Siva

This document provides an overview of Recipe management API.

A

Recipe Management Handbook

Contents

[**1.** **Overivew:** 2](#_Toc111024875)

[**2.** **Solution requirements & choices** 2](#_Toc111024876)

[**3.** **Current Limitation** 2](#_Toc111024879)

[**4.** **Steps to run the application from windows** 3](#_Toc111024880)

[5. **Technology and IDE used** 4](#_Toc111024881)

[**6.** **Steps to Configure the application in Spring tool suite.** 4](#_Toc111024882)

[**7.** **API Test Inputs/Data** 5](#_Toc111024883)

[**8.** **API Usage** 7](#_Toc111024884)

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/recipe-management-2.0.git>

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>

1. **Solution requirements & choices**

Solution requires persistence of data, 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. **Current Limitation**

The current solution uses embedded Mongodb for data persistence. The persisted data’s will be available or having lifetime till application shutdown. Every time application data has to be created when application has been shutdown.

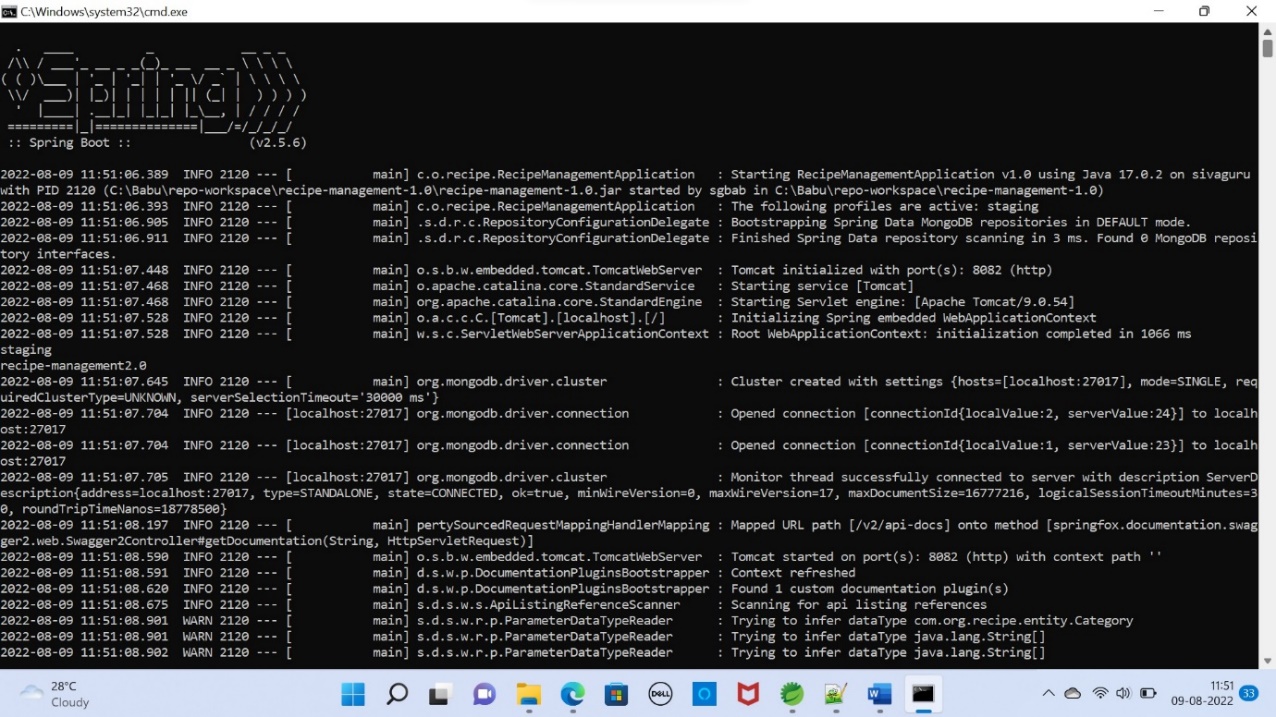
1. **Steps to run the application from windows**

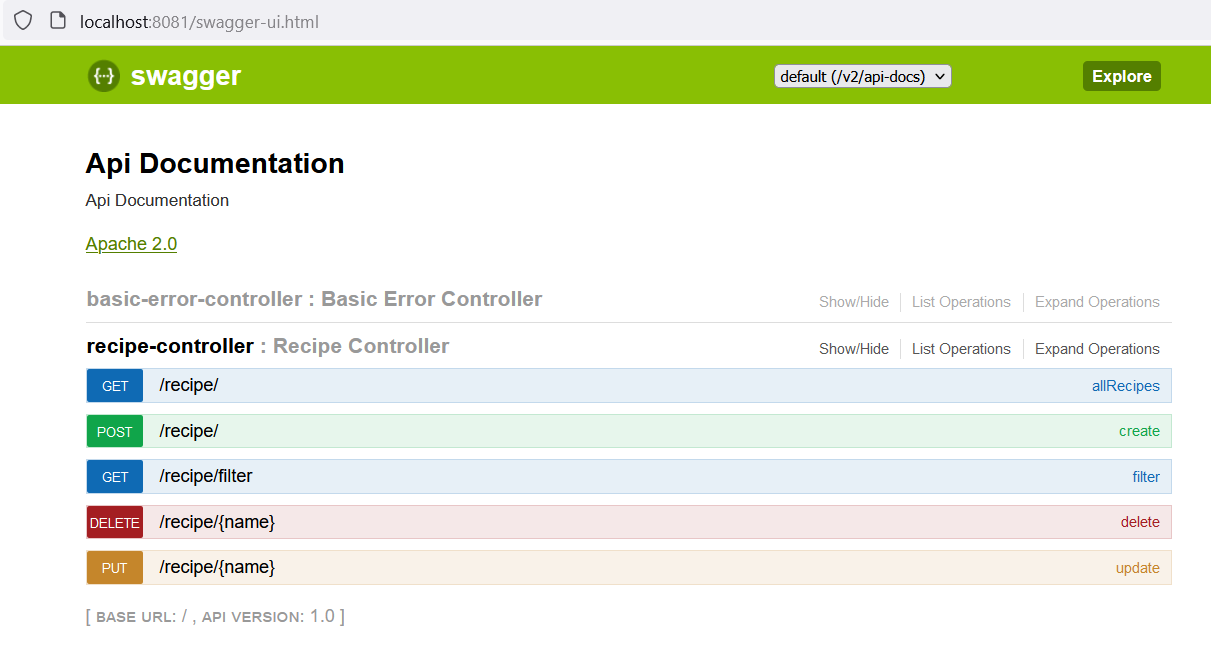
The application jar is stored in git hub public repository download the jar along with config folder and applicationStart.bat

<https://github.com/sivagurunathbabu/recipe-management-2.0.git>

* 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 as shown below. Also open the swagger API screen and see whether all API’s are listed.

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

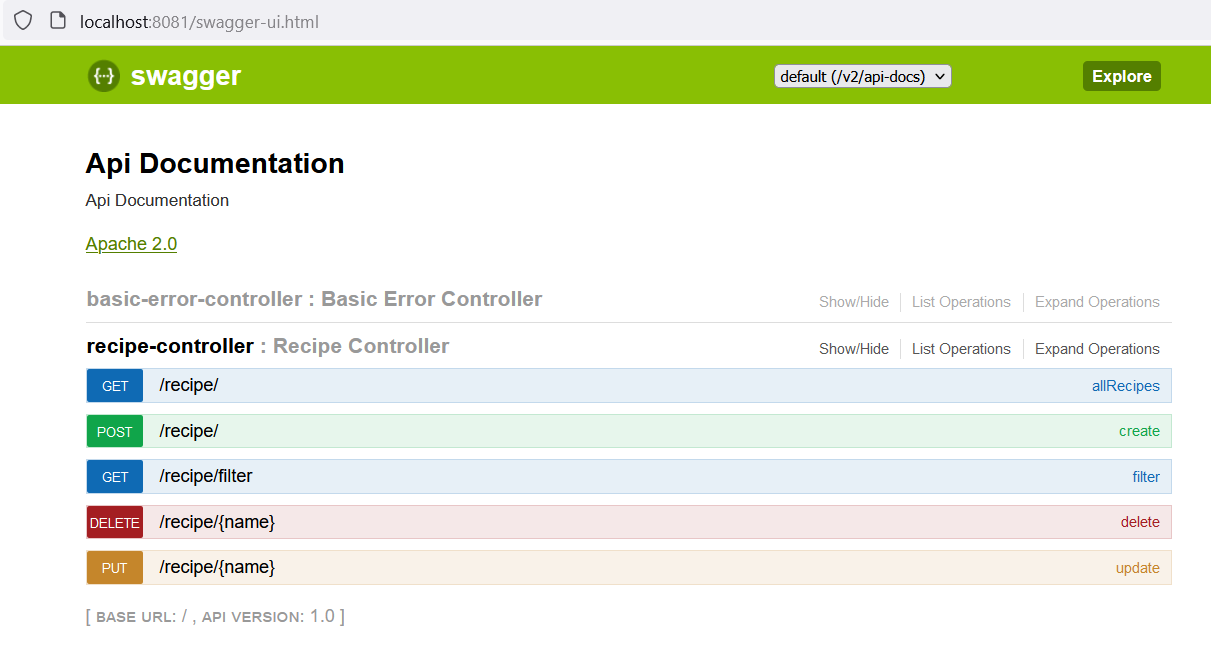




# Technology and IDE used:

* Technology Stack- Java, Spring Boot, Maven, Embedded 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. server.address=localhost
   2. server.port=8081
   3. spring.application.name=recipe-management
7. Application port is configured to 8081.
8. Launch the main application Application.java as SpringBoot application.
9. 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>



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

**Note:** Currently unit testing using @DataMongoTest not working in this embedded mongodb version hence integration test approach has been followed for all concreate classes.

1. **API Test Inputs/Data**

Input 1:

{

    "name": "Meat Potato Wedge",

    "noOfServings": 2,

    "ingredients": [

        "Potato",

        "Meat",

        "Chilly flakes"

    ],

    "instructions": "Marinate meat with spices and coat it with smashed potatoes and fry with oil",

    "category": "NON\_VEG"

}

Input 2:

{

    "name": "Meat Schnitzel",

    "noOfServings": 2,

    "ingredients": [

        "Egg",

        "Meat",

        "Breadcrumb"

    ],

    "instructions": "Cut slice of meat, coat it with egg and breadcrumb then fry until golden brown",

    "category": "NON\_VEG"

}

Input 3:

{

    "name": "Salmon Gravy ",

    "noOfServings": 4,

    "ingredients": [

        "Salmon",

        "Chilly",

        "Tomato"

    ],

    "instructions": "Marinate Salmon fish with olive oil, garlic and ginger. After 30 min of marination cook in oven for 10 min",

    "category": "NON\_VEG"

}

Input 4:

{

    "name": "Tower Dosa",

    "noOfServings": 1,

    "ingredients": [

        "Water",

        "Rice Flour",

        "Ghee"

    ],

    "instructions": "Mix Dosa batter with little water and fry the Dosa with ghee",

    "category": "VEG"

}

Input 5:

{

    "name": "Potato Wedge",

    "noOfServings": 2,

    "ingredients": [

        "Potato",

        "Breadcrumb",

        "Chilly flakes"

    ],

    "instructions": "Mix boiled Potato, breadcrumb and Chilly flakes and keep it in oven for 10 min",

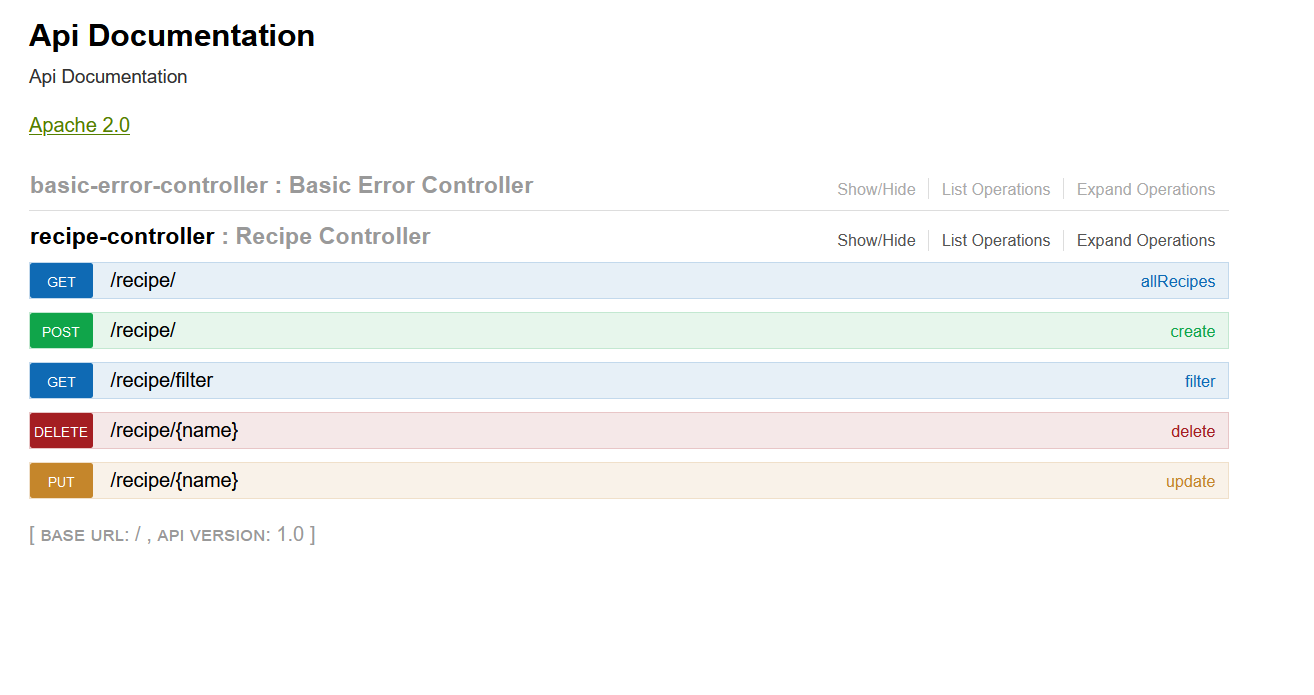
    "category": "VEG"

}

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.

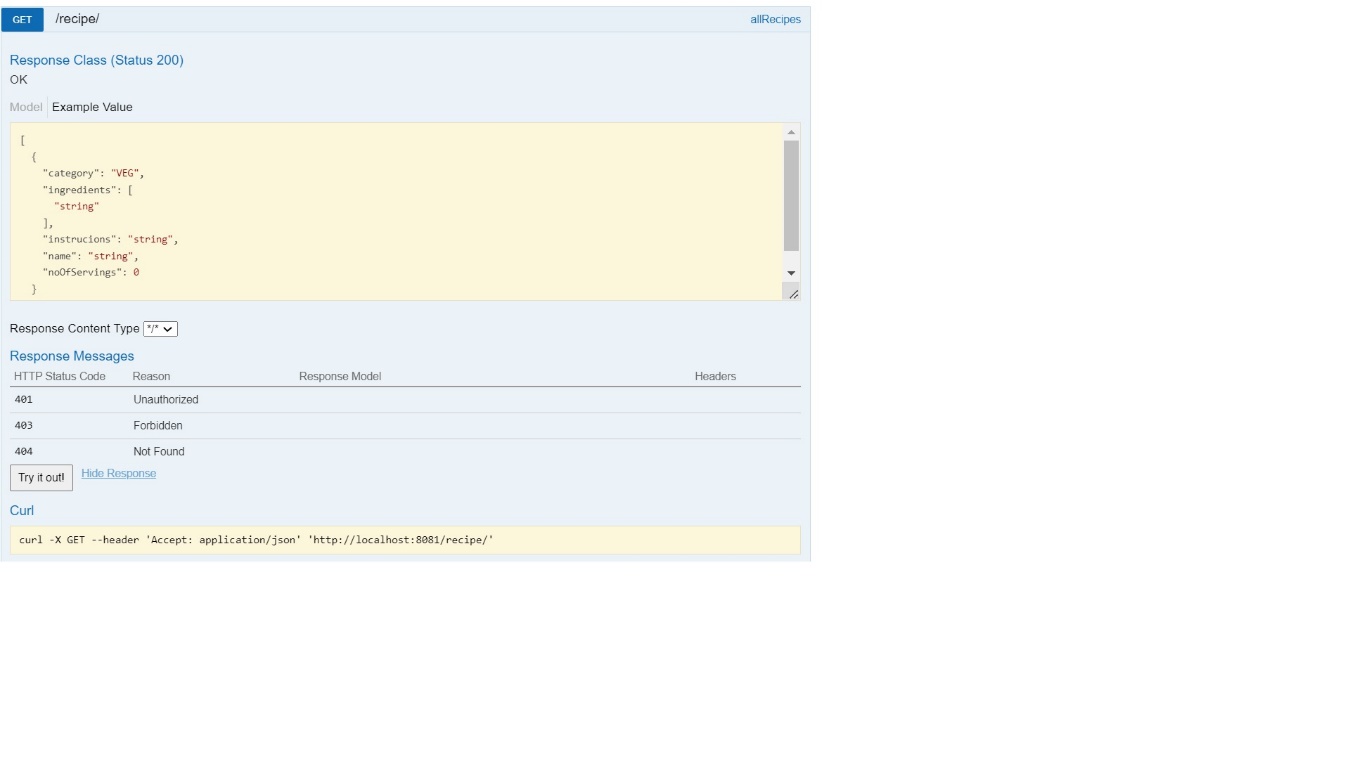
[http://localhost:8081/swagger-ui.html#](http://localhost:8081/swagger-ui.html)!



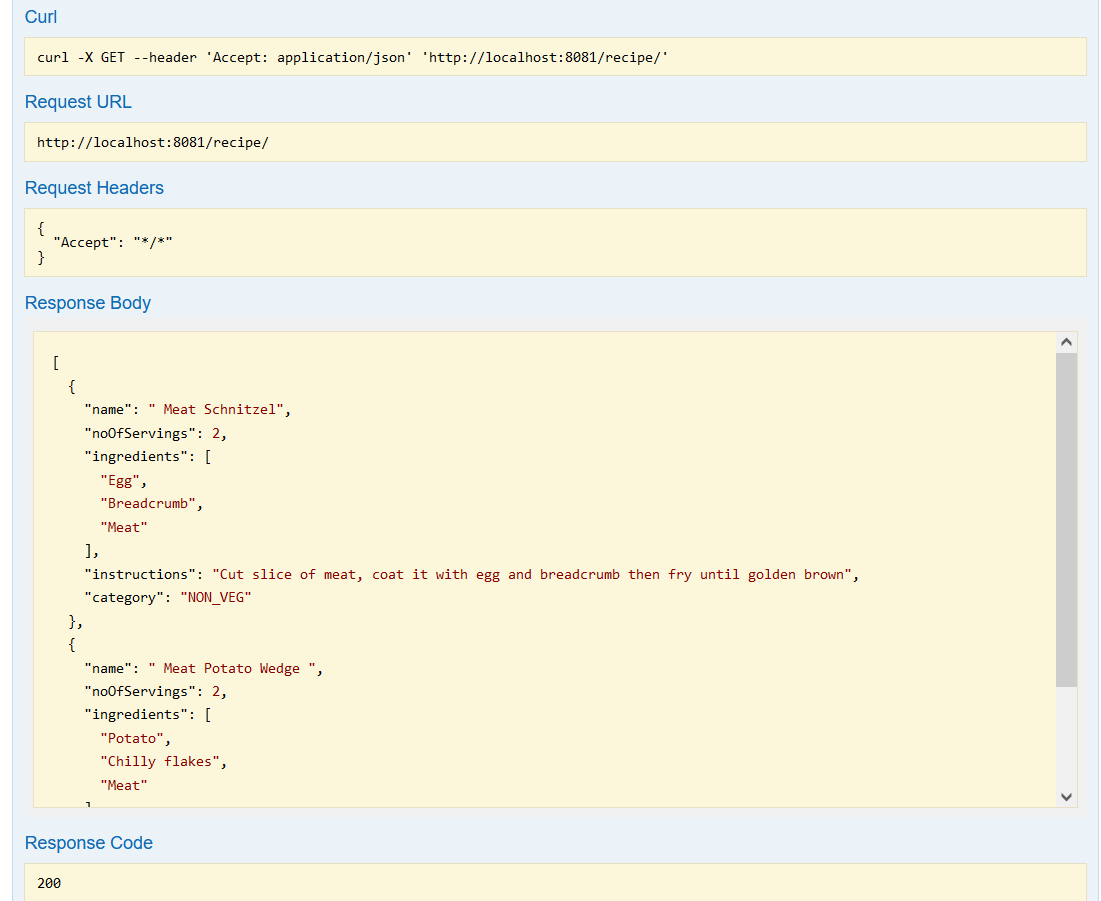
To get all recipes available:

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

Method: Get



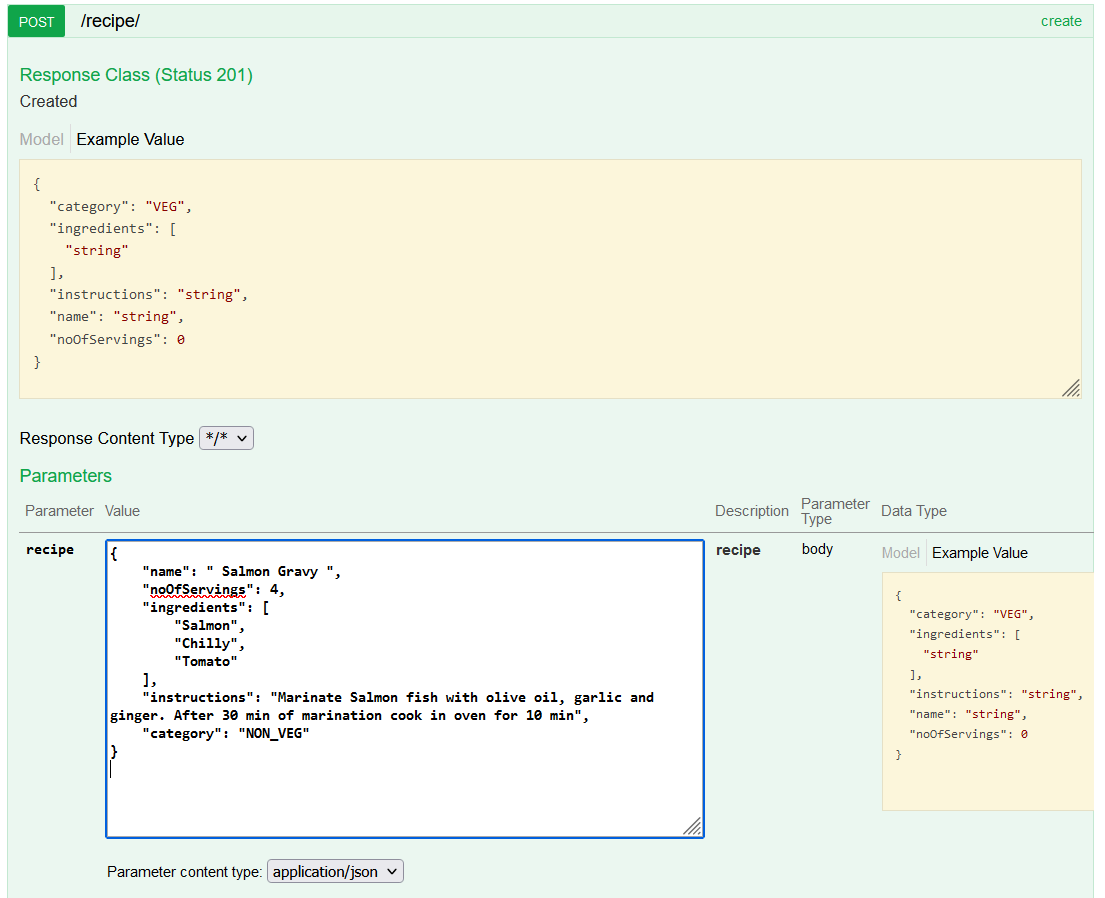
Response showing all the available recipes



To create a new recipe:

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

Method: Post



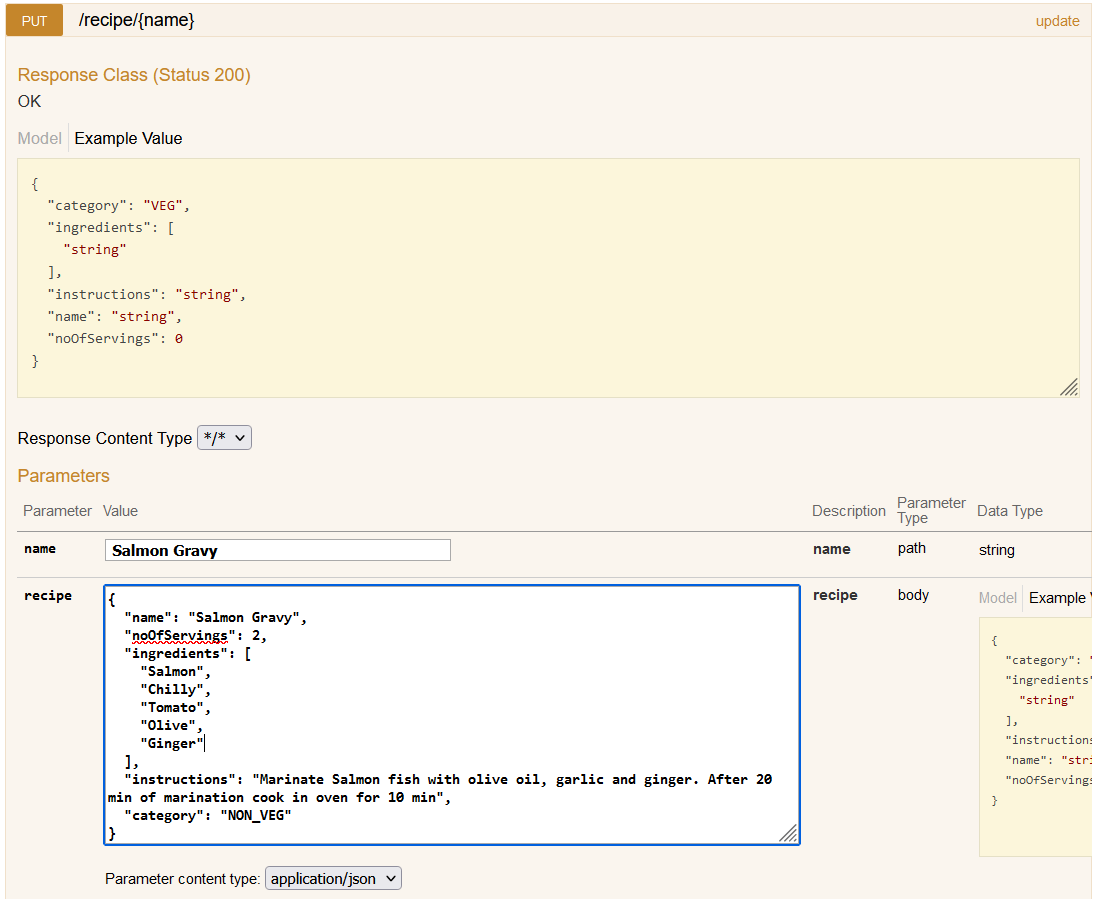
Response showing the saved recipe



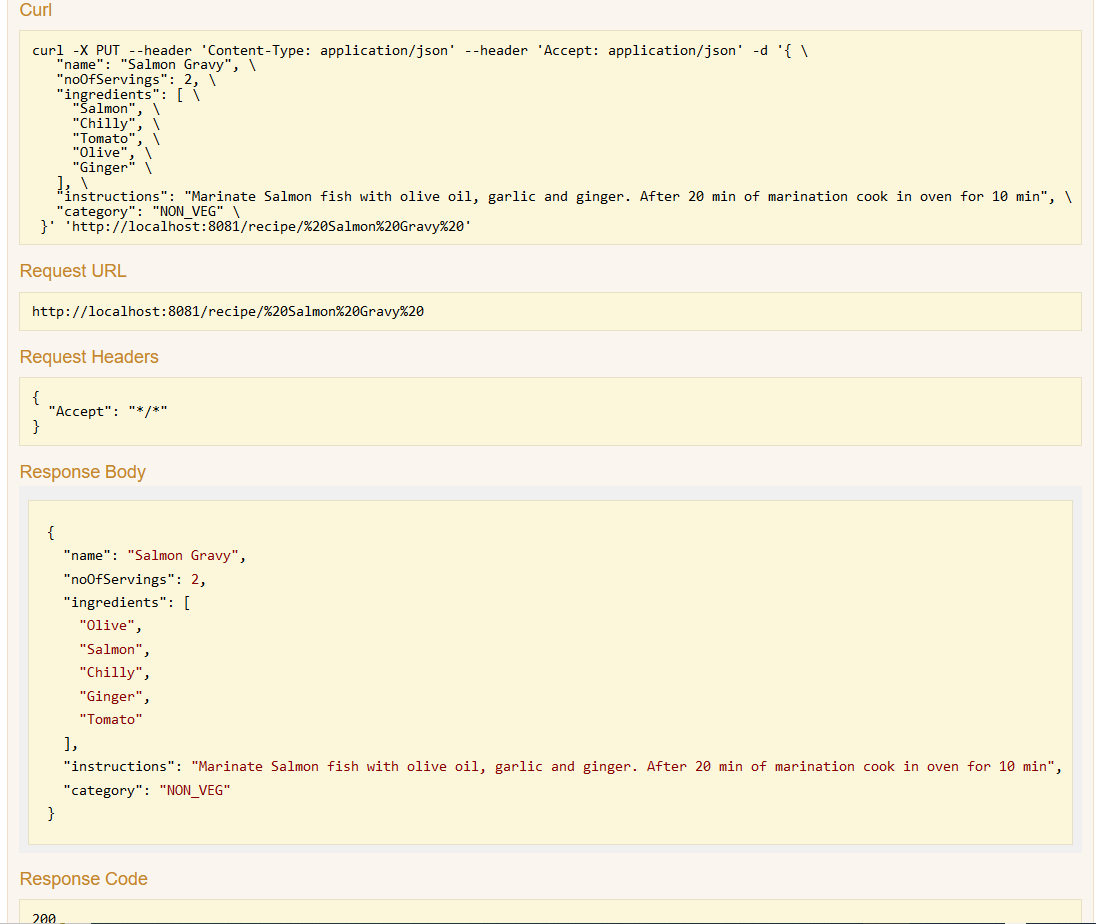
To edit/modify a recipe:

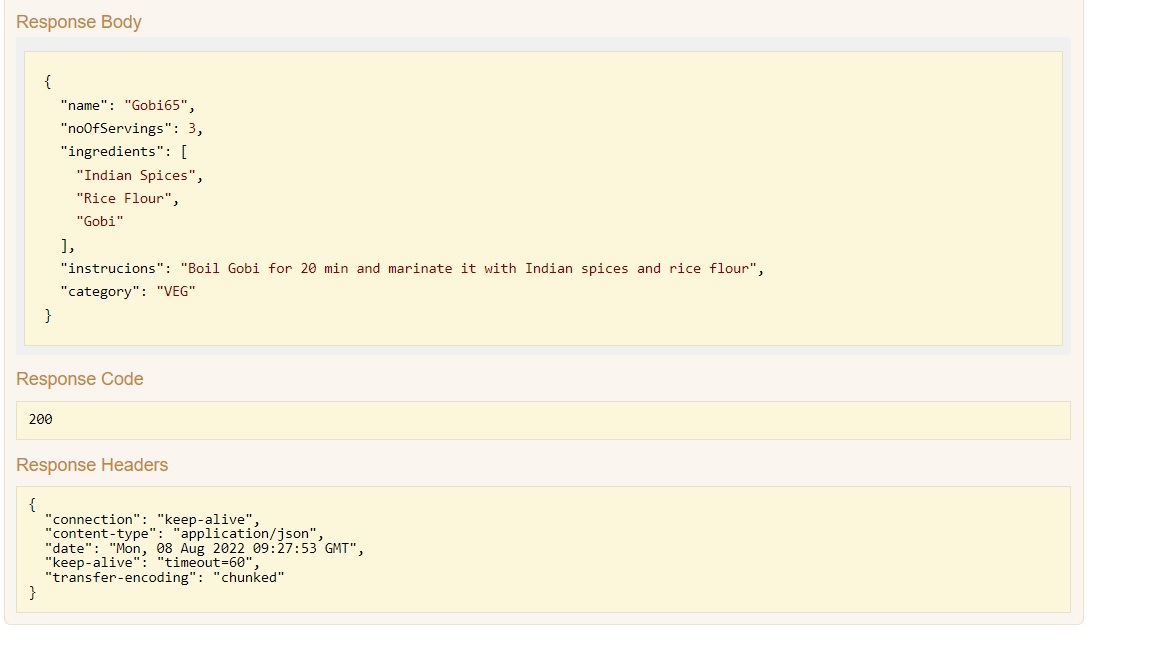
URI: <http://localhost:8081/recipe/>%20Salmon%20Gravy%20

Method: Put



Response showing the saved recipe

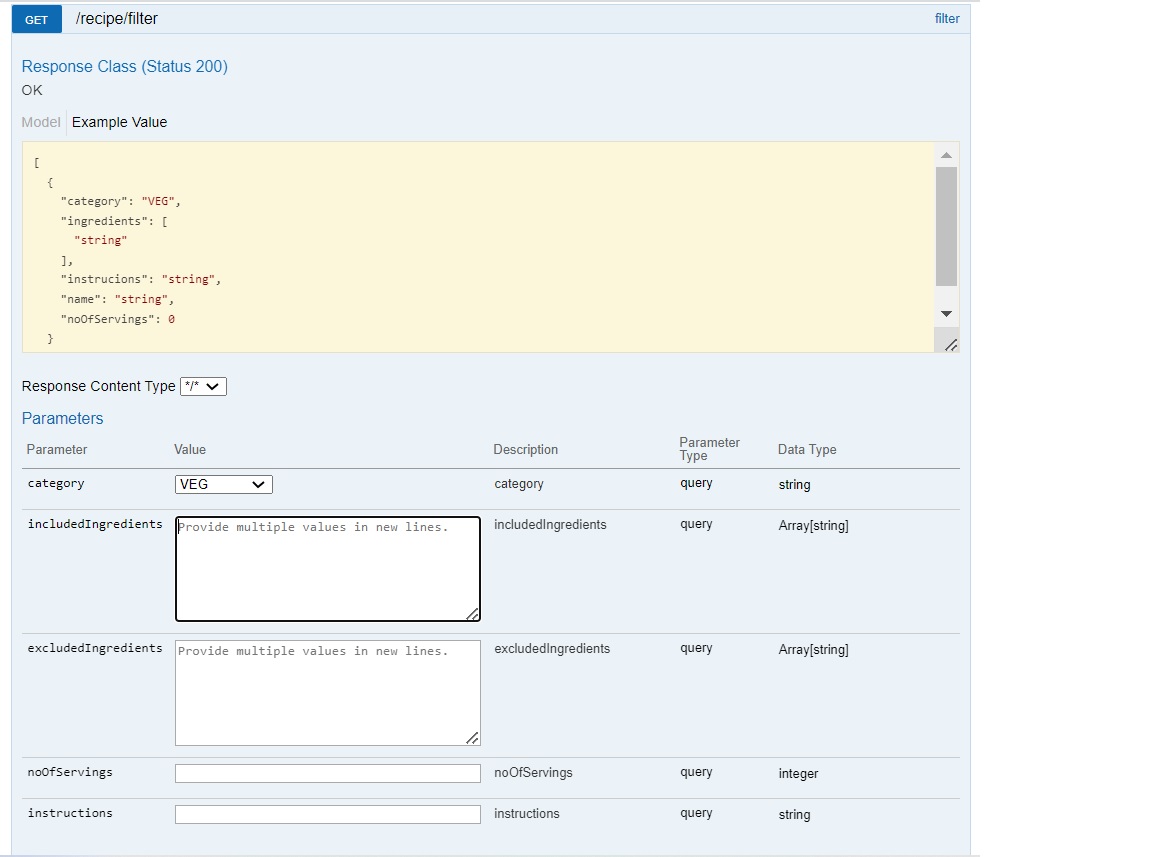




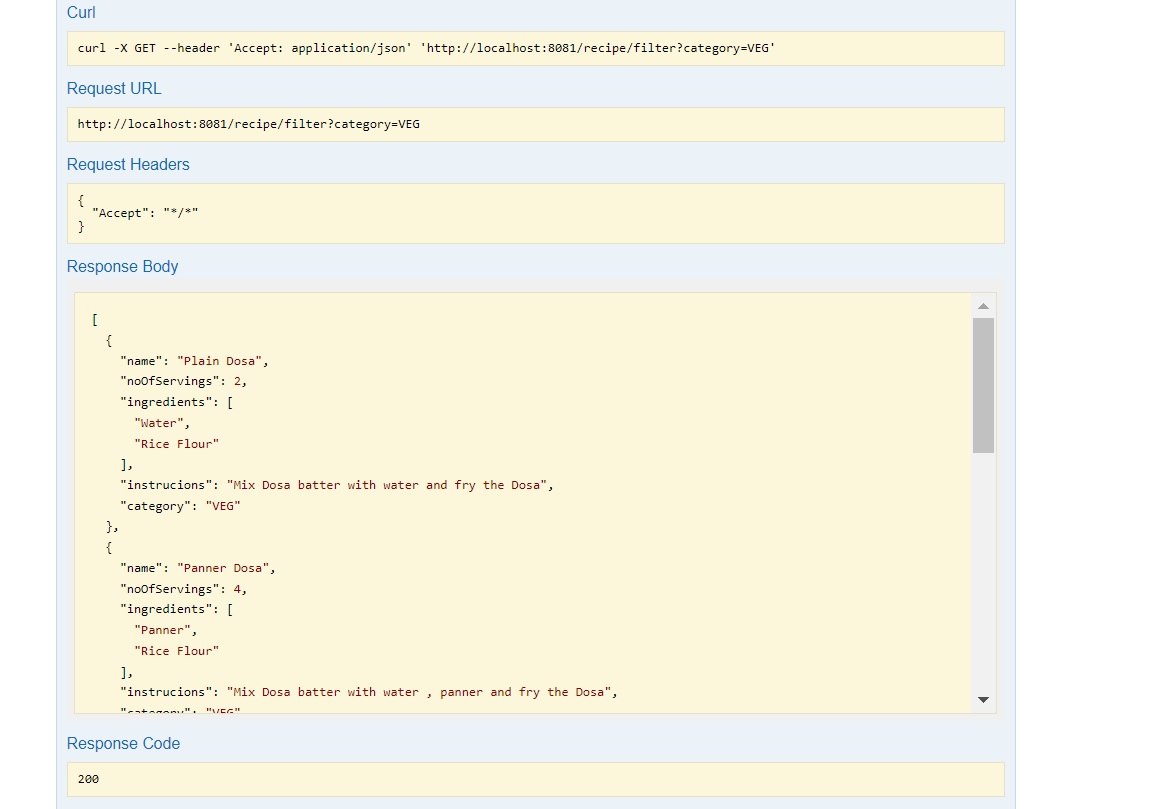
To filter a Veg recipe from the saved recipes:

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

Method: Get



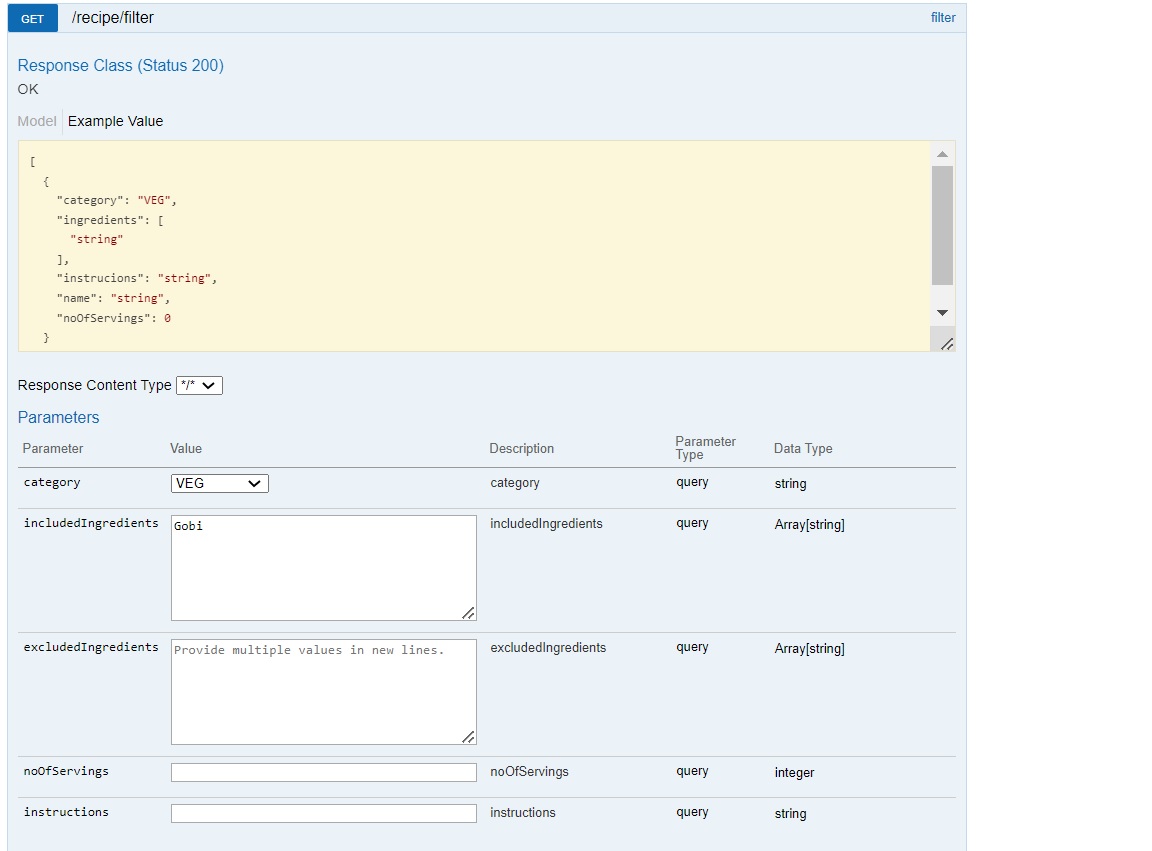
Response showing the filtered Vegetarian recipes.



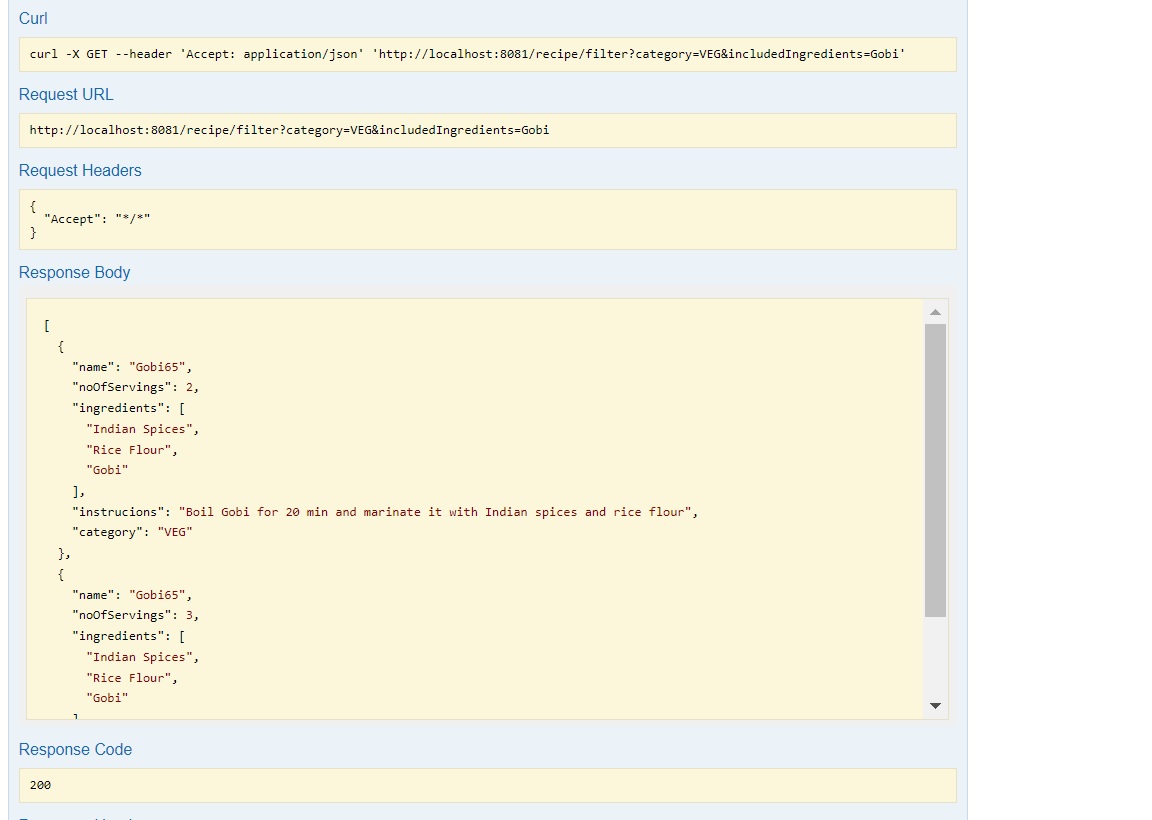
To filter a recipe with category Veg and ingredient Gobi65

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

Method: Get



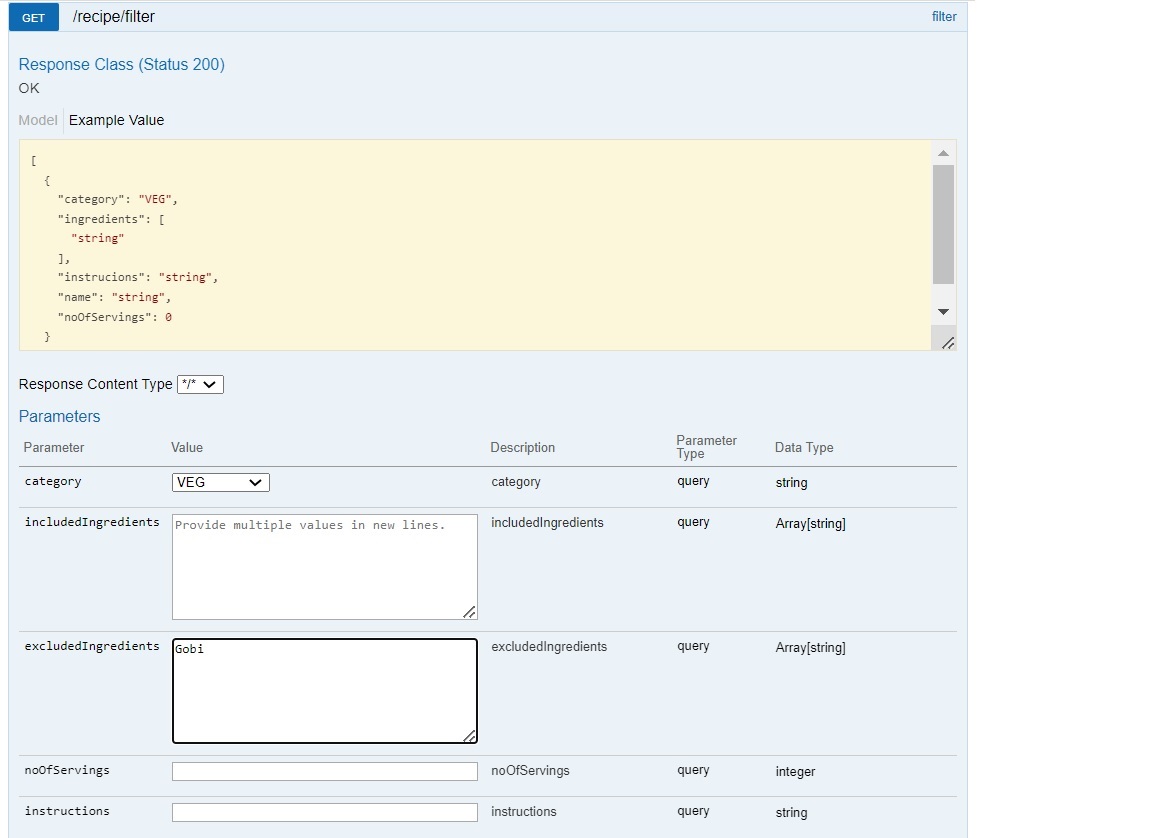
Response showing filtered recipe



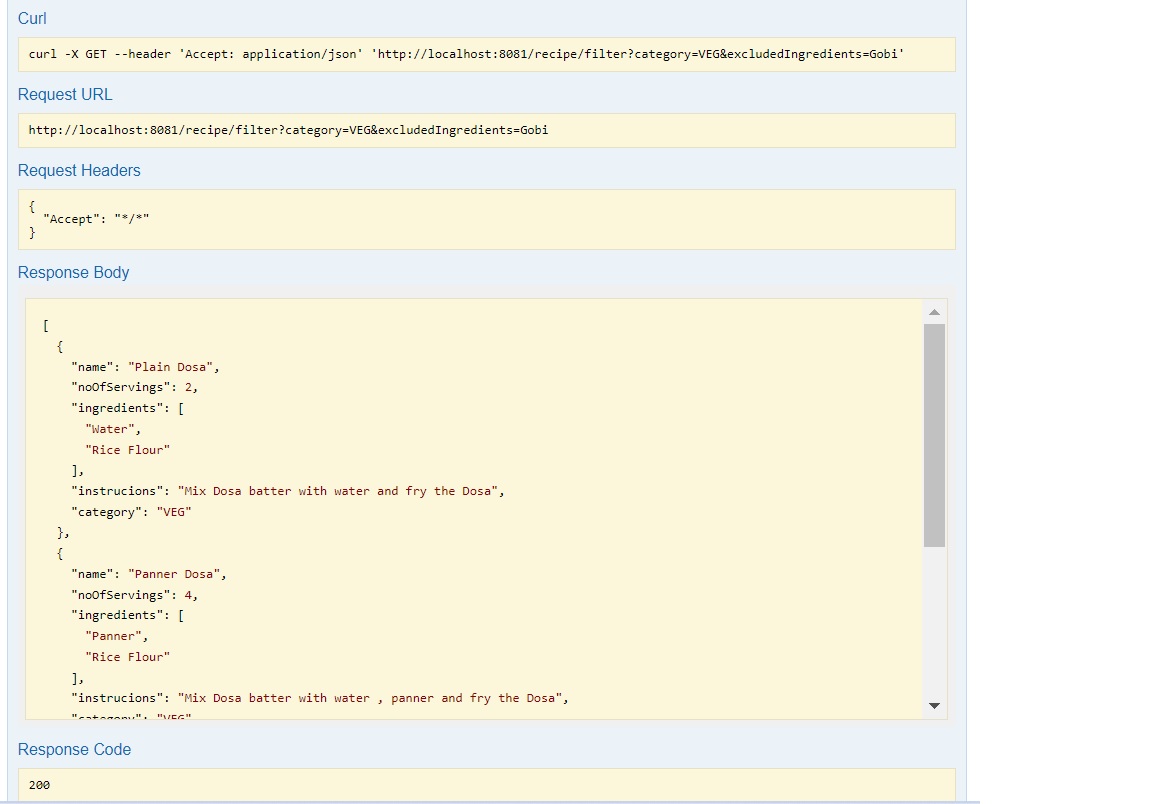
To filter a recipe with category Veg and without ingredient Gobi65

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

Method: Get



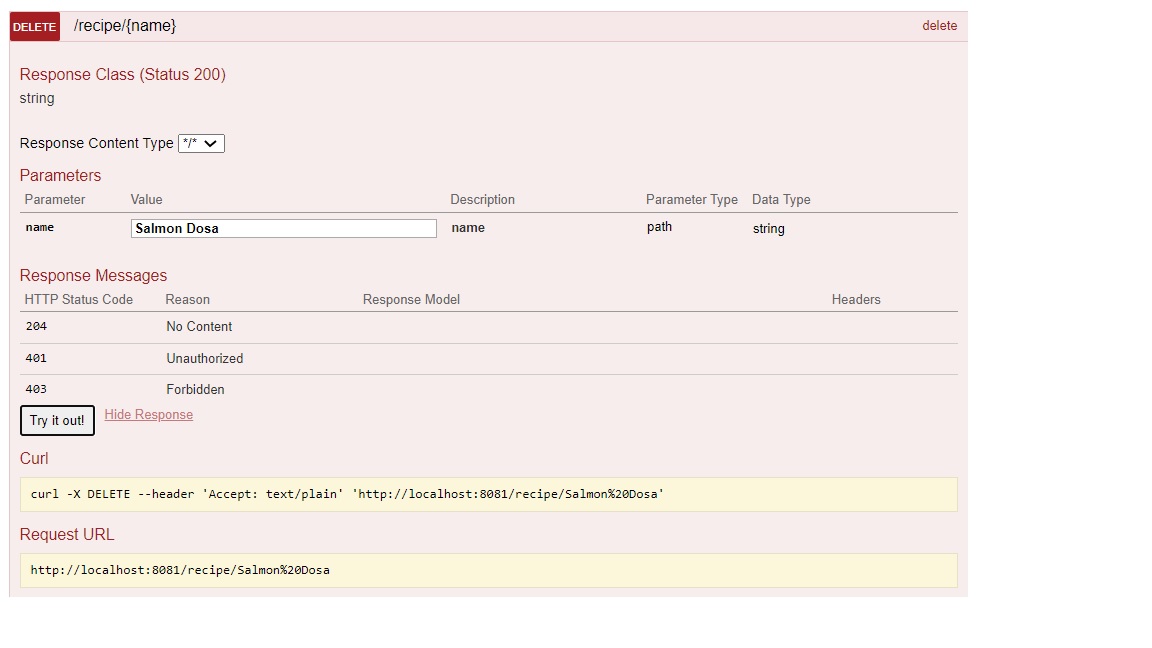
Response showing filtered recipe



To delete a recipe with name Salmon Dosa

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

Method: Delete



Response showing the deleted recipe

