

Hugo MAURIN

Lina RHIM

Rapport de Projet Agile : Application Planning Poker

Table des matières

1. Introduction	3
2. Objectifs et Contexte du Projet	3
2.1 Objectifs	3
2.2 Pourquoi utiliser Scrum et Kanban ?	3
3. Organisation Agile et Pair Programming	4
3.1 Pair Programming	4
3.2 Gestion Scrum et Suivi Kanban	4
4. Analyse des Versions et Fonctionnalités	5
4.1 Version 1 : Implémentation de la Page de Login	5
4.2 Version 2 : Ajout du Product Owner (PO)	5
4.3 Version 3 : Intégration du Scrum Master (SM)	6
4.5 Diagramme de Classe	8
5. Implémentation et Pipeline CI/CD	10
5.1 Structure Modulaire du Code	10
5.2 Pipeline CI/CD	11
6. Validation et Tests	13
6.1 Tests Unitaires	13
6.2 Documentation Générée	14
7. Interface Utilisateur	15
7.1 Éléments principaux de l'interface :	15
7.2 Cas d'utilisation :	20
8. Conclusion	22

1. Introduction

Dans le cadre de ce projet, nous avons développé une application Planning Poker, un outil interactif conçu pour aider les équipes agiles à estimer les efforts nécessaires pour réaliser les tâches d'un backlog. L'application est conçue pour être utilisée sur un seul ordinateur, où les membres de l'équipe se connectent à tour de rôle selon leur rôle (PO, SM, ou votant). En respectant les bonnes pratiques agiles et Scrum, notre équipe de deux membres a mis en œuvre une méthodologie structurée pour produire une solution fiable, évolutive et adaptée aux besoins des utilisateurs.

Le développement s'est déroulé en trois versions incrémentales :

1. **Version 1** : Mise en place de la page de connexion (login).
2. **Version 2** : Ajout des fonctionnalités liées au Product Owner (PO).
3. **Version 3** : Introduction des responsabilités du Scrum Master (SM) et des états des votes.

Cette approche incrémentale nous a permis de valider progressivement les fonctionnalités, tout en assurant une qualité constante grâce à des tests automatisés, un suivi Kanban et l'utilisation de pair programming.

Ensuite nous avons mergé notre dernière version 3 à la branche main.

Nous avons utilisé chatgpt pour nous aider à réaliser le code pour les tests, la documentation Doxygen et quelques implémentations de code dans notre application.

2. Objectifs et Contexte du Projet

2.1 Objectifs

L'objectif principal était de créer une application conviviale et robuste permettant :

- Une connexion avec nom utilisateur pour débiter les sessions de Planning Poker.
- La gestion des parties par le Product Owner, responsable de définir les fonctionnalités.
- Une supervision de la réunion par le Scrum Master, qui valide les résultats des votes.
- La sauvegarde des parties pour reprise ou analyse ultérieure via des fichiers JSON (carte café)

2.2 Pourquoi utiliser Scrum et Kanban ?

Nous avons choisi une méthodologie Scrum pour structurer le développement en sprints, permettant de livrer un produit fonctionnel après chaque itération. Le tableau Kanban a complété cette approche en offrant une vue claire et visuelle des tâches à réaliser, en cours, et terminées. Ces deux outils combinés nous ont permis :

- **D'assurer une transparence** dans l'avancement du projet.
- **De rester flexibles** face aux imprévus.
- **De prioriser efficacement** les fonctionnalités critiques.

3. Organisation Agile et Pair Programming

3.1 Pair Programming

Le développement a été réalisé en pair programming, ce qui a permis :

- Une double vérification immédiate du code, réduisant les erreurs.
- Une meilleure compréhension partagée des fonctionnalités développées.
- Une optimisation des choix techniques grâce à une collaboration constante entre les deux développeurs.

Nous alternions entre deux rôles :

1. **Driver** : Responsable d'écrire le code en suivant les tâches du tableau Kanban.
2. **Observer** : Analyse et valide le code, tout en anticipant les problèmes potentiels.

3.2 Gestion Scrum et Suivi Kanban

Le projet a été découpé en trois sprints, chacun correspondant à une version majeure :

1. **Sprint 1 (Version 1)** : Mise en place de la page de login.
2. **Sprint 2 (Version 2)** : Ajout du rôle du Product Owner et gestion du backlog.
3. **Sprint 3 (Version 3)** : Intégration du Scrum Master et gestion avancée des votes.

Le tableau Kanban a servi à organiser les tâches comme suit :

- **To Do** : Liste des tâches identifiées (implémenter l'authentification, gérer les votes).
- **In Progress** : Tâches en cours de développement.
- **Review** : Fonctionnalités prêtes à être validées par le PO ou le SM.
- **Done** : Tâches terminées et validées.

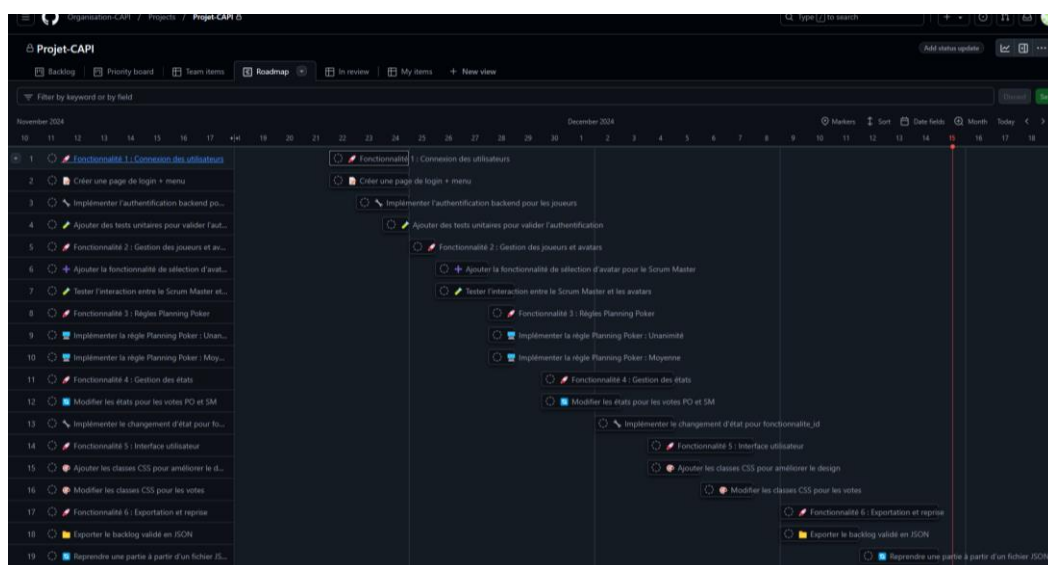


Figure 1: Planification Kanban du développement de l'application

4. Analyse des Versions et Fonctionnalités

4.1 Version 1 : Implémentation de la Page de Login

Fonctionnalités développées :

- Création d'une page de connexion intuitive avec un fichier de template HTML (login.html) situé dans le dossier templates/.
- La page permet aux utilisateurs se connecter s'ils sont autorisés (vérification des pseudos dans le backlog).
- Mise en place d'un système de gestion des erreurs :
 - Affichage d'un message clair en cas d'identifiants invalides.
 - Blocage de l'accès tant que les informations ne sont pas valides.
- Intégration d'une logique backend dans models/app_manager.py, qui traite les identifiants saisis.

Choix techniques :

- Langage Python pour sa simplicité et sa compatibilité avec des frameworks légers pour gérer les fichiers HTML.
- Templates HTML situés dans templates/login.html pour séparer l'interface utilisateur de la logique backend.
- Utilisation d'un fichier JSON (data/backlog.json) pour gérer des données.

Pourquoi commencer par le login ?

La gestion des utilisateurs constitue une étape fondamentale pour l'application. Sans cette fonctionnalité, il aurait été impossible d'attribuer des rôles spécifiques ou de personnaliser les sessions de Planning Poker. Ce choix a également permis de poser les bases de l'application.

4.2 Version 2 : Ajout du Product Owner (PO)

Fonctionnalités développées :

- Ajout d'un rôle spécifique pour le Product Owner (PO) qui permet de gérer les éléments suivants via des interfaces interactives :
 - Définir les joueurs participant à la partie.
 - Configurer le backlog, notamment en ajoutant de nouvelles fonctionnalités à estimer.
 - Choisir les règles du jeu, telles qu'unanimité ou moyenne.
- Sauvegarde des données du backlog (y compris les fonctionnalités ajoutées ou modifiées) dans un fichier JSON accessible dans le dossier data/backlog.json.
- Gestion dynamique des fonctionnalités ajoutées via des interactions avec le backend (app.py et models/app_manager.py).

Fonctionnalité supplémentaire :

- Ajout d'un bouton interactif permettant d'accéder à une vue détaillée du **backlog des fonctionnalités**. Cette fonctionnalité offre :
 - **Aux utilisateurs** : La possibilité de vérifier si leur pseudo est associé à une fonctionnalité lors d'une réunion.
 - **Au Product Owner** : Une vue claire des joueurs associés à chaque fonctionnalité, facilitant le suivi et la gestion des responsabilités au sein du backlog.

Rôle du fichier app.py :

- Gestion des routes et logique principale pour :
 - Charger les templates HTML comme backlog.html ou ajout_fonctionnalite.html.
 - Traiter les formulaires soumis via les interfaces (ex : ajouter une fonctionnalité au backlog).
 - Sauvegarder les modifications apportées par le PO dans le fichier backlog.json.
- Communication entre le backend et le frontend :
 - Les données du backlog sont récupérées depuis le fichier JSON et affichées dynamiquement dans backlog.html.
 - Les formulaires de ajout_fonctionnalite.html et edit_fonctionnalite.html sont traités pour mettre à jour les fonctionnalités existantes.

Templates HTML associés :

- backlog.html : Présente les fonctionnalités actuelles sous forme de liste. Les boutons permettent au PO d'ajouter, modifier ou supprimer des fonctionnalités.
- ajout_fonctionnalite.html : Interface pour saisir les détails d'une nouvelle fonctionnalité.
- edit_fonctionnalite.html : Permet de modifier une fonctionnalité existante.

Pourquoi intégrer le PO à ce stade ?

Le rôle du PO est crucial pour définir les objectifs des sessions et gérer le backlog. Implémenter cette fonctionnalité dans la deuxième version nous a permis de structurer les données et de poser les bases pour les interactions futures, notamment celles impliquant les votes du Scrum Master.

4.3 Version 3 : Intégration du Scrum Master (SM)

Fonctionnalités développées :

- Ajout d'un rôle spécifique pour le Scrum Master (SM), lui permettant de superviser les sessions et valider les fonctionnalités selon les règles choisies (unanimité ou moyenne).
- Gestion des états des fonctionnalités :
 - Chaque fonctionnalité passe par des états comme "Validée", "En cours" ou "À revoir", en fonction des résultats des votes.
- Implémentation de la sauvegarde et de la reprise des sessions :
 - Les parties peuvent être exportées et sauvegardées dans data/backlog.json.

- Possibilité de recharger une session existante pour reprendre le processus à partir des données sauvegardées.
- Le Scrum Master est responsable de déconnecter tous les utilisateurs à la fin de la session via le bouton Logout

Rôle du fichier app.py :

- Gestion des routes et du flux de données pour :
 - Charger et afficher la salle de vote via `salle_de_vote.html`.
 - Gérer les actions du Scrum Master depuis `acces_sm.html`, comme la validation ou la modification d'une fonctionnalité.
 - Sauvegarder les votes et les décisions du SM dans le fichier `backlog.json`.
- **Supervision des votes :**
 - Les règles choisies (unanimité ou moyenne) sont appliquées dynamiquement lors du processus de validation des fonctionnalités.

Choix techniques :

- **Extension de la classe Vote dans `models/app_manager.py` :**
 - Permet de gérer les états des fonctionnalités (validée, en cours etc...).
 - Implémente la logique pour calculer les résultats des votes en fonction des règles définies (moyenne dans notre cas).
- **Fonctionnalités JSON avancées :**
 - Ajout de la gestion des états et des votes directement dans le fichier JSON.
 - Simplifie la reprise des parties en rendant les données accessibles de manière structurée.

Templates HTML associés :

- `acces_sm.html` : Interface dédiée au Scrum Master pour superviser les fonctionnalités et voir leur état (validée ou non).
- `salle_de_vote.html` : Page où les participants peuvent voter sur les fonctionnalités en cours.

Pourquoi terminer par le Scrum Master ?

Le rôle du Scrum Master est axé sur la supervision des sessions de vote et la validation des fonctionnalités. Une fois que les joueurs et le backlog sont définis (versions 1 et 2), il devient logique d'ajouter le SM en tant que dernier rôle pour compléter le flux de l'application. Cette fonctionnalité clôture également le cycle en ajoutant un mécanisme de validation robuste et en garantissant la continuité des parties.

4.5 Diagramme de Classe

Pour illustrer la structure logique de l'application, voici le diagramme de classe représentant les interactions entre les principales entités de l'application. Ce diagramme met en évidence les responsabilités des classes et leurs relations.

Les relations entre ces classes montrent comment **AppManager** utilise et manipule les fonctionnalités pour gérer les sessions de Planning Poker.

Classe Fonctionnalite

Responsabilité : Représente une tâche ou fonctionnalité dans le backlog.

Attributs :

- id, nom, description, priorite, difficulté, etc., définissent les détails de la fonctionnalité.
- participants : Liste des développeurs impliqués dans la fonctionnalité.

Méthodes :

- `to_dict()` : Retourne une représentation sous forme de dictionnaire pour faciliter la sérialisation.
- `__str__()` et `__repr__()` : Fournissent des représentations textuelles utiles pour le débogage et la journalisation.

Classe AppManager

Responsabilité : Gère l'ensemble des fonctionnalités et des participants de l'application.

Attributs :

- backlog : Liste des instances de Fonctionnalite.
- state : Dictionnaire contenant les participants, les indicateurs globaux et le contexte.
- backlog_file : Fichier JSON utilisé pour sauvegarder et charger le backlog.

Méthodes :

- Gestion des fonctionnalités : Ajouter, modifier, supprimer, lister, et récupérer la fonctionnalité prioritaire.
- Gestion des participants : Ajouter, vérifier les doublons, et déconnecter.
- Gestion des votes : Initialiser, enregistrer, valider, révéler, et réinitialiser.
- Sauvegarde/Chargement : Sérialisation et désérialisation des fonctionnalités dans un fichier JSON.
- Indicateurs d'état (state) : Vérifie si l'équipe est complète ou si tous les participants ont voté.

Relation entre les classes

- La classe AppManager utilise une liste d'instances de Fonctionnalite pour gérer les tâches.
- Les participants ne sont pas des instances d'une classe distincte, mais des dictionnaires manipulés au sein de l'attribut state.

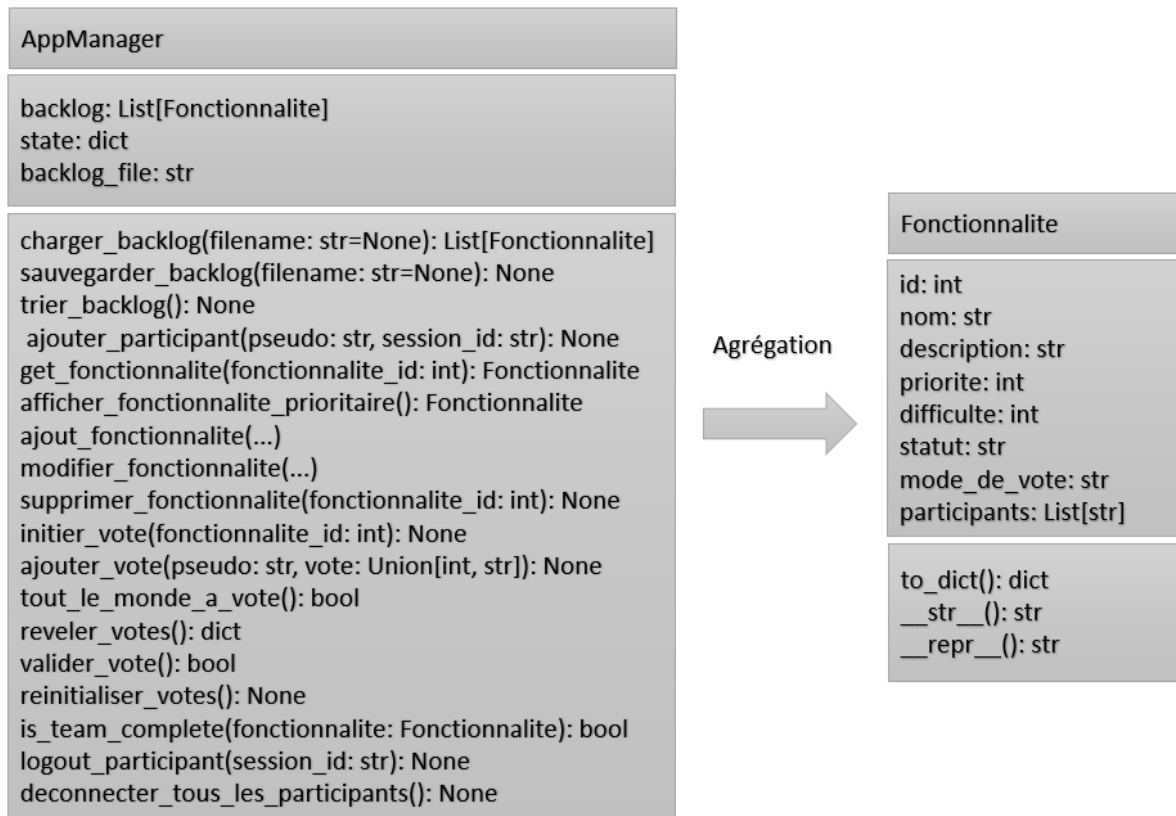


Figure 2: Diagramme de classe

5. Implémentation et Pipeline CI/CD

5.1 Structure Modulaire du Code

La structure du projet suit une logique modulaire pour séparer les responsabilités entre le backend, le frontend et les fichiers statiques :

- **app.py** : Logique principale, gère les interactions entre les classes.

Backend :

- **models/app_manager.py** :
 - Gère la logique principale de l'application, notamment l'authentification, la gestion du backlog et les votes.
 - Contient les fonctions nécessaires à la validation des fonctionnalités par le Scrum Master.
- **models/fonctionnalite.py** :
 - Définit les classes et les méthodes pour représenter les fonctionnalités dans le backlog.

Frontend :

- **Templates HTML** situés dans **templates/** :
 - login.html : Interface de connexion.
 - backlog.html : Gestion du backlog par le Product Owner.
 - ajout_fonctionnalite.html et edit_fonctionnalite.html : Interfaces pour ajouter ou modifier des fonctionnalités.
 - acces_sm.html et salle_de_vote.html : Pages pour le Scrum Master et la gestion des votes.
- **CSS** dans **static/css/** :
 - style_layout.css : Définit les styles généraux des pages.
 - style_salle.css : Ajoute des styles spécifiques à la salle de vote.

Données :

- **data/backlog.json** :
 - Contient les informations sur les fonctionnalités et les états des votes.
 - Sert de stockage persistant et léger pour les parties.

Tests :

- **tests_unitaires/ :**
 - `test_app_manager.py` : Valide les fonctions principales liées à l'authentification, au backlog, et à la gestion des votes.
 - `Test_app.py` : Vérifie les interactions globales de l'application, y compris les routes principales (login, backlog, vote), la gestion des formulaires soumis par les utilisateurs, et la sauvegarde ou reprise des données via le backend.

5.2 Pipeline CI/CD

Pour garantir la qualité et la stabilité du code tout au long du projet, un pipeline d'intégration continue et de déploiement continu (**CI/CD**) a été mis en place à l'aide de **GitHub Actions**. Ce pipeline repose sur des fichiers de configuration YAML, spécifiquement conçus pour automatiser les processus essentiels à chaque mise à jour du code.

Automatisation des processus

Exécution des tests unitaires après chaque push :

Un fichier YAML dédié configure l'exécution des tests unitaires à chaque fois qu'une modification est envoyée au dépôt GitHub (via un push). Ce fichier spécifie les étapes nécessaires pour :

- Installer les dépendances requises.
- Exécuter les tests unitaires situés dans le répertoire **tests_unitaires/**.

Les tests couvrent les fonctionnalités critiques telles que l'authentification, la gestion du backlog, et les votes, garantissant que tout nouveau code respecte les normes de qualité définies.

Si un test échoue, le pipeline arrête son exécution et signale immédiatement l'erreur au contributeur.

Génération de la documentation avec Doxygen :

- Un autre fichier YAML est utilisé pour automatiser la génération de la documentation technique. Ce fichier configure Doxygen pour analyser les commentaires inclus dans le code source et produire une documentation détaillée des classes, méthodes, et fonctions.
- Ce processus assure que la documentation reste constamment à jour avec l'évolution du projet.

Déploiement de la documentation sur GitHub Pages :

- Une fois la documentation générée, un fichier YAML supplémentaire est utilisé pour orchestrer son déploiement automatique sur **GitHub Pages**.
- Ce déploiement permet à l'équipe ou aux évaluateurs d'accéder facilement à une version en ligne de la documentation sans avoir besoin de la générer manuellement.

Impact du pipeline CI/CD

Amélioration de la qualité du code :

- Les tests unitaires sont exécutés automatiquement à chaque push, ce qui aide à identifier et corriger rapidement les bugs ou régressions.
- Cela garantit que chaque mise à jour respecte les normes de qualité définies pour le projet.

Documentation toujours à jour :

- L'automatisation de la génération et du déploiement de la documentation assure qu'elle reflète en permanence l'état actuel du code, même après des modifications importantes.

Collaboration facilitée :

- L'automatisation des processus réduit les tâches répétitives pour les développeurs, leur permettant de se concentrer sur l'ajout de nouvelles fonctionnalités.
- La transparence du pipeline CI/CD renforce la collaboration en rendant visibles les erreurs ou les mises à jour pour toute l'équipe.

6. Validation et Tests

6.1 Tests Unitaires

Les tests unitaires sont un aspect essentiel pour garantir la fiabilité et la stabilité de l'application. Ils ont été mis en place pour valider les fonctionnalités critiques, en utilisant une architecture modulaire qui facilite leur exécution et leur extension.

Nous avons effectués dans la version finale 16 tests pour AppManager.py et 16 tests pour app.py.

Quelques fonctionnalités testées :

1. Login : Validation des identifiants

- Tests pour vérifier que seuls les utilisateurs autorisés peuvent accéder à l'application.
- Simulation de plusieurs scénarios : identifiants corrects, identifiants incorrects, champs vides.
- Les tests garantissent qu'un message d'erreur est renvoyé en cas de problème.

2. Backlog : Gestion des entrées utilisateur

- Validation des fonctionnalités ajoutées ou modifiées dans le backlog par le Product Owner.
- Tests pour s'assurer que les données du backlog sont correctement sauvegardées et rechargées depuis le fichier JSON (data/backlog.json).
- Vérification que les entrées invalides (e.g., champs vides, doublons) sont correctement rejetées.

3. Votes : Vérification des résultats

- Validation des deux modes de jeu implémentés :
 - **Unanimité** : Tous les joueurs doivent être d'accord pour valider une fonctionnalité.
 - **Moyenne** : Le calcul de la moyenne des votes est effectué correctement.
- Tests pour garantir que les résultats sont sauvegardés correctement et que les règles sont respectées.

Les autres tests complémentaires sont également inclus dans le dossier tests_unitaires/, garantissant une couverture étendue de l'ensemble des fonctionnalités du projet.

Mise en place des tests :

- Les tests sont situés dans le dossier **tests_unitaires/**.
- Utilisation de pytest pour automatiser l'exécution et fournir des rapports de couverture.

6.2 Documentation Générée

Documentation technique :

Une documentation technique complète a été générée à l'aide de **Doxygen**. Cette documentation est essentielle pour les développeurs permettant de détailler la structure et le fonctionnement de l'application.

Description des classes et méthodes :

Chaque classe (e.g., PO, Backlog, Vote) est décrite, avec une explication de son rôle et de ses responsabilités.

Les méthodes sont détaillées avec leurs paramètres, leur fonctionnement et leurs cas d'utilisation.

Interactions entre les composants :

Les relations entre le backend (app.py, app_manager.py) et le frontend (fichiers HTML) sont expliquées, avec des diagrammes simplifiés pour représenter les flux de données.

Sa mise en œuvre :

Les commentaires dans le code suivent un format standard compatible avec Doxygen, assurant une génération cohérente.

La documentation est déployée automatiquement via **GitHub Actions** sur GitHub Pages, permettant un accès facile aux développeurs.

Instructions d'installation et d'utilisation (Readme):

Ce fichier permet de détailler la façon d'utiliser l'application :

- Comment cloner le dépôt depuis GitHub.
- Les étapes pour installer les dépendances via un fichier requirements.txt.
- Les commandes pour lancer l'application et les tests.

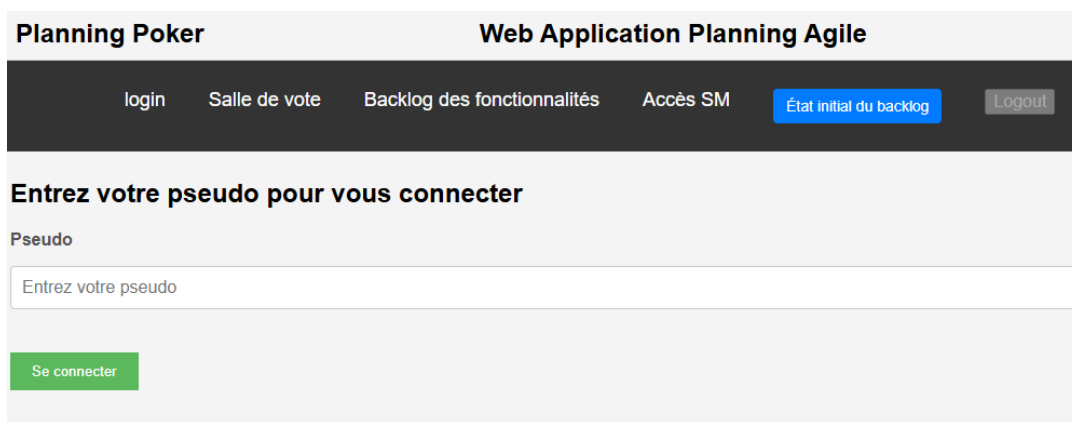
7. Interface Utilisateur

L'interface utilisateur de l'application a été conçue pour offrir une expérience intuitive, adaptée aux rôles spécifiques des utilisateurs. Chaque rôle dispose d'actions dédiées pour garantir un flux structuré et conforme aux principes du Planning Poker.

7.1 Éléments principaux de l'interface :

1. Page de Login :

- Accessible via login.html.
- Chaque utilisateur se connecte séquentiellement en sélectionnant son pseudo
- Les rôles (PO, SM, votants) sont automatiquement attribués en fonction du pseudo



The screenshot shows the login page of the 'Planning Poker' web application. At the top, there is a header with the title 'Planning Poker' and 'Web Application Planning Agile'. Below the header is a navigation bar with links: 'login', 'Salle de vote', 'Backlog des fonctionnalités', 'Accès SM', 'État initial du backlog' (highlighted in blue), and 'Logout'. The main content area has the heading 'Entrez votre pseudo pour vous connecter'. Below this is a label 'Pseudo' and a text input field with the placeholder 'Entrez votre pseudo'. At the bottom of the form is a green button labeled 'Se connecter'.

2. Menu interactif pour les rôles :

Une fois connecté, chaque utilisateur accède à un menu avec des options adaptées à son rôle :

PO (Product Owner) :

- Gère le backlog (ajout, modification ou suppression des fonctionnalités).
- Configure les règles de vote (unanimité, moyenne, etc.).
- Ajoute les participants à chaque tâche.

SM (Scrum Master) :

- Supervise le processus de vote, initie les sessions, et gère les résultats.
- Facilite les discussions en cas de désaccord.
- Valide les votes ou réinitialise les tâches si nécessaire.

Votants :

Consultent la tâche en cours de vote et soumettent leurs estimations via des cartes de vote.



3. Section de vote :

- Accessible via `salle_de_vote.html`.
- Les votants sélectionnent leurs estimations pour les fonctionnalités en cours à partir d'options prédéfinies.
- Le SM supervise les votes et révèle les résultats pour valider ou ajuster les tâches.

Flux fonctionnel de l'application :

1. Connexion :

Les utilisateurs se connectent **l'un après l'autre** en sélectionnant leur pseudo dans la page de login.

La fonction associée à chaque pseudo (PO, SM ou votant) est définie par avance.

2. Gestion du backlog :

Le PO configure les attributs de la fonctionnalité : le nom, la description, la priorité, le statut, le mode de vote et la liste des participants.

Les modifications sont sauvegardées automatiquement dans un fichier JSON.

3. Démarrage du vote :

Le Scrum Master initie les votes pour les tâches définies dans le backlog.

Planning Poker
Web Application Planning Agile

[login](#)
[Salle de vote](#)
[Backlog des fonctionnalités](#)
[Accès SM](#)
[État initial du backlog](#)
[Déconnecter tout le monde](#)

Accès Scrum Master

Bonjour , vous êtes connecté en tant que Scrum Master.

Initier le vote

Participants attendus :

- lina
- hugo

Participants connectés :

- lina
- hugo

Fonctionnalité prioritaire :

- Nom :** Fonctionnalité 5
- Description :** jh
- Mode de vote :** unanimité

Tous les participants attendus sont connectés.

Initier le vote

Actions après le vote

Révéler les votes

Résultat des votes

- lina : 1
- hugo : 1

Valider le vote

Réinitialiser les votes

Vote validé avec succès. Résultat : 1.

Passage à la fonctionnalité suivante : Fonctionnalité 5.

4. Soumission des votes :

Les votants sélectionnent leurs estimations via une interface ergonomique.

Une fois tous les votes soumis, le SM peut révéler les résultats.

5. Gestion des désaccords :

En cas de désaccord, le SM facilite une discussion entre les participants.

Si nécessaire, il réinitialise les votes pour un nouveau tour.

6. Validation des tâches :

Une fois un consensus atteint, le SM valide la tâche.

Le statut de la tâche est mis à jour dans le backlog (e.g., "Validée", "Terminé").

7. Sauvegarde et reprise :

À la fin de la session, l'état du backlog est sauvegardé automatiquement dans un fichier JSON.

Ce fichier peut être chargé ultérieurement pour reprendre une session interrompue.

Planning Poker

Web Application Planning Agile

login

Salle de vote

Backlog des fonctionnalités

Accès SM

État initial du backlog


Logout

Salle de vote

Participants connectés



lina



hugo

Votez pour la fonctionnalité prioritaire :

Participant actif : po

1

2

3

5

8

13

20

40

80

100

?

cafe

L'équipe est au complet ! Vous pouvez commencer le vote.

Voici les détails de la fonctionnalité sélectionnée :

Nom : Fonctionnalité 5

Statut : A faire

Mode de vote : unanime

Détails

Options PO

Ajouter une fonctionnalité

Modifier le backlog

po est maintenant actif.

Planning Poker

Web Application Planning Agile

login

Salle de vote

Backlog des fonctionnalités

Accès SM

État initial du backlog

Logout

Ajouter une fonctionnalité

Nom de la fonctionnalité :

Description :

Priorité :

Difficulté :

Participants ajoutés :

Aucun participant ajouté.

Ajouter un participant :

Entrez le pseudo du participant

Ajouter

Enregistrer la fonctionnalité

Retour

Planning Poker

Web Application Planning Agile

login

Salle de vote

Backlog des fonctionnalités

Accès SM

État initial du backlog

Logout

Backlog des fonctionnalités

Nom	Description	Priorité	Difficulté	Statut	Mode de vote	Participants	Actions
Fonctionnalité 5	jh	5	1	A faire	unanimite	lina hugo	<div>Modifier</div> <div>Supprimer</div>
Fonctionnalité 6	sdscvc xxv	6	20	A faire	moyenne	lina hugo	<div>Modifier</div> <div>Supprimer</div>
Fonctionnalité 8	kjkjkjkj	8	5	A faire	unanimite	lina hugo	<div>Modifier</div> <div>Supprimer</div>
Fonctionnalité 18	fdgf hjhj	18	4	A faire	unanimité	lina hugo	<div>Modifier</div> <div>Supprimer</div>
Fonctionnalité 1	cvcv	1	25	Terminé	unanimite	lina hugo	<div>Modifier</div> <div>Supprimer</div>
Fonctionnalité 3	TEST	3	1	Terminé	moyenne	lina hugo	<div>Modifier</div> <div>Supprimer</div>
Fonctionnalité 4	qssqs	4	8	Terminé	unanimite	lina hugo	<div>Modifier</div> <div>Supprimer</div>
Fonctionnalité 7	test1	7	1	Terminé	unanimite	lina hugo	<div>Modifier</div> <div>Supprimer</div>
Fonctionnalité 13	ffd	13	66	Terminé	moyenne	lina hugo	<div>Modifier</div> <div>Supprimer</div>

Retour

Planning Poker

Web Application Planning Agile

login

Salle de vote

Backlog des fonctionnalités

Accès SM

État initial du backlog

Déconnecter tout le monde

Salle de vote

Participants connectés

lina

hugo

Votez pour la fonctionnalité prioritaire :

Participant actif : sm

1

2

3

5

8

13

20

40

80

100

?

cafe

L'équipe est au complet ! Vous pouvez commencer le vote.

sm est maintenant actif.

po

sm

Voici les détails de la fonctionnalité sélectionnée :

Nom : Fonctionnalité 5

Statut : A faire

Mode de vote : unanime

Détails

Choix techniques :

- **HTML + CSS :**
 - Templates HTML dans templates/ pour structurer les pages.
 - CSS dans static/css/ pour une présentation claire et professionnelle.
- **Backend dynamique :**
 - Gestion des interactions utilisateur et des données via app.py.
- **Accessibilité multiplateforme :**
 - Accessible via un navigateur, sans dépendances spécifiques.

7.2 Cas d'utilisation :

Un cas d'utilisation décrit une interaction entre un utilisateur et un système pour atteindre un objectif spécifique. Dans ce contexte, les fonctionnalités décrites montrent comment les différents utilisateurs (Product Owner, Scrum Master, développeurs) interagissent avec le système pour gérer les réunions Agile.

Pour formaliser cela comme un **cas d'utilisation**, voici les éléments nécessaires :

1. Acteurs

Les rôles des participants au système :

- **Product Owner (PO)** : Responsable du backlog, ajoute/modifie les fonctionnalités et supervise la priorisation.
- **Scrum Master (SM)** : Facilite le processus de vote, initie/réinitialise les votes et valide les décisions.
- **Développeurs (Hugo, Lina)** : Votent sur les fonctionnalités en cours pour définir les priorités.

2. Scénarios de cas d'utilisation

Cas 1 : Connexion des participants

- **Objectif** : Permettre aux utilisateurs de se connecter au système avec un rôle défini.
- **Acteurs impliqués** : PO, SM, Développeurs.
- **Étapes** :
 1. Un utilisateur ouvre la page de connexion.
 2. Il entre un pseudo autorisé (par exemple : "lina").
 3. Le système attribue un rôle et génère une session unique.
 4. L'utilisateur est redirigé vers la page appropriée (salle de vote ou gestion backlog).
- **Exception** : Si le pseudo est non autorisé, un message d'erreur s'affiche.

Cas 2 : Gestion du backlog

- **Objectif** : Gérer les fonctionnalités (ajouter, modifier, supprimer) pour le processus de vote.
- **Acteurs impliqués** : PO.
- **Étapes** :
 1. Le PO se connecte au système.
 2. Il accède à la page du backlog.
 3. Il ajoute une fonctionnalité en remplissant le formulaire requis.
 4. La fonctionnalité est ajoutée et triée par priorité.
- **Exceptions** :
 1. Si les données du formulaire sont invalides, un message d'erreur est affiché.
 2. Les modifications ou suppressions échouent si la fonctionnalité est introuvable.

Cas 3 : Processus de vote

- **Objectif** : Faciliter le vote sur une fonctionnalité prioritaire pour la priorisation ou validation.
- **Acteurs impliqués** : SM, Développeurs.
- **Étapes** :
 1. Le SM initie le vote pour la fonctionnalité prioritaire.
 2. Les développeurs votent en utilisant des cartes numériques.
 3. Le SM peut révéler les votes une fois que tout le monde a voté.
 4. Si les critères du mode de vote sont remplis, la fonctionnalité est validée.
 5. La fonctionnalité est marquée comme "Terminée" et la suivante devient prioritaire.
- **Exceptions** :
 1. Si les votes ne sont pas unanimes, une discussion est initiée.
 2. Si un participant choisit "?", une discussion est nécessaire.

Cas 4 : Suspension de la réunion

- **Objectif** : Suspendre la réunion et sauvegarder l'état en cas de besoin.
- **Acteurs impliqués** : SM, Développeurs.
- **Étapes** :
 1. Tous les participants votent "Café".
 2. Le SM valide la suspension.
 3. L'état du backlog est sauvegardé dans un fichier séparé.
 4. Lors de la reprise, le SM charge l'état sauvegardé.

- **Exceptions :**

1. Si un participant ne vote pas "Café", la suspension n'est pas possible.

Cas 5 : Modes de vote

- **Objectif :** Gérer différentes règles de validation pour les votes.

- **Acteurs impliqués :** SM, Développeurs.

- **Étapes :**

1. Le mode de vote est défini dans la fonctionnalité (Unanimité, Moyenne).
2. En mode "Unanimité" :
 - Tous les votes doivent être identiques.
 - Si ce n'est pas le cas, une discussion est initiée.
3. En mode "Moyenne" :
 - La moyenne des votes est calculée.
 - La carte la plus proche de cette moyenne est choisie.
4. Le SM valide les résultats en fonction des règles.

- **Exceptions :**

1. Si des votes manquent, la validation échoue.

8. Conclusion

Grâce à cette application, les équipes peuvent organiser leurs sessions Planning Poker de manière fluide, en alternant les rôles et en garantissant un contrôle centralisé par le Scrum Master. Cette centralisation, associée à une interface intuitive et à des fonctionnalités adaptées à chaque fonction, permet de renforcer la collaboration et d'assurer une gestion efficace des tâches et des votes dans un cadre structuré et agile.

Le développement de ce projet a été une occasion précieuse de mettre en pratique les principes fondamentaux de l'agilité, notamment la transparence, l'adaptabilité et l'amélioration continue.

En appliquant une méthodologie structurée avec des itérations incrémentales et en utilisant le pair programming, nous avons non seulement atteint nos objectifs techniques, mais aussi assuré une qualité constante à chaque étape.

De plus, la mise en œuvre d'un pipeline CI/CD a permis d'automatiser les tests et la documentation, garantissant ainsi une application robuste et évolutive.

Ces outils ont non seulement accéléré le cycle de développement, mais ont également facilité la collaboration en fournissant des retours rapides et des processus bien définis.

En conclusion, l'application **Planning Poker** répond pleinement aux besoins des utilisateurs en offrant une solution conviviale, fonctionnelle et extensible. Elle illustre parfaitement comment une approche agile, associée à des choix techniques pertinents, peut transformer une idée en un produit concret, fiable et adapté aux exigences du travail en équipe.