**RMS – RECEPE MANAGEMENT SYSTEM**

In this document you will get to know about Recipe management system. You will also get to know that how to use this application and what are the various components.

**Overview of Steps**

1. Application Architecture

2. Review project structure

3. Maven POM file

4. Application configuration properties file

6. GET: Get a list of Recipes

7. GET: Get a single Recipes by id

8. POST: Add a new Recipe

9. PUT: Update an Recipe

10. DELETE: Delete an Recipes

11. GET, POST, PUT, DELETE urls

12. Database Structure and credentials

**1. Application Architecture**

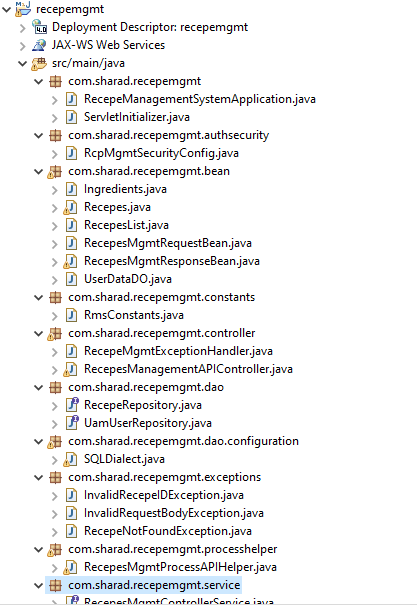
At a low-level, the application will have the following architecture.

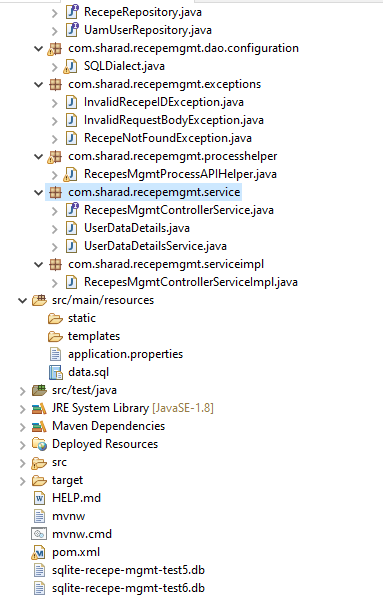
The RMS App REST will have a new Recipe Service implementation.

We will simply make calls to the CRM REST API to retrieve, delete and update the data.

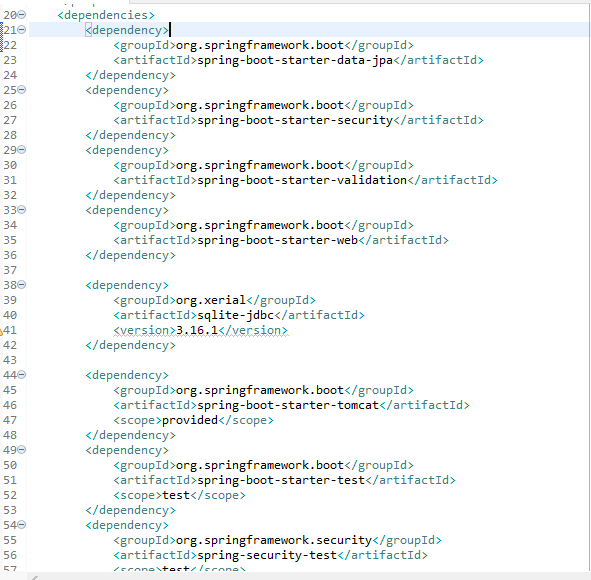
**2. Review Project Structure**

Here’s a description of the key components of the project.

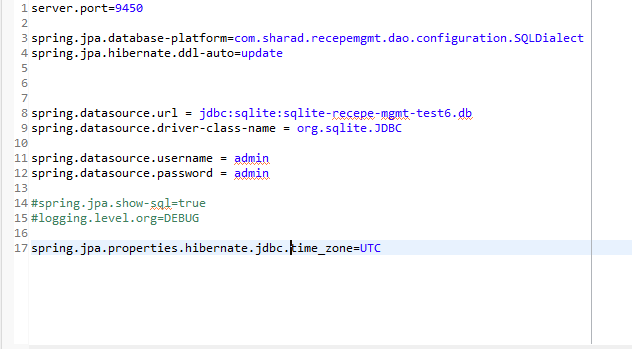




**3. Maven POM File**



**4. Application Configuration Properties**



**6. GET: Get a list of Customers**

The REST API exposes the following endpoint for getting a list of Recipes.

[**http://localhost:9800/restapi/recepes/**](http://localhost:9450/restapi/recepes/)

**Sample request:**

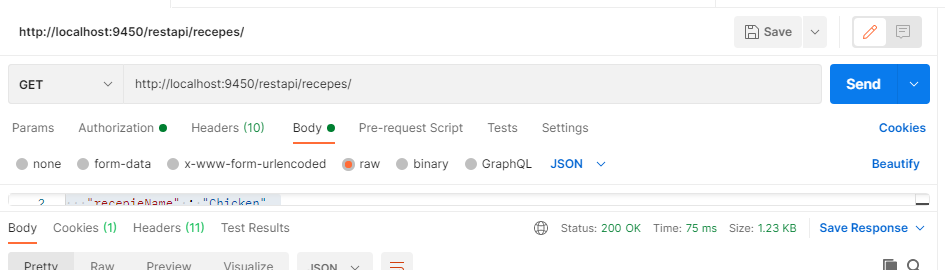
Please add Auth in header of POSTMAN as basic auth and provide

Username (sharad)and password(sharad)

For this url no need to set body.

Use Method type ‘GET’

Request:



**Response:**

{

    "responseStatus": "0",

    "responseDesc": "SUCCESS:Recepe Details have been successfully fetched",

    "recepiesDetails": **null**,

    "recepeList": [

        {

            "recepieId": 1,

            "recepieName": "Palak Paneer",

            "recepieCreationDate": "30‐06‐2021 13:08",

            "isVegeterian": "Y",

            "survingPeople": "17",

            "recIngredients": [

                {

                    "ingredentsId": 2,

                    "ingredents": "Palak"

                },

                {

                    "ingredentsId": 3,

                    "ingredents": "Paneer"

                }

            ],

            "cookingIntructions": "1.Boil chicken 2.boil rice 3. Mix both"

        },

        {

            "recepieId": 6,

            "recepieName": "Chicken",

            "recepieCreationDate": "30‐06‐2021 19:14",

            "isVegeterian": "N",

            "survingPeople": "18",

            "recIngredients": [

                {

                    "ingredentsId": 7,

                    "ingredents": "Chiken"

                }

            ],

            "cookingIntructions": "1.Boil chicken 2.boil rice 3. Mix both"

        }

    ]

}

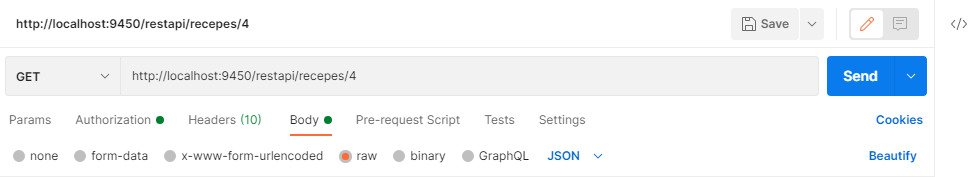
**7. GET: Get a single Recipe by ID**

The REST API exposes the following endpoint for getting a list of Recipes.

[**http://localhost:9800/restapi/recepes/1**](http://localhost:9450/restapi/recepes/1)

Here 1 refers to recipe id.

Use GET method for this as well.



Response:

{

    "responseStatus": "0",

    "responseDesc": "SUCCESS:Recepe Details have been successfully fetched",

    "recepeList": [

        {

            "recepieId": 1,

            "recepieName": "Palak Paneer",

            "recepieCreationDate": "30‐06‐2021 13:08",

            "isVegeterian": "Y",

            "survingPeople": "17",

            "recIngredients": [

                {

                    "ingredentsId": 2,

                    "ingredents": "Palak"

                },

                {

                    "ingredentsId": 3,

                    "ingredents": "Paneer"

                }

            ],

            "cookingIntructions": "1.Boil chicken 2.boil rice 3. Mix both"

 }

]

}

**8. POST: Add a new Recipe**

The REST API exposes the following endpoint for creating Recipes.

**http://localhost:9800/restapi/recepes/1**

Use POST method for this.

Sample Request:



Response:

{

"responseStatus": "0",

"responseDesc": "SUCCESS:Recepe Details have been successfully created",

"recepeList": null

}

**9. PUT: Update Recipe**

The REST API exposes the following endpoint for update Recipes.

**http://localhost:9800/restapi/recepes/1**

Use PUT method for this.

Here 1 refers to the recipe id.

**Sample Request:**



**Response**:

{

"responseStatus": "0",

"responseDesc": "SUCCESS:Recepe Details have been successfully updated",

"recepeList": null

}

**10. DELETE: DELETE Recipe**

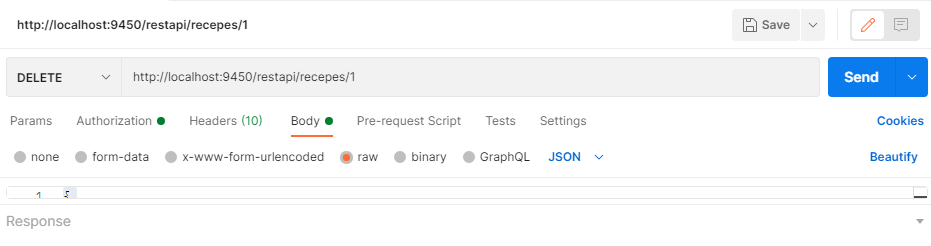
The REST API exposes the following endpoint for delete Recipes.

**http://localhost:9800/restapi/recepes/1**

Use delete method for this.

Here 1 refers to the recipe id.

Request:



**Response**:

{

"responseStatus": "0",

"responseDesc": "SUCCESS:Recepe Details have been successfully deleted",

"recepeList": null

}

**11. GET, POST, PUT, DELETE urls**

GET http://localhost:9800/restapi/recepes/

GET http://localhost:9800/restapi/recepes/1

POST http://localhost:9800/restapi/recepes/

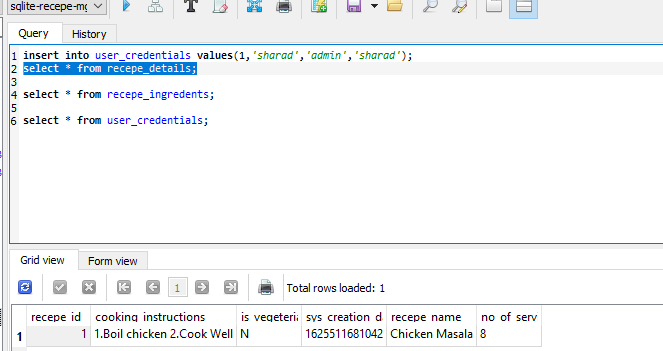
PUT http://localhost:9800/restapi/recepes/1

DELETE [http://localhost:9800/restapi/recepes/1](http://localhost:9450/restapi/recepes/1)

**12. Database Structure**

If want to see database, need to download SQLiteStudio from internet. It is open source.

Structure:



Commands to run application. (If not running in eclipse)

Extract the provided war and go to target directory.

Java –jar target/recepemgmt-0.0.1.war