Master 1 Data sciences

Personal project:

Freezer manager

Loïc Lejoly s141123

Academic year 2017-2018

June 2018

Contents

1 Introduction

In the context of the personal project course given at ULiège. I decided to develop a smart freezer. This idea emerges due do a personal experience. Every year my family and I reorganize the home's freezers. I was a bit shocked when I saw the quantity of products that were in the freezers and for which I was not aware. This is a problem that anyone can potentially face one day.

The main problem of forgotten products is they cannot stay indefinitely to a freezer due to they physical properties. For instance after some time a meat staying for a long time into a freezer will not have a good taste. As results, some products were thrown to the rubbish. We could avoid to throw to the rubbish products if we are aware of the existence of that products. The solution of that problem would be an application or a tool that gives the possibility to manage the content of your freezers and more. It is the direction that I took for my personal project.

1.1 Explanation of the project itself

The project consists of an application that will give you some information about the products stored into the user's freezers. The period since a product is stored into a specific freezer. Each product will be linked to a unique user due to the fact that most of the products stored in a freezer are *home-made products*. This means that a same product cannot be shared with two different people.

The application gives the possibility to directly find what you want with a specific nomenclature. Each product has a serial number by box and by freezer (if the user has several freezers). Each time a product comes in/out of the freezer the user needs to notify this manipulation to the application.

Moreover, the application has the possibility to do suggestions to the user according to his previous consumption and depending on what his / her freezer(s) contain for some time. The goal being that the application helps the user in his choices and also remind him what he has in his freezer and what should perhaps be consumed in first.

For this project the application is restricted to a web application but can be easily ported to others terminals due to the fact that the back-end part is based on a REST API which is cross-platform.

Front-end part and back-end part a aborder

2 Content storing

2.1 Database model discussion

To store the different information about a user and the data related to his freezers a database is required. There exists several types of databases and it is no so easy to make a choice that will not have bad impacts on the future if the choice made at the beginning was not good. The choice made is a relational database. this choice is motivated by several things. First of all, the problem is known and I know the links to establish between data. Moreover, data would not change on the long-term which means that the architecture of the database would not change a lot and the quantity of information to store will never be huge. Finally, data to store is essentially texts and numbers, thus no need to have a database that can manage video streams etc like noSQL database.

The relational database management system chosen is MySQL since it is an open-source software which is commonly use.

2.2 Relational database structure

The information that the database need to store will be explained in detailed.

First of all, we need to store some information about a user such as an email and a password that allow the user to establish a connection with the main website. This website gives additional information about the user such as the token linked to the user account. This token is mandatory to have access to the API used to manage his freezers. A user has also two additional information which are the language and learning fields. The language field is used to specify the language used by the user. It can be interesting for third applications to have some knowledge about the user. The learning is used to save some parameters that could be useful for learning algorithm. For this project I do no use it because I did not have enough time to find and test some of

these.

The database contains several types of products. These types of products can only be added by the administrator of the system. A product type is described by an identification a name given in English and a name given in French.

The users's freezers should be described by the user. A freezer is descried with three elements . The first one is a unique identifier to be identified into the database. This parameter is automatically done by the system itself, the user does not need to manage this parameter. The two other parameters are the number of boxes that the freezer contains and the second one is the name given by the user to the freezer.

A freezer product from a user contains several fields. First of all, a unique identifier given by the system itself. A date of input, this date must be anterior to the date of its entrance in the freezer. In other words it is impossible to encode product with a date that refers to a place in the future. A date of output, by default this field is set to null. when this field is not null it means that the product is not any more in the freezer. The date of output must be greater or equal to the date of input. A product also contains a period field which is used to define the period in months for which the product should be in the freezer. Then third applications can display products that overcome this period. The quantity is also a piece of information of a product. It gives the quantity in terms of person for which the product can be used. Finally, a product also have two additional fields to allow a user to locate more easily a product. These two additional fields are the box number which give the box for a specific freezer and the product identifier which allows to locate the product in this box. This identifier is unique for the specific box.

A product also receives a description . The product description contains three different fields which are the description identifier, the product name and a free description of the product. This description can be a structured description or a simple text.

2.3 Entity-relationship model

The Figure 1 shows the entity-relationship model of the freezer database. As we can see on this schema the relations *list_freezer* and *product_to_type* are not mandatory since the information that they linked can be retrieved from the rest of the database. These two relations have been inserted into the database for a simplicity since these information are often asked. Knowing that it is interesting for the backup part.

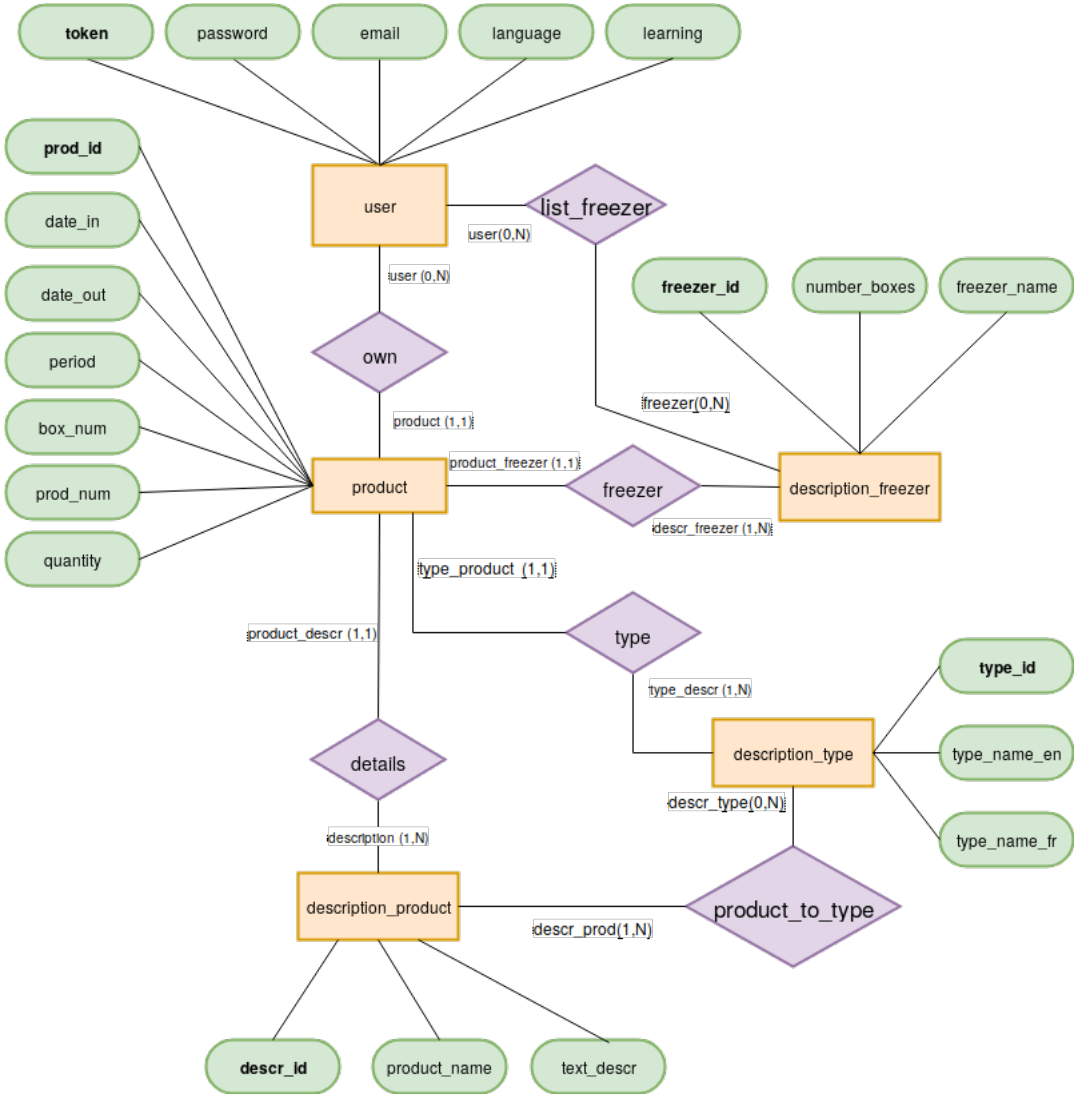

Figure 1: entity-relationship model of the database

2.4 Relational model

From the entity-relationship we can go into the relational format which allow to have a representation for integrating the database into a relational database management system.

1. User(**token**, password, email, language, learning)
2. Description_type(**type_id**, type_name_en , type_name_fr)
3. Description_product(**descr_id**, product_name, text_descr)
4. Description_freezer(**freezer_id**, number_boxes, freezer_name)
5. Product_to_type(**#descr_id, #type_id**)
6. List_freezers(**#freezer_id, #token**)
7. Product(**prod_id**, #token, #descr_id, #freezer_id, #type_id, date_in, date_out, period, box_num, prod_num, quantity)

The database is encoded in MySQL language. The engine used is InnoDB with utf8mb4_unicode_ci as collation. The choice of this engine has been done because is the default engine and manage ACID transactions and foreign keys. The Figure 2 depicts the MySQL implemented and the types associated to each variable. The choices made for the length size of variables are personal choices. But the length size can be changed with no difficulties if the need arose. The encoding chosen allows to be flexible as much as possible.

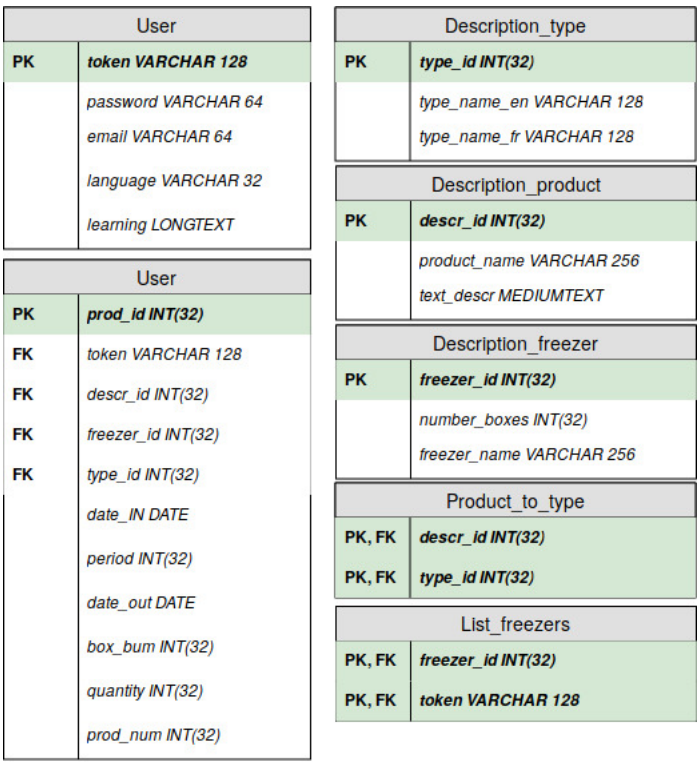

Figure 2: slq tables of the database

As discussed above the architecture of the database allows to skip some tables to backup like Product_to_type and List_freezers. These two table can be reconstruct with the others tables. Thus it is not mandatory to backups these ones.

2.5 Account creation

To use the API a user need to create an account with an e-mail address and a password. A webpage (Figure 3) is built to achieve this goal(*website/register.php*). To have a the password encrypted using the function *crypt()* with a salt. By default crypt uses the DES algorithm and it is the algorithm chosen for this project. Note that the DES algorithm is not secure but sufficient to hide a password from the admin.
If the objective is to put in production the project. It is recommended to change the password encryption to use another type of encryption like SHA 512 or another secure encryption algorithm.

When the account has been created it is linked to a token. This token is generated on the server side with the PHP function *random_bytes* which is recommended to generate string of cryptographic random bytes that are suitable for cryptographic use. The length specified for the generation is 32 bytes.Thus the number of possibilities is $(2^8)^{32} = 2^{256}$. After that, the random sequence generated is transformed into a hexadecimal string to be more readable by a novice user.

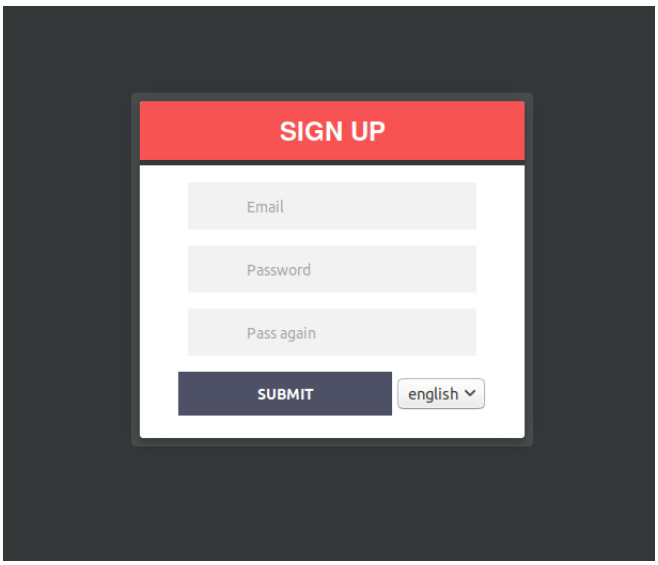

Figure 3: register page

When the account is created a user can use the API to build its own software and manage his freezers with the help of The API and the token linked to its account or use an existing application that uses the API. In this second case the third part application should only asked your token and anything else.

pourquoi choix du relationel db mettre les schema entite relation expliquer les choix etc
mentionner que l'encodage a été fait à la main et essentiellement pour un utilisateur -> fastidieux à faire et insérer des éléments aléatoirement n'a pas vraiment de sens du fait qu'un utilisateur n'a pas ce comportement. Ce la ne poserait pas de probleme pour la db mais c'est un travail inutile a réaliser dans l'optique du mon projet que de remplir la db avec des données aléatoire.

3 User panel

When a user possesses an account. He has the possibility to access to a user panel with the *website/login.php* page (Figure 5). If the information given are correct, you will be redirected to the user panel panel page *website/user/user_interface.php*.

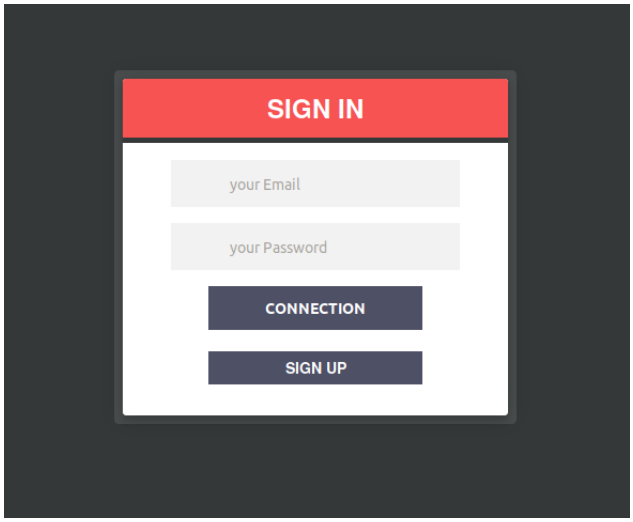

Figure 4: Login page

This user panel is used to give to the user more information about its account. i.e: the email address, the token linked to the account, the possibility to change the password and the language. For the scope of the project the user panel is not developed more than that but for production project it should be the case. We could add some features to manage our freezers, etc.

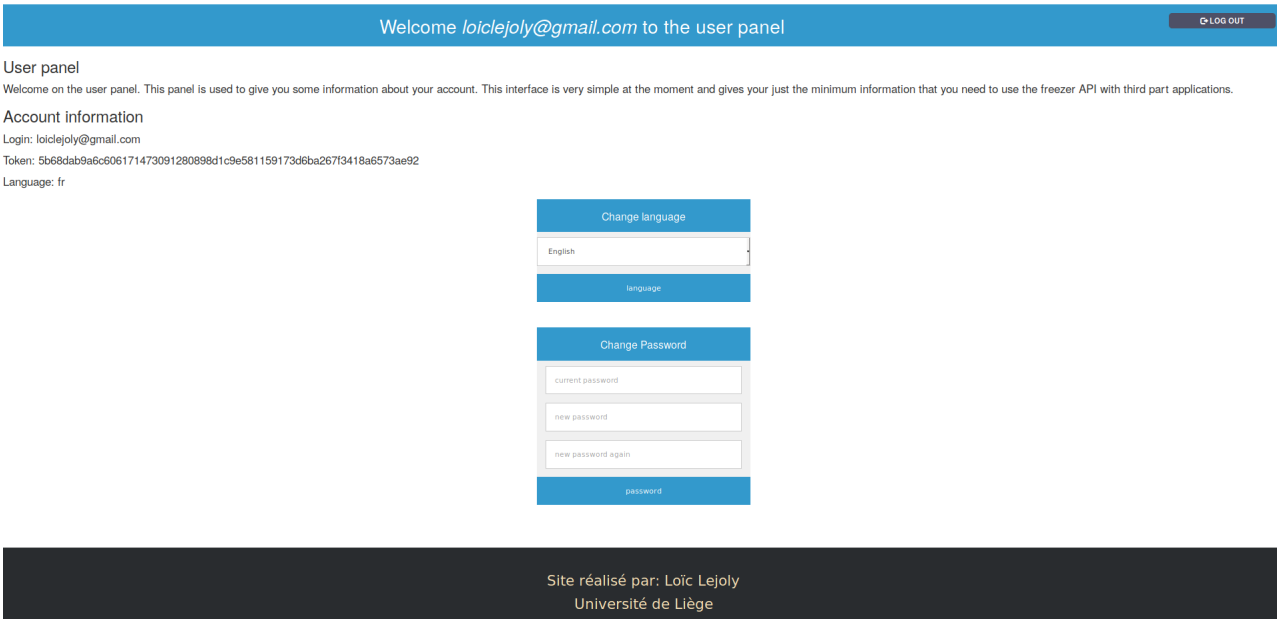

Figure 5: User panel

4 API REST

4.1 Sever

4.2 functionalities implemented

Several functionalities have been implemented.

4.2.1 check_token

This request implements a single method which is *GET* method. It gives the possibility to check the validity of a given token. If this one is correct a response with the status code 200 is sent. Otherwise, a response with the status code 400 is returned. Moreover, when the status code 400 is returned a JSON object is also sent back with two fields *details and status*. Where details field explains the reason of this status code. And status field which give the status a second time to help third applications to obtain that easily.

The prototype of this request is the following:

$$< domainUrl > /check\_token/ < token >$$

Parameters:

1. <domailUrl>: The url of the server API. In the case of the project the server is run on the *localhost:5000*
2. <token>: A user's token that corresponds to an existing account.

4.2.2 types

This request implements a single method which is *GET* method. It gives the possible types of products that can be encoded into the database. To obtain these types you need to give an existing token. If this token is correct a JSON object containing the different types of products is sent and the status code of the response is 200. Otherwise the status code 400 is returned with the two fields *details and status* described earlier.

The prototype of this request is the following:

$$< domainUrl > /types/ < token >$$

Parameters:

1. <domailUrl>: The url of the server API. In the case of the project the server is run on the *localhost:5000*
2. <token>: A user's token that corresponds to an existing account.

The architecture of the JSON file returned is the following:

```
[
    {
        "type_id": 1,
        "type_name_en": "soup",
        "type_name_fr": "soupe"
    },
    {
        "type_id": 2,
        "type_name_en": "meal soup",
        "type_name_fr": "soupe repas"
    },
    ...
]
```

Explanation of the JSON object:

1. *type_id*: An 'integer' that represents the id of the product's type
2. *type_name_en*: The type name in English
3. *type_name_fr*: The type name in French

Note that it is not possible for a user to add a product type. It is the desired behaviour since we do not want a user with the possibility to add random product types or define several times the same product type. This functionality should only given to the admin of the system.

4.3 Documentation

To understand how to use the REST API a website with a documentation has been made. The tool used to built this documentation is *MkDocs*. This tool is commonly used to build documentation and are used for instance by *Keras*. Markdown is used to write the documentation and the tools translates it into static html page. It is really easy to use if you know the Markdown language.

This documentation explain what are the available requests that can be done, how to achieve a correct request and what are the results sent back. For each request an example using the *client URL request library (cURL)* is given. The choice of this tool has been made because it is cross-platform and not difficult to use. But it can be used with any other tool that can achieve http requests.

explication des differentes fonction mentioner l'instoration d'une documentation détaillée et de l'utilisation mentioner cORS pour prendre en compte les different domaines Pas de problemes pour ajouter de nouvelles fonctionnalitées ou de changer les existantes

Pourquoi une API REST

Mentioner l'essai du preférence learning ce qui a été fait et pas fait les probleme rencontrés etc

4.4 formules

5 Admin panel

Pas vraiment développé Panel qui donne acces à toute une série d'informations globales sur les utilisateurs. Pas bcp de fonctionalités pour le moment mais peut etre étendu sans problème

6 External website

Ce veut être une application "externe" a l'API REST et à la DB. Peut etre vu comme une société externe utilisant un service

Le but de cettte application est d'avoir une sorte de dashboard

7 unit tests

8 problemes rencontrés

l'architecture n'est pas la plus adéquate pour faire les unit test. sur le site flask une autre approche d'architecutre est proposée

9 future work

10 Conclusion