Università degli Studi di Salerno
Corso di Ingegneria del Software

Enoteca
Il Gocciolatoio
Bevi poco... Ma bevi bene!

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
08/01/2021	0.3	Rettifica Class Interface e Package	Francesco Di Palma Giovanni Di Mauro Maria Giuseppina Mosca
09/01/2021	1.0	Modifiche e pubblicazioni documento	Francesco Di Palma Giovanni Di Mauro Maria Giuseppina Mosca

SOMMARIO

1.0 INTRODUZIONE

1.1 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.

1.2 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 (Camel Case). 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

1.3 DEFINIZIONI, ACRONIMI E ABBREVIAZIONI

RAD = Requirement Analysis Document

SDD = System Design Document

ODD = Object Design Document

1.4 RIFERIMENTI

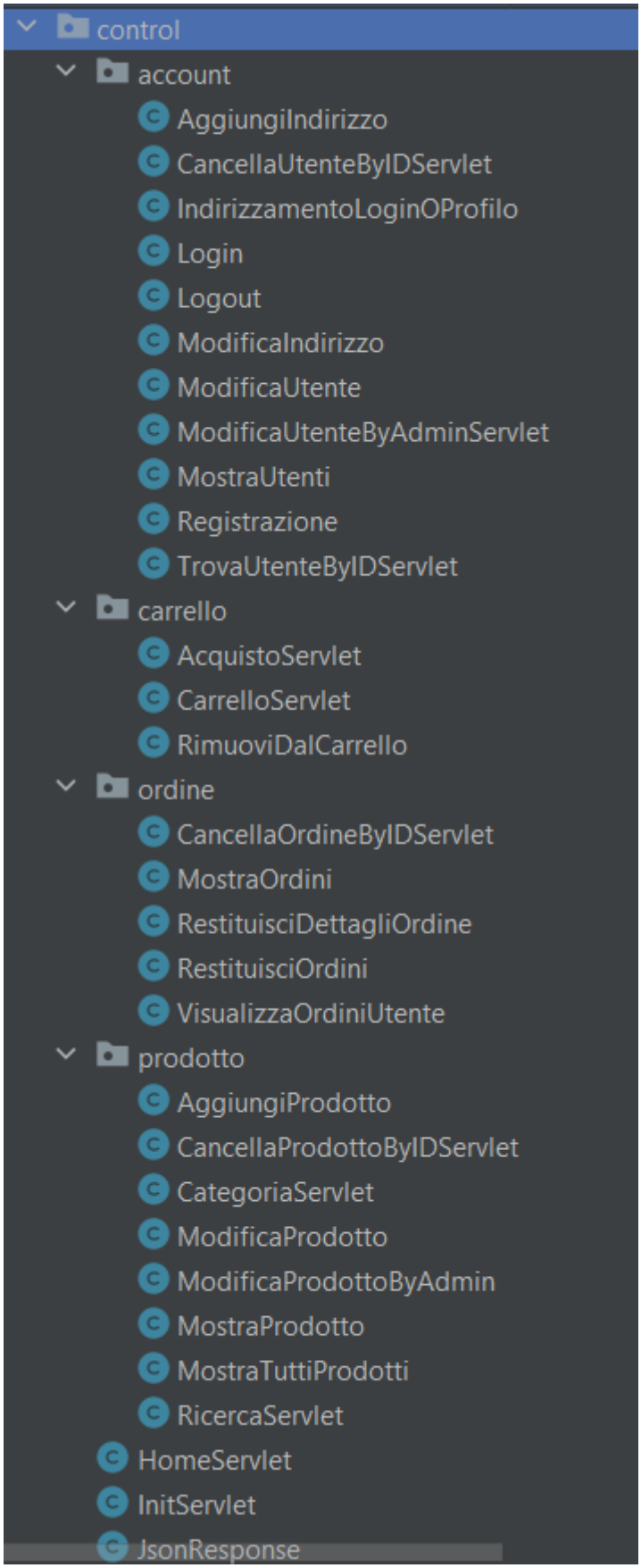
Riferimenti a RAD_IlGocciolatoio e a SDD_IlGocciolatoio

2.0 PACKAGES

Abbiamo diviso il sistema in:

- Control
- Dao
- Model
- View

2.1 control


control
account
AggiungiIndirizzo
CancellaUtenteByIDServlet
IndirizzamentoLoginOProfilo
Login
Logout
ModificaIndirizzo
ModificaUtente
ModificaUtenteByAdminServlet
MostraUtenti
Registrazione
TrovaUtenteByIDServlet
carrello
AcquistoServlet
CarrelloServlet
RimuoviDalCarrello
ordine
CancellaOrdineByIDServlet
MostraOrdini
RestituisciDettagliOrdine
RestituisciOrdini
VisualizzaOrdiniUtente
prodotto
AggiungiProdotto
CancellaProdottoByIDServlet
CategoriaServlet
ModificaProdotto
ModificaProdottoByAdmin
MostraProdotto
MostraTuttiProdotti
RicercaServlet
HomeServlet
InitServlet
JsonResponse

Account	
Classe	**Descrizione**
AggiungiIndirizzo	Servlet che gestisce l'aggiunta di un indirizzo
CancellaUtenteByIDServlet	Servlet che gestisce la rimozione di un utente tramite "id"
IndirizzamentoLoginOProfilo	Servlet che gestisce il reidirizzamento del profilo
Login	Servlet che gestisce il login al sito
Logout	Servlet che gestisce il logout dal sito
ModificaIndirizzo	Servlet che gestisce la modifica di un indirizzo
ModificaUtente	Servlet che gestisce la modifica di un utente
ModificaUtenteByAdminServlet	Servlet che gestisce la modifica di un utente da parte di un Admin
Registrazione	Servlet che gestisce la registrazione al sito
TrovaUtenteByIDServlet	Servlet che gestisce la ricerca di un utente tramite id

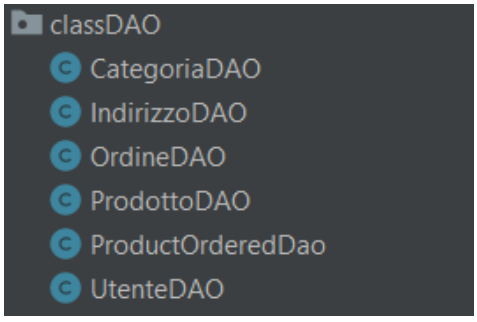
Carrello	
Classe	**Descrizione**
AcquistoServlet	Servlet che gestisce l'acquisto
CarrelloServlet	Servlet che gestisce il carrello
RimuoviDalCarrello	Servlet che gestisce la rimozione di un elemento dal carrello

Ordine	
Classe	**Descrizione**
CancellaOrdineByIDServlet	Servlet che gestisce la rimozione di un ordine tramite "id"
MostraOrdini	Servlet che gestisce la visualizzazione degli ordini
RestituisciDettagliOrdini	Servlet che gestisce la visualizzazione dei dettagli di un ordine
RestituisciOrdini	Servlet che gestisce la visualizzazione della lista degli ordini
VisualizzaOrdiniUtente	Servlet che gestisce la visualizzazione di tutti gli ordini di un utente

Prodotto	
Classe	**Descrizione**
AggiungiProdotto	Servlet che gestisce l'aggiunta di un prodotto
CancellaProdottoByIDServlet	Servlet che gestisce la rimozione di un prodotto tramite "id"
CategoriaServlet	Servlet che gestisce le categorie dei prodotti
ModificaProdotto	Servlet che gestisce la modifica di un Prodotto
ModificaProdottoByAdmin	Servlet che gestisce la lista di Prodotti da modificare
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
RicercaServlet	Servlet che gestisce la ricerca di un prodotto nella barra di ricerca

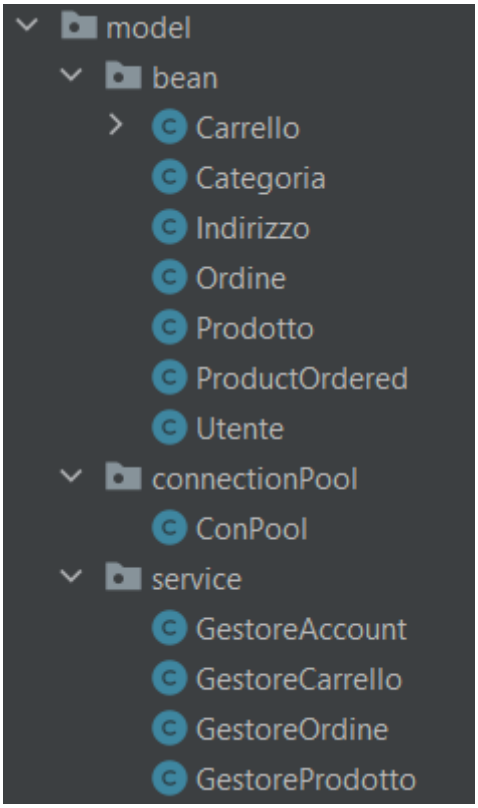
Control	
Classe	**Descrizione**
HomeServlet	Servlet che gestisce i prodotti destinati nella "HomePage"
InitServlet	Servlet che gestisce l'avvio della web application
JsonResponse	Servlet che gestisce i suggerimenti di ricerca

2.2 Classi DAO



ClassDao	
Classe	**Descrizione**
CategoriaDao	Descrive l'interazione con il database per "Categoria"
IndirizzoDao	Descrive l'interazione con il database per "Indirizzo"
OrdineDAO	Descrive l'interazione con il database per "Ordine"
ProdottoDao	Descrive l'interazione con il database per "Prodotto"
ProductOrderedDao	Descrive l'interazione con il database per "ProdottoOrdinato"
UtenteDao	Descrive l'interazione con il database per "Utente"

2.3 Model: Bean, ConnectionPool

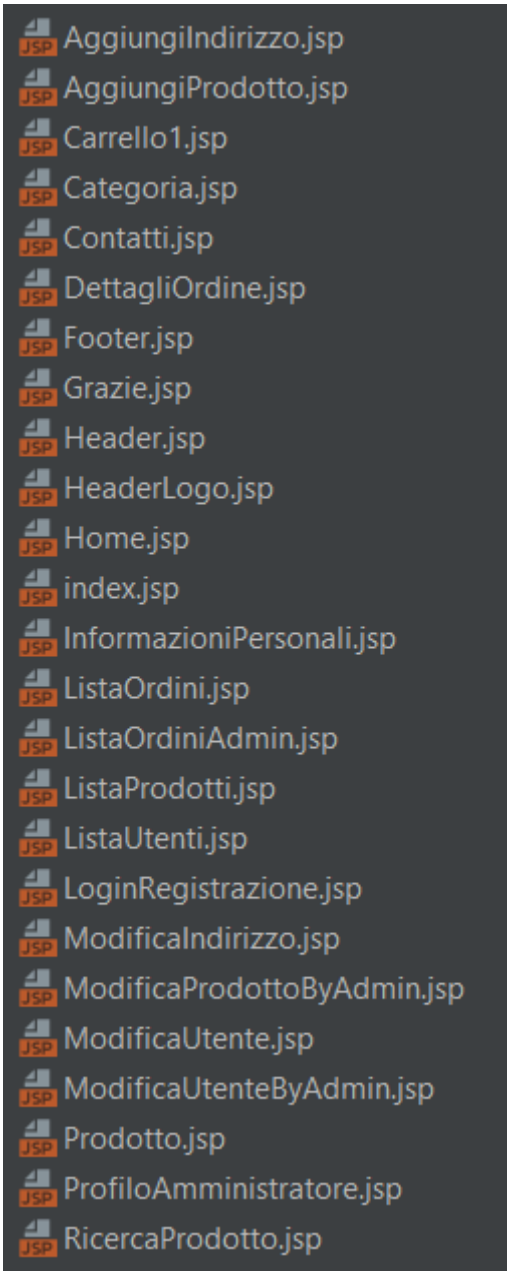


Bean	
Classe	**Descrizione**
Carrello	Descrive un carrello
Categoria	Descrive una Categoria di un prodotto
Indirizzo	Descrive un indirizzo
Ordine	Descrive un ordine
Prodotto	Descrive un Prodotto
ProductOrdered	Descrive un Prodotto Ordinato da un Utente
Utente	Descrive un Utente

Service	
Classe	**Descrizione**
GestoreAccount	Contiene la logica di business relativa all'account
GestoreCarrello	Contiene la logica di business relativa al carrello
GestoreLogin	Contiene la logica di business relativa al login
GestoreOrdine	Contiene la logica di business relativa all'ordine
GestoreProdotto	Contiene la logica di business relativa al prodotto

connectionPool	
Classe	**Descrizione**
ConPool	Classe che implementa l'object Pool Pattern, responsabile di fornire connessione con database.

2.4 View

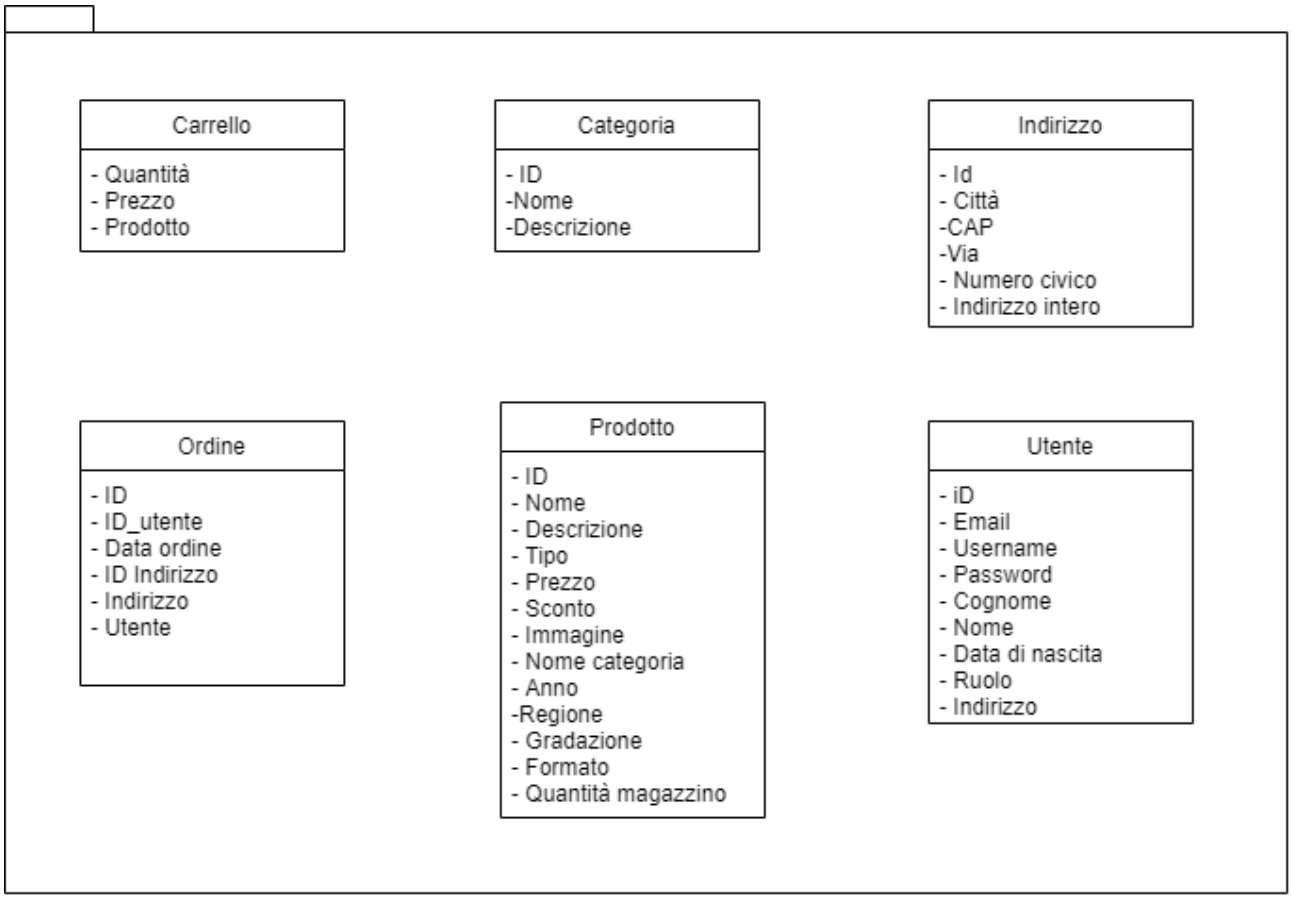


View	Descrizione
AggiungiIndirizzo.jsp	Pagina che mostra un form per aggiungere un indirizzo
AggiungiProdotto.jsp	Pagina che mostra un form per aggiungere un prodotto
Carrello1.jsp	Pagina che mostra il Carrello
Categoria.jsp	Pagina che mostra la categoria selezionata

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
Header.jsp	Header principale del sito
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 di un utente
ListaOrdiniAdmin.jsp	Pagina che mostra la lista di tutti gli 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
ModificaIndirizzo.jsp	Pagina che permette di modificare un indirizzo selezionato
ModificaProdottoByAdmin.jsp	Pagina che mostra la lista di prodotti che si vuole modificare
ModificaUtente.jsp	Pagina che permette la modifica dei dati di un utente
ModificaUtenteByAdmin.jsp	Pagina che permette la modifica di uno specifico utente
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



Nome classe	Carrello
Descrizione	Questa classe rappresenta l'oggetto "Carrello"
Signature dei metodi	+ getQuantita(): int + setQuantita(quantita: int): void + getProdotto(): Prodotto + getPrezzoTotCent(): double + getProdotti(): Collection<ProdottoQuantita> + get(int prodId): ProdottoQuantita + put(Prodotto p, int quantita): void + remove(int prodId): ProdottoQuantita + getPrezzoTotCent(): double
Pre-condizioni	
Post-condizioni	
Invariante	

Nome classe	Categoria
Descrizione	Questa classe rappresenta l'oggetto "Categoria"
Signature dei metodi	+ getId(): int + setId(int id): void + getNome(): String + setNome(String nome): void + getDescrizione(): String + setDescrizione(String descrizione): void
Pre-condizioni	
Post-condizioni	
Invariante	

Nome classe	Indirizzo
Descrizione	Questa classe rappresenta l'oggetto "Indirizzo"
Signature dei metodi	+ getId(): int + setId(int id): void +getCitta():String +setCitta(String citta):void +getCap():int +setCap(int cap):void +getVia():String +setVia(String via):void +getNumCivico():int +getNazione():String +getFullIndirizzo():String
Pre-condizioni	
Post-condizioni	
Invariante	

Nome classe	Ordine
Descrizione	Questa classe rappresenta l'oggetto Ordine
Signature dei metodi	+ getId(): int + setId(int id): void + getId_utente(): int + setId_utente(int id_utente): void +getData_Ordine():Date +setData_Ordine(Date data):void +getIdIndirizzo():int +setIdIndirizzo(int indirizzo):void +getIndirizzo():Indirizzo +setIndirizzo(Indirizzo indirizzo):void +getUtente():Utente +setutente(Utente utente):void
Pre-condizioni	
Post-condizioni	
Invariante	

Nome classe	Prodotto
Descrizione	Questa classe rappresenta l'oggetto Prodotto
Signature dei metodi	+ getId(): int + setId(int id): void + getNome(): String + setNome(String nome): void + getDescrizione(): String + setDescrizione(String descrizione): void + getTipo(): String + setTipo(String tipo): void + getPrezzo(): double + setPrezzo(double prezzo): void + getSconto(): double + setSconto(double sconto): void + getImmagine(): String + setImmagine(String immagine): void + getNome_categoria():String + setNome_categoria(nome_categoria): void + getAnno(): int

	+ setAnno(int anno): void + getRegione(): String + setRegione(String regione): void + getGradazione(): int + setGradazione(int gradazione): void + getFormato(): int + setFormato(int formato): void + getQuantita_magazzino(): int + setQuantita_magazzino(int quantita_magazzino): void
Pre-condizioni	
Post-condizioni	
Invariante	

Nome classe	Utente
Descrizione	Questa classe rappresenta l'oggetto Utente
Signature dei metodi	+ getId(): int + setId(int id): void + getEmail(): String + setEmail(String email): void + getUsername(): String + setUsername(String username): void + getPassword(): String + setPassword(String password): void +getCognome():String +setCognome(Strig cognome):void +getNome():String +setNome(String nome):void +getDataNascita():Date +setDataNascita(Date dataNascita):void + getRuolo(): String + setRuolo(String ruolo): void +getIndirizzoList():List<Indirizzo> +setIndirizzoList(List<Indirizzo> indirizzoList):void +addIndirizzoToList(Indirizzo i):void +removeIndirizzoToList(Indirizzo i):void
Pre-condizioni	
Post-condizioni	
Invariante	

Nome classe	ProductOrdered
Descrizione	Questa classe rappresenta l'oggetto "ProdottoOrdinato"
Signature dei metodi	+ getId(): int + setId(int id): void + getNome(): String + setNome(String nome): void + getDescrizione(): String + setDescrizione(String descrizione): void + getTipo(): String + setTipo(String tipo): void + getSconto(): double + setSconto(double sconto): void + getImmagine(): String + setImmagine(String immagine): void + getNome_categoria():String + setNome_categoria(nome_categoria): void + getQuantity(): int + setQuantity(int quantity): void + getPrezzo(): double + setPrezzo(double prezzo): void + getAnno(): int + setAnno(int anno): void + getRegione(): String + setRegione(String regione): void + getGradazione(): int + setGradazione(int gradazione): void + getFormato(): int + setFormato(int formato): void + getOrdineId(): int + setOrdineId(int ordine): void
Pre-condizioni	
Post-condizioni	
Invariante	

3.2 DAO

Nome classe	CategoriaDAO
Descrizione	la classe “CategoriaDAO” interagisce con il database e restituisce tutti i prodotti di una data categoria
Signature dei metodi	+retriveAll(): ArrayList<Prodotto>
Pre-condizioni	
Post-condizioni	
Invariante	

Nome classe	IndirizzoDAO
Descrizione	La classe “indirizzoDAO” rappresenta tutte le interazioni con il database riguardo all’indirizzo
Signature dei metodi	+retriveAllById(int id):ArrayLisy<Indirizzo> +doSave(Indirizzo i, int id):int +deleteIndirizzo(int id):int +doUpdate(Indirizzo i):int +retriveById(int i):Indirizzo +removeInd(int id):int +retriveIndirizzoOdineById(int id):Indirizzo
Pre-condizioni	
Post-condizioni	
Invariante	

Nome classe	OrdineDAO
Descrizione	La classe “OrdineDAO” interagisce con il database
Signature dei metodi	+doSave(Ordine ordine, int idIndirizzo): int +retriveAll(): ArrayList<Ordine> +retiveByIdUser(int idUser): ArrayList<Ordine> +deleteOrder(int IdOrder): int +addAdressToOrder(Indirizzo indirizzo): int
Pre-condizioni	
Post-condizioni	
Invariante	

Nome classe	ProdottoDAO
Descrizione	La classe “ProdottoDAO” rappresenta tutte interazioni con in database riguardo al prodotto
Signature dei metodi	+retriveOne(int idProdotto): Prodotto +retriveQunat(int n): int +retriveBySearch(String nomeProdotto): List<Prodotto> +retriveAllProductNames(): ArrayList<String> +retriveByName(String nome): Prodotto +retriveAll(): ArrayList<Prodotto> +retriveRand(): ArrayList<Prodotto> +retriveCategory(String cat): ArrayList<Prodotto> +doSave(Prodotto p): int +deleteProduct(int id): int +doUpdateQuantity(int idP , int quantitaAcquistata): int +doUpdate(Prodotto p): int
Pre-condizioni	
Post-condizioni	
Invariante	

Nome classe	ProductOrderedDAO
Descrizione	La classe “ProductOrderedDAO” riguarda i prodotti presenti negli ordini
Signature dei metodi	+retriveByOrderId(int idOrder): ArrayList<ProductOrdered> +doSave(ProductOrdered p, int idOrdine): int +deleteProductOrdered(int id):int
Pre-condizioni	
Post-condizioni	
Invariante	

Nome classe	UtenteDAO
Descrizione	La classe “utenteDAO” riguarda tutte le interazioni con il database per quanto riguarda l’utente
Signature dei metodi	+retriveAll():ArrayList<Utente> +doRetriveByUsernamePassword(String username, String password): Utente +doRetriveByUsernameEmail(username: String, email:String): Utente +doSave(Utente utente): int +doUpdate(Utente utente): int +deleteUser(int iduser): int +retriveById(int idUtente): Utente +deRetriveUsername(String username): Utente +doRetriveByEmail(String email):Utente
Pre-condizioni	
Post-condizioni	
Invariante	

3.3 CONTROL

Nome Classe	AcquistoServlet
Descrizione	Questa classe è un control che passa dei dati al “GestoreCarrello” che li invia a OrdineDAO, ProductOrderedDAO e ProdottoDAO
Signature dei metodi	#doGet(request: HttpServletRequest, response: HttpServletRequest):void
Pre-Condizioni	Context AcquistoServlet:: doGet(request: HttpServletRequest, response: HttpServletRequest) Pre: request.getSession().getAttribute(“utente”)!=null and Request.getSession().getAttribute(“carrello”)!=null
Post-Condizioni	
Invariante	

Nome Classe	AggiungiIndirizzo
Descrizione	Questa classe è un control che passa dei dati all’UtenteDAO col fine di salvare l’indirizzo relativo all’Utente
Signature dei metodi	#doGet(request: HttpServletRequest, response: HttpServletRequest):void
Pre-Condizioni	Context AggiungiIndirizzo:: doGet(request: HttpServletRequest, response: HttpServletRequest) Pre: request.getSession().getAttribute(“utente”)!=null
Post-Condizioni	
Invariante	

Nome Classe	AggiungiProdotto
Descrizione	Questa classe è un control che si occupa di passare i dati al ProdottoDAO col fine di creare un prodotto che verrà inserito nel DataBase
Signature dei metodi	#doGet(request: HttpServletRequest, response: HttpServletRequest):void
Pre-Condizioni	Context AggiungiProdotto:: 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	
Invariante	

Nome Classe	CancellaOrdineByIdServlet
Descrizione	Questa è una classe control che si occupa di cancellare un ordine tramite un id, e ne passa i dati all'OrdineDAO
Signature dei metodi	#oGet(request: HttpServletRequest, response: HttpServletRequest):void
Pre-Condizioni	Context CancellaOrdineByIdServlet:: doGet(request: HttpServletRequest, response: HttpServletRequest) Pre: request.getParameter("id")
Post-Condizioni	
Invariante	

Nome Classe	CancellaProdottoByIdServlet
Descrizione	Questa è una classe control che si occupa di cancellare un prodotto tramite un id, e ne passa i dati all'ProdottoDAO
Signature dei metodi	#doGet(request: HttpServletRequest, response: HttpServletRequest):void
Pre-Condizioni	Context CancellaProdottoByIdServlet:: doGet(request: HttpServletRequest, response: HttpServletRequest) Pre: request.getParameter("id")
Post-Condizioni	
Invariante	

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	
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	
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	
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	
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	
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	#doPost(request: HttpServletRequest, response: HttpServletRequest):void
Pre-condizioni	
Post-condizioni	
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	
Invariante	

Nome classe	ModificaIndirizzo
Descrizione	Servlet che consente di modificare l'indirizzo apartenente ad un utente
Signature dei metodi	#doget(request: HttpServletRequest, response: HttpServletRequest):void
Pre-condizioni	
Post-condizioni	
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	
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	
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	
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	
Invariante	

Nome classe	MostraOrdini
Descrizione	Questa
Signature dei metodi	
Pre-condizioni	
Post-condizioni	
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	
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	
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	
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	
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	
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	
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	
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	
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	

3.4 MODEL

Nome classe	GestoreAccount
Descrizione	Classe model che si occupa di inviare i dati ad "UtenteDAO" e ad "IndirizzoDAO"
Signature dei metodi	+gestioneRegistzione(HttpServletRequest request, HttpServletResponse response):void +gestioneAddIndirizzo(HttpServletRequest request, HttpServletResponse response): void +gestoreCancellaUtente(HttpServletRequest request, HttpServletResponse response):void +gestoreReindirizzamenteLoginProfilo(HttpServletRequest request, HttpServletResponse response):void +gestoreLogin(HttpServletRequest request, HttpServletResponse response):void +gestoreLogout(HttpServletRequest request, HttpServletResponse response):void +gestoreModificaIndirizzo(HttpServletRequest request, HttpServletResponse response):void

	+gestoreModificaUtente(HttpServletRequest request, HttpServletResponse response):void +gestoreModificaUtenreByAdmin(HttpServletRequest request, HttpServletResponse response):void +gestoreMostraUtenti(HttpServletRequest request, HttpServletResponse response):void +gestoreTrovaUtenteById(HttpServletRequest request, HttpServletResponse response):void
Pre-condizioni	
Post-condizioni	
Invariante	

Nome classe	GestoreCarrello
Descrizione	Classe model che si occupa di inviare i dati a “ProdottoDAO” e “IndirizzoDAO”
Signature dei metodi	+controlloIndirizzo(HttpServletRequest request, HttpServletResponse response):boolen +gestoreAggiungiProdottoCarrello(HttpServletRequest request, HttpServletResponse response):void +gestoreRimuoviProdottoCarrello(HttpServletRequest request, HttpServletResponse response):void +gestoreAcquisto(HttpServletRequest request, HttpServletResponse response):void
Pre-condizioni	
Post-condizioni	
Invariante	

Nome classe	GestoreOrdine
Descrizione	Classe model che si occupa di inviare i dati a “IndirizzoDAO”, “OrdineDAO”, “UtenteDAO”
Signature dei metodi	+gestoreCancellaOrdine(HttpServletRequest request, HttpServletResponse response):void +gestoreOrdini(HttpServletRequest request, HttpServletResponse response):void +gestoreRestituisciDettagliOrdine(HttpServletRequest request, HttpServletResponse response):void +gestoreRestituisciOrdini(HttpServletRequest request, HttpServletResponse response):void +gestoreRestituisciOrdiniUtenti(HttpServletRequest request, HttpServletResponse response):void
Pre-condizioni	
Post-condizioni	
Invariante	

Nome classe	GestoreProdotto
Descrizione	Classe model che si occupa di inviare i dati a “ProdottoDAO”
Signature dei metodi	+gestoreAddProdot to(HttpServletRequest request, HttpServletResponse response):void +gestoreCancellaProdotto(HttpServletRequest request, HttpServletResponse response):void +gestoreCategoria(HttpServletRequest request, HttpServletResponse response):void +modificaProdotto(HttpServletRequest request, HttpServletResponse response):void +gestoreModificaProdottoByAdmin(HttpServletRequest request, HttpServletResponse response):void +gestoreMostraProdotto(HttpServletRequest request, HttpServletResponse response):void +gestoreMostraTuttiProdotti(HttpServletRequest request, HttpServletResponse response):void +gestoreRicercaProdotto(HttpServletRequest request, HttpServletResponse response):void
Pre-condizioni	
Post-condizioni	
Invariante	