Università degli Studi di Salerno

Corso di Ingegneria del Software

DressApp

Test Execution Plan

Versione 1.0

Data: 07/01/2023

Coordinatore del progetto:

Nome	Matricola
Francesco Ferrara	0512109789

Partecipanti:

Nome	Matricola
Alaia Michele	0512112861
Auleta Antonio	0512106822
Ferrara Francesco	0512109789
Giammarino Josè Luis	0512107086

Scritto da:
Alaia Michele
Auleta Antonio
Ferrara Francesco
Giammarino Josè Luis

Revision History:

Data	Versione	Descrizione	Autori
29/12/2022	**0.1**	**Inizio stesura ODD**	**Tutti i partecipanti**
07/01/2023	**1.0**	**Completamento ODD**	**Tutti i partecipanti**

Indice

1. Introduzione

Dopo la realizzazione del **Requirement Analysis Document** e **System Design Document**, abbiamo descritto quello che sarà il nostro sistema e i nostri obiettivi, tralasciando gli aspetti implementativi.
L'Object Design Document 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 signatures dei sottosistemi definiti nel System Design Document.
Inoltre, sono specificati i trade-offs e le linee guida.

1.1 Object Design Trade-offs

- **Comprensibilità →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, ma questa caratteristica incrementerà il tempo di sviluppo del nostro progetto.

- **Interfaccia → Usabilità**
  L'interfaccia grafica è stata realizzata in modo da essere molto semplice, chiara e concisa: tra le altre cose vengono utilizzati form e pulsanti disposti in maniera da rendere più semplice l'utilizzo del sistema.

- **Sicurezza → Efficienza**
  La sicurezza, come descritto nei requisiti non funzionali del Requirement Analysis Document, rappresenta uno degli aspetti chiave della piattaforma. Tuttavia, dati i tempi di sviluppo molto limitati, ci limiteremo ad implementare sistemi di sicurezza basati sulla crittografia della password degli utenti.

- **Response time → Hardware**
  Il sistema garantisce una certa reattività alle richieste, e quindi è in grado di offrire diversi servizi agli utenti, ma tale caratteristica verrà limitata dall'hardware del sistema.

- **Prestazioni → Costi**
  Il sistema prevede l'utilizzo di fogli di stile semplici e open per mantenere prestazioni elevate, e quindi non saranno utilizzate oggetti o librerie non open source.

1.2 Linee guida per la documentazione delle interfacce

Per lo sviluppo del codice verranno seguite le seguenti convenzioni:

- **Naming Convention**
  È buona norma utilizzare nomenclature che rispettino le seguenti caratteristiche: 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, la prima parola inizierà con la lettera minuscola, mentre le altre parole che seguiranno inizieranno con la lettera maiuscola (es: nomePrimaVariabile).

- **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 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()**.

- **Classi e File**
  I nomi delle classi e dei file devono iniziare con una lettera maiuscola e devono fornire informazioni utili relative al loro scopo.
  Ogni file sorgente contiene una singola classe e deve essere strutturato in un determinato modo:
  1. L' istruzione package che permette di inserire la classe in un determinato package.
  2. L' istruzione import che importa le librerie necessarie alla class.
  3. Una piccola descrizione della classe.
  4. Il codice effettivo del file sorgente.

1.3 Riferimenti

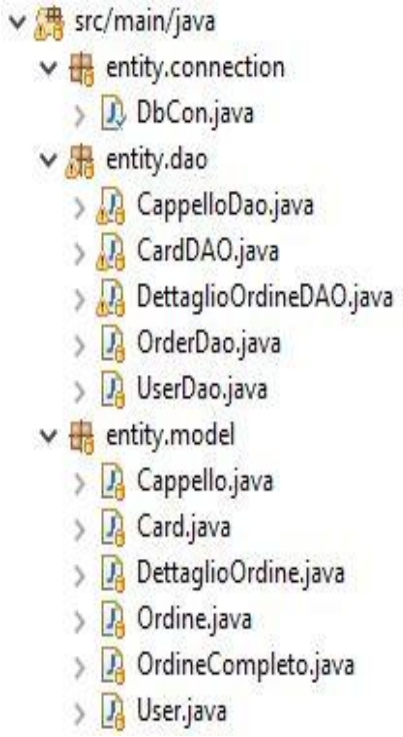
Questo documento utilizza riferimenti a concetti introdotti ed esplicitati nel RAD e nel SDD di DressApp.

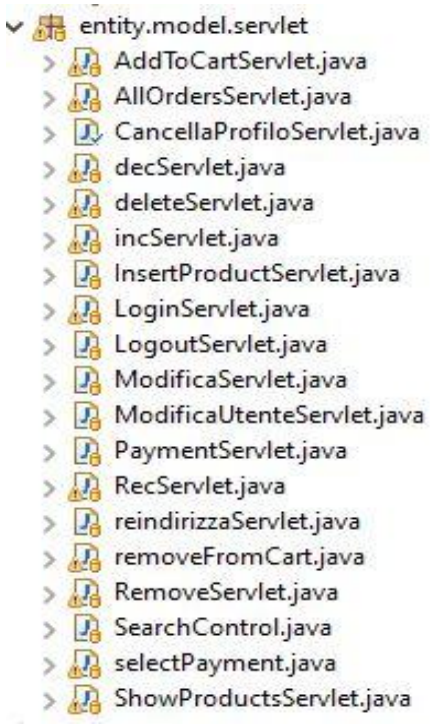
2. Packages

2.1 Back-end Packages

L'implementazione del back-end si declina nei seguenti pacchetti, ognuno dei quali è contenuto nella directory src/main/java/:

- Connection
- Dao
- Model
- Servlet





Connection	
Classe	**Descrizione**
DBConnectionPool	Classe che permette l'interazione con il database

Dao/	
Classe	**Descrizione**
CappelloDao	Pattern architetturale per la gestione della persistenza dei cappelli nel database.
CardDao	Pattern architetturale per la gestione della persistenza delle card nel database.
DettaglioOrdineDao	Pattern architetturale per la gestione della persistenza dei dettagli degli ordini nel database.
OrderDao	Pattern architetturale per la gestione della persistenza degli ordini nel database.
UserDao	Pattern architetturale per la gestione della persistenza degli utenti nel database.

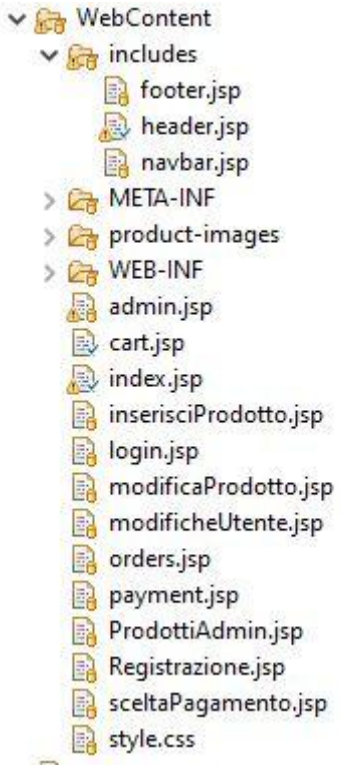
Model/	
Classe	**Descrizione**
Cappello	Classe java che delinea le caratteristiche di un Cappello
Card	Classe java che delinea le caratteristiche di una card
DettaglioOrdine	Classe java che delinea le caratteristiche di un Dettaglio Ordine
Ordine	Classe java che delinea le caratteristiche di un Ordine
OrdineCompleto	Classe java che delinea le caratteristiche di un Ordine Completo
User	Classe java che delinea le caratteristiche di un Utente

Servlet/	
Classe	**Descrizione**
AddToCartServlet	*Servlet che gestisce l'aggiunta dei prodotti al carrello*
AllOrdersServlet	*Servlet che gestisce tutti gli ordini degli Utenti*
CancellaProfiloServlet	*Servlet che gestisce la cancellazione del profilo di un utente*
decServlet	*Servlet che gestisce il decremento della quantità di un cappello dal carrello*
deleteServlet	*Servlet che gestisce la cancellazione del profilo di un utente*
incServlet	*Servlet che gestisce l'incremento della quantità di un cappello dal carrello*
InsertProductServlet	Servlet che gestisce l'inserimento di un prodotto nel catalogo
LoginServlet	Servlet che gestisce il Login
LogoutServlet	Servlet che gestisce il Logout
ModificaServlet	*Servlet che gestisce la modifica di un cappello da parte dell'Admin*
ModificaUtenteServlet	*Servlet che gestisce la modifica delle informazioni di un utente*
PaymentServlet	*Servlet che gestisce il pagamento di un Ordine*
RecServlet	*Servlet che gestisce la registrazione di un nuovo utente*
removeFromCartServlet	*Servlet che gestisce la rimozione di un cappello dal carrello*
RemoveServlet	*Servlet che gestisce la rimozione di un cappello dal catalogo*
SearchControl	*Servlet che gestisce la ricerca di un cappello*
selectPayment	*Servlet che gestisce la selezione del metodo di Pagamento*
ShowProductsServlet	*Servlet che gestisce la visualizzazione dei Cappelli*

2.2 Front-end Packages

L'implementazione del front-end si declina nei seguenti pacchetti, ognuno dei quali è contenuto nella directory /WebContent/:

- Include
- Product-images
- JSP e CSS



Includes/	
JSP	**Descrizione**
Footer	View del footer
Header	View dell header
Navbar	View della navbar

WebContent/	
JSP	**Descrizione**
admin	View dell'admin
cart	View del carrello
index	View della Home
inserisciProdotto	View della pagina di Inserimento di un Cappello
login	View del login
modificaProdotto	View per la modifica di un cappello
modificheUtente	View delle modifiche del profilo di un utente
orders	View della lista degli ordini di un utente
payment	View per il pagamento e completamento dell'ordine
ProdottiAdmin	View del catalogo con opzioni aggiuntive per l'admin
sceltaPagamento	View della Scelta del metodo di pagamento
style.css	style per pagina di autenticazione

3. Class Interface

3.1 Connection

Nome Classe	DBConnectionPool
Descrizione	Classe che permette l'interazione con il database
Signature dei metodi	+ createDBConnection() : Connection + getConnection() : Connection + releaseConnection(Connection) : void

3.2 Dao

Nome Classe	CappelloDao
Descrizione	Pattern architetturale per la gestione della persistenza dei cappelli nel database.
Signature dei metodi	+ getAllProducts() : List<Cappello> + searchItems(String) : ArrayList<Cappello> + removeProduct(int) : void + insertProduct(String,String,float,String,String,int) : void + retriveProductById(int) : Cappello + updateProdotto(int,String,String,float,String,String,int,Date): boolean + updateDisp(int, int) : void + updateModificato(int) : void + riduciDisponibilita(int, int) : void

Nome Classe	CardDao
Descrizione	Pattern architetturale per la gestione della persistenza delle card nel database.
Signature dei metodi	+ getAllCards(int) : ArrayList<Card> + removeCard(int) : void + insertCard(String, String, String, int, int) : boolean + retriveCardById(int) : Card

Nome Classe	DettaglioOrdineDao
Descrizione	Pattern architetturale per la gestione della persistenza dei dettagli degli ordini nel database.
Signature dei metodi	+ getAllDettaglioOrdini() : List<DettaglioOrdine> + insertDettaglioOrdine(int, int ,int) : void + searchDettaglioOrdineByOrdineId(int) : ArrayList<DettaglioOrdine> + searchOrdiniCompleti(int) : ArrayList<OrdineCompleto> + updateQuantita(int, int, int) : void + getQuantita(int, int) : int + removeById(int, int) : void

Nome Classe	OrderDao
Descrizione	Pattern architetturale per la gestione della persistenza degli ordini nel database.
Signature dei metodi	+ changeState(int) : boolean + getAllOrders() : ArrayList<Ordine> + getMailUserbyOrderId(int) : String + doSave(Date, int, boolean) : void + doRetriveById(int) : Ordine + doRetriveByIdBuy(int) : Ordine + doDelete(int) : boolean + doUpdateById(int, String, String) : boolean + searchOrdersFromNameProduct(String) : ArrayList<Ordine> + idGrande() : int + getOrdersByUser(int) : ArrayList<Ordine>

Nome Classe	UserDao
Descrizione	Pattern architetturale per la gestione della persistenza degli utenti nel database.
Signature dei metodi	+ userLogin(String,String) : User + retrivebyMail(String) : boolean + getMailById(int) : String + userRec(String, String, String, String, int, int) : User + doDelete(String) : boolean + doDeletebyId(int) : boolean + doUpdate(User, String, String) : void + doUpdateById(int, String, String) : boolean

3.3 Model

Nome Classe	**Cappello**
Descrizione	Classe che descrive i cappelli
Signature dei metodi	+ Cappello(String, String, Float, String, String, Int) + getId() : int + getNome() : String + getDescrizione() : String + getPrezzo() : float + getCategoria() : String + getFoto() : String + getDisponibilità() : int + getModificato() : boolean + setNome(String) : void + setDescrizione(String) : void + setPrezzo(float) : void + setCategoria(String) : void + setFoto(String) : void + setDisponibilità(int) : void + setModificato(boolean) : void + toString() : String + equals(Object) : boolean

Nome Classe	**User**
Descrizione	Classe che descrive gli utenti
Signature dei metodi	+ User(String, String, String, String, Boolean) + getId() : int + getEmail() : String + getPassword() : String + getNome() : String + getCognome() : String + getIsAdmin() : Boolean + setEmail(String) : void + setPassword(String) : void + setNome(String) : void + setCognome(String) : void + setIsAdmin(Boolean) : void + toString() : String + equals(Object) : boolean

Nome Classe	**Card**
Descrizione	Classe che descrive i metodi di pagamento degli utenti
Signature dei metodi	+ Card(String, String, String, Int, Int) + getId() : int + getProprietario() : String + getNumeroCarta() : String + getDataScadenza() : String + getCVV() : int + getUser() : int + setProprietario(String) : void + setNumeroCarta(int) : void + setDataScadenza(String) : void + setCVV(int) : void + setUser(int) : void + toString() : String + equals(Object) : boolean

Nome Classe	**Ordine**
Descrizione	Classe che descrive gli ordini degli utenti
Signature dei metodi	+ Ordine(Date, Int, Boolean) + getId() : int + getData() : Date + getUser() : int + getIsBuy() : Boolean + setId(int) : void + setData(Date) : void + setUser(int) : void + setIsBuy(Boolean) : void + toString() : String + equals(Object) : boolean

Nome Classe	Dettaglio Ordine
Descrizione	Classe che associa i cappelli agli ordini degli utenti
Signature dei metodi	+ DettaglioOrdine(Int, Int, Int) + getCappello() : int + getOrdine() : int + getQuantita() : int + setCappello(int) : void + setOrdine(int) : void + setQuantita(int) : void + toString() : String + equals(Object) : boolean

3.4 Servlet

Nome classe	AddToCartServlet
Descrizione	Servlet che gestisce l'aggiunta dei prodotti al carrello

Signature dei metodi	# doGet(HttpServletRequest , HttpServletResponse) : void
	action:
	● orderDao
	● dettaglioOrdineDao
	● ordine
	● dettaglioOrdine
	● user
	
	# doPost(HttpServletRequest, HttpServletResponse) : void
	doGet(HttpServletRequest,HttpServletResponse)

Nome classe	AllOrdersServlet
Descrizione	*Servlet che gestisce tutti gli ordini degli Utenti*

Signature dei metodi	# doGet(HttpServletRequest , HttpServletResponse) : void
	action:
	● orderDao
	● user
	
	# doPost(HttpServletRequest, HttpServletResponse) : void
	doGet(HttpServletRequest,HttpServletResponse)

Nome classe	**CancellaProfiloServlet**
Descrizione	*Servlet che gestisce la cancellazione del profilo di un utente*

Signature dei metodi	# doGet(HttpServletRequest , HttpServletResponse) : void
	action:
	● userDao
	
	# doPost(HttpServletRequest, HttpServletResponse) : void
	doGet(HttpServletRequest,HttpServletResponse)

Nome classe	**decServlet**
Descrizione	*Servlet che gestisce il decremento della quantità di un cappello dal carrello*

Signature dei metodi	# doGet(HttpServletRequest , HttpServletResponse) : void
	action:
	● user
	● dettaglioOrdineDao
	
	# doPost(HttpServletRequest, HttpServletResponse) : void
	doGet(HttpServletRequest,HttpServletResponse)

Nome classe	**deleteServlet**
Descrizione	*Servlet che gestisce la cancellazione del profilo di un utente*

Signature dei metodi	# doGet(HttpServletRequest , HttpServletResponse) : void
	action:
	● cardDao
	● user
	
	# doPost(HttpServletRequest, HttpServletResponse) : void
	doGet(HttpServletRequest,HttpServletResponse)

Nome classe	**incServlet**
Descrizione	*Servlet che gestisce l'incremento della quantità di un cappello dal carrello*
Signature dei metodi	# doGet(HttpServletRequest , HttpServletResponse) : void action: ● dettaglioOrdineDao ● user ● CappelloDao ● Cappello

	# doPost(HttpServletRequest, HttpServletResponse) : void
	doGet(HttpServletRequest,HttpServletResponse)

Nome classe	**InsertProductServlet**
Descrizione	Servlet che gestisce l'inserimento di un prodotto nel catalogo

Signature dei metodi	# doGet(HttpServletRequest , HttpServletResponse) : void
	action:
	● cappelloDao
	
	# doPost(HttpServletRequest, HttpServletResponse) : void
	doGet(HttpServletRequest,HttpServletResponse)

Nome classe	**LoginServlet**
Descrizione	Servlet che gestisce il Login

Signature dei metodi	# doGet(HttpServletRequest , HttpServletResponse) : void
	doPost(HttpServletRequest,HttpServletResponse)
	
	# doPost(HttpServletRequest, HttpServletResponse) : void
	action:
	● userDao
	● user

Nome classe	**LogoutServlet**
Descrizione	Servlet che gestisce il Logout

Signature dei metodi	# doGet(HttpServletRequest , HttpServletResponse) : void
	action:
	● user
	
	# doPost(HttpServletRequest, HttpServletResponse) : void
	doGet(HttpServletRequest,HttpServletResponse)

Nome classe	**ModificaServlet**
Descrizione	*Servlet che gestisce la modifica di un cappello da parte dell'Admin*
Signature dei metodi	# doGet(HttpServletRequest , HttpServletResponse) : void

	doPost(HttpServletRequest,HttpServletResponse) # doPost(HttpServletRequest, HttpServletResponse) : void action: ● cappelloDao ● cappello

Nome classe	**ModificaUtenteServlet**
Descrizione	*Servlet che gestisce la modifica delle informazioni di un utente*

Signature dei metodi	# doGet(HttpServletRequest , HttpServletResponse) : void
	action:
	● userDao
	
	# doPost(HttpServletRequest, HttpServletResponse) : void
	doGet(HttpServletRequest,HttpServletResponse)

Nome classe	**PaymentServlet**
Descrizione	*Servlet che gestisce il pagamento di un Ordine*

Signature dei metodi	# doGet(HttpServletRequest , HttpServletResponse) : void
	action:
	● cardDao
	
	# doPost(HttpServletRequest, HttpServletResponse) : void
	doGet(HttpServletRequest,HttpServletResponse)

Nome classe	**RecServlet**
Descrizione	*Servlet che gestisce la registrazione di un nuovo utente*

Signature dei metodi	# doGet(HttpServletRequest , HttpServletResponse) : void
	doPost(HttpServletRequest,HttpServletResponse)
	# doPost(HttpServletRequest, HttpServletResponse) : void
	action:
	● userDao
	● user

Nome classe	**removeFromCartServlet**
Descrizione	*Servlet che gestisce la rimozione di un cappello dal carrello*

Signature dei metodi	# doGet(HttpServletRequest , HttpServletResponse) : void
	action:
	● dettaglioOrdineDao
	
	# doPost(HttpServletRequest, HttpServletResponse) : void
	doGet(HttpServletRequest,HttpServletResponse)

Nome classe	**RemoveServlet**
Descrizione	*Servlet che gestisce la rimozione di un cappello dal catalogo*

Signature dei metodi	# doGet(HttpServletRequest , HttpServletResponse) : void
	action:
	● cappelloDao
	● user
	
	# doPost(HttpServletRequest, HttpServletResponse) : void
	doGet(HttpServletRequest,HttpServletResponse)

Nome classe	**SearchControl**
Descrizione	*Servlet che gestisce la ricerca di un cappello*

Signature dei metodi	# doGet(HttpServletRequest , HttpServletResponse) : void
	action:
	● cappello
	● cappelloDao
	
	# doPost(HttpServletRequest, HttpServletResponse) : void
	doGet(HttpServletRequest,HttpServletResponse)

Nome classe	**selectPayment**
Descrizione	*Servlet che gestisce la selezione del metodo di Pagamento*

Signature dei metodi	# doGet(HttpServletRequest , HttpServletResponse) : void
	action:
	● orderDao
	
	# doPost(HttpServletRequest, HttpServletResponse) : void
	doGet(HttpServletRequest,HttpServletResponse)

Nome classe	**ShowProductsServlet**
Descrizione	*Servlet che gestisce la visualizzazione dei Cappelli*

Signature dei metodi	# doGet(HttpServletRequest , HttpServletResponse) : void
	action:
	● cappelloDao
	● cappello
	● user
	
	# doPost(HttpServletRequest, HttpServletResponse) : void
	doGet(HttpServletRequest,HttpServletResponse)