UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Faculty of Engineering.

Subject: Advanced Programming

Teacher: Carlos Andres Sierra

Project: WorkShops

Daniel Santiago Pérez **20231020203**

Bogotá, D.C.

2024.

Workshop 2

Program Description:

The program aims to develop an internal platform for a vehicle constructor company. The main objective is to create a catalog of vehicles that allows users to manage information about different types of vehicles, including their specific features such as engine, chassis, model, fuel consumption, among others. Additionally, a command-line menu should be implemented to create and display vehicles and engines.

User Stories:

- As a user, I want to be able to create new vehicles specifying their type, engine, chassis, model, and other relevant details.
- As a user, I want to be able to register different types of engines with their respective specifications.
- As a user, I want to calculate the fuel consumption of a vehicle based on its engine characteristics.
- As a user, I want to see all vehicles registered in the catalog.
- As a user, I want to see all engines registered in the system.
- As a user, I want to have an intuitive command-line interface to navigate and use the program's functions, such as navigating menus and filtering searches.

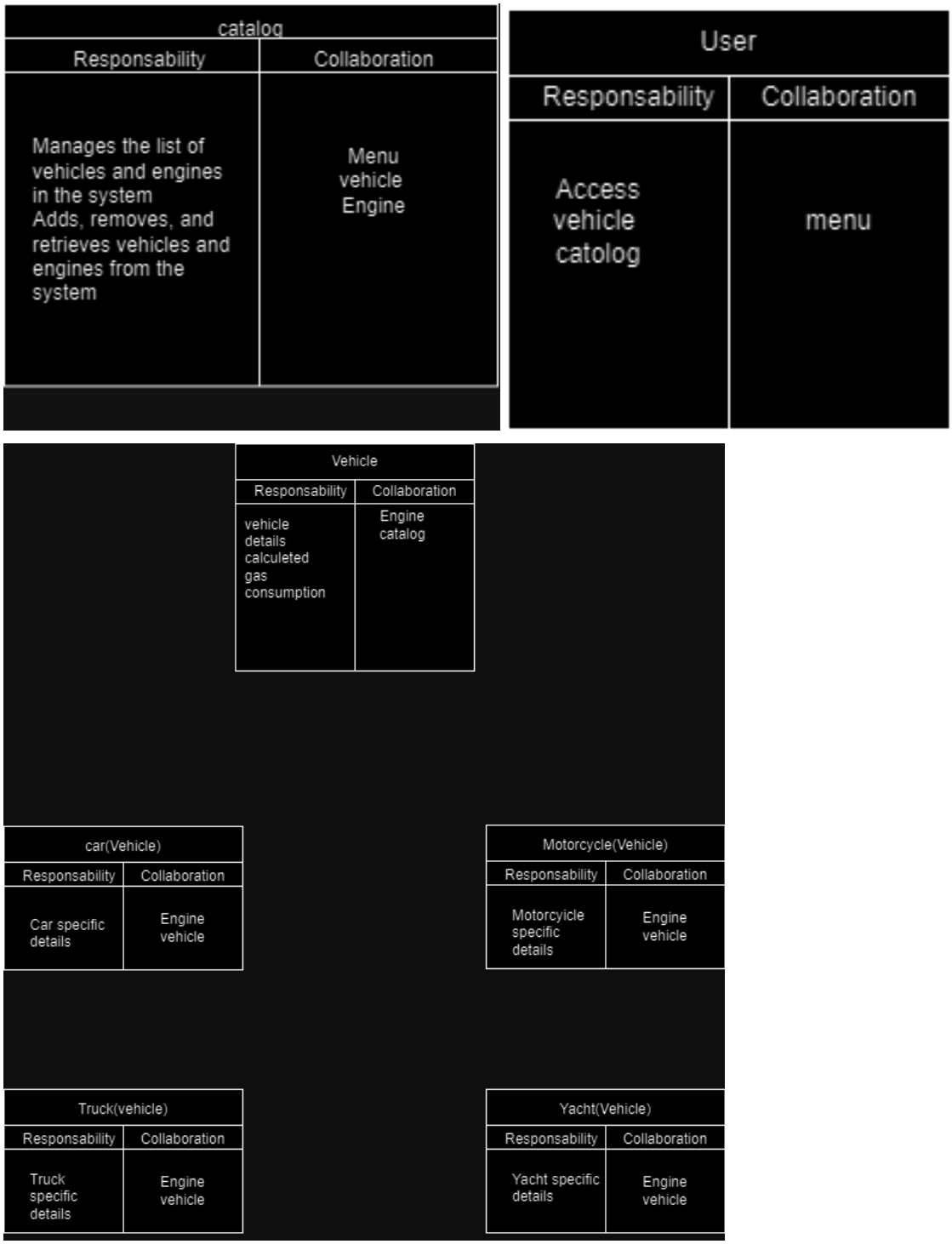
Stakeholders:

- *Company CTO:* Responsible for the strategic direction of the project and making important decisions.
- *Developers:* Responsible for implementing the program's functionalities and meeting the established requirements.
- *End Users:* Employees of the vehicle constructor company who will use the platform to manage the vehicle catalog.

Main Entities:

- *Vehicle:* Represents a vehicle registered in the catalog, with properties such as type, engine, chassis, model..
- *Engine:* Represents an engine registered in the system, with properties such as name, type, potency, weight-
- *Command-Line Menu:* Interface that allows users to interact with the program through text commands in the console.

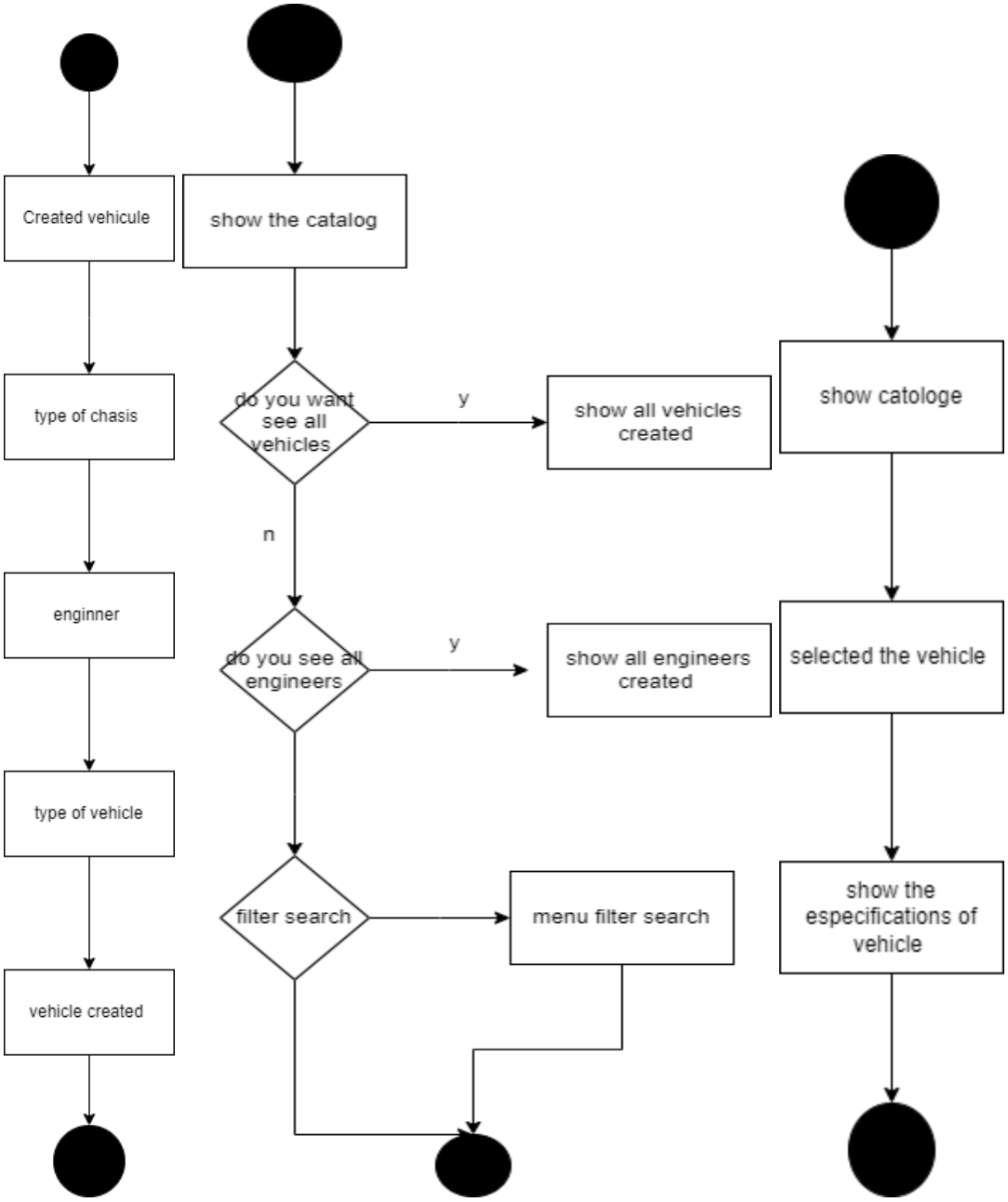
CRC Cards

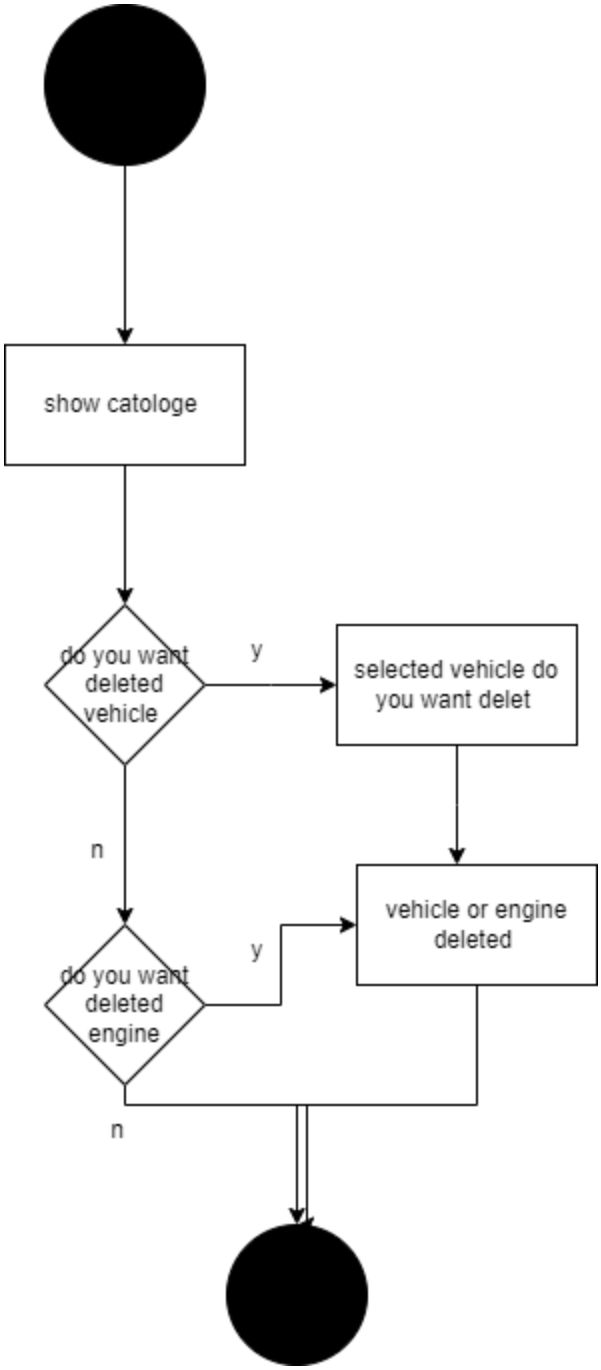

catalog
Responsability
Collaboration
Manages the list of vehicles and engines in the system
Adds, removes, and retrieves vehicles and engines from the system
Menu
vehicle
Engine
User
Responsability
Collaboration
Access vehicle catolog
menu
Vehicle
Responsability
Collaboration
vehicle details calculeted gas consumption
Engine catalog
car(Vehicle)
Responsability
Collaboration
Car specific details
Engine vehicle
Motorcycle(Vehicle)
Responsability
Collaboration
Motorcyicle specific details
Engine vehicle
Truck(vehicle)
Responsability
Collaboration
Truck specific details
Engine vehicle
Yacht(Vehicle)
Responsability
Collaboration
Yacht specific details
Engine vehicle

Menu	
Responsability	Collaboration
Manage comand-line menu	vehicle Engine

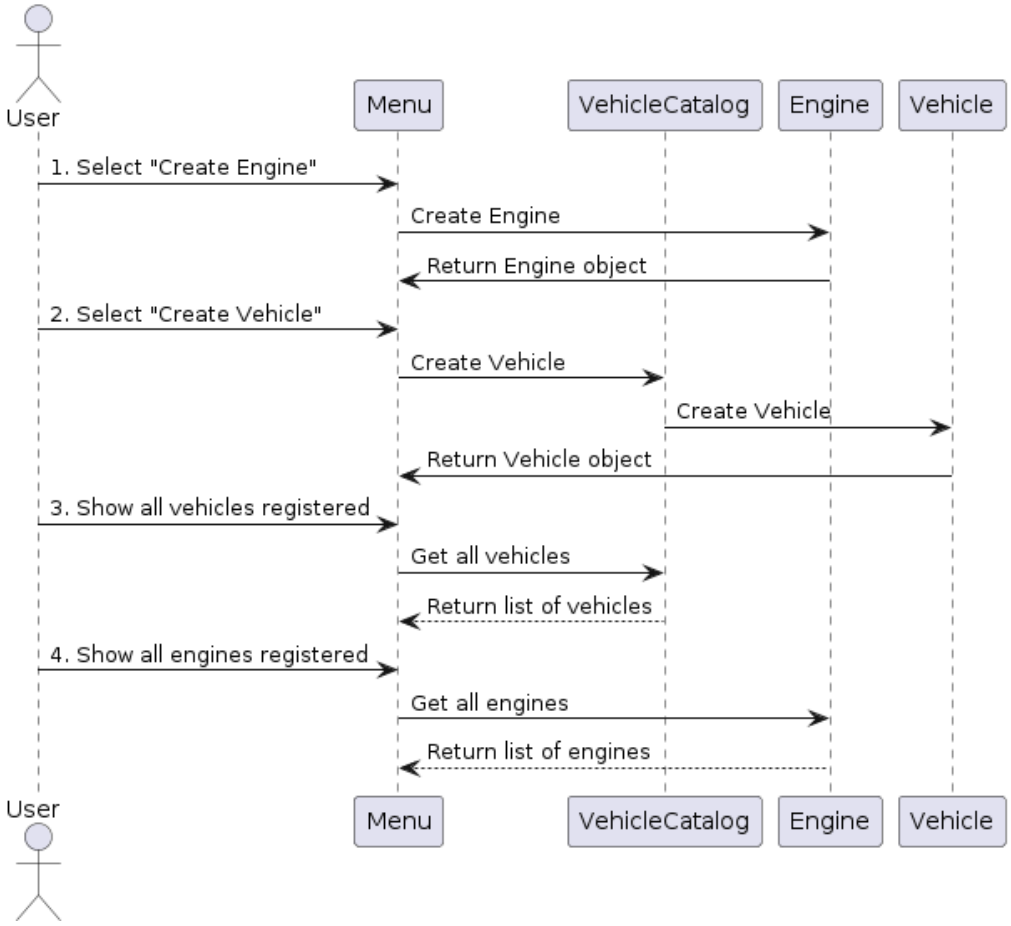
Engine	
Responsability	Collaboration
Engine details calculeted gas consumption	Vehicle catalog

Activity diagrams

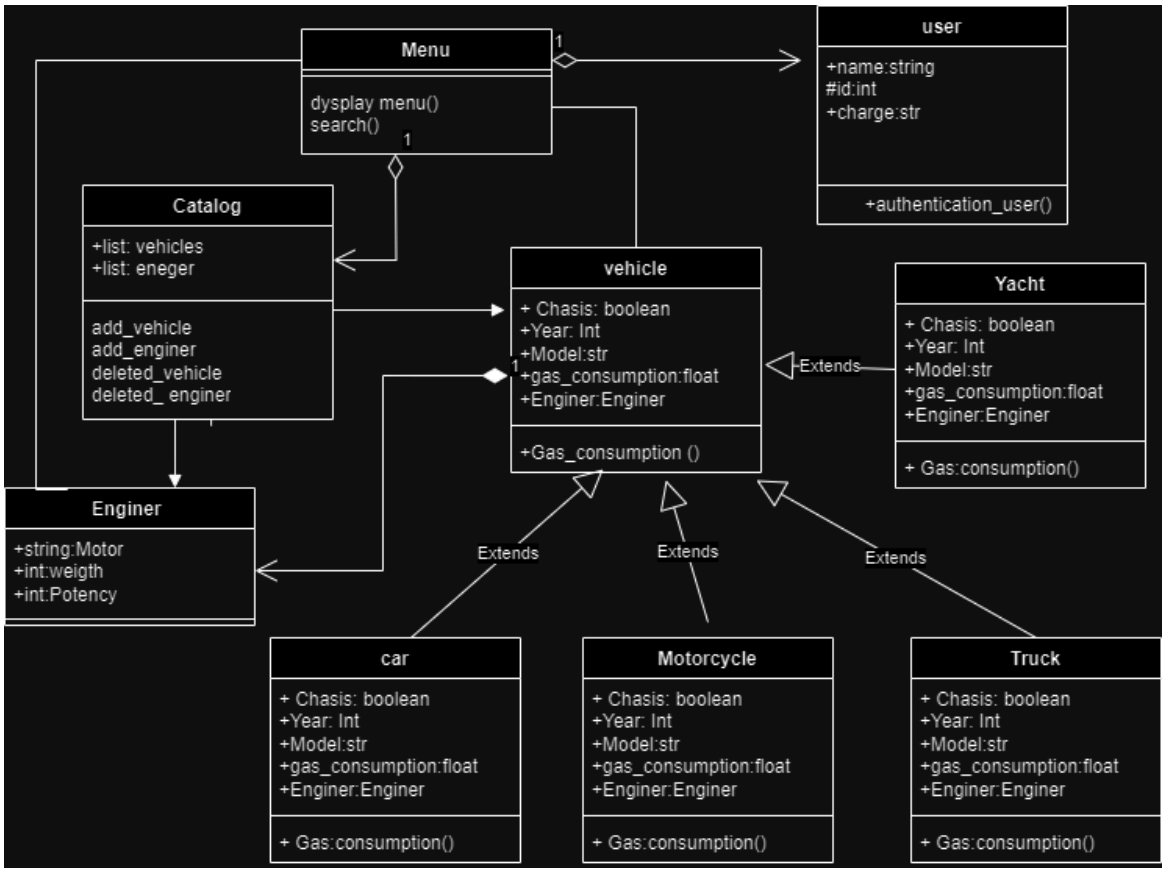

Created vehicule
type of chasis
enginner
type of vehicle
vehicle created
show the catalog
do you want see all vehicles
y
show all vehicles created
n
do you see all engineers
y
show all engineers created
filter search
menu filter search
show catologe
selected the vehicle
show the especifications of vehicle


show catologe
do you want deleted vehicle
y
selected vehicle do you want delet
n
do you want deleted engine
y
vehicle or engine deleted
n

Sequence Diagram

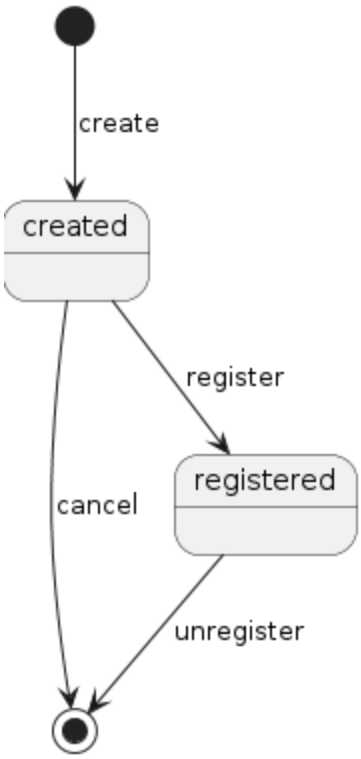


Class Diagram

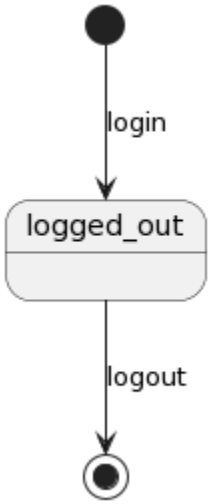


State Diagram

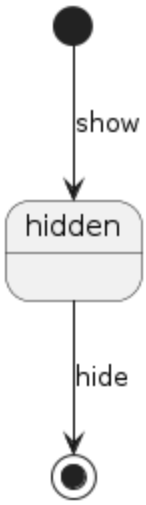
vehicle and engine state



user state



menu state


show
hidden
hide

Workshop 3

User stories

1. Create New Vehicles

   As a user, I want to be able to create new vehicles specifying their type, engine, chassis, model,and other relevant details, So that I can accurately add new vehicles to the catalog.
2. Register Engines

   As a user,I want to be able to register different types of engines with their respective specifications, So that I can accurately define engine details for vehicles.
3. Calculate Fuel Consumption

   As a user, I want to calculate the fuel consumption of a vehicle based on its engine characteristics, So that I can understand the efficiency of each vehicle.
4. View All Vehicles

   As a user, I want to see all vehicles registered in the catalog, So that I can browse the entire inventory of vehicles.
5. View All Engines

   As a user, I want to see all engines registered in the system,
   So that I can browse all engine options available.
6. Command-Line Interface

   As a user, I want to have an intuitive command-line interface to navigate and use the program's functions, such as navigating menus and filtering searches, So that I can efficiently use the platform without a graphical user interface.

Acceptance Criteria

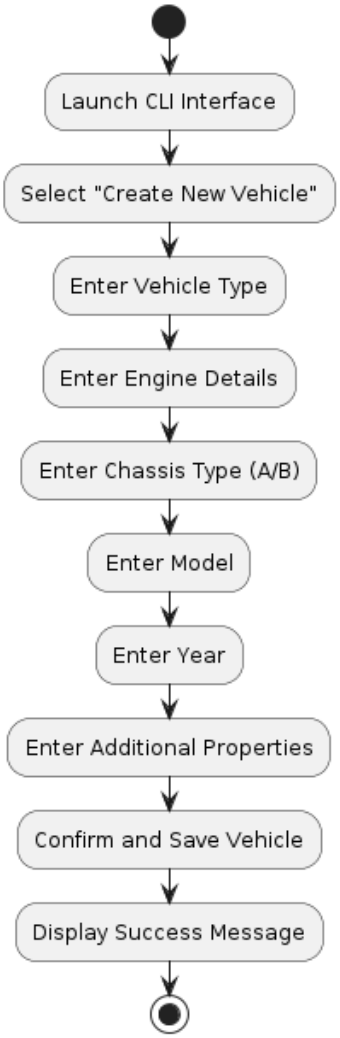
- The platform must allow specifying the vehicle type (car, truck, yacht, motorcycle).
- The platform must require entering the engine, chassis type (A or B), model, and year.
- Additional fields relevant to each vehicle type should be available for input.
- The platform must allow entering the engine name, type, potency, and weight.
- The engine registration must validate that all necessary properties are provided and correctly formatted.
- The platform should provide a method to calculate gas consumption using the formula: Gas Consumption = 1.1 × engine.potency + 0.2 × engine.weight − (0.3 if chassis is A or 0.5 if chassis is B) Gas Consumption=1.1×engine.potency+0.2×engine.weight−(0.3 if chassis is A or 0.5 if chassis is B)

- The calculation should be automatically available after entering engine details and chassis type.
- The platform should display a list of all registered vehicles.
- Each vehicle listing should include its type, engine, chassis, model, gas consumption, and year.
- The platform should display a list of all registered engines.
- Each engine listing should include its name, type, potency, and weight.
- The command-line interface should provide clear instructions and prompts for each action.
- Users should be able to navigate through menus, create and view records, and filter searches using intuitive commands.
- The interface should handle errors gracefully, providing helpful messages and guidance for correcting input mistakes.

Activity Diagrams

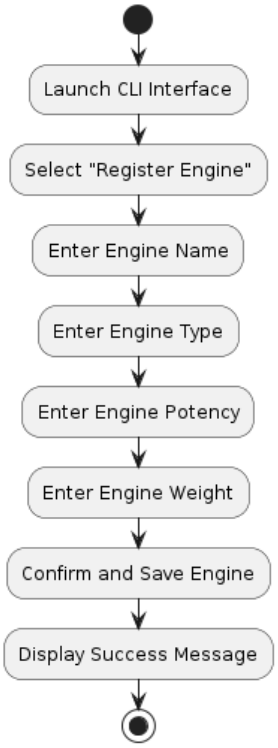
Creating a new vehicle

- Launch CLI Interface: The user initiates the command-line interface.
- Select "Create New Vehicle": The user chooses the option to create a new vehicle.
- Enter Vehicle Type: The user specifies the type of vehicle (car, truck, yacht, motorcycle).
- Enter Engine Details: The user inputs the engine details including name, type, potency, and weight.
- Enter Chassis Type (A/B): The user specifies whether the chassis is type A or B.
- Enter Model: The user enters the model of the vehicle.
- Enter Year: The user specifies the manufacturing year of the vehicle.
- Enter Additional Properties: The user inputs any additional properties specific to the vehicle type.
- Confirm and Save Vehicle: The user confirms the entered details and saves the new vehicle entry.
- Display Success Message: The system displays a success message indicating that the vehicle has been successfully added to the catalog.



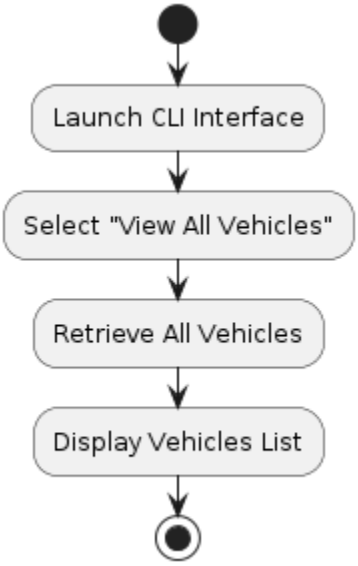
Register Engine

- Launch CLI Interface: The user starts the command-line interface.
- Select "Register Engine": The user selects the option to register a new engine.
- Enter Engine Name: The user inputs the name of the engine.
- Enter Engine Type: The user specifies the type of the engine.
- Enter Engine Potency: The user inputs the potency of the engine.
- Enter Engine Weight: The user specifies the weight of the engine.
- Confirm and Save Engine: The user confirms the details and saves the new engine entry.
- Display Success Message: The system displays a success message indicating that the engine has been successfully registered.



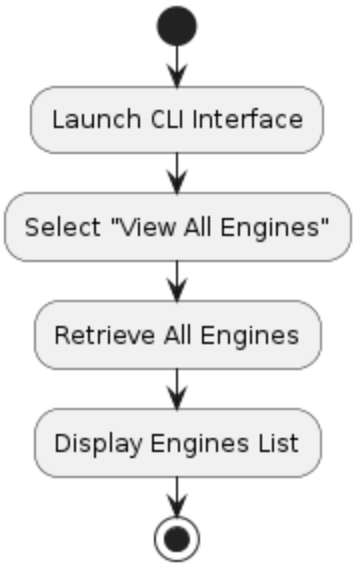
View all vehicles

- Launch CLI Interface: The user initiates the command-line interface.
- Select "View All Vehicles": The user selects the option to view all vehicles in the catalog.
- Retrieve All Vehicles: The system retrieves all vehicle entries from the catalog.
- Display Vehicles List: The system displays a list of all registered vehicles, showing details like type, engine, chassis, model, gas consumption, and year.



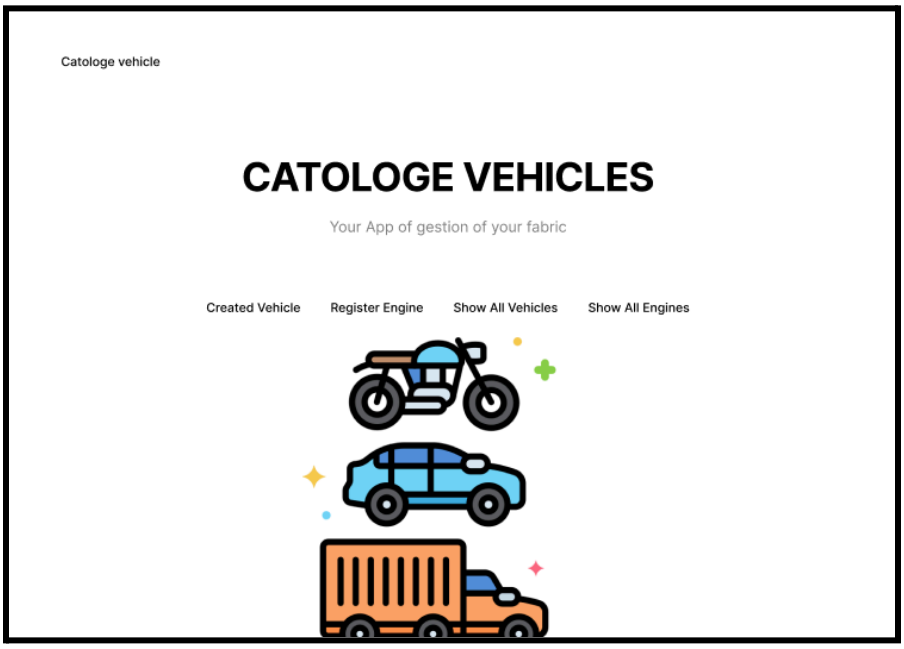
View all engines

- Launch CLI Interface: The user starts the command-line interface.
- Select "View All Engines": The user selects the option to view all registered engines.
- Retrieve All Engines: The system retrieves all engine entries from the database.
- Display Engines List: The system displays a list of all registered engines, showing details like name, type, potency, and weight.

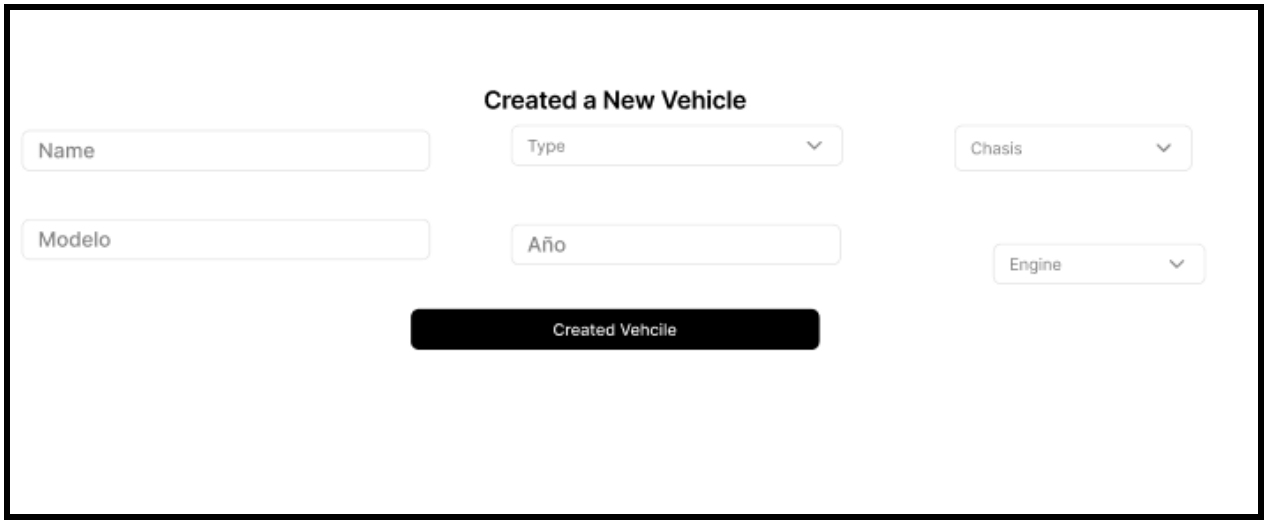


IU Prototypes

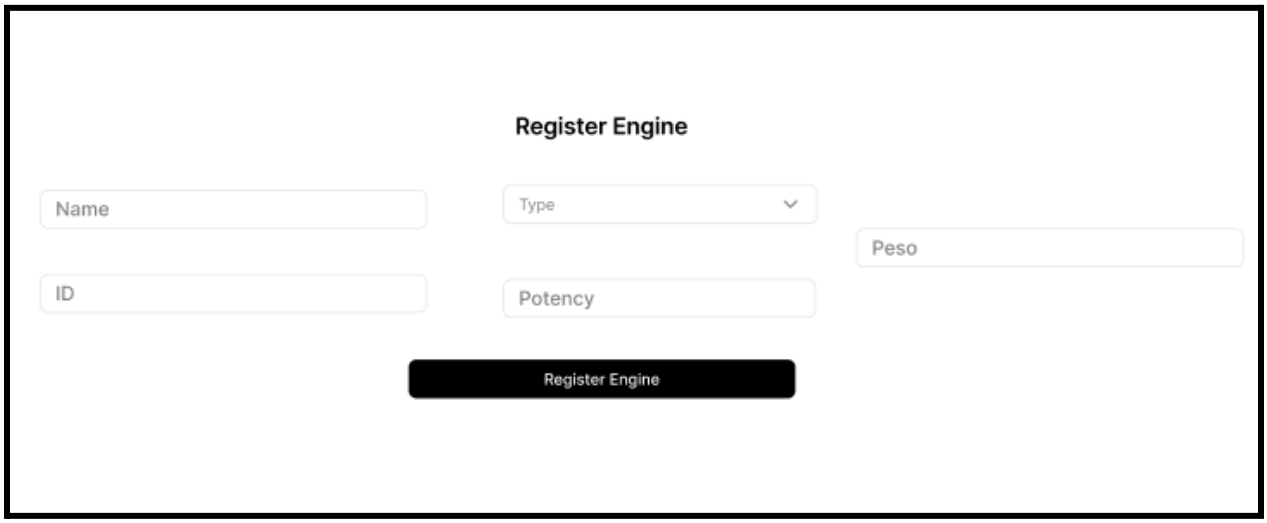
CLI INTERFACE



CREATED A NEW VEHICLE



REGISTER ENGINE



SHOW ALL VEHICLES

All Vehicles

Name	Chasis	Modelo	Año	Type	Engine	consumo gas
carens	A	kia	2022	carro	V8	1200
Toyota	B	Lancha	2012	Yate	motorlancha	1500

SHOW ALL ENGINES

All Engines

Name	Id	Potencia	Peso	Type
V8	1428	1200	1000	gas
Motorlancha	1235	1400	800	electrico

Data Structure JSON

1.Creating New Vehicles

json

```
{
  "vehicle_type": "car",
  "engine": {
    "name": "V8 Turbo",
    "type": "Petrol",
```

```
    "potency": 400,
    "weight": 350
  },
  "chassis_type": "A",
  "model": "Sedan 2024",
  "year": 2024,
  "additional_properties": {
    "color": "Red",
    "seats": 5
  }
}
```

Explanation:

vehicle_type: Specifies the type of vehicle (e.g., car, truck, yacht, motorcycle).
engine: An object containing details about the engine.
name: The name of the engine.
type: The type of engine (e.g., Petrol, Diesel).
potency: The engine's potency.
weight: The engine's weight.
chassis_type: Specifies the chassis type (A or B).
model: The model of the vehicle.
year: The manufacturing year of the vehicle.
additional_properties: An object containing any additional properties specific to the vehicle type (e.g., color, seats).// tentative

Response JSON:

json

```
{
  "status": "success",
  "message": "Vehicle created successfully",
  "vehicle_id": "12345"
}
```

Explanation:

status: Indicates the result of the operation (e.g., success, error).
message: A descriptive message regarding the result.
vehicle_id: A unique identifier for the newly created vehicle.

2. Registering Engines

json

```
{
  "name": "V8 Turbo",
```

```
  "type": "Petrol",
  "potency": 400,
  "weight": 350
}
```

Explanation:

- name: The name of the engine.
- type: The type of engine (e.g., Petrol, Diesel).
- potency: The engine's potency.
- weight: The engine's weight.

Response JSON:

json

```
{
  "status": "success",
  "message": "Engine registered successfully",
  "engine_id": "67890"
}
```

Explanation:

- status: Indicates the result of the operation (e.g., success, error).
- message: A descriptive message regarding the result.
- engine_id: A unique identifier for the newly registered engine.

3. Viewing All Vehicles

json

```
{
  "action": "view_all_vehicles"
}
```

Explanation:

- action: Specifies the requested action, which in this case is to view all vehicles.

Response JSON:

json

Copiar código

```
{
  "status": "success",
  "vehicles": [
    {
      "vehicle_id": "12345",
      "vehicle_type": "car",
      "engine": {
```

```
      "name": "V8 Turbo",
      "type": "Petrol",
      "potency": 400,
      "weight": 350
    },
    "chassis_type": "A",
    "model": "Sedan 2024",
    "year": 2024,
    "gas_consumption": 450.5,
    "additional_properties": {
      "color": "Red",
      "seats": 5
    }
  },
  {
    "vehicle_id": "12346",
    "vehicle_type": "truck",
    "engine": {
      "name": "Diesel XL",
      "type": "Diesel",
      "potency": 600,
      "weight": 800
    },
    "chassis_type": "B",
    "model": "Truck 2024",
    "year": 2024,
    "gas_consumption": 720.1,
    "additional_properties": {
      "color": "Blue",
      "capacity": "20 tons"
    }
  }
 ]
}
```


Explanation:

status: Indicates the result of the operation (e.g., success, error).
vehicles: An array of objects, each representing a vehicle in the catalog.
vehicle_id: The unique identifier of the vehicle.
vehicle_type: The type of the vehicle.
engine: An object containing details about the engine.
name: The name of the engine.
type: The type of engine.
potency: The engine's potency.

weight: The engine's weight.
chassis_type: The chassis type (A or B).
model: The model of the vehicle.
year: The manufacturing year of the vehicle.
gas_consumption: The calculated fuel consumption for the vehicle.
additional_properties: An object containing any additional properties specific to the vehicle type.

4. Viewing All Engines

json

```
{
  "action": "view_all_engines"
}
```

Explanation:

action: Specifies the requested action, which in this case is to view all engines.

Response JSON:

json

```
{
  "status": "success",
  "engines": [
    {
      "engine_id": "67890",
      "name": "V8 Turbo",
      "type": "Petrol",
      "potency": 400,
      "weight": 350
    },
    {
      "engine_id": "67891",
      "name": "Diesel XL",
      "type": "Diesel",
      "potency": 600,
      "weight": 800
    }
  ]
}
```

Explanation:

status: Indicates the result of the operation (e.g., success, error).

engines: An array of objects, each representing an engine registered in the system.
engine_id: The unique identifier of the engine.
name: The name of the engine.
type: The type of engine.
potency: The engine's potency.
weight: The engine's weight.

Web Services

CreateNewVehicle

- **Method:** POST
- **URL:** /api/vehicles
- **Request:** Vehicle details including type, engine, chassis, model, year, and additional properties.
- **Response:** Status, message, and unique vehicle ID.

RegisterEngine

- **Method:** POST
- **URL:** /api/engines
- **Request:** Engine details including name, type, potency, and weight.
- **Response:** Status, message, and unique engine ID.

ViewAllVehicles

- **Method:** GET
- **URL:** /api/vehicles
- **Request:** N/A
- **Response:** Status and list of all vehicles with details.

ViewAllEngines

- **Method:** GET
- **URL:** /api/engines
- **Request:** N/A
- **Response:** Status and list of all engines with details.