

Rapport du Projet Planning Poker

BAH Mohamed Habib Julien Costa

December 20, 2024

Contents

1	Introduction	3
2	Présentation du Projet	3
2.1	Objectifs	3
2.2	Fonctionnalités principales	3
3	Architecture et Choix Techniques	3
3.1	Structure du Projet	3
3.2	Choix Techniques	4
4	Interface Utilisateur	4
4.1	Écran Principal	4
4.2	Configuration du Nombre de Joueurs	5
4.3	Entrée des Noms des Joueurs	6
4.4	Vote et Chronomètre	6
4.5	Résultats des Votes	6
5	Mode de Compilation et Exécution	7
5.1	Compilation	7
6	Choix Techniques et Justification	7
6.1	Langage et Bibliothèques	7
7	Documentation et Tests	8
7.1	Documentation	8
7.2	Tests Unitaires	8

8	Intégration Continue	8
9	Résultats et Analyse	9
9.1	Résultats Générés	9
9.2	Analyse	9
10	Conclusion	9

1 Introduction

Ce projet s'inscrit dans le cadre d'une méthode agile pour la gestion de projets. L'objectif est de développer une application interactive permettant de réaliser une estimation collective des fonctionnalités grâce à la méthode **Planning Poker**.

2 Présentation du Projet

2.1 Objectifs

L'objectif principal du projet est de proposer une interface simple et interactive pour effectuer des estimations en groupe.

L'application inclut :

- La configuration du nombre de joueurs.
- La gestion des fonctionnalités du backlog.
- La possibilité de choisir différents modes de vote : unanimité stricte, moyenne, médiane ou majorité absolue.

2.2 Fonctionnalités principales

1. Configuration du nombre de joueurs et de leurs noms.
2. Gestion d'un backlog de fonctionnalités.
3. Chronomètre de vote par joueur pour une meilleure organisation.
4. Sauvegarde et visualisation des résultats des votes dans un fichier JSON.
5. Mode de reprise à partir d'un backlog sauvegardé.

3 Architecture et Choix Techniques

3.1 Structure du Projet

La structure du projet suit une architecture modulaire avec les composants suivants :

```

agile/
src/
    __init__.py
    app.py          # Point d'entrée principal
    ui.py           # Gestion de l'interface utilisateur
    utils.py        # Fonctions utilitaires
    game_logic.py   # Logique métier
data/
    backlog.json    # Backlog des fonctionnalités
docs/               # Documentation générée
tests/              # Tests unitaires
output_images/      # Cartes converties en images PNG
svg_cards/          # Cartes SVG
README.md           # Documentation utilisateur

```

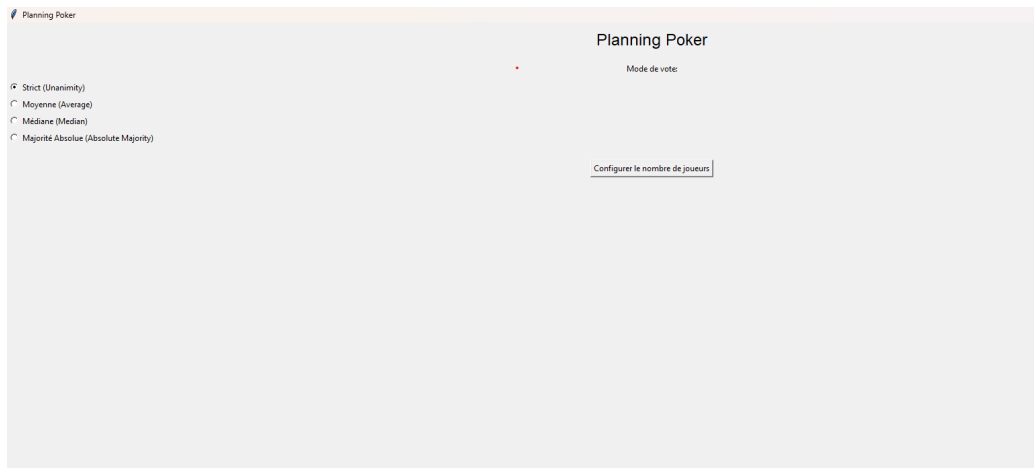
3.2 Choix Techniques

- **Langage** : Python 3.11
 - Simplicité pour la gestion de l'interface utilisateur et des données.
- **Bibliothèque graphique** : tkinter
 - Intégrée à Python, elle permet de créer une interface utilisateur portable.
- **Sauvegarde des données** : JSON
 - Format léger et facile à manipuler pour le backlog et les résultats.
- **Gestion des cartes** : Conversion des fichiers SVG en PNG avec cairosvg.

4 Interface Utilisateur

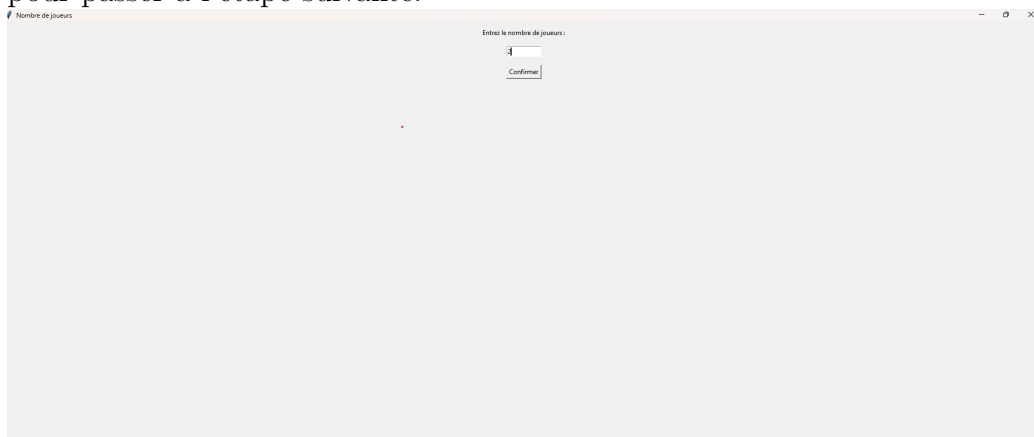
4.1 Écran Principal

L'utilisateur commence par sélectionner un mode de vote parmi les options disponibles.



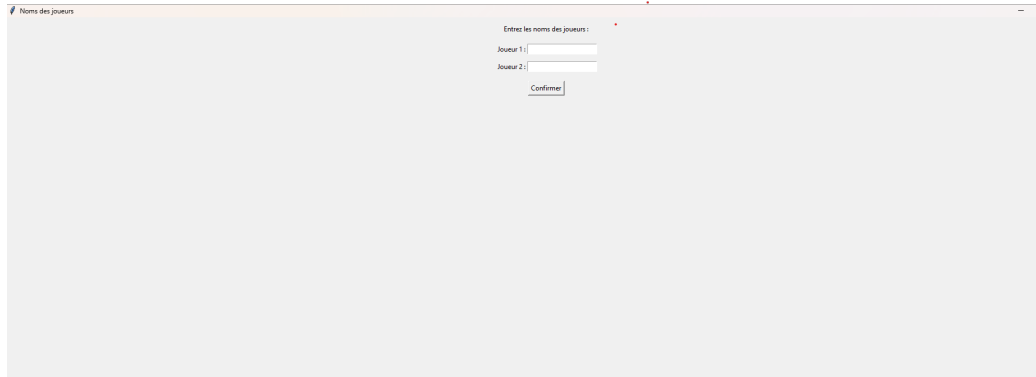
4.2 Configuration du Nombre de Joueurs

L'utilisateur entre le nombre de joueurs. Ce paramètre est indispensable pour passer à l'étape suivante.



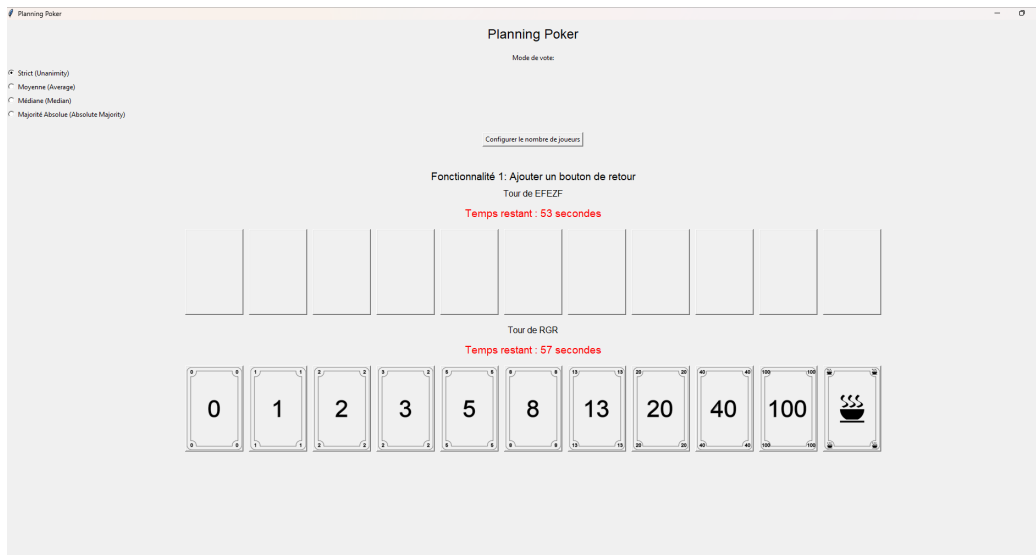
4.3 Entrée des Noms des Joueurs

Chaque joueur entre son nom pour faciliter l'identification lors du vote.



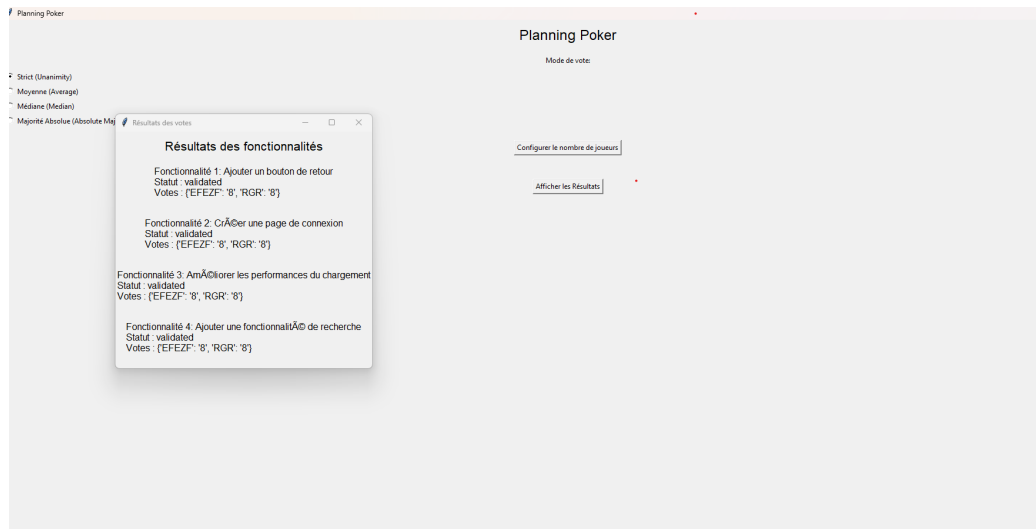
4.4 Vote et Chronomètre

Chaque joueur vote pour une fonctionnalité en utilisant les cartes disponibles. Un chronomètre de 60 secondes est affiché pour limiter le temps de réflexion.



4.5 Résultats des Votes

Une fois tous les votes terminés, l'utilisateur peut afficher les résultats dans une nouvelle fenêtre et sauvegarder les données dans un fichier JSON.



5 Mode de Compilation et Exécution

5.1 Compilation

L'application est compilée et lancée comme suit:

- Placez-vous dans le répertoire parent du dossier `src` (dossier `agile`).
- Exécutez la commande suivante :

```
python -m src.app
```

Cette méthode garantit le bon fonctionnement des imports relatifs grâce à la structure modulaire du projet.

6 Choix Techniques et Justification

6.1 Langage et Bibliothèques

- **Python** : Choisi pour sa simplicité et son efficacité pour le prototypage rapide.
- **Tkinter** : Bibliothèque native pour construire des interfaces graphiques interactives.

- **Pillow** : Manipulation d'images pour les cartes.
- **Cairosvg** : Conversion des fichiers SVG en PNG.
- **pdoc** : Génération automatique de documentation.
- **flake8** : Analyse statique pour respecter les normes de codage.

7 Documentation et Tests

7.1 Documentation

La documentation est générée automatiquement avec **pdoc**. Chaque module contient des **docstrings** expliquant le rôle de ses fonctions et classes.

7.2 Tests Unitaires

Des tests unitaires garantissent la fiabilité de l'application :

- **Tests de la logique métier** : Validation des calculs des modes de jeu (unanimité, moyenne, etc.).
- **Tests de l'interface utilisateur** : Vérification des interactions utilisateur et de l'affichage des résultats.

8 Intégration Continue

Une pipeline CI a été configurée avec **GitHub Actions** pour automatiser les étapes suivantes :

- **Tests automatisés** : Les tests unitaires sont exécutés à chaque modification du code.
- **Analyse statique** : Utilisation de **flake8** pour détecter les erreurs de style ou de structure.
- **Génération de documentation** : Création automatique de documentation HTML à jour.

9 Résultats et Analyse

9.1 Résultats Générés

Les votes de chaque joueur sont enregistrés pour chaque fonctionnalité. Voici un exemple de fichier JSON généré :

```
[
  {
    "feature_id": 1,
    "description": "Ajouter un bouton de retour",
    "status": "validated",
    "votes": {"Joueur 1": "8", "Joueur 2": "8"}
  }
]
```

9.2 Analyse

- Le mode de vote garantit que toutes les fonctionnalités sont discutées jusqu'à obtenir un consensus ou une validation.
- Le chronomètre contribue à maintenir une dynamique rapide et efficace.

10 Conclusion

Ce projet répond aux objectifs initiaux en offrant une solution interactive et modulaire pour le Planning Poker. Grâce à la simplicité de son interface et à sa modularité, cette application est adaptée pour les équipes agiles cherchant à estimer leurs tâches de manière collaborative.