

|  |  |  |  |
| --- | --- | --- | --- |
| Cognome | Nome | Matricola | Indirizzo e-mail istituzionale |
| Cesarano | Agostino | N86003587 | [ag.cesarano@studenti.unina.it](mailto:ag.cesarano@studenti.unina.it) |
| Fortino | Alessandro | N86003884 | [ale.fortino@studenti.unina.it](mailto:ale.fortino@studenti.unina.it) |

1 Febbraio 2022

Sommario

[1. Analisi del problema 3](#_Toc95144732)

[1.1 WireFrame 3](#_Toc95144733)

[1.2 Tecnica di implementazione 5](#_Toc95144734)

[1.3 Individuazione degli Oggetti 6](#_Toc95144735)

[2. Stesura del Codice 7](#_Toc95144736)

[2.1 Le Classi DAO 8](#_Toc95144737)

[2.1.1 UML 9](#_Toc95144738)

[2.2 Le Classi Controller 9](#_Toc95144739)

[2.2.1 UML 10](#_Toc95144740)

[2.3 Dipendenze 12](#_Toc95144741)

[3. Diagramma delle Classi 13](#_Toc95144742)

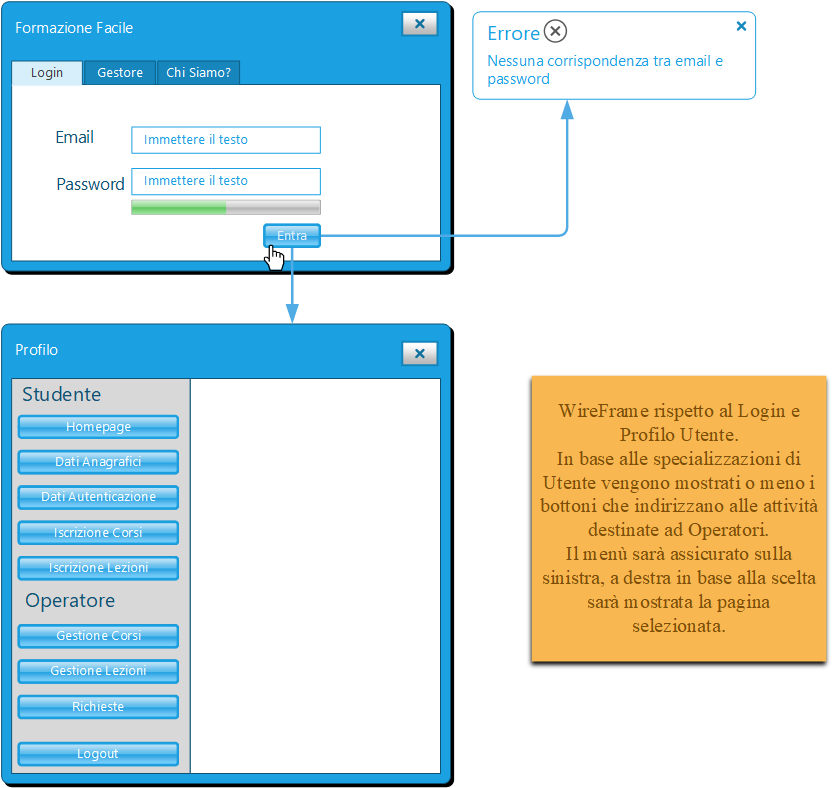
[4. Sequence Diagram 14](#_Toc95144743)

[5. CRC Card 15](#_Toc95144744)

1. Analisi del problema

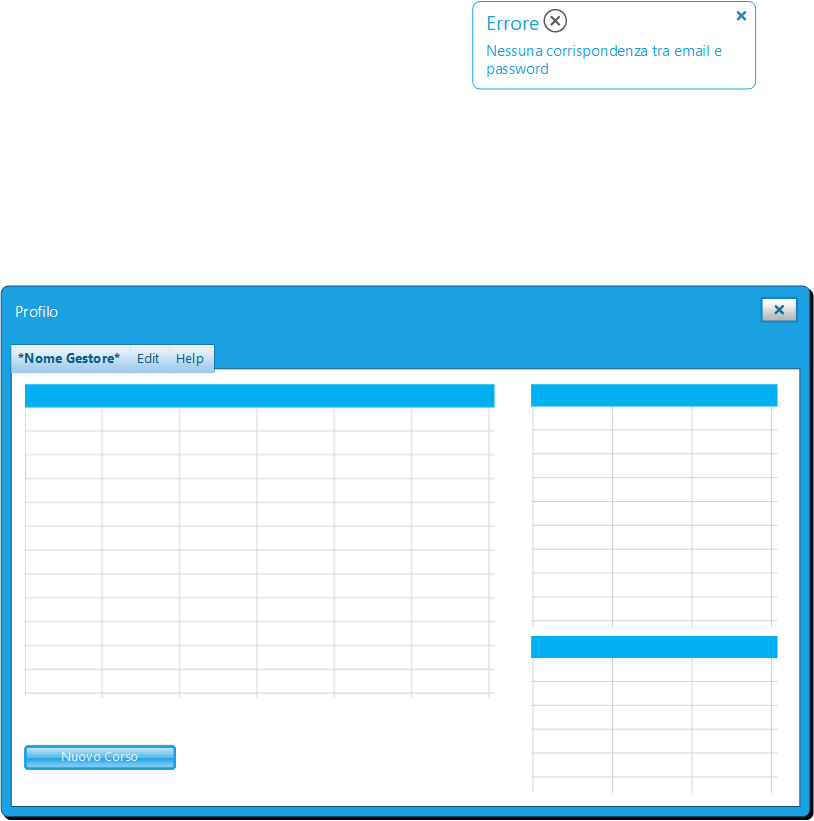
Dopo un accurata analisi della traccia,si è proseguito con l’analizzare il problema posto ricercando una soluzione Software che sià adeguata al problema stesso.

In primis si è ragionato ad un implementazione grafica **wireframe** che sia semplice e intuitiva, l’obbiettivo prefissato è stato di mantenere un interfaccia minimal che quindi presenti pochi colori e che presenti meno Stage / Frame possibili.

1.1 WireFrame

Le pagine visualizzate a destra sono composte per lo più da TableView che permettono l’interazione tramite click destri sulle righe che ci interessano.

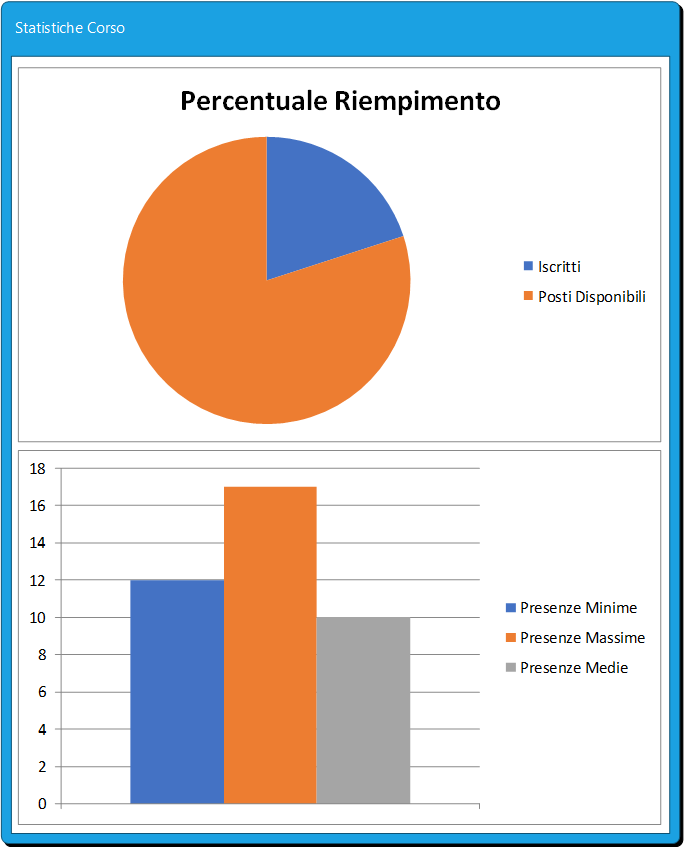
La pagina principale,tramite un Tab Pane permette di differenziare il Login destinato agli Utenti rispetto a quello destinato ai Gestori di Corsi.

Effettuando il login tramite il tab riservato ai Gestori si accede al Profilo Gestore.

**WireFrame Profilo Gestore**

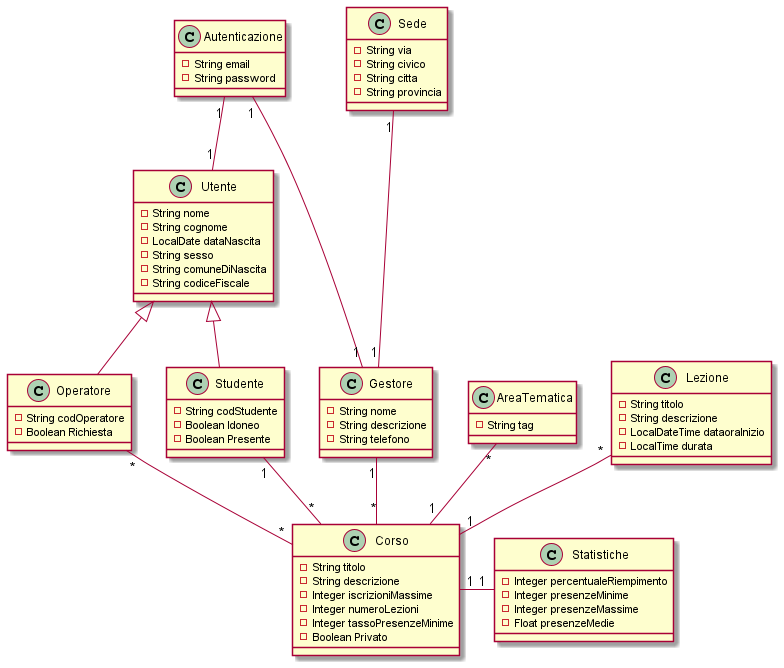
A differenza del Profilo Utente il Profilo Gestore tramite tre TableView mostra i dettagli su **Corsi,Operatori e Iscritti** dei Corsi erogati.

Come nel Profilo Utente si può interagire con ogni tabella tramite click destri sulle righe.

Gli**Operatori** e i **Gestori** hanno accesso alla pagina che permette la visualizzazione delle **Statistiche** del Corso,che vengono mostrate tramite Grafici.

1.2 Tecnica di implementazione

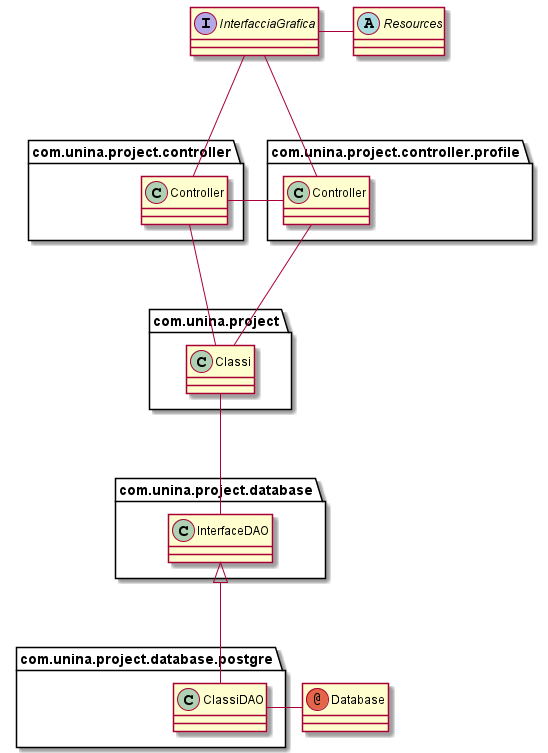
In seguito abbiamo pensato come implementare l’idea sviluppata precedentemente in **wireframe** tramite **Java**,valutando tra **Swing** e **Java FX**,la scelta è ricaduta sull’ultimo poiché mette a disposizione la possibilità di mostrare i dati tramite Grafici inoltre permette di avere un interfaccia decisamente più moderna.

* 1. Individuazione degli Oggetti

Individuati ciò che saranno gli Oggetti del Programma possiamo procedere con la stesura del codice.

I dettagli di ogni Classe sono riportati nel paragrafo dedicato alle [Carte di Class Responsibility Collaboration ( **CRC Cards** ) .](#_5._CRC_Card)

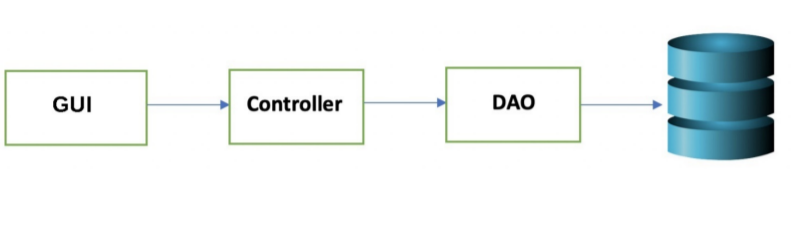
2. Stesura del Codice

Avendo le idee chiare su come implementare l’idea tramite Software si è iniziato suddividendo il programma in vari **package**.

Ogni **package** comunica con l’altro in modo bidirezionale, ogni classe richiama elementi di altri **package**. La suddivisione in più pacchetti è utile per suddividere le varie parti del Programma.

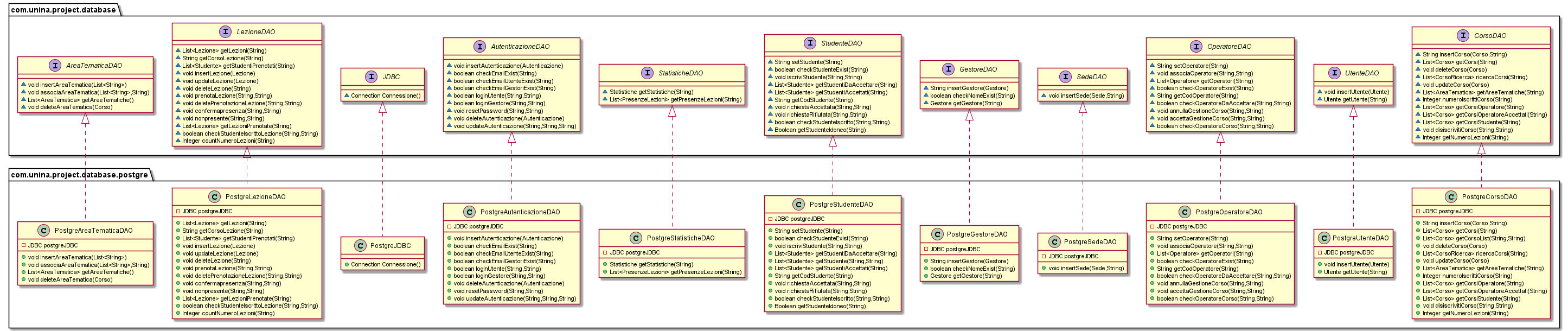
|  |  |
| --- | --- |
| *Package* | *Funzione* |
| **com.unina.project** | contiene gli Oggetti/Tipi definiti nel Programma. |
| **com.unina.project.controller** e **com.unina.project.controller.profile** | contengono tutte le classi Controller che interagiscono con l’interfaccia grafica e con gli fxml contenuti nelle **resources**. |
| **com.unina.project.database** | contiene le interfacce dei DAO relativi al Database. |
| **com.unina.project.databse.postgre** | contiene le classi DAO specifiche **postgre** che implementano le interfacce. |

## 2.1 Le Classi DAO

Le Classi **Data Access Object** sono state aggiunte per permettere al resto del Programma di essere indipendente dal Database. Le Query sono contenute esclusivamente al loro interno per permettere al Software nel caso in cui si cambiasse Database di modificare esclusivamente ciò che sono le classi che implementano le **Interface**.

Le implementazioni delle **interface DAO** a loro volta verranno interfacciate con i **Controller**, che avranno il compito di fare da intermediari con le interfacce grafiche, e quindi far **isolare** completamente le interfacce grafiche dal mondo della base dati.

### 2.1.1 UML

[](file:///C:\Users\agost\OneDrive\Desktop\OO-BDI-Project\UML\umldatabase.png)UML dei package che contengono Interfacce e Classi DAO

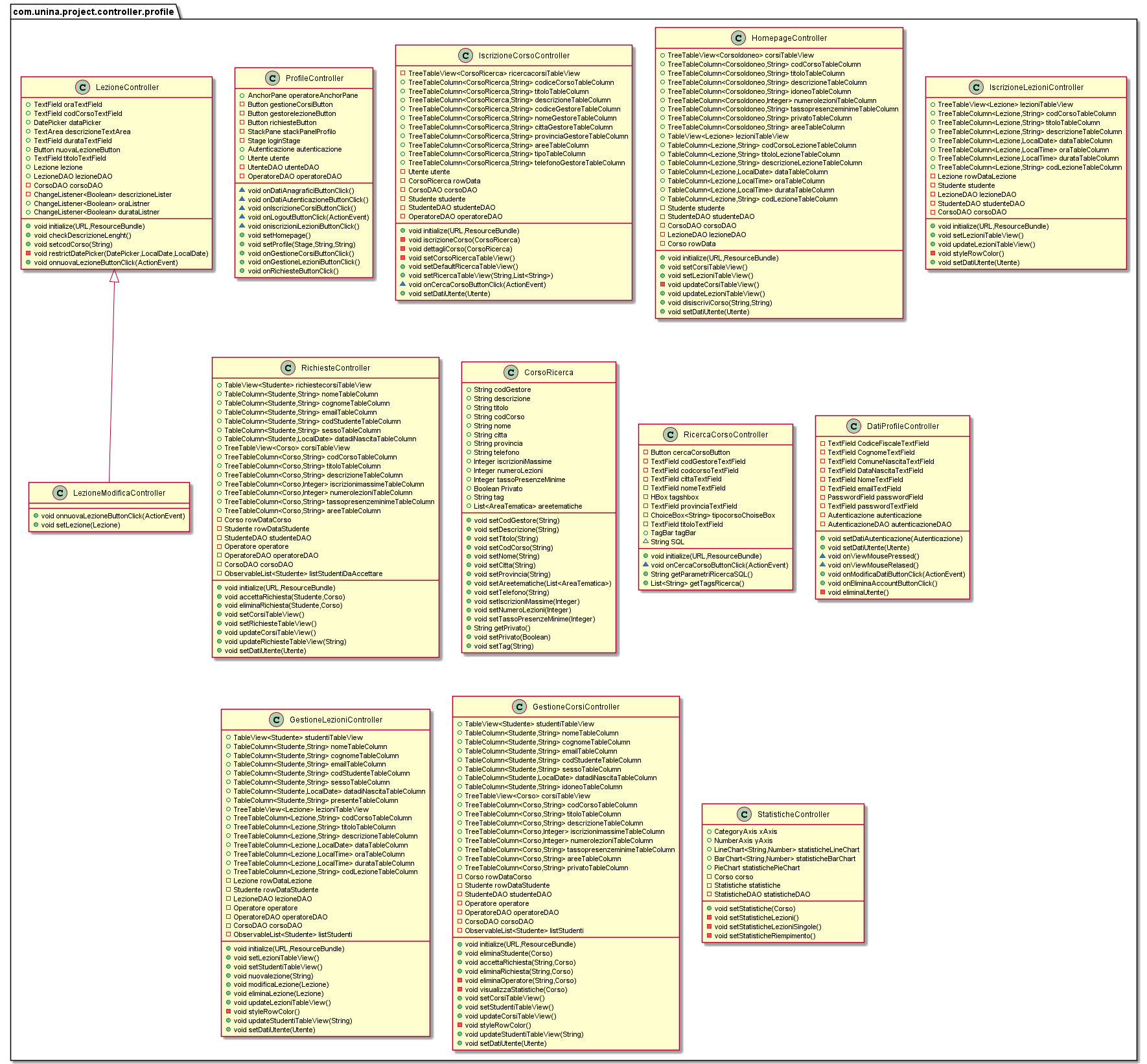
*Si consiglia di fare click sopra l’immagine che apre il file PNG per una visione migliore*

## 2.2 Le Classi Controller

Le classi Controller sono le classi che gestiscono l’interfaccia grafica. Ogni **Controller** è associato ad un **fxml**, contenuto nelle **resources**, che definisce tutti gli elementi grafici dell’interfaccia. Ad ogni elemento può essere associato un **id,** che definito nel Controller, permette la gestione/personalizzazione dell’elemento.

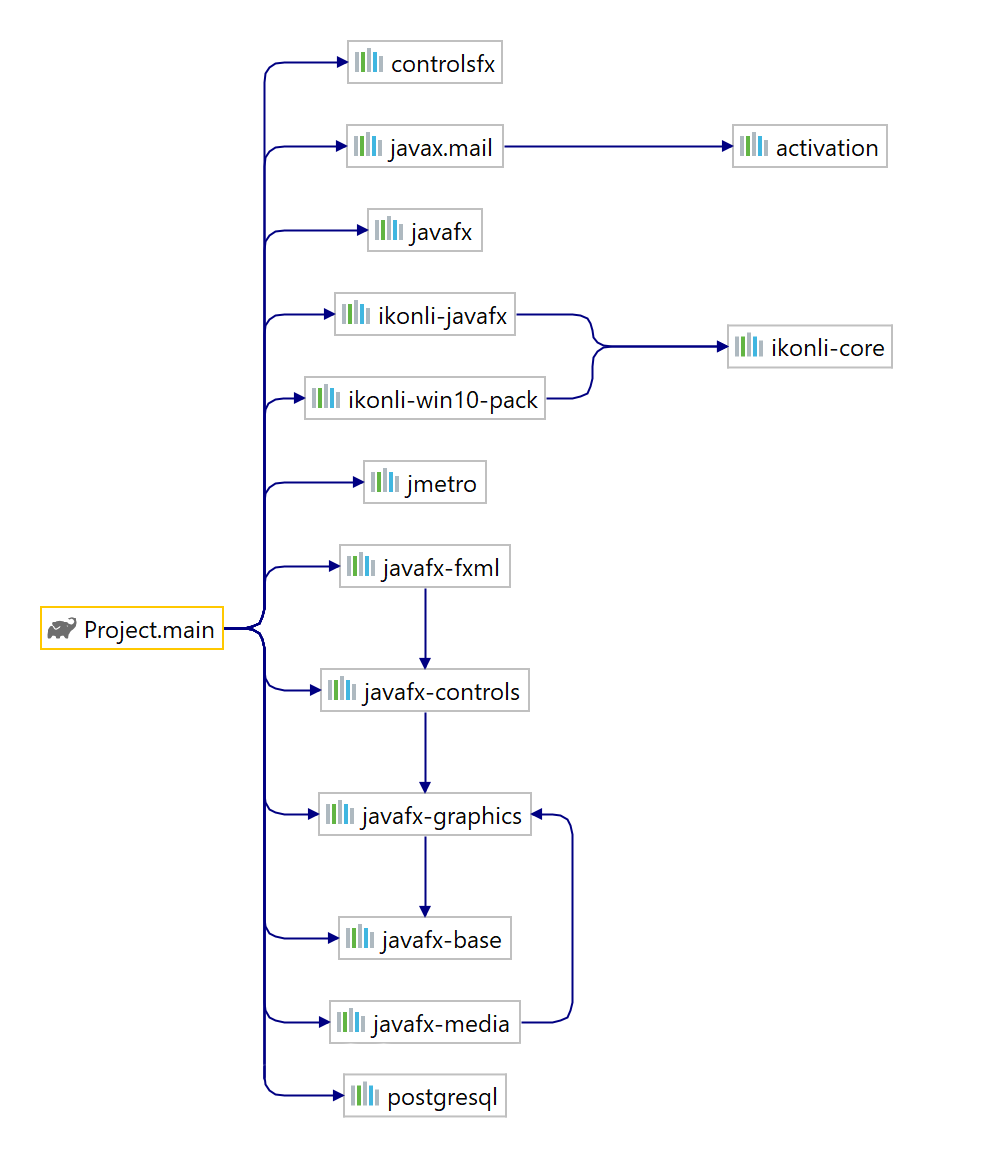
I Controller sono distribuiti in **due package**, un package è dedicato ai Controller del Profilo Utente, l’altro ai restanti.

### 2.2.1 UML

[](file:///C:\Users\agost\OneDrive\Desktop\OO-BDI-Project\UML\umlcontroller1.png)*Si consiglia di fare click sopra l’immagine che apre il file PNG per una visione migliore*

## 2.3 Dipendenze

Il funzionamento del programma dipende da dei **moduli terzi** che permettono l’uso di funzioni non implementate nativamente nel **JDK.**

Le dipendenze sono gestite da **Gradle** che è uno strumento di automazione della compilazione **(build automation tool).**

|  |  |
| --- | --- |
| *Moduli* | *Funzione* |
| **JavaFX** | Aggiunge tutte le funzionalità grafiche di JavaFX, anche se sviluppato da Oracle non è contenuto nel JDK. |
| **ControlsFX** | Permette di aggiungere elementi grafici avanzati e funzionalità extra.  Nel progetto è stato usato per le TextField con Suggerimenti. |
| **JavaX.Mail** | Permette di interfacciarsi con il mondo mail tramite Java.  Nel progetto si inviano codici di verifica durante la registrazione. |
| **Ikonli** | Permette di aggiungere Icone all’interno degli FXML senza ricorrere ad immagini. |
| **JMetro** | Tema Grafico ispirato all’interfaccia Windows. |
| **PostgreSQL** | Aggiunge le funzionalità di iterazione Java-Database **JDBC** (Java DataBase Connectivity). |

[*Consulta la JavaDoc per visualizzare i dettagli.*](file:///C:\Users\agost\OneDrive\Desktop\OO-BDI-Project\documentazione\javadoc\index.html)

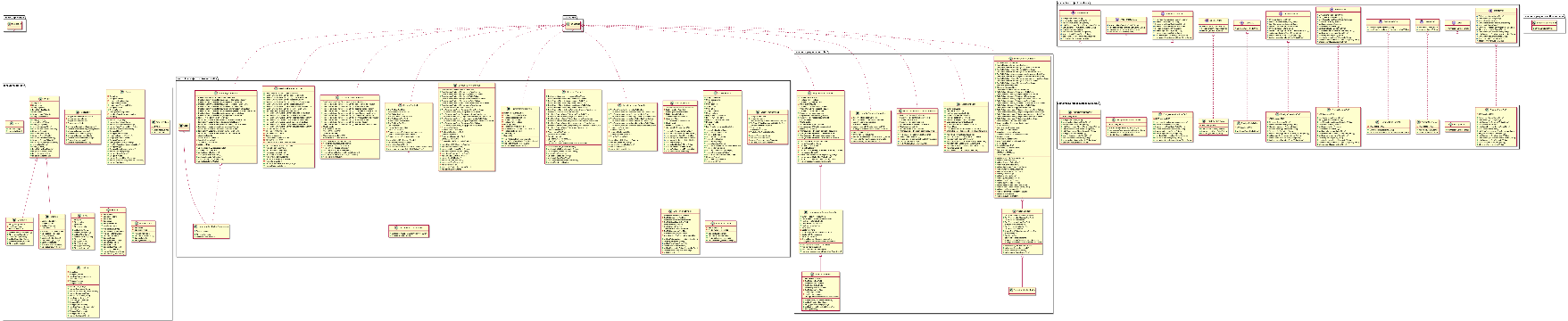
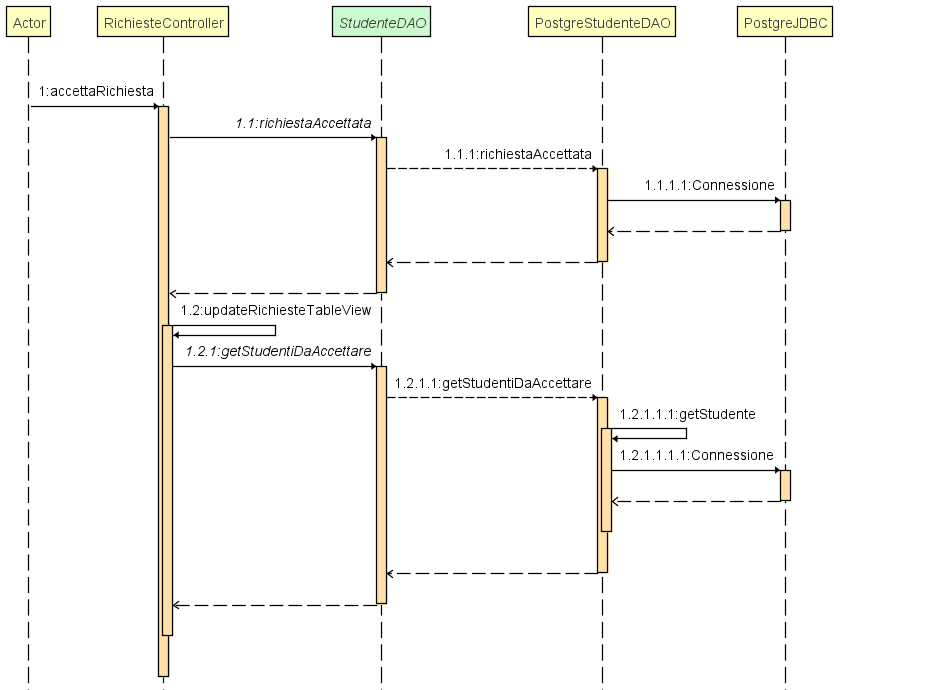
[](file:///C:\Users\agost\OneDrive\Desktop\OO-BDI-Project\UML\umlcompleto.png)3. Diagramma delle Classi

Diagramma delle Classi Completo dell’intero proggetto, a causa della dimenzione elevata, si consiglia la visione esterna.

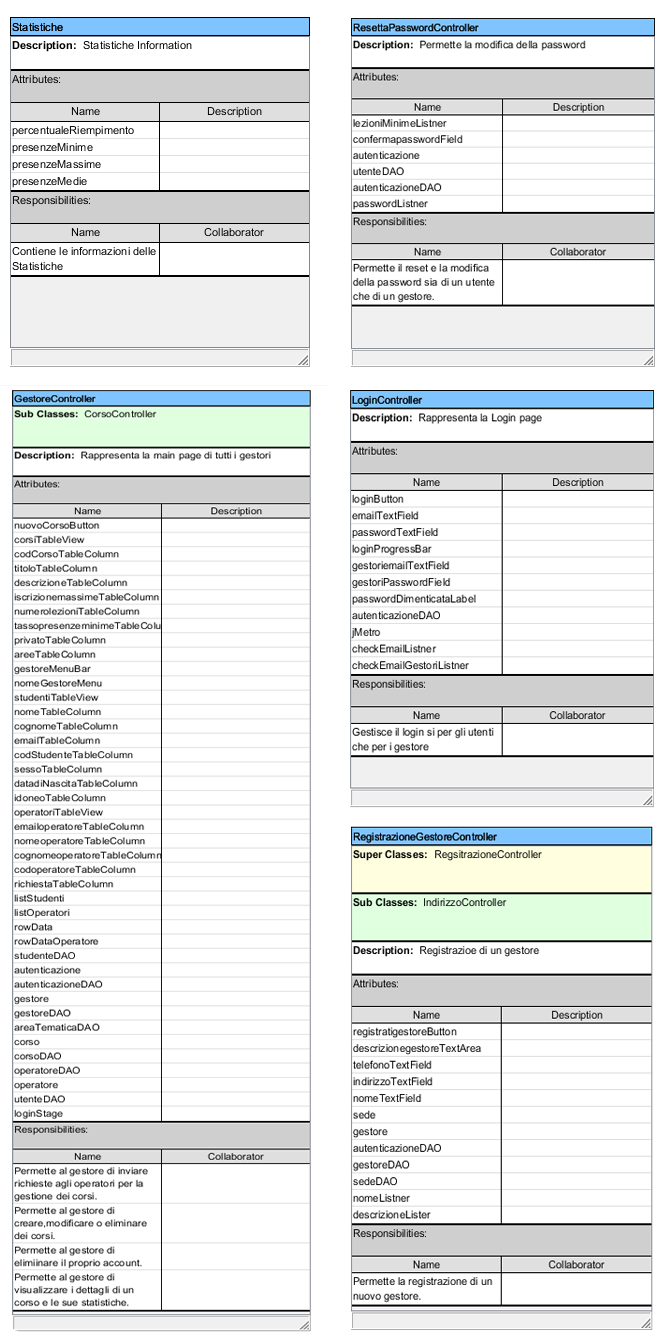
[*Consulta il PDF dedicato.*](file:///C:\Users\agost\OneDrive\Desktop\OO-BDI-Project\UML\umlcompleto.pdf)

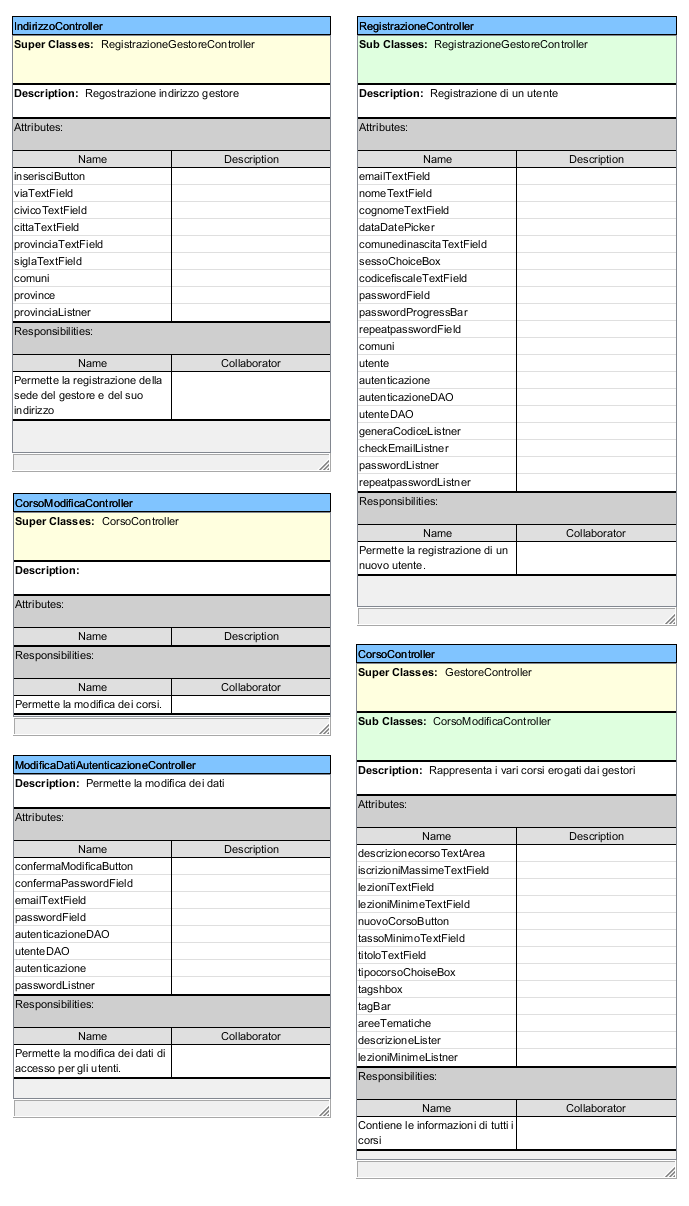
# 4. Sequence Diagram

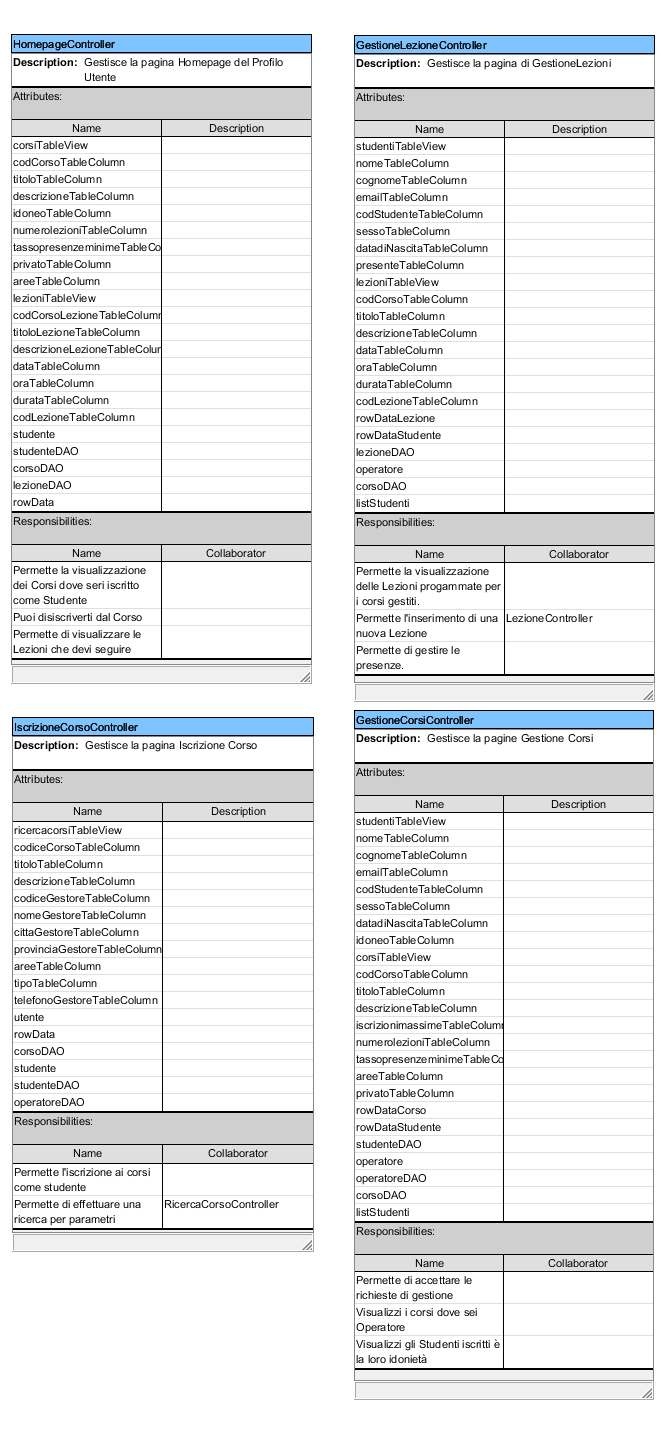
[](file:///C:\Users\agost\OneDrive\Desktop\OO-BDI-Project\UML\RichiesteController_accettaRichiesta.png)*Sequence Diagram del metodo oninviasottoscrizioneButtonClick del Controller RegistrazioneGestoreController*

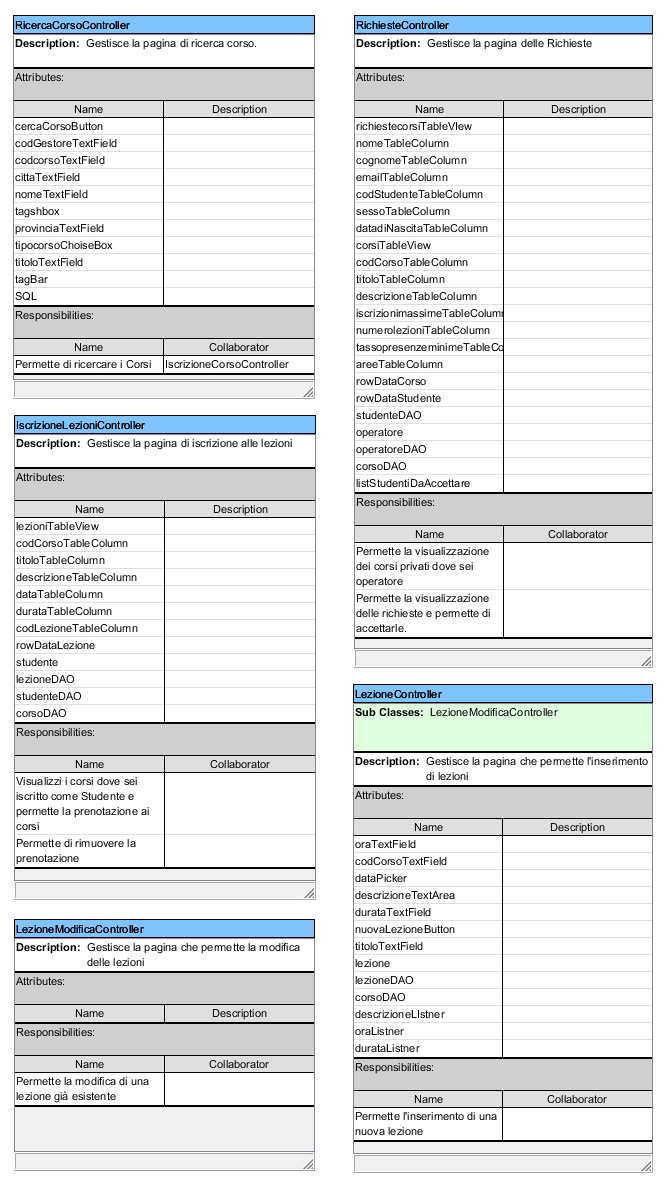
*Sequence Diagram del metodo accettaRichiesta del Controller RichiesteController*

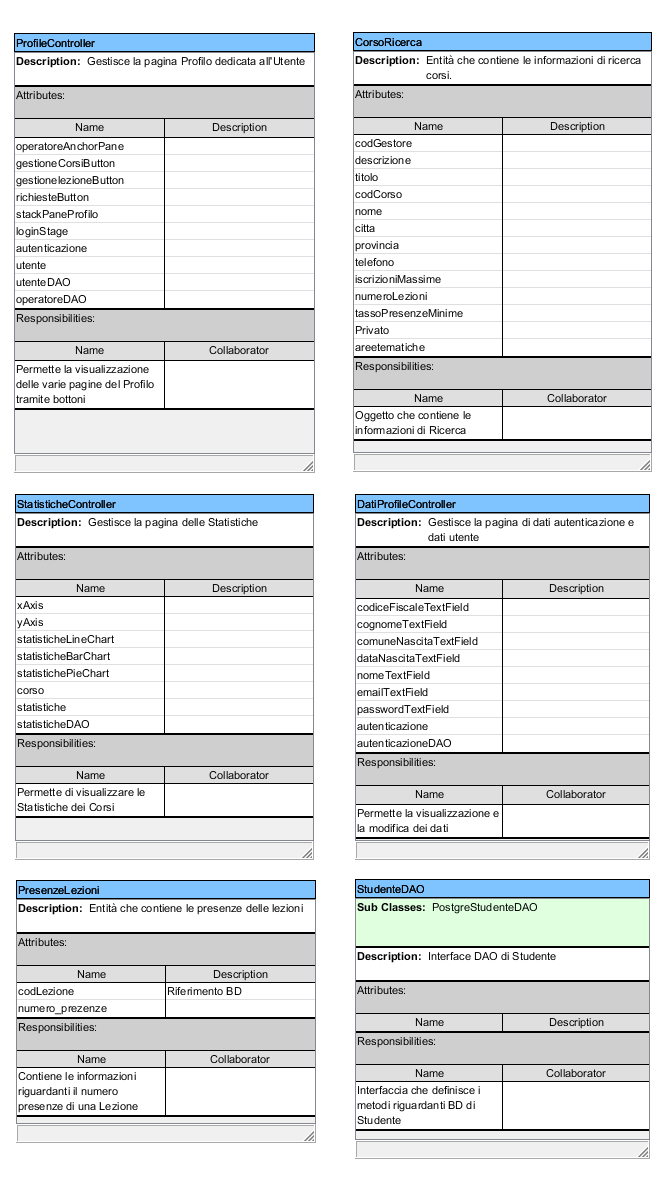
# 5. CRC Card



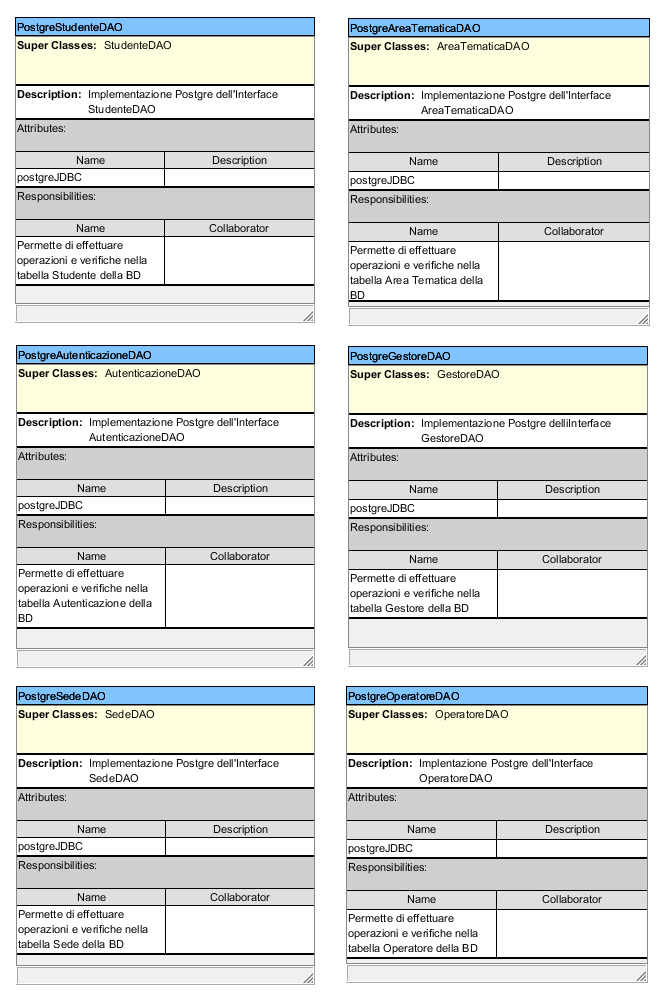
**

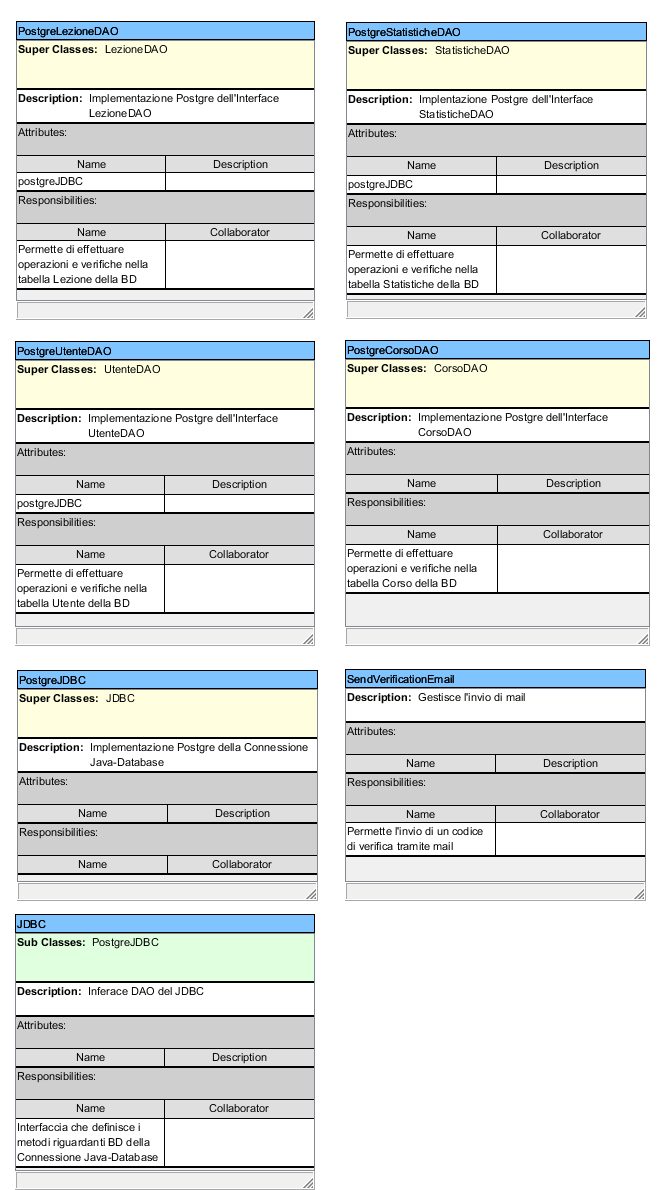
**

**

**



**

**