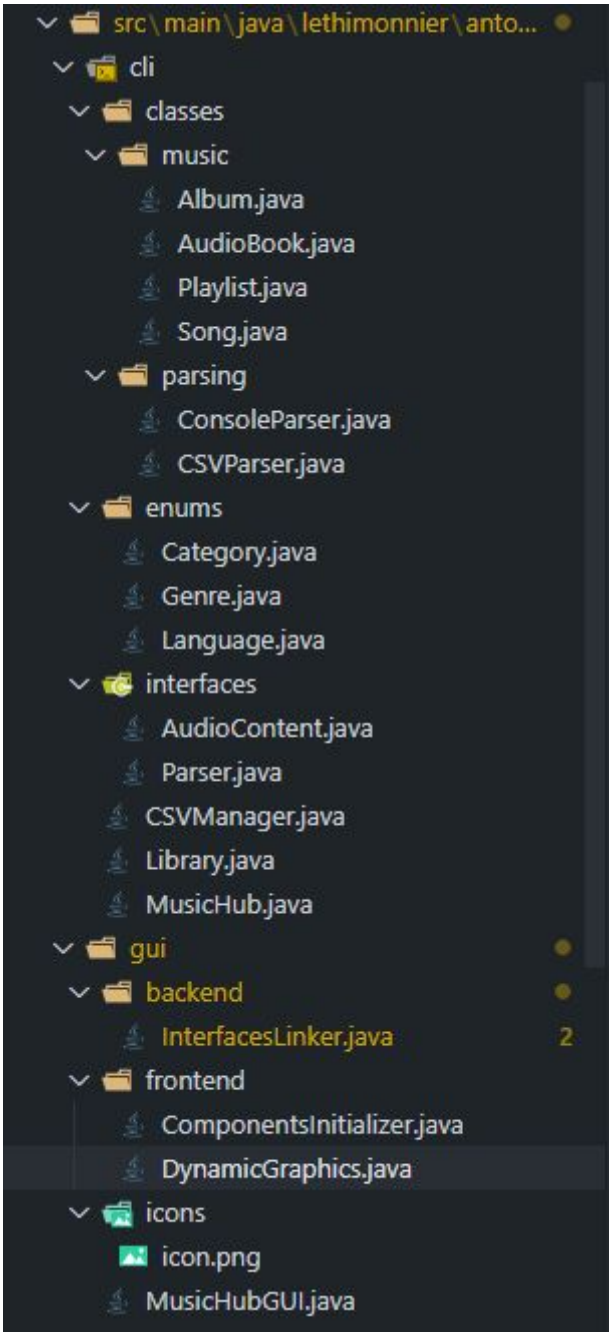
Rapport Projet POO MusicHub

Antoine Lethimonnier
Jérémy Rodrigues

Table des matières :

I/ Bilan du travail

1. Organisation du code



Le code se divise majoritairement en deux parties. En effet, nous avons pris la liberté de créer un version en interface graphique de ce projet, en plus de la version en ligne de commandes.
Les deux sont accessibles via la même base, avec l'un sous le dossier **cli** et l'autre sous le dossier **gui**.

Le gui fonctionne avec les méthodes du cli. Il est possible autant pour le gui que pour le cli d'utiliser plusieurs instances de chaque en même temps : leurs mécanismes sont identiques mais leurs données sont uniques.
La version en ligne de commandes se déroule ainsi:

- Un sous-dossier **classes** pour les classes "basiques". Il contient deux autres dossiers :
    - Un sous-dossier **music** pour les classes principales. Celles-ci sont les 4 objets principaux nécessaires : la Song, l'AudioBook, l'Album et la Playlist
    - Un sous-dossier **parsing**, permettant la gestion des entrées utilisateur. Elles permettent donc de gérer les entrées clavier, et de les transformer en objet du dossier **music**. Elles permettent aussi la conversion des entrées dans le fichier csv en objet **music** également.
- Un sous-dossier **enums**, contenant toutes les classes énumératrices. Ce sont en fait les objet Genre, mais aussi Langue et Catégorie, utilisés respectivement dans les objets Song et AudioBook.
- Un sous-dossier **interfaces** qui contient les interfaces Java et les classes abstraites, utilisés notamment en superclasse de Song et AudioBook (AudioContent), mais aussi en superclasse des classes parseurs dans le sous-dosser **parsing**.
- Enfin, les classes majeurs sont situées à la racine du cli, avec:
    - **CSVManager**, qui gère l'import et l'export au niveau du fichier csv
    - **Library**, qui stocke et gère tout le contenu de la bibliothèque musicale
    - **MusicHub**, la classe dans laquelle se trouve le main, qui gère les entrées-sorties utilisateur et les redirige dans les classes de traitement correspondantes

Aussi, la version en interface graphique est composée de :

- Un sous-dossier **backend**, qui contient la classe InterfacesLinker, permettant d'utiliser les méthodes existantes du cli pour faire fonctionner le gui
- Un sous-dossier **frontend**, qui contient les deux classes de gestion et de génération de composants de l'interface graphique
- Un sous-dossier **icons**, qui continent simplement une image libre de droits qui est utilisée en tant qu'icône de l'application
- Enfin, la classe majeure **MusicHubGUI** est située à la racine du gui, qui permet d'initialiser la fenêtre et d'appeler les autres classes délégatrices.

2. Fonctionnement

a. Command Line Interface (CLI)

Voici comment fonctionne l'application MusicHub via terminal :
L'utilisateur est invité, et ce dès le lancement de l'application, à choisir un fichier csv à importer dans l'application.
Si l'utilisateur a malencontreusement passé l'import proposé lors du lancement de l'application, il peut toujours utiliser l'option "i" afin de pouvoir importer un fichier.
L'utilisateur a la possibilité d'utiliser diverses commandes afin de réaliser des actions précises qui sont :

- c : permet à l'utilisateur d'ajouter une nouvelle chanson dans l'application.

- a : permet à l'utilisateur de créer un nouvel album.
- + : permet à l'utilisateur d'ajouter des chansons présentes dans l'application dans un album.
- l : permet à l'utilisateur de créer un nouvel AudioBook.
- p : permet à l'utilisateur de créer une nouvelle playlist.
- - : permet à l'utilisateur de supprimer une playlist existante.
- i : permet à l'utilisateur d'importer un nouveau fichier CSV.
- s : permet à l'utilisateur de sauvegarder toutes la librairie de l'application dans le fichier CSV souhaité.
- t : permet d'afficher les albums triés par genre, les albums triés par date de sortie, les audiobooks triés par auteur et les playlist triées par ordre alphabétique.
- o : permet à l'utilisateur d'afficher tous les contenus de la librairie.
- x : permet à l'utilisateur de supprimer tous les éléments de la librairie
- h : permet d'afficher la liste de toutes les commandes disponibles
- q : permet à l'utilisateur de quitter proprement l'application

b. Bonus - Graphical User Interface (GUI)

Voici comment fonctionne l'application MusicHub via le GUI:
Une fenêtre apparaît dans laquelle l'utilisateur peut effectuer diverses actions.

Dans la barre des menus, l'utilisateur peut cliquer sur l'onglet "File" afin qu'il lui soit proposé d'importer, d'exporter des fichiers CSV, de rafraîchir les tableaux ou bien encore de quitter l'application.
Il a aussi à disposition un onglet "help" dans lequel il peut obtenir des informations sur l'application telles que les identités des créateurs ainsi que le copyright.

L'utilisateur a à sa disposition deux menu, l'un lui permettant de jongler entre les chansons, albums, playlists et audiobooks, l'autre lui permettant de choisir ce qu'il souhaite faire (ajouter, retirer ou seulement voir les éléments) dans l'un des quatres types de contenus qu'il a choisi auparavant dans le premier menu.
Ainsi, si l'utilisateur va dans l'onglet playlist puis qu'il clique sur l'onglet "view" il pourra voir toutes les playlists dont il dispose dans sa librairie. Cela fonctionne de la même façon avec les onglets "add" et "remove".

II/ Contribution du binôme

Antoine

Je me suis occupé de la création initiale des classes de bases. Ces classes sont celles décrites ci-dessus, dans le sous-dossier **music**. Ce sont les classes racine du projet.

J’ai également été en charge de la structure du code, mais aussi de toutes les classes ne touchant pas le CSV ni le frontend GUI, c’est-à-dire tout le dossier **enums**, **interfaces**, et les classes principales (MusicHub, MusicHubGUI, Library).

Pour la partie GUI, je me suis chargé de la classe principale et de la partie de liaison **backend**, en aidant mon coéquipier un peu sur la classe DynamicGraphics.

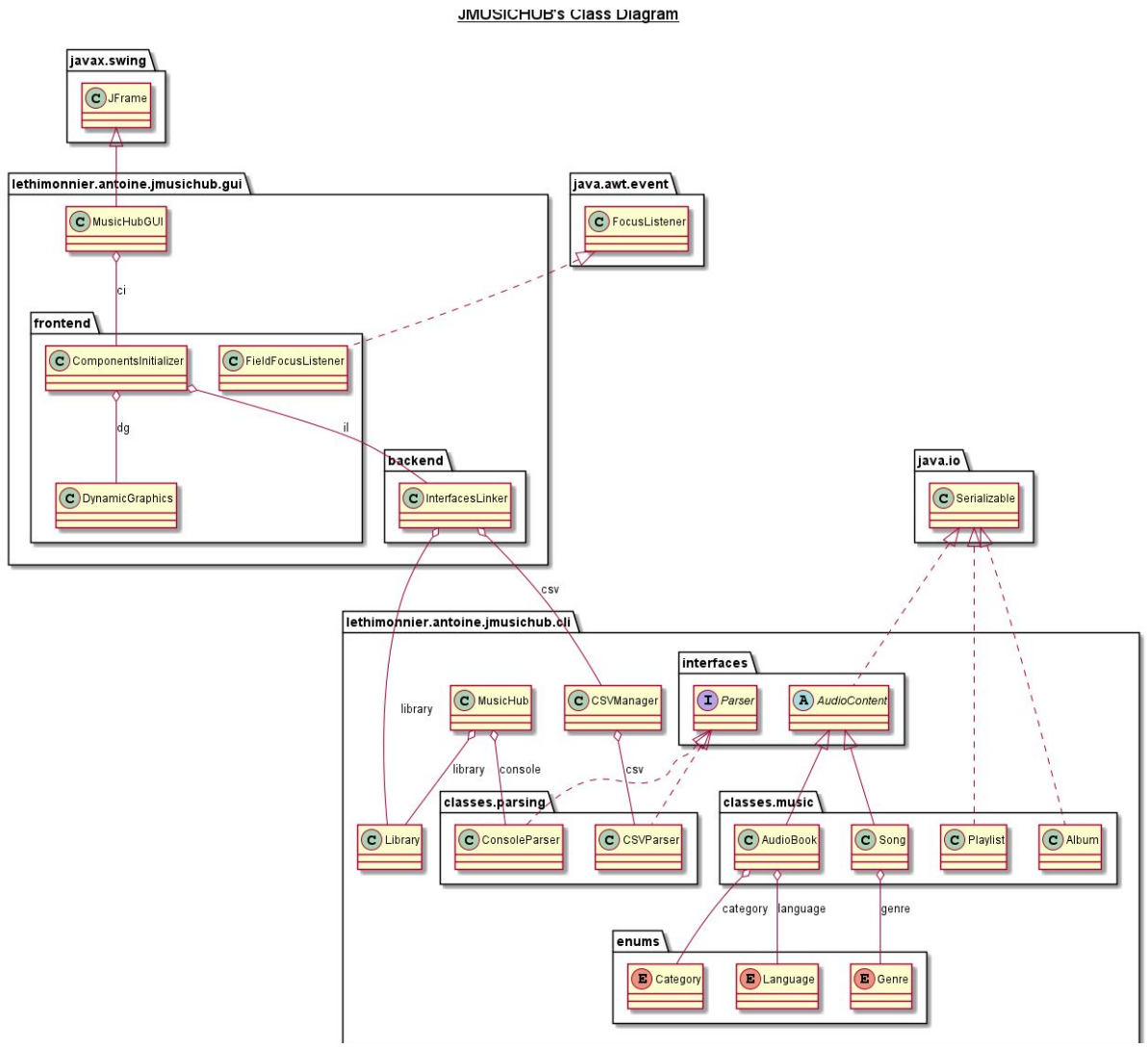
Jérémy

Je me suis occupé majoritairement à tous ce qui touche la partie CSV et la partie Interface du GUI.
Pour la partie CSV, je me suis donc chargé de l’import/export, des différents parseurs que nous utilisons pour l’affichage dans la console, l’écriture et la lecture des fichiers CSV.

Pour la partie GUI, je me suis chargé de toute la partie Interface, c'est-à-dire la gestion des divers Component et Containers qui permettent un affichage en temps réel interactif avec l’utilisateur de l’application MusicHub.

III/ Diagramme UML

Voici le diagramme UML du projet (L'UML est aussi disponible au format .png dans le .zip du projet) :



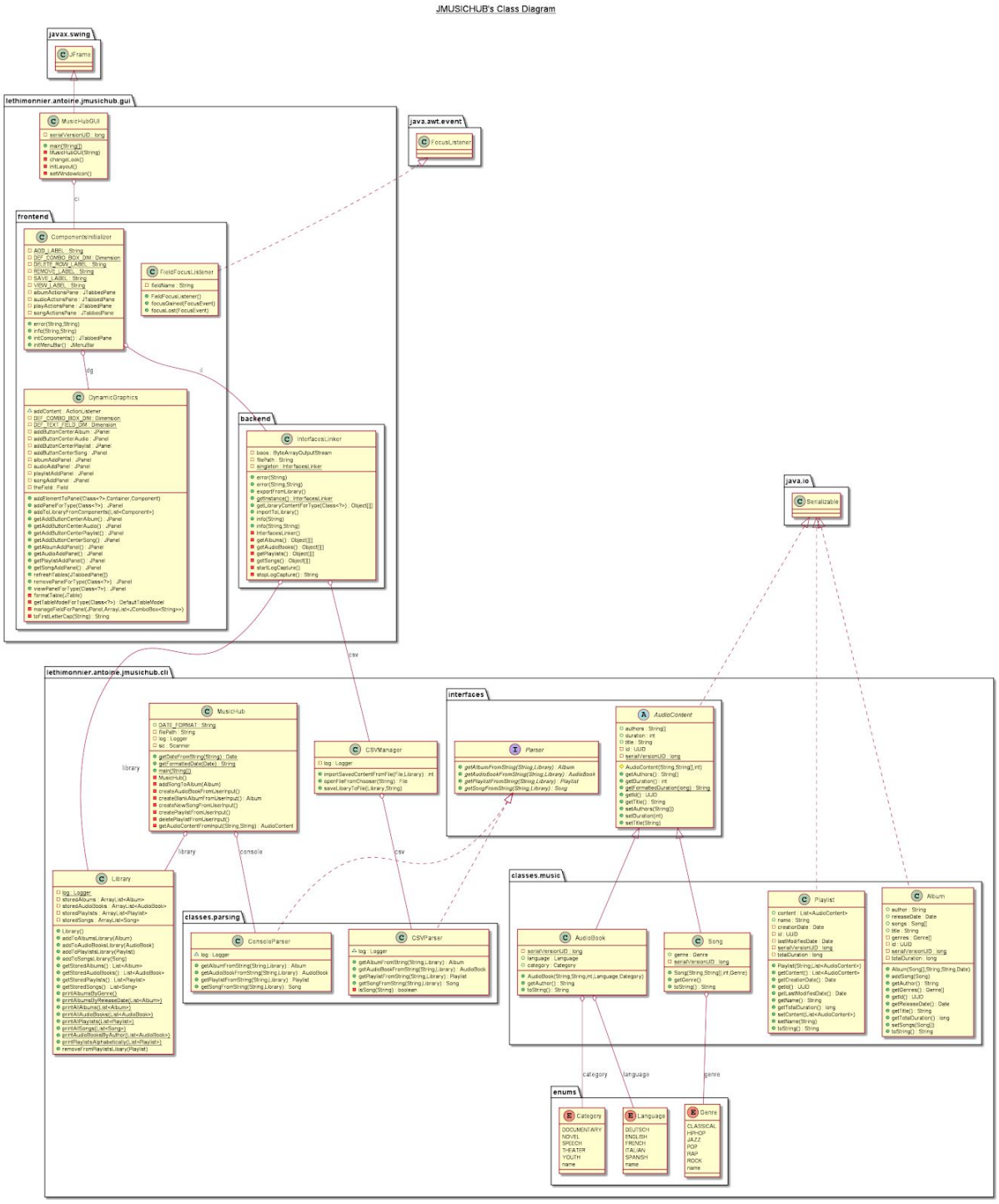


Diagramme complet avec méthodes et variables

IV/ Difficultés et solutions

1. Problème avec les Scanners

Problème 1 : Les Scanner rentraient en conflit les uns avec les autres, entraînant des exceptions liées au management de stdin.
Solution 1 : Tous les scanners ont été enlevés, ne laissant plus qu'un scanner global pour toutes les utilisations.

Problème 2 : Le scan d'entier (int) empêchait le scanner suivant de s'afficher, lié au fonctionnement de scanner.nextInt().
Solution 2 : Un scanner.nextLine() non récupéré a été ajouté en dessous de chaque nextInt().

2. Problèmes d'import et d'export

Problème : La bibliothèque externe openCSV n'était pas tout le temps compatible avec le séparateur (;) que nous avons utilisé dans notre fichier.
Solution : Pour l'import, nous avons remplacé, pour chaque ligne, les virgules par des point-virgules; pour l'export, nous avons spécifié le séparateur désiré, afin de pouvoir garder une cohérence entre les fichiers importés et exportés, et ainsi pouvoir réimporter des fichiers exportés.

3. Réorganiser les classes

Problème : Les classes étaient trop grandes et brouillon, entraînant des difficultés de compréhension du code.
Solution : Les classes ont été réorganisées en regroupant les méthodes similaires dans des classes dédiées. Cela a entraîné quelques ruptures du mécanisme, qui ont dû être résolues.

4. Problèmes d'affichage avec les Layouts du GUI

Problème:
Les composants ne s'affichaient pas correctement avec les Layouts que nous utilisions.

Solution :
Nous avons dû chercher diverses solutions afin de trouver une façon optimale de tous afficher correctement en étudiant tous les layouts disponibles et en les comparant afin de nous rendre compte duquel était le mieux pour le contenu que nous souhaitions afficher.

5. Problème avec l'interface de Vue/Remove du GUI

Problème:

Afficher les éléments d'Album dans le JTable a été un vrai calvaire car nous n'arrivions pas bien gérer toutes les boucles for que nous avions afin de bien afficher tout le contenu des albums sous la forme :

```
        Album                   Song
Titre Auteur Genre | Titre Auteur Genre Durée (song 1 de Album 1)
                   | Titre Auteur Genre Durée (song 2 de Album 1)
                   | Titre Auteur Genre Durée (song X de Album 1)
Titre Auteur Genre | Titre Auteur Genre Durée (song 1 de Album 2)
```

Solution :
Nous avons utilisé un compteur global pour gérer les lignes au lieu d'utiliser les variables présentes dans les for car celles-ci ne s'incrémentaient pas au moment voulu. Le problème a été identique pour les Playlists, et une solution similaire a été appliquée.