Rapport du Projet Planning Poker

1. Introduction et Contexte

Le projet Planning Poker s'inscrit dans le cadre de la matière : "Conception Agile de Projet Informatiques" du M1 Informatique. L'objectif de ce projet est de concevoir une application qui permet à une équipe de réaliser des sessions de Planning Poker afin d'estimer la complexité de fonctionnalités d'un backlog, comme vu en cours.

L'application développée permet de gérer un backlog de fonctionnalités, d'ajouter des joueurs, de choisir un mode de calcul des votes (dans notre projet strict ou majorité absolue), puis de réaliser une phase de vote à l'aide de cartes (0, 1, 2, 3, 5, 8, 13, 20, 40, 100 et la carte spéciale "café").

Le projet a été réalisé en binôme en s'appuyant sur les principes du pair programming. Cette manière de travailler nous a permis d'échanger régulièrement sur les choix techniques, de relire mutuellement notre code et de corriger rapidement nos erreurs.

Le développement s'est fait de façon progressive, en ajoutant les fonctionnalités une par une. Cette approche nous a permis de tester rapidement chaque étape, d'ajuster certaines décisions en cours de route et d'améliorer progressivement la stabilité de l'application.

2. Choix Techniques et Architecture

2.1 Choix du langage et des bibliothèques

Le langage Python a été choisi pour ce projet en raison de sa simplicité, de sa lisibilité et de sa rapidité de développement. Ces caractéristiques sont particulièrement adaptées à un projet de ce type. De plus, Python est un langage qui nous est enseigné à Lyon 2 toutes les années depuis la L1. Il est donc naturellement celui dans lequel nous avons le plus de facilité et de connaissances. Pour ces 2 raisons, il nous a paru évident que Python était le langage le plus cohérent pour mener à bien ce projet de Planning Poker.

L'interface graphique a été développée à l'aide de la bibliothèque Tkinter, intégrée nativement à Python. Ce choix permet de proposer une interface fonctionnelle sans dépendances externes lourdes, garantissant ainsi une exécution immédiate du projet sur la machine de l'évaluateur. Nous avions quelques bases de la bibliothèque Tkinter auparavant ce qui nous a aussi poussé vers ce choix pour faire l'interface.

L'application utilise une base de données SQLite pour stocker les informations (joueurs, votes, parties), ce qui permet de les retrouver même après la fermeture du

programme. SQLite a été privilégiée car elle ne nécessite pas de serveur dédié, ce qui simplifie le déploiement et respecte la contrainte de portabilité du projet. Elle permet néanmoins une modélisation relationnelle robuste des données du Planning Poker, telles que les parties, les joueurs, le backlog et les votes.

2.2 Architecture logicielle

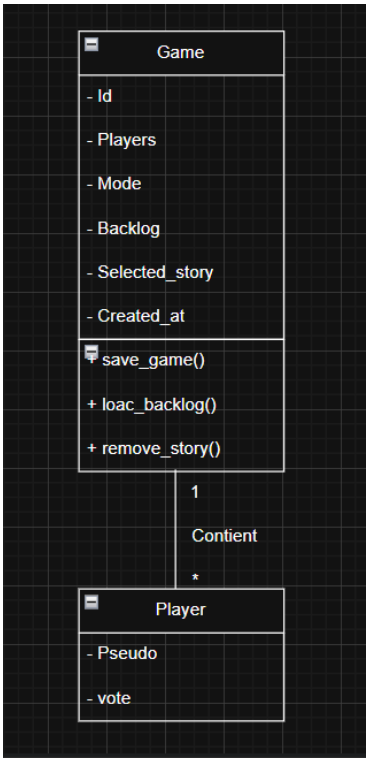
L'application est structurée de manière à séparer les règles de fonctionnement du Planning Poker, la gestion des données et l'interface utilisateur. Même si l'architecture ne suit pas formellement le modèle MVC, elle améliore la lisibilité du code et facilite sa maintenance. Mais en effet l'interface (GameUI) contient encore un peu de logique.

La logique métier est encapsulée dans les classes Game et Player, responsables de la gestion des joueurs, du backlog, des votes et de la sauvegarde des parties. L'interface utilisateur est gérée par la classe GameUI, qui se charge des interactions avec l'utilisateur via Tkinter.

L'accès à la base de données est isolé dans le module db.py, ce qui centralise la gestion des connexions et facilite un passage éventuel à un autre système. Cette séparation rend le code plus simple à maintenir et limite les dépendances entre les différentes parties de l'application.

2.3 Modélisation – Diagrammes

Diagramme de classe :



Base de données :

Game : games table

Player : players table

Votes : game_players table

Backlog : backlog table

Explication du diagramme :

Un diagramme de classes a été réalisé afin de représenter les principales entités du système. La classe Game constitue le cœur de l'application et regroupe les informations liées à une partie de Planning Poker : mode de jeu, backlog, story sélectionnée et joueurs participants.

La classe Player représente un joueur identifié par un pseudo et associé à un vote.Les votes sont enregistrés dans la table game_players de la base de données. La relation entre Game et Player est de type un-à-plusieurs, une partie pouvant contenir plusieurs joueurs.

Le diagramme se concentre volontairement sur les classes principales afin de rester lisible et pertinent, conformément aux recommandations du sujet.

2.4 Gestion des données

Les données du projet sont stockées dans une base SQLite structurée autour de plusieurs tables : backlog, games, players et game_players. Cette organisation permet de représenter correctement les relations entre les parties, les joueurs et leurs votes.

Le backlog est enregistré dans la base de données, ce qui permet de conserver les stories même après la fermeture de l'application. Il peut être enrichi dynamiquement via l'interface graphique. Chaque partie est sauvegardée avec son mode de jeu, la story estimée et les votes associés à chaque joueur.

Des scripts d'initialisation (init_db.sql, init_db.py) et de remplissage (fill_backlog.py) sont fournis afin de garantir une reproductibilité complète du projet lors de son installation.

3. Mise en place de l'Intégration Continue

Pour assurer la qualité et la stabilité du projet, nous avons mis en place l''intégration continue (CI) avec GitHub Actions. Ce pipeline s'exécute automatiquement à chaque push ou pull request sur la branche principale (main).

3.1 Workflow

Le workflow est défini dans le fichier test.yml et se compose des étapes suivantes :

1) Checkout du code : Récupération automatique du code depuis le dépôt GitHub

2) Configuration de l'environnement Python : Installation de Python 3.8 sur une machine virtuelle Ubuntu

3) Installation des dépendances : Mise à jour de pip et installation de pytest pour les tests unitaires

4) Configuration du PYTHONPATH : Permet à Python de reconnaître le répertoire du projet pour l'exécution des tests

5) Exécution des tests : Lancement des tests définis dans test_menu.py et génération d'un rapport HTML

Ce workflow garantit que toute modification du code est vérifiée automatiquement avant d'être intégrée dans la branche principale, ce qui assure une base stable pour le développement.

3.2 Tests Unitaires

Les tests unitaires permettent de vérifier la logique métier, notamment :

- La création et gestion des joueurs (Player) et parties (Game)

- La sélection d'une story et la phase de vote

- Le calcul des résultats en fonction du mode choisi (strict ou majorité relative)

Grâce à ces tests, chaque fonctionnalité importante est validée automatiquement à chaque changement de code.

3.3 Génération de documentation

Pour documenter le projet, nous utilisons Doxygen. La configuration est définie dans le fichier Doxyfile :

- Le code source se situe dans le répertoire src

- La documentation est générée au format HTML dans le dossier documentation

- Les diagrammes UML et les graphes d'appels sont activés pour mieux visualiser les relations entre classes et fonctions

Ce processus de génération de documentation peut également être intégré dans la CI, permettant d'obtenir une documentation toujours à jour à chaque version du projet.

4. Manuel Utilisateur et Fonctionnalités

4.1 Installation et lancement

Pour utiliser le projet, il suffit de cloner le dépôt GitHub et d'installer Python (version 3.8 recommandée). Aucune dépendance externe lourde n'est nécessaire, car l'interface graphique utilise Tkinter, intégré nativement à Python.

Commandes pour lancer le projet :
git clone <URL_DU_DEPOT>
cd <nom_du_dossier>
python backend.py

Les scripts d'initialisation (init_db.py) et de remplissage (fill_backlog.py) doivent être exécutés avant le premier lancement pour créer la base de données et insérer les stories initiales.

4.2 Fonctionnalités principales

L'application propose toutes les fonctionnalités essentielles d'un outil de Planning Poker :

1) Gestion du backlog
    - Chargement automatique des stories depuis la base de données
    - Ajout, suppression et actualisation des stories via l'interface graphique
    - Les modifications sont persistantes et conservées même après la fermeture de l'application
2) Gestion des joueurs
    - Ajout de joueurs avec un pseudo unique
    - Affichage dynamique des joueurs participants à la partie
3) Sélection du mode de calcul
    - Mode strict : le résultat de la story est déterminé uniquement si tous les joueurs sont unanimes
    - Mode majorité relative : le vote le plus fréquent est retenu
4) Sélection de la story à estimer
    - L'utilisateur choisit une story depuis la liste du backlog pour la session de vote
5) Phase de vote

- Chaque joueur choisit une carte parmi : 0, 1, 2, 3, 5, 8, 13, 20, 40, 100, café
- Les votes sont enregistrés dans la base de données et associés aux joueurs et à la partie en cours

6) Résumé et sauvegarde de la partie
    - Affichage d'un résumé complet : joueurs, votes et story sélectionnée
    - Sauvegarde automatique de la partie dans la base SQLite

4.3 Interface utilisateur

L'interface, réalisée avec Tkinter, offre un flux simple et intuitif :

1) Menu principal : affichage du backlog
2) Ajout ou suppression de stories
3) Ajout de joueurs et choix du mode de jeu
4) Sélection de la story à estimer
5) Phase de vote avec saisie des cartes
6) Résumé final et sauvegarde de la partie

L'interface met à jour dynamiquement les informations affichées, permettant à l'utilisateur de suivre facilement le déroulement de la session.

Conclusion

Le projet Planning Poker a permis de mettre en pratique les concepts de Conception Agile. L'application développée offre une interface fonctionnelle pour gérer un backlog, ajouter des joueurs, voter et sauvegarder les parties.

Les choix techniques, notamment Python, Tkinter et SQLite, ont permis un développement rapide, une interface intuitive et une sauvegarde fiable des informations. Le code est structuré pour séparer la logique métier, la gestion des données et l'interface, facilitant ainsi sa maintenance et son évolution future.

Pour aller plus loin, il serait possible d'ajouter de nouveaux modes, une interface web ou une gestion multi-utilisateurs à distance, afin de rendre l'outil encore plus complet pour une utilisation réelle en équipe.