e con la descrizione fornita |
| Public void inviaMailBan(AccountAzienda\_Bean azienda, String motivazione) | Permette di inviare una mail di ban all’azienda fornita e con la motivazione indicata |
| Public void inviaMailOrdine(Ordine\_Bean ordine) | Permette di inviare una mail di dettaglio al cliente per l’ordine fornito. |

**Di seguito sono specificate le interfacce delle classi utilizzate. Non sono indicate le interfacce dei metodi get e set per gli attributi delle classi e non saranno indicate le interfacce delle classi che presentano solo tali metodi.**

|  |  |
| --- | --- |
| **Carrello** | |
| **Metodo** | **Descrizione** |
| Public void aggiornaQtaCarrello(Prodotto\_Bean prod, int qta) | Permette di aggiornare la quantità di un prodotto presente nel carrello. |
| Public boolean isEmpty() | Permette di valutare se il carrello è vuoto |
| Public void rimuoviProdotto(Prodotto\_Bean prodotto) | Permette di rimuove il prodotto indicato dal carrello |
| Public AccountAzienda\_Bean getCurrentAzienda() | Permette di ricavare l’attuale azienda associata ai prodotti del carrello |
| Public void svuota() | Permette di svuotare il carrello |
| Public boolean checkAziendaCarrello(Prodotto\_Bean prod) | Permette di valutare se l’azienda associata al prodotto è uguale all’azienda corrente associata al carrello |
| Public boolean checkInCarrelo(Prodotto\_Bean prod) | Permette di valutare se il prodotto indicato è presente nel carrello |
| Public void aggiungiProdotto(Prodotto\_Bean prod) | Permette di aggiungere un prodotto al carrello |

|  |  |
| --- | --- |
| **AccountAzienda\_Bean** | |
| **Metodo** | **Descrizione** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
| **AccountUtente\_Bean** | |
| **Metodo** | **Descrizione** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
| **AccountFattorino\_Bean** | |
| **Metodo** | **Descrizione** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
| **Ordine\_Bean** | |
| **Metodo** | **Descrizione** |
| Public void setStato(String stato) | Permette di settare lo stato passato come parametro tra (In preparazione, Ritirato, Consegnato) |
| Public void getStato() | Permette di ricavare lo stato di un Ordine |

## Object Constrain Language

**In questa sezione verranno specificati i contratti per le classi del sistema.**

|  |  |
| --- | --- |
| Nome Classe | Carrello |
| Descrizione | Classe che gestisce il carrello |
| Pre-condizioni | **Context**::**Carrello** public aggiornaQtaCarrello(prod : Prodotto, qta : Int)  **pre**: prod !=null and qta > 0 and prodotti->include(p | p.getCodice() = prod.getCodice())  **Context**::**Carrello** public rimuoviProdotto(prod : Prodotto)  **pre**: prod !=null and prodotti->include(p | p.getCodice() = prod.getCodice())  **Context**::**Carrello** public svuota()  **pre**: prodotti->size()>0  **Context**::**Carrello** public checkAziendaCarrello(prod: Prodotto)  pre: prod != null  **Context**::**Carrello** public checkInCarrello(prod: Prodotto)  **pre**: prod != null  **Context**::**Carrello** public aggiungiProdotto(prod: Prodotto)  **pre**: prod!= null and prod.getAzienda() = self.getCurrentAzienda() |
| Post-condizioni | **Context**::**Carrello** public aggiornaQtaCarrello(prod : Prodotto, qta : Int)  **post**: self.prodotti->include(p| p.getCodice() = prod.getCodice() and p.getQta() = qta)  **Context**::**Carrello** public rimuoviProdotto(prod : Prodotto)  **post**: prodotti-> !include(p | p.getCodice() = prod.getCodice())  **Context**::**Carrello** public svuota()  **post**: prodotti->size() = 0 and self.CurrentAzienda() = null  **Context**::**Carrello** public checkAziendaCarrello(prod: Prodotto)  **post:** True if self.isEmpty() or prod.getAzienda().getEmail().equals(self.currentAzienda.getEmail())  **Context**::**Carrello** public checkInCarrello(prod: Prodotto)  **post:**  **Context**::**Carrello** public aggiungiProdotto(prod: Prodotto)  **post**: prodotti->include(p | p.getCodice() = prod.getCodice()) |
| Invarianti | **Context::Carrello** **inv** self.prodotti->forAll(p | p.getAzienda() = self.getCurrentAzienda()) |

|  |  |
| --- | --- |
| Nome Classe | AccountAzienda\_Bean |
| Descrizione | Classe che rappresenta l’account di una Azienda |
| Pre-condizioni | **Context**::**AccountAzienda** public dammiProdotto(id :Long)  **pre**: id != null and prodotti->include(p | p.getCodice() = id)  **Context**::AccountAzienda public aggiungiProdotto(prod : Prodotto)  **pre**: prod != null and prodotti->!include(p | p.getCodice() = prod.getCodice())  **Context**::AccountAzienda public rimuoviProdotto(prod : Prodotto)  **pre**: prod != null and prodotti->include(p | p.getCodice() = prod.getCodice())  **Context**::AccountAzienda public aggiornaProdotto(prod : Prodotto)  **pre**: prod != null and prodotti->include(p | p.getCodice() = prod.getCodice()) |
| Post-condizioni | **Context**::AccountAzienda public dammiProdotto(id :Long)  **post**:  **Context**::AccountAzienda public aggiungiProdotto(prod : Prodotto)  **post**: prodotti->include(p | p.getCodice() = prod.getCodice())  **Context::**AccountAzienda public rimuoviProdotto(prod : Prodotto)  **post**: prodotti->!include(p | p.getCodice() = prod.getCodice())  **Context**::AccountAzienda public aggiornaProdotto(prod : Prodotto)  **post**: prodotti->include(p | p.getCodice() = prod.getCodice() and p.getNome() = prod.getNome() and p.getImmagine() = prod.getImmagine() and p.getDescrizione() = prod.getDescrizione() and p.getPrezzo() = prod.getPrezzo()) |
| Invarianti |  |

|  |  |
| --- | --- |
| Nome Classe | Ordine\_Bean |
| Descrizione | Classe che rappresenta un ordine |
| Pre-condizioni | **Context**::Ordine public cambiaStato(stato :String)  **pre** : !self.stato.equals(“Consegnato”) |
| Post-condizioni | **Context**::Ordine public cambiaStato(stato :String)  **post** : if stato.equals(“Ritirato”) and @pre self.stato.equals(“In preparazione”) then self.stato = stato  if stato.equals(“Consegnato”) and @pre self.stato.equals(“Ritirato”) then self.stato = stato |
| Invarianti |  |

|  |  |
| --- | --- |
| Nome Classe | GestoreOrdineDAOImpl |
| Descrizione | Classe che implementa l’interfaccia GestoreOrdineDao per la gestione dei dati persistenti riguardanti l’ordine. |
| Pre-condizioni | **Context::GestoreOrdineDAOImpl** public controlloEsistenzaOrdine(long : idOrdine)  **pre** : idOrdine != null  **Context::GestoreOrdineDAOImpl** public creaOrdine(order : Ordine)  **pre** : order != null  **Context::GestoreOrdineDAOImpl** public dammiOrdiniPreparazione(azienda : AccountAzienda)  **pre** : azienda != null  **Context::GestoreOrdineDAOImpl** public  dammiOrdine(codice : Long)  **pre** : codice != null  **Context::GestoreOrdineDAOImpl** public dammiConsegne(fattorino : AccountFattorino)  **pre** : fattorino != null  **Context::GestoreOrdineDAOImpl** public ordineSetRitirato(orderCode : Long)  **pre** : order != null  **Context::GestoreOrdineDAOImpl** public ordineSetConsegnato(orderCode : Long)  **pre** : order != null |
| Post-condizioni | - |
| Invarianti | - |

|  |  |
| --- | --- |
| Nome Classe | GestoreMailImpl |
| Descrizione | Classe che implementa l’interfaccia GestoreMail per la gestione e l’invio di mail da parte del sistema |
| Pre-condizioni | **Context::GestoreMailImpl** public inviaMailModetatore(idOrder :Long, description : String)  pre: idOrder != null and description != null  **Context::GestoreMailImpl** public inviaMailBan(azienda :AccountAzienda, description : String)  pre: azienda != null and description != null  **Context::GestoreMailImpl** public inviaMailOrdine(order :Ordine)  pre: azienda != null and description != null |
| Post-condizioni | - |
| Invarianti | - |

|  |  |
| --- | --- |
| Nome Classe | GestoreUtenteDaoImpl |
| Descrizione | Classe che implementa l’interfaccia GestoreUtenteDao per la gestione dai dati degli utenti |
| Pre-condizioni | **Context::GestoreUtenteDaoImpl** public controlloEsistenzaMail(email: String)  pre: email != null  **Context::GestoreUtenteDaoImpl** public registrazioneCliente(cliente: AccountCliente)  pre: cliente != null  **Context::GestoreUtenteDaoImpl** public registrazioneAzienda(azienda: AccountAzienda)  pre: azienda != null  **Context::GestoreUtenteDaoImpl** public registrazioneFattorino(fatt: AccountFattorino)  pre: fatt != null  **Context::GestoreUtenteDaoImpl** public controllaBan(email: String)  pre: email != null  **Context::GestoreUtenteDaoImpl** public controllaEsistenzaAccount(user: String, pass: String)  pre: user != null and pass != null  **Context::GestoreUtenteDaoImpl** public dammiUtente(email: String)  pre: email != null  **Context::GestoreUtenteDaoImpl** public aggiornaCliente(cliente: AccountCliente)  pre: cliente != null  **Context::GestoreUtenteDaoImpl** public aggiornaAzienda(azienda: AccountAzienda)  pre: azienda != null  **Context::GestoreUtenteDaoImpl** public aggiornaFattorino(fatt: AccountFattorino)  pre: fatt != null  **Context::GestoreUtenteDaoImpl** public dammiAziendaConOrdine(ordine: Ordine)  pre: ordine != null  **Context::GestoreUtenteDaoImpl** public banAzienda(azienda: AccountAzienda)  pre: azienda != null  **Context::GestoreUtenteDaoImpl** public dammiListaAziende(citta: String)  pre: citta != null  **Context::GestoreUtenteDaoImpl** public aggiungiAlListino(azienda: AccountAzienda, prod : Prodotto)  pre: azienda != null and prod != null  **Context::GestoreUtenteDaoImpl** public aggiornaProdotto(azienda: AccountAzienda, prod : Prodotto)  pre: azienda != null and prod != null  **Context::GestoreUtenteDaoImpl** public rimuoviProdotto(azienda: AccountAzienda, prod : Prodotto)  pre: azienda != null and prod != null |
| Post-condizioni | - |
| Invarianti | - |

## SERVLETS

**Di seguito i contratti per le classi servlets.**

|  |  |
| --- | --- |
| Nome Classe | DoRegistrazioneCliente |
| Descrizione | Servlet per la registrazione del Cliente |
| Pre-condizioni | **Context::DoRegistrazioneCliente** public doGet(request: HttpServletRequest, response: HttpServletResponse)  **pre**: request!=null and response!=null and request.getSession().getAttribute(“utente”) = null and request.getParameter("email")!=null and request.getParameter("password")!=null and request.getParameter("nome")!=null and request.getParameter("cognome")!=null |
| Post-condizioni | - |
| Invarianti | - |

|  |  |
| --- | --- |
| Nome Classe | DoRegistrazioneAzienda |
| Descrizione | Servlet per la registrazione dell’ Azienda |
| Pre-condizioni | **Context::DoRegistrazioneAzienda** public doGet(request: HttpServletRequest, response: HttpServletResponse)  pre: request!=null && response!=null and  request.getSession().getAttribute(“utente”) = null and request.getParameter("email")!=null and request.getParameter("password")!=null and request.getParameter("nome")!=null and request.getParameter("telefono")!=null and request.getParameter("indirizzo")!=null and request.getParameter("civico")!=null and request.getParameter("citta")!=null and request.getParameter("provincia")!=null and request.getParameter("iva")!=null and request.getParameter("start-time")!=null and request.getParameter("end-time")!=null |
| Post-condizioni | - |
| Invarianti | - |

|  |  |
| --- | --- |
| Nome Classe | DoRegistrazioneFattorino |
| Descrizione | Servlet che effettua la registrazione del Fattorino |
| Pre-condizioni | **Context::DoRegistrazioneFattorino** public doGet(request: HttpServletRequest, response: HttpServletResponse)  pre: request!=null && response!=null and request.getSession().getAttribute(“utente”) = null request.getParameter("email")!=null and request.getParameter("password")!=null and request.getParameter("nome")!=null and request.getParameter("cognome")!=null and request.getParameter("telefono ")!=null and request.getParameter("citta")!=null and request.getParameter("provincia")!=null and request.getParameter("start-time")!=null and request.getParameter("end-time")!=null |
| Post-condizioni | - |
| Invarianti | - |

|  |  |
| --- | --- |
| Nome Classe | DoLogin |
| Descrizione | Servlet che effettua il login |
| Pre-condizioni | **Context::DoLogin** public doPost(request: HttpServletRequest, response: HttpServletResponse)  pre: request!=null && response!=null && request.getParameter("email")!=null && request.getParameter("password")!=null && |
| Post-condizioni | - |
| Invarianti | - |

|  |  |
| --- | --- |
| Nome Classe | DoModificaProfiloCliente |
| Descrizione | Servlet che modifica il profilo del Cliente |
| Pre-condizioni | **Context::DoModificaProfiloCliente** public doGet(request: HttpServletRequest, response: HttpServletResponse)  pre: request!=null && response!=null and request.getSession().getAttribute(“utente”) != null and request.getParameter("password")!=null and request.getParameter("nome")!=null and request.getParameter("cognome")!=null |
| Post-condizioni | - |
| Invarianti | - |

|  |  |
| --- | --- |
| Nome Classe | DoModificaProfiloAzienda |
| Descrizione | Servlet che modifica il profilo dell’Azienda |
| Pre-condizioni | **Context::DoModificaProfiloAzienda** public doGet(request: HttpServletRequest, response: HttpServletResponse)  pre: request!=null && response!=null and request.getSession().getAttribute("utente")!=null and request.getParameter("password")!=null and request.getParameter("nome")!=null and request.getParameter("telefono")!=null and request.getParameter("indirizzo")!=null and request.getParameter("civico")!=null and request.getParameter("citta")!=null and request.getParameter("provincia")!=null and request.getParameter("iva")!=null and request.getParameter("start-time")!=null and request.getParameter("end-time")!=null |
| Post-condizioni | - |
| Invarianti | - |

|  |  |
| --- | --- |
| Nome Classe | DoModificaProfiloFattorino |
| Descrizione | Servlet che modifica il profilo del Fattorino |
| Pre-condizioni | **Context::DoModificaProfiloFattorino** public doGet(request: HttpServletRequest, response: HttpServletResponse)  pre: request!=null && response!=null and request.getSession().getAttribute("utente")!=null and request.getParameter("password")!=null and request.getParameter("nome")!=null and request.getParameter("cognome")!=null and request.getParameter("telefono ")!=null and request.getParameter("citta")!=null and request.getParameter("provincia")!=null and request.getParameter("start-time")!=null and request.getParameter("end-time")!=null |
| Post-condizioni | - |
| Invarianti | - |

|  |  |
| --- | --- |
| Nome Classe | DoCreaSegnalazione |
| Descrizione | Classe che crea la segnalazione di un cliente al moderatore |
| Pre-condizioni | **Context::DoCreaSegnalazione** public doGet (request: HttpRequest, response: HttpResponse)  Pre: request != null and response != null and request.getSession().getAttribute(“utente”) != null and request.getParameter(“id-order”) != null and request.getParameter(“reason”) != null |
| Post-condizioni | - |
| Invarianti | - |

|  |  |
| --- | --- |
| Nome Classe | DoAggiungiAlCarrello |
| Descrizione | Servlet che gestisce l’inserimento di un prodotto nel carrello |
| Pre-condizioni | **Context::DoAggiungiAlCarrello** public doGet (request : HttpRequest, response : HttpResponse)  pre: request != null and response != null and request.getSession().getParameter(“utente”) != null and request.getParameter(“id”) != null and request.getParameter(“azienda”) != null |
| Post-condizioni | - |
| Invarianti | - |

|  |  |
| --- | --- |
| Nome Classe | DoBannaAzienda |
| Descrizione | Servlet che consente di bannare una azienda |
| Pre-condizioni | **Context::DoBannaAzienda** public doGet (request : HttpRequest, response : HttpResponse)  pre: request != null and response != null and request.getSession().getParameter(“utente”) != null and request.getParameter(“id”) != null and request.getParameter(“azienda”) != null |
| Post-condizioni | - |
| Invarianti | - |

|  |  |
| --- | --- |
| Nome Classe | DoLogout |
| Descrizione | Servlet che effettua il logout |
| Pre-condizioni | **Context::DoLogout** public doPost(request: HttpServletRequest, response: HttpServletResponse)  pre: request!=null and response!=null |
| Post-condizioni | - |
| Invarianti | - |

|  |  |
| --- | --- |
| Nome Classe | DoRicercaAzienda |
| Descrizione | Servlet che consente di cercara le aziende di una città |
| Pre-condizioni | **Context::DoRicercaAzienda** public doGet(request: HttpServletRequest, response: HttpServletResponse)  pre: request!=null and response!=null ad request.getSession().getAttribute(“utente”) != null and request.getParameter(“ricerca”) != null |
| Post-condizioni | - |
| Invarianti | - |

|  |  |
| --- | --- |
| Nome Classe | DoInserimentoProdotto |
| Descrizione | Classe che inserisce un nuovo prodotto all’interno del listino dell’Azienda |
| Pre-condizioni | **Context::DoInserimentoProdotto** public doGet(request: HttpServletRequest, response: HttpServletResponse)  pre: request!=null and response!=null and request.getSession().getAttribute(“utente”)!=null and request.getParameter("nome")!=null and request.getParameter("costo")!=null and request.getParameter("img\_path")!=null and request.getParameter("descrizione")!=null |
| Post-condizioni | - |
| Invarianti | - |

|  |  |
| --- | --- |
| Nome Classe | DoVaiModificaProdotto |
| Descrizione | Servlet che consente di visualizzare la pagina di modifica un prodotto all’interno del listino dell’Azienda |
| Pre-condizioni | **Context::DoVaiModificaProdotto** public doPost(request: HttpServletRequest, response: HttpServletResponse)  pre: request!=null && response!=null && request.getSession().getAttribute(“utente”)!=null && request.getParameter("idProdotto")!=null |
| Post-condizioni | - |
| Invarianti | - |

|  |  |
| --- | --- |
| Nome Classe | DoModificaProdotto |
| Descrizione | Servlet che modifica un prodotto all’interno del listino dell’Azienda |
| Pre-condizioni | **Context::DoModificaProdotto** public doPost(request: HttpServletRequest, response: HttpServletResponse)  pre: request!=null && response!=null && request.getSession().getAttribute(“utente”)!=null && request.getParameter("id")!=null && request.getParameter("nome")!=null && request.getParameter("costo")!=null && request.getParameter("img\_path")!=null && request.getParameter("descrizione")!=null |
| Post-condizioni | - |
| Invarianti | - |

|  |  |
| --- | --- |
| Nome Classe | DoEliminaProdotto |
| Descrizione | Servlet che elimina un prodotto dall’interno del listino dell’Azienda |
| Pre-condizioni | **Context::DoEliminaProdotto** public doPost(request: HttpServletRequest, response: HttpServletResponse)  pre: request!=null && response!=null && request.getSession().getAttribute(“utente”)!=null && request.getParameter("idProdotto")!=null |
| Post-condizioni | - |
| Invarianti | - |

|  |  |
| --- | --- |
| Nome Classe | DoVisualizzaListinoAzienda |
| Descrizione | Servlet che consente di visualizzare il listino di una azienda |
| Pre-condizioni | **Context::DoVisualizzaListinoAzienda** public doGet(request: HttpServletRequest, response: HttpServletResponse)  pre: request!=null && response!=null && request.getSession().getAttribute(“utente”)!=null && request.getParameter("id")!=null |
| Post-condizioni | - |
| Invarianti | - |

|  |  |
| --- | --- |
| Nome Classe | DoModificaQuantita |
| Descrizione | Servlet che consente di modificare la quantità di un prodotto nel carrello |
| Pre-condizioni | **Context::DoModificaQuantita** public doGet(request: HttpServletRequest, response: HttpServletResponse)  pre: request!=null && response!=null and request.getSession().getAttribute(“utente”)!=null and request.getParameter("productId")!=null and request.getParameter(“productQuantity”) != null and request.getParameter(“companyMail”) != null |
| Post-condizioni | - |
| Invarianti | - |

|  |  |
| --- | --- |
| Nome Classe | DoRimuoviDalCarrello |
| Descrizione | Servlet che consente di eliminare un prodotto dal Carrello |
| Pre-condizioni | **Context::DoRimuoviDalCarrello** public doGet(request: HttpServletRequest, response: HttpServletResponse)  pre: request!=null && response!=null and request.getSession().getAttribute(“utente”)!=null and request.getParameter("Id")!=null |
| Post-condizioni | - |
| Invarianti | - |

|  |  |
| --- | --- |
| Nome Classe | DoVisualizzaProdotto |
| Descrizione | Servlet che consente di visualizzare i dettagli di un prodotto di una azienda |
| Pre-condizioni | **Context::DoVisualizzaProdotto** public doGet(request: HttpServletRequest, response: HttpServletResponse)  pre: request!=null && response!=null and request.getSession().getAttribute(“utente”)!=null and request.getParameter("azienda")!=null and request.getParameter(“prod”) != null |
| Post-condizioni | - |
| Invarianti | - |

|  |  |
| --- | --- |
| Nome Classe | DoOrdinazione |
| Descrizione | Servlet che consente di Creare un ordine |
| Pre-condizioni | **Context::DoOrdinazione** public doGet(request: HttpServletRequest, response: HttpServletResponse)  pre: request!=null && response!=null and request.getSession().getAttribute(“utente”)!=null and request.getParameter("address")!=null and request.getParameter(“notes”) != null and request.getParameter(“credit-card”) != null and request.getSession.getAttribute(“carrello”) != null and !request.getSession.getAttribute(“carrello”).isEmpty() |
| Post-condizioni | **Post:**  request.getSession.getAttribute(“carrello”).isEmpty() = True |
| Invarianti | - |

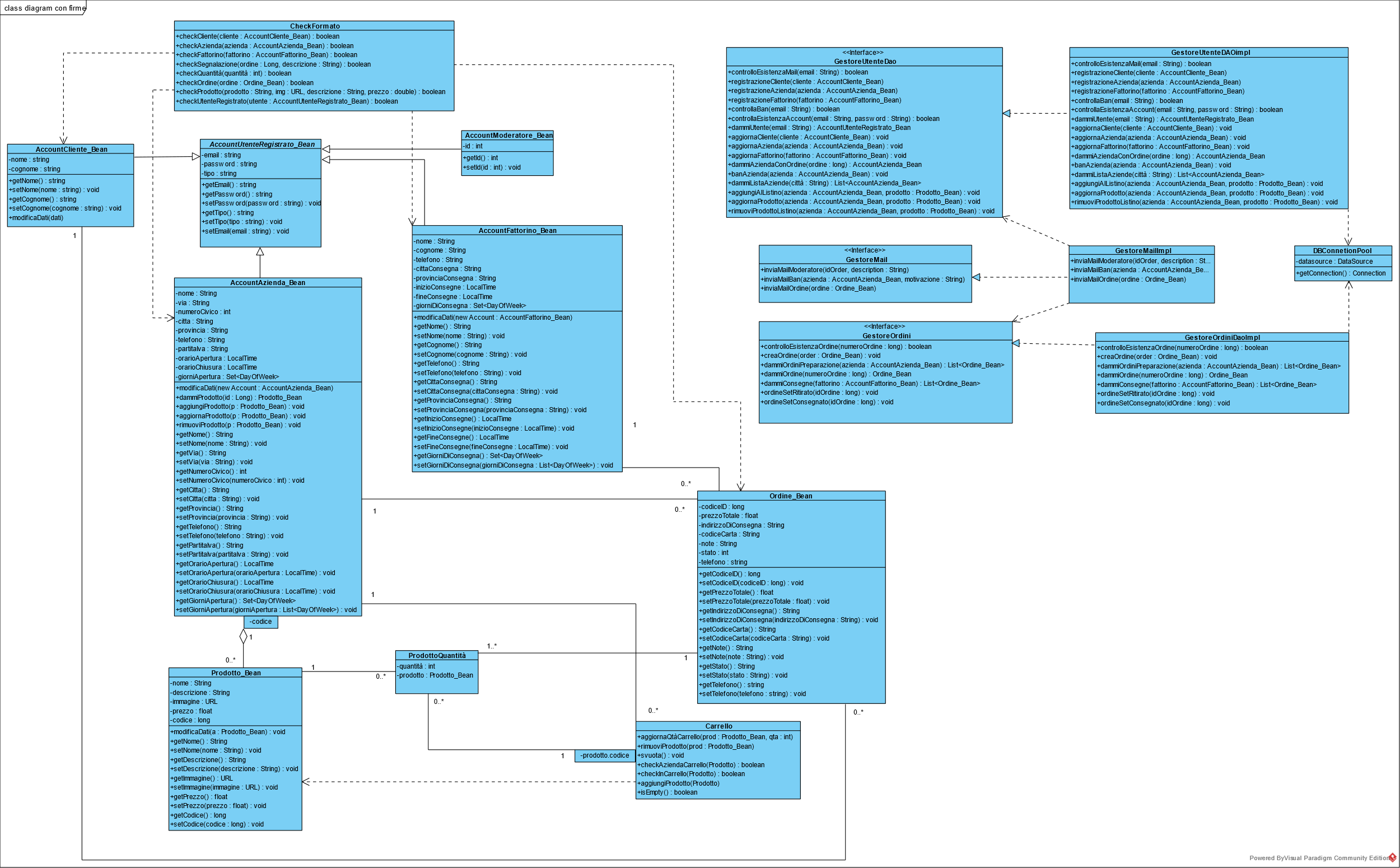
|  |  |
| --- | --- |
| Nome Classe | DoVaiVisualizzaOrdine |
| Descrizione | Servlet che consente all’azienda di visualizzare un ordine in preparazione |
| Pre-condizioni | **Context::DoVaiVisualizzaOrdine** public doGet (request: HttpRequest, response : HttpResponse)  pre: request !=null && response!=null and request.getSession().getAttribute(“utente”)!=null and request.getParameter(“idOrdine”) != null |
| Post-condizioni | - |
| Invarianti | - |

|  |  |
| --- | --- |
| Nome Classe | DoSetConsegnato |
| Descrizione | Servlet che consente di settare lo stato di un Ordine a Consegnato |
| Pre-condizioni | **Context::DoSetConsegnato** public doGet (request: HttpRequest, response : HttpResponse)  pre: request !=null && response!=null and request.getSession().getAttribute(“utente”)!=null and request.getParameter(“id”) != null |
| Post-condizioni | - |
| Invarianti | - |

|  |  |
| --- | --- |
| Nome Classe | DoVisualizzaConsegne |
| Descrizione | Servlet che consente di visualizzare le consegne da effettuare di un fattorino |
| Pre-condizioni | **Context::DoVisualizzaConsegne** public doGet (request: HttpRequest, response : HttpResponse)  pre: request !=null && response!=null and request.getSession().getAttribute(“utente”)!=null |
| Post-condizioni | - |
| Invarianti | - |

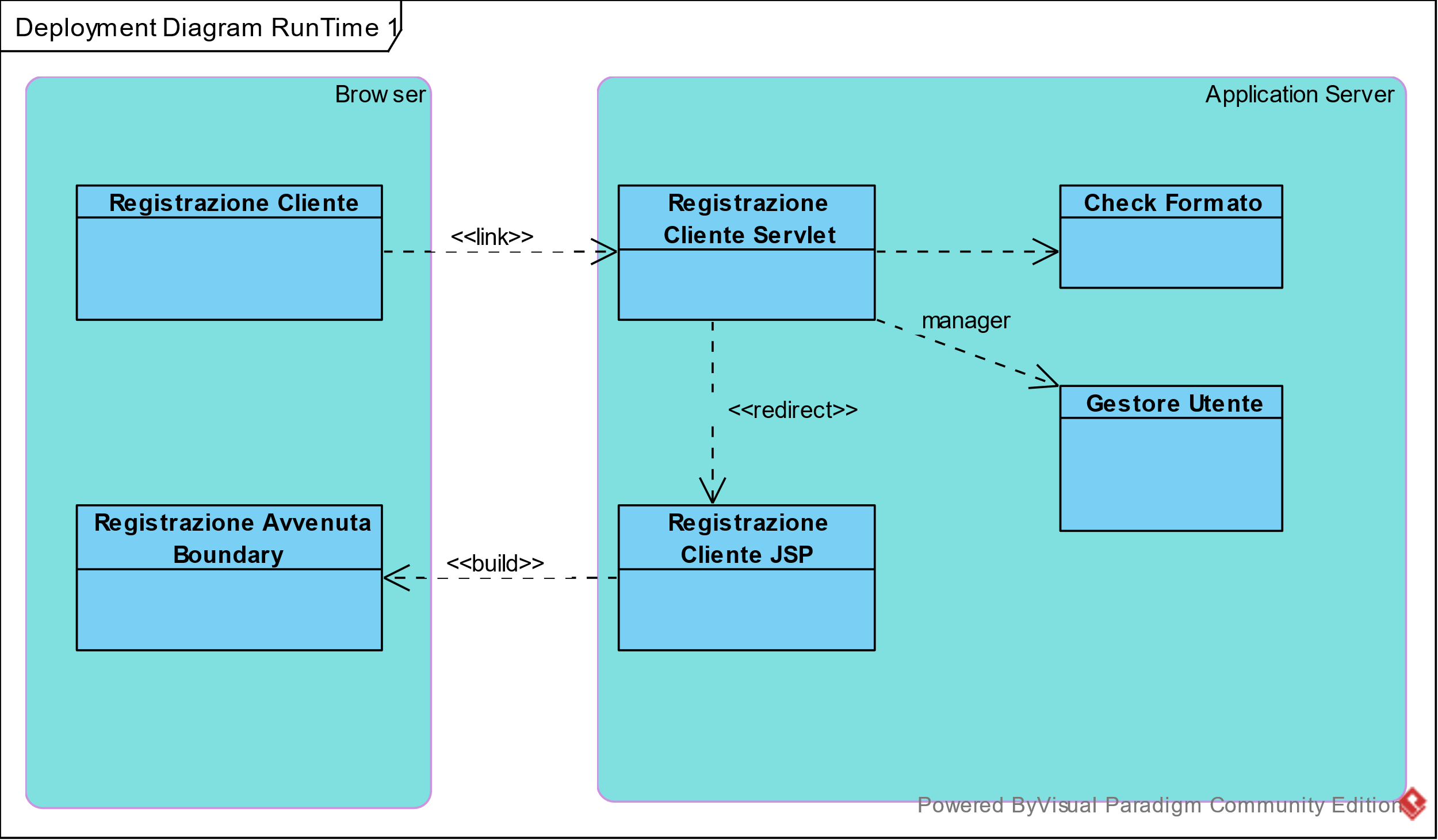
|  |  |
| --- | --- |
| Nome Classe | DoSetRitirato |
| Descrizione | Servlet che consente di settare lo stato di un Ordine a Ritirato |
| Pre-condizioni | **Context::DoSetRitirato** public doGet (request: HttpRequest, response : HttpResponse)  pre: request !=null && response!=null and request.getSession().getAttribute(“utente”)!=null and request.getParameter(“id”) != null |
| Post-condizioni | - |
| Invarianti | - |

# Class Diagram



# 5.Runtime Deployement

Il sistema Eat & Reorder verrà implementato utilizzando un web server che realizza le funzionalità tramite l’utilizzo di tecnologie Servlet e JSP (Java server page). Le servlet sono utilizzate per gestire la logica di controllo mentre le JSP verranno utilizzate per la pura visualizzazione formattata dei dati così come specificato dall’utilizzo del pattern MVC(Model View Controller). Nei seguenti diagrammi viene mostrato il deployement dei componenti sulle varie macchine del sistema.



Il browser visualizza la pagina HTML all’interno della quale possono essere presenti form con dati da inserire. Alla conferma, tali dati sono passati tramite una richiesta http al web server che manda tale richiesta alla servlet designata. Tale servlet, gestisce i dati utilizzando i Gestori (che si occupano dell’accesso ai dati del DB). Terminata l’elaborazione, la servlet rimanda i dati elaborati ad una JSP anch’essa eseguita lato server che si occuperà della creazione del codice HTML contenti le informazioni da visualizzare lato Browser.

Tale schema riassume l’utilizzo dei vari componenti all’interno del sistema e di dove essi vengono eseguiti.