University of Pisa

SCUOLA DI INGEGNERIA

Corso di Laurea in Artificial Intelligence and Data Engineering

Task3 Documentation

Candidati

Alice Nannini
Giacomo Mantovani
Marco Parola
Stefano Poleggi

Relatore

Prof. Pietro Ducange

Anno Accademico 2019–2020

Contents

1 Introduction

This is an application to browse and evaluate university courses, called **Student-Evaluation**.
The application is developed to allow students (users) to view all the courses of a specific university with related comments.
Looking the table on the right side, a user can browse all subjects and by clicking an element of this table, on the left section, the user can see more information about the chosen element: general information and comments.
In order to filter the list of subjects, there is a choice box, thanks to which the user can select a specific degree course.
In order to leave a comment, it is necessary to log in, otherwise, the application will allow interaction in read-only.
There are two buttons in the bottom left corner to allow students to update or delete their comments. There is no form to register into the application, it is assumed that users are already registered into the system, but there is an administrator that can add, update and delete subjects and all the informations related.

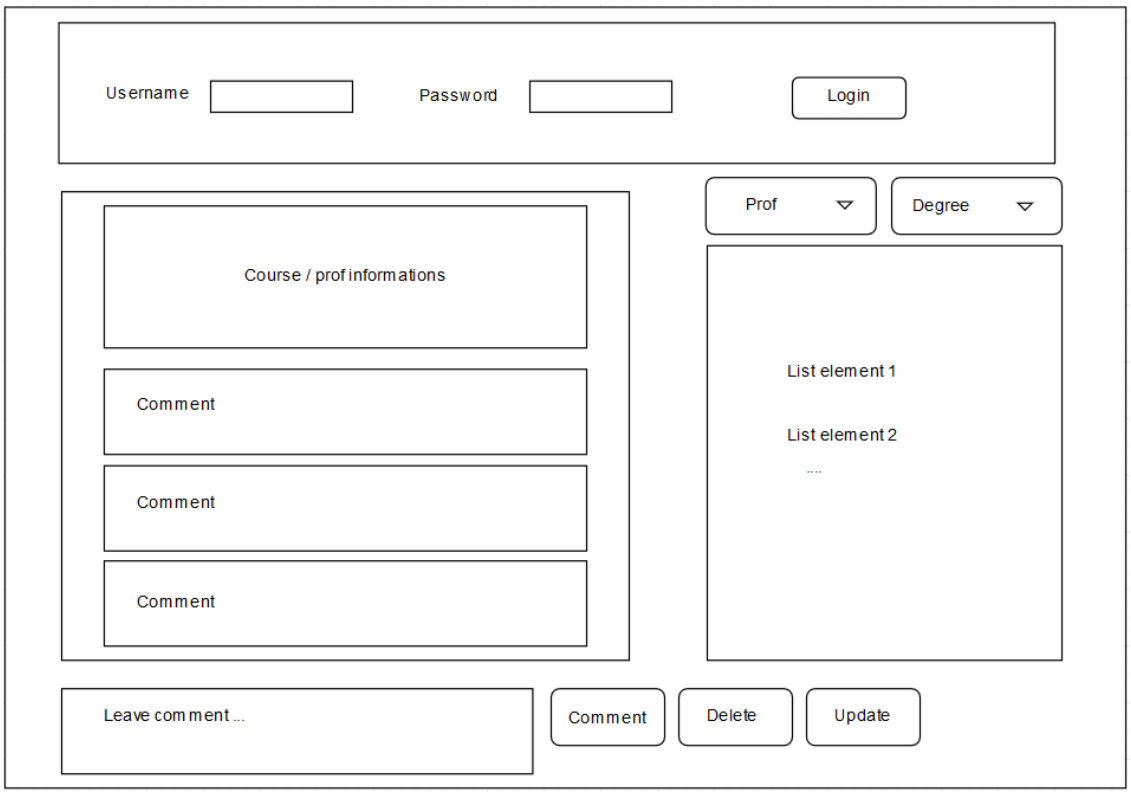

Figure 1: Mockup

2 Analysis and workflow

2.1 Requirements

2.1.1 Functional requirements

The system has to allow the guest to carry out basic functions such as:

- To select a course from the list and view information and comments.
- To select a degree course from the list, filtering subjects.

In addiction to the guest functions, the system has to allow the user to carry out basic functions such as:

- To login to the system.
- To upload comments on a course.
- To update a comment of a course only if the user is the owner.
- To delete a comment of a course only if the user is the owner.

The system has to allow the administrator to carry out basic functions such as:

- To login to the system.
- To add a course.
- To update a course.
- To delete a course.
- To associate a professor to a course.
- To delete any comment.

2.1.2 Non-functional requirements

- Usability, ease of use and intuitiveness of the application by the user.
- Avaliablility, with the service guaranteed h24.
- The system should support simultaneous users.
- The system should provide access to the database with a few seconds of latency.

2.2 Use cases

Actors

- Guest : this actor represents a user who is not logged into the system
- Student : this actor represents a user who is logged into the system
- Admin : this actor represents the administrator of the system

2.2.1 Use Cases Description

Event	UseCase	Actor(s)	Description
Log in, Log out	Login, Logout	Admin, Student	The user logs in/out the application. The system browses the professors' list by the degree course of the logged user and returns it on the interface.
View all the subjects	Browse, Find, View P/S	User	The user chooses that he wants to view the list of all subjects. The system browses the data on the db and returns them on the interface.
View the comments and information of a subject	Browse, Find, View C	User	The user clicks on a record of the subject table. The system browses on the db the comments related to that subject and returns them on the interface.
Add a comment	Add C	Admin, Student	The user submits the text of his comment. The system updates the db and the interface.
Update a comment	Update C	Admin, Student	The user selects the comment and commits the new text. The system updates the db and the interface.
Delete a comment	Delete C	Admin, Student	The user selects the comment and submits the delete. The system updates the db and the interface.
View the subjects by degree	Browse, Find, View P/S	User	The user selects from the choice-boxes the degree course and the list (subjects) he's interested in. The system browses on the db the subjects filtered by the chosen degree and returns them on the interface.
Add a subject	Add P/S	Admin	The user submits the name and other information of the new subject. The system updates the db and the interface.
Update a subject	Update P/S	Admin	The user selects the subject and commits the new information. The system updates the db and the interface.
Delete a subject	Delete P/S	Admin	The user selects the subject and submits the delete. The system updates the db and the interface.

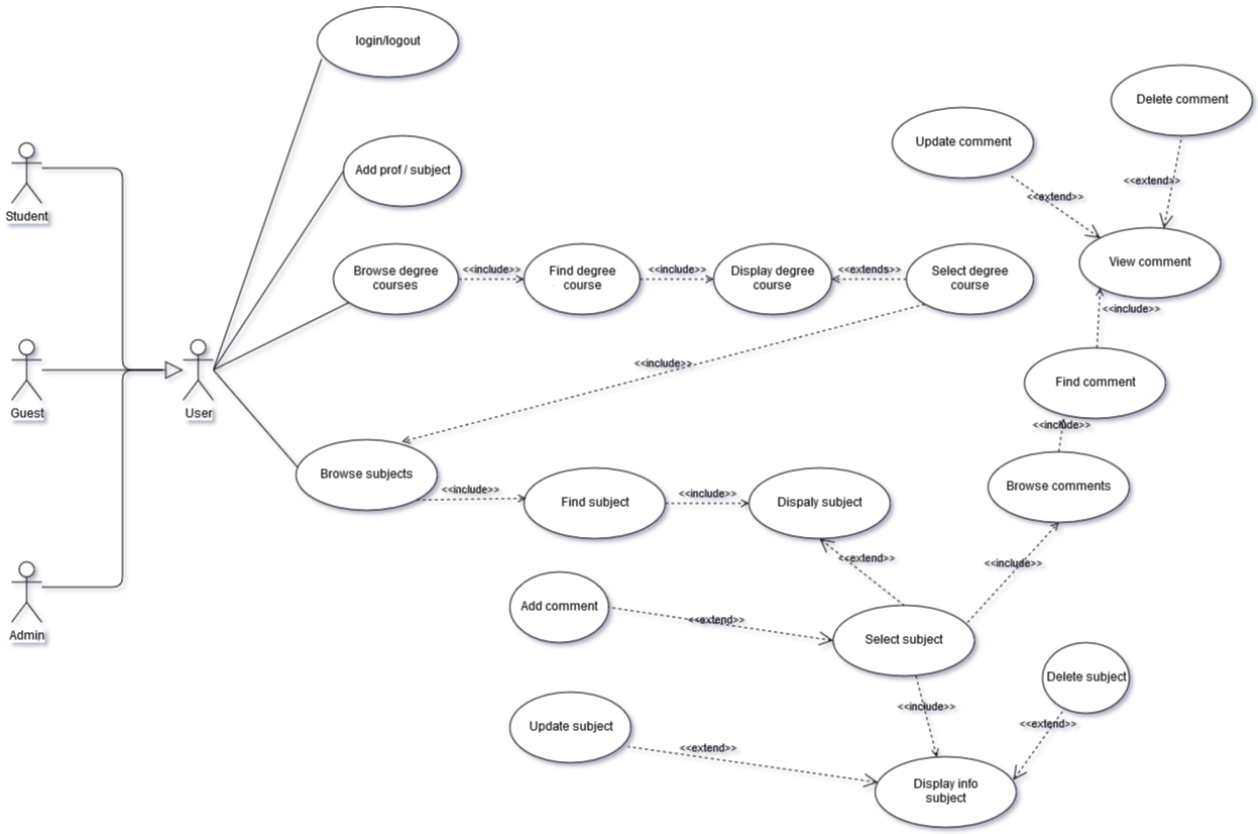

Figure 2: Use cases diagram

2.3 Analysis of entities

This diagram represent the main entities of the application and the relations between them.

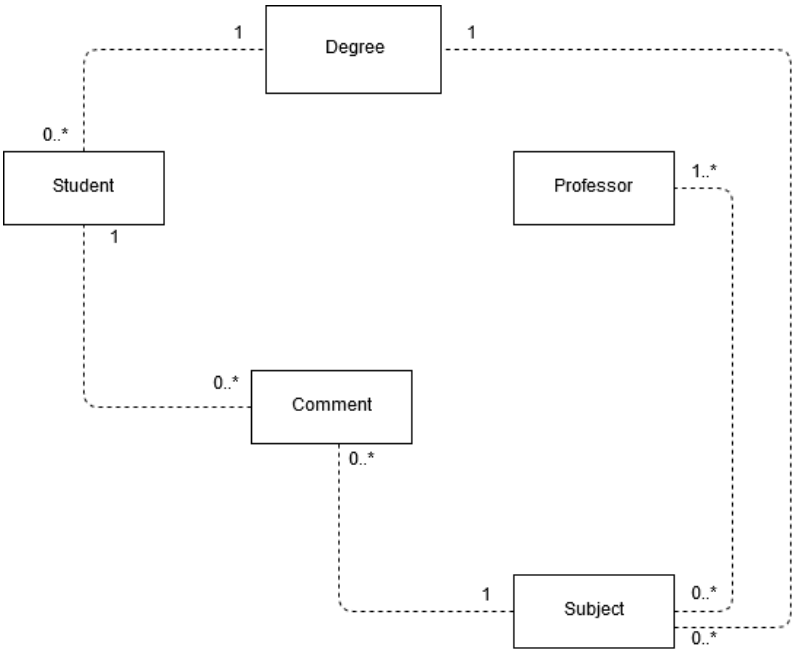

Figure 3: UML analysis diagram

3 Design

3.1 Database Choice

This application is based on many join operations between the entities, in order to obtain each professor associated with a subject, or each comment associated with a subject, or yet display all the subjects associated with a degree course. For this reason, if the amount of data available is very high as expected, a relational database will be computationally expensive. Then the choice fell on a graph database, which manage to work very well and very fast on a model made with a lot of entities and relations.

3.2 Software Architecture

The application is designed over 2 different layers, see figure 4:

- Front-end
- Back-end

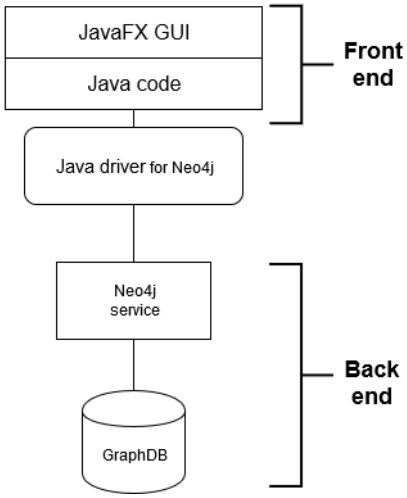

Figure 4: Software architecture diagram

3.3 Structure of the database

In graph databases the entities are modeling using the vertexes, in our model there are the following entities:

- Professor
- Student
- Degree
- Subject
- SubjectComment

Moreover the relations are modeling using the edges, in our model there are the following edges:

- Attends, connects a students to a degree

- Teaches, connects a professor to a subject
- Belongs, connects a course to a degree
- Wrote, connects a student to a subject

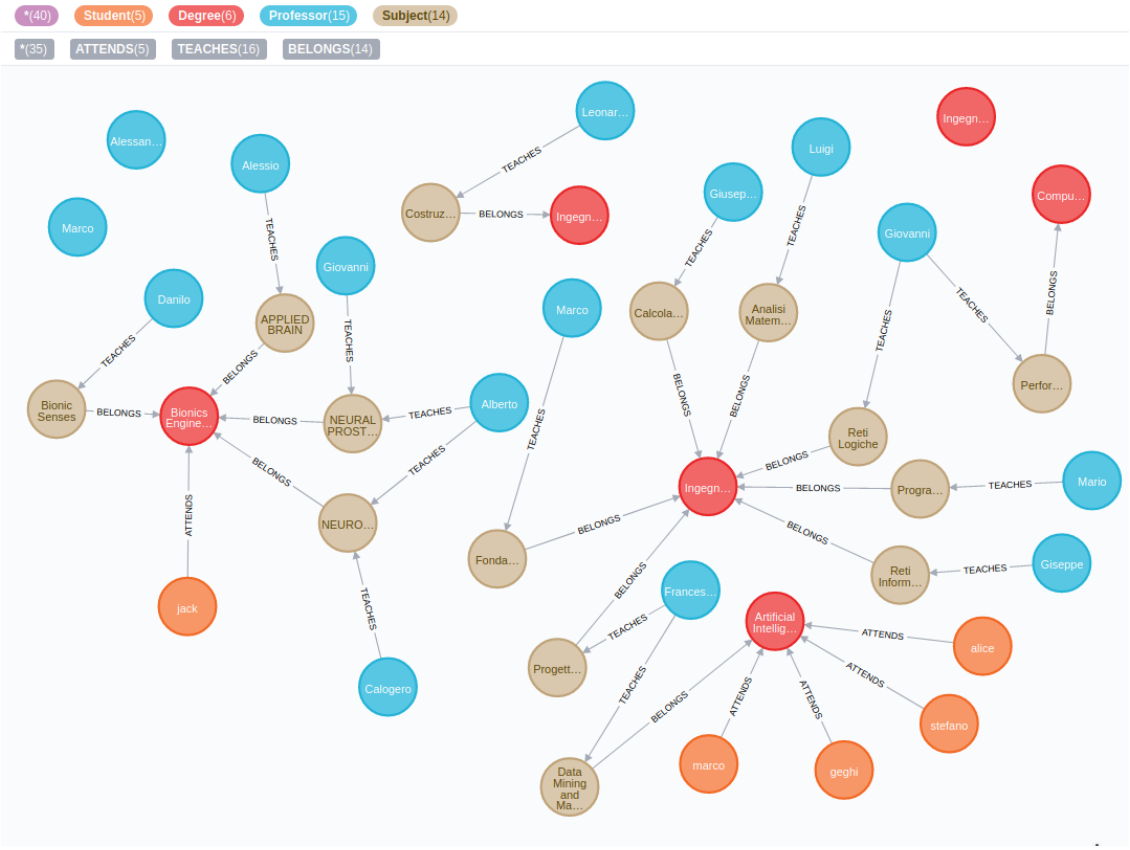

Figure 5: Graph model

4 Implementation

4.1 Used Technologies

The application is developed in java programming language, version 11.0.4, and in JavaFX system to create the GUI, version 11, so it should run on each platform in which JVM is installed, but the application is tested and guardantee on Ubuntu 16 and Window OS. Moreover Maven is used to build and mantain the project, version 3.8.0.
The java driver for Neo4j manage the comunication between client application layer and mongo backend layer, version 3.2.1.
For the backend layer it is used a graph database: Neo4j, version 3.2.1.
So this application is tested using these technologies, considering these particular versions: for other versions the correct execution isn't guaranteed .

4.2 Java Classes Description

4.2.1 Entities

In this section is listed the classes who model the entities.

- Student
- Professor
- Subject
- Comment
- Degree

Each of them have some attributes and the related Get and Set methods, in order to avoid too much redundancy code, it is reported only an example of declaration.

```
public class Student{
   private final SimpleIntegerProperty id;
   private final SimpleBooleanProperty admin;
   private final SimpleStringProperty username;
   private final SimpleStringProperty password;
   private final SimpleObjectProperty<Degree> degree;
   private List<Comment> comments = new ArrayList<Comment>();

   public Student(int i, String u, String p, Degree d, boolean a) {
       id = new SimpleIntegerProperty(i);
       admin = new SimpleBooleanProperty(a);
       username = new SimpleStringProperty(u);
       password = new SimpleStringProperty(p);
       degree = new SimpleObjectProperty(d);
   }

   public int getId(){ return id.get(); }
   public void setId(int i){ id.set(i); }
   public boolean getAdmin() { return admin.get(); }
   public void setAdmin(Boolean b) { admin.set(b); }
   public String getUsername() { return username.get(); }
   public void setUsername(String n) { username.set(n); }
```

```
    public String getPassword() { return password.get(); }
    public void setPassword(String n) { password.set(n); }
    public Degree getDegree() { return degree.getValue(); }
    public void setDegree(Degree d) { degree.set(d); }
  }
```

4.2.2 Database manager

The other class in the project is DbManager. It handles the connection with Neo4j and provides all methods to interact with it.

```
  public class DbManager implements AutoCloseable {
       private final Driver driver;
       private final String uri = "bolt://localhost:7687";
       private final String user = "user";
       private final String password = "pwd";

       public DbManager() {
       driver = GraphDatabase.driver(uri, AuthTokens.basic(user, password));
       }

    // ... crud operation ...

       @Override
       public void close() throws Exception {
       driver.close();
       }
  }
```

4.3 CRUD operations

All crud operations are implemented using **Cypher query language**; it is a declarative graph query language that allows for expressive and efficient querying and updating of a property graph.

4.3.1 Create

This method of DbManager class guarantees the creation of a subject. A new vertex is inserted on the graph and is connected with a professor node using an incoming edge called TEACHES, moreover it is connected with degree node, using an outcoming edge called BELONGS.

```
public void createSubject(String name, int credits, String info, int profId, int degreeId) {
  try(Session session = driver.session(AccessMode.WRITE)){
     session.run("MATCH (p:Professor) WHERE id(p) = $profId " +
                 "MATCH (d:Degree) WHERE id(d) = $degreeId " +
                 "CREATE (a:Subject {name:$name, cfu:$cfu, info:$info})," +
                 "(p)-[:TEACHES]->(a), (a)-[:BELONGS]->(d);",
     Values.parameters("name",name,"cfu",credits,"info",info,
                       "profId",profId,"degreeId",degreeId) );
   }
}
```

4.3.2 Read

This method reads from database all degrees and inserts them in a list.

```
public List<Degree> getDegreeCourses() {
   List<Degree> list = new ArrayList<>();

   try(Session session = driver.session(AccessMode.READ)){
     StatementResult sr = session.run("MATCH (dd:Degree)''+
                                      "RETURN ID(dd), dd.name;");

     while(sr.hasNext()) {
        Record r = sr.next();
        System.out.println(r.toString());
        list.add(new Degree(r.get("ID(dd)").asInt(),r.get("dd.name").asString()));
     }
  }
  return list;
}
```

4.3.3 Update

The following snippet of shows how to execute an updating operation of an entity professor.

```
  public void updateProfessor(int profId, String name, String surname) {
     try(Session session = driver.session(AccessMode.WRITE)){
        session.run("MATCH (p:Professor) WHERE ID(s) = $profId\n" +
                    "SET p.name = $name, p.surname = $surname;",
        Values.parameters("profId",profId,"name",name,"surname",surname) );
     }
  }
```

4.3.4 Delete

This example of deleting operation shows how to remove subject, given its id and related relations.

```
   public void deleteSubject(int subjectId) {
       try(Session session = driver.session(AccessMode.WRITE)){
              session.run("MATCH (n:Subject)" +
                          "WHERE id(n) = $idSubject" +
                          "DETACH DELETE n;",
              Values.parameters("idSubject",subjectId));
        }
   }
```

5 User Manual

When you first run the application, the interface you get is the one in figure 6.

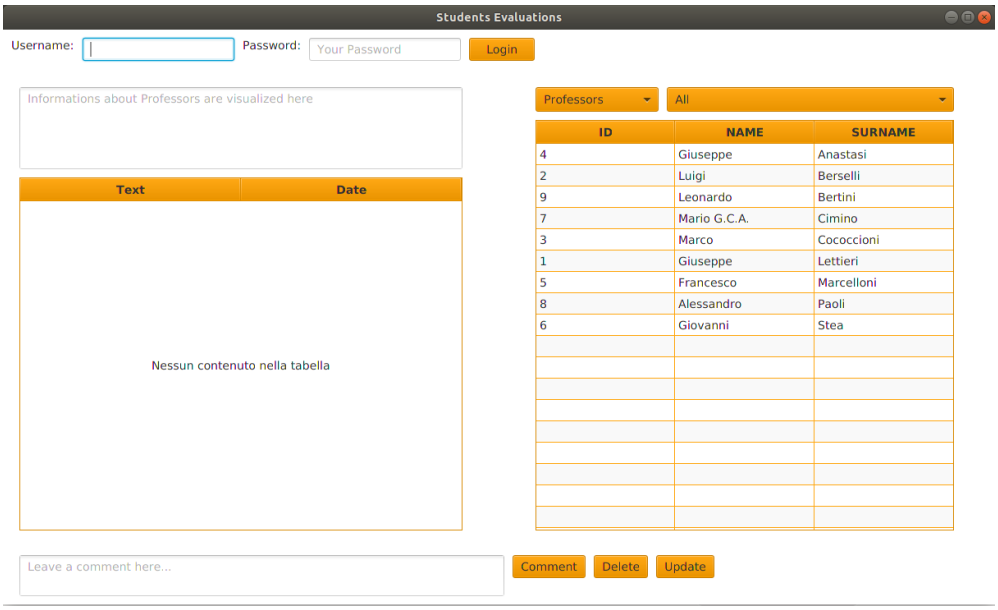

Figure 6: First view of the application

The default display includes the list of all registered professors in the table on the right. You can choose to display the professors of a single degree course, using the drop-down menu on the right (fig. 7), or decide to view the list of subjects (fig. 8), for which is also available the degree course's filter.

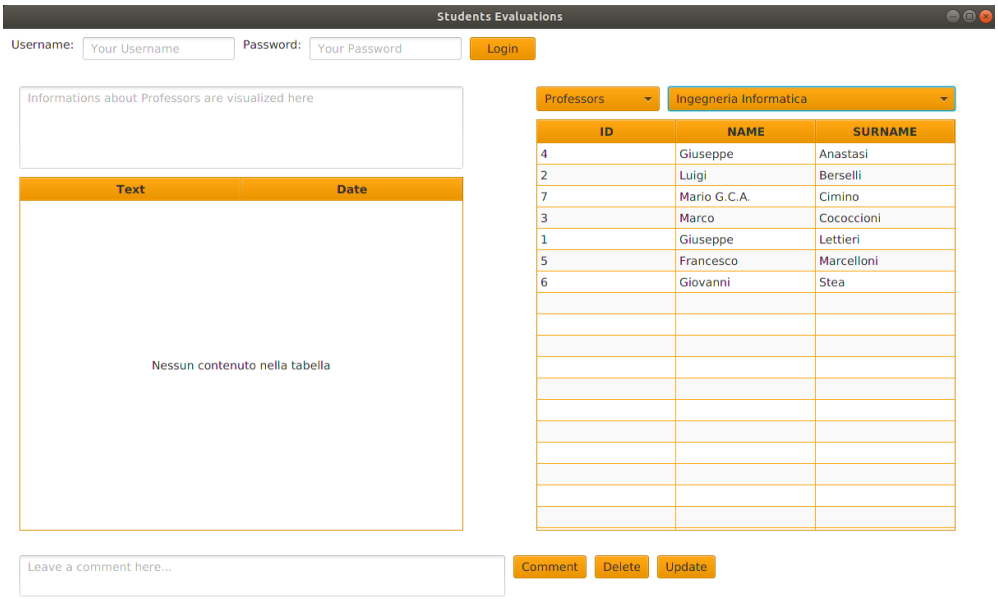

Figure 7: Selection of professors filtered by "Ingegneria Informatica" degree course

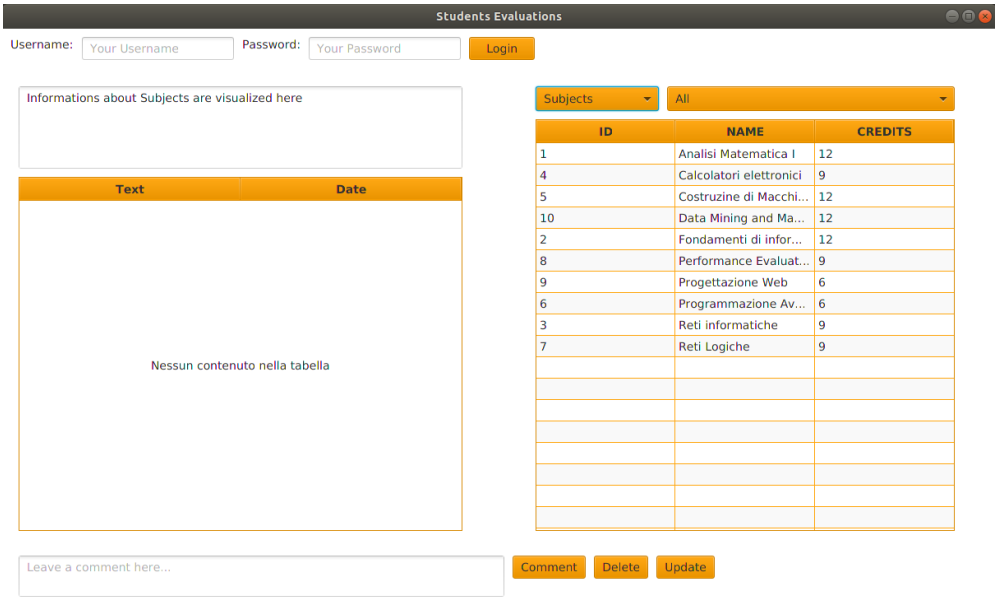

Figure 8: Selection of subjects

If you have a registered account, you can log in to the application, so that the comments' operations aren't blocked. Enter your username and your password in the suited fields at the top and click on "Login" (fig. 9).

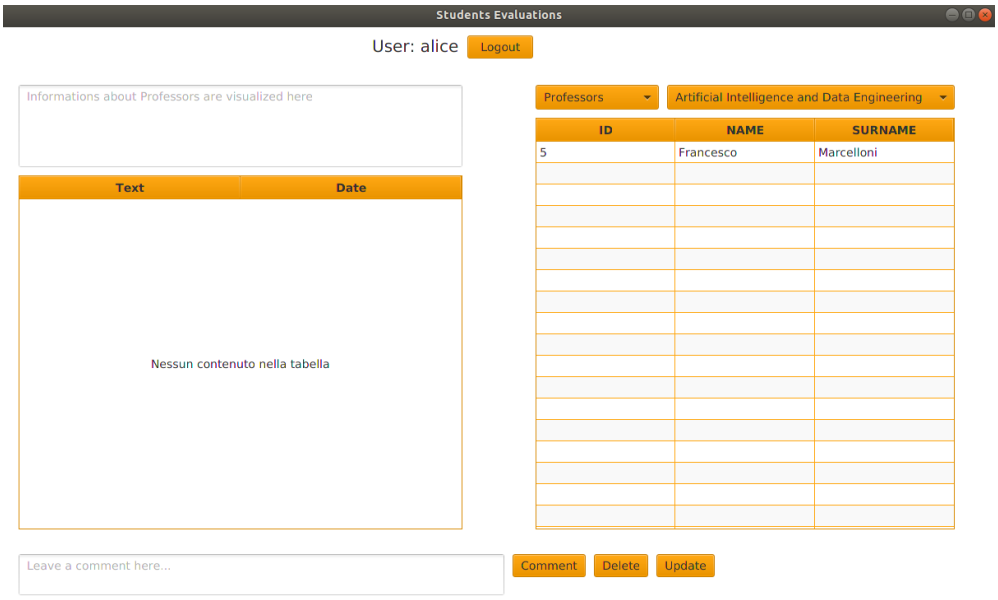

Figure 9: Application interface after the user "Alice" has logged in

If you now want to be able to see the comments associated with a particular professor, you have to click on the name of the professor: in the table on the left the list of comments already received will appear (fig. 10). With this operation, you'll be able to visualize also the information related to that professor.

To leave a comment, you need to enter the text in the field below the table and then click on the "Comment" button. The result obtained from these operations is shown in fig. 11.

You can also decide to modify the comment you just uploaded or another comment you made on a previous session. To do so, you need to click on the comment you want to update, change

Figure 10: Displaying the comments related to a professor

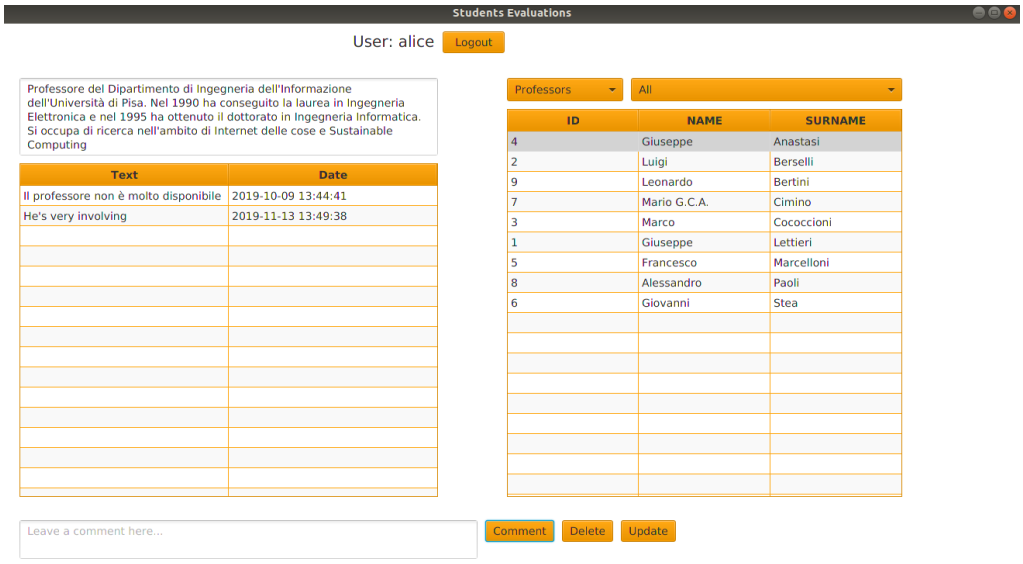

Figure 11: Interface after adding a comment

the text in the field below the table and then click on the "Update" button (fig. 12). Finally you have the chance to delete your comment, by clicking on "Delete" after selecting it. Notice that you can modify or delete just the comments that you made.

The operations of adding, updating and deleting work as well for the the subjects' comments.

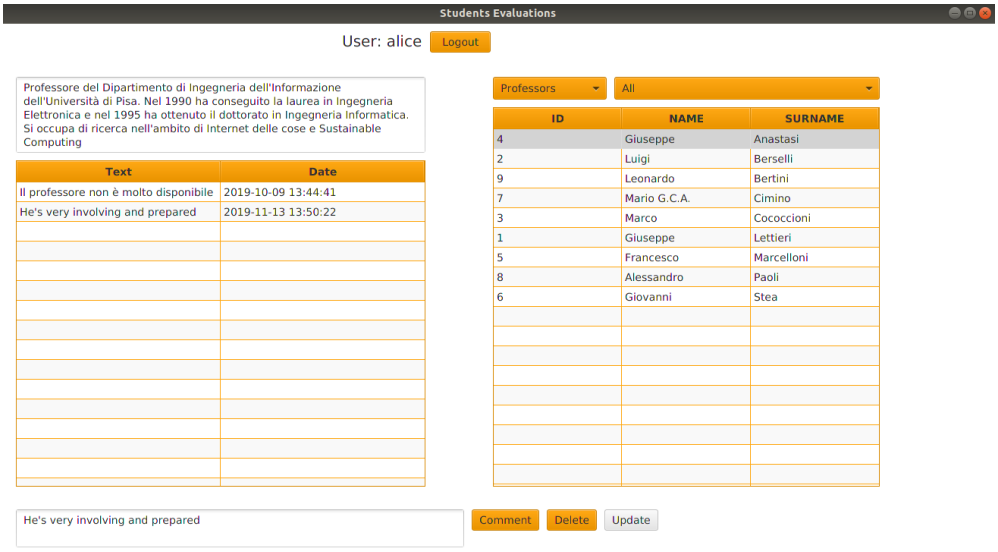

Figure 12: Interface after updating a comment

To log out, just click on the appropriate button at the top, next to the user label.

Moreover, if you don't have a registered username, you can still browse through the application, search for professors 'and subjects' information and read all comments. You are just unable to leave or change any comments.

5.1 Admin Manual

If you have an admin user, you are entitled to make changes both on the professors' and the subjects' lists. You need to log in inserting your username and password, and the application will recognize you as the administrator and show up the buttons for modifying the data (fig. 13).

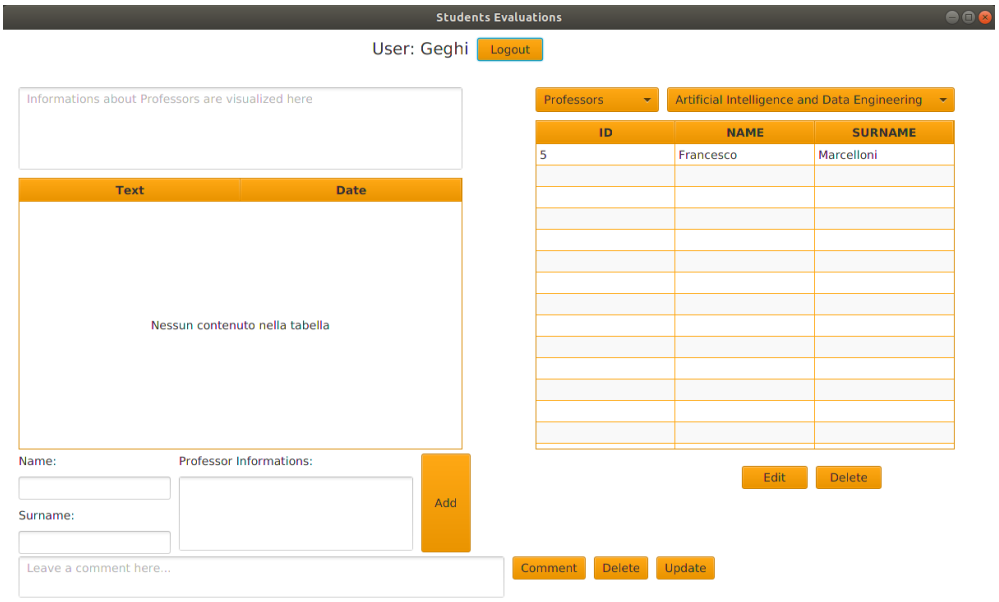

Figure 13: Interface after the administrator has logged in

You can choose to add a new professor, using the input fields at the bottom left. You have to specify the name, surname, and description, then press the "Add" button (fig. 14).

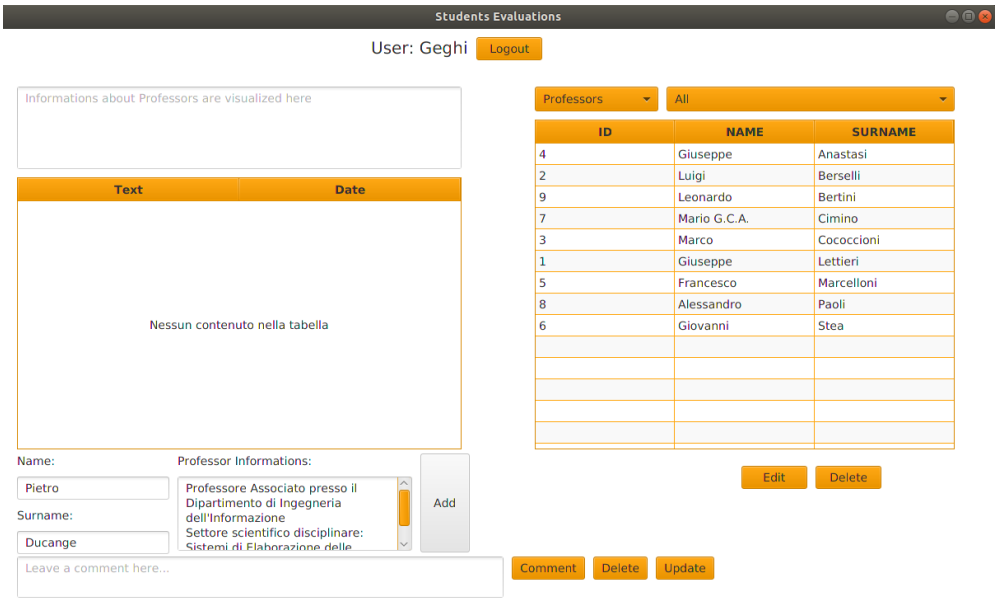

Figure 14: Adding a new professor

You can also modify the data related to a professor: click on the professor you are interested in and change the information shown in the apposite input fields. Finally, you have the chance to delete a professor by clicking on the "Delete" button after selecting the wanted professor (fig. 15).

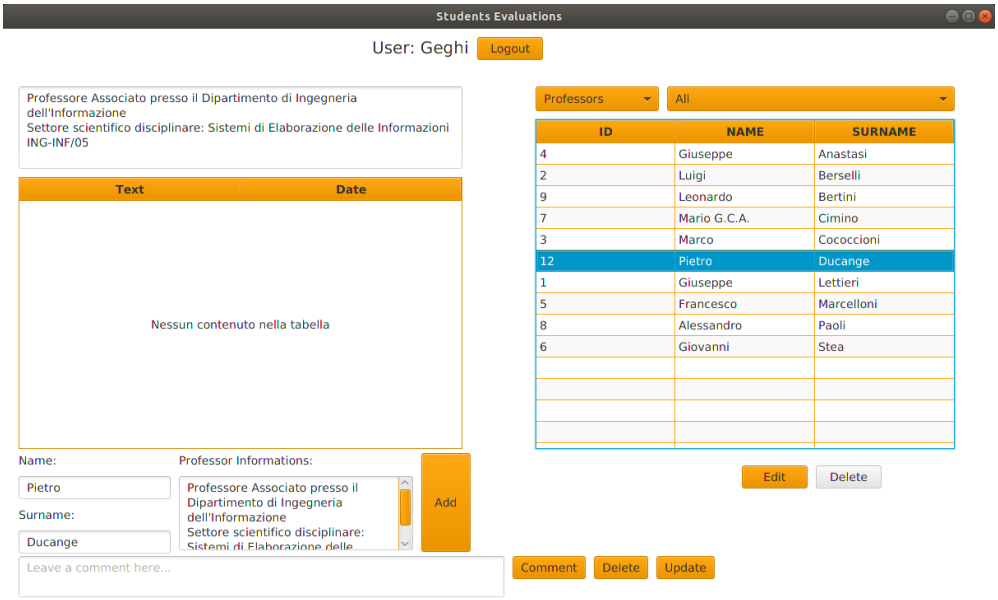

Figure 15: Screen of the application's interface from which you can either update or delete a professor

All these operations are available for the subjects as well. The only difference is that when you want to add a new subject you also need to specify the id of the professor teaching it (or a list of ids, separeted by commas, if there are more professors teaching it). Moreover, you must have precisely displayed in the table the subjects of the same degree course of the new one (fig. 16).

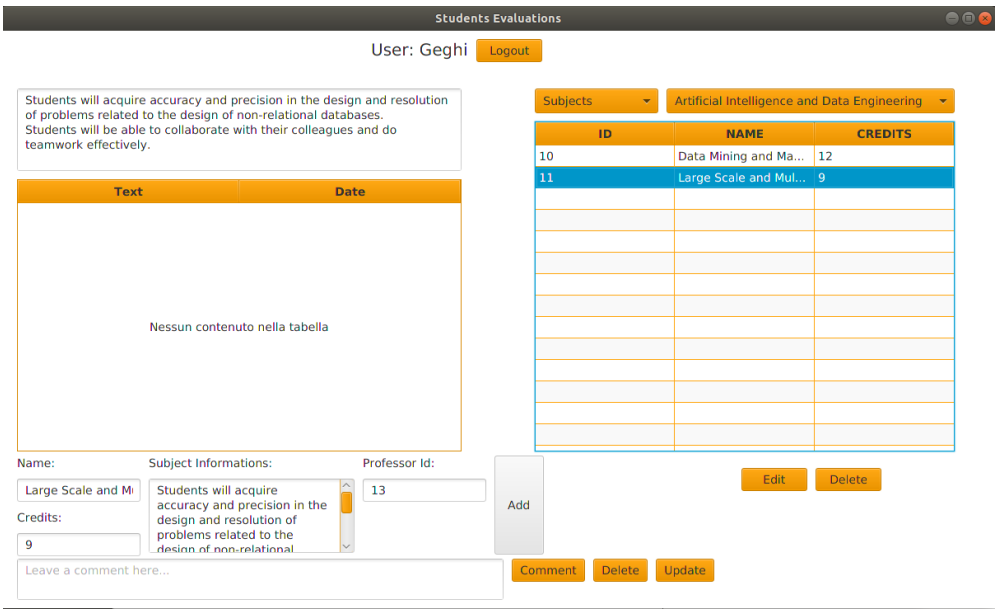

Figure 16: Interface after adding a new subject, ready to modify or delete it

The administrator can delete comments posted by all the users, too. Just click on the comment and then on the "Delete" button (fig. 17).

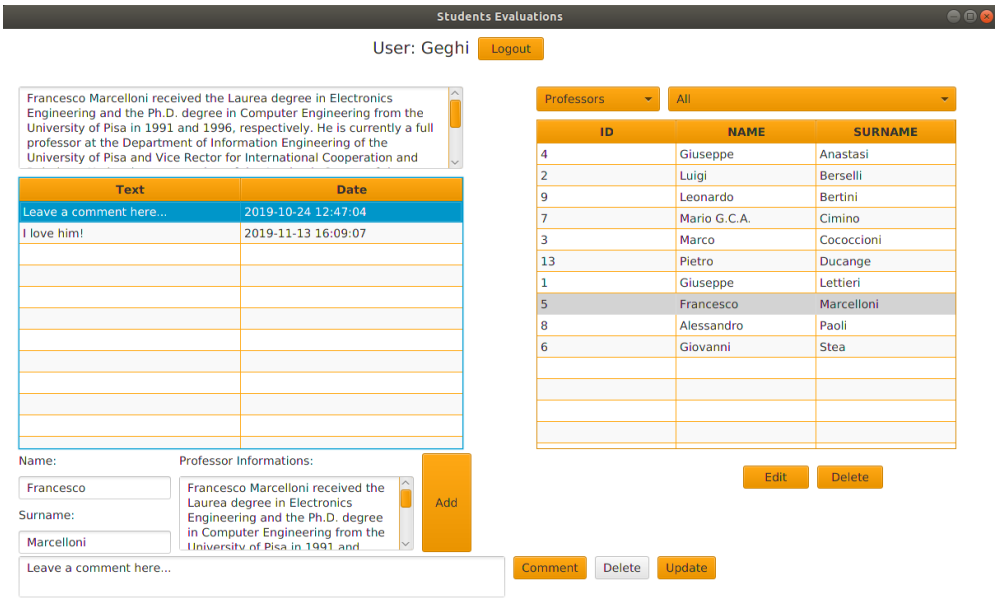

Figure 17: Screen of the application's interface from which the admin can delete a comment