Architecture

Vision globale sur l'architecture d'un projet

Le projet est composé 4 packages principaux :

- `electoral_systems` : un package fournissant les modules et les sous-packages nécessaires pour une élection. Plus précisément, les utilitaires : la classe Singleton, les itérateurs pour générer des noms, des prénoms et des IDs (package `utls`), les extensions : les sondages, la démocratie liquide (package `extensions`) et les règles de vote (package `voting_rules`).
- `people` : un package fournissant les classes permettant de représenter les candidats et des électeurs dans une élection.
- `graphics` : un package fournissant les modules et les sous-packages de la partie graphique. Tous code de **PyQt** se trouve ici. Les sous-packages principalement des widgets utilisés sur les widgets principaux, i.e. les pages.
- `sqlite` : un package fournissant les modules permettant d'interagir (importer, exporter) avec la base des données à l'aide de SQLite.

Une telle séparation des modules par packages permet de regrouper le code par ses fonctions dans l'application et de retrouver facilement ce qu'il faut rechercher dans le code.

Justification des choix principaux de développement

Stockage des données de l'élection

Module `electoral_systems.election`.

Pour stocker des candidats et des électeurs on a décidé d'utiliser les listes simples car on n'avait pas besoin de rechercher un candidat ou sûrement un électeur particulier. De toute façon pour calculer des résultats d'une élection il fallait traiter chaque électeur un par un.

Dataclass

Package `people`.

Les classes de ce package ont été écrites à l'aide des `dataclass` pour simplifier l'écriture des méthodes spécifiques des classes (`__init__, __gt__, __lt__` etc) et mieux typer et structures des attributs.

Génération des données (itérateurs)

Module : `electoral_systems.utls.id_iterator`, `electoral_systems.utls.name_iterator` .

Les itérateurs ont été implémentés pour générer des IDs uniques des personnes (cf. `people.person`) et des noms et des prénoms des candidats (cf. `people.candidate`).

Les noms et des prénoms des candidats sont générés uniquement si l'utilisateur génère les données aléatoirement. Dans ce cas les noms, ainsi que les prénoms sont générés simplement de **A** à **Z**, après de **AB** à **AZ** etc. Cela permet d'identifier simplement les candidats sur la page des résultats.

Les simples générateurs de *Python* n'étaient pas suffisant car il fallait de faire redémarrer la génération des IDs et des noms et des prénoms si l'utilisateur revient sur la page d'accueil.

Une partie graphique

Pour implémenter la partie graphique de l'application on a utilisé le framework *C++* **Qt** porté en *Python* ***PyQt***.

Nous avons utilisé plusieurs outils de dessin fournis par **PyQt**, notamment :

- `pyqtgraph` pour la visualisation des graphiques de la distribution normale.
- `QtCharts` pour afficher les diagrammes des résultats des règles de vote à un et à plusieurs tours. Ces classes de PyQt offrent toutes les fonctionnalités nécessaires pour travailler avec différents types de diagrammes.
- `QGraphicsScene`, `QGraphicsItem`, etc. pour représenter graphiquement les résultats des règles de vote Condorcet-cohérentes. Ces classes sont particulièrement utiles pour afficher des objets en 2D.
- `QPainter` pour dessiner la carte politique. Cette classe offre les fonctionnalités nécessaires pour dessiner différentes formes (par exemple, des points, des lignes, etc.). En particulier, elle a permis de simplifier les calculs de conversion entre le système de coordonnées mathématiques et celui de l'ordinateur en utilisant la transformation `QTransform`.

Partage des données entre les différents widgets de Py*Qt*

Module `electoral_systems.utls.singleton`.

Plusieurs widgets de **Qt** a besoin soit d'accéder des données de l'élection, soit de les modifier. Pour gérer cela une classe principale qui représente une élection `Election` (module `electoral_systems.election`) hérite de la classe `Singleton` écrite manuellement. Cette approche garantit qu'une seule instance de la classe `Election` est créée et que toute référence ultérieure pointe vers cette même instance déjà existante.

Base des données

Package `sqlite`.

Notre application permet d'importer et d'exporter les données. Pour implémenter cette fonctionnalité on a décidé d'utiliser le *SQL*, et plus précisément *SQLite* à cause de la taille relativement petite du projet et de son intégration aisée avec Python.