# Compte-Rendu Projet 8

Le but de ce projet est la création d’un site Web permettant de comparer des produits alimentaires sur la base de leur qualité nutritionnelle.

Les contraintes essentielles des attendus sont l’utilisation du framework Django et un déploiement sur Heroku.

## LIENS UTILES

Site sur Heroku : <https://papounet-diet.herokuapp.com/home/>

Git Hub : <https://github.com/Fabrice-64/papounet_diet>

LucidChart : [lucidchart](https://app.lucidchart.com/lucidchart/d90d5817-293e-45ce-8b62-a6dce5a37387/view?page=LVDFdq5QNebm) (dont le plan de tests)

Trello : <https://trello.com/b/7oAUZGUZ/p8betterdiet>

## Démarche CHOISIE

La démarche choisie a été de considérer l’écriture du code comme la conséquence à la fois du travail de conception amont et du développement d’une architecture de tests la plus solide possible.

C’est pourquoi ont été développés des diagrammes d’impact, de classe, de relation entre les entités, ainsi qu’un BPMN. Si le déroulement du développement a conduit à dévier de ces travaux préliminaires, ils lui ont tout de même servi d’ossature[[1]](#footnote-1).

Dans un deuxième temps, le développement s’est fait autour de deux user stories, jalonnées par des tests d’assertion utilisant Selenium. Autant que faire se peut, le reste du code en a été déduit, ce qui a permis d’atteindre un taux de couverture des tests d’un minimum de 84%.

Pour autant, des modules repris directement des tutoriels officiels de Django ont fait l’objet de tests et d’amélioration limités : le code employé pour l’identification des utilisateurs a été à peine adapté. En effet, dans le cadre d’une optimisation du temps, il m’a semblé pertinent de faire confiance à un code développé par des experts et éprouvé par des milliers de développeurs.

L’effort de test a donc porté sur l’app food\_items, destinée à la gestion des produits alimentaires, depuis leur importation jusqu’à leur mise à disposition de l’utilisateur.

## PARTIS PRIS

1. Le cahier des charges exprime la contrainte d’accéder à une sélection de produits en 1 clic. Pour rester dans ce cadre, la méthode de recherche de produits de substitution s’appuie sur les mots-clés saisis par l’utilisateur et non pas les catégories : il s’agissait bien de considérer le cas d’une jeune femme pressée qui saisit quelques mots clés, obtient une sélection de produits semblables et peut d’emblée obtenir quelques renseignements complémentaires.

2. Heroku, sur lequel l’application doit être déployée, autorise un maximum de 10.000 lignes de données dans une base PostgreSQL[[2]](#footnote-2). Or les catégories Open Food Facts comme les magasins sont au nombre de plusieurs milliers. Le choix a donc été fait de charger les catégories rattachées à plus de 10.000 articles et les magasins offrant plus de 1.000 références de produits. Ainsi, plus de lignes sont restées disponibles pour les produits, puisqu’ils sont au cœur de ce projet. Pour garder un échantillon de produits représentatif (près de 3.000), tout en restant sous le plafond de 10.000 lignes, il a été décidé in extremis de ne garder qu’une seule catégorie par produit.

## DIFFICULTES RENCONTREES

1. La compréhension du cheminement suivi par les requêtes a pris beaucoup de temps. Comment elles passent de la page HTML à des aiguillages, pour atterrir sur les Views et repartir soit vers la même page, soit vers une autre, en reprenant ces mêmes aiguillages a pris du temps.

Les tests ont beaucoup aidé à comprendre cette logique (HttpResponse, et RequestFactory).

2. L’élaboration de tests pertinents a provoqué un foisonnement de tests s’entrecroisant et une multiplication des données utilisées, soit avec des fichiers utilisés soit Mock de DB, des Mocks de requêtes, etc. Ce qui a été initialement perçu comme une surabondance de données de test confère en revanche plus de modularité et permet d’adapter le test au plus près de chaque méthode ou chaque fonctionnalité. Le bilan reste donc positif.

3. Le déploiement sur Heroku s’est accompagné de régressions, notamment en local et sur les tests. L’obstacle a été contourné en créant des environnements différents, permettant ainsi d’avoir une configuration compatible avec un déploiement et une autre pour le développement et les tests.

4. Pour aider à la résolution de cette difficulté, un niveau d’arborescence a été supprimé dans le projet et un nouveau repository a été créé sur GitHub. Pour faciliter l’évaluation de ce projet, l’ancien repository a été conservé sous son nom d’origine : OC\_Project\_8.

## DEVELOPPEMENTS ULTERIEURS

Ce projet m’a permis de tout apprendre de ce framework très riche et de m’apercevoir qu’il me faudrait des mois de travail pour en exploiter tout le potentiel.

Les développements que je discerne pour une future version de cette application sont :

1. Un renforcement de la sécurité en exploitant mieux les possibilités de Django. Actuellement la fonctionnalité ‘manage.py –check deploy’ met à jour 4 points d’amélioration.

2. Les contraintes portant sur les mots de passe devraient être également renforcées.

3. Ajouter des stop-words pour la recherche des produits et éviter ainsi l’omission de résultats pertinents.

4. Ajouter une carte Google à la page coordonnées de Pur Beurre.

5. Ajouter un login par Facebook ou un autre réseau social.

6. Ajouter une fonctionnalité de modification des produits (ex : ajout de magasins).

7. Modification des coordonnées de l’utilisateur.

8. Améliorer les graphismes.

Bien évidemment, cela vient en complément des évolutions demandées par la société Pur Beurre.

1. Ces travaux sont accessibles sur LucidChart, dont le lien est donné en tête de ce document. [↑](#footnote-ref-1)
2. Il s’agit ici de rester dans la version hobby-dev. [↑](#footnote-ref-2)