Planning Poker

Réalisation d’un site web

Thiméa Holleville

Alexis Gabrysch

28 decembre 2023

Sommaire

Introduction

Le but du projet est de réaliser une interface interactive qui permet de simuler des parties de planning poker en local. Le planning poker est une méthode qui consiste à jouer avec des cartes les uns contre les autres pour édifier une hiérarchisation des tâches au sein d'une équipe. C'est pour cela que de plus en plus d'entreprises mettent en place ce principe et ont donc besoin d'une interface graphique utilisable, ce que nous avons fait.

Il a fallu réaliser ce projet en utilisant les méthodes agiles vu en cours mais aussi les transmettre d'un point de vue logiciel.

L'interface permet de jouer au planning poker a plusieurs selon plusieurs modes et on peut même sauvegarder une partie en cours pour la reprendre plus tard.

1 Design patterns

1.1 Présentation des design patterns

Nous avons utilisé les patterns suivants :

- Le singleton
- Design Pattern Factory
- Memento

Le singleton consiste à restreindre l'instanciation d'un seul objet à la fois. Nous l'avons utilisé dans notre projet afin de vérifier et de s'assurer qu' une seule partie est jouée à la fois.

Le factory (la fabrique) est un patron de conception qui consiste à créer des objets sans avoir à spécifier la classe exacte de ces objets.

Et le memento est un patron de conception qui permet de sauvegarder un objet et de le rétablir à l'état précédent d'un objet sans révéler les détails de son implémentation.

1.2 Usage pour le site

Le singleton a été utilisé au moment de l'instanciation de la classe Game qui permet de créer un jeu avec un plateau lorsque tous les

paramètres sont complétés et validés.

```
}

// Singleton implementation
const gameInstance = (function() {
    let instance = null;

    function createInstance(players, difficulty) {
        return new Game(players, difficulty);
    }

    return {
        getInstance: function(players, difficulty) {
            if (!instance) {
                instance = createInstance(players, difficulty);
            }
            return instance;
        }
    };
})();
```

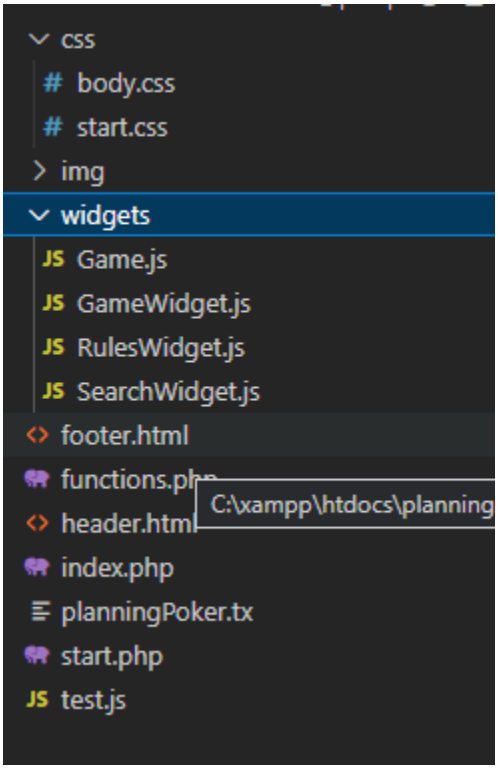
Quant au factory, nous n'avons pas eu le temps de l'implémenter. Mais nous l'aurions mis afin de créer plusieurs instances de jeu dans le localStorage (qui stock les parties terminées ou mises en pause via la carte café) et de vérifier le fonctionnement avec ou sans paramètres.

Puis, le momento est utile pour notre application afin de réutiliser une ancienne sauvegarde du partie, par exemple dans le cas d'une erreur dans le résultat d'une des Features (fonctionnalité choisie), ou encore afin de changer le nombre de joueur etc.

2 Techniques de réalisation

2.1 Architecture

Pour l'architecture du site, nous avons fait plusieurs fichiers/dossiers notamment en javaScript afin que le code soit plus clair.
Nous avons implémenté des classes JavaScript pour la gestion du jeu : RulesWidget, GameWidget et Game.
Voici une image de notre architecture.



L'utilisation de classe, nous a permis de créer des instances unique à chaque utilisation et à les utiliser comme bon nous semble. Ce n'est pas une utilisation basique de classe mais cela nous a permis de pouvoir modifier chaque partie du projet sans casser les autres. De plus, ce sera bien plus simple pour modifier ou ajouter/améliorer le code de certaines parties du jeu.

Et bien sûr, index permet de lancer le code plus simplement, et lance les fichiers nécessaires au bon déroulement du jeu. C'est à dire que

petit à petit que l'utilisateur entre les données, il est facile de valider ou non puis de continuer sur une autre phase du jeu. Par exemple, si tous les joueurs n'ont pas de pseudo, la partie ne peut être lancée. Si la partie ne contient aucune fonctionnalité pour laquelle les joueurs doivent voter, la partie ne peut pas non plus être lancée etc... Le code se déroule petit à petit et par VALIDATION.

2.2 Language et classes

Notre projet est implémenté sous forme de site web, nous utilisons donc les langages du web comme html css mais aussi beaucoup de javascript qui est au final le cœur de notre projet.
Comme montré plus haut, nous utilisons 3 classes dans notre projet : GameWidget qui permet de créer une nouvelle partie, c'est à dire entrer la liste des joueurs ainsi que la difficulté souhaitée ; RulesWidget permet simplement l'affichage des règles avec un changement dynamique selon la règle sélectionnée. Enfin, la classe Game permet de gérer tout le jeu du début de la partie à la fin, ainsi que le stockage en JSON.

La classe GAME est la plus complexe et la plus complète de notre jeu car elle gère le jeu en lui-même. SearchGame permet simplement de reprendre une instance de GAME afin de reprendre la partie.

2.3 Sauvegarde de la partie

La sauvegarde de la partie est effectuée lorsque la partie est terminée, c'est -à -dire lorsque chaque fonctionnalité à été votée et validée (selon les règles de la difficulté en cours. Ou alors lorsque tous les joueurs jouent la carte café, la partie se sauvegarde et se met en pause.

Nous n'avons pas encore eu le temps de la faire mais pour le moment il faut, après avoir mis la carte café pour chaque joueur, recharger la

page et sélectionner la partie en question dans le menu “reprendre une partie”. Il faudrait ajouter un message confirmant la sauvegarde de la partie et un retour au menu principal mais la sauvegarde fonctionne bien.

La sauvegarde se fait donc en stockant en JSON dans le localStorage du navigateur. C’est une propriété qui permet d’accéder à un objet local même lorsque la session navigateur prend fin. Nous avons ajouté une fonctionnalité qui permet de reset le localStorage mais nous aimerions par la suite implémenter une fonction qui permet de choisir une partie sauvegarder et de la supprimer, au cas par cas.

3 Intégration continue

3.1 Définitions

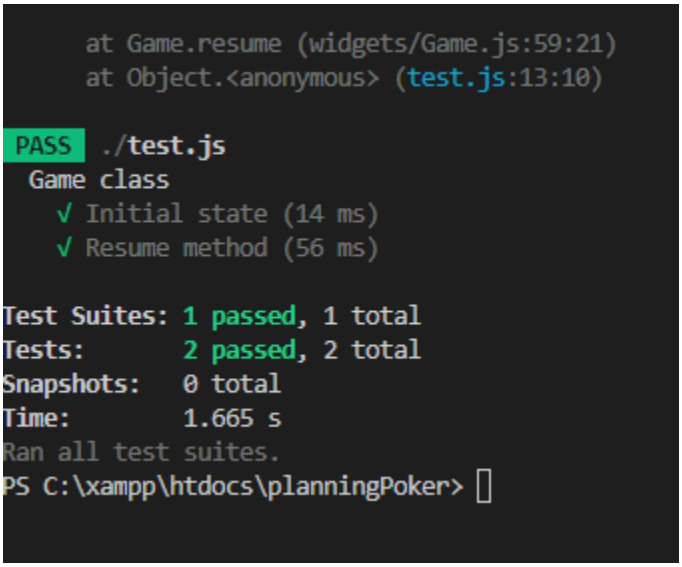
L'intégration continue est une façon de concevoir et de maintenir un programme dans le génie logiciel. Cela consiste à réduire les éventuels risques dûs aux bugs, mais aussi à améliorer la clarté du code.

Pour ce faire, certaines pratiques sont utilisées et notamment la mise en œuvre tout au long du processus de tests unitaires. Il s’agit de recouvrir le programme d’instructions de test avec warning, notamment dans les zones sensibles telles que les interactions utilisateurs (boutons, zone de saisie) pour ainsi révéler où se situent les problèmes et bugs dans le code. Cela facilite grandement la correction des problèmes pour une efficacité et une rapidité de livraison accrues.

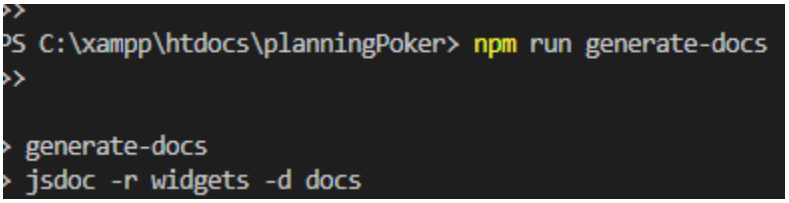
Les projets sont souvent réalisés en équipe, il est ainsi primordial d'améliorer la collaboration entre les membres. Pour ce faire, des outils sont exploités tels que Gît dans notre cas, qui permet de travailler sur la même source, mais en utilisant des branches séparées et ainsi avoir un code toujours à jour pour tous.

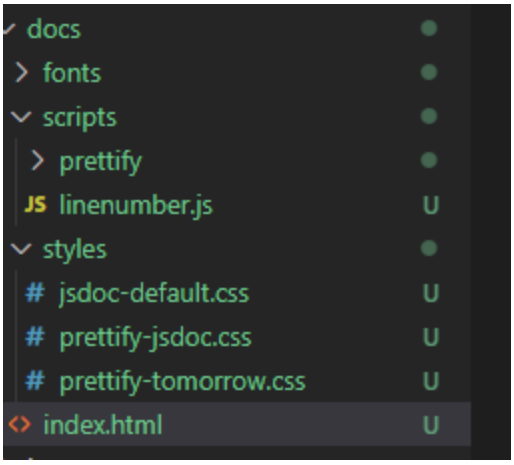
3.2 Intégrations

Pour les tests nous avons implémenté le code avec des try catch.
Puis nous avons utilisé jest afin d'effectuer des test unitaire sur la classe Game via le fichier test.js avec npx jest après avoir configuré les fichiers nécessaires comme jest.config (node js).



Enfin, pour la documentation nous avons utilisé JSDOC afin de créer une documentation du code automatiquement. De plus, nous avons bien commenté le code afin que nous puissions comprendre et reprendre facilement le code plus tard.





4 Bilan

Le projet nous a permis d'utiliser les connaissances vues en cours et en TD et cela de plusieurs façons différentes. Que cela soit d'un point de vue pratique en combinant les acquis d'autres cours et les combiner entre eux dans un espace utilisant les méthodes agiles, mais aussi sur le projet en lui-même, le planning poker et l'instauration des règles de celui-ci. On a ainsi pu jouer à ce jeu à la fois sur l'écran mais aussi dans la vraie vie.

Cela a aussi permis aussi la mise en évidence de la difficulté qui peut faire face lors de la réalisation de ce type de projet collaboratif et abouti. De la communication, à la mise à disposition des membres de l'équipe mais aussi du management du temps et des livrables. Chaque partie du processus est vouée à faire émerger des problèmes, chose qu'on a dû faire face, et ainsi devoir les résoudre organiquement.