Università degli Studi di Salerno

Corso di Ingegneria del Software

OBJECT DESIGN DOCUMENT

Nome progetto:

EsteticaMente

Anno accademico 2018/2019

Partecipanti:	Matricola:
Aurora Scola	0512103834
Lucia Forte	0512103948
Marco Minucci	0512106088

Revision History

Data	Versione	Descrizione	Autori
09/01/2019	1.0	Prima stesura ODD	Minucci Marco, Scola Aurora, Forte Lucia
11/01/2019	1.1	Aggiunti paragrafi	Minucci Marco, Scola Aurora, Forte Lucia

INDICE

1. Introduzione

1.1. Object design trade-offs

Dopo aver prodotto i documenti di Requirement Analysis e di System Design nei quali è stato presentato il nostro sistema tralasciando i dettagli implementativi, andiamo ora a stilare il documento di Object Design ove andremo a definire un modello capace di integrare in maniera precisa le funzionalità individuate nei documenti precedenti.

In maniera particolare si vogliono definire le interfacce delle classi, le operazioni, i tipi, gli argomenti, la signature dei sottosistemi definiti nella progettazione del sistema software, i trade-offs e le linee guida.

Comprensibilità vs Tempo

Il codice del nostro sistema dovrà essere di facile comprensione e lettura, quindi sarà corredato di commenti per facilitare anche la fase di testing o eventuali future modifiche. Questo, ovviamente, comporterà un aumento del tempo di sviluppo.

Interfaccia vs Usabilità

L'interfaccia è stata realizzata per rendere l'utilizzo del nostro sistema di facile utilizzo, così che gli utenti possano interfacciarvisi in maniera semplice ed immediata, andando ad aumentare l'usabilità. Per questo scopo utilizzeremo il framework opensource Bootstrap per il front-end.

Sicurezza vs Efficienza

La sicurezza rappresenta un requisito non funzionale del nostro sistema, come riportato nel RAD. Avendo però tempi di realizzazione ristretti, realizzeremo un sistema di sicurezza concentrato sull'autenticazione degli utenti tramite username e password e la conseguente cifratura di queste ultime. Saranno inoltre aggiunti dei controlli per evitare l'accesso non autorizzato alle funzionalità del sistema controllando i dati della sessione. Questi accorgimenti potrebbero influire sull'efficienza del sistema in favore della sicurezza.

1.2. Linee guida per la documentazione delle interfacce

Gli sviluppatori dovranno rispettare determinate linee guida per la stesura del codice:

NAMING CONVENTION:

È buona norma utilizzare nomi:

- Descrittivi
- Pronunciabili
- Di uso comune
- Di lunghezza medio-corta
- Evitando la notazione ungherese

- Utilizzando solo caratteri consentiti (a-z, A-Z, 0-9)

VARIABILI:

- I nomi delle variabili dovranno cominciare con la lettera minuscola ed eventuali altre parole successive dovranno avere la prima lettera maiuscola. La dichiarazione delle variabili deve essere effettuata ad inizio blocco.
- Sarà possibile usare il carattere "_" in proprietà statiche o variabili costanti.

METODI:

- I nomi dei metodi dovranno cominciare con la lettera minuscola e le altre successive parole dovranno avere la prima lettera maiuscola. I nomi dei metodi dovranno essere rappresentati da verbi esemplificativi della funzionalità che andranno a svolgere. I nomi dei metodi che saranno utilizzati per accesso o modifica di variabili devono essere del tipo getNomeVariabile() e setNomeVariabile(). (Es. getNome(), setNome()).
- I metodi devono essere corredati da commenti che ne indicano la funzione, parametri e valori di ritorno.

CLASSI E PAGINE:

- I nomi delle classi e delle pagine devono cominciare con la lettera maiuscola, così come le altre parole successive. Anche i nomi delle classi devono essere evocativi circa il loro scopo.
- 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

1.3. Definizioni, acronimi e abbreviazioni

Acronimi

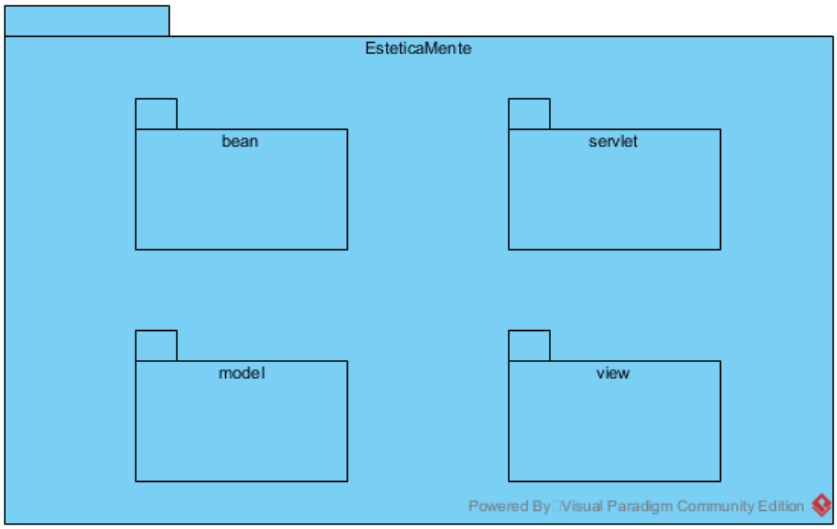
- **RAD:** Requirement Analysis Document

1.4. Riferimenti

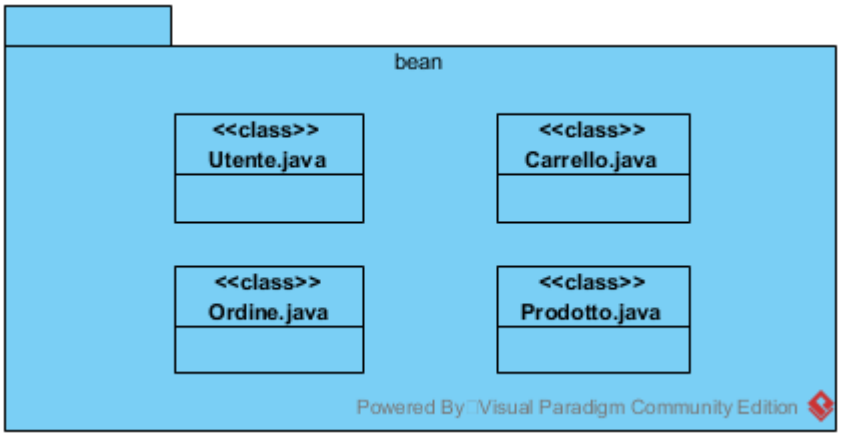
- B. Bruegge, A. H. Dutoit, Object Oriented Software Engineering - Using UML,
- Pattern and Java, Prentice Hall, 3rd edition, 2009
- Documento SDD del progetto EsteticaMente
- Documento RAD del progetto EsteticaMente

2. Packages

2.1. Package Core

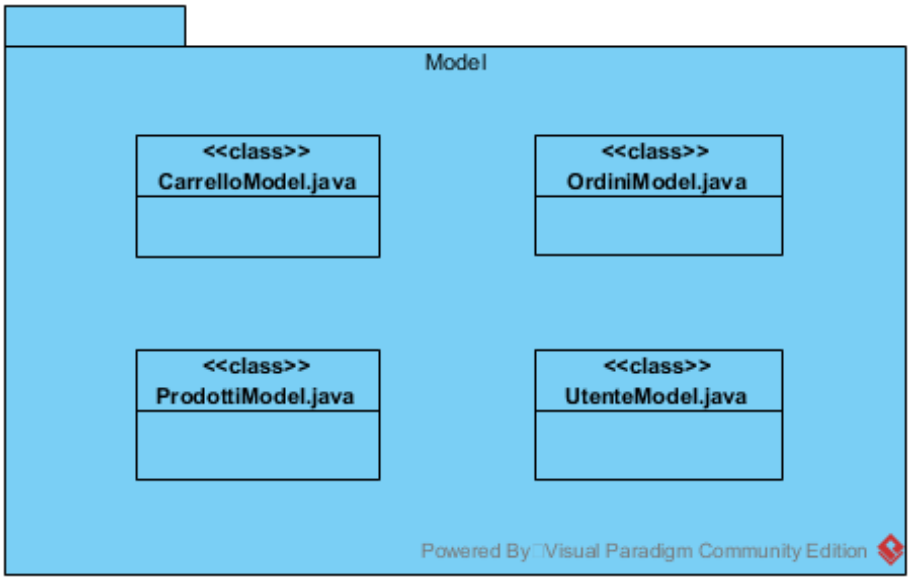


2.1.1. Package bean



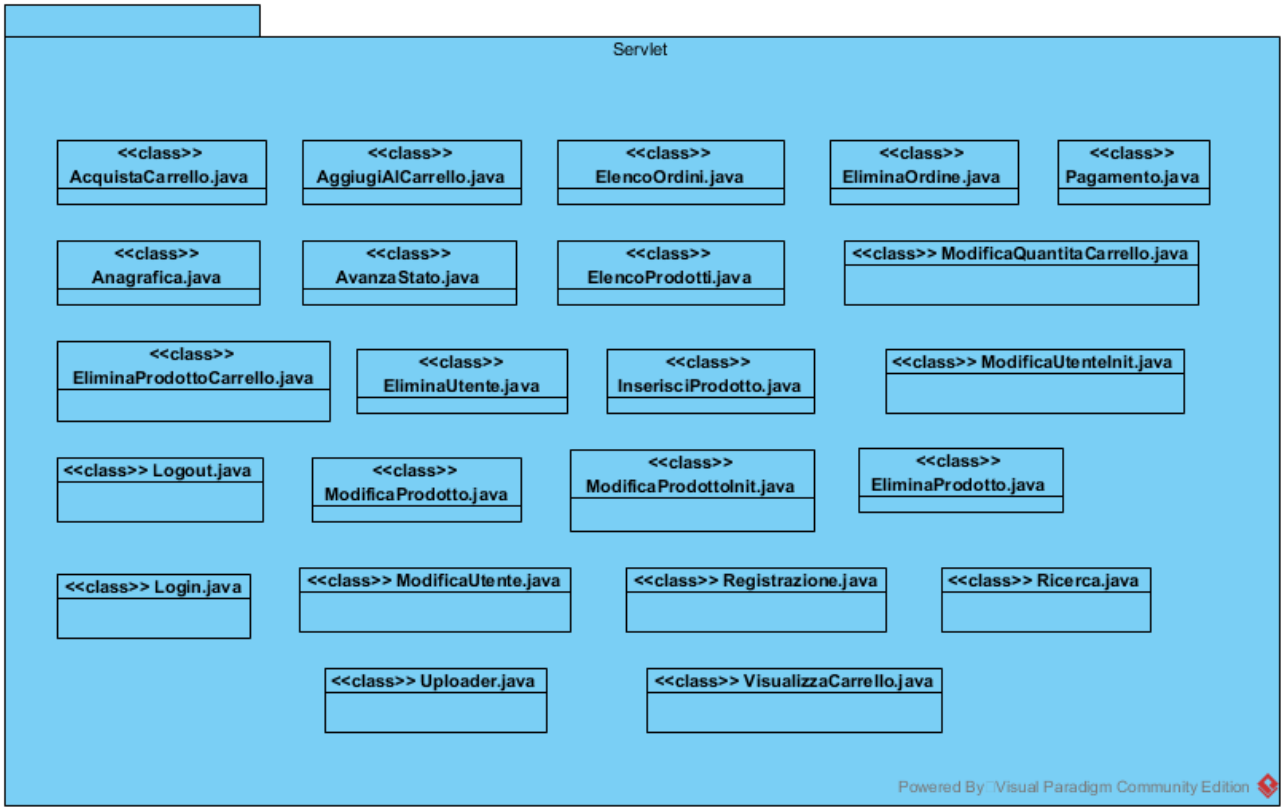
Classe	**Descrizione**
Utente.java	Descrive un utente registrato del sistema.
Carrello.java	Descrive un carrello di un utente.
Prodotto.java	Descrive un prodotto del sistema.
Ordine.java	Descrive un ordine effettuato da un utente del sistema.

2.1.2. Package model



Classe	Descrizione
CarrelloModel.java	Il model che effettua le operazioni riguardanti il carrello, interfacciandosi al DB a cui è connesso.
ProdottiModel.java	Il model che effettua le operazioni riguardanti i prodotti del nostro sistema, interfacciandosi al DB al quale è connesso.
OrdiniModel.java	Il model che esegue le operazioni legate agli ordini del sistema, interfacciandosi al DB a cui è connesso.
UtenteModel.java	Il model che esegue le operazioni legate agli utenti del nostro sistema, interfacciandosi al DB a cui è connesso.

2.1.3. Package servlet



Classe	Descrizione
AcquistaCarrello.java	Controller che interviene quando viene confermato l'acquisto del carrello.
AggiungiAlCarrello.java	Controller che Interviene quando viene aggiunto un prodotto ad un carrello.
Anagrafica.java	Controller che si occupa di far visualizzare i dati anagrafici dell'utente.
AvanzaStato.java	Controller che interviene quando l'amministratore cambia lo stato di un ordine.
ElencoOrdini.java	Controller che si attiva quando viene richiesta la visualizzazione degli ordini effettuati.
ElencoProdotti.java	Controller che si attiva quando viene richiesta la visualizzazione dei prodotti del sistema.
EliminaOrdine.java	Controller invocato quando l'utente decide di eliminare un ordine.
EliminaProdotto.java	Controller che permette di eliminare un prodotto dal sistema da parte dell'amministratore.
EliminaProdottoCarrello.java	Controller che permette all'utente di rimuovere un prodotto dal carrello.
EliminaUtente.java	Controller che permette ad un utente di eliminare il proprio account dal sistema.

InserisciProdotto.java	Controller che permette all'amministratore di inserire un prodotto nel sistema.
Login.java	Controller che permette ad un utente di effettuare il login al sistema.
Logout.java	Controller che permette ad un utente di effettuare il logout al sistema.
ModificaProdotto.java	Controller che permette all'amministratore di modificare i dati di un prodotto presente nel sistema.
ModificaProdottoInit.java	Controller che interviene per visualizzare i campi del prodotto che si vuole modificare.
ModificaQuantitaCarrello.java	Controller che permette di modificare la quantità di un prodotto nel carrello.
ModificaUtente.java	Controller che permette all'utente di modificare i propri dati dell'account.
ModificaUtenteInit.java	Controller che mostra all'utente la sua pagina di modifica dei dati in base al proprio ruolo nel sistema.
Pagamento.java	Controller che perette all'utente di effettuare il pagamento del proprio ordine.
Registrazione.java	Controller che permette la gestione della registrazione.
Ricerca.java	Controller che permette ad un utente di effettuare la ricerca dei prodotti presenti nel sistema.
Uploader.java	Controller che permette di effettuare l'upload dell'immagine di un prodotto.
VisualizzaCarrello.java	Controller che permette all'utente di visualizzare il carrello.

2.1.4. Package view

- AnagraficaAmministratore.jsp
- AnagraficaErrore.jsp
- AnagraficaUtente.jsp
- Carrello.jsp
- Catalogo.jsp
- ChiSiamo.jsp
- Database.jsp
- DatabaseErrore.jsp
- EliminazionePSuc.jsp
- EliminazioneUtente.jsp
- EliminazioneUtenteSucc.jsp
- ErroreGenerale.jsp
- Index.jsp
- InserimentoPFallita.jsp
- InserimentoPSuccesso.jsp
- InserisciProdotto.jsp

- Login.jsp
- LoginErrore.jsp
- LoginNotifica.jsp
- ModificaProdotto.jsp
- ModificaProdottoS.jsp
- ModificaUtente.jsp
- ModificaUtenteE.jsp
- ModificaUtenteS.jsp
- ModificaUtenteUserE.jsp
- Occasioni.jsp
- OrdineAmminsitratore.jsp
- OrdineCliente.jsp
- Pagamento.jsp
- Registrazione.jsp
- RegistrzioneEsistente.jsp
- RegistrazioneFallita.jsp
- RegistrazioneSuccesso.jsp

3. Interfaccia delle classi

3.1. Gestione Utente

- public void modificaAccount(Utente usr, String username)
  Effettua la modifica dei dati dell'account utente.

- public boolean eliminaUtente(String username)
  Elimina l'account utente dal sistema.

- public void modificaUtente(Utente usr, String username)
  Effettua la modifica dei dati dell'utente senza alterare l'username.

- public Utente returnInfo(String username)
  Restituisce un oggetto Utente contenente i dati dell'utente passato come parametro.

3.1.1. Gestione Autenticazione

- public String verifyAccess(String username, String password)
  Verifica i dati di accesso dell'utente durante il login.

3.1.2. Gestione Registrazione

- public void insertUser(Utente usr)
  Aggiunge un nuovo utente al sistema.

- public boolean controllUser(String username)
  Controlla se esiste già l'username nel sistema.

3.2. Gestione Prodotti

- public void insertProdotto(Prodotto usr)
  Inserisce un nuovo prodotto nel sistema.

- public boolean eliminaProdotto(String idProdotto)
  Elimina un prodotto dal sistema.

- public ArrayList<Prodotto> returnProdotti()
  Restituisce la lista dei prodotti nel sistema.

- public void modificaImmagine(int idProdotto, String Url)
  Modifica l'immagine di un prodotto.

- public void modificaProdotto(int idProdotto, Prodotto prd)
  Modifica le informazioni di un prodotto del sistema.

- public Prodotto returnInfo(int idProdotto)
  Restituisce un oggetto della classe Prodotto con le informazioni relative all'ID passato come parametro.

- public ArrayList<Prodotto> ricerca(int numero, String nome)
  Effettua la ricerca di un prodotto all'interno del sistema.

3.3. Gestione Ordini

- public int creaOrdine(int idCarrello, String utente, Double prezzo)
  Crea un nuovo ordine.

- public boolean eliminaOrdine(int idOrdine)
  Elimina un ordine.

- public void avanzaStato(String Stato, int idOrdine)
  Aggiorna lo stato relativo ad un ordine effettuato.

- public ArrayList<Ordine> returnOrdini()
  Restituisce una lista degli ordini effettuati presenti nel sistema.

- public ArrayList<Ordine> returnOrdiniUtente(String username)
  Mostra all'utente i propri ordini.

3.3.1. Gestione Pagamento

- public void inserisciIban(int idOrdine, String iban)
  Inserisce l'iban dell'utente per il pagamento.

3.4. Gestione Carrello

- public boolean creaCarrello(String username)
  Crea un nuovo carrello per l'utente.

- public boolean eliminaProdottoCarrello(int numeroCarrello, int idProdotto)
  Elimina un prodotto presente nel carrello specifico.

- public boolean aggiungiProdottoCarrello(String username, int idProdotto, int quantita)
  Aggiunge un prodotto, con le relative quantità, al carrello dell'utente.

- public boolean cambiaQuantitaCarrello(int numeroCarrello, int quantita, int idProdotto)
  Modifica la quantità di un prodotto nel carrello.

- public Carrello returnCarrello(String User)
  Restituisce il carrello di un utente con i prodotti selezionati.

4. Object Constraints Language (OCL)

4.1. Gestione Utente

Nome classe: UtenteModel

Descrizione: Rappresenta il manager che svolge le attività relative all'utente

Invariante: -

Context:		
returnInfo(username: String): Utente	**Pre:**	username!=null

	Post:	-
modificaUtente(usr: User, username: String)	**Pre:**	username != null usr != null
	Post:	-
modificaAccount(usr: Utente, username: String): void	**Pre:**	usr!=null username!=null
	Post:	-
eliminaUtente(username: String): boolean	**Pre:**	username!=null
	Post:	-

4.1.1. Gestione Autenticazione

Nome classe: UtenteModel

Descrizione: Rappresenta il manager che svolge le attività relative all'autenticazione

Invariante: -

Context:		
verifyAccess(username: String, password: String): String	**Pre:**	username!=null password!=null
	Post:	-

4.1.2. Gestione Registrazione

Nome classe: UtenteModel

Descrizione: Rappresenta il manager che svolge le attività relative alla registrazione

Invariante: -

Context:		
insertUser(usr: Utente): void	**Pre:**	usr!=null
	Post:	-
controllUser(username: String): boolean	**Pre:**	username != null
	Post:	-

4.2. Gestione Prodotto

Nome classe: ProdottiModel

Descrizione:
Rappresenta il manager che esegue operazioni di inserimento, eliminazione, ricerca e modifica dei prodotti.

Invariante: -

Context:		
insertProdotto(prodotto: Prodotto): void	**Pre:**	prodotto!=null
	Post:	-
eliminaProdotto(idProdotto: String): boolean	**Pre:**	idProdotto!=null
	Post:	-

returnProdotti(): ArrayList<Prodotto>	**Pre:**	-
	Post:	-
modificaImmagine(idProdotto: int, url: String): void	**Pre:**	idProdotto!=null url!=null
	Post:	-
modificaProdotto(idProdotto: int, prd: Prodotto): void	**Pre:**	idProdotto!=null prd!=null
	Post:	-
returnInfo(idProdotto: int): Prodotto	**Pre:**	idProdotto!=null
	Post:	-
ricerca(numero: int, nome: String): ArrayList<Prodotto>	**Pre:**	numero!=null nome!=null
	Post:	-

4.3. Gestione Ordini

Nome classe: OrdiniModel

Descrizione: Rappresenta il manager che esegue operazioni riguardo gli ordini.

Invariante: L'utente deve aver scelto dei prodotti da acquistare

Context:		
creaOrdine(idCarrello: int, utente: String, prezzo: double): int	**Pre:**	idCarrello!=null utente!=null prezzo!=null

	Post:	-
eliminaOrdine(idOrdine: int): boolean	**Pre:**	idOrdine!=null
	Post:	-
avanzaStato(stato: String, idOrdine: int): void	**Pre:**	stato!=null idOrdine!=null
	Post:	-
returnOrdini(): ArrayList<Ordine>	**Pre:**	-
	Post:	-
returnOrdiniUtente(username: String): ArrayList<Ordine>	**Pre:**	username!=null
	Post:	-

4.3.1. Gestione Pagamento

Nome classe: OrdiniModel

Descrizione: Rappresenta il manager che esegue l'operazione del pagamento.

Invariante: L'utente deve aver confermato l'ordine

Context:		
inserisciIban(idOrdine: int, iban String): void	**Pre:**	idOrdine!=null iban!=null
	Post:	-

4.4. Gestione Carrello

Nome classe: ProdottiModel

Descrizione:
Rappresenta il manager che esegue le operazioni di inserimento, cancellazione e modifica del carrello.

Invariante: -

Context:		
creaCarrello(username: String): boolean	**Pre:**	username!=null
	Post:	-
eliminaProdottoCarrello(numeroCarrello: int, idProdotto: int): boolean	**Pre:**	numeroCarrello!=null idProdotto!=null
	Post:	-
aggiungiProdottoCarrello(username: String, idProdotto: int, quantita: int): boolean	**Pre:**	username!=null idProdotto!=null quantita!=null
	Post:	-
cambiaQuantitaCarrello(numeroCarrello: int, quantita: int, idProdotto: int): boolean	**Pre:**	numeroCarrello!=null quantita!=null idProdotto!=null
	Post:	-
returnCarrello(user: String): Carrello	**Pre:**	user!=null
	Post:	-