

Rapport de projet

*Conception Agile - Application
Planning Poker*

Réalisé par :

- MOURID Fatine
 - SAADANE Manel
-

1. Introduction et Contexte

Dans le cadre du module “**Conception agile de projets informatiques**” , nous avons réalisé une application portant sur le développement d'une application de **Planning Poker**.

Le Planning Poker est une méthode utilisée en gestion de projet agile afin d'aider une équipe à estimer la complexité des tâches, appelées **user stories**.

L'objectif principal de cette application est de permettre à **plusieurs joueurs** de participer à une **session de vote** pour estimer les tâches d'un projet. Chaque joueur choisit une carte correspondant à son estimation. Une fois les votes révélés, l'application vérifie si une estimation peut être validée selon le mode de calcul choisi : **unanimité, moyenne, médiane ou majorité (absolue/ relative)**. En cas de désaccord entre les joueurs, un nouveau tour de vote est alors lancé.

Ce projet a été réalisé en binôme par **Fatine Mourid** et **Manel Saadane**. Nous avons travaillé ensemble en suivant le principe du **pair programming** vu en cours. Pour collaborer, nous avons utilisé **CodeSandbox**, un outil de développement en ligne, avec un **lien partagé**, ce qui nous a permis de travailler sur le même projet, d'échanger régulièrement et de suivre l'avancement du travail.

2. Choix Techniques et Architecture

2.1 Langage et framework utilisé

Nous avons choisi de développer l'application en utilisant le **framework React JS**. Ce choix s'explique notamment par le fait que nous avions déjà travaillé **ensemble en binôme sur des projets utilisant React lors de notre Licence 2**. Cette expérience commune avec ce framework nous a permis de gagner du temps sur la prise en main des outils et de nous concentrer davantage sur la logique de l'application et le respect des règles du Planning Poker.

Nous trouvons que React est bien adapté à ce projet, car il permet de créer une interface interactive et dynamique. Les actions des utilisateurs (**votes, révélation des cartes, changement de tour, affichage des résultats**) entraînent des mises à jour fréquentes de l'interface, que React gère de manière efficace.

Le fonctionnement par composants nous a permis de séparer clairement les différentes parties de l'application, comme **le menu, les cartes de vote, l'affichage des votes, les résultats et le chronomètre**. Cette organisation rend le code plus lisible et plus facile à maintenir.

2.2 Architecture logicielle

Notre application est organisée de manière **simple**, en séparant l'interface et la logique. On peut dire que c'est une **architecture en couches**, et aussi une version simplifiée de **MVC**.

- **Partie interface (Vue)** : tout ce qui est affichage et interaction utilisateur est dans le dossier **components/**.
 - **Par exemple** : le menu, les cartes, l'affichage des votes, le résultat, le chronomètre et le chat.
- **Partie logique métier (Contrôle/ règles)** : les règles de vote et les vérifications sont regroupées dans le dossier **services/**.
 - On y trouve par exemple **le calcul de la moyenne/médiane/majorité**, la vérification de l'unanimité et la gestion des fichiers Json.
- **Composant principal (App.jsx)** : c'est le **centre** de l'application. Il **récupère** les informations du menu, garde l'état de la partie (joueurs, backlog, votes, tour, etc.) et envoie ce qu'il faut aux composants pour **l'affichage**.

Grâce à cette organisation, le code est plus clair : chaque fichier a **un rôle précis**, et on évite de mélanger l'interface avec les règles de calcul. Cela rend aussi l'application **plus facile à tester**, car les fonctions importantes (règles de vote) sont séparées de l'interface.

2.3 Modélisation (diagrammes)

Pour la modélisation, **nous n'avons pas généré un diagramme automatique** de tout le projet (car cela donnerait un schéma illisible). À la place, nous avons choisi de représenter seulement les éléments importants du modèle de données, ceux qui expliquent vraiment le fonctionnement du Planning Poker.

Le but du diagramme est de montrer clairement :

- Les joueurs
- Le backlog (liste de user stories)
- Les votes
- Les règles de calcul (unanimité, moyenne, médiane, majorités)

Nous avons donc fait un diagramme **simple** qui se concentre sur ces objets et leurs relations, afin qu'un lecteur comprenne rapidement comment une partie se déroule.

Figure 1 : Modèle métier

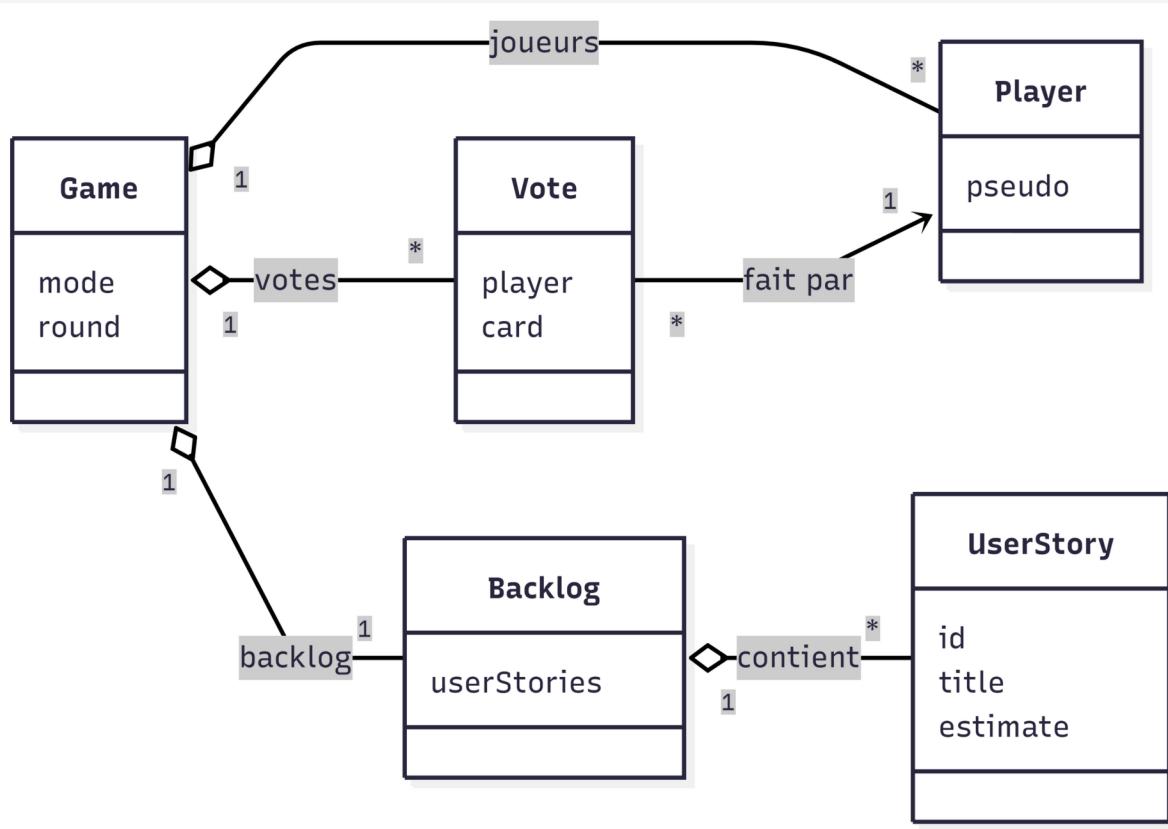
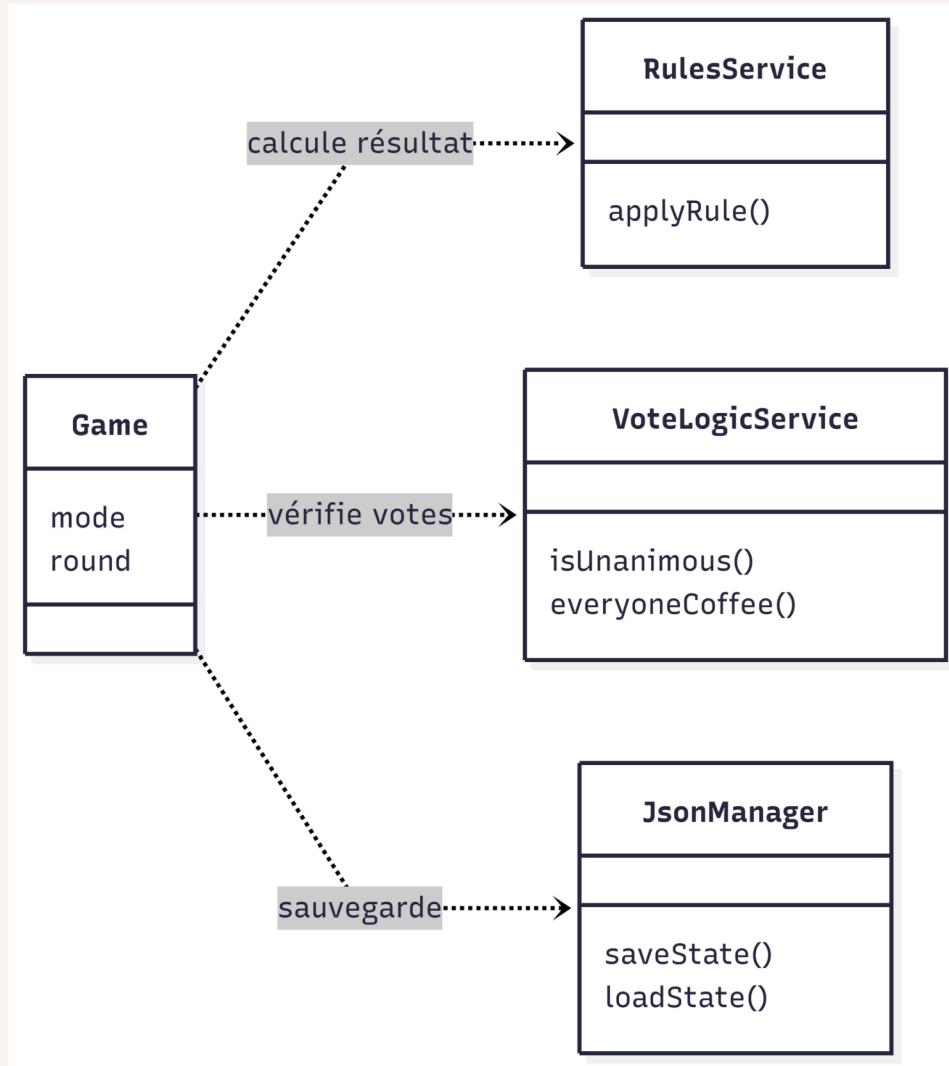


Figure 2 : Services



2.4 Gestion des données

Conformément à l'énoncé, nous avons choisi d'utiliser des **fichiers Json** pour gérer le **backlog, la sauvegarde de partie et les résultats finaux.**

2.4.1 Fichier Json du backlog

Le backlog est un tableau (liste) de user stories.

Chaque user story contient au minimum :

- id : identifiant de la tâche (ex : US1)
- title : description courte de la tâche (ex: Créer une session de jeu)
- estimate : estimation (au début null, puis remplie après validation)

2.4.2. Sauvegarde/ reprise de partie (export/ import de l'état)

L'application permet de **sauvegarder une partie en cours** sous la forme d'un fichier Json appelé état de la partie.

Ce fichier contient toutes les informations nécessaires pour **reprendre la partie ultérieurement à l'aide du fichier sauvegardé.**

Grâce à cette sauvegarde, il est possible de **reprendre la partie exactement là où elle s'était arrêtée**, sans perdre l'avancement du Planning Poker.

2.4.3 Cas spécial : carte café

Si tous les joueurs votent la carte café, l'application génère un fichier Json de type **snapshot**, qui enregistre l'état d'avancement du backlog. Ce fichier peut ensuite être chargé via le menu pour reprendre la partie.

2.4.4 Export des résultats finaux

Quand toutes les user stories du backlog sont validées, l'application exporte **un fichier Json final** contenant le backlog complet avec, pour chaque user story, l'estimation finale de l'équipe.

3. Mise en place de l'Intégration Continue

Afin d'améliorer la qualité du projet et de vérifier automatiquement le bon fonctionnement du code, nous avons mis en place une intégration continue à l'aide de GitHub Actions.

3.1 Workflow du pipeline ci

Le pipeline d'intégration continue est déclenché automatiquement :

- lors d'un **push** sur la branche **principale main**
- lors de la création ou de la mise à jour d'une pull request

Le workflow suit les étapes suivantes :

1. Récupération du dépôt (Checkout)

2. Le code du projet est récupéré depuis le dépôt GitHub.

3. Installation de l'environnement

4. L'environnement **Node.js** est configuré, puis les dépendances du projet sont installées à l'aide de **pnpm**.

5. Exécution des **tests unitaires**

6. Les **tests** sont lancés automatiquement

7. Génération de la **documentation**

Ce workflow permet de vérifier rapidement que le projet reste fonctionnel après chaque modification.

3.2 Tests Unitaires

Dans ce projet, les tests unitaires sont exécutés automatiquement grâce à la commande :

pnpm test

Les tests portent sur **la logique métier du Planning Poker**, en particulier **la validation de l'unanimité en mode strict et le comportement des règles de vote comme la majorité absolue**.

Ces tests sont implémentés dans le fichier **rules.test.js**.

3.3 Génération de la documentation dans le projet

Dans ce projet, la documentation est générée automatiquement à l'aide de JSDoc.

La commande suivante est exécutée dans le pipeline ci : **pnpm run doc**

Cette commande génère une documentation HTML à partir des commentaires présents dans le code source de notre application.

→ Les fichiers générés sont stockés dans le dossier **docs/**.

4. Manuel Utilisateur et Fonctionnalités

4.1 Installation et lancement de l'application

4.1.1 Lancement en local

Le projet peut être cloné et lancé directement en local, sans configuration complexe.

Prérequis

- Node.js
- pnpm ou npm ou yarn
- Un navigateur web

Clonage du projet

- **git clone https://github.com/msaadane2/PlanningPokerFM.git**
- **cd PlanningPokerFM**

Installation des dépendances

- **pnpm install ou yarn install ou npm install** : installe l'ensemble des dépendances du projet
- **pnpm install firebase ou yarn add firebase ou npm install firebase** : installe Firebase, utilisé dans l'application pour la fonctionnalité de chat en temps réel.

Lancement de l'application

pnpm start

Une fois la commande exécutée, l'application se lance automatiquement dans le navigateur.

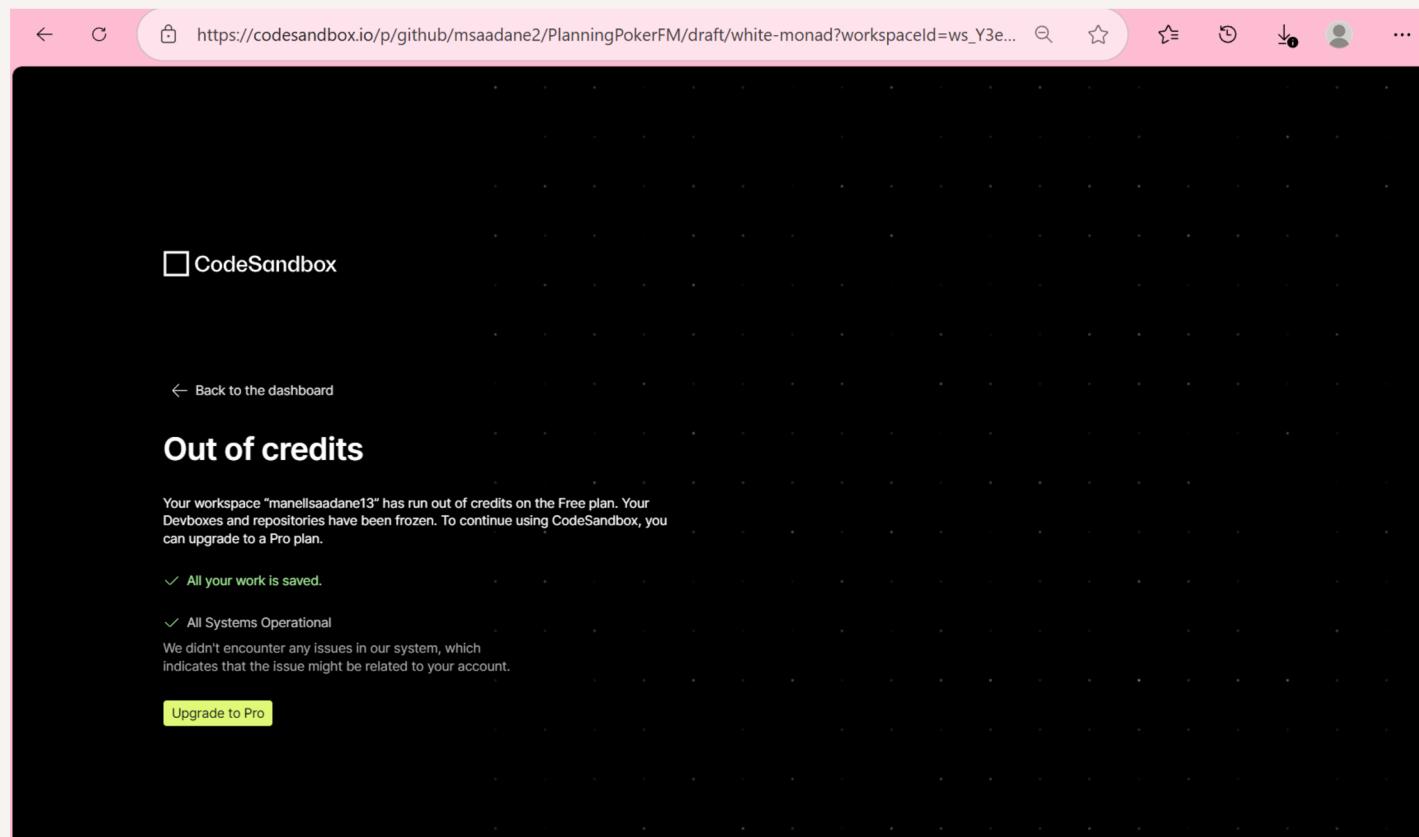
4.1.2 Accès en ligne (CodeSandbox)

Le projet a également été développé et testé en ligne via **CodeSandbox**, qui a servi d'environnement principal de travail collaboratif. Cette plateforme nous a permis de partager le code, de lancer l'application et de tester les fonctionnalités sans installation locale.

https://codesandbox.io/p/github/msaadane2/PlanningPokerFM/draft/white-monad?workspaceId=ws_Y3ejotG8QJSKrHYMjqxBqD

Cependant, au moment du rendu, l'accès à l'environnement est temporairement limité en raison de l'atteinte du nombre de crédits disponibles sur la version gratuite. Une capture d'écran est jointe afin d'illustrer cette contrainte technique.

L'ensemble du travail a néanmoins été entièrement sauvegardé avec le dépôt GitHub du projet, qui contient le **code final, la documentation, les tests et le rapport**.



4.1.2 Déploiement en ligne (Vercel)

L'application est également accessible via un déploiement en ligne réalisé avec Vercel, permettant de tester la version finale directement depuis un navigateur.

<https://fmplanningpoker.vercel.app/>

4.2 Mode de jeu

Dans notre application, l'utilisateur choisit un mode de jeu au début de la partie depuis le menu. Le mode sélectionné est ensuite utilisé pour valider les estimations pendant la session.

1) Mode strict (unanimité)

Le mode strict correspond à la règle d'unanimité :

- Tous les joueurs doivent voter exactement la même valeur pour qu'une user story soit validée
- si les votes sont différents, l'estimation n'est pas validée et un nouveau tour de vote est relancé jusqu'à obtenir l'unanimité.

2) Autres modes disponibles (moyenne/ médiane/ majorité)

En plus du mode strict, notre application propose aussi les modes suivants :

- Moyenne : on calcule la moyenne des votes numériques
- Médiane : on garde la valeur centrale après tri
- Majorité absolue : une valeur est validée seulement si elle dépasse 50 % des votes
- Majorité relative : on retient la valeur la plus fréquente.

Spécificité importante : premier tour à l'unanimité

Même lorsque l'on choisit un mode comme la moyenne, la médiane ou une majorité, le premier tour de vote pour chaque user story se fait toujours à l'unanimité.

4.3 Interface

4.3.1 Présentation de l'interface

L'application propose une interface **simple et intuitive**, pensée pour guider l'utilisateur tout au long d'une partie de Planning Poker. Les différentes étapes sont clairement séparées et accessibles depuis l'écran principal.

L'interface est composée de plusieurs zones principales :

- Un menu de configuration pour définir les joueurs et le mode de jeu et une zone de chargement et de sauvegarde des fichiers Json

Bienvenue sur l'application de Planning Poker



- Une zone de vote avec les cartes de Planning Poker et une zone d'affichage des votes et des résultats

Vote par cartes de Planning Poker

Fatine							
0	1	2	3	5	8	13	20
0	1	2	3	5	8	13	20
0	1	2	3	5	8	13	20
40	100	?	?	?	?	?	?
40	100	?	?	?	?	?	?
40	100	?	?	?	?	?	?

Vote : —

Votes

0/2 joueur(s) ont voté

[Révéler les votes](#)

Résultat

- Des éléments complémentaires comme le chronomètre et le chat.

⌚ Temps restant : 75s

Échanges entre joueurs

Fatine : Bonjour

Tout supprimer

Écrire un message... Envoyer

Vote par cartes de Planning Poker

Fatine

⌚ Temps restant : 52s

Échanges entre joueurs

Fatine : Bonjour

Manel : Salut

Tout supprimer

Écrire un message... Envoyer

Vote par cartes de Planning Poker

Manel

0 1 2 3 5 8 13 20

4.3.2 Flux d'une partie

Le déroulement d'une partie suit les étapes suivantes :

1. L'utilisateur commence par configurer la partie : **ajout des joueurs, choix du mode de jeu et accès aux options d'import ou d'export.**
2. Chargement du backlog json
3. Un fichier json contenant le backlog est importé. Les user stories sont alors chargées dans l'application.
4. Vote
5. Les joueurs votent à l'aide des cartes de Planning Poker. Les votes restent cachés tant que tous les joueurs n'ont pas voté.
6. Révélation et validation
7. Les votes sont révélés. Selon le mode de jeu choisi, l'application valide l'estimation ou relance un nouveau tour de vote.
8. Sauvegarde et export
9. L'utilisateur peut sauvegarder l'état de la partie à tout moment ou exporter le fichier json final une fois le backlog entièrement estimé.

5. Conclusion

Ce projet nous a permis de mettre en pratique concrètement **les notions vues en cours de conception agile**, tout en développant une **application complète et collaborative**. Il nous a permis d'améliorer nos compétences aussi bien sur le plan technique que sur l'organisation du travail en binôme.

Tout au long du projet, nous avons appris de nouvelles notions, notamment **la mise en place de l'intégration continue, l'écriture de tests unitaires et la génération automatique de documentation**, des aspects que nous n'avions pas encore l'habitude d'utiliser dans nos projets précédents.

Nous avons rencontré plusieurs difficultés, notamment au début du projet avec notre premier dépôt GitHub, où les **commits** ne fonctionnaient pas correctement. Face à ces contraintes et au temps limité, nous avons su nous adapter en utilisant **CodeSandbox** comme plateforme de collaboration. Cet outil nous a permis de travailler ensemble sur le même environnement, sans configuration complexe, et d'avancer efficacement sur le projet, comme si nous travaillions directement sur **GitHub**.

Cependant, CodeSandbox présente aussi **certaines limitations**, notamment le passage au mode payant avec un **nombre de crédits restreint**.

Tout au long du projet, nous avons également utilisé **ChatGPT** comme **assistant**, principalement pour nous aider lorsque nous étions bloquées, par exemple pour **comprendre l'utilisation de GitHub avec CodeSandbox, la mise en place des tests unitaires ou la génération de la documentation**.

Enfin, ce projet s'inscrit dans la continuité du projet réalisé en Licence 2, notamment avec **React**, ce qui nous a permis de démarrer avec **une base connue** tout en approfondissant nos compétences. Malgré les contraintes de temps et les difficultés rencontrées, nous avons réussi à mener le projet à terme **en trouvant des solutions adaptées et en respectant les objectifs fixés**.