Groupe I
COTTREAU Arthur
LIPKO Mathias
ZHENG Zehua

Rapport de Projet

Objectif:

Faire une application Planning Poker locale destinée à évaluer l'effort et la difficulté associés au développement de certaines fonctionnalités. Cette application doit au moins avoir deux modes de jeux.

L’utilisateur doit pouvoir entrer dans une liste de fonctionnalités qui sera sauvegardée dans un JSON. Il doit également être capable d'importer ce JSON pour débuter ou poursuivre une partie, et sauvegarder les résultats dans un JSON une fois que la partie est terminée. Si, au cours d'une partie, tous les joueurs utilisent la carte café, l'application doit enregistrer la progression dans un fichier JSON, permettant ainsi de reprendre la partie ultérieurement.

Organisation:

En sachant qu’on était un groupe de trois pour ce projet, on a mis en place un groupe Discord afin de faciliter la communication et le partage de ressources.

Sur ce dernier, une réunion était organisée chaque lundi pour partager le travail qu’on avait effectué lors de la semaine. Ce travail était ensuite évalué et potentiellement modifié avant son inclusion dans la branche principale du dépôt GitHub. Ces réunions nous permettaient également de répartir de nouvelles tâches chaque semaine aux membres de l’équipe et de débattre certains choix techniques.

Pour faciliter cette organisation, nous nous servions d’un tableau Kanban pour tracker les progrès sur chaque fonctionnalité et la personne qui travaillait dessus.

Afin de valider le travail de chacun, personne n'avait le droit de faire des push direct sur la branche principale du projet, et on devait faire un pull request pour chaque modification, de sorte que le groupe puisse vérifier les modifications et additions qui ont été faites et voir s'il y avait des problèmes.

Choix techniques:

Language: Python
Framework: Flet
Intégration continue: Doxygen, Github Actions
Architecture: Monolithique, Locale

Pour prendre en compte les besoins et les envies de chacun dans le groupe, on a décidé d'utiliser Python pour sa simplicité, accompagné du Framework Flet, car l'un des membres de notre groupe était familier avec la technologie sur laquelle Flet est construit (Flutter). De plus, nous avions la possibilité de développer le projet pour diverses plateformes avec ce framework (Web, Linux, Mac, Windows), ce qui était intéressant.

Le choix d'utiliser Python et Doxygen a également été influencé par les ressources à notre disposition sur le Moodle liées à l'intégration continue.

Pour les choix au niveau de l'architecture, l'idée était de garder le projet simple. Donc, on a choisi de faire une application locale.

Gestion des données

Le JSON qui sert à démarrer une partie ne contiendra que le nom des utilisateurs, le backlog et le mode de jeu sélectionné par l'utilisateur. Tandis que le JSON de sauvegarde (et de résultat) contiendra en plus la note d'évaluation de chaque tâche.

Lorsqu'un utilisateur charge un JSON de sauvegarde, la quantité d'évaluations des tâches dans la sauvegarde déterminera à quelle tâche la partie reprendra.

Mise en place de l'intégration continue:

Workflow:

Pour les tests unitaires (unit-test) :
- Checkout Code : Permet au workflow d'accéder au dépôt GitHub
- Set up Python : Installe une version de Python pour le job
- Install dependencies : Installe les librairies Python nécessaires pour exécuter le job
- Run tests and generate HTML report : Exécute le fichier Python pour les tests unitaires et génère un rapport HTML

Pour la documentation Doxygen (documentation):
- Checkout Repository : Permet au workflow d'accéder au dépôt GitHub
- Install Doxygen : Installe Doxygen pour le job
- Install Dot: Installe Dot pour le job
- Generate Documentation : Génère la documentation avec les configurations du fichier Doxyfile
- Deploy Documentation : Déploie la documentation Doxygen dans la branche gh-pages

Tests Unitaires:

Lorsque une modification est faite sur la branche principale (main), le job 'unit-test' est exécuté. Il met d'abord l'environnement en place pour pouvoir faire les tests unitaires, puis exécute la fonction test() du fichier test.py pour faire les tests. Si tous les assert() sont égaux à True, le test est passé, mais si un des assert est égal à False, le test échoue, et on aura une notification sur GitHub pour le savoir.

```python
def dict_avg_tests():
    dict_data = {'0':50, '1':0}
    assert (DictOperations.dict_avg(dict_data) == 25.0)

    dict_data = {'0':0, '1':0, '2':0, '3':200}
    assert (DictOperations.dict_avg(dict_data) == 50.0)

    dict_data = {'0':-50, '1':50}
    assert (DictOperations.dict_avg(dict_data) == 0.0)

    dict_data = {'0':5}
    assert (DictOperations.dict_avg(dict_data) == 5.0)
```

```python
def dict_unanimous_tests():
    dict_data = {'0':50, '1':50}
    assert (DictOperations.is_dict_unanimous(dict_data) == True)

    dict_data = {'0':10, '1':50, '2':50}
    assert (DictOperations.is_dict_unanimous(dict_data) == False)

    dict_data = {'0':-5}
    assert (DictOperations.is_dict_unanimous(dict_data) == True)
```

Tests unitaires pour le calcul de la moyenne, et test unitaires pour la validation de l'unanimité.

Génération de la Documentation:

Lorsque une modification est faite sur la branche principale (main), le job ‘documentation’ est exécuté. Il met d’abord l'environnement en place pour pouvoir générer la documentation, puis la génère en suivant les configurations du fichier Doxyfile, et il déploie la documentation sur la branche gh-pages. La branche gh-pages qui elle est configurée à publier un site en utilisant le contenu dans la branche avec GitHub Pages.

Manuel Utilisateur et Fonctionnalités:

Installation et Lancement:

En assumant que vous ayez déjà installé Python (<= 3.9) avec Pip.

```
pip install ‘flet[all]’
pip install pyinstaller
git clone https://github.com/ArthurCottreau/PlanningPoker.git
cd PlanningPoker/src
pyinstaller --onefile main.py
```

Le fichier exécutable se trouve dans le dossier 'dist'.

Modes de jeu:

Dans notre application, nous avons deux modes de jeu :
- Strict : L’évaluation de chaque tâche lors de la partie se joue sur l’unanimité.
- Moyenne : Le premier tour de l’évaluation d’une tâche se joue sur l’unanimité, mais le deuxième sur la moyenne.

Interface: