Université Lumière - Lyon 2 Campus Porte des Alpes
Master informatique

Rapport du projet CAPI
“ Pocker Planning Project ”

Module : Conception agile de projets informatiques

41LIAA02

Etudiants :
SHOUT Lina Rayane
ABID Abdeldjebbar

Année : 2024 - 2025
Enseignant chargé du module :
Valentin Lachand Pascal

..

Table des matières

.Chapitre 1

Présentation du sujet

1.1 Introduction

Le Planning Poker est une méthode agile utilisée par les équipes de développement pour estimer les efforts nécessaires à la réalisation de différentes tâches ou fonctionnalités dans un projet. Cette technique collaborative permet de faciliter la discussion et d'arriver à un consensus au sein de l'équipe.

1.1.1 Objectif

Ce projet vise à développer une application web de Planning Poker qui offre une plateforme simple et intuitive aux équipes de développement. Elle permettra aux utilisateurs de participer à des sessions de vote pour estimer les efforts nécessaires sur des fonctionnalités, tout en garantissant une traçabilité des résultats et une flexibilité dans les modes de jeu proposés (par exemple, strict, moyenne, ou médiane).

1.1.2 Solutions proposées

Pour atteindre cet objectif, l'application intègre les fonctionnalités suivantes :
— **Gestion des participants** : ajout et gestion des rôles (administrateurs et joueurs).
— **Gestion des parties** : création, configuration et suivi des sessions de Planning Poker.
— **Système de vote** : chaque joueur peut voter sur les fonctionnalités, et les résultats sont calculés en fonction des règles choisies.
— **Sauvegarde et reprise** : possibilité de sauvegarder l'état des parties en cours et de les reprendre ultérieurement.
— **Génération de rapports** : exportation des résultats des votes sous forme de fichier JSON, incluant les estimations des tâches validées.

Cette approche garantit que l'application répond aux besoins des équipes tout en restant adaptable et évolutive pour des améliorations futures. Ce rapport détaille la conception, le développement et la mise en œuvre de cette application.

1.2 Conception de l'Application

Le développement de l'application de Planning Poker a été guidé par des principes de conception clairs visant à fournir une expérience utilisateur fluide et efficace tout en respectant les règles du Planning Poker. La conception a été divisée en plusieurs phases clés, incluant la modélisation des données, la définition des interactions utilisateur et la conception de l'interface

1.2.1 Construction des Diagrammes

Dans le cadre de ce projet, nous avons utilisé PlantUML, un outil de création de diagrammes basé sur un langage de description textuelle

.

Diagramme de Cas d'Utilisation PlantUML : Ce diagramme illustre les interactions entre les utilisateurs (participants et administrateurs) et le système :

```
@startuml
left to right direction
actor Participant
actor Administrateur

rectangle PlanningPoker {
  Participant -- (Rejoindre une Partie)
  Participant -- (Voter sur Fonctionnalité)
  Participant -- (Voir Résultats)
  Participant -- (Reprendre Partie)

  Administrateur -- (Créer Partie)
  Administrateur -- (Gérer Fonctionnalités)
  Administrateur -- (Sauvegarder Partie)
}

@enduml
```

FIGURE 1.1 – Diagramme de Cas d'Utilisation PlantUML

.**Diagramme de Cas d'Utilisation :**

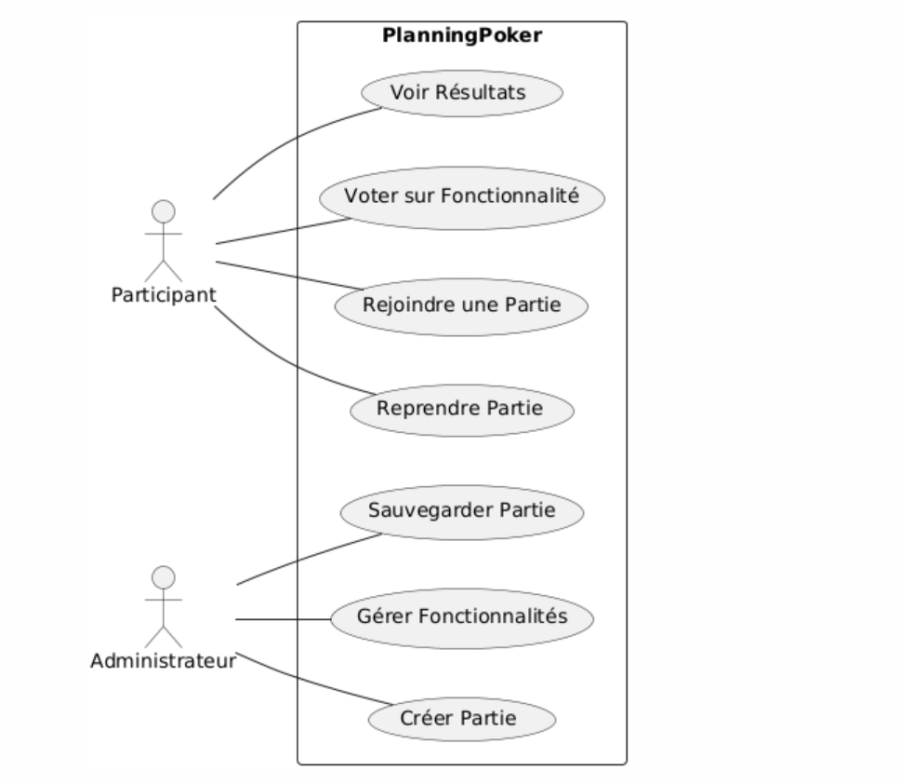

FIGURE 1.2 – Diagramme de Cas d'Utilisation

.

Diagramme de Classes PlantUML : Ce diagramme représente les modèles et leurs relations dans L'application de planning pocker : :

```
@startuml
class Partie {
  - nom: string
  - mode_jeu: string
  - statut: string
  -- sauvegarderBacklog()
  -- sauvegarderEtatPartie()
}

class Participant {
  - pseudo: string
  - est_admin: boolean
}

class Fonctionnalite {
  - name: string
  - description: string
  - valide: boolean
  - difficulte: float
}

class Vote {
  - vote: int
  -- estUnanime(): boolean
}
```

FIGURE 1.3 – Diagramme de Classes PlantUML

Diagramme de Classes :

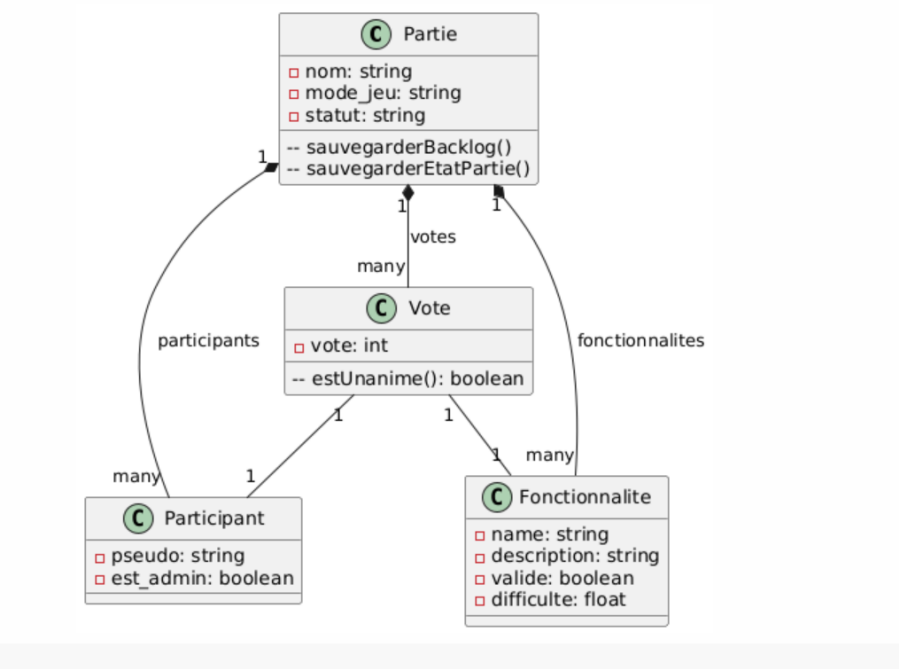

FIGURE 1.4 – Diagramme de Classes

1.3 Développement

1.3.1 Architecture

Le code de l'application est organisé en modules selon les principes MVC, où chaque modèle (Modèle), vue (Vue), et contrôleur (Contrôleur) est clairement défini

- **Modèles** : Les classes de modèles en Django représentent la structure de la base de données. Par exemple, les modèles Participant, Partie, et Fonctionnalité encapsulent les données et les comportements relatifs aux utilisateurs, aux jeux, et aux éléments à estimer.
- **Vues** : Les vues gèrent la logique de présentation et d'interaction avec l'utilisateur. Les vues comme creer partie et liste partie traitent les requêtes des utilisateurs et retournent des réponses appropriées.
- **Contrôleurs** : Bien que Django n'ait pas de contrôleurs explicites comme dans certains autres frameworks MVC, les "views" en Django agissent comme des contrôleurs, orchestrant la logique métier et les interactions modèle-vue.

1.3.2 Langage

Python a été choisi pour le backend en raison de sa simplicité et de sa popularité dans le développement web. Ses avantages incluent :

Une courbe d'apprentissage rapide, utile pour les projets avec des délais serrés. Une vaste communauté offrant des bibliothèques comme Django et Django REST Framework. Sa compatibilité avec des outils de gestion de données comme PostgreSQL.

1.3.3 Classes et Modélisation

La conception des classes a été guidée par les besoins métier, en suivant une approche orientée objet. Les classes principales incluent :

Participant : Représente un joueur ou un administrateur avec des informations comme le pseudo et le rôle.
Partie : Contient les détails des sessions de Planning Poker, comme les participants, le mode de jeu, et l'état de la session.
Fonctionnalité : Gère les tâches ou fonctionnalités à estimer, avec des informations comme la description et le statut (validée ou non).
Vote : Enregistre les votes soumis par les participants pour chaque fonctionnalité.
Ces classes sont reliées de manière logique, ce qui reflète les relations réelles entre les entités dans une session de Planning Poker. Cette approche garantit une modularité et une extensibilité du système.

1.3.4 Choix des technologies

1.3.4.1 Environnement de développement

- **Backend** : Django 5.1.4 et Django REST Framework pour les services API.
- **Frontend** : React pour une interface utilisateur réactive.
- **Base de données** : PostgreSQL pour la gestion de données robuste.
- **Outils de développement** : Docker pour la conteneurisation et GitHub pour le contrôle de version et l'intégration continue.

1.4 Interface Utilisateur

L'interface utilisateur de l'application Planning Poker a été conçue pour offrir une expérience fluide et intuitive, avec une organisation claire des différentes fonctionnalités. Voici une description des principales sections et fonctionnalités.

.

1.4.1 Menu Principal

La page d'accueil offre un accès rapide à deux sections principales :

- **Gestion des Participants** : Options pour créer un participant et lister les participants existants.
- **Gestion des Parties** : Options pour créer une nouvelle partie ou afficher la liste des parties existantes.

FIGURE 1.5 – Menu Principal de l'application.

1.4.2 Création de Participants

Cette interface permet d'ajouter de nouveaux participants au système. Chaque participant peut se voir attribuer un rôle d'administrateur ou non grâce à une simple case à cocher.

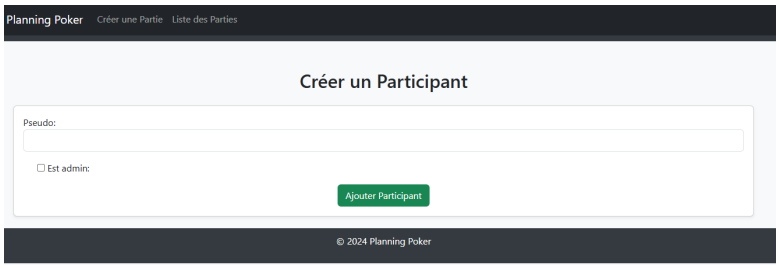

FIGURE 1.6 – Interface pour la création d'un participant.

1.4.3 Liste des Participants

Une liste détaillée des participants inscrits est affichée. Les administrateurs sont identifiés par un badge "Admin", facilitant ainsi leur identification.

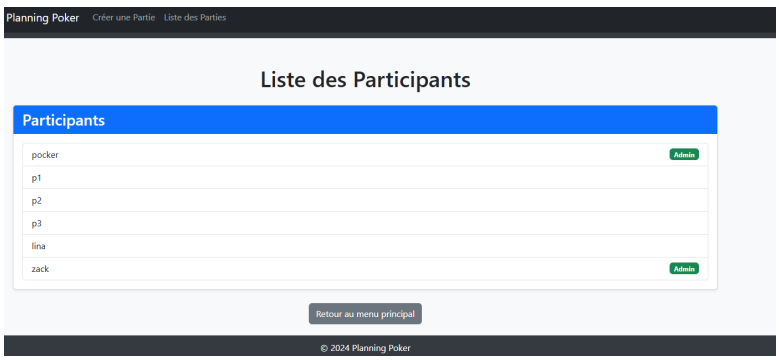

FIGURE 1.7 – Liste des participants avec identification des administrateurs.

1.4.4 Création d'une Partie

L'interface de création de partie permet de configurer rapidement une session. Les options incluent :
— Le nom de la partie.
— Le mode de jeu (ex. strict).
— La sélection d'un administrateur et des participants.

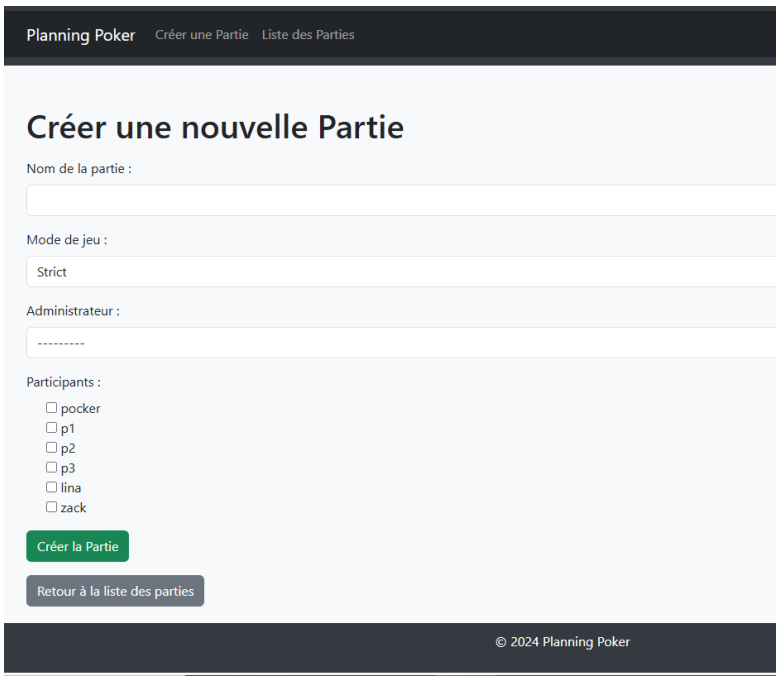

FIGURE 1.8 – Interface pour la création d'une nouvelle partie.

1.4.5 Liste des Parties

Cette section affiche toutes les parties créées, avec des options pour :
— Lancer une partie.
— Voir les détails d'une partie spécifique.

FIGURE 1.9 – Liste des parties avec options pour lancer ou voir les détails.

.

1.4.6 Vote sur une Fonctionnalité

Une interface dédiée au vote permet aux participants de sélectionner une carte correspondant à leur estimation pour une fonctionnalité donnée. Les participants peuvent également utiliser des cartes spéciales comme "Café" ou " ?" pour indiquer une pause ou une incertitude.

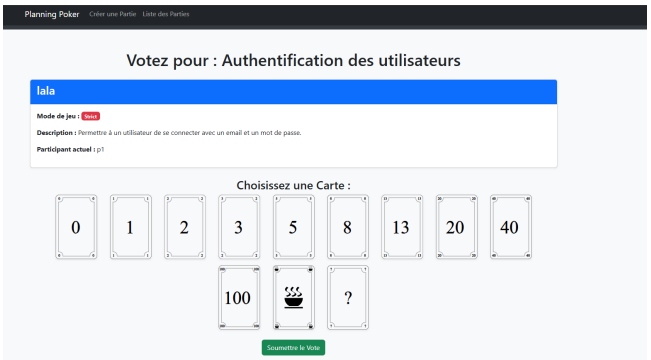

FIGURE 1.10 – Interface de vote sur une fonctionnalité.

1.4.7 Détails de la Partie

Les détails d'une partie incluent :
— Les participants de la partie.
— Les fonctionnalités en cours avec leur statut (validé ou non).
— Les votes soumis pour chaque fonctionnalité.
Cette page fournit une vue d'ensemble claire de l'état actuel d'une partie et de ses résultats.

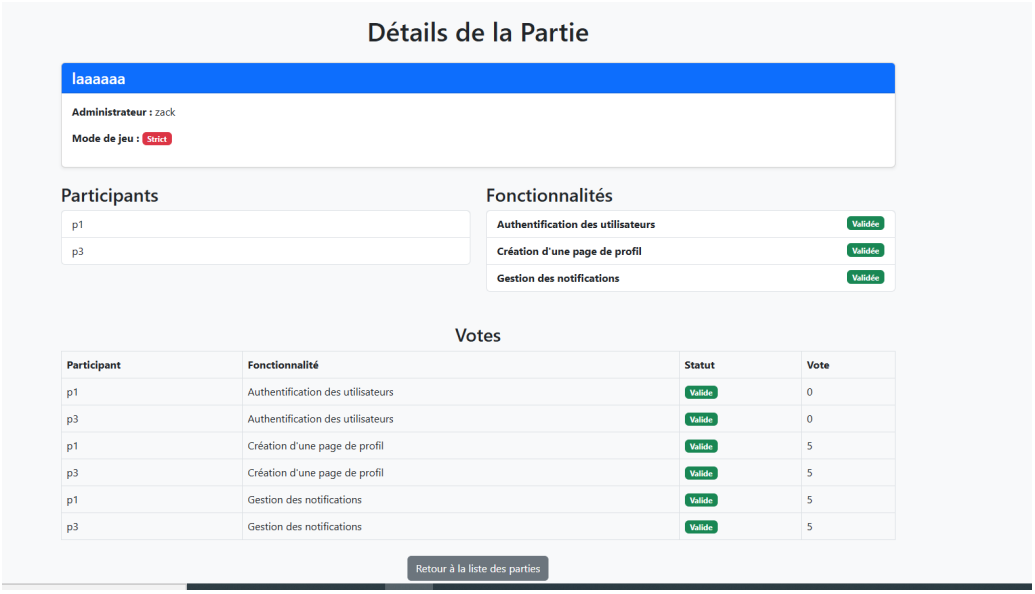

FIGURE 1.11 – Détails d'une partie avec les participants, fonctionnalités et votes.

1.5 Intégration Continue

L'intégration continue (**CI**) a été mise en place à l'aide de **GitHub Actions**. Elle permet d'automatiser les tests et les vérifications de qualité du code à chaque modification apportée au dépôt. Voici les étapes suivies pour configurer ce workflow :

1.5.1 Mise en place des workflows

— Un fichier `ci.yml` a été créé dans le dossier `.github/workflows`, définissant les étapes du pipeline.
— Chaque *commit* déclenche automatiquement les tests unitaires et d'intégration.

1.5.2 Étapes du pipeline

Le pipeline de CI se compose des étapes suivantes :
— **Installation des dépendances** : Les bibliothèques nécessaires, mentionnées dans le fichier `requirements.txt`, sont installées.
— **Exécution des tests** : Tous les tests définis dans le projet sont exécutés à l'aide de l'outil `pytest`.
— **Linting** : Des outils comme `flake8` sont utilisés pour garantir la qualité et la conformité du code.
— **Déploiement** (optionnel) : Si tous les tests réussissent, le code peut être déployé automatiquement sur un serveur distant.

1.5.3 Bénéfices

— **Détection rapide des erreurs** : L'automatisation des tests permet d'identifier rapidement les problèmes dans le code.
— **Qualité du code maintenue** : Les tests et le linting garantissent que le code reste de haute qualité avant toute fusion dans la branche principale.

1.6 Tests et Validation

Pour garantir la fiabilité et la robustesse de l'application, une stratégie de test rigoureuse a été mise en place en suivant les pratiques standards de développement logiciel. Les tests ont été réalisés en deux étapes principales : tests unitaires et tests d'intégration, en utilisant l'outil pytest pour Django.

1.6.1 Tests Unitaires

- Chaque composant principal (modèles, vues, formulaires) a été isolé et testé individuellement. Les tests unitaires ont permis de valider les comportements des modèles, notamment : La création de participants, de parties, et de fonctionnalités. La gestion des votes et la validation des fonctionnalités. Les méthodes personnalisées comme la sauvegarde d'état et le calcul des moyennes. Des cas de test spécifiques ont été écrits pour vérifier les erreurs attendues, comme des entrées invalides dans les formulaires ou des conflits de pseudo lors de la création de participants.

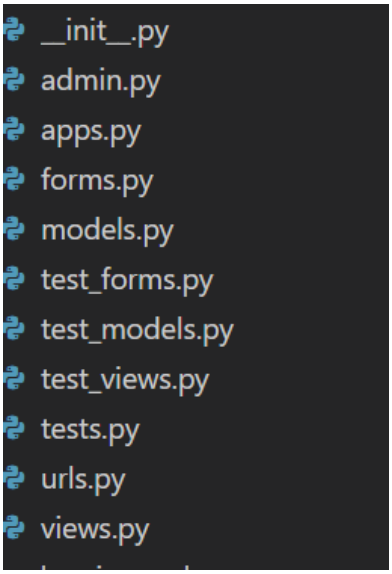

FIGURE 1.12 – tests

1.6.2 Tests d'Intégration

Ces tests ont été utilisés pour valider les interactions entre les différents composants de l'application : La gestion de flux utilisateur comme la création de parties, l'ajout de participants, et la reprise de parties existantes. La validation des routes principales, vérifiant que chaque URL retourne le contenu attendu. L'intégration entre les modèles et les vues, notamment pour la persistance des données et la gestion des relations (participants, fonctionnalités, votes). L'objectif principal était de s'assurer que toutes les fonctionnalités de l'application fonctionnent correctement dans un contexte réel.

1.6.3 Résultats des Tests

Un total de 18 tests unitaires et d'intégration ont été réalisés pour valider le bon fonctionnement de l'application. Voici un résumé des résultats :

Les principaux cas d'utilisation, comme la création d'une partie, l'ajout de participants, la gestion des votes, et la sauvegarde d'état, ont tous été validés avec succès.
Tous les modèles ont été testés pour garantir leur intégrité, y compris la gestion des relations entre les participants, les fonctionnalités, et les parties.

1.7 Génération de documentation avec Doxygen

L'utilisation de Doxygen dans ce projet joue un rôle essentiel en assurant une documentation technique claire, organisée et accessible. Doxygen permet de générer automatiquement une documentation à partir des commentaires du code, ce qui réduit les efforts manuels et garantit que la documentation reste synchronisée avec l'évolution du projet. Cela est particulièrement utile pour un projet collaboratif, où de nouveaux contributeurs ou développeurs peuvent rapidement comprendre la structure, les fonctionnalités et les dépendances du code. En fournissant des détails sur les classes, fonctions et relations, Doxygen améliore la lisibilité et la maintenabilité du projet, tout en facilitant les futures extensions

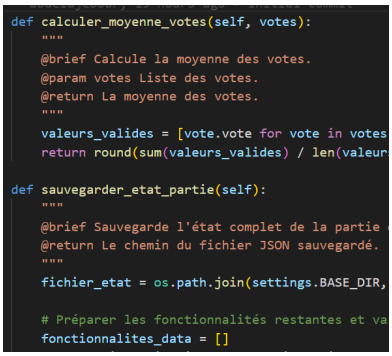

FIGURE 1.13 – Documentation avec Doxygen

1.8 Utilisation de GitHub pour le Projet

1.8.1 Introduction à GitHub

GitHub a été utilisé comme plateforme principale pour la gestion de version et la collaboration sur ce projet. En tant qu'outil de contrôle de version basé sur Git, GitHub permet de suivre les modifications apportées au code source, de collaborer avec d'autres développeurs et d'assurer une intégration continue.

1.8.2 Organisation du Dépôt

Le dépôt GitHub du projet a été structuré pour faciliter la gestion des fichiers et la navigation :

- **Code source** : Tous les fichiers Python liés à Django, y compris les modèles, vues, et templates, sont organisés dans des répertoires logiques.
- **Documentation** : Une documentation détaillée, y compris le fichier `README.md`, offre des instructions pour installer et utiliser le projet.
- **Dépendances** : Le fichier `requirements.txt` liste toutes les dépendances nécessaires à l'exécution de l'application.
- **Scripts de configuration** : Des scripts pour le déploiement et la configuration de l'environnement de développement, comme les fichiers `Dockerfile` ou de configuration pour GitHub Actions.

1.8.3 Accès au Dépôt

Le dépôt GitHub peut être consulté via l'URL suivante :

`https://github.com/Abdeldjebbar10/POKER`$_{PROJECT}$

Les utilisateurs peuvent cloner le dépôt en utilisant la commande suivante dans leur terminal :

```
git clone https://github.com/Abdeldjebbar10/POKER_PROJECT.git
```

.

1.9 Conclusion

Ce projet de développement d'une application web de Planning Poker a permis de répondre aux besoins des équipes de développement en fournissant un outil intuitif et efficace pour estimer les efforts nécessaires à la réalisation de tâches ou fonctionnalités. En s'appuyant sur l'architecture **Model-View-Controller (MVC)** et le framework **Django**, l'application offre une structure robuste et modulable qui facilite la maintenance et l'évolution.

La réalisation du projet a impliqué plusieurs étapes importantes, notamment la conception de l'application à l'aide de diagrammes UML, la mise en œuvre des fonctionnalités principales telles que la gestion des participants, des parties et des votes, ainsi que l'intégration d'un système de sauvegarde et reprise pour garantir une flexibilité optimale. Le choix des outils comme **GitHub** pour la gestion de version et l'intégration continue a permis de maintenir un développement structuré et collaboratif.

Au terme de ce projet, l'application atteint son objectif principal en fournissant une plateforme complète pour organiser et gérer des sessions de Planning Poker. Les tests réalisés ont démontré la fiabilité du système et la satisfaction des exigences fonctionnelles.

Cependant, des améliorations restent possibles, notamment :

— l'ajout de fonctionnalités supplémentaires, telles qu'un espace de discussion en temps réel pour les participants ;
— une personnalisation des modes de jeu ;
— une optimisation des performances pour les grandes équipes.

En conclusion, ce projet constitue une base solide pour un outil collaboratif qui peut être enrichi et étendu à l'avenir. Il illustre également l'importance d'une conception bien pensée et d'un développement structuré pour garantir le succès d'une application logicielle.