Reyane Redjem

Rapport Projet Planning Poker

Présentation globale du projet :

L'objectif de ce projet est de développer une application de planning poker en utilisant Python. Le planning poker est une technique utilisée dans le domaine de la gestion de projet agile pour estimer l'effort requis pour chaque tâche. L'application permettra à un groupe de joueurs de voter sur la complexité des fonctionnalités (backlog) à implémenter. Étant donné que je travaille en solo suite à l'abandon de mon collègue, le projet sera orienté vers une utilisation en local pour simplifier le développement et pour éviter la complexité liée à la gestion des connexions distantes. Trois versions de planning poker seront mises en place, chacune avec des règles différentes (stricte, moyenne, majorité relative), offrant ainsi une flexibilité dans le processus d'estimation.

Choix d'architecture :

J'ai choisi Python pour sa simplicité et sa facilité d'utilisation, des atouts importants pour le développement d'une application légère comme le planning poker.

Dans mon code j'ai des choses qui se rapproche de designs pattern

- La gestion du type de jeu (**strict**, **moyenne**, **majoriteRelative**) dans la fonction '**saveJson**' et son utilisation dans la fonction '**Game**' pour déterminer le comportement en fonction du type de jeu se rapproche un peu du modèle de conception de stratégie. Chaque type de jeu peut être considéré comme une stratégie différente.
- Dans la boucle principale du jeu, il y a une boucle infinie qui réagit aux événements. Les parties du code qui réagissent aux événements (par exemple, les clics de souris) pourraient être considérées comme des modèle de conceptions d'observateurs qui réagissent aux changements d'état.
- L'utilisation de la fonction '**get_font**' pour obtenir une police avec une taille spécifique s'approche un peu du modèle de conception de fabrique. Cela ressemble à une méthode de création centralisée pour obtenir des objets de police.

Intégration continue :

J'ai configuré Doxygen pour générer automatiquement la documentation à partir des commentaires dans le code. Cette documentation automatique offre une vue claire de la

structure du code, des classes, et des méthodes, facilitant la compréhension et la maintenance du projet.

En parallèle, j'ai mis en place Pytest, un framework de test pour Python. À chaque modification du code, Pytest est automatiquement déclenché pour exécuter les tests définis, permettant ainsi une détection rapide d'éventuels problèmes. Cette approche garantit la stabilité du code tout au long du développement.