Note :

* le rapport en PDF (format libre)
* Rendre avant *le jeudi 28 décembre à 12h00*
* 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

sujet : 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.

Sommaire généré (On peut toujours modifier) :

[Présentation globale du projet 3](#_Toc154566350)

[Application de planning poker 3](#_Toc154566351)

[Choix d’architecture 3](#_Toc154566352)

[Motivation 3](#_Toc154566353)

[Explication 3](#_Toc154566354)

[Choix des patterns 6](#_Toc154566355)

[Motivation 8](#_Toc154566356)

[Explication 8](#_Toc154566357)

[Singleton 8](#_Toc154566358)

[Intégration continue 8](#_Toc154566359)

[Explication 8](#_Toc154566360)

[Automatisation des tests 8](#_Toc154566361)

[Génération de la documentation 8](#_Toc154566362)

# Présentation globale du projet

## Application de planning poker

Le projet vise la conception et le développement d'une application de Planning Poker, un jeu utilisé pour évaluer la complexité des fonctionnalités d'un backlog de développement. L'utilisation de l'application se fait localement, où les joueurs sélectionnent individuellement leurs cartes.

Elle offre la flexibilité de définir un nombre de joueurs compris entre 2 et 12, chacun possédant un pseudo unique. De plus, l'application propose deux modes de jeu distincts : unanimité et moyenne. Les fonctionnalités du backlog sont entrées sous forme de liste JSON.

Chaque joueur a en sa possession un jeu de cartes avec les valeurs suivantes : 0, 1, 2, 3, 5, 8, 13, 20, 40, 100, le point d'interrogation, et la tasse de café. Le point d'interrogation est utilisé lorsque l'estimateur ne peut pas estimer la tâche, et la tasse de café exprime le besoin de faire une pause. Plus le joueur vote une carte avec une valeur élevée, plus la tâche a une charge importante.

Chaque joueur vote pour chaque fonctionnalité, et l'application valide ou rejette la fonctionnalité en fonction des règles sélectionnées. En cas de rejet, un nouveau vote est initié. Une fois toutes les fonctionnalités validées, l'application enregistre un fichier JSON contenant les estimations de difficulté pour chaque fonctionnalité.

Si tous les joueurs votent pour la "carte café", cela enregistre l'état d'avancement du backlog dans un fichier JSON. Ce dernier peut être chargé via le menu pour reprendre une partie précédente.

# Choix d’architecture

## Motivation

Nous avons opté pour le langage Java en raison de sa polyvalence et de sa popularité dans le développement logiciel. Java offre une portabilité élevée, ce qui signifie que notre application pourrait être exécutée sur différentes plateformes sans nécessiter de modifications majeures. De plus, nous avons choisi IntelliJ comme IDE en raison de sa facilité d'utilisation.

## Explication et utilisation

On s’est concentrée sur le développement de l'interface utilisateur de l'application de Planning Poker, Notamment, sur la mise en place des différents menus, ainsi que la navigation fluide vers le plateau de jeu.

Le menu d'accueil, première interaction visuelle avec les utilisateurs, propose des options pour commencer une nouvelle partie ou reprendre une partie en cours, si disponible.

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

Nous avons choisi de limiter le nombre de joueurs à douze pour optimiser l'expérience de jeu.

Une image contenant texte, Appareils électroniques, capture d’écran, logiciel

Description générée automatiquement

Concernant la gestion du nombre de joueurs, on a mis en place des contrôles pour garantir que chaque pseudo soit unique et non nul, évitant ainsi des confusions lors du vote.

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquementUne image contenant texte, capture d’écran, logiciel, affichage

Description générée automatiquement

Ensuite, la conception inclut un menu permettant aux joueurs de choisir entre les modes de jeu, chacun étant expliqué avec ses règles spécifiques, notamment "Unanimité" et "Moyenne".

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Par la suite, nous avons introduit un menu qui permet aux joueurs de rédiger les fonctionnalités qu'ils souhaitent voter au cours de cette partie de Planning Poker. Nous avons pris soin de garantir une transition fluide entre ces étapes, assurant ainsi une compréhension claire des options disponibles pour les joueurs.

Une image contenant texte, capture d’écran, logiciel, Police

Description générée automatiquement

La transition vers le plateau de jeu a été réalisée avec succès, offrant aux joueurs la possibilité d'exprimer leurs votes de manière fluide et intuitive. Lorsqu'un joueur sélectionne une carte, son contour change de couleur pour une meilleure visibilité. De plus, l'intitulé de la fonctionnalité, le mode de jeu choisi, le tour en cours, ainsi que le nom du joueur appelé à voter sont clairement affichés, garantissant une expérience de jeu claire et interactive pour tous les participants.

Pour améliorer l'interaction, un chronomètre a été intégré. Lorsque le résultat du vote est en attente avec le symbole « ? », le chronomètre se met en pause, permettant aux joueurs de discuter de la fonctionnalité avant que le chronomètre ne reprenne. Cette fonctionnalité contribue à favoriser une communication efficace et une compréhension mutuelle entre les joueurs, enrichissant ainsi l'expérience globale du jeu.

Une image contenant texte, capture d’écran

Description générée automatiquement Une image contenant texte, capture d’écran, Police

Description générée automatiquement

 Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

Lorsque tous les joueurs ont voté pour une fonctionnalité ou que la partie est terminée, un message s'affiche pour informer les joueurs. Ce message peut indiquer que la partie est terminée et a été enregistrée dans un fichier JSON ou qu'elle a été enregistrée dans un fichier JSON. Après avoir cliqué pour fermer ce message, la fenêtre se ferme automatiquement après quelques secondes.

# Choix des patterns

Single patterns

Diagramme version 1

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

Description générée automatiquement

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 fonctionnalité.

->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 rajouté les dépendances 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

### Singleton

Le premier design pattern que nous avons utilisé est le singleton. Dans notre implémentation, la classe PlanningPoker a été conçue comme une classe singleton, assurant ainsi une unique instance de cette classe dans tout le programme. Les variables de la classe, sont privées pour garantir un contrôle strict sur leur accès.

Le constructeur de la classe PlanningPoker a été défini comme privé, empêchant ainsi la création d'instances multiples de la classe. Pour accéder à l'instance unique, nous avons créé une méthode statique getInstance(). Cette méthode crée l'instance si elle n'existe pas encore, sinon elle renvoie simplement l'instance existante.

Cette approche assure que notre application ne dispose que d'une seule instance de la classe PlanningPoker, évitant ainsi des problèmes potentiels liés à la gestion de multiples instances.

# Intégration continue

## Explication

### Automatisation des tests unitaires : JUnit

On a créé quatre fichiers de tests unitaires dans le répertoire "Test/java" de notre projet, avec le chemin complet étant "CAPI\CAPI\_Projet\src\test\java".

Ces fichiers de test sont conçus pour vérifier le bon fonctionnement de certaines parties de notre code.

### Génération de la documentation : Doxygen

Dans le cadre de notre projet de Planning Poker, nous avons pris la décision d'utiliser Doxygen comme outil de génération de documentation. Doxygen est un générateur de documentation largement utilisé dans l'écosystème du développement logiciel. Son principal objectif est de simplifier et d'automatiser le processus de création de documentation à partir du code source.

Doxygen :

* Copiez le chemin d'installation de Doxygen (par exemple, C:\Program Files\doxygen\bin).
* Ajoutez ce chemin au PATH système. Vous pouvez le faire en suivant ces étapes :
  + Cliquez avec le bouton droit sur "Ce PC" ou "Ordinateur" sur le bureau ou dans l'Explorateur de fichiers.
  + Choisissez "Propriétés".
  + Cliquez sur "Paramètres système avancés" sur le côté gauche.
  + Cliquez sur "Variables d'environnement".
  + Dans la section "Variables système", recherchez la variable "Path" et cliquez sur "Modifier".
  + Cliquez sur "Nouveau" et ajoutez le chemin d'installation de Doxygen.
  + Cliquez sur "OK" pour fermer les fenêtres.
* Redémarrez votre fenêtre de commande : Fermez toutes les fenêtres de commande ouvertes et ouvrez-en une nouvelle. Cela garantira que les modifications apportées au PATH prennent effet.
* Maintenant, essayez d'exécuter à nouveau la commande doxygen -g Doxyfile dans la nouvelle fenêtre de commande.
* (tu devrais avoir ça : C:\Users\user\_name>doxygen -g Doxyfile

Configuration file 'Doxyfile' created.

Now edit the configuration file and enter

Doxygen

to generate the documentation for your project)

* Tu ouvres Doxyfile avec VSC
* Tu modifies les variables suivantre :
  + PROJECT\_NAME           = "CAPI\_Projet"
  + PROJECT\_NUMBER         = 1
  + OUTPUT\_DIRECTORY       = "Chemin\Conception\_agile\_de\_projets\_info\projet\CAPI\CAPI\_Projet\Doc"
  + OUTPUT\_LANGUAGE        = French
  + INPUT                  = "Chemin\Conception\_agile\_de\_projets\_info\projet\CAPI\CAPI\_Projet\src\main\java"
* Après tu fais doxygen Doxyfile
* Et tu ouvres inndex.html qui se trouve dans le dossier "Chemin\Conception\_agile\_de\_projets\_info\projet\CAPI\CAPI\_Projet\Doc\html"