UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

ZOO MANAGEMENT SYSTEM

-DATABASE & OBJECT-ORIENTED PROGRAMMING PROJECT-

2nd YEAR, 1st SEMESTER

Student: Mara Mureșan

Group: 30423

Project Supervisor: Dr. Ing. Călin Cenan

Project Supervisor: Ing. Alexandru Ștefan Gârleanu

9.01.2024

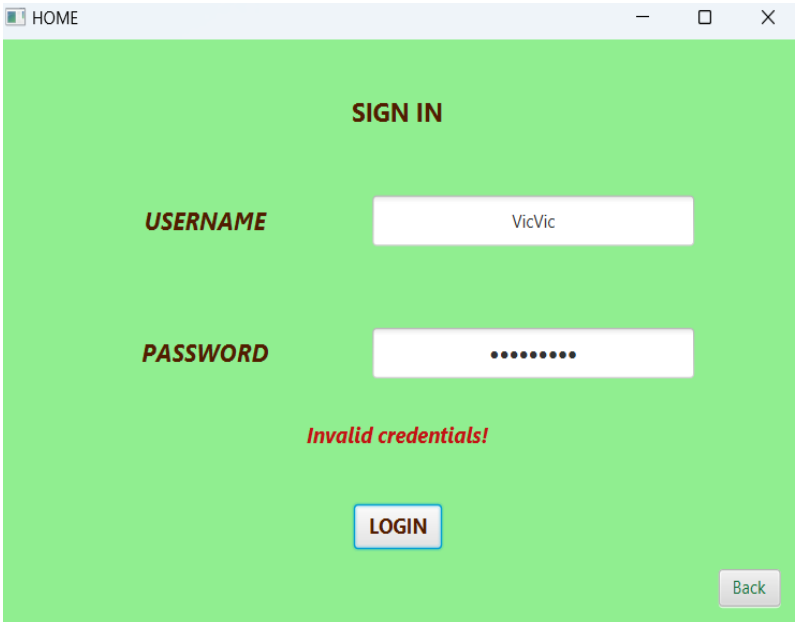
TABLE OF CONTENTS

1. TASK DESCRIPTION

This is a Java application designed for zoo management. The application utilizes a zoo database to enhance the experience for both workers and visitors. It covers key functionalities such as worker portals, environment and animal tracking, visitor management, ticketing, and reviews.

The first window that appears when the application is run.

When pressing the ‘Go On!’ button, the next window appears.

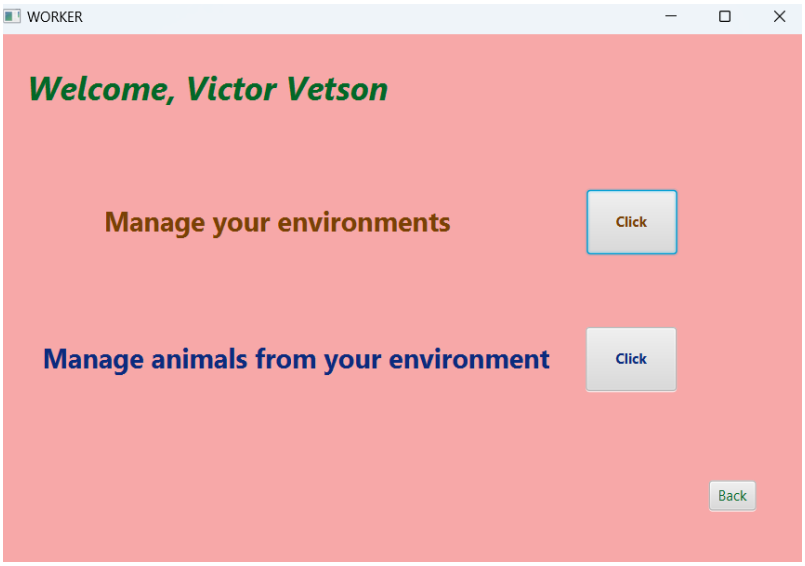


The user should write his username/password and press ‘LOGIN’ button.

If this information was wrongly introduced, “Invalid credentials” message is displayed.

Else, it goes to the account window depending on the type of the user: worker/visitor.

‘Back’ button goes to the previous window.

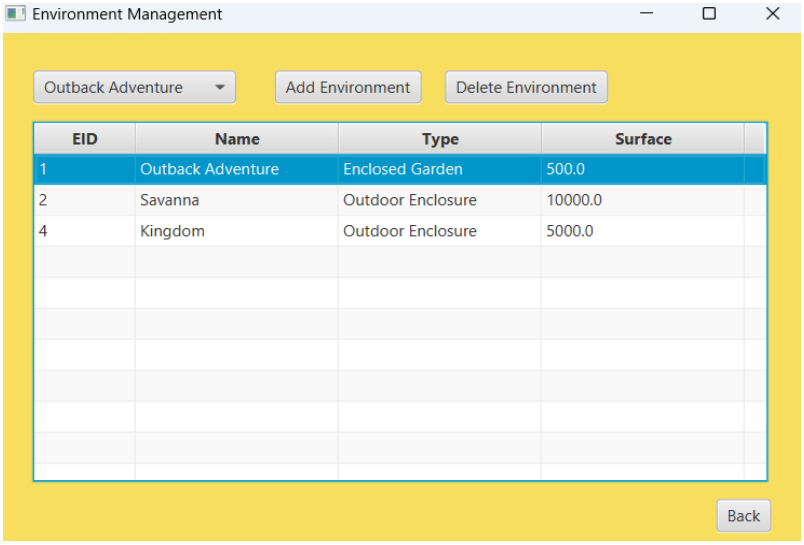


If the user is a worker, it has 2 options:

to manage the environments in which he works (by clicking the first button)

to manage the animals from the environments in which he works (by clicking the second button)

'Back' button goes to the previous window.

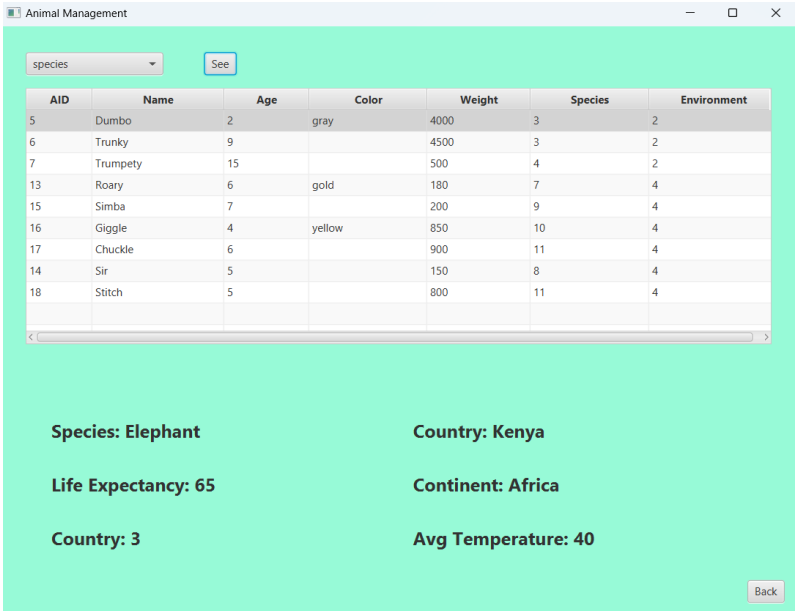


The worker can see the environments in which he works.

He can choose an environment from the ComboBox and add it to his list. ('Add' button)

If he selects an environment from the table, he can delete it. ('Delete' button)

'Back' button goes to the previous window.



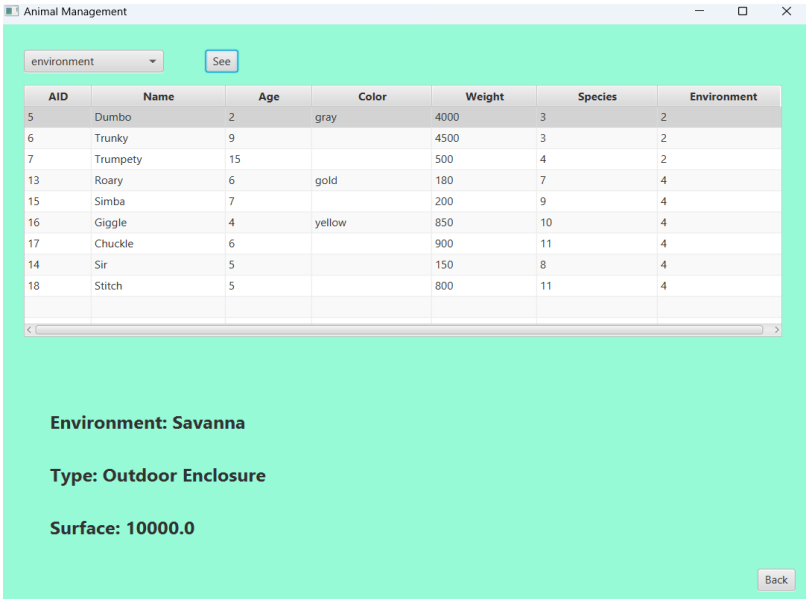
The worker can see the animals from the environments in which he works.

He can select an animal from the table, and an option from the ComboBox.

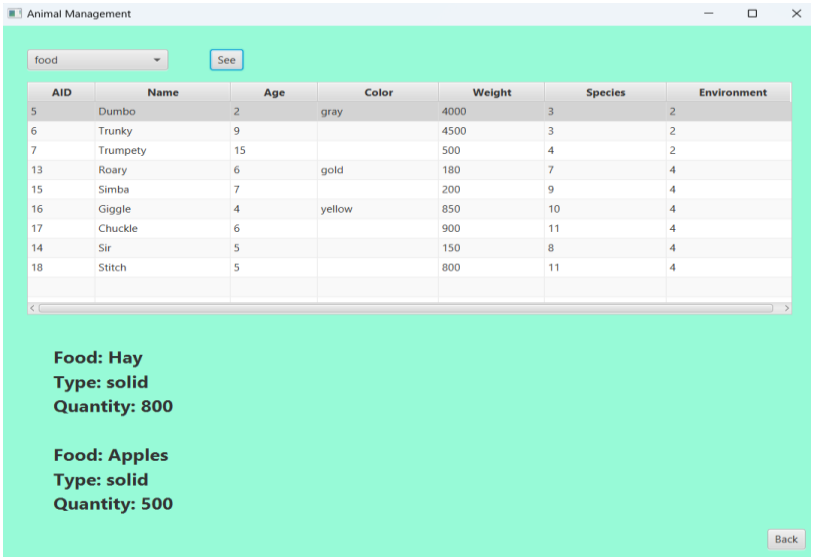
If he presses the 'See' button he can see the selected properties for the selected animal.

If he chooses 'species' he can see the species and the country.

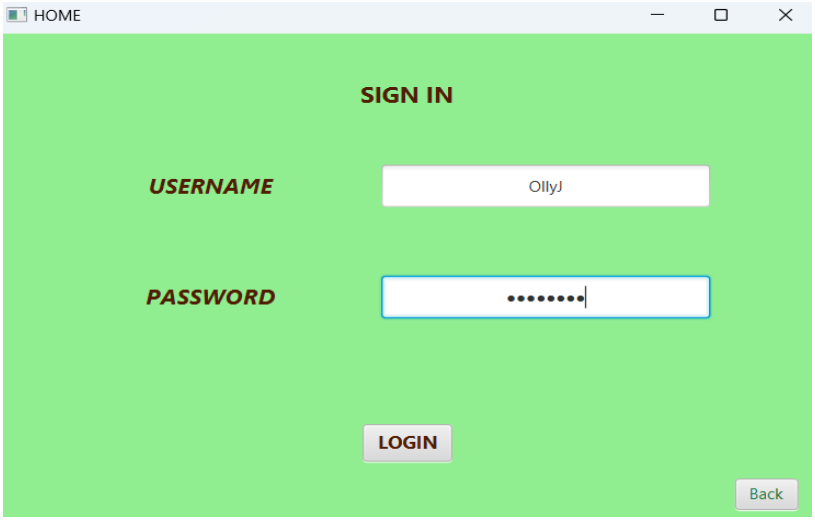
'Back' button goes to the previous window.



If he chooses 'environment' he can see the environment in which the selected animal lives.



If he chooses 'food' he can see the list of foods that the selected animal eats.



Now, the user is a visitor.

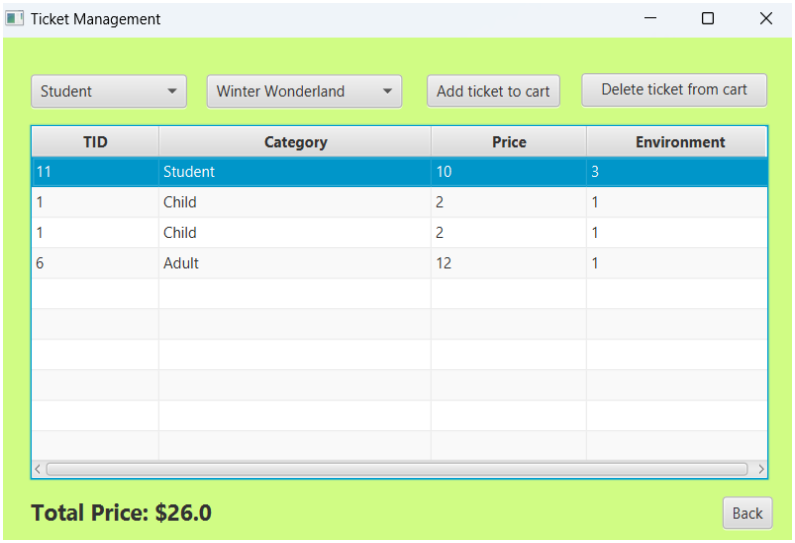
'Back' button goes to the previous window.

If the user is a visitor, it has 2 options:

to buy a ticket to the zoo (by clicking the first button)

to write a review of his visit (by clicking the second button)

'Back' button goes to the previous window.

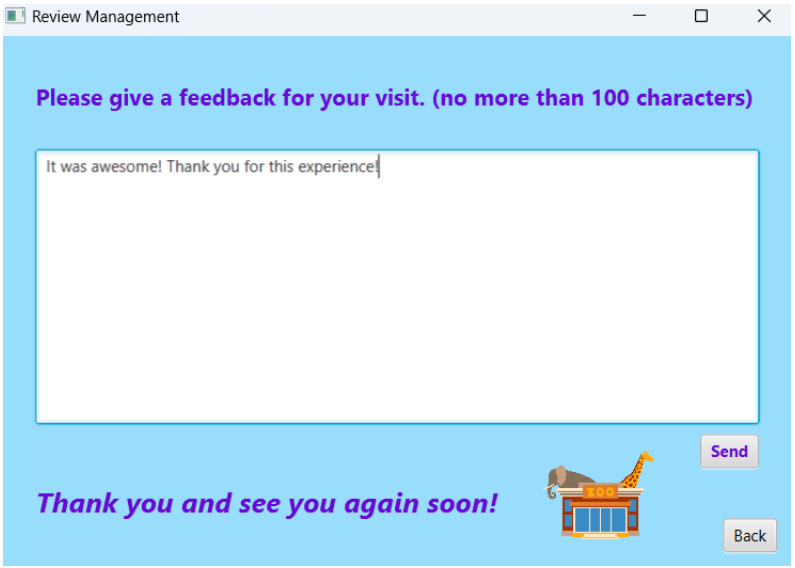


The visitor can see his shopping cart and the total price of the tickets from his cart.

He can choose a category and an environment from the ComboBoxes and buy the ticket with these criteria. ('Add' button)

If he selects a ticket from the table, he can delete it from the cart. ('Delete' button)

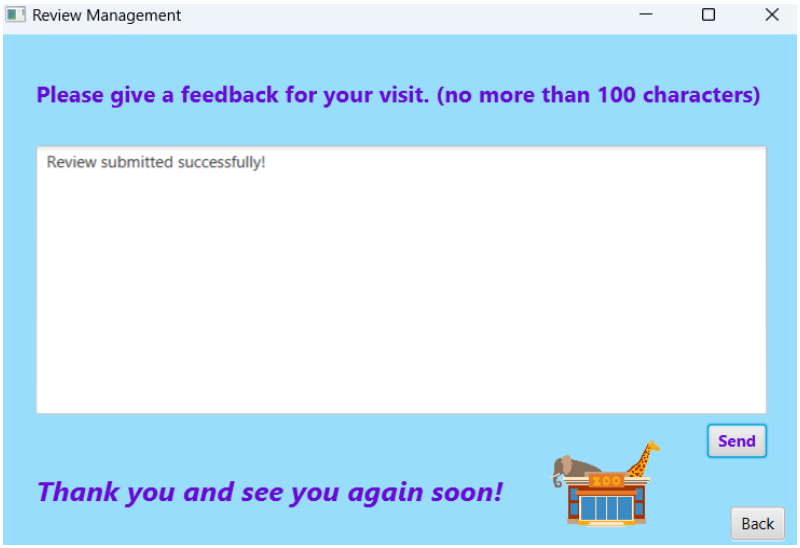
'Back' button goes to the previous window.



The visitor can leave a feedback message.

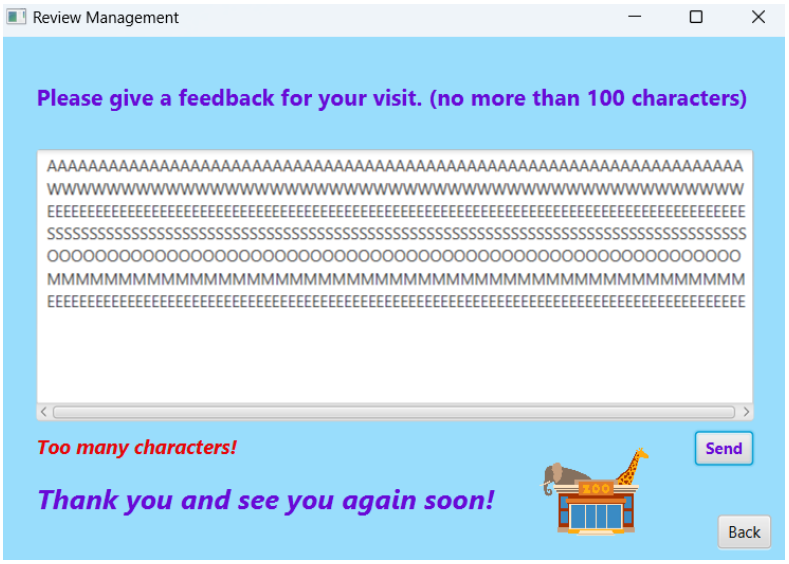
He can write in the TextArea a text with maximum 100 characters.

'Back' button goes to the previous window.



If he presses the 'Send' button, "Review submitted successfully" message will appear in the TextArea and the message will be stored in the Database.

(if the message is shorter or equal to 100 characters)



If he presses the 'Send' button, "Too many characters!" message will appear in a Label.

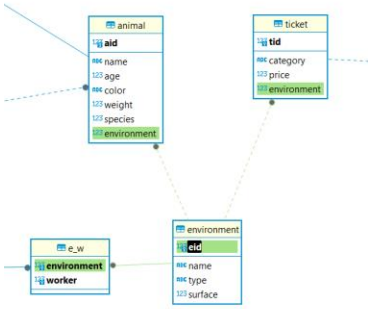
(if the message has more than 100 characters)

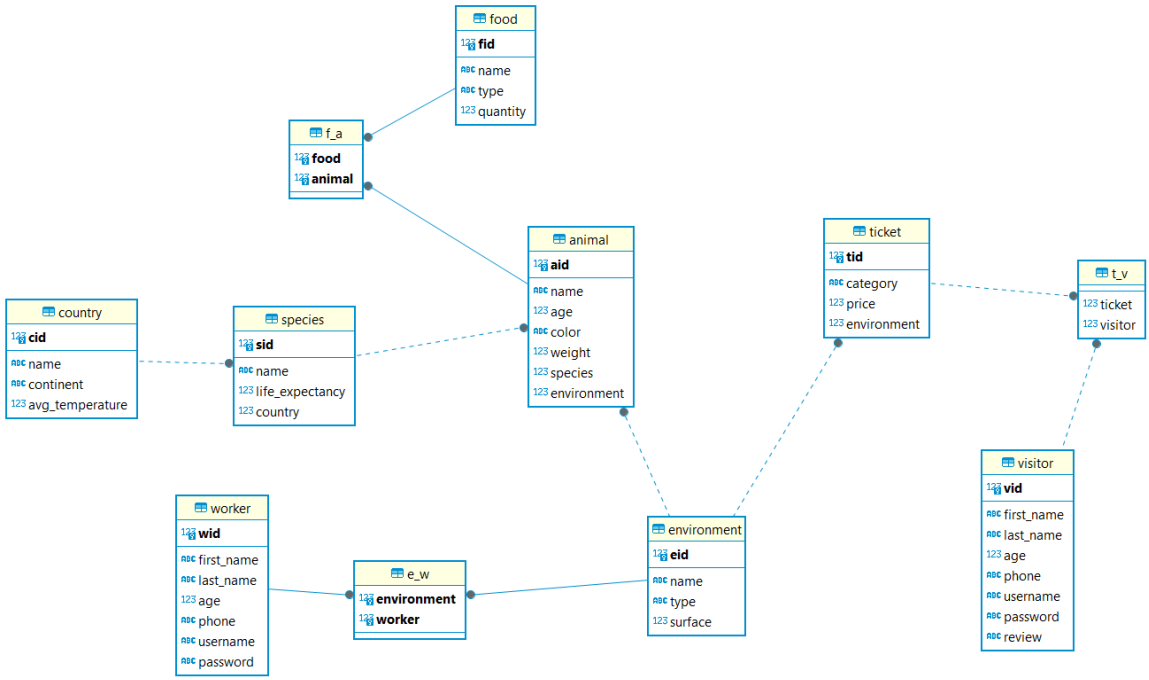
2. DATABASE

First of all, I made a Zoo database using PostgreSQL, in DBeaver.

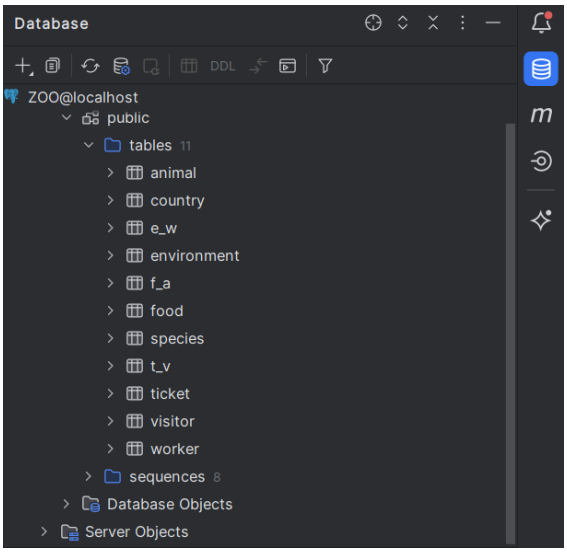
Components

- 11 tables
- 1-to-many relationships (ex: a country can have many species)
- many-to-many relationships, solved by adding auxiliary tables (ex: table f_a, a food can be eaten by many animals and an animal can eat many foods)
- constraints:
    - primary keys
    - unique keys (for first_name & last_name)
    - check constraint (price < 50)
- foreign keys





After that, I connected the database with my Java application project as New Data Source, using IntelliJ IDEA.




```
1 usage
private Connection establishConnection() {
        try {
            // JDBC URL, username, and password of MySQL server
            String jdbcUrl = "jdbc:postgresql://localhost:5432/ZOO";
            String username = "postgres";
            String password = "postgres";

            // Establish the connection
            return DriverManager.getConnection(jdbcUrl, username, password);
        } catch (SQLException e) {
            e.printStackTrace();
            throw new RuntimeException("Failed to connect to the database");
        }
}
```

Scripts

In the Java code, I integrated some SQL scripts in order to retrieve, add or delete information from the tables.

- Get all the environments

```
//'e-w' is the join table between 'Worker' and 'Environment'
String query = "SELECT DISTINCT e.* FROM environment e " +
        "JOIN e_w ew ON e.eid = ew.environment";
```

- Get the environments for a specific worker

```
String query = "SELECT e.* FROM environment e " +
        "JOIN e_w ew ON e.eid = ew.environment " +
        "WHERE ew.worker = ?";
```

- Add environment for a worker

```
String query = "INSERT INTO e_w (environment, worker) VALUES (?, ?)";
```

- Delete environment for a worker

```
String query = "DELETE FROM e_w WHERE environment = ? AND worker = ?";
```

- Get the country for an animal

```
String query = "SELECT c.* FROM country c " +
        "JOIN species s ON c.cid = s.country " +
        "JOIN animal a ON a.species = s.sid " +
        "WHERE a.aid = ?";
```

- Get the ticket by category and environment

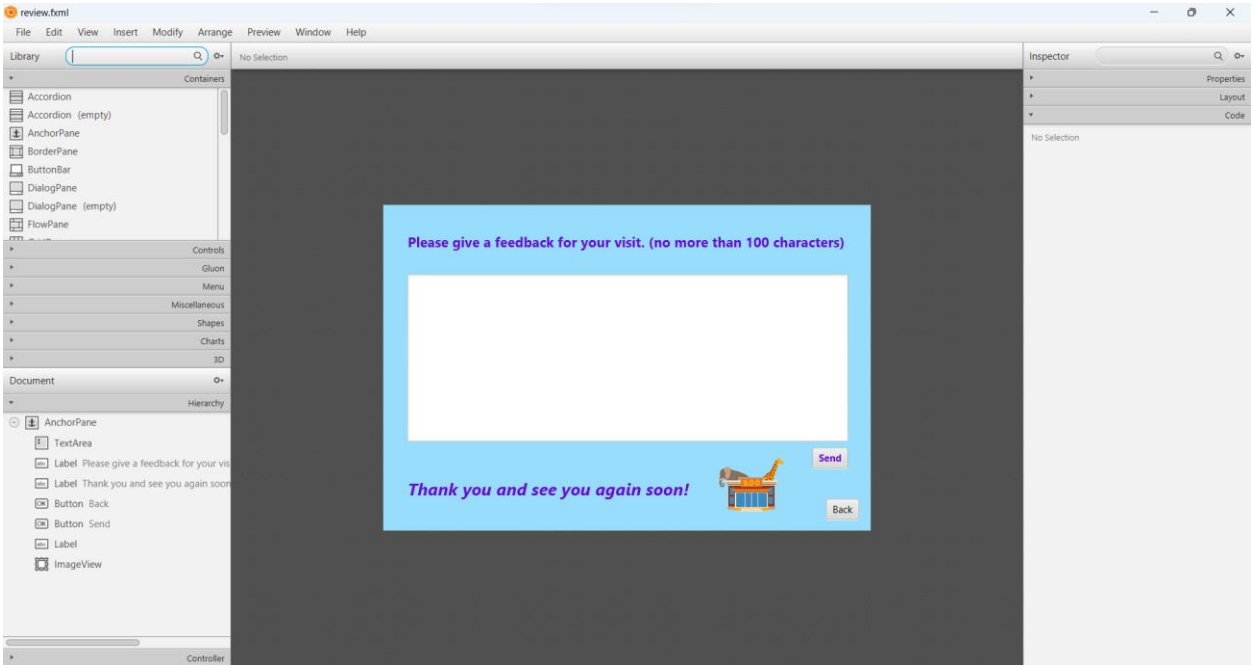
```
String query = "SELECT * FROM ticket WHERE category = ? AND environment = ?";
```

- Update the visitor's review

```
String query = "UPDATE visitor SET review = ? WHERE vid = ?";
```

3. JAVA INTERFACE

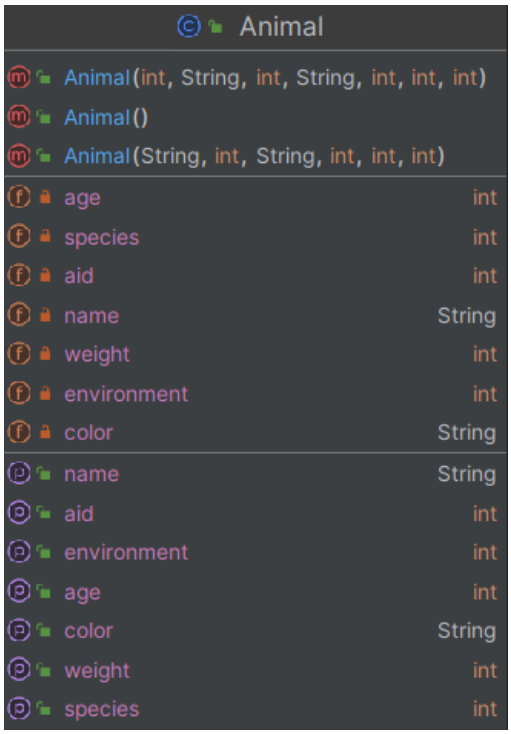
I used Java logic for creating useful methods and JavaFX & FXML files for making the GUI interface. All the .fxml files were opened and visually customized using SceneBuilder.

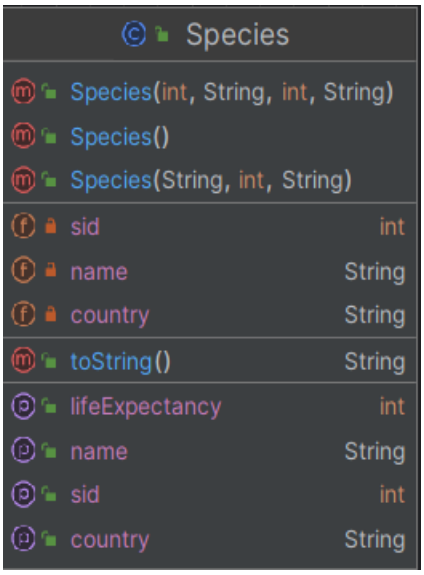


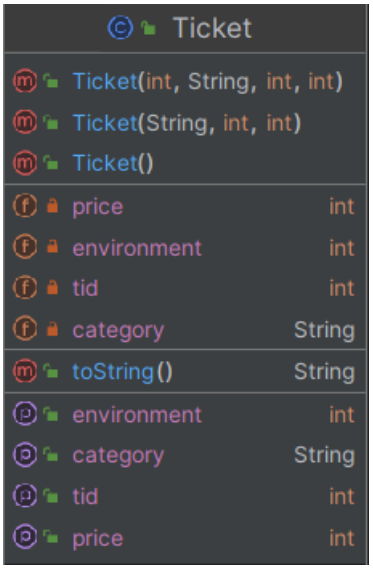
Components

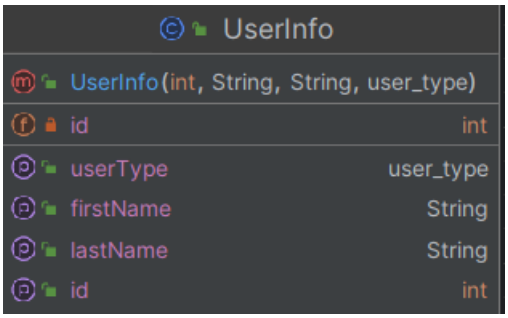
- Classes for tables
  - constructors
  - getters & setters for each column of the table
- Repositories
  - general methods that interact with the database
- Controllers
  - methods that load/interact with the .fxml files
  - methods called when pressing the buttons

Diagrams



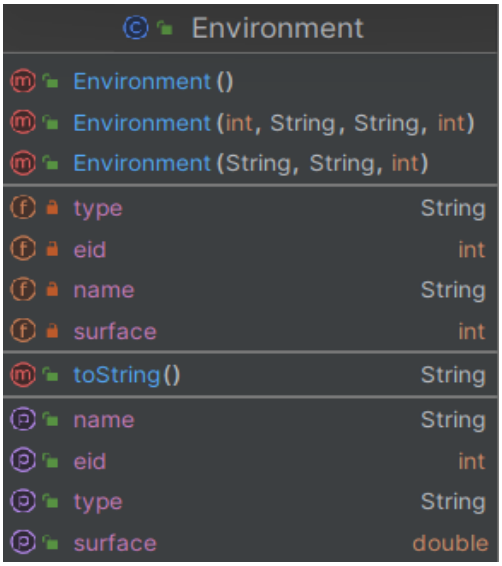


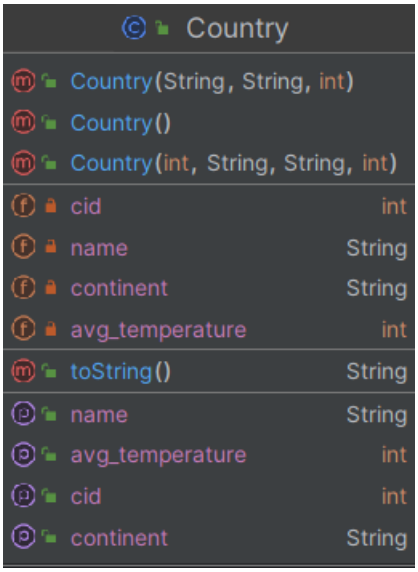


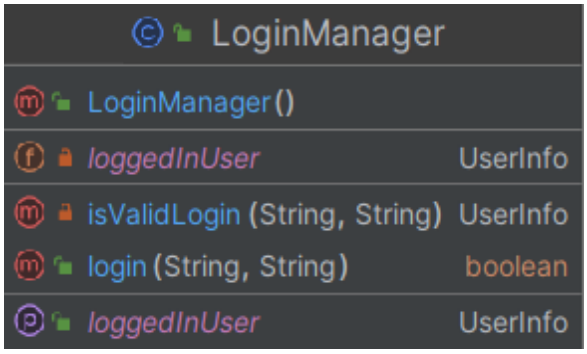


Classes have short methods for getting and setting the values for each column of the table. They are very useful for finding information about a class.

(ex: I use 'species.getName()' for easily finding the name of a certain species)







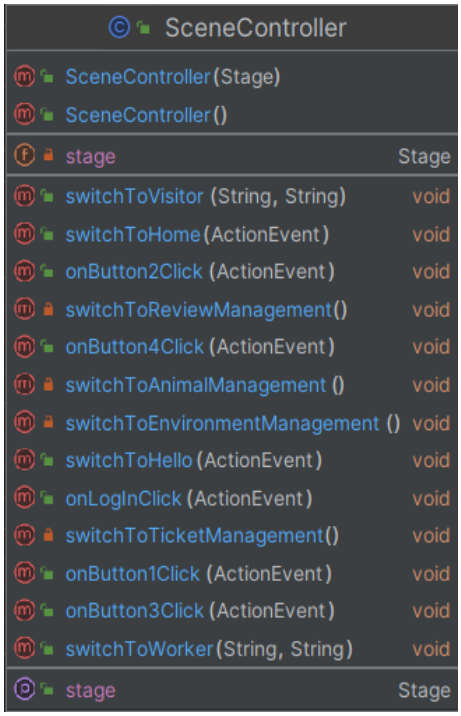
Repositories have methods that take information from the database and return it in the Java code.

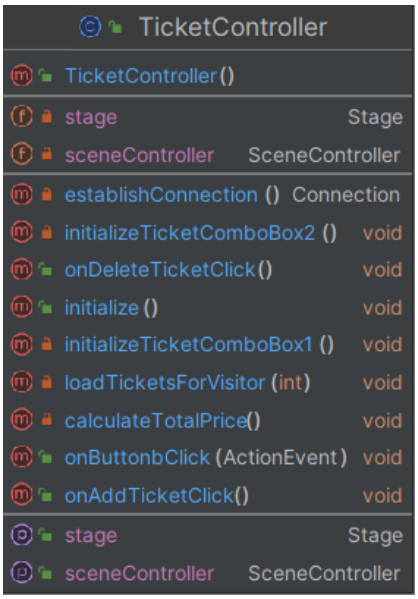
ex: 'isValidLogin()' method gets, using SQL query, the username and password of the user that logged in and searches it in the 'visitor' and 'worker' tables;

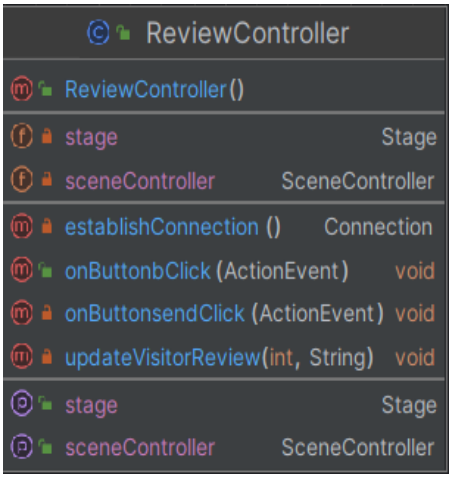
if the credentials were found in the 'visitor' table, the method returns a UserInfo (the class previously defined) with 'vid', 'first_name', 'last_name' and an enum type 'VISITOR';

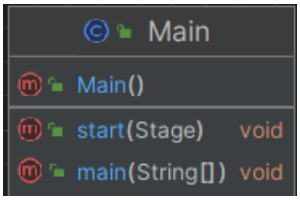
if the credentials were found in the 'worker' table, the method returns a UserInfo (the class previously defined) with 'wid', 'first_name', 'last_name' and an enum type 'WORKER';

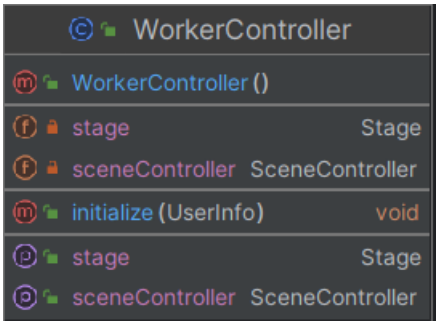
else, if the credentials were not found anywhere, the method returns the id 0, empty strings for the names and the enum type 'INVALID'

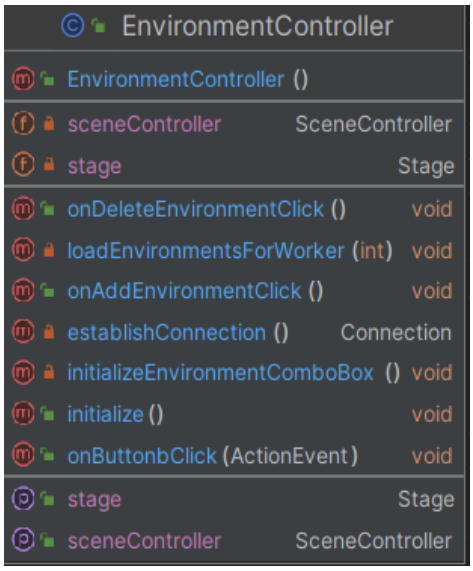


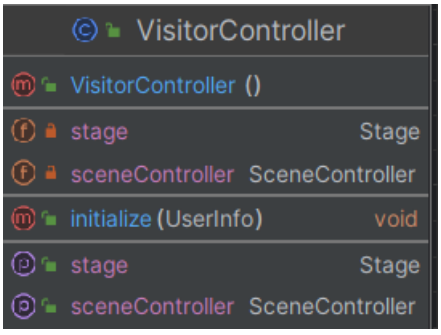


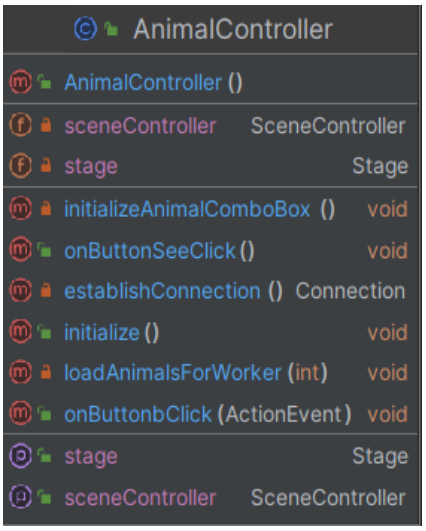












Controllers have methods written mainly for the FXML interface that use methods from the repositories. The functions' headers are written in the buttons' 'onAction' fields.

They are used for connecting with the database, setting the proper stage, initializing tables, comboBoxes, buttons, labels and switching between different scenes.

4. FUTURE DEVELOPMENTS

By now, I couldn't find an appropiate method to implement a 'Back' button that switches between scenes from different controllers. It gives an error like 'No specified controller'. So, solving this issue would be a future goal.

Also, it could be a good idea to try to make a Sign-Up page and a window that keeps track of the environments of the zoo seen by each visitor.

5. REFERENCES

- Database Lab Works and Courses
- OOP Lab Works and Courses
- Youtube tutorials
- Web resources