

STRATÉGIE DE TESTS

PROJET 8

Ce document présente la stratégie de tests adoptée pour le premier projet Django de la formation.

Organisation

Le projet web PurBeurre fait appel à une base de données. On identifie donc les parties à tester ainsi:

- Backend
- Modèles
- Templates et navigateur

De manière plus détaillée PurBeurre se compose de:

- 1 module noyau créé avec la commande *manage.py startproject PurBeurre*
- 2 applications Django (*foodfacts* et *roles*)

Je choisis de créer un fichier de test dans chaque application qui testera les vues et formulaires de chacune.

L'utilisateur trouvera donc *test_foodfacts.py* et *test_roles.py* dans leurs répertoires respectifs.

J'ai déjà importé les modèles dans *foodfacts*. J'y place donc **d'une part** les commandes permettant le remplissage de la base de données et l'appel des différents outils de tests avec *manage.py* et **d'autre part** le module qui permet d'interagir avec la base de données.

Les modèles seront utilisés par les deux applications donc les tests de modèles seront placés dans le dossier *PurBeurre* dans *test_roles.py*.

De la même manière, les tests côté client (dans le navigateur) seront appelés depuis *Purbeurre* car ils concernent les deux applications.

Du fait de la complexité de ce projet par rapport au précédent :

- Chaque nouvelle vue sera testée avant d'être (si besoin) complétée pour vérifier qu'elle retourne bien un code 200.
- Puis les modèles seront testés
- Puis chaque fonctionnalité sera testée une fois achevée
- Puis l'ensemble des fonctionnalités sous la forme de tests d'intégration
- Puis les tests de fonctionnalités dans le navigateur (Chrome)
- Puis de nouveau les tests de modèles si jamais les modèles ont été modifiés au cours du projet
- L'ajout du lancement de l'ensemble des tests depuis *manage.py* permettra de tous les lancer.
- Un test de couverture est prévu afin de s'assurer d'un taux de 80 %
- La conformité à PEP-8 sera évaluée tout à la fin

Outils

Bakery permettra de tester les modèles.

Pytest permettra de lancer les tests unitaires et d'intégration. Il commencera en effet par les précédentes fonctionnalités déjà vérifiées et finira par la dernière à avoir été développée.

Selenium permettra de tester les fonctionnalités dans le navigateur.

Un module *commands* ajoutera la commande de lancement des tests avec simplement *python manage.py test*.

Coverage permettra d'évaluer le taux de couverture.

Flake8 sera utilisé pour la conformité à PEP-8.

Notez que Coverage requiert d'exclure les fichiers non testés d'où la commande :

```
coverage report --omit=foodfacts\test_foodfacts.py  
,test_roles.py,roles\apps.py,roles\views.py,foodfacts\views.py,PurBeurre\  
test_selenium_checks.py,roles>EmailBackend.py,foodfa  
cts\migrations\0002_favorites_front_img.py,foodfacts\migrations\0001_initial.py,roles\  
migrations\0001_initial.py,roles\migrat  
ions\0002_delete_users.py,roles\models.py,roles\apps.py,foodfacts\  
collector.py,foodfacts\apps.py,PurBeurre\wsgi.py,PurBeurre\  
test_model.py,PurBeurre\asgi.py,roles\test_roles.py,PurBeurre\constants.py
```