



Task1 documentation

Candidati

Alice Nannini

Giacomo Mantovani

Marco Parola

Stefano Poleggi

Relatore

Prof. Pietro Ducange

Contents

1	Introduction	1
2	Requirements	2
2.1	Functional requirement	2
2.2	Non-functional requirements	2
3	Use case	3
3.1	Use Cases Description	3
4	Class diagram	5
5	Design	6
5.1	Software architecture	6
5.2	Database design	7
6	Implementation	8
6.1	Used technologies	8
6.2	Snippets of code	8
6.3	Create	9
6.4	Read	10
6.5	Update	10
6.6	Delete	11
6.7	GUI	11
7	Level DB	12
8	User Manual	15
8.1	Admin Manual	19

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.

The mockup shows a web application interface with the following components:

- Login Section:** A horizontal box at the top containing "Username" and "Password" labels, each followed by a text input field, and a "Login" button to the right.
- Filtering Section:** Two dropdown menus labeled "Prof" and "Degree" are positioned above a large rectangular area for the list of elements.
- Left Panel:** A vertical stack of four boxes. The top box is labeled "Course / prof informations". The following three boxes are each labeled "Comment".
- Right Panel:** A large rectangular area containing the text "List element 1", "List element 2", and "...".
- Bottom Section:** A horizontal box at the bottom containing a text input field labeled "Leave comment ..." and three buttons labeled "Comment", "Delete", and "Update".

Figure 1: Mockup

2 Requirements

2.1 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 addition 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.

2.2 Non-functional requirements

- Usability, ease of use and intuitiveness of the application by the user.
- Availability, with the service guaranteed 24/7.
- The system should support simultaneous users.
- The system should provide access to the database with a few seconds of latency.

3 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

3.1 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.

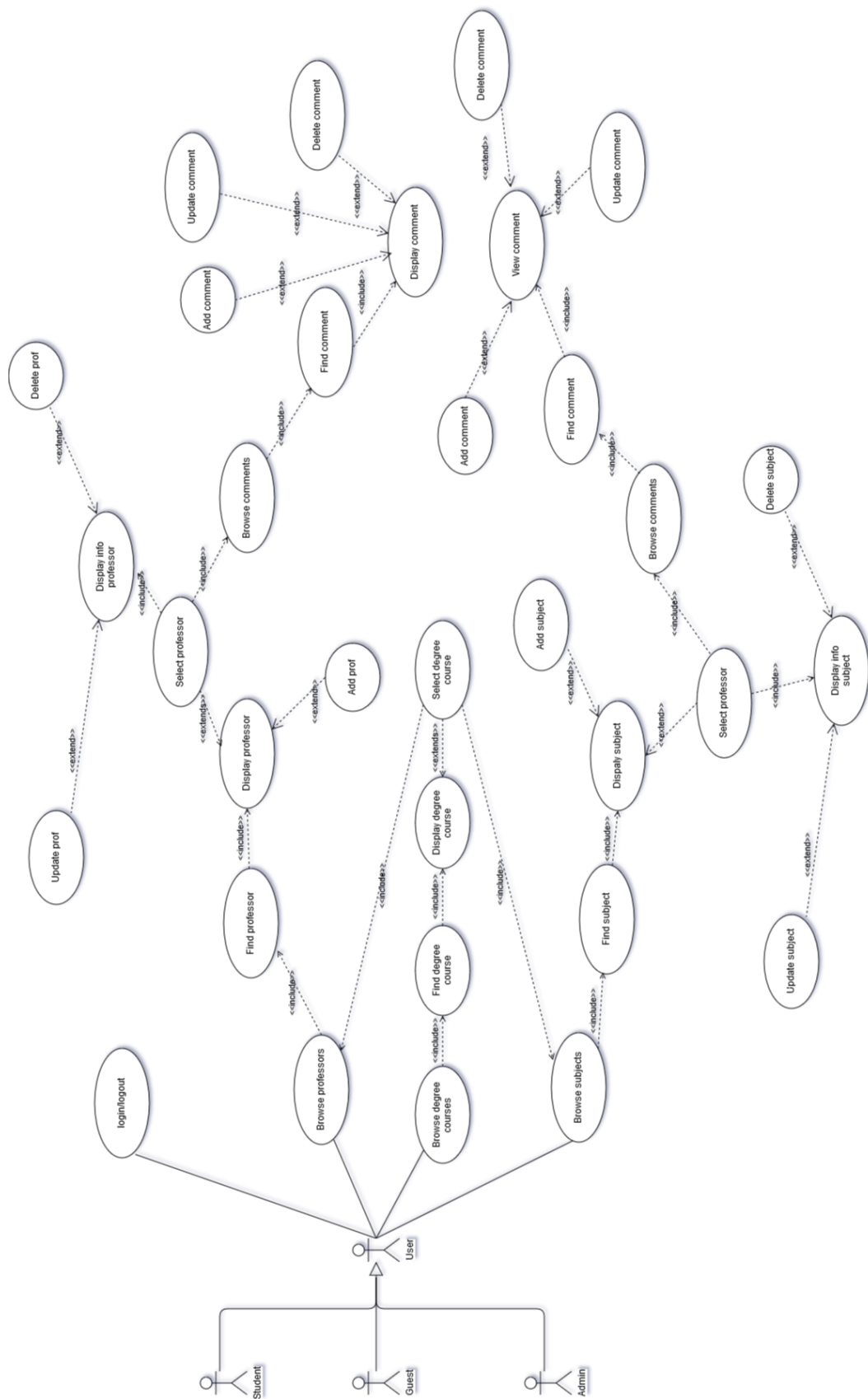


Figure 2: Use cases diagram

4 Class diagram

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

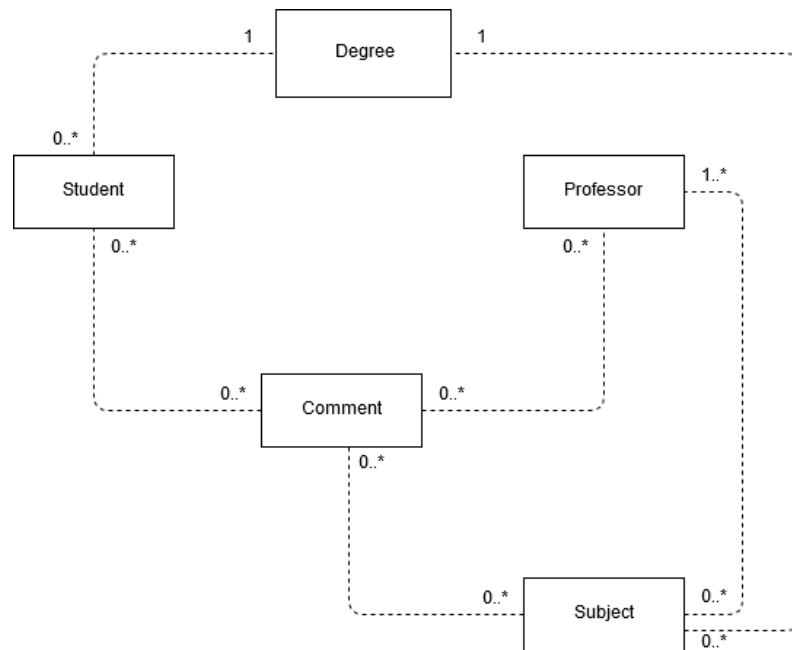


Figure 3: UML analysis diagram

5 Design

5.1 Software architecture

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

- Front-end
- Middleware
- Back-end

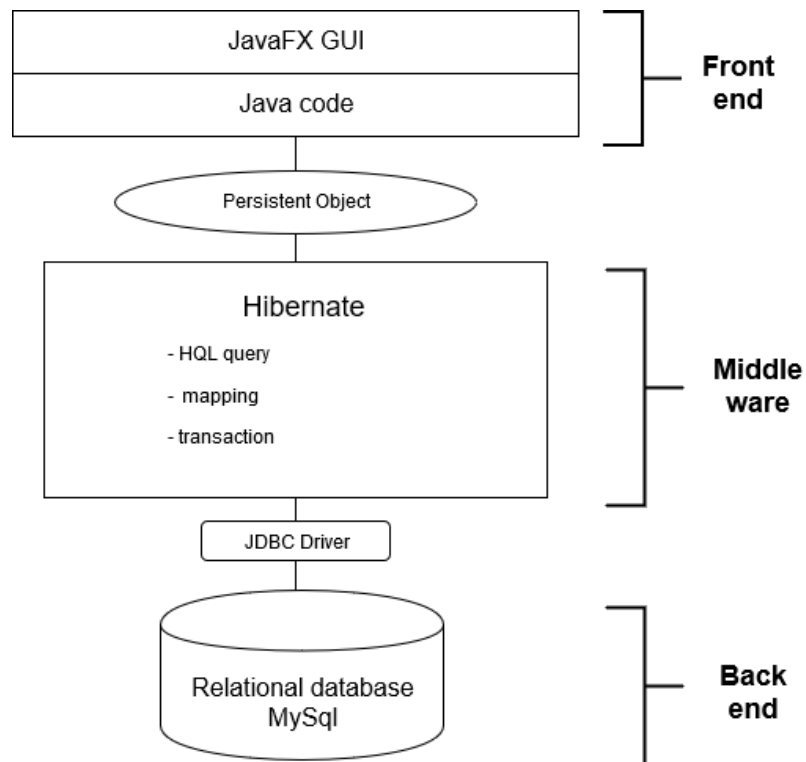


Figure 4: Software architecture diagram

5.2 Database design

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

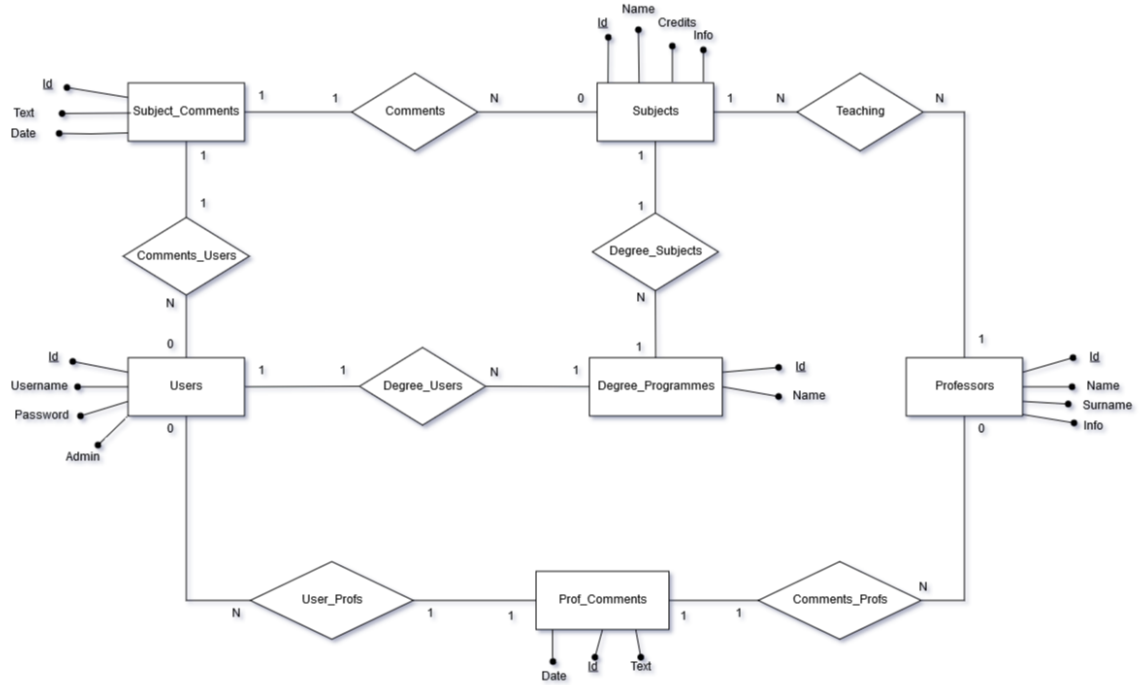


Figure 5: ER diagram

6 Implementation

6.1 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 guaranteed on Ubuntu 16 and Window OS. Moreover Maven is used to build and maintain the project, version 3.8.0.

The middleware layer is built thanks to Hibernate, version 5.4.4.0.

The jdbc driver manage the communication 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 .

6.2 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

These classes are declared using the Hibernate annotations syntax.

The following java-code shows a declaration of the class Subject as example, and contains the annotations used also in the other classes.

```
@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>();
```

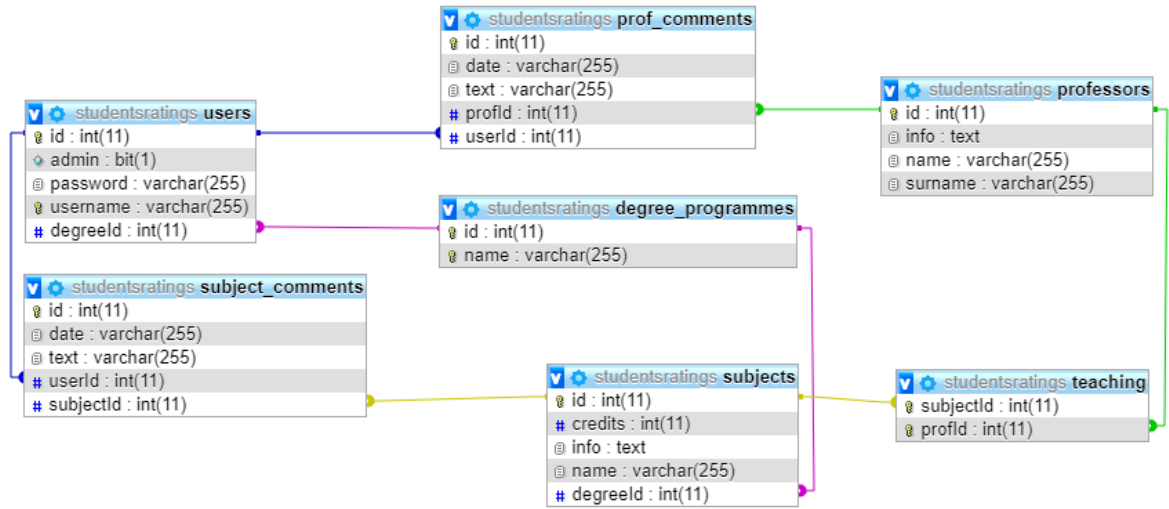


Figure 6: ER diagram

Another fundamental 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 described as follows.

6.3 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 professor Id doesn't exist");
                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;
    }
}
  
```

```

    }
    ...
}

```

6.4 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<>();
    String selectionSubjectByDegree = "SELECT_s" +
                                      "FROM Subjects_s" +
                                      "WHERE_degreeId_s=?1" +
                                      "ORDER_BY_s.name";

    try {
        entityManager = factory.createEntityManager();
        TypedQuery<Subject> query = entityManager.createQuery(selectionSubjectByDegree,
                                                                Subject.class);

        query.setParameter(1, degree);
        results = query.getResultList();
    }
    ...
    return results;
}

```

6.5 Update

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

```

public boolean updateCommentSubject(int subjectCommentId, String text, int userId)
    boolean updated = false;
    Date date = new Date();

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

        // if the user is the owner of that comment.
        if (subjectComment.getStud().getId() == userId) {
            entityManager.getTransaction().begin();
            subjectComment.setText(text);
            subjectComment.setDate(date.toString());
            entityManager.getTransaction().commit();
            updated = true;
        } else { // if the user is not the owner of that comment —> false.
            entityManager.close();
            return updated;
        }
    }
    ...
    return updated;
}

```

6.6 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 (subjectComment.getStud().getId() == userId) {
            entityManager.remove(subjectComment);
            entityManager.getTransaction().commit();
            deleted = true;
        } else {
            entityManager.close();
            return deleted;
        }
    }
    ...
    return deleted;
}
```

6.7 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.

7 Level DB

An alternative implementation of this application developed using MySQL relational database can be done using a key-value database, so in this chapter we describe an implementation of the application using levelDB.

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 unique identifier, which takes the name of "key"; couples of key-values are stored in namespaces (buckets) and in each bucket, keys must be unique. The main strengths of the key-value database are:

- Simplicity
- Speed
- Scalability

Key-value databases 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, classic RDBMS are preferred when organization and management are more important than performances; typically, when a database presents a lot of relations among entities, key-value 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 the key-value database as well, just by building up keys in a predefined way.

The application we realized is a multi-users one, so it needs a remote and shared database, to whom every user can access at every moment. LevelDB does not support remote usage (differently from other key-value databases), so it is not feasible to re-implement our kind of application realizing that tool. Moreover, one of the strength of LevelDB is the possibility to handle flexible data but, in our case, the data does not vary a lot from their original structure (etc. NULL value are not expected, etc.). In conclusion, we don't have many advantages in replicating the application in a key-value architecture.

Just for an illustrative purpose, we tried to recreate the structure of our application using LevelDB. First of all, we needed to define the key structure for our entities; to do so, we have chosen to represent the tables in our buckets using the table name, the id of the object we are referring to and the attribute name for the value. Let's consider our "users" table, used to store information about the users of our application.

	Id	Username	Password	degree	Admin
►	1	Geghi	Geghi	3	1
	2	alice	alice	3	0
	4	stefano	stefano	3	0
	6	marco	marco	3	0

In the key-value database, we use keys to retrieve the values of our objects. To define the three main components of the key (prefix, identifiers, and suffix), we extract that information from the entity. So, we use the entity name (users) to define the prefix, the id (called `user_id`) as the identifiers of the key and the suffix is specified by its related attribute. Our schema can be represented by a group of key-value pair specified as follows:

user:\$user_id:\$attribute_name = \$value

The table “users” becomes:

```
Users:1:Username = “Geghi”
Users:1:Password = “Geghi”
Users:1:degree = “3”
Users:1:Admin = “1”
Users:2:Username = “alice”
Users:2:Password = “alice”
Users:2:degree = “3”
Users:2:Admin = “0”
Users:4:Username = “stefano”
Users:4:Password = “stefano”
Users:4:degree = “3”
Users:4:Admin = “0”
Users:6:Username = “marco”
Users:6:Password = “marco”
Users:6:degree = “3”
Users:6:Admin = “0”
```

Relations among entities are represented in RDBMS with foreign keys. We used a similar approach in translating the relational database into the key-value schema. The foreign key becomes an additional identifier in the key for the pair.

For that example, we focus on the table “subject_comments” that contains the comments of the users about the subjects. Among the others, we store in that table information regarding the user that made a comment (using his id) and the subject of the comment (using the id of that subject).

id	userId	subjectId	text	date
1	1	1	Spiega bene ma va troppo veloce a mio parere	2019-10-09 15:25:57
2	1	4	L'esame è molto difficile ma le spiegazioni sono molto precise. Da migliorare forse la preparazione in lab...	2019-10-09 15:27:13
3	1	10	Seguire le lezioni per questo corso è molto consigliato	2019-10-23 21:21:30

Figure 7: Subject comments table

We can see the userId, foreign key related to the user who made the comment, and subjectID, foreign key of the subject.

To represent that relation, the identifiers of the key becomes the id of the comment (subject_comment_id) the user id (user_id) and the subject id (subject_id).

The prefix is again the name of the entity and the suffix is the name of the fields. The table is represented by a group of key-value pairs specified as follow:

```
subject_comments:$subject_comment_id:$user_id:$subject_id:$attribute_name
= $value
```

The table “subject_comments” becomes:

```
Subject_comments:1:1:1:text = “Spiega bene ma va troppo veloce a mio parere”
Subject_comments:1:1:1:date = “2019-10-09 15:25:57”
Subject_comments:2:1:4:text = “L'esame è molto difficile ma le spiegazioni sono ... “
Subject_comments:2:1:4:date = “2019-10-09 15:27:13”
Subject_comments:3:1:10:text = “Seguire le lezioni per questo corso è molto consigliato”
Subject_comments:3:1:10:date = “2019-10-23 21:21:30”
```

LevelDB saves the key-value pairs in a file. To initialize that file it is necessary to instantiate the class DB, Option and File, which will provide the tools to work with this architecture. That task must be computed only the first time, so we put the “init.txt” file into the folder project. At every start, the application will check that file, but only the first time we run the code it will instantiate all the buckets; after that initialization, it will update the file writing “true” on it.

Here is the snippet of the code used in our application to create, initialize and populate the buckets with some basic information:

```
DB levelDBStore;
Options options = new Options();
File f = new File("levelDBStore");
levelDBStore = factory.open(f, options);

//key -> users:$user_id:$attribute name
levelDBStore.put("users:1:username".getBytes(), "Geghi".getBytes());
levelDBStore.put("users:1:password".getBytes(), "Geghi".getBytes());
levelDBStore.put("users:1:degree".getBytes(), "3".getBytes());
levelDBStore.put("users:1:admin".getBytes(), "1".getBytes());
```

In order to convert the “teaching” table, representing an N-M relation in RDBMS, to a key-value schema, we introduced a new attribute, changing a bit the structure of the table. Thanks to that table we can map the relation among subjects and professors who teach them.

	subjectId	professorId
▶	1	2
	2	3
	3	4
	4	1
	5	9
	6	7
	7	6
	9	5
	10	5
*	NULL	NULL

Figure 8: Teaching table

The key for that table becomes:

teaching:\$subjectId:professorId = \$value

where \$value represents the role of the professor in the teaching of that subject.

8 User Manual

When you first run the application, the interface you get is the one in figure 9.

The screenshot shows the 'Students Evaluations' application window. At the top, there is a login section with 'Username:' and 'Password:' labels, input fields, and a 'Login' button. Below the login section, on the left, is a placeholder box with the text 'Informations about Professors are visualized here'. To the right of this is a table with columns 'ID', 'NAME', and 'SURNAME'. The table contains 12 rows of professor data. Below the table, there is a comment section with a text input field labeled 'Leave a comment here...' and three buttons: 'Comment', 'Delete', and 'Update'.

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 9: 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. 10), or decide to view the list of subjects (fig. 11), for which is also available the degree course's filter.

The screenshot shows the 'Students Evaluations' application window with the 'Professors' drop-down menu set to 'Ingegneria Informatica'. The table on the right now displays only the professors associated with this degree course. The rest of the interface, including the login section and the comment section, remains the same.

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 10: Selection of professors filtered by "Ingegneria Informatica" degree course

Students Evaluations

Username: Password:

Informations about Subjects are visualized here

Text	Date
Nessun contenuto nella tabella	

Leave a comment here...

Subjects:

ID	NAME	CREDITS
1	Analisi Matematica I	12
4	Calcolatori elettronici	9
5	Costruzione 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

Figure 11: 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. 12).

Students Evaluations

User: alice

Informations about Professors are visualized here

Text	Date
Nessun contenuto nella tabella	

Leave a comment here...

Professors:

ID	NAME	SURNAME
5	Francesco	Marcelloni

Figure 12: 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. 13). 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. 14.

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 Logout

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

Leave a comment here...

Comment
Delete
Update

Professors

All

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: Displaying the comments related to a professor

Students Evaluations

User: alice Logout

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

Leave a comment here...

Comment
Delete
Update

Professors

All

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 14: Interface after adding a comment

8.1 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. 16).

Students Evaluations

User: Geghi Logout

Informations about Professors are visualized here

Text	Date
Nessun contenuto nella tabella	

Name: Professor Informations:

Surname:

Leave a comment here...

Add

Professors Artificial Intelligence and Data Engineering

ID	NAME	SURNAME
5	Francesco	Marcelloni

Edit Delete

Comment Delete Update

Figure 16: 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. 17).

Students Evaluations

User: Geghi Logout

Informations about Professors are visualized here

Text	Date
Nessun contenuto nella tabella	

Name: Professor Informations:

Surname:

Leave a comment here...

Add

Professors All

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

Edit Delete

Comment Delete Update

Figure 17: 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. 18).

Students Evaluations

User: Geghi Logout

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	

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

Surname: Ducange

Leave a comment here...

Add

Professors All

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

Edit Delete

Comment Delete Update

Figure 18: 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, separated 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. 19).

Students Evaluations

User: Geghi Logout

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	

Name: Large Scale and Multi-Tasking Subject Informations: Students will acquire accuracy and precision in the design and resolution of problems related to the design of non-relational databases

Credits: 9

Leave a comment here...

Add

Subjects Artificial Intelligence and Data Engineering

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

Edit Delete

Comment Delete Update

Figure 19: 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. 20).

Students Evaluations

User: Geghi Logout

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

Name:

Francesco

Surname:

Marcelloni

Leave a comment here...

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

Add

Professors

All

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

Edit

Delete

Comment

Delete

Update

Figure 20: Screen of the application's interface from which the admin can delete a comment