Laurea Magistrale in informatica-Università di Salerno
Corso di *Gestione dei Progetti Software*- Prof. Andrea DE LUCIA

Object Design Document

Riferimento	
Versione	1.0
Data	09/03/2021
Destinatario	Prof. Andrea DE LUCIA
Presentato da	Vincenzo Raia (VR) - 0512106140 Raffaele Scarpa (RS) - 0512105708 Giuseppe Pepe (GP) - 0512105930

Revision History

Data	Versione	Descrizione	Autori
16/01/2021	0.1	Prima stesura	VR
09/03/2021	1.0	Revisione finale	VR

Sommario

1. Introduzione

1.1 Object Design Trade-offs

Il seguente documento 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 la signature dei sottosistemi definiti nel System Design.

Inoltre sono specificati i trade-off e le linee guida.

Comprensibilità vs 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 che ne semplifichino la comprensione. Questa caratteristica aggiungerà un incremento di tempo allo sviluppo del nostro progetto.

Interfaccia vs Usabilità:

L'interfaccia grafica è stata realizzata in modo da essere molto semplice, chiara e concisa.

Fa uso di form e pulsanti disposti in maniera da rendere semplice l'utilizzo del sistema da parte dell'utente finale.

Sicurezza vs Efficienza:

La sicurezza, come descritto nei requisiti non funzionali del RAD, rappresenta uno degli aspetti importanti del sistema. Tuttavia, dati i tempi di sviluppo molto limitati, ci limiteremo ad implementare sistemi di sicurezza basati su username e password degli utenti.

1.2 Linee Guida per la Documentazione delle Interfacce

Gli sviluppatori dovranno seguire alcune linee guida per la scrittura del codice:

Naming Convention

È buona norma utilizzare nomi:

1. Descrittivi
2. Pronunciabili
3. Di uso comune
4. Lunghezza medio-corta
5. Non abbreviati
6. Evitando la notazione ungherese
7. Utilizzando solo caratteri consentiti (a-z, A-Z, 0-9)

Variabili

I nomi delle variabili devono cominciare con una lettera minuscola, e le parole seguenti con la maiuscola. Quest'ultime devono essere dichiarate ad inizio blocco, solamente una per riga e devono essere tutte allineate per facilitare la leggibilità. (Esempio: libroVenduto)

E' inoltre possibile, in alcuni casi, utilizzare il carattere underscore "_", ad esempio quando utilizziamo delle variabili costanti oppure quando vengono utilizzate delle proprietà statiche. (Esempio: TIPO_UTENTE)

Metodi

I nomi dei metodi devono cominciare con una lettera minuscola, e le parole seguenti con la lettera maiuscola. 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(). Le variabili dei metodi devono essere dichiarate appena prima del loro utilizzo e devono servire per facilitarne la leggibilità.

Esistono però casi particolari come ad esempio nell'implementazione dei model, dove viene utilizzata l'interfaccia CRUD.

Esempio: getId(), setId()

I commenti dei metodi devono essere raggruppati in base alla loro funzionalità, la descrizione dei metodi deve apparire prima di ogni dichiarazione di metodo, e deve descriverne lo scopo.

Deve includere anche informazioni sugli argomenti, sul valore di ritorno e, se applicabile, sulle eccezioni.

Classi e pagine

I nomi delle classi devono cominciare con una lettera maiuscola, e anche le parole seguenti all'interno del nome devono cominciare con una lettera maiuscola. I nomi di queste ultime devono fornire informazioni sul loro scopo.

I nomi delle pagine devono cominciare con una lettera minuscola. Tutte le parole al loro interno sono scritte in minuscolo e separate dal carattere '-'.

I nomi delle servlet sono analoghi a quelli delle classi, con l'aggiunta della parola "Servlet" come ultima parola.

Esempio: ManagerAutenticazione.java, registra-utente.jsp, LoginServlet.java

Ogni file sorgente java contiene una singola classe e dev'essere strutturato in un determinato modo:

- Una breve introduzione alla classe
    - L'introduzione indica: l'autore, la versione e la data.
      Esempio:

```
/**
*
* @author nome dell'autore
* @version numero di versione della classe
* @since data dell'implementazione
*/
```

- L'istruzione include che permette di importare all'interno della classe gli altri oggetti che la classe utilizza.
- La dichiarazione di classe 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.

1.3 Definizioni ,acronimi e abbreviazioni

Acronimi

- RAD: Requirements Analysis Document
- SDD: System Design Document
- ODD: Object Design Document
- CRUD: Create Read Update Delete

Abbreviazioni

- DB: DataBase

1.4 Riferimenti

- Documento SDD del progetto BookStore
- Documento RAD del progetto BookStore

2.Packages

La gestione del nostro sistema è suddivisa in tre livelli (three-tier):

● Presentation layer

● Application layer

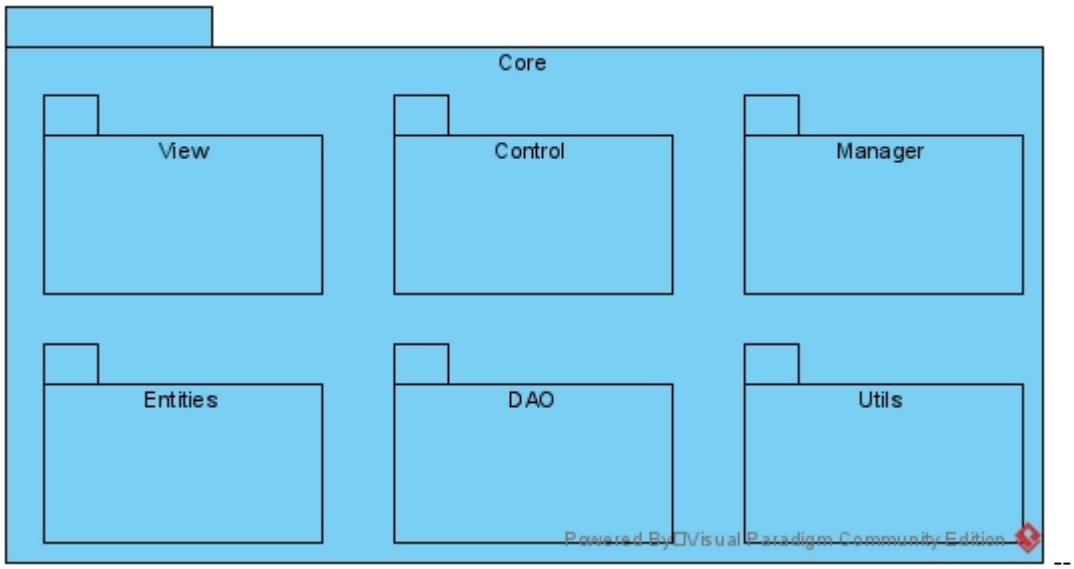
● Storage layer

Il package BookStore contiene sottopackage che a loro volta inglobano classi atte alla gestione delle richieste utente. Le classi contenute nel package svolgono il ruolo di gestore logico del sistema.

Presentation layer	**Include tutte le interfacce grafiche e in generale i boundary objects, come le form con cui interagisce l'utente. L'interfaccia verso l'utente è rappresentata da un Web server e da eventuali contenuti statici (es. pagine HTML).**
Application layer	Include tutti gli oggetti relativi al controllo e all'elaborazione dei dati. Questo avviene interrogando il database tramite lo storage layer per generare contenuti dinamici e accedere a dati persistenti. Si occupa di varie gestioni quali: • Gestione utente • Gestione carrello • Gestione libro • Gestione carrello • Gestione categoria

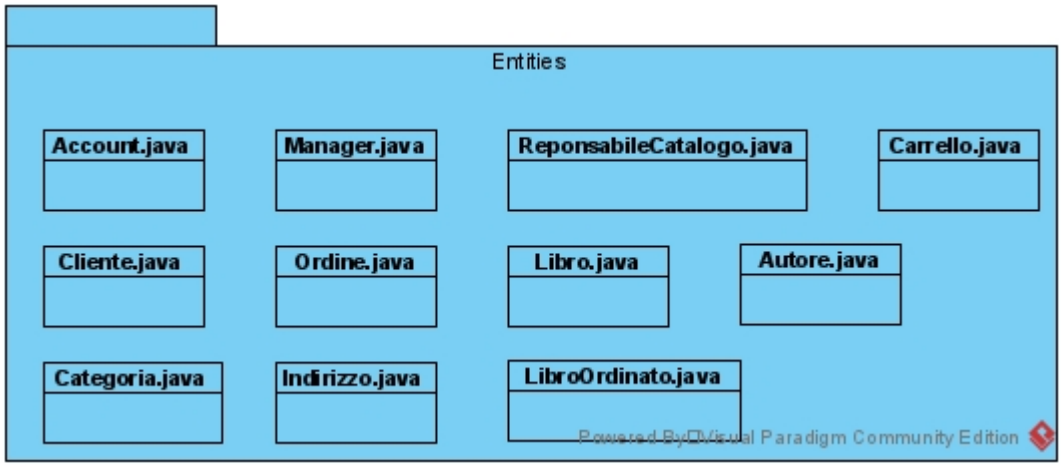
	• Gestione libro
Storage layer	Ha il compito di effettuare memorizzazione, il recupero e l'interrogazione degli oggetti persistenti. I dati, i quali possono essere acceduti dall'application layer, sono depositati in maniera persistente su un database tramite DBMS.

2.1 Packages core



--

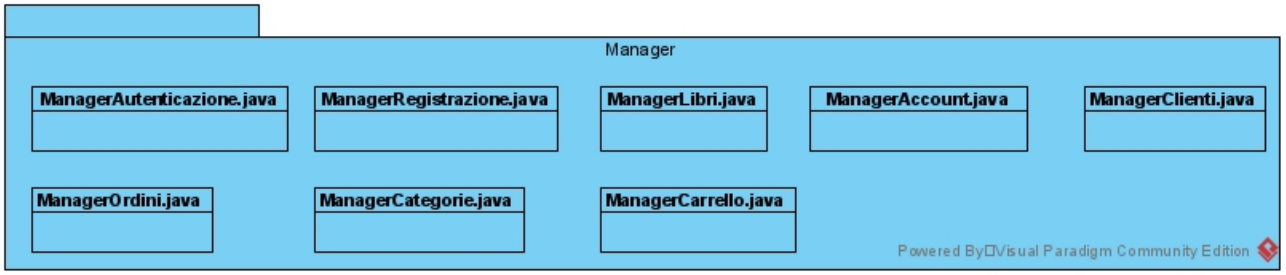
2.1.1 Packages Entitites



CLASSE	DESCRIZIONE
Account.java	Descrive una qualsiasi utente che fa uso del sistema.
Manager.java	Descrive un amministratore del sistema.
ResponsabileCatalogo.java	Descrive un responsabile catalogo del sistema.
Carrello.java	Descrive il carrello del cliente presente nel sistema
Cliente.java	Descrive un cliente del sistema
Libro.java	Descrive un libro presente nel sistema
Categoria.java	Descrive una categoria presente nel sistema
Ordine.java	Descrive un ordine effettuato da un cliente
LibroOrdinato.java	Descrive un libro acquistato da un cliente

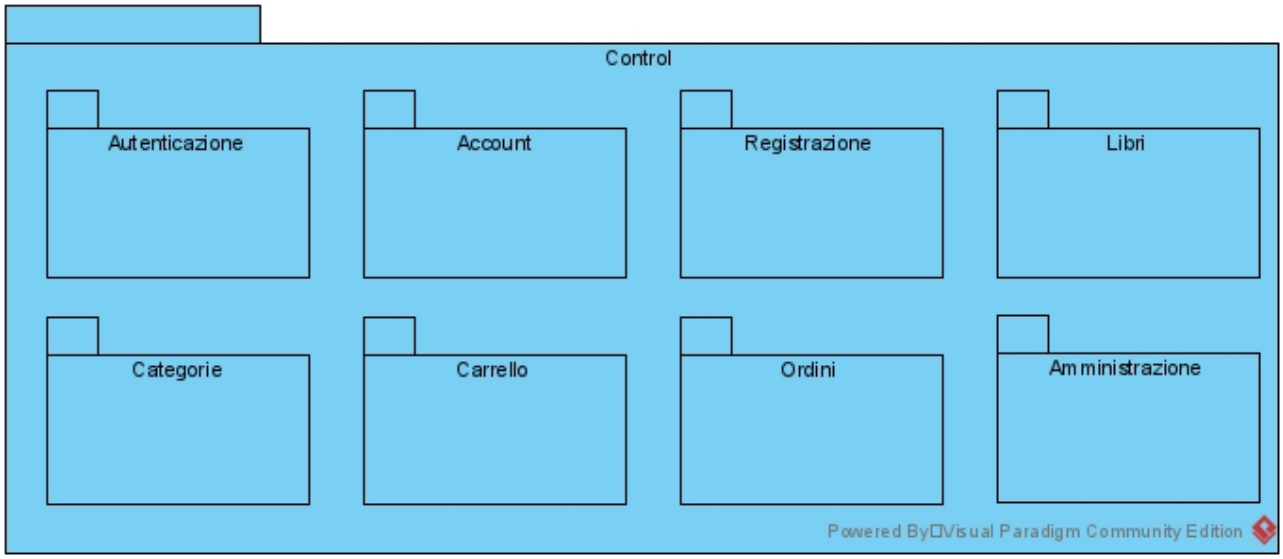
Autore.java	Descrive un autore di un libro presente nel sistema

2.1.2 Packages Manager

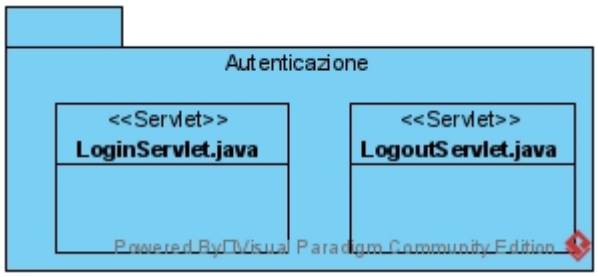


CLASSE	DESCRIZIONE
ManagerAutenticazione.java	Si tratta di una classe che funge da interfaccia del sottosistema che gestisce l'autenticazione.
ManagerAccount.java	Si tratta di una classe che funge da interfaccia del sottosistema che gestisce gli account.
ManagerLibri.java	Si tratta di una classe che funge da interfaccia del sottosistema che gestisce i libri.
ManagerRegistrazione.java	Si tratta di una classe che funge da interfaccia del sottosistema che gestisce la registrazione
ManagerClienti.java	Si tratta di una classe che funge da interfaccia del sottosistema che gestisce i clienti.
ManagerCategorie.java	Si tratta di una classe che funge da interfaccia del sottosistema che gestisce le categorie.
ManagerOrdini.java	Si tratta di una classe che funge da interfaccia del sottosistema che gestisce gli ordini.
ManagerCarrello.java	Si tratta di una classe che funge da interfaccia del sottosistema che gestisce il carrello.

2.1.3 Packages Control

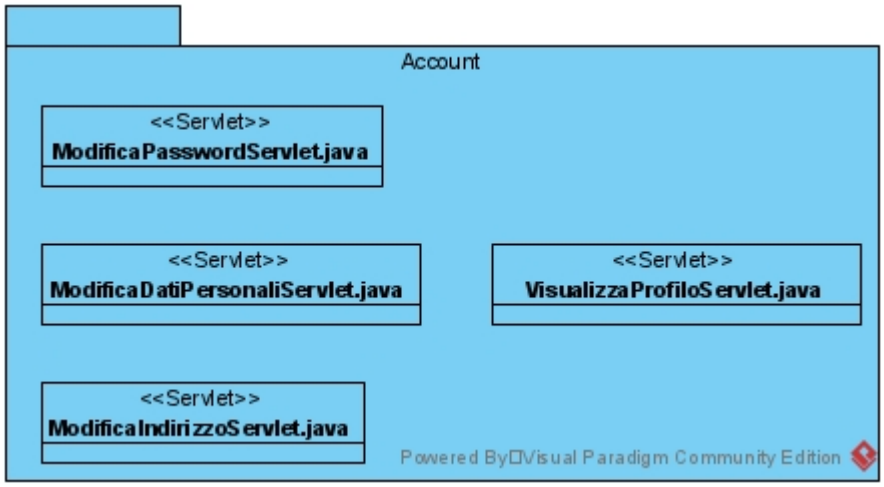


2.1.3.0 Autenticazione



CLASSE	DESCRIZIONE
LoginServlet.java	Gestisce il login degli utenti.
LogoutServlet.java	Gestisce il logout degli utenti.

2.1.3.1 Account

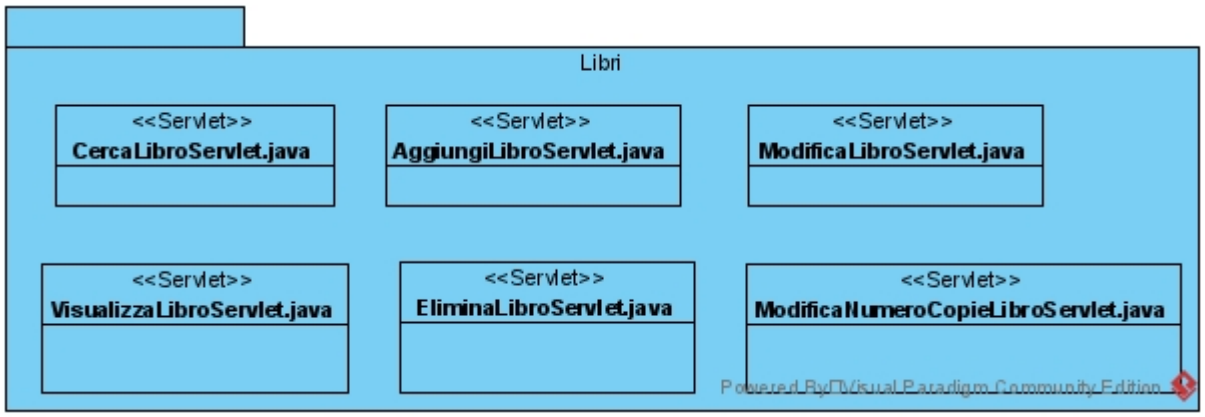


CLASSE	DESCRIZIONE
ModificaPassowrdServlet.java	Permette di modificare la password.
ModificaDatiPersonaliServlet.java	Permette di modificare i dati personali.
VisualizzaProfiloServlet.java	Permette di visualizzare le informazioni personali.
ModificaIndirizzoServlet.java	Permette di modificare l'indirizzo del cliente

2.1.3.2 Registrazione

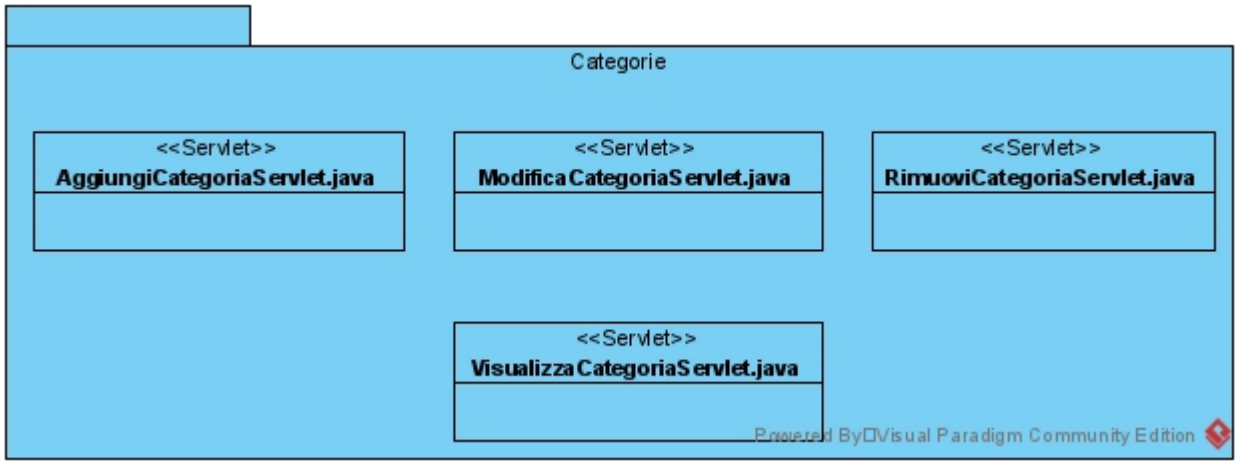
CLASSE	DESCRIZIONE
RegistrazioneClienteServlet.java	Permette di registrare un nuovo cliente
ConfermaMailRegistrazioneServlet.java	Permette di confermare la mail di un nuovo cliente

2.1.3.3 Libri



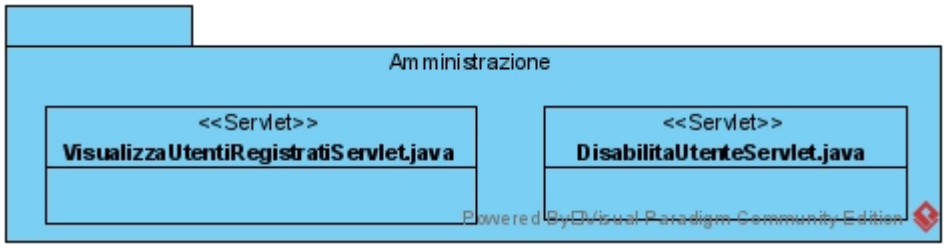
CLASSE	DESCRIZIONE
CercaLibroServlet.java	Permette la ricerca di un libro.
AggiuntaLibroServlet.java	Permette l'aggiunta di un libro.
EliminaLibroServlet.java	Permette l'eliminazione di un libro.
ModificaLibroServlet.java	Permette la modifica di un libro.
VisualizzaLibroServlet.java	Permette di visualizzare un libro e tutte le sue informazioni
ModificaNumeroCopieLibroServlet.java	Permette la modifica del numero copie disponibili di un libro

2.1.3.4 Categorie



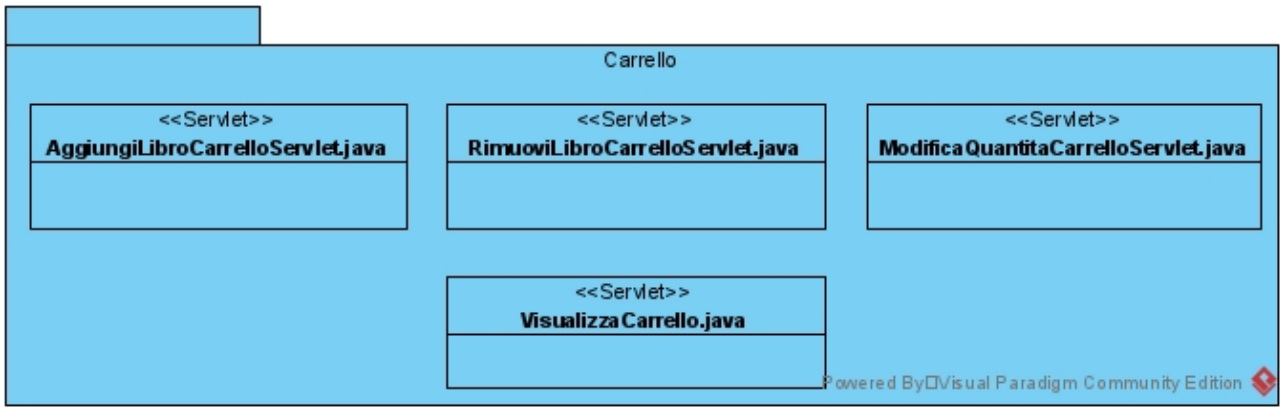
CLASSE	DESCRIZIONE
AggiuntaCategoriaServlet.java	Permette l'aggiunta di una categoria.
EliminaCategoriaServlet.java	Permette l'eliminazione di una categoria.
ModificaCategoriaServlet.java	Permette la modifica di una categoria.
VisualizzaCategoriaServlet.java	Permette di visualizzare una categoria con i relativi libri.

2.1.3.5 Amministrazione



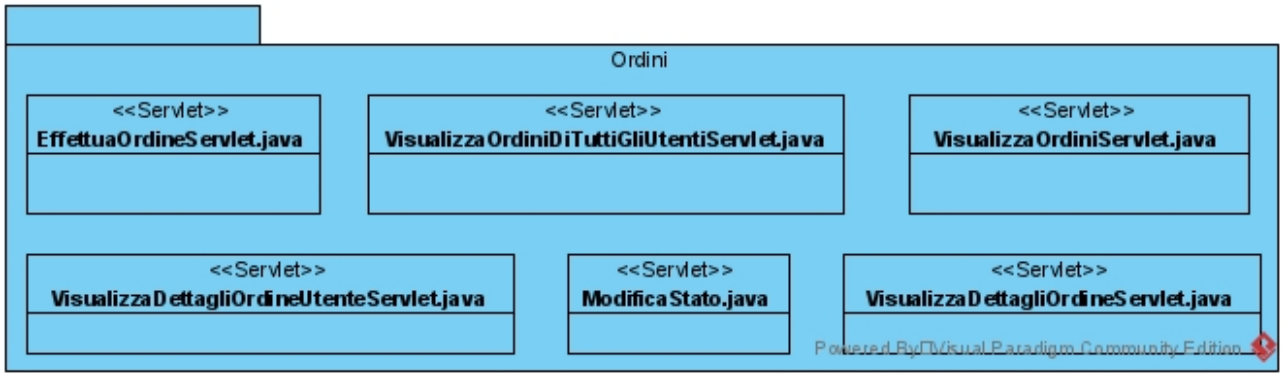
CLASSE	DESCRIZIONE
VisualizzaUtentiRegistratiServlet.java	Permette all'amministratore di visualizzare tutti gli utenti registrati nel sistema.
DisabilitaUtente.java	Permette di disabilitare un cliente

2.1.3.6 Carrello



CLASSE	DESCRIZIONE
AggiungiLibroCarrelloServlet.java	Permette di aggiungere un libro nel carrello.
RimuoviLibroCarrelloServlet.java	Permette di rimuovere un libro nel carrello
VisualizzaCarrello.java	Permette di visualizzare i libri presenti nel carrello
ModificaQuantitaCarrelloServlet.java	Permette di modificare la quantità di un libro nel carrello

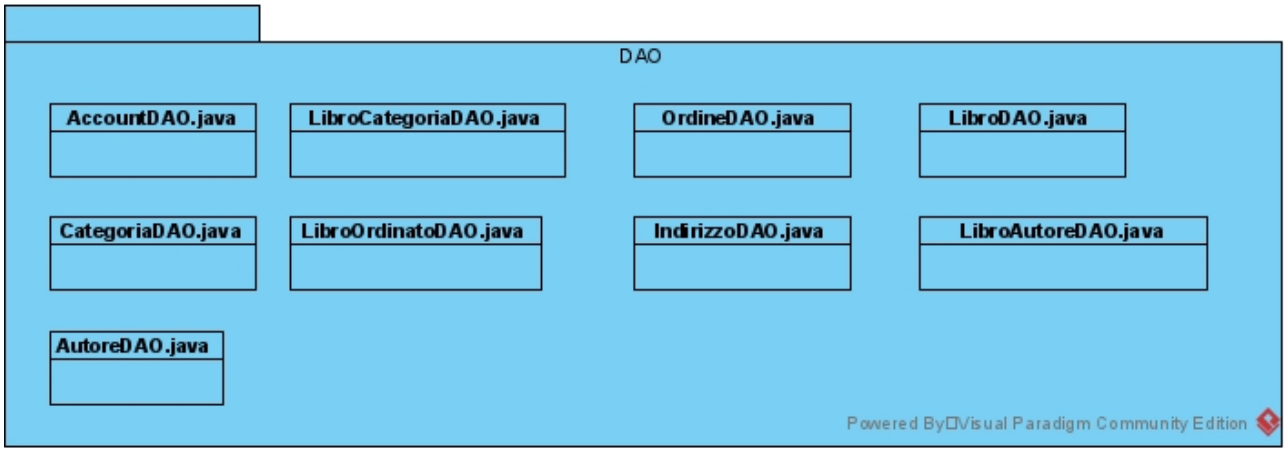
2.1.3.7 Ordini



CLASSE	DESCRIZIONE
EffettuaOrdineServlet.java	Permette di effettuare un ordine
VisualizzaOrdinitDiTuttiGliUtentiServlet.java	Permette all'amministratore di visualizzare gli ordini tutti i clienti
VisualizzaOrdiniServlet.java	Permette di visualizzare tutti gli ordini effettuati

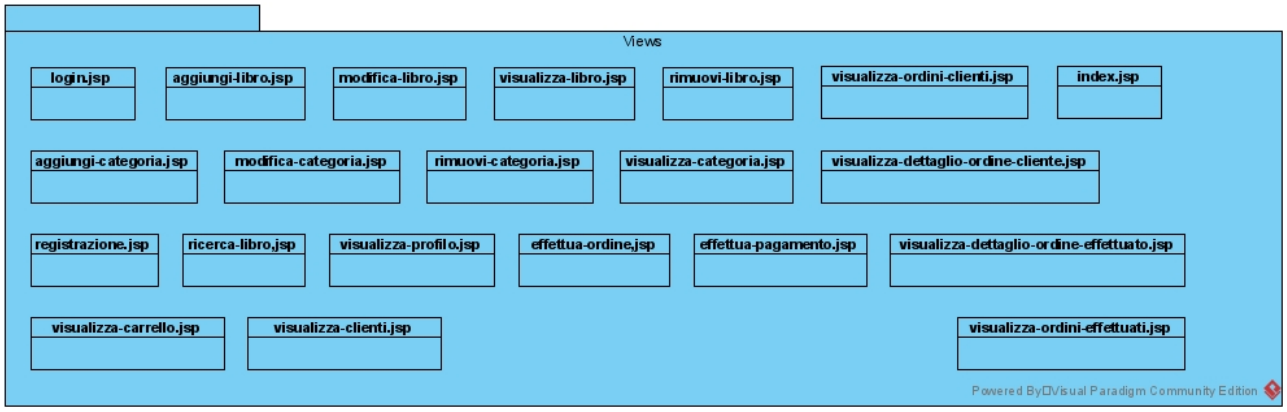
VisualizzaDettagliOrdineUtenteServlet.java	Permette all'amministratore di visualizzare di un ordine di un cliente
VisualizzaDettagliOrdineServlet.java	Permette al cliente di visualizzare i dettagli di un ordine
ModificaStato.java	Permette ad un amministratore di modificare uno stato di una ordine

2.1.4 Pakcages DAO



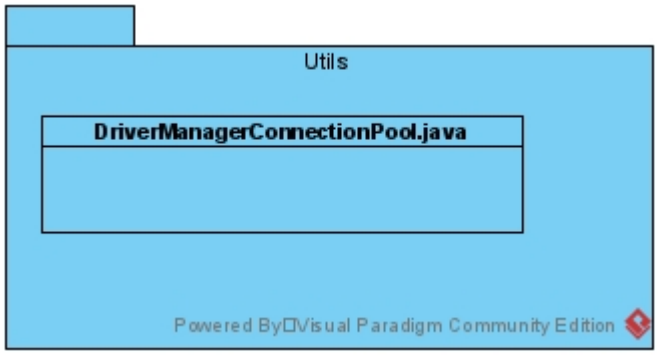
CLASSE	DESCRIZIONE
AccountDAO.java	Permette di interagire con la tabella Account all'interno del database.
OrdineDAO.java	Permette di interagire con la tabella Ordine all'interno del database.
LibroDAO.java	Permette di interagire con la tabella Libro all'interno del database.
IndirizzoDAO.java	Permette di interagire con la tabella Indirizzo all'interno del database.
CategoriaDAO.java	Permette di interagire con la tabella Categoria all'interno del database.
LibroOrdinato.java	Permette di interagire con la tabella LibroOrdinato all'interno del database.
AutoreDAO.java	Permette di interagire con la tabella Autore all'interno del database.
LibroCategoriaDAO.java	Permette di interagire con la tabella Libro_Categoria all'interno del database.
LibroAutoreDAO.java	Permette di interagire con la tabella Libro_Autore all'interno del database.

2.1.5 Packages View



CLASSE	DESCRIZIONE
login.jsp	Visualizza la pagina che permette di effettuare il login.
index.jsp	Visualizza la pagina iniziale del sistema.
conferma-registrazione.jsp	Visualizza la pagina che conferma l'iscrizione al sito.
registrazione.jsp	Visualizza la pagina che permette di registrare una persona al sito.
aggiungi-libro.jsp	Visualizza la pagina che permette di aggiungere un libro al sistema.
modifica-libro.jsp	Visualizza la pagina che permette di modificare un libro nel sistema.
rimuovi-libro.jsp	Visualizza la pagina che permette di rimuovere un libro dal sistema.
ricerca-libro.jsp	Visualizza la pagina che permette di mostrare i risultati di una ricerca di libri in base ad una parola chiave nel sistema.
aggiungi-categoria.jsp	Visualizza la pagina che permette di aggiungere una categoria al sistema.
modifica-categoria.jsp	Visualizza la pagina che permette di modificare una categoria al sistema.
rimuovi-categoria.jsp	Visualizza la pagina che permette di rimuovere una categoria dal sistema.
visualizza-libro.jsp	Visualizza la pagina che permette di visualizzare i dettagli di un libro.
visualizza-categoria.jsp	Visualizza la pagina che permette di visualizzare tutti i libri associati ad un determinata categoria
visualizza-ordini-clienti.jsp	Visualizza la pagina che permette di visualizzare gli ordini effettuati dai clienti
visualizza-carrello.jsp	Visualizza la pagina che permette di mostrare i libri aggiunti al carrello
visualizza-ordini-effettuati.jsp	Visualizza la pagina che permette ad un cliente di visualizzare i propri ordini effettuati
visualizza-dettaglio-ordine-effettuato.jsp	Visualizza la pagina che permette ad un cliente di visualizzare i dettagli di un ordine da lui effettuato
visualizza-dettaglio-ordine-cliente.jsp	Visualizza la pagina che permette ad un amministratore di visualizzare i dettagli di un ordine effettuato da un cliente
visualizza-profilo.jsp	Visualizza la pagina che permette di mostrare tutte le informazione di un utente
effettua-ordine.jsp	Visualizza la pagina che permette di effettuare un ordine da un cliente
effettua-pagamento.jsp	Visualizza la pagina che contiene un form dove inserire i dati di pagamento di un cliente
visualizza-clienti.jsp	Visualizza tutit gli utenti registrati alla piattaforma

2.1.6 Packages Utils



CLASSE	DESCRIZIONE
DriverManagerConnectionPool.java	Permette al sistema di collegarsi al database e di lanciare richieste al database

3. Interfaccia delle classi

3.1 Package Manager

3.1.0 Manager Autenticazione

Si tratta di una classe che funge da interfaccia del sottosistema che gestisce l'autenticazione

public Account login(String email,String password)	Questo metodo permette di effettuare l'accesso
public boolean logout (HttpSession session)	Questo metodo permette di effettuare il logout

3.1.1 Manager Account

public boolean modificaPassword(String username, String newPassword, String oldPassword)	Questo metodo permette di modificare la password corrente dell'utente e ritorna un valore booleano
public boolean modificaDatiPersonali(String email, String username,String password, String nome, String cognome)	Questo metodo permette di modificare i dati personali di un utente del sistema
public boolean modificaIndirizzo(String via, String comune,String provincia,String noteCorriere,int cap,String username,String password)	Questo metodo permette di modificare i dati di un indirizzo di un cliente
public Cliente recuperaCliente(Account account)	Questo metodo permette di recuperare le info di un cliente
public Account recuperaAccount(String username)	Questo metodo permette di recuperare le info di un account

3.1.2 Manager Registrazione

public boolean registraCliente(String nome, String cognome, String password, String username,String email, Indirizzo indirizzoCliente)	Questo metodo permette di aggiungere un cliente al sistema

public boolean confermaRegistrazione(String username,String email)	Questo metodo permette di verificare l'esistenza di una mail e di confermare la registrazione al sistema.

3.1.3 Manager Libri

public Collection<Libro> cercaLibro(String parolaChiave)	Questo metodo permette la ricerca di un libro all'interno del sistema
public boolean aggiuntaLibro(String isbn, String titolo, String trama, Calendar dataPubblicazione, Float prezzo, double quantita, String coperina,boolean disabilitato, List<Autore> autori)	Questo metodo permette l'aggiunta di un libro all'interno del database di libri.
public boolean eliminaLibro(String isbn)	Questo metodo permette l'eliminazione di un libro dal database di libri
public boolean modificaLibro(String isbn, String titolo, String trama, Calendar dataPubblicazione, Float prezzo, double quantita, String copertina,boolean disabilitato, List<Autore> autori)	Questo metodo permette la modifica di un libro all'interno del database di libri.
public boolean modificaNumeroCopieLibro(String isbn,double quantita)	Questo metodo permette la modifica del numero copie di un libro all'interno del database di libri
public Libro acquisisciLibro(String isbn)	Questo metodo permette di ricevere le informazioni di un libro presente del database
public Collection< Libro > acquisisciTuttiLibri()	Questo metodo permette di ricevere tutte le informazioni di ogni libro presente del database

3.1.4 Manager Categorie

public boolean aggiungiCategoria(String nome,String descrizione)	Questo metodo permette di aggiungere una categoria all'interno del sistema
public boolean modificaCategoria(String nome,String descrizione,int id)	Questo metodo permette di modificare una categoria all'interno del sistema
public boolean eliminaCategoria(int id)	Questo metodo permette di eliminare una categoria e i libri associati all'interno del sistema
public Categoria acquisisciCategoria(int id)	Questo metodo permette di ricevere le informazioni di una categoria presente del database
public Collection< Categoria > acquisisciTutteLeCategorie()	Questo metodo permette di ricevere tutte le informazioni di ogni categoria presente del database

3.1.5 Manager Amministrazione

public Collection<Account> visualizzaUtentiRegistrati()	Questo metodo permette di prendere dalla base di dati tutti gli utenti del sistema
public boolean disabilitaCliente(String username)	Questo metodo permette di disabilitare un cliente dal sistema
public boolean abilitaCliente(String username)	Questo metodo permette di abilitare un cliente dal sistema

3.1.6 Manager Carrello

public boolean aggiungiLibroCarrello(Carrello carrello, Libro libroDaAggiungere)	Questo metodo permette di aggiungere un libro nel carrello

public boolean rimuoviLibroCarrello(Carrello carrello, Libro libroDaRimuovere)	Questo metodo permette di rimuovere un libro nel carrello
public boolean modificaQuantitaCarrello(Carrello carrello, Libro libro, double quantita)	Questo metodo permette di modificare la quantità di copie di un libro presente nel carrelo

3.1.7 Manager Ordini

public int effettuaOrdine(Cliente cliente, Carrello libriOrdinati)	Questo metodo permette di effettuare un ordine
public Collection<Ordine> visualizzaOrdini(Cliente cliente)	Questo metodo permette di visualizzare ad un cliente tutti gli ordini effettuati
public Ordine visualizzaDettaglioOrdine(Cliente cliente, int id)	Questo metodo permette di visualizzare ad un cliente nel dettaglio un ordine effettuato
public Collection<Ordine> visualizzaOrdiniUtenti()	Questo metodo permette di visualizzare ad un amministratore tutti gli ordini effettuati da tutti i clienti
public Ordine visualizzaDettaglioOrdineUtente(int id)	Questo metodo permette di visualizzare ad un amministratore nel dettaglio un ordine effettuato da un cliente
public boolean modificaStatoOrdine(int id, boolean stato)	Questo metodo permette di modificare lo stato di un ordine effettuato da un cliente

4. Design Pattern

4.1. DAO Pattern

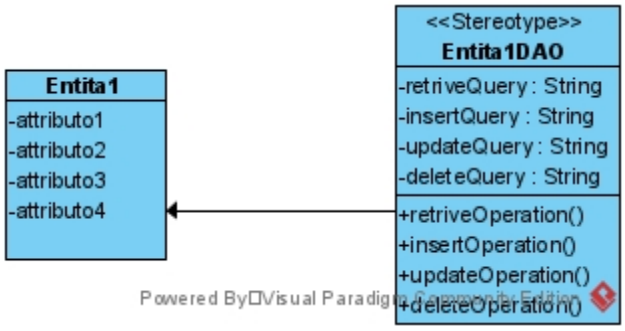
Per l'accesso al database è stato impiegato il pattern DAO (Data Access Object).

Il pattern è usato per separare le API di accesso ai dati dalla logica di business di alto livello.

Per ogni tabella presente nel DataBase esisterà una sua corrispondente classe DAO che possiederà metodi per effettuare le operazioni CRUD.

Ogni classe DAO utilizzerà esclusivamente la classe Entity corrispondente alla tabella su cui effettua le operazioni.

Di seguito è presentato un modello di utilizzo del pattern:



4.2 Façade Pattern

Per la realizzazione dei servizi dei sottosistemi è stato usato il pattern Façade. Il pattern si basa sull'utilizzo di un oggetto che permette, attraverso un'interfaccia, l'accesso a sottosistemi che espongono interfacce complesse e molto diverse tra loro, nonché a blocchi di codice complessi.

Nel nostro caso, ogni sottosistema possiede una classe chiamata manager.

Questa implementa dei metodi che corrispondono ai servizi offerti dal sottosistema. Tali metodi, nella loro implementazione, utilizzeranno le classi entity e le classi DAO per eseguire il servizio da essi offerto. In output presenteranno un oggetto che servirà alla servlet per la presentazione del risultato dell'operazione. In questo modo si svincola completamente la logica di business, implementata dai manager, e la logica di controllo, implementata dalla servlet.

Viene qui presentato un modello di utilizzo del pattern: