Rapport du Projet: Application de Planning Poker

1. Présentation Globale du Projet

Le projet **Planning Poker** est une application collaborative qui facilite l'estimation des efforts nécessaires pour réaliser les tâches d'un projet en utilisant une méthode de vote simple et efficace. L'objectif principal est de permettre à plusieurs participants (développeurs, chefs de projet, etc.) de voter sur la difficulté des tâches dans un backlog, en respectant des modes de jeu variés :

- **Mode strict** (unanimité obligatoire),
- **Moyenne** (calcul de la moyenne des votes),
- **Médiane** (valeur médiane des votes),
- **Majorité** (relative ou absolue).

L'application inclut :

- Une interface utilisateur intuitive,
- Une gestion en temps réel grâce à **Socket.IO**,
- La possibilité de charger et sauvegarder des données de backlog sous forme de fichiers JSON.

2. Motivation des Choix:

Choix Techniques :

- Node.js et Socket.IO : Ces technologies ont été choisies pour gérer la communication en temps réel, garantissant une synchronisation fluide entre les participants.
- HTML/CSS et JavaScript : Fournissent une interface utilisateur réactive et simple à naviguer.
- JSON : Format utilisé pour importer/exporter les tâches, en raison de sa simplicité et de sa compatibilité avec les outils modernes.

Choix Fonctionnels :

- Modes de jeu variés : Pour répondre aux besoins différents des équipes et projets.
- Reprise de partie : Enregistrement et chargement de l’état du backlog permettent de continuer une partie interrompue.
- Votes anonymes : Encourage la transparence et évite les biais liés à l'influence.

3. Explications et Diagrammes

3.1. Diagramme de Cas d'Utilisation Un diagramme UML représentant les principales interactions :

Acteurs :

Joueur

Serveur

Cas d’utilisation :

Créer une partie

Rejoindre une partie

Voter pour une tâche

Consulter les résultats

3.2. Diagramme de Séquence Un exemple de séquence pour le processus de vote :

Un joueur vote en cliquant sur une carte.

Le vote est envoyé au serveur via Socket.IO.

Le serveur vérifie si tous les joueurs ont voté.

Une fois le vote terminé, le serveur calcule le résultat et l’envoie à tous les joueurs.

3.3. Diagramme d’Architecture L'architecture générale comprend :

Frontend : Gère l'interface utilisateur et les interactions.

Backend : Gère la logique métier et la synchronisation.

Base de données temporaire (en mémoire) : Stocke les votes et les joueurs connectés.

Fonctionnalités Principales:

1. Ajout dynamique des utilisateurs.
2. Choix parmi plusieurs modes de jeu (strict, moyenne, médiane, etc.).
3. Gestion du backlog via un fichier JSON.
4. Réinitialisation automatique si les votes ne sont pas unanimes.
5. Affichage en temps réel des résultats de votes.
6. Possibilité de sauvegarder l'état du backlog et de reprendre une partie.
7. Synchronisation entre plusieurs clients via WebSockets.

Flux de Travail:

1. L'administrateur configure le jeu en entrant les détails initiaux (nom des joueurs, backlog).
2. Chaque utilisateur se connecte en utilisant son propre navigateur.
3. Les joueurs votent pour chaque tâche selon les règles du mode choisi.

4. Une fois tous les votes reçus, les résultats sont calculés et affichés.

5. Si les votes ne respectent pas les règles, un nouveau vote est initié.

6. À la fin du backlog, les résultats sont sauvegardés dans un fichier JSON.

Détails Techniques:

1. Frontend: HTML, CSS, JavaScript avec Socket.IO pour la communication en temps réel.

2. Backend: Node.js avec Express et Socket.IO.

3. Sauvegarde des données: Utilisation de fichiers JSON pour stocker les résultats et l'état.

4. Gestion des erreurs: Validation des votes, synchronisation des joueurs.

Conclusion:

Ce projet répond aux besoins fondamentaux de collaboration dans l'estimation de tâches en équipe. Grâce à l'utilisation de technologies modernes et une conception modulaire, l'application est extensible et facile à utiliser.

Hossein ABDOLI 2247198

Mohammad Moloudi 2240890