OBJECT DESIGN DOCUMENT

INTRODUZIONE

Descrive i compromessi di progettazione degli oggetti fatti dagli sviluppatori, le linee guida seguite per le interfacce dei sottosistemi, la scomposizione dei sottosistemi in pacchetti e classi e le interfacce di classe. L'ODD viene utilizzato per scambiare informazioni sull'interfaccia tra i team e come riferimento durante i test.

OBJECT DESIGN TRADE-OFFS

Per quanto riguardo la realizzazione del sistema sono stati individuati i seguenti **trade-offs**:

Memoria vs estendibilità:

Il sistema favorisce l'estendibilità allo spazio di allocazione così da rendere più efficiente lo sviluppo delle nuove funzionalità del sito.

Tempo di risposta vs Affidabilità:

Si farà molta attenzione all'affidabilità in modo tale da garantire la sicurezza dei propri dati personali invece che avere un tempo di risposta breve.

Criteri di manutenzione vs Criteri di performance:

Sarà implementato il sistema in modo tale da preferire la manutenibilità alla performance in modo tale da agevolare lo sviluppo e l'aggiornamento del sito per via delle continue richieste degli utenti. A sfavore delle performance.

Comprensibilità vs Tempo:

Il codice sarà scritto rispettando lo standard di Google per la programmazione nel linguaggio Java in modo tale da dare più comprensibilità, agevolando così il mantenimento del progetto. Assecondando la comprensibilità del codice, il tempo per lo sviluppo aumenterà.

COMPONENTI OFF-THE-SHELF

Le componenti **off-the-shelf** che utilizzeremo saranno:

Il sito web sarà sviluppato con **HTML, CSS** e **Javascript**.

JQuery, Javascript e **Ajax** che renderanno l'interfaccia capace di reagire alle azioni dell'utente, Esse daranno possibilità all'utente di facilitare la navigazione nel sito web.

Per realizzare una connessione con il DB, si utilizzerà un sistema di connection pool.

INTERFACE DOCUMENTATION GUIDELINES

Durante l'implementazione si seguiranno le seguenti linee guida.

Classi e interfacce Java

Si utilizzerà la notazione a gobba di cammello "notazioneGobbaDiCammello"

Nella scrittura delle classi Java si seguirà lo standard Google Java.

Per quanto riguarda le classi Java, si dovranno rispettare:

1) Non inserire spazi tra il nome del metodo e la parentesi tonda che apre la lista dei parametri.
2) La parentesi graffa aperta si trova alla fine della stessa linea dell'istruzione di dichiarazione.
3) La parentesi graffa chiusa inizia in una nuova riga vuota allo stesso livello di indentazione del nome della classe o metodo.
4) Le istruzioni composte devono essere indentate di uno shift all'interno dell'istruzione composta.
5) Le istruzioni composte seguono le stesse regole delle classi e dei metodi

JSP

Il codice Java delle pagine JSP deve combaciare con le convenzioni per la codifica in Java le istruzioni dovranno essere racchiuse negli scriptlet.

HTML

Le pagine HTML, sia in forma statica che dinamica, devono essere conformi allo standard HTML 5. Inoltre, il codice HTML statico deve utilizzare l'indentazione, per facilitare la lettura, secondo le seguenti regole:

1) Un'indentazione consiste in una tabulazione.

2) Ogni tag deve avere un'indentazione maggiore del tag che lo contiene.

3) Ogni tag di chiusura deve avere lo stesso livello di indentazione del corrispondente tag di apertura.

4) I tag di commento devono seguire le stesse regole che si applicano ai tag normali.

CSS

I fogli di stile (CSS) devono seguire le seguenti convenzioni:

Ogni foglio di stile deve essere iniziato da un commento analogo a quello presente nei file Java.

1) I selettori della regola si trovano a livello 0 di indentazione, uno per riga.

2) L'ultimo selettore della regola è seguito da parentesi graffa aperta.

3) Le proprietà che costituiscono la regola sono listate una per riga e sono indentate rispetto ai selettori.

4) La regola è terminata da una parentesi graffa chiusa, collocata da sola su una riga.

SQL

I nomi delle tabelle devono seguire le seguenti regole:

1) Devono essere costituiti da sole lettere maiuscole.

2) Il nome deve essere un sostantivo singolare.

I nomi dei campi devono seguire le seguenti regole:

1) Seguono le stesse regole delle tabelle

2) Se il nome è costituito da più parole, si utilizza l'underscore tra le due parole.

DEFINIZIONI, ACRONIMI E ABBREVIAZIONI

JSP: acronimo di Java Scripting Preprocessor, è una tecnologia di programmazione web in java per lo sviluppo della logica di presentazione (tipicamente secondo il pattern MVC) di applicazioni web.

ODD: Object Design Document .

MVC: acronimo di Model-view-controller, è un pattern architetturale molto diffuso nello sviluppo di sistemi software. Off-The-Shelf: Servizi esterni al sistema di cui viene fatto utilizzo.

Bootstrap: è una raccolta di strumenti liberi per la creazione di siti e applicazioni per il web.

HTML: Linguaggio di programmazione utilizzato per lo sviluppo di pagine Web.

CSS: acronimo di Cascading Style Sheets è un linguaggio usato per definire la formattazione delle pagine Web.

JavaScript: JavaScript è un linguaggio di scripting orientato agli oggetti e agli eventi, comunemente utilizzato nella programmazione Web lato client per la creazione di effetti dinamici interattivi.

JQuery: JQuery è una libreria JavaScript per applicazioni web.

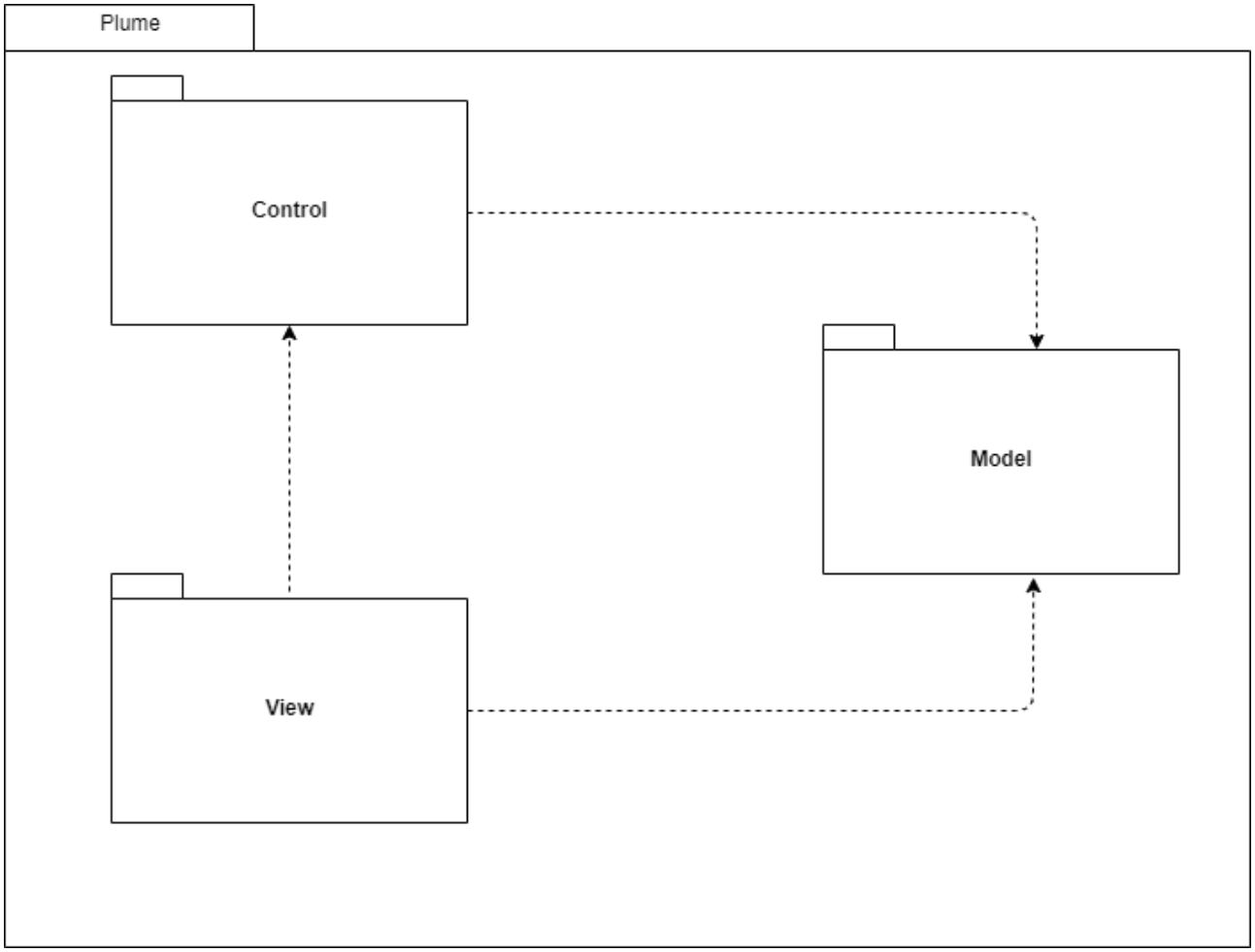
AJAX: AJAX, acronimo di Asynchronous JavaScript and XML, è una tecnica di sviluppo software per la realizzazione di applicazioni web interattive.

Servlet: i servlet sono oggetti scritti in linguaggio Java che operano all'interno di un server web.

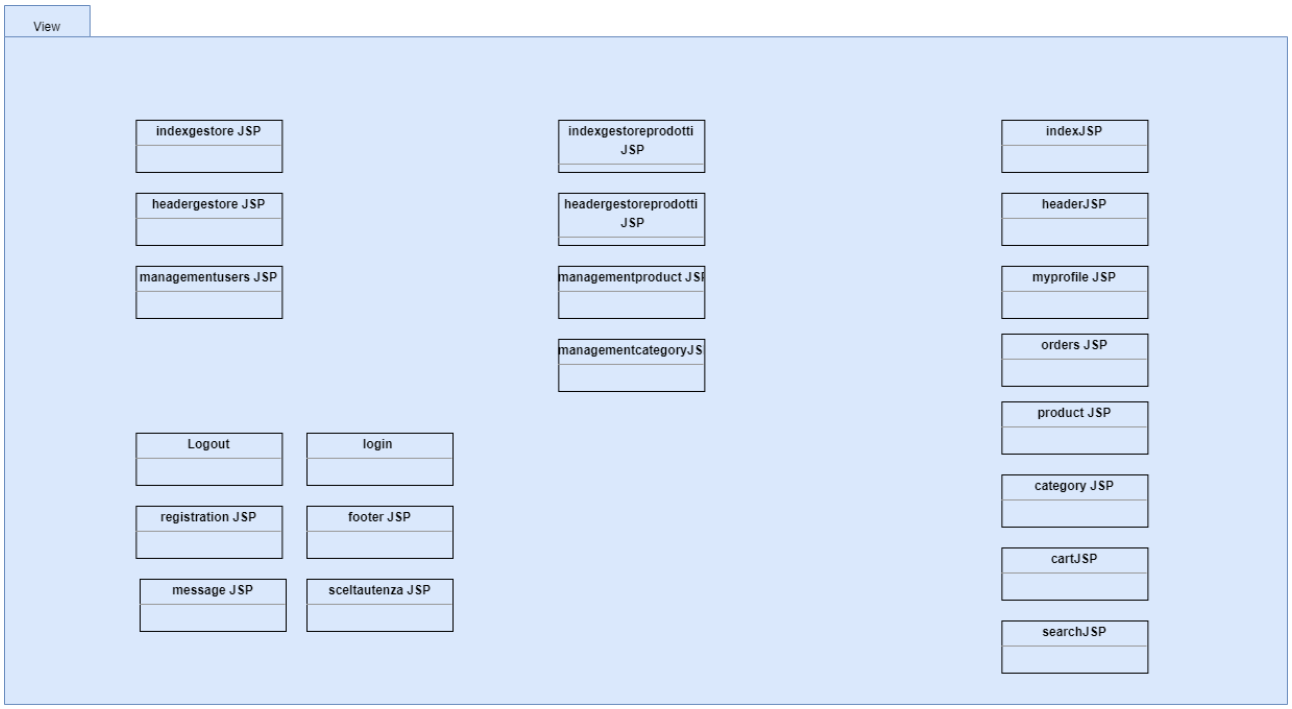
REFERENCES

- Problem Statement-Plume
- Requirements Analysis Document-Plume
- System Design Document-Plume
- Slide prof. De Lucia pubblicate sulla piattaforma e-learning.

PACKAGES

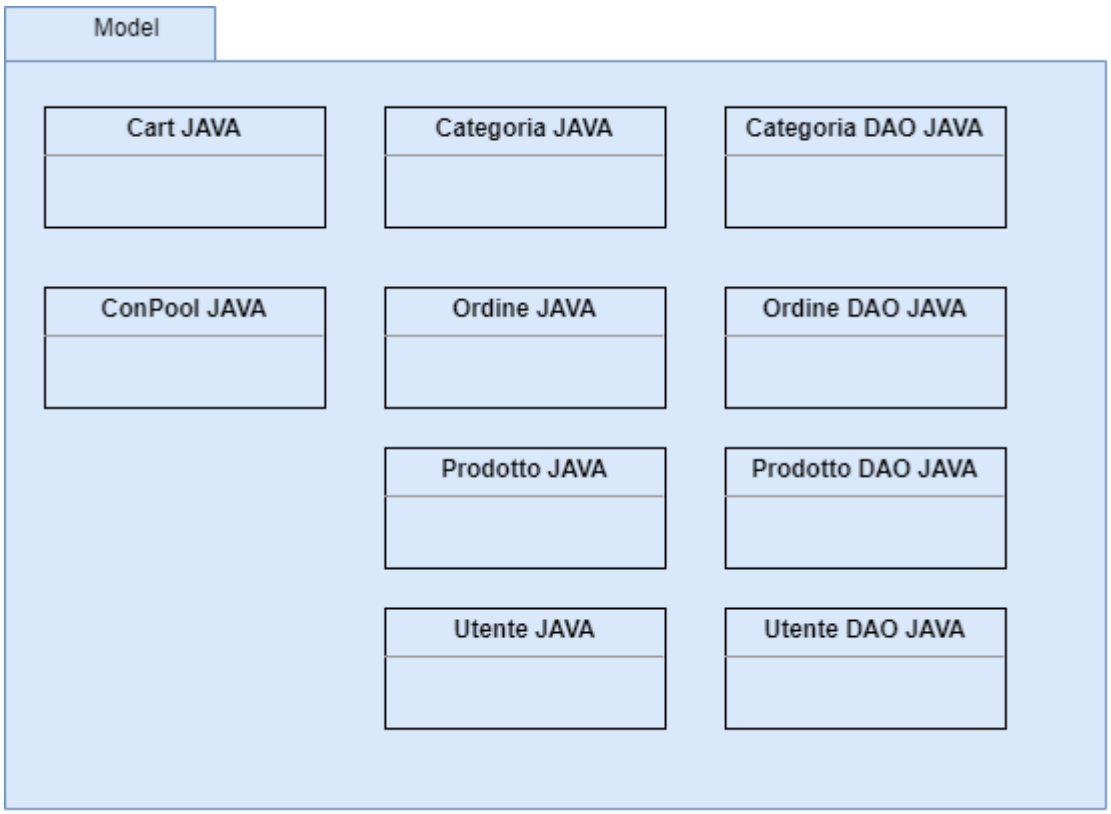


Il design pattern **MVC** consente la suddivisione del sistema in tre blocchi principali: **Model, View e Control**. Il **Model** che fornisce le classi Bean e i DAO che forniscono l'uso dei metodi per gestire il DB, il **View** si occupa dell'iterazione tra l'utente e il sistema (contiene le classi che vanno a formare l'interfaccia grafica), il **Control** riceve i comandi dall'utente tramite il package **View** e gestisce ed elabora le richieste tramite le servlet.

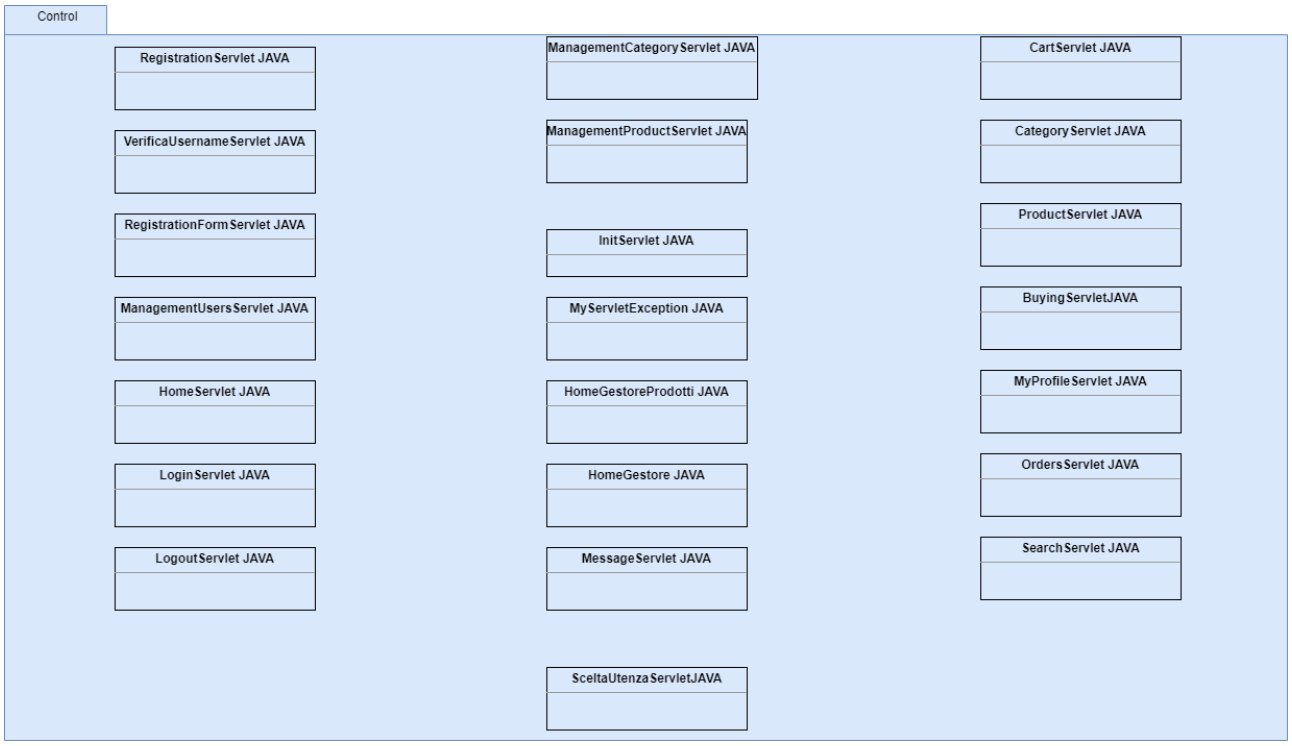


- Il Package **View** è formato:
  - **registration JSP**: viene usata per registrare un nuovo utente al sito web Plume tramite un'interfaccia grafica apposita;
  - l**ogin JSP, logout JSP**: vengono implementate per gestire le azioni per l'autenticazione e disconnessione;
  - **message JSP**: è la pagina che comunica all'utente l'esito dell'operazione da lui effettata;
  - **index JSP**: è la prima pagina che viene mostrata all'utente quando viene aperto il sito web;
  - **headerI JSP**: è la pagina che verrà utilizzata per implementare il menù della pagina principale del sito e della pagina principale del cliente;
  - **footer.html**: è la parte inferiore della struttura del sito web che conterrà le informazioni essenziali del sito;
  - **headerGestore JSP**: è la pagina usata per implementare il menù del Gestore;
  - **indexGestore JSP:** è la pagina che apparirà quando il gestore del sito web effettuerà l'autenticazione;
  - **headerGestoreProdotti JSP**: è la pagina usata per implementare il menù del gestoreProdotti;
  - **indexGestoreProdotti JSP**: è la pagina che apparirà quando un Gestore prodotti effettuerà l'autenticazione;
  - **gestioneUsers JSP**: è la pagina che permette al Gestore tramite l'interfaccia grafica di poter rendere dare una promozione ad un Cliente rendendolo GestoreProdotti o di eliminare un utente;
  - **gestioneCategorie JSP**: è la pagina che permette al GestoreProdotti di aggiungere/eliminare una categoria;
  - **gestioneProdotti JSP**: è la pagina che permette al gestoreProdotti di aggiungere/eliminare un prodotto;
  - **search JSP**: è la pagina che visualizza tutti i prodotti ricercati dal Cliente

- **category JSP**: la pagina che permette al Cliente di visualizzare tutte le informazioni della categoria riguardante uno specifico prodotto;
- **product JSP**: la pagina che permette al Cliente di visualizzare il prodotto con tutte con tutte le sue caratteristiche;
- **cart JSP**: la pagina che permette all'utente di acquistare i prodotti dopo che sono stati selezionati e inseriti nel carrello;
- **myProfile JSP**: la pagina che permette al cliente di visualizzare i propri dati;
- **orders JSP**: pagina che permette al Cliente di visualizzare tutti i suoi ordini che sono stati effettuati;



- Il package **Model** contiene tutte le classi che gestiscono i dati persistenti. Ogni classe contenuta fornisce i metodi per accedere ai dati dell'applicazione. Le classi contenute sono:
  - **Cart**
  - **Categoria**
  - **CategoriaDAO**
  - **ConPool**
  - **Ordine**
  - **OrdineDAO**
  - **Prodotto**
  - **ProdottoDAO**
  - **Utente**
  - **UtenteDAO**



- Il package **Control** riceve i comandi dal package View che vengono effettuati dall'utente. Le Servlet contenute al suo interno sono:
    - **LoginServlet JAVA**: Questa servlet servirà per effettuare l'autenticazione al sito. Essa controllerà se i dati immessi dall'utente siano corretti e indirizzerà l'utente sulla sua homepage;
    - **LogoutServlet JAVA**: Questa servlet interrompe la sessione e reindirizza l'utente alla pagina principale;
    - **RegistrationServlet JAVA**: controlla se i dati durante la registrazione rispetto il giusto formato ed effettua la registrazione;
    - **RegistrationFormServlet JAVA**: indirizza l'utente sulla pagina di registrazione;
    - **ManagementCategoryServlet JAVA**: gestisce e comunica se la rimozione e l'aggiunta di una categoria è andata a buon fine;
    - **ManagementProductServlet JAVA**: gestisce e comunica se la rimozione e l'aggiunta di un prodotto è andata a buon fine;
    - **ManagementUsersServlet JAVA**: gestisce e comunica se la rimozione o la promozione degli utenti è andata a buon fine;
    - **BuyingServelt JAVA**: gestisce, comunica gli ordini e controlla se l'utente è autenticato per effettuarlo;
    - **CategoryServlet JAVA**: nel caso in cui la categoria non avrà nessun prodotto al suo interno avviserà il cliente altrimenti visualizzerà tutti i prodotti contenuti nella categoria;
    - **MyProfileServlet:** indirizza l'utente sulla pagina per la visualizzazione degli ordini;
    - **HomeServlet JAVA**: è la servlet della HomePage;
    - **HomeGestoreProdottiServlet JAVA**: è la servlet della HomePage del gestore prodotti;
    - **HomeGestoreServlet JAVA**: è la servlet della HomePage del gestore;

- **InitServelt JAVA**: sovrascriverà il metodo service () in modo tale da non riscrivere le stesse istruzioni avrà un metodo per controllare se l'utente ha i permessi per effettuare alcune operazioni. Questa classe estenderà tutte le classi.
- **MyServletException JAVA**: servlet per gestire le eccezioni personali;
- **OrdersServlet JAVA**: gestisce la visualizzazione degli ordini;
- **ProductServlet JAVA**: gestirà la visualizzazione del prodotto, nel caso in cui non esiste essa farà visualizzare un messaggio;
- **SearchServlet JAVA**: gestisce la ricerca facendo visualizzare tutti i prodotti correlati ad essa nel caso in cui nessun prodotto corrisponderà invierà un messaggio di esito negativo;
- **CartServlet JAVA**: pagina che gestirà i prodotti nel carrello;
- **VerificaUsernameServlet JAVA:** Controlla quando verrà effettuata la registrazione se verrà inserito l'username, se è uguale ad un altro username presente nel sistema nega la registrazione finchè non ne sarà inserito uno corretto.
- **SceltaUtenza JAVA:** Gestisce la richiesta di cambio utenza di un qualsiasi utente;
- **MessageServlet JAVA:** Servlet che gestirà i messaggi;

CLASS INTERFACES

Nome classe	Categoria
Descrizione	Classe che definisce l'entità categoria
Attributi e metodi	-id : int -nome : String -descrizione : String +getId() +setId(int) +getNome() +setNome(String) +getDescrizione() +setDescrizione(String) +toString() +hashCode() +equals(Object)
Precondizione	
Postcondizione	
Invariante	

Nome classe	Ordine
Descrizione	Classe che definisce l'entità ordine
Attributi e metodi	-id : int -idproduct : int -quantity : double -price : double -product : Prodotto +getId() +setId(int) +getIdproduct() +setIdproduct(int) +getQuantity() +setQuantity(double) +getPrice() +setPrice(double) +getProduct() +setProduct(Prodotto) +toString()
Precondizione	
Postcondizione	
Invariante	

Nome classe	Prodotto
Descrizione	Classe che definisce l'entità prodotto
Attributi e metodi	-id : int -nome : String -descrizione : String -prezzo : double -categoria : Categoria -taglia : String -image : byte[] +getId() +setId(int) +getNome() +setNome(String) +getDescrizione() +setDescrizione(String) +getPrezzo() +setPrezzo(double) +getCategoria() +setCategoria(Categoria) +getTaglia() +setTaglia(String) +getImage() +setImage(byte[]) +toString() +hashCode() +equals(Object)
Precondizione	
Postcondizione	
Invariante	

Nome classe	Utente
Descrizione	Classe che definisce l'entità utente
Attributi e metodi	-id : int -username : String -passwordhash : String -nome : String -email : String -gestore : boolean -gestoreprodotti : Boolean +getId() +setId(int) +getUsername() +setUsername(String) +getPasswordhash() +setPassword(String) +setPasswordhash(String) +getNome() +setNome(String) +getEmail() +setEmail(String) +IsGestore() +setGestore(boolean) +IsGestoreProdotti() +setGestoreProdotti(boolean) +toString() +hashCode() +equals(Object)
Precondizione	
Postcondizione	
Invariante	

Nome classe	Cart
Descrizione	Classe che definisce l'entità carrello
Attributi e metodi	-prodotto : Prodotto -quantita : double -ProdottoQuantita(Prodotto, double) +getQuantita() +setQuantita(double) +getProdotto() +getPrice() -prodotti : LinkedHashMap<Integer, ProdottoQuantita> +getProdotti() +get(int) +put(Prodotto, int) +remove(int) +getPriceTot() +toString() +hashCode() +equals(Object)

Precondizione	
Postcondizione	
Invariante	

Nome classe	CategoriaDAO
Descrizione	Classe che definisce i metodi dell'entità categoria
Attributi e metodi	+doRetrieveAll() +doSave(Categoria) +doDelete(int)
Precondizione	**Context** CategoriaDAO::doSave(categoria:Categoria): void **Pre** categoria != null **Context** CategoriaDAO::doDelete(id : int): void **Pre** id != null
Postcondizione	
Invariante	

Nome classe	OrdineDAO
Descrizione	Classe che definisce i metodi dell'entità categoria
Attributi e metodi	doSave(Ordine, Utente) doSaveP(Ordine) doRetrieveByUser(int, int, int) doRetrieveNumberByUser(int) doRetrieveAll(int, int)
Precondizione	**Context** ProdottoDAO::doSave(ordine Ordine, user Utente) : int **Pre** ordine != null && user != null **Context** OrdineDAO::doSaveP(order: Ordine):void **Pre** order!= null **Context** OrdineDAO::doRetriveByUser(id: int, offset: int, limit: int): List<Ordine> **Pre** id != null Context OrdineDAO:: `doRetrieveNumberByUser(id: int): List<Ordine>` Pre id != null
Postcondizione	
Invariante	

Nome classe	ProdottoDAO
Descrizione	Classe che definisce i metodi dell'entità prodotto
Attributi e metodi	doRetrieveAll(int, int) getCategoria(Connection, int) doRetrieveById(int) doRetrieveByMatch(String, int, int) doRetrieveByCategoria(int, int, int) doDelete(int) doSave(Prodotto)
Precondizione	**Context** ProdottoDAO::getCategoria(connection:Connection, idCategory:int):Categoria

	Pre connection != null && idCategory != null **Context** ProdottoDAO::getdoRetriceById(id:int):Prodotto **Pre** id != null **Context** ProdottoDAO::doRetriveByMatch(string:String, offset:int, limit:int):List<Prodotto> **Pre** string != null && offset != null && limit != null **Context** ProdottoDAO::doRetriveByCategoria(categoria int, offset:int, limit:int):List<Prodotto> **Pre** int != null && offset != null && limit != null **Context** ProdottoDAO::doDelete(id:int):void **Pre** id != null **Context** ProdottoDAO::doSave(product: Prodotto):void **Pre** product != null
Postcondizione	
Invariante	

Nome classe	UtenteDAO
Descrizione	Classe che definisce i metodi dell'entità utente
Attributi e metodi	doRetrieveAll(int, int) doRetrieveByUsernamePassword(String, String) doRetrieveByUsername(String) doSave(Utente) doGestore(int) doGestoreProdotti(int) doDelete(int)
Precondizione	**Context** UtenteDAO::doRetriveByUsernamePassword(username:String, password:String):Utente **Pre** username != null && password != null **Context** UtenteDAO::doRetriveByUsername(username:String):Utente **Pre** username != null **Context** UtenteDAO::doSave(utente:Utente):void **Pre** utente != null **Context** UtenteDAO::doGestore(id:int):void **Pre** id != null && utente.gestore:false **Context** UtenteDAO::doGestoreProdotti(id:int):void **Pre** id != null && utente.gestoreProdotti:false **Context** UtenteDAO::doDelete(id:int):void **Pre** id != null
Postcondizione	**Context** UtenteDAO::doGestore(id:int):void **Post** utente.gestore:true **Context** UtenteDAO::doGestoreProdotti(id:int):void **Post** utente.gestoreProdotti:true
Invariante	

Nome classe	ConPool
Descrizione	Classe che permette la connessione al DB
Attributi e metodi	-freeDbConnections : List<Connection> +createDBConnection() +getConnection()

	+releaseConnection(Connection))
Precondizione	
Postcondizione	
Invariante	

Nome classe	ManagementCategoryServlet
Descrizione	Gestisce e comunica se la rimozione e l'aggiunta di una categoria è andata a buon fine
Attributi e metodi	-categoriaDAO : CategoriaDAO #doGet(HttpServletRequest, HttpServletResponse) #doPost(HttpServletRequest, HttpServletResponse)

Precondizione	**Context** ManagementCategoryServlet::doGet(request:HttpServletRequest, response HttpServletResponse)
	Pre request != null && response != null &&
	IF(request.getParameter("operation") == "add") **then**
	request.getParameter("name") != null &&
	request.getParamenter("description") != null
	IF(request.getParameter("operation") =="remove") **then**
	Integer.parseInt(request.getParameter("category") > 0

Postcondizione	**Context** ManagementCategoryServlet::doGet(request:HttpServletRequest, response HttpServletResponse) **Post** **IF**(request.getParameter("operation") == "add") || (request.getParameter("operation") == "remove") **then** request.getAttribute("message") != null
Invariante	

Nome classe	ManagementProductServlet
Descrizione	Gestisce e comunica se la rimozione e l'aggiunta di un prodotto è andata a buon fine
Attributi e metodi	#doGet(HttpServletRequest, HttpServletResponse) #doPost(HttpServletRequest, HttpServletResponse) -productDAO : ProductDAO

Precondizione	**Context** ManagementCategoryServlet::doGet(request:HttpServletRequest, response HttpServletResponse)
	Pre request != null && response != null &&
	Integer.parseInt(request.getParameter("cat") > 0
	IF(request.getParameter("operation") == "add") **then**
	request.getParameter("name") != null &&
	request.getParameter("description") != null &&
	Double.parseDouble(request.getParameter("price") > 0 &&
	Integer.parseInt(request.getParameter("category") > 0 &&
	request.getParameter("taglia") != null &&
	request.getPart("url") != null &&
	filePart.getInputStream()
	IF(request.getParameter("operation") == "remove") **then**
	Integer.parseInt(request.getParameter("product") > 0

Postcondizione	**Context** ManagementCategoryServlet::doGet(request:HttpServletRequest, response HttpServletResponse) **Post**

	IF(request.getParameter(“operation”) == “add”) || (request.getParameter(“operation”) == “remove”) **then** request.getAttribute(“message”) != null
Invariante	

Nome classe	ManagementUserServlet
Descrizione	Gestisce e comunica se la rimozione o la promozione degli utenti è andata a buon fine
Attributi e metodi	-doGet(HttpServletRequest, HttpServletResponse) -doPost(HttpServletRequest, HttpServletResponse) utenteDAO : UtenteDAO

Precondizione	**Context** ManagementCategoryServlet::doGet(request:HttpServletRequest, response HttpServletResponse)
	Pre request != null && response != null &&
	IF(request.getParameter(“operation”) == “remove”) **then**
	Integer.parseInt(request.getParameter(“id”) > 0
	IF(request.getParameter(“operation”) == “gestore”) **then**
	Integer.parseInt(request.getParameter(“id”) > 0

Postcondizione	**Context** ManagementCategoryServlet::doGet(request:HttpServletRequest, response HttpServletResponse) **Post** **IF**(request.getParameter(“operation”) == “remove”) || (request.getParameter(“operation”) == “gestore”) **then** request.getAttribute(“message”) != null
Invariante	

Nome classe	BuyingServlet
Descrizione	Gestisce, comunica gli ordini e controlla se l’utente è autenticato per effettuarlo
Attributi e metodi	-doGet(HttpServletRequest, HttpServletResponse) -doPost(HttpServletRequest, HttpServletResponse) orderDAO : OrdineDAO
Precondizione	**Context** ManagementCategoryServlet::doGet(request:HttpServletRequest, response HttpServletResponse) **Pre** request != null && response != null && request.getSession().getAttribute(“utente”) != null && request.getSession().getAttribute(“carrello”) != null
Postcondizione	**Context** ManagementCategoryServlet::doGet(request:HttpServletRequest, response HttpServletResponse) **Post** request.getAttribute(“message”) != null
Invariante	

Nome classe	CartServlet
Descrizione	Pagina che gestirà i prodotti nel carrello
Attributi e metodi	-doGet(HttpServletRequest, HttpServletResponse) -doPost(HttpServletRequest, HttpServletResponse) prodottoDAO : ProdottoDAO
Precondizione	**Context** ManagementCategoryServlet::doGet(request:HttpServletRequest, response HttpServletResponse)

	Pre request != null && response != null &&
	Session.getAttribute(“carrello”) != null &&
	IF(session.getAttribute(“carrello”) == null) **then**
	Integer.parseInt(request.getParameter(“prodId”) > 0 &&
	Integer.parseInt(request.getParameter(“quantity”) > 0
	IF(session.getAttribute(“carrello”) != null) **then**
	Integer.parseInt(request.getParameter(“setNum”) > 0

Postcondizione	
Invariante	

Nome classe	CategoryServlet
Descrizione	Nel caso in cui la categoria non avrà nessun prodotto al suo interno avviserà il cliente altrimenti visualizzerà tutti i prodotti contenuti nella categoria
Attributi e metodi	#doGet(HttpServletRequest, HttpServletResponse) #doPost(HttpServletRequest, HttpServletResponse) -prodottoDAO : ProdottoDAO
Precondizione	**Context** ManagementCategoryServlet::doGet(request:HttpServletRequest, response HttpServletResponse) **Pre** request != null && response != null && **IF** (request.getParameter(“id”) != null) Id = Integer.parseInt(request.getParameter(“id”)) >0
Postcondizione	**Context** ManagementCategoryServlet::doGet(request:HttpServletRequest, response HttpServletResponse) **Post** request.getAttribute(“category”) != null
Invariante	

Nome classe	HomegestoreProdottiServlet
Descrizione	Servlet della HomePage del gestore prodotti
Attributi e metodi	#doGet(HttpServletRequest, HttpServletResponse) #doPost(HttpServletRequest, HttpServletResponse)
Precondizione	**Context** ManagementCategoryServlet::doGet(request:HttpServletRequest, response HttpServletResponse) **Pre** request != null && response != null
Postcondizione	
Invariante	

Nome classe	HomegestoreProdottiServlet
Descrizione	Servlet della HomePage del gestore
Attributi e metodi	#doGet(HttpServletRequest, HttpServletResponse) #doPost(HttpServletRequest, HttpServletResponse)
Precondizione	**Context** ManagementCategoryServlet::doGet(request:HttpServletRequest, response HttpServletResponse) **Pre** request != null && response != null

Postcondizione	
Invariante	

Nome classe	HomeServlet
Descrizione	Servlet pagina iniziale e homepage cliente
Attributi e metodi	#doGet(HttpServletRequest, HttpServletResponse) #doPost(HttpServletRequest, HttpServletResponse)
Precondizione	**Context** ManagementCategoryServlet::doGet(request:HttpServletRequest, response HttpServletResponse) **Pre** request != null && response != null
Postcondizione	
Invariante	

Nome classe	InitServlet
Descrizione	Sovrascriverà il metodo service () in modo tale da non riscrivere le stesse istruzioni avrà un metodo per controllare se l'utente ha i permessi per effettuare alcune operazioni. Questa classe estenderà tutte le classi
Attributi e metodi	#doGet(HttpServletRequest, HttpServletResponse) #doPost(HttpServletRequest, HttpServletResponse) -productDAO : ProdottoDAO -categoryDAO : CategoriaDAO -userDAO : UtenteDAO
Precondizione	**Context** ManagementCategoryServlet::doGet(request:HttpServletRequest, response HttpServletResponse) **Pre** request != null && response != null **IF**(request.getSession().getAttribute("utente") || utente.isGestore==false || utente.isGestoreProdotti == false)
Postcondizione	
Invariante	

Nome classe	LoginSevlet
Descrizione	Questa servlet servirà per effettuare l'autenticazione al sito. Essa controllerà se i dati immessi dall'utente siano corretti e indirizzerà l'utente sulla sua homepage in caso di esito positivo
Attributi e metodi	#doGet(HttpServletRequest, HttpServletResponse) #doPost(HttpServletRequest, HttpServletResponse) -utenteDAO : UtenteDAO
Precondizione	**Context** ManagementCategoryServlet::doGet(request:HttpServletRequest, response HttpServletResponse) **Pre** request != null && response != null request.getParameter("username") != null request.getParamenter("password") != null
Postcondizione	**Context** ManagementCategoryServlet::doGet(request:HttpServletRequest, response HttpServletResponse) **Post** request.getAttribute("utente") != null
Invariante	

Nome classe	LogoutSevlet
Descrizione	Questa servlet interrompe la sessione e reindirizza l'utente alla pagina principale
Attributi e metodi	#doGet(HttpServletRequest, HttpServletResponse) #doPost(HttpServletRequest, HttpServletResponse)
Precondizione	**Context** ManagementCategoryServlet::doGet(request:HttpServletRequest, response HttpServletResponse) **Pre** request != null && response != null && Request.getSession().getAttribute("utente") != null
Postcondizione	**Context** ManagementCategoryServlet::doGet(request:HttpServletRequest, response HttpServletResponse) **Post** request.getSession().getAttribute("utente") == null
Invariante	

Nome classe	MyProfileServletServlet
Descrizione	Indirizza l'utente sulla pagina per la visualizzazione degli ordini
Attributi e metodi	#doGet(HttpServletRequest, HttpServletResponse) #doPost(HttpServletRequest, HttpServletResponse)
Precondizione	**Context** ManagementCategoryServlet::doGet(request:HttpServletRequest, response HttpServletResponse) **Pre** request != null && response != null request.getParameter("user") != null
Postcondizione	
Invariante	

Nome classe	MessageServlet
Descrizione	Servlet che gestirà i messaggi
Attributi e metodi	#doGet(HttpServletRequest, HttpServletResponse) #doPost(HttpServletRequest, HttpServletResponse)
Precondizione	**Context** ManagementCategoryServlet::doGet(request:HttpServletRequest, response HttpServletResponse) **Pre** request != null && response != null
Postcondizione	
Invariante	

Nome classe	OrdersServlet
Descrizione	Gestisce la visualizzazione degli ordini
Attributi e metodi	#doGet(HttpServletRequest, HttpServletResponse) #doPost(HttpServletRequest, HttpServletResponse) -orderDAO : OrdineDAO -userDAO : UtenteDAO
Precondizione	**Context** ManagementCategoryServlet::doGet(request:HttpServletRequest, response HttpServletResponse) **Pre** request != null && response != null request.getParamenter("user") != null
Postcondizione	**Context** ManagementCategoryServlet::doGet(request:HttpServletRequest, response HttpServletResponse) **Post**

	request.getAttribute(“orders”) != null
Invariante	

Nome classe	ProductServlet
Descrizione	Servlet che gestisce i prodotti
Attributi e metodi	#doGet(HttpServletRequest, HttpServletResponse) #doPost(HttpServletRequest, HttpServletResponse) -prodottoDAO : ProdottoDAO
Precondizione	**Context** ManagementCategoryServlet::doGet(request:HttpServletRequest, response HttpServletResponse) **Pre** request != null && response != null **IF**(request.getParamenter(“id”) != null Id = Integer.ParseInt(request.getParameter(“id”)) > 0
Postcondizione	**Context** ManagementCategoryServlet::doGet(request:HttpServletRequest, response HttpServletResponse) **Post** request.getAttribute(“prod”) != null
Invariante	

Nome classe	RegistrationFormServlet
Descrizione	Indirizza l’utente sulla pagina di registrazione
Attributi e metodi	#doGet(HttpServletRequest, HttpServletResponse) #doPost(HttpServletRequest, HttpServletResponse)
Precondizione	**Context** ManagementCategoryServlet::doGet(request:HttpServletRequest, response HttpServletResponse) **Pre** request != null && response != null
Postcondizione	**Context** ManagementCategoryServlet::doGet(request:HttpServletRequest, response HttpServletResponse) **Post** request.getSession().getAttribute(“utente”) == null
Invariante	

Nome classe	RegistrationServlet
Descrizione	Controlla se i dati durante la registrazione rispettano il formato giusto ed effettua la registrazione
Attributi e metodi	#doGet(HttpServletRequest, HttpServletResponse) #doPost(HttpServletRequest, HttpServletResponse) -utenteDAO : UtenteDAO
Precondizione	**Context** ManagementCategoryServlet::doGet(request:HttpServletRequest, response HttpServletResponse) **Pre** request != null && response != null && request.getParameter(“username”) != null && request.getParameter(“password”) != null && request.getParameter(“passwordConferma”) != null && request.getParameter(“nome”) != null && request.getParameter(“email”) != null && `IF( (username.length() >= 6 && username.matches("^[0-9a-zA-Z]+$"))  ||`

	(password.length() >= 8 && !password.toUpperCase().equals(password) && !password.toLowerCase().equals(password) && password.matches(".*[0-9].*"))|| password.equals(passwordConferma) || (nome.trim().length() > 0 && nome.matches("^[a-zA-Z\u00C0-\u00ff]+$")) \|\| (email.matches("^\\w+([\\.-]?\\w+)*@\\w+([\\.-]?\\w+)*(\\.\\w+)+$")))
Postcondizione	**Context** ManagementCategoryServlet::doGet(request:HttpServletRequest, response HttpServletResponse) **Post** request.getSession("user") != null
Invariante	

Nome classe	SceltaUtenzaServlet
Descrizione	Gestisce la richiesta di cambio utenza di un qualsiasi utente
Attributi e metodi	#doGet(HttpServletRequest, HttpServletResponse) #doPost(HttpServletRequest, HttpServletResponse)
Precondizione	**Context** ManagementCategoryServlet::doGet(request:HttpServletRequest, response HttpServletResponse) **Pre** request != null && response != null
Postcondizione	
Invariante	

Nome classe	SearchServlet
Descrizione	Gestisce la ricerca facendo visualizzare tutti i prodotti correlati ad essa nel caso in cui nessun prodotto corrisponderà invierà un messaggio di esito negativo
Attributi e metodi	#doGet(HttpServletRequest, HttpServletResponse) #doPost(HttpServletRequest, HttpServletResponse) -prodottoDAO : ProdottoDAO
Precondizione	**Context** ManagementCategoryServlet::doGet(request:HttpServletRequest, response HttpServletResponse) **Pre** request != null && response != null request.getParameter("str") != null
Postcondizione	**Context** ManagementCategoryServlet::doGet(request:HttpServletRequest, response HttpServletResponse) **Post** request.getAttribute("search") != null
Invariante	

Nome classe	VerificaUsernameServlet
Descrizione	Controlla se l'username ha un duplicato
Attributi e metodi	#doGet(HttpServletRequest, HttpServletResponse) #doPost(HttpServletRequest, HttpServletResponse)

	utenteDAO : UtenteDAO
Precondizione	**Context** ManagementCategoryServlet::doGet(request:HttpServletRequest, response HttpServletResponse) **Pre** request != null && response != null Username = request.getParamenter(“username”) != null `username.length() >= 6 && username.matches("^[0-9a-zA-Z]+$") && utenteDAO.doRetrieveByUsername(username) == null`
Postcondizione	**Context** ManagementCategoryServlet::doGet(request:HttpServletRequest, response HttpServletResponse) **Post** validaUsername().usernameOK = true
Invariante	