Site web 2.0 - caractéristiques et améliorations

1. Amélioration de l’architecture modulaire

Nous optimisons notre architecture de site web en réduisant les problèmes de notre conception monolithique actuelle. Nous allons :

- réorganiser notre code en plusieurs applications distinctes ;
- déplacer les fichiers HTML du site dans des dossiers de templates spécifiques à chaque application.

Cette optimisation améliorera la flexibilité, la maintenabilité et l'évolutivité de notre code.

Aspects techniques

Nous prévoyons de séparer “oc_lettings_site” en deux applications distinctes, "lettings" et "profiles", pour améliorer la modularité, l'extensibilité et la séparation des fonctionnalités. Voici les modifications à effectuer pour cette optimisation :

- créer une nouvelle application "lettings", contenant les modèles "Address" et "Letting" ;
- remplir les nouvelles tables avec les données déjà présentes dans la base de données en utilisant les fichiers de migration Django. Attention, il ne faut pas utiliser le langage SQL directement dans le fichier de migration ;
- créer une nouvelle application "profiles", contenant le modèle "Profile" ;
- répéter l’opération de migration pour cette nouvelle application ;
- en utilisant les migrations Django, supprimer les anciennes tables de la base de données ;
- remplacer les templates de manière cohérente dans les nouvelles applications ;

- déplacer les URL de "lettings", "profiles", "lettings_index" et "profiles_index" de "oc_lettings_site" vers les nouvelles applications, en gardant ROOT_URLCONF identique ;
- créer des espaces de nom pour les URL des deux nouvelles applications ;
- renommer "lettings_index.html" en "index.html" et la vue "lettings_index" en "index". Effectuer la même opération l'application "profiles", renommer "profiles_index.html" en "index.html" et la vue "profiles_index" en "index" ;
- supprimer les fichiers inutiles dans l'application "oc_lettings_site".

Le site (y compris la section d'administration) ne doit pas changer d'apparence ni de fonctionnalités, car il s'agit d'une pure refactorisation. Les outils de développement locaux tels que pytest et flake8 devront également continuer à être utilisables.

2. Réduction de divers problèmes sur le projet

Nous avons plusieurs problèmes à résoudre dans l'application :

- Le linting signale de nombreuses erreurs qu'il faut donc corriger. Merci de ne pas modifier la configuration du linting dans setup.cfg et de conserver les commentaires ;
- La section d'administration du site contient une erreur de pluralisation pour le terme "adresse", qui est affiché comme "Addresss". Nous devons nous assurer que la pluralisation est correcte sur l'ensemble du site ;
- Il est important de prendre en compte la gestion des erreurs 404 et 500 sur notre site web. Actuellement, ces pages d'erreur sont génériques et peu utiles pour l'utilisateur final ;
- Il est crucial de documenter le code en utilisant des docstrings claires et concises afin de faciliter la compréhension du fonctionnement du code par les développeurs. Pour assurer une documentation complète, nous devons inclure des docstrings sur chaque module, classe et fonction ;
- Nous devons maintenant mettre en place des tests unitaires et d'intégrations pour les vues, les URL et les modèles afin de garantir la qualité et le bon fonctionnement de notre application. Il est important que chaque test soit placé dans l'application appropriée pour assurer une structure et une organisation cohérentes. N'oubliez pas de vérifier que la couverture de test est supérieure à 80 %.

3. Surveillance de l'application et suivi des erreurs via Sentry

Dans le cadre de l'optimisation de notre application, nous souhaitons mettre en place un système de surveillance et de gestion des erreurs plus robuste. Nous avons choisi d'utiliser Sentry comme outil de surveillance des erreurs. Une fois Sentry intégré, vous devrez compléter la gestion des erreurs en insérant des logs appropriés dans le code,

en utilisant le module de logging. Ces logs doivent être placés aux endroits stratégiques du code, tels que les fonctions critiques, les blocs try/except et les points de validation des données.

Voici les étapes à suivre :

1. Installation de Sentry ;
2. Configuration de Sentry ;
3. Configuration de logging ;
4. Insertion de logs.

Veuillez ne pas stocker les identifiants Sentry ou toute autre donnée sensible dans le code source (utilisez des variables d'environnement). Cela est valable également pour toutes les données sensibles du site.

Veuillez mettre à jour la documentation sur le déploiement en conséquence (là encore, votre successeur devrait pouvoir mettre en place facilement la journalisation de Sentry décrite ici).

4. Pipeline CI/CD et le déploiement

Le pipeline CI/CD à mettre en place doit être composé :

- d'un travail de compilation et de tests qui reproduit l'environnement de développement local et exécute le linting et la suite de tests et vérifie que la couverture de test est supérieure à 80 % ;
- d'un travail de conteneurisation qui :
    - construit une image du site pour Docker et la pousse vers le registre des conteneurs du Docker Hub,
    - tague les images avec un label distinct (par exemple, le "hash" de commit),
    - permet de récupérer l'image du registre sur votre machine locale et de lancer le site localement en utilisant l'image (uniquement avec Docker),
    - permet de faire ce qui précède par le biais d'une commande unique ;
- d'un travail de déploiement pour mise en production qui met en service le site en utilisant un hébergeur que vous devrez choisir, comme Render, AWS webapp, Azure, etc.
    - Configurez le déploiement de manière à ce que seules les modifications apportées à la branche master dans GitHub déclenchent la conteneurisation et le déploiement du site,
        - La tâche de conteneurisation ne doit être exécutée que si la compilation et la suite de tests sont réussies,
        - Le travail de déploiement et de production ne doit s'exécuter que si le travail de conteneurisation est réussi,
    - Les modifications apportées aux autres branches doivent uniquement déclencher la compilation et les tests (sans déployer le site ou effectuer la conteneurisation).

Il s'agit d'un site destiné aux consommateurs, il est donc important que l'apparence du site soit correcte. Il faut donc bien vérifier le chargement des fichiers statiques après le déploiement. De plus, le CTO est un utilisateur fréquent de l'interface admin, l'apparence est importante également. Le déploiement doit correspondre à ce que nous voyons localement.

Ajoutez également une section "Déploiement" au README, expliquant brièvement :

- un récapitulatif haut niveau du fonctionnement du déploiement ;
- la configuration requise pour que le déploiement fonctionne correctement ;
- les étapes nécessaires pour effectuer le déploiement (votre successeur doit être capable de suivre vos instructions et de faire le travail sans problème, sans avoir à passer du temps à rechercher le problème/la solution lui-même).

5. Documentation de l'application

Dans le cadre du projet, nous allons créer une documentation de spécification technique. Cette documentation servira à décrire respectivement les aspects techniques du projet.

La documentation doit comporter :

- la description du projet ;
- les instructions sur l'installation du projet ;
- un guide de démarrage rapide ;
- les technologies et les langages de programmation à utiliser ;
- une description de la structure de la base de données et des modèles de données ;
- une description des interfaces de programmation ;
- un guide d'utilisation (avec des cas d'utilisation) ;
- les procédures de déploiement et de gestion de l'application.

La création de cette documentation contribuera à une meilleure organisation et compréhension du projet, favorisant ainsi une collaboration plus efficace et un développement plus fluide.