SCIENCES SORBONNE UNIVERSITÉ

Rapport de projet :

Simulation d'un système d'élections

BATACHE Toufic
FARAH Karl
HATOUM Nehmat
SIDDIQUE Zahra

Chargés de TD :
MAUDET Nicolas et LE GAY Sebastien

21 avril 2023

Sommaire

I - Vision globale du code

I - a. Architecture et structure

Notre code a été écrit et conçu de façon modulaire. Ce qui signifie que les différentes fonctionnalités sont structurées dans de multiples fichiers. Chaque module est autonome et indépendant, permettant une meilleure intégration et compréhension du code. De plus, la division de ce dernier nous permet d'effectuer des tests plus élaborés pour chaque fonction du module.

Notre logiciel est formé à partir des fichiers suivant :

1. **candidate.py et voter.py**

Ce sont les dataclass pour les objets "Candidat" et "Votant".
L'objet "Candidat" regroupe les paramètres spécifiques pour ce type d'élément : le label, les coordonnées et la couleur. Les fonctions de gestion de ces structures sont aussi définies dans le fichier.
Similairement pour l'objet "Votant", ce dernier possède par contre les paramètres label et coordonnées uniquement.

2. **file_manager.py**

Ce module regroupe les fonctionnalités de gestion des fichiers. À l'aide des fonctions au sein de ce module, on peut manipuler l'interface en utilisant des fichiers csv. On pourra par la suite, importer et exporter des données à partir de multiples fichiers.

3. **graph_manager.py**

Ce module manipule tout ce qui est en rapport avec le graphe de l'interface. Les fonctions de ce module permettent l'ajout de points sur le graphe, le

calcul de la distance entre différents points, et plusieurs autres fonctionnalités.

4. keyboard_manager.py

Ce module nous permet la manipulation de l'interface grâce aux touches du clavier. Les fonctions du module créent les liens entre les touches "Entrée", "Esc", "Tab" et "Shift+Tab".

5. data_manager.py

Ce module manipule la gestion des dataclass avec l'interface graphique. Les fonctions permettront la création de listes regroupant les votants et candidats. Elles nous autoriserons d'ajouter les candidats et votants sur le graph de l'interface et de réinisialiser les listes en cas de nécessité.

6. voting_manager.py

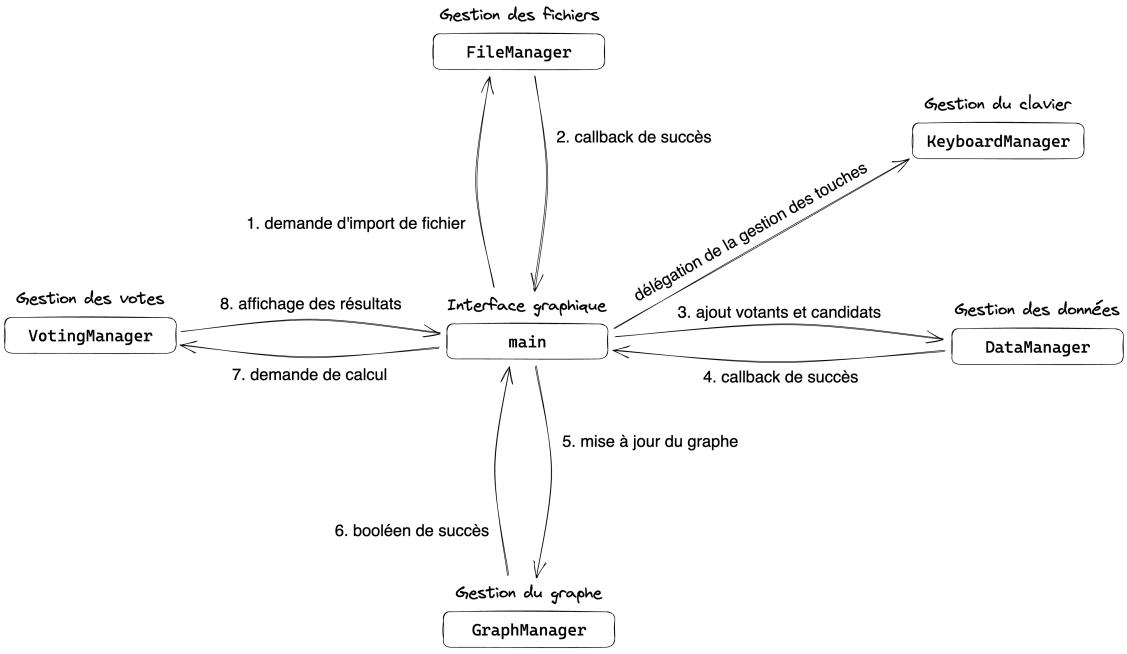
Ce module manipule les calculations relatives aux différentes méthodes de vote. Les fonctions prennent comme paramètre les profiles de votes et retournent le gagnant selon un certain système de vote afin de l'afficher.

7. voting_details_manager.py

Ce module permet l'affichage des details de vote et des informations. Les fonctions affiche des fenêtres avec les details relatifs demandés.

8. main.py

Ce module regroupe les différents modules définis précédemment pour manipuler l'interface et la générer.



I - b. Organisation et documentation

Au fur et à mesure du développement du projet, nous avons soigneusement commenté tout le code, en décrivant l'utilité de chaque fonction, détaillant les paramètres attendus et spécifiant la valeur de retour. Cela nous a permis par la suite de pouvoir générer la documentation de tout le projet très rapidement, en utilisant Sphinx.

Ce qui nous a aussi beaucoup aidé, c'est l'organisation sur le GitLab. Tout d'abord, nous avons travaillé en sprints, qui eux contiennent des tickets assignés à différents membres de l'équipe. Une demande de fusion a été créée pour chaque ticket, et le code écrit par l'un a été revu par l'autre pour s'assurer qu'il n'y a pas de bugs. Cela nous a permis de bien définir les grands chantiers sur lesquels on travaillait, tout en s'assurant que le code écrit est conforme aux contraintes mises en place (conventions de nommage, documentation, pas de bugs, etc).

II - Politique de tests

Des tests unitaires ont été réalisés sur chaque module et pour chaque fonction, en faisant bien attention à avoir traité les différents cas de bords. Les fichiers de test sont contenus dans le répertoire `tests_package`, et sont facilement identifiables en suivant la convention de nommage suivante : `test_<module>`.

III - Optimisation du code

Le programme a été optimisé pour pouvoir supporter un nombre élevé de candidats et de votants. Les améliorations apportées ont permis d'optimiser l'utilisation des ressources informatiques disponibles, réduisant ainsi le temps de calcul et améliorant les performances globales du système. En conséquence, les résultats de la simulation peuvent être obtenus plus rapidement, permettant une analyse plus rapide et efficace des données. Dans les tests effectués, nous étions capable d'ajouter environ 4500 objects sur le graphe et calculer les gagnants suivant les différentes méthodes de votes dans un temps de calcul inférieur à 5 secondes.