




Work on project. Stage 4/5: Sort & update

Project: [Recipes](#)

Sort & update

 Hard  2 hours  469 users solved this stage. Latest completion was **about 1 hour ago**.

Description

In this stage, we continue with improving our application. It would be good to retrieve all recipes related to a category (beverages, salads, desserts, and so on), to search for a recipe by name, to update a recipe, or to find out when a recipe was uploaded or updated. To do that, you need two additional fields: `category` and `date`. One field is a recipe category, the other field stores the date. You also need to add two new endpoints. One endpoint will update recipes, the other receives the query parameters that will allow searching for recipes by a category or name.

Objectives

Don't forget to keep the functionality from the previous stages. This is what your program can do:

- `POST /api/recipe/new` receives a recipe as a JSON object and returns a JSON object with one `id` field;
- `GET /api/recipe/{id}` returns a recipe with a specified `id` as a JSON object;
- `DELETE /api/recipe/{id}` deletes a recipe with a specified `id`.

In this stage, the recipe structure should contain two new fields:

- `category` represents a category of a recipe. The field has the same restrictions as `name` and `description`. It shouldn't be blank;
- `date` stores the date when the recipe has been added (or the last update). You can use any date/time format, for example `2021-09-05T18:34:48.227624` (the default `LocalDateTime` format), but the field should have at least 8 characters.

Also, the service should support the following endpoints:











- `PUT /api/recipe/{id}` receives a recipe as a JSON object and updates a recipe with a specified `id`. Also, update the `date` field too. The server should return the `204 (No Content)` status code. If a recipe with a specified `id` does not exist, the server should return `404 (Not found)`. The server should respond with `400 (Bad Request)` if a recipe doesn't follow the restrictions indicated above (all fields are required, string fields can't be blank, arrays should have at least one item);
- `GET /api/recipe/search` takes one of the two mutually exclusive query parameters:
 - `category` – if this parameter is specified, it returns a JSON array of all recipes of the specified category. Search is case-insensitive, sort the recipes by date (newer first);
 - `name` – if this parameter is specified, it returns a JSON array of all recipes with the names that **contain** the specified parameter. Search is case-insensitive, sort the recipes by date (newer first).If no recipes are found, the program should return an empty JSON array. If 0 parameters were passed, or more than 1, the server should return `400 (Bad Request)`. The same response should follow if the specified parameters are not valid. If everything is correct, it should return `200 (Ok)`.

There is a couple of ways to do that. Check the examples below for details. If you need more theory on how to get data from a database or how to process query parameters, take a look at [Query Methods](#) paragraph in the Official Documentation (check the table), or the [@RequestParam](#) [annotation](#).

Examples

Example 1: `POST /api/recipe/new` request

6 / 6 Prerequisites

-  [LocalTime](#) 
-  [LocalDate](#) 
-  [LocalDateTime](#) 
-  [Comparing dates and time](#) 
-  [The utility class Collections](#) 

Show all

[Join a study group for the project Recipes](#)

Discuss your current project with fellow learners and help each other.

```
{
  "name": "Fresh Mint Tea",
  "category": "beverage",
  "description": "Light, aromatic and refreshing beverage, ...",
  "ingredients": ["boiled water", "honey", "fresh mint leaves"],
  "directions": ["Boil water", "Pour boiling hot water into a mug", "Add fresh mint leaves", "Mix and let the mint
5 minutes", "Add honey and mix again"]
}
```

Response:

```
{
  "id": 1
}
```

Further `GET /api/recipe/1` response:

```
{
  "name": "Fresh Mint Tea",
  "category": "beverage",
  "date": "2020-01-02T12:11:25.034734",
  "description": "Light, aromatic and refreshing beverage, ...",
  "ingredients": ["boiled water", "honey", "fresh mint leaves"],
  "directions": ["Boil water", "Pour boiling hot water into a mug", "Add fresh mint leaves", "Mix and let the mint
5 minutes", "Add honey and mix again"]
}
```

Hint

Example 2: `PUT /api/recipe/1` request

```
{
  "name": "Warming Ginger Tea",
  "category": "beverage",
  "description": "Ginger tea is a warming drink for cool weather, ...",
  "ingredients": ["1 inch ginger root, minced", "1/2 lemon, juiced", "1/2 teaspoon manuka honey"],
  "directions": ["Place all ingredients in a mug and fill with warm water (not too hot so you keep the beneficial h
10 minutes", "Drink and enjoy"]
}
```

Further response for the `GET /api/recipe/1` request:

```
{
  "name": "Warming Ginger Tea",
  "category": "beverage",
  "date": "2021-04-06T14:10:54.009725",
  "description": "Ginger tea is a warming drink for cool weather, ...",
  "ingredients": ["1 inch ginger root, minced", "1/2 lemon, juiced", "1/2 teaspoon manuka honey"],
  "directions": ["Place all ingredients in a mug and fill with warm water (not too hot so you keep the beneficial h
10 minutes", "Drink and enjoy"]
}
```

Example 3: A database with several recipes

```
{
  "name": "Iced Tea Without Sugar",
  "category": "beverage",
  "date": "2019-07-06T17:12:32.546987",
  ....
},
{
  "name": "vegan avocado ice cream",
  "category": "DESSERT",
  "date": "2020-01-06T13:10:53.011342",
  ....
},
{
  "name": "Fresh Mint Tea",
  "category": "beverage",
  "date": "2021-09-06T14:11:51.006787",
  ....
},
{
  "name": "Vegan Chocolate Ice Cream",
  "category": "dessert",
  "date": "2021-04-06T14:10:54.009345",
  ....
},
{
  "name": "warming ginger tea",
  "category": "beverage",
  "date": "2020-08-06T14:11:42.456321",
  ....
}
}
```

Response for the `GET /api/recipe/search/?category=dessert` request:


```
[
  {
    "name": "Vegan Chocolate Ice Cream",
    "category": "dessert",
    "date": "2021-04-06T14:10:54.009345",
    ....
  },
  {
    "name": "vegan avocado ice cream",
    "category": "DESSERT",
    "date": "2020-01-06T13:10:53.011342",
    ....
  },
]
```

Response for the `GET /api/recipe/search/?name=tea` request:

```
[
  {
    "name": "Fresh Mint Tea",
    "category": "beverage",
    "date": "2021-09-06T14:11:51.006787",
    ....
  },
  {
    "name": "warming ginger tea",
    "category": "beverage",
    "date": "2020-08-06T14:11:42.456321",
    ....
  },
  {
    "name": "Iced Tea Without Sugar",
    "category": "beverage",
    "date": "2019-07-06T17:12:32.546987",
    ....
  },
]
```

Search is case-insensitive, the recipes are sorted by date.

 Report a typo

 See hint

 Write a program

[Code Editor](#)

[IDE](#)

CONNECTION STATUS

✓ IDE is responding IntelliJ IDEA 2022.1.2

✓ EduTools plugin is responding 2022.5-2022.1-343

Solve in IDE

 Synchronizing IDE may take a while

Look up solution (💎 100)

[Comments \(30\)](#)

[Hints \(12\)](#)

[Useful links \(3\)](#)

[Solutions \(104\)](#)

[Show discussion](#)