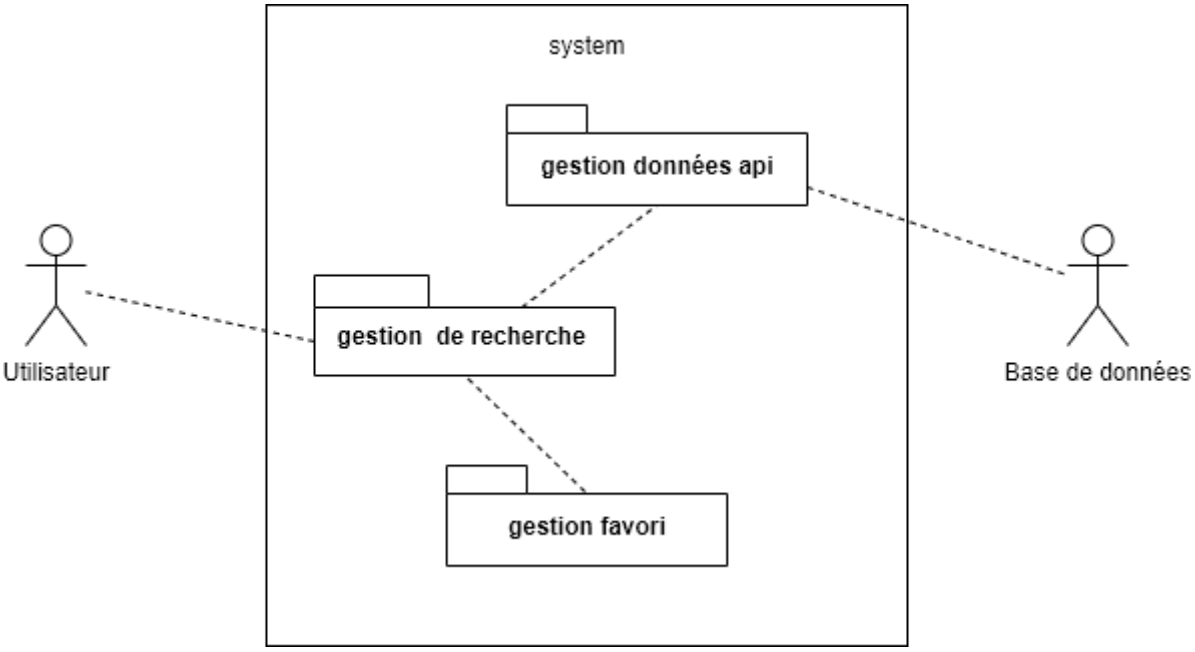
P5 – Utilisez les données publiques de l’openfoodfacts

Morgan Facorat

Démarche choisie

Dans un premier temps j'ai cherché à comprendre le fonctionnement de l' API OpenFoodFacts qui est une api au format JSON. OpenFoodFacts a été la seule difficulté liée à ce projet. En effet chaque api à son mode de fonctionnement. Je me la suis donc approprie en lisant la documentation.Pour mieux organiser mon projet j'ai cree un tableau trello disponible
ici :https://trello.com/invite/b/HnBxIpuu/47a9b83f96a805669b06b567900ef807/projet-p5-application-pur-beur

Je me suis permis par la suite de creer un diagramme de packages.



Grace à celui-ci nous avons defini trois packages bien distincts les uns des autres. Qui m'ont permis de regrouper les fonctionnalités majeures de notre application en plusieurs groupes.

Apres avoir défini les differentes carracteristiques de l'application à l'aide du cahier des charges et du diagramme de packages . j'ai cherche les différents modules qui seraient utiles au developpement de celle-ci. Je me suis donc retrouve avec requests , json et mysql.connector.

le module requests , permet d'utilser le protocole http ce qui m'a permis d'interagir directement avec l'api.

Par la suite le module json m'a permis de manipuler les données aux format json , récupere à l'aide de request.

Et pour finir le module mysql.connector m'a permis d'effectuer les requettes sql donc d'intéragir directement avec ma base de donnée , crée une data base , et l'allimenter à l'aide des données de l'api.

J'ai donc transcrit mon idée directement en ligne de code disponible sur github :
https://github.com/fakko77/OpenFoodFacts

Mon applicaiton est diviser en 4 fichiers distincts

Pur beurre.py permet d'exécuter celle-ci, classes py contient la classe qui permet de se connecter à la base de données et effectuer les requêtes, fonction.py contient toutes les fonctions qui permettent au bon fonctionnement de l'application, récupération, mise en forme des données collecter. Le fichier constant.py quant à lui les informations de connexion à la base de

données Ip , user et quelques requêtes Sql.

```
def create_data():
    data_create = data_connect(ip, user, password, "")
    data_create.req("DROP DATABASE IF EXISTS pur_beurre")
    data_create.req("CREATE DATABASE IF NOT EXISTS pur_beurre")
    data_create.close


data = data_connect(ip, user, password, db)
close = data.close


def create_table():
    data.req(create_category)
    data.req(create_produit)
    data.req(create_favori)
    close


def ajout_category():
    maxi = len(dic)
    compteur = 0
    compteur_category = 0
    compteur_produit = 0
    while compteur < maxi:
        requette = "INSERT INTO category " \
                   "(nom) values ('" + str(dic[compteur]) + "')"
        data.req(requette)
        compteur += 1
    while compteur_category < 5:
        while compteur_produit < 100:
            print(compteur_category, str(dic[compteur_category]))
            url = "https://fr.openfoodfacts.org/cgi/search.pl?action=process" \
                  "&tagtype_0=categories&tag_contains_" \
                  "0=contains&tag_0=" + str(dic[compteur_category]) + "&json=" \
                                                                      "true&page_size=100"
            payload = {}
            headers = {}
            response = requests.request("GET", url,
```

Comme l’on peut le voir sur la capture d’écran ci-dessus, j’ai créé plusieurs fonctions qui ont chacun un but précis à l’application. En effet j’ai trouvé avantageux de coder de cette façon d’une part cela éclaircie le code et d'une autre pour le maintien et l’évolution de celui-ci car l’on sait directement ou interagir pour effectuer une modification. Sur la base de données par exemple.

```
import mysql.connector
import json
from constantes import *


class data_connect:
    def __init__(self, ip, user, password, db):
        self.ip = ip
        self.user = user
        self.password = password
        self.db = db

        conn = mysql.connector.connect(
            host=ip,
            port=3306,
            user=user,
            password=password,
            database=db)
        cur = conn.cursor(buffered=True)
        self.con = conn
        self.cur = cur


    def req(self, req):
        cur = self.cur
        cur.execute(req)

    def req_return(self):
        cur = self.cur
        return cur.fetchall()
    def close_conn(self):
        conn = self.con
        return conn.close()
```

La seule difficulté rencontrée au niveau du code à été de créer la classe permettant la connexion !

le problème à été résolu une fois après avoir compris comment incorporer le module mysql connector dans la class data connect.