

Software design

Documentation

Assignment 3

Name: Cristian Muntean

Group:30434

Contents

1. Objective	3
1.1. General requirement.....	3
1.2. Specific requirement	3
2. Tools used.....	4
2.1. List of used tools	4
2.2. Justification for using the selected tools.....	4
2.2.1. Java.....	4
2.2.2. Java Swing	4
2.2.3. Sergeds library.....	4
2.2.4. Jackson Dataformat XML.....	5
2.2.5. Gradle	5
2.2.6. JUnit, Groovy and Spock	5
3. UML diagrams	5
3.1. Use case diagram	5
3.2. Class diagram	7
3.3. Entity-Relationship diagram.....	9
3.4. Activity diagrams.....	10
3.4.1. Activity diagram to change the language of the app	10
3.4.2. Activity diagram to login the app.....	11
3.4.3. Activity diagram to view the cars in service.....	13
3.4.4. Activity diagram to filter the table containing the cars in service	14
3.4.5. Activity diagram to search cars by owner	15
3.4.6. Activity diagram to save car information in different formats	16
3.4.7. Activity diagram to perform CRUD operations on tables.....	17
3.4.8. Activity diagram to view car statistics	19
4. Implementation	20
4.1. Login window	20
4.2. Employee window.....	21
4.3. Manager window	24
4.4. Admin window	25

1. Objective

The objective of this assignment is to become familiar with the Model-View-Controller architectural pattern, as well as the Observer behavioral design pattern. A relational database (SQL Server, MySQL, etc.) will be used for information persistence.

1.1. General requirement

The general requirement for the first homework is:

- ❖ In the analysis phase, the use case diagram and activity diagrams will be created for each use case.
- ❖ In the design phase, the following will be created:
 - Class diagram respecting the MVC architecture and using the Observer behavioral design pattern;
 - Entity-relationship diagram corresponding to the database.
- ❖ In the implementation phase, code will be written to fulfill all the functionalities specified by the use case diagram using:
 - the design provided by the class diagram
 - one of the following programming languages: C#, C++, Java, Python.
- ❖ During the testing phase, unit tests (testing project) corresponding to the operations of creating the database, establishing a connection to the database, creating tables, and querying tables of the database will be implemented.
- ❖ Completion of the topic will consist of submitting a directory containing:
 - A file with the UML diagrams created;
 - The database;
 - The software application;
 - Documentation (minimum of 10 pages) - a file that includes:
 - the student's name, group;
 - the statement of the problem;
 - the tools used;
 - the justification for the chosen programming language;
 - description of the UML diagrams;
 - description of the application.

1.2. Specific requirement

The requirement for the specific homework is:

Develop an application that can be used in an auto service. The client application will have 3 types of users: employee, manager, and administrator.

After authentication, employee users can perform the following operations:

- ❖ View the list of all existing vehicles in the service, sorted by brand and fuel type;
- ❖ Filter vehicles by certain criteria: owner, brand, color, fuel type;
- ❖ Search for a vehicle by owner;
- ❖ CRUD operations regarding the persistence of vehicles and their owners;
- ❖ Save lists with information about vehicles in multiple formats: csv, json, xml, txt.

After authentication, manager users can perform the following operations:

- ❖ View the list of all existing vehicles in the service, sorted by brand and fuel type;
- ❖ Filter vehicles by certain criteria: owner, brand, color, fuel type;
- ❖ Search for a vehicle by owner;
- ❖ Save lists with information about vehicles in multiple formats: csv, json, xml, txt;
- ❖ View statistics related to vehicles using graphs (radial structure, ring structure, column structure, etc.).

After authentication, administrator users can perform the following operations:

- ❖ CRUD operations for information related to application users;
- ❖ View the list of all users and filter it by user type.

The graphical interface of the application will be available in at least 3 languages of international circulation.

2. Tools used

2.1. List of used tools

For the implementation of the application I used several tools:

- Java – for the development of the production code which is run by the application
- Java Swing – for the graphical user interface
- Sergeds swing mvvm – library for creating the mvvm pattern
- Jackson Dataformat XML – library for deserializing XML files as POJOs
- Gradle – for dependency management
- JUnit, Groovy and Spock – for writing the unit tests

2.2. Justification for using the selected tools

2.2.1. Java

I decided to use Java for the development of the production code in order to try to understand better how different architectural patterns and principles are applied in said programming language. During the development of this project, I learned how to implement the MVP architectural pattern and how to apply the SOLID principles in order to create classes and method which are more comprehensible and easier to understand.

2.2.2. Java Swing

I used Java Swing for the graphical user interface because I was already familiar with it and had some experience in creating different layouts of GUIs in order to present the required data in a user friendly way.

2.2.3. Sergeds library

I used this external library in order to create the MVVM architectural pattern in swing. This library offers several bindings that can be used to bind the text fields, combo boxes and check boxes from the view classes in the view-model classes.

2.2.4. Jackson Dataformat XML

I used this external library in order to deserialize XML files into POJOs, which were further used for implementing the internationalization feature of the app. The app contains 3 XML files, each one containing every phrase used in all of the views. When a language is selected, the XML corresponding to that language is deserialized in a POJO, which is further used to dynamically set the labels and buttons text to the phrases translated in the selected language.

2.2.5. Gradle

I recently learned Gradle from my workplace, so I decided to try to deepen my knowledge in it and try to make a clear separation between Maven and Gradle in order to create an opinion on which one I would prefer to use day by day.

2.2.6. JUnit, Groovy and Spock

Knowing that the implementation of the application was going to be created in Java, I used JUnit to create the unit tests for the production code.

I also recently learned Groovy and Spock at my workplace. I discovered that writing tests in Groovy makes the tests more readable and are easier to understand what is a “prerequisite” for a test, what is actually tested and what is expected from a test. Also, Groovy provides more readable test names that

On top of that, Spock helped to remove redundant code, which would have been copy-pasted in multiple tests which tested similar functionalities. For example, when testing if a specific table is created after running a given method, the expected result should be that they are created. This test should be done for every table in the database, which would have the same structure: call method which creates a table, then check if it exists. By using the “where” from Spock, this redundancy is removed and a single test is created which will be iterated over using different parameters for functions.

3. UML diagrams

During the analysis and the design phase, the use case diagram, the class diagram and the entity-relationship diagram had to be created.

3.1. Use case diagram

The first one was the use case diagram, which was created during the analysis phase of the project (Figure 1).

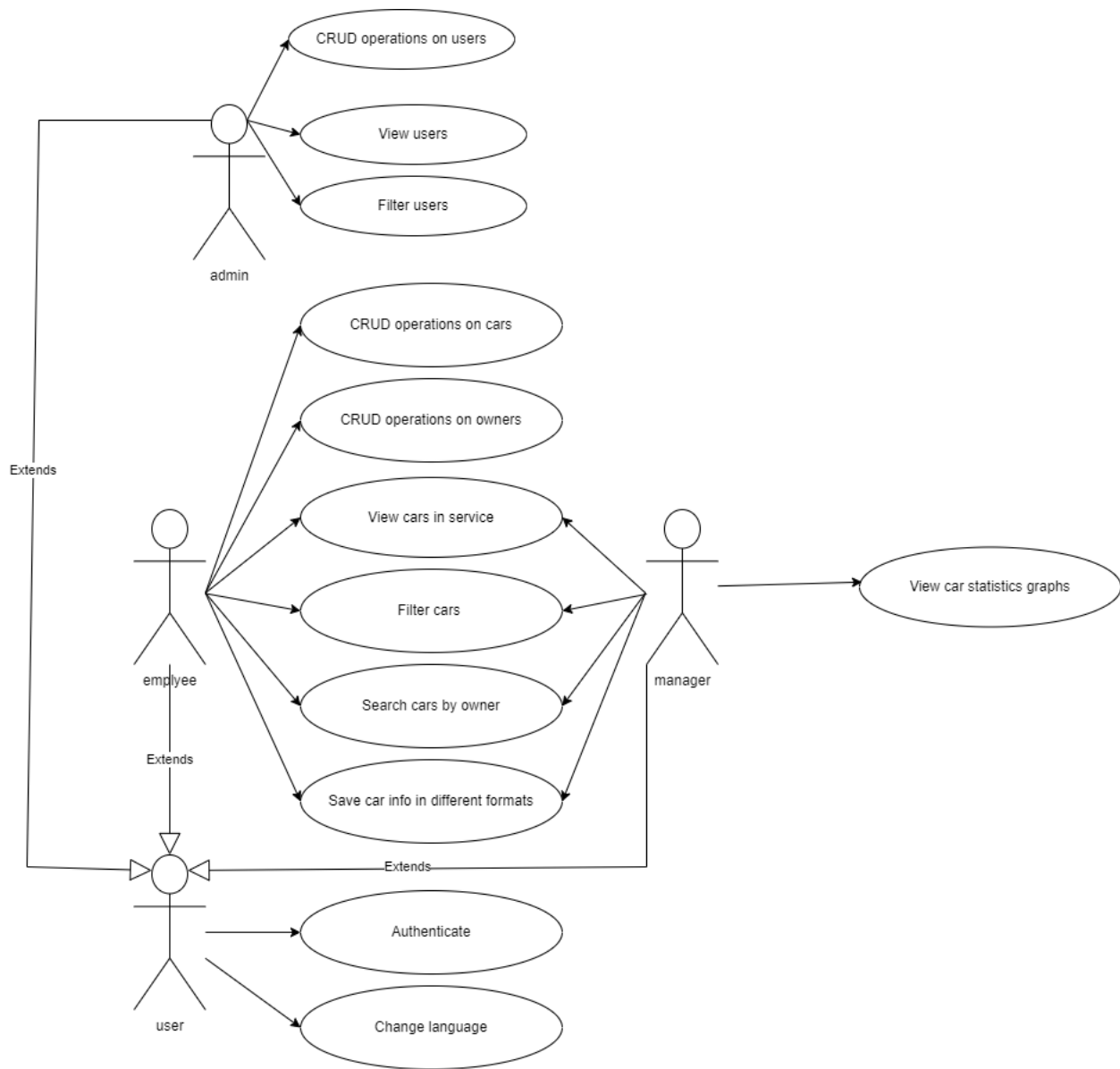


Figure 1

We can see that in order to use the application, the user must first authenticate into it. After the authentication is done, based on their role, the user will be able to perform some given actions. On top of that, every user is able to change the language of the application.

If the user is an employee, he will be able to perform Create, Read, Update and Delete operations on cars and owners of the cars, in order to create reservation in the service shop. Also, he can see the cars that are currently in the shop, sort them by brand and fuel type and filter them by owner, car brand, fuel type and color of the car. Also, the employee is able to search cars by the owner of the car and save the car information in different formats (csv, xml, json, txt).

If the user is a manager, he will only be able to see the cars that are available in the shop, sort them by brand and fuel type and filter them by owner, car brand, fuel type and color of the car. Also, the employee is able to search cars by the owner of the car and save the car information in different formats (csv, xml, json, txt) just like the employee can. The manager can also view statistics graphs about the cars that are present in the service shop.

If the user is an admin, he will be able to view the users that are present in the database, filter them by the role, perform Create, Update and Delete operations on the current users of the application and see the users that are currently registered into the application's database.

3.2. Class diagram

During the design phase of the project, the class diagram was created which shows the classes of the application and the relationship between them (Figure 2).

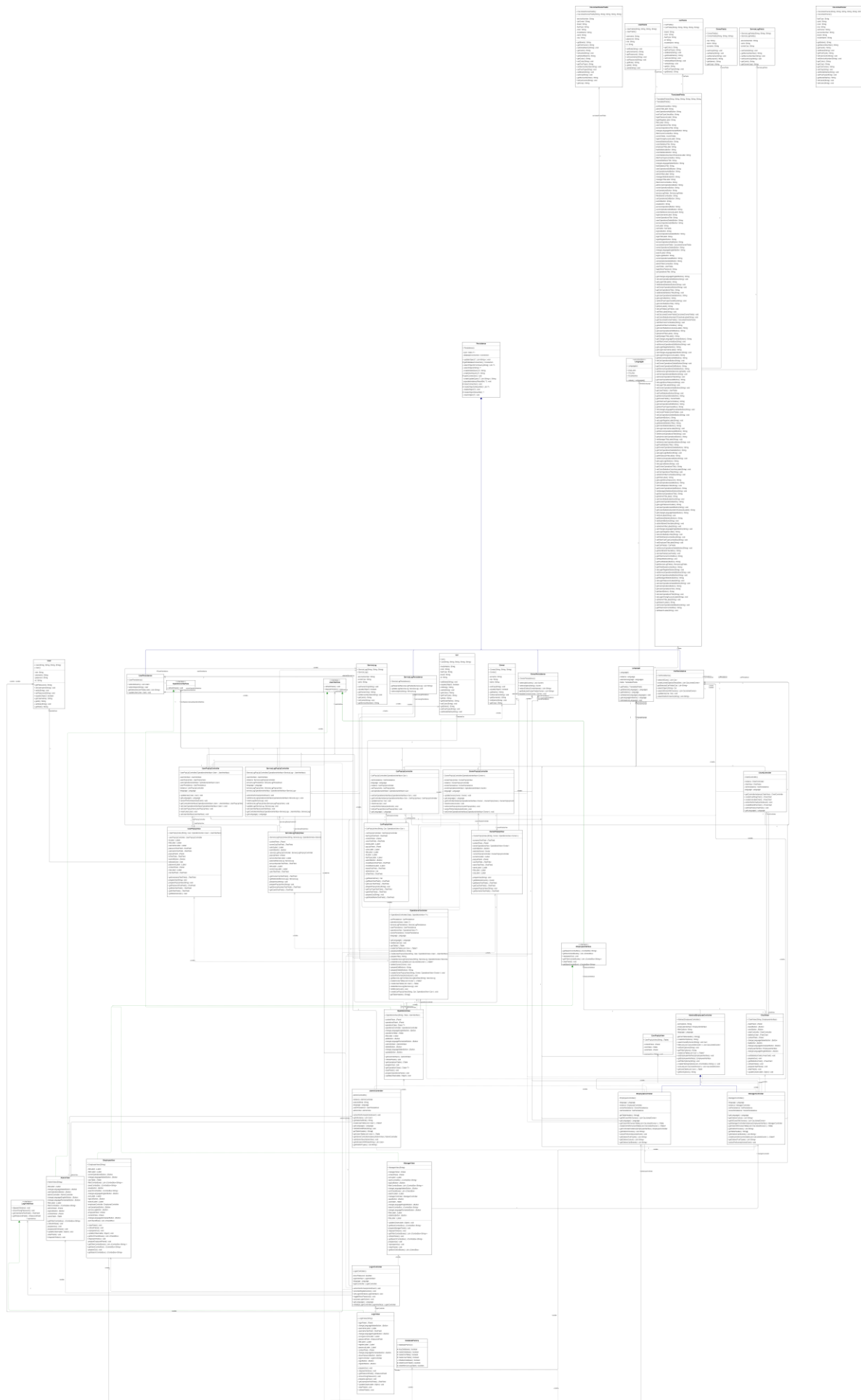


Figure 2

In this class diagram we can see that the MVC architectural pattern was respected, each class being part of one of the following packages: model, persistence, view or controller. In order for the user to perform an action at a given point in time, each element from the view classes (that has an action associated to it) has an action listener in the view's respective controller that, when triggered, will perform a specific action such as login or CRUD operations on a table. Each view has a controller associated to it that acts as a bridge between the view and the model and it contains all the necessary logic to perform the desired actions.

Furthermore, every time the application starts, it will try to create a new database if doesn't currently exist. On every operation that requires the access to the database, first a connection will be established, then the operation will be performed, then the connection will be closed. This is done in order to promote the efficiency of the application.

3.3. Entity-Relationship diagram

The entity-relationship diagram was also created during the design phase of the project. It represents the structure of the database and the relationships that are present between different tables (Figure 3).

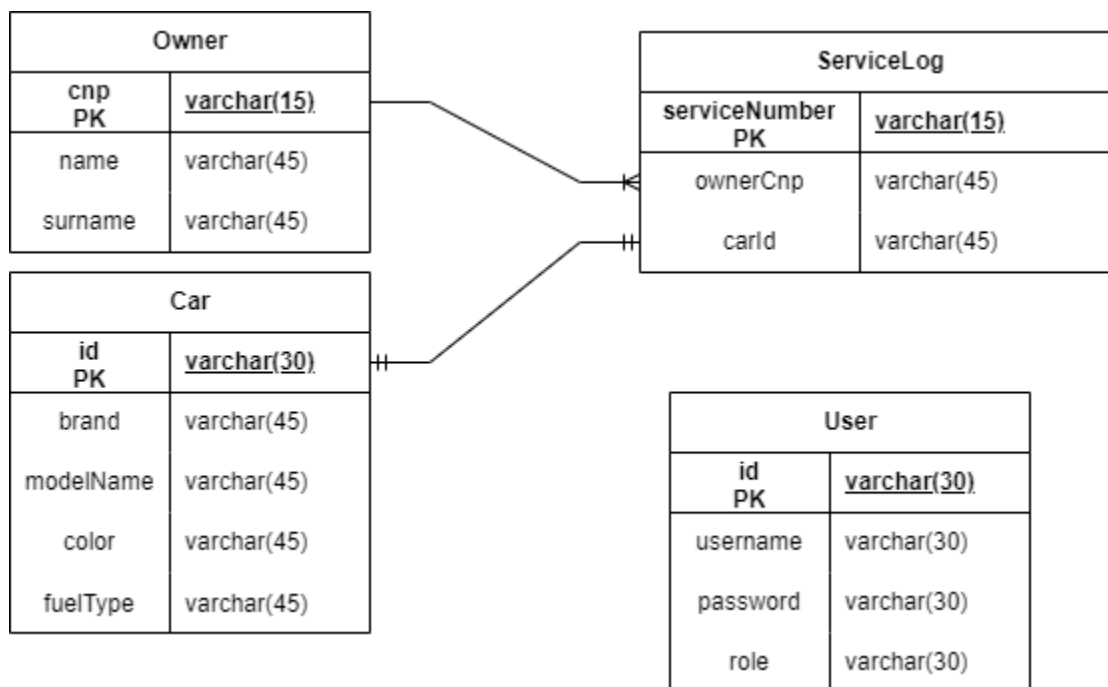


Figure 3

The User table will store information about all the users. Each user has a unique id and username, a password and a role. The role can be one of the following 3 options: "employee", "manager" or "admin".

The Owner table will store information about each owner of a car. Every owner has a unique cnp, a name and a surname.

The Car table will store information about each car. Every car has a unique id, a brand, a model name, a color, and a type of fuel.

The Service Log table will store information about a car and its owner that is present in the shop. An owner can have multiple cars in the shop at once, but a unique car can be in the shop only once at a given time.

3.4. Activity diagrams

The activity diagrams are used along side the use case diagram in order to provide a more detailed explanation for each use case presented in the use case diagram. Thus, the following activity diagrams were created.

3.4.1. Complete activity diagram

The complete activity diagram is presented in Figure 4.

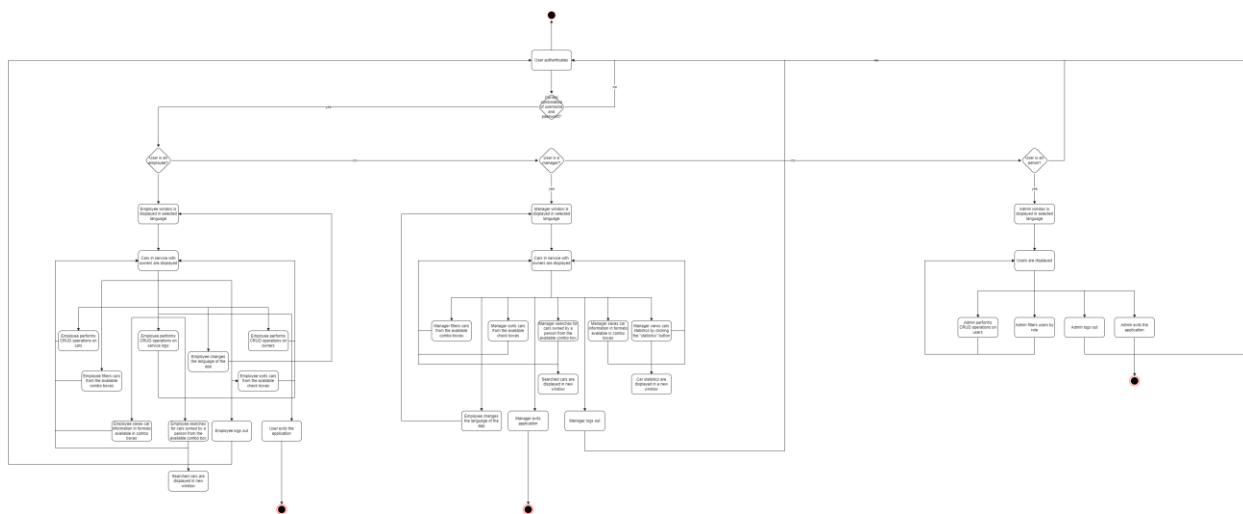


Figure 4

As presented above, the application starts with authentication. Then, based on whether the credentials that the user has entered, either the credentials are wrong and he has to enter a new username and password, or the authentication is successful and the user enters the application. After a successful authentication, the application displays the employee window, the manager window or the admin window. In each of these three cases there are multiple activities available, which are further presented one by one in the following activity diagrams.

3.4.2. Activity diagram to change the language of the app

The following activity diagram presents the steps a user has to take in order to change the language of the application (Figure 5).

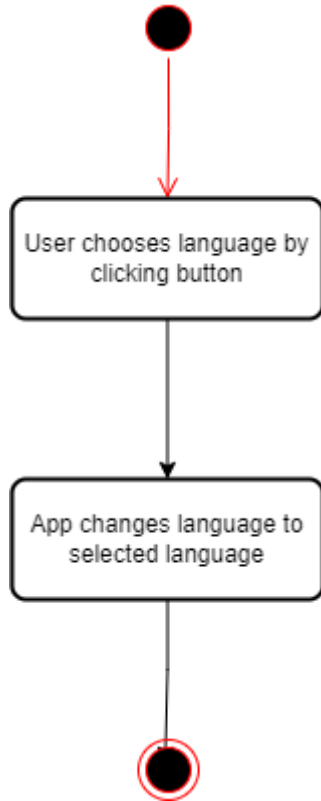


Figure 5

Since the user is able to change the language in the same way in every window of the application, the diagram contains only two steps. First, the user has to click on one of the 2 available buttons that are responsible with changing the language of the app to a different one. These 2 buttons change based on the selected language. So, if the application is currently in English, the two available buttons are “Romanian” and “Italian”. If the application is currently in Romanian, the two available buttons are “English” and “Italian”. If the application is currently in Italian, the two available buttons are “English” and “Romanian”. The second step is the app changing the language to the one that the user has selected.

3.4.3. Activity diagram to login the app

The following diagram presents the steps a user has to perform in order to successfully log in the application (Figure 6).

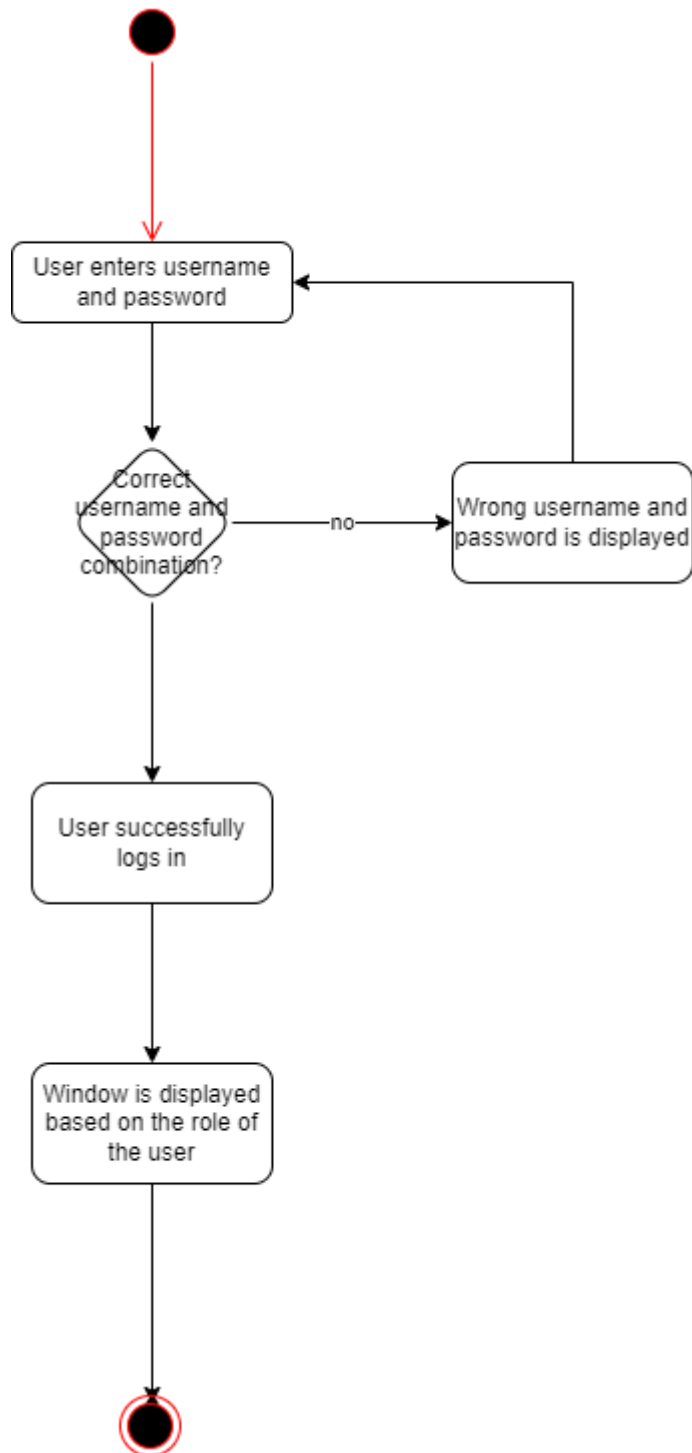


Figure 6

This activity diagram contains the following steps: First, the user has to enter a username and password. Then, if the username and password match an account in the database, the user is logged in and a new window is displayed based on the role of the user. The roles can be employee, manager or administrator. If the username and password combination don't match an account in the database, the user is not

successfully logged in and a text will be displayed that informs the user that he introduced a wrong username and password combination.

3.4.4. Activity diagram to view the cars in service

The following diagram represents the steps that the user must take in order to see what cars are in service and takes into account that only managers and employees can see the cars in service, so it only applies if the user is logged into an account with the role “employee” or “manager” (Figure 7).

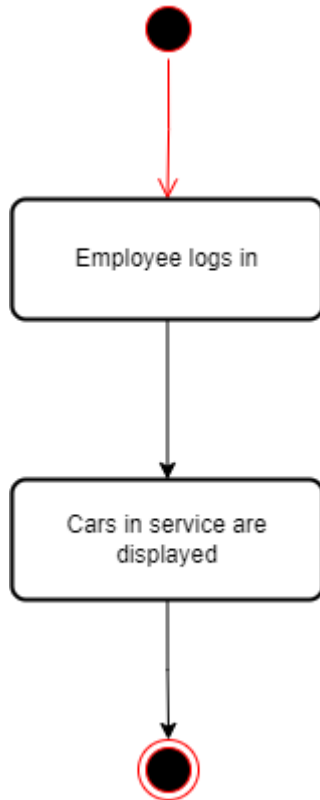


Figure 7

The diagram for the manager is the same as the one showed above, but in stead of the employee log in it is manager log in. The steps that are necessary to be taken in order to see the cars in service are the log in, the table being shown on the window that is shown right after the log in.

3.4.5. Activity diagram to filter the table containing the cars in service

The following activity diagram is used to represent the steps that must be taken in order for the employee/manager to filter the cars in service table (Figure 8).

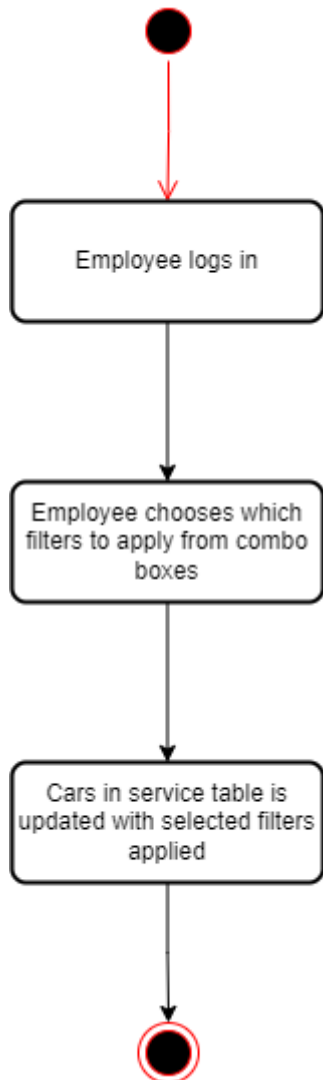


Figure 8

The activity diagram for the manager is the same as the one displayed above. The user must log in either as an employee or as a manager and select which filters to apply on the table from the given combo boxes. The table is then updated with the information containing only the rows that satisfy the given filters.

3.4.6. Activity diagram to search cars by owner

The following diagram is used to show the steps that an employee or manager has to take in order to search all the cars that a person has in the shop (Figure 9).

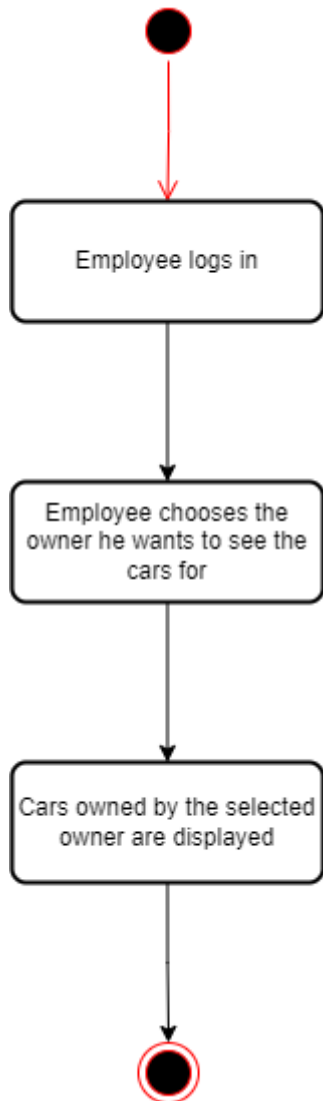


Figure 9

The activity diagram for the manager is the same as the one displayed above. The steps that need to be accomplished are the same as in the case of the filtering activity diagram, except rather than selecting which filters to apply, the user must select the owner he wants to see the car info for.

3.4.7. Activity diagram to save car information in different formats

The activity diagram to save the car information is the same for both employee and manager user types (Figure 10).

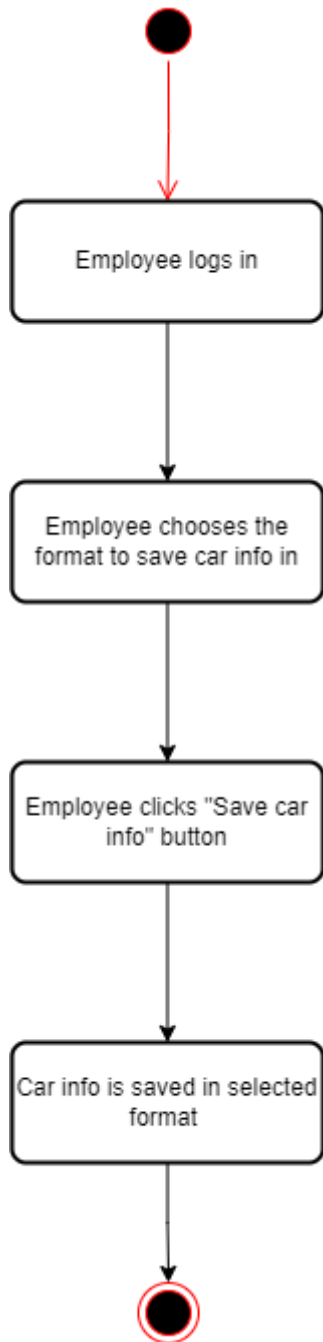


Figure 10

The user must first log in, then select which format he wants the car information to be saved in, then he must click the button to perform the information save action.

3.4.8. Activity diagram to perform CRUD operations on tables

The following activity diagram applies for all CRUD operations on tables and for all types of operations. In the diagram that is presented below, the “Add” operation is exemplified, however both the “Edit” and the “Delete” follow the same format, only the type of operation changes. The possible tables that are available to perform CRUD operation on are: Car, Owner, ServiceLog and User. The activity diagram is presented in Figure 11.

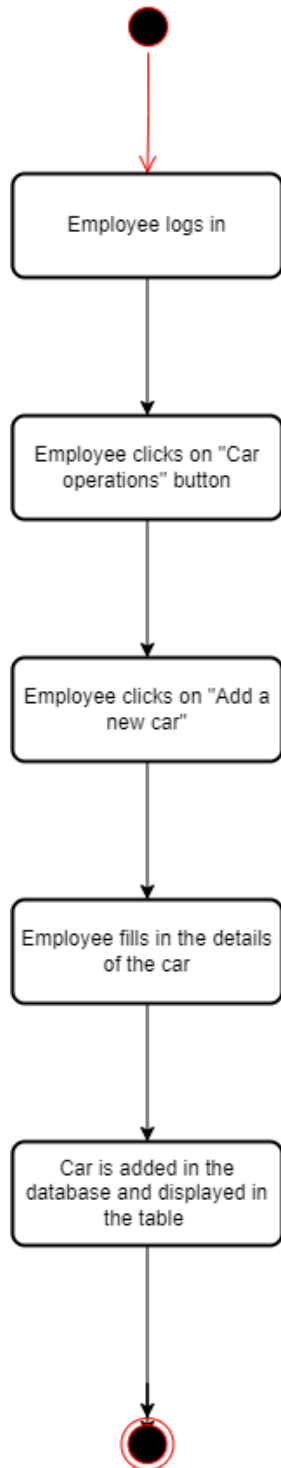


Figure 11

In order to perform operation on the tables Car, Owner and ServiceLog, the user must be an employee. To perform operations on the User table, the user must be an administrator.

The steps necessary to perform an addition are: log in, click on the "operations" button, click on the "add" button, fill in the details and submit.

For the update and delete the flows are the same.

3.4.9. Activity diagram to view car statistics

The following activity diagram shows the necessary steps a manger needs to take in order to see statistics about cars (Figure 12).

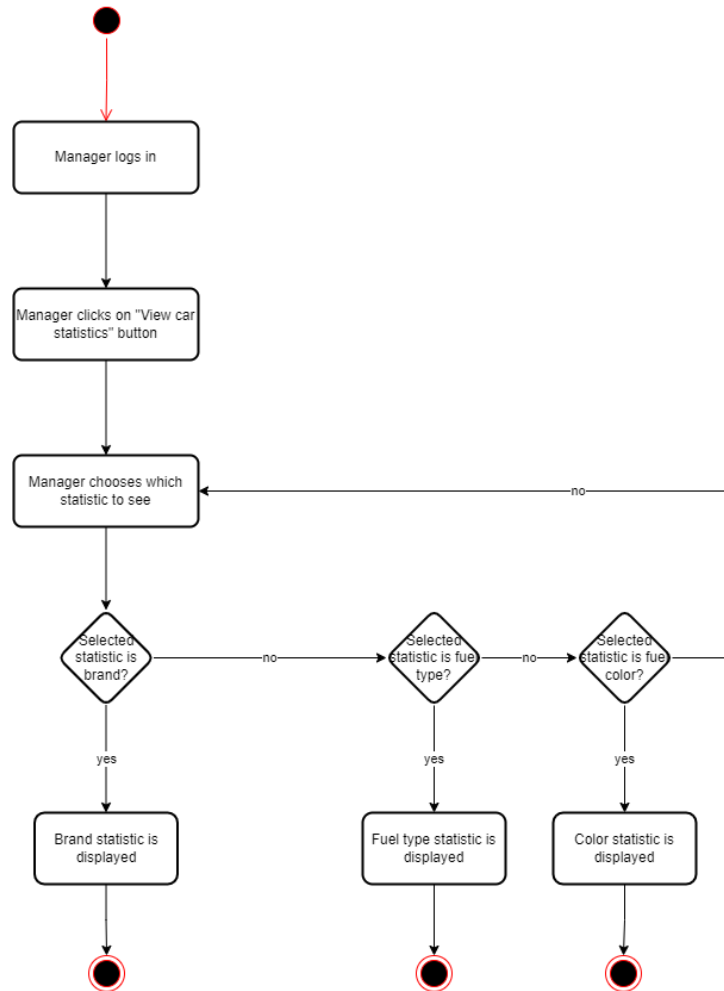


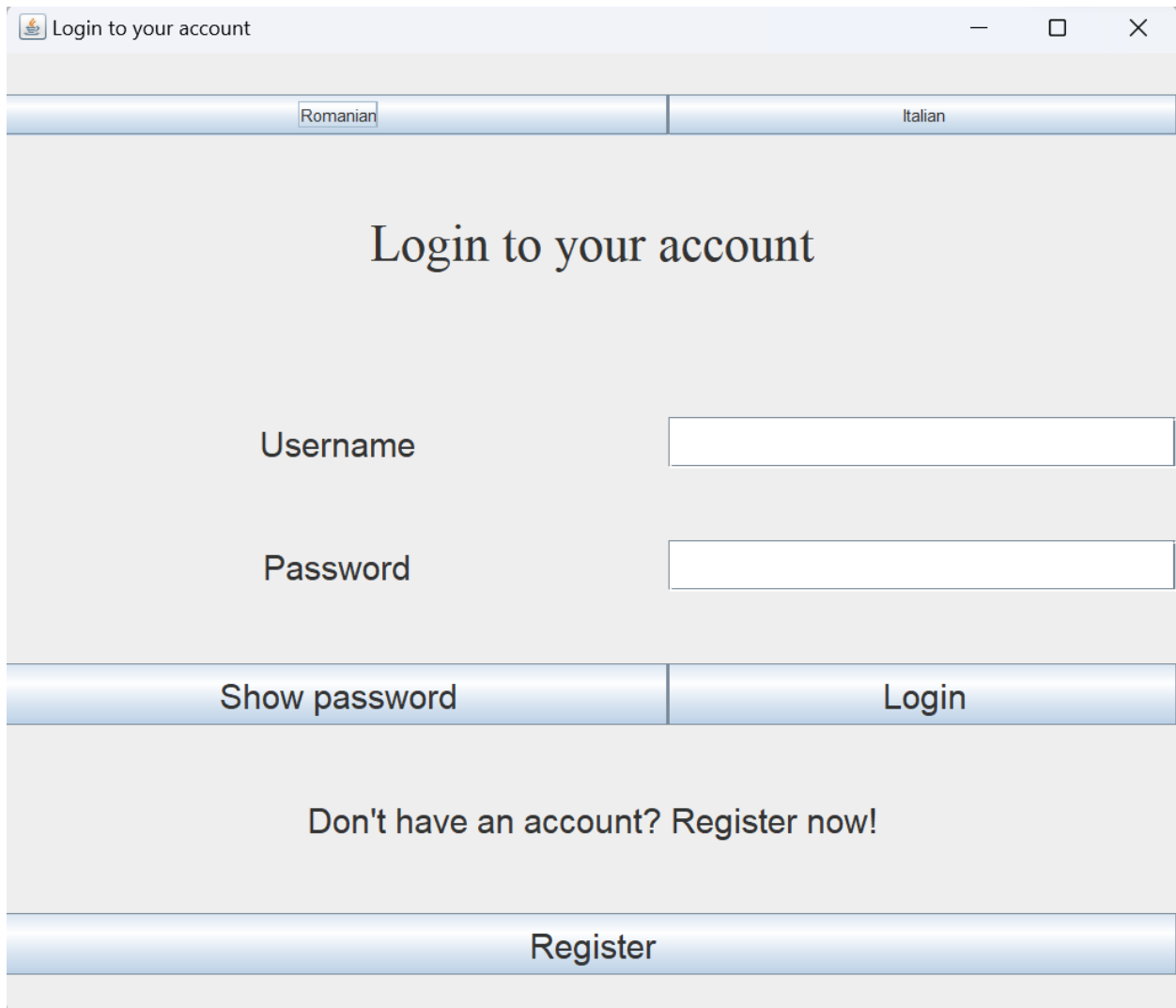
Figure 12

There are 3 available statistics: brand, type and color. In order to show each one, a manager must log in, click on the “statistics” button, choose which statistic he wants to see by pressing one of the 3 available buttons and based on the clicked button, the selected statistic is displayed.

4. Implementation

4.1. Login window

The application was implemented using multiple View classes, which are created and made visible at dynamically at runtime, based on the actions of the user. The application starts with the login view, which presents 2 text fields and a button, along with some labels that prompt the user to enter a username and password in order to login. Along these, a register button is present for users to register. When the register button is pressed, the username and password present in the text fields are taken and a new user is created. A registration will automatically give the role of “employee” to the user. The login view is presented below.



Login to your account

Romanian Italian

Login to your account

Username

Password

Show password Login

Don't have an account? Register now!

Register

Figure 13

4.2. Employee window

If the user logs in as an employee, a new employee view is shown and the login view disappears. In this employee view, a table is presented which contains information about all cars and their owners that are present in the shop at the current time. The records of the shown table can be sorted by brand and/or fuel type and filtered by owner, car brand, fuel type and/or color. The combo boxes which allow the user to select what he wants to filter the table by are generated dynamically from the current cars and owners present in the database. Below the filtering options, there are 4 buttons: “owner operations”, “car operations”, “service operations” and “log out”. Clicking on any button from the first 3 will open a pop-up window which will allow the user to perform Create, Read, Update and Delete actions on the table described by the name of the button. So, for example if the “owner operations” button is clicked, a pop-up window will open which allows the user to perform the operations already described above on the Owner table. The “search cars by owner” opens a pop up window with the cars that are owned by a certain owner. This window is opened whenever an option from the combo box on the right of the label is selected. The “Save car info in format” button saves the car information in the format specified in the combo box. By clicking the “log out” button, the user will be redirected to the login window once again. The employee window is presented below:

The screenshot shows a window titled "Employee window" with a header bar containing "Romanian" and "Italian" tabs. Below the header is a table with the following data:

serviceNumber	carId	brand	modelName	color	fuelType	cnp	carOwner
1	1	mercedes	cls	black	diesel	2	suciu cezar
2	2	nissan	silvia s15	white	gasoline	1	name surname
3	3	nissan	gtr	black	diesel	1	name surname
4	4	bmw	e36	purple	gasoline	4	one one

Below the table, there are several controls:

- "Sort car table by:" with checkboxes for "Brand" and "Fuel Type".
- "Select filters to apply on the table above:" with dropdown menus for "Owner", "Brand", "Fuel Type", and "Color".
- "Search cars by owner:" with a dropdown menu for "Owner".
- A row of buttons: "Owner operations", "Car operations", "Service operations", and "Save car info in format".
- A dropdown menu for "csv" next to the "Save car info in format" button.
- A "Log Out" button at the bottom right.

Figure 14

The if the buttons “owner operations”, “car operations” and “service operations” are clicked, a operations window will be dynamically created. For example, the window generated by clicking “owner

operations” contains a table that shows all the owners that are currently present in the database, as well as 3 buttons from which a new owner can be inserted, a selected owner can be edited or a selected owner can be deleted. If the first 2 buttons are clicked, a new pop-up window will be created and shown, where the user must enter values for each column of an owner, then press “submit”. The owner, car and service operations windows are presented below.

Owner operations

Romanian Italian

Owner operations

cnp	name	surname
1	name	surname
2	suciu	cezar
3	adi	adi
4	one	one

Add a new Owner

Update selected Owner

Delete selected Owner

Figure 15

Car operations

Romanian

Italian

Car operations

id	brand	modelName	color	fuelType
1	mercedes	cls	black	diesel
10	mercedes	g63	black	diesel
2	nissan	silvia s15	white	gasoline
3	nissan	gtr	black	diesel
4	bmw	e36	purple	gasoline
5	bmw	m3	blue	diesel
6	audi	e-tron	yellow	electric
7	mazda	mx5	red	gasoline
8	mazda	rx7	grey	gasoline
9	ferrari	f40	red	gasoline

Add a new Car

Update selected Car

Delete selected Car

Figure 16

Service operations

Romanian
Italian

ServiceLog operations

serviceNu...	carId	brand	modelName	color	fuelType	cnp	carOwner
1	1	mercedes	cls	black	diesel	2	suciu cezar
2	2	nissan	silvia s15	white	gasoline	1	name surn...
3	3	nissan	gtr	black	diesel	1	name surn...
4	4	bmw	e36	purple	gasoline	4	one one

Add a new ServiceLog

Update selected ServiceLog

Delete selected ServiceLog

Figure 17

4.3. Manager window

If the user logs in as a manager, a manager window will be generated which shows the same table, sorting options and filtering options as the ones in the employee window and a log out button which has the same functionality the other log out. Also, the search and save functionalities are the same as in the employee window. The manager window is presented below:

Manager View

Romanian

Italian

Manager View

serviceNumber	carId	brand	modelName	color	fuelType	cnp	carOwner
1	1	mercedes	cls	black	diesel	2	suciu cezar
2	2	nissan	silvia s15	white	gasoline	1	name surname
3	3	nissan	gtr	black	diesel	1	name surname
4	4	bmw	e36	purple	gasoline	4	one one

Sort car table by:

☐ Brand
 ☐ Fuel Type

Select filters to apply on the table above:

Owner

Brand

Fuel Type

Color

Search cars by owner:

Owner

Save car info in format

csv

Car statistics

Log Out

Figure 18

4.4. Admin window

The admin contains a table which shows the users that are present in the User table in the database, a combo box that filters the user table based on the role of the users, a button that takes the user to the Create, Read, Update and Delete operations on users and a log out button. If the “user operations” button is clicked, a pop-up window will open which contains a table with the current users and 3 buttons: “add a new user”, “update selected user”, “delete selected user”. If the first button is clicked, a new pop-up window will open where the current admin will have to complete the text fields corresponding to each column of the user table, then click “submit”. If the second button is clicked, the selected user from the table will be updated with the fields that the current admin will complete (the same procedure as in the insert operation). If the third button is clicked, the selected user from the table will be deleted. For each action performed, the table will refresh automatically both in the admin view and in the pop-up operations view. The admin window and the pop-up window that contains the user operations are presented below:

Admin Window

Romanian

Italian

Admin Window

id	username	password	role
1	admin	admin	admin
2	employee	employee	employee
3	manager	manager	manager

Filter users by:

Role

User operations

Log Out

Figure 19

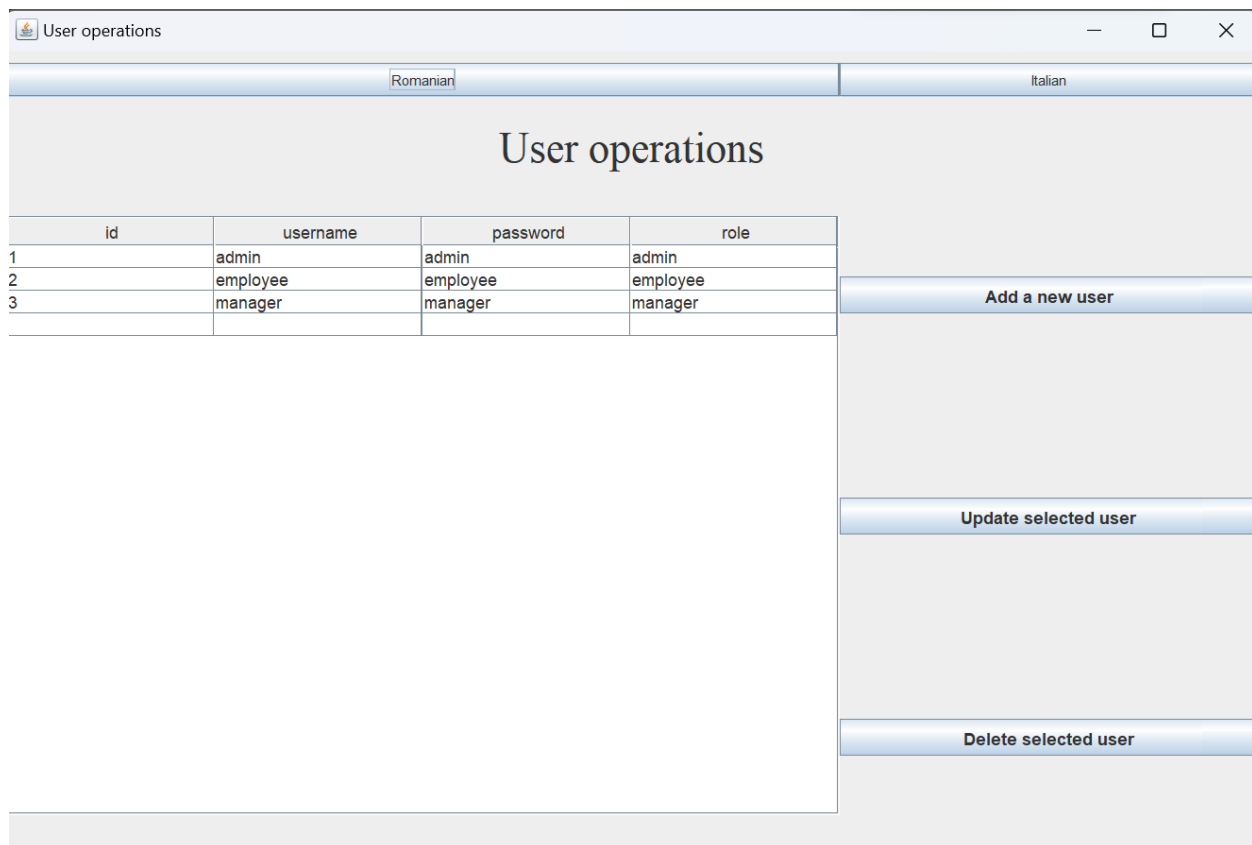


Figure 20