Conception agile de projets informatiques

Projet de planning poker

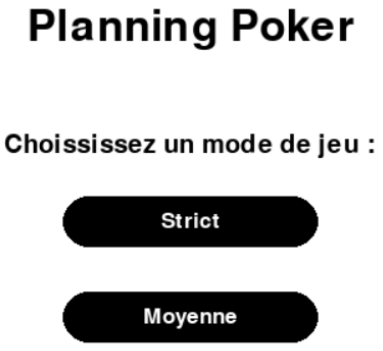
Guillaume Panighetti
Arthur Perret

I – Introduction

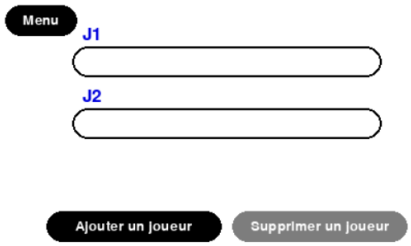
Le projet dont nous allons parler aujourd'hui est issus d'un module de master 1 informatique à l'université Lyon 2, Conception agile de projets informatiques. Le projet consiste à créer une application de planning poker afin d'estimer la difficulté de développement d'un ensemble de tâches basé la discussion entre collègue/équipe. Pour comprendre notre déroulé du projet, nous commencerons par voir ce que fait l'application et son fonctionnement. Nous verrons ensuite les procédés techniques ainsi que nos différents choix et nous finirons par parler de l'intégration continue.

II – Fonctionnement de l'application

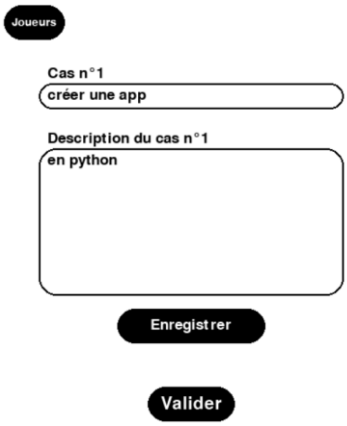
Comme dit précédemment, un planning poker est comme un jeu qui sert à estimer les difficultés d'un ensemble de tâche. Quand nous lançons l'application nous avons le choix entre le mode strict et le mode moyenne.



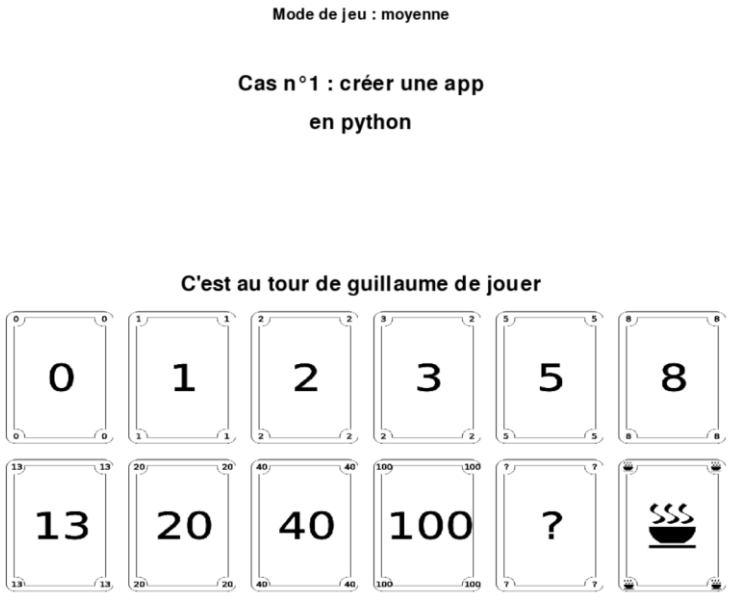
Une fois le mode choisi, nous choisissons le nombre de joueurs (au maximum 5) et le pseudo de chaque joueur.



Ensuite chacun peut écrire plusieurs tâches avec une description.



La partie peut commencer. Chacun leur tour, les joueurs ont des cartes affichées où ils peuvent cliquer dessus pour estimer la difficulté de la tâche actuellement affichée, tout en débattant entre eux. En fonction du mode de jeu, le tour recommence ou non.



A la fin de la partie, une fois que tous les cas ont été voté, un fichier json s'enregistre dans les fichiers du pc avec toutes les tâches ainsi que leur difficulté (moyenne ou unanime).

III – Procédés techniques et architecture

Pour programmer cette application, nous avons utilisé le langage Python. Il est très flexible, plutôt facile à prendre en main et parfait pour ce genre d'application. De plus, ce genre de projet permet de nous améliorer fortement à ce langage utile à plein de type d'application. En ce qui concerne les librairies, la principale utilisée pour ce projet est pygame. C'est une librairie qui permet de faire des interfaces graphiques, en particulier pour du jeu vidéo, facilement. Nous avons également utilisé d'autres librairies comme json afin de créer des fichiers json ou encore tkinter pour ouvrir des boites de dialogues et d'afficher des informations, ainsi que d'autres.

La structure de notre code est faite en programmation orienté objet, c'est-à-dire que nous utilisons des classes. Nous avons fait plusieurs fichiers python pour séparer les différentes fonctionnalités de l'application afin que ce soit plus lisible. Les modules et classes sont donc importés en début de fichier afin de les relier entre eux. Ce choix nous parait le plus logique car la POO permet de modifier facilement le code, notamment grâce à l'héritage, l'encapsulation et la séparation des interfaces. Cela facilite la maintenance, une étape importante de la création d'application.

Dans ce code, nous avons utilisés trois designs pattern. Un design pattern est une solution récurrente à un problème de conception logicielle. Il s'agit d'une description générale d'une solution, qui peut être appliquée à de nombreux contextes différents, cela facilite alors les programmeurs.

Singleton : surement l'un des plus connus, le singleton garantit qu'il n'y aura qu'une seule instance d'une classe. Dans notre application, elle garantit une seule instance pour la classe Partie.

Observer : un design pattern de structure, il permet à un objet de notifier d'autres objets lorsqu'il change d'état. Ici, il permet de mettre en pause la partie quand tous les joueurs utilisent la carte « café ».

Factory : un design pattern de création qui permet de créer des objets de manière dynamique en fonction de certains critères. Ici il créer une instance de la classe Cartes.

Notre code est organisé en 5 fichiers Python avec à l'intérieur différentes classes ainsi que 3 designs pattern :

Main.py (fichier principal)

Fenetres.py (qui gère les fenêtres de l'application)

- FntAccueil
- FntConfigJoueurs
- FntConfigTaches
- FntJeu

Interface.py (qui gère l'interface de l'application)

- Fenetres
- Rectangle
- Bouton
- BoiteTexte
- BoiteSaisie

Jeu.py (qui gère le jeu en lui-même)

- Joueurs
- Taches
- Cartes
- CartesFactory (design pattern)
- PartieObserver (design pattern)
- Partie

TestUnit.py (qui fait des tests unitaires)

- TestPartie

IV – Intégration continue

En parlant de tests unitaires, nous avons utilisé la librairie pytest afin de faire des tests sur des bouts de code. Pour ce faire nous avons choisis de faire de l'intégration continue avec github actions. Le principe est d'utiliser un fichier de format yaml qui utilise pytest et fait un test unitaire dès qu'un fichier est push dans le repository github. Malheureusement après plusieurs tentatives, notre test unitaire ne fonctionne toujours pas. Le problème ne vient pas de l'intégration continue qui, elle fonctionne, mais bien du fichier python de test unitaire en lui-même.

Avec des projets de taille conséquente, il n'est pas toujours facile de se retrouver dans tout ce code. C'est pourquoi ici nous avons utilisé Doxygen, un logiciel permettant de générer de la documentation. Pour cela, nous utilisons dans notre code un format spécifique. La documentation se fait entre trois guillemets ‘’’ ‘’’ ainsi que des @ pour chaque partie. Par exemple, pour documenter le nom des classes nous utilisons « @class NomClasse », ou encore pour documenter un argument d'une méthode nous écrivons « @brief argument : description de l'argument » etc. Au final, nous nous retrouvons avec un site où tout notre code est documenté, on s'y retrouve facilement et c'est très accessible.

Voici comment s'est déroulé le projet de la création de cette application de planning poker, merci pour votre attention.