VIVIER Maude

RANDRIAMITANDRINA Finaritra

Planning Poker

Application sous Java

VIVIER Maude

RANDRIAMITANDRINA Finaritra

Table des matières

VIVIER Maude

RANDRIAMITANDRINA Finaritra

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.

Particularité :
- Nous avons mis en place un chronomètre pour estimer la durée de la partie du planning Poker, il est également enregistré dans le fichier JSON.
- Si tous les joueurs utilisent la carte café, l'application enregistre un fichier JSON avec l'état d'avancement du backlog. Ce fichier JSON peut être chargé via le menu dans "reprendre une partie".
- Si tous les jours optent pour la carte "?" (donc s'ils ne savent pas comment estimer la difficulté) un temps de pause de 10 secondes s'enclenche. Le temps que tout le monde se concerte pour décider d'un niveau de difficulté. Puis, on reprend le tour.
- Dans notre projet, nous avons utilisé un seul design patterns, le singleton.

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.

Prérequis : Télécharger Maven

Dans notre projet, nous utilisons un fichier JSON comme structure pour le Backlog, afin d'obtenir une liste de fonctionnalité avec ces difficultés. Pour gérer ce fichier (ex : sauvegarder ou récupérer le fichier JSON), nous utilisons l'outil de gestion **Maven.**

Ainsi, **si vous n'avez pas encore Maven** d'installer sur IntelliJ, vois les étapes à suivre pour l'obtenir:

VIVIER Maude

RANDRIAMITANDRINA Finaritra

1. Faire **Ctrl + Shit + A**, cette commande permet de rechercher une commande
2. Rechercher la commande **Add FrameWorksSupport**
3. Sélectionner **Maven**
4. Aller dans le fichier **pom.xml**
5. Rajouter la dépendance **Jackson**, voir image ci-dessous :

```
<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>
    </dependency>
</dependencies>
```

6. Ne pas oublier de faire **clic-droit sur pom.xml > Maven > Reload Project .**

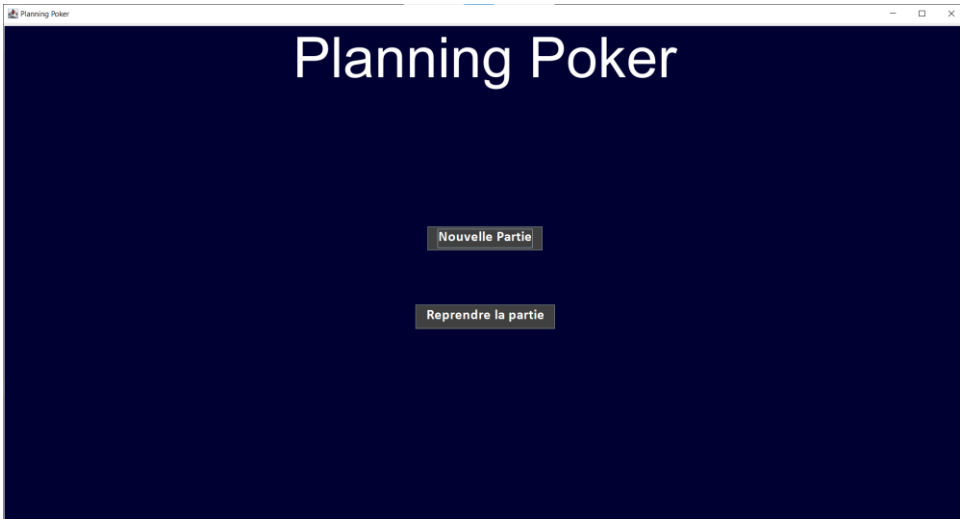
Run le projet

1. Sur IntelliJ, ouvrir le projet CAPI_*Projet*.
2. Lancer le programme principal *Fenetre*
   Le programme principal *main* se trouve dans le fichier *Fenetre.java*
   Son chemin d'accès est : *CAPI\CAPI_Projet\src\main\java\Fenetre.java*
3. Pour lancer une partie, vous pouvez :
   a. Lancer une nouvelle partie de Planning Poker
   b. Ou reprendre une partie déjà sauvegardée dans un fichier JSON

Explication et utilisation de l'application

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, étant la première interaction visuelle avec les utilisateurs, propose des options pour commencer une nouvelle partie ou reprendre une partie en cours, si disponible. Le bouton « reprendre une partie» permet de récupérer le fichier JSON *backlog.json ,* qui se trouve dans le chemin d'accès *CAPI\CAPI_Projet\src\json .*



VIVIER Maude

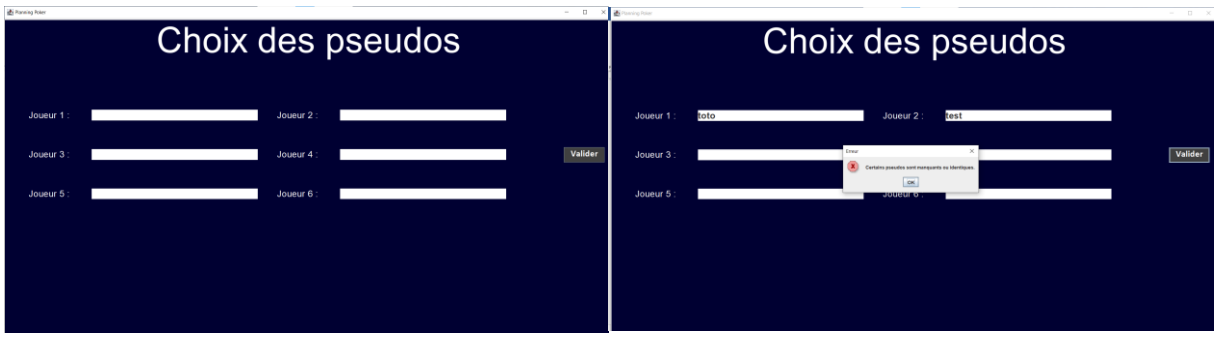
RANDRIAMITANDRINA Finaritra

Voici un exemple de structure du fichier JSON.


```
{
  "fonctionnalites" : [ {
    "description" : "Fonction1",
    "difficulte" : "3"
  }, {
    "description" : "Fonction2",
    "difficulte" : "8"
  } ],
  "modeDeJeu" : "MOYENNE",
  "joueurs" : [ {
    "id" : 1,
    "pseudo" : "Bob",
    "voteCourant" : "8"
  }, {
    "id" : 2,
    "pseudo" : "Bobette",
    "voteCourant" : "8"
  } ],
  "tempsEcoule" : 19000
}
```


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

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.

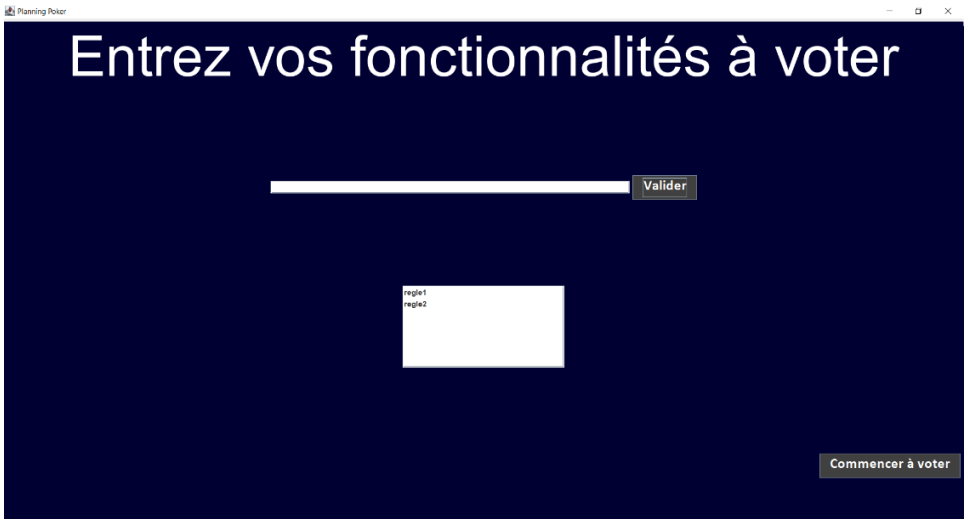


VIVIER Maude

RANDRIAMITANDRINA Finaritra

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". Les règles sont affichées en bas à droite.

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.

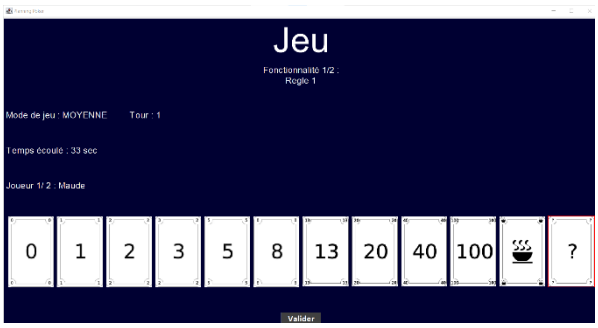


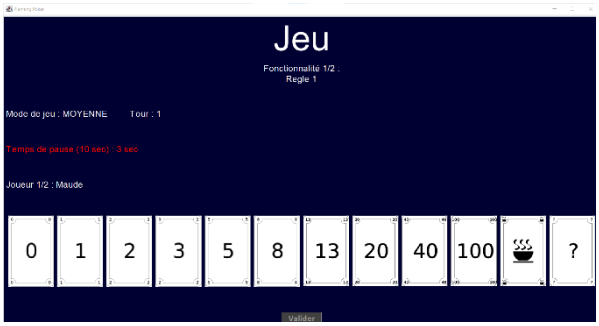
La transition vers le plateau de jeu a été réalisée avec succès, offrant aux joueurs la possibilité d'exprimer leurs votes de manière fluide et intuitive. Lorsqu'un joueur sélectionne une carte, son contour change de couleur pour une meilleure visibilité (elle devient rouge). De plus, l'intitulé de la 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 voter, sélectionner une carte, cette dernière aura un contour rouge puis cliquer sur le bouton « Valider ».

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.





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.

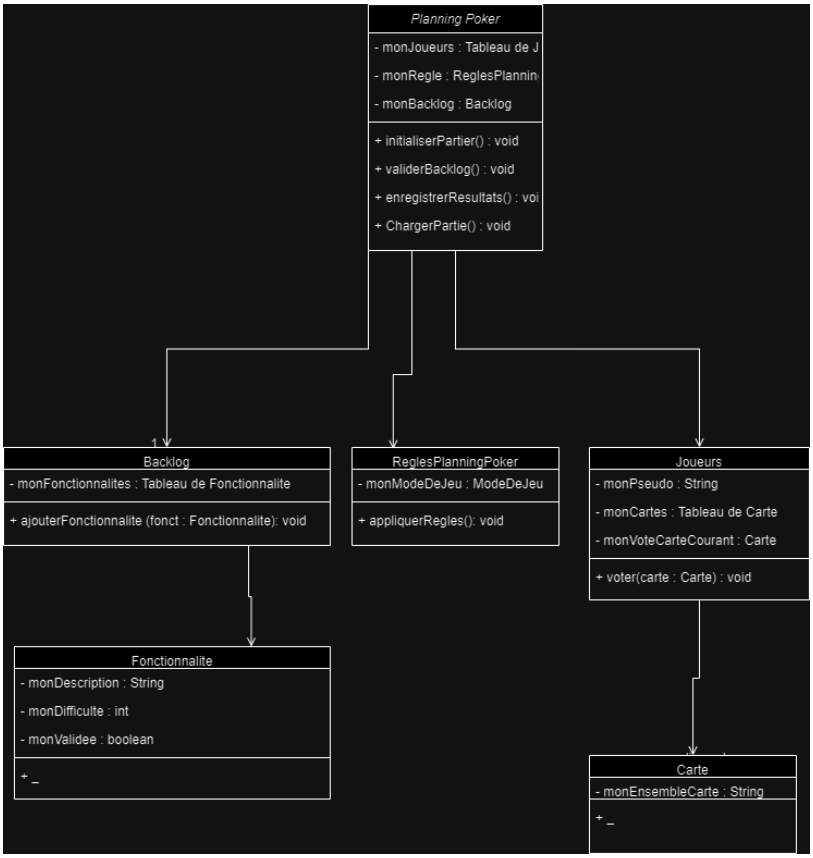
Choix d'organisation

Diagramme

Voici comment nous avons imaginé le diagramme des classes au début de notre projet.

VIVIER Maude

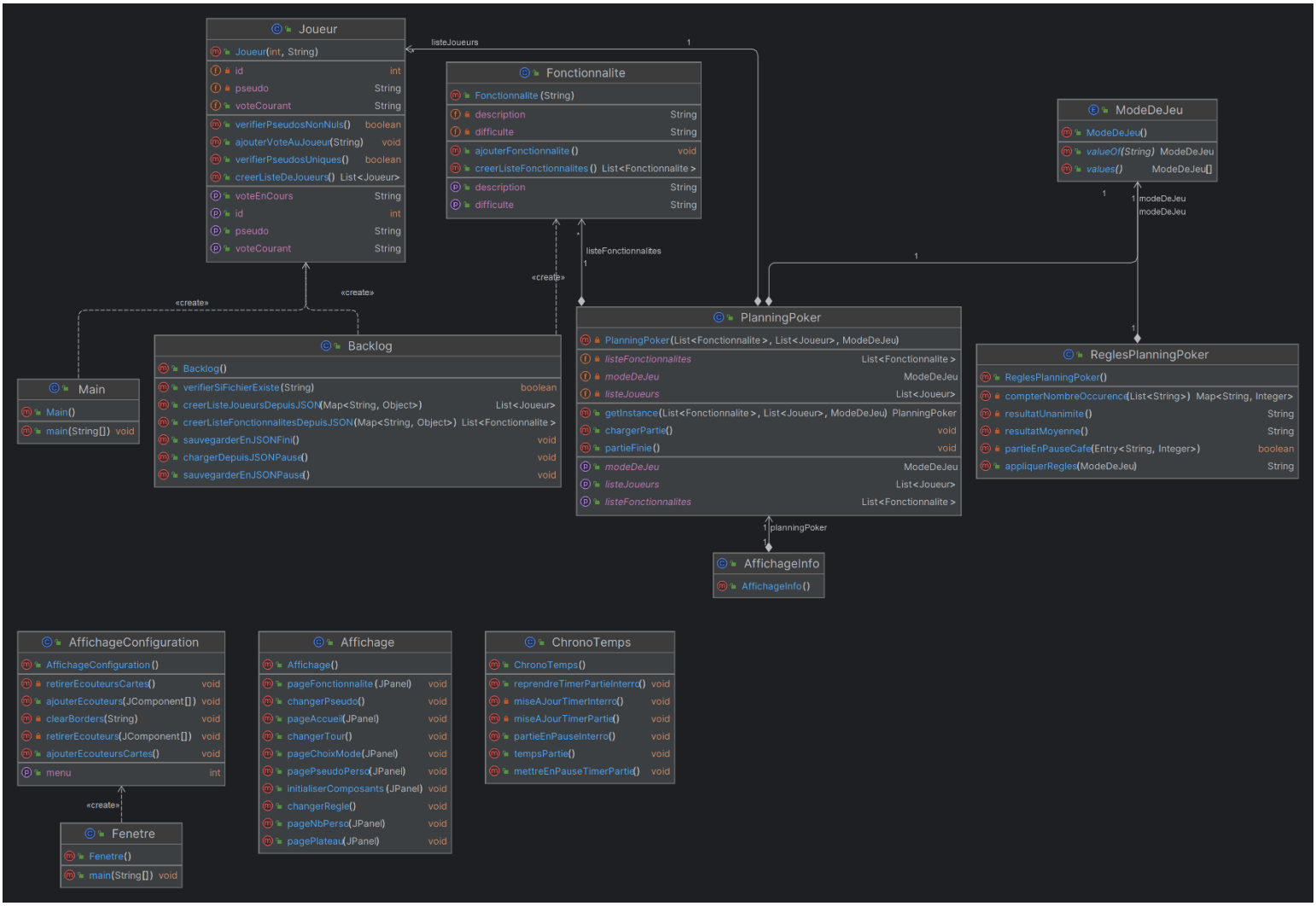
RANDRIAMITANDRINA Finaritra



Voici le diagramme des classes finals générés avec l'éditeur de code IntelliJ:

VIVIER Maude

RANDRIAMITANDRINA Finaritra



Design Pattern : Singleton

Motivation et explication

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.

VIVIER Maude

RANDRIAMITANDRINA Finaritra

Intégration continue

Tests unitaires : JUnit

On a créé quatre fichiers de tests unitaires dans le répertoire "*Test/java*" de notre projet. Le chemin d'accès est "*CAPI**CAPI_Projet**src**test**java*". Ces fichiers de test sont conçus pour vérifier le bon fonctionnement de certaines parties de notre code. Nous avons utilisé le framework de test unitaire **Junit**.

- *BacklogTest* : Teste la sauvegarde et le chargement en JSON pendant une pause.
- ChronoTempsTest : Teste le fonctionnement du timer de partie.
- FenetreTest : Teste la création de la fenêtre.
- ReglesPlanningPokerTest : Teste l'application des règles en mode « UNANIMITE » et « MOYENNE».

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. Vous trouverez le fichier de documentation de projet dans le fichier **index.html.**

Voici son chemin d'accès **: CAPI\CAPI_Projet\Doc\html\index.html**