University of Pisa

SCUOLA DI INGEGNERIA
Corso di Laurea in Artificial Intelligence and Data Engineering

Task1 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 and professors, called **Student-Evaluation**.
The application is developed to allow students (users) to view all the courses and professors of a specific university with related comments.
There are two buttons to switch the view of the list of professors to the list of courses: this action will update the table below. 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 professors and 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 professors, subjects and all the informations related.

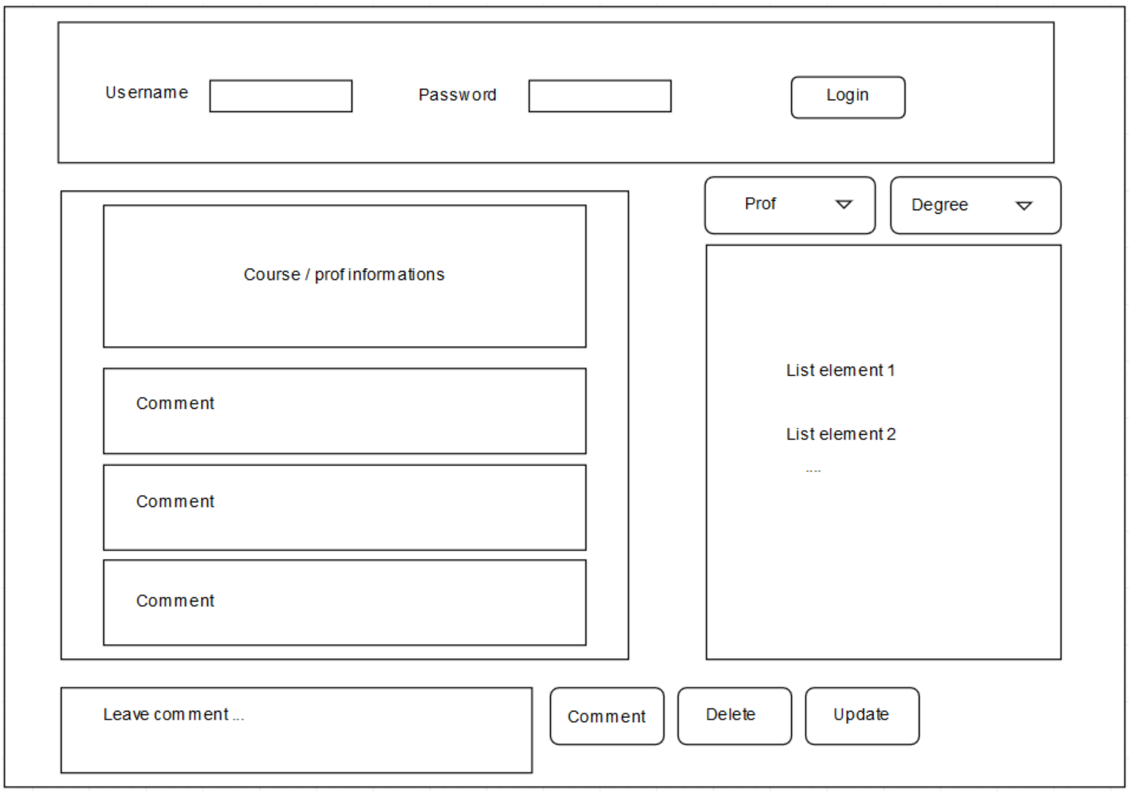


Figure 1: Mockup

2 Analysis and workflow

2.1 Requirements

2.1.1 Functional requirement

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

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

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

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

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

- To login into the system.
- To add a course/professor.
- To update a course/professor.
- To delete a course/professor.
- To associate to a course a professor.
- 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 case

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

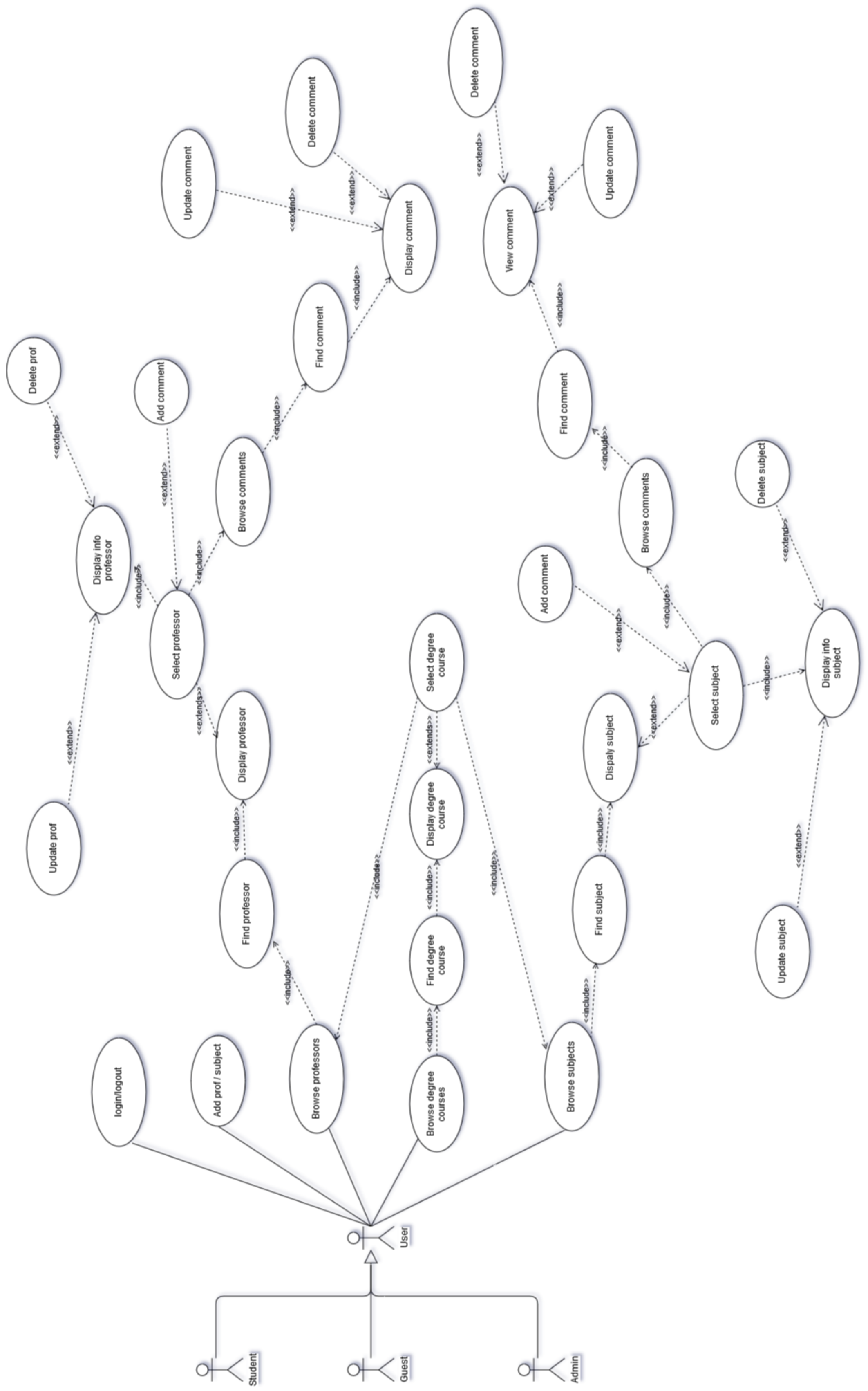


Figure 2: Use cases diagram

2.3 Class diagram

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

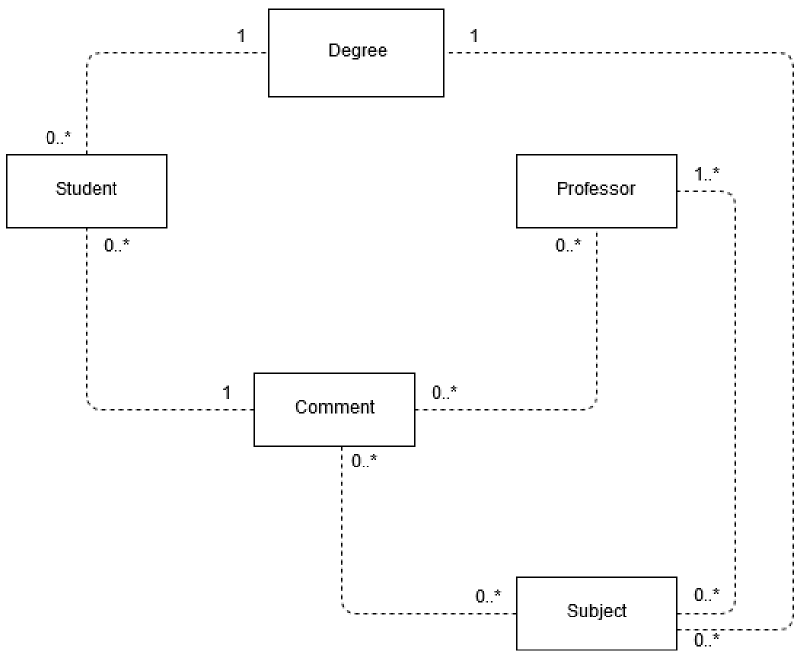


Figure 3: UML analysis diagram

3 Design

3.1 Software architecture

The application is designed over 3 different layers, see figure 4:

- Front-end
- Middleware
- Back-end

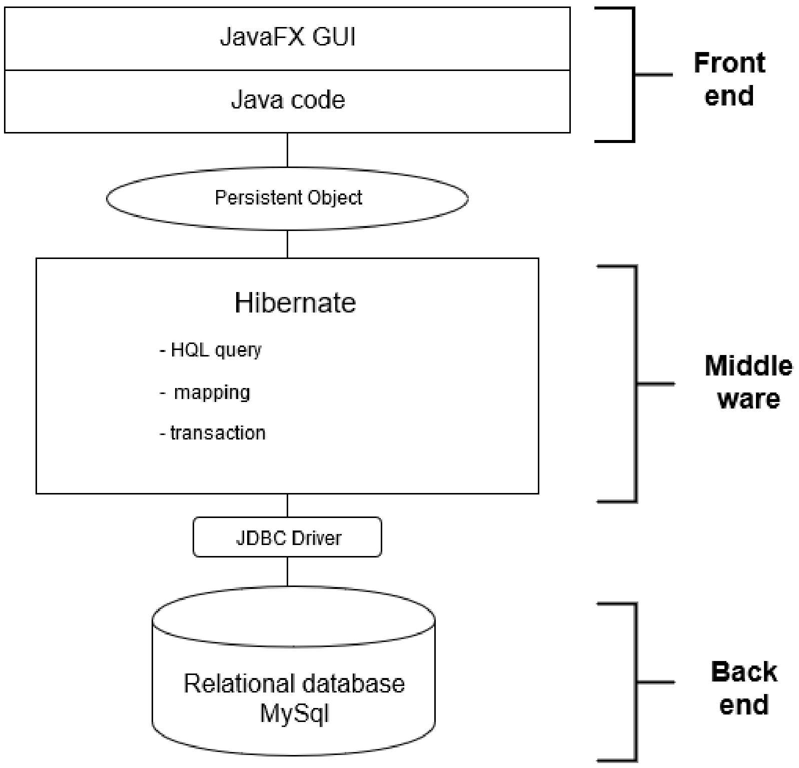


Figure 4: Software architecture diagram

3.2 Database design

The application is developped over a relational-database, using the MySql platform. The figure 5 shows the ER-diagram of the database.

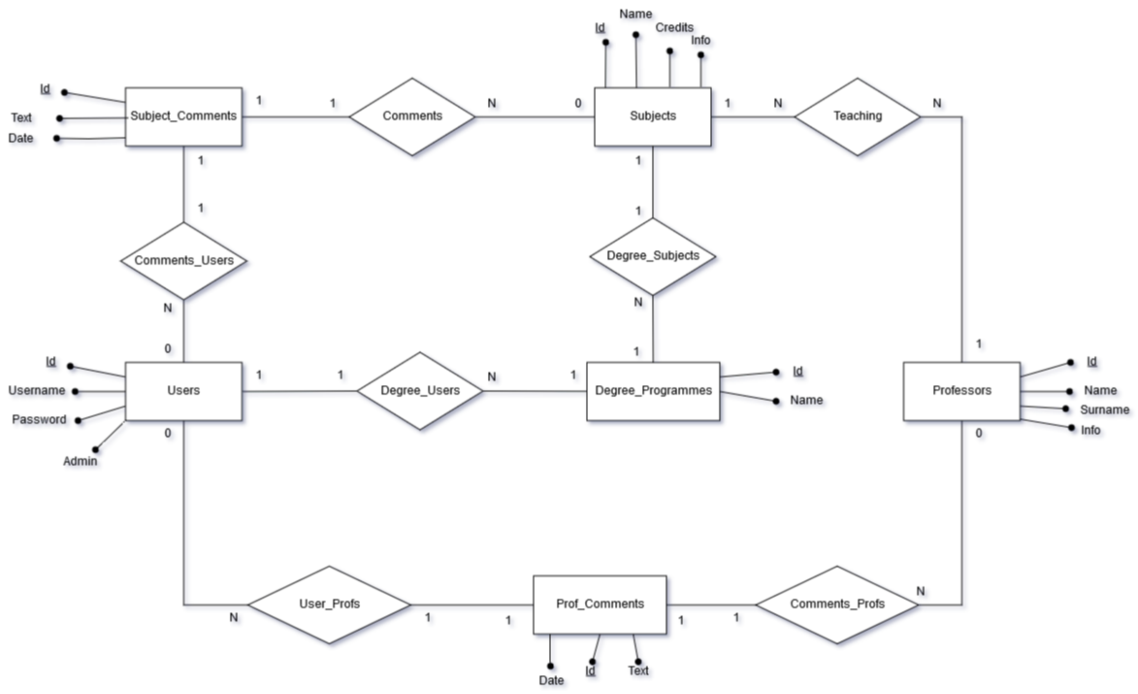


Figure 5: ER diagram

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 middleware layer is built thanks to Hibernate, version 5.4.4.0.
The jdbc driver manage the comunication between middleware layer and backend layer, version 8.0.17.
For the backend layer it is used Apache as web-server, version 2.4 (including Php, MySql).
So this application is tested using these technologies, considering these particular versions: for other versions the correct execution isn't guaranteed .

4.2 Snippets of code

This phase describes the most interesting parts about the creation and interaction with the database.
The following classes are mapped in the table of the database:

- Degree
- Student
- Professor
- Subject
- ProfessorComment
- ProfessotSubject

- Degree
- Student
- Professor
- Subject
- ProfessorComment
- ProfessotSubject

These classes are declared using the Hibernate annotations syntax.

4.2.1 Declaretion of class Subject

The following java-code shows a declaration of the class Subject.

```
@Entity(name = "Subjects")
@Table(name = "subjects")
public class Subject {

        @Column(name = "id")
        @Id
        @GeneratedValue(strategy = GenerationType.IDENTITY)
        private int id;
        private String name;
        private int credits;
        @Column(name = "info", columnDefinition="TEXT")
        private String info;

        // relation with degree
        @ManyToOne(fetch = FetchType.LAZY)
        @JoinColumn(name = "degreeId")
        private Degree deg;

        // relation with Subject comments.
        @OneToMany(mappedBy = "subj", cascade = CascadeType.ALL, orphanRemoval = true)
        private List<SubjectComment> subjectComments = new ArrayList<SubjectComment>();

        // relation with Professors.
        @ManyToMany
        @JoinTable(name = "teaching", joinColumns = @JoinColumn(name = "subjectId"),
                inverseJoinColumns = @JoinColumn(name = "profId"))
        private Set<Professor> professor = new HashSet<Professor>();
```

4.2.2 Declaretion of class Person

The following java-code shows a declaration of the class Person. The @MappedSuperclass annotation is used to map the superclass.

```
@MappedSuperclass
public abstract class Person {

        @Column(name = "id")
        @Id
        @GeneratedValue(strategy = GenerationType.IDENTITY)
        private int id;
    ...
}
```

4.2.3 Declaretion of class Professor

The following java-code shows a declaration of the class Professor.

```
@Entity(name = "Professors")
@Table(name = "professors")
public class Professor extends Person {

        @Column(name = "info", columnDefinition="TEXT")
        private String info;
        private String name;
        private String surname;

        // relation with profComments.
        @OneToMany(mappedBy = "prof", cascade = CascadeType.ALL, orphanRemoval = true,
                                                            fetch = FetchType.LAZY)
        private List<ProfessorComment> professorComments = new ArrayList<ProfessorComment>();

        // relation with Subjects.
        @ManyToMany(mappedBy = "professor", cascade = CascadeType.ALL)
        private Set<Subject> subject = new HashSet<Subject>();
    ...
}
```

4.2.4 Declaretion of class Student

The following java-code shows a declaration of the class Student.

```
@Entity(name = "Users")
@Table(name = "users")
public class Student extends Person {

        private final boolean admin;
        @Column(name = "username", unique = true)
        private String username;
        private String password;

        // relation with Degree.
        @ManyToOne(fetch = FetchType.LAZY)
        @JoinColumn(name = "degreeId")
        private Degree deg;

        // relation with SubjectComments.
        @OneToMany(mappedBy = "stud", cascade = CascadeType.ALL, orphanRemoval = true)
        private List<SubjectComment> subjectComments = new ArrayList<SubjectComment>();

        // relation with ProfessorComments.
        @OneToMany(mappedBy = "stud", cascade = CascadeType.ALL, orphanRemoval = true)
        private List<ProfessorComment> professorComments = new ArrayList<ProfessorComment>();
    ...
}
```

4.2.5 Declaretion of class Comment

The following java-code shows a declaration of the class Comment. The @MappedSuperclass annotation is used to map the superclass.

```
@MappedSuperclass
public abstract class Comment {

        @Column(name = "id")
        @Id
        @GeneratedValue(strategy = GenerationType.IDENTITY)
        private int id;
        private String text;
        private String date;

        private static String format = "yyyy-MM-dd HH:mm:ss";
    ...
}
```

4.2.6 Declaretion of class SubjectComment

The following java-code shows a declaration of the class SubjectComment.

```
@Entity(name = "SubjectComments")
@Table(name = "subject_comments")
public class SubjectComment extends Comment {

        @ManyToOne(fetch = FetchType.LAZY)
        @JoinColumn(name = "userId")
        private Student stud;

        @ManyToOne(fetch = FetchType.LAZY)
        @JoinColumn(name = "subjectId")
        private Subject subj;
    ...
}
```

4.2.7 Declaretion of class ProfessorComment

The following java-code shows a declaration of the class ProfessorComment.

```
@Entity(name = "ProfComments")
@Table(name = "prof_comments")
public class ProfessorComment extends Comment {

        @ManyToOne(fetch = FetchType.LAZY)
        @JoinColumn(name = "profId")
        private Professor prof;

        @ManyToOne(fetch = FetchType.LAZY)
        @JoinColumn(name = "userId")
        private Student stud;
   ...
}
```

4.2.8 Declaretion of class Degree

The following java-code shows a declaration of the class Degree.

```
@Entity(name = "Degree")
@Table(name = "degree_programmes")
public class Degree {

        @Column(name = "id")
        @Id
        @GeneratedValue(strategy = GenerationType.IDENTITY)
        private int id;

        @Column(name = "name", unique = true)
        private String name;

        // relation with students.
        @OneToMany(mappedBy = "deg", cascade = CascadeType.ALL, orphanRemoval = true)
        private List<Student> student = new ArrayList<Student>();

        // relation with subjects.
        @OneToMany(mappedBy = "deg", cascade = CascadeType.ALL, orphanRemoval = true)
        private List<Subject> subject = new ArrayList<Subject>();
   ...
}
```

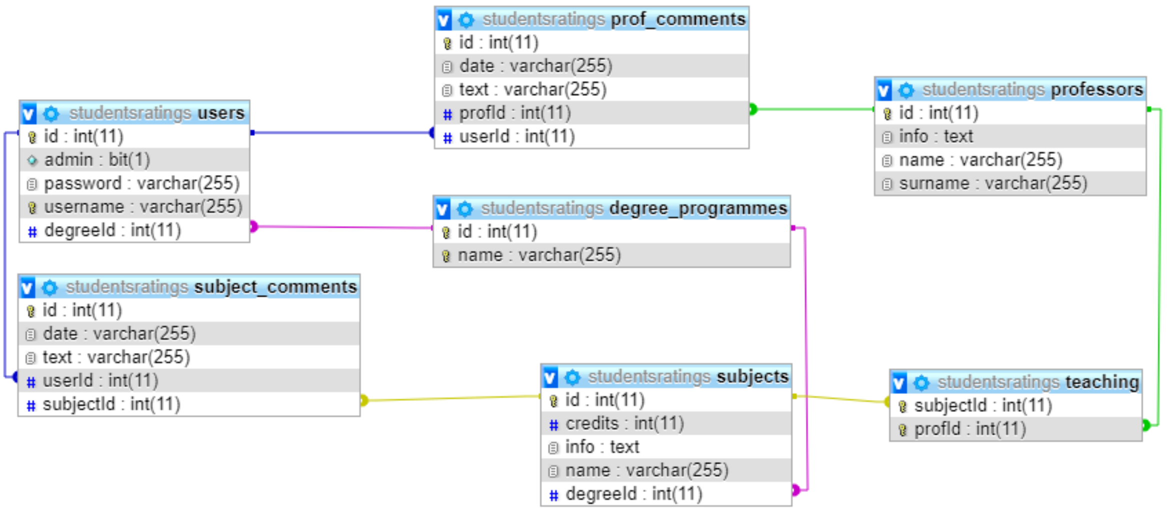


Figure 6: ER diagram

Another foundamental class is ManagerEM, that manages the db-connection and the related operations. The implementation of a set of CRUD operations, regarding the class Subject, is desribed as follows.

4.3 Create

Creating a new subject specifying general informations, the professor holding the course and the associated degree program as parameters.

```
public Subject createSubject(String name, int credits, String info,
                                    String profIdStr, int degreeId){
        System.out.println("Creating␣a␣new␣subject");

        Subject subject = new Subject(0, name, credits, info);
        String[] professorsId = profIdStr.split(",", 5);
        try {
                entityManager = factory.createEntityManager();
                Degree degree = entityManager.find(Degree.class, degreeId);
                int profId = 0;
                for (String p : professorsId) {
                        profId = Integer.parseInt(p);
                        Professor professor = null;
                        if (profId > 0) {
                                professor = entityManager.find(Professor.class, profId);
                        }
                        if (profId > 0 && professor == null) {
                                System.err.println("the␣inserted␣prof␣Id␣doesn't␣exixst");
                                entityManager.close();
                                return null;
                        }
                        subject.getProfessor().add(professor);
                        professor.getSubject().add(subject);
                }
                degree.getSubject().add(subject);
                subject.setDeg(degree);

                entityManager.getTransaction().begin();
                entityManager.persist(subject);
                entityManager.getTransaction().commit();
                System.out.println("subject␣Added");
                return subject;
        } catch (Exception ex) {
                ex.printStackTrace();
                System.err.println("A problem occurred in updating a subject!");
        } finally {
                entityManager.close();
        }
        return subject;

}
```

4.4 Read

This functionality returns a list of subjects interrogating the database using a query written in Hibernate Query Language (HQL).

```
public List<Subject> getSubjects(int degree) {

        List<Subject> results = new ArrayList<>();
        System.out.println("Getting a List of subjects based on the degree");

        String selectionSubjects = "SELECT s FROM Subjects s ORDER BY s.name";
        String selectionSubjectByDegree =
                         "SELECT s FROM Subjects s WHERE degreeId = ?1 ORDER BY s.name";
        try {
                entityManager = factory.createEntityManager();
                if (degree < 0) {
                        TypedQuery<Subject> query = entityManager.createQuery
                                (selectionSubjects, Subject.class);
                        results = query.getResultList();
                } else {

                        TypedQuery<Subject> query = entityManager.createQuery
                                (selectionSubjectByDegree, Subject.class);
                        query.setParameter(1, degree);
                        results = query.getResultList();
                }

        } catch (Exception ex) {
                ex.printStackTrace();
                System.err.println("A problem occurred in retriving subjects!");

        } finally {
                entityManager.close();
        }
        return results;
}
```

4.5 Update

This operation allows students to update their comments about a selected subject.

```
public boolean updateCommentSubject(int subjectCommentId, String text, int userId) {
        System.out.println("Updating a subject comment");
        boolean updated = false;
        Date date = new Date();

        try {
                entityManager = factory.createEntityManager();
                SubjectComment subjectComment = entityManager.find
                                        (SubjectComment.class, subjectCommentId);
```

```
            // if the user is the owner.
            if (subjectComment.getStud().getId() == userId) {
                    entityManager.getTransaction().begin();
                    subjectComment.setText(text);
                    subjectComment.setDate(date);
                    entityManager.getTransaction().commit();
                    System.out.println("subject␣comment␣updated");
                    updated = true;

            } else { // if user is not owner --> error
                    System.err.println("You␣are␣not␣the␣owner␣of␣that␣comment,
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣please␣select␣another␣comment");
                    entityManager.close();
                    return updated;
            }
    } catch (Exception ex) {
            ex.printStackTrace();
            System.err.println("A␣problem␣occurred␣in␣updating␣a␣subject␣comment!");

    } finally {
            entityManager.close();
    }
    return updated;
}
```

4.6 Delete

This operation allows a student to delete their comments about a selected subject.

```
public boolean deleteCommentSubject(int subjectCommentId, int userId, boolean admin) {
        boolean deleted = false;
        try {
                entityManager = factory.createEntityManager();
                entityManager.getTransaction().begin();
                SubjectComment subjectComment = entityManager.find(SubjectComment.class,
                                                        subjectCommentId);

                // if user is owner OR admin he can delete the comment
                if (subjectComment.getStud().getId() == userId || admin) {
                        entityManager.remove(subjectComment);
                        deleted = true;
                } else { // if user is not owner AND he is not admin --> error
                        System.err.println("You␣are␣not␣the␣owner␣of␣that␣comment,
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣please␣select␣another␣comment");
                        return deleted;
                }

                entityManager.getTransaction().commit();
                System.out.println("subject␣comment␣removed");

        } catch (Exception ex) {
                ex.printStackTrace();
                System.err.println("A␣problem␣occurred␣in␣removing␣a␣subject␣comment!");

        } finally {
                entityManager.close();
        }
        return deleted;
}
```

4.7 GUI

Moreover there are 3 classes to manage the graphic user interface:

- GraphicInterface
- CommentTable
- ProfSubjectTable

These classes use the javaFX framework to build the GUI and handle the related events.

5 Level DB

Key-values databases are one of the simplest examples of NoSQL stores that can be found. In a key-value database, data are persistently stored assigning a value to a uniqle identifier, which takes the name of key; couples of key-values are stored in namespaces (buckets) and in each buckets, keys must be unique.
The main strenghts of the key-value database are:

- Simplicity
- Speed
- Scalability

Key-value database are mostly used when speed in retrieving data and ease of storage are more important than the organization of data into complex structures. So, in general, classics RDBMS are preferred when organization and management is more important than perfomances; typically, when a database presents a lot of relations among entities, key-values stores aren't the best choice. When the structure of our database is simple, instead, we could use the key-value stores because they are more capable of providing higher performances than RDBMS.
It is also true that simple relations between entities are represented in key-value database as well, just by building up keys in a predefined way.
The new architecture is shown in figure 7:

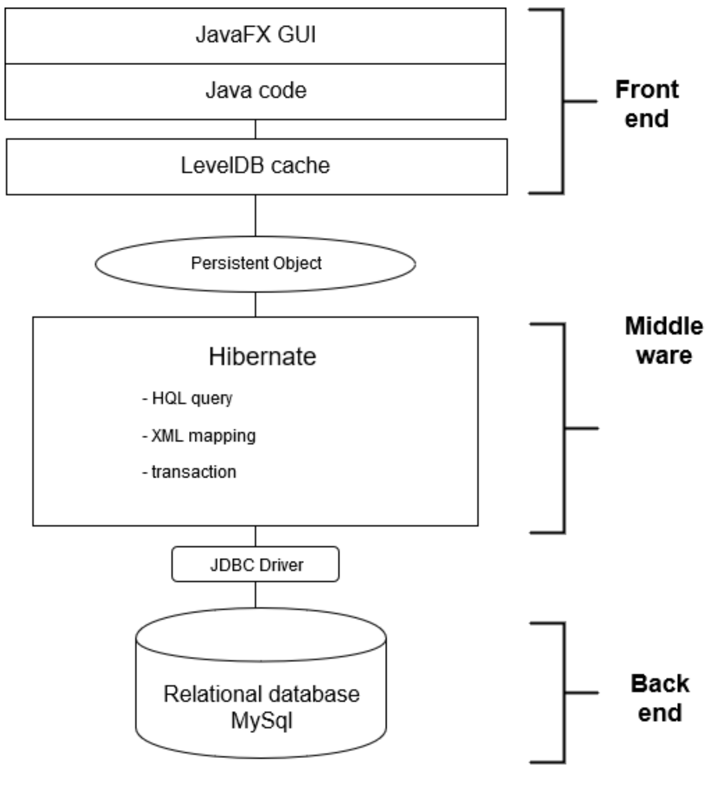


Figure 7: Software architecture diagram

LevelDB is a tool used to create a key-value database. It stores data under the form of key-value pairs locally in .sst file. Thanks to the property of those kind of databases we can think to set up a cache memory for our application using LevelDB.

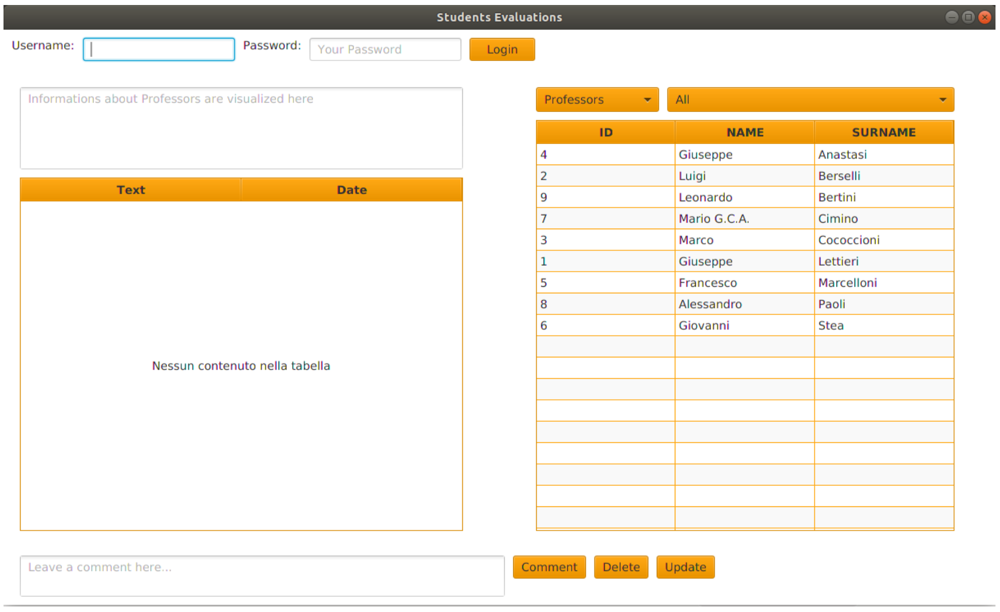
We need to store information about professors/courses, with their relative comments, belonging to the same degree of the student that is actually logged. The idea is that the student will check those information more frequently than in the case of professors or courses of different degree.
When a student logs in, we retrieve and store the sequent data: professors (with their relative comments) and courses (with relative comments) of the same degree of the student. Those informations are then used to populate the tables needed during the normal operations of the application.
We create the key for that data in the following way:

- For the professors' data:
  professors:$professor_id:$attribute_name
  where the attributes are: name, surname and info.
- For the subjects' data:
  subjects:$subject_id:$attribute_name
  where the attributes are: name, credits and info.
- For the professor's comments data:
  prof_comments:$prof_comments_id:$user_id:$professor_id:$attribute_name
  where the attributes are: text and date.
- For the subject's comments data:
  subject_comments:$subject_comment_id:$user_id:$subject_id:$attribute_name
  where the attributes are: text and date.

Every time a users choose to see the informations about professors or subjects belonging to his same degree, we display the data stored locally. Once the user select a professor or a subject, the comment for that professor or for that subject are retrieved from the local file.
During the normal operation of the application, a student can add a new comment or update or modify one of his previously submitted comments.
When a user add a comment on one of the target objects (professors or subjects of his same degree), the new comment is first added onthe local file then the adding is propagated on the database. When a user modify or delete a comment, the action is only operated on the local file, when the user logs out those operations are then propagated on the relational database.
The admin, which is a super-user whith higher privileges, has to skip those step; in fact, the admin's operations skip the local file and affect first the relational database, and then the key-value file, without the asynchronous steps.

6 User Manual

When you first run the application, the interface you get is the one in figure 8.



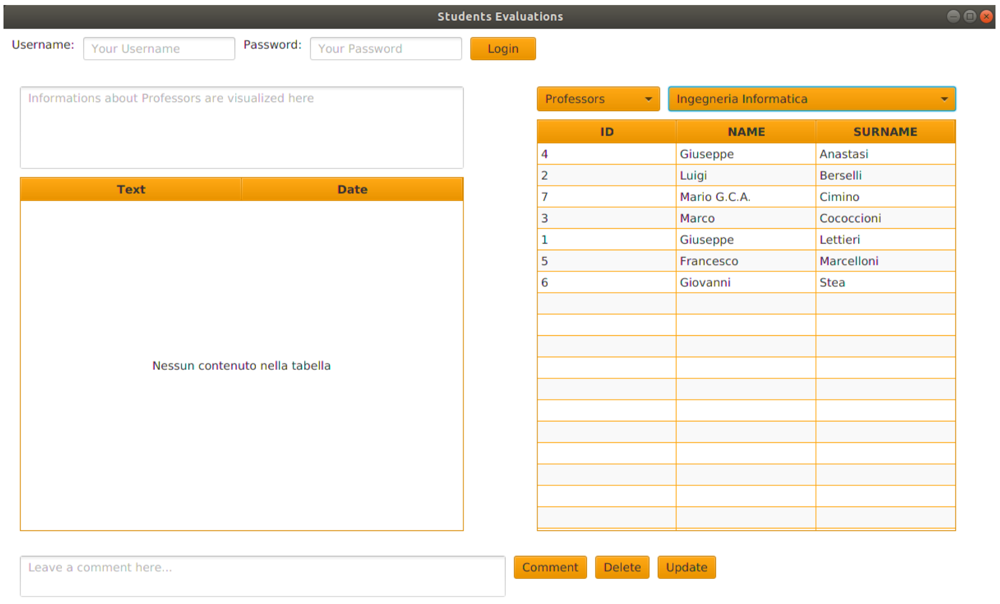
Text	Date

Nessun contenuto nella tabella

ID	NAME	SURNAME
4	Giuseppe	Anastasi
2	Luigi	Berselli
9	Leonardo	Bertini
7	Mario G.C.A.	Cimino
3	Marco	Cococcioni
1	Giuseppe	Lettieri
5	Francesco	Marcelloni
8	Alessandro	Paoli
6	Giovanni	Stea

Figure 8: 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. 9), or decide to view the list of subjects (fig. 10), for which is also available the degree course's filter.

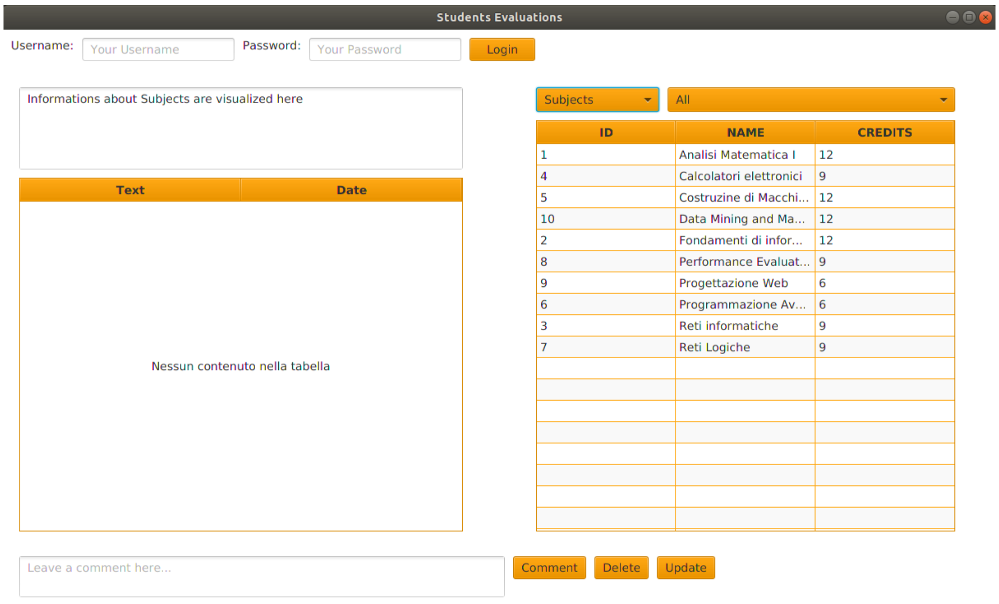


Text	Date

Nessun contenuto nella tabella

ID	NAME	SURNAME
4	Giuseppe	Anastasi
2	Luigi	Berselli
7	Mario G.C.A.	Cimino
3	Marco	Cococcioni
1	Giuseppe	Lettieri
5	Francesco	Marcelloni
6	Giovanni	Stea

Figure 9: Selection of professors filtered by "Ingegneria Informatica" degree course


Students Evaluations

Username: Your Username Password: Your Password Login

Informations about Subjects are visualized here

Text	Date

Nessun contenuto nella tabella

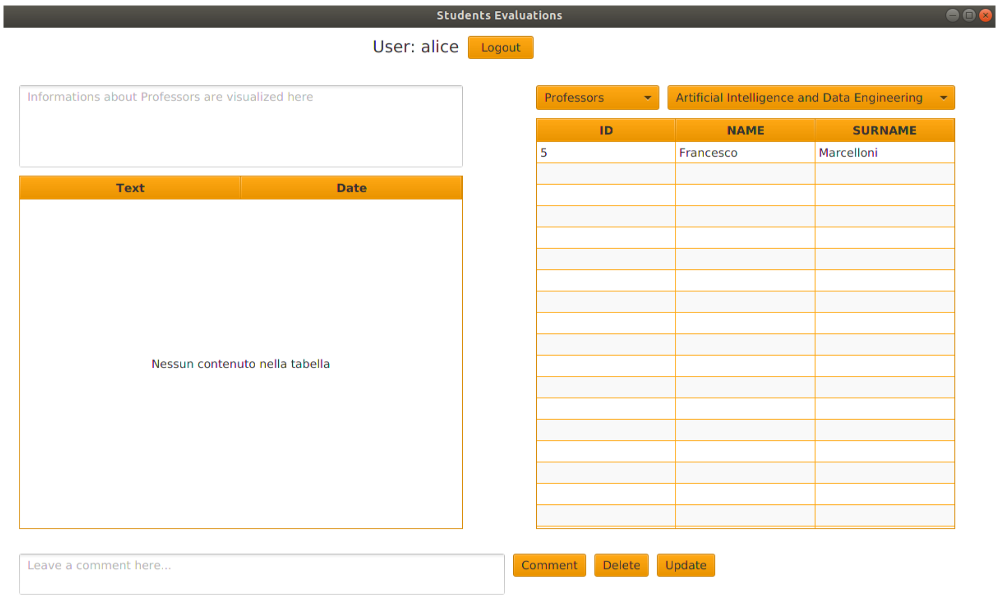
Subjects All

ID	NAME	CREDITS
1	Analisi Matematica I	12
4	Calcolatori elettronici	9
5	Costruzine di Macchi...	12
10	Data Mining and Ma...	12
2	Fondamenti di infor...	12
8	Performance Evaluat...	9
9	Progettazione Web	6
6	Programmazione Av...	6
3	Reti informatiche	9
7	Reti Logiche	9

Leave a comment here... Comment Delete Update

Figure 10: 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. 11).


Students Evaluations

User: alice Logout

Informations about Professors are visualized here

Text	Date

Nessun contenuto nella tabella

Professors Artificial Intelligence and Data Engineering

ID	NAME	SURNAME
5	Francesco	Marcelloni

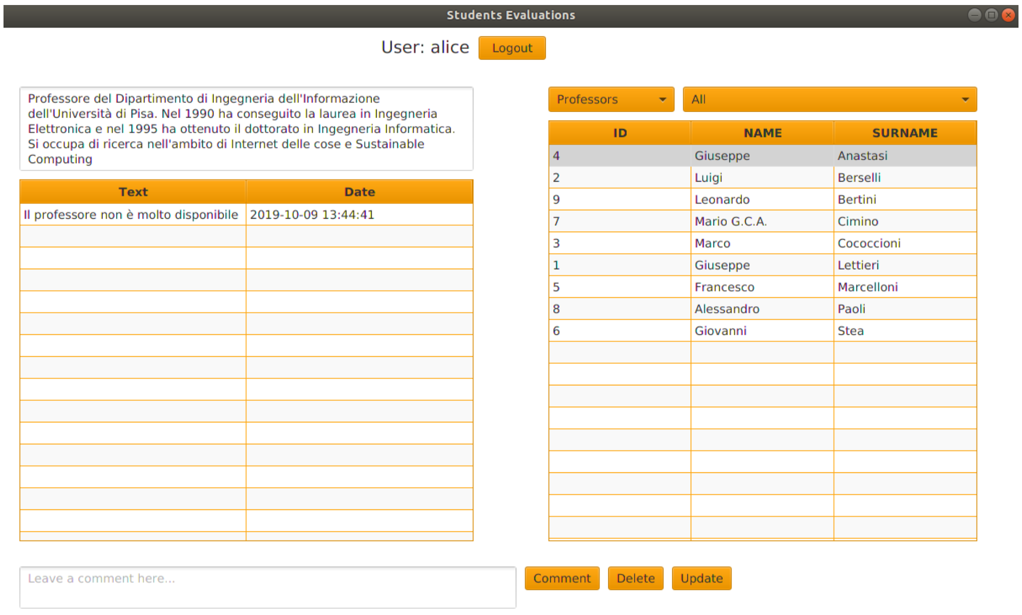
Leave a comment here... Comment Delete Update

Figure 11: 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. 12). 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. 13.

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



Students Evaluations

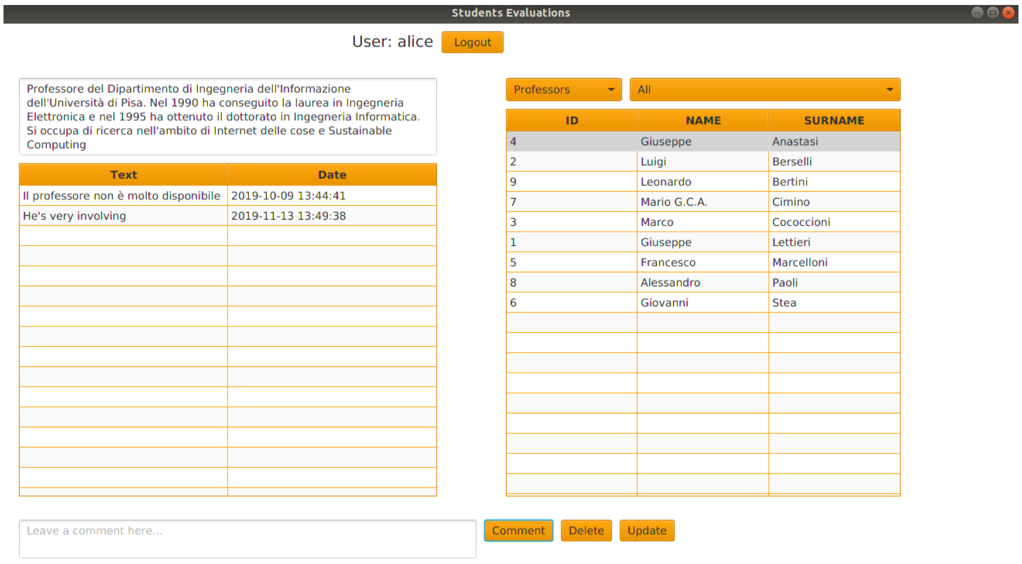
User: alice

Professore del Dipartimento di Ingegneria dell'Informazione dell'Università di Pisa. Nel 1990 ha conseguito la laurea in Ingegneria Elettronica e nel 1995 ha ottenuto il dottorato in Ingegneria Informatica. Si occupa di ricerca nell'ambito di Internet delle cose e Sustainable Computing

Text	Date
Il professore non è molto disponibile	2019-10-09 13:44:41

ID	NAME	SURNAME
4	Giuseppe	Anastasi
2	Luigi	Berselli
9	Leonardo	Bertini
7	Mario G.C.A.	Cimino
3	Marco	Cococcioni
1	Giuseppe	Lettieri
5	Francesco	Marcelloni
8	Alessandro	Paoli
6	Giovanni	Stea

Figure 12: Displaying the comments related to a professor



Students Evaluations

User: alice

Professore del Dipartimento di Ingegneria dell'Informazione dell'Università di Pisa. Nel 1990 ha conseguito la laurea in Ingegneria Elettronica e nel 1995 ha ottenuto il dottorato in Ingegneria Informatica. Si occupa di ricerca nell'ambito di Internet delle cose e Sustainable Computing

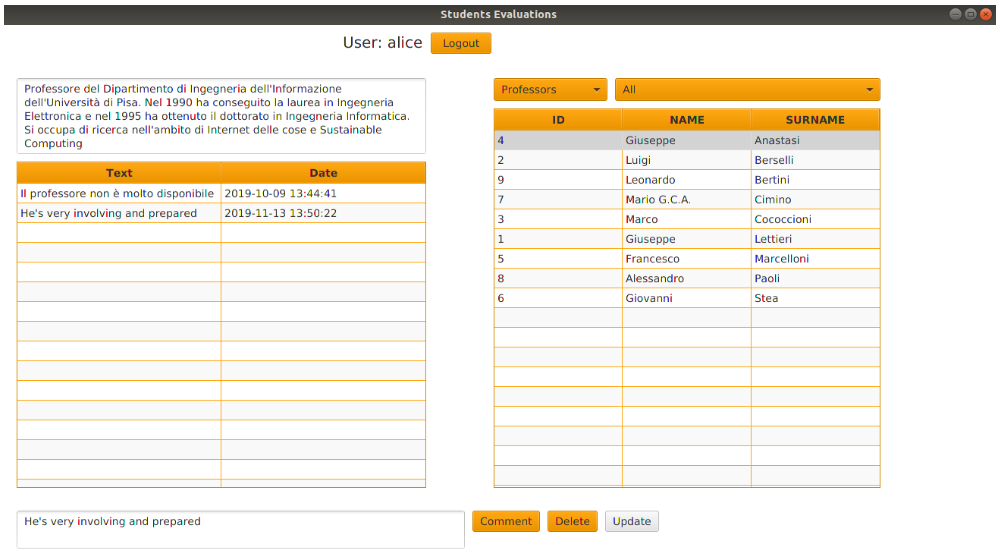
Text	Date
Il professore non è molto disponibile	2019-10-09 13:44:41
He's very involving	2019-11-13 13:49:38

ID	NAME	SURNAME
4	Giuseppe	Anastasi
2	Luigi	Berselli
9	Leonardo	Bertini
7	Mario G.C.A.	Cimino
3	Marco	Cococcioni
1	Giuseppe	Lettieri
5	Francesco	Marcelloni
8	Alessandro	Paoli
6	Giovanni	Stea

Figure 13: Interface after adding a comment

the text in the field below the table and then click on the "Update" button (fig. 14). 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.



Professore del Dipartimento di Ingegneria dell'Informazione dell'Università di Pisa. Nel 1990 ha conseguito la laurea in Ingegneria Elettronica e nel 1995 ha ottenuto il dottorato in Ingegneria Informatica. Si occupa di ricerca nell'ambito di Internet delle cose e Sustainable Computing

Text	Date
Il professore non è molto disponibile	2019-10-09 13:44:41
He's very involving and prepared	2019-11-13 13:50:22

ID	NAME	SURNAME
4	Giuseppe	Anastasi
2	Luigi	Berselli
9	Leonardo	Bertini
7	Mario G.C.A.	Cimino
3	Marco	Cococcioni
1	Giuseppe	Lettieri
5	Francesco	Marcelloni
8	Alessandro	Paoli
6	Giovanni	Stea

He's very involving and prepared

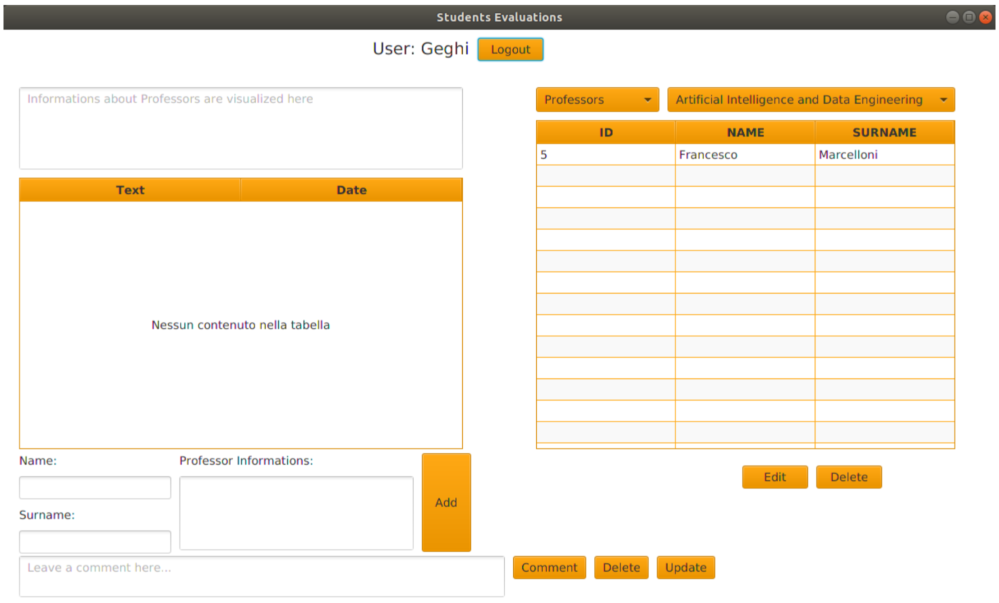
Figure 14: Interface after updating a comment

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.

6.1 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. 15).

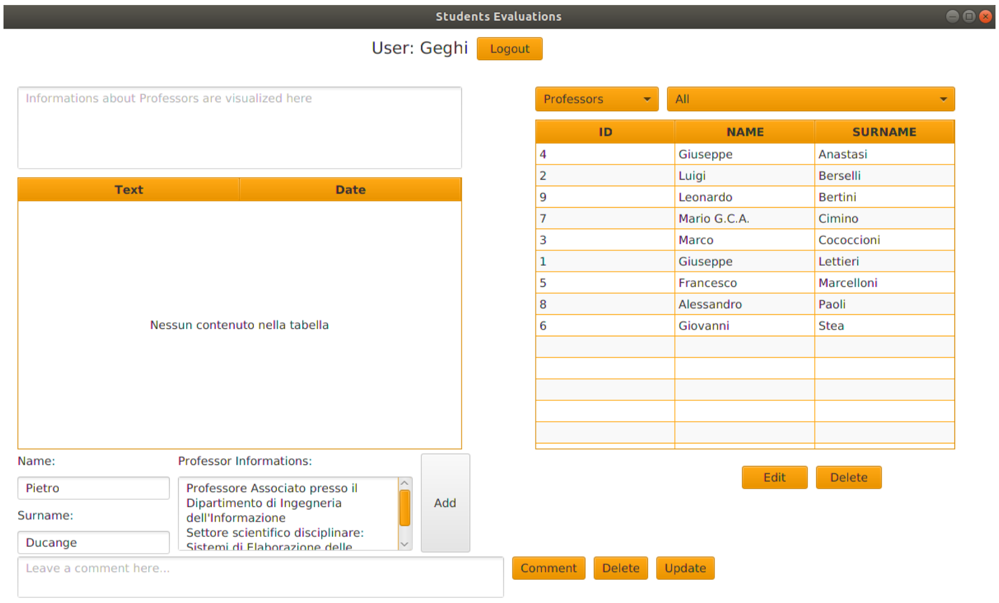


Text	Date

ID	NAME	SURNAME
5	Francesco	Marcelloni

Figure 15: 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. 16).



Text	Date

ID	NAME	SURNAME
4	Giuseppe	Anastasi
2	Luigi	Berselli
9	Leonardo	Bertini
7	Mario G.C.A.	Cimino
3	Marco	Cococcioni
1	Giuseppe	Lettieri
5	Francesco	Marcelloni
8	Alessandro	Paoli
6	Giovanni	Stea

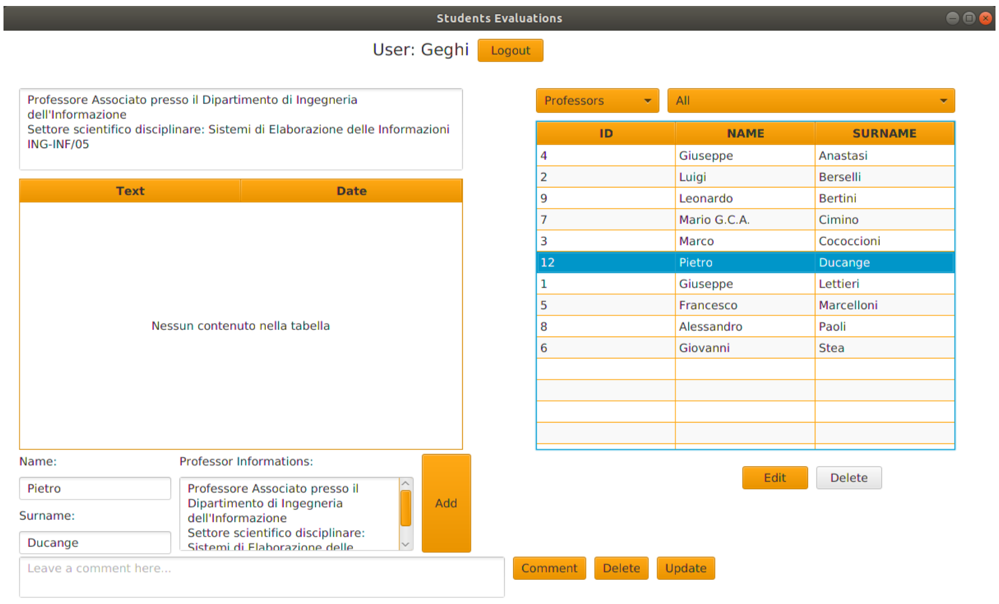
Name: Pietro

Surname: Ducange

Professor Informations: Professore Associato presso il Dipartimento di Ingegneria dell'Informazione Settore scientifico disciplinare: Sistemi di Elaborazione delle

Figure 16: 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. 17).



Professore Associato presso il Dipartimento di Ingegneria dell'Informazione
Settore scientifico disciplinare: Sistemi di Elaborazione delle Informazioni ING-INF/05

Text	Date

Nessun contenuto nella tabella

ID	NAME	SURNAME
4	Giuseppe	Anastasi
2	Luigi	Berselli
9	Leonardo	Bertini
7	Mario G.C.A.	Cimino
3	Marco	Cococcioni
12	Pietro	Ducange
1	Giuseppe	Lettieri
5	Francesco	Marcelloni
8	Alessandro	Paoli
6	Giovanni	Stea

Name: Pietro

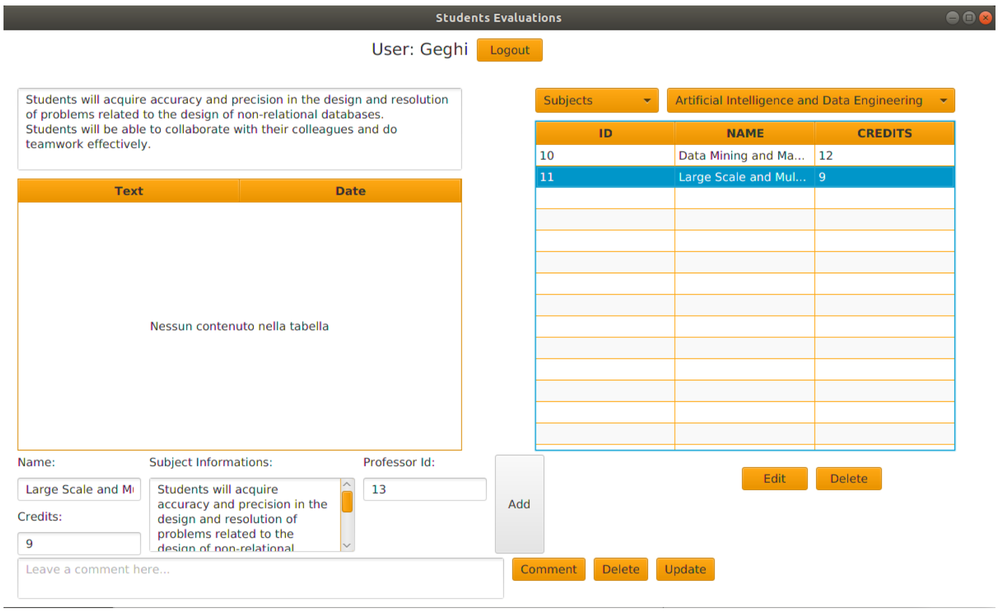
Surname: Ducange

Professor Informations: Professore Associato presso il Dipartimento di Ingegneria dell'Informazione Settore scientifico disciplinare: Sistemi di Elaborazione delle

Leave a comment here...

Figure 17: 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. 18).



Students will acquire accuracy and precision in the design and resolution of problems related to the design of non-relational databases.
Students will be able to collaborate with their colleagues and do teamwork effectively.

Text	Date

Nessun contenuto nella tabella

ID	NAME	CREDITS
10	Data Mining and Ma...	12
11	Large Scale and Mul...	9

Name: Large Scale and M

Credits: 9

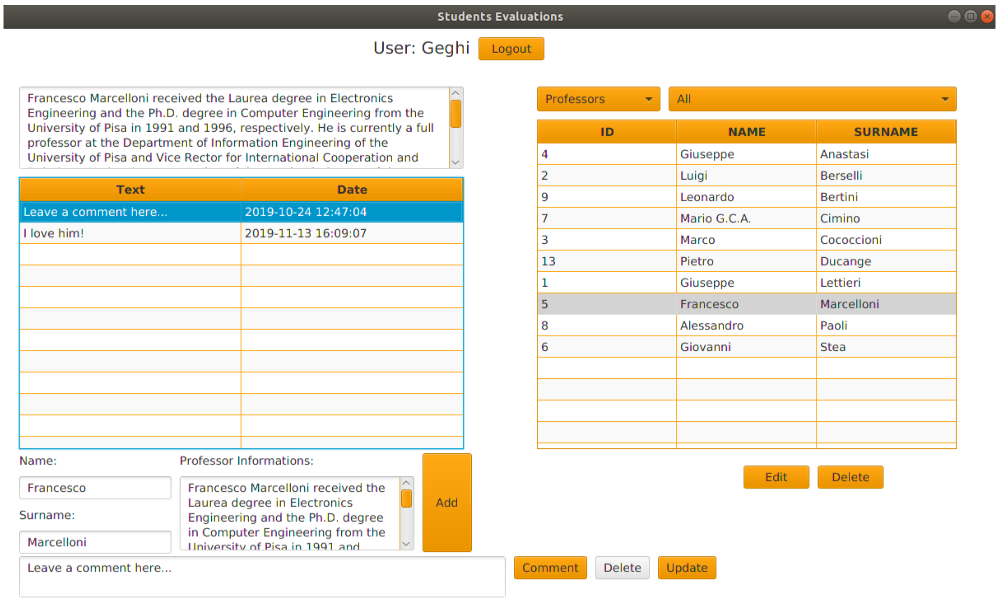
Subject Informations: Students will acquire accuracy and precision in the design and resolution of problems related to the design of non-relational

Professor Id: 13

Leave a comment here...

Figure 18: 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. 19).



User: Geghi

Francesco Marcelloni received the Laurea degree in Electronics Engineering and the Ph.D. degree in Computer Engineering from the University of Pisa in 1991 and 1996, respectively. He is currently a full professor at the Department of Information Engineering of the University of Pisa and Vice Rector for International Cooperation and

Text	Date
Leave a comment here...	2019-10-24 12:47:04
I love him!	2019-11-13 16:09:07

ID	NAME	SURNAME
4	Giuseppe	Anastasi
2	Luigi	Berselli
9	Leonardo	Bertini
7	Mario G.C.A.	Cimino
3	Marco	Cococcioni
13	Pietro	Ducange
1	Giuseppe	Lettieri
5	Francesco	Marcelloni
8	Alessandro	Paoli
6	Giovanni	Stea

Name: Francesco

Surname: Marcelloni

Professor Informations: Francesco Marcelloni received the Laurea degree in Electronics Engineering and the Ph.D. degree in Computer Engineering from the University of Pisa in 1991 and

Leave a comment here...

Figure 19: Screen of the application's interface from which the admin can delete a comment