



UNIVERSITÀ
DEGLI STUDI
DE L'AQUILA



Dipartimento di Ingegneria e Scienze
dell'Informazione e Matematica

Università degli Studi dell'Aquila

UnivAQ Library

2019/2020

Corso: *Object Oriented Software Design*

Docente: *Romina Eramo*



Federico Di Menna **253962**

federico.dimenna@student.univaq.it



INDICE

[INDICE](#)

[SPECIFICA](#)

[INTRODUZIONE](#)

[Utente](#)

[Pubblicazione](#)

[REQUISITI](#)

[Documento dei Requisiti](#)

[Actors](#)

[Requisiti Funzionali](#)

[Requisiti Non Funzionali](#)

[Use Case Diagram](#)

[Descrizione Use Case](#)

[SYSTEM DESIGN](#)

[ARCHITETTURA DEL SISTEMA](#)

[Distinzione tra Controller Back-End e Front-End](#)

[DAO - Architettura](#)

[Controller Di Back-End](#)

[Descrizione delle entità](#)

[Pubblicazione](#)

[Utente](#)

[Recensione](#)

[Like](#)

[Sorgente](#)

[Metadati](#)

[Storia](#)

[Ruolo](#)

[Class Diagrams](#)

[Model Class Diagram](#)

[Controller Class Diagram](#)



UNIVERSITÀ
DEGLI STUDI
DE L'AQUILA

[Back End Controller Class Diagram](#)
[Front End Controller Class Diagram](#)
[DAO Class Diagram](#)
[Interfaces](#)
[MySQL Implementation Class Diagram](#)
[Main Class Diagram](#)

TECNOLOGIE UTILIZZATE

INTERFACCIA GRAFICA

[Root Layout](#)
[Splash Screen](#)
[Registrazione](#)
[Login](#)
[Pannello Gestione](#)
[Profilo Utente](#)
[Classifiche](#)
[Gestione Utenti](#)
[Logs](#)
[Ricerca](#)
[Inserimento Pubblicazione](#)
[Aggiungi Autori](#)
[Aggiungi Parole Chiave](#)
[Aggiungi Ristampe](#)
[Aggiungi Sorgenti](#)
[Visualizza Catalogo](#)
[Visualizza Pubblicazione](#)
[Visualizza Pubblicazioni con gli stessi autori](#)
[Scrivi Recensione](#)
[Approva Recensioni](#)
[Informazioni Generali](#)

SPECIFICA

La Biblioteca rappresenta un semplice gestore di catalogo bibliografico online. Il catalogo conterrà una serie di pubblicazioni, ciascuna delle quali caratterizzata da un titolo, una lista di autori, un editore, una serie di metadati (vedi dopo) e, optionalmente, una descrizione testuale (sunto del contenuto o presentazione della pubblicazione), un indice (composto dai titoli dei vari capitoli/sezioni della pubblicazione, numerati e ordinati) e una serie di sorgenti tramite le quali poter accedere alla pubblicazione o a parti di essa. Ciascuna sorgente sarà caratterizzata da un tipo, una URI, da un formato e da una descrizione. Esempi di sorgenti valide potrebbero essere i seguenti:

- tipo="immagine", URI="http://server.net/cover.jpg", formato="image/jpeg", descrizione="copertina"
- tipo="download", URI="http://server.net/book.pdf", formato="application/pdf", descrizione="versione elettronica gratuita"
- tipo="acquisto", URI="http://www.amazon.it/xyz", formato="cartaceo, copertina rigida", descrizione="acquista online"

Per quanto riguarda invece i metadati, questi saranno costituiti (almeno) dalle seguenti informazioni: codice ISBN, numero di pagine, lingua, data di pubblicazione, lista delle ristampe (numero e data) e parole chiave identificative (zero o più). Ciascuna pubblicazione potrà essere associata a un certo numero di recensioni testuali, le quali dovranno anche indicare l'utente che le ha inserite e la data/ora dell'inserimento. Le recensioni saranno moderate, quindi è necessario prevedere un flag che indichi se la recensione è stata approvata oppure no dai moderatori. Infine, gli utenti potranno anche assegnare il loro like alle pubblicazioni, che quindi dovranno essere associate anche alla lista dei like ricevuti, con utente e data.

Sarà inoltre necessario prevedere opportune strutture per immagazzinare i dati dell'utenza. Il sistema, infatti, prevede due tipologie di utenza: attiva e passiva. Gli utenti potranno registrarsi nel sito fornendo i loro dati anagrafici, un indirizzo email (che verrà usato anche come username) e, ovviamente, la password scelta. Il livello di utenza iniziale sarà sempre quello passivo.

Per tener traccia delle modifiche effettuate alla bibliografia dai vari utenti attivi, il sistema dovrà mantenere una "storia" di ciascuna scheda bibliografica. Ogni entry di tale storia conterrà un timestamp, il nome dell'utente e una descrizione della modifica. Potete scegliere voi il dettaglio di questa descrizione: di base, sono accettabili anche descrizioni del tipo "ha modificato la pubblicazione", "ha inserito la pubblicazione", "ha approvato una recensione dell'utente X", ecc.



INTRODUZIONE

Il sistema verte su due concetti fondamentali che interagiscono: **Utente** e **Pubblicazione**.

Utente

L'utente è l'utilizzatore del nostro sistema. Abbiamo determinati ruoli di utenti che si occupano della gestione oltre che l'utilizzo dell'applicativo, e altri che si occupano soltanto di utilizzarlo.

I ruoli che il sistema gestisce sono: **Utente Anonimo**, **Utente (Attivo / Passivo)**, **Moderatore**, **Amministratore**, **Super Amministratore** che verranno illustrati meglio in seguito.

Pubblicazione

Una pubblicazione è una scheda bibliografica di un libro, opera letteraria, e tutto quello che può essere identificato mediante codice ISBN (*International Standard Book Number*). Le pubblicazioni vengono inserite nel catalogo dagli utenti (di grado moderatore o superiore).

Il sistema gestisce inoltre le interazioni che ci sono tra le due classi di oggetti (likes, recensioni...).

REQUISITI

Documento dei Requisiti

Actors

Gli attori definiti per il nostro sistema sono, in ordine di rilievo, i seguenti (Ogni utente di grado superiore eredita i privilegi dei sottostanti):

Utente Anonimo : Visitatore anonimo del nostro sistema che non ha effettuato nessun accesso. Un utente anonimo all'interno del sistema può eseguire l'accesso oppure registrarsi al sistema. Inoltre ha la possibilità di visualizzare elenco e dettagli delle pubblicazioni ed eseguire una ricerca.

Utente (Attivo / Passivo) : Gli Utenti (attivi / passivi) sono quegli utenti che sono registrati al sistema come utilizzatori e quindi non si occupano della gestione di tale sistema. Tali utenti possono inserire i propri

feedback (quindi recensioni e like relativi a pubblicazioni). Inoltre possono gestire il proprio profilo e visualizzare le classifiche. La distinzione tra utente attivo e utente passivo è di carattere statistico, e non si riflette sulle capabilities del ruolo. Tale distinzione viene fatta nel seguente modo

- **Utente Passivo** : Utenti registrati che non hanno mai inserito una recensione oppure non hanno inserito una recensione negli ultimi due mesi.
- **Utente Attivo** : Utenti registrati che hanno inserito una recensione negli ultimi due mesi.

Moderatore : Prima categoria di utenti che si occupa della gestione del sistema oltre che del suo utilizzo. Infatti essi sono abilitati all'inserimento e alla modifica delle pubblicazioni e all'approvazione ed eliminazione delle recensioni di altri utenti. Gli utenti di grado moderatore o superiore vengono considerati sempre attivi indipendentemente dall'inserimento delle recensioni.

Amministratore : Utente che possiedono privilegi di cancellazione di una pubblicazione e di gestione di un utente (promozione da utente a moderatore e degradazione).

Super Amministratore : Ruolo che possiede un solo utente del sistema, ovvero l'amministratore principale. Tale utente ha tutte le capabilities che fornisce il sistema. In particolare gestisce tutte le tipologie di utenti, promuovendo e degradando anche gli amministratori.

Requisiti Funzionali

Date le funzionalità espresse dalla specifica di progetto, i requisiti funzionali sono i seguenti.

Il sistema fornisce (*numero requisito della specifica*) :

- un pannello di gestione utenti che permetterà agli utenti amministratori e super amministratore di promuovere e degradare gli utenti, e un meccanismo automatizzato per permettere ad un utente di passare da attivo a passivo e viceversa (1) in base ai criteri stabiliti. (Questa procedura è implementata con un evento a livello database)
- una sezione dedicata alle classifiche che illustra le ultime 10 pubblicazioni inserite (2), quelle aggiornate negli ultimi 30 giorni (3) e gli utenti che hanno inserito più pubblicazioni (4)
- una pagina che illustra il profilo di un utente illustrandone le proprie caratteristiche e le pubblicazioni inserite (5)
- una sezione dedicata al catalogo completo (6) gestito mediante un meccanismo di paginazione dell'intera raccolta, con un filtro per la visualizzazione di quelli con possibilità di download (16). Il catalogo contiene le informazioni sostanziali come codice ISBN, Titolo ecc... e in aggiunta la data di ultima ristampa (17)
- una pagina che espone i dettagli di una pubblicazione in particolare mostrano oltre alle sue



caratteristiche anche il numero dei like (12), le recensioni approvate (13) che si riferiscono alla stessa e la possibilità di avere l'elenco di tutte le pubblicazioni del catalogo con gli stessi autori (18)

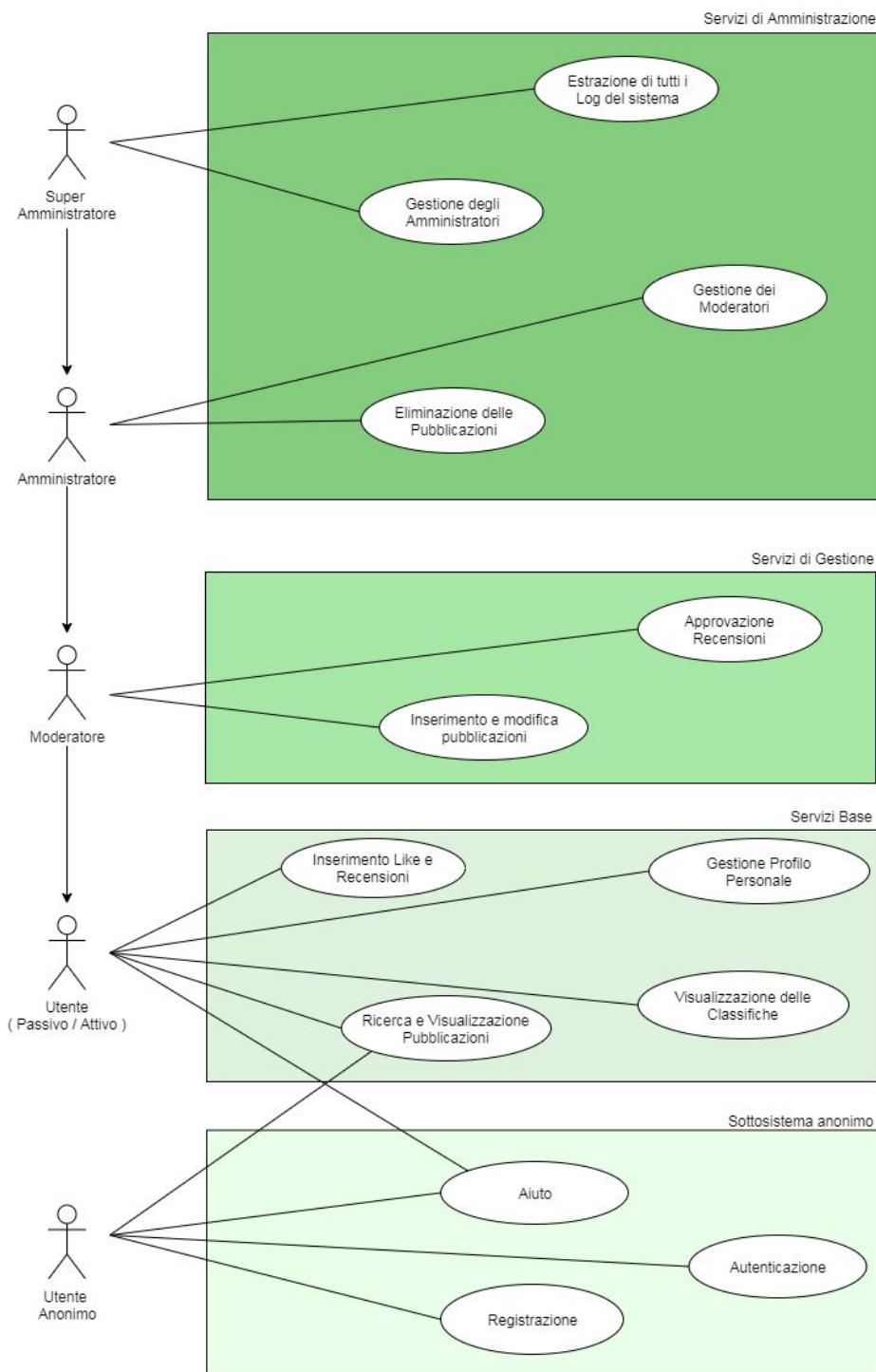
- un meccanismo di ricerca di pubblicazioni per ID (7), ISBN, titolo, autore, parole chiave (8)
- un pannello che permette l'inserimento di una pubblicazione con tutte le sue caratteristiche, comprese ristampe, parole chiave, autori e sorgenti
- una sezione sotto la pubblicazione in dettaglio che permette l'inserimento di una recensione (9) legata ad essa e di inserimento e rimozione like (11)
- un pannello dedicato agli utenti di grado moderatore o superiore che espone tutte le recensioni in lista di approvazione (14) e permette l'approvazione o l'eliminazione delle recensioni (10)
- una pagina dedicata al super amministratore che permette la visualizzazione del log di sistema contenente tutte le modifiche relative a una pubblicazione (15)
- una pagina illustrativa sul progetto

Requisiti Non Funzionali

TIPO DI REQUISITO	DETTAGLI
Availability	Il sistema deve essere accessibile ogni giorno a tempo pieno per dare la possibilità agli utenti utilizzare il sistema
Security	L'accesso per utilizzare propriamente il sistema deve essere gestito mediante autenticazione con username e password. Quest'ultima non deve essere inserita in chiaro nel database ma deve essere cifrata
Usability	Il sistema deve essere dotato di un'interfaccia grafica che sia di semplice utilizzo e comprensione anche per persone senza confidenza con il mondo digitale. La fase di avvio dell'applicazione non deve durare più di un 30 secondi e deve essere arricchita con un'animazione di splash
Efficiency	Il sistema deve essere progettato in modo da gestire un elevato numero di pubblicazioni ed utenti, mantenendo dei tempi di risposta brevi. A tal proposito si propone un meccanismo di paginazione per la gestione del catalogo
Integrity	Il sistema deve garantire l'autenticità dei commenti inseriti



Use Case Diagram





Descrizione Use Case

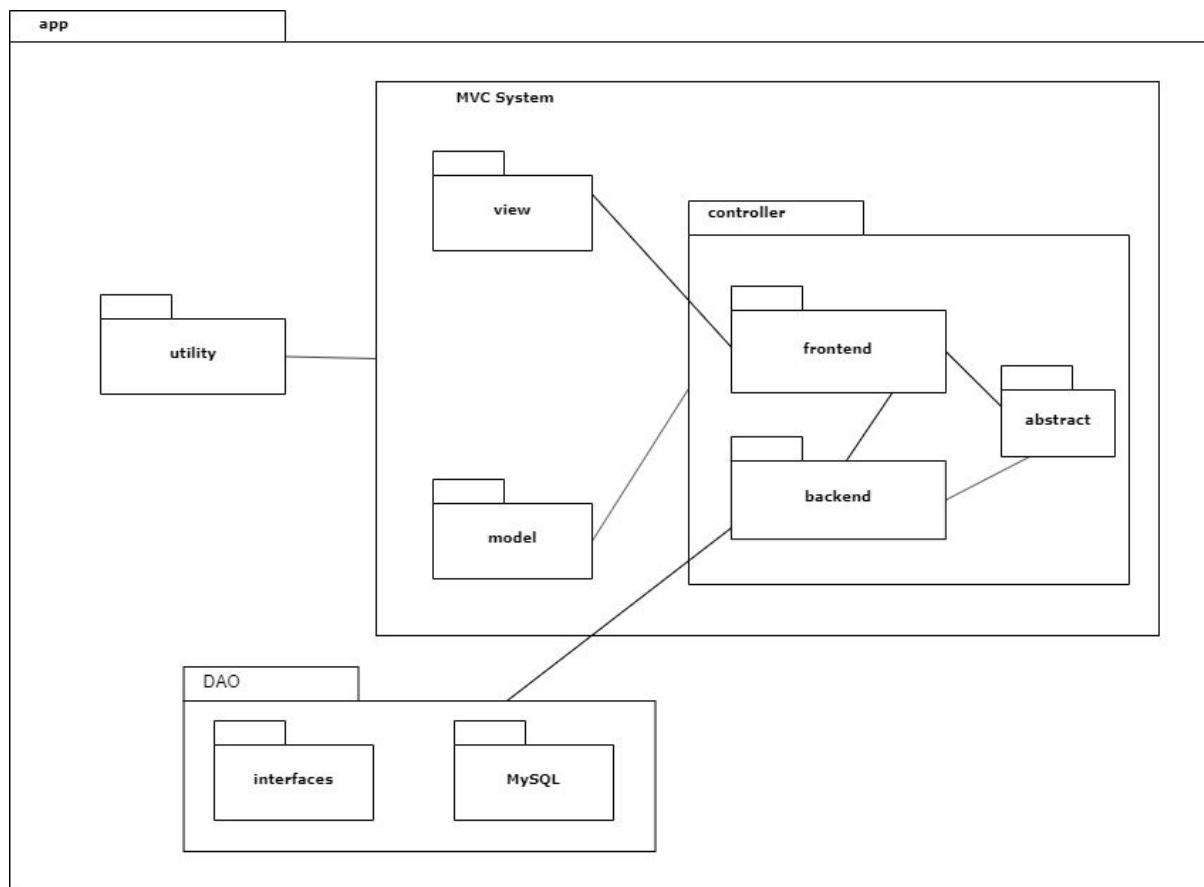
Task	Descrizione
Registrazione	<i>Modulo nel quale l'utente anonimo inserirà i propri dati quali e verrà registrato nel sistema, e automaticamente avrà accesso alle funzionalità riservate all'utente normale</i>
Autenticazione	<i>Pagina nella quale l'utente anonimo avrà la possibilità di inserire nome utente e password al fine di eseguire una login nel sistema</i>
Aiuto	<i>Pagina di informazioni generali sul sistema</i>
Ricerca e visualizzazione pubblicazioni	<i>Comprende la pagina nella quale sarà visualizzato il catalogo completo e un'altra pagina nella quale sarà possibile eseguire una ricerca delle pubblicazioni filtrata per ID, ISBN, Titolo, Autore e parole chiave</i>
Visualizzazione delle classifiche	<i>Pagina nella quale verranno visualizzati gli utenti più collaborativi e le ultime pubblicazioni inserite e modificate</i>
Gestione profilo personale	<i>Sezione che si occupa della modifica degli attributi descrittivi del singolo utente del sistema</i>
Inserimento like e recensioni	<i>Funzionalità di inserimento di un like relativo ad una pubblicazione e inserimento di una recensione che sarà soggetta ad approvazione da parte di un utente moderatore</i>
Inserimento e modifica pubblicazioni	<i>Modulo del sistema (riservato agli utenti di grado moderatore o superiore) che si occupa di gestire le pubblicazioni del catalogo, quindi del loro inserimento e la loro modifica</i>
Approvazione recensioni	<i>Pagina (riservata agli utenti di grado moderatore o superiore) che lista le recensioni in attesa di approvazione e ha la funzionalità di far approvare o eliminare una recensione</i>
Eliminazione delle pubblicazioni	<i>Funzionalità (riservata agli amministratori) che permette di eliminare una pubblicazione dal catalogo</i>



Gestione dei moderatori	<i>(Riservato agli amministratori) Permette di promuovere gli utenti base a moderatori e viceversa</i>
Gestione degli amministratori	<i>(Riservato al super amministratore) Permette di promuovere gli utenti moderatori a amministratori e viceversa</i>
Estrazione dei log di sistema	<i>(Riservato al super amministratore) Pagina che illustra tutti i log delle modifiche effettuate alle pubblicazioni</i>

SYSTEM DESIGN

ARCHITETTURA DEL SISTEMA



L'architettura del sistema è costruita sul pattern architettonale **MVC** (Model-View-Controller) che ha permesso una distinzione tra le logiche di business e presentazione dei dati.

Infatti troviamo i package model e view che contengono rispettivamente le classi del modello (che descrive le entità delle quali il sistema ha bisogno per operare) e le viste dell'applicazione. Per quanto riguarda il data retrieval è stato utilizzato il pattern **DAO** (Data Access Object), che fornisce un layer indipendente di comunicazione con il database.

Distinzione tra Controller Back-End e Front-End

La distinzione sopra citata deriva dal fatto che utilizzando la libreria JavaFX per la realizzazione dell'interfaccia grafica, abbiamo bisogno dei controller per gestire le viste. Questo perché la libreria JavaFX è di per sé MVC. Così facendo dovevamo unire i controller della nostra applicazione con i controller di JavaFX, che per il principio della “*separation of concerns*” non è adeguato. Quindi la soluzione scelta è stata quella di dividere in due package i controller, inserendo in quelli di backend i controller che si occupano dello scambio di dati con il DAO e di gestione del model, e in quelli di frontend i controller che gestiscono le viste.

Sono state inoltre definite due classi astratte a questo proposito, *ControllerFrontend.java* e *ControllerBackend.java* che sono servite nel caso in cui i due tipi avessero avuto bisogno di implementare in ogni classe dei metodi. La scelta si è rivelata utile per il metodo *load()* all'interno di *ControllerFrontend.java* che viene invocato ogni volta che una schermata viene attivata, e quindi da ogni controller di frontend.

I controller-backend si occupano dell'interazione con il DAO per ricavare i dati che servono e gestirli negli oggetti del dominio (model) e di gestire le problematiche di sessione e autorizzazione dell'utente.

DAO - Architettura

Il Data Access Object è stato progettato utilizzando il design pattern **Factory**. Esso in questo caso ci permette di astrarre le funzionalità del DAO da come devono essere implementate. Infatti è stata definita la classe astratta *DAOFactory.java* che al suo interno contiene tutti i metodi astratti che vanno implementati nella Factory concreta legata a uno specifico DBMS (es. *MySQL_DAOFactory.java*). Inoltre la classe astratta precedente contiene anche un metodo statico (*getDAOFactory(DBMS)*), che restituisce un'implementazione specifica del DAO in base al parametro che noi passiamo. Questo metodo sarà utilizzato ogni volta che vogliamo creare un oggetto del DAO.

Per permettere al sistema di avere questa flessibilità, è necessario che tutte le classi del DAO siano opportunamente definito tramite interfacce che specificano quali metodi bisogna implementare e la loro *signature*. Esse si trovano nel package *app.DAO.interfaces*.

L'altro package che troviamo all'interno del DAO è MySQL (*app.DAO.MySQL*) che contiene tutte le implementazioni delle astrazioni definite precedentemente su DBMS MySQL, ovvero implementazione della classe astratta DAOFactory e di tutte le interfacce presenti in *DAO.interfaces*.

Controller Di Back-End

Tutti i controller di Back End sono stati implementati utilizzando il design pattern **Singleton** che ci permette di modellare la situazione in cui una classe viene istanziata una sola volta. Nel nostro caso è proprio così, infatti non c'è nessuna necessità di avere più istanze di un controller, anzi questo potrebbe causare conflitti (es. se avessimo più istanze del controller schermo potremmo rischiare di avere una schermata caricata nel primo e una diversa nel secondo, che non ha senso nella nostra applicazione).

Oltre ai controller che gestiscono il Model, il sistema è dotato di 3 controller aggiuntivi:

- **ControllerAuth:** controller che gestisce l'autorizzazione degli utenti.

Metodi specifici:

- **boolean abilita(ManageCapability capability):** al metodo viene passata la macroarea alla quale accedere enumerata con la java enum ManageCapability. Esso restituirà true o false in base alla sessione dell'utente corrente e al suo ruolo.

- **ControllerSchermo:** controller che gestisce i cambi di layout dell'applicazione.

Metodi specifici:

- **void caricaLayouts():** metodo invocato durante il processo di starting dell'applicazione che permette all'FXML loader di ottenere già tutte le risorse di tipo view che il sistema richiede. Il metodo inserisce inoltre in un hashmap i nomi delle view caricate con la rispettiva risorsa.
- **void activateOnCenter(String name):** metodo invocato ogni qualvolta si ha la necessità di cambiare schermata. Inserisce nello spazio centrale del BoderPane la schermata desiderata.
- **void openPaneInWindow(String name):** metodo invocato quando si desidera aprire una schermata in una nuova finestra.

- **ControllerSessione:** controller che gestisce le sessioni degli utenti.

Metodi specifici:

- **Utente *getUtenteLoggato()*:** restituisce l'oggetto Utente che ha effettuato l'accesso, se nessuno ha effettuato l'accesso restituisce null.
- **void *activateOnCenter(String name)*:** setta il parametro Utente loggato. Il metodo viene invocato quando un utente esegue l'accesso, e quando il logout.
- **void *logout()*:** metodo invocato quando si desidera effettuare il logout.
- **Pubblicazione *getPubblicazioneSelezionata()*:** restituisce la pubblicazione presa in esame dall'utente, null se non esiste
- **void *setPubblicazioneSelezionata(Pubblicazione p)*:** setta la pubblicazione presa in esame dall'utente
- **String *getPaginaPrecedente()*:** restituisce la pagina visitata precedentemente a quella attuale.
- **void *setPaginaPrecedente(String name)*:** setta la pagina da assegnare come target al button “indietro”
- **void *check()*:** setta il master layout (navbar, footer, ecc...) in base all'utente loggato.

Descrizione delle entità

Pubblicazione

Una pubblicazione è una scheda bibliografica di un libro, opera letteraria, e tutto quello che può essere identificato mediante codice ISBN. Le pubblicazioni vengono inserite nel catalogo dagli utenti (di grado moderatore o superiore).

Utente

Con il termine utente indichiamo qualsiasi tipo di utilizzatore del sistema. Gli utenti base possono consultare il catalogo e inserire recensioni e like. I gestori possono in aggiunta inserire o modificare le pubblicazioni. Gli utenti base si suddividono in attivi e passivi in base alle azioni che effettuano nel corso del tempo (e non hanno privilegi differenti, la distinzione è a scopo statistico). Invece gli utenti gestori, che ovviamente sono considerati attivi dal sistema, sono di tipo moderatore, amministratore e super amministratore. Ogni ruolo ha i propri privilegi, in aggiunta a quelli delle categorie sottostanti.

Recensione

Una recensione è un breve testo di carattere qualitativo che un utente associa ad una pubblicazione nel catalogo. Le recensioni dovranno essere approvate da parte di un utente moderatore prima di essere pubblicamente visibili. Ogni recensione è associata alla data e ora di inserimento.

Like

I like vengono assegnati dagli utenti alle pubblicazioni nel catalogo. Per cui ogni like è rappresentato come una relazione tra utente e pubblicazione. Ogni pubblicazione può ricevere un solo like per ogni utente.

Sorgente

Le sorgenti legate ad una pubblicazione rappresentano qualsiasi tipo di risorsa digitale (immagini, link per download, ecc...) e sono associate al proprio URI.

Metadati

I metadati di una pubblicazione rappresentano l'insieme dei dati aggiuntivi associati ad una pubblicazione (ISBN, numero di pagine, lingua, data di pubblicazione, lista delle ristampe, parole chiave identificative).

Storia

Tiene traccia di tutte le modifiche fatte dagli utenti al catalogo bibliografico. Ogni entry della storia è associata ad una breve descrizione della modifica e la data e l'ora di quando essa è stata effettuata (scritta sul database).

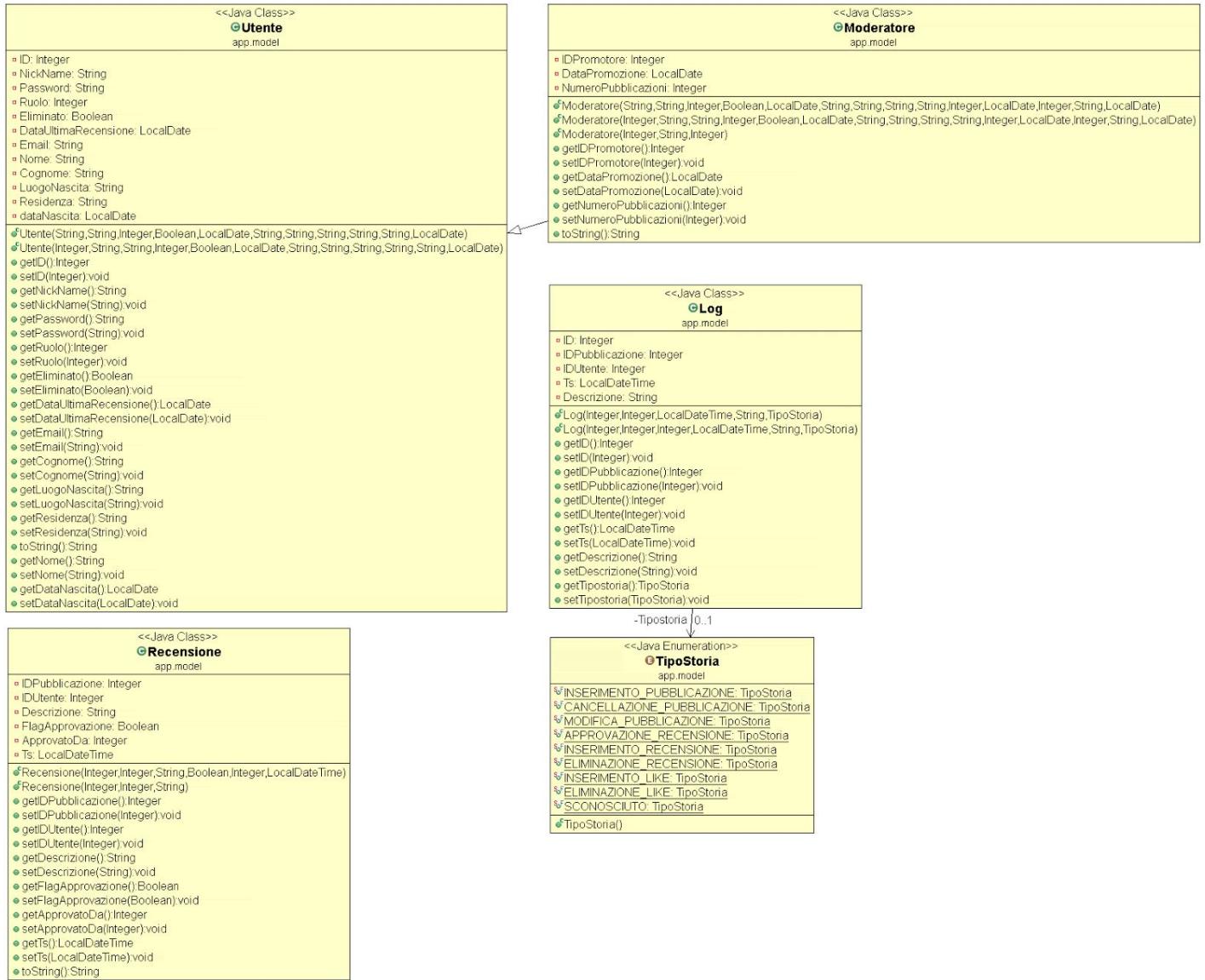
Ruolo

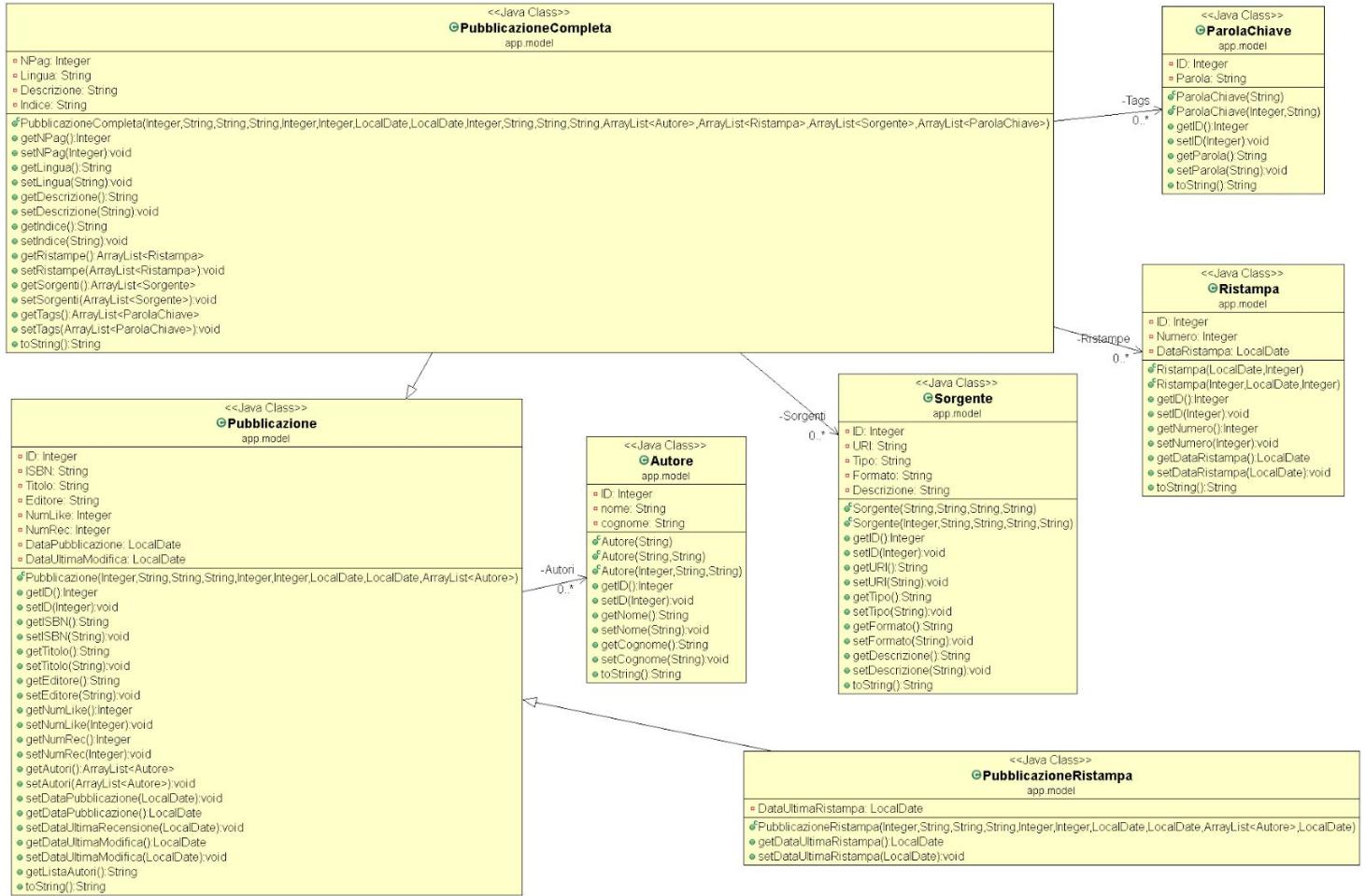
Il ruolo indica la tipologia di utenza riferita ad un particolare utente con i vari privilegi permessi. I ruoli da noi proposti sono : attivo, passivo, moderatore, amministratore, superamministratore.



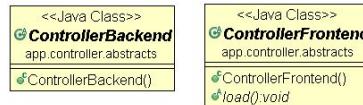
Class Diagrams

Model Class Diagram

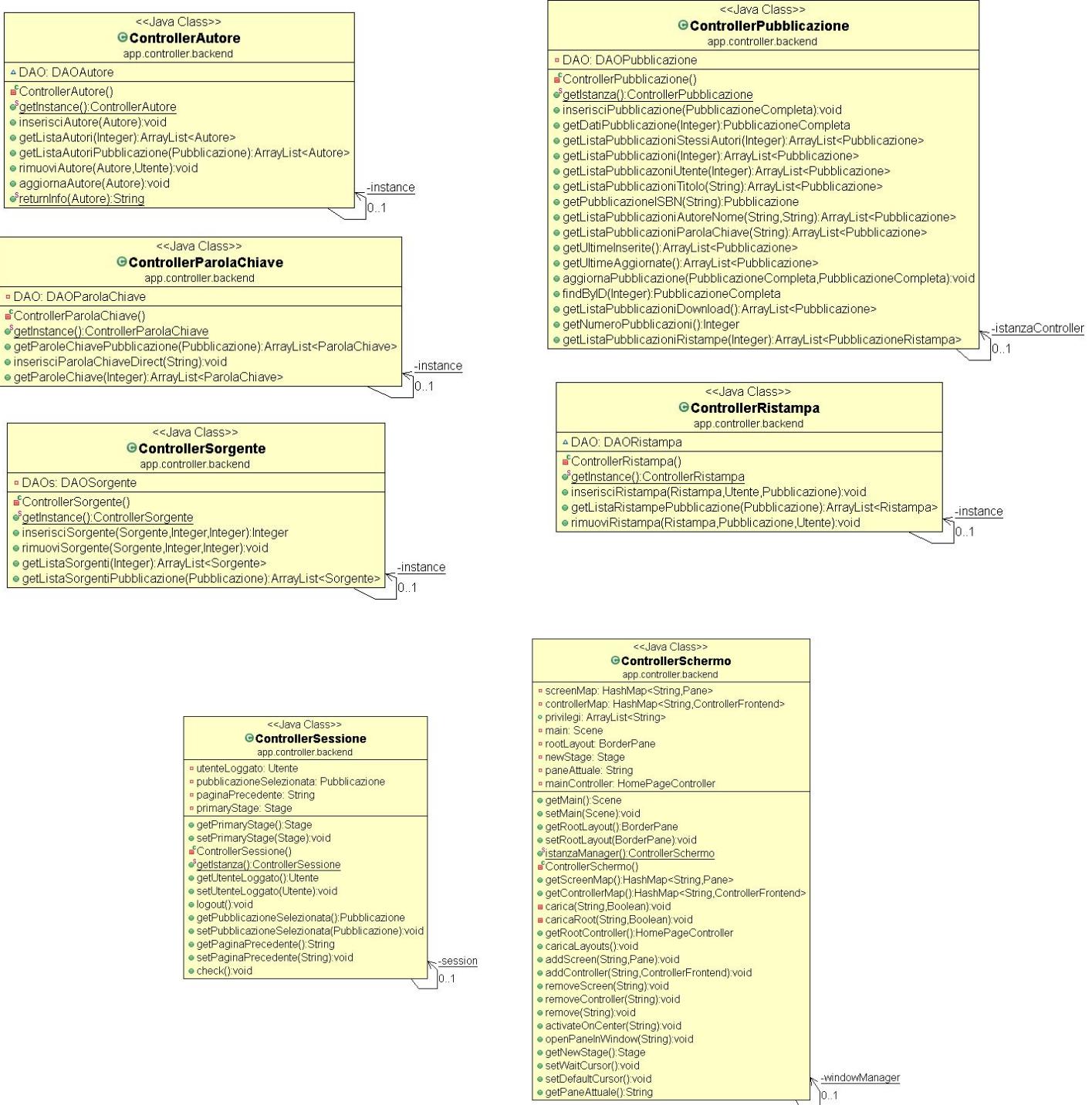


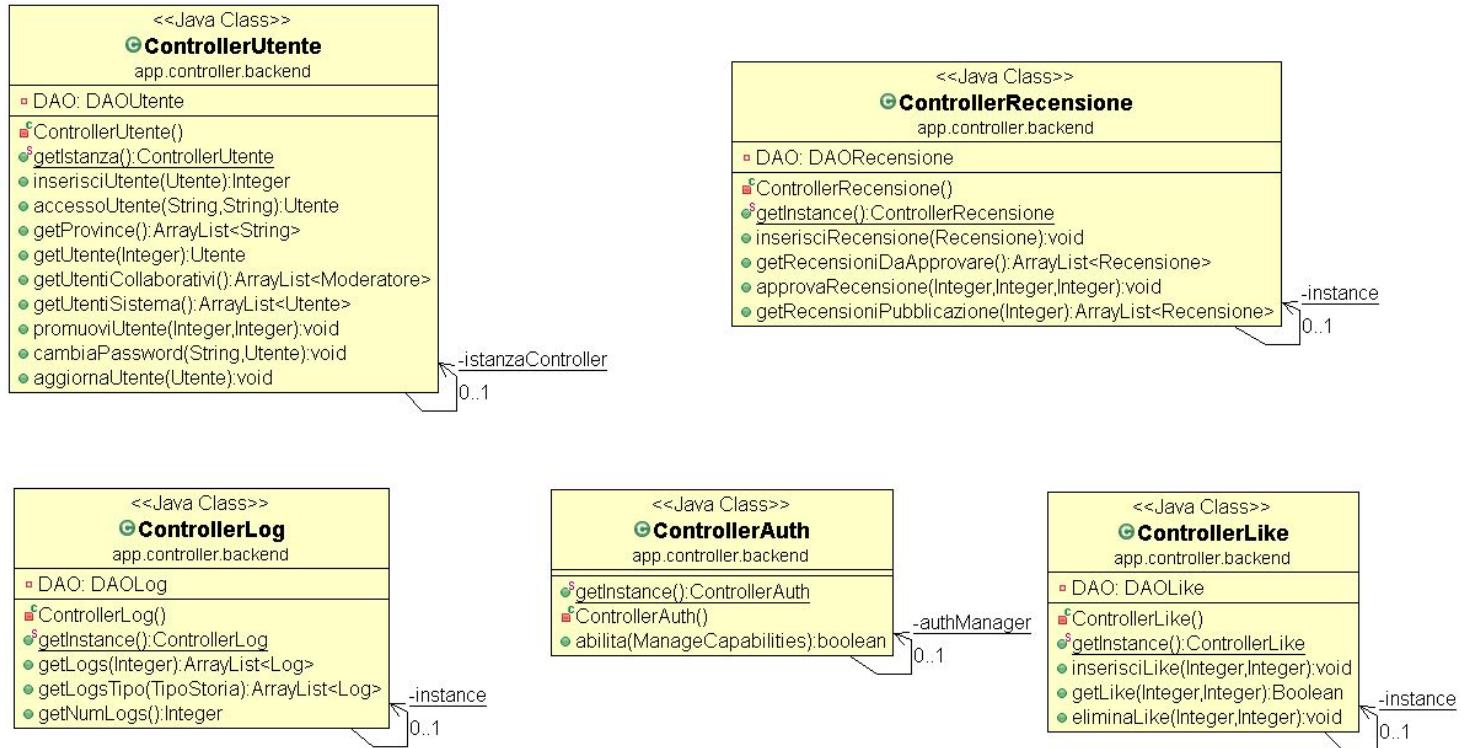


Controller Class Diagram



Back End Controller Class Diagram





Front End Controller Class Diagram



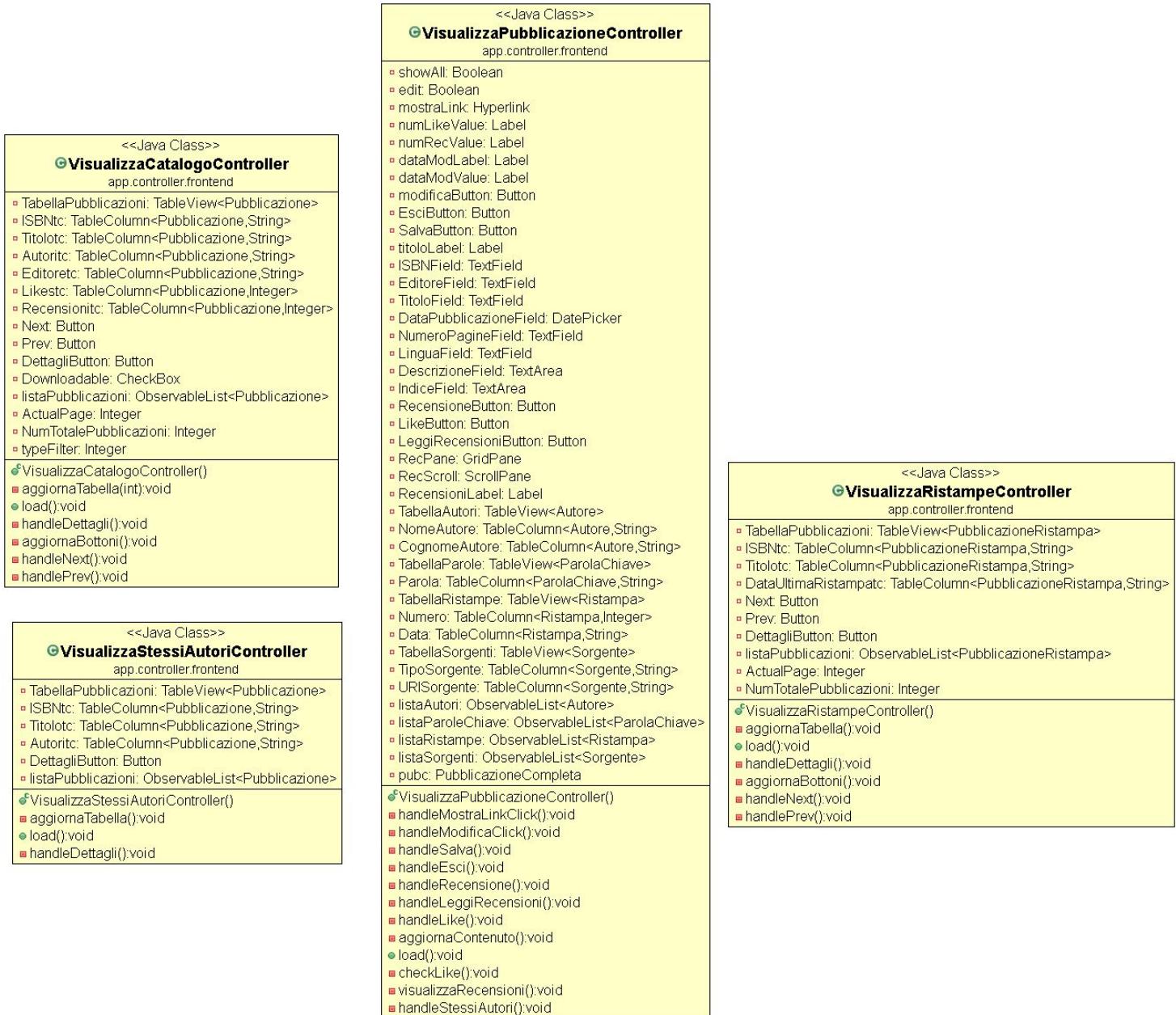
AggiungiAutoreController app.controller.frontend	AggiungiParolaChiaveController app.controller.frontend	AggiungiRistampaController app.controller.frontend	AggiungiSorgenteController app.controller.frontend
<ul style="list-style-type: none"> ▫ NomeField: TextField ▫ CognomeField: TextField ▫ AggiungiNuovoButton: Button ▫ AggiungiEsistenteButton: Button ▫ CancelButton: Button ▫ TabellaAutori: TableView<Autore> ▫ Nometc: TableColumn<Autore, String> ▫ Cognometc: TableColumn<Autore, String> ▫ listaAutori: ObservableList<Autore> ▫ pubblicazione: PubblicazioneCompleta <p>AggiungiAutoreController()</p> <ul style="list-style-type: none"> ■ handleAggiungiNuovo(): void ■ handleAggiungiEsistente(): void ■ handleCancel(): void ■ isInputValid(): boolean ■ refreshTableAndFields(): void ■ load(): void ■ getPubblicazione() PubblicazioneCompleta ■ setPubblicazione(PubblicazioneCompleta): void 	<ul style="list-style-type: none"> ▫ ParolaField: TextField ▫ AggiungiNuovoButton: Button ▫ AggiungiEsistenteButton: Button ▫ CancelButton: Button ▫ TabellaParole: TableView<ParolaChiave> ▫ Parolatc: TableColumn<ParolaChiave, String> ▫ listaParole: ObservableList<ParolaChiave> ▫ pubblicazione: PubblicazioneCompleta <p>AggiungiParolaChiaveController()</p> <ul style="list-style-type: none"> ■ initialize(): void ■ handleAggiungiNuovo(): void ■ handleAggiungiEsistente(): void ■ handleCancel(): void ■ isInputValid(): boolean ■ load(): void ■ getPubblicazione() PubblicazioneCompleta ■ setPubblicazione(PubblicazioneCompleta): void 	<ul style="list-style-type: none"> ▫ NumeroField: TextField ▫ DataField: DatePicker ▫ pubblicazione: PubblicazioneCompleta <p>AggiungiRistampaController()</p> <ul style="list-style-type: none"> ■ initialize(): void ■ handleSalva(): void ■ handleCancel(): void ■ isInputValid(): boolean ■ load(): void ■ getPubblicazione() PubblicazioneCompleta ■ setPubblicazione(PubblicazioneCompleta): void 	<ul style="list-style-type: none"> ▫ URIField: TextField ▫ TipoField: TextField ▫ FormatoField: TextField ▫ DescrizioneField: TextArea ▫ pubblicazione: PubblicazioneCompleta <p>AggiungiSorgenteController()</p> <ul style="list-style-type: none"> ■ initialize(): void ■ handleSalva(): void ■ handleCancel(): void ■ isInputValid(): boolean ■ load(): void ■ getPubblicazione() PubblicazioneCompleta ■ setPubblicazione(PubblicazioneCompleta): void
BarraLateraleController app.controller.frontend	GestioneCatalogoController app.controller.frontend	ApprovaRecensioniController app.controller.frontend	ClassificheController app.controller.frontend
<ul style="list-style-type: none"> ▫ GlobalPane: AnchorPane ▫ Pannello: Button ▫ Logout: Button <p>BarraLateraleController()</p> <ul style="list-style-type: none"> ■ load(): void ■ handlePannello(): void ■ handleLogout(): void 	<ul style="list-style-type: none"> ▫ TabellaPubblicazioni: TableView<Pubblicazione> ▫ ISBNtc: TableColumn<Pubblicazione, String> ▫ Titolotc: TableColumn<Pubblicazione, String> ▫ Autoritc: TableColumn<Pubblicazione, String> ▫ Editoretc: TableColumn<Pubblicazione, String> ▫ Likestc: TableColumn<Pubblicazione, Integer> ▫ Recensionitc: TableColumn<Pubblicazione, Integer> ▫ Avanti: Button ▫ Indietro: Button ▫ InserisciButton: Button ▫ listaPubblicazioni: ObservableList<Pubblicazione> <p>GestioneCatalogoController()</p> <ul style="list-style-type: none"> ■ aggiornaLista(int): void ■ handleInserisci(): void ■ handleDettagli(): void ■ load(): void 	<ul style="list-style-type: none"> ▫ TabellaRecensioni: TableView<Recensione> ▫ Utentetc: TableColumn<Recensione, Integer> ▫ Pubbttc: TableColumn<Recensione, Integer> ▫ Testotc: TableColumn<Recensione, String> ▫ listaRecensioni: ObservableList<Recensione> ▫ TestoField: TextArea ▫ ApprovaButton: Button ▫ NickField: TextField ▫ TitoloField: TextField ▫ ISBNField: TextField <p>ApprovaRecensioniController()</p> <ul style="list-style-type: none"> ■ handleApprova(): void ■ aggiorna(): void ■ mostraRecensione(Recensione): void ■ load(): void 	<ul style="list-style-type: none"> ▫ TabellaUtenti: TableView<Moderatore> ▫ Nicknametc: TableColumn<Moderatore, String> ▫ NumPubbttc: TableColumn<Moderatore, Integer> ▫ TabellaPubblicazioni: TableView<Pubblicazione> ▫ ISBNtc: TableColumn<Pubblicazione, String> ▫ Titolotc: TableColumn<Pubblicazione, String> ▫ TabellaPAGgiornate: TableView<Pubblicazione> ▫ ISBNAtc: TableColumn<Pubblicazione, String> ▫ TitoloAtc: TableColumn<Pubblicazione, String> ▫ Avanti: Button ▫ Indietro: Button ▫ listaUtenti: ObservableList<Moderatore> ▫ listaUltimeInserite: ObservableList<Pubblicazione> ▫ listaUltimeAggiornate: ObservableList<Pubblicazione> <p>ClassificheController()</p> <ul style="list-style-type: none"> ■ load(): void



<<Java Class>>	GestioneUtentiController	app.controller.frontend
▪ TabellaUtenti: TableView<Utente>		
▪ IDUtentetc: TableColumn<Utente, Integer>		
▪ Nicknametc: TableColumn<Utente, String>		
▪ Ruolotc: TableColumn<Utente, String>		
▪ PromuoviButton: Button		
▪ RuoloField: ComboBox<String>		
▪ mappaRuoli: HashMap<Integer, String>		
▪ mappaNomi: HashMap<String, Integer>		
▪ listaUtenti: ObservableList<Utente>		
❖ GestioneUtentiController()		
▪ initialize(): void		
▪ handleRuoli(): void		
▪ refreshTable(): void		
● load(): void		
● mostraUtente(Utente): void		
<<Java Class>>	InformazioniController	app.controller.frontend
❖ InformazioniController()		
● load(): void		
<<Java Class>>	LoginController	app.controller.frontend
▪ Accedi: Button		
▪ Username: TextField		
▪ Password: TextField		
▪ NonValido: Label		
▪ linkRegistrazione: Hyperlink		
❖ LoginController()		
● handleAccedi(): void		
● handleRegistrazione(): void		
● load(): void		
<<Java Class>>	HomePageController	app.controller.frontend
▪ Entra: Menu		
▪ Accedi: MenuItem		
▪ Registrati: MenuItem		
▪ Catalogo: MenuItem		
▪ Cerca: MenuItem		
▪ ruololmg: ImageView		
▪ userlmg: ImageView		
▪ GestioneButton: Button		
▪ EsciButton: Button		
▪ IndietroButton: Button		
▪ RuoloLabel: Label		
▪ UserLabel: Label		
❖ HomePageController()		
▪ handleCatalogo(): void		
▪ handleAccedi(): void		
▪ handleRegistrati(): void		
▪ handleCerca(): void		
▪ handleIndietro(): void		
▪ handleGestione(): void		
▪ handleEsci(): void		
▪ handleRistampe(): void		
▪ handleChiudi(): void		
● load(): void		
● setRuololmgVix(Boolean): void		
● setUserlmgVix(Boolean): void		
● setUserLabelText(String): void		
● setRuoloLabelText(String): void		
● refreshAfterLogout(): void		
● disabilitalndietro(): void		
● abilitalndietro(): void		
● setRootLayout(Boolean, Boolean): void		
<<Java Class>>	InserisciPubblicazioneController	app.controller.frontend
▪ numLikeLabel: Label		
▪ numLikeValue: Label		
▪ numRecLabel: Label		
▪ numRecValue: Label		
▪ dataModLabel: Label		
▪ dataModValue: Label		
▪ AnnullaButton: Button		
▪ InserisciButton: Button		
▪ ISBNField: TextField		
▪ EditoreField: TextField		
▪ DataPubblicazioneField: DatePicker		
▪ ErrorLabel: Label		
▪ TitoloField: TextField		
▪ NumeroPagineField: TextField		
▪ LinguaField: TextField		
▪ DescrizioneField: TextField		
▪ IndiceField: TextField		
▪ EditAutori: Button		
▪ EditSorgenti: Button		
▪ EditRistampe: Button		
▪ EditParole: Button		
▪ TabellaAutori: TableView<Autore>		
▪ NomeAutore: TableColumn<Autore, String>		
▪ CognomeAutore: TableColumn<Autore, String>		
▪ TabellaParole: TableView<ParolaChiave>		
▪ Parola: TableColumn<ParolaChiave, String>		
▪ TabellaRistampe: TableView<Ristampa>		
▪ Numero: TableColumn<Ristampa, Integer>		
▪ Data: TableColumn<Ristampa, String>		
▪ TabellaSorgenti: TableView<Sorgente>		
▪ TipoSorgente: TableColumn<Sorgente, String>		
▪ URISorgente: TableColumn<Sorgente, String>		
▪ listaAutori: ObservableList<Autore>		
▪ listaParoleChiave: ObservableList<ParolaChiave>		
▪ listaRistampe: ObservableList<Ristampa>		
▪ listaSorgenti: ObservableList<Sorgente>		
▪ dalInserire: PubblicazioneCompleta		
❖ InserisciPubblicazioneController()		
▪ handleInserisci(): void		
▪ handleAnnulla(): void		
▪ handleAutori(): void		
▪ handleParole(): void		
▪ handleRistampe(): void		
▪ handleSorgenti(): void		
● load(): void		
● update(): void		
<<Java Class>>	LogsController	app.controller.frontend
▪ TabellaStoria: TableView<Log>		
▪ ISBNNtc: TableColumn<Log, Integer>		
▪ Nicknametc: TableColumn<Log, Integer>		
▪ Timestamptc: TableColumn<Log, String>		
▪ Operazionetc: TableColumn<Log, String>		
▪ IDtc: TableColumn<Log, Integer>		
▪ TipoField: ComboBox<String>		
▪ DescrizioneText: Text		
▪ Next: Button		
▪ Prev: Button		
▪ listaLog: ObservableList<Log>		
▪ filteredLogs: FilteredList<Log>		
▪ NumTotaleLogs: Integer		
▪ ActualPage: Integer		
❖ LogsController()		
● load(): void		
▪ aggiornaTabella(): void		
▪ aggiornaBottoni(): void		
▪ handleNext(): void		
▪ handlePrev(): void		
▪ handleTipo(ActionEvent): void		



<p><<Java Class>></p> <p>PannelloGestioneController app.controller.frontend</p> <ul style="list-style-type: none">▫ Ricerca: Label▫ Classifiche: Label▫ Impostazioni: Label▫ Profilo: Label▫ Recensioni: Label▫ Logs: Label▫ GestioneCatalogo: Label▫ GestioneUtenti: Label▫ Box00: HBox▫ Box01: HBox▫ Box02: HBox▫ Box03: HBox▫ Box10: HBox▫ Box11: HBox▫ Box12: HBox▫ Box13: HBox <p>PannelloGestioneController()</p> <ul style="list-style-type: none">● caricaPannello():void● handleRicerca():void● handleClassifiche():void● handleImpostazioni():void● handleProfilo():void● handleRecensioni():void● handleLogs():void● handleGestioneCatalogo():void● handleGestioneUtenti():void● load():void	<p><<Java Class>></p> <p>ProfiloUtenteController app.controller.frontend</p> <ul style="list-style-type: none">▫ NomeField: TextField▫ CognomeField: TextField▫ EmailField: TextField▫ NicknameField: TextField▫ newPass: PasswordField▫ ResidenzaField: ComboBox<String>▫ LuogoNascitaField: ComboBox<String>▫ DataNascitaField: DatePickerController▫ SalvaButton: Button▫ ModificaButton: Button▫ PasswordButton: Button▫ TabellaPubblicazioni: TableView<Pubblicazione>▫ Titolotc: TableColumn<Pubblicazione, String>▫ listaPubblicazioni: ObservableList<Pubblicazione>▫ activeChangePassword: Boolean <p>ProfiloUtenteController()</p> <ul style="list-style-type: none">● load():void■ handlePassword():void■ handleModifica():void■ handleSalva():void	<p><<Java Class>></p> <p>RegistrazioneController app.controller.frontend</p> <ul style="list-style-type: none">▫ Residenza: ComboBox<String>▫ ProvNascita: ComboBox<String>▫ NickName: Label▫ NickField: TextField▫ Email: Label▫ EmailField: TextField▫ Password: Label▫ PassField: PasswordField▫ Nome: Label▫ Error: Label▫ NomeField: TextField▫ Cognome: Label▫ CognomeField: TextField▫ DataNascita: Label▫ DataNascitaField: DatePickerController▫ PassField2: PasswordField▫ Registrati: Button <p>RegistrazioneController()</p> <ul style="list-style-type: none">● initialize():void● handleRegistrati():void● isValidEmail(String):boolean● isValidNickname(String):boolean● isValidPassword(String, String):boolean● criptaPassword(String):String● load():void	<p><<Java Class>></p> <p>RicercaController app.controller.frontend</p> <ul style="list-style-type: none">▫ TabellaPubblicazioni: TableView<Pubblicazione>▫ ISBNtc: TableColumn<Pubblicazione, String>▫ Titolotc: TableColumn<Pubblicazione, String>▫ Autoretc: TableColumn<Pubblicazione, String>▫ Editoretc: TableColumn<Pubblicazione, String>▫ Liketc: TableColumn<Pubblicazione, Integer>▫ Recensionitc: TableColumn<Pubblicazione, Integer>▫ Dattatc: TableColumn<Pubblicazione, String>▫ KeyField: TextField▫ FilterField: ChoiceBox<String>▫ CercaButton: Button▫ NomeAutoreField: TextField▫ CognomeAutoreField: TextField▫ ErrorLabel: Label▫ listaPubblicazioni: ObservableList<Pubblicazione> <p>RicercaController()</p> <ul style="list-style-type: none">● RicercaController()■ handleCerca():void■ visualizzaPubblicazione(Pubblicazione):void■ checkValid():boolean■ handleBox():void● initialize():void● load():void
	<p><<Java Class>></p> <p>ScriviRecensioneController app.controller.frontend</p> <ul style="list-style-type: none">▫ TitoloLabel: Label▫ AutoreLabel: Label▫ TestoRecensione: TextArea▫ PubblicaButton: Button <p>ScriviRecensioneController()</p> <ul style="list-style-type: none">● load():void■ handlePubblica():void		



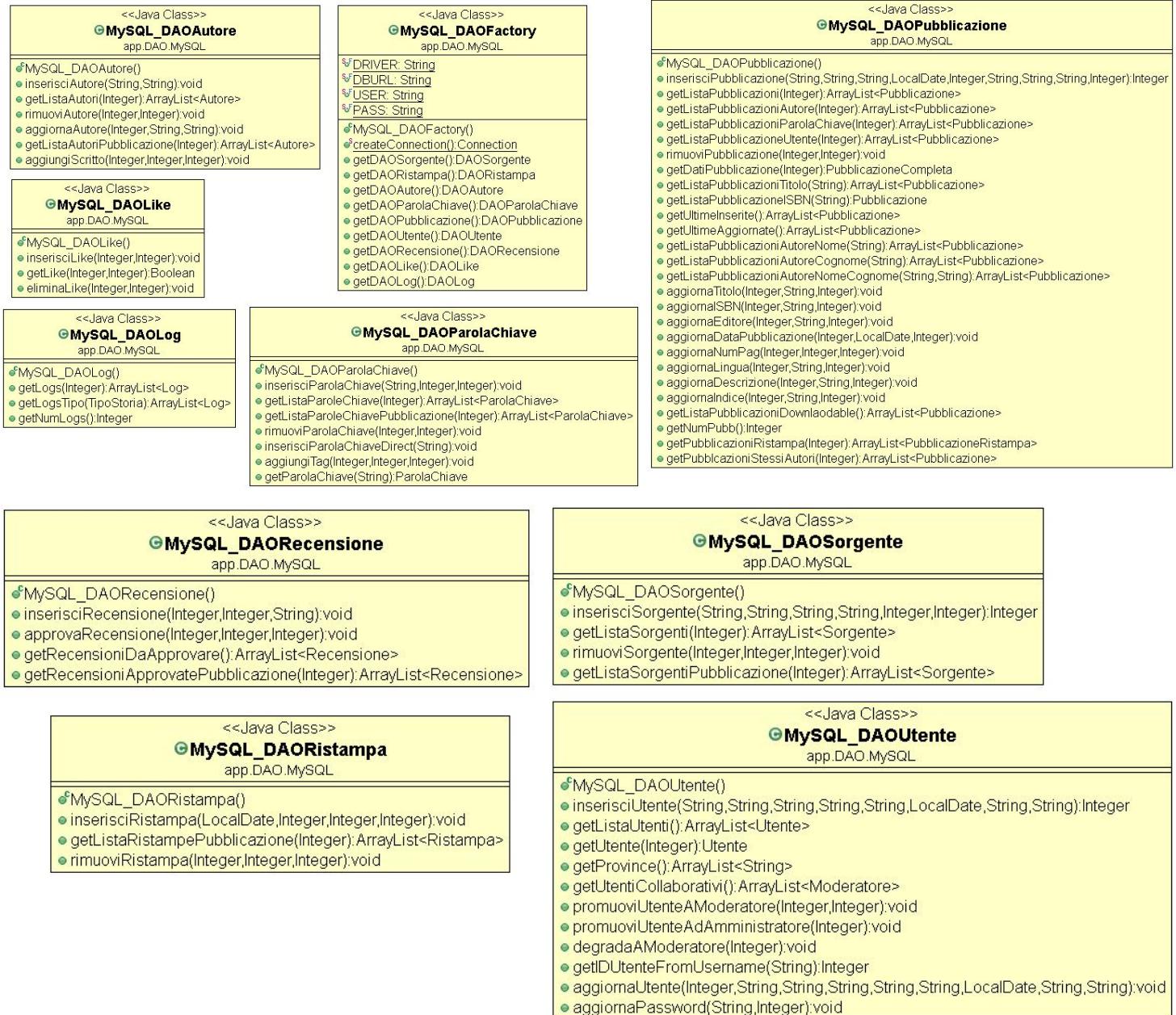
DAO Class Diagram



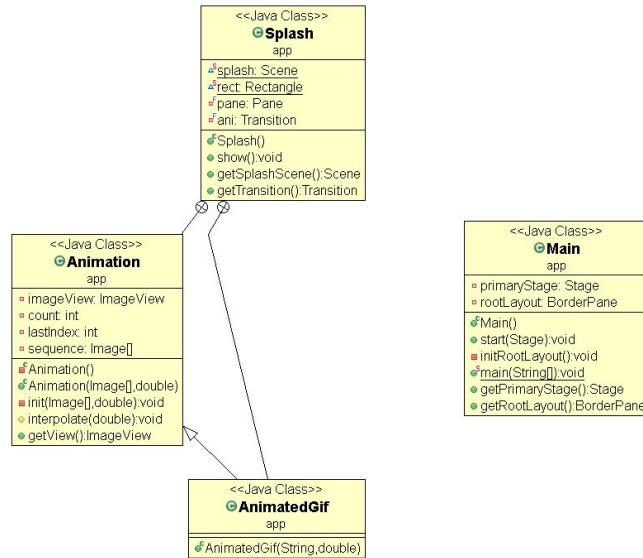
Interfaces



MySQL Implementation Class Diagram



Main Class Diagram

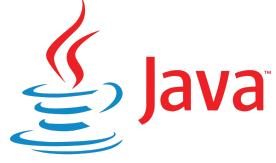


Dettagli Implementativi

Per quanto riguarda l'implementazione del controller schermo sono state utilizzate delle strutture `HashMap` per memorizzare il pannello e il controller con il relativo nome per permettere un cambio di schermate più facilmente gestibile. Sono stati definiti anche dei tipi **Enum** per definire le tipologie della Storia (ovvero del log) e per le capabilities dell'utente. Per la gestione dei **ruoli** (Amministratore, Moderatore ecc...) è stata definita una mappa Hash dentro la classe del model “Utente” e due metodi che permettono di passare dal valore numerico a quello stringa per avere maggiore trasparenza su quello che succede nell'applicazione.

Per ottimizzare il flusso dell'applicativo si è deciso di effettuare il caricamento delle schermate con `FXMLLoader` all'inizio tramite il metodo `caricaLayouts()`. Per cifrare la password è stata utilizzata la funzione di hashing robusta **BCrypt**. I controlli di validità sui campi sono stati fatti manualmente oppure tramite le **regex** (es. email valida).

TECNOLOGIE UTILIZZATE



Il progetto è stato sviluppato su **Eclipse**, con l'aiuto della libreria **JavaFX**. La connessione al database è stata implementata utilizzando il driver **JDBC**. Il DBMS utilizzato è **MySQL**, per tutti i dettagli progettuali e implementativi del database consultare la documentazione del progetto database fornita nella stessa cartella.

Il layout del sistema è stato progettato sullo **Scene Builder** di JavaFX, utilizzando il linguaggio **CSS** per lo stile. I diagrammi sono stati realizzati con il plugin di Eclipse **ObjectAid** e **draw.io**, mentre la documentazione è stata scritta con l'aiuto della suite Google, in particolare **Google Docs**.

Per lo sharing del progetto si è utilizzata una repository su **GitHub**.

Link alla repository: <https://github.com/federix98/UnivAQLibrary>

INTERFACCIA GRAFICA

Root Layout

Il root layout contiene le schermate descritte successivamente. Esso è stato realizzato con un border pane che ha la seguente struttura:



UNIVERSITÀ
DEGLI STUDI
DE L'AQUILA



Al top troviamo un Menu bar che se non siamo loggati contiene shortcut all'accesso e alla registrazione e contiene sempre le shortcut per la ricerca e il catalogo. Inoltre troviamo in ogni caso la barra di navigazione che contiene un pulsante per andare indietro e uno per uscire dall'applicazione.

Esempio top senza login:



Esempio top con login:



Come possiamo notare in caso di login effettuato abbiamo collegamenti a Logout e delle label che indicano nome dell'utente e ruolo.

In basso invece abbiamo un footer contenente delle informazioni generali sul progetto:



Le schermate di funzionamento dell'applicazione verranno caricate al centro.

Splash Screen

Schermata di splash che viene mostrata durante il caricamento dell'applicazione.



UNIVERSITÀ
DEGLI STUDI
DE L'AQUILA



Registrazione

Maschera con un form che permette di effettuare la registrazione al sistema tramite data-entry.

Il layout è il seguente:

Benvenuto in 
Compila il seguente modulo di registrazione

NickName	<input type="text"/>	Nome	<input type="text"/>
Email	<input type="text"/>	Cognome	<input type="text"/>
Password	<input type="text"/>	Residenza (PR)	<input type="text"/>
Conferma Pass	<input type="text"/>	Luogo nascita (PR)	<input type="text"/>
Data di nascita	<input type="text"/> 	<input type="button" value="Registrati"/>	

Login


ACCEDI

Username
Password

Non hai ancora un account? [Registrati subito!](#)

Permette di effettuare l'accesso al sistema una volta che si è registrati al suo interno.

Contiene anche un link che punta alla pagina di registrazione.

Pannello Gestione

Pannello che permette all'utente di accedere alle varie aree del sistema in base al suo ruolo.

Esempio amministratore e moderatore:



Profilo Utente

Mostra tutte le informazioni o metadati dell'utente e permette di aggiornare i valori modificabili. Sulla destra troviamo anche l'elenco delle pubblicazioni che abbiamo inserito con un bottone per modificarle.

Il layout è il seguente:

Classifiche

Come richiesto esplicitamente dalla specifica, mostra le ultime 10 pubblicazioni inserite, quelle che negli ultimi 30 giorni hanno subito un aggiornamento e gli utenti più collaborativi ovvero che hanno inserito più



pubblicazioni.

Gestione Utenti

IDUtente	Nickname	Ruolo
10	ciccio	Amministratore
11	franz94	Moderatore
12	ricky66	Amministratore
15	sandro23	Moderatore
16	doge99	Moderatore
...

Ruolo

Pannello riservato agli utenti gestori del sistema, permette di promuovere o degradare gli altri utenti di grado inferiore.

Logs

ID Pubblicazione	ID	Tipo	Timestamp	Operazione
206	5	29	2019-11-29T13:48:37	INSERIMENTO_LIKE
205	7	1	2019-11-26T22:34:05	ELIMINAZIONE_LIKE
204	7	1	2019-11-26T22:34:02	INSERIMENTO_LIKE
203	7	28	2019-10-25T20:35:43	ELIMINAZIONE_LIKE
202	7	28	2019-10-25T20:35:38	INSERIMENTO_LIKE
201	7	28	2019-10-25T20:35:29	ELIMINAZIONE_LIKE
200	7	28	2019-10-25T20:35:24	INSERIMENTO_LIKE
...

< >

Mostra l'elenco dei log del sistema in formato paginato. Contiene dei pulsanti per filtrare in base a ID della pubblicazione e Tipologia.

Ricerca



The screenshot shows a dark blue search interface. At the top, there is a search bar with the placeholder text "limbo" and a magnifying glass icon. To the right of the search bar are two buttons: "Cerca" (Search) and "per Titolo" (by Title). Below the search bar is a horizontal menu with the following labels: ISBN, Titolo, Autore, Editore, # Like, # Recensioni, and Data. Underneath this menu, a single search result is displayed in a light gray box. The result includes the ISBN "978881713...", the title "Limbo. Pen...", the author "Stendhal, ...", the publisher "Rizzoli", the number of likes "5", the number of reviews "5", and the date "2019-07-02".

Permette di effettuare tra le pubblicazioni una ricerca per ID, ISBN, Titolo, Autore e Parola Chiave. I dati delle pubblicazioni che otteniamo sono: ISBN, Titolo, Autore, Editore, Numero Like, Numero Recensioni e Data Inserimento. Cliccando su una delle pubblicazioni del risultato possiamo accedere alla pagina “Visualizza Pubblicazione” della stessa.

Inserimento Pubblicazione

Fornisce una maschera per l'inserimento di una pubblicazione.

Aggiungi Autori

Si attiva come finestra di Pop Up per inserire gli autori e collegarli alla pubblicazione che si sta inserendo.

Aggiungi Parole Chiave

Si attiva come finestra di Pop Up per inserire parole chiave e collegarle alla pubblicazione che si sta inserendo.

Aggiungi Ristampe

Si attiva come finestra di Pop Up per inserire le ristampe e collegarle alla pubblicazione che si sta inserendo.

Aggiungi Sorgenti

Si attiva come finestra di Pop Up per inserire sorgenti e collegarli alla pubblicazione che si sta inserendo.

Esempio di inserimento:



The screenshot shows a software interface for managing publications. On the left, there's a sidebar with fields for 'Titolo', 'Codice ISBN', 'Editore', 'Data pubblicazione', 'Numero pagine', 'Lingua', 'Descrizione', and 'Indice', each with an 'Inserisci' button. The main area has two tabs: 'Aggiungi Autore' (Add Author) and 'Autori Esistenti' (Existing Authors). In 'Aggiungi Autore', there are fields for 'Nome' and 'Cognome' with an 'Aggiungi' button. In 'Autori Esistenti', there's a table with columns 'Nome' and 'Cognome' showing entries like Stendhal, Zerocalcare, Italo, Stefano, and Calvino, Benni. At the bottom are 'Esci' and 'Aggiungi' buttons. To the right, there are sections for 'Parole Chiave' (Keywords) and 'Sorgenti' (Sources), both currently empty. A footer bar includes the UnivAQLibrary logo and the text 'Progetto realizzato per scopi didattici Corso: Object Oriented Software Design'.

Visualizza Catalogo

Visualizza il catalogo delle pubblicazioni nel sistema in formato paginato.

Visualizza Pubblicazione

Visualizza i dettagli di una pubblicazione e offre, sotto autenticazione, la possibilità di inserire e rimuovere likes, e rimuovere le recensioni da una pubblicazione. Inoltre è possibile accedere all'elenco delle pubblicazioni aventi gli stessi autori descritto successivamente. Permette anche di accedere alla pagina per lasciare una recensione alla pubblicazione. Cliccando sul bottone “Modifica” nella parte inferiore della pagina possiamo modificare i metadati della pubblicazione. Il bottone compare soltanto se l'utente è autorizzato alla modifica.



The screenshot shows a library application interface. On the left, a book detail page is displayed for "L'eredità di Agneta. Le signore di Lowenhof". It includes fields for ISBN 13 (9788809869097), Editore (Giunti Editore), Data pubblicazione (10/07/2019), Pagine (640), Lingua (Italiano), and a "Mostra tutto" button. Below this are sections for Indice (Indice), Descrizione (Descrizione...), Autori (Agatha Christie), and Parole Chiave (letteratura). A "Like" button is also present. On the right, a list of publications by the same author is shown, titled "Lista delle pubblicazioni con gli stessi autori". The table has columns for Titolo, Editore, Giunti Editore, Data pubblicazione, Ristampe, and Sorgenti. The "Ristampe" column shows entries like erin33, testo, darius88, testo, swq, CiaoCiao, and 23. The "Sorgenti" column shows download and https://duck... The table also includes buttons for Elimina, Modifica, and Scrivi Recensione.

Visualizza Pubblicazioni con gli stessi autori

Visualizza un elenco di pubblicazioni aventi gli stessi autori della pubblicazione dalla quale si proviene.

Scrivi Recensione

Permette di scrivere una recensione al libro selezionato che poi sarà soggetta ad approvazione.

Approva Recensioni

Permette di approvare le recensioni scritte ma non ancora approvate.

Informazioni Generali

Contiene le informazioni generali del progetto.

The screenshot shows the "Informazioni Generali" section of the application. It features the UnivaQLibrary logo at the top. Below it is a heading "Informazioni Generali" and a paragraph stating: "Progetto realizzato per il corso di Object Oriented Software Design dell'Università degli Studi dell'Aquila. (2018/2019)". A detailed description follows: "L'applicativo è un software gestionale di una biblioteca, che offre servizi di gestione delle pubblicazioni e dell'utenza. Esso è stato sviluppato secondo il pattern architettonico Model View Controller (MVC) utilizzando la piattaforma JavaFX.". At the bottom, there is a note: "Il progetto si appoggia al database realizzato per il corso Laboratorio di Basi di Dati." Logos for Eclipse and JavaFX are also present.