RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

UNIVERSITÉ LUMIÈRE LYON 2

Rapport de Projet

Planning Poker

Développement d'une application Planning Poker pour faciliter l'estimation collaborative en équipe.

Encadré par :
Valentin Lachand-Pascal

Réalisé par :
M. SAHRANE Mohamed Riad
M. TABET Idir
M. TABET Nassim

1. Introduction

Le projet Planning Poker est une application collaborative permettant aux équipes de développement de prioriser et d'évaluer les difficultés des tâches à réaliser. Conçue en Python avec Flask, cette application facilite la planification agile tout en favorisant une approche ludique et participative. Elle inclut des fonctionnalités de création de parties, de vote, de sauvegarde et de reprise des parties, ainsi que des modes de jeu personnalisés.

2. Objectifs

- Faciliter la planification agile grâce à une application intuitive.
- Offrir plusieurs modes de jeu pour s'adapter aux différents besoins d'évaluation.
- Assurer une expérience utilisateur fluide avec une interface propre et ergonomique.
- Implémenter des tests unitaires pour garantir la fiabilité de l'application.
- Fournir des mécanismes robustes de sauvegarde et de reprise des parties.

3. Architecture Technique

a. Langage et Framework

- **Langage :** Python 3.10
- **Framework Web :** Flask 3.1.0
- **Dépendances :** Werkzeug, Jinja2, click, MarkupSafe

b. Structure du Code

- **App/**
  - app.py : Contient la logique principale de l'application.
  - backlog.json : Données de tâches initiales pour les parties.
- **templates/** : Fichiers HTML pour les différentes pages.
- **static/** : Ressources CSS et images (cartes, icônes).
- **tests/** : Fichier de test unitaire.
- **partie_sauvegardee/** et **resultats/** : Dossiers de sauvegarde et de résultats.

4. Fonctionnalités Implémentées

- **Modes de Jeu :**
    - **Moyenne** : Moyenne arithmétique des votes.
    - **Médiane** : Calcul de la médiane pour un résultat plus neutre.
    - **Unanimité** : accord unanime requis pour avancer.
    - **Majorité** : Mode de vote où la valeur la plus fréquente est choisie.

- **Sauvegarde/Reprise des Parties :**
    - Exportation des états de partie au format JSON.
    - Récupération facile grâce à un code unique.

- **Interface Utilisateur :**
    - Conception ergonomique basée sur Bootstrap.
    - Cartes visuelles pour les votes.

- **Chat Intégré :**
    - Envoi de messages entre joueurs.
    - Historique de discussion associé à chaque partie.

- **Tests Unitaires :**
    - Couverture de plusieurs scénarios (création de partie, votes, sauvegarde).

5. Implémentation des Modes de Jeu

- **Moyenne :**
    - Calcul : Somme des valeurs divisée par le nombre de joueurs ayant voté.
    - Utilisation : Adapté pour les équipes cherchant un consensus approximatif.
- **Médiane :**
    - Calcul : Valeur centrale après tri des votes.
    - Utilisation : Réduit l'impact des valeurs extrêmes.
- **Unanimité :**
    - Règle : Tous les joueurs doivent voter la même valeur.
    - Utilisation : Garantit un accord total avant de passer à la tâche suivante.
- **Majorité :**
    - Règle : La valeur la plus fréquente est choisie.
    - Utilisation : Pratique pour les équipes nombreuses.

6. Sauvegarde et Reprise

a. Mécanisme de Sauvegarde

- Chaque partie est identifiée par un code unique.
- L'état complet (mode, votes, backlog) est sauvegardé au format JSON dans le dossier partie_sauvegardee/.

b. Reprise de Partie

- En saisissant le code de partie, les joueurs peuvent reprendre où ils se sont arrêtés.
- Les données sont chargées depuis le fichier JSON correspondant.

7. Tests Unitaires

- **Envoi d'un message dans le chat :**
  Vérifie que les messages sont envoyés correctement dans le chat de la partie.
  Résultat : Message 'Hello World' envoyé avec succès par 'host_player'.

- **Création d'une partie :**
  Teste la création d'une partie avec le mode de jeu défini.
  Résultat : Partie créée avec succès, mode de jeu 'moyenne'.

- **Vérification de la page d'accueil :**
  Vérifie que la page d'accueil fonctionne correctement.
  Résultat : La page d'accueil affiche le contenu prévu et fonctionne comme attendu.

- **Rejoindre une partie :**
  Vérifie qu'un joueur peut rejoindre une partie existante avec un code valide.
  Résultat : Le joueur 'player2' a rejoint la partie avec succès.

- **Chargement d'une partie sauvegardée :**
  Teste le chargement d'une partie sauvegardée à partir d'un fichier JSON.
  Résultat : Partie avec le code 'TEST123' chargée avec succès.

- **Fin d'une partie :**
  Simule la fin d'une partie et vérifie que les résultats sont sauvegardés correctement.
  Résultat : Partie terminée avec succès, résultats sauvegardés dans le dossier résultats.

- **Sauvegarde et chargement d'une partie :**
  Vérifie que l'état d'une partie est sauvegardé et peut être rechargé sans perte d'information.
  Résultat : Fichier sauvegardé et chargé avec succès, état conforme aux attentes.

- **Soumission d'un vote :**
  Vérifie qu'un joueur peut soumettre un vote et que celui-ci est enregistré correctement.
  Résultat : Vote soumis avec succès pour le joueur 'player1'.

8. Justification des Choix Techniques

- **Flask :**
    - Framework léger et adapté pour des projets de taille moyenne.
    - Simplicité d'intégration avec les fichiers statiques et les templates.

- **Python :**
    - Langage polyvalent, facile à lire et à maintenir.
    - Grande communauté offrant un support abondant.

- **Architecture Basée sur JSON :**
    - Format léger et universel pour stocker les états des parties.

- **Interface Basée sur Bootstrap :**
    - Rapidité de conception et résultat réactif.

9. Conclusion

L'application Planning Poker répond aux besoins des équipes agiles grâce à son interface conviviale, ses multiples modes de jeu et ses fonctionnalités robustes de sauvegarde et reprise. La mise en place de tests unitaires et d'intégration continue garantit la fiabilité et la qualité globale du projet.