Université Lumière Lyon 2
INSTITUT DE LA COMMUNICATION

Rapport de Projet : Application de Planning Poker

Travail élaboré par :

Maram NASR

Skander HAJ MABROUK

ANNÉE UNIVERSITAIRE : 2024 – 2025

Table des matières

Liste des figures

Introduction générale

Dans le cadre de nos études en M1 Informatique, nous avons choisi de développer une application de Planning Poker, un outil collaboratif permettant aux équipes de développement de planifier et d'estimer la complexité des tâches d'un projet. L'objectif principal de ce projet est de mettre en œuvre les concepts étudiés en programmation, gestion de projet et méthodologies agiles, tout en appliquant des techniques de modélisation et d'intégration continue.

Cette application permettra de faciliter le processus de vote et de prise de décision dans un cadre ludique et structuré, en proposant divers modes de jeu et des fonctionnalités additionnelles comme une visualisation graphique des résultats.

Le projet se décompose en trois grandes étapes : la définition des besoins et des spécifications, la conception de l'application, et enfin la phase de développement et de mise en œuvre.

Ce mémoire se termine avec une synthèse des réalisations pour l'application qui expose les valeurs ajoutées du projet ainsi que les perspectives.

Analyse des besoins 1

Introduction

Dans ce chapitre, nous nous intéressons principalement à la première étape de l'approche SCRUM, où il est question de collecter les exigences fonctionnelles et non fonctionnelles.

1.1 Capture des besoins

Dans le domaine du développement collaboratif, l'estimation et la planification des tâches sont des étapes cruciales pour garantir le succès des projets. Le Planning Poker, une méthode agile populaire, s'impose comme un outil essentiel pour estimer les efforts nécessaires à la réalisation de tâches tout en favorisant l'engagement et la participation des membres de l'équipe. Ce projet vise à concevoir et développer une application numérique de Planning Poker, qui centralise et simplifie ce processus dans un environnement interactif, intuitif, et structuré.

La solidité et la réussite de cette application résident dans sa capacité à répondre aux besoins des utilisateurs tout en garantissant une expérience fluide, fiable et évolutive. Ce chapitre aborde la définition des besoins, fonctionnels et non fonctionnels, en identifiant les fonctionnalités principales et les acteurs impliqués dans l'utilisation du système.

1.1.1 Les besoins fonctionnels

Les besoins fonctionnels décrivent les services fondamentaux que l'application doit fournir pour répondre aux attentes des utilisateurs. Ces fonctionnalités reflètent les objectifs du système et permettent de définir les interactions principales entre les différents acteurs et le système.

- **BF1** : Le système doit permettre aux utilisateurs de créer une partie de Planning Poker, en définissant le nombre de joueurs, le mode de la partie et le backlog à évaluer.
- **BF2** : Les utilisateurs doivent pouvoir voter pour chaque tâche selon les modes choisis : mode strict ou moyenne.
- **BF3** : Le système doit collecter et valider les votes en fonction des règles définies, puis afficher les résultats sous forme graphique.
- **BF4** : Le système doit permettre la sauvegarde et la reprise d'une session interrompue.
- **BF5** : Le système doit sauvegarder dans la base de données les résultats des sessions précédentes.

1.1.2 Les besoins non fonctionnels

Au-delà des fonctionnalités essentielles, le système doit respecter un ensemble de critères non fonctionnels garantissant sa qualité et sa fiabilité.

- **BNF1** : Ergonomie – L'application doit proposer une interface conviviale et facile à manipuler, avec un design intuitif et des interactions fluides.
- **BNF2** : Simplicité – Les interfaces utilisateur doivent être compréhensibles, même pour des utilisateurs non techniques.
- **BNF3** : Performance – Les temps de réponse doivent être rapides pour garantir une expérience utilisateur optimale.
- **BNF4** : Scalabilité – Le système doit être extensible pour intégrer de nouvelles fonctionnalités à l'avenir.
- **BNF5** : Disponibilité – L'application doit être accessible à tout moment pour éviter toute interruption du service.

1.1.3 Identification des acteurs

Les acteurs de l'application représentent les entités qui interagissent directement avec le système pour accomplir leurs objectifs.

- **Administrateur** : Responsable de la gestion des paramètres globaux de la partie. Il est un membre de l'équipe qui configure les règles du jeu et peut également voter en tant que joueur.
- **Joueur** : Participe activement aux parties en votant pour les estimations des tâches.

1.2 Conclusion

Ce chapitre a présenté les objectifs et la structure de notre application en définissant les besoins fonctionnels et non fonctionnels, ainsi que les rôles des différents acteurs. Ces éléments constituent la base de conception et de développement de notre système. Dans le prochain chapitre, nous détaillerons les diagrammes de conception nécessaires pour structurer l'application.

Conception 2

Introduction

La phase de conception est une étape essentielle dans le développement d'un système d'information. Elle consiste à modéliser les aspects structurels et comportementaux du projet en utilisant des diagrammes adaptés, qui servent de pont entre les exigences identifiées et le code à développer. Ce chapitre présente la conception de notre application Planning Poker, en détaillant ses principales interactions à travers des diagrammes UML.

L'objectif est de garantir une conception cohérente, robuste et évolutive, tout en facilitant la compréhension et la collaboration entre les membres de l'équipe. Nous aborderons un diagramme de cas d'utilisation global, un diagramme de classe pour illustrer les relations structurelles, et un diagramme de séquence détaillant le déroulement de la fonctionnalité voter.

2.1 Diagrammes de conception

2.1.1 Diagrammes de cas d'utilisation

Le diagramme de cas d'utilisation représente les interactions entre les acteurs et les fonctionnalités principales du système.

Il identifie les cas d'utilisation répondant aux exigences fonctionnelles définies au chapitre précédent, comme illustré çi-dessous.

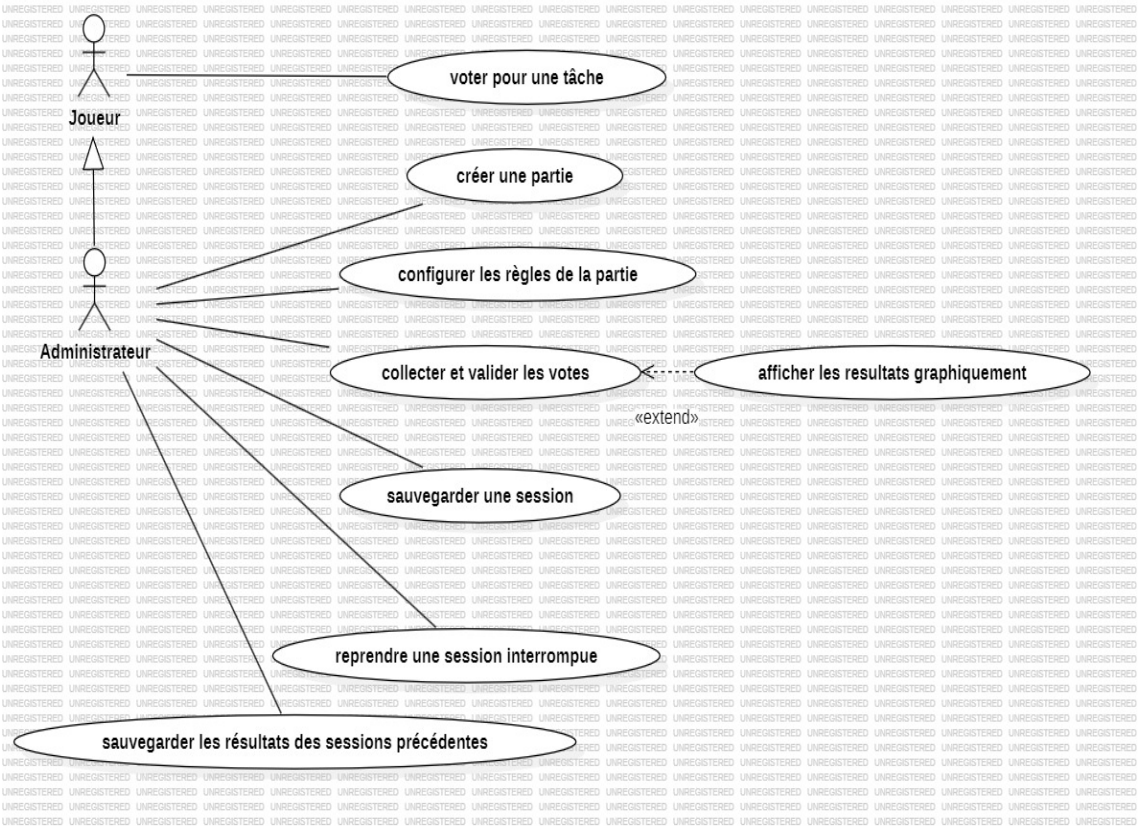

FIGURE 2.1 – *Diagramme de cas d'utilisation*

2.1.2 Diagrammes de classe

Le diagramme de classe illustre la structure statique du système en décrivant les classes, leurs attributs, leurs méthodes, ainsi que les relations entre elles.

La figure suivante décrit notre diagramme de classe.

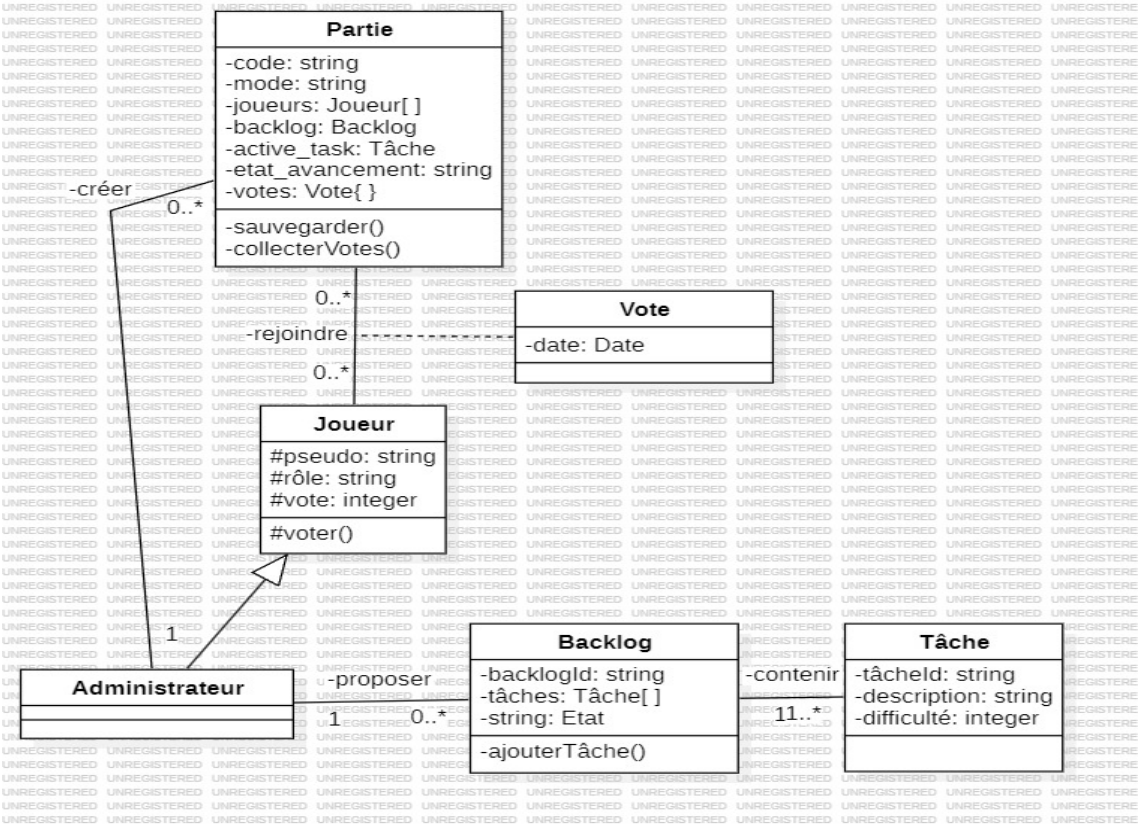

FIGURE 2.2 – *Diagramme de classe*

Ce diagramme de classes présenté offre une vision simplifiée de la structure statique du système, en mettant en avant les relations principales entre les classes et leurs attributs clés. Cependant, dans la mise en œuvre du code, certaines abstractions et détails spécifiques du diagramme n'ont pas été strictement respectés. Cela découle d'une volonté de simplifier le développement, en adoptant une approche plus pragmatique et orientée vers l'efficacité. Bien que cette simplification puisse s'écarter des bonnes pratiques d'un design UML rigoureux, elle répond aux besoins immédiats du projet tout en favorisant une implémentation rapide et fonctionnelle.

2.1.3 Diagramme de séquence : Fonctionnalité "Voter"

Le diagramme de séquence çi-dessous illustre en détail le processus de la fonctionnalité centrale de l'application : le système de vote. Cette fonctionnalité est au cœur de l'application Planning Poker, car elle permet aux joueurs de collaborer pour estimer les tâches d'un backlog selon deux modes de jeu : strict et moyenne.

Le diagramme décrit les interactions entre les différents acteurs et composants du système, depuis la soumission d'un vote jusqu'à la validation finale d'une tâche. Il met également en évidence les spécificités des deux modes, avec une gestion des tours de vote et des calculs adaptés à chaque cas. Cette fonctionnalité clé garantit une dynamique de jeu fluide et collaborative, tout en respectant les règles définies pour chaque mode.

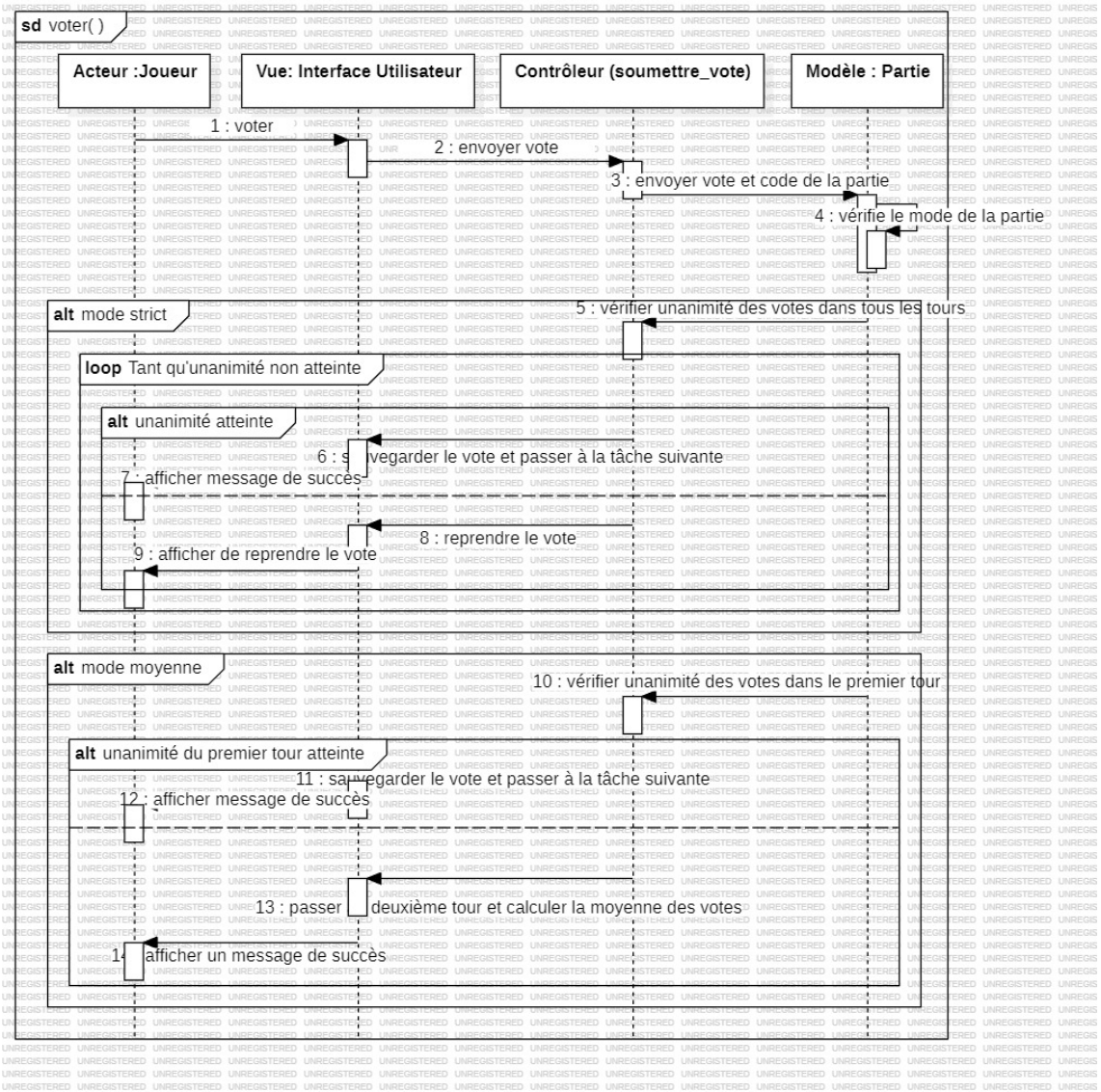

FIGURE 2.3 – *Diagramme de séquence*

Conclusion

Ce chapitre a exposé les éléments de conception de notre application Planning Poker. Le diagramme de cas d'utilisation a permis de visualiser les principales fonctionnalités du système et leurs interactions avec les acteurs. Le diagramme de classe a détaillé la structure statique et les relations entre les différentes entités du système, tandis que le diagramme de séquence de la fonctionnalité voter a illustré le flux d'interactions entre les objets. Ces modèles UML constituent une base solide pour la phase de développement, qui sera détaillée dans le chapitre suivant.

Développement et mise en œuvre

3

Introduction

Ce chapitre détaille la phase de réalisation du projet Planning Poker, couvrant la méthodologie adoptée, les outils et technologies choisis, ainsi que les étapes du développement accompagnées de captures d'écran illustrant le fonctionnement de l'application. Cette phase concrétise la conception établie précédemment en mettant en œuvre les fonctionnalités définies et en assurant leur adéquation avec les besoins fonctionnels et non fonctionnels.

3.1 Technologies utilisées

3.1.1 Environnements logiciels et langages de programmation :

La réalisation de notre projet repose sur l'utilisation de technologies modernes et performantes, choisies pour répondre efficacement aux exigences fonctionnelles et faciliter le développement collaboratif.

Visual Studio Code (VSCode) :

Visual Studio Code est un éditeur de code puissant et léger, largement adopté par les développeurs pour sa flexibilité et ses nombreuses fonctionnalités. Il offre un écosystème riche en extensions, permettant une personnalisation adaptée aux besoins du projet. Sa compatibilité avec Python et Django, grâce aux extensions dédiées, en fait un choix idéal pour un projet basé sur ces technologies. De plus, son intégration native avec Git simplifie la gestion du contrôle de version directement depuis l'éditeur.

Git et GitHub :

Git est un système de contrôle de version distribué qui permet une gestion efficace du code source, facilitant le travail collaboratif et la gestion des modifications. En combinant Git avec GitHub, un service d'hébergement en ligne, nous avons pu bénéficier d'outils puissants comme le suivi des issues, les pull requests pour intégrer les contributions, et un historique clair des versions. Ce choix a permis une coordination fluide entre les membres de l'équipe, en tirant parti des commandes de Git Bash sur Windows pour gérer efficacement les dépôts et synchroniser les changements.

le lien vers notre repository sur git

Python et Django :

Le langage Python a été choisi pour sa simplicité, sa lisibilité et sa vaste communauté de développeurs. Il offre une multitude de bibliothèques pour résoudre des problématiques complexes et se prête bien au développement rapide de solutions web. Django, un framework web basé sur Python, a été retenu pour sa robustesse, son architecture basée sur le modèle MVC (Modèle-Vue-Contrôleur) et ses outils intégrés comme l'administration automatisée et la gestion des bases de données. Django permet également une meilleure scalabilité et une sécurité accrue grâce à des protections intégrées contre des vulnérabilités courantes (injections SQL, XSS, etc.).

SQLite3 :

Pour la gestion de la base de données, nous avons choisi SQLite3, une solution légère et efficace fournie nativement par Django. SQLite3 est une base de données relationnelle qui est intégrée directement dans le projet sans nécessiter d'installation supplémentaire, ce qui simplifie le déploiement pour des projets de petite à moyenne envergure. Elle est parfaitement adaptée pour notre application, car elle permet de gérer les données utilisateurs, les tâches et les votes de manière simple tout en garantissant la cohérence des données.

JSON dans Django :

Un aspect important de notre choix technologique est l'utilisation du format JSON pour le stockage de certaines données complexes, telles que les tâches du backlog et les résultats des votes. Le format JSON est particulièrement adapté dans ce contexte car il permet de stocker des structures de données flexibles et imbriquées tout en étant facilement manipulable dans Django grâce à son support natif via les champs JSONField. Cette approche facilite la gestion de données dynamiques et permet une évolution rapide de la structure du modèle sans nécessiter de migrations complexes.

HTML5 et CSS3 :

Pour la partie front-end, HTML5 et CSS3 ont été utilisés pour concevoir les interfaces utilisateur. HTML5 a permis de structurer les pages web de manière sémantique, tandis que CSS3 a offert les outils nécessaires pour styliser les pages avec des designs modernes et responsive, améliorant ainsi l'expérience utilisateur.

3.2 Méthodologie de développement

Pour le développement de cette application, nous avons adopté une méthodologie agile, et plus spécifiquement le cadre SCRUM, en raison de ses nombreux avantages et de sa capacité à répondre efficacement aux besoins d'un projet dynamique et collaboratif. SCRUM est une approche agile largement utilisée dans le développement logiciel qui se distingue par sa capacité à gérer des projets complexes de manière flexible et itérative. Voici les principaux avantages de cette méthodologie dans le contexte de notre projet :

Itérations courtes et incrémentales

Les sprints, d'une durée généralement de deux à trois semaines, nous ont permis de décomposer le projet en petites tâches concrètes et réalisables sur des périodes relativement courtes. À la fin de chaque sprint,

nous avons pu non seulement livrer des fonctionnalités spécifiques, mais aussi tester et valider leur bon fonctionnement. Ce processus incrémental a facilité l'ajustement rapide de notre approche et la prise en compte des retours, nous permettant de livrer une version du produit plus stable à chaque cycle.

Flexibilité et adaptabilité

L'un des principes fondamentaux de la méthodologie agile est sa capacité à s'adapter aux changements. En fonction des imprévus, des nouvelles exigences ou des retours des parties prenantes, nous avons pu ajuster nos priorités et redéfinir les objectifs pour chaque sprint. Cette flexibilité a permis d'intégrer de nouvelles idées ou de réagir rapidement aux défis techniques tout en maintenant le cap sur les objectifs globaux du projet.

Collaboration accrue et communication continue

SCRUM repose sur une communication constante entre les membres de l'équipe et les parties prenantes. Les réunions quotidiennes, appelées "daily stand-ups", ont permis de suivre l'avancement du projet, de discuter des obstacles rencontrés, et de coordonner les actions à entreprendre. Ces réunions ont favorisé une meilleure collaboration, assurant que chaque membre de l'équipe était bien informé des priorités et des ajustements nécessaires. De plus, les revues de sprint et rétrospectives ont été des moments privilégiés pour évaluer les performances du sprint précédent et améliorer continuellement notre approche.

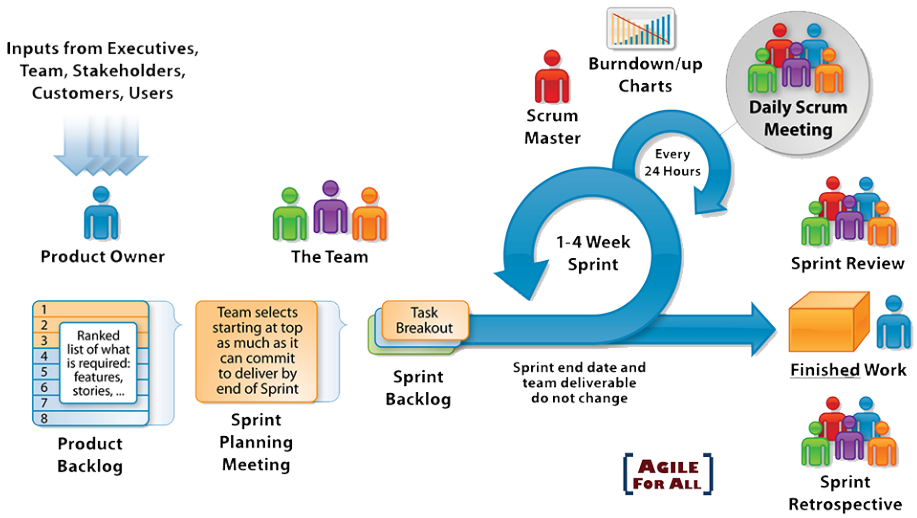

FIGURE 3.1 – *Méthodologie SCRUM*

3.3 DÉVELOPPEMENT

Au cours du développement de cette application, nous avons mis en place des pratiques de développement logiciel modernes, notamment l'intégration continue, afin de garantir une livraison continue et de maintenir une qualité optimale tout au long du projet. Après avoir développé les fonctionnalités principales du système, nous avons intégré ces éléments dans un flux de travail automatisé, permettant de tester, documenter et déployer les différentes composantes du projet de manière efficace.

L'intégration continue a été facilitée par l'utilisation d'outils tels que GitHub Actions, qui ont permis d'automatiser plusieurs processus, notamment la génération de documentation et l'exécution de tests unitaires à chaque modification du code. Cela a assuré que les mises à jour du projet étaient testées en temps réel et que la documentation était constamment mise à jour.

Dans les sections suivantes, nous détaillerons les différentes étapes de ce processus : la génération automatique de la documentation avec Doxygen, l'exécution des tests unitaires avec Python, ainsi que l'automatisation de ces étapes avec GitHub Actions. Nous fournirons également des illustrations des interfaces développées, alimentées par les données JSON de la base de données, et expliquerons comment ces outils ont contribué à la gestion de la qualité du code et à la fluidité du processus de développement.

Présentation des Interfaces

Une fois les fonctionnalités principales du système développées, la prochaine étape a consisté à vous présenter les interfaces utilisateur avec les données provenant de la base de données. Ces interfaces sont alimentées par du contenu JSON, qui constitue le format de stockage principal dans la base de données SQLite3 de Django.

Pour illustrer cela, des captures d'écran des interfaces avec le contenu JSON seront fournies ci-dessous, montrant le rendu des données dans l'application. Ces captures serviront à démontrer l'interaction entre l'interface utilisateur et la structure de données dynamique.

[Insérer ici une capture d'écran de l'interface utilisateur avec des données JSON.]

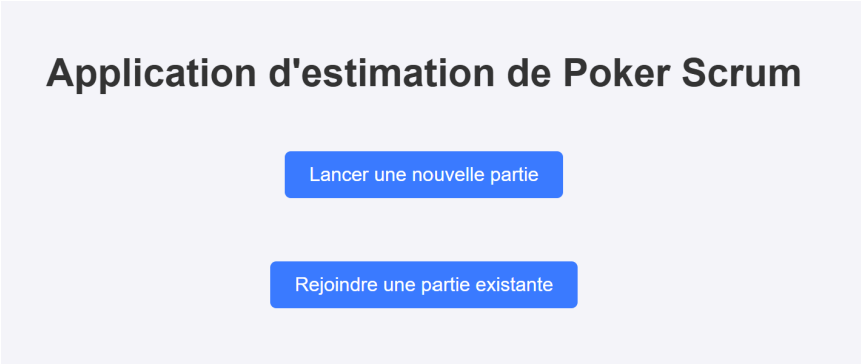

FIGURE 3.2 – *interface d'acceuil*

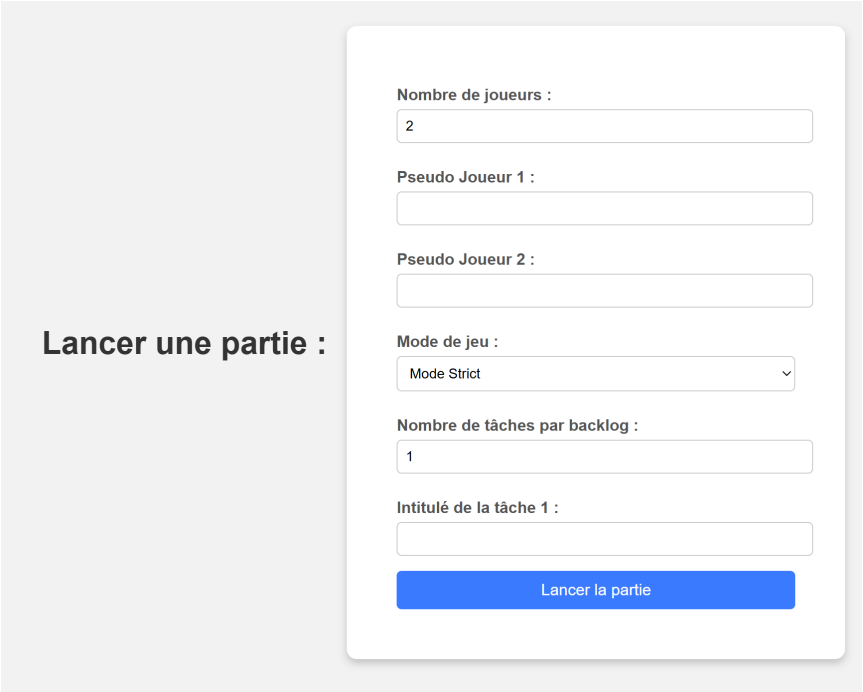

FIGURE 3.3 – *interface lancer une partie*

FIGURE 3.4 – *interface partie créée avec succès*

FIGURE 3.5 – *interface rejoindre partie*

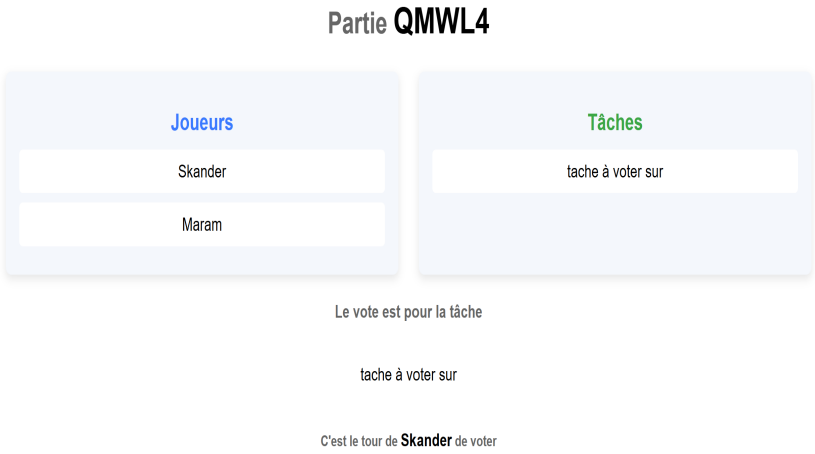

FIGURE 3.6 – *interface du vote Partie 1*

FIGURE 3.7 – *interface du vote Partie 2*

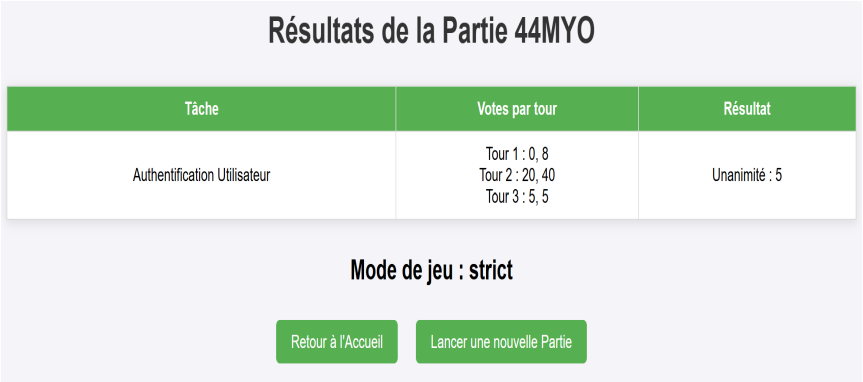

FIGURE 3.8 – *interface résultats du vote*

FIGURE 3.9 – *interface partie suspendue*

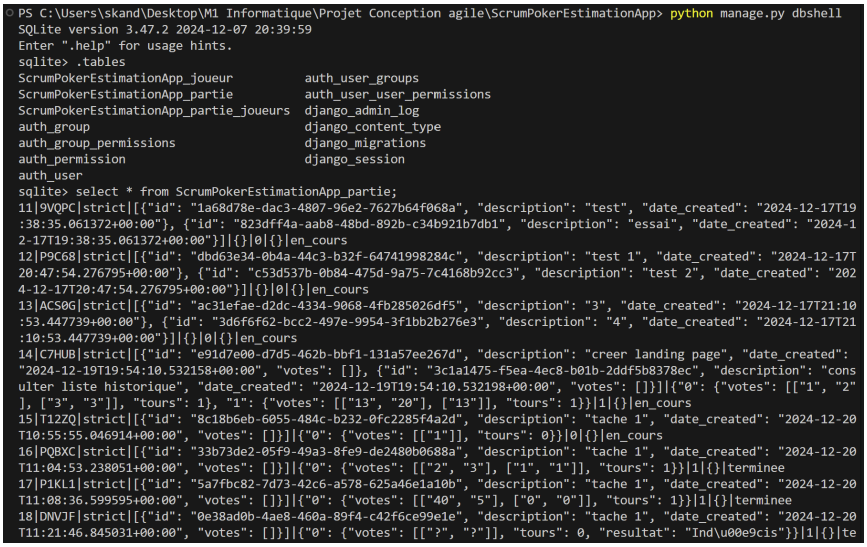

FIGURE 3.10 – *Capture d'une partie de la base de données sous format JSON*

Génération de Documentation et Automatisation avec GitHub Actions

Pour faciliter la compréhension et la maintenance du projet, une documentation détaillée du code a été générée à l'aide de Doxygen, un outil de documentation pour les projets logiciels en C++, Python, et d'autres langages.

Dans le cadre de ce projet, nous avons utilisé Doxygen pour automatiser la génération de documentation à partir du code Python. Chaque fois qu'une modification majeure est apportée au code, la documentation

est régénérée pour refléter les nouveaux ajouts ou changements dans les fonctionnalités.

En parallèle, GitHub Actions a été utilisé pour automatiser le processus de génération de la documentation à chaque modification du code. Une fois qu'un push est effectué sur le dépôt GitHub, un workflow est lancé pour régénérer la documentation et la déployer automatiquement, permettant ainsi d'avoir une version constamment mise à jour de la documentation du projet sans intervention manuelle.

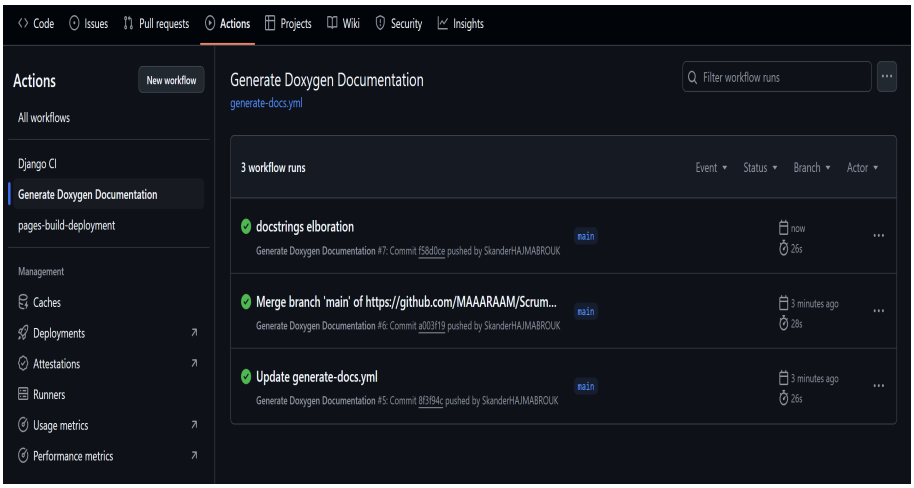

FIGURE 3.11 – *Documentation automatique*

Génération et Automatisation des Tests Unitaires avec Python et GitHub Actions

Pour garantir la fiabilité du code développé, on a mis en place des tests unitaires en utilisant Python. Ces tests permettent de vérifier le bon fonctionnement de chaque fonctionnalité en s'assurant que chaque unité de code se comporte comme prévu, même après des modifications ou des ajouts. Les tests sont essentiels pour éviter les régressions et valider chaque portion de code indépendamment du reste de l'application.

Afin d'automatiser l'exécution de ces tests à chaque modification du code, on a également utilisé GitHub Actions. Dès qu'une modification est apportée au projet et poussée sur GitHub, un workflow est déclenché automatiquement pour exécuter tous les tests unitaires. Cette automatisation permet de s'assurer que les tests sont toujours exécutés de manière cohérente et en temps réel, réduisant ainsi le risque d'introduire des erreurs non détectées dans le projet.

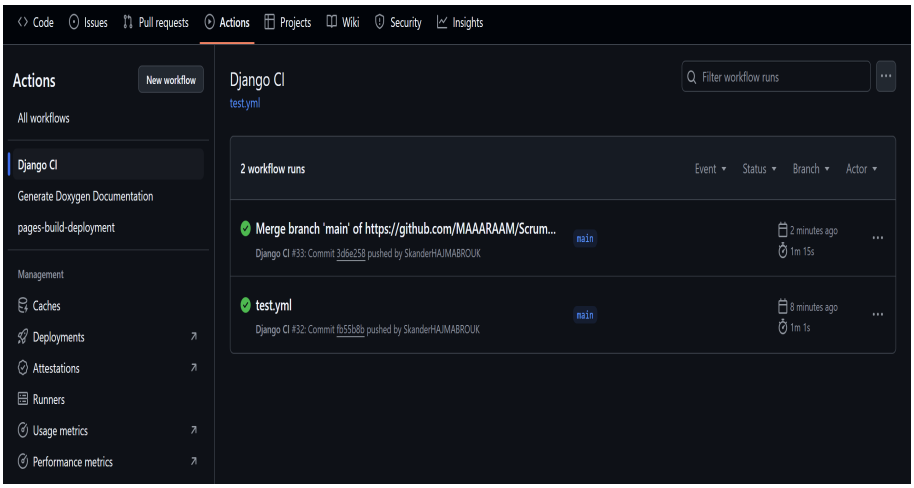

FIGURE 3.12 – *Tests unitaires automatiques*

CONCLUSION

Ce chapitre a détaillé la méthodologie SCRUM adoptée, les outils et technologies utilisés, ainsi que les étapes clés du développement de l'application Planning Poker. Les choix technologiques ont permis de garantir un développement efficace et modulable, tandis que les tests ont assuré une conformité aux besoins initiaux. Cette phase marque une transition vers l'évaluation finale de l'application et l'analyse des résultats obtenus.

Conclusion générale

Le développement de l'application Planning Poker constitue une expérience enrichissante à plusieurs égards, en alliant la théorie à la pratique dans un contexte collaboratif et méthodologique. Ce projet nous a permis de mettre en œuvre une solution concrète répondant aux besoins des équipes de développement, tout en intégrant les concepts essentiels abordés durant notre formation en M1 Informatique, tels que les méthodologies agiles, les principes de conception logicielle, et les technologies web modernes.

L'application se distingue par sa simplicité d'utilisation et sa capacité à s'adapter à divers scénarios grâce à ses fonctionnalités robustes, telles que :

L'intégration de multiples modes de vote pour s'ajuster aux besoins spécifiques des équipes.

Une interface utilisateur ergonomique, favorisant une navigation intuitive.

La possibilité de sauvegarder et de reprendre des sessions pour garantir une continuité de travail.

Une visualisation graphique claire et interactive des résultats, facilitant les prises de décision.

Le développement de cette application nous a également permis d'explorer des technologies récentes, telles que Github Actions. Nous avons ainsi renforcé nos compétences tout en apprenant à collaborer efficacement dans un cadre agile.

Enjeux et défis relevés

Au cours de ce projet, nous avons été confrontés à plusieurs défis, notamment la gestion des communications en temps réel, l'organisation des données structurées en JSON, et la conception d'une interface web responsive. De plus, un incident imprévu a perturbé notre travail : le vol du PC d'un des membres de l'équipe, ce qui a entravé la progression du projet pendant plusieurs semaines. Malgré ce contretemps, grâce à une méthodologie rigoureuse et un travail d'équipe soutenu, nous avons su surmonter ces obstacles et livrer un produit fonctionnel respectant les exigences définies.

Perspectives

Bien que notre application réponde aux objectifs initiaux, elle ouvre également la voie à des améliorations et extensions futures. Sur le plan des fonctionnalités avancées, l'ajout de modes de jeu personnalisables et d'une

fonction de brainstorming avec un tableau blanc collaboratif pourraient offrir des options supplémentaires aux utilisateurs. Nous projetons également une extension multiplateforme, notamment par le développement d'une application mobile native pour iOS et Android, afin de rendre l'application accessible à un plus grand nombre d'utilisateurs, ainsi qu'une compatibilité optimisée pour les tablettes et écrans tactiles.

En termes d'optimisation, nous souhaitons améliorer les performances des sessions de vote pour gérer un nombre accru de participants et renforcer les mécanismes de sécurité pour garantir une utilisation fiable en milieu professionnel.

De même, nous n'avons pas pu implémenter un chronomètre et un espace de discussion pour enrichir l'interaction des joueurs durant les sessions de vote en raison des contraintes de temps. Ces fonctionnalités restent néanmoins sur notre liste d'améliorations futures.