Rapport final - Projet long
Lecteur de musique

Jed Moutahir, Nathan Foucher, Héloïse Lafargue,
Yvan Kiegain, Aicha Bennaghmouch, Jordan Ramassamy, Martin Guidez

Technologie objet
Département Sciences du Numérique - Première année
2021-2022

Table des matières

Table des figures

Première partie
Rappel du sujet

1 Contexte

L'objectif du projet est de faire une application de lecture de musique.
Dans notre vie quotidienne, la musique nous accompagne constamment (trajets, détente, travail, soirées...). Néanmoins nous avons pris l'habitude d'écouter de la musique compressée et de moins bonne qualité qu'a l'époque. En effet les platformes de streaming (prédominantes sur ce marché aujourd'hui) sont bien obligées de compresser leurs fichiers audio pour ne pas surcharger les serveurs et la connexion interent. Or certains utilisateurs souhaitent retrouver la qualités CD pour écouter de la musique. C'est pourquoi nous avons choisi de créer cette application capable de lire les fichiers en haute qualité stockés localement tout en gardant l'avantage des platformes de streaming capable de gérer et de créer automatiquement une bibliothèque propre et à jour.

2 Pitch Elevator

Pour (public concerné par le produit) :	Les personnes disposant d'un ordinateur
Qui souhaitent (besoin des cibles) :	écouter de la musique en local
Notre produit est :	un lecteur de musique sur ordinateur
Qui (le bénéfice majeur, l'utilité de la solution) :	permet de lire des fichiers audios de qualité CD
A la différence (pratique actuelle, concurrence) :	de la radio ou des applications utilisant internet (qualité dégradée)
Permet (éléments différentiables majeurs) :	d'organiser les musiques en playlist, de gérer automatiquement la bibliothèque (ajout automatique des informations sur les fichiers audios) de les écouter sans connexion internet.

FIGURE 1 – Pitch Elevator

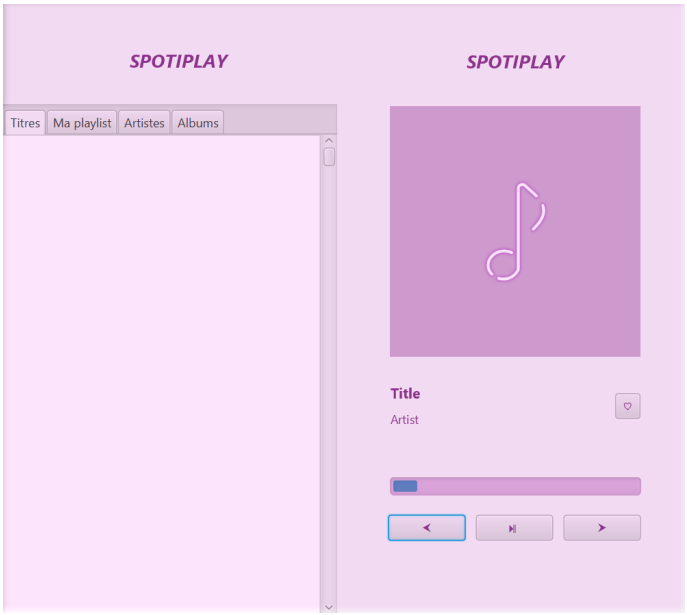

FIGURE 2 – Lecteur de musique

Deuxième partie

Principales fonctionnalités

3 Backlog produit

Les principales fonctionnalités de l'application sont identifiées dans la figure 3.
Nous avons représenté le thème en rose, les features en vert et les user-stories en bleu.

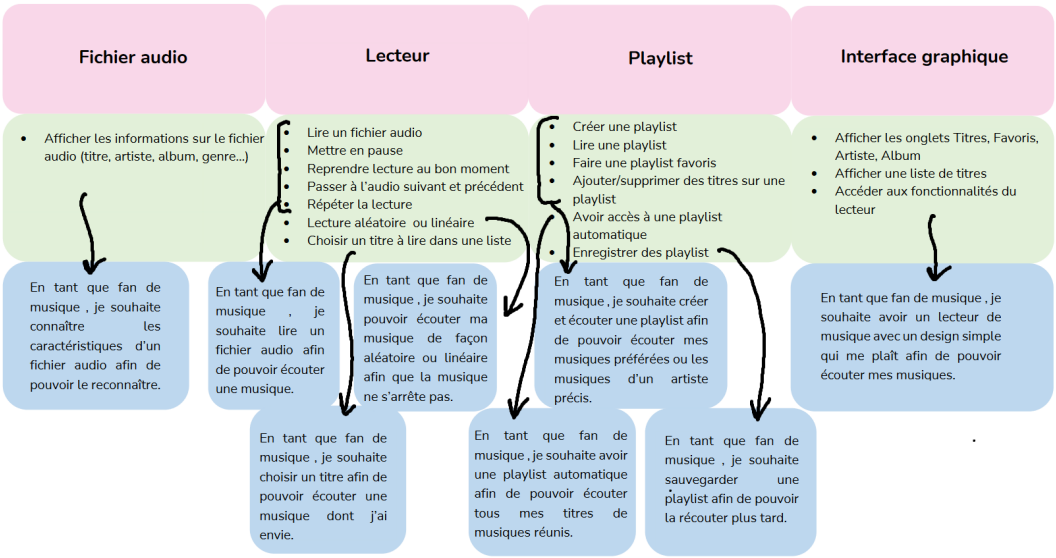

FIGURE 3 – Backlog produit du lecteur de musique

4 Etat de réalisation

4.1 Fichier audio

— Afficher les informations sur le fichier audio (titre, artiste, album, genre...) : commencé à l'itération 1 et terminé à l'itération 3 (avec l'API)

4.2 Lecteur

— Lire un fichier audio : réalisé à l'itération 1
— Mettre en pause : réalisé à l'itération 1
— Reprendre lecture au bon moment : réalisé à l'itération 1
— Passer à l'audio suivant et précédent : réalisé à l'itération 1
— Répéter la lecture : commencé
— Lecture aléatoire ou linéaire : réalisé à l'itération 1
— Choisir un titre à lire dans une liste : réalisé à l'itération 2
— Modification du volume depuis l'application : non réalisé
— Écouter de la musique en arrière-plan avec barre de notification persistante pour faire des actions simples : non réalisé

4.3 Playlist

— Créer une playlist : commencé à l'itération 2 et terminé à l'itération 3

— Lire une playlist : commencé à l'itération 2 et terminé à l'itération 3
— Faire une playlist favoris : réalisé à l'itération 3
— Ajouter/supprimer des titres sur une playlist : commencé à l'itération 2 et terminé à l'itération 3
— Avoir accès à une playlist automatique : réalisé à l'itération 2
— Enregistrer des playlist : réalisé à l'itération 3

Troisième partie

Découpage de l'application

Le découpage de l'application en sous paquetages permet de mieux organiser le projet et d'éviter différents conflits entre les modifications des étudiants. Il à été décidé de créer les paquetages :

- Interface : gérant l'interface graphique de l'application
- API : gérant les API de l'application
- Lecteur : gérant le back-end de l'application pour lire et réaliser la gestion des fichiers audios

Quatrième partie

Diagramme de classes

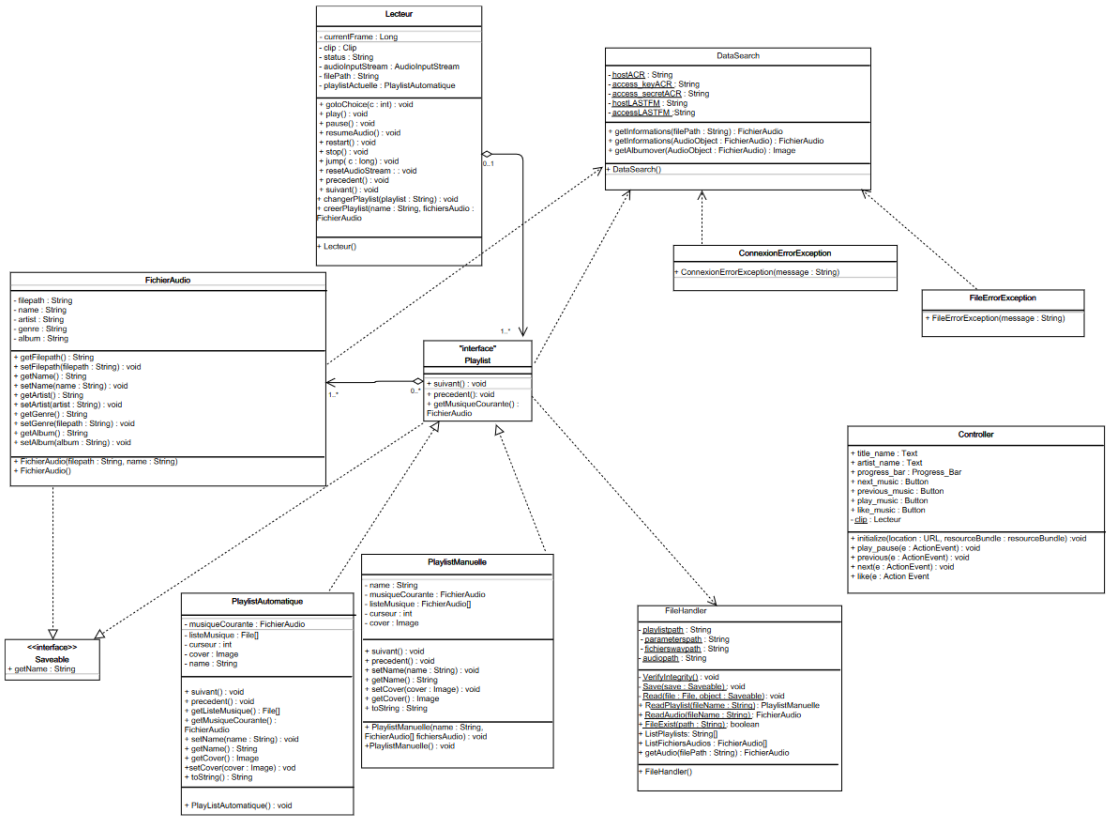

FIGURE 4 – Diagramme de classes

Le diagramme de classes qui donne l'architecture de notre application lecteur de musique est donné dans la figure 4.

Cinquième partie

Principaux choix & problèmes

5 Choix de conception significatifs

5.1 Interface Playlist

Il à été décidé dès le début de créer l'interface playlist qui sera l'une des plus utilisée. Cette dernière nous premet de créer des réalisations qui seront utilisées pour gérer le classement des musiques par playlist (comme l'on s'y attend) mais aussi par album, auteur, genre. En effet ces derniers élèments fonctionnent de la même manière qu'une playlist (liste de musique avec un nom et une image), ainsi par soucis de simplicité tout ces élèments sont des réalisations de playlist.

5.2 Classe FileHandler

Comme notre application à pour but d'être utilisée quotidiennement par l'utilisateur, il semblait évident de devoir sauvegarder les paramètres et réglages utilisateurs pour les retrouver à chaques ouvertures. Il faut notament sauvegarder les informations sur les fichiers audio et les playlists. Par mesure de simplicité nous avons décidés de créer une classe à part entière permettant de gèrer les sauvegarde de chaque réalisation de Saveable (interface des élèments enregistrables). Au début le choix avait été fait d'utiliser des fichiers JSON pour sauvegarder les informations. Mais cela était fastidieu et peut interressant pour sauvegarder des grosses quantités de données. Ainsi le choix à été fait d'utiliser objectOStream pour sauvegarder directements les attribus des objects "Saveable".

5.3 Classe Controller

La classe controller est la classe qui fera le liens entre l'interface graphique, les élèments et les méthodes. Cette classe est en quelque sorte le chef d'orchestre de l'application.

5.4 Classe DataSearch

L'un des points clé de notre projet etait que l'application de musique serait capable de chercher automatiquement les informations sur une musique (artiste, titre, album, jacquette de l'album). Pour cela on utilisera des bases de données sur internet en utilisant des API. La classe DataSearch s'occupe de l'utilisation des API en définissant des méthodes adpatées à nos utilisations. Les fichiers JSON en sortie de requêtes étant fastidieux à gérer, le choix d'utiliser la bibliothèque importée java-json.

5.5 Utilisation de java-fx

Pour l'interface graphique nous avons utilisé la bibliothèque et le sdk de java-fx permettant de réaliser des interfaces simplement avec un système de drag and drop.

6 Principaux problèmes rencontrées

6.1 java-fx

Java fx à l'avantage d'être très simple pour créer l'interface graphique en elle même. Néanmoins ce dernier demande l'utilisation de plusieurs bibliothèques et jdk. Cela à pris beaucoup de temps à une partie l'équipe gérant l'interface graphique pour prendre completement en main ces outils retardant notre avancé dans le projet.

6.2 Sauvegarde des élèments

Nous avons eu des soucis pour conserver les objects à l'aide de objectOStream. En effet certains attribus étaient perdu à la réimportation. Ce problème à su être résolu en réadapatant le code.

6.3 Travail en collaboration

Nous avons fait le choix de travailler en équipe grâce à un git. Chacun pouvait ainsi avoir sa branche et l'idée était de merge les grosse modifications sur la branche principale. Néanmoins comme nous travaillons sur divers systèmes d'exploiatations différents et sur des versions différentes d'éclipse, de nombreux compflits ont eux lieux ralentissant l'avancée sur le projet.

Sixième partie

Organisation de l'équipe & méthodes agiles

7 L'organisation de l'équipe

7.1 Jed Moutahir

— Mise en place d'un lecteur pouvant lire un seul fichier audio à la fois (simpleAudioPlayer.java)
— fonctionnalités implémentées : lecture, pause/reprendre, recommencer, choisir un instant du morceau où reprendre, arrêter.
— Travail sur le coeur du lecteur : création de playlists, enregistrement et chargement de ces dernières).
— Rendus : PlaylistManuelle.java, FichierAudio.java, FileHandler.java, MainIteration2.java. Ainsi que des fichiers test.
— Travail sur le coeur du lecteur : création d'un lecteur possédant toutes les méthodes nécessaires à une implémentation simple de l'interface graphique. Le lecteur est donc capable de manipuler des playlists, des fichiers audio, d'enregistrer les playlists créées et les métadonnées des fichiers audio pour les recharger lors d'une prochaine utilisation.
— Rendus : Playlist.java, PlaylistManuelle.java, PlaylistAutomatique.java, FichierAudio.java, FileHandler.java, Lecteur.java. Ainsi que des fichiers test.

7.2 Nathan Foucher

— organisation d'un vote pour le choix du sujet
— organisation de réunions et repartition des taches essentielles
— travail sur le FileHandler
— gestion des API
— gestion des requêtes web
— améliorations de classes déjà crées pour rajouter des éléments manquants
— remise en commun du travail des autres étudiants
— liens entre le back-end et front-end
— restructuration totale et néttoyage de l'application
— rédaction du rapport final

7.3 Héloïse Lafargue

— réalisation de l'interface graphique : design et réalisation de la page principale sur Scene Builder en accord avec la charte graphique. On retrouve les fonctionalités pour lire un titre
— découverte de JavaFX
— lien entre l'interface et les classes : réalisation complète du controller et main associés à ma page, en JavaFX
— rédaction de plusieurs rapports dont le rapport final
— organisation de réunions pour suivre les sprints

7.4 Yvan Kiegain

— réalisation de la classe playlist automatique
— réalisation de la classe de test playlist automatique
— réalisation de la classe Controller et ajustement du code général pour l'interface graphique et la gestion des événements afin de la rendre fonctionnelle

7.5 Aicha Bennaghmouch

— répertorier les décisions prises lors des TD de méthodes agiles (fonctionnalités par priorités)
— premier diagramme de classes
— rapport de la 1ère itération
— réalisation d'une partie de l'interface graphique avec Héloïse et Yvan (page avec les différents titres, playlist, albums) avec une partie du contrôleur associé. Il a fallu se renseigner sur JavaFX.

7.6 Jordan Ramassamy

— réalisation des différents diagrammes de classe.
— écriture de classes de test pour les différentes classes sous Junit.
— écriture du début de certaines classes comme FichierAudio ou bien Playlist.

7.7 Martin Guidez

— simplifications de méthodes pour optimiser le programme
— définition de la charte graphique de l'interface graphique avec la maquette
— essaie d'interface sur Swing

8 Mise en œuvre des méthodes agiles

8.1 Pitch elevator et backlog produit

Après avoir choisi lors d'un vote le sujet final parmis les 7 proposés, nous avons établi un pitch elevator pour s'assurer du besoin et de la cible. Puis, lors des séances de méthodes agiles nous avons pu établir le backlog produit avec les thèmes, features et user-stories. Cela nous a permis de définir les attentes de notre projet et de prioriser les fonctionnalités.

8.2 Sprint

Nous avons répartis le travail à faire lors des 3 sprints (itérations) dès le début du projet pour avoir une ligne de conduite à tenir.
Pour chaque itération nous avons défini un sprint backlog (liste des tâches priorisées).

8.3 Sprint review et rendus

A la fin de chaque itération nous prenions le temps de faire un sprint review pour faire la démonstration de l'avancée du projet à l'ensemble de l'équipe. De plus, chacun rédigeait un rapport personnel sur son travail et nous faisions aussi un manuel utilisateur et un rapport global.

8.4 Scrum et Trello

Nous avons utilisé l'application web Trello pour organiser nos tâches en srcum-ban (figure 5). Cela nous a permis de prévenir les autres membres de l'équipe sur quelles tâches l'on travaillé durant les sprints. L'avantage est aussi d'avoir une vue globale sur l'avancement et ce qu'il reste à faire.

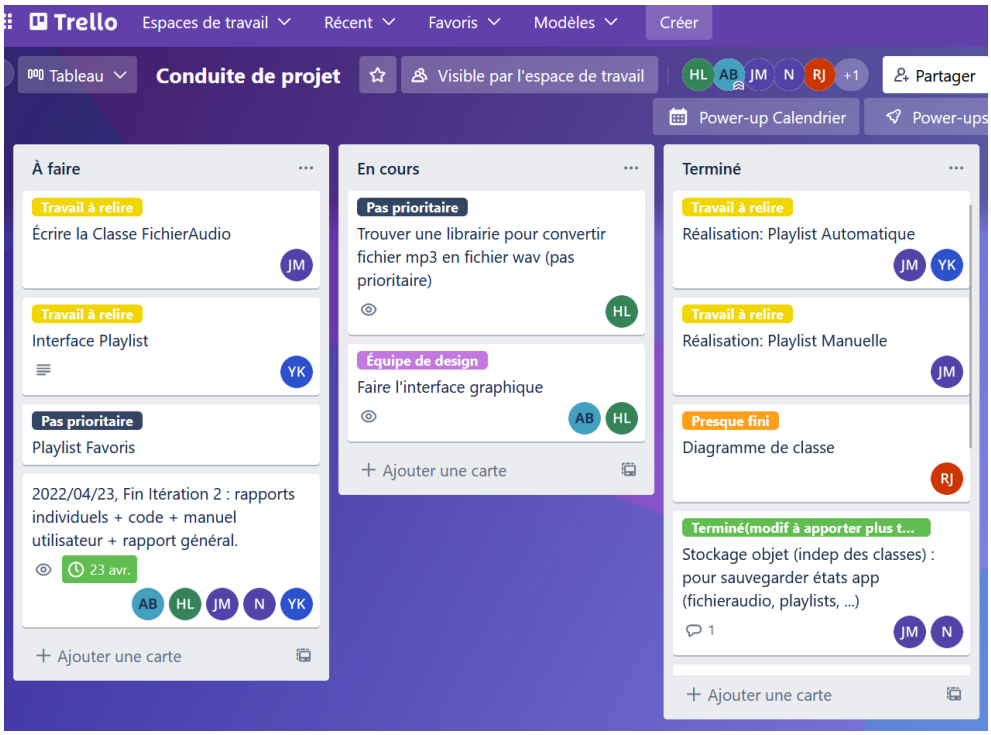

FIGURE 5 – Srum-ban, backlog d'itération

8.5 Mêlées quotidiennes

Lors de la dernière semaine, avant les rendus finaux, nous avons organisé des daily meeting (mêlée quotidienne) pour que chacun présente ce qu'il avait terminé depuis le dernier daily meeting, quels problèmes il avait rencontré et quels étaient ses objectifs pour le suivant.
Cela nous a permis de finir à temps les fonctionnalités que nous avions désignées comme prioritaires en aidant les membres de l'équipe sur les problèmes qu'ils rencontraient.

8.6 Notre bilan

Ce projet en groupe de 7 personnes nous a permis d'appréhender le travail en équipe et l'importance du srcum master ainsi que des méthodes de travail comme le sprint planning, les mêlées quotidiennes ou encore l'importance dees user-stories et des les backlog review.
Nous avons pu développer nos connaissances en Java et c'était l'occasion pour certains membres de l'équipe réaliser une première application.