Note :

* le rapport en PDF (format libre)
* Rendre avant *vendredi 22 décembre à 17h59*
* Le rapport doit comprendre
  + une présentation globale du projet
  + une motivation
    - des choix d’architecture
      * *( la façon dont vous les avez implémenté dans votre projet.)*
    - des patterns choisis
      * *(utiliser au moins trois des design patterns présentés en cours.)*
  + explication
    - des choix d’architecture
    - et des patterns choisis
      * en s’aidant de diagrammes appropriés et adaptés au degré de précision et au type d’explication.
        + *Par exemple des diagrammes de classe, mais pas que cela, et pas de plats de spaghettis généré automatiquement représentant tout le code.*
    - votre intégration continue
      * *expliquer comment vous avez mis en place*
      * à minima, automatisation des tests
      * et génération de la documentation

Sommaire généré (On peut toujours modifier) :

[Présentation globale du projet 2](#_Toc150521879)

[Application de planning poker 2](#_Toc150521880)

[Choix d’architecture 2](#_Toc150521881)

[Motivation 3](#_Toc150521882)

[Explication 3](#_Toc150521883)

[Choix des patterns 3](#_Toc150521884)

[Motivation 3](#_Toc150521885)

[Explication 3](#_Toc150521886)

[Intégration continue 3](#_Toc150521887)

[Explication 3](#_Toc150521888)

[Automatisation des tests 3](#_Toc150521889)

[Génération de la documentation 3](#_Toc150521890)

# Présentation globale du projet

## Application de planning poker

L'objectif de l'application est de permettre à des joueurs de faire une partie de planning poker, en respectant les règles vues en cours.

L'application peut être à distance (chaque joueur utilise son propre dispositif) ou locale (les joueurs choisissent chacun à leur tour leurs cartes).

Un menu permet de décider du nombre de joueurs et de rentrer un pseudo pour chacun des joueurs. Le menu doit aussi permettre de choisir parmi différentes règles de planning poker (règles strictes, moyenne, médiane, etc.)

On doit pouvoir entrer une liste de fonctionnalités (backlog) en JSON (vous êtes libre d'utiliser la structure que vous souhaitez).

Une fois que chacun a voté, l'application valide ou non la fonctionnalité en fonction des règles choisies via le menu. Si la fonctionnalité n'est pas validée, on recommence le vote.

Lorsque tout le backlog est validé, l'application enregistre un fichier JSON avec, pour chaque fonctionnalité, la difficulté estimée par l'équipe.

**Note :**Si tous les joueurs utilisent la carte café, l'application doit enregistrer un fichier JSON avec l'état d'avancement du backlog. Ce fichier JSON doit pouvoir être chargé via le menu pour "reprendre" une partie.

**Note 2 :**Vous êtres très fortement encouragés à ajouter des fonctionnalités qui vous semblent utiles au bon déroulement du planning poker (chronomètre, espace de discussion, etc.)

Vous devez, dans votre projet, utiliser au moins trois des design patterns présentés en cours. Vous devez justifier l'utilisation de ces designs patterns dans votre rapport, ainsi que de la façon dont vous les avez implémentés dans votre projet.

# Choix d’architecture

Java

IntelliJ

## Motivation

## Explication

# Choix des patterns

Diagramme version 1

Une image contenant texte, capture d’écran, diagramme, conception

Description générée automatiquement

Il manque :

- un constructeur de la classe Joueur

- un attribut identifiant unique : «  monId » à Joueur

- un constructeur de Plannig Poker

Maude : s’est occupé de l’implémentation de l’affichage du menu (selection et récupération du nombre de joueur, des pseudos des joueurs)

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Une image contenant texte, capture d’écran, Police

Description générée automatiquementUne image contenant texte, capture d’écran, Police

Description générée automatiquement

On a pu récupérer les valeurs

Fina : Dans le fichier main, je teste les classes pour jouer à la partie PlanningPoker. Je crée un tableau de joueurs, j’insère un mode de jeu.

J’implémente également le fichier JSON (le Backlog), pour obtenir une liste de fonctinnalité.

->Ctr+Shit+A > Add FrameWorksSupport> Maven> pom.xml >

<dependencies>  
 <!-- Autres dépendances de votre projet -->  
  
 <!-- Dépendance Jackson -->  
 <dependency>  
 <groupId>com.fasterxml.jackson.core</groupId>  
 <artifactId>jackson-databind</artifactId>  
 <version>2.13.4.1</version> <!-- Utilisez la dernière version disponible -->  
 </dependency>  
  
</dependencies>

Ici, j’ai rajouter les dependances Jackson. Après, Ne pas oublier de faire clic-droit sur pom.xml > « Maven »> « Reload Project» .

Pour compiler maintenant : CAPICAPI\_Projet\_implementation diagramme des classes/src /src /main/java /Fenetre /Fenetre.java

## Motivation

## Explication

# Intégration continue

## Explication

### Automatisation des tests

### Génération de la documentation