Projet 5 : Utilisez les données publiques de l'OpenFoodFacts

...

Parcours développeur d'application - Python
Openclassrooms

Contexte

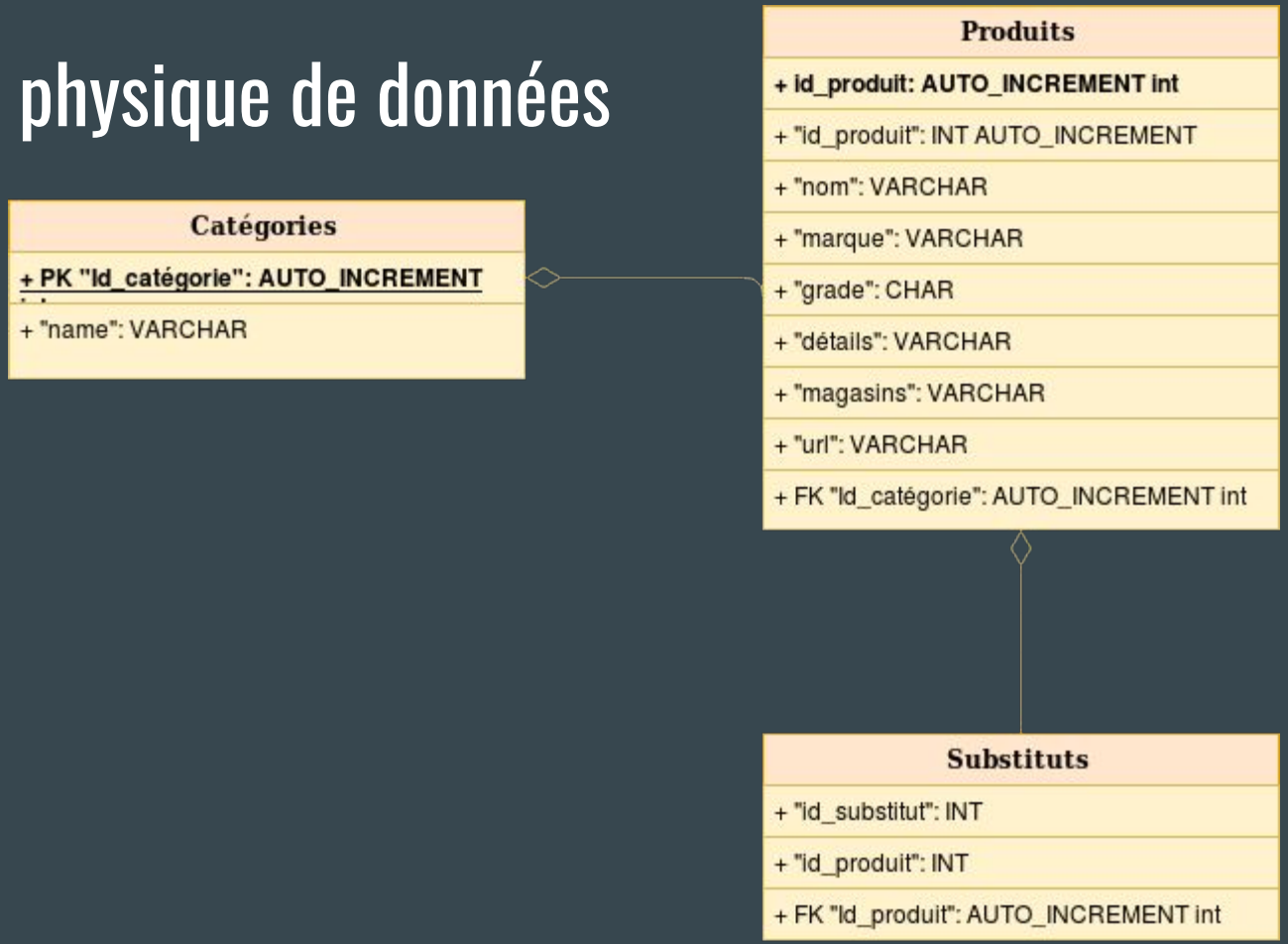
La startup Pur Beurre travaille connait bien les habitudes alimentaires françaises. Leur restaurant, Ratatouille, remporte un succès croissant et attire toujours plus de visiteurs sur la butte de Montmartre.

La startup à remarqué que ses clients seraient prêt à accepter de changer leur alimentation pour manger plus sainement mais ne savent pas comment remplacer certains aliments.

L'idée est donc de créer un programme interagissant avec la base Open Food Facts pour en récupérer les aliments, les comparer et proposer un substitut plus sain.

Démonstration du programme

Modèle physique de données



Connexion à la base de données Mysql

-Utilisation de la librairie PyMysql

-Méthode permettant de retourner un tuple que l'on récupérera pour nos autres méthodes de création de tables et d'insertion de données

```
def db_connection():

    """
    Connect to the database
    """

    connexion = pymysql.connect(host='localhost',
                                user='root',
                                password='jojo',
                                db='Projet_5_Openfoodfacts',
                                charset='utf8')
    curs = connexion.cursor()
    return curs, connexion


def create_db():
    curs, con_db = db_connection()

    curs.execute(
        "CREATE DATABASE Projet_5_Openfoodfacts CHARACTER SET 'utf8';"
    )
    print("Création de la base de données - Projet_5_Openfoodfacts -")
    con_db.commit()
```

La récupération des données dans l'API Open Food Facts

-Boucle sur le nombre de page à parcourir

-Librairie "Requests"

-Données converties en .json sur le site

-Boucle sur les données produits pour filtre des données

-Boucle sur sur le dictionnaire "prod_dict" pour insertion dans la base de données et insertion des données dans la base

```
def openfoodfacts_produits(cat_id):
    """read open food facts api and select id name brand grade detail
    from a selected categorie"""
    nb_page = 1
    while nb_page <= NB_PROD_REQUEST/NB_PROD_PAGE:
        url = requests.get("https://fr.openfoodfacts.org/langue/francais/categorie/"+str(cat_id)+"/"+str(nb_page)+".json")

        data_raw = url.json()
        data_produits = data_raw["products"]
        prod_dict = {}
        nb_page = nb_page +1
        for produits in data_produits:

            try:
                name_prod = (produits["product_name"])
                id_prod = (produits["_id"])
                brand_prod = (produits["brands"])
                grade_prod = (produits["nutrition_grades"])
                detail_prod = (produits["generic_name_fr"])
                stores_prod = (produits["stores"])
                url_prod = "https://fr.openfoodfacts.org/produit/"+str(produits["_id"])
                cat_prod = cat_id[0][0]

                prod_dict[id_prod] = name_prod, brand_prod, grade_prod, detail_prod, stores_prod, url_prod, cat_prod

            except Exception as e:
                pass

        for prod_id, prod in prod_dict.items():
            name, brand, grade, detail, stores, url, cat = prod
            try:
                insert_products_db(prod_id, name, brand, grade, detail, stores, url, cat)
            except Exception as e:
                print(e)
```

L'insertion des données dans la base de données Mysql

-Tuple pour les données en paramètre

-ON DUPLICATE KEY UPDATE sur le nom pour ne pas bloquer à l'insertion

```
def insert_products_db(id_prod, name_prod, brand_prod, grade_prod, detail_prod, stores_prod, url_prod, cat_prod):
    """insert the product values in database"""
    product_var = id_prod, name_prod, brand_prod, grade_prod, detail_prod, stores_prod, url_prod, cat_prod, name_prod
    curs, con_db = db_connection() #tuples
    try:
        curs.execute(
            "INSERT INTO Produits VALUES (NULL, %s, %s, %s, %s, %s, %s, %s, %s) ON DUPLICATE KEY UPDATE nom=%s",product_var
        )
    except Warning:
        pass
    con_db.commit()
```

Suppression des tables

- Appel de ces 3 méthodes pour la suppression des tables

```
def del_substituts_table():
    """delete substituts table"""
    curs, con_db = db_connection() #tuples
    curs.execute(
        "DROP TABLE Substituts;"
    )
    con_db.commit()

def del_products_table():
    """delete products table"""
    curs, con_db = db_connection() #tuples
    curs.execute(
        "DROP TABLE Produits;"
    )
    con_db.commit()

def del_categories_table():
    """delete categories table"""
    curs, con_db = db_connection() #tuples
    curs.execute(
        "DROP TABLE Categories;"
    )
    con_db.commit()
```

Affichage des données sauvegardées par l'utilisateur

-Jointure des tables Produits et Substituts

-Boucle pour l'affichage des données de la requête

```
def show_substituts_db():
    """display the product and the substituts informations from the table substituts"""
    curs, con_db = db_connection()

    curs.execute(
    """SELECT DISTINCT p1.id_produit, p1.ean_produit, p1.nom, p1.marque, p1.grade, p1.details, p1.magasins, p1.url,
    p2.id_produit, p2.ean_produit, p2.nom, p2.marque, p2.grade, p2.details, p2.magasins, p2.url
    FROM Substituts s LEFT JOIN Produits p1 ON p1.id_produit = s.id_produit LEFT JOIN Produits p2 ON p2.id_produit = s.id_substitut"""
    )

    data = curs.fetchall()
    con_db.commit()

    for rows in data:
        print("-"*50)
        print("-"*50)
        print("PRODUIT SUBSTITUÉ :")
        print("\n-Nom : "+str(rows[2])+"\n-Marque : "+str(rows[3])+"\n-Nutriscore : "+str(rows[4])+"\n-Description : "+str(rows[5])+"\n-Magasins : "+str(rows[6])+"\n-Lien : "+str(rows[7]))
        print("-"*50)
        print("SUBSTITUT DU PRODUIT :")
        print("\n-Nom : "+str(rows[10])+"\n-Marque : "+str(rows[11])+"\n-Nutriscore : "+str(rows[12])+"\n-Description : "+str(rows[13])+"\n-Magasins : "+str(rows[14])+"\n-Lien : "+str(rows[15]))
        print("-"*50)
        print("-"*50)
```

Réponse utilisateur et réaction du programme

-Affichage des catégories avec la méthode show_categories_db()

-Réponse utilisateur avec la fonction category_choice() avec une boucle pour filtrer les réponses non désirées

```
while first_screen:

    del_screen = False
    selection_screen = False
    display_choice()
    main_answer = select_choice()
    print(main_answer)
    first_screen = False
    if main_answer == "1":
        selection_screen = True

        while selection_screen:
            data_cat = show_categories_db()
            answer = category_choice(data_cat)
            cat_select = select_categories_db(answer)

            openfoodfacts_produits(cat_select)

            selection_screen = False
            product_screen = True
```

```
def category_choice(data):
    """ input for the category selection """
    print("-"*50)
    print("CHOIX DE LA CATEGORIE")
    print("-"*50)

    user_answer = input("Quelle catégorie choisissez-vous?")

    liste_ids = [str(elem[0]) for elem in data]#convert element in string

    while user_answer not in liste_ids:
        print("id inexistant - Veuillez reessayer")
        user_answer = input("Quelle catégorie choisissez-vous?")

    return user_answer
```

```
def show_categories_db():
    """display the categories"""
    curs, con_db = db_connection()

    curs.execute(
        "SELECT id_categorie, nom FROM Categories ORDER BY id_categorie ASC"
    )
    data = curs.fetchall()
    con_db.commit()

    print("affichage des catégories -")
    print("-"*50)
    for line in data:
        print(str(line[0])+"-"+line[1])

    return data
```

Affichage des produits substituts

-Requête dans la base pour la sélection des données en fonction du nutri-score et de la catégorie du produit choisi par l'utilisateur

-Affichage d'un numéro de substitut pour chaque produit substitut

```
def show_substituts(grade, categorie):
    """display the product and the substituts informations for the selected product"""
    selection = grade, categorie

    curs, con_db = db_connection()

    curs.execute(
        "SELECT id_produit, ean_produit, nom, marque, grade, details, magasins, url FROM Produits WHERE grade<=%s AND categorie
    )
    data = curs.fetchall()
    num_substitut = 1
    dict_sub = {}
    code_sub = []
    for line in data:

        code_sub = str(num_substitut)
        id_sub = str(line[0])
        dict_sub[code_sub] = id_sub
        print("substitut n°"+str(num_substitut)+" code "+str(line[0]))
        print("-"*50)
        print("\n-Nom : "+str(line[2])+"\n-Marque : "+str(line[3])+"\n-Nutriscore : "+str(line[4])+"\n-Description : "+str(line
        print("-"*50)
        #print(dict_sub)
        num_substitut += 1
    con_db.commit()

    return dict_sub
```

```
def show_user_selection(selection):
    """show product details from the selected product"""
    curs, con_db = db_connection()

    curs.execute(
        "SELECT id_produit, ean_produit, nom, marque, grade, categorie, details, magasins, url
    )
    data = curs.fetchall()

    print("affichage détails produits -")
    for line in data:
        grade = str(line[4])
        categorie = str(line[5])
        print("-"*50)
        print("\n-Nom : "+str(line[2])+"\n-Marque : "+str(line[3])+"\n-Nutriscore : "+str(line[
        print("-"*50)
        selection_substitut = show_substituts(grade, categorie)

    con_db.commit()

    return selection_substitut
```

Amélioration du programme

Plusieurs choses améliorables concernant ce programme:

- L'affichage des produits non pas par leur identifiant dans la base de données mais plutôt de manière incrémentale.
- Si le produit demandé par l'utilisateur possède un substitut, lui affiché ce substitut déjà enregistré
- Avoir une approche plus orientée objet notamment au niveau des affichages de texte et des input
- Ajouter une interface graphique