Projet POO (Python)

Un projet de gestion de données musicales dans le cadre de la confirmation des acquis du D.U. II.

Binôme : BEKIER Maxence & LACAILLE Gustave.

Python, syntaxe simple et polyvalence.

- **Troisième** langage **le plus utilisé** par les développeurs, ce qui rend sa communauté très large.
- **Flexible** et **facile** à apprendre.
- Une **polyvalence** de programmation.
- Un accès **facile** à de la documentation **compréhensible**.
- Une **bibliothèque standard riche**, un accès aux nombreux modules qui permettent d'effectuer une variété de tâches, ce qui rend la programmation **plus simple**.
- **Populaire et très demandé**, il est aujourd'hui sollicité par de **nombreuses** entreprises et projets open-source.

Le diagramme de GANTT, la gestion du temps et la répartition des tâches, une étape importante du projet.

- Établissement d'un **diagramme de GANTT** pour une gestion efficace du temps.
- Répartition des tâches et autres au sein du binôme, prise en compte des **contraintes** personnelles/professionnelles de chacun et établissement d'un **plan global**.
- Mise en place d'outils numériques pour collaborer à **distance.** (GitHub, Discord, Google Drive)
- **Mises au point** avec Mr. BOURDON le **27 mars** et le **9 avril 2023**, pour un suivi particulier du projet.

Les principaux éléments de conception.

- **Architecture modulaire** : le code est **organisé** en **modules** distincts. (classes, fonctions)
- **Encapsulation** : caractéristique **essentielle** de la **POO** (programmation orientée objet), la plupart des méthodes sont **encapsulées** dans des classes. Le code est donc plus **protégé** et **lisible**.
- **Bibliothèques externes** : utilisation de bibliothèques externes pour **gérer les playlists** (ElementTree), ou **l'interface utilisateur**. (Kivy)
- **Interface utilisateur** : les fenêtres graphiques sont essentielles aux **utilisateurs non techniques** qui veulent interagir avec les logiciels.
- **Conformité aux normes** : le format **XSPF** est un format **standard** pour les fichiers de playlists, permettant une **utilisation universelle** des playlists créées et utilisées.

Défis, obstacles et solutions (?)

- Obstacle rencontré sur la fenêtre graphique, difficultés lors de la mise en place de la **modification** et de l'**affichage** des playlists.
- Mise en application de l'architecture modulaire et son **organisation** dans le programme.
- Prise de connaissance de la **norme XSPF** et de son architecture pour adapter le code à sa gestion.
- Organisation, la mise en place des tâches et leur accomplissement ont été rendu possible grâce à une **communication efficace** et une gestion des **outils numériques** optimale.

Conclusion : les points (non) traités et les extensions.

- Point console (‘cli.py’) traité à 100%. (recherche de fichiers musicaux, création de playlist à partir d’un dossier, ...)
- Point GUI (‘gui.py’) traité à 70%. (recherche des fichiers musicaux, affichage des métadonnées, création de playlist)
- Pas d’extensions.