Recipe Management API REST

Technical Document

Author	Version	Date	Details
Yunus Sezgin	1.0	26.12.2021	Initial version.

Table of Contents

Running the Recipe API

Recipe API Github Repo Address

https://github.com/yunussezgin/recipe-service

Required tools to run the API

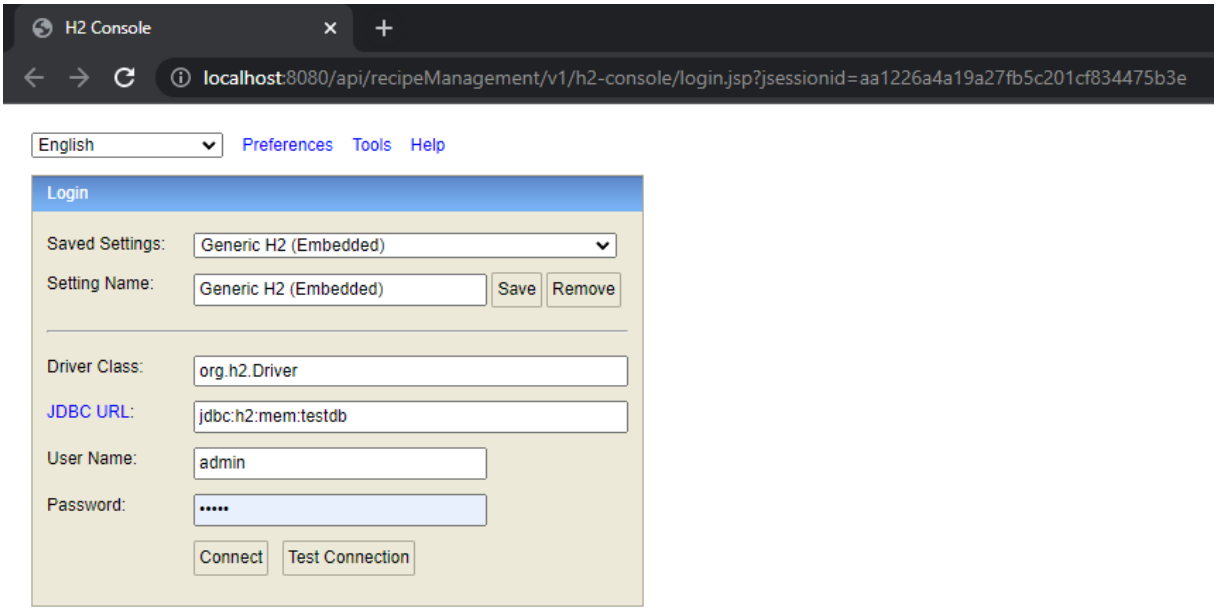
- JDK 8
- Maven
- Eclipse or IntelliJ Idea
- Lombok plugin for IDE
- Docker Desktop (optional, you can run on your PC)
- PostgreSQL (optional, you can use H2 DB)

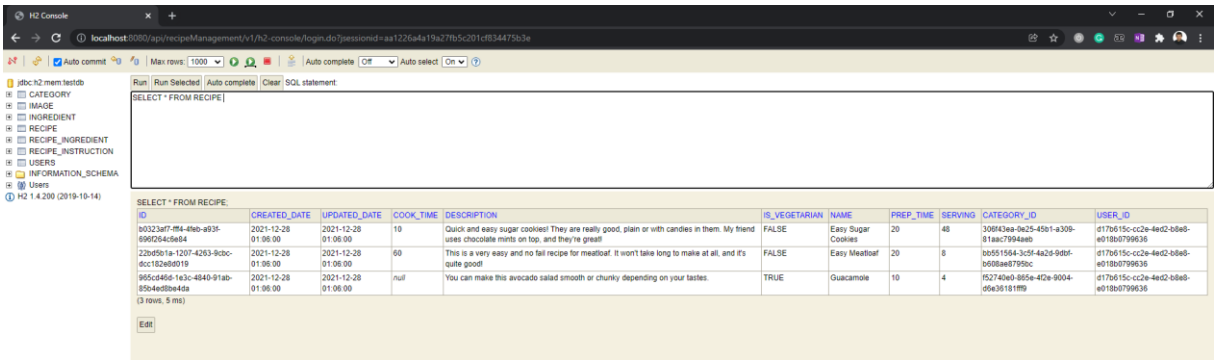
1. Running on your PC

API config file includes two DB configurations: PostgreSQL and H2 in-memory DB. The project uses H2 DB as default. If you have PostgreSQL on your PC, you can change the configuration. You don't need to do anything if you want to run with H2 DB. The configuration file name is application.yml under the resources directory.

H2 DB URL: http://localhost:8080/api/recipeManagement/v1/h2-console
JDBC URL: jdbc:h2:mem:testdb

User Name: admin
Password: admin





Run the below commands at the root of the project directory. You can also do the below on your IDE.

1. Build Recipe API using maven.
   mvn clean install
2. Run spring boot application.
   mvn spring-boot:run

2. Running on Docker

API includes docker config files which are docker file and docker-compose file. Docker creates two containers. The first one is for spring application, and the second is for PostgreSQL DB. Docker uses the application-docker.yml file under the resources directory for application config.

Run the below commands at the root of the project directory.

1. Build Recipe API using maven.
   mvn clean install
2. Create containerized images for the application using a docker-compose command.
   docker-compose build
3. Run all containers using the single command as below.
   docker-compose up

- In addition, to stop all containers, you can use the following command.
  docker-compose down

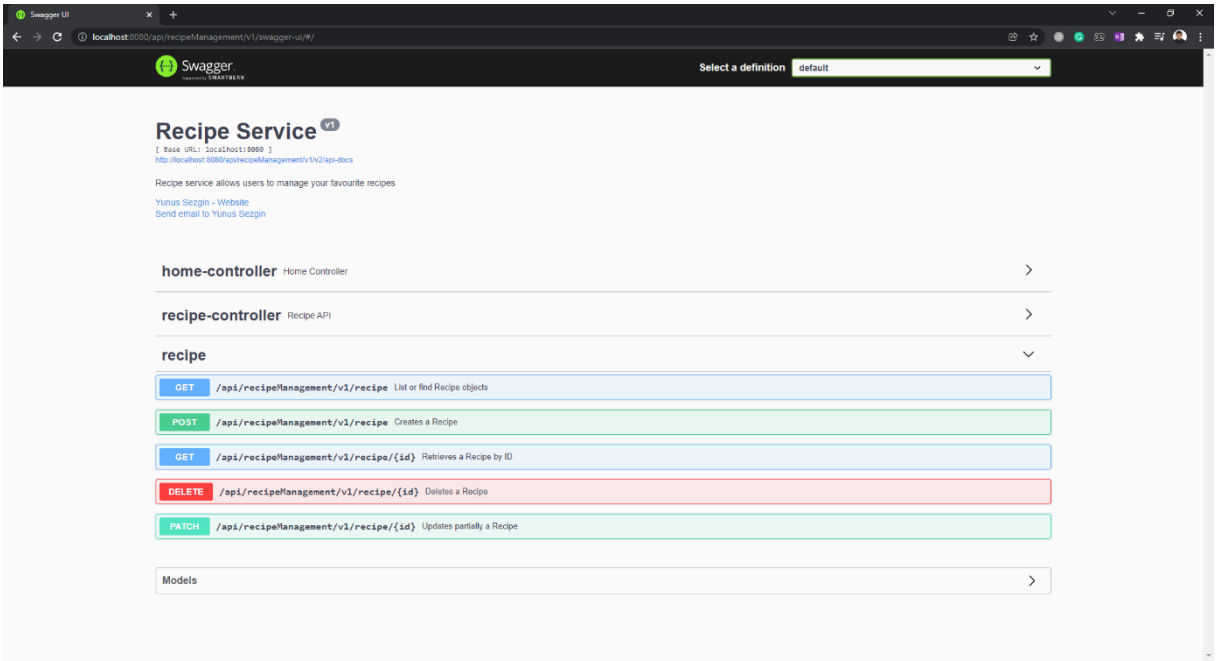
Invoke the Recipe API

1. Invoke with Swagger UI

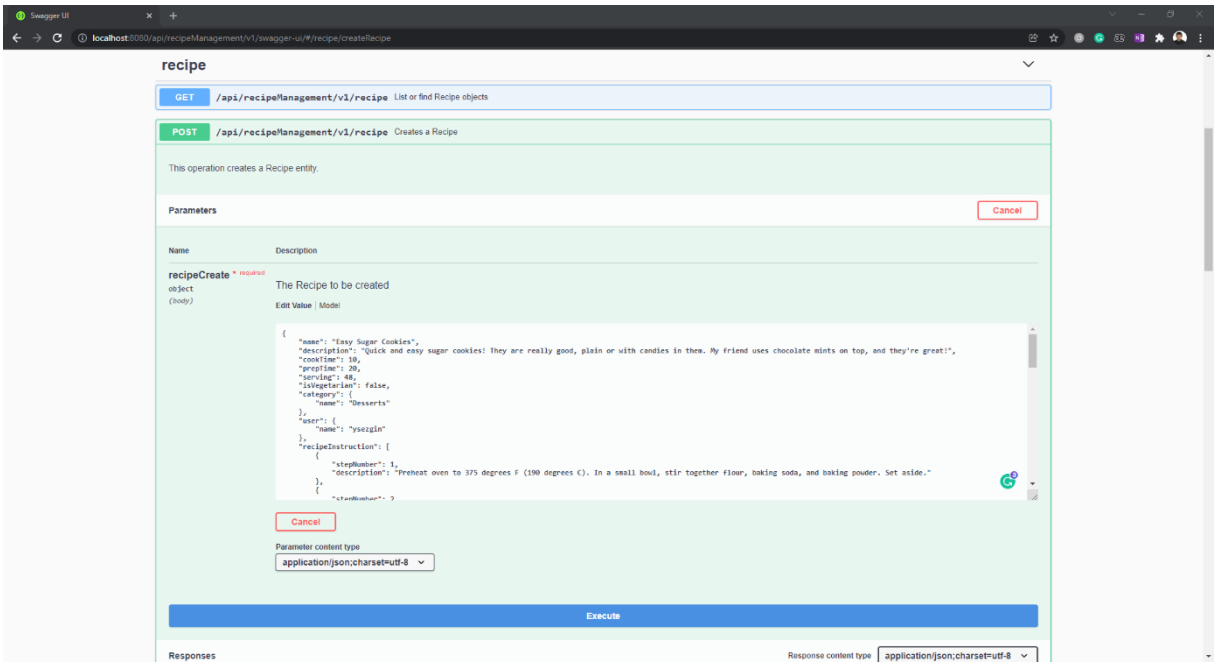
Recipe API includes Swagger UI implementation. You can get information about API endpoints and models on Swagger UI. You can invoke the API endpoints with Swagger UI.

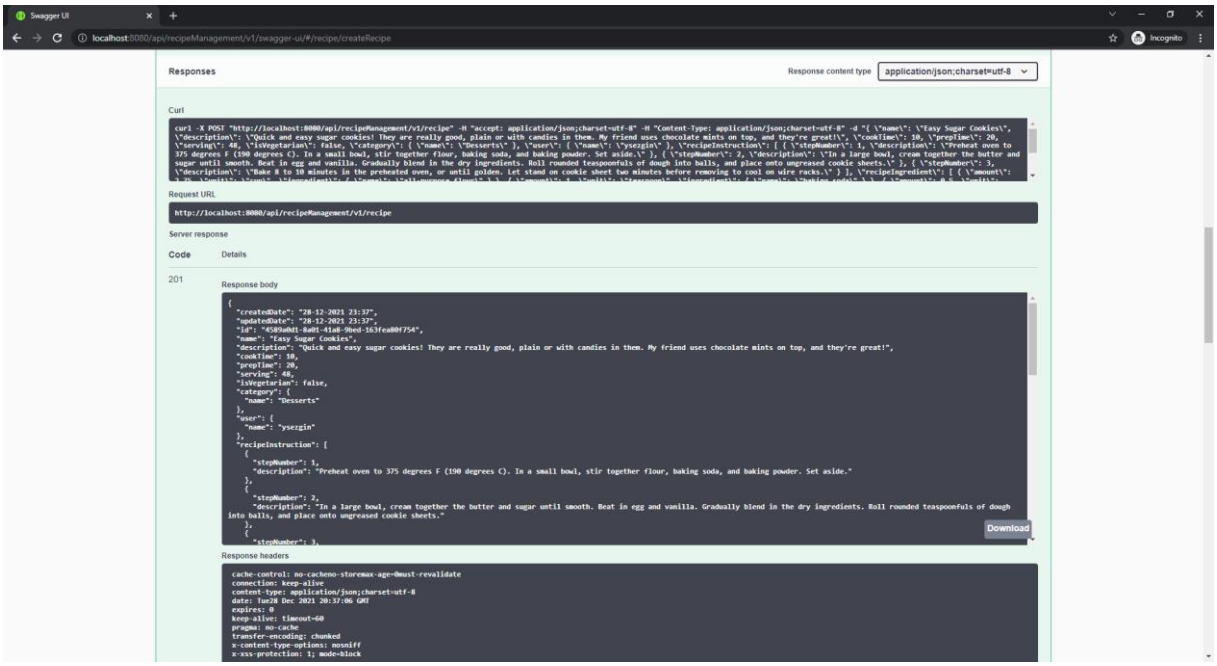
Swagger UI: http://localhost:8080/api/recipeManagement/v1/swagger-ui/

Username: admin
Password: admin



You can reach the endpoints under the recipe. After choosing an operation, click the Try it out button. Then fill in the request body or parameters. After then click the Execute button. You can see the response in the response body box.



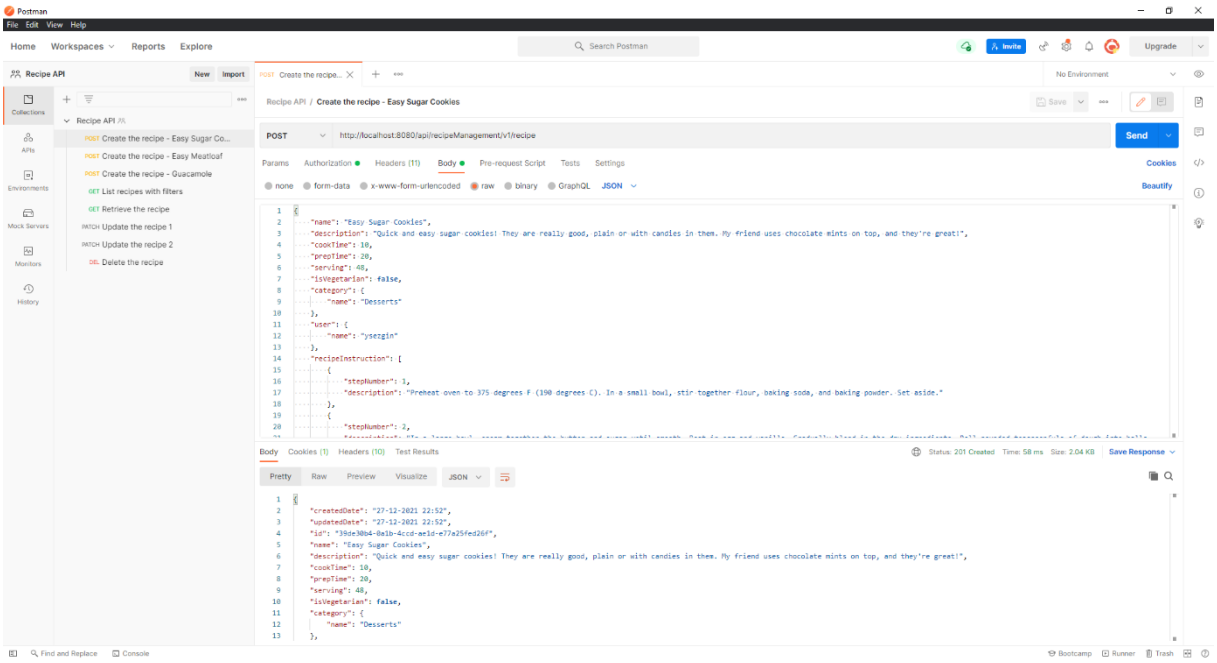


2. Invoke with Postman

The project includes postman collection, which provides sample API requests. The postman collection is under the resources/postman directory. You can import the collection to your postman.

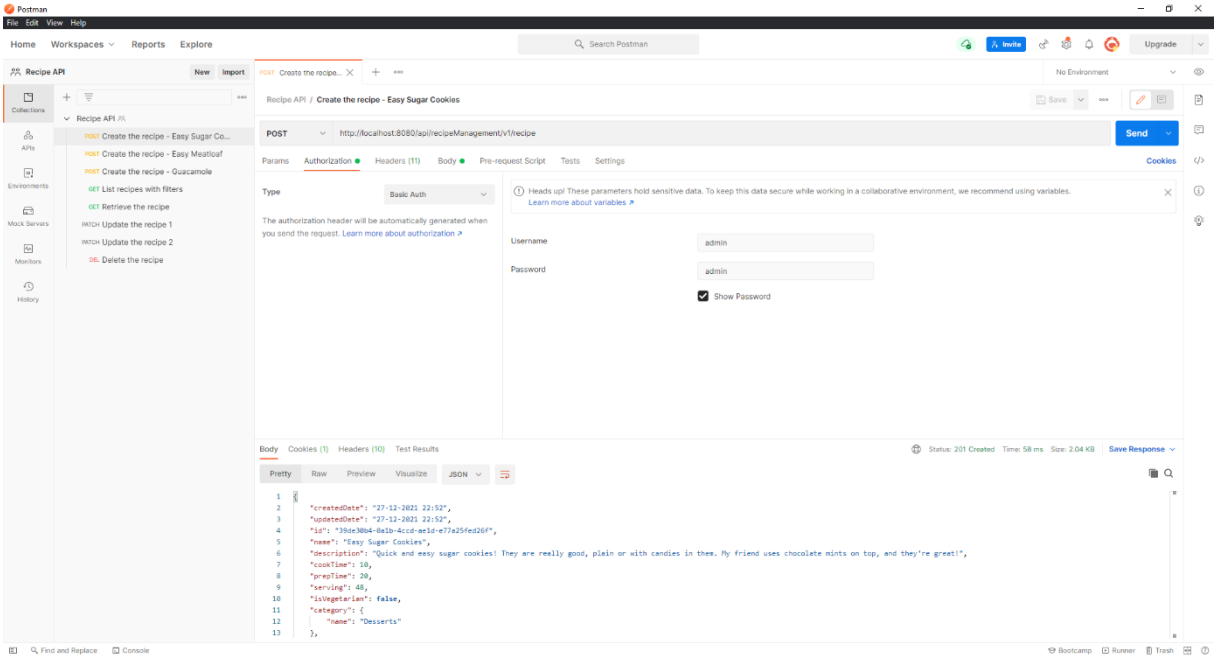
Postman collection: https://github.com/yunussezgin/recipe-service/blob/dev/src/main/resources/postman/Recipe%20API.postman_collection.json

Unit test data: https://github.com/yunussezgin/recipe-service/tree/dev/src/test/resources/data



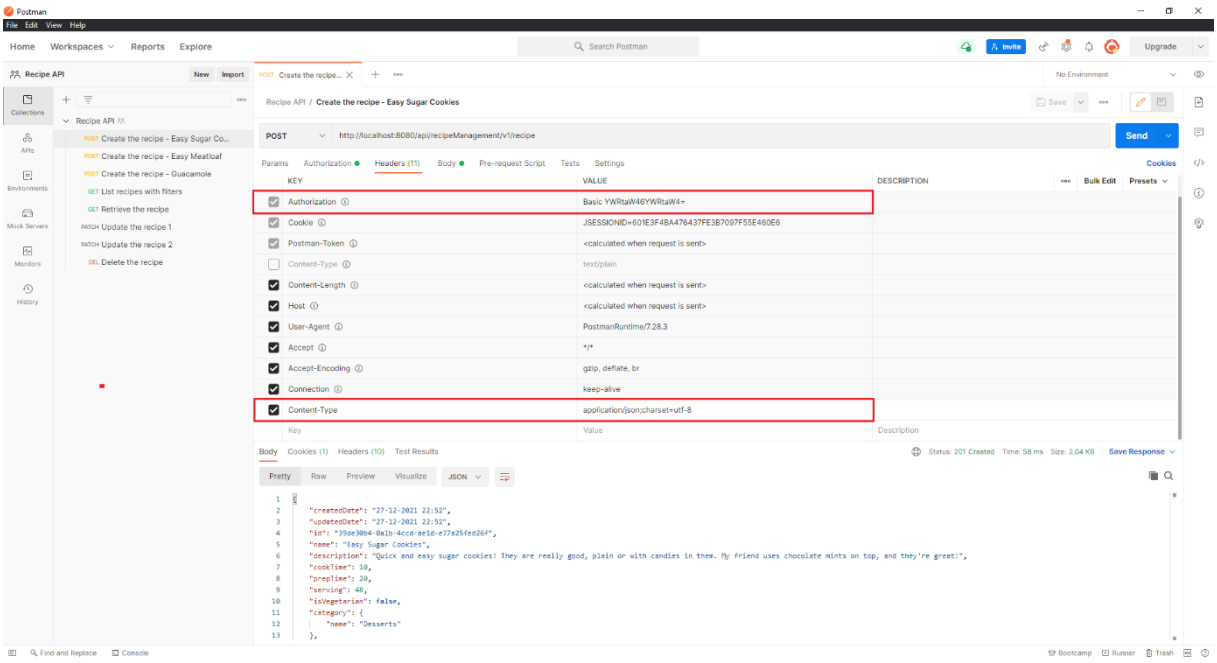
API supports basic authentication. You can change user name and password on application.yml. Currently defined values the below.

Username: admin
Password: admin



You must send two keys/values with the header. “admin:admin” is equal “YWRtaW46YWRtaW4=” as Base64 encoded.

Authorization: Basic YWRtaW46YWRtaW4=
Content-Type: application/json;charset=utf-8



Architectural Details

API Highlight Features

1- API was developed with the spring boot framework. This framework made the API more maintainable and reliable.

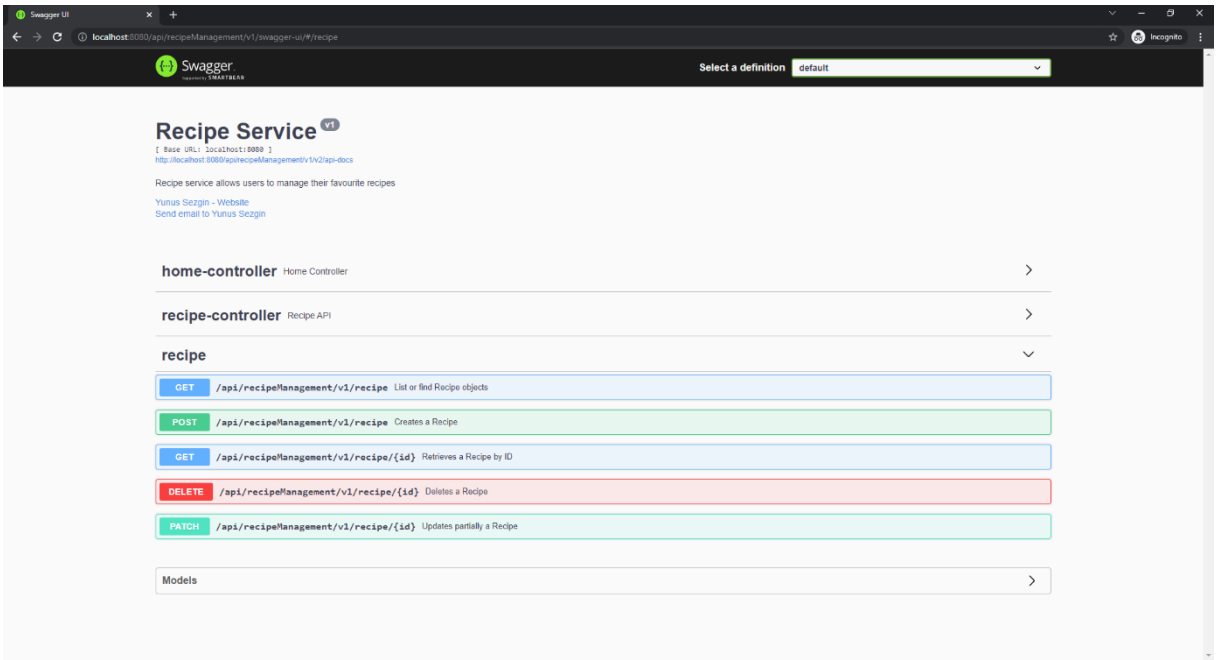
2- Querydsl library was implemented to filter records. The below get request filters records according to user.name, ingredient.name, and category.name. You can search records safely with this method.

http://localhost:8080/api/recipeManagement/v1/recipe?**user.name=ysezgin&recipeIngredient.ingredient.name=baking soda&category.name=Desserts**

3- Swagger UI is used to get information about API endpoints and models. You can also create a client service with the swagger to invoke the Recipe API from another API.

Swagger UI: http://localhost:8080/api/recipeManagement/v1/swagger-ui/
API Swagger Docs: http://localhost:8080/api/recipeManagement/v1/v2/api-docs

Username: admin
Password: admin



4- Spring Security Basic Authentication was implemented to provide security. Login credentials were defined on the application.yml file.

5- Developed a global exception handler to give meaningful error messages to users. The global exception handler handles the below example.

Body Cookies (1) Headers (10) Test Results

Pretty Raw Preview Visualize JSON

```
{
    "code": "ERR404",
    "message": "Recipe with id:f7696df4-3ce1-414f-b792-6c265e5f6491 not found!"
}
```

6- Patch operation was developed to update entities. If you want to update a record, you can send a partial request body. Firstly, the record will be found on DB. Then your request body will merge with the DB record. After then the record will update with merged body request. If you invoke the API from UI, this method provides less data transfer between UI and backend.

7- The unit tests were developed with the Given-When-Then pattern. Unit tests run on in-memory DB. This method prevents side effects. Tests use sample data files that are in the test/resources directory.