AOO Python – Projet

Consignes générales

Le projet est à effectuer par groupe de deux personnes.

Rendus

- Un dépôt GitHub contenant :
  - Un fichier README.md avec les informations générales sur votre projet : noms des contributeurs, description, structure générale du projet, comment exécuter votre projet. N'hésitez pas à inclure des images, schéma etc...
  - Le code source dans un dossier « src »
  - Les tests unitaires dans un dossier « tests »

- Un rapport (un seul par groupe, et au format PDF) dans lequel figurera :
  - Le résumé de votre démarche ainsi que la répartition des tâches
  - Une explication générale du fonctionnement de votre application
  - Les problèmes rencontrés et ce que vous avez fait pour les résoudre
  - Les différentes ressources utilisées pour mener à bien le projet

- A rendre individuellement en plus : votre impression sur la bonne répartition de la charge de travail au sein des membres du groupe.
  Google Forms à envoyer : https://forms.gle/KCsk9Ngr6h7nhqZN6

Tous les rendus se feront par email. Une personne du groupe sera chargée d'envoyer les rendus aux adresses suivantes : loic.riegel@uha.fr et francois.ludwinski@uha.fr.

Indiquez clairement le nom de tous les membres du groupe dans l'email et dans le rapport.

Date de rendu : dimanche 1 juin 2025, à 23h59

Travail demandé

Contexte

L'entreprise MeetingPro est leader dans la location d'espace de coworking.

MeetingPro met à disposition plusieurs types de salles à louer pour ses clients : des salles standard, des salles de conférence et des salles informatiques.

Elle souhaite disposer d'un système informatique permettant de gérer la réservation des salles pour ses clients.

Votre manager vous a chargé de concevoir une application avec une interface graphique en Python pour répondre au besoin de l'entreprise MeetingPro.

Besoin de MeetingPro

Voici le compte rendu de l'interview client passée par votre business manager :

« Nous avons chez MeetingPro plusieurs types de salles :

- Standard : une salle de réunion classique avec une capacité d'accueil variable (1 personne à 4 personnes).

- Conférence : une salle de réunion avec un estrade de présentation, un vidéo projecteur et une capacité d'accueil entre 4 et 10 personnes.

- Informatique : une salle de réunion avec un ordinateur par place. Capacité d'accueil de 1 personne à 4 personne.

Pour identifier un client nous avons besoin de son nom et de son adresse email.

On lui attribue aussi un identifiant unique qui sert lors de la facturation.

Un créneau de réservation est caractérisé par une date de début et une date de fin. Un créneau peut être aussi long que nécessaire.

Sa durée minimum est 30 min.

Nous souhaiterions que le programme s'utilise avec une interface graphique simple.

Voici les fonctionnalités principales attendues :

- Ajouter un nouveau client
- Ajouter une nouvelle salle
- Afficher les salles réservables
- Afficher les réservations pour un client
- Identifier si une salle est disponible sur un créneau
- Afficher les salles disponibles pour un créneau
- Réserver une salle

Pour réserver une salle, le parcours client est le suivant :

- Le client choisi son créneau
- Les salles disponibles lui sont proposées
- Le client choisi sa salle
- La réservation est effectuée

Le programme doit avoir un menu principal comportant les onglets suivants :

1. Ajouter un nouveau client
2. Ajouter une nouvelle salle
3. Afficher les salles réservables
4. Afficher les réservations pour un client
5. Identifier si une salle est disponible sur un créneau
6. Afficher les salles disponibles pour un créneau
7. Réserver une salle

Pour la base de données, un simple fichier au format JSON pour rendre les données persistantes sera suffisant. »

Conseil de résolution

- Analyser le besoin pour identifier les différentes parties logicielles :
  - Modélisation des données (clients, salles, réservations ...)
  - Gestion de la base de données (fichier JSON)
  - Interface graphique

Des schémas d'architecture, de classes, de séquences, maquette de l'interface graphique pourront être très utiles pour la suite du développement du projet et illustrer votre rapport de projet.

- Rendre le code modulaire, séparer le programme en différentes parties pour se partager le travail :
  - On pourra utiliser ou s'inspirer du pattern « Model, View, Controller (MVC) ».
  - Gestion de la base de données (lire les données du fichier JSON au lancement de l'application, et écrire dans ce même fichier avec les nouvelles données avant de terminer le programme). On pourra utiliser les fonction json.load et json.dump.

- Développer de manière itérative. Modélisez, implémentez et testez une petite partie du système et assurez-vous que cette partie fonctionne bien. Faites un commit. Ensuite seulement, ajoutez une nouvelle fonctionnalité et assurez-vous que cette nouvelle fonctionnalité fonctionne bien, et que les fonctionnalités précédentes continuent à bien fonctionner. Refaites un commit, et recommencez, jusqu'à avoir toutes les fonctionnalités.

- Préparer des questions pour éclaircir des zones du besoin qui vous semblent obscures. Votre Buisness Manager tentera d'obtenir des informations auprès du client.

Ressources

Packages Python :

- json
- uuid
- datetime
- tkinter

Le pattern « Model View Controller » :

- https://www.youtube.com/watch?v=ihtIcGkTFBU
- https://www.youtube.com/watch?v=DUg2SWWK18I

Critères d'évaluation

- Votre application fonctionne et respecte le besoin formuler par le client.
- Le projet est bien structuré.
- Le projet est bien documenté.
- Le code est testé et tous les tests passent. On ne demande pas 100% de couverture. Ecrivez des tests qui ont du sens et pertinents pour vérifier le bon fonctionnement de votre application. Toute la partie du code en lien avec l'interface graphique n'a pas besoin de tests unitaires.
- La code source respecte les règles de bonnes pratiques vues en cours : conventions PEP8, formatage, annotations de types, etc... Séparez votre code source en plusieurs modules.
- L'utilisation des principes de la programmation orienté objet dans le code source. Attention, cela ne veut pas dire que tout doit devenir une classe ! Il vaut toujours mieux privilégier la simplicité.
- L'utilisation de Git et GitHub dont vous aurez fait preuve pour collaborer à plusieurs. Chacun devra toujours travailler sur une branche, puis ouvrir une Pull Request. Demandez ensuite aux autres membres du projet de relire votre contribution, de suggérer des améliorations ou de valider. Mergez ensuite sur la branche principale.
- Bonus : on pourra utiliser un linter et un formatter comme « ruff ».

La note sera normalement la même pour tous les membres du projet, sauf si une grande différence de contribution est mise en évidence. Cela pourra être fait en regardant l'historique des activités sur les dépôts GitHub ainsi que grâce aux retours individuels.