Rapport de Projet : Application Planning Poker Collaboratif

Réalisation par : Fatma Gamra, Wejdene Helal

Introduction

Pour ce projet, nous avons travaillé en binôme sur la réalisation d'une application web nommée Planning Poker Collaboratif, dans le cadre de notre formation. L'objectif principal était de créer un outil permettant d'estimer des tâches de façon collaborative en utilisant la méthode Agile du Planning Poker.

Le projet visait à mettre en pratique nos compétences en développement front-end moderne, en gestion de projet et en collaboration. Dans ce rapport, nous détaillons toutes les étapes de notre réalisation, les choix techniques, les difficultés rencontrées, ainsi que les améliorations possibles pour l'avenir.

1. Réalisation et exécution technique

1.1. Qualité et structure du code

Nous avons mis un point d'honneur à écrire un code propre et lisible. Les composants React sont modulaires et réutilisables, et nous avons utilisé des noms explicites pour les variables et fonctions afin de faciliter la compréhension. La logique est séparée en différentes couches pour éviter toute confusion et faciliter la maintenance future.

L'architecture globale suit les standards React/Vite : - `main.jsx` initialise l'application. - Chaque fonctionnalité est encapsulée dans un composant autonome. - L'arborescence est claire pour qu'un autre développeur puisse comprendre rapidement le projet.

1.2. Interface utilisateur et fonctionnalités

L'utilisateur peut créer ou rejoindre une session, choisir ses cartes pour estimer une tâche et voir les résultats en temps réel. Deux modes ont été implémentés : - Mode individuel pour s'entraîner seul. - Mode collaboratif pour estimer en groupe.

Pour améliorer l'expérience utilisateur, nous avons travaillé sur l'ergonomie de l'interface, le placement des boutons, la lisibilité des cartes, et la clarté des résultats. Nous avons également ajouté des animations légères pour rendre la révélation des votes plus dynamique.

1.3. Gestion des données

Firebase a été utilisé pour stocker les données et gérer la synchronisation : - Persistance : les sessions et votes sont conservés même si l'utilisateur ferme le navigateur. - Temps réel : les votes s'affichent instantanément pour tous.

Nous avons aussi mis en place des règles pour garantir l'intégrité des données et éviter les conflits lors des mises à jour simultanées par plusieurs participants.

1. Architecture, modélisation et qualité

2.1. Choix technologiques

- React pour l'interface réactive et modulable.
- Vite pour un environnement de développement rapide.
- Firebase pour la persistance et la synchronisation en temps réel.

Cette combinaison nous a permis de livrer rapidement un MVP fonctionnel et scalable.

2.2. Architecture

L'application repose sur une architecture front-end first : toute la logique est côté client. Firebase agit comme service externe pour les données, simplifiant le déploiement et séparant clairement interface et données. Nous avons structuré les composants en plusieurs catégories : - Composants de navigation - Composants de session - Composants de vote - Composants d'affichage des résultats

Chaque composant est autonome, réutilisable et facile à tester.

2.3. Documentation

La documentation est essentielle pour la maintenabilité : - README.md : description complète, technologies et instructions pour lancer l'application. - Commentaires dans le code : explications pour les parties complexes et choix techniques. - Structure des fichiers explicite : facilite la compréhension du flux de l'application.

2.4. Intégration continue

GitHub Actions a été utilisé pour vérifier automatiquement l'installation, la compilation et l'intégrité du code à chaque push. Cela assure que la branche principale reste stable et déployable, et réduit le risque de régressions.

2.5. Pistes d'amélioration

Pour rendre le projet plus robuste et complet, nous envisageons : - L'ajout de tests unitaires et fonctionnels. - La génération automatique de la documentation technique avec JSDoc. - La création d'un backend dédié pour gérer l'authentification des utilisateurs et l'historique des sessions.

Nous pourrions également améliorer l'UI avec plus de retours visuels, comme l'indication de l'état des votes en cours, et optimiser la performance pour un grand nombre de participants.

1. Difficultés rencontrées et solutions

Pendant le développement, nous avons rencontré plusieurs défis : - Synchronisation en temps réel des votes : nous avons résolu ce problème en utilisant les fonctionnalités de Firebase Realtime Database. - Gestion des sessions multiples : nous avons implémenté une logique côté client pour gérer les sessions actives et éviter les collisions. - Structuration du code pour rester modulable et lisible : nous avons régulièrement refactoré les composants et utilisé des conventions strictes de nommage.

1. Conclusion

Ce projet a permis de mettre en pratique des compétences clés : développement front-end avec React et Vite, synchronisation temps réel avec Firebase, bonnes pratiques de code et intégration continue.

L’application finale est fonctionnelle, intuitive et prête à évoluer. Les prochaines étapes incluent le développement d’un backend complet et des tests automatisés. Nous avons aussi amélioré notre collaboration en binôme et appris à gérer un projet de manière structurée et maintenable.