Siva

This document provides an overview of Recipe management API.

A

Recipe Management Handbook

Contents

[**1.** **Overivew:** 2](#_Toc110939588)

[**2.** **Prerequisite** 2](#_Toc110939589)

[**3.** **Solution requirements & choices** 2](#_Toc110939590)

[**4.** **Steps to run the application from windows** 3](#_Toc110939593)

[**5.** **API Usage** 4](#_Toc110939594)

[6. **Technology and IDE used**: 12](#_Toc110939596)

[**7.** **Steps to Configure the application in Spring tool suite.** 12](#_Toc110939597)

[**8.** **Current Limitation** 13](#_Toc110939598)

1. **Overivew:**

Recipe management API provides way to perform Create, Read, Update and delete operations on the user’s favorite recipes.

The application source code is stored in git hub public repository location ***https://github.com/sivagurunathbabu/***.

<https://github.com/sivagurunathbabu/recipe-management-1.0.git>

Postman collection for requests can be accessed in the below location.

<https://www.getpostman.com/collections/1e07a5800fa539ce4aeb>

To access all API’s via Swagger (URL may change in case of any changes to be made to the configuration params as per need)

[http://localhost:8081/swagger-ui.html#](http://localhost:8081/swagger-ui.html)!

1. **Prerequisite**

* A mongo db installation is required.

1. **Solution requirements & choices**

Solution requires persistence of data for later use, search records and amend modifications to existing records. Following are few possible options which works well for the above need.

**Relational database**

**NoSQL / Document database**

Relation database are best candidates for managing structured data they store information in rows and columns hence it occupies larger space, whereas No SQL database maintains it as a key value pairs or document based or graph database based on requirement. No SQL database comparatively takes lesser space.

Records are accessed based on primary key, foreign keys and column names so search is fast in case of RDBMS for structured data, whereas unstructured data search is time consuming compared to No SQL database. Searching unstructured datas like JSON in NoSQL is quick compared to RDBMS.

Storage of information in database consumes space. No SQL databases works well for large volume of unstructured information compared to RDBMS.

The current solution has been solved using NoSQL database based on the development time, space and search complexities.

1. **Steps to run the application from windows**

Prerequisite - Mongo database instance/service should be up and running

The application jar is stored in git hub public repository download the jar along with config folder and applicationStart.bat

<https://github.com/sivagurunathbabu/recipe-management-1.0.git>

* Modify the application config params stored in application-staging.properties file under config folder as per mongo database instance.

**Below properties helps in configuring the Mongo db URI and name**

spring.data.mongodb.uri=mongodb://localhost:27017/mongoDB

spring.data.mongodb.database=mongoDB

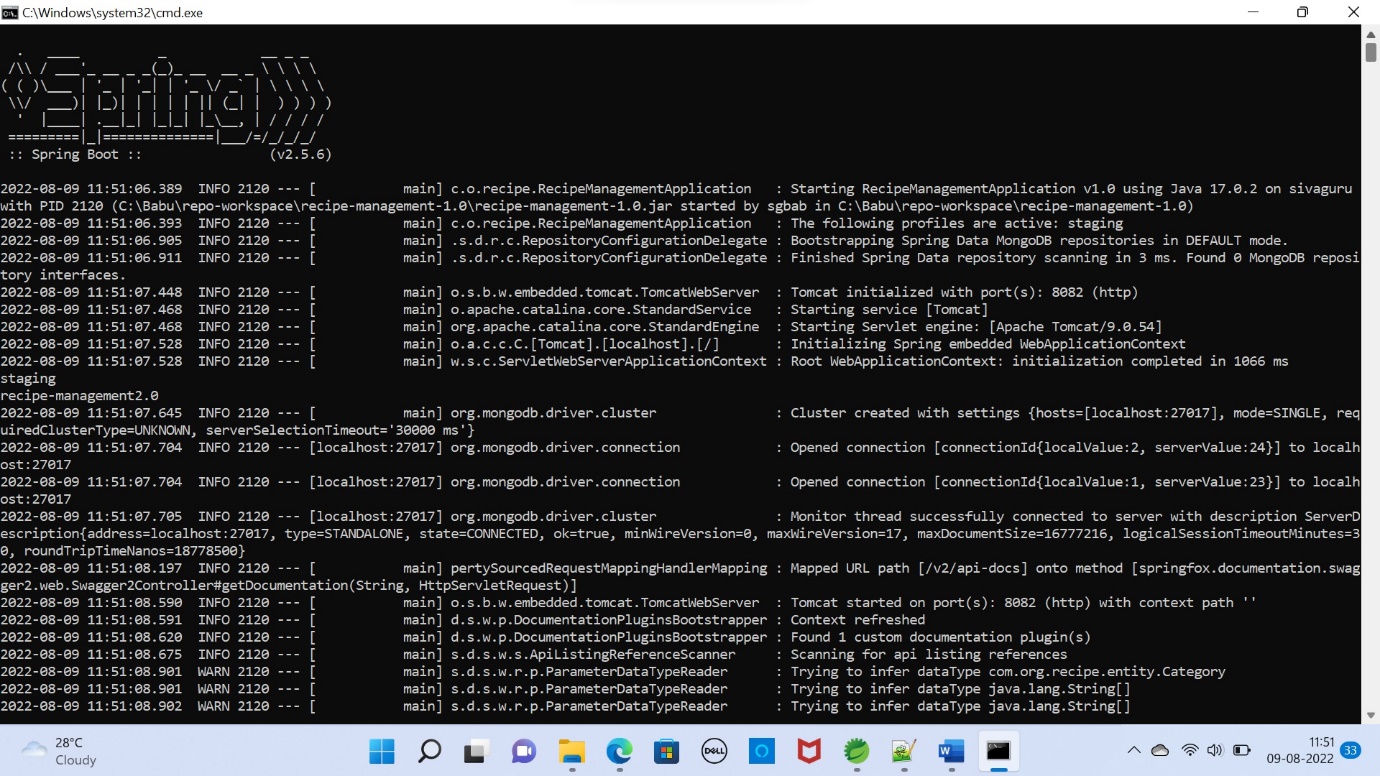
**Below properties helps in configuring the post and IP address**

server.address=localhost

server.port=8081

* Start the application by executing the applicationStart.bat (Currently configured to start the application with config params updated in config location).
* Check the console for the application start and open the swagger API screen

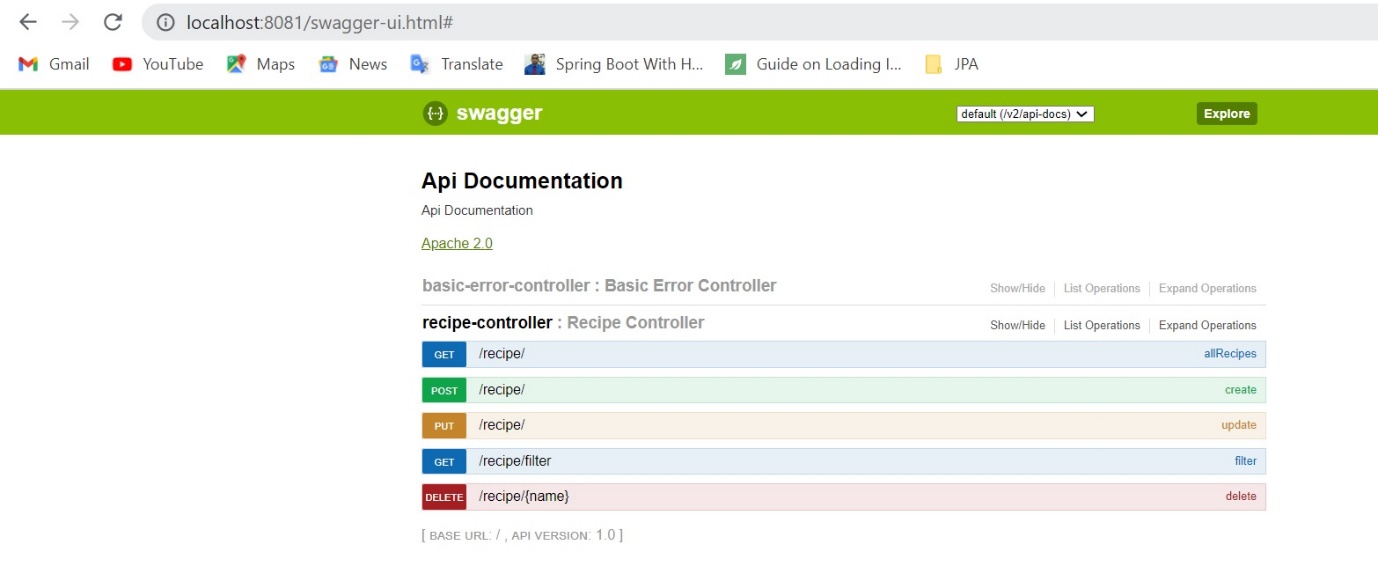
<http://localhost:8081/swagger-ui.html#>!



1. **API Usage**

All API’s can be accessed via Swagger. Click each operations and start testing your API’s. All the API’s were tested and responses were captured and shared below.

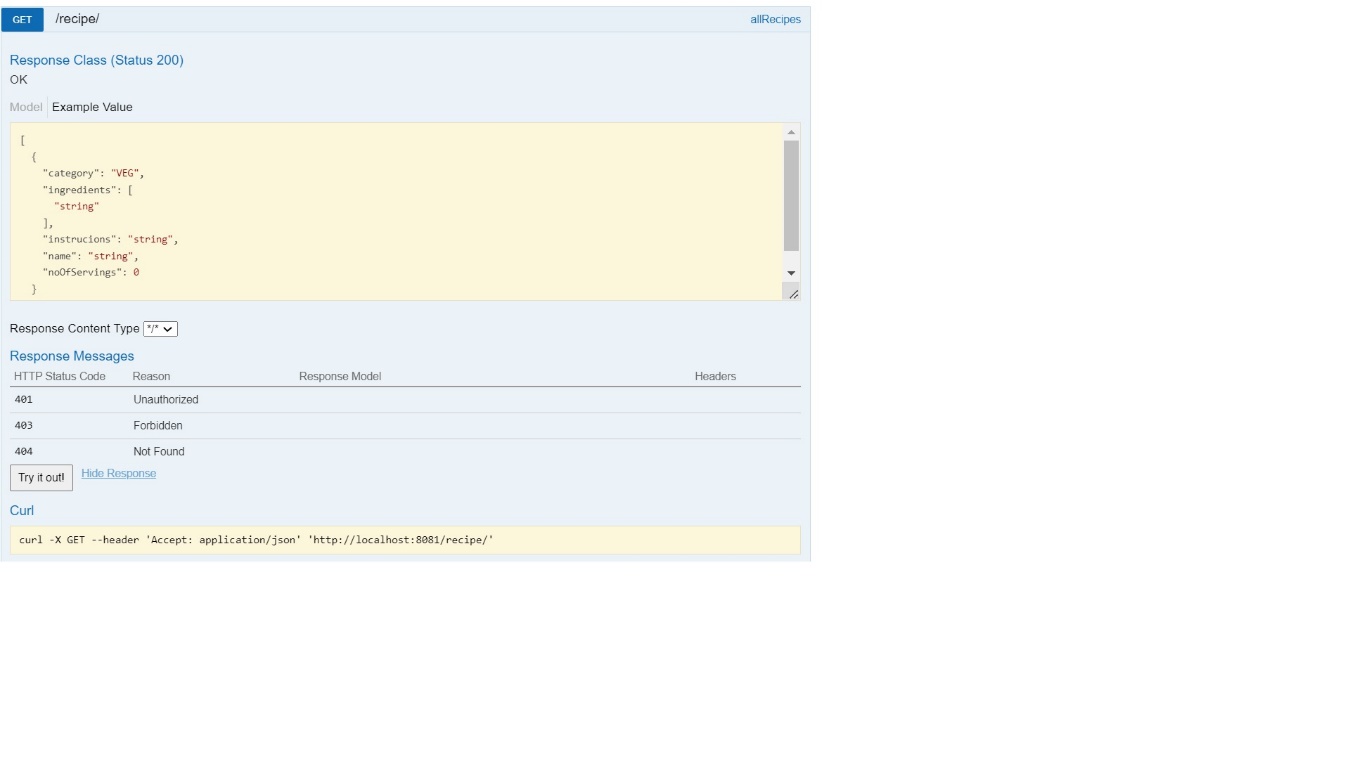
[http://localhost:8081/swagger-ui.html#](http://localhost:8081/swagger-ui.html)!



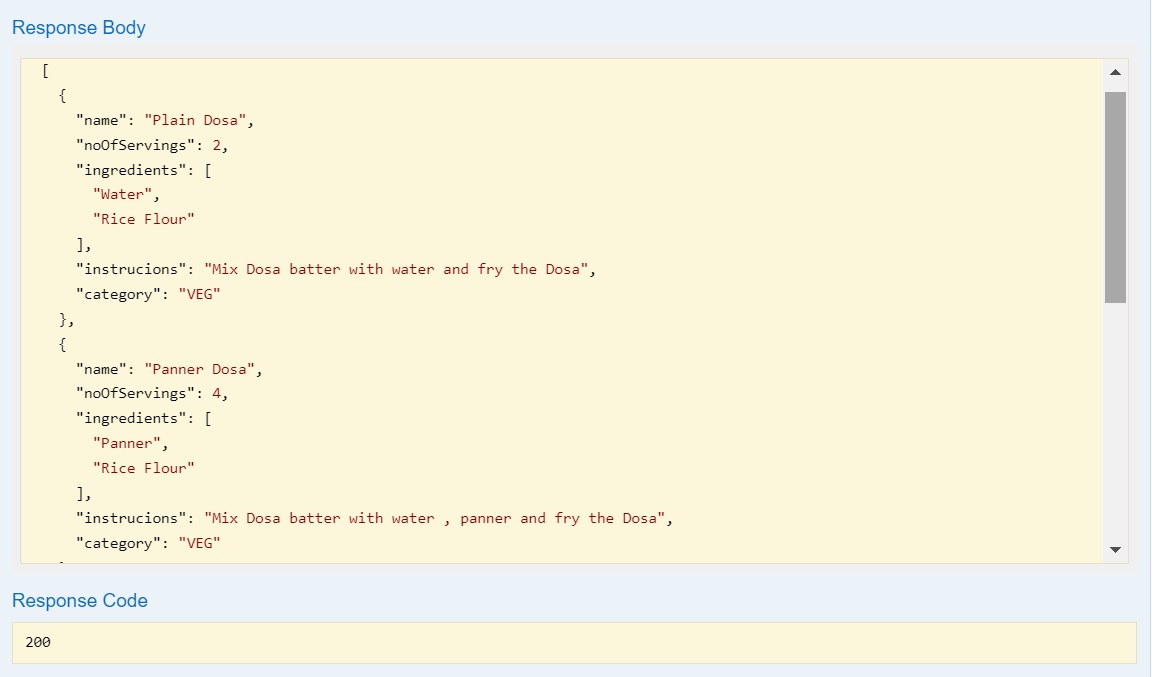
To get all recipes available:

URI: <http://localhost:8081/recipe/>

Method: Get



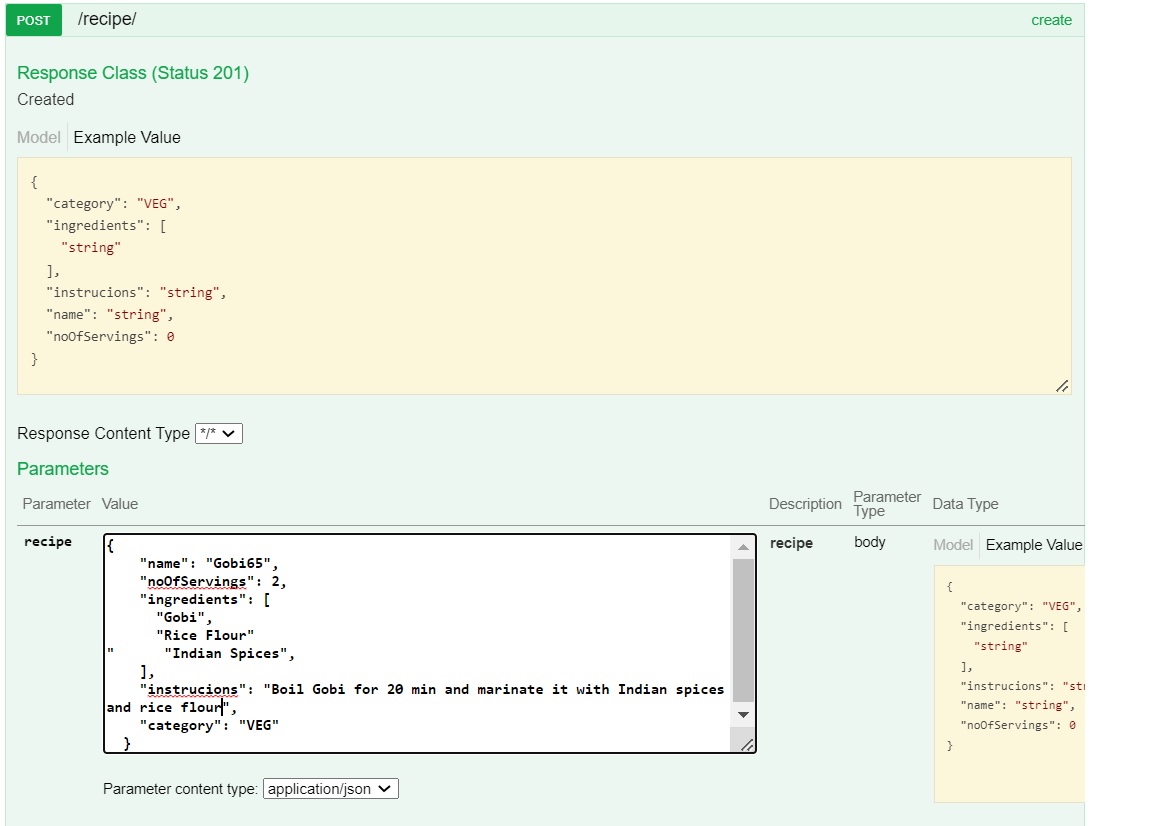
Response showing all the available recipes



To create a new recipe:

URI: <http://localhost:8081/recipe/>

Method: Post



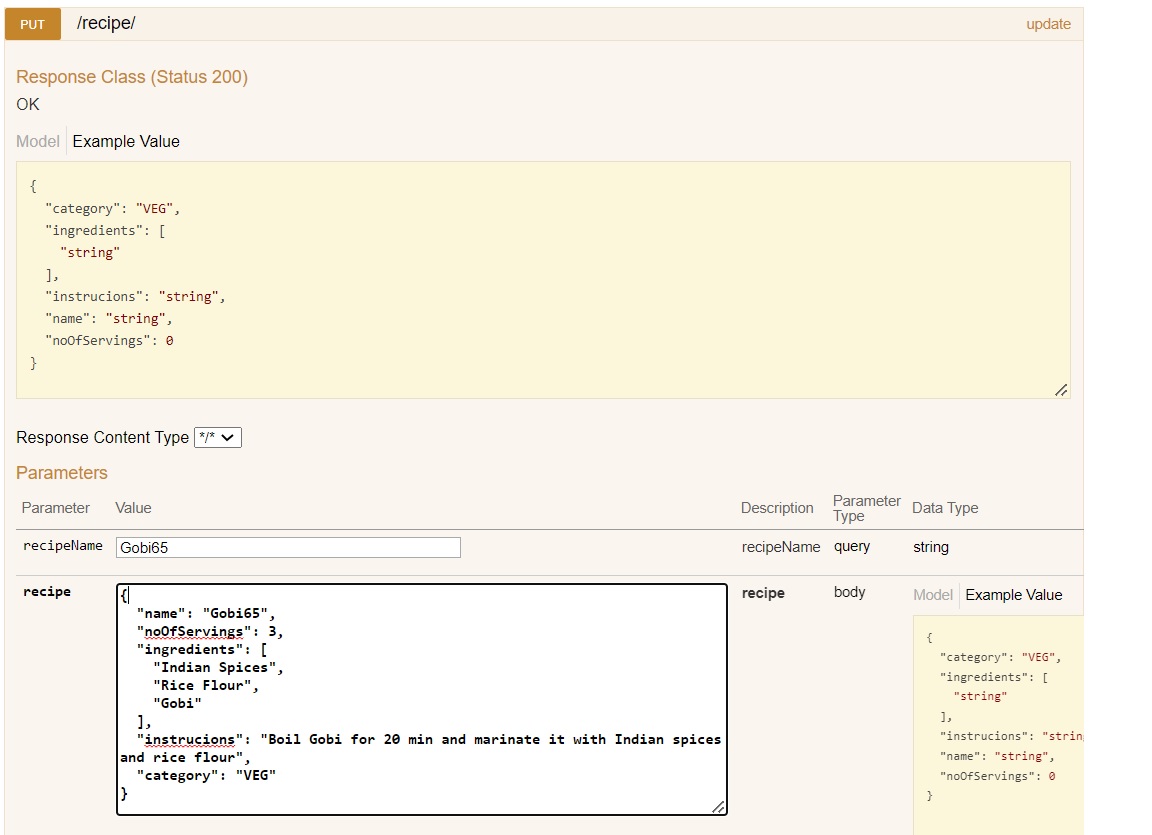
Response showing the saved recipe



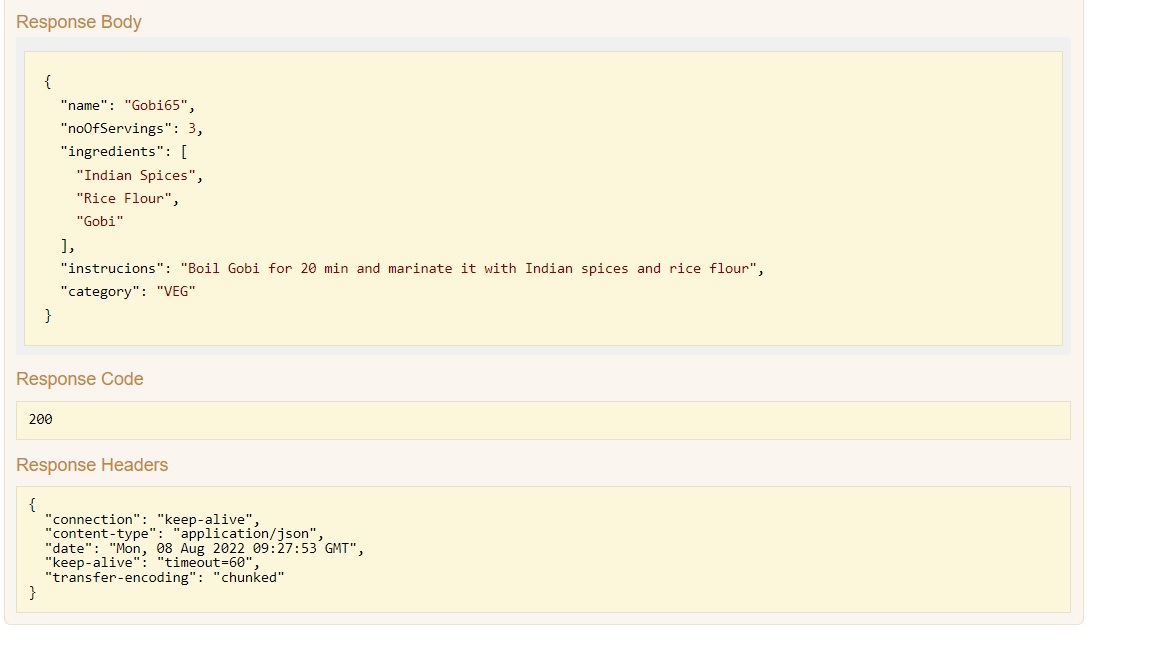
To edit/modify a recipe:

URI: <http://localhost:8081/recipe/>

Method: Put



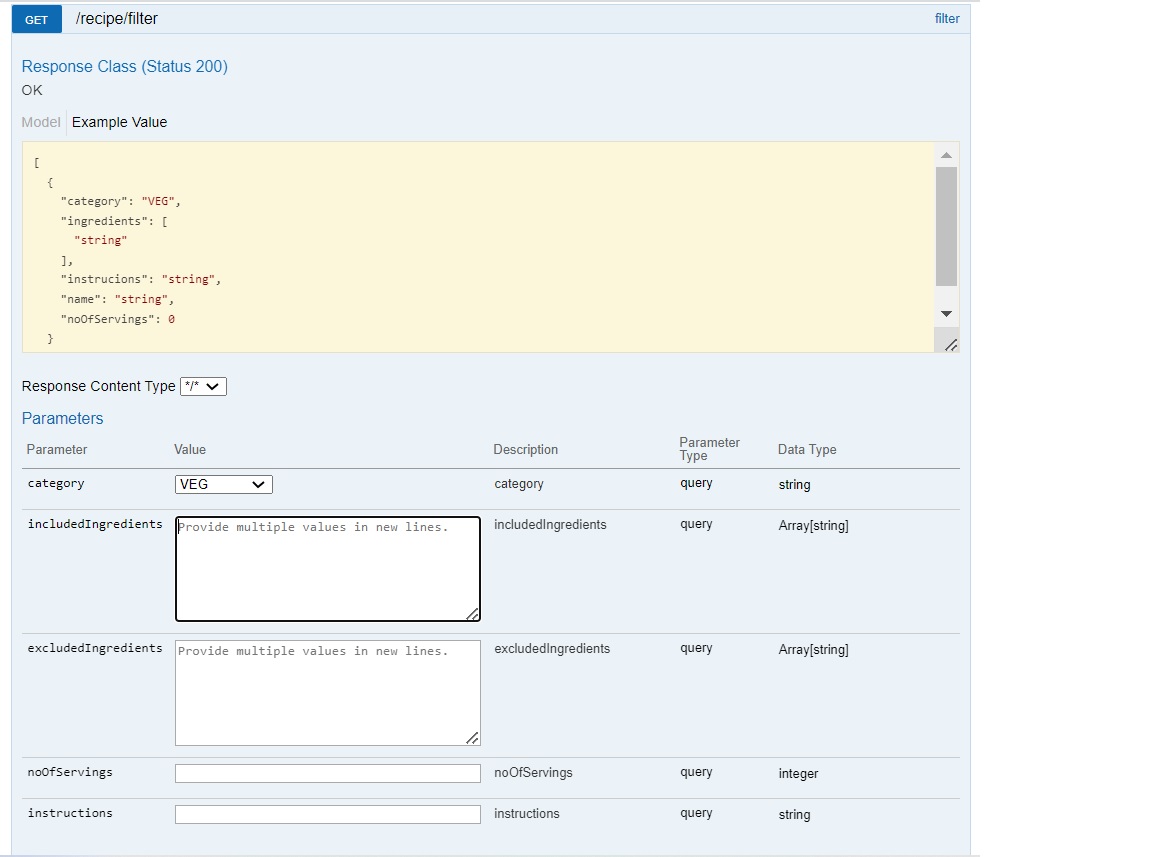
Response showing the saved recipe



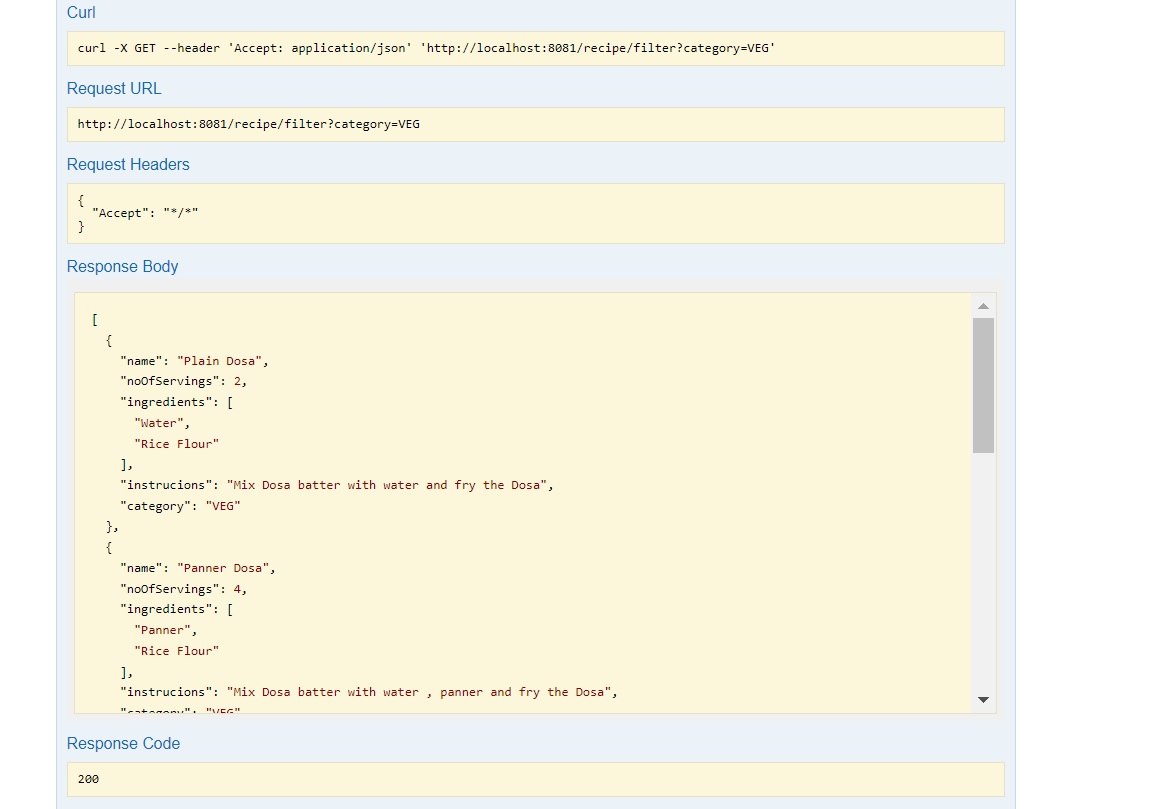
To filter a Veg recipe from the saved recipes:

URI: <http://localhost:8081/recipe/filter?category=VEG>

Method: Get



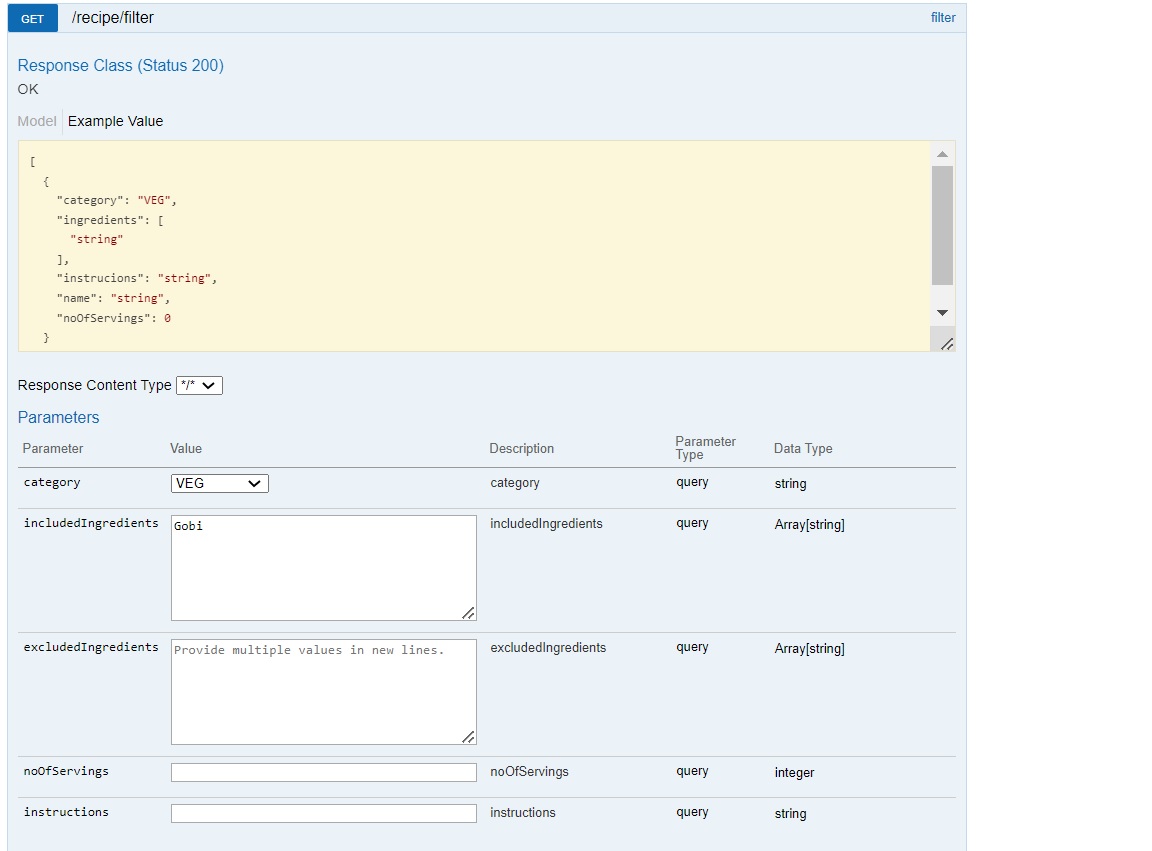
Response showing the filtered Vegetarian recipes.



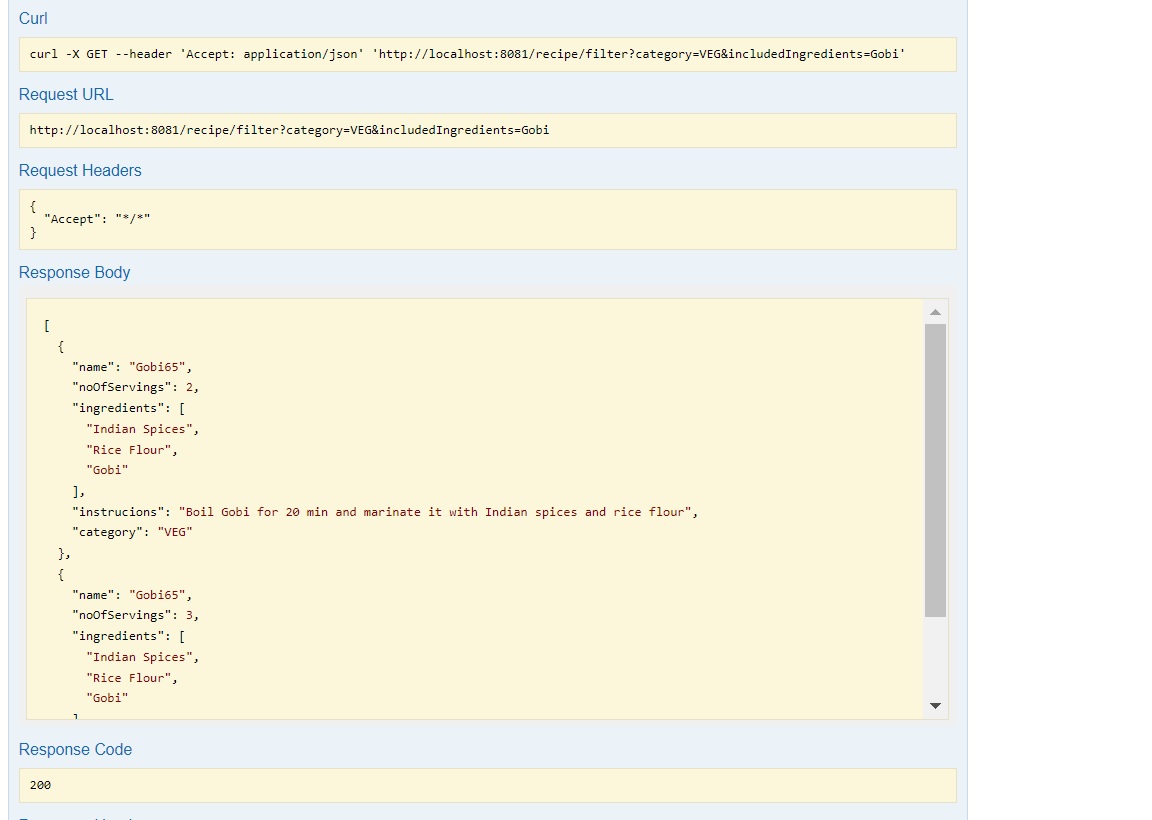
To filter a recipe with category Veg and ingredient Gobi65

URI: <http://localhost:8081/recipe/filter?category=VEG&includedIngredients=Gobi65>

Method: Get



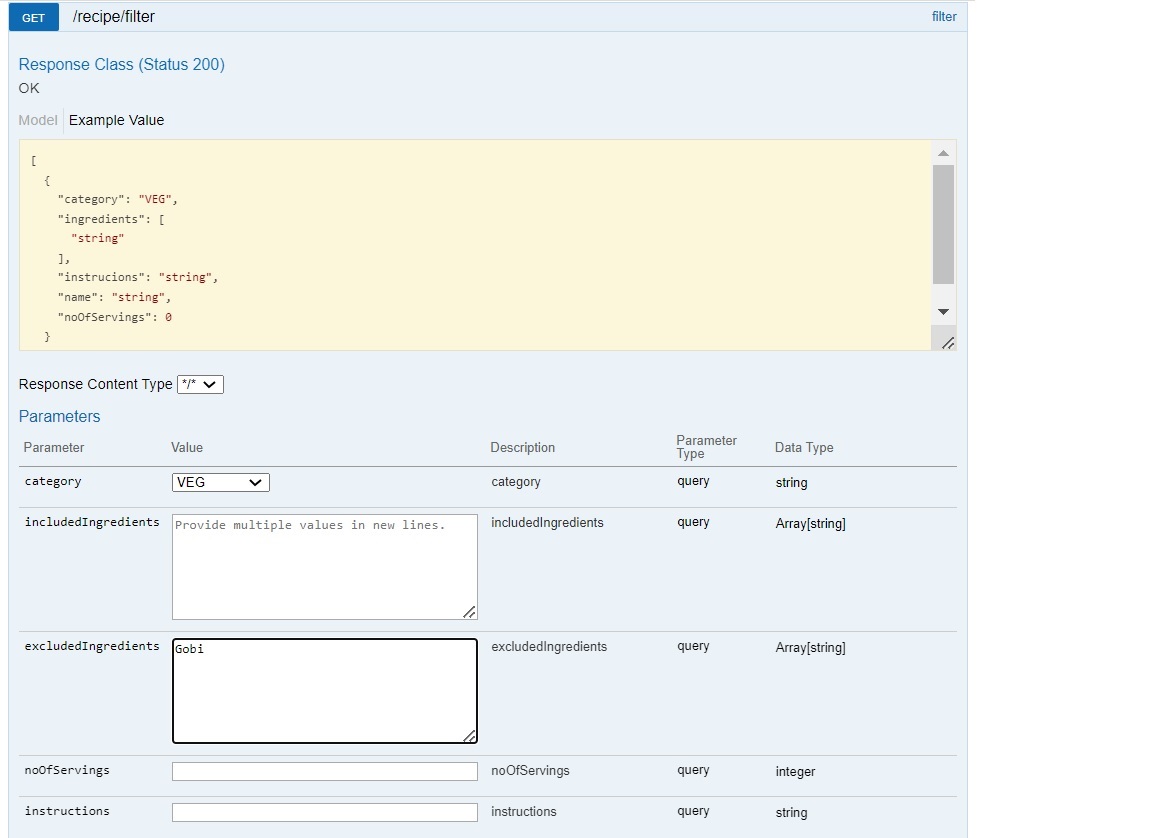
Response showing filtered recipe



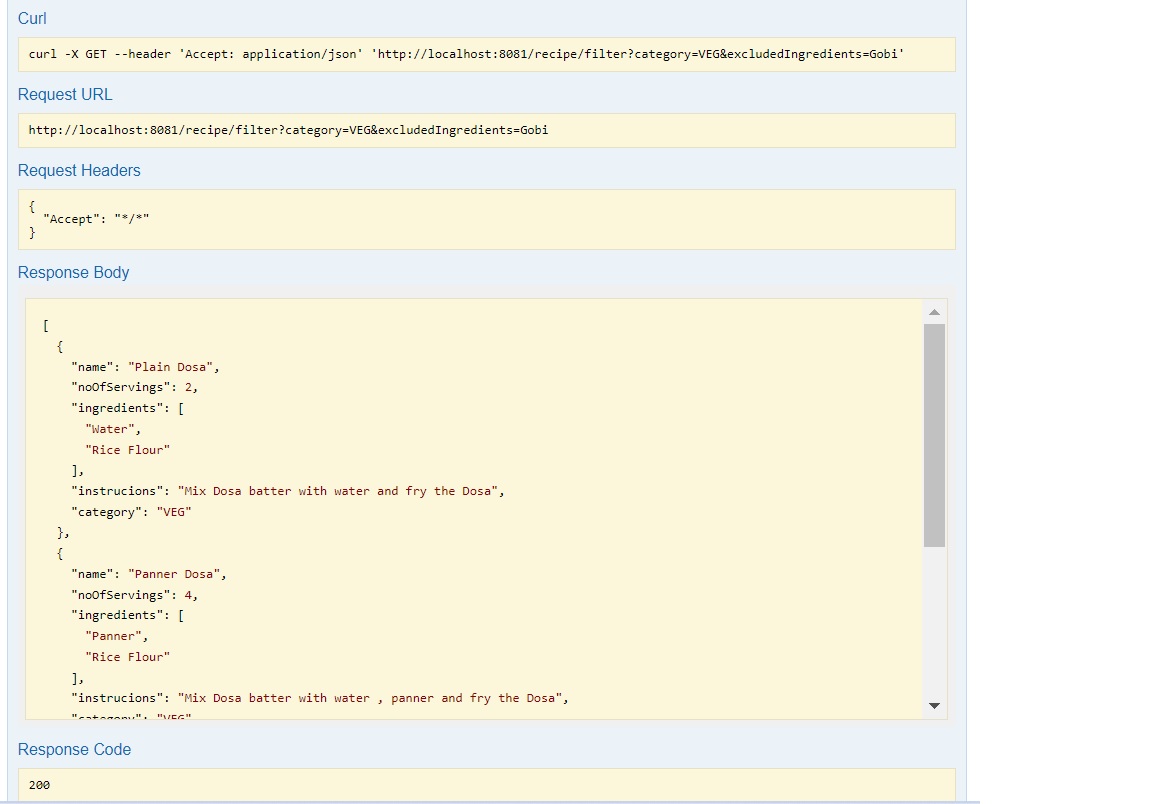
To filter a recipe with category Veg and without ingredient Gobi65

URI: <http://localhost:8081/recipe/filter?category=NON_VEG&excludedIngredients=Gobi65>

Method: Get



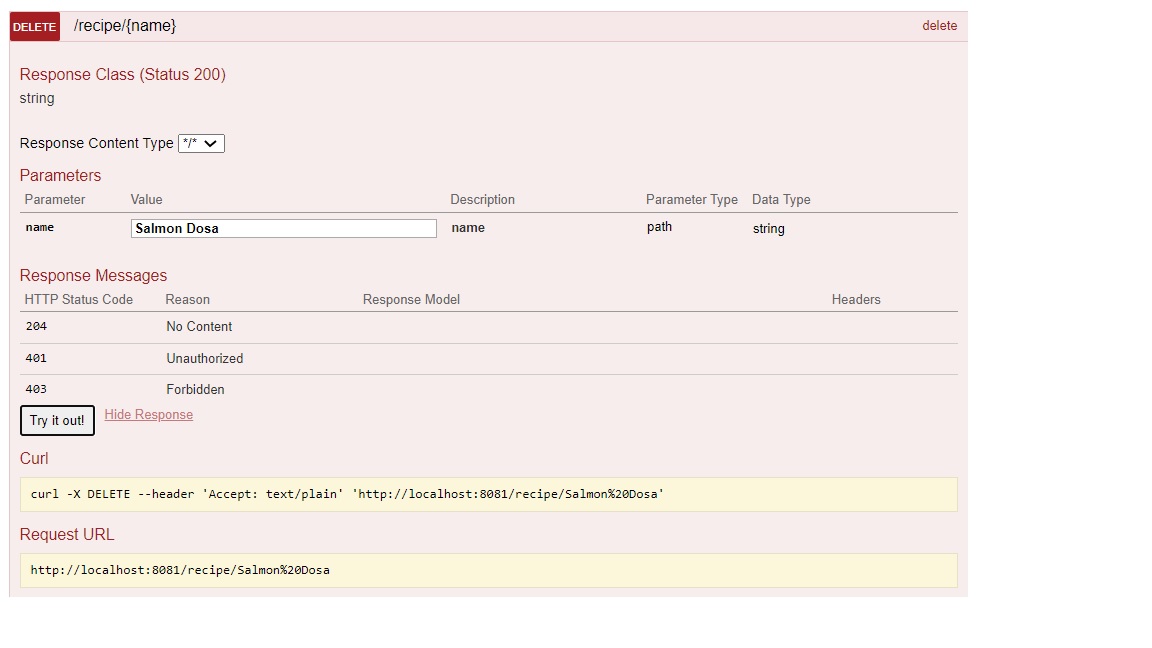
Response showing filtered recipe



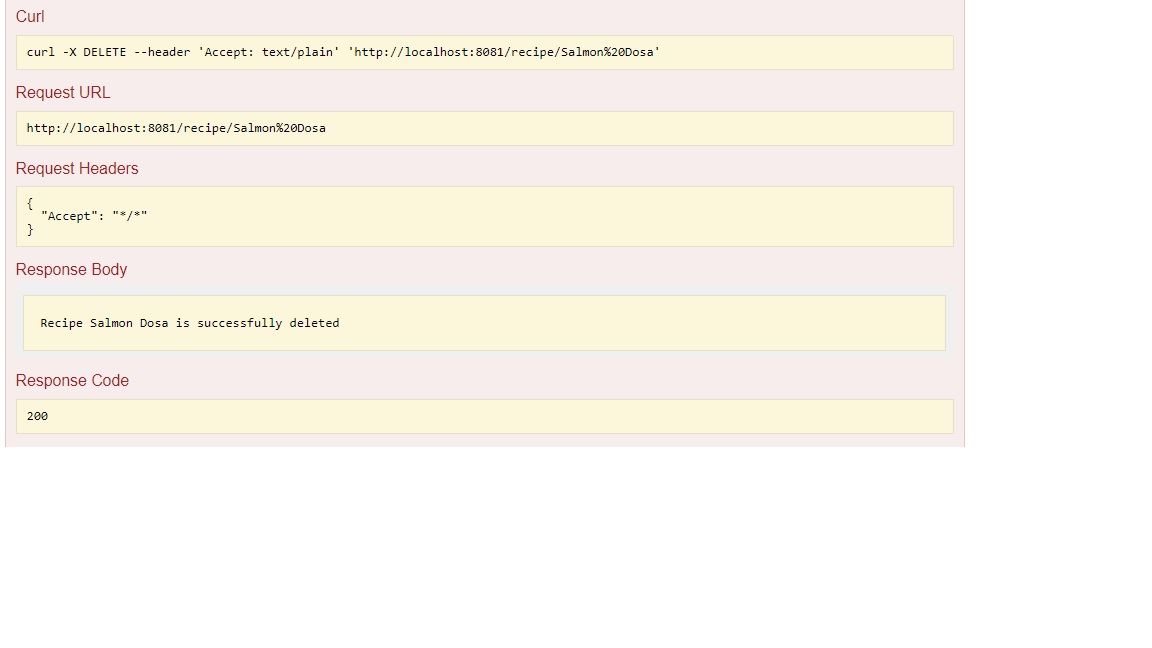
To delete a recipe with name Salmon Dosa

URI: <http://localhost:8081/recipe/Salmon%20Dosa>

Method: Delete



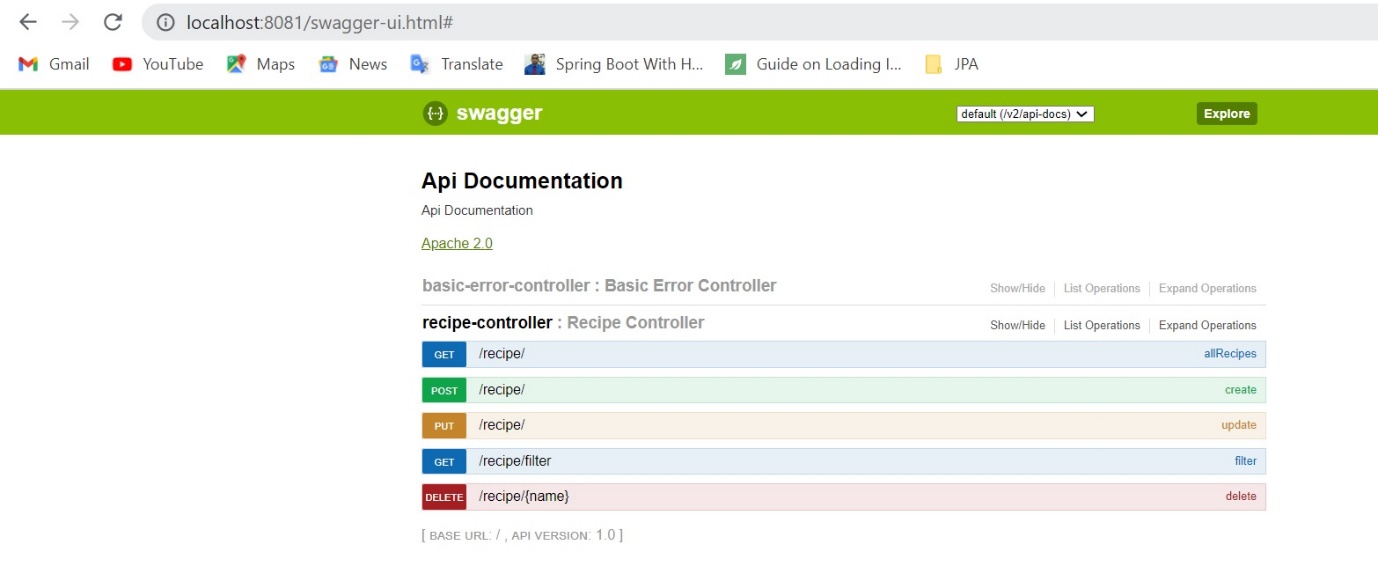
Response showing the deleted recipe



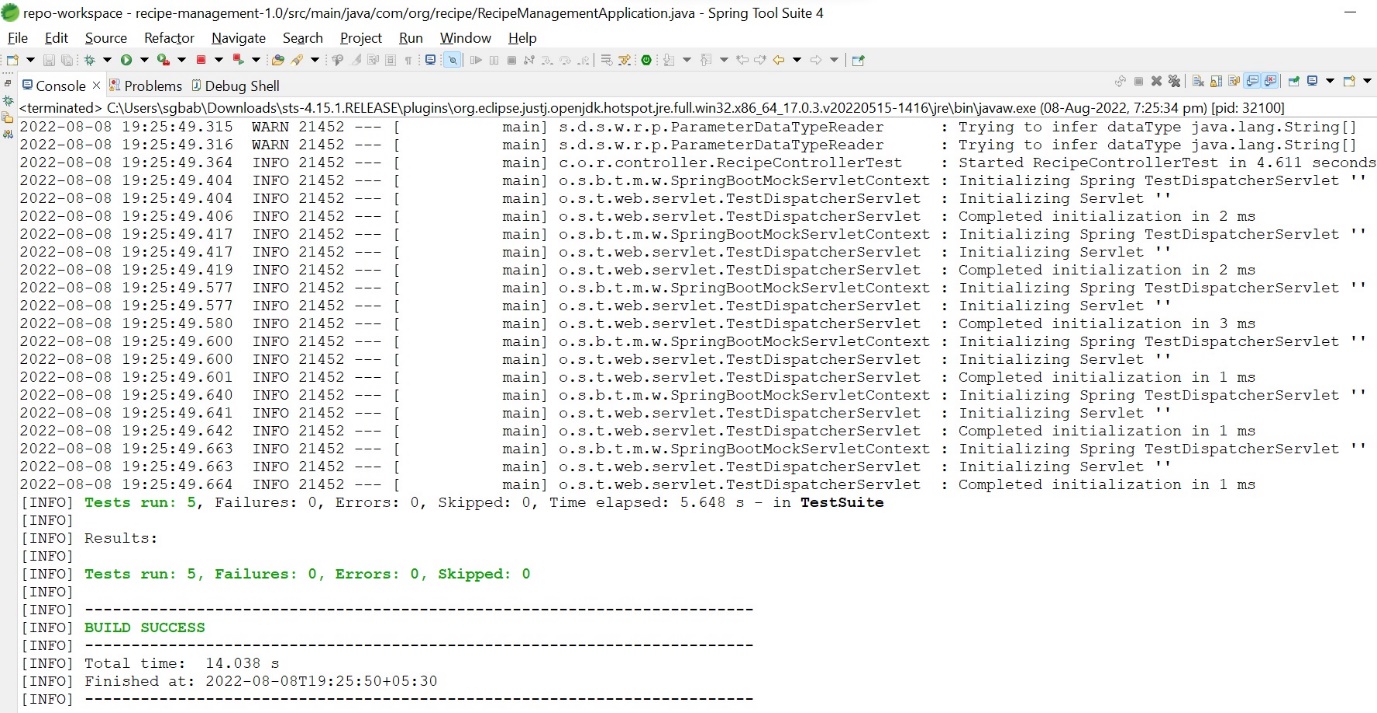
# Technology and IDE used:

* Technology Stack- Java, Spring Boot, Maven, Mongo database,
* IDE - Spring Tool Suite.

1. **Steps to Configure the application in Spring tool suite.**
2. Download the source from the repository in the zip format.
3. Unzip the source.
4. Import the workspace from the unzip location into the Spring Tool Suite IDE.
5. Once the maven downloads all the jars.
6. Modify the configuration params defined in application.properties as per need.
   1. spring.data.mongodb.uri=mongodb://localhost:27017/mongoDB
   2. spring.data.mongodb.database=mongoDB
   3. server.address=localhost
   4. server.port=8081
   5. spring.application.name=recipe-management
7. Current configuration has database name as mongoDB running in local machine on port 27017. Application port is configured to 8081.
8. Start the mongo database.
9. Launch the main application RecipeManagementApplication.java as SpringBoot application.
10. Launch the browser with below URL (As per the current configurations). Change as per your local configurations. Swagger generated screen opens with easier access to test our recipe-management API’s.
    1. [http://localhost:8081/swagger-ui.html#](http://localhost:8081/swagger-ui.html)!



1. Also run the maven build with goal test. It tests all the Integrations tests currently added.



1. **Current Limitation**

* Currently Integration tests updates the live database. Incase of RecordExistsException arises from application then delete the existing recipe (which is under test) record before rerunning the test RecipeBoServiceTest.java
* Migration to Embedded Mongodb or Testcontainer will solve the above problem, since no test result are saved in live database.