Rapport POO – DU II 2023

Gustave Lacaille
Maxence Bekier

Introduction :

Dans le cadre de cette fin d'année en D.U II à CYTech, il nous a été proposé de réaliser un projet (POO) en Python à faire sur environ deux mois. Le projet qui nous a été attribué consistait en la mise en place d'un code permettant le tri et la récupération des métadonnées de fichiers MP3 et FLAC, leur restitution, ainsi qu'une fenêtre graphique permettant de manipuler des playlists au format XSPF.

L'ambition de ce projet est de tirer parti de nos compétences en matière de programmation orientée objet, notamment dans la gestion de l'héritage, des instances de classes et autres. Il fallait prendre en compte la consigne avec des règles bien précises pour chaque ligne de code à effectuer.

Dans ce rapport, nous nous proposons d'évoquer les grandes étapes de ce projet, de cerner les objectifs que nous nous sommes fixés, et d'explorer comment nous avons su gérer notre travail collaboratif en dépit des contraintes horaires professionnelles de Maxence. Les différentes difficultés rencontrées et les leçons tirées au cours de l'élaboration du programme final seront également discutées.
En somme, ce rapport se veut être une exploration approfondie du processus de réalisation de notre projet. Il s'attardera sur les détails techniques liés à la mise en œuvre du cahier des charges, la gestion des outils disponibles, la maîtrise du temps et l'application des compétences acquises en programmation.

Planification du Projet :

Le sujet du projet nous a été donné le 27 mars 2023. L'une des premières étapes consistait à établir un diagramme de GanTT afin de gérer efficacement notre temps. Nous l'avons effectué dans les délais avec une mise en commun des informations comprises et des buts individuels à atteindre. En sachant tout cela, il nous a été facile de le faire assez rapidement, la communication au sein du groupe étant fluide.

Ensuite, le premier point d'avancement du projet a été fait le vendredi 7 mars où un PNG du diagramme de GanTT était à rendre. Il devait montrer d'une façon globale comment nous allions gérer la progression du projet et ses imprévus dans le temps.

Nous avons finalement très peu utilisé le diagramme de GanTT, allant à un rythme efficace, le travail a pu avancer grâce à une communication et un échange d'information efficaces grâce aux outils mis en place. Cependant, certaines phases ont été difficiles à mettre en place en raison du chevauchement du projet WEB en parallèle et pour certains, des obligations professionnelles en plus.
Une fois tout cela fait, il ne nous restait plus qu'à commencer la phase de développement du projet en Python.

Description du programme du projet POO 2023

Le code produit à l'aide de l'énoncé du Projet POO 2023 a pour objectif de rechercher des fichiers MP3 ou FLAC en vérifiant le type MIME dans un dossier et ses sous dossiers, récupérer les métadonnées d'une ou plusieurs musiques, de créer une playlist au format XSPF, tout cela dans le terminal et de manipuler cette dernière dans une fenêtre graphique. Différentes bibliothèques ont été utilisées dans ce code, comme TinyTag ou Kivy, c'est l'un des avantages de Python, sa variété de bibliothèque. Ce point lui permet de pouvoir développer une grande diversité de programmes.
Tout ces points devront être utilisable en l'état en suivant les instructions fournies dans le readme.txt

Description des différentes étapes de la programmation objet Python

1 / Mise en place de l'environnement de travail collectif numérique

La mise en œuvre de notre projet a été orchestrée par nous deux de manière séparée. Nous avons utilisé différents environnements de développement pour mener à bien nos tâches. Gustave a choisi d'utiliser Visual Studio comme plateforme de développement principale. Après avoir créé et révisé ses codes sur Visual Studio, il les a ensuite testés dans son terminal Unix pour assurer leur bon fonctionnement.

De son côté, Maxence a préféré utiliser IntelliJ, un autre environnement de développement très populaire. IntelliJ est réputé pour sa flexibilité et son interface utilisateur intuitive, ce qui le rend idéal pour une variété de projets de développement. Il est également possible d'installer des plugins pour étendre ses possibilités (comme la compréhension par IntelliJ d'un autre langage), et il possède un terminal intégré.

Afin de faciliter la collaboration et la synchronisation de nos efforts, nous avons choisi d'utiliser GitHub, une plateforme de partage de code. Cette plateforme nous a permis de partager efficacement nos progrès dans le code, tout en garantissant la transparence et la clarté du travail accompli. C'était d'autant plus crucial que l'intégralité de notre travail a été réalisé à distance, ce qui souligne l'importance d'une plateforme de collaboration efficace.

En plus de GitHub, nous avons utilisé Discord pour la communication. Discord, une plateforme de communication en temps réel, a été utilisée pour les appels audio et les échanges textuels entre nous. Cela nous a permis de rester en contact constant, de résoudre rapidement les problèmes et de discuter de manière productive des avancées et des obstacles du projet. Ainsi, malgré la distance physique entre nous, nous avons réussi à travailler ensemble de manière efficace et productive grâce à l'utilisation intelligente de ces outils de collaboration et de communication.

2 / Mise en place des premières classes et méthodes

La phase initiale de notre projet a commencé avec la création du fichier 'cli.py'. Cette étape préliminaire a été d'une importance cruciale, car elle a jeté les bases de l'infrastructure de notre système.

La première tâche à laquelle nous nous sommes attaqués a été d'étudier et de comprendre la nature des métadonnées liées aux fichiers audio. Les métadonnées sont des ensembles d'informations attachées à chaque fichier audio, offrant des détails précis sur divers aspects du contenu audio. Ces informations comprennent le titre du morceau, l'artiste qui l'a interprété, l'album dans lequel il est inclus, l'année de publication, la durée totale du morceau, l'artiste de l'album, le genre musical, le numéro de piste, le nombre total de pistes dans l'album, et même le compositeur du morceau. Ces informations, tout en étant invisibles pour l'auditeur moyen, jouent un rôle clé dans la gestion, le tri et l'organisation de la musique dans une application audio.
Pour manipuler efficacement ces métadonnées, nous avons décidé de créer une classe spécifique, nommée “Metadata”. Cette classe a été conçue avec des attributs qui correspondent à chaque élément de métadonnées que nous avions identifié. Chaque instance de la classe Metadata représente ainsi un ensemble unique d'informations liées à un fichier audio particulier.
En outre, nous avons également ajouté une méthode externe pour extraire les métadonnées d'un fichier audio, et créer une nouvelle instance de la classe Metadata. Cette méthode utilise la bibliothèque TinyTag, un outil puissant capable de lire les informations de métadonnées à partir de divers types de fichiers audio. Grâce à cette méthode, nous pouvons facilement récupérer et utiliser les informations de métadonnées à partir de n'importe quel fichier audio dans notre système. Cette approche nous permet de gérer de manière flexible et efficace la musique dans notre application, en facilitant des opérations telles que le tri et l'organisation des pistes, la création de playlists…

3 / Fonction MAIN et affichage console

Une fonction main() a été mise en place dans le fichier ‘cli.py’ afin d’interpréter et d’analyser les entrées de l'utilisateur qui sont fournies par la ligne de commande dans le terminal. Cette fonction est le point d'entrée du programme et c'est là que les interactions avec l'utilisateur ont lieu.

L'un des principaux rôles de cette fonction main() est d'utiliser un module de Python qui analyse les arguments de la ligne de commande entrée. Ces arguments permettent à l'utilisateur de spécifier précisément ce qu'il veut que le programme fasse et sur quels fichiers ou répertoires il doit travailler.
Il est possible d'envoyer au programme plusieurs arguments. Le premier est -d ou --directory, qui peut être utilisé pour indiquer un répertoire spécifique que le programme doit explorer. Le programme parcourra récursivement ce répertoire et tous ses sous-répertoires, cherchant des fichiers musicaux sur lesquels travailler.
Le deuxième argument est -f ou --file, qui peut être utilisé pour spécifier un seul fichier à analyser. Contrairement à l'option de répertoire, cette option se concentre sur un seul fichier et permet à l'utilisateur d'obtenir des informations sur ce fichier spécifique.
Le troisième argument est -o ou --output, qui peut être utilisé pour spécifier un chemin de sortie pour une playlist XSPF qui sera développée dans la suite du programme. La playlist XSPF (XML Shareable Playlist Format) est un type de format de liste de lecture basé sur XML qui est largement supporté par de nombreux lecteurs de musique. Une fois que le programme a analysé les fichiers de musique et/ou exploré les répertoires, il peut utiliser cette option pour générer une playlist XSPF contenant tous les fichiers de musique trouvés.

Ces trois arguments de ligne de commande sont essentiels pour la flexibilité du programme, permettant à l'utilisateur d'adapter le comportement du programme à ses besoins spécifiques.

4 / Les Playlists

Cette phase de développement du projet s'est principalement concentrée sur la création et la gestion des "listes de lecture", également appelées "playlists". Une liste de lecture est une liste de chansons ou de morceaux de musique qui sont lus dans un certain ordre.

Une classe spécifique a été créée pour cela, appelée "Playlist". Cette classe est comme un ensemble d'instructions pour créer et gérer ces listes de lecture. Ces instructions comprennent diverses fonctions qui permettent de créer une nouvelle liste de lecture, d'ajouter des chansons à la liste, de lire la liste existante et de faire beaucoup d'autres choses.

Nous avons aussi mis en place une fonction qui permet de récupérer toutes les playlists existantes d'un certain dossier sur un ordinateur, une autre qui affiche les chansons de la playlist à l'écran pour qu'on puisse les voir, et une qui ajoute une nouvelle chanson à la playlist.

En résumé, cette phase de développement a permis de créer une série d'outils qui permettent de créer et de gérer des listes de lecture de musique, ce qui rend l'application plus utile et facile à utiliser actuellement et dans le futur avec l'interface graphique. Tout cela a été correctement intégré dans la fonction main().

5/ Fenêtre Graphique et gui.py

Après avoir finalisé les méthodes et classes du fichier 'cli.py', nous nous sommes orientés vers le développement de l'interface graphique utilisateur dans le fichier 'gui.py', une tâche qui n'a pas été sans difficultés. Notre premier défi a été d'apprendre à utiliser la bibliothèque Kivy, qui est essentielle pour la gestion de l'interface graphique. Cependant, sa mise en œuvre s'est avérée être un obstacle considérable.

Néanmoins, nous avons surmonté ces défis et avons réussi à mettre en place une fenêtre sur le côté gauche de l'interface utilisateur. Cette fenêtre permet de rechercher des fichiers FLAC ou MP3 dans les dossiers à partir de la racine. Lorsqu'un fichier musical est sélectionné, ses métadonnées sont affichées au centre de l'interface, y compris l'image de couverture. Cette dernière a requis une attention particulière dans le code pour s'afficher correctement, et le code du fichier 'cli.py' a été utilisé à certains endroits.

Par la suite, nous avons intégré la gestion des playlists. La barre de recherche des fichiers XSPF a été placée sur le côté droit de l'affichage des métadonnées, et les boutons de modification ont été placés en dessous. Nous avons pensé cette disposition pour améliorer l'expérience utilisateur, la rendant plus intuitive et facile à comprendre.

Enfin, l'utilisateur a la possibilité de sélectionner ses fichiers et de construire la playlist de son choix au format XSPF. Cette fonctionnalité offre une grande flexibilité à l'utilisateur, lui permettant de personnaliser ses playlists selon ses préférences musicales.

Conclusion :

Au terme de ce projet, nous avons réussi à réaliser un programme en Python qui permet le tri, la récupération des métadonnées de fichiers MP3 et FLAC, et la gestion des playlists au format XSPF. Grâce à un travail collaboratif efficace, malgré des contraintes de temps et de distance, ils ont pu relever les défis techniques que représentait le projet.

Tout d'abord, la planification, via l'établissement d'un diagramme de GanTT, a été une étape cruciale pour la gestion du temps et la répartition des tâches. Ensuite, la mise en place d'un environnement de travail numérique a facilité la collaboration à distance et la synchronisation des efforts.

Dans la partie technique du projet, la création des classes et méthodes pour gérer les métadonnées des fichiers audio et pour manipuler les playlists a été une étape importante. Par la suite, la mise en place de l'interface graphique a permis de rendre le programme plus intuitif et plus facile à utiliser pour l'utilisateur.

Malgré les difficultés rencontrées, notamment pour la mise en œuvre de la bibliothèque Kivy pour l'interface graphique, le projet a été mené à bien. Les compétences acquises en programmation orientée objet ont été mises à profit et le travail réalisé respecte le cahier des charges initial.

En conclusion, ce projet a été l'occasion pour nous de mettre en pratique nos connaissances en programmation Python et en programmation orientée objet. C'est une expérience enrichissante qui nous a permis de développer des compétences techniques, un esprit de collaboration et la capacité à gérer des projets d’envergure de groupe de travail.