JustRecipe

Group Project for Large Scale and Multi-Strutured Databases

Francesco Campilongo
Daniele Cioffo
Francesco Iemma

Academic Year 2020/21

Contents

Introduction

Chapter 1

Dataset

In this first chapter of this document we will talk about searching for the initial dataset.
As specified in the project documentation, the dataset had to be at least 50MB large, and this quantity could not be generated directly within the application. So we did an initial search, finding two datasets, which were generated by their authors by performing the scraping on the sites *www.FoodNetwork.com* and *www.Epicurios.com*. The second dataset was more complete (more nutritional values), so it was used as the main dataset. The other dataset was used to complement the other, reaching a total of 67.8 MB, with 45349 recipes.
To correctly extract the data present in the two datasets we wrote a program in Java, called *RecipeReade*r, thanks to which we adapted the two different formats and removed the duplicates (recipes with the same title that were present in both datasets). To implement this program we used the GSon library and the Jackson library.
The variety property is ensured by using two different sources. The velocity / variability properties are ensured because comments and recipes are eliminated and added inside the application, indeed this data may lose importance after a certain time interval since new data quickly arrives.

Chapter 2

Design

2.1 Introduction To The Application

The topic of cuisine is extensively widespread in our society. In fact we can think at the success achieved by tv shows related to cooking in the last years and also at the fact that a lot of chefs are becoming superstars. Then there is another important factor: the coronavirus outbreak.
With the coronavirus outbreak a lot of people became cuisine lovers, in fact at the first moments of the pandemia several ingredients as flour and yeast were very hard to find, because people were confined in their home and so they had more free time.
But this topic is not a recente one. The first recipe book dates back to eigth century B.C. and it is the so-called *Eraclio* (by the name of the city in which he was found). Then also an important latin writer, Apicio, wrote one of the most important recipe books of the roman era: *De Re Coquinaria* which dates back to the first century B.C..
So the topic of cuisine is inherent to human nature, because the necessity of eating is a basic need. Furthermore, everyone has experimented the infamous question: "What will I eat this evening?". JustRecipe has the aim of answer to this question, it has the aim of helping university student or workers to retrieve and to do fast and simple recipes.
So this application is basically a recipe book but it is also more than this.
JustRecipe is also a social network which allow people to enjoy, to ex-change ideas about cooking, to feel less lonely in this hard period.

2.2 Requirements

2.2.1 Main Actors

The main actors of the application are four:

- Unregistered User

  He is the user which open the application for the first time, in order to access he must sign-up.

- User

  He is the normal user (the registered one).

- Moderator

  He is in charge of controlling the comments and eventually delete the ones which contain abuses.

- Administrator

  He is the most powerful actor, he can delete users and recipes and he is also in charge of elect moderators

Each actor can do all the features of the previous ones in the list.

2.2.2 Functional Requirements

Features offered to the Unregistered User

- Registration

  In order to access the application an user must sign-up. Otherwise he is not allowed to access and to use all the functionalities.

Features offered to the Registered User

- Login/Logout

  The only way to access the application, as we said previously, is to sign-up and login. At the end the user can logout and close the session.

- Search a recipe

  It's possible to search a recipe searching for the title and for categories.

- Browse suggested recipes

  The suggestions will be offered in a proper section, they are done considering the relationships between the user logged, the users followed by the user logged and so on so forth.

- Browse recipes of following users

  In a proper section (i.e. the Homepage) the user can browse the recipes of the following user. Indeed he can see only a snapshot of the recipes. If he wants a more in-depth view, he can click on it and see the recipe page in which he is able to see all the recipe details.

- Add a recipe

  The user can insert a new recipe.

- Edit own recipes

  The user can edit the recipes previously added by himself.

- Comment recipes

  Every user can make a comment about recipes

- Follow another user

  The most important feature of each social network: the users can follow each others.

- Like a recipe

  In order to evaluate a recipe each user can like its.

Features offered to the Moderator

- Delete comments

  The moderator is in charge of delete comments which contain racist abuse, crude terms and so on so forth.

Features offered to the Administrator

- Delete users

  The admin can delete the users which don't respect the application guidelines.

- Delete recipes

  The admin can delete recipes not correctly inserted

- Elect moderators

  In order to handle better the application, the admin can elect some users as moderators.

2.2.3 Non-Functional Requirements

2.2.4 Actors and Use Cases

The use case diagram of the application is described in the figure 2.1

Figure 2.1: Use Case Diagram

Some observations on the diagram are necessary:

- The circles in blue are the ones which described actions available only for the owner of the object on which the actions are applied.

  So, in detail, this means that a **User** can edit/delete a profile if and only if he owns this profile. Then he can edit a recipe and/or a comment if and only if he adds that recipe or that comment.

- When we are seeing the recipe detail we can go on the user that have been added that recipe. So the extend relation between *View Recipe* and *View User* means this.

- The action *Browse Recipes of following users* is available only if the **User** follows at least one user. Otherwise he can start to follow users and only after this he can see suggested

recipes (*Browse Suggested Recipes*). In this case, due to the fact that the user follows nobody, he will see the most famous recipes in general because it's impossible to suggest specific recipes due to the fact that he has no following and no likes or comments.

- The actions in red are the ones that can be performed only by the **Administrator**
- The actions in orange are the ones that can be performed only by the **Moderators**

2.3 UML Class Diagram

Let analyze the UML Class Diagram. There are three main entities: User, Comment, Recipe. It's important to point out that the **User** of the Use Case Diagram is here the so-called *Registered User* and the *User* of this diagram is the generic user. Then we undeline the fact that, in order to represent the three actors of the use case diagram, a generalization is needed.

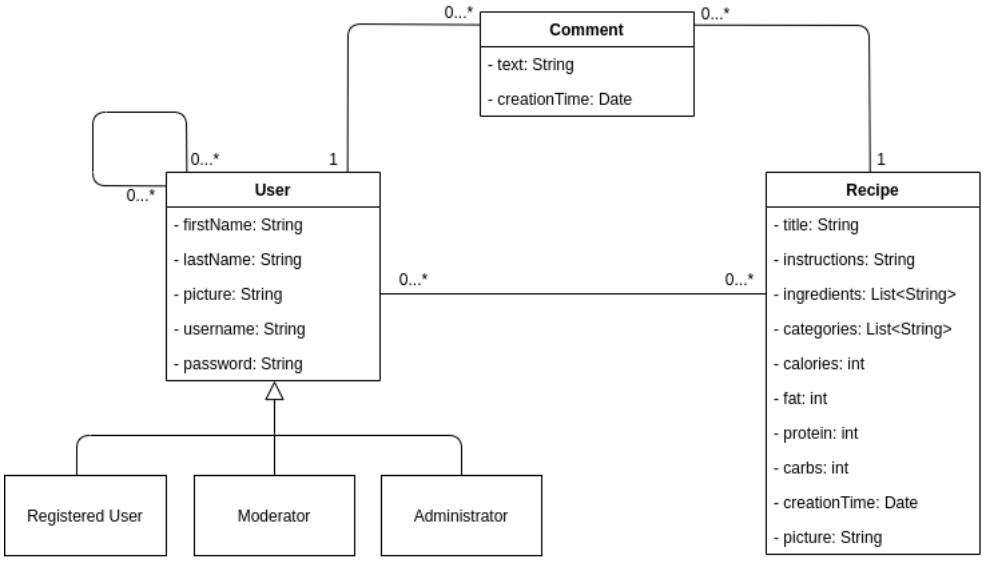

Figure 2.2: UML Analysis Classes Diagram with generalization unsolved

Observing the figure 2.2 it's possible to understand that we can resolve the generalization putting an attribute in the entity *User*. It is an integer and we call it *role*: if it's a *Normal User* role is 0; if *Moderator* then 1; if *Administrator* then 2.

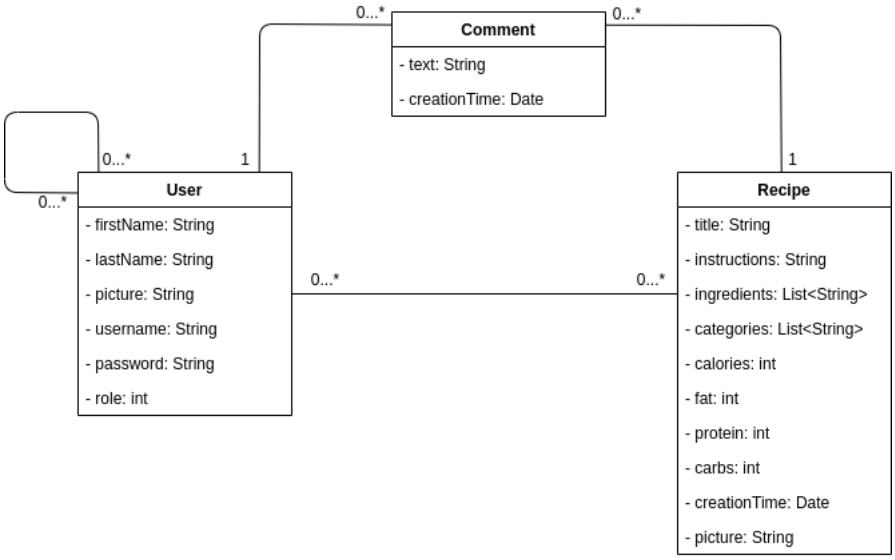

Figure 2.3: UML Analysis Classes Diagram

2.4 Data Model

2.4.1 DocumentDB

```
{
  "_id":
      {"$oid": "5fdb5fd86796ee4e73ef5b84"},
  "title":
      "Lentil, Apple, and Turkey Wrap ",
  "instructions":
      "1. Place the stock, lentils, celery, carrot, thyme, and
      salt in a medium saucepan and bring to a boil. Reduce heat
      to low and simmer until the lentils are tender, about 30
      minutes, depending on the lentils. (If they begin to
      dry out, add water as needed)..." ,
  "ingredients":
      ["4 cups low-sodium vegetable or chicken stock", "1 cup
         dried brown lentils", ...],
  "categories":
      ["Sandwich", "Bean", "Fruit", "Tomato", "turkey", ...],
  "calories":
      426,
  "fat":
      7,
  "protein":
      30,
  "carbs":
      20,
  "creationTime":
      {
        "$date": "2020-12-17T13:40:40.658Z"
      },
  "authorUsername":
      "oscar.evans",
  "picture":
      "https://assets.epicurious.com/photos/551
         b0595e7851a541a30b23f/master/pass/239173_lentil-apple-
         and-turkey-wrap_6x4.jpg",
  "comments":
      [
        {
          "authorUsername": "oliver.smith",
          "text": "Very good!!!",
          "creationTime":
          {
            "$date": "2020-12-17T13:50:40.658Z"
          }
        },
        {
          "authorUsername": "jessica.evans",
          "text": "Fantastic",
          "creationTime":
          {
            "$date": "2020-12-17T13:52:40.658Z"
          }
        },....
      ]
}
```

2.4.2 GraphDB

2.5 Distributed Database Design

2.5.1 Replicas

2.5.2 Sharding

2.6 Software Architecture

Chapter 3

Implementation and Test

3.1 Main Modules

3.2 Main Packages and Classes

3.3 Most Relevant Queries

3.3.1 MongoDB

Recipes of the user X - Campilongo

```
db.recipe.find({"title":"Pasta"});
```

Search for recipe title X

Given a portion of the title, this query is capable of returning as a result a set of recipes whose titles contain it. Research must be case insensitive (see *options:"i"* in the regexMatch step of the aggregation)

- Input: portion of the title, how many recipes to skip and how many recipes to get.
- Output: set of recipes.

```
db.recipes.aggregate (
  [
    {
      $match :
      {
        title:
        {
          $regex: /^.*<portionOfTheTitle>.*$/,
          $options: "i"
        }
      }
    },
    {
      $sort :
      {
        creationTime : -1
      }
    },
    {
      $skip: <howManyToSkip>
    },
```

```
    {
      $limit: <howManyToGet>
    }
  ]
)
```

Search for recipe category X - Iemma

Search for recipe ingredients X - Campilongo

Most common recipe categories

This query allows you to get a ranking of the categories most used by users for their recipes.

- Input: no one.
- Output: list with the categories ordered by the number of use.

```
db.recipes.aggregate (
  [
    {
      $unwind : "$categories"
    },
    {
      $group :
      {
        _id : "$categories",
        numberOfRecipes:
        {
          $sum: 1
         }
      }
    },
    {
      $project:
      {
        'category': '$_id',
        numberOfRecipes: 1,
        _id: 0
      }
    },
    {
      $sort :
      {
        numberOfRecipes : -1
      }
    },
    {
      $skip: <howManyToSkip>
    },
    {
      $limit: <howManyToGet>
    }
  ]
)
```

Last Comments X - Iemma

3.3.2 Neo4J

Tabella con dominio grafo e descrizione

Suggested Recipes

We have three levels of suggestions with different relevance.

- First Level

  Recipe R is a first level suggestion for the user X if R has been added by the user Y where Y is followed by X ($X \rightarrow Y$). Or R has been added by Z where Z is followed by Y ($X \rightarrow Y \rightarrow Z$).

- Second Level

  Recipe R is a second level suggestion for the user X if R has been added by the user Y where Y is the owner of at least N recipes liked by X.

- Third Level

  Recipe R is a third level suggestion for the user X if R has been commented by the user Y where Y is followed by X ($X \rightarrow Y$).

Recipes of following users X - Campilongo

```
MATCH (n:User)
RETURN n;
```

Most followed and active users

Top Commentators

Most liked users

Best Recipes

Search for username

Given a portion of the username, this query is able to return all users whose usernames contain it. The search must be case insensitive, so the toLower function is used.

- Input: portion of the username, how many users to skip and how many users to get.
- Output: set of users.

```
MATCH (u:User)
WHERE toLower(u.username) CONTAINS toLower($username)
OPTIONAL MATCH (u)<-[f1:FOLLOWS]-(:User)
OPTIONAL MATCH (u)-[f2:FOLLOWS]->(:User)
OPTIONAL MATCH (u)-[a:ADDS]->(:Recipe)
RETURN u.firstName AS firstName, u.lastName AS lastName,
u.picture AS picture, u.username AS username,
u.password AS password, u.role AS role,
COUNT(DISTINCT f1) AS follower,
COUNT (DISTINCT f2) AS following,
COUNT (DISTINCT a) AS numRecipes
SKIP $skip LIMIT $limit
```

Search for user's fullname X - Iemma

3.4 Unit Test

3.5 Tests and Statistical Analysis

Chapter 4

User Manual