Plan de tests : OC projet 13 :

Périmètre des tests :

Tests unitaires :

Pour chaque app seront testés les modules views, models, forms lorsqu'ils existent et que c'est pertinent.

Tests d'intégration :

Test d'affichage des postes disponibles à l'échange (page d'accueil).

Test d'envoi de mail via le formulaire de contact.

Test de création de compte (avec un formulaire rempli correctement, avec une confirmation de mot de passe invalide, avec un e-mail déjà enregistré).

Test de connection à son compte (avec un formulaire rempli correctement, avec un e-mail erroné, avec un mot de passe erroné).

Test de déconnection à son compte.

Test de manipulations dans la page "mon compte".

Test de manipulations dans la page "ma recherche" (enregistrement d'une recherche de permutation, modification d'une recherche de permutation, suppression d'une recherche de permutation).

Test de modification du statut d'une permutation.

Outils de tests :

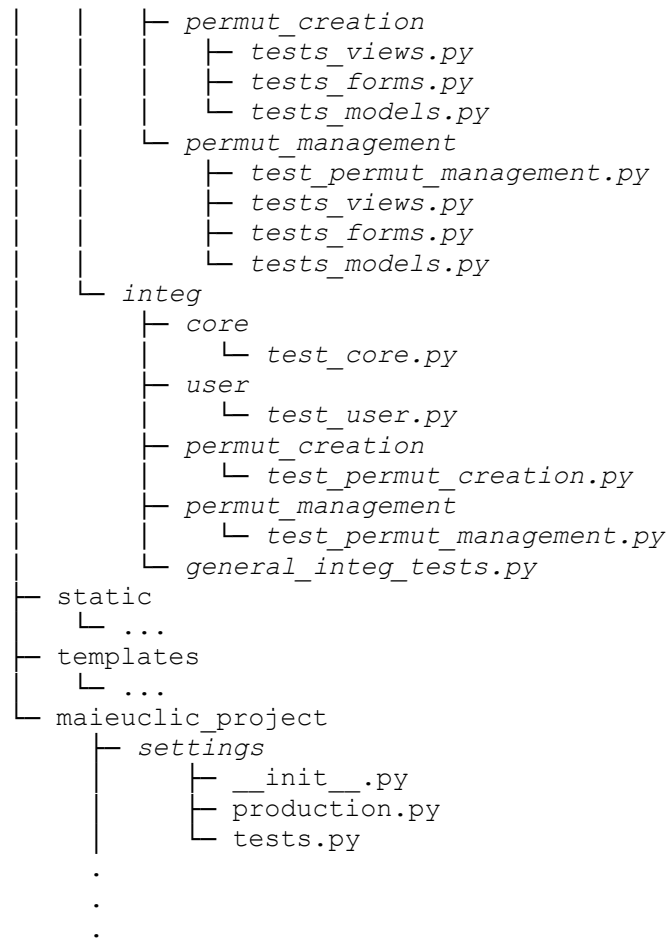
Utilisation de unittest avec django.test pour les tests unitaires.

Utilisation de selenium pour les tests d'intégration.

Organisation des tests :

Le projet contient un dossier tests, comprenant un sous-dossier unit qui contient un sous-dossier spécifique pour chaque app, avec un fichier de tests pour chaque module à tester (views, forms, models...). Le dossier tests contient également un dossier integ lui-même réparti en sous-dossiers pour chaque app :

```
racine
 ├─ manage.py
 ├─ user
 │   └─ ...
 ├─ permut_creation
 │   └─ ...
 ├─ permut_management
 │   └─ ...
 ├─ core
 │   └─ ...
 ├─ tests
 │   ├─ unit
 │   │   ├─ core
 │   │   │   └─ tests_views.py
 │   │   ├─ user
 │   │   │   ├─ tests_views.py
 │   │   │   ├─ tests_forms.py
 │   │   │   └─ tests_models.py
```



Execution des tests :

Pour ce projet, j'ai choisi d'utiliser le Test Driven Development et donc de développer les tests avant de développer les fonctionnalités, afin d'expérimenter cette méthode (pour les projets précédents j'avais développé les tests après les fonctionnalités). Les tests sont exécutés grâce à coverage avec la commande coverage run, suivie de coverage html si l'on souhaite un rapport détaillé des tests.

Pour la parti permut_management, j'ai développé l'application avant de développer les tests.

Rapport des tests :

Pour ce projet, lors de sa première mise en production, nous avions une couverture de 95%. L'exécution des tests a passé 51 tests avec succès. Pour plus de détails concernant les résultats et la couverture : https://github.com/elwaze/Maieuclic/tree/master/maieuclic_project/coverage_html_report