Guide d’étapes clés : Mettez à l'échelle une application Django en utilisant une architecture modulaire

Comment utiliser ce document ?

Ce guide vous propose un découpage du projet en étapes. Vous pouvez suivre ces étapes selon vos besoins. Dans chacune, vous trouverez :

- des recommandations pour compléter la mission ;
- les points de vigilance à garder en tête ;
- une estimation de votre avancement sur l’ensemble du projet (attention, celui-ci peut varier d’un apprenant à l’autre).

Suivre ce guide vous permettra ainsi :

- d’organiser votre temps ;
- de gagner en autonomie ;
- d’utiliser les cours et ressources de façon efficace ;
- de mobiliser une méthodologie professionnelle que vous pourrez réutiliser.

Gardez en tête que votre progression sur les étapes n’est qu’une estimation, et sera différente selon votre vitesse de progression.

Recommandations générales

Il est recommandé de faire les 5 étapes dans l’ordre.

Étape 1 : Améliorez l’architecture modulaire

20 % de progression

Avant de démarrer cette étape, je dois avoir :

- forké et cloné l’application sur mon ordinateur ;

- lancé et testé l'application en local ;
- consulté des ressources pour savoir comment modifier les fichiers de migrations pour copier des données vers de nouvelles tables, si besoin.

Une fois cette étape terminée, je devrais avoir :
- 3 applications : oc_letting_site, profile et letting ;
- implémenté tous les points d'amélioration de l'architecture modulaire.

Recommandations :
- Suivez les indications dans les Aspects techniques du document Site web 2.0 - caractéristiques et améliorations

Points de vigilance :
- Veuillez être prudent pour suivre le bon ordre de déplacement, de mise à jour et de suppression des éléments du site ;
- Ne pas utiliser du SQL directement dans le fichier de migration à l'aide de la méthode RunSQL().

Ressources :
- Le tutoriel How to Move a Django Model to Another App de realpython.com (rédigé en anglais).

Étape 2 : Réduisez les divers problèmes sur le projet

45 % de progression

Avant de démarrer cette étape, je dois avoir :
- une application fonctionnelle ;
- accès à la partie admin de l'application.

Une fois cette étape terminée, je devrais avoir :
- flake8 qui ne renvoie aucune erreur ;
- la correcte pluralisation des modèles dans la page admin de l'application ;
- une page personnalisée en cas d'erreur 404 ou 500 ;
- une docstring sur chaque module, classe et fonction ;
- une couverture de test supérieure à 80 %.

Recommandations :

- Les docstrings devraient expliquer la fonctionnalité, les paramètres, la valeur de retour et tout autre détail important du code ;
- Organisez les tests de manière à ce qu'ils soient regroupés avec l'application respective pour faciliter la localisation et la gestion des tests ;

- Assurez-vous que la couverture de test est supérieure à 80 % en utilisant un outil de vérification de couverture tel que "coverage" ou "pytest-cov".

Point de vigilance :
Lorsque vous évaluez le taux de couverture, veillez à exclure les fichiers de test, car ils affichent automatiquement une couverture de 100 %.

Ressources :
- Les chapitres Implémentez vos tests pour framework Django avec pytest-django et Mesurez votre couverture de test du cours Testez votre projet Python ;
- Le tutoriel Python Docstrings de programiz.com (rédigé en anglais).

Étape 3 : Surveillez l'application et ses erreurs via Sentry

65 % de progression

Avant de démarrer cette étape, je dois avoir :
- une application fonctionnelle sans bug.

Une fois cette étape terminée, je devrais avoir :
- les erreurs et les logs de l'application qui remontent sur Sentry.

Recommandations :
- Lors de la configuration de Sentry, n'oubliez pas de récupérer la clé d'API nécessaire pour l'intégration et pour l'ajouter aux fichiers de configuration de l'application ;
- Pensez à définir les niveaux de log appropriés pour les différentes parties de l'application ;
- Identifiez les points critiques de l'application où les logs doivent être insérés, tels que les fonctions sensibles, les blocs try/except ;
- Communiquez des informations pertinentes dans les logs, telles que les messages d'erreur, les variables importantes, etc. ;
- Vérifiez le fonctionnement de vos logs en provoquant des erreurs. Sont-elles correctement capturées et enregistrées ?

Ressources :
- La documentation d'installation avec Django de Sentry (rédigé en anglais) ;
- La documentation de logging avec Python de Sentry (rédigé en anglais) ;
- La documentation de journalisation de Django.

Étape 4 : Mettez en place le pipeline CI/CD et le déploiement

80 % de progression

Avant de démarrer cette étape, je dois avoir :

- lu la documentation sur circle CI ;
- lu la documentation sur Docker ;
- fait l'ensemble des tests de l'application ;
- une couverture de test supérieure à 80 %.

Une fois cette étape terminée, je devrais avoir :

- la CI qui se lance automatiquement à chaque commit ;
- une image Docker qui tourne en local ;
- une image pushée dans le Docker hub ;
- une application déployée et accessible avec une URL publique.

Recommandations :

- Utilisez le template fourni par CircleCI pour lancer la CI avec la vérification sur les tests ;
- Pensez à merger la branche créée automatiquement par CircleCI sur les autres branches ;
- N'oubliez pas de placer les valeurs sensibles en variable d'environnement, car le fichier de configuration "config.yml" est public ;
- Veillez à vérifier que le conteneur fonctionne correctement en local avant d'essayer de le déployer.

Points de vigilance :

- Ne pas oublier de configurer l'application Django pour la production ;
- Les fichiers statiques doivent correctement se charger en production ;
- L'image pushée dans Docker hub doit être clonée en local et se lancer sans bug ;
- Le déploiement doit être répétable ;
- Si vous utilisez Render comme solution de déploiement, veillez à désactiver le déploiement automatique à chaque commit.

Ressources :

- Le cours Mettez en place l'intégration et la livraison continues avec la démarche DevOps ;
- Le cours Optimisez votre déploiement en créant des conteneurs avec Docker ;
- Le tutoriel How to Create Django Docker Images de Section (rédigé en anglais) ;

- Le tutoriel How to Deploy Docker Container on Heroku? de Featurepreneur (rédigé en anglais) ;
- Le tutoriel Deploying Docker Images to Heroku With Circle CI de Nexton (rédigé en anglais).

Étape 5 : Documentez l'application

100 % de progression

Avant de démarrer cette étape, je dois avoir :

- terminé le déploiement de l'application.

Une fois cette étape terminée, je devrais avoir :

- la documentation du projet publiée sur Read The Docs.

Recommandations :

- Suivez le tutoriel fourni dans les ressources pour effectuer la configuration de la documentation sur Read The Docs ;
- Créez la structure et le contenu du document en amont avant de les retranscrire dans le projet ;
- Vérifiez que les sections et les informations sont bien organisées pour une navigation facile ;
- La documentation doit être claire, facile à comprendre et concise, il faut éviter les gros paragraphes.

Points de vigilance :

- Vérifiez que la documentation se met à jour automatiquement à chaque modification.

Ressources :

- Le tutoriel de Read The Docs ;
- Introduction au reStructuredText dans la documentation officielle de Sphinx.

Projet terminé !