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

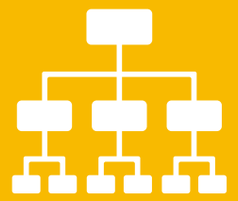
Modèle de la base de données

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

create_db.py / création de la base de données

```
11
12      try:
13          cursor = cnx.cursor()
14          create_db = "CREATE DATABASE IF NOT EXISTS `OPENFOODFACTS` CHARACTER SET 'utf8'"
15          cursor.execute(create_db)
16
17          param_user = "GRANT ALL PRIVILEGES ON OPENFOODFACTS.* TO 'root'@'localhost' "
18          cursor.execute(param_user)
19          cnx.commit()
20
21      except:
22          print("Un problème est survenu lors de la création de la base de données.")
23      finally:
24          cnx.close()
```

Analyse des scripts

sql.py / création des tables

```
11      try:
12          CURSOR = CNX.cursor()
13          CREATE_TABLE = "CREATE TABLE IF NOT EXISTS `Category` (" \
14                         "`id` SMALLINT unsigned NOT NULL AUTO_INCREMENT," \
15                         " `Produit` VARCHAR(40) NOT NULL," \
16                         " PRIMARY KEY (`id`)" \
17                         ") ENGINE=InnoDB"
18          CURSOR.execute(CREATE_TABLE)
19          CNX.commit()
20
21          CREATE_TABLE_1 = "CREATE TABLE IF NOT EXISTS `Product` (" \
22                           " `id` SMALLINT unsigned NOT NULL AUTO_INCREMENT," \
23                           " `produit_id` SMALLINT unsigned NOT NULL," \
24                           "`sub` TEXT NOT NULL," \
25                           "`nom` TEXT NOT NULL," \
26                           "`marque` TEXT NOT NULL," \
27                           "`nutriscore` TEXT  NOT NULL," \
28                           "`url` TEXT NOT NULL," \
29                           "PRIMARY KEY (`id`)," \
30                           "KEY `fk_produit_id` (`produit_id`)," \
31                           "CONSTRAINT `fk_produit_id` FOREIGN KEY (`produit_id`) REFERENCES Category(id)" \
32                           ") ENGINE=InnoDB"
33          CURSOR.execute(CREATE_TABLE_1)
34          CNX.commit()
35
36
37      except:
38          print(Fore.RED + Back.WHITE + Style.BRIGHT +
39                "Les tables existent déjà dans la base de données OPENFOODFACTS.")
40      finally:
41          CNX.close()
```

Analyse des scripts

cnx_db.py / connexion à la base de données

```
5   import mysql.connector
6
7   CNX = mysql.connector.connect(user='root',
8                                 password='',
9                                 host='localhost',
10                                database='OPENFOODFACTS',
11                                auth_plugin='mysql_native_password')
```

Analyse des scripts

constantes.py / constantes du programme

```
 8      CAT_DICT = {1: 'https://fr.openfoodfacts.org/categorie/jus-d-orange/',
 9                  2: 'https://fr.openfoodfacts.org/categorie/pates-a-tartiner-au-chocolat/',
10                  3: 'https://fr.openfoodfacts.org/categorie/biscottes/'}
11
12      NUTRI_DICT = {1: 'a', 2: 'b', 3: 'c', 4: 'd', 5: 'e'}
13
14      for key, value in CAT_DICT.items():
15          if key:
16              R = requests.get(value + '1.json')
17              TABLE_JSON_PAGE = R.json()
18              PAGE_SIZE = TABLE_JSON_PAGE[u'page_size']  # return number of products by page
19              COUNT = TABLE_JSON_PAGE[u'count']  # return number of product by category
20              TOTAL_PAGE_NUMBER = int(COUNT / PAGE_SIZE) + 1  # return number of pages by category
```

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

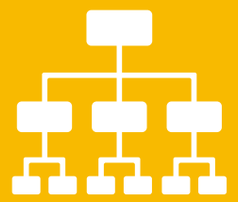
feed_in.py /
la classe product /
méthode de classe
get_products_from_off

```
22      class Product:
23          """Class Product : get products from OFF, send products to database """
24
25          def __init__(self):...
27
28          def get_products_from_off(self):  # method to get data with requests' module
29              try:
30                  for page in range(0, 1):
31                      for key, value in CAT_DICT.items():
32                          if key:
33                              rpage = requests.get(value + str(page + 1) + '.json')
34                              TABLE_JSON_PAGE = rpage.json()
35                              products_by_page = TABLE_JSON_PAGE[u'products']
36
37                              for products in products_by_page:  # Fill the list of pro
38
39                                  id_product = key
40                                  sub = 0
41                                  name = products['product_name']
42                                  brand = products['brands']
43
44                                  nutri_origin = products['nutrition_grades_tags'][0]
45                                  for key_nutri, value_nutri in NUTRI_DICT.items():
46                                      if nutri_origin == value_nutri:
47                                          nutriscore = key_nutri
48
49                                  url = products['url']
50                                  self.products_data.append([id_product,
51                                                             sub,
52                                                             name,
53                                                             brand,
54                                                             nutriscore,
55                                                             url])
56              except:
57                  print(Fore.RED + Back.WHITE + Style.BRIGHT +
58                        "Un problème est survenu lors de la " +
59                        "récupération des produits.")
60
61          def send_products_to_db(self):...
75
```

Analyse des scripts

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

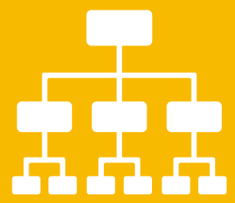
```
22      class Product:
23          """Class Product : get products from OFF, send products to database """
24
25          def __init__(self):...
27
28          def get_products_from_off(self):...
60
61      def send_products_to_db(self):  # method to send products to database
62          try:
63              data = self.products_data[:]  # all data got from OFF
64              cursor = cnx.cursor()
65              cursor.execute("INSERT INTO `Category` (`id`, `Produit`) "
66                             "VALUES (NULL, 'Jus d''orange'), "
67                             "(NULL, 'Pâte à tartiner au chocolat'),"
68                             "(NULL, 'Biscottes')")
69              cursor.executemany("INSERT INTO Product(id, produit_id, sub, nom, marque, nutriscore, url) "
70                                 "VALUES (NULL, %s, %s, %s, %s, %s, %s)", data)
71              cnx.commit()
72          except:
73              print(Fore.RED + Back.WHITE + Style.BRIGHT +
74                    "Un problème est survenu lors de l'insertion des données dans la BDD.")
```



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

```
23      def main():  # Main function
24
25          print(Fore.BLACK + Back.CYAN + Style.BRIGHT + "\nChargement en cours...")
26
27          program = True
28          new_product = Product()
29          new_product.get_products_from_off()
30          new_product.send_products_to_db()
31
32          while program:  # Main loop
33              print(Fore.WHITE + Back.BLACK + Style.BRIGHT + "\n- Menu -" +
34                    "\n1 - Quel aliment souhaitez-vous remplacer ?" +
35                    "\n2 - Retrouver mes aliments substitués" +
36                    "\n3 - Quitter le programme")
37              menu_input = input(
38                  Fore.BLACK + Back.CYAN + Style.BRIGHT + "\nEntrez le chiffre correspondant à votre souhait : ")
39
40              if menu_input == '1':...
51
52              if menu_input == '2':...
59
60              if menu_input == '3':...
63
64
65  if __name__ == '__main__':  # Encapsulation of main function
66      main()
```

Analyse des scripts

mainscript.py / Main loop / menu_input 1

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

    print(Fore.BLACK + Back.CYAN + Style.BRIGHT + "\nChargement en cours...")

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

    while program:  # Main loop
        print(Fore.WHITE + Back.BLACK + Style.BRIGHT + "\n- Menu -" +
              "\n1 - Quel aliment souhaitez-vous remplacer ?" +
              "\n2 - Retrouver mes aliments substitués" +
              "\n3 - Quitter le programme")
        menu_input = input(
            Fore.BLACK + Back.CYAN + Style.BRIGHT + "\nEntrez le chiffre correspondant à votre souh

        if menu_input == '1':  # Show main menu
            menu()
            cat_id = input(Fore.BLACK + Back.CYAN + Style.BRIGHT + "\nEntrez le chiffre correspondan
                           "catégorie que vous désirez afficher : ")
            if cat_id <= '3':  # Show the chosen category among cat id
                show_category(cat_id)
                user_idproduct_choosen = input(
                    Fore.WHITE + Back.BLACK + Style.BRIGHT + "\nTapez l\'ID attribué au " +
                    "produit pour afficher une proposition de substitut : ")
                if user_idproduct_choosen:  # Save a substitute
                    substitutes(cat_id, user_idproduct_choosen)

        if menu_input == '2':...

        if menu_input == '3':...
```

queries.py

fonction menu

```python
def menu():  # display main menu
    """ THE MAIN MENU"""
    try:
        print(Fore.BLACK + Back.WHITE + Style.BRIGHT + '\n- Sélectionnez la catégorie -')
        cursor = CNX.cursor()
        cursor.execute("SELECT * FROM Category")
        for (id, Produit) in cursor:
            print("{} - {}".format(id, Produit))
    except:
        print(Fore.RED + Back.WHITE + Style.BRIGHT + "Impossible d'afficher le 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, nutriscore, url "
                       " FROM Product WHERE produit_id = " + str(cat_id) +
                       " AND NOT nutriscore='unknown'")
        for (id, nom, marque, nutriscore, url) in cursor:
            print("ID : {} NOM : {} MARQUE : {} NUTRISCORE : {} LIEN : {}".format(
                id, nom, marque, nutriscore, url))
    except:
        menu()
```

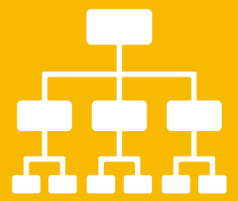
fonction substitutes

```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, 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(Fore.BLACK + Back.CYAN + Style.BRIGHT + "\nNous vous proposons le substitut suivant:")

        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(Fore.BLACK + Back.CYAN + Style.BRIGHT +
                          '\nVoulez-vous sauvegarder ce substitut ? ' +
                          '(tapez 1, sinon tapez "entrer") : ')

        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(Fore.BLACK + Back.CYAN + Style.BRIGHT + "Produit sauvegardé !")
    except:
        show_category(cat_id)
```



Analyse des scripts

mainscript.py / Main loop / menu_input 2

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

    print(Fore.BLACK + Back.CYAN + Style.BRIGHT + "\nChargement en cours...")

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

    while program:  # Main loop
        print(Fore.WHITE + Back.BLACK + Style.BRIGHT + "\n- Menu -" +
              "\n1 - Quel aliment souhaitez-vous remplacer ?" +
              "\n2 - Retrouver mes aliments substitués" +
              "\n3 - Quitter le programme")
        menu_input = input(
            Fore.BLACK + Back.CYAN + Style.BRIGHT + "\nEntrez le chiffre correspondant à votre souhait : ")

        if menu_input == '1':...

        if menu_input == '2':  # Show saved products
            saving_products = True
            while saving_products:
                show_saved_products()
                saved_menu = input("\nVoulez-vous afficher d'autres substituts ? (tapez 'entrer' sinon tapez 1) : ")
                if saved_menu == '1':
                    saving_products = False

        if menu_input == '3':...

```

queries.py

fonction show_saved_products

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

    try:
        print(Fore.BLACK + Back.WHITE + Style.BRIGHT + '\n- Mes produits sauvegardés -')

        #Affiche tous les produits d'origine
        cursor = CNX.cursor()

        cursor.execute(" SELECT id, sub, nom, marque, nutriscore, url FROM Product WHERE sub > 0")
        for (id, sub, nom, marque, nutriscore, url) in cursor:
            print("Votre produit d'origine -> ID : {} SUBSTITUT : {} NOM : {} MARQUE : {} NUTRISCORE : {} LIEN : {}\n".format(
                id, sub, nom, marque, nutriscore, url))

        user_menu = input("\nTapez le numéro du SUBSTITUT que vous désirez afficher : ")
        if user_menu: # display on screen the substitute choosen for the initial product
            cursor.execute(" SELECT id, nom, marque, nutriscore, url FROM Product WHERE id= "+ str(user_menu))
            for (id, nom, marque, nutriscore, url) in cursor:
                print("Votre substitut -> ID : {} NOM : {} MARQUE : {} NUTRISCORE : {} LIEN : {}\n".format(
                    id, nom, marque, nutriscore, url))
    except:
        print("Impossible d'afficher les produits sauvegardés.")
```

Analyse des scripts

mainscript.py / Main loop / menu_input 3

```
23      def main():  # Main function
24
25          print(Fore.BLACK + Back.CYAN + Style.BRIGHT + "\nChargement en cours...")
26
27          program = True
28          new_product = Product()
29          new_product.get_products_from_off()
30          new_product.send_products_to_db()
31
32          while program:  # Main loop
33              print(Fore.WHITE + Back.BLACK + Style.BRIGHT + "\n- Menu -" +
34                    "\n1 - Quel aliment souhaitez-vous remplacer ?" +
35                    "\n2 - Retrouver mes aliments substitués" +
36                    "\n3 - Quitter le programme")
37              menu_input = input(
38                  Fore.BLACK + Back.CYAN + Style.BRIGHT + "\nEntrez le chiffre correspondant à votre souhait : ")
39
40              if menu_input == '1':...
51
52              if menu_input == '2':...
59
60              if menu_input == '3':  # Quit prog
61                  stop_program()
62                  program = False
63
```

```
95      def stop_program():  # Delete all data and quit
96          """ STOP PROGRAM FUNCTION"""
97          try:
98              cursor = CNX.cursor()
99              cursor.execute("DROP TABLE IF EXISTS Product, Category")
100             cursor.execute("DROP DATABASE IF EXISTS OPENFOODFACTS")
101         except:
102             print("Un problème est survenu lors de la suppression des données.")
103         finally:
104             CNX.close()
105             print(Fore.BLACK + Back.WHITE + Style.BRIGHT +
106                   "\nMerci d'avoir utilisé notre programme. À bientôt.")
107
```

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

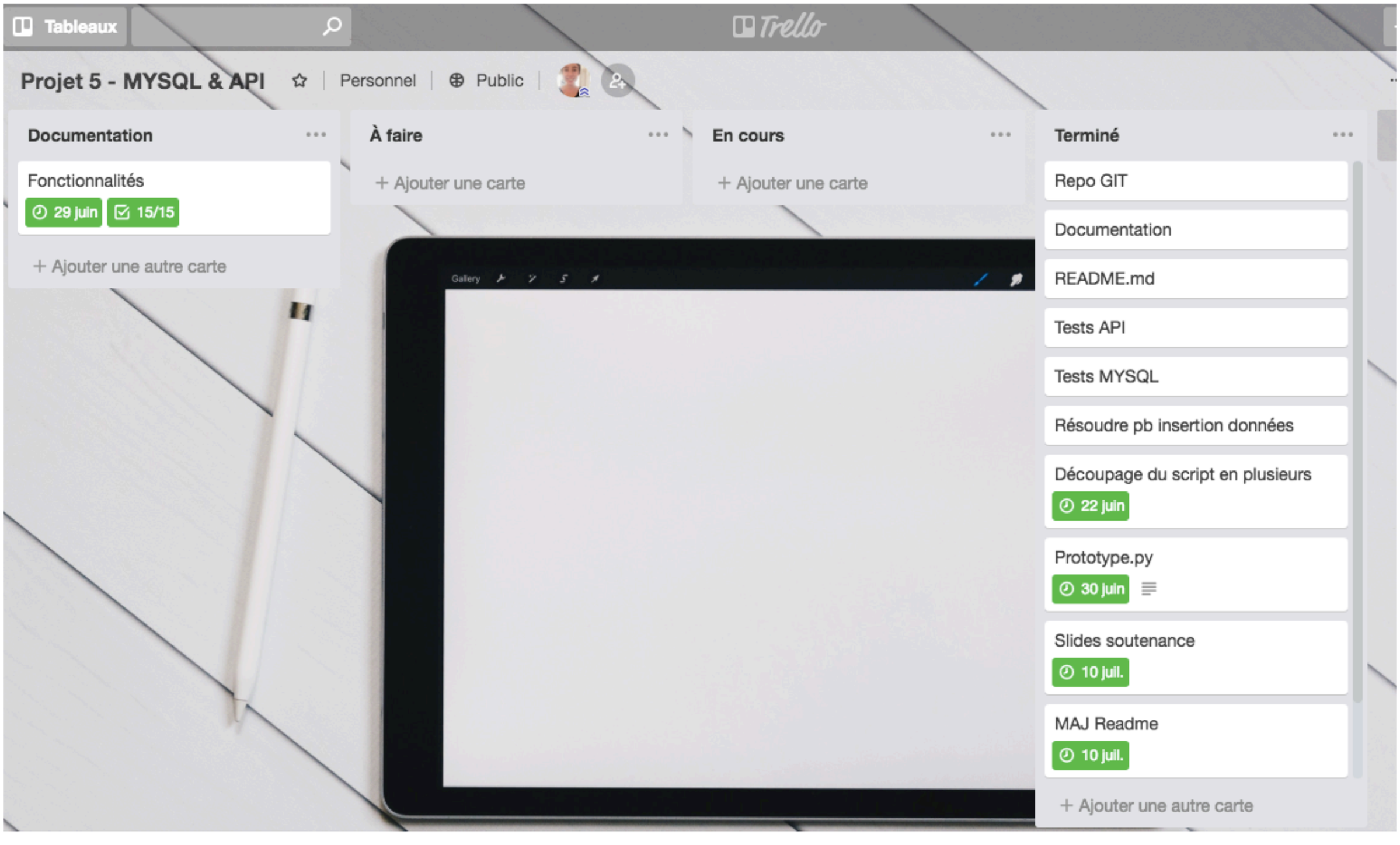
5. Insertion des données récupérées dans les tables de la BDD précédemment créée

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