Develop two endpoints one GET and one POST according to the rules below.

- Use **Java 11**, **Maven**(to manage dependencies), and mysql with **root password** and **root user**.

- The service should have an entity, dtos, a repository (for database connection), a controller , a service (which should house the logic when saving and fetching from the database)
- The connection to the database must be made using JPA and JDBC. **Hint**: **use an application.properties or application.yaml to be able to do this.**

- The payload sent to register a car in the database must have the following URLs:
  - POST: /cars/post
  - GET: /cars/get/{idChassi}

```
{
        "idChassi": "123", (type:long(this should be the primary key of your table and
unique)
        "name": "New fiesta", (type:String)
        "brand" : "Ford" ,(type:String, must accept only Ford, Chevrolet, BMW, Volvo)
        "color": "blue",  (type:String)
        "fabricationYear": "2014/2015"  (type:String)
}
```

- Rule 1 - The “brand” field should accept **only the brands** (**Ford**, **Chevrolet**, **BMW**, **Volvo**) and in case of sending another field that is not these **4 brands**, send an **exception** and not let the request execute successfully.
- Rule 2 - The above payload must be **registered in the database**.
- Rule 3- When a **GET** is called, the **IdChassi** must be passed and the corresponding car saved by the **POST** in the database must be returned.
- Rule 4- Nulls must not be saved in the database or returned in the **OUTPUT** or **ENTRY** DTOs, this validation must occur **and in case of null**, throw **an exception as described above.**
- Rules 5 - Projects that do not have **the correct database configuration will have a discounted grade.**
- Rule 6- The fields must be **in English like the payload above**.