Università degli Studi di Salerno

Corso di Ingegneria del Software

Object Design Document

Partecipanti:

Nome	**Matricola**
Carlo Lerro	0512105440
Alfonso Fiorentino	0512102994
Antonio Botticchio	0512105458

Revision History

Data	***Versione***	***Cambiamenti***	***Autori***
17/12/2019	0.1	Aggiunta sezione 1	Antonio Botticchio
17/01/2020	0.2	Aggiunta sezione 2	Carlo Lerro
17/01/2020	0.3	Revisione	Alfonso Fiorentino
18/01/2020	0.4	Aggiunta sezione 1.4	Carlo Lerro
18/01/2020	0.5	Aggiunto Packages	Antonio Botticchio

Introduction

1.1. Object Design trade-offs

Comprensibilità vs Tempo:
Il codice del sistema deve essere comprensibile, in modo da facilitare la fase di testing ed eventuali future modifiche da apportare. Al fine di rispettare queste linee guida il codice sarà integrato da commenti volti a migliorarne la leggibilità; tuttavia questo richiederà una maggiore quantità di tempo necessario per lo sviluppo del nostro progetto.

Prestazioni vs Costi:
Dal momento che il budget allocato è spendibile principalmente in risorse umane e non consente l'acquisto di tecnologie(tranne per l'acquisto Balsamiq ,usato per i mock-up) proprietarie specifiche verranno utilizzati template open source e componenti hardware di nostra proprietà.

Interfaccia vs Usabilità:
Verrà realizzata un'interfaccia grafica chiara e concisa, usando form e pulsanti predefiniti che hanno lo scopo di rendere semplice l'utilizzo del sistema da parte dell'utente finale.

Sicurezza vs Efficienza:
La sicurezza rappresenta uno degli aspetti principali del sistema. Tuttavia, a causa di tempi di sviluppo molto limitati, ci limiteremo ad implementare sistemi di sicurezza basati su email e password.

1.2. Interface documentation guidelines

Gli sviluppatori dovranno seguire precise linee guida per la stesura del codice:

Naming Convention:

Per la documentazione delle interfacce bisognerà utilizzare nomi:

- Descrittivi
- Pronunciabili;

- Di lunghezza medio-corta;

Variabili:

- I nomi delle variabili dovranno iniziare con la lettera minuscola, e le parole successive con la lettera maiuscola.
- In ogni riga dovrà esserci un'unica variabile dichiarata, eventualmente allineata con quelle del blocco dichiarativo.
- In determinati casi, è possibile utilizzare il carattere underscore "_"

Metodi:

- I nomi dei metodi dovranno iniziare con la lettera minuscola, e le parole successive con la lettera maiuscola, secondo la "Camel Notation".
- Il nome del metodo sarà costituito da un verbo che ne identifica l'azione seguito da un sostantivo, eventualmente aggettivato.
- Ai metodi verrà aggiunto un commento JavaDoc, il quale deve essere posizionata prima della dichiarazione del metodo, e deve descriverne lo scopo.

Classi Java e pagine JSP:

- I nomi delle classi e delle pagine dovrà iniziare con la lettera maiuscola, così come le parole successive all'interno del nome.
- I nomi delle classi e delle pagine dovrà corrispondere alle informazioni e le funzioni fornite da quest'ultime.
- Le classi saranno strutturate prevedendo rispettivamente:
    1. Dichiarazione della classe pubblica;
    2. Dichiarazione di costanti;
    3. Dichiarazioni di variabili di classe;
    4. Dichiarazioni di variabili di istanza;
    5. Costruttore;
    6. Metodi;

Packages:

1. I nomi dei packages dovranno essere scritti in minuscolo concatenando insieme diversi sostantivi o sigle, separate dal carattere "." .

2. Non saranno ammessi caratteri speciali.

1.3. Definitions, acronyms and abbreviation

Acronimi:

- RAD:Requirements Analysis Document
- SDD: System Design Document
- ODD:Object Design Document

1.4. Design Pattern

1.4.1. DAO Pattern

Il design pattern architetturale DAO permette di astrarre i dettagli implementativi della persistenza dei dati: invece di usare una comunicazione diretta e strettamente accoppiata con il database, questa viene svolta attraverso un'interfaccia DAO; ciò permette di favorire la manutenibilità e la versatilità della gestione dei dati.

Il design pattern DAO ha come scopo:

- Separare le operazioni di accesso ai dati dalle operazioni di business;
- Avere un'interfaccia comune di accesso ai dati.

Fornisce diversi vantaggi, tra cui:

- Manutenibilità
- Facilità di utilizzo
- Riuso del codice

Utilizzo

Il design pattern DAO sarà utilizzato per gestire le operazioni di persistenza, aggiornamento, rimozione e caricamento dei dati da effettuare sul database.

1.4.2. DAO Factory Pattern

Questo pattern utilizza i principi del DAO pattern e del Factory pattern. Il primo permette di astrarre i dettagli implementativi della persistenza dei dati utilizzando un'interfaccia DAO.
Il secondo nasconde il processo di creazione di un oggetto al client e separare la logica di accesso di basso livello dai servizi di alto livello. Quindi il factory restituisce un' interfaccia del DAO che il client vuole, disaccoppiando quindi le diverse implementazioni con il client che l'utilizza.

Nel nostro caso verrà utilizzato per dividere la logica di persistenza utilizzando l'interfaccia DAO e utilizzando il Factory in modo da poter utilizzare diverse implementazioni di un DAO all'occorrenza.

1.4.3. Singleton Pattern

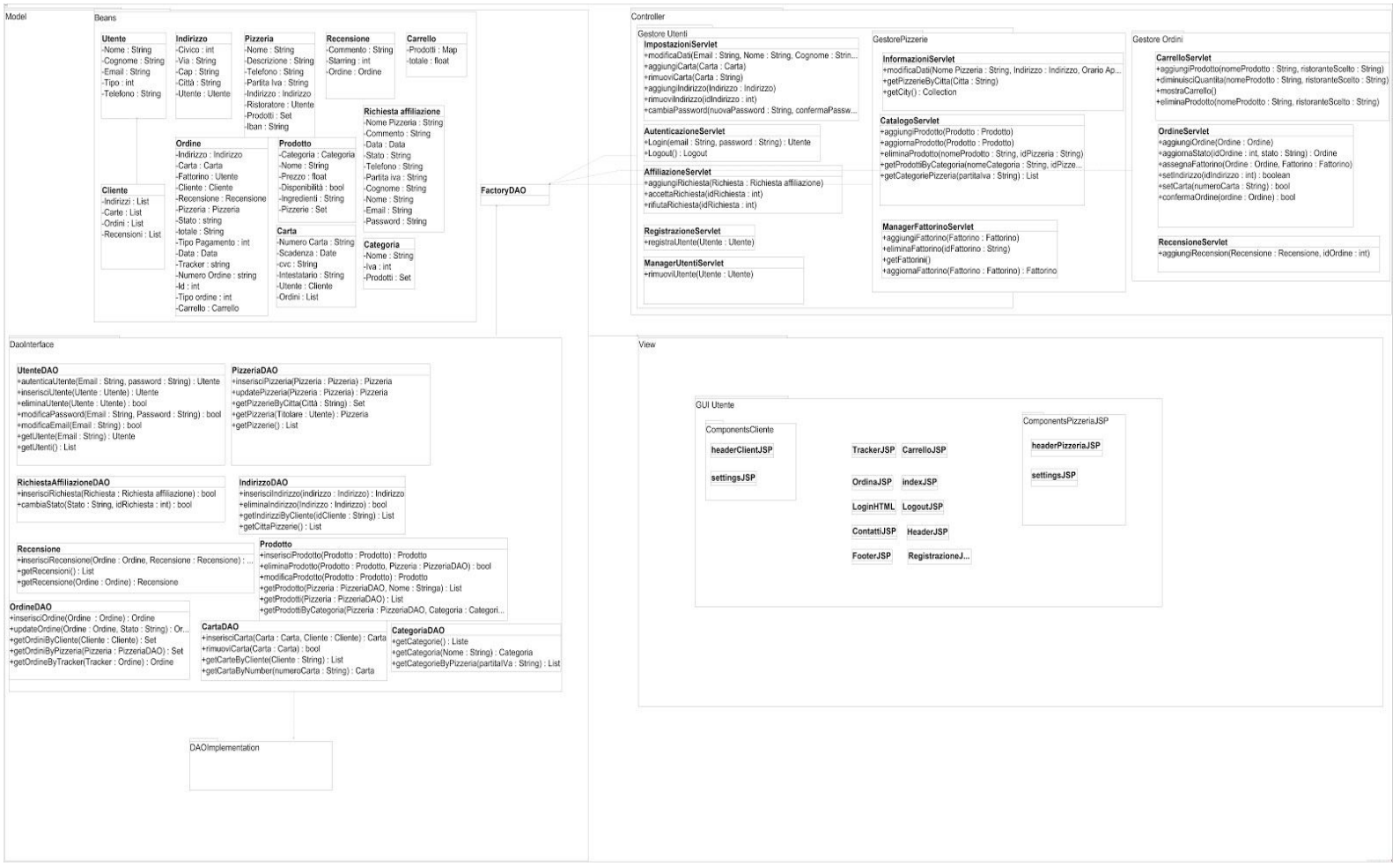
Il Singleton è un design pattern di tipo strutturale che garantisce l'istanziazione di un singolo oggetto della classe di interesse, fornendo un punto di accesso globale all'interno del sistema; lo scope dell'oggetto istanziato è di applicazione e supporta la gestione degli accessi concorrenti ai metodi esposti.

Il design pattern Singleton ha come scopo:
• Avere un accesso controllato all’unica istanza della classe
• Ridurre il numero di oggetti condivisi
• Centralizzare informazioni e comportamenti in un'unica entità condivisa dagli utilizzatori

Il principale vantaggio offerto è:
• Mutua esclusione

Utilizzo
Il design pattern verrà utilizzato per gestire l’accesso alla classe DriverManagerConnectionPool che si occupa dell’accesso al DB..
Qualsiasi classe che intende utilizzare i metodi di questa classe dovrà farlo attraverso l’utilizzo dell’unica istanza della classe presente all’interno del sistema.

Packages


Model
Beans
Utente
-Nome : String
-Cognome : String
-Email : String
-Tipo : int
-Telefono : String
Indirizzo
-Civico : int
-Via : String
-Cap : String
-Città : String
-Utente : Utente
Pizzeria
-Nome : String
-Descrizione : String
-Telefono : String
-Partita Iva : String
-Indirizzo : Indirizzo
-Ristoratore : Utente
-Prodotti : Set
-Iban : String
Recensione
-Commento : String
-Starring : int
-Ordine : Ordine
Carrello
-Prodotti : Map
-totale : float
Richiesta affiliazione
-Nome Pizzeria : String
-Commento : String
-Data : Data
-Stato : String
-Telefono : String
-Partita iva : String
-Cognome : String
-Nome : String
-Email : String
-Password : String
Ordine
-Indirizzo : Indirizzo
-Carta : Carta
-Fattorino : Utente
-Cliente : Cliente
-Recensione : Recensione
-Pizzeria : Pizzeria
-Stato : string
-totale : String
-Tipo Pagamento : int
-Data : Data
-Tracker : string
-Numero Ordine : string
-Id : int
-Tipo ordine : int
-Carrello : Carrello
Prodotto
-Categoria : Categoria
-Nome : String
-Prezzo : float
-Disponibilità : bool
-Ingredienti : String
-Pizzerie : Set
Carta
-Numero Carta : String
-Scadenza : Date
-cvc : String
-Intestatario : String
-Utente : Cliente
-Ordini : List
Categoria
-Nome : String
-Iva : int
-Prodotti : Set
Cliente
-Indirizzi : List
-Carte : List
-Ordini : List
-Recensioni : List
FactoryDAO
Controller
Gestore Utenti
ImpostazioniServlet
+modificaDati(Email : String, Nome : String, Cognome : Strin...
+aggiungiCarta(Carta : Carta)
+rimuoviCarta(Carta : String)
+aggiungiIndirizzo(Indirizzo : Indirizzo)
+rimuoviIndirizzo(idIndirizzo : int)
+cambiaPassword(nuovaPassword : String, confermaPassw...
AutenticazioneServlet
+Login(email : String, password : String) : Utente
+Logout() : Logout
AffiliazioneServlet
+aggiungiRichiesta(Richiesta : Richiesta affiliazione)
+accettaRichiesta(idRichiesta : int)
+rifiutaRichiesta(idRichiesta : int)
RegistrazioneServlet
+registraUtente(Utente : Utente)
ManagerUtentiServlet
+rimuoviUtente(Utente : Utente)
GestorePizzerie
InformazioniServlet
+modificaDati(Nome Pizzeria : String, Indirizzo : Indirizzo, Orario Ap...
+getPizzerieByCitta(Citta : String)
+getCity() : Collection
CatalogoServlet
+aggiungiProdotto(Prodotto : Prodotto)
+aggiornaProdotto(Prodotto : Prodotto)
+eliminaProdotto(nomeProdotto : String, idPizzeria : String)
+getProdottiByCategoria(nomeCategoria : String, idPizze...
+getCategoriePizzeria(partitaIva : String) : List
ManagerFattorinoServlet
+aggiungiFattorino(Fattorino : Fattorino)
+eliminaFattorino(idFattorino : String)
+getFattorini()
+aggiornaFattorino(Fattorino : Fattorino) : Fattorino
Gestore Ordini
CarrelloServlet
+aggiungiProdotto(nomeProdotto : String, ristoranteScelto : String)
+diminuisciQuantita(nomeProdotto : String, ristoranteScelto : String)
+mostraCarrello()
+eliminaProdotto(nomeProdotto : String, ristoranteScelto : String)
OrdineServlet
+aggiungiOrdine(Ordine : Ordine)
+aggiornaStato(idOrdine : int, stato : String) : Ordine
+assegnaFattorino(Ordine : Ordine, Fattorino : Fattorino)
+setIndirizzo(idIndirizzo : int) : boolean
+setCarta(numeroCarta : String) : bool
+confermaOrdine(ordine : Ordine) : bool
RecensioneServlet
+aggiungiRecensioni(Recensione : Recensione, idOrdine : int)
DaoInterface
UtenteDAO
+autenticaUtente(Email : String, password : String) : Utente
+inserisciUtente(Utente : Utente) : Utente
+eliminaUtente(Utente : Utente) : bool
+modificaPassword(Email : String, Password : String) : bool
+modificaEmail(Email : String) : bool
+getUtente(Email : String) : Utente
+getUtenti() : List
PizzeriaDAO
+inserisciPizzeria(Pizzeria : Pizzeria) : Pizzeria
+updatePizzeria(Pizzeria : Pizzeria) : Pizzeria
+getPizzerieByCitta(Città : String) : Set
+getPizzeria(Titolare : Utente) : Pizzeria
+getPizzerie() : List
RichiestaAffiliazioneDAO
+inserisciRichiesta(Richiesta : Richiesta affiliazione) : bool
+cambiaStato(Stato : String, idRichiesta : int) : bool
IndirizzoDAO
+inserisciIndirizzo(indirizzo : Indirizzo) : Indirizzo
+eliminaIndirizzo(Indirizzo : Indirizzo) : bool
+getIndirizziByCliente(idCliente : String) : List
+getCittaPizzerie() : List
Recensione
+inserisciRecensione(Ordine : Ordine, Recensione : Recensione) : ...
+getRecensioni() : List
+getRecensione(Ordine : Ordine) : Recensione
Prodotto
+inserisciProdotto(Prodotto : Prodotto) : Prodotto
+eliminaProdotto(Prodotto : Prodotto, Pizzeria : PizzeriaDAO) : bool
+modificaProdotto(Prodotto : Prodotto) : Prodotto
+getProdotto(Pizzeria : PizzeriaDAO, Nome : Stringa) : List
+getProdotti(Pizzeria : PizzeriaDAO) : List
+getProdottiByCategoria(Pizzeria : PizzeriaDAO, Categoria : Categori...
OrdineDAO
+inserisciOrdine(Ordine : Ordine) : Ordine
+updateOrdine(Ordine : Ordine, Stato : String) : Or...
+getOrdiniByCliente(Cliente : Cliente) : Set
+getOrdiniByPizzeria(Pizzeria : PizzeriaDAO) : Set
+getOrdineByTracker(Tracker : Ordine) : Ordine
CartaDAO
+inserisciCarta(Carta : Carta, Cliente : Cliente) : Carta
+rimuoviCarta(Carta : Carta) : bool
+getCarteByCliente(Cliente : String) : List
+getCartaByNumber(numeroCarta : String) : Carta
CategoriaDAO
+getCategorie() : Liste
+getCategoria(Nome : String) : Categoria
+getCategorieByPizzeria(partitaIVa : String) : List
DAOImplementation
View
GUI Utente
ComponentsCliente
headerClientJSP
settingsJSP
TrackerJSP
CarrelloJSP
OrdinaJSP
indexJSP
LoginHTML
LogoutJSP
ContattiJSP
HeaderJSP
FooterJSP
RegistrazioneJ...
ComponentsPizzeriaJSP
headerPizzeriaJSP
settingsJSP