Rapport individuel
SAE R 3.01c
Ilane Riotte

Synthèse :

MobiList vise à développer une application web pour la gestion d'inventaires de biens mobiliers et d'assurance.
Durant cette semaine de projet, notre groupe s'est concentré sur le développement du site et de la majorité des fonctionnalités. Mon rôle a été principalement technique, avec dans l'ensemble la stabilisation des différentes page et structures, la correction de nombreux bugs et ajout de quelques fonctionnalités.

Mes contributions principales ont été :
- Correction et amélioration des vues login et changer_mot_de_passe, incluant la gestion du hachage et la correction du système de profils (User/Assure).
- J'ai développé la logique back-end Python pour le tableau_de_bord (calcul des statistiques) et Python et Flask pour la vue ajouter_logement.
- J'ai eu un rôle majeur dans la correction de la commande flask loaddb [] pour qu'elle remplisse correctement les tables de la base de données, notamment les relations de la table possède.
- J'ai aidé mon groupe sur certains problèmes de conception et apporté des petites améliorations visuelles sur certaines pages après leur développement (CSS, JavaScript pour l'icône "œil").

Analyse :
Le plus gros problème que j'ai rencontré a été de trouver et corriger les erreurs de liaison entre la base de données et les vues, notamment la fonctionnalité de connexion et de chargement des données.
- La commande flask loaddb [] échouait en provoquant des erreurs lors du remplissage des tables. De plus, la connexion utilisateur était non fonctionnelle car le système ne faisait pas le lien entre l'objet User (pour les connexions) et l'objet Assure (pour les informations clientes), provoquant par exemple des erreurs d'attributs inexistants (comme l'utilisation de l'attribut logement de User qui n'existait tout simplement pas dans la base de donnée) dans certaines pages.

Démarche et Solution :
- J'ai compris que nous ne pouvions pas mélanger l'insertion SQL manuelle (db.session.execute(possede.insert()...) avec l'ORM. J'ai réécrit le script pour qu'il utilise la méthode ORM (assure.logements.append(logement)), ce qui a résolu certains conflits de relation.
- J'ai redéfini le "Profile Pattern" (pour la connexion). Le LoginForm n'authentifie plus que le User. Le user_loader ne charge que le User. Ensuite, dans les vues (comme tableau_de_bord), j'accède aux données de l'assuré via la relation current_user.assure_profile qui permet de récupérer les informations du client via l'objet Assure.
Ces corrections on permis une stabilisation de la base de données pour que tout notre groupe puisse l'utiliser efficacement lors du développement de leurs vues.

Démonstration de compétence :

AC21.01 Élaborer et implémenter les spécifications fonctionnelles :
J'ai transformer les fonctionnalités "connexion" et "changement de mot de passe" en un code fonctionnel, gérant le hachage (sha256) et la mise à jour des mots de passe en base de données, ainsi que certaines autre fonctionnalités comme la création d'un logement....

AC21.03 Adopter de bonnes pratiques de conception et de programmation :
J'ai séparé l'authentification (modèle User) des données clientes (modèle Assure), améliorant la clarté du code.
J'ai corrigé la commande loaddb pour qu'elle respecte la convention ORM au lieu d'écrire du SQL brut, améliorant la cohérence des données.

AC22.01 Choisir des structures de données complexes adaptées au problème :
J'ai aidé à implémenter et corriger les relations ORM complexes : la relation 1-1 (User → Assure) et la relation N-N (Assure ↔ Logement via la table possede), ainsi que la gestion des données utilisateurs pour certaines vues telles que la connexion, le tableau de bord, les création des logements, etc..

AC23.01 Concevoir et développer des applications communicantes :
J'ai développé plusieurs vues Flask gérant les requêtes GET (afficher le formulaire ajouter_logement) et POST (traiter les données, créer l'objet Logement, et le lier au current_user.assure_profile).

AC24.04 Manipuler des données hétérogènes :
Dans la vue tableau_de_bord, j'ai développé la logique Python pour agréger des données, j’ai également manipulé les données lors de la création des logements, la modifications de mot de passe, les connexions utilisateurs, etc..

AC26.03 Mobiliser les compétences interpersonnelles pour travailler dans une équipe :
J'ai aidé mon groupe en corrigeant certaines partie de leur code (ex: problèmes dans les formulaires, erreurs de logique dans les vues) et en partageant les potentiels problèmes de conceptions notamment de la base de données et des templates.