Rapport du projet GLPOO

I-/ L’organisation du projet

Liste des étudiants :

-Anas Obadi : Création des 2 Pattern design / diagramme des cas d’utilisations / diagramme UML des classes

-Adrien Engel : code des class / GitHub

-Erwan Lepelletier : Script Maven / Test unitaires

-Lorenzo Lamberti : Trello / Rapport / Diagramme de séquence / Diagramme de déploiement / diagramme de composants

II-/ La spécification

1. Exigences

Exigences Fonctionnelles :

* Le système doit être capable de jouer de la musique
* Le système doit être capable de lire un livre audio
* L’utilisateur doit pouvoir rechercher une musique
* L’utilisateur doit pouvoir rechercher un livre audio
* L’utilisateur doit pouvoir créer une playlist
* L’utilisateur doit pouvoir ajouter et supprimer des éléments de sa playlist
* L’utilisateur doit pouvoir augmenter ou baisser le volume
* Le système supportera le français, l’anglais, l’Italien, l’espagnol et l’allemand
* Le système doit pouvoir accéder à une musique depuis un fichier MP3 ou Wave
* Le système doit pouvoir assurer les fonctionnalité « Play » et « Stop »
* Pouvoir se connecter à l’application en tant qu’Admin ou Utilisateur

Exigences Non-Fonctionnelles :

* Les données seront stockées dans des fichiers XML
* L’application est programme de bureau
* L’application est développée en Java
* La musique doit se lancer en moins de 2 secondes après sélection

1. Diagramme UML

Diagramme des cas d’utilisations :

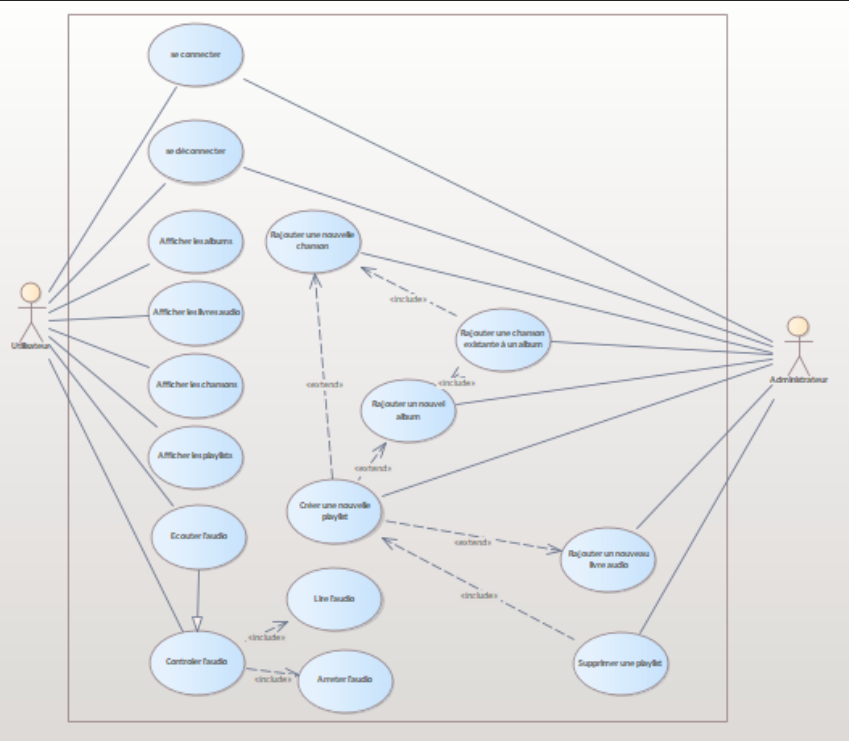
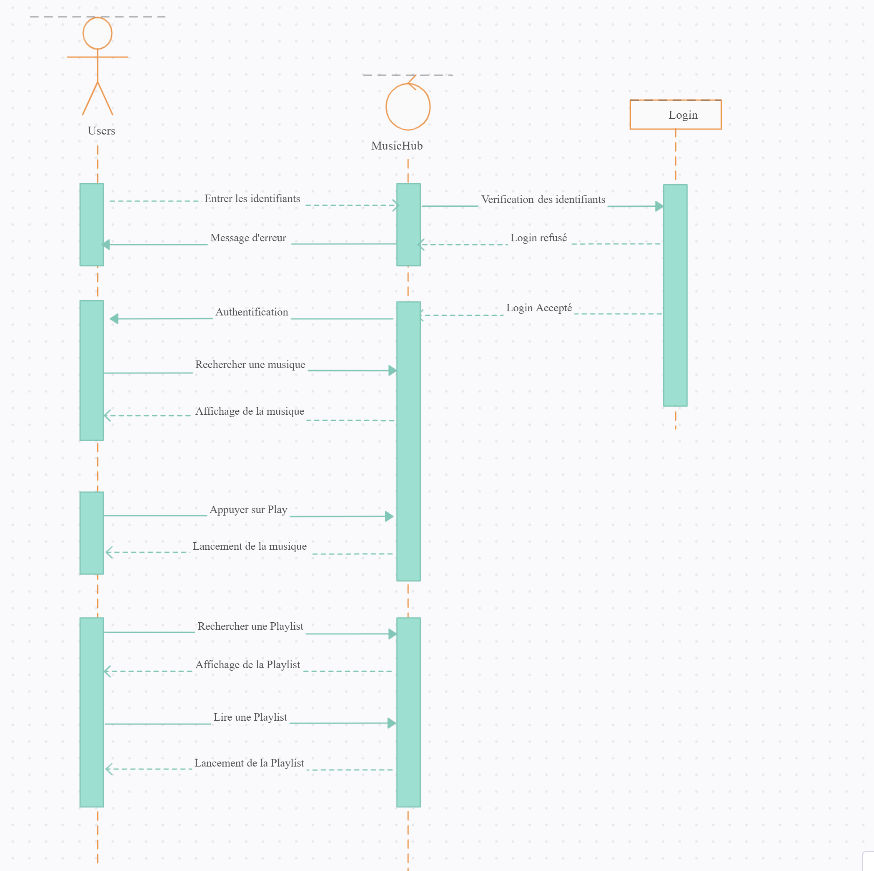


Diagramme de séquence :



1. Lien du Trello

<https://trello.com/b/kGI4ZopQ/trello-glpoo>

III-/ La conception

1. Diagrammes UML

Diagramme de package :

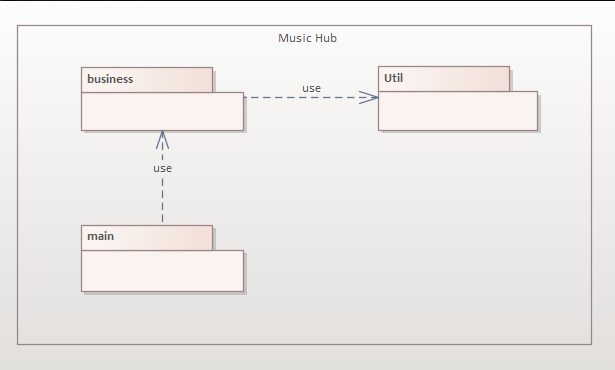


Diagramme de composants :

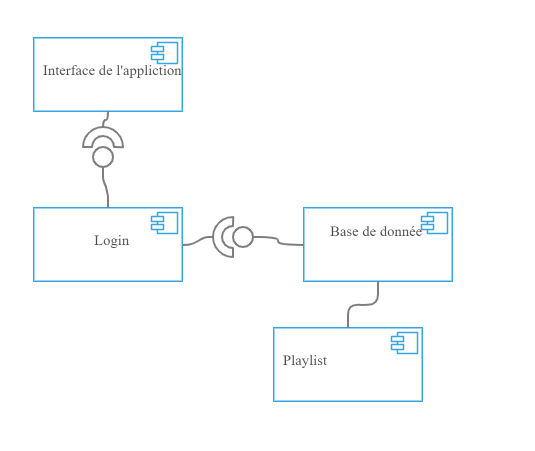


Diagramme de déploiement :

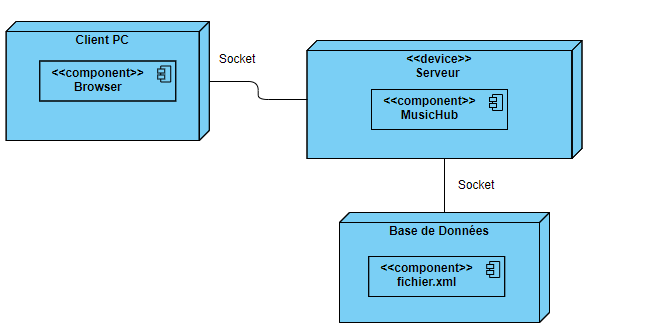
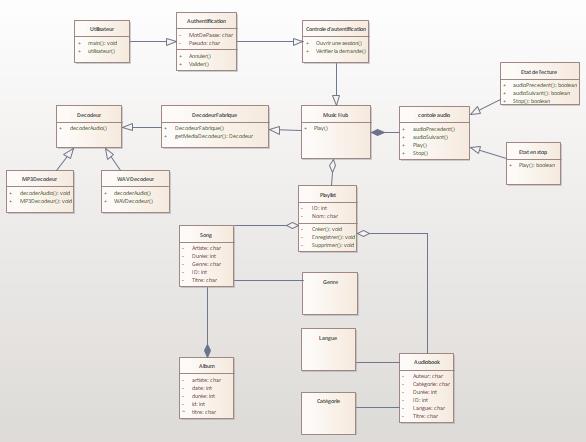
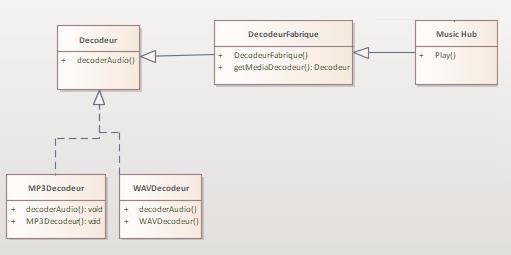


Diagramme de classes :



1. Patrons de conceptions



1. Analyse du principe SOLID

Responsabilité unique : Les différentes classes du projet sont responsables chacune de la ou les fonctionnalités qu’elles gèrent. La classe MusicHubUser.java utilise des méthodes uniquement liées à un User.

Ouvert/Fermé : Lorsqu’on utilise les méthodes issues d’autres classes dans MusicHubUser.java, on ne se soucie pas de modifier ces classes (ex : Albums.java, AudioBook.java, Main.java, ClientConnection.java, etc…)

Liskov : En utilisant AbstractServer as = new MusicHubServer(); pour initialiser le serveur, on respecte le principe de Liskov. On pourrait remplacer AbstractServer par MusicHubServer et cela ne modifierait pas le programme.

Ségrégation des Interfaces : On utilise une interface et des énumérations spécifiques.

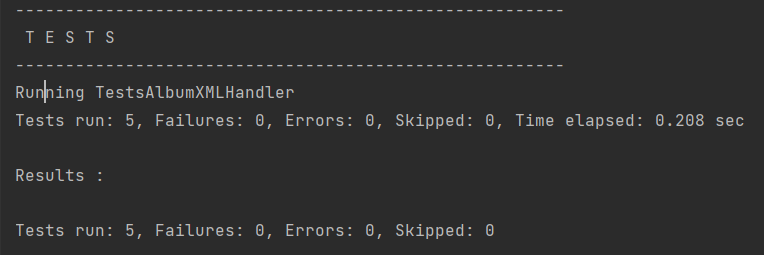
Inversion des dépendances : MusicHubServer dépend uniquement d’une abstraction : AbstractServer

IV-/ Le Git-Hub

<https://github.com/adriengel974/Projet_GLPOO>

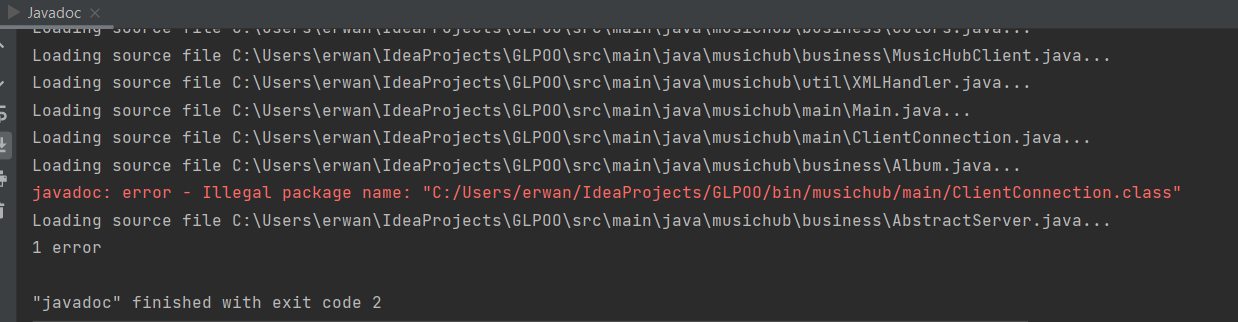
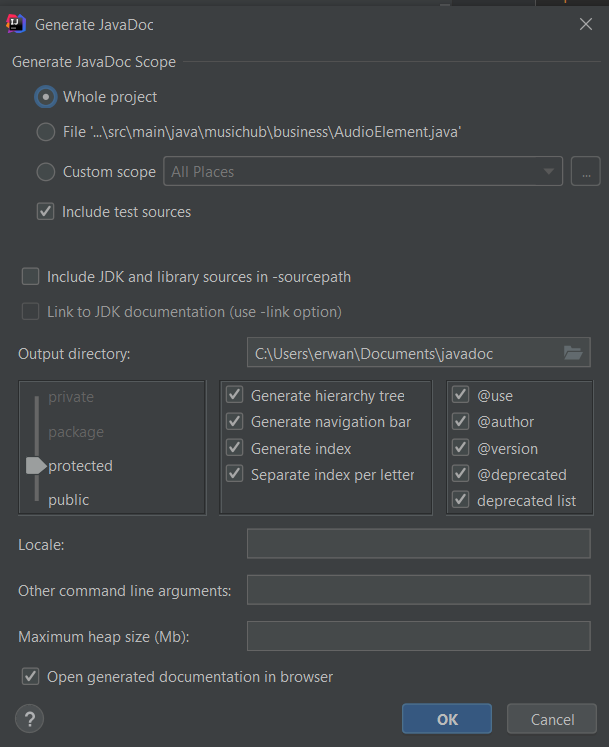
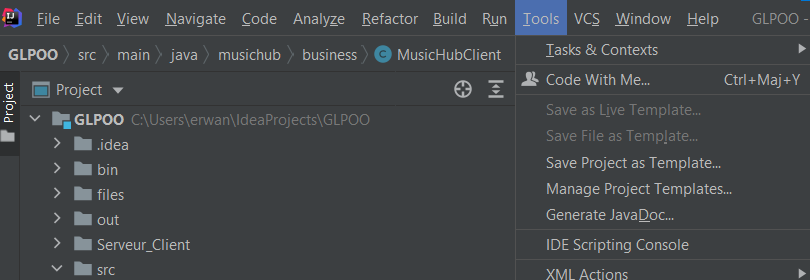
V-/ Les tests

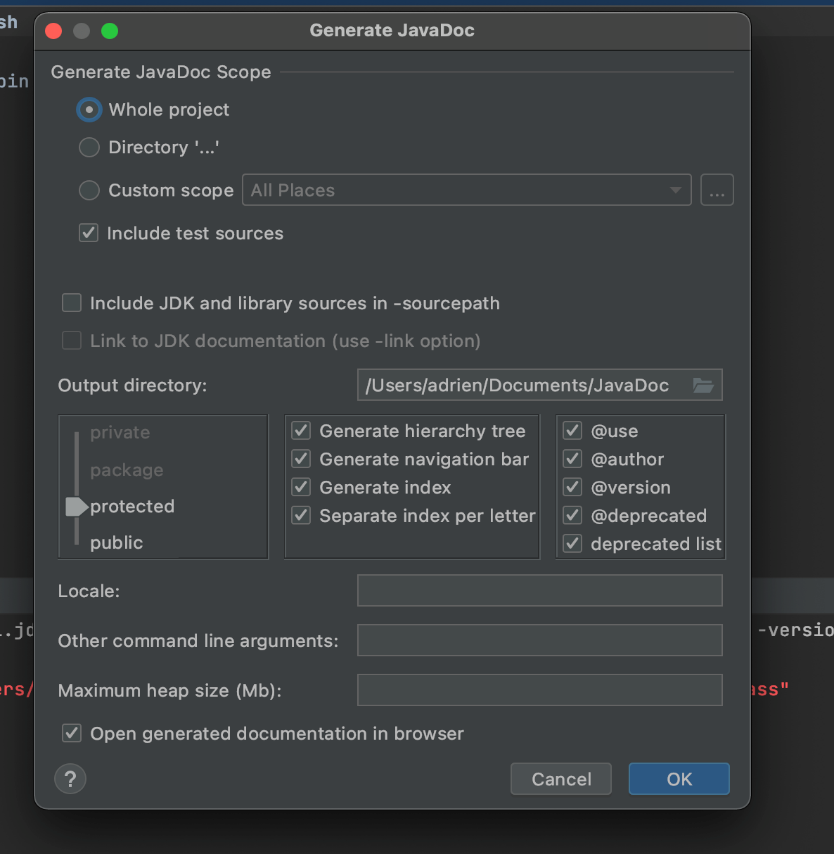
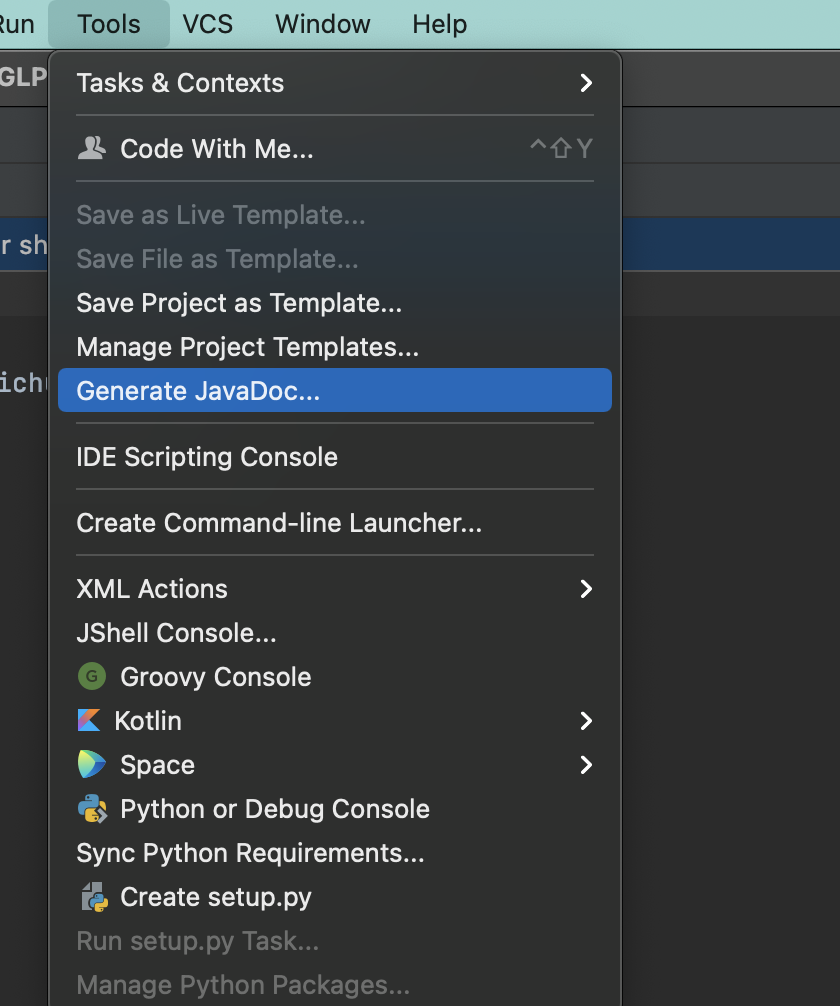




Les tests on été effectué sur les Class **Album** et **XMLHandler**

VI-/ Problème





On a eu des problèmes lors de la génération de la Javadoc. On a essayé avec la même version du projet sur deux ordinateurs différents de générer la Javadoc mais on a obtenu deux erreurs distinctes. Du coup on n’arrive pas à comprendre d’où vient le problème et on ne sait donc pas comment le régler.