deti universidade de aveiro
departamento de eletrónica,
telecomunicações e informática

Projeto 2 - TPW

Trabalho de Tecnologias e Programação Web
Professor: Hélder Zagalo

João Viegas nº113144, Jorge Domingues nº113278, João Monteiro nº114547

Abstract

CarStand is a web-based system designed to manage a vehicle stand. It allows users to check what brands are available in the stand and view the groups that own those brands. For customers, it is possible to explore available vehicles in the stand, add them to their favourites and express interest in a certain vehicle. Administrators, on the other hand, can accept or decline requests from the customers and can also add, remove or edit vehicles.

Keywords: *Car; Stand; Customer; Brand; Angular; Tailwind; Django; TypeScript.*

---

1. Local execution

In our projects we have two folders: **CarStand** and **carStand-app**, the first runs the Django Project and the second the Angular.

1.1. How to Run:

Open two terminals in the root of the project. On the first one do:

```
python3 -m venv venv
source venv/bin/activate
pip install -r requirements.txt
cd CarStand/
python3 manage.py makemigrations app
python3 manage.py migrate
python3 manage.py runserver
```

On the second do:

```
cd carStand-app/
ng serve
```

In the browser navigate to the following url: http://localhost:4200/home.

2. Users

2.1. User access

The system includes two types of users: **non-logged users** and **logged users**. Logged users have additional functionalities compared to non-logged users, including:

- Demonstrating interest in cars and motorcycles.
- Adding vehicles to their list of favourites.
- Editing their profile.

User Accounts:

- **User 1:**

```
Username: joaoaugusto
Password: joaocars12345

```

- **User 2:**

```
Username: antoniojose32
Password: joseaveiro27

```

2.2. Administrator Access

The system administrator is responsible for managing purchase requests and vehicles. Their responsibilities include:

- Accepting or denying purchase requests submitted by users who demonstrate interest in a vehicle.
- Managing vehicles in the system:
  - Adding new vehicles.
  - Deleting vehicles.
  - Updating vehicle details.
  - Adding new models.

Administrator Account:

```
Username: admin
Password: admin
```

3. Deployment

3.1. Backend

The system backend was deployed using PythonAnywhere, following the same approach we were instructed to use in the first project.

To access the web page, use the following link: https://joaov2345678.pythonanywhere.com/index/

3.2. Frontend

Deploying the frontend presented some challenges. Initially, we attempted to use platforms such as Heroku, Render, Railway, and GitHub Pages. However, the best solution we identified was Vercel, where the system is now accessible.

Frontend deployment can be accessed at: https://carstand-app.vercel.app/home

4. Authentication

Our authentication system is based on caching user information and reusing Django's authentication framework to maintain and ensure security.

The credentials required for login and sign-up are sent via the body of a POST request. During sign-up, if the provided data is valid, a new 'User' and 'Profile' will be created through the 'UserSerializer'. These will be cached with the key derived from 'getUniqueID(request)'. During login, the system verifies whether the username and password are valid using Django's 'authenticate' method. Subsequently, the system checks if the user is active by calling the 'checkAuthStatus' function from 'auth.service.ts'.

The 'checkAuthStatus' function is responsible for verifying if a user is active. If they are, their information is stored in a 'BehaviorSubject'. This structure emits data to its subscribers, allowing components to fetch

user information only once while enabling reuse across various components without the need for repeated fetch requests.

Additionally, logging out removes the cached information of the user associated with the current session. This ensures that the session-specific data is properly cleared, maintaining consistency and security without affecting other users in the cache.

5. Implementation

5.1. Components

- **Button to Go-Back**
  - Provides a button that allows navigating to the previous page. It is reusable across different views.
- **Cards**
  - Displays items in a card layout. Each card contains essential information about the item. There are three different types of cards in the system.
  - Two of these cards components, can be named as 'main' cards, since they show key information about components, such as brands, for example, in their associated page.
  - There's also 'secondary' cards for all the main components, that are used when these are shown in another component view (e.g. showing all the brand of a given group).
- **Details**
  - Shows detailed information about a specific vehicle, group or brand when a user selects it from a list of cards.
- **Brands**
  - Lists the available brands. Details are accessible by clicking each card.
- **Carousel**
  - Provides a carousel view to display vehicles in a promotional form.
- **Cars**
  - Lists the available cars. Details are accessible by clicking each card.
- **Create Vehicle**
  - Handles the creation of a new vehicle. This component is typically used by administrators.
- **Create-Vehicle-Model**
  - Similar to the `Create Vehicle` component but specifically designed for creating new vehicle models, that can be later used to create a new car or motorcycle.
- **Desired Vehicles**
  - Allows users to visualize the list of vehicles they have shown interest in.
- **Edit Vehicle**
  - Enables administrators to edit details of an existing vehicle.
- **Favourites**
  - Allows users to visualize their list of favorite vehicles.
- **Footer**
  - Contains the footer information displayed at the bottom of each page, such as contact details or links.

- **Groups**
  - Lists the available groups. Details are accessible by clicking each card.
- **Home**
  - The main page of the application, providing an overview of the system.
- **Login**
  - Handles user authentication, allowing users to log in with their credentials.
- **Manager Table**
  - Displays a table of vehicles or purchase requests for the administrator to review and manage.
- **MotorBikes**
  - Lists the available motorcycles. Details are accessible by clicking each card.
- **Navbar**
  - Provides navigation links for moving between the different views of the application.
- **Profile**
  - Allows users to view and edit their profile information.
- **Purchased Vehicles**
  - Allows users to visualize the list of vehicles they have shown interest in and that were accepted by the system administrator.
- **Search Bar**
  - Implements a search functionality to filter and find vehicles, brands or groups.
- **Search Bar Table**
  - A search bar used to search items in the `Manager Table` component.
- **Sign Up**
  - Handles user registration, allowing new users to create accounts.
- **Filter Vehicles**
  - Provides filtering functionality for vehicles, enabling users to narrow down search results based on criteria such as price, number of doors or if it is eletric or no.

5.1.1. Interconnection Between Components

- **Shared Components:**

  Components such as `Navbar`, `Footer`, and `Go-Back Button` are shared across multiple pages to provide consistent navigation and layout.

  Inside the `Home Page`, the `Carousel` component is included when logging in as a regular user, or without login at all. When logging in as an administrator, that information is replaced with the `Manager Table` component, where the admin can accept or deny purchase requests from different users.

  For each main component (`Car`, `Motorbike`, `Group` and `Brand`) there is a search bar, that allows users to search items more quickly. For each model, it is also possible to apply a search using more specific filters (e.g., price range).

  When seeing detailed information about a `Group` it is possible to see which brands are owned by that specific group.

  In the same way, when seeing detailed information about a `Brand` it is possible to see the vehicles that were produced by that same brand.

  Each of the `Desired Vehicles`, `Purchased Vehicles` and `Favourites` components is accessible through the `Profile` view, when logging in as an user who is not an administrator.

6. Endpoints

6.1. Authentication

- **'api/signup'**
  - Endpoint to create an account.
- **'api/login'**
  - Endpoint to login with an existent account.
- **'api/logout'**
  - Endpoint to logout of an account.
- **'api/isAuth'**
  - Endpoint to check if the current user is logged in. If so, his information his fetched.

6.2. Vehicles

- **'api/cars'**
  - Endpoint to list all cars.
- **'api/car'**
  - Endpoint to get one car.
- **'api/cars/create'**
  - Endpoint to create a new car.
- **'api/cars/update'**
  - Endpoint to edit a car.
- **'api/cars/delete/<int:id>'**
  - Endpoint to delete a car.
- **'api/motos'**
  - Endpoint to list all motos.
- **'api/moto'**
  - Endpoint to get one moto.
- **'api/motos/create'**
  - Endpoint to create a new moto.
- **'api/motos/update'**
  - Endpoint to edit a moto.
- **'api/motos/delete/<int:id>'**
  - Endpoint to delete a moto.
- **'api/models/<str:vehicle_type>/'**
  - Endpoint to get all models of a specific type (car or motorbike).
- **'api/carmodel/create'**
  - Endpoint to create a new model (only possible as an administrator).
- **'api/vehicles/<int:vehicle_id>/<str:type>/status/'**
  - Endpoint to get the current status of a vehicle (if it is purchased or not).

6.3. Brands

- **'api/brands'**
  - Endpoint to list all brands.
- **'api/brand'**
  - Endpoint to get one brand.
- **'api/brands/<int:brand_id>/vehicles'**
  - Endpoint to get all vehicles of a given brand.

6.4. Groups

- **'api/groups'**
  - Endpoint to list all groups.
- **'api/group'**
  - Endpoint to get one groups.
- **'api/groups/<int:id>/brands'**
  - Endpoint to get all brands of a given group.

6.5. Search

- **'api/search/<str:type>/'**
  - Endpoint that allows search based by name of every main component.
- **'api/<str:type>/filters'**
  - Endpoint that allows search based on filters for vehicles.

6.6. Profile

- **'api/profile/'**
  - Endpoint that allows to obtain information about an user, but also to edit it.
- **'api/profile/desired/'**
  - Endpoint that allows to fetch the vehicles an user desires.
- **'api/vehicles/<int:vehicle_id>/<str:type>/interest/'**
  - Endpoint that allows an user to show interest in a vehicle.
- **'api/profile/purchased/'**
  - Endpoint that allows to fetch the vehicles an user purchased.
- **'api/favorites/<str:type>/get/'**
  - Endpoint that allows to fetch the vehicles an user added as favourite.
- **'api/favorites/<str:type>/'**
  - Endpoint that allows an user to add a vehicle to his favourites.

6.7. Administrator Management

- **'api/vehicles/approval/'**
  - Endpoint to get all vehicles that are waiting for manager approval.
- **'api/vehicles/<int:vehicle_id>/<int:profile_id>/<str:type>/approve/'**
  - Endpoint that allows a manager to approve a request to buy a vehicle.
- **'api/vehicles/<int:vehicle_id>/<int:profile_id>/<str:type>/negate/'**
  - Endpoint that allows a manager to negate a request to buy a vehicle.

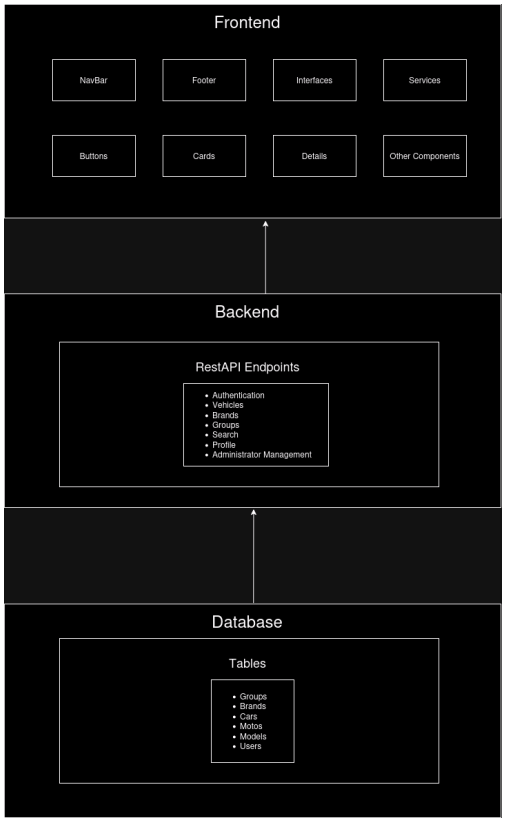

Figure 1. System Architecture Diagram

7. Architecture

Here is a overall view of the architecture of our project