P5

Utiliser les données publiques d’Open Food Facts

Les objectifs

1. Créer une base de données et ses tables à l'aide de MySQL

2. Insérer des données récupérées sur le site OFF

3. Créer une interface client/BDD en ligne de commande

4. Obtenir un produit final utilisable par un client

Les librairies

- **Requests** : récupération des données

- **MySQL.connector** : lien entre MySQL et Python, gestion de la BDD

- **Virtualenv** : environnement virtuel pour faciliter la portabilité du programme

- **Colorama** : optimisation de l’ergonomie du programme, meilleure expérience utilisateur

Modèle de la base de données

Base de données
"OPENFOODFACTS"

category	
PK	**id : SMALLINT unsigned NOT NULL AUTO_INCREMENT**
	id : SMALLINT unsigned NOT NULL AUTO_INCREMENT produit : VARCHAR(40) NOT NULL

product	
PK	**id : SMALLINT unsigned NOT NULL AUTO_INCREMENT**
FK	**produit_id : SMALLINT unsigned NOT NULL**
	id : SMALLINT unsigned NOT NULL AUTO_INCREMENT produit_id : SMALLINT unsigned NOT NULL sub : TEXT NOT NULL nom : TEXT NOT NULL marque : TEXT NOT NULL nutriscore : TEXT NOT NULL url : TEXT NOT NULL

Démo et analyse des scripts

Analyse des scripts

user_param.py / informations de connexion à la base de données

```
5
6       USER_NAME = 'root' # your username
7       USER_PASSWORD = '' # your password
8       USER_HOST = 'localhost' # your host
9       DB_NAME = 'OPENFOODFACTS'
10      PASSWORD_TYPE = 'mysql_native_password'
```

Analyse des scripts

create_db.py / création de la base de données

```
4     import mysql.connector
5     from mysql.connector import errorcode
6     from const_msg import *
7     from user_param import *
8
9     CNX = mysql.connector.connect(user=USER_NAME,
10                                  password=USER_PASSWORD,
11                                  host=USER_HOST,
12                                  auth_plugin=PASSWORD_TYPE)
13
14    try:
15        CURSOR = CNX.cursor()
16        create_db = "CREATE DATABASE IF NOT EXISTS `OPENFOODFACTS` CHARACTER SET 'utf8'"
17        CURSOR.execute(create_db)
18
19        param_user = "GRANT ALL PRIVILEGES ON OPENFOODFACTS.* TO '"+USER_NAME+"'@'"+USER_HOST+"' "
20        CURSOR.execute(param_user)
21        CNX.commit()
22
23    except:
24        print(PROBLEM_DB)
25    finally:
26        CNX.close()
```

Analyse des scripts

sql.py / création des tables

```
16      try:
17          CURSOR = CNX.cursor()
18          CREATE_TABLE = "CREATE TABLE IF NOT EXISTS Category (" \
19                         "id SMALLINT unsigned NOT NULL AUTO_INCREMENT," \
20                         "Produit VARCHAR(40) NOT NULL," \
21                         " PRIMARY KEY (id)" \
22                         ") ENGINE=InnoDB"
23          CURSOR.execute(CREATE_TABLE)
24          CNX.commit()
25
26          CREATE_TABLE_1 = "CREATE TABLE IF NOT EXISTS `Product` (" \
27                           "id SMALLINT unsigned NOT NULL AUTO_INCREMENT," \
28                           "produit_id SMALLINT unsigned NOT NULL," \
29                           "sub TEXT NOT NULL," \
30                           "nom TEXT NOT NULL," \
31                           "marque TEXT NOT NULL," \
32                           "shop TEXT NOT NULL," \
33                           "nutriscore TEXT  NOT NULL," \
34                           "url TEXT NOT NULL," \
35                           "PRIMARY KEY (id)," \
36                           "KEY fk_produit_id (produit_id)," \
37                           "CONSTRAINT fk_produit_id FOREIGN KEY (produit_id) REFERENCES Category(id)" \
38                           ") ENGINE=InnoDB"
39          CURSOR.execute(CREATE_TABLE_1)
40          CNX.commit()
41
42
43      except:
44          print(PROBLEM_DB_TABLES)
45      finally:
46          CNX.close()
47
```

Analyse des scripts

constantes.py
const_msg.py
les constantes du programme

```python
CAT_DICT = {1: 'https://fr.openfoodfacts.org/categorie/jus-d-orange/',
            2: 'https://fr.openfoodfacts.org/categorie/pates-a-tartiner-au-chocolat/',
            3: 'https://fr.openfoodfacts.org/categorie/biscottes/'}

NUTRI_DICT = {1: 'a', 2: 'b', 3: 'c', 4: 'd', 5: 'e'}

for key, value in CAT_DICT.items():
    if key:
        R = requests.get(value + '1.json')
        TABLE_JSON_PAGE = R.json()
        PAGE_SIZE = TABLE_JSON_PAGE[u'page_size']  # return number of products by page
        COUNT = TABLE_JSON_PAGE[u'count']  # return number of product by category
        TOTAL_PAGE_NUMBER = int(COUNT / PAGE_SIZE) + 1  # return number of pages by category
```

```python
t en cours..."


- " \
                                                    "\n1 - Quel aliment souhaitez-vous remplacer ? " \
                                                    "\n2 - Retrouver mes aliments substitués " \
                                                    "\n3 - Quitter le programme "

ENTER_NUMBER = Fore.BLACK + Back.CYAN + Style.BRIGHT + "\nEntrez le chiffre correspondant à votre souhait : "
ENTER_CAT_NUMBER = Fore.BLACK + Back.CYAN + Style.BRIGHT + "\nEntrez le chiffre correspondant à la " \
                                                           "catégorie que vous désirez afficher : "

ENTER_IDSUB_NUMBER = Fore.WHITE + Back.BLACK + Style.BRIGHT + "\nTapez l'ID attribué au " \
                                                              "produit pour afficher une proposition" \
                                                              " de substitut \nou appuyez sur entrer " \
                                                              "pour quitter : "
SHOW_MORE_SUB = "\nVoulez-vous afficher d'autres substituts ? (tapez 'entrer' sinon tapez 1) : "

# messages for queries.py
SELECT_CAT = Fore.BLACK + Back.WHITE + Style.BRIGHT + '\n- Sélectionnez la catégorie -'
SUB_RESULT = Fore.BLACK + Back.CYAN + Style.BRIGHT + "\nNous vous proposons le substitut suivant:"
SAVE_SUB = Fore.BLACK + Back.CYAN + Style.BRIGHT +'\nVoulez-vous sauvegarder ce substitut ? ' \
                                                  '(tapez 1, sinon tapez "entrer") : '
SUB_SAVED_CONFIRMATION = Fore.BLACK + Back.CYAN + Style.BRIGHT + "Produit sauvegardé !"

SUB_SAVED_MENU = Fore.BLACK + Back.WHITE + Style.BRIGHT + '\n- Mes produits sauvegardés -'

ENTER_IDSUB_NUMBER_WANTED = "\nTapez le numéro du SUBSTITUT que vous désirez " \
                            "afficher ou tapez 'entrer' pour quitter : "

# goodbye message
GOODBYE_MSG = Fore.BLACK + Back.WHITE + Style.BRIGHT + "\nMerci d'avoir utilisé notre programme. À bientôt."
```

Analyse des scripts

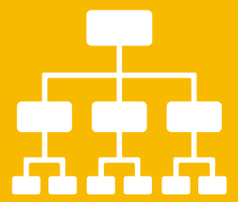
feed_in.py / la classe Product

```
22      class Product:
23          """Class Product : get products from OFF
24
25      def __init__(self):...
27
28      def get_products_from_off(self):...
60
61      def send_products_to_db(self):...
75
```

Analyse des scripts

feed_in.py / la classe Product / méthode de classe init

```
22      class Product:
23          """Class Product : get products from O
24
25          def __init__(self):  # constructor
26              self.products_data = []
27
28          def get_products_from_off(self):  # me
29              try:...
56              except:...
60
61          def send_products_to_db(self):...
```

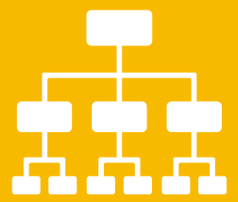


Analyse des scripts

```
def get_products_from_off(self):  # method to get data with requests' module
    try:
        for page in range(0, TOTAL_PAGE_NUMBER):
            for key, value in CAT_DICT.items():
                if key:
                    rpage = requests.get(value + str(page + 1) + '.json')
                    TABLE_JSON_PAGE = rpage.json()
                    products_by_page = TABLE_JSON_PAGE[u'products']

                    for products in products_by_page:...
    except:
        print(PROBLEM_INSERTION)
```

feed_in.py
la classe product
méthode de classe : get_products_from_off

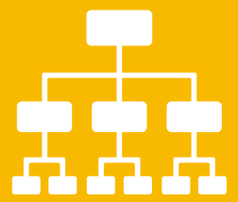
```
for products in products_by_page:  # Fill the list of products

    id_product = key
    sub = 0

    try:
        name = products['product_name']
    except KeyError:
        name = '/'

    try:
        brand = products['brands']
    except KeyError:
        brand = '/'

    try:
        shop = products['stores']
    except KeyError:
        shop = '/'

    try:
        nutri_origin = products['nutrition_grades_tags'][0]
        for key_nutri, value_nutri in NUTRI_DICT.items():
            if nutri_origin == value_nutri:
                nutriscore = key_nutri
    except KeyError:
        nutriscore = '/'

    try:
        url = products['url']
    except KeyError:
        url = '/'

    self.products_data.append([id_product,
                               sub, name,
                               brand, shop,
                               nutriscore, url])
```



Analyse des scripts

feed_in.py / la classe Product / méthode de classe send_products_to_db

```
def send_products_to_db(self):  # method to send products to database
    try:
        data = self.products_data[:]  # all data got from OFF
        CURSOR = CNX.cursor()
        CURSOR.execute("INSERT INTO Category (id, Produit) "
                       "VALUES (NULL, 'Jus d''orange'), "
                       "(NULL, 'Pâte à tartiner au chocolat'),"
                       "(NULL, 'Biscottes')")
        CURSOR.executemany("INSERT INTO Product(id, produit_id, sub, nom, marque, shop, nutriscore, url) "
                           "VALUES (NULL, %s, %s, %s, %s, %s, %s, %s)", data)
        CNX.commit()
    except:
        print(PROBLEM_INSERTION)
```



Analyse des scripts

queries.py / fonctions

```
18      def menu():...
28
29
30      def show_category(cat_id):...
42
43
44      def substitutes(cat_id, user_idproduct_choosen):...
70
71
72      def show_saved_products():...
94
95      def stop_program():...
107
```

Analyse des scripts

mainscript.py / Main loop

```
def main():  # Main function

    print(LOADING_MSG)

    program = True
    new_product = Product()
    new_product.get_products_from_off()
    new_product.send_products_to_db()

    while program:  # Main loop
        print(THE_MAIN_MENU)
        menu_input = input(ENTER_NUMBER)

        if menu_input == '1':  # Show main menu
            menu()
            cat_id = input(ENTER_CAT_NUMBER)
            if cat_id == '1' or '2' or '3':  # Show the chosen category among cat id
                show_category(cat_id)

        if menu_input == '2':  # Show saved products
            saving_products = True
            while saving_products:
                show_saved_products()
                saved_menu = input(SHOW_MORE_SUB)
                if saved_menu == '1':
                    saving_products = False

        if menu_input == '3':  # Quit program and delete all data
            stop_program()
            program = False


if __name__ == '__main__':  # Encapsulation of main function
    main()
```

Analyse des scripts

mainscript.py / Main loop / menu_input 1

```python
def main():  # Main function

    print(LOADING_MSG)

    program = True
    new_product = Product()
    new_product.get_products_from_off()
    new_product.send_products_to_db()

    while program:  # Main loop
        print(THE_MAIN_MENU)
        menu_input = input(ENTER_NUMBER)

        if menu_input == '1':  # Show main menu
            menu()
            cat_id = input(ENTER_CAT_NUMBER)
            if cat_id == '1' or '2' or '3':  # Show the chosen category among cat id
                show_category(cat_id)

        if menu_input == '2':  # Show saved products
            saving_products = True
            while saving_products:
                show_saved_products()
                saved_menu = input(SHOW_MORE_SUB)
                if saved_menu == '1':
                    saving_products = False

        if menu_input == '3':  # Quit program and delete all data
            stop_program()
            program = False


if __name__ == '__main__':  # Encapsulation of main function
    main()
```

queries.py

fonction menu

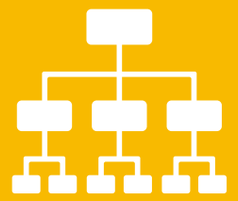
```python
def menu():  # display main menu
    """ THE MAIN MENU"""
    try:
        print(SELECT_CAT)
        cursor = CNX.cursor()
        cursor.execute("SELECT * FROM Category")
        for (id, Produit) in cursor:
            print("{} - {}".format(id, Produit))
    except:
        print(PROBLEM_CAT_MENU)
```

fonction show_category

```python
def show_category(cat_id):  # show a chosen category
    """ SHOW THE CATEGORY CHOSEN BY THE USER"""
    try:
        cursor = CNX.cursor()
        cursor.execute("SELECT id, nom, marque, shop, nutriscore, url "
                       " FROM Product WHERE produit_id = " + str(cat_id) +
                       " AND NOT nutriscore='unknown'")
        for (id, nom, marque, shop, nutriscore, url) in cursor:
            print("ID : {} NOM : {} MARQUE : {} MAGASIN : {} NUTRISCORE : {} LIEN : {}".format(
                id, nom, marque, shop, nutriscore, url))

        user_idproduct_choosen = input(ENTER_IDSUB_NUMBER)
        if user_idproduct_choosen:  # Save a substitute
            substitutes(cat_id, user_idproduct_choosen)
    except:
        menu()
```

fonction substitutes

```python
def substitutes(cat_id, user_idproduct_choosen):  # find and save a healthier substitute
    """ SUBSTITUTES FUNCTION"""
    try:
        cursor = CNX.cursor()
        cursor.execute("SELECT id, sub, nom, marque, shop, nutriscore, url"
                       " FROM Product WHERE produit_id = " + str(cat_id) +
                       " AND NOT id = " + str(user_idproduct_choosen) + " AND NOT sub > 1"
                                                                        " ORDER BY nutriscore, RAND() LIMIT 1")
        print(SUB_RESULT)

        result_sub_product = cursor.fetchall()[0]  # gives the substitute
        id_sub = result_sub_product[0]  # gives the id of the substitute in order to save it later
        print(result_sub_product)  # print the substitute

        user_menu = input(SAVE_SUB)

        if user_menu == '1':
            cursor = CNX.cursor()
            # update the table , saving the id of the substitute
            cursor.execute("UPDATE Product SET sub=" + str(id_sub) + " WHERE id=" + str(user_idproduct_choosen))
            CNX.commit()
            print(SUB_SAVED_CONFIRMATION)
    except:
        show_category(cat_id)
```



Analyse des scripts

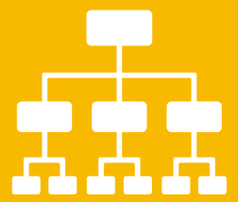
mainscript.py / Main loop / menu_input 2

```python
def main():  # Main function

    print(LOADING_MSG)

    program = True
    new_product = Product()
    new_product.get_products_from_off()
    new_product.send_products_to_db()

    while program:  # Main loop
        print(THE_MAIN_MENU)
        menu_input = input(ENTER_NUMBER)

        if menu_input == '1':  # Show main menu
            menu()
            cat_id = input(ENTER_CAT_NUMBER)
            if cat_id == '1' or '2' or '3':  # Show the chosen category among cat id
                show_category(cat_id)

        if menu_input == '2':  # Show saved products
            saving_products = True
            while saving_products:
                show_saved_products()
                saved_menu = input(SHOW_MORE_SUB)
                if saved_menu == '1':
                    saving_products = False

        if menu_input == '3':  # Quit program and de
            stop_program()
            program = False


if __name__ == '__main__':  # Encapsulation of main
    main()
```

queries.py

fonction show_saved_products

```python
def show_saved_products():  # shows saved products
    """ SHOW SAVED PRODUCTS"""

    try:
        print(SUB_SAVED_MENU)

        cursor = CNX.cursor()
        # display on screen all products with a substitute
        cursor.execute(" SELECT id, sub, nom, marque, shop, nutriscore, url FROM Product WHERE sub > 0")
        for (id, sub, nom, marque, shop, nutriscore, url) in cursor:
            print("Votre produit d'origine -> ID : {} SUBSTITUT : {} NOM : {} MARQUE : {} "
                  "MAGASIN : {} NUTRISCORE : {} LIEN : {}\n".format(id, sub, nom, marque, shop, nutriscore, url))

        user_menu = input(ENTER_IDSUB_NUMBER_WANTED)
        if user_menu: # display on screen the substitute choosen for the initial product
            cursor.execute(" SELECT id, nom, marque, shop, nutriscore, url FROM Product WHERE id= "+ str(user_menu))
            for (id, nom, marque, shop, nutriscore, url) in cursor:
                print("Votre substitut -> ID : {} NOM : {} MARQUE : {} MAGASIN : {} NUTRISCORE : {} LIEN : {}\n".format(
                    id, nom, marque, shop, nutriscore, url))
    except:
        print(PROBLEM_SUB_MENU)
```



Analyse des scripts

mainscript.py / Main loop / menu_input 3

```python
def main():  # Main function

    print(LOADING_MSG)

    program = True
    new_product = Product()
    new_product.get_products_from_off()
    new_product.send_products_to_db()

    while program:  # Main loop
        print(THE_MAIN_MENU)
        menu_input = input(ENTER_NUMBER)

        if menu_input == '1':  # Show main menu
            menu()
            cat_id = input(ENTER_CAT_NUMBER)
            if cat_id == '1' or '2' or '3':  # Show the chosen category among cat id
                show_category(cat_id)

        if menu_input == '2':  # Show saved products
            saving_products = True
            while saving_products:
                show_saved_products()
                saved_menu = input(SHOW_MORE_SUB)
                if saved_menu == '1':
                    saving_products = False

        if menu_input == '3':  # Quit program and delete all data
            stop_program()
            program = False

if __name__ == '__main__':  # Encapsulation of main function
    main()
```

```python
def stop_program():  # Delete all data and quit
    """ STOP PROGRAM FUNCTION"""
    try:
        cursor = CNX.cursor()
        cursor.execute("DROP TABLE IF EXISTS Product, Category")
        cursor.execute("DROP DATABASE IF EXISTS OPENFOODFACTS")
    except:
        print(PROBLEM_DB_SUPPR)
    finally:
        CNX.close()
        print(GOODBYE_MSG)
```

fonction stop_program

queries.py

L’algorithme dans son ensemble

1. Lancement du programme en ligne de commande

2. Création de la base de données relationnelle MySQL « OPENFOODFACTS » en local ainsi que ses tables (avec clef étrangère)

3. Récupération des données avec la librairie Requests via l’API OpenFoodFacts

4. Stockage des données dans une liste Python

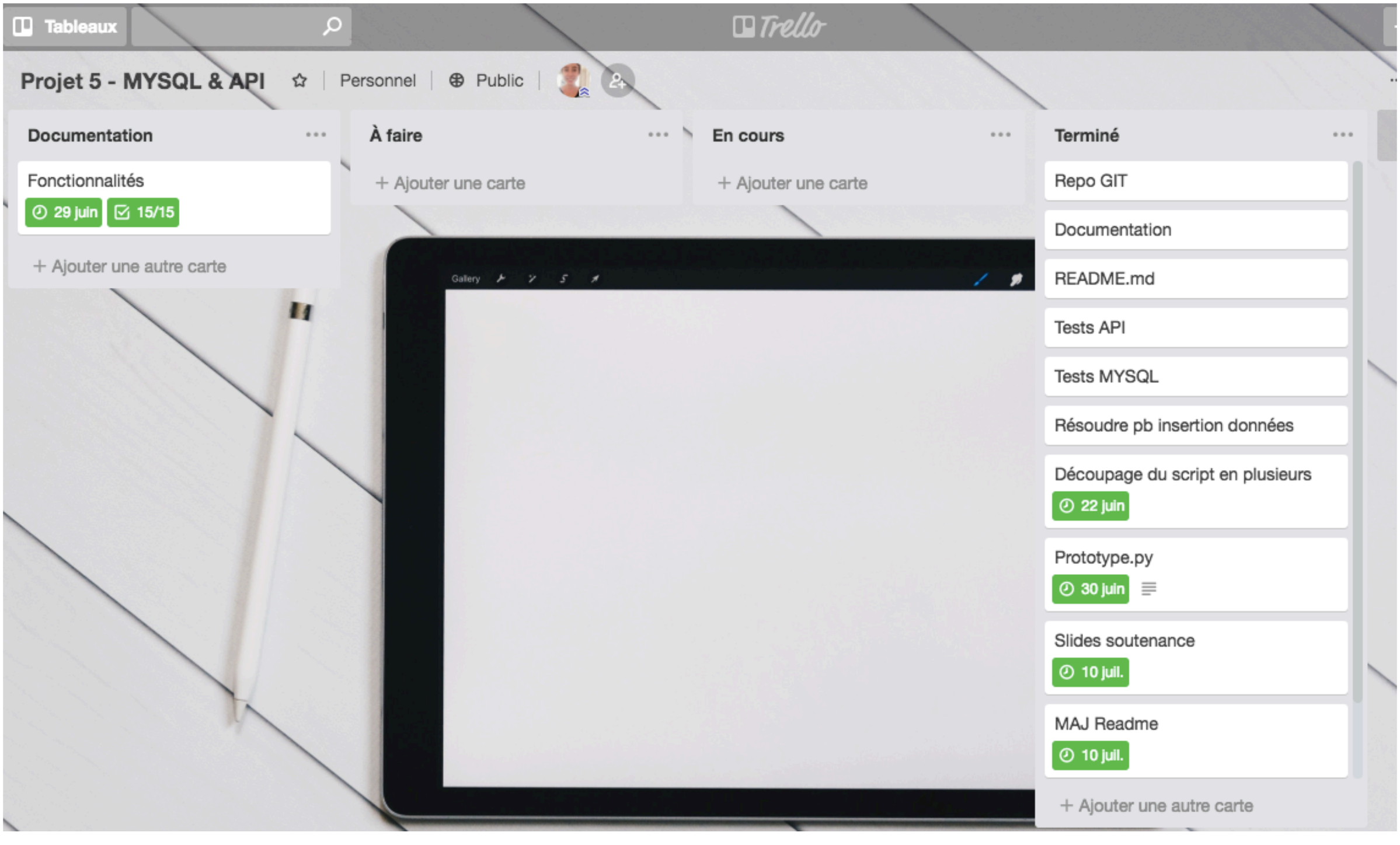
5. Insertion des données récupérées dans les tables de la BDD précédemment créée

6. Affichage du menu principal permettant au client de choisir entre le choix d’une catégorie de produit, l’affichage de ses produits sauvegardés ou de quitter le programme.

7. Si le client décide d’afficher une catégorie de produit, il peut choisir d’en sauvegarder un à la fois. A terme, les produits sauvegardés sont accessibles via le menu principal

8. A tout moment, le client peut quitter le programme. Si il fait ce choix, toutes les données sont effacées.

Gestion du projet de manière Agile



Etapes de la production

1. Découvertes des API et de MySQL

2. Tests des librairies requests et MySQL.connector puis des insertions de données

3. Création de scripts python avec commits réguliers sur GitHub

4. Améliorations des scripts

5. Obtention d’un produit fonctionnel

Optimisations potentielles

- Ajout de catégories de produits
- Création d’une interface graphique
- Optimisation des scripts python