Java Backend test

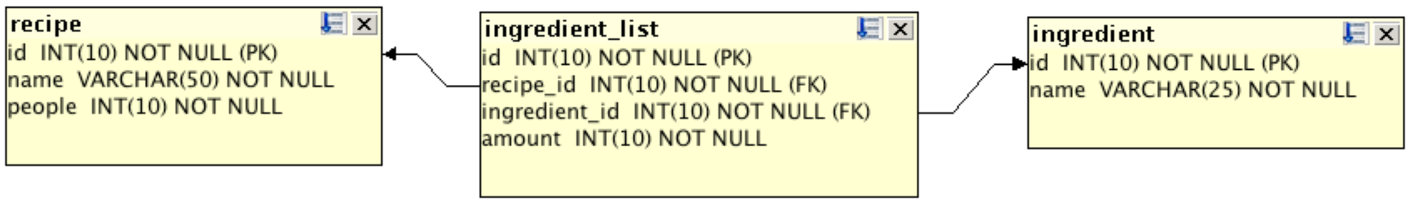
Backend test The task is to create an API in for example `Spring Boot`. A database dump is provided in data.sql. It contains table creation and value insertion. All you need before is a database. The database contains recipes and ingredients for these recipes. It is made up by 3 tables:

Java Backend test

Requirements

Endpoints

- `GET: /api/recipe/{recipe_id}`
- `POST: /api/recipe`
- `PUT: /api/recipe/{recipe_id}`
- `GET: /api/recipe?q={search_string}`
- `GET: /api/ingredient`
- `GET: /api/ingredient/{ingredient_id}`
- `POST: /api/ingredient`
- `PUT: /api/ingredient/{ingredient_id}`
- `GET: /api/ingredient?q={search_string}`



- A recipe contains ingredients in different amount.
- The recipe table contains an id, the name of a recipe and how many people it serves.
- The ingredient table contains an id and the name of the ingredient.
- The ingredient_list table connects the other two tables to form a recipe. The amount column says how much of a certain ingredient you should have in the specific recipe.

Requirements

- We want an web application that is self contained, like a Spring Boot project with this stack:
  - An in-memory database like H2 (you still can use other stuff but the code should be self contained and runnable)
  - Embedded container (Jetty/Tomcat/Wildfly) running on port 8080
  - Database initialization should take place in the code itself
  - The code should be buildable just by the build tool (Maven/Gradle) so you shouldn't bundle IDE specific stuff and all the dependencies should be fetched by the build tool.
  - The code should be clean and be structured well and we check how you document your code
- Not necessary but a plus
  - If you provide a way of integeration testing (the test should be runnable by Maven/Gradle and be self contained)

Tip. This is a nice place to find out more about Spring Boot: https://projects.spring.io/spring-boot/

Endpoints

- `Content-Type` for both requests and responses should be `application/json`

- When doing an operation on an ID that doesn't exist, an HTTP 404 should be returned.
- Where there are no example JSONs, design them yourself.

The API should have the following endpoints:

- Get/put/post for the ingredients
- Get/put/post for the recipe including the ingredient list
- Search for recipe by name
- Search for ingredient by name

as described below:

`GET: /api/recipe/{recipe_id}`

This request should also support an optional queryparam (?p={people}) that recalculates the recipe to fit for that amount of people. Example response:

```
{
  "id": 1,
  "name": "Recipe_1",
  "people": 2,
  "ingredients": [
    {
      "id": 1,
      "name": "Carrot",
      "amount": 5
    },
    {
      "id": 2,
      "name": "Salt",
      "amount": 2
    }
  ]
}
```

`POST: /api/recipe`

This endpoint should consume JSON to create a new recipe and produce a JSON response with the new id.

Example request body:

```
{
    "name": "New Recipe",
    "people": 4,
    "ingredients":
        [
          {
            "id": 1,
            "amount": 2
          },
          {
            "id": 4,
            "amount": 6
          }
        ]
}
```

`PUT: /api/recipe/{recipe_id}`

Endpoint to update an already existing recipe. Example request body: * assume that the ingredients are already created.

```
{
  "name": "New recipe name",
  "ingredients": [
    {
      "id": 1,
      "amount": 4
    }
  ]
}
```

`GET: /api/recipe?q={search_string}`

Search endpoint to search on recipe name. All recipes with names containing the search string should be returned. Example response:

```
{
"recipes":[
      {
        "id": 1,
        "name": "Recipe_1",
        "people": 2,
        "ingredients": [
          {
            "id": 1,
            "name": "Carrot",
            "amount": 5
          },
          {
            "id": 2,
            "name": "Salt",
            "amount": 2
          }
        ]
      }
   ]
}
```

`GET: /api/ingredient`

This endpoint should return the name of all ingredients and their id.

`GET: /api/ingredient/{ingredient_id}`

Return ingredient name and id.

`POST: /api/ingredient`

Create a new ingredient.

`PUT: /api/ingredient/{ingredient_id}`

Changed the name of the ingredient.

`GET: /api/ingredient?q={search_string}`

Search for ingredient name. All ingredients with names containing the search string should be returned.