Rapport

Application de planning poker

Réalisé par : Aamara Aya, Bouaouich Nouhayla et Amirat Meroua

1. Introduction

Dans le cadre de ce projet, nous avons développé une application de Planning Poker, une méthode d'estimation collaborative largement utilisée dans les méthodes agiles. Cette application permet à plusieurs joueurs d'estimer la difficulté des fonctionnalités d'un backlog, en respectant des règles précises vues en cours.

Le projet vise à mettre en pratique plusieurs notions importantes telles que la modélisation, la justification des choix techniques, la mise en place de l'intégration continue, les tests unitaires ainsi que la génération de documentation.

2. Objectifs de l'application

L'objectif principal de l'application est de permettre à plusieurs joueurs de participer à une session de planning poker afin d'estimer la complexité des fonctionnalités d'un backlog.

Les objectifs secondaires sont :

- Gérer plusieurs joueurs avec des pseudos personnalisés
- Proposer différents modes de validation des estimations
- Permettre le chargement et la sauvegarde des données via des fichiers JSON
- Respecter les règles spécifiques du planning poker (unanimité, carte café, etc.)

3. Organisation du travail

Le projet a été réalisé par un groupe de trois personnes. Étant donné qu'il s'agissait de notre première expérience de travail collaboratif avec Git et GitHub, l'organisation initiale a nécessité une phase d'apprentissage.

Au début du projet, plusieurs dépôts ont été utilisés afin que chaque membre puisse expérimenter les fonctionnalités de Git (commits, branches, synchronisation). Cette phase explique le fait que les commits n'aient pas été réguliers dans un premier temps.

Une fois les outils maîtrisés, le travail a été rassemblé dans un dépôt GitHub final, servant de version officielle du projet. Cette démarche nous a permis de consolider l'ensemble des fonctionnalités, d'unifier le code et de mettre en place l'intégration continue sur un dépôt stable.

Cette expérience a été formatrice et a permis au groupe de mieux comprendre les bonnes pratiques du travail collaboratif et de la gestion de versions.

4. Fonctionnalités principales

Gestion des joueurs :

- Un menu permet de définir le nombre de joueurs
- Chaque joueur doit saisir un pseudo
- Le jeu peut être :
    - À distance : chaque joueur vote depuis son propre dispositif (fonctionnalité bonus)
    - En local : les joueurs votent chacun à leur tour sur le même dispositif

Choix des règles de Planning Poker :

L'application permet de choisir différentes règles de validation des estimations :

- Mode strict (obligatoire) : unanimité requise
- Un mode supplémentaire parmi :
    - Moyenne
    - Médiane
    - Majorité absolue
    - Majorité relative

Gestion du backlog :

- Le backlog est fourni sous forme de fichier JSON
- La structure du JSON est libre
- Chaque fonctionnalité est estimée indépendamment

Une fois une fonctionnalité validée :

- Son estimation finale est enregistrée
- L'application passe à la fonctionnalité suivante

Sauvegarde et reprise de partie :

- À la fin du backlog, l'application génère un fichier JSON contenant :
    - Chaque fonctionnalité
    - Sa difficulté estimée par l'équipe

5. Choix Techniques et Architecture

1. Langage et bibliothèques utilisées

Le projet a été développé sous la forme d'une application web, avec une séparation claire entre le backend en PHP et le frontend en JavaScript.

Le langage PHP a été choisi pour le backend pour plusieurs raisons :

- Il est bien adapté au développement d'applications web
- Il permet une manipulation simple des fichiers JSON
- Il offre une exécution côté serveur sécurisée
- Il est largement documenté et utilisé

PHP permet d'implémenter efficacement la logique métier du planning poker.

Le JavaScript a été utilisé pour le frontend afin de :

- Rendre l'interface dynamique
- Gérer les événements utilisateurs (clics, votes)
- Communiquer avec le backend

Ce choix améliore l'expérience utilisateur et permet une interaction fluide pendant la partie.

Ce choix technologique est cohérent avec le besoin d'une application accessible, facilement testable et exécutable sans configuration complexe sur la machine de l'évaluateur.

2.Architecture logicielle

L'application suit une architecture en couches, permettant de séparer clairement :

- la logique métier (gestion des votes, calcul des estimations),
- la gestion des données (lecture et écriture des fichiers JSON),
- l'interface utilisateur (interaction avec le joueur).

Cette architecture améliore la maintenabilité du code, facilite les tests unitaires et rend le projet plus évolutif.

3. Modélisation et diagrammes de classes

3.1 Diagramme de cas d'utilisation

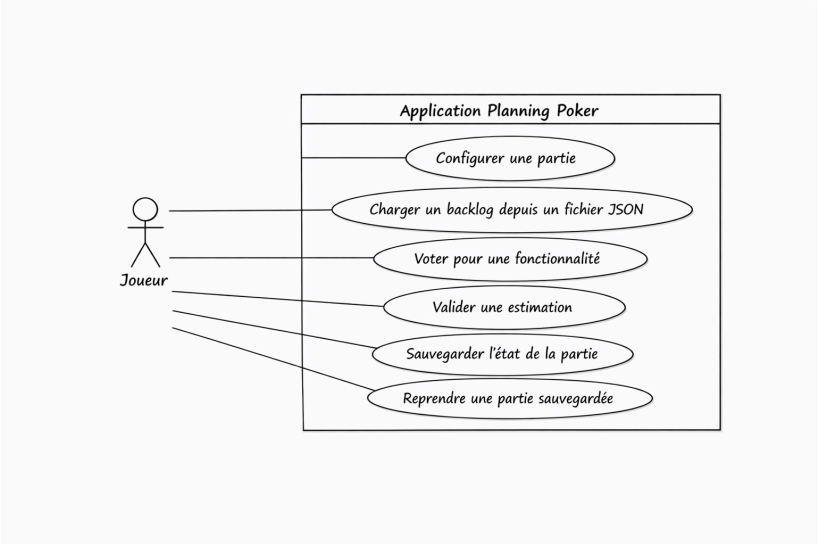
Le diagramme de cas d'utilisation décrit les interactions entre les utilisateurs (acteurs) et le système.

Acteur principal : Joueur

Cas d'utilisation principaux :

- Configurer une partie (nombre de joueurs, pseudos, règles)
- Charger un backlog depuis un fichier JSON
- Voter pour une fonctionnalité
- Valider une estimation
- Sauvegarder l'état de la partie
- Reprendre une partie sauvegardée

Ce diagramme met en évidence les fonctionnalités offertes par l'application du point de vue utilisateur.



[DIGRAMME DE CAS D'UTILISATION SIMPLIFIE]

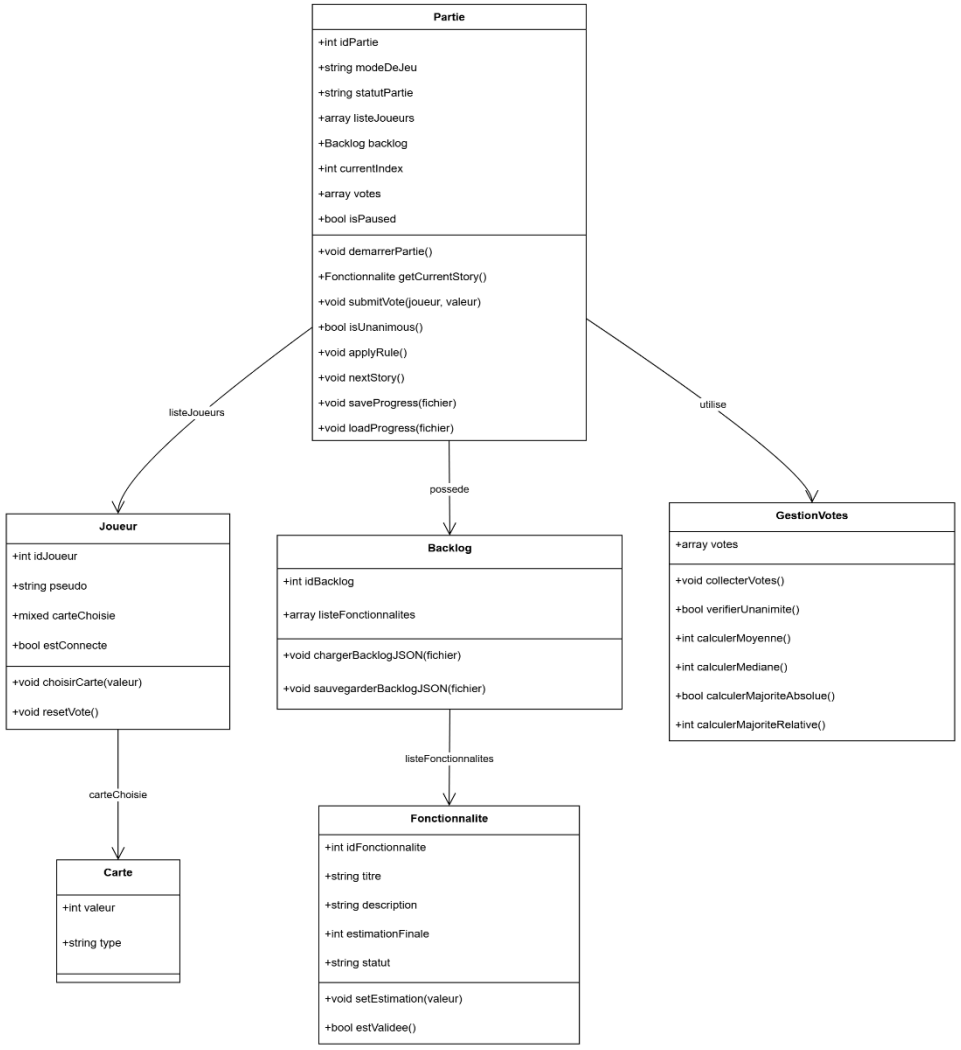
3.2 Diagramme de classes

La modélisation du projet repose sur plusieurs entités métier essentielles au fonctionnement du Planning Poker :

- Joueur : représente un participant avec un pseudo et un vote courant.
- Backlog : contient la liste des fonctionnalités à estimer.

- Fonctionnalité : représente une tâche du backlog avec son état (validée ou non) et son estimation finale.
- Vote : gère les valeurs choisies par les joueurs lors d'un tour.
- Règles de jeu : implémente les différentes stratégies de validation (strict, moyenne, médiane, majorité).

Le diagramme de classes présentés dans le rapport se concentrent uniquement sur ces éléments clés afin de garantir une bonne lisibilité et une compréhension rapide du modèle de données.



[DIGRAMME DE CLASSES SIMPLIFIE]

4. Gestion des données (JSON)

Les données sont stockées sous forme de fichiers JSON, ce qui offre :

- une structure claire et lisible,
- une facilité de modification manuelle,
- une compatibilité native avec Python.

Deux types de fichiers sont utilisés :

- un fichier Backlog JSON, contenant les tâches à estimer,
- un fichier de sauvegarde de partie, permettant de reprendre une session ultérieurement.

6. Mise en place de l'Intégration Continue (CI)

1. Objectif de l'intégration continue

La CI permet d'automatiser les vérifications du projet à chaque modification du code source. Elle garantit :

- la stabilité du code
- la détection précoce des erreurs
- la génération automatique de la documentation
- le respect des bonnes pratiques de développement

La CI repose sur GitHub Actions, directement intégré à GitHub.

2. Déclenchement et environnement

Le pipeline CI est défini dans .github/workflows/ci.yml et se déclenche :

- à chaque push sur n'importe quelle branche
- à chaque pull request

Toutes les étapes du pipeline sont exécutées automatiquement par GitHub Actions, ce qui garantit que le code et la documentation sont toujours vérifiés et à jour.

3. Workflow du pipeline

Checkout : récupération du code depuis GitHub (actions/checkout@v4)

Configuration Node.js : installation de Node.js version 20 (actions/setup-node@v4) pour compatibilité avec npm, Vitest et JSDoc

Installation des dépendances : npm install pour installer toutes les dépendances définies dans package.json

Exécution des tests unitaires : npm run test sur estimation.js pour vérifier la logique métier (calcul des estimations, validation des votes, unanimité)

- Les tests s'exécutent automatiquement à chaque push ou pull request
- Si un test échoue, le pipeline s'arrête et le code n'est pas intégré
- résultats des tests dans GitHub Actions

tests > JS estimations.test.js > ...

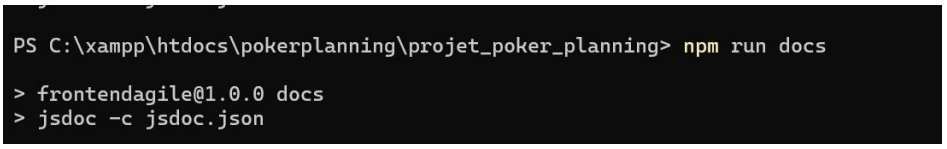
```
1  import { describe, it, expect } from "vitest";
2  import { average, median, absoluteMajority, relativeMajority } from "../src/estimation.js";
3
4  describe("estimation", () => {
5    it("average", () => {
6      expect(average([1, 2, 3])).toBe(2);
7      expect(average([])).toBe(null);
8    });
9
10   it("median", () => {
11     expect(median([1, 3, 2])).toBe(2);
12     expect(median([1, 2, 3, 4])).toBe(2.5);
13     expect(median([])).toBe(null);
14   });
15
16   it("absoluteMajority", () => {
17     expect(absoluteMajority([3, 3, 3, 2])).toBe(3);
18     expect(absoluteMajority([1, 2, 2, 3])).toBe(null);
19   });
20
21   it("relativeMajority", () => {
22     expect(relativeMajority([1, 2, 2, 3])).toBe(2);
23     expect(relativeMajority([])).toBe(null);
24   });
25 });
26
```

```
Run `npm audit` for details.
PS C:\xampp\htdocs\pokerplanning\projet_poker_planning> npm run test

> frontendagile@1.0.0 test
> vitest run


 RUN  v2.1.9 C:/xampp/htdocs/pokerplanning/projet_poker_planning

 ✓ tests/estimations.test.js (4)
   ✓ estimation (4)
     ✓ average
     ✓ median
     ✓ absoluteMajority
     ✓ relativeMajority

 Test Files  1 passed (1)
      Tests  4 passed (4)
   Start at  12:49:43
   Duration  868ms (transform 77ms, setup 0ms, collect 111ms, tests 17ms, environment 1ms,

PS C:\xampp\htdocs\pokerplanning\projet_poker_planning>
```

Génération de la documentation : npm run docs via JSDoc

- npm lit package.json et trouve le script docs : jsdoc -c jsdoc.json
- JSDoc lit jsdoc.json et analyse le fichier poker.js
- Les commentaires /** ... */ sont transformés en pages HTML dans le dossier docs/ (docs/index.html)

- Documentation générée et pipeline CI vert



4. Apports de la CI

La mise en place de l'intégration continue apporte plusieurs avantages :Automatisation des contrôles : plus besoin de lancer manuellement les tests ou la documentation

- Fiabilité accrue : toute modification du code est immédiatement vérifiée
- Qualité logicielle : les erreurs sont détectées rapidement
- Documentation systématique : toujours cohérente avec le code
- Approche professionnelle : conforme aux pratiques utilisées en entreprise et dans les projets Agile

7. Manuel Utilisateur et Fonctionnalités

1. Installation et lancement

a. Cloner le dépôt :

git clone https://github.com/aya-spongebob/projet_poker_planning.git

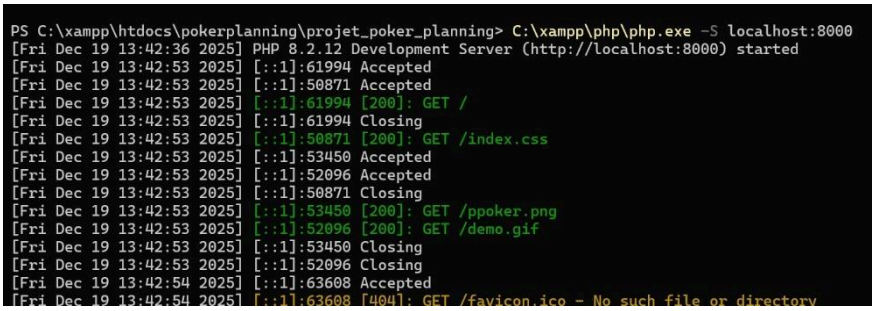
cd projet_poker_planning

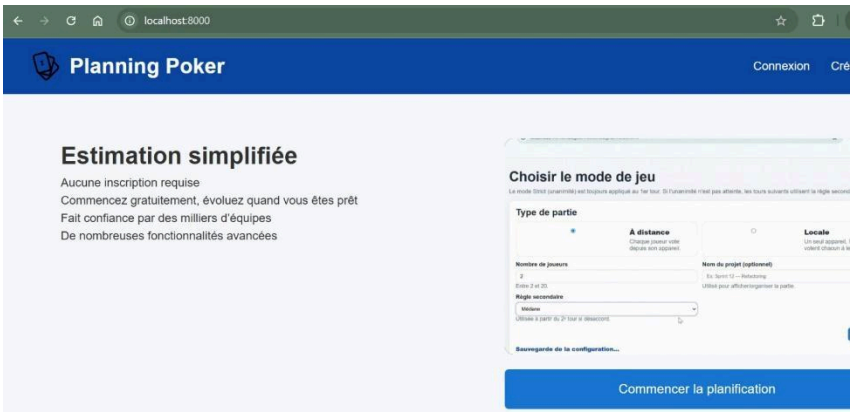
b. Lancer un serveur local :

C:\xampp\php\php.exe -S localhost:8000

c. Ouvrir le projet dans le navigateur :

http://localhost:8000



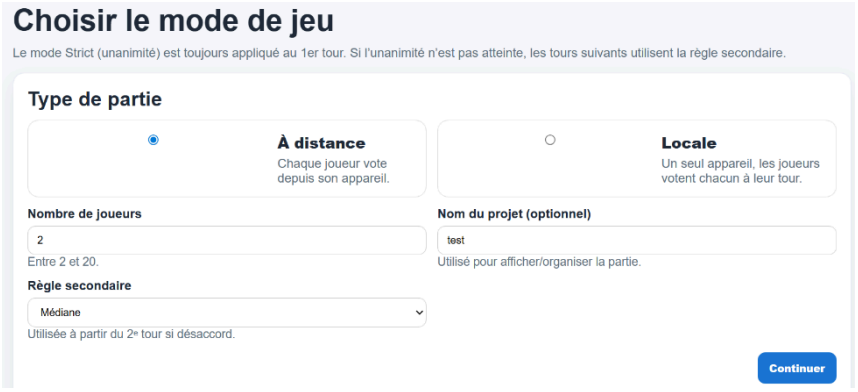


2. Interface et Déroulement d'une Partie

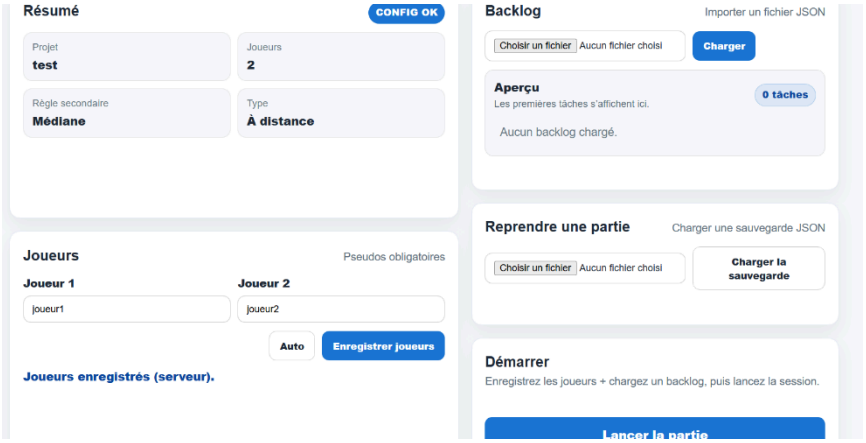
Le flux d'une partie est le suivant :

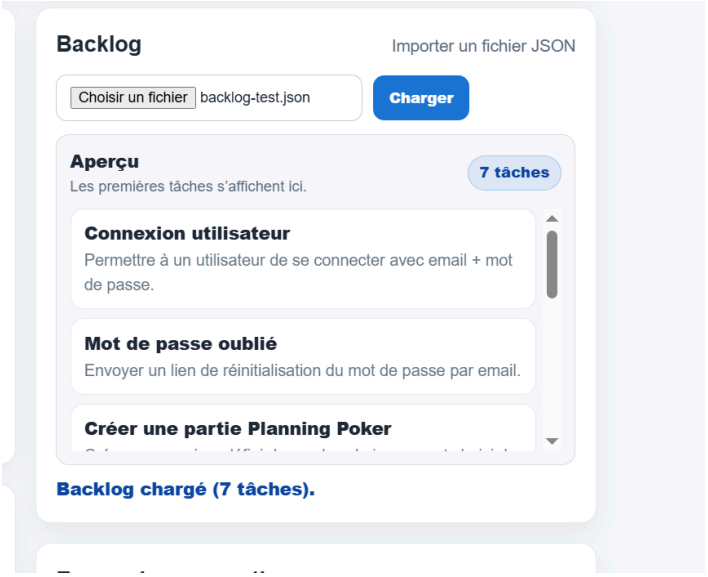
1. **Menu principal** : choisir le nombre de joueurs et entrer leurs pseudos.
2. **Sélection des règles** : choisir le mode strict et le mode secondaire (moyenne, médiane ou majorité).
3. **Chargement du backlog** : importer un fichier JSON contenant la liste des fonctionnalités à estimer.
4. **Vote des joueurs** : chaque joueur choisit sa carte pour chaque fonctionnalité.
5. **Validation ou nouveau vote** : selon les règles sélectionnées, la fonctionnalité est validée ou un nouveau tour de vote est lancé.
6. **Sauvegarde de la partie** : l'état de la partie est enregistré dans un fichier JSON, permettant de reprendre la partie ultérieurement.
7. **Reprise d'une partie** : possibilité de charger un fichier JSON sauvegardé pour continuer la session.

- Menu principal et saisie des pseudos Sélection des règles et mode de jeu

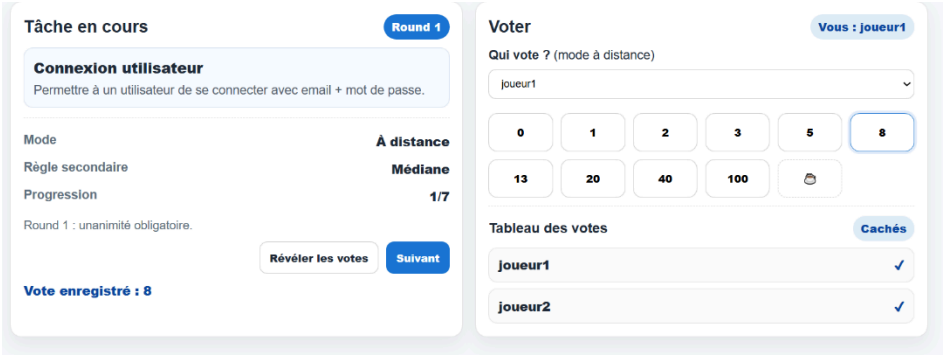


- Saisie des pseudos et chargement du backlog

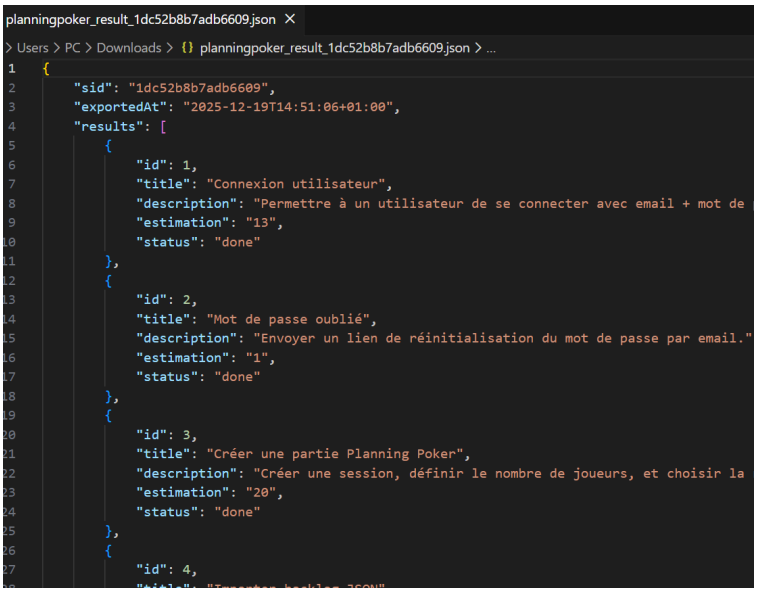




- Interface de vote pour une fonctionnalité



- Aperçu du backlog et des résultats des votes



8. Conclusion

Le projet Planning Poker nous a permis de développer une application complète pour estimer collectivement la complexité des fonctionnalités d'un backlog en respectant les règles du Planning Poker.

Au cours de ce projet, nous avons consolidé nos compétences en développement web avec PHP et JavaScript, appris à travailler efficacement en groupe de trois personnes, et découvert l'utilisation de Git et GitHub pour la gestion collaborative du code.

La mise en place d'une intégration continue avec GitHub Actions a permis d'automatiser les tests unitaires et la génération de documentation, assurant ainsi la qualité et la fiabilité du projet.

L'application propose différents modes de jeu, une interface intuitive et la possibilité de sauvegarder et reprendre les parties, répondant ainsi aux besoins fonctionnels définis au départ.

Ce projet nous a permis non seulement d'appliquer les concepts théoriques appris en cours, mais également de découvrir les bonnes pratiques professionnelles en matière de développement Agile et de gestion de projet logiciel.