TECHNICAL REPORT

USER STORIES

As a developer, I want to optimize the memory usage of the application, to reduce resource consumption.

As a developer, I want to separate the application into subsystems, to maintain modular and scalable code.

As a developer, I want to implement a monitoring system, to check memory consumption and execution time of searches.

As a developer, I want to implement a caching system, to store the last five vehicle search results and improve performance.

As a developer, I want to properly encapsulate the application components, to increase security and access control.

As a user, I want to be able to register in the application, to have access to the functionalities.

As a user, I want to be able to log in to the application, to access my data and preferences.

As a user, I want to be able to view and search by filter to find the information I need.

As a user, I want to have a simple interface to interact with the subsystems, for a smooth user experience.

As an administrator, I want to be able to create, update and delete vehicles and engines, to keep catalog information up to date.

As an administrator, I want to calculate gas consumption for trucks, so I can provide more accurate information about performance.

As a manager, I want to create high and low-end engines, so I could offer vehicles of different price ranges.

As an administrator, I want everything made by users to be logged, to track changes and audit activity.

As an administrator, I want to access performance statistics saved in a log file, to identify and resolve bottlenecks.

ENTITIES

Vehicle

Engine

User

Administrator

Subsystem

Catalog

Search

Log

Performance

CRC CARDS

VehicleFlyweight	
- Contain vehicle data for sharing by reducing memory	Catalog

VehicleFactory	
- Obtain VehicleFlyweight instances efficiently, reusing existing instances and avoiding object duplication	Catalog VehicleFlyweight

Engines	
Store information about the engine, such as engine type, capacity, power, so on.	Vehicles Catalog

Truck	
- Provide truck information -Calculate gas consumption	Vehicle

User	
Register and log in to the app. Access the application's functionalities. Show menu Handle user interaction options	Search Log

AdminProxy	
Manage the application and its content. Create, update and delete vehicles and engines. Access performance statistics and activity logs Calculate gas consumption Acces toperformance stats Acces activity log	Catolg Log Performance

Sub system	
Divide the application into modular and scalable parts.	Search Log Catalog

Search	
Search the vehicle catalog.	Catalog

Log	
Record activities carried out in the application.	Administrator

Performance	
Collect and store application performance statistics.	Administrator

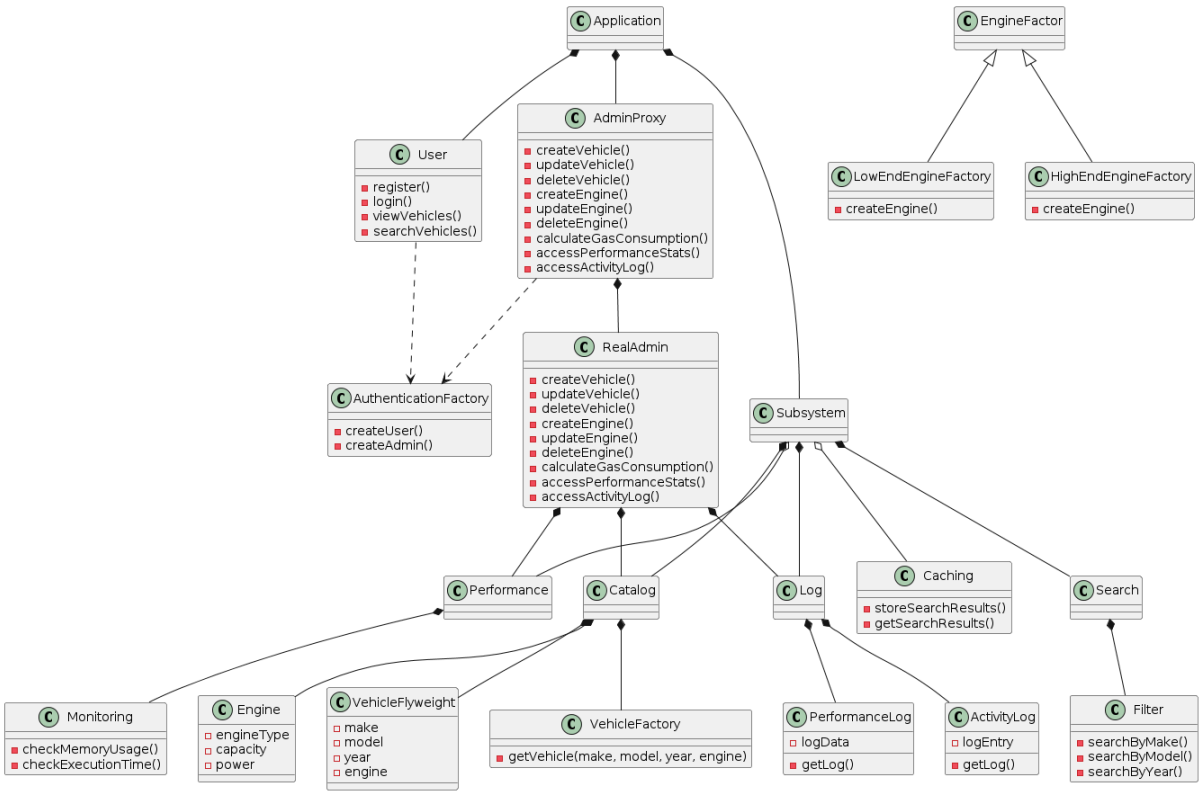
Catalog

- show a list of vehicles - add new vehicle (just manager) - let user apply filters to find vehicles	Vehicles Engines

FactoryLowEngines	
- create low price engines	Vehicles

FactoryHighEngines	
- create high price engines	Vehicles

UML



The idea with this last image it's implement structural patterns like flyweight, in this case for vehicle class, avoiding create instances duplicated with the same information and using a control with Factory pattern using the last flyweight we could use an already created instance.

Pattern proxy was created like an intermediate layer that controls access to real objects. This allows adding additional logic, such as authentication, authorization, and validation

Regarding encapsulation, it can be applied as follows: by encapsulating the logic for creating User and AdminProxy objects within a Factory class, this responsibility is separated from other classes. This promotes the principle of Single Responsibility. If, in the future, there's a need to change how User and AdminProxy objects are created, only the Factory class would require modification, without affecting other parts of the application.