Projet CAPI-
Planning Poker

Adjamé Tellier-Rozen

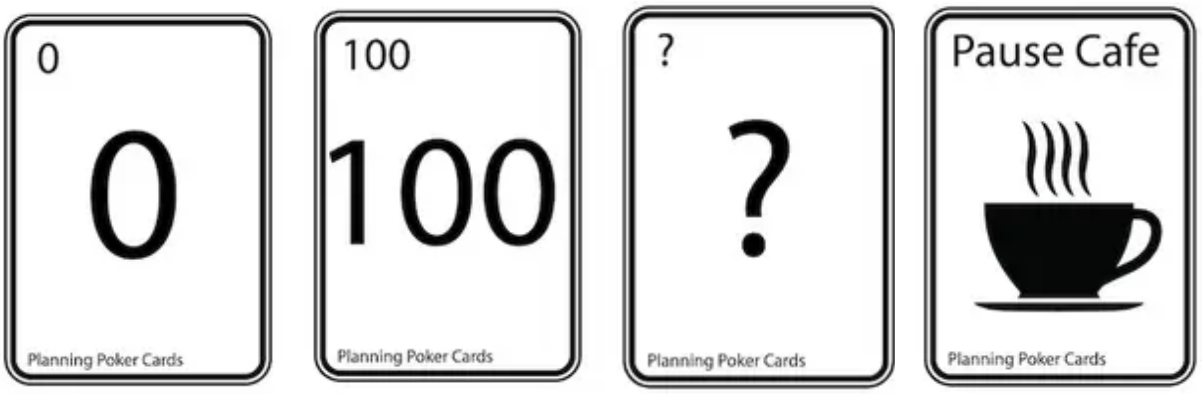


Table des matières

Partie 1: Le projet

Contexte et Objectifs du Projet CAPI

Le Projet CAPI, réalisé dans le cadre de la matière "Conception agile de projets informatiques", consiste en la création d'un jeu de Planning Poker sur une période de six semaines. Ce projet vise à appliquer les principes du développement agile appris en cours.

Défi personnel et adaptation

Initialement prévu en binôme, le projet a pris une tournure différente pour moi. En raison de divergences de rythme et de méthodologie avec ma partenaire, nous avons convenu qu'il serait plus judicieux de poursuivre nos travaux séparément.

Aperçu du Planning Poker

Le planning poker est une technique utilisée dans le développement agile pour estimer la complexité des tâches.
Les membres de l'équipe utilisent des cartes numérotées pour représenter leurs estimations d'effort, et après une discussion structurée, ils sélectionnent une carte qui reflète le consensus atteint.
Cette méthode favorise la participation de tous et aide à atteindre une compréhension commune de la charge de travail.

image1: Decks cartes

Objectif et consignes du projet

La réalisation du jeu de Planning Poker doit refléter notre compréhension et notre application des pratiques agiles.
Bien que certains aspects majeurs du projet soient laissés à notre préférence, comme le choix du langage de programmation, du style visuel et de l'interface, il est impératif d'intégrer certaines spécificités :

1. Réalisation Technique:
    - Un projet sur GitHub fonctionnel et clonable, accompagné d'un README détaillant les instructions d'utilisation, qui utilise l'intégration continue.
    - L'implémentation des différents modes de jeu, ainsi que des fonctionnalités pour l'enregistrement et le chargement de fichiers de Backlog. Le tout en respectant 3 designs patterns.

2. Rédaction et justification
    - Une explication détaillée des choix techniques
    - Une documentation complète et accessible du projet.

Partie 2: Le produit

Aperçu du produit final et des fonctionnalités

Dans le développement de notre jeu de Planning Poker, nous avons choisi une approche axée sur l'utilisateur, visant à fournir une interface claire et facile à utiliser. Notre but était de concevoir une expérience fluide, permettant un accès rapide et direct aux fonctionnalités clés sans surcharge visuelle. La simplicité de La conception minimaliste avec un fond blanc uni et des boutons aux couleurs distinctives contribue à une navigation aisée et à une compréhension immédiate des actions disponibles.

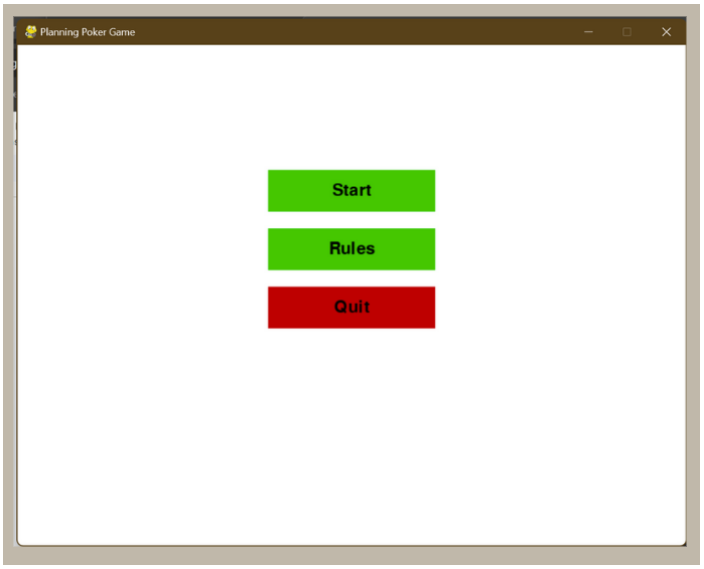


image2: L'accueil

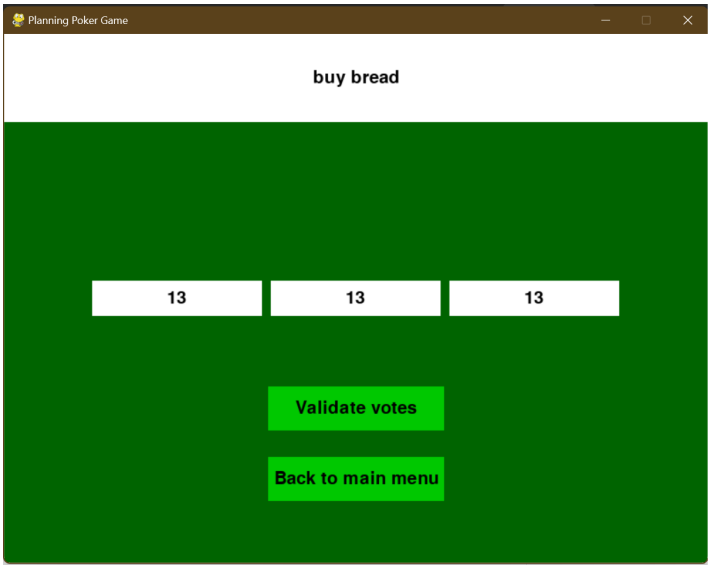


image3: Validation d'une tâche

Expérience de Jeu Simplifiée

Le design du jeu utilise une gamme de couleurs choisies pour leur signification intuitive: le vert indique la progression, le rouge signale l'annulation ou le rejet, et d'autres teintes spécifiques, comme le vert foncé pour confirmer et le bleu pour les situations exceptionnelles. Ces couleurs ont été sélectionnées pour leur fonctionnalité, aidant à orienter l'utilisateur à travers les différentes étapes du jeu avec une clarté visuelle. Pendant le processus de vote, le jeu dirige les joueurs, affichant la tâche actuelle et offrant un retour visuel sur les votes dès qu'ils sont enregistrés.

Flexibilité et Personnalisation

Pour avoir un maximum de flexibilité, les joueurs ont la possibilité de choisir parmi plusieurs méthodes d'estimation. Strictes, Moyenne, Médiane, Majorité absolue et Majorité relative permettant une personnalisation selon les préférences de l'équipe et les exigences du projet.

En outre, les joueurs ont la flexibilité d'utiliser leurs propres fichiers JSON pour les backlogs, ce qui leur permet d'importer des données sans être obligés de les créer directement dans le jeu. Cette fonctionnalité assure une continuité fluide de l'expérience de jeu.

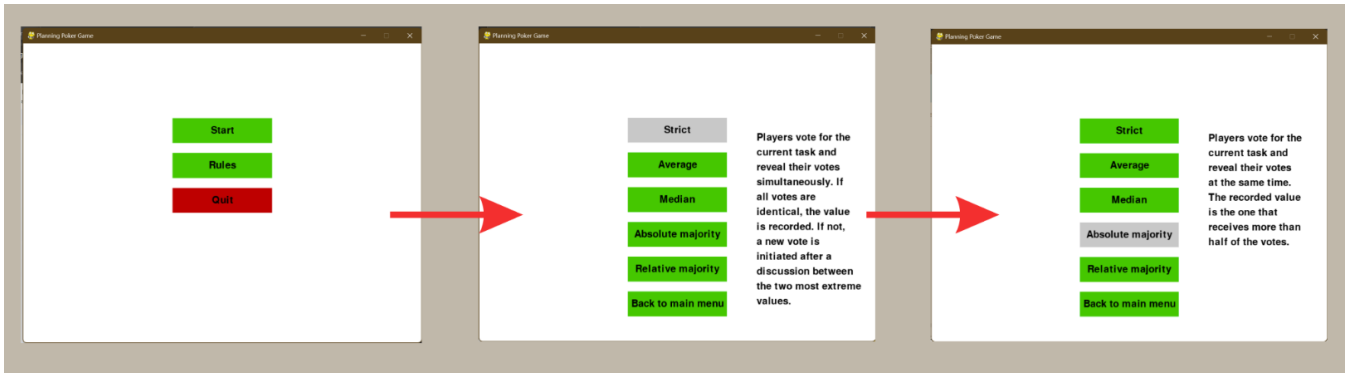


image3: Le choix des règles

image4: Le chargement ou la création de Backlog

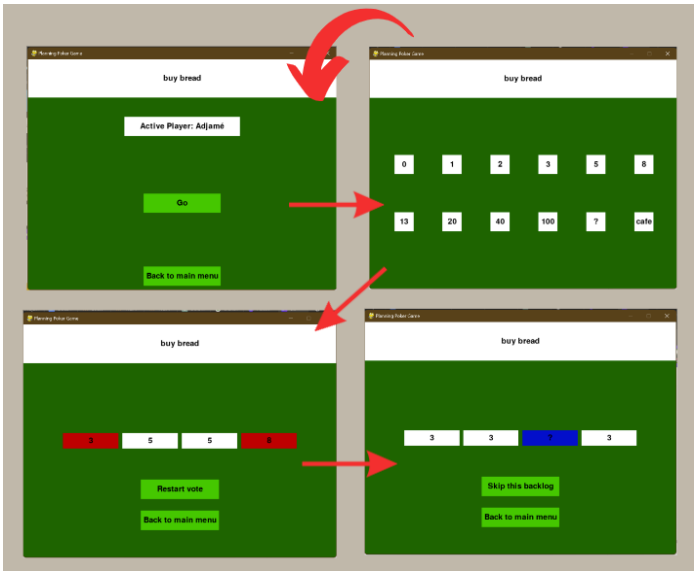


image5: Boucle de jeu répété pour chaque joueur et affichage des résultats

Choix du langage de programmation et des libraires

Langage de programmation

Pour assurer une portabilité et une accessibilité maximales, nous avons choisi Python comme langage de programmation principal pour notre projet . Python se distingue par sa syntaxe claire et sa capacité à fonctionner sur différents systèmes. Cela permet à notre jeu d'être exécuté aisément sur n'importe quelle machine.

Librairies et Gestion des fonctionnalités

- **Pygame :** Pour la création d'une interface utilisateur interactive et réactive, nous avons utilisé Pygame, qui offre de nombreuses fonctionnalités pour le développement de jeux en Python. Cela permet de gérer les entrées des utilisateurs et de rendre l'expérience de jeu fluide et engageante. (*https://www.pygame.org/news*)
- **Tkinter :** Choisi pour la simplicité d'ouverture et de gestion de fichiers. (*https://wiki.python.org/moin/TkInter*)
- **Numpy** : Pour les opérations mathématiques et les calculs, en particulier ceux liés aux votes. (*https://numpy.org/*)
- **JSON :** Pour la mise en forme des données de Backlog, JSON est utilisé en raison des contraintes de gestions de backlogs. (*https://docs.python.org/fr/3/library/json.html*)

Partie 3: Approfondissement

Choix de designs patterns

Notre architecture logicielle repose sur l'implémentation de trois design patterns, chacun correspondant à un aspect fondamental de la conception logicielle :

- **Singleton (Création):** Le pattern Singleton a été choisi pour garantir une unique instance des composants clés de notre jeu: Jeu, Board, et Visual. Cette méthode assure l'uniformité et la stabilité des configurations durant toute l'exécution du programme. Les éléments vitaux sont sécurisés en privé, tandis que l'accès est régulé par des getters et setters via des constructeurs publics, permettant ainsi une gestion unifiée des variables globales et préservant l'intégrité du système.
(*https://refactoring.guru/fr/design-patterns/singleton*)

- **Composite (Structurel):** Nous avons implémenté le pattern Composite pour organiser l'architecture hiérarchique de notre interface utilisateur. Cette approche nous permet de gérer des collections d'objets, tels que les cartes chez un joueur ou les joueurs dans le jeu, de façon cohérente. Elle facilite ainsi la gestion et l'interaction au sein des différents niveaux de l'interface, en traitant des ensembles d'objets comme une entité unique, simplifiant considérablement la complexité structurelle.
(*https://refactoring.guru/fr/design-patterns/composite*)

- **Commande (Comportemental):** Le pattern Commande est utilisé pour encapsuler les actions du jeu en objets distincts et autonomes. Cette approche permet de programmer, mettre en attente, ou annuler des actions de manière flexible, en particulier dans le contexte de notre interface, où chaque bouton est associé à une commande spécifique qui comprend un traitement visuel, une action déclenchée par un clic, et une modification des états, facilitant ainsi une gestion claire et modulaire des interactions utilisateur. (*https://refactoring.guru/fr/design-patterns/command*)

Structure et Fonctionnement du Code

Le code (et commentaires) est rédigé en anglais pour respecter les conventions d'écriture et de programmation, seul le rapport, le README et le fichier de tests unitaire sont en français pour être certains de correspondre aux consignes.
Il est organisé autour de classes dédiées à des responsabilités uniques :

- **Classe Carte**: Représente une carte individuelle et gère ses attributs.

- **Classe Joueur**: Encapsule les données et les actions d'un joueur, incluant la gestion de son identité et de ses cartes.

- **Classe Jeu**: Coordonner le jeu, les joueurs, et les backlogs, et centralise la logique de gestion des votes et des backlogs.

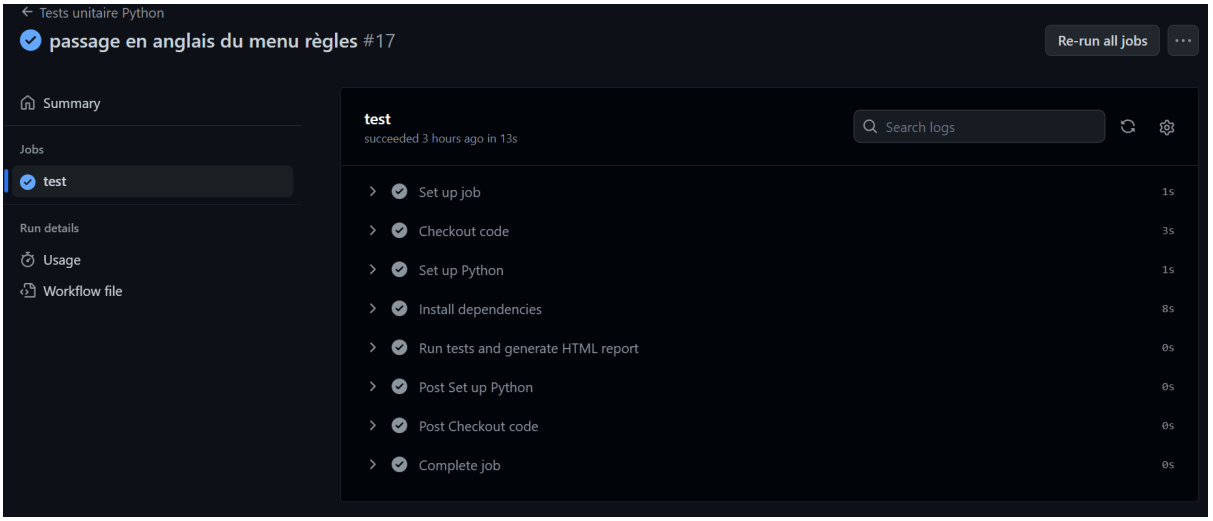
- **Classe Visual**: Contrôle l'affichage et interagit avec l'utilisateur, en s'appuyant sur les informations de jeu.

- **Classe Board:** Gère l'état du plateau de jeu et les interactions du menu.

Cette séparation en classe est essentielle dans la méthode agile et contribue à la flexibilité du code, ainsi qu'à l'implémentation des différents design patterns.

Génération de tests unitaires

Pytest est un framework de tests unitaires pour le langage de programmation Python qui permet de créer des tests simples avec une syntaxe concise et une intégration possible avec les workflows de python ce qui permet de vérifier continuellement le fonctionnement de chaque partie de notre code.
(https://docs.pytest.org/en/7.4.x/)
Nous avons adopté pytest pour l'élaboration de nos tests unitaires, facilitant la vérification indépendante de chaque partie du code. L'intégration de ces tests dans les workflows GitHub nous permet d'assurer une qualité continue à chaque mise à jour du code. Les tests automatisés sont un élément crucial de notre intégration continue, assurant que les régressions sont rapidement identifiées et corrigées.



Génération de documentation

Doxygen est un outil de génération de documentation à partir du code source, qui prend en charge plusieurs langages de programmation et permet de créer une documentation logicielle à la fois technique et lisible sous divers formats. (https://www.doxygen.nl/)

Bien que la documentation générée par Doxygen ne soit pas encore opérationnelle, la documentation est cruciale pour la compréhension et la durabilité du projet. Les workflows GitHub sont également utilisés pour automatiser la génération de la documentation, assurant ainsi que chaque contribution au code est accompagnée des explications nécessaires à sa compréhension.
Pour le moment cela ne se voit malheureusement que part les commentaires de descriptions de fonctions.

Partie 4: Ouverture du projet

Difficultés rencontrés

La collaboration a émergé comme l'un des défis majeurs de ce projet. Les différences d'approches et de préparation ont mené à une continuation indépendante du projet, soulignant ainsi l'importance cruciale de la communication et de l'alignement au sein d'une équipe.
Par ailleurs, la mise en place de la documentation automatique avec Doxygen a présenté ses propres complications, nécessitant une attention détaillée et finalement mise de côté en raison de contraintes de temps. Cela a mis en évidence le besoin de prudence lors de la configuration des outils de développement.
La gestion des backlogs a aussi posé problème, notamment dans la création et le stockage des fichiers. Un souci particulier est apparu lors de l'ajout manuel de backlogs, provoquant un retour inattendu au menu principal. Bien que les données ne soient pas perdues, cette redirection erronée nécessite une résolution, mais l'origine du problème reste à déterminer.

Axes d'amélioration

Améliorer l'expérience utilisateur est essentiel, notamment en affinant le processus de génération de backlogs pour une intégration plus harmonieuse et pour résoudre les problèmes de redirection.
La fonctionnalité de documentation automatique doit aussi être révisée pour garantir que le code soit accompagné d'une documentation complète et actualisée.
Sur le plan esthétique, l'introduction d'animations et de retours sonores rendrait l'interaction avec le jeu plus riche et engageante.

À plus long terme, envisager une version en ligne du jeu ouvrirait la possibilité de jouer à distance et nécessiterait l'intégration de nouvelles fonctionnalités, comme un chat intégré, pour faciliter la communication entre les joueurs durant les sessions de planning poker.

conclusion

Le projet CAPI s'est révélé être une initiative ambitieuse qui m'a incité à développer mes compétences techniques et à naviguer à travers divers défis imprévus. L'expérience acquise en travaillant avec la bibliothèque Pygame, pour la première fois, a été particulièrement instructive, nécessitant une période extensive de tests avant d'atteindre des résultats satisfaisants. En résumé, bien que le projet ait été exigeant, il a été extrêmement enrichissant, me permettant d'étendre mes compétences et de comprendre l'application pratique des principes de la programmation agile dans des scénarios réels.