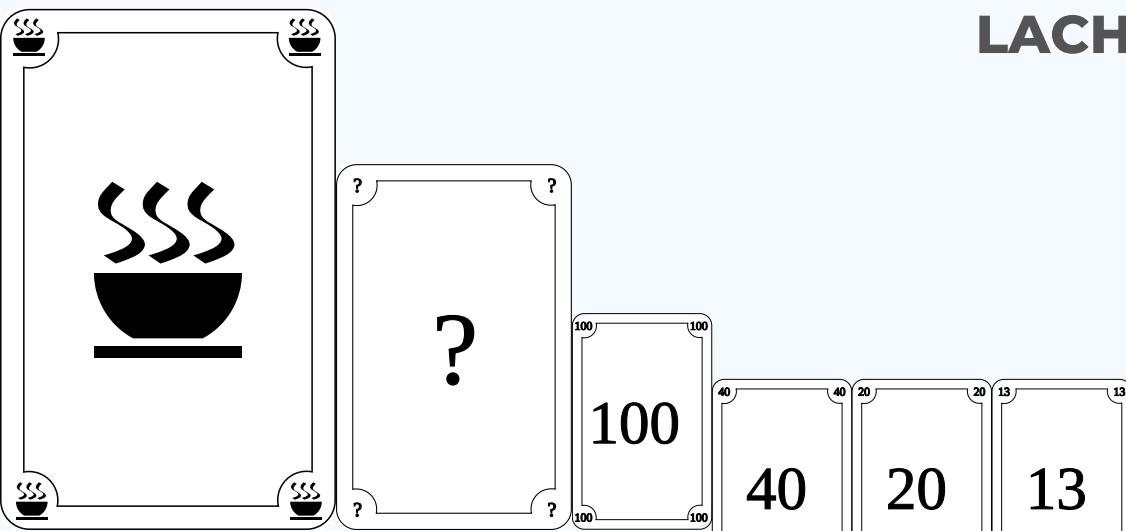


RAPPORT CONCEPTION AGILE DE PROJETS INFORMATIQUES - M1 INFO

APPLICATION PLANNING POKER

Romain DUC
Senebou DIARRA
Encadrant : Valentin
LACHAND-PASCAL



SOMMAIRE

01 Présentation du projet

02 Choix techniques

A - Technologies

B - Architectures

C - Classes

03 Mise en place de l'intégration continue

A - Tests unitaires

B - Documentation

04 Annexes

PRÉSENTATION DU PROJET

L'objectif principal de ce projet est de développer une application dédiée au Planning Poker, également connu sous le nom de Scrum Poker, une méthode largement adoptée pour estimer les tâches au sein des équipes de développement logiciel agiles. Cette approche collaborative et ludique implique l'utilisation d'un jeu de cartes numériques, permettant aux membres de l'équipe d'attribuer des estimations à différentes tâches.

Le Planning Poker vise à faciliter la discussion au sein de l'équipe, favorisant ainsi l'émergence d'une estimation consensuelle qui prend en compte l'expertise et les perspectives de chacun. Cette méthode aide les équipes à établir des priorités et à planifier efficacement leur travail dans le contexte de la gestion de projets Agiles.

Notre application, disponible en local, offre aux joueurs la possibilité de participer à une session de Planning Poker. Elle dispose d'un menu permettant de définir le nombre de joueurs et d'attribuer un pseudonyme à chacun. De plus, elle propose différentes règles de Planning Poker parmi lesquelles choisir, à savoir : stricte, moyenne et médiane.

Ces règles déterminent la validation d'une fonctionnalité lors du vote :

- Stricte : uniquement si tous les joueurs sont unanimes.
- Moyenne : si une fonctionnalité obtient plus de la moitié des votes.
- Médiane : la fonctionnalité aura l'estimation du vote qui se situe entre les opposés

Les joueurs ont ensuite la possibilité d'écrire manuellement leurs fonctionnalités ou de les importer au format JSON. Une fois ces étapes complétées, chaque joueur peut voter pour les fonctionnalités présentées successivement, parmi les cartes suivantes 0, 1, 2, 3, 5, 8, 13, 20, 40, 100, ☕, ?.

L'application valide ou rejette une fonctionnalité en fonction des règles définies dans le menu. En cas de rejet, un nouveau vote est lancé.

Une fois toutes les fonctionnalités validées, l'application génère un fichier JSON contenant, pour chaque fonctionnalité, la difficulté estimée par l'équipe. De plus, si tous les joueurs utilisent la carte café, l'application enregistre un fichier JSON reflétant l'état d'avancement du backlog. Ce fichier peut être chargé ultérieurement via le menu pour reprendre une partie.

CHOIX TECHNIQUES

A - Technologies

Pour la réalisation de cette application, nous avons choisi d'utiliser le langage python. En effet, c'est un langage qui permet d'avoir énormément de solution en ce qui concerne les applications.

Nous avons pu alors utiliser :

- Pygame : C'est une bibliothèque qui permet de créer des applications en Python (orientées "jeux")
- Flask : C'est un framework micro-web de Python utilisé pour créer des applications en ligne. Il offre une méthode simple et adaptable pour développer des applications Web et des API (Application Programming Interfaces) basées sur Python

Flask est connu pour sa simplicité et sa flexibilité. Il n'impose pas de structure ou de dépendances particulières.

Par ailleurs nous avons choisi comme IDE, Visual Studio Code.

B - Architecture

Concernant le choix de nos architectures, nous avons opté pour un modèle architectural et 3 design patterns.

1 - Modèle-Vue-Contrôleur

Nous avons initialement adopté le modèle architectural MVC (Modèle-Vue-Contrôleur).

Ce modèle offre une séparation claire de la structure d'une application en trois composants principaux : les modèles, les vues et les contrôleurs.

L'adoption de ce modèle a grandement contribué à la structuration efficace de notre code. Il nous a également conféré la flexibilité nécessaire pour effectuer des modifications sur l'interface utilisateur et les contrôles de manière indépendante du modèle.

2 – Observer Pattern

En ce qui concerne notre premier design pattern, nous avons décidé d'utiliser le pattern Observateur. Son activité principal est de notifier ses observateurs en cas de changement d'état. C'est pour cela que nous avons décider d'utiliser ce modèle afin de permettre aux joueurs d'observer les changements dans l'état de la partie.

Par exemple, lorsqu'une fonctionnalité est validée ou que le vote est terminé, l'Observateur va notifier tous les joueurs de cette mise à jour.

Par ailleurs, chaque joueur est un observateur qui est enregistrer auprès d'une classe nommé "VoteManager" afin d'être notifié des résultats.

Création d'un système de publication/souscription où les clients peuvent s'inscrire pour recevoir des mises à jour.

3 – Singleton Pattern

Par la suite, nous avons opté pour le pattern Singleton. En effet, nous avions besoin d'un modèle nous permettant de vérifier l'unicité des fonctionnalités, c'est-à-dire une garantie qu'il n'y ai qu'une seule instance des fonctionnalités mais aussi du gestionnaire de jeu.)

Judicieux pour gérer l'instance de la base de données ou le gestionnaire de session. Cela garantit qu'une seule instance est utilisée dans toute l'application.

4 – Factory Pattern

Ce pattern peut être utilisé pour créer des objets basés sur les règles du jeu sélectionnées (strict, moyenne, médiane, etc.).

Création d'une RuleFactory qui génère des instances de règles spécifiques selon le choix de l'utilisateur.

Autre DP pertinent : Strategy Pattern afin mettre en œuvre différents modes de jeu sous forme d'algorithmes interchangeables.

AverageRule	MedianRule	RuleFactory	SessionManager	StrictRule
evaluate_votes(votes)		get_rule(rule_type)	get_instance()	evaluate_votes(votes)

Figure 1 : Exemple de classes pour Singleton & Factory

C - Classes, conception

Afin de mieux structurer le modèle, nous avons utilisé des classes.

Premièrement, il y a la classe Player qui gère les joueurs (comprenant un nom d'utilisateur)

Puis la classe Tache, qui gère les différentes fonctionnalités liés aux intitulés des tâches afin de permettre aux joueurs d'effectuer des estimations

Le champ 'votes' de la classe **Tache** associe à un joueur son vote pour une tâche donnée.

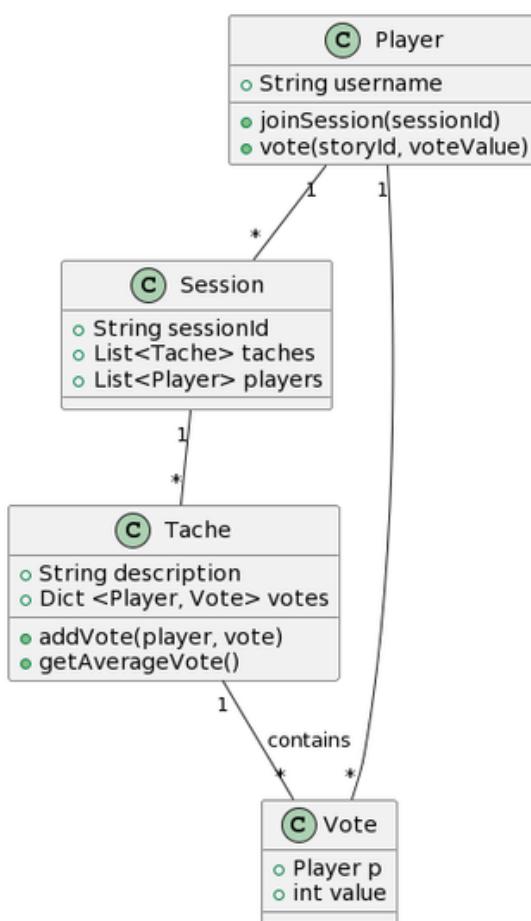


Figure 2 : Diagramme de classes pour le modèle

INTÉGRATION CONTINUE

A – Mise en place

Pour la mise en place de l'intégration continue, nous avons opté pour la génération de documentation à l'aide de Doxygen. Doxygen est un outil de documentation largement répandu pour les langages de programmation tels que C, C++, et Java, mais il peut également être configuré pour documenter du code Python.

Nous avons également utilisé des tests unitaires. Ces tests restent très basiques, triviaux et pourraient être élargis pour couvrir plus de cas de figure.

B – Tests unitaires

Pour exécuter les tests unitaires exécuter le fichier unittests.py à la racine du projet, après avoir installé le package unittest (requirements.txt)

```
python3 unittests.py
```

La mise en place de tests permet de détecter les bugs et les problèmes dans le code dès les premières étapes du développement. Cela facilite la correction des erreurs avant qu'elles ne soient trop coûteuses et/ou complexes à résoudre. Les tests unitaires peuvent être automatisés, ce qui permet de les exécuter fréquemment dans le cycle de vie d'une application.

C - Documentation

La documentation a été générée à l'aide d'un Doxyfile, ce dernier étant un fichier de configuration qui indique à Doxygen comment générer la doc.

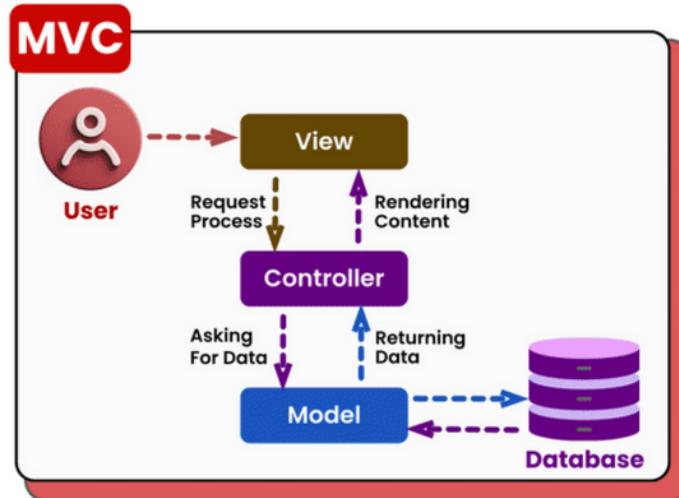
Les paramètres importants à configurer ont été les suivants :

- PROJECT_NAME : Le nom du projet.
- INPUT : Les répertoires et fichiers à documenter.
- FILE_PATTERNS : Les motifs de fichiers à inclure (par exemple, *.py pour les fichiers Python).
- EXTRACT_ALL : Permet d'extraire toutes les informations du code, même si elles ne sont pas documentées.

Consulter la documentation générée : Après l'exécution de Doxygen (*doxygen conf/Doxyfile*), la documentation générée se trouve généralement dans un dossier nommé html. Ouvrez le fichier index.html.

ANNEXES

Annexe 1 : Pattern architectural MVC



crédits : blog.amigoscode.com

Annexe 2 : Captures d'utilisation

A screenshot of a web browser showing a landing page for a planning poker application. The address bar at the top shows the URL <http://127.0.0.1:8080>. The main content area displays the text "Planning poker!". Below this are three blue buttons: "Create session", "Join session", and "Write tasks".

Page d'accueil pour créer/rejoindre une session, et ou créer une session en indiquant les tâches

i http://127.0.0.1:8080/enter_backlog?

Planning poker!

Enter Backlog

Task:

Changer la photo de profil

Task:

Accéder à l'interface administrateur

Task:

Modifier des caractéristiques de son profil

Task:

Planifier un prêt de trottinette

Task:

Se déconnecter du service

Submit

Entrer manuellement les intitulés des user-stories

i http://127.0.0.1:8080/addplayer?

Planning poker!

Add User

Username

Eric

Username

Jérôme

Username

Antoine

Username

Elisa

Username

Olivier

Submit!

Ajouter des joueurs

i http://127.0.0.1:8080

Planning poker!

e79a91ef-6d90-4cb4-ad27-9c3a2656c494

Olivier

[Start poker](#) [Export data](#) [Reset](#)

Players:

Eric
Jérôme
Antoine
Elisa
Olivier

Tasks:

Changer la photo de profil
Accéder à l'interface administrateur
Modifier des caractéristiques de son profil
Planifier un prêt de trotinette
Se déconnecter du service

Card configuration:

1 2 3 5 8 13 20 40 100 ?

Page récapitulative avant de lancer

i http://127.0.0.1:8080/results/0

Planning poker!

Results round 1

Task: Changer la photo de profil

Players voted: 1

Average score: 3.0

[Change vote](#) [Next task](#)

Votes:
Olivier: 3

Resultat d'un tour

Planning poker!

e79a91ef-6d90-4cb4-ad27-9c3a2656c494

Olivier

Round 2

Task: Accéder à l'interface administrateur



Planning poker!

e79a91ef-6d90-4cb4-ad27-9c3a2656c494

Start poker

Export data

Reset

Players:

Eric
Jérôme
Antoine
Elisa
Olivier

Tasks:

Changer la photo de profil: 3.0
Accéder à l'interface administrateur
Modifier des caractéristiques de son profil
Planifier un prêt de trottinette
Se déconnecter du service

Card configuration:



i http://127.0.0.1:8080

Planning poker!

e79a91ef-6d90-4cb4-ad27-9c3a2656c494

Start poker Export data Reset

Players:

Eric
Jérôme
Antoine
Elisa
Olivier

Tasks:

Changer la photo de profil: 3.0
Accéder à l'interface administrateur
Modifier des caractéristiques de son profil
Planifier un prêt de trottinette
Se déconnecter du service

Card configuration:

Réinitialiser la session en cours

i http://127.0.0.1:8080

Planning poker!

ddd6690e-0f54-45ca-b34b-ecdc6092853c

Start poker Export data Reset

Players:

ercf
er
as
ju
pom

Tasks:

Add login and registration functionality
Design and implement the main dashboard
Allow users to modify their profile
Allow users to add a picture to their profile

Card configuration:

Exemple de session avec la liste 'backlog.json'