Guide d’étapes clés : Développez une architecture back-end sécurisée en utilisant Django ORM

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 qu’elle sera différente selon votre vitesse de progression.

Recommandations générales

Ce projet se fait en trois temps :

1. Une partie d’initialisation et de définition (étapes 1 à 2) ;
2. Une partie de développement (étapes 3 à 7) ;
3. Une partie de finalisation (étapes 8 à 9).

Évitez de commencer les développements sans la partie de définition ; cela vous fera plus perdre du temps qu’autre chose. Bien que la partie développement soit la plus longue, veuillez ne pas négliger les parties de définition (avant le développement) et de journalisation et documentation (après le développement).

Étape 1 : Mettez en place de l'environnement SQL et Python

5 % de progression

Avant de démarrer cette étape, je dois avoir :

- lu le projet entier, incluant le cahier des charges.

Une fois cette étape terminée, je devrais avoir :

- mis en place une base de données fonctionnelle et accessible.

Recommandations :

- Choisir un moteur de base de données (SQLite, MySQL, PostgreSQL) ;
- Installer les outils d'administration associés (PgAdmin, MySQL Workbench, etc.) ;
- Créer un compte utilisateur pour l'application ;
- Créer une base de données pour l'application ;
- S'assurer que cet utilisateur dispose des droits nécessaires sur la base de données ;
- Installer une librairie ORM en Python (par exemple SQL Alchemy) ;
- Se familiariser avec son utilisation.

Points de vigilance :

- Avant de démarrer le projet, familiarisez-vous avec ce qu'est un système Customer Relationship Management (CRM) – l'application que vous développerez ;
- N'oubliez pas d'utiliser un compte non privilégié pour la base de données ;
- Évitez de renseigner les informations de connexion à la base de données (y compris le mot de passe) en dur dans le code.

Ressources :

- Pour mettre en place votre base de données, suivez ce chapitre Créez votre base de données (BDD) du cours OpenClassrooms Implémentez vos bases de données relationnelles avec SQL ;
- Tutoriel SQLAlchemy (en anglais) ;
- Pour mieux comprendre le contexte d'un CRM, suivez ce cours OpenClassrooms : Utilisez un CRM dans votre activité de commercial.

Étape 2 : Définissez les modèles et implémentation dans la base

15 % de progression

Avant de démarrer cette étape, je dois avoir :

- initialisé la base de données.

Une fois cette étape terminée, je devrais avoir :

- défini les classes Python des entités métiers
  - pour les clients,
  - les contrats,
  - et les événements ;
- modélisé les classes.

Recommandations :

- Un diagramme entity-relationship (ERD) peut être utile pour mieux visualiser les relations entre les différents modèles

Points de vigilance :

- Attention à définir les relations entre les différents modèles ;
- Soyez conscient qu'il sera peut-être difficile d'implémenter les relations dans l'ORM.

Ressources :

- Le cours OpenClassrooms Modélisez vos bases de données ;
- Si vous voulez utiliser un diagramme UML, vous pouvez suivre le chapitre Définissez un glossaire commun à partir d'un diagramme de classes dans le cours OpenClassrooms Appliquez le principe du Domain-Driven Design à votre application ;
- Tutoriel : diagramme de classes en UML (vidéo).

Partie de développement

Après les étapes précédentes, vous aurez déjà initialisé la BDD et défini les modèles, c'est-à-dire que vous serez prêt à coder l'application. Nous avons découpé les tâches de développement de l'application dans les étapes suivantes en procédant d'abord au code des utilisateurs et de

l'authentification, puis au code des opérations CRUD, et enfin au code de l'interface.

Toutefois, il est possible de suivre un ordre différent, mais veuillez garder les étapes distinctes pour respecter la séparation des principes. Si vous voulez suivre un autre ordre, validez-le avec votre mentor.

Par ailleurs, nous vous conseillons de suivre cette étape parallèle ci-dessous tout au long de la partie de développement.

Étape parallèle : Écrivez du code propre

Avant de démarrer cette étape, je dois avoir :

- défini les classes Python.

Une fois cette étape terminée, je devrais avoir :

- une application écrite du code propre qui fonctionne bien.

Recommandations :

- L'application va comporter de nombreux éléments interdépendants. Pour faciliter le processus de développement et rendre la maintenance du code plus facile sur le long terme, il faudra faire des choix sur l'architecture de l'application ;
- Il existe une collection de « bonnes solutions » aux problèmes courants, typiquement regroupées sous le nom de *design patterns*. Vous allez très certainement en utiliser plusieurs dans ce projet. Soyez sûr de vous familiariser avec les plus classiques.
    - Repository, DAO, Active Record… pour l'interface entre la base de données et la logique métier (*data access layer*) ;
    - MVC sera par exemple très utile pour « connecter » l'interface utilisateur à la logique métier.
- Le code devra bien sûr suivre les bonnes pratiques de développement :
    - formatage et PEP8 (black, autopep8, flake8…) ;
    - le code est couvert par des tests (unitaires et intégration). N'oubliez pas la couverture du code !
    - le code est documenté et bien organisé ;
    - tout au long du projet, gardez en tête que la **sécurité** de la plateforme est la principale préoccupation. Cela se traduira également dans le code !

Points de vigilance :

- Certains concepts « objet » en Python sont différents d'autres langages, comme Java ;
- Les design patterns sont des guides vers une solution, et non LA solution ;
- Il n'y a pas de design pattern miracle : choisissez celui qui vous convient le mieux !

Ressources:

- Design patterns / Database Access Layer ;
- Si vous utilisez l'architecture MVC dans votre projet, vous pouvez suivre le cours OpenClassrooms Écrivez du code Python maintenable, particulièrement les chapitres suivants :
  - Implémentez le modèle pour votre application ;
  - Implémentez le contrôleur et la vue pour votre application.

Étape 3 : Créez les comptes utilisateurs et permissions

25 % de progression

Avant de démarrer cette étape, je dois avoir :

- défini les classes Python.

Une fois cette étape terminée, je devrais avoir :

- codé les classes et les fonctions Python pour l'identification des utilisateurs.

Recommandations :

- Réfléchir au stockage des mots de passe : ils doivent être salés et hachés ;
- Il est possible d'utiliser des librairies tierces (par exemple *bcrypt* ou *argon2*) ;
- Tous les collaborateurs possèdent des éléments d'identification pertinents :
  - un numéro d'employé ;
  - un nom ;
  - une adresse email ;
  - une affiliation à un département ;
  - des permissions différentes suivant leur département.

Points de vigilance :

- Respectez des bonnes pratiques de sécurité pour le stockage d'informations sensibles dans la base de données (obfuscation du mot de passe, salage, etc.) ;
- Comprenez la différence entre autorisation et authentification ;
- Évitez l'implémentation des rôles « en dur » dans la table des comptes utilisateur au lieu de créer une relation vers un « rôle ».

Ressources :

- Article sur le hachage des mots de passe (en anglais) ;
- Argon2 : exemple d'utilisation (en anglais).

Étape 4 : Développez le code permettant l'authentification et l'autorisation des utilisateurs

35 % de progression

Avant de démarrer cette étape, je dois avoir :

- créé les classes des utilisateurs.

Une fois cette étape terminée, je devrais avoir :

- une fonction d'authentification permettant une authentification persistante des utilisateurs ;
- une fonction d'autorisation permettant de vérifier le niveau de permissions de l'utilisateur en cours.

Recommandations :

- Il est possible d'utiliser des jetons JSON Web Token (JWT ; par exemple avec la librairie *pyjwt*). Faites attention au stockage du secret/à l'algorithme utilisé !
- Après l'authentification, l'utilisateur reçoit un jeton qui lui permet d'accéder aux fonctionnalités auxquelles il a droit :
    - le jeton est stocké sur la machine et permet d'autoriser les actions dans l'application ;
    - par exemple, l'utilisateur pourrait s'authentifier en utilisant la commande : "python epicevents.py login".

Points de vigilance :

- La réalisation d'une authentification dans le terminal peut être une tâche ardue ;
- Utilisez le fichier .netrc / _netrc peut être trop compliqué pour rendre l'application compatible avec d'autres plateformes : vous pouvez utiliser votre propre méthode (fichier dédié, variable d'environnement, argument en ligne de commande, etc.).
- Les jetons ont une date d'expiration. Comment allez-vous gérer un jeton expiré ?

Ressources :

- Comment utiliser JWT en Python (en anglais) ;
- Authentification vs autorisation (en anglais) ;
- Building an authenticated CLI (en anglais) ;
- Référez-vous au cours OpenClassrooms Sécurisez vos applications web avec l'OWASP pour vous rappeler les bonnes pratiques de sécurisation.

Étape 5 : Développez les premiers éléments de code permettant la lecture de données dans la base

40 % de progression

Avant de démarrer cette étape, je dois avoir :

- l'identification et l'authentification des utilisateurs.

Une fois cette étape terminée, je devrais avoir :

- codé les classes et les fonctions Python permettant la lecture de données depuis la base.

Recommandations :

- Le code doit permettre, *au minimum*, d'obtenir tous les instances des entités métiers ;
- Il doit permettre :
    - d'obtenir tous les clients,
    - d'obtenir tous les contrats,
    - d'obtenir tous les événements.

Points de vigilance :

- Vérifiez les permissions avant de récupérer les données ;
- Assurez-vous que les utilisateurs doivent être authentifiés pour accéder aux données ;
- Ajoutez les relations entre les modèles « utilisateur » et les modèles métier.
    - Les modèles « utilisateur » sont 'employé' et les modèles métier sont 'contrats', 'événements', ou 'clients'.

Ressources :

- Le cours OpenClassrooms Implémentez vos bases de données relationnelles avec SQL, notamment le chapitre Sélectionnez les données présentes dans votre BDD ;
- Aide-mémoire SQLAlchemy (en anglais).

Étape 6 : Développez le code permettant la création et la mise à jour de données dans la base

50 % de progression

Avant de démarrer cette étape, je dois avoir :

- une application qui permet la lecture des données

Une fois cette étape terminée, je devrais avoir :

- codé les classes et les fonctions Python permettant la création et la mise à jour d'informations dans la base de données.

Recommandations :

- Le code doit permettre, *au minimum*, de :
    - créer un collaborateur ;
    - modifier un collaborateur (y compris son département) ;
    - créer un contrat ;
    - modifier un contrat (tous les champs, y compris relationnels) ;
    - créer un événement ;
    - modifier un événement (tous les champs, y compris relationnels) ;
    - les fonctionnalités ci-dessous doivent uniquement être accessibles aux utilisateurs qui y ont effectivement accès.

Points de vigilance :

- Validez les données dans le code avant l'écriture dans la base ;
- Vérifiez les permissions/rôles de l'utilisateur en cours avant la modification des données.

Ressources :

- Le cours OpenClassrooms Implémentez vos bases de données relationnelles avec SQL, notamment les chapitres Insérez des données dans votre BDD et Mettez à jour les données de votre BDD ;
- Aide-mémoire SQLAlchemy (en anglais).

Étape 7 : Développez l'interface en ligne de commande

80 % de progression

Avant de démarrer cette étape, je dois avoir :

- une application qui permet la création et la mise à jour des données.

Une fois cette étape terminée, je devrais avoir :

- codé l'interface utilisateur.

Recommandation :

- Utilisez des librairies tierces pour faciliter l'implémentation du programme (par exemple: *click* ou *rich*).

Points de vigilance :

- Il faut toujours valider les données utilisateur avant leur utilisation ;
- Il faut réutiliser (et si besoin, améliorer) le code développé à l'étape précédente pour assurer une bonne séparation des responsabilités (affichage vs manipulation des données).

Ressources :

- Rich sur GitHub ;
- Tutoriel pour rich (en anglais) ;
- Tutoriel pour click (en anglais) ;
- Tutoriel : développez une application sécurisée en ligne de commande avec rich et click (en anglais).

Partie de la finalisation

À ce point, l'application marche déjà. Elle permet la manipulation de données (les opérations CRUD) sécurisées par un interface. Pour finaliser votre projet, vous ajoutez la journalisation et vous documentez votre code avant de le publier.

Étape 8 : Ajoutez la journalisation avec Sentry.io

90 % de progression

Avant de démarrer cette étape, je dois avoir :

- une application qui permet à l'utilisateur d'effectuer les opérations CRUD.

Une fois cette étape terminée, je devrais avoir :

- ajouté la journalisation avec Sentry à l'application.

Recommandations :

- Lire la documentation de Sentry pour Python ;
- Assurez-vous que les situations suivantes sont journalisées :
    - toutes les exceptions inattendues,
    - chaque création/modification d'un collaborateur,
    - la signature d'un contrat (pour l'option scénarisée).

Points de vigilance :

- Soyez conscient de la difficulté à mettre en place le kit Sentry ;
- Soyez attentif aux problèmes de sécurité avec les clés Sentry (par exemple : publication sur GitHub, ou en dur dans le code).

Ressources :

- La documentation officielle de Sentry pour Python.

Étape 9 : Documentez et publiez sur GitHub

100 % de progression

Avant de démarrer cette étape, je dois avoir :

- une application qui effectue toutes les fonctionnalités attendues.

Une fois cette étape terminée, je devrais avoir :

- le repository GitHub de mon projet bien documenté et publié.

Recommandations :

- Vérifiez qu'aucune donnée sensible n'est exposée publiquement (secret, clé privée, clés Sentry, informations de connexion à la base de données, etc.) ;
- Assurez-vous que le repository est complet et permet le déploiement de l'application (fichier requirements, instructions de mise en place de la base de données, etc.) ;
- Prévoyez un script Python (ou un fichier SQL) permettant la création des tables initiales.

Points de vigilance :

- Ne livrez jamais du code mal ou non documenté ;
- Assurez-vous que le contenu du fichier README est clair et suffisant.

Ressources :

- Le cours OpenClassrooms Devenez un expert de Git et GitHub sert de point de référence pour les bonnes pratiques.

Projet terminé !