Axiom Mobile Handset Java assignment application

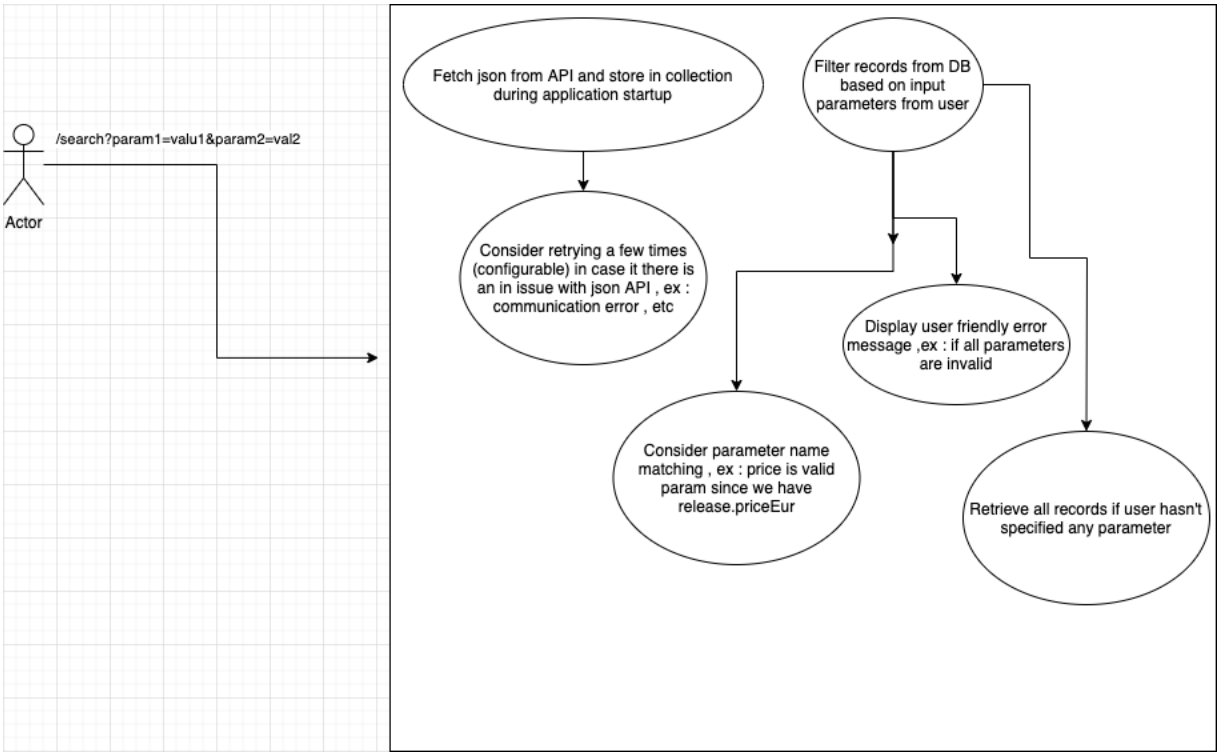
Requirements and Goals of the System

Functional Requirements:

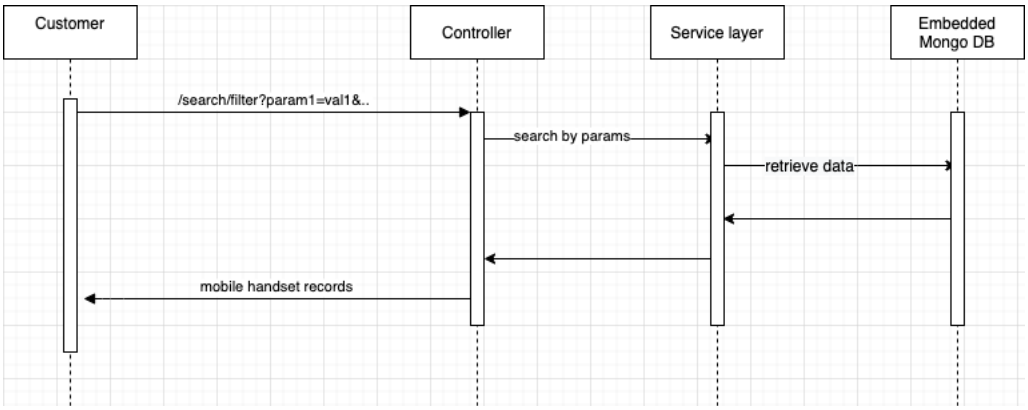
At this URL: https://a511e938-a640-4868-939e-6eef06127ca1.mock.pstmn.io/handsets/list, you will find a JSON file with a sample "Mobile handset" database. The data in this JSON is static, I.e it doesn't get updated.

1. Create a Spring Boot application exposing a search API (**GET /mobile/search?)** that will allow the caller to retrieve one or more mobile handset record based on the passed search criteria. The criteria can be any field in the handset data along with any value. Examples: • */search?priceEur=200*. Will return 10 devices. • */search?sim=eSim*. Will return 18 devices. • /search?announceDate=1999&price=200. Will return 2 devices.
2. Consume the JSON API in the best way you see suitable.
3. Create Unit Test cases as you see suitable.
4. Provide a documentation and a high-level architecture of your application. You can do one or more diagrams, as you see suitable, to describe your application functionality.

Use case diagram



Sequence diagram



System APIs

```
ResponseEntity<List<MobileHandset>> search(MultiValueMap<String,Object> requestParams)
```

Parameters:

Query parameters MultiValueMap<String, Object> requestParams
Example : /search?announceDate=1999&price=200

Returns: List of handset records
Successful request gets list of mobile handset records back based on query criteria, otherwise user friendly error will be specified

Swagger docs : http://localhost:8687/swagger-ui.html#/

Security and Permissions

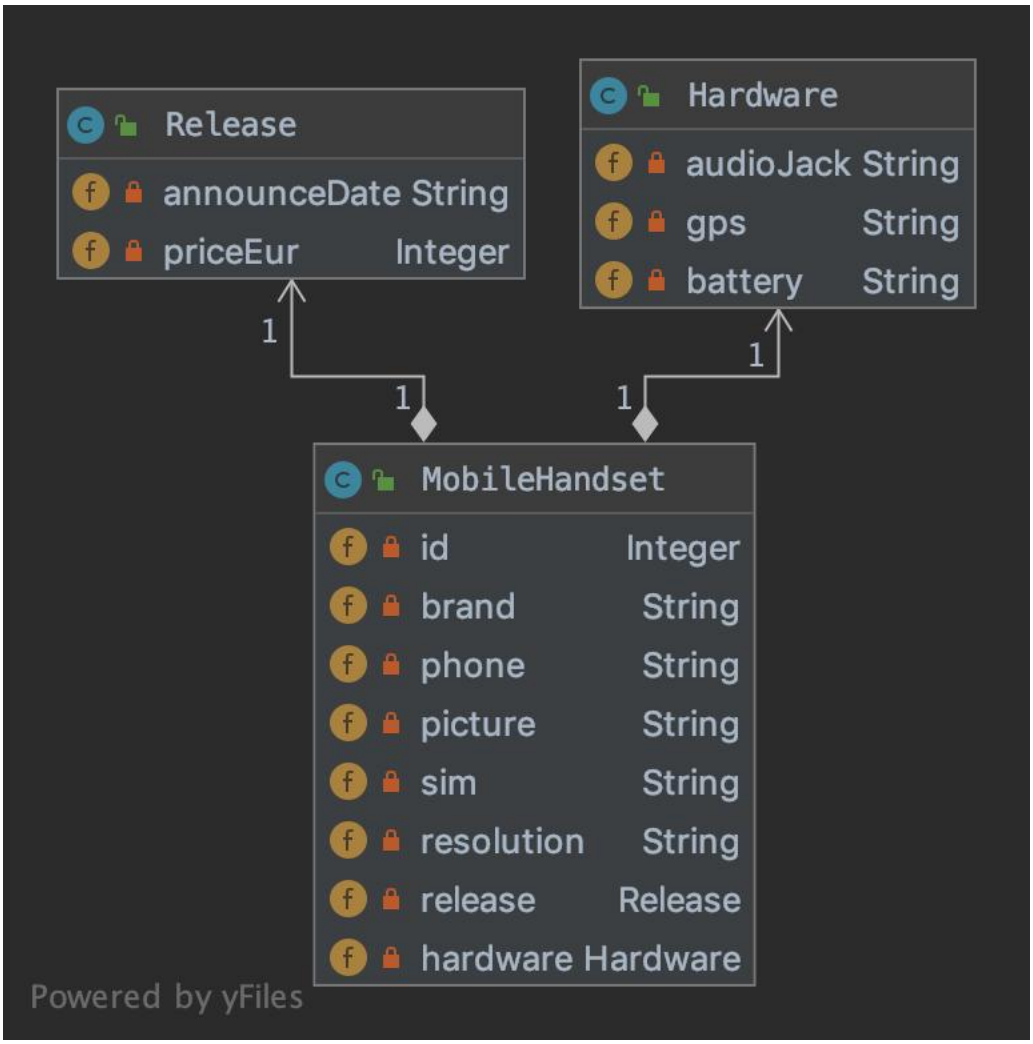
System API doesn't have a requirement for security and I keep it as it is for sake of simplicity

In general we can set up stateless (token based ex :JWT) or stateful (session based ex: Spring security , or third party provider, etc) authentication mechanism in application .

For this scenario we can also use API Gateway to handle authentication mechanism in order to keep application code simpler.

Database Design

MongoDB collections based on provided json files



Scalability

In order to make system easily scalable we still need to do a few more configurations.

1. Currently data is inserted to DB during startup for sake of simplicity which would cause an issue if we have multiple instances running at the same time , the best practice for this scenario is using database migration tools to make sure data (structure) consistency. Tools like **Mongoose** , **Mongock** etc . (Or Liquibase/Flyway if we decide to have relational DB).

As per the requirements we don’t have DB writes. For database scalability we can use built-in sharding feature from Mongo DB.

2. In order to scale application itself we would need to setup docker file plugin to containerize it then use orchestration tools like EKS,Openshift , etc
3. Load balancer to spread traffic to the instances