Progetto Object Oriented Software Design

Docente : **Romina Eramo**

Studente : **Lapenna Fulvio 252334** fulvio.lapenna@student.univaq.it

2018/2019

repository GIT : https://github.com/BestDogeGitHub/Progetto-OOSD.git

Specifica

Il catalogo bibliografico realizzato ha lo scopo di poter mantenere in formato digitale informazioni riguardanti una lista di pubblicazioni.

L'applicazione è stata creata per tutte quelle piccole - medie realtà, come ad esempio una biblioteca, che hanno interesse a fornire ai propri clienti una piattaforma dove poter consultare le pubblicazioni a cui si è interessati.

oltre alla consultazione delle pubblicazioni il catalogo permette anche una gestione degli utenti iscritti, che verrà descritta piu' avanti.

il catalogo fornisce anche alcune funzioni 'social': gli utenti una volta essersi identificati tramite il login potranno dare un like, o una recensione, ad una pubblicazione presente nel catalogo, le recensioni, una volta approvate da un moderatore, potranno essere visualizzate dagli altri utenti.

Per qualsiasi informazione sul funzionamento del Sistema o sulle caratteristiche di gestione delle informazioni potete consultare la documentazione del progetto di database.

Sistema

Il sistema si divide in tre sotto-sistemi :

1. utenti
2. pubblicazioni
3. iterazioni

Utenti

Questa parte del sistema si occupa della conservazione e visualizzazione delle informazioni sugli utenti presenti nel sistema, si occupa inoltre della gestione e della modifica degli utenti.

per ulteriori informazioni riguardanti i vari tipi di utenti si consulti la sezione **'Attori'**

Pubblicazioni

Questo sottosistema si occupa di conservare tutte le informazioni riguardanti le pubblicazioni per permettere agli utenti di visualizzarle e modificarle.

Iterazioni

Questo sottosistema si occupa di tutte le iterazioni tra le pubblicazioni e gli utenti, quindi della conservazione e gestione delle informazioni riguardanti i like, le recensioni e lo storico delle modifiche delle pubblicazioni.

Documento dei Requisiti

Attori del Sistema

Gli attori del sistema si dividono in:

- Utente non Autenticato
- Utente Base
- Moderatore
- Amministratore
- Super-Amministratore

Utente non Autenticato

Rappresenta l'utente che sta utilizzando il catalogo ma che non ha ancora eseguito il login, le uniche azioni che può svolgere sono le procedure di registrazione al catalogo e di login, l'utente può inoltre accedere alla lista delle pubblicazioni presenti nel catalogo, eseguire una ricerca e visualizzare il profilo delle pubblicazioni.

Utente Base

Rappresenta l'utente che si è registrato al sito ed ha eseguito il login, oltre alle azioni dell'utente non autenticato, l'utente base, una volta visualizzato il profilo di una pubblicazione, può scegliere di mettere like, oppure di scrivere una recensione.

può inoltre visualizzare la lista degli utenti registrati e accedere al profilo di un utente e può inoltre accedere allo storico del sistema.

Gli utenti base sono a loro volta divisi in utenti passivi ed utenti attivi, questa divisione è stata creata puramente a scopo informativo e non c'è alcuna differenza per quanto riguarda le azioni possibili.

Un utente viene classificato come attivo se ha scritto una recensione negli ultimi due mesi,

voceversa un utente viene classificato come passivo se non ha inserito nessuna recensione negli ultimi due mesi.

Moderatore

Oltre alle azioni svolte da un utente Base un Moderatore può approvare le Recensioni scritte da un utente, oppure eliminarle, i moderatori possono inoltre inserire nuove schede bibliografiche all'interno del catalogo, possono inoltre modificare i dati delle pubblicazioni già inserite.

Amministratore

Oltre alle azioni svolte da un moderatore gli amministratori si occupano della gestione dei moderatori, possono infatti promuovere un utente base a moderatore e viceversa rimuovere il permesso di moderatore ad un utente facendolo tornare al grado di utente base.

Gli amministratori possono inoltre eliminare dal catalogo una pubblicazione.

Possono anche eliminare un utente, ma solo se questo utente non è a sua volta un amministratore.

Super-Amministratore

Unico, può svolgere qualsiasi azione sul catalogo, si occupa della gestione degli amministratori, può promuovere quindi un moderatore ad amministratore oppure viceversa degradare un amministratore a moderatore.

Pubblicazione

Una pubblicazione rappresenta una scheda bibliografica associata ad un libro, un'opera letteraria, o qualsiasi scritto che può essere identificato mediante codice ISBN.
Le pubblicazioni vengono inserite nel catalogo dagli utenti (di grado moderatore o superiore).

Ogni scheda bibliografica contiene una serie di informazioni, con struttura comune a tutte le pubblicazioni**.**

Oltre a queste informazioni una Pubblicazione è associata a quattro liste:

- Autori

rappresenta la lista di autori che hanno scritto la pubblicazione.

- Parole Chiave

Rappresenta una lista di **keyword** da associare alla pubblicazione definite dall'utente che inserisce la Pubblicazione

- Ristampe

La lista delle ristampe di quella pubblicazione che sono state inserite nel database.

- Sorgenti

Una lista di Sorgenti esterne associate alla pubblicazione, contengono il **link** esterno è una breve descrizione del contenuto digitale.

Iterazioni

Una pubblicazione è infine associata ai like inseriti dagli utenti e da una lista di recensioni, queste recensioni saranno visibili sul profilo della pubblicazione solo una volta approvate da un moderatore.

Ogni utente può inserire un solo like e scrivere una sola recensione per la stessa pubblicazione.

Ogni modifica effettuata dagli utenti ad una pubblicazione verrà automaticamente registrata all'interno dello storico, dove sarà presente un "entry" per ogni azione sulle pubblicazione.

Ogni entry dello storico contiene una breve descrizione della modifica e il timestamp di quando è avvenuta.

Altri Requisiti

Il sistema deve essere associato ad una **Interfaccia Grafica** capace di modificarsi in base al tipo di utente connesso.

Il sistema deve essere **scalabile,** deve quindi essere in grado di supportare eventuali

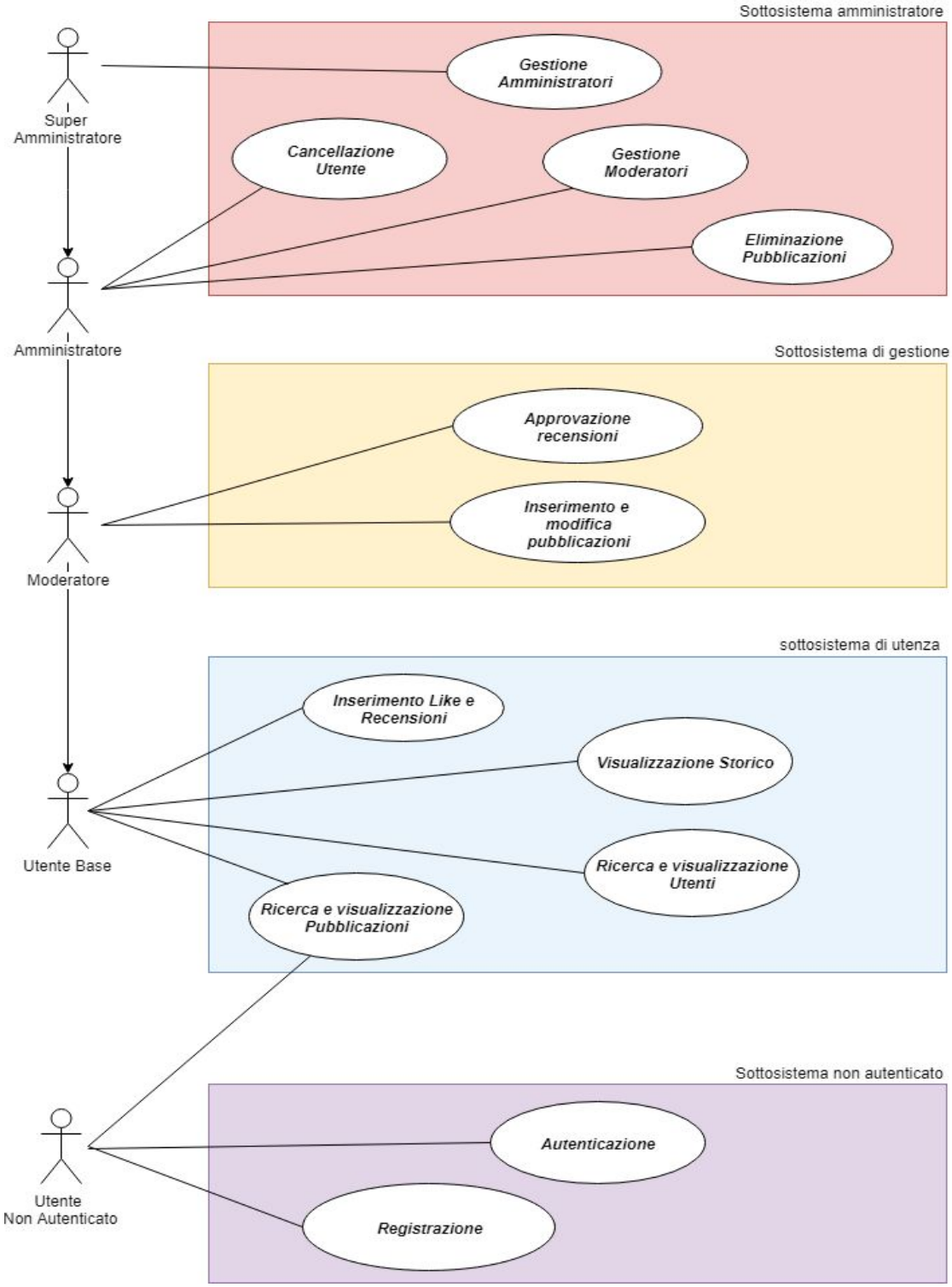
modifiche ed espansioni funzionali future.

L'interfaccia grafica deve essere semplice e intuitiva.

Il sistema deve fornire un livello di **Sicurezza** di base per i propri utenti, conservando le loro password nel database solo dopo averle criptate.

Il sistema deve appoggiarsi ad un database per permettere la conservazione delle informazioni.

Modello Use Case



Descrizione Use Case

In questa sezione verranno descritte tutte le azioni possibili dagli utenti del catalogo:

Autenticazione

processo di identificazione di un utente non autenticato in cui verrà richiesto l'inserimento di username e password, se i dati inseriti sono collegati ad un account l'accesso verrà eseguito con successo.

Registrazione

processo in cui verrà chiesto all'utente di inserire: Email, Username, Password, Nome, Cognome, DataNascita, LuogoNascita e Residenza, se i dati sono corretti, la registrazione avviene con successo, il profilo viene salvato nel database e sarà possibile eseguire il login

Visualizzazione lista Pubblicazioni (Catalogo)

In questa sezione sarà possibile visualizzare l'intera lista di pubblicazioni presenti.

Le pubblicazioni potranno essere ordinate per : Titolo (alfabetico), Data di inserimento nel catalogo, Data di ultimo aggiornamento.

Si potrà inoltre visualizzare le pubblicazioni con allegato fra le sorgenti un download del libro.

Scelta una pubblicazione, si potrà visualizzare il suo profilo.

Ricerca Pubblicazioni

Permette all'utente di eseguire una ricerca su tutte le pubblicazioni in catalogo.

La ricerca può essere fatta tramite: l'isbn di un libro, il titolo del libro, una parola chiave, oppure inserendo il nome o il cognome di un autore.

dopo scelto il tipo di ricerca e aver inserito i dati richiesti verrà visualizzata la lista di tutte le pubblicazioni con cui avverrà un matching.

Scelta una pubblicazione, si potrà visualizzare il suo profilo.

Visualizzazione profilo Pubblicazione

Il profilo di una pubblicazione del catalogo in cui sarà possibile visualizzare tutti i dati riguardanti la pubblicazione e la lista delle sue recensioni.

Inserimento Like e Recensioni

Gli utenti mentre visualizzano il profilo di una pubblicazione possono mettere Like (oppure togliere Like) alla pubblicazione e possono inoltre inserire una recensione alla pubblicazione se non ne hanno già scritta una in precedenza per la stessa pubblicazione (a meno che non è stata eliminata) .

una recensione può essere eliminata da un utente di grado moderatore o superiore, oppure dall'utente che l'ha scritta.

Modifica pubblicazioni

Gli utenti mentre visualizzano il profilo di una pubblicazione, se hanno i permessi adeguati, potranno modificare i dati riguardanti quella pubblicazione, potranno inoltre aggiornare le liste di Autori,Parole Chiave, Sorgenti e Ristampe associate alla pubblicazione.

Se l'utente è un amministratore la pubblicazione potrà anche essere eliminata.

Visualizzazione Lista Utenti

Si potrà visualizzare la lista completa degli utenti inseriti nel database in ordine alfabetico, oppure visualizzare gli utenti più 'collaborativi' ordinati per numero di pubblicazioni inserite.

Scelto un utente, si potrà visualizzare il suo profilo.

Visualizzazione profilo Utente

Qui saranno mostrati tutti i dati di un utente che si è registrato al catalogo, sarà inoltre possibile modificare il grado dell'utente oppure eliminare l'account in base alle regole precedentemente descritte riguardo la gestione degli utenti.

Sarà inoltre possibile visualizzare la lista delle sue recensioni e la lista delle pubblicazioni inserite da lui.

Visualizzazione Profilo Utente (Personale)

La pagina personale dell'utente che ha effettuato il login, qui potrà visualizzare i suoi dati.

L'utente potrà inoltre modificare tutti i suoi dati, compreso cambiare la propria password, oppure potrà visualizzare la lista delle recensioni che ha inserito nel catalogo.

Visualizzazione Storico

Qui si potrà visualizzare la lista di tutte le modifiche che vengono effettuate sulle pubblicazioni nel database, compreso l'inserimento di like e recensioni e l'approvazione di queste ultime.

Ogni entry dello storico contiene una breve descrizione della modifica effettuata, il tipo di modifica e il timestamp di quando è avvenuta.

Sarà possibile inoltre aprire il profilo della pubblicazione inerente una entry, oppure aprire il profilo dell'utente che ha modificato la pubblicazione

Approvazione Recensioni

I moderatori del catalogo (e gli utenti di grado superiore) potranno accedere ad una sezione del catalogo dove potranno trovare una lista contenente solamente le recensioni in attesa di approvazione.

Le recensioni non approvate non saranno visibili dagli utenti che visitano il profilo di una pubblicazione.

Le recensioni scritte da utenti di grado moderatore o superiore verranno approvate automaticamente dal sistema nel momento dell'inserimento

Inserimento Pubblicazioni

Gli utenti potranno inserire una pubblicazione nel catalogo inserendo tutti i suoi dati e associandoli inoltre:

- una lista di autori
- una lista di parole chiave
- una lista di sorgenti
- una lista di ristampe

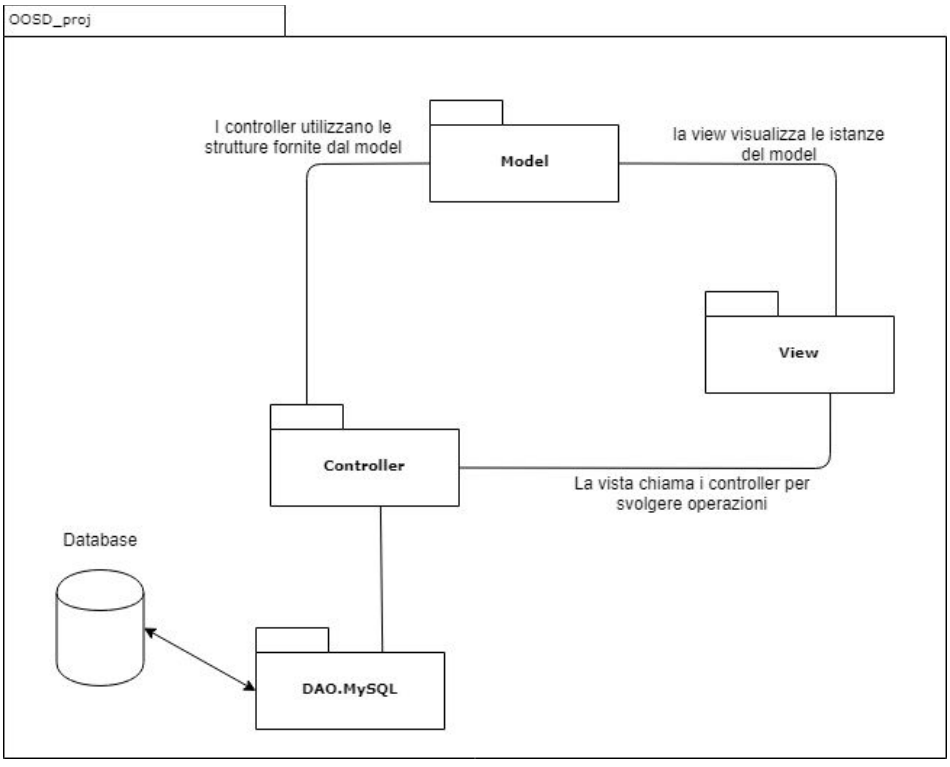
Design del Sistema

Modello Architettura software

L'applicazione è stata sviluppata basandosi sul design pattern fondamentale **MVC** (Model View Controller) con cui è stata decisa la struttura del progetto, dandogli la sua prima forma.

Un altro pattern utilizzato è il DAO (Data Access Object). Tale pattern è stato adottato per isolare l'accesso al database, questo renderà più facili operazioni di manutenzione.

L'utilizzo del DAO ha permesso al programma di raggiungere un maggior livello di astrazione, riuscendo anche a semplificare le operazioni di modifica del sistema.



Il sistema finale è diviso nei seguenti package:

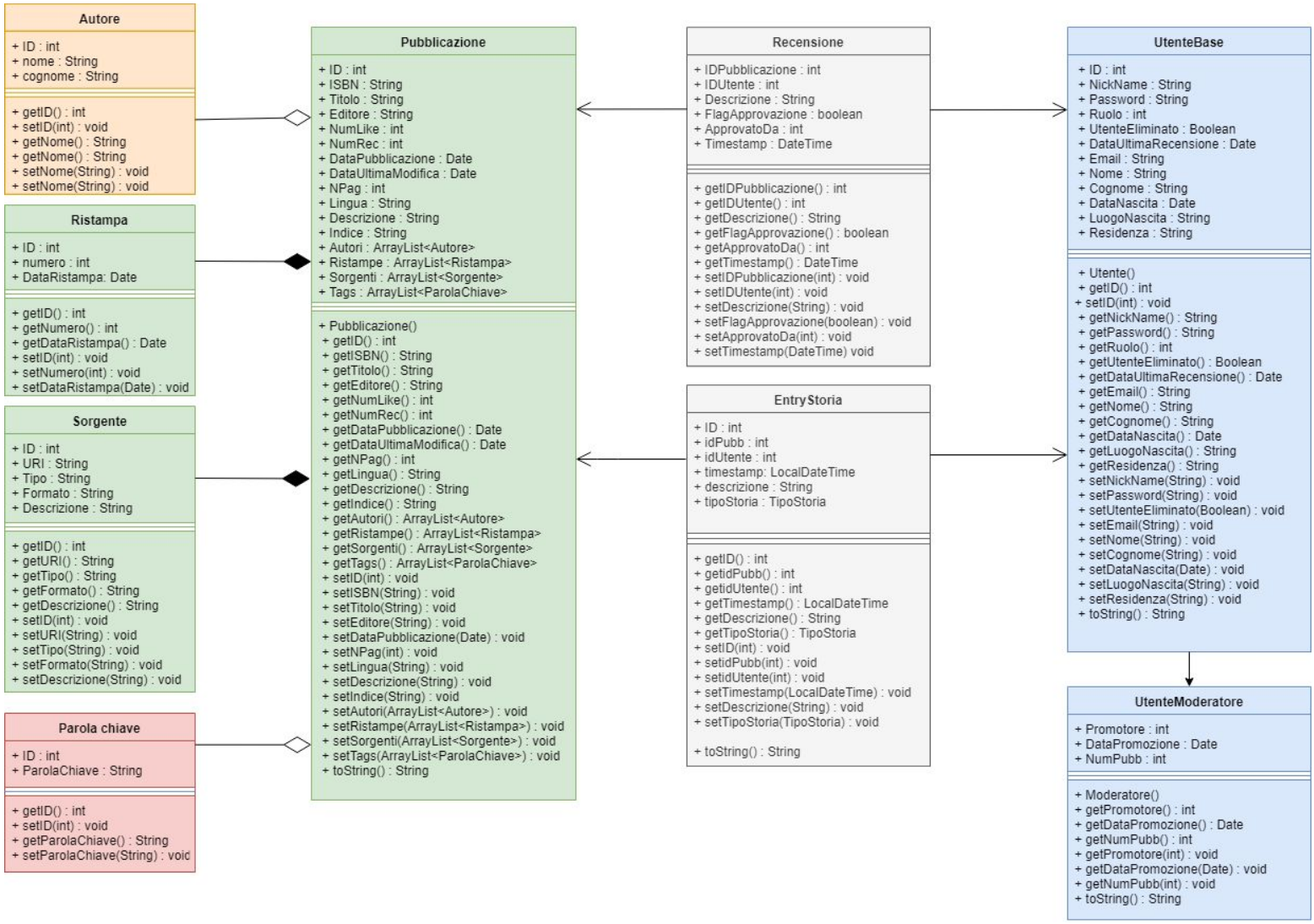
- **Model**
- **Controller**
- **View e ViewController**
- **DAO**

Model

contiene le classi che rappresentano la parte statica del sistema.

Queste classi verranno istanziate durante l'esecuzione del applicazione per conservare le informazioni delle entità del sistema.

Model. Class Diagram



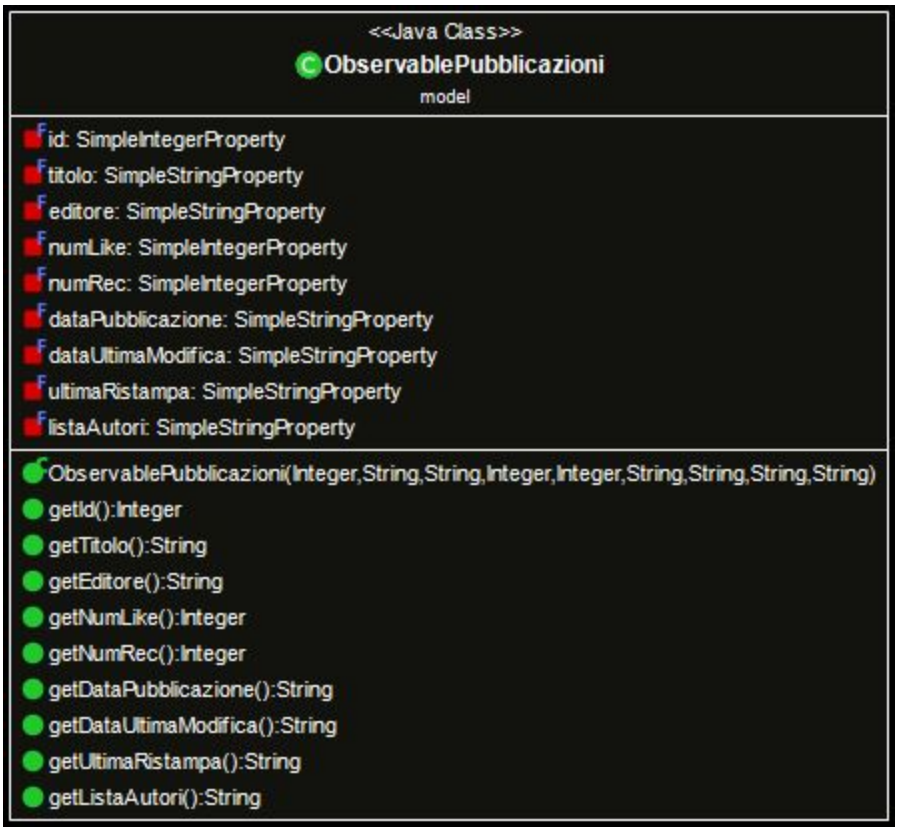
Nel **Model Class DIagram** possiamo osservare le principali entità del programma e i loro attributi, abbiamo la **Pubblicazione** associata alle classi **Autore,Ristampa,Sorgente e Parole Chiave,** gli oggetti di queste 4 classi, anche se rappresentano entità a sé stanti, sono sempre associati ad una pubblicazione e arricchiscono quest'ultima.

Gli utenti del sistema vengono rappresentati tramite la classe **UtenteBase,** mentre per gli utenti di grado moderatore o superiore è stata creata la sottoclasse **UtenteModeratore** che oltre ai dati comuni a tutti gli utenti contiene anche dati riguardo la data di promozione, l'utente che lo ha promosso e il numero di pubblicazioni inserite nel database.

Le **Recensioni** e le **Entry dello Storico** rappresentano invece oggetti associati alla pubblicazione a cui si riferiscono e all'utente che li ha generati.

oltre alle classi definite nel precedente "class diagram" il package **Model** contiene anche le classi:

- **TipiEnum :** in cui sono stati definiti alcuni tipi enumerativi che sono stati utilizzati durante la programmazione.
- **ObservablePubblicazioni :** classe utilizzata nelle interfacce per la visualizzazione delle pubblicazioni all'interno delle liste.



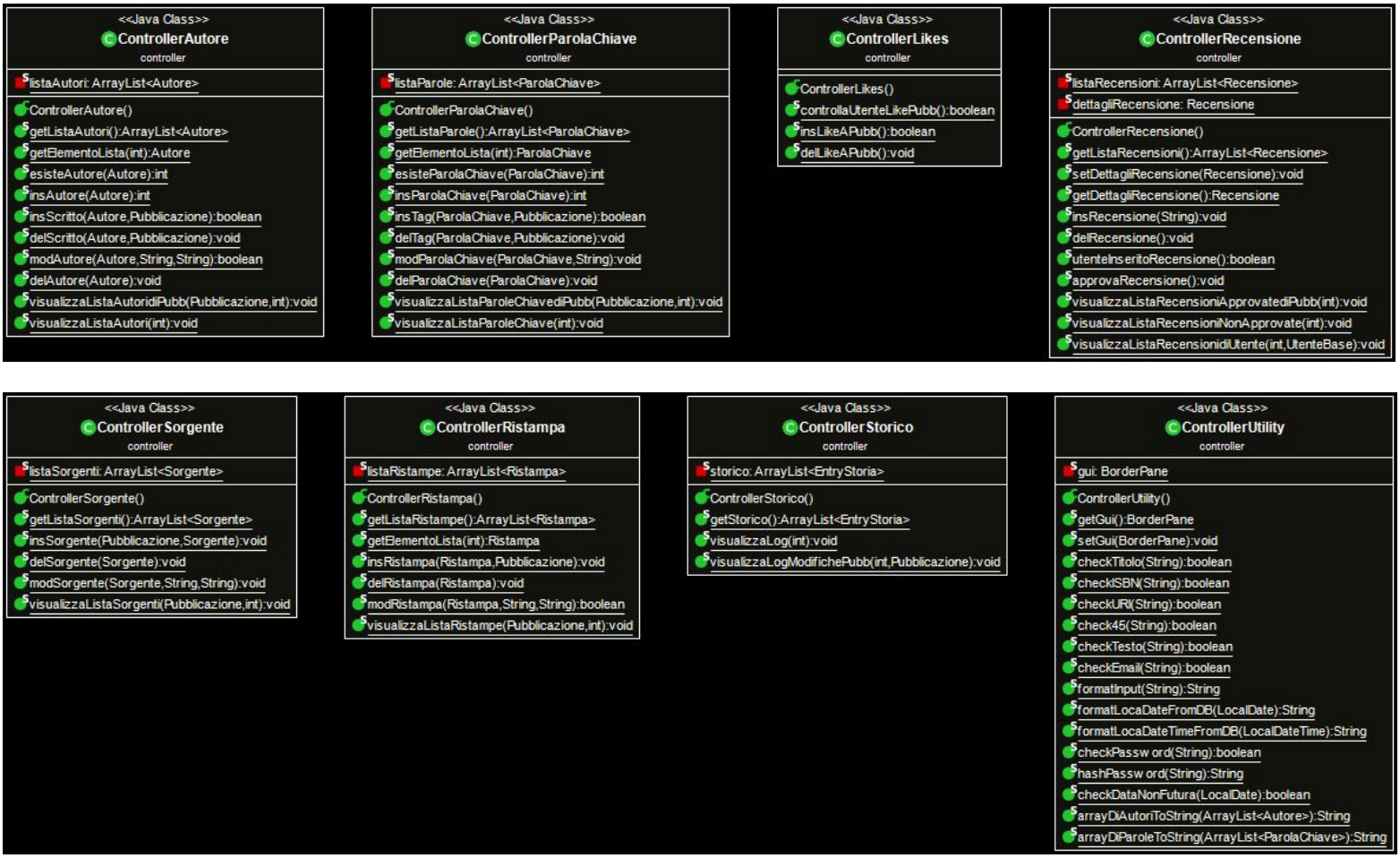
Questa classe è stata creata seguendo il pattern **JavaBeans** per permettere la visualizzazione delle pubblicazioni tramite lista all'interno delle **TableView** in **JavaFX.**

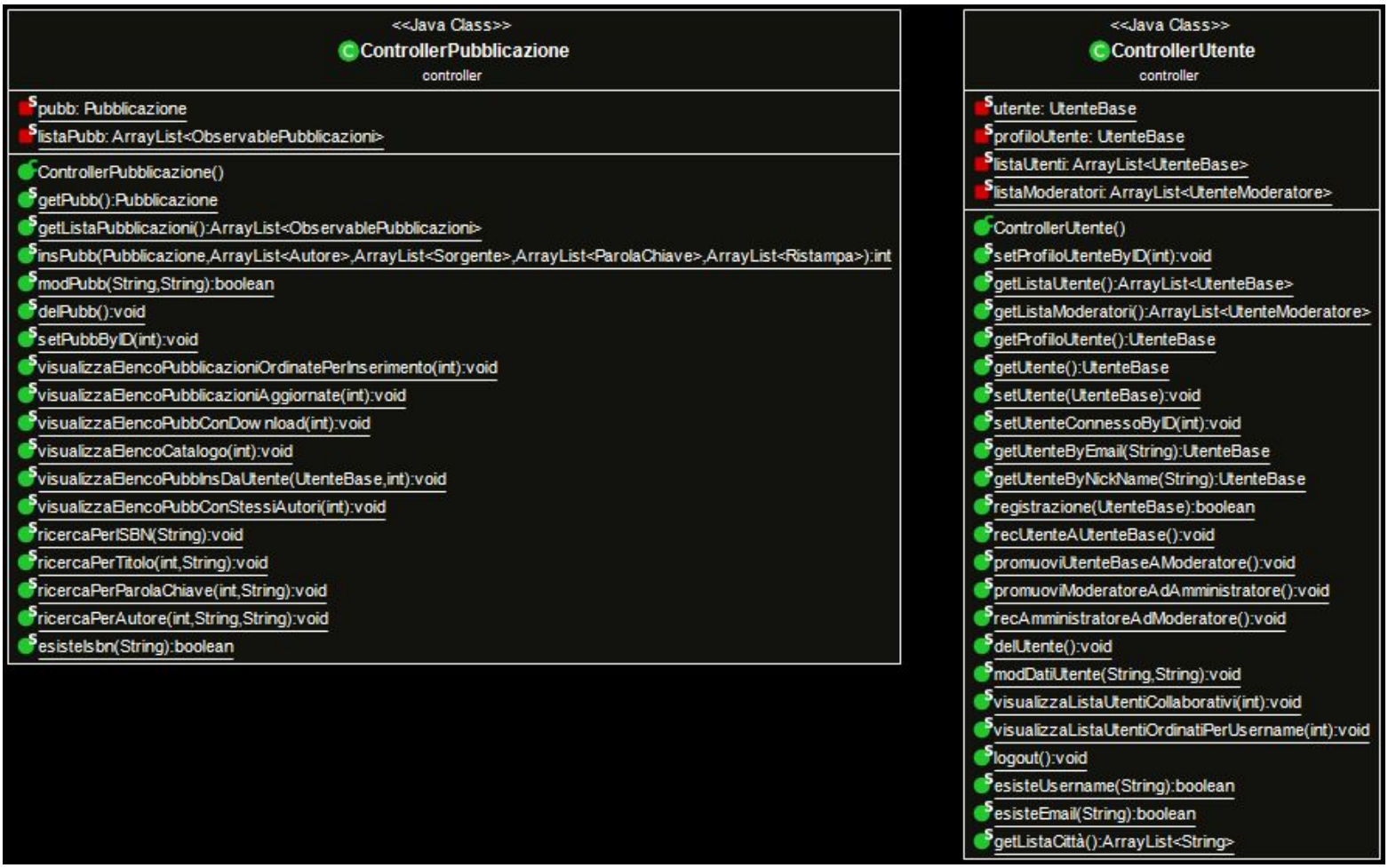
Un oggetto di 'ObservablePubblicazioni' non raccoglie tutti i dati di una pubblicazione, ma solo quelli che si vuole mostrare all'utente mentre sfoglia un elenco di varie pubblicazioni.

Controller

La parte che connette gli altri elementi del sistema, viene chiamata dalla view in base alle richieste dell'utente, si relaziona con il database servendosi del DAO e manipola le informazioni in output dal DB istanziando le classi del Model.

Ogni Controller contiene al suo interno un oggetto della classe DAO con cui si relaziona per la comunicazione con il database





Oltre ai controller per la gestione di ogni entità del sistema è stato creata la classe **ControllerUtility** che racchiude principalmente metodi per la formattazione dei testi, la convalida degli input dell'utente e il metodo per la criptazione della password

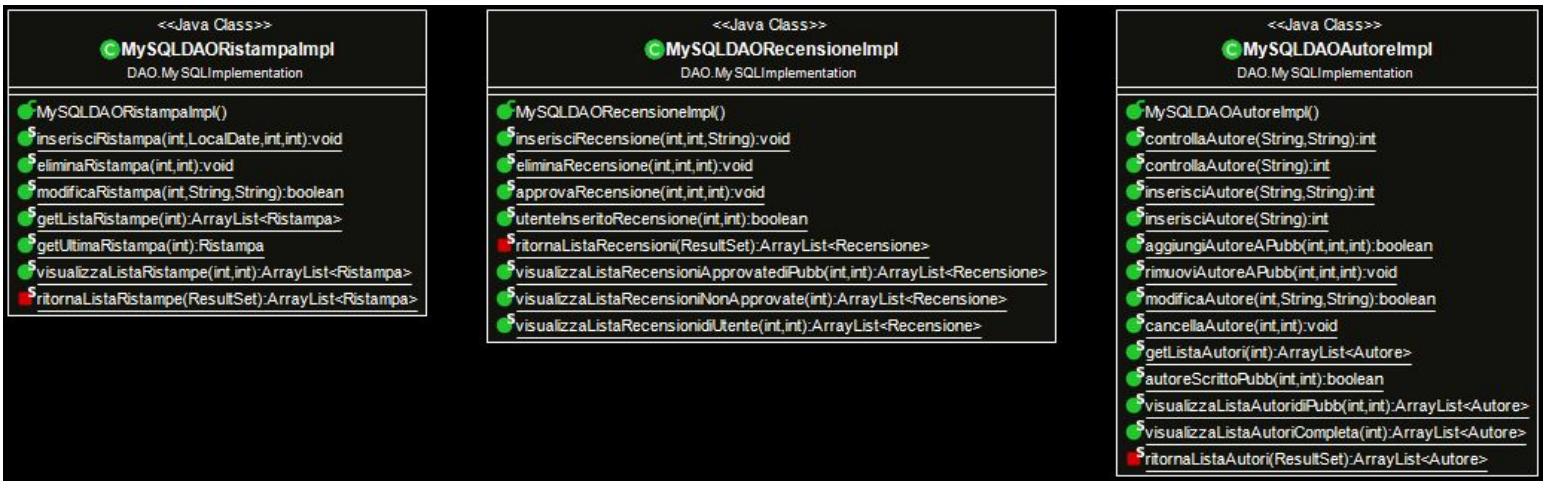
DAO

Le classi relative al DAO gestiscono la comunicazione con il DBMS, si occupano quindi dell'interrogazione del database e della successiva manipolazione del **ResultSet** restituito, per poi passare il risultato al controller, che non si deve quindi preoccupare delle specifiche del database con cui il sistema è connesso.

Ogni classe del DAO è stata scritta a partire da un interfaccia che definisce le operazione da implementare, questo permetterà in futuro la possibilità di scrivere un nuovo DAO per una piattaforma differente (per esempio un DB della Oracle) e di riutilizzare gli stessi controller che gestiscono l'implementazione di MySQL, basterà che il nuovo DAO rispetti le interfacce definite.


<<Java Class>>
MySQLDBConnector
DAO.connectors
MySQLDBConnector()
creaConnessione():Connection


<<Java Class>>
MySQLDAOPubblicazioneImpl
DAO.MySQLImplementation
MySQLDAOPubblicazioneImpl()
getElencoSenzaParametri(String,int):ArrayList<ObservablePubblicazioni>
getElencoConParametroString(String,int,String):ArrayList<ObservablePubblicazioni>
visualizzaElencoPubblicazioniOrdinatePerInserimento(int):ArrayList<ObservablePubblicazioni>
visualizzaElencoPubblicazioniAggiornate(int):ArrayList<ObservablePubblicazioni>
visualizzaElencoPubbConDow nload(int):ArrayList<ObservablePubblicazioni>
visualizzaElencoCatalogo(int):ArrayList<ObservablePubblicazioni>
visualizzaElencoPubblnsDaUtente(int,int):ArrayList<ObservablePubblicazioni>
getPubb(int):Pubblicazione
ricercaPerISBN(String):ArrayList<ObservablePubblicazioni>
ricercaPerTitolo(int,String):ArrayList<ObservablePubblicazioni>
ricercaPerParolaChiave(int,String):ArrayList<ObservablePubblicazioni>
ricercaPerAutore(int,String,String):ArrayList<ObservablePubblicazioni>
visualizzaElencoPubbConStessiAutori(int,int):ArrayList<ObservablePubblicazioni>
inserisciPubb(int,String,String,String,LocalDate,int,String,String,String):int
modificaPubblicazione(int,String,String,int):boolean
rimuoviPubblicazione(int,int):void
ritornaListaPubblicazioni(ResultSet):ArrayList<ObservablePubblicazioni>
esisteISBN(String):boolean
<<Java Class>>
MySQLDAOLikeImpl
DAO.MySQLImplementation
MySQLDAOLikeImpl()
utenteLikePubb(int,int):boolean
aggiungiLikeAPubb(int,int):boolean
rimuoviLikeAPubb(int,int):void
<<Java Class>>
MySQLDAOParolaChiaveImpl
DAO.MySQLImplementation
MySQLDAOParolaChiaveImpl()
controllaParolaChiave(String):int
inserisciParolaChiave(String):int
aggiungiParolaChiaveAPubb(int,int,int):boolean
rimuoviParolaChiaveAPubb(int,int,int):void
modificaParolaChiave(int,String):void
cancellaParolaChiave(int,int):void
getListaParoleChiave(int):ArrayList<ParolaChiave>
parolaChiaveTagPubb(int,int):boolean
visualizzaListaParolaChiavediPubb(int,int):ArrayList<ParolaChiave>
visualizzaListaParolaChiaveCompleta(int):ArrayList<ParolaChiave>
ritornaListaParoleChiave(ResultSet):ArrayList<ParolaChiave>


<<Java Class>>
MySQLDAORistampaImpl
DAO.MySQLImplementation
MySQLDAORistampaImpl()
inserisciRistampa(int,LocalDate,int,int):void
eliminaRistampa(int,int):void
modificaRistampa(int,String,String):boolean
getListaRistampe(int):ArrayList<Ristampa>
getUltimaRistampa(int):Ristampa
visualizzaListaRistampe(int,int):ArrayList<Ristampa>
ritornaListaRistampe(ResultSet):ArrayList<Ristampa>
<<Java Class>>
MySQLDAORecensioneImpl
DAO.MySQLImplementation
MySQLDAORecensioneImpl()
inserisciRecensione(int,int,String):void
eliminaRecensione(int,int,int):void
approvaRecensione(int,int,int):void
utenteInseritoRecensione(int,int):boolean
ritornaListaRecensioni(ResultSet):ArrayList<Recensione>
visualizzaListaRecensioniApprovatediPubb(int,int):ArrayList<Recensione>
visualizzaListaRecensioniNonApprovate(int):ArrayList<Recensione>
visualizzaListaRecensionidiUtente(int,int):ArrayList<Recensione>
<<Java Class>>
MySQLDAOAutoreImpl
DAO.MySQLImplementation
MySQLDAOAutoreImpl()
controllaAutore(String,String):int
controllaAutore(String):int
inserisciAutore(String,String):int
inserisciAutore(String):int
aggiungiAutoreAPubb(int,int,int):boolean
rimuoviAutoreAPubb(int,int,int):void
modificaAutore(int,String,String):boolean
cancellaAutore(int,int):void
getListaAutori(int):ArrayList<Autore>
autoreScrittoPubb(int,int):boolean
visualizzaListaAutoridiPubb(int,int):ArrayList<Autore>
visualizzaListaAutoriCompleta(int):ArrayList<Autore>
ritornaListaAutori(ResultSet):ArrayList<Autore>

sono state create una classe DAO ed una Classe controller per ogni entità del sistema, in modo da dividere le operazioni su diverse entità.

Il package DAO contiene inoltre la classe connector, che contiene il metodo per aprire la connessione con il database, ed il metodo per l'inizializzazione degli oggetti DAO all'interno dei Controller.

View

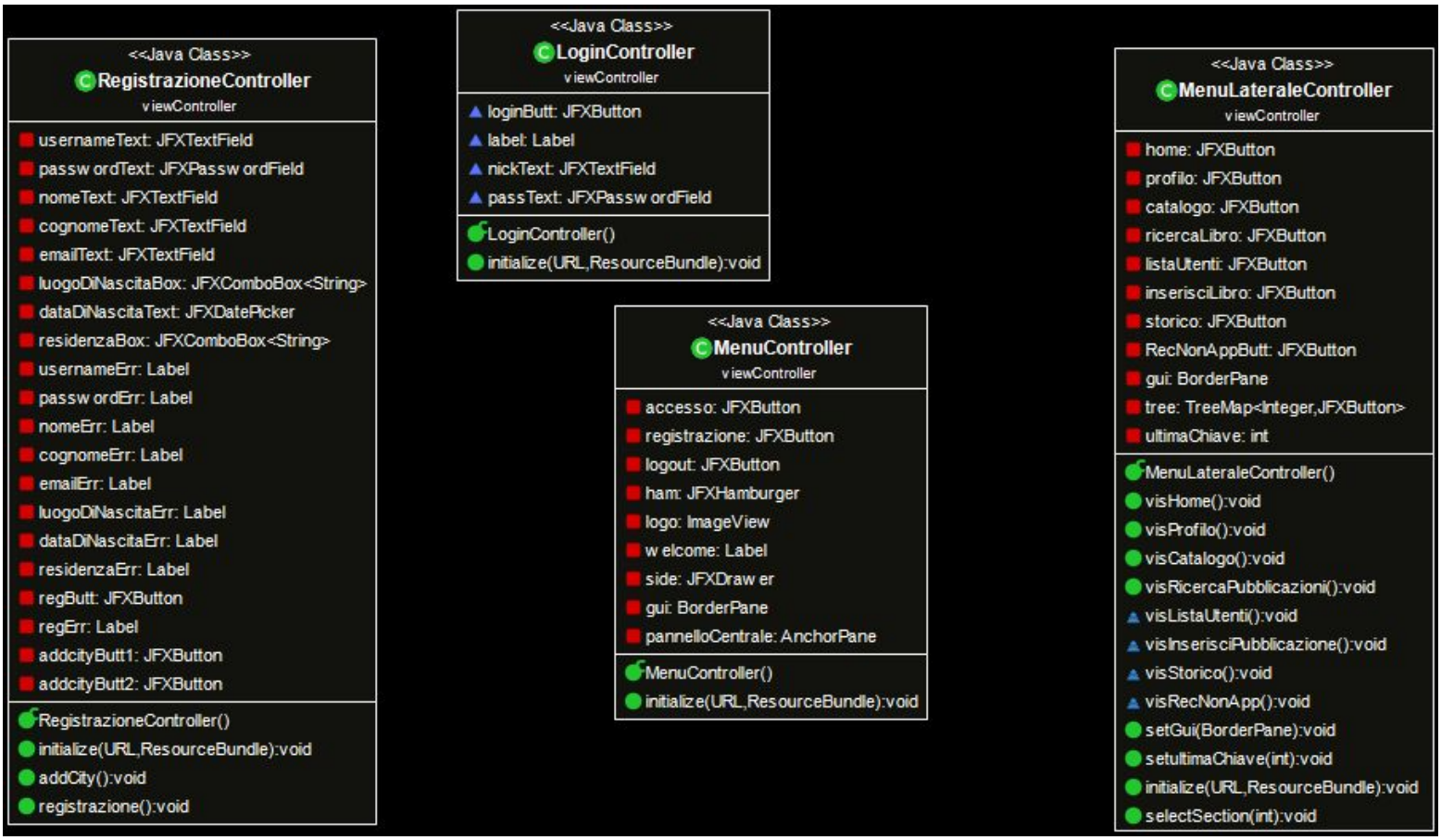
Questo package contiene tutti i file .fxml delle interfacce della GUI.

Le interfacce sono state sviluppate tramite il programma **Scene Builder**, che ha permesso la modifica e la creazione della vista in maniera interattiva.

E' stato utilizzato il design pattern **Decorator** tramite @FXML per connettere gli elementi delle interfacce grafiche al rispettivo controller e aggiungerli caratteristiche e funzioni.

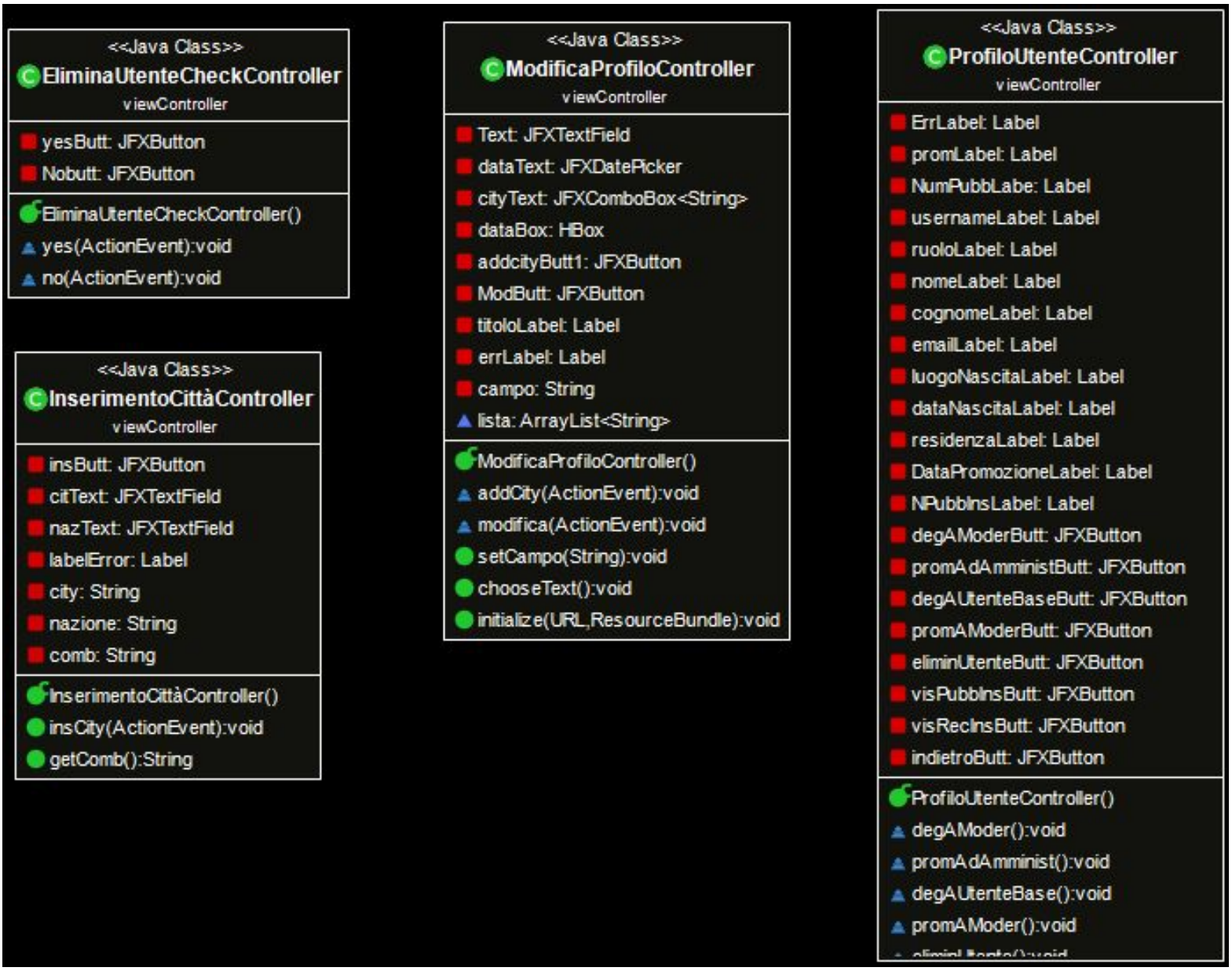
ViewController

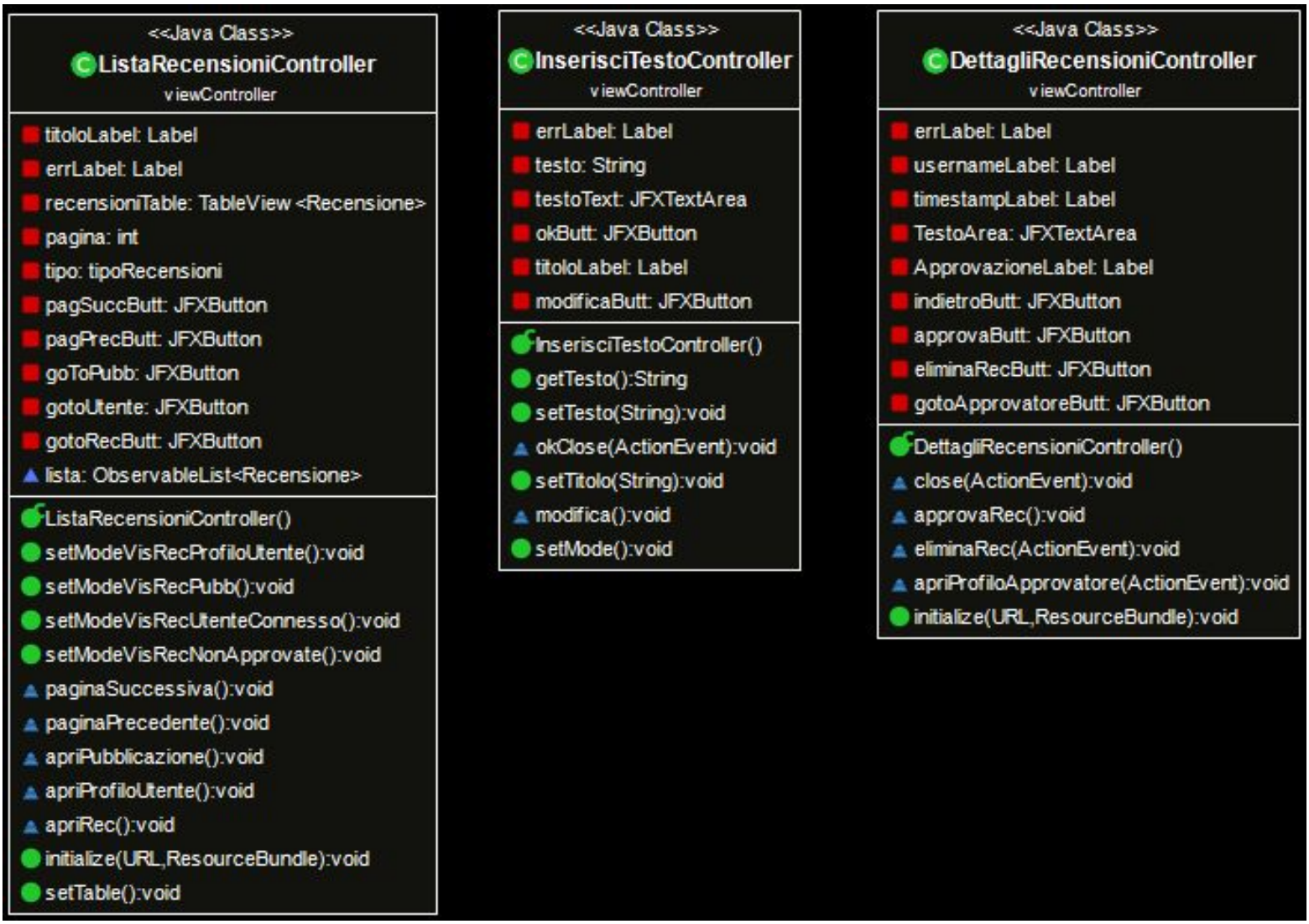
Qui sono contenute le classi dei controller della view, uno per ogni interfaccia sviluppata, permettono di modificare la GUI in base al grado dell'utente connesso, mostrando cosa può fare e nascondendo le azioni per cui non ha i permessi necessari.


<<Java Class>>
RegistrazioneController
viewController
usernameText: JFXTextField
passwordText: JFXPasswordField
nomeText: JFXTextField
cognomeText: JFXTextField
emailText: JFXTextField
luogoDiNascitaBox: JFXComboBox<String>
dataDiNascitaText: JFXDatePicker
residenzaBox: JFXComboBox<String>
usernameErr: Label
passwordErr: Label
nomeErr: Label
cognomeErr: Label
emailErr: Label
luogoDiNascitaErr: Label
dataDiNascitaErr: Label
residenzaErr: Label
regButt: JFXButton
regErr: Label
addcityButt1: JFXButton
addcityButt2: JFXButton
RegistrazioneController()
initialize(URL,ResourceBundle):void
addCity():void
registrazione():void
<<Java Class>>
LoginController
viewController
loginButt: JFXButton
label: Label
nickText: JFXTextField
passText: JFXPasswordField
LoginController()
initialize(URL,ResourceBundle):void
<<Java Class>>
MenuController
viewController
accesso: JFXButton
registrazione: JFXButton
logout: JFXButton
ham: JFXHamburger
logo: ImageView
welcome: Label
side: JFXDrawer
gui: BorderPane
pannelloCentrale: AnchorPane
MenuController()
initialize(URL,ResourceBundle):void
<<Java Class>>
MenuLateraleController
viewController
home: JFXButton
profilo: JFXButton
catalogo: JFXButton
ricercaLibro: JFXButton
listaUtenti: JFXButton
inserisciLibro: JFXButton
storico: JFXButton
RecNonAppButt: JFXButton
gui: BorderPane
tree: TreeMap<Integer,JFXButton>
ultimaChiave: int
MenuLateraleController()
visHome():void
visProfilo():void
visCatalogo():void
visRicercaPubblicazioni():void
visListaUtenti():void
visInserisciPubblicazione():void
visStorico():void
visRecNonApp():void
setGui(BorderPane):void
setultimaChiave(int):void
initialize(URL,ResourceBundle):void
selectSection(int):void


<<Java Class>>
ListaUtentiController
viewController
ErrLabel: Label
utentiTable: TableView <UtenteBase>
moderatoriTable: TableView <UtenteModeratore>
pagina: int
pagSuccButt: JFXButton
pagPrecButt: JFXButton
gotoUtente: JFXButton
collaborativiButt: JFXButton
alfOrdButt: JFXButton
listaUtenti: ObservableList<UtenteBase>
listaMod: ObservableList<UtenteModeratore>
ListaUtentiController()
paginaSuccessiva():void
paginaPrecedente():void
apriProfiloUtente():void
collaborativi():void
ordineAlf():void
initialize(URL,ResourceBundle):void
setUtentiTable():void
setModeratoriTable():void
<<Java Class>>
UtenteConnessoController
viewController
ErrLabel: Label
usernameLabel: Label
ruoloLabel: Label
nomeLabel: Label
cognomeLabel: Label
emailLabel: Label
luogoNascitaLabel: Label
dataNascitaLabel: Label
residenzaLabel: Label
modUsernameButt: JFXButton
modPassw ordButt: JFXButton
modNomeButt: JFXButton
modCognomeButt: JFXButton
modEmailButt: JFXButton
modLuogoNascitaButt: JFXButton
modDataNascitaButt: JFXButton
modResidenzaButt: JFXButton
visRecInsButt: JFXButton
UtenteConnessoController()
modUsername(ActionEvent):void
modPassw ord(ActionEvent):void
modNome(ActionEvent):void
modCognome(ActionEvent):void
modEmail(ActionEvent):void
modDataNascita(ActionEvent):void
modLuogoNascita(ActionEvent):void
modResidenza(ActionEvent):void
modificaUtente(String):void
visRecIns(ActionEvent):void
initialize(URL,ResourceBundle):void
updateProfilo():void
<<Java Class>>
CatalogoController
viewController
ListaPubbTable: TableView <ObservablePubblicazioni>
pagSuccButt: JFXButton
pagPrecButt: JFXButton
goToPubb: JFXButton
ErrLabel: Label
catalogoButt: JFXButton
inserimentoButt: JFXButton
AggiornateButt: JFXButton
conDow nloadButt: JFXButton
pagina: int
lista: ObservableList<ObservablePubblicazioni>
CatalogoController()
paginaSuccessiva():void
paginaPrecedente():void
apriPubblicazione():void
visCatalogo():void
visInserimento():void
visAggiornate():void
visConDow nload():void
initialize(URL,ResourceBundle):void
setTable():void
<<Java Class>>
InserisciPubblicazioneController
viewController
ISBNText: JFXTextField
titoloText: JFXTextField
editoreText: JFXTextField
dataPubbPicker: JFXDatePicker
numPaggText: JFXTextField
linguaBox: JFXComboBox<String>
insElemLabel: Label
ErrLabel: Label
insPubbButt: JFXButton
indice: String
insIndiceButt: JFXButton
descrizione: String
insDescrizioneButt: JFXButton
AutoriTable: TableView <Autore>
insAutoreButt: JFXButton
paroleTable: TableView <ParolaChiave>
insParolaButt: JFXButton
sorgentiTable: TableView <Sorgente>
insSorgenteButt: JFXButton
ristampeTable: TableView <Ristampa>
insRistampaButt: JFXButton
dataRistampaPicker: JFXDatePicker
firstText: JFXTextField
secondText: JFXTextField
tipoBox: JFXComboBox<String>
formatoBox: JFXComboBox<String>
insErrLabel: Label
listaAutori: ArrayList<Autore>
listaParole: ArrayList<ParolaChiave>
listaSorgenti: ArrayList<Sorgente>
listaRistampe: ArrayList<Ristampa>
insElementButt: JFXButton
inserisciPubblicazioneController()
insPubb():void
insIndice():void
insDescrizione():void
inserisciTesto(String):void
insAutore():void
insParola():void
insSorgente():void
insRistampa():void
insElement():void
initialize(URL,ResourceBundle):void
<<Java Class>>
RicercaPubblicazioniController
viewController
ListaPubbTable: TableView <ObservablePubblicazioni>
ErrLabel: Label
pagina: int
tipo: tipoRicerca
gui: BorderPane
pagSuccButt: JFXButton
pagPrecButt: JFXButton
goToPubb: JFXButton
isbnRicButt: JFXButton
titoloRicButt: JFXButton
parolaRicButt: JFXButton
autoreRicButt: JFXButton
ricercaHBox: HBox
cercaText: JFXTextField
CognomeAutoreText: JFXTextField
noElemLabel: Label
CercaButt: JFXButton
lista: ObservableList<ObservablePubblicazioni>
RicercaPubblicazioniController()
setGui(BorderPane):void
paginaSuccessiva():void
paginaPrecedente():void
apriPubblicazione():void
ricercaPerIsbn():void
ricercaPerTitolo():void
ricercaPerParola():void
ricercaPerAutore():void
Ricerca():void
initialize(URL,ResourceBundle):void
setTable():void
<<Java Class>>
StoricoController
viewController
StoricoTableTable: TableView <EntryStoria>
ErrLabel: Label
pagina: int
pagSuccButt: JFXButton
pagPrecButt: JFXButton
goToPubb: JFXButton
gotoUtente: JFXButton
lista: ObservableList<EntryStoria>
StoricoController()
paginaSuccessiva():void
paginaPrecedente():void
apriPubblicazione():void
apriProfiloUtente():void
setTable():void
initialize(URL,ResourceBundle):void


<<Java Class>>
EliminaPubblicazioneCheckController
viewController
Nobutt: JFXButton
yesButt: JFXButton
EliminaPubblicazioneCheckController()
no(ActionEvent):void
yes(ActionEvent):void
<<Java Class>>
StoricoPubblicazioneController
viewController
pagina: int
storicoPubbTable: TableView <EntryStoria>
titoloLabel: Label
errLabel: Label
pagSuccButt: JFXButton
pagPrecButt: JFXButton
goToPubb: JFXButton
goToUtente: JFXButton
lista: ObservableList<EntryStoria>
StoricoPubblicazioneController()
paginaSuccessiva():void
paginaPrecedente():void
apriPubblicazione():void
apriProfiloUtente():void
initialize(URL,ResourceBundle):void
setTable():void
<<Java Class>>
GestisciTableController
viewController
autoriTable: TableView <Autore>
paroleTable: TableView <ParolaChiave>
ristampeTable: TableView <Ristampa>
sorgentiTable: TableView <Sorgente>
dataRistampaPicker: JFXDatePicker
firstText: JFXTextField
tipoBox: JFXComboBox<String>
secondText: JFXTextField
formatoBox: JFXComboBox<String>
errLabel: Label
campo: String
titoloLabel: Label
cancRow Butt: JFXButton
s: Sorgente
r: Ristampa
modRow Butt: JFXButton
indietroModRow Butt: JFXButton
okModRow Butt: JFXButton
insRow Butt: JFXButton
GestisciTableController()
cancRow ():void
modRow ():void
indietroModRow ():void
okModRow ():void
insRow ():void
setTitolo(String):void
initialize(URL,ResourceBundle):void
checkUtente():void
<<Java Class>>
ProfiloPubblicazioneController
viewController
modTitoloButt: JFXButton
modAutoriButt: JFXButton
modEditoreButt: JFXButton
modParoleButt: JFXButton
modIsbnButt: JFXButton
modDataPubbButt: JFXButton
modNPagButt: JFXButton
modLinguaButt: JFXButton
titoloLabel: Label
autoriLabel: Label
editoreLabel: Label
paroleLabel: Label
isbnLabel: Label
nLikeLabel: Label
nRecensioniLabel: Label
dataPubbLabel: Label
dataUltModLabel: Label
nPagLabel: Label
linguaLabel: Label
eliminaPubbButt: JFXButton
visDescrizioneButt: JFXButton
visIndiceButt: JFXButton
visRistampeButt: JFXButton
visSorgentiButt: JFXButton
visRecensioniButt: JFXButton
visStoricoButt: JFXButton
insLikeButt: JFXButton
delLikeButt: JFXButton
<<Java Class>>
ListaPubblicazioniUtenteController
viewController
pagina: int
ListaPubbTable: TableView <ObservablePubblicazioni>
titoloLabel: Label
errLabel: Label
pagSuccButt: JFXButton
pagPrecButt: JFXButton
goToPubbButt: JFXButton
goToUtenteButt: JFXButton
lista: ObservableList<ObservablePubblicazioni>
ListaPubblicazioniUtenteController()
paginaSuccessiva():void
paginaPrecedente():void
apriPubblicazione():void
tornaAlProfiloUtente():void
initialize(URL,ResourceBundle):void
setTable():void


<<Java Class>>
EliminaUtenteCheckController
viewController
yesButt: JFXButton
Nobutt: JFXButton
EliminaUtenteCheckController()
yes(ActionEvent):void
no(ActionEvent):void
<<Java Class>>
InserimentoCittàController
viewController
insButt: JFXButton
citText: JFXTextField
nazText: JFXTextField
labelError: Label
city: String
nazione: String
comb: String
InserimentoCittàController()
insCity(ActionEvent):void
getComb():String
<<Java Class>>
ModificaProfiloController
viewController
Text: JFXTextField
dataText: JFXDatePicker
cityText: JFXComboBox<String>
dataBox: HBox
addcityButt1: JFXButton
ModButt: JFXButton
titoloLabel: Label
errLabel: Label
campo: String
lista: ArrayList<String>
ModificaProfiloController()
addCity(ActionEvent):void
modifica(ActionEvent):void
setCampo(String):void
chooseText():void
initialize(URL,ResourceBundle):void
<<Java Class>>
ProfiloUtenteController
viewController
ErrLabel: Label
promLabel: Label
NumPubbLabe: Label
usernameLabel: Label
ruoloLabel: Label
nomeLabel: Label
cognomeLabel: Label
emailLabel: Label
luogoNascitaLabel: Label
dataNascitaLabel: Label
residenzaLabel: Label
DataPromozioneLabel: Label
NPubblnsLabel: Label
degAModerButt: JFXButton
promAdAmministButt: JFXButton
degAUtenteBaseButt: JFXButton
promAModerButt: JFXButton
eliminUtenteButt: JFXButton
visPubblnsButt: JFXButton
visRecInsButt: JFXButton
indietroButt: JFXButton
ProfiloUtenteController()
degAModer():void
promAdAmminist():void
degAUtenteBase():void
promAModer():void


<<Java Class>>
ListaRecensioniController
viewController
titoloLabel: Label
errLabel: Label
recensioniTable: TableView <Recensione>
pagina: int
tipo: tipoRecensioni
pagSuccButt: JFXButton
pagPrecButt: JFXButton
goToPubb: JFXButton
gotoUtente: JFXButton
gotoRecButt: JFXButton
lista: ObservableList<Recensione>
ListaRecensioniController()
setModeVisRecProfiloUtente():void
setModeVisRecPubb():void
setModeVisRecUtenteConnesso():void
setModeVisRecNonApprovate():void
paginaSuccessiva():void
paginaPrecedente():void
apriPubblicazione():void
apriProfiloUtente():void
apriRec():void
initialize(URL,ResourceBundle):void
setTable():void
<<Java Class>>
InserisciTestoController
viewController
errLabel: Label
testo: String
testoText: JFXTextArea
okButt: JFXButton
titoloLabel: Label
modificaButt: JFXButton
InserisciTestoController()
getTesto():String
setTesto(String):void
okClose(ActionEvent):void
setTitolo(String):void
modifica():void
setMode():void
<<Java Class>>
DettagliRecensioniController
viewController
errLabel: Label
usernameLabel: Label
timestampLabel: Label
TestoArea: JFXTextArea
ApprovazioneLabel: Label
indietroButt: JFXButton
approvaButt: JFXButton
eliminaRecButt: JFXButton
gotoApprovatoreButt: JFXButton
DettagliRecensioniController()
close(ActionEvent):void
approvaRec():void
eliminaRec(ActionEvent):void
apriProfiloApprovatore(ActionEvent):void
initialize(URL,ResourceBundle):void

Descrizione dell'Interfaccia grafica

Definiremo ora per ogni Interfaccia la sua principale funzione nel programma.

Home

La schermata di benvenuto, appare all'apertura del programma.

Login

Finestra per permettere agli utenti di eseguire l'accesso al sistema.

Menu

Menu principale del sistema, sempre visibile, da cui è possibile accedere al Menu Laterale, contiene inoltre i bottoni per accedere,registrarsi e uscire.

MenuLaterale

Menu sul lato destro, permette all'utente di navigare tra le varie sezioni del programma.

Registrazione

Permette all'utente di inserire i propri dati per registrarsi.

Catalogo

Qui è possibile accedere alla lista completa delle pubblicazioni inserite.

UtenteConnesso

Qui l'utente che ha eseguito l'accesso può visualizzare e modificare le sue informazioni personali, può inoltre cambiare password e visualizzare le recensioni che ha scritto.

ModificaProfilo

finestra che permette all'utente di modificare i propri dati.

si modifica in base al tipo di informazione che l'utente deve inserire (data,città o stringa)

InserimentoCittà

finestra che permette all'utente di aggiungere una città tra quelle del sistema.

RicercaPubblicazioni

Sezione che permette la ricerca di una pubblicazione tra quelle del catalogo tramite ISBN,autore,parola chiave o titolo.

InserisciPubblicazione

Qui un utente potrà inserire una nuova pubblicazione nel sistema per renderla visualizzabile agli altri utenti.

ListaUtenti

Sezione del programma in cui è possibile visualizzare la lista completa degli utenti iscritti al catalogo.

Storico

Sezione che permette di visualizzare tutta la storia delle modifiche effettuate dagli utenti sulle pubblicazioni.

StoricoPubblicazione

Storico delle modifiche di una particolare pubblicazione, contiene le 'entry' di modifica solo della pubblicazione visualizzata

ProfiloPubblicazione

Finestra che contiene tutti i dati di una pubblicazione, permette anche di accedere alle sue recensioni e allo storico delle sue modifiche.

EliminaPubblicazioneCheck

Finestra di allarme che si apre nel momento che un utente cerca di eliminare una pubblicazione.

GestisciTable

finestra che permette di modificare le liste (autori,parole,sorgenti e ristampe) associate ad una pubblicazione.

ProfiloUtente

Finestra che contiene tutti i dati di un utente, permette anche di accedere alle sue recensioni.

Qui è possibile anche eliminare l'utente oppure modificare il suo grado.

EliminaUtenteCheck

Finestra di allarme che si apre nel momento che un utente cerca di eliminare un'altro utente.

ListaRecensioni

Lista di recensioni, può contenere la lista di recensioni di una pubblicazione, scritte da un utente, oppure la lista delle recensioni non approvate.

DettagliRecensione

Finestra in cui è possibile visualizzare tutte le informazioni di una recensione.

Ci si accede da ListaRecensioni, aprendo una delle recensioni.

InserisciTesto

Finestra generica che permette all'utente di inserire un testo, esso può essere l'indice, una descrizione, oppure una recensione.

ListaPubblicazioniUtente

Contiene la lista di Pubblicazioni inserite da un determinato utente.

Tecnologie Utilizzate

Database

Il DBMS utilizzato è **MySQL.**
Il database è stato generato e utilizzato su **MySQL Server**.
Per la progettazione logica e fisica del database, stesura di query, procedure e codice SQL è stato utilizzato **MySQL Workbench**.

Applicativo

Per la scrittura del codice è la generazione dei file UML e delle javadoc è stato utilizzato **Eclipse**.

Per alcuni Diagrammi UML è stato utilizzato **Draw.IO.**

Per la creazione della GUI del sistema è stato utilizzato **JavaFX** assieme a **Scene Builder** e **Css** e la libreria **JFoenix**.

Per la connessione dell'applicativo al database è stato utilizzato il driver **JDBC** nella sua versione 8.0.16 **.**