Rapport de projet

Conception agile de projets informatiques

Romane TOUZAIN
Mathias VACHER

Master 1 – Informatique – 2023/2024

Ce projet consistait en la création d'une application de planning poker.
Ce type de projet ont pour but de faciliter la planification des tâches au sein d'une équipe de développement. Cette application permet à chaque membre de l'équipe de voter sur la difficulté des tâches à réaliser, et d'en discuter ensuite de manière ludique. C'est très apprécié et utilisé dans le monde professionnel, et est un aspect important de la gestion de projet.

Ce rapport va vous informer sur nos choix de conception, les patterns utilisés, les décisions techniques etc ... mais aussi sur notre manière d'aborder un tel projet.

- **Patterns utilisés**

Observateur (Observer) :
Le modèle de conception Observer est utilisé dans la gestion des votes et de la progression du jeu. Par exemple, dans la fonction on_card_selected, les observateurs (les différentes parties du jeu) sont notifiés lorsque chaque joueur effectue son vote. La logique de mise à jour est déclenchée lorsqu'un joueur sélectionne une carte.

Grâce à cet observateur, on va pouvoir changer le numéro de la tâche en haut de la fenêtre, mettre à jour le nom du joueur en fonction de qui doit voter etc... C'est très important pour un projet comme celui-ci.

MVC (Modèle-Vue-Contrôleur) :
L'architecture du projet suit le modèle de conception MVC. La classe PlanningPoker agit comme le modèle, la classe CustomDialog et les fonctions dans gui.py agissent comme la vue, tandis que les fonctions dans main.py/rules.py agissent comme les contrôleurs. Cette séparation est pratique pour isoler les problèmes, et pour attribuer à chaque fichier un rôle propre et unique.

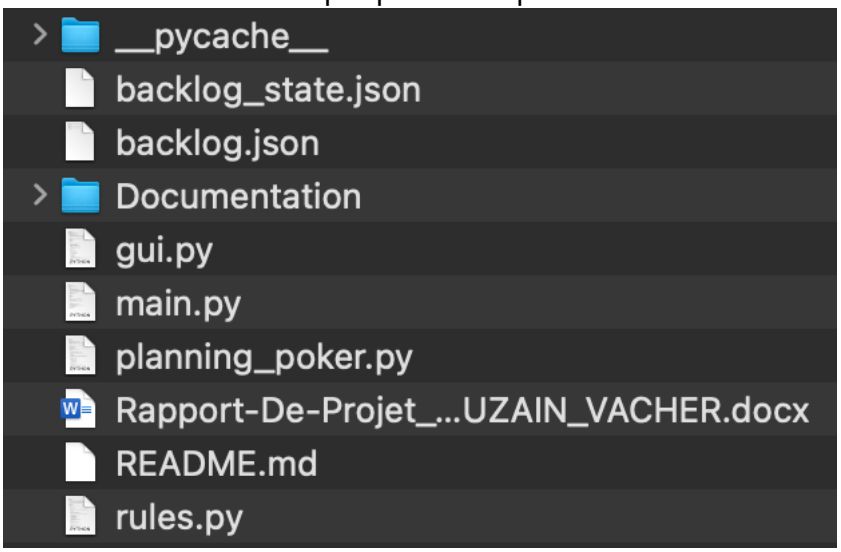
Stratégie :
Le pattern Stratégie a été employé pour gérer les différentes méthodes de calcul de la difficulté des tâches en fonction des règles choisies. Cela rend

notre système flexible, permettant l'ajout facile de nouveaux « modes de jeu » sans modifier la structure globale. (Voir fichier rules.py)

- **Choix techniques**

Architecture :
Grâce au design pattern MVC, nous avons des fichiers bien distincts avec des rôles propres à chacun. L'architecture de notre projet est somme toute très classique avec les fichiers à la racine du dossier, tous accessibles par des chemins très courts & simples.

Nous allons vous expliquer chaque fichier de notre arborescence :



__pycache__ : dossier « inutile » contenant le cache de chaque exécution de notre programme
Backlog_state.json : Fichier enregistrant temporairement les votes pour chaque tâche si jamais l'application crash, idéalement nous pourrions le rouvrir avec un bouton pour repartir de ce point depuis l'application, mais par manque de temps nous n'avons pas pu développer cette fonctionnalité.
Backlog.json : Fichier contenant chaque tâche à analyser et à voter.
Documentation : Dossier contenant notre documentation générée automatiquement
Gui.py : Fichier responsable de l'affichage des différentes fenêtres
Main.py : Fichier responsable de la bonne exécution de notre projet
Planning_poker.py : Fichier responsable de la logique du jeu du planning poker, du vote tour à tour, de la récupération des tâches dans le fichier backlog...

Rapport de projet.docx : Ce document
README.md : Fichier contenant une courte introduction & explication de comment exécuter notre code
Rules.py : Fichier contenant la logique des différents « modes de jeu » et la manière de les gérer

Langage :
Nous avons choisi python car nous pensions que cela allait être plus simple de gérer un projet de ce type, choix que nous regrettons légèrement aujourd'hui. Pour plusieurs raisons : il est difficile de faire une interface agréable à utiliser et joli à l'œil avec les simples libraires de python, là où par exemple en utilisant des langages web, nous n'aurions été limités que par notre imagination. Au début, nous pensions python plus riche en librairie/documentation/tutoriels, mais il s'avère que les langages web le sont tout autant. Nous nous sommes tout de même débrouillés pour avoir un projet fonctionnel, et nous avons fait de notre mieux pour le rendre agréable à utiliser et ludique, c'est le principal ! Tout ça en plus d'avoir acquis de très bonnes connaissances dans ce langage important de la programmation.

Classes & structure du code :
Nous avons structuré notre code en classes pour encapsuler chaque fonctionnalité, et assurer une modification/un maintien à jour du code rapide et simple.
Chaque classe est commentée, et a une documentation complète rédigée par nos soins puis complétée automatiquement.
Nous pensons notre code simple, compréhensible, et pouvant être repris par n'importe quelle personne le souhaitant. De par notre niveau en python qui n'est pas expert, nous avons été forcés de faire au plus simple, ce qui nous croyons être un avantage et un grand plus.

- ## Documentation

La documentation a été rédigée et créée par nos soins puis de manière automatique par le logiciel Doxygen. Nous pensons que générer de manière automatique cette documentation rend le projet bien plus attractif, et le rend plus simple à être repris, car tout est rédigé sur la même base, mais nous l'aborderons dans le point suivant. Toutes les fonctions, classes, attributs... sont documentés !

Un point important à noter :
Pour changer les tâches : nous n'avons pas eu le temps de mettre en place le système de gestion de fichier demandé, c'est pourquoi nous vous demandons de bien vouloir accéder au fichier « backlog.json » et de modifier à la main chaque tâche en fonction de votre besoin. Merci de bien respecter le format de notre fichier.

Mais tout cela est expliqué dans le readme présent dans nos dossiers/sur notre GitHub bien évidemment.

- ## Intégration continue

Tests unitaires :

Par manque de temps, notre projet n'aura pas de tests unitaires... C'est dommage car cela facilite grandement la découverte de bug/la correction de problèmes, et aide à tester le programme bien plus aisément. Le manque de temps aura été notre plus gros problème ici, car les documents mis à notre disposition par M. Lachand-Pascal sur le Moodle était clairs et compréhensibles, même si complexes à appliquer à un tel projet selon nous...

Documentation :

La documentation aura été créée par nous puis générée par le logiciel Doxygen dont la documentation sur le Moodle nous aura été bien utile. Celle-ci comprend chaque classe, chaque fonction, chaque attribut, chaque paramètre de notre projet. Elle vous en apprendra plus sur celui-ci, mais vous pouvez aussi consulter notre code.

Merci d'avoir lu ce rapport, nous sommes bien évidemment disponibles par mail pour toutes questions ou interrogations.

Romane Touzain
Mathias Vacher