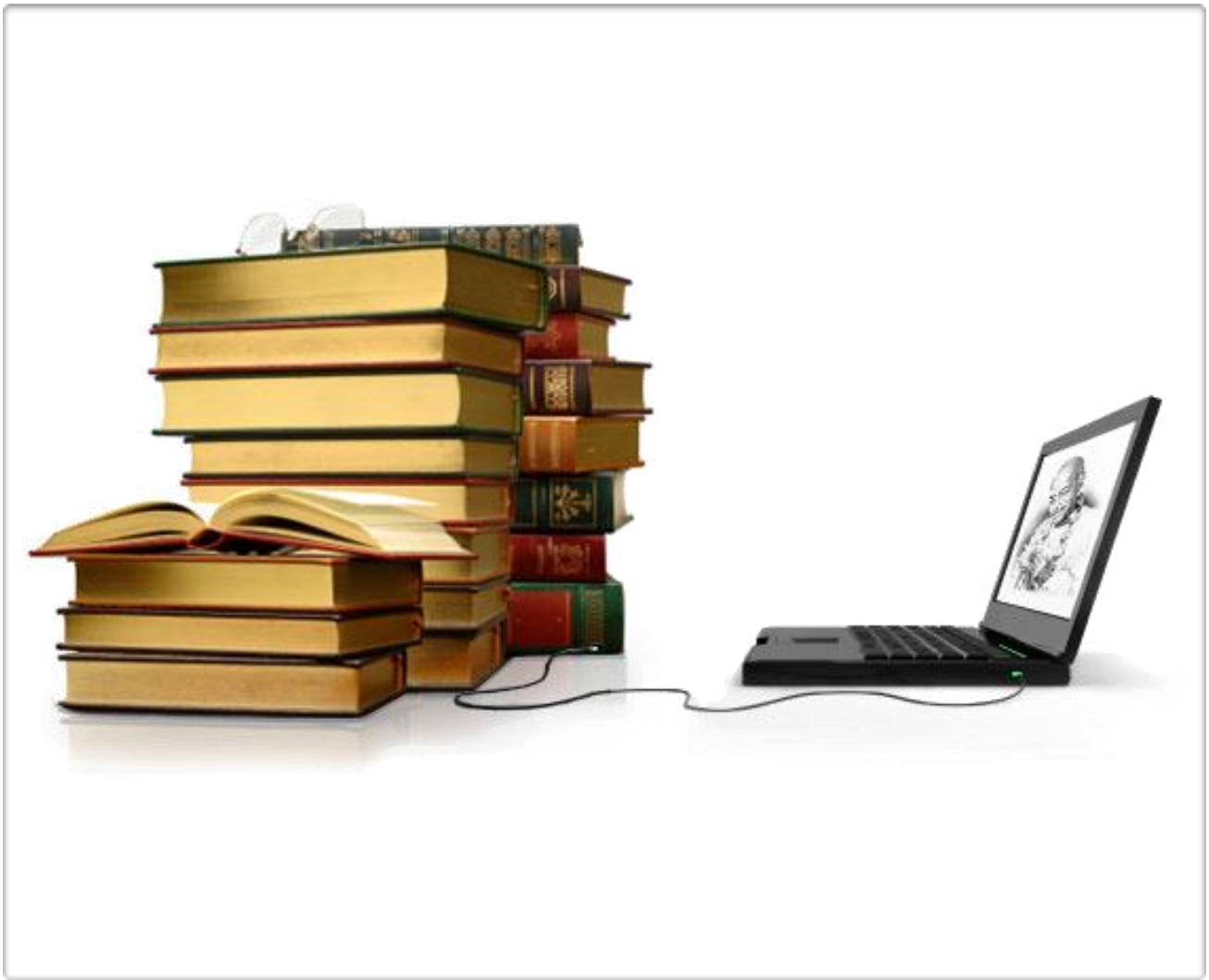


Biblioteca Digitale

Preparato per: Esame di Object-Oriented Software Design
Preparato da: Francesco Pennacchia, Simone De Angelis, Riccardo Scarpellino
Matricole: 247848 247304 249900

ANALISI DEI REQUISITI

Prefazione

Ci è stato richiesto di realizzare un software in grado di gestire digitalmente dei manoscritti antichi.
Il software è in grado di gestire diverse tipologie di utenti, con diversi permessi e diverse funzionalità a loro disposizione. I manoscritti andranno manualmente trascritti tramite editor di testo incluso nel software.
Si presuppone che questo programma sarà utilizzato all'interno di una biblioteca munita di computer dedicato.

Tipologie Utenti

Nel programma sono disponibili cinque tipi di utenti differenti:

• **Utente Semplice**: è in grado di accedere al programma, modificare i suoi dati personali ed eventualmente cancellare il suo account. Oltre questo, può cercare le opere tramite il campo di ricerca, consultarle e, se dispone dei permessi, scaricarle.

• **Uploader**: Oltre a poter gestire il proprio profilo e consultare le opere, può caricare immagini dei testi che dovranno essere poi trascritti.

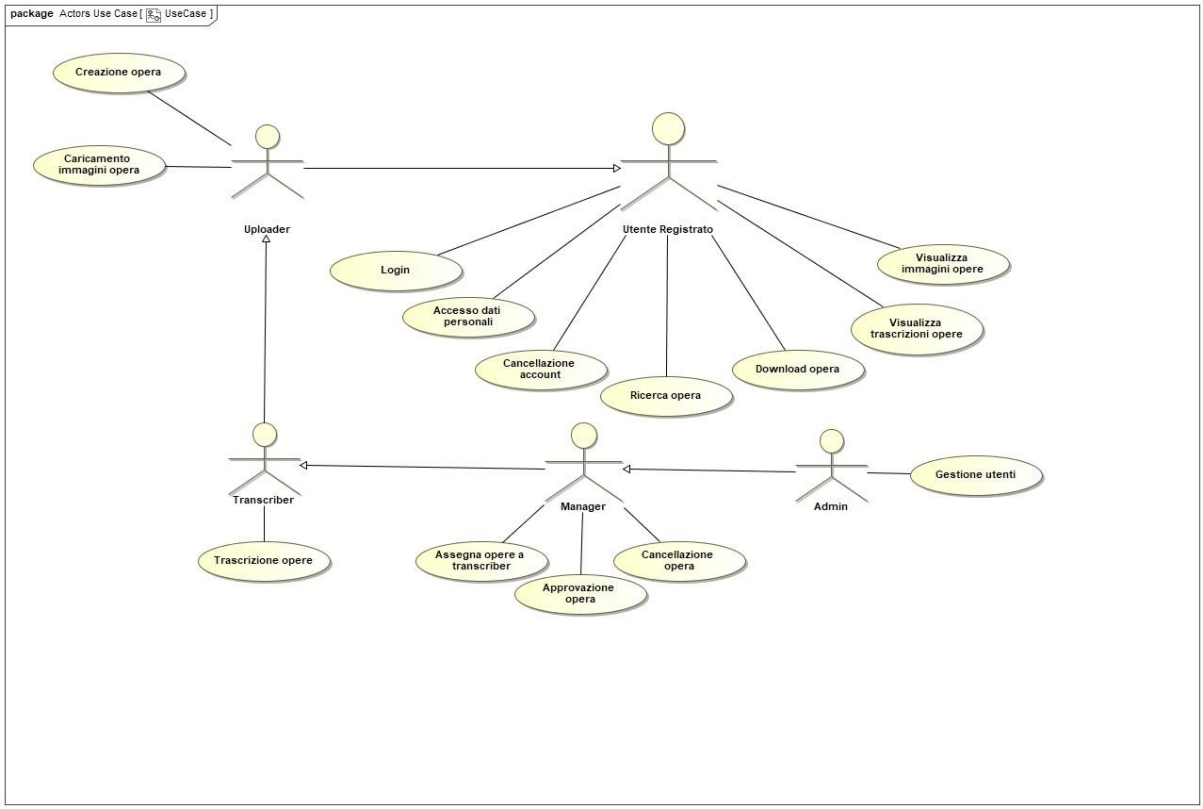
• **Transcriber**: questo utente è un'estensione dell'uploader, quindi, oltre a caricare le immagini, è in grado di trascrivere i testi tramite l'editor incluso nel programma.
I transcriber hanno un punteggio (da 1 a 5) a loro assegnato in base all'esperienza.

• **Manager**: Il manager è colui che si occupa della gestione delle opere, fa si che le trascrizioni procedano per il meglio dando o meno la sua approvazione e decide a quale trascrittore associare una o più opere.
Inoltre è in grado di cancellare definitivamente un'opera.

• **Admin**: Il suo ruolo è quello di gestire il sistema, di fare in modo che tutto funzioni e quindi ha libero accesso a tutte le opere e può, se necessario, modificare gli account altrui.

Use Case

Ecco il modello “Use case” che ci aiuta a capire come saranno gestiti i ruoli e le funzioni svolte.

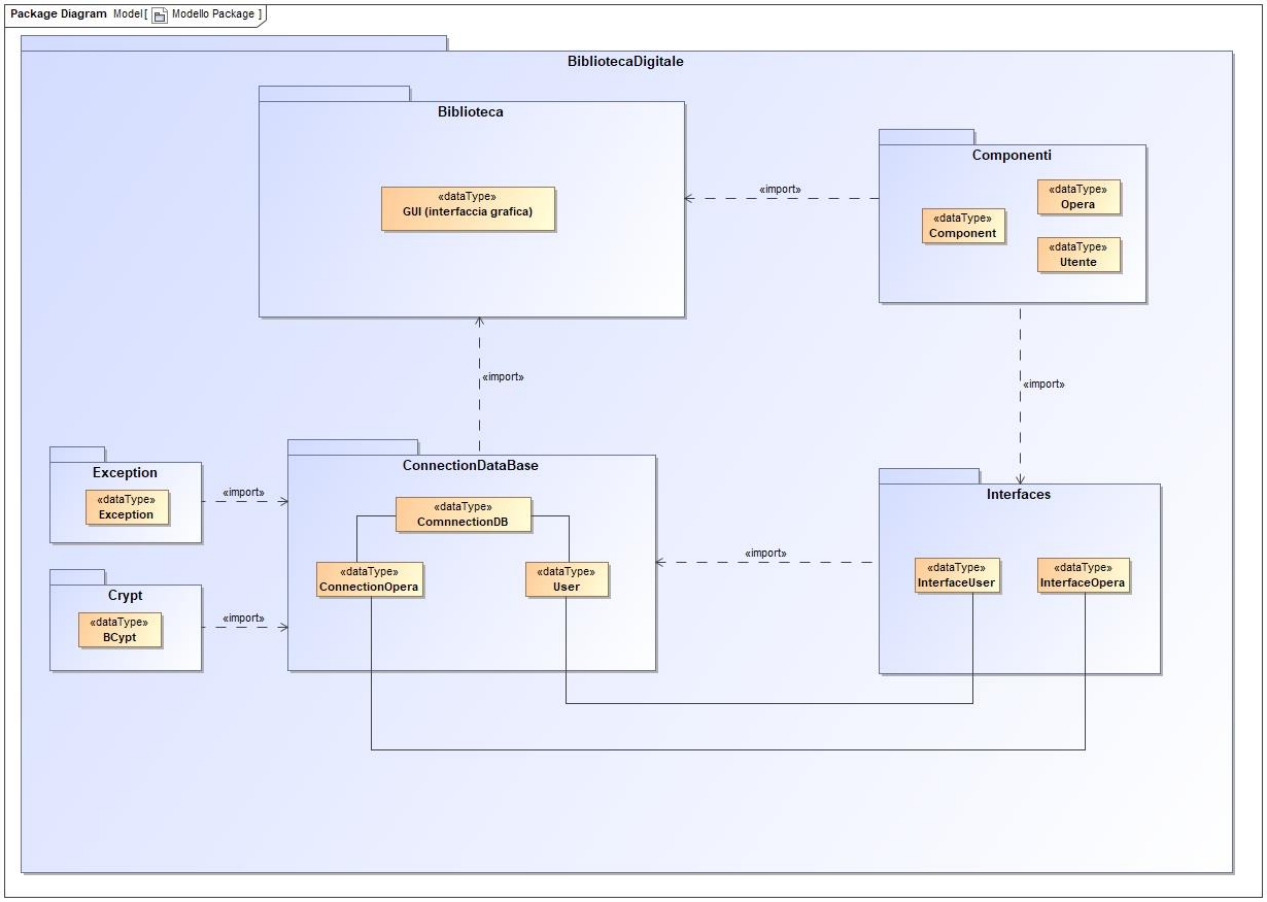


Requisiti principali del software:

- **Login**: L’utente dovrà accedere e in base al ruolo potrà svolgere le proprie mansioni.
- **Registrazione**: Nel caso non si è registrati c’è un opportuno modulo per potersi registrare. Al momento della registrazione verrà assegnato il ruolo di “utente”.
- **Visualizzazione e Modifica del profilo**: Gli utenti registrati possono visualizzare e modificare i propri dati.
- **Visualizzazione della lista utenti**: Solo l’admin potrà accedere a questa sezione, con i suoi permessi potrà modificare i loro ruoli.
- **Visualizzazione e gestione delle opere**: Tutti gli utenti possono visualizzare le opere inserite, ma solo quelli aventi i ruoli opportuni possono gestirle.

Il tutto è gestito tramite una comoda interfaccia grafica.

Architettura Software



In questo diagramma vengono mostrati i package del software e come vengono collegati tra loro.
Come architettura è stata applicata la **CLIENT – SERVER**.
Come possiamo vedere dal diagramma il package "**Biblioteca**" è il nostro **CLIENT**, invece il package "**ConnectionDataBase**" consiste nel nostro **SERVER**.
Il Client è l'interfaccia (GUI) per l'utente che permette di gestire tutte le operazione, in modo facile e veloce.
Il Server si occupa della gestione degli accessi alla banca dati e di fornire tutti i costrutti necessari a far funzionare il nostro Client.

CLIENT

Per quanto riguarda lo sviluppo del Client sono state utilizzate le librerie "**Java Swing**". Queste librerie ci hanno fornito il necessario per la realizzazione dell'interfaccia grafica (GUI).
Inoltre per agevolare il lavoro abbiamo utilizzato "**WindowBuilder**". Window Builder è una comoda estensione per Eclipse che ci permette di realizzare la nostra interfaccia grafica, sfruttando sempre le librerie Java Swing.

SERVER

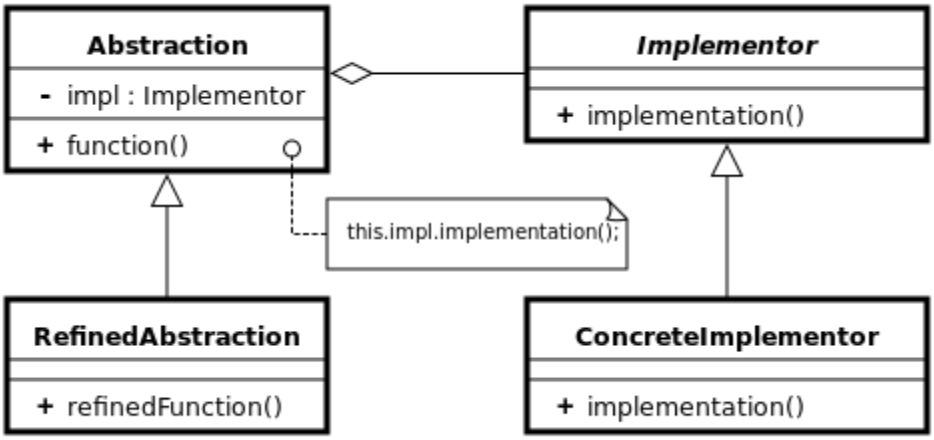
Utilizzo del pattern **DAO**
Il ***DAO*** (*Data Access Object*) è un pattern architetturale per la rappresentazione tabellare di un RDBMS, usata principalmente per stratificare e isolare l'accesso ad una tabella tramite query.
Per quanto riguarda il **database** dove sono archiviate tutti i dati degli utenti e delle opere abbiamo optato per **MySQL**

un RDBMS open source.

Altri pattern:

Bridge Pattern

Il bridge pattern è un design pattern (modello di progettazione) della programmazione ad oggetti che permette di separare l'interfaccia di una classe, in modo tale si può usare l'ereditarietà per fare evolvere l'interfaccia o l'implementazione in modo separato.



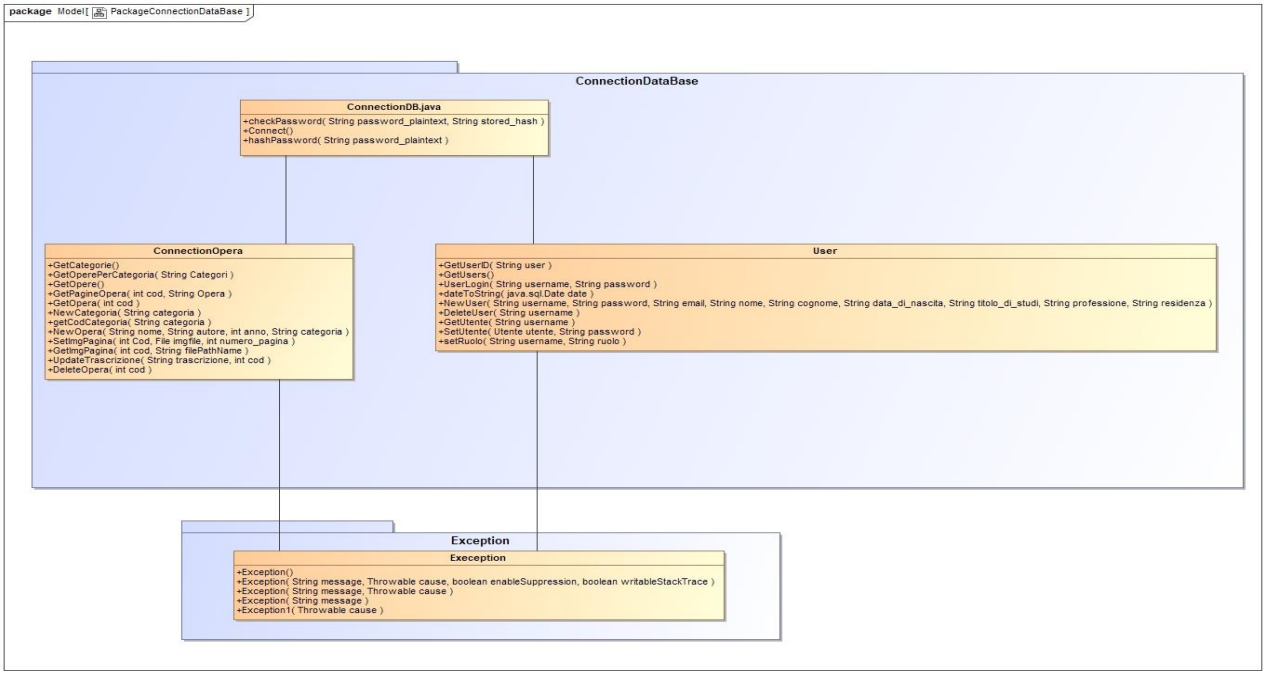
CLASSE DIAGRAM

Rappresentazioni dei package:

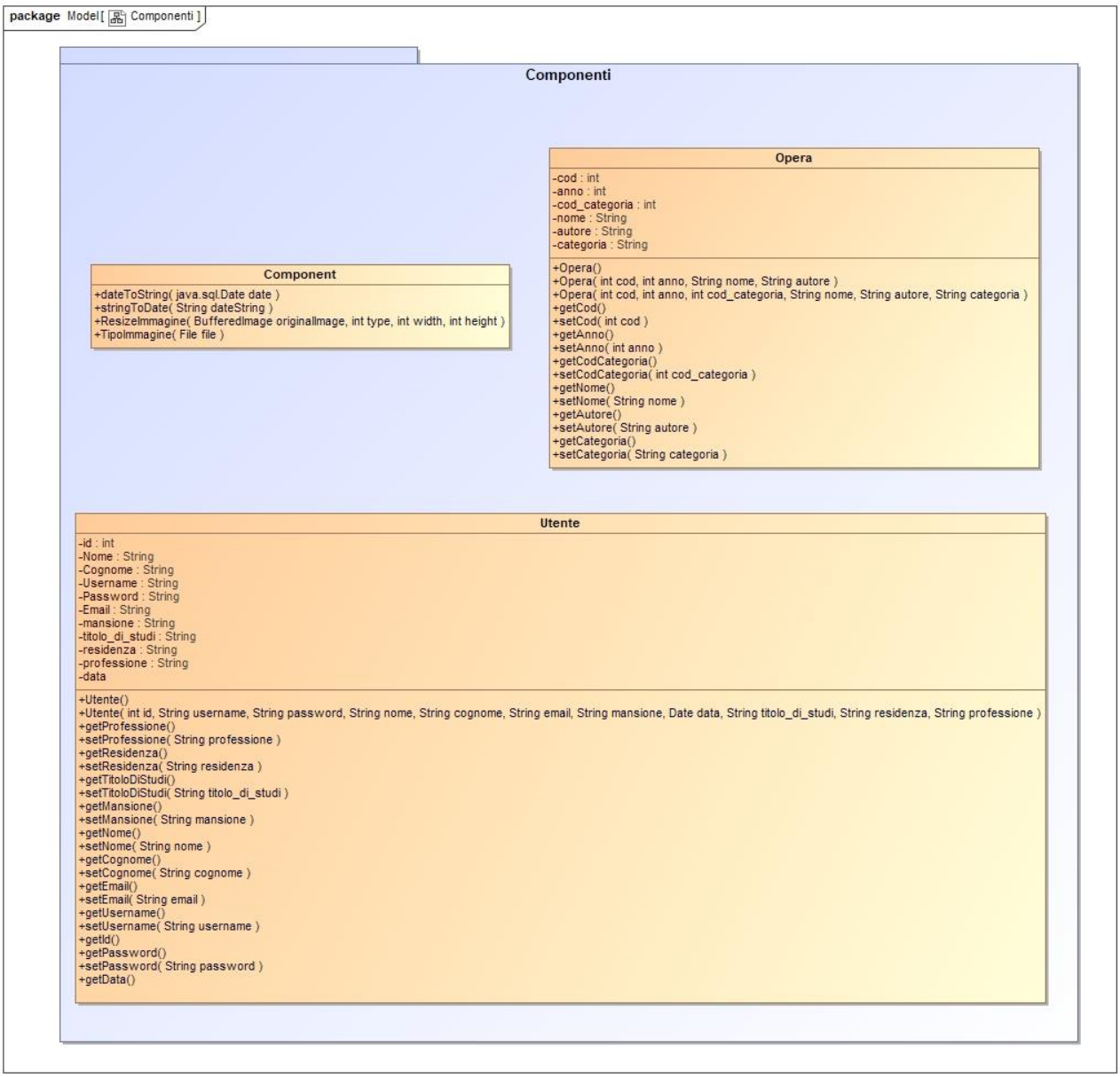
Il nostro software è sviluppato principalmente su questi package: "**ConnectionDataBase**", "**Exception**", "**Componenti**" e "**Biblioteca**".

Rappresentazione delle classi all'interno dei package:

- **ConnectionDataBase**: Si occupa della gestione e della connessione al Database, in particolare le classi:
    1. **ConnectionDB** si occupa della connessione al DB;
    2. **ConnectioOpera** si occupa della gestione dell'opera e tutte le query riguardanti l'opera;
    3. **User** si occupa degli utenti e tutte le query riguardanti la gestione degli utenti.



- **Componenti**: all'interno ci sono delle componenti aggiuntive che ci aiutano nella gestione di opere e utenti, in particolare sono contenute le seguenti classi:
    1. **Opera** dedicata all'opera;
    2. **Utente** dedicata all'utente;
    3. **Component** qui ci sono alcuni metodi per la gestione delle immagini e delle Date.


package Model[Componenti]
Componenti
Component
+dateToString(java.sql.Date date)
+stringToDate(String dateString)
+ResizeImmagine(BufferedImage originalImage, int type, int width, int height)
+TipoImmagine(File file)
Opera
-cod : int
-anno : int
-cod_categoria : int
-nome : String
-autore : String
-categoria : String
+Opera()
+Opera(int cod, int anno, String nome, String autore)
+Opera(int cod, int anno, int cod_categoria, String nome, String autore, String categoria)
+getCod()
+setCod(int cod)
+getAnno()
+setAnno(int anno)
+getCodCategoria()
+setCodCategoria(int cod_categoria)
+getNome()
+setNome(String nome)
+getAutore()
+setAutore(String autore)
+getCategoria()
+setCategoria(String categoria)
Utente
-id : int
-Nome : String
-Cognome : String
-Username : String
-Password : String
-Email : String
-mansione : String
-titolo_di_studi : String
-residenza : String
-professione : String
-data
+Utente()
+Utente(int id, String username, String password, String nome, String cognome, String email, String mansione, Date data, String titolo_di_studi, String residenza, String professione)
+getProfessione()
+setProfessione(String professione)
+getResidenza()
+setResidenza(String residenza)
+getTitoloDiStudi()
+setTitoloDiStudi(String titolo_di_studi)
+getMansione()
+setMansione(String mansione)
+getNome()
+setNome(String nome)
+getCognome()
+setCognome(String cognome)
+getEmail()
+setEmail(String email)
+getUsername()
+setUsername(String username)
+getId()
+getPassword()
+setPassword(String password)
+getData()

- **Biblioteca** in questo package sono contenuti tutti i moduli che compongono l'interfaccia grafica (**GUI**) del **CLIENT.** Ogni modulo è stato realizzato in modo da essere user-friendly, inoltre nel sistema interfacce viene gestito anche il sistema dei permessi, dove solo utenti con un certo ruolo possono accedervi.