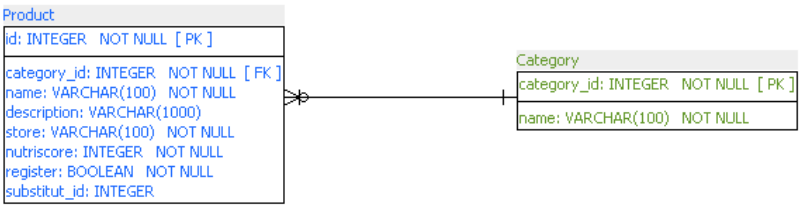
En premier lieu, lorsque je me suis lancé dans ce projet, j'ai choisi d'appliquer les instructions à la lettre. **Organisation du travail.**

J'ai choisi Trello et créer un tableau pour séquencer mon projet, et ainsi appliquer une méthode agile pour en venir à bout. Voici le lien : https://trello.com/b/aYvwzICu/projet-5-da-python

Puis j'ai créé mon premier repository GitHub à l'adresse : https://github.com/cenotapHe/OpenFoodFacts. Dans la foulée j'ai aussi écrit le fichier README associé, afin de me donner une direction.

J'ai alors rencontré mon mentor pour ce projet : Benoît Prieur. Avec qui nous avons fixé des objectifs pour ce projet. Une semaine pour maîtriser l'API, construire les scripts et la base de données. Une semaine de plus pour avoir une application fonctionnelle et ainsi demandé le passage en soutenance. Cette fixation d'objectif m'a ainsi porté dans ma motivation tout au long de ce projet.

Ayant déjà fait le projet 6 de DA Python avant celui-ci, je n'ai eu aucun soucis à créer une base de donnée fonctionnelle ainsi que le Modèle Physique de Données. Je me sers de MySQL pour administrer ma base de données, et je l'ai créé au travers de SQL Power Architect.



J'ai eu plusieurs difficultés avant de prendre correctement l'API d' OpenFoodFacts en main. Car j'étais parti dans la mauvaise direction en collectant les produits à travers leur code barre. En plus d'avoir énormément d'erreur que je n'expliquais pas à ce stade du développement, j'avais aussi un script extrêmement lent. Le tout premier lancement m'a pris une nuit entière pour collecter un peu moins de 100 articles. Mais grâce à un algorithme qui me permettait de switcher entre les noms des catégories, plusieurs pages pour chaque catégorie et plusieurs articles sur chaque page, j'ai pu créer un fichier JSON me permettant de construire par la suite une base de données suffisamment conséquente pour cette application.

```
# Variable for the main boucle
i = 1
i_max = 20

# Variable for the category
count = 0
list_category = []
i_category = 0
name_category = ['hamburgers', 'viennoiseries', 'bonbons', 'mueslis', 'pizzas']

# Main Boucle
while i <= i_max:

    # API request
    os.system("curl -X GET https://fr-en.openfoodfacts.org/category/{}/{}.json --output fichier2.json".format(
        name_category[i_category], str(i)))
```

Je me suis d'ailleurs permit de créer deux scripts différents en fonction de l'utilisation que l'on veut en faire. Le premier script importe chaque article dans un fichier SQL, qui permet de remplir facilement la base de données précédemment créé par le fichier de SQL Power Architect. Le deuxième script quant à lui, utilise directement les fichiers SQL pour créer un fichier de base de données en « .db » avec les tables créées et remplies.

Ainsi la première semaine s'est conclue avec les objectifs réalisés en avance. Vient alors le temps de créer l'application en elle-même avec la rédaction du fichier « main.py ». La création des menus et des choix pour l'utilisateur ne posa aucuns soucis.

Par contre en naviguant sur le net, j'ai commencé à essayer de faire dialoguer mon application python et ma base données MySQL avec le module sqlite3. Mais j'ai rencontré beaucoup d'erreur. Lors de mon rendez-vous hebdomadaire, mon mentor m'aiguilla alors sur « MySQL connector ». Un module qui me prit beaucoup d'énergie pour arriver à le faire fonctionner correctement. Finalement je parviens à l'installer sur une ancienne version de python, la 3.4. Du coup pour exécuter mon main, je dois taper la commande « py -3.4 main.py », mais tout fonctionne correctement.

```
import mysql.connector
```

```
def select_from(self):
    """With mySQL connector, this fonction do a SELECT FROM"""
    cnx = mysql.connector.connect(
        host='localhost', database='openfoodfacts', user='user', password='password')

    cursor = cnx.cursor()

    query = (self)
    cursor.execute(query)

    tupple = []

    for i in cursor:
        variable = i
        tupple.append(variable)

    return tupple
```

Ayant une appli qui fonctionne correctement à ma base de données, il ne me restait plus qu'à sauvegarder en mémoire les substituts trouvés. Pour cela, dans ma base de données, sur la table recensant les produits, je me suis servi de deux colonnes : register & substitut_id.

La première est un booléen qui me permet de dire si le produit a déjà été substitué ou non. La deuxième conserve en mémoire le l'id du substitut ainsi sauvegardé.

Tout au long de ce projet, j'ai effectué des push réguliers sur mon git hub. Et aujourd'hui, j'écris les livrables de ce projet. Dans l'attente de la soutenance pour le valider.