**EJB-JPA Project**

**Glassfish and Wildfly application servers**

**Car Dealership**

Catana Victor-Cristian

SDI – 244

Functional aspects

The application is designed as a car dealership’s website, featuring accesibility to two elements of the business, vehicles and customers. There are two lists on the main menu, as well as submission forms, namely for both vehicles and customers. One can choose to view a customer or a vehicle separately, delete said customer or vehicle, or have a customer buy a vehicle by typing in the vehicle’s ID.

Access to these different features and views is handled by three Servlets:

* MainServlet – for the home page, which offers the lists with details about the customers and vehicles in the application as well as the forms for creating customers and vehicles
* VehicleServlet – for the vehicles page, which displays the details of the selected vehicle and offers the option of deleting the vehicle
* CustomerServlet – for the customers page, which displays the details of the selected customer, offers the option of deleting said customer and also offers the option of buying a vehicle by providing its ID

MainServlet only features a doGet method, while the other two servlets both feature two different methods, doGet and doPost, with the help of which they provide functionalities to the user.

As mentioned above, two entities are present in the application, Vehicle and Customer. Each of the two is handled by a bean, which in turn implements a service interface which is afterwards injected into the servlets either by EJB injection or JNDI.

Technical aspects

The two entities the application is based on are handled, as said before, by two Beans which implement two interfaces. The beans have the following structure:

* Vehicles
  + void addVehicle (String make, String model, double price, String registration)
  + void deleteVehicle(long id)
  + Vehicle find(long id)
  + List<Vehicle> findAll()
* Customers
  + void addCustomer (String first\_name, String last\_name)
  + void deleteCustomer(long id)
  + Customer find(long id)
  + Vehicle findVehicle(long id)
  + void addVehicleToCustomer(long customer, long vehicle)
  + void removeVehicleFromCustomer(long vehicle)
  + List<Customer> findAll()

As far as the database is concerned, the relationship between the two tables, Vehicle and Customer, is unidirectional one to many, where one customer can have many vehicles, but one vehicle can only belong to one customer at one specific moment in time. The Vehicle contains a foreign key to the customer ID which is stored in the table under the id\_customer column.

The database used for the application is MySQL, version 8, and handled in the application by using the MySQL Java Connector.

Deployment of the application is done by using two different application servers, GlassFish and WildFly, each with their own dedicated project because they both need specific configuration details. In the development and deployment of the application for these two application severs, JDK 17 with Glassfish 7 and Wildfly 27 were used. Maven was used for building and dependency management purposes.