Università degli Studi di Salerno
Corso di Ingegneria del Software

MalTour
Object Design Document
Versione 2.0

Data: 25/12/2021

Progetto: MalTour	Versione: 2.0
Documento: SSD	Data: 25/11/2021

Coordinatore del progetto:

Nome	Matricola
Teresa Vitagliano	052105622

Partecipanti:

Nome	Matricola
Teresa Vitagliano	0512105622

Scritto da:	Teresa Vitagliano

Revision History

Data	Versione	Descrizione	Autore
24/11/2021	1.0	Prima stesura del documento	Teresa Vitagliano
01/12/2021	1.1	criteri	Teresa Vitagliano
02/12/2021	1.2	Proposta architettura software	Teresa Vitagliano

1. Introduction

1.1 Object Design Trade-off

Complessità vs Tempo

Il codice deve essere quanto più comprensibile possibile per facilitare future modifiche. Il codice deve essere quindi commentato opportunamente, ma ciò richiederà più tempo per lo sviluppo.

Sicurezza vs Efficienza

La sicurezza rappresenta un elemento importante nel sistema, come indicato anche nei requisiti non funzionali. Però, dati i tempi di sviluppo limitati, ci limiteremo ad implementare funzioni di sicurezza basati su username e password.

Response Time vs Hardware

Il tempo di risposta rappresenta un fattore importante nel sistema, in particolare su determinate funzionalità. Tutto ciò dipenderà però anche dal tipo hardware su cui verrà fatto eseguire il sistema.

1.2 Interface documentation guidelines

Gli sviluppatori seguiranno le seguenti linee guida per la definizione delle interfacce:

- Classi Java e Servlet:
  - I nomi dovranno iniziale con la lettera maiuscola.
  - Se il nome contiene più parole, ognuna di esse dovrà iniziare con una lettera maiuscola.
  - I nomi dovranno corrispondere alle informazioni e/o funzionalità che offre quella classe o Servlet.
- Metodi:
  - I nomi dovranno iniziare con la lettera minuscola.
  - Se il nome contiene più parole, ognuna di esse dovrà iniziare con la lettera maiuscola.
  - I nomi dovranno corrispondere alle informazioni e/o funzionalità che offre quel metodo. Si utilizzerà un verbo o aggettivi.
  - I nomi dei metodi per ottenere e settare attributi seguiranno la regola di nominazione get[nomeAttributo] e set[nomeAttributo].
- Variabili:
  - I nomi delle variabili dovranno iniziale con lettera minuscola.
  - Se il nome contiene più parole, ognuna di esse dovrà iniziare con la lettera maiuscola.

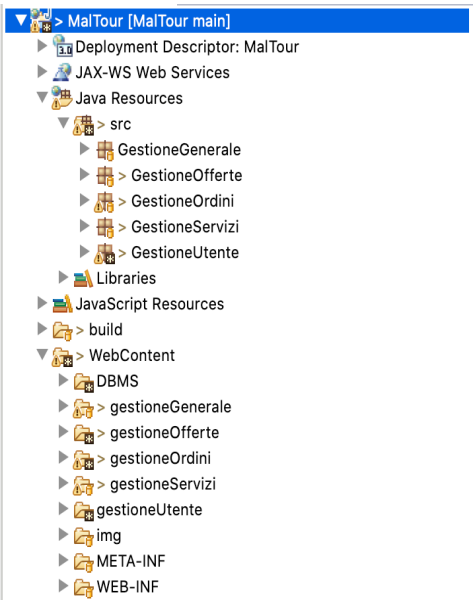
1.3 References

RAD: Requirement Analysis Document

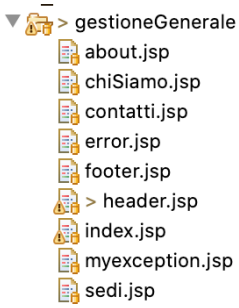
SDD: System Design Document
ODD: Object Design Document

2. Packages

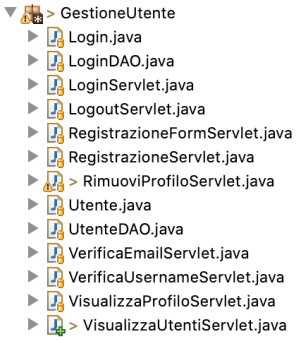
Il sistema è suddiviso in packages nel modo seguente:

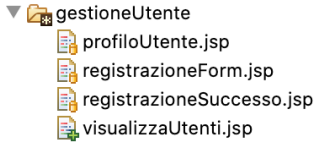


- gestioneGenerale
-

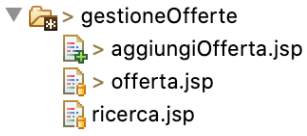


- gestioneUtente

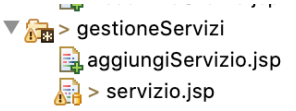




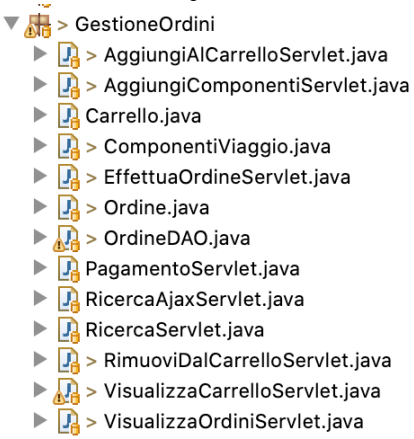
- gestioneOfferte



- gestioneServizi



- gestioneOrdini



3. Class interfaces

Nome Classe	Utente
Descrizione	Questa classe Bean rappresenta l'oggetto Utente
Signature dei metodi	+ getId():int + setId(id: int): void

	+ getUsername():String + setUsername(username: String): void + getPassword (): String + setPassword(password: String): void + getNome(): String + setNome(nome: String): void + getEmail():String + setEmail(email: String) + isCliente(): boolean + isGestoreOfferte() :boolean + isGestoreOrdini():boolean + setGestore(gestore: int): void
Pre-condizioni	Context Utente:: getEmail() Pre: l'email deve avere una corrispondenza nel DB, Context Utente:: getUsername() Pre: l'username deve avere una corrispondenza nel DB
Post-condizioni	Context Utente:: setPassword(password) Post: la password ha una corrispondenza nel DB, con l'email
Invariante	

Nome Classe	**UtenteDAO**
Descrizione	Questa classe è un manager che si occupa di interagire con il database. Gestisce le query riguardanti l'Utente.
Signature dei metodi	+ doRetrieveByUsernamePassword(String username, String password): Utente + doRetrieveAll(): List<Utente> + verificaGestore(email: String): int + doRetrieveById(id: int): Utente + doRetrieveByUsername(username: String): Utente + doRetrieveByEmail(email: String): Utente + doSave(utente: Utente): boolean + doUpdate(utente: Utente): int + doDelete(id: int): boolean
Pre-condizioni	Context UtenteDAO::doRetrieveByUsernamePassword(username,password) Pre: username != null, password != null Context UtenteDAO::doRetrieveAll() Pre: Context UtenteDAO::verificaGestore(email) Pre: email != null Context UtenteDAO::doRetrieveById(id) Pre: id != null

	Context UtenteDAO::doRetrieveByUsername(username) Pre: username != null Context UtenteDAO::doRetrieveByEmail(email) Pre: email != null Context UtenteDAO::doSave(utente) Pre: utente != null Context UtenteDAO::doUpdate(utente) Pre: utente != null Context UtenteDAO::doDelete(id) Pre: id != null
Post-condizioni	Context UtenteDAO::doRetrieveByUsernamePassword(username,password) Post: utente->/select(u | utente.username = username and utente.password =password)) Context UtenteDAO::doRetrieveAll(offset, limit) Post: database.utente Context UtenteDAO::verificaGestore(email) Post: database.utente->(select(u | u.email = utente.gestore) Context UtenteDAO::doRetrieveById(id) Post: utente -> (select (u | utente.id =id) Context UtenteDAO::doRetrieveByUsername(username) Post: utente -> (select (u | utente.id =id)) Context UtenteDAO::doRetrieveByEmail(email) Post: utente -> (select (u | utente.email = email)) Context UtenteDAO::doSave(utente) Post: database.utente -> incudes(select(u | u.id = utente.id)) Context UtenteDAO::doUpdate(utente) Post: Context UtenteDAO::doDelete(id) Post: database.utente -> not includes(select(u | id=id)
Invariante	

Nome Classe	**Offerta**
Descrizione	Questa classe Bean rappresenta l'oggetto Offerta
Signature dei metodi	+ getId():int + setId(id: int): void + getDestinazione():String

	+ setDestinazione(destinazione: String): void + getDescrizione (): String + setDescrizione(descrizione: String): void + getData_partenza(): Date + setData_partenza(data_partenza: Date): void + getOra_partenza(): int + setOra_partenza (int: String): void + getData_ritorno():date + setData_ritorno (data_ritorno: String): void + getOra_ritorno(): int + setOra_ritorno(ora_ritorno: ing): void + getPartenza_da(): String + setPartenza_da(partenza_da: String): void + getArrivo_a(): String + setArrivo_a(arrivo_a: String): void + getPernottamento(): String + setPernottamento(pernottamento: String): void + getPosti_disponibili(): int + setPosti_disponibili(posti_disponibili: int): void + getPrezzo(): long + setPrezzo(prezzoCent: long): void + getServizi():List<Servizio> + setServizi(List<Servizio> servizi): void
Pre-condizioni	
Post-condizioni	
Invariante	

Nome Classe	**OfferteDAO**
Descrizione	Questa classe è un manager che si occupa di interagire con il database. Gestisce le query riguardanti le Offerte.
Signature dei metodi	+ doRetrieveAll(offset: int, limit: int): List<Offerta> + doRetrieveById(id: int): Offerta + doRetrieveByServizio(servizio: int, offset: int, limit: int): List<Offerta> + doRetrieveByNomeOrDescrizione(against: String, offset: int, limit: int): List<Offerta> + doRetrieveByDestinazione(against: String, offset: int, limit: int): List<Offerta> + doSave(offerta: Offerta): int + doUpdate(offerta: Offerta): boolean + doDelete(id: int): int - **static** getServizi(Connection con, idOfferta: int): List<Servizio>
Pre-condizioni	Context OfferteDAO::doRetrieveAll(offset, limit) Pre: Context OfferteDAO::doRetrieveById(id) Pre: id!= null Context OfferteDAO::doRetrieveByServizio(servizio, offset, limit) Pre: servizio != null

	Context OfferteDAO::doRetrieveByNomeOrDescrizione(against, offset, limit) Pre: against != null Context OfferteDAO::doRetrieveByDestinazione(against, offset, limit) Pre: against != null Context OfferteDAO::doSave(offerta) Pre: offerta != null Context OfferteDAO::doUpdate(offerta) Pre: offerta != null Context OfferteDAO::doDelete(id) Pre: id != null Context OfferteDAO::getServizi(con, idOfferta) Pre: idOfferta != null
Post-condizioni	Context OfferteDAO:: doRetrieveAll(offset, limit) Post: database.offerta Context OfferteDAO:: doRetrieveById(id) Post: offerta -> select(o| offerta.id =id) Context OfferteDAO:: doRetrieveByServizio(servizio, offset, limit) Post: offerta -> select(o| offerta.servizio = servizio) Context OfferteDAO:: doRetrieveByNomeOrDescrizione(against, offset, limit) Post: offerta -> select(o| offerta.against = against) Context OfferteDAO:: doRetrieveByDestinazione(against, offset, limit) Post: offerta -> select(o| offerta.against = against) Context OfferteDAO:: doSave(offerta) Post: database.offerta ->include(select(o| offerta.email =id.getEmail())) Context OfferteDAO:: doUpdate(offerta) Post: Context OfferteDAO:: doDelete(id) Post: offerta -> not include(select(o| offerta.email=email) Context OfferteDAO:: getServizi(con, idOfferta) Post: offerta -> select(o| offerta.servizi = servizi)
Invariante	

Nome Classe	**Servizi**
Descrizione	Questa classe Bean rappresenta l'oggetto Servizi

Signature dei metodi	+ getId(): int + setId(id: int): void + getNome(): String + setNome(nome: String): void + getDescrizione(): String + setDescrizione(descrizione: String): void
Pre-condizioni	
Post-condizioni	
Invariante	

Nome Classe	**ServizioDAO**
Descrizione	Questa classe è un manager che si occupa di interagire con il database. Gestisce le query riguardanti l'Servizio.
Signature dei metodi	+ doRetrieveAll(): List<Servizio> + doRetrieveByNome(nome: String): Servizio + doRetrieveByDescrizione(descrizione: String): Servizio + doSave(id: int): void + doUpdate(id: int): void + doDelete(id: int):void
Pre-condizioni	Context ServizioDAO::doRetrieveAll(): Pre: Context ServizioDAO::doRetrieveByNome(nome): Pre: nome != null Context ServizioDAO::doRetrieveByDescrizione(descrizione) Pre: descrizione != null Context ServizioDAO::doSave(id) Pre: id != null Context ServizioDAO::doUpdate(id) Pre: id != null Context ServizioDAO::doDelete(id) Pre: id != null
Post-condizioni	Context ServizioDAO::doRetrieveAll(): Post: database.servizio Context ServizioDAO::doRetrieveByNome(nome): Post: servizio -> select(o| servizio.nome = nome) Context ServizioDAO::doRetrieveByDescrizione(descrizione) Post: servizio -> select(o| servizio.nome = nome) Context ServizioDAO::doSave(id) Post: database.servizio ->include(select(o| servizio.id =id.getId()))

	Context ServizioDAO::doUpdate(id) Post: Context ServizioDAO::doDelete(id) Post: servizio -> not include(select(o | servizio.id=id)
Invariante	

Nome Classe	**Ordine**
Descrizione	Questa classe Bean rappresenta l'oggetto Ordine
Signature dei metodi	+ getId(): int + setId(id: int): void + getIdutente(): int + getIdofferta(): int + getEmail(): String + setEmail(email: String): void + isStato(): boolean + setStato(stato: boolean): void + getComponenti():ArrayList<ComponentiViaggio> + setComponenti(componenti: ArrayList<ComponentiViaggio>): void
Pre-condizioni	
Post-condizioni	
Invariante	

Nome Classe	**OrdineDAO**
Descrizione	Questa classe è un manager che si occupa di interagire con il database. Gestisce le query riguardanti l'Ordine.
Signature dei metodi	+ doRetrieveAll(offset: int, limit: int): List<Ordine> + doSave(ordine: Ordine): int + doRetrieveByUtente(utente: int): List<Ordine>
Pre-condizioni	Context OrdineDAO::doRetrieveAll(offset, limit) Pre: Context OrdineDAO::doSave(ordine) Pre: ordine != null Context OrdineDAO::doRetrieveByUtente(idUtente) Pre: idUtente != null
Post-condizioni	Context OrdineDAO::doRetrieveAll(offset, limit) Post: database.ordine Context OrdineDAO::doSave(ordine) Post: database.ordine -> select(o | ordine.idUtente = idUtente)

	Context OrdineDAO::doRetrieveByUtente(utente) Post: ordine -> select(o | ordine.idUtente = idUtente)
Invariante	

Nome Classe	**Carrello**
Descrizione	Questa classe Bean rappresenta l'oggetto Carrello
Signature dei metodi	+ OffertaQuantita(Offerta offerta, **int** quantita) + getQuantita(): int + setQuantita(quantita: int): void + getOfferta(): int + getPrezzoTotCent() + getPrezzoTotEuro() - LinkedHashMap<Integer, OffertaQuantita> offerte + Collection<OffertaQuantita> getProdotti() + get(prodId: int): OffertaQuantita + put(Offerta, **int** quantita): void + remove(prodId: int): OffertaQuantita + getPrezzoTotCent(): long + getPrezzoTotEuro(): strin
Pre-condizioni	
Post-condizioni	
Invariante	

Nome Classe	**CarrelloDAO**
Descrizione	Questa classe è un manager che si occupa di interagire con il database. Gestisce le query riguardanti il Carrello.
Signature dei metodi	+ doRetrieveAll(offset: int, limit: int): List<Carrello> + doReciveByUtente(id:int): List<Carrello> + doDelete(idOfferta): boolean
Pre-condizioni	Context CarrelloDAO::doRetrieveAll(offset, limit) Pre: Context CarrelloDAO:: doReciveByUtente(id) Pre: id != null Context CarrelloDAO:: doDelete(idOfferta) Pre: idOfferta != null
Post-condizioni	Context CarrelloDAO::doRetrieveAll(offset, limit) Post: database.carrello Context CarrelloDAO:: doRetrieveByUtente(utente) Post: ordine -> select(o | carrello.idUtente = idUtente)

	Context CarrelloDAO:: doDelete(idOfferta) Post: carrello -> not include(select(c| accrello.idOfferta=idOfferta)
Invariante	

Nome Classe	**ComponentiViaggio**
Descrizione	Questa classe Bean rappresenta l'oggetto ComponentiViaggio
Signature dei metodi	+ getId() + setId(intid) + getNome() + setNome(String nome) + getCognome(): String + setCognome(String cognome) + getData_nascita() + setData_nascita(String data_nascita)
Pre-condizioni	
Post-condizioni	
Invariante	

Nome Classe	**ComponentiViaggioDAO**
Descrizione	Questa classe è un manager che si occupa di interagire con il database. Gestisce le query riguardanti i ComponentiViaggio.
Signature dei metodi	+ doRetrieveByOrdine(ordine: int, offset: int, limit: int): List<ComponentiViaggio> + getComponenti(con: Connection, idordine: int): List<ComponentiViaggio>
Pre-condizioni	Context ComponentiViaggioDAO:: doRetrieveByOrdine(ordine) Pre: ordine != null Context ComponentiViaggioDAO:: getComponenti(con, idOrdine) Pre: idOrdine != null
Post-condizioni	Context ComponentiViaggioDAO:: doRetrieveByOrdine(ordine) Post: componentiViaggio -> select(c| componentiViaggio.idOrdine = idOrdine) Context ComponentiViaggioDAO:: getComponenti(con, idordine) Post: componentiViaggio -> select(c| componentiViaggio.idOrdine = idOrdine)
Invariante	

4. Design Pattern