Backend Challenge

The main goal of this challenge is to understand how you build software and what you consider production-ready code. Take your time to read the challenge, understand the requirements, and start building it when you feel ready. You don't need to keep track of your time, this is not a race.

Things we consider:

- Architecture and design of the solution
- Easiness to setup/execute
- Code quality
- Unit/Integration tests (we are not asking for full test coverage -given time constraint- however, you still demonstrate your testing skills)
- Version Control (GitHub preferred)

Try to guide us on your approach to solve the problem by explaining:

- Problems you discovered
- Executed tests and results
- Ideas you would like to implement (explain how you would implement them)
- Decisions you had to take and why
- Tested architectures and reasons why you chose them instead of other options

Technical requirements

We kindly ask you to solve the following code challenge in (Java/Kotlin and Spring Boot) as part of our selection process.

Extras

- Usage of OCI-Containers (Docker for example)

Problem

Context

Imagine you are creating the first version of heycar. We need to provide a platform that can receive the listings from the dealers through different providers, and make them available on the platform.

- Dealer - the company that is publishing the car

- Listing - the car that is being published
- Provider - the platform the dealers already use to manage their own listings

Description

We need to get listings from different sources and make them available on the platform in a standardized format. Some of our providers are sending data via CSV, with the following format (+example) to the `/upload_csv/<dealer_id>/`endpoint via POST, where `<dealer_id>` will be the id of the dealer who is sending the data:

```
code,make/model,power-in-ps,year,color,price
1,mercedes/a 180,123,2014,black,15950
2,audi/a3,111,2016,white,17210
3,vw/golf,86,2018,green,14980
4,skoda/octavia,86,2018,16990
```

The rest of the providers are sending the information via JSON API, using the following format to the `/vehicle_listings/`endpoint via POST:

```
[
  {
    "code": "a",
    "make": "renault",
    "model": "megane",
    "kW": 132,
    "year": 2014,
    "color": "red",
    "price": 13990
  }
]
```

Requirements

- Different dealers may have listings with the same code, and the listings should be treated as different listings
- If a listing is sent again by the same dealer (based on the code), it should be updated
- All the uploaded listings should be available in JSON on the `/search` endpointsvia GET
- A client should be able to search using the following parameters: make, model, year, and color

For this challenge you don't need to cover:

- Authentication
- Authorization
- Data removal