Rapport

Projet GLPOO

2020/2021

MEMBRES :

Collaty Srikanth

Niakate Boubacar

Razafimahefa Timoty

Wahabi Mohammed

Sommaire

Introduction

Le but du projet génie logiciel est de vous faire travailler de manière collaborative sur un sujet donné, en mettant à l'œuvre les notions du cours, en améliorant vos compétences en POO et Java, et en produisant tous les artefacts d'un processus de développement de projet.

Mission

On vous demande de faire évoluer l'application de gestion de contenus musicaux jMusicHub développée sur le premier semestre dans le module POO - Java. Vous pouvez soit partir de votre implémentation de jMusicHub, soit de celle fournie sur Moodle. L'application sera un programme de bureau développé en Java (pas de Web, ni mobile) et elle sera conçue en architecture client-serveur. Il est impératif que minimum un client puisse se connecter au serveur. On pourra créer des chansons, des albums ou des playlists sur le serveur. On pourra afficher sur le(s) client(s) les chansons, les albums et les playlists du serveur. On pourra écouter les morceaux de musique sur le(s) client(s). Indication: La lecture de la musique se fera en utilisant la classe SourceDataLine dans la Java Sound API ; un AudioInputStream peut être utilisé afin de recevoir les données audios depuis le serveur. L'application n'a pas d'interface graphique, c'est une application console. L'information sur la console serveur sera mise à jour automatiquement quand du nouveau contenu (par exemple une nouvelle chanson) est disponible sur le serveur. Sur le client, la mise à jour se fera sur commande utilisateur. Il n'y a pas de persistance de données sur le client. L'application disposera d'un système de journalisation des erreurs qui permettra l'écriture des erreurs, avertissements ou informations survenues lors de l'exécution dans un fichier texte. Chaque ligne d'erreur, avertissement ou information contiendra un horodatage (date, heure).

Livrables

On attend une application opérationnelle. Une application avec moins de fonctionnalités mais sans bugs est préférable a une application pleine de fonctionnalités qui ne marchent pas !

Vous devez former des équipes de 3 ou 4 étudiants et les enregistrer sur la page Moodle du cours.

Un projet avec l'information suivante sera livré avec le code :

3.1. L'organisation du projet La liste des étudiants dans l'équipe avec leurs rôles et la description de ces rôles, avec les responsabilités de chacun

3.2. La spécification: Le Cahier de charges: - la liste des exigences fonctionnelles et non-fonctionnelles - le diagramme UML des cas d'utilisation, de séquence - le backlog des User Stories et Constraint Stories, chacune ayant aussi ses critères d'acceptance (lien vers le projet Trello dans lequel vous avez invité l'utilisateur felicia.ionascu@esiea.fr)

3.3. La conception : - 4 diagrammes UML : package, composants, déploiement, classes; démontrant la décomposition en modules de l'application et les relations entre les différentes entités - les patrons de conception utilisés - une analyse de la conception selon les principes SOLID avec argumentaire

3.4. Un lien vers le projet sous GitHub, qui contient : - le projet Eclipse/IntelliJ et les sources - des tests unitaires exécutables avec JUnit - un script Maven pour le build, génération de documentation avec javadoc, le test utilisant JUnit et le packaging en .jar Indication https://maven.apache.org/plugins/maven-javadoc-plugin/usage.html - tout autre fichier nécessaire à l'exécution du programme (par exemple, un readme).

3.5. Le test : - un rapport de test avec: les tests d'acceptance exécutés, la liste des bugs trouvés avec leur priorité et sévérité.

Equipe

Le groupe est composé de 4 membres.

Collaty Srikanth (3A-31)

lien GitHub: https://github.com/srikanthsc/projet.git (principal)

Niakate Boubacar (3A-31)

Razafimahefa Timoty (3A-31)

Wahabi Mohammed (3A-31)

Trello

Lien : <https://trello.com/b/5AYx2n5t/projet-glpoo>

Trello est un outil de gestion de projet en ligne, lancé en septembre 2011 et inspiré par la méthode Kanban de Toyota. Il repose sur une organisation des projets en planches listant des cartes, chacune représentant des tâches

Repartions des Taches

Srikanth COLLATY : Diagrammes UML de package, composants, déploiement, classes démontrant la décomposition en modules et classes de l'application. Conception de l'application client-serveur, et Deux design patterns codés. Gestion du Trello, User Stories et Contraint Stories, exigence fonctionnelle et non fonctionnelle.

Timoty Razafimahefa : la liste des tests d'acceptance exécutés, la liste des bugs trouvés - javaDoc -Code : Un script Maven pour le build, la génération de documentation avec javadoc, le lancement des tests unitaires et le packaging.

Mohamed Wahabi : Des classes de tests unitaires exécutables avec JUnit Dans le rapport : critères rightBICEP utilisé.

Boubacar Niakate : Une analyse de la conception selon les principes SOLID et test unitaire.

Exigences fonctionnelles et non-fonctionnelles

Définition exigences fonctionnelles et non fonctionnelles :

Exigences fonctionnelles : Affirmation concernant : les services qu'un système doit offrir comment le système doit réagir à des entrées particulières, comment le système doit se comporter dans des situations particulières, Peuvent aussi affirmer ce que le système ne doit pas faire, Dépendent du type de logiciel et de ses utilisateurs

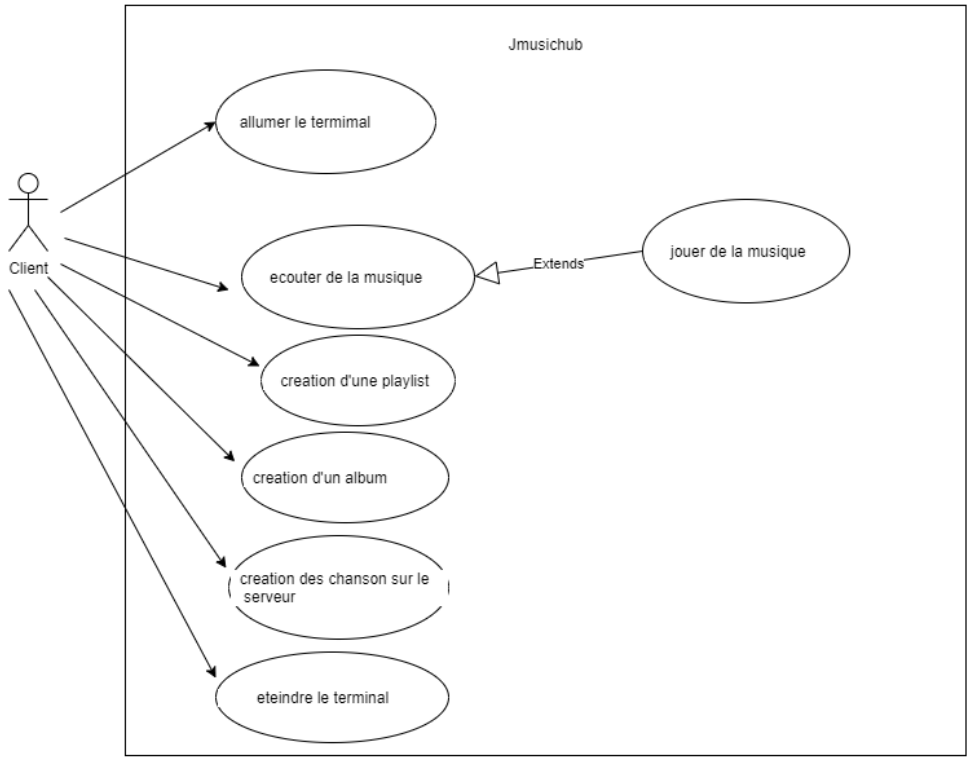
Exigences non fonctionnelles : Des contraintes sur les services ou les fonctions offertes par le système : contraintes de temps, contraintes du processus de développement, des standards, etc., Souvent ils s'appliquent au système entier plutôt qu'aux fonctionnalités ou services séparés, Exigences du domaine, Contraintes provenant du domaine d'exploitation

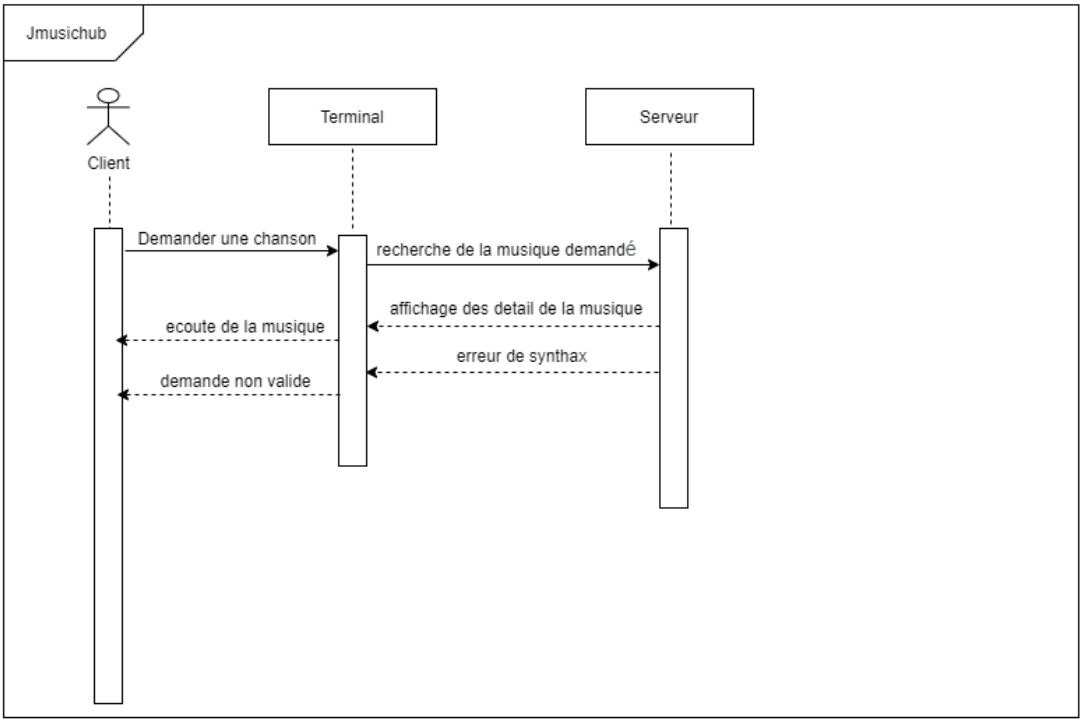
ID	Nom	Type	Description
SY_REQ_1	Fonctionnalité de base	F	L'application permettra d'écouter du contenu musical.
SY_REQ_2	Disponibilite	F	La Lecture de musique se fait gratuitement
SY_REQ_3	commentaires	NF	Commentaires dans le code
SY_REQ_4	Affichage	F	Affichage des informations sur le terminal
SY_REQ_5	Journal des erreurs	NF	un système de journalisation des erreurs qui permettra l'écriture des erreurs
SY_REQ_6	Format	F	Format des fichiers: « mp3 ou wav »
SY_REQ_7	Mise jour	F	L'information sur la console serveur sera mise à jour

			automatiquement quand du nouveau contenu (par exemple une nouvelle chanson) est disponible sur le serveur
SY_REQ_8	Communication	F	La communication se fait entre le client et le serveur
SY_REQ_9	classe	NF	La lecture de la musique se fera en utilisant la classe SourceDataLine
SY_REQ_10	Autre Fonctionnalité	F	L'utilisateur pourra créer des chansons, des albums et des playlist

- Il y a 7 exigences fonctionnelles et 3 exigences non fonctionnelles.

Diagramme UML de cas d'utilisation et diagramme UML de séquence





Diagrammes UML de package

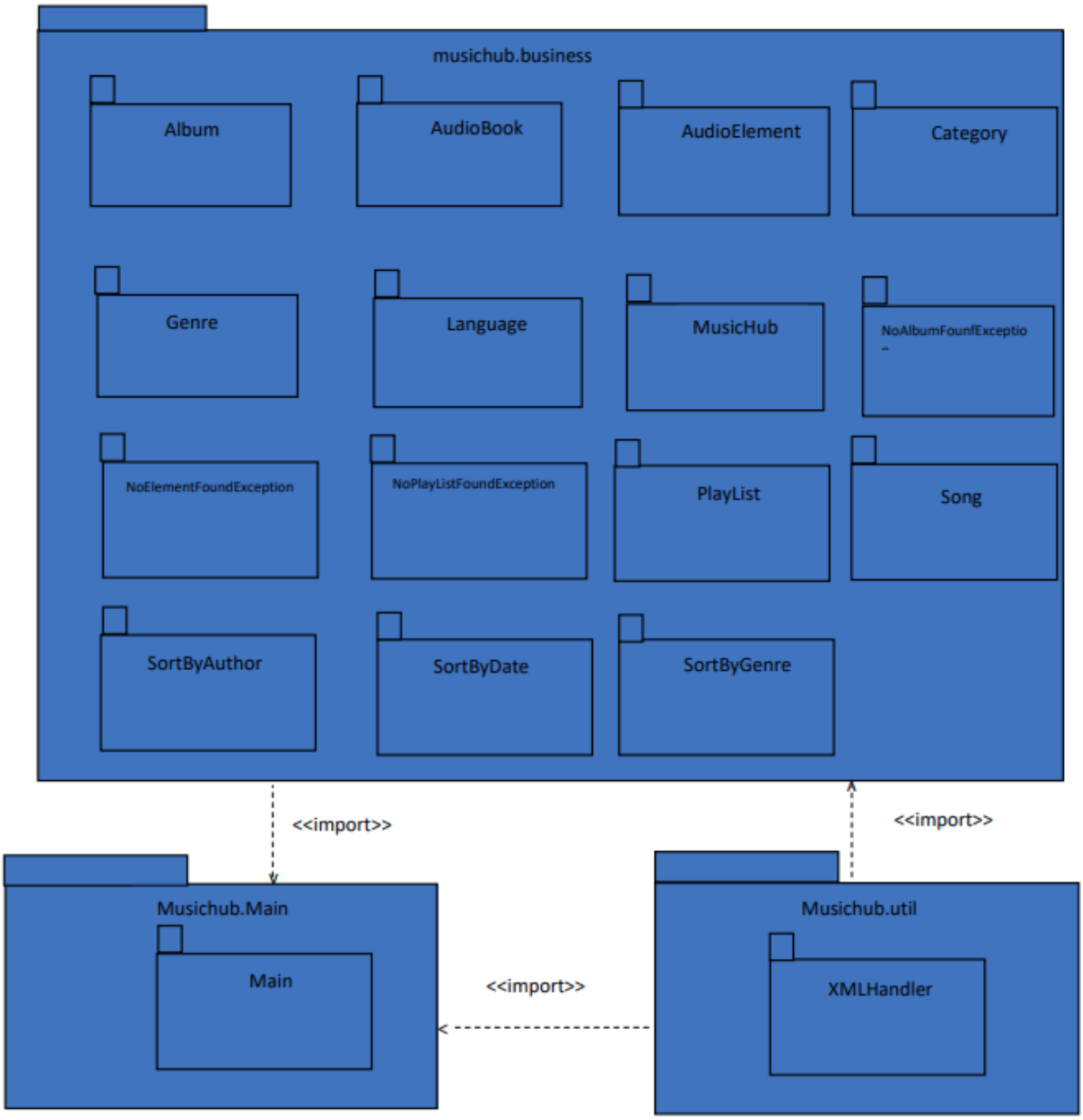


Diagramme UML de déploiement

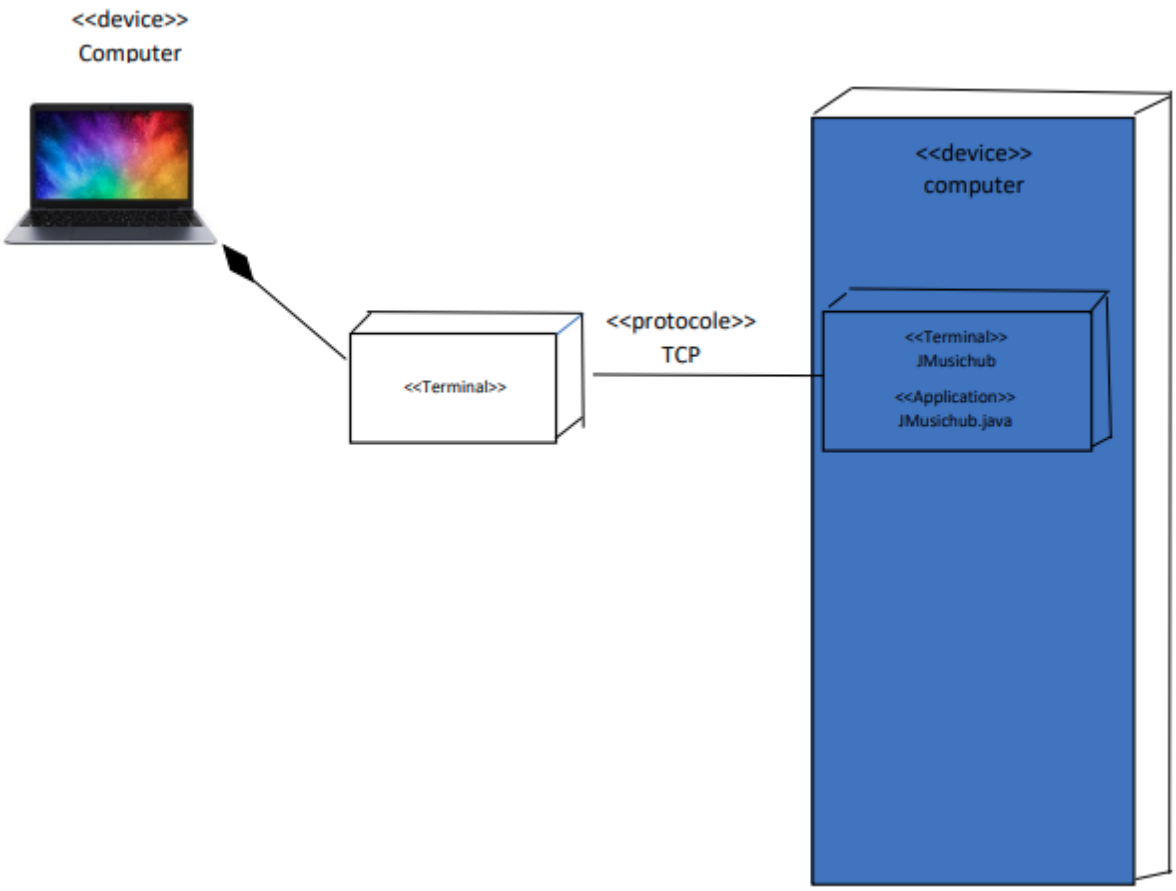
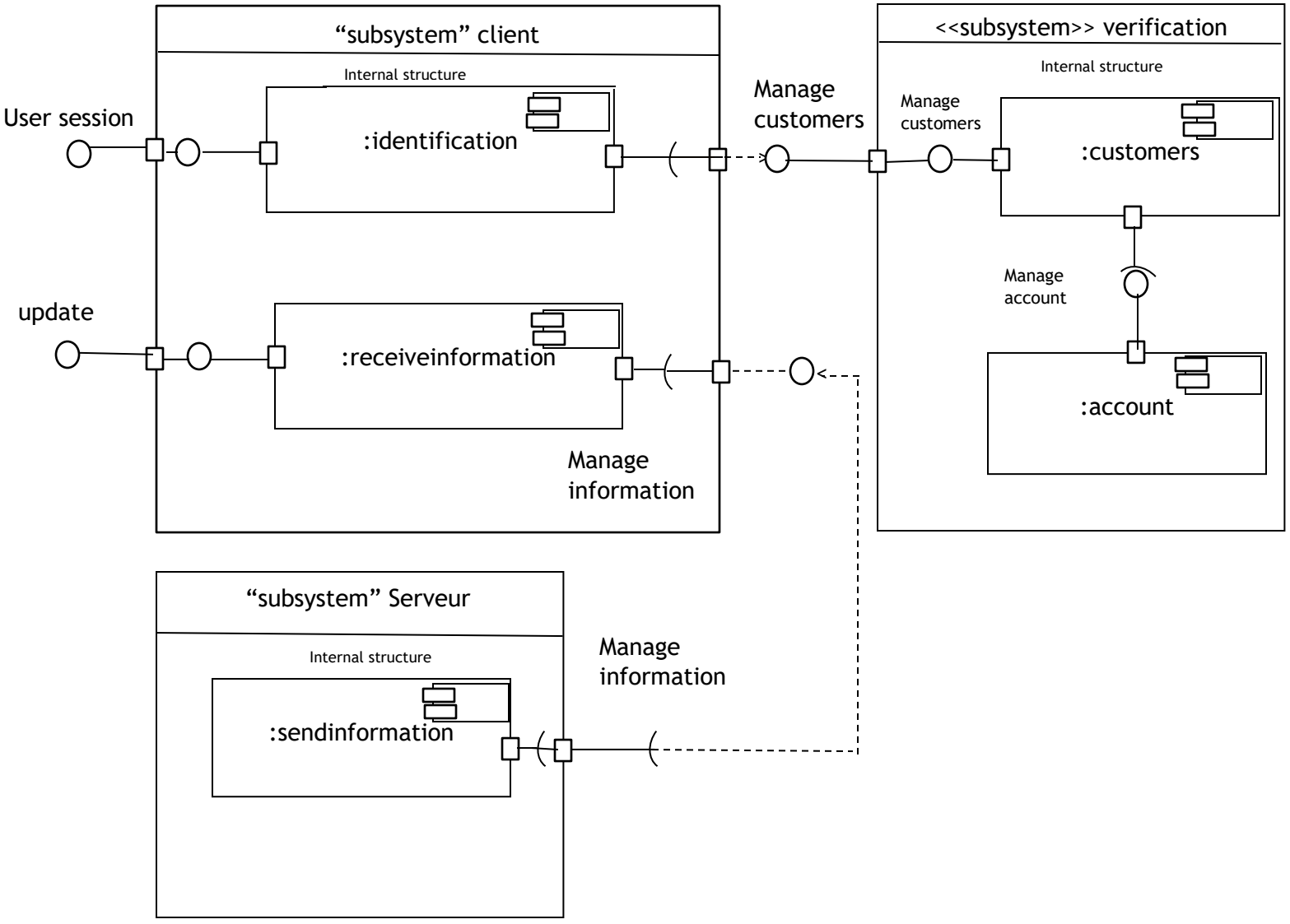


Diagramme UML de classe

Le diagramme uml est a retrouver sur le lien github suivant :
https://github.com/srikanthsc/projet/blob/main/diagramme%20uml%20de%20classe.png

Diagramme UML de composant



Principe solide

Le principe de responsabilité unique est respecté :
Chaque module, classe, ou méthode doit être responsable d'une seule partie de la fonctionnalité que le logiciel fournit, et cette responsabilité devrait être entièrement encapsulée par la classe, le module ou la méthode
Album.java : respecté
AudioBook.java : respecté
AudioElement.java : respecté
PlayList.java : respecté
Song.java : respecté
La structure des autres fichiers .java n'implique pas une vérification
.
Le principe ouvert et fermé n'est pas respecté :
Toutes les entités logicielles doivent être ouvertes à l'extension, mais fermées à la modification
En effet dans la classe MusicHub on a if (ae instanceof AudioBook) ou bien if (el instanceof Song) etc.., le problème ici est que la classe MusicHub subira des modifications si l'on modifie la class Song et la classe AudioBook.

Le principe de substitution de Liskov est respecté :

Si S est un sous-type de T, alors les objets de type T peuvent être remplacés avec des objets de type S sans altérer aucune des propriétés du programme
Dans notre programme on a Song et Audiobook qui sont des classes qui sont implementé de AudioElement
Toutes les méthodes de AudioElement sont utilisées par les classes implementés par AudioElement donc le principe de Liskov est respecté.

La ségrégation des interfaces est respectée :

Une classe ne doit jamais être forcée à implémenter une interface qu'elle n'utilise pas ou une méthode qui n'a pas de sens pour elle.
Dans le programme il n'y a aucune class qui est forcée à implémenter une interface qu'elle n'utilise pas ou une méthode qui n'a pas de sens pour elle.

Le principe d'inversion des dépendances :

Les entités doivent dépendre uniquement des abstractions.
Les classes de haut niveau ne doivent pas dépendre des classes de bas niveau. Les deux doivent dépendre des abstractions
On a une qu'une seule classe abstraite AudioElement et ses classes filles Song et AudioBook qui dépendent d'elle donc on peut conclure que ce principe est respecté.

User Stories et Constraint Stories

us 1.1 En tant qu'user je veux que les informations des chansons soit affiché sur le terminal.

us 1.2 En tant qu'user je veux écouter a partir du terminal des chansons.

us 1.3 En tant qu'user je veux créer une playlist, un album, une chanson sur le serveur

us 1.4 En tant qu'user je veux écouter gratuitement de la musique.

us 1.5 En tant qu'user je veux écouter plusieurs musiques différentes.

us 1.6 En tant qu'user je veux que les informations des chansons soit affiché sur le terminal.

cs 1.1 (Interopérabilité)

Le système sera codé en java

cs 1.2 (Interopérabilité)

Les données seront stockées dans un fichier txt ou xml

cs 1.3 (Portabilité)

L'application peut être compatible avec windows ou linux.

cs 1.4 (Sécurité)

Pour accéder aux services de l'application, le client doit entrer un identifiant et un mot de passe

Conception de l’application Client-Serveur

Le Serveur

Le serveur java initialise la connexion, il lance l'écoute sur un port et se met en attente des connexions entrantes pour qu'il les accepte. Par exemple, le numéro de port est 5000, le client envoie une demande au serveur avec ce numéro de port 5000. Le serveur accepte la demande et transmit ses informations (adresse ip) au client. Maintenant, la connexion est établie et un échange de messages peut se faire.

Java fournit un package java.net qui traite tout ce qui est réseau, on utilise deux classes :

java.net.ServerSocket: cette classe accepte les connexions venues des clients.

java.net.Socket: cette classe permet de se connecter à la machine distante.

On a besoin aussi d'un outil pour saisir, envoyer et recevoir le flux:

Scanner: lire les entrées clavier.

BufferedReader : lire le texte reçu à partir de l'émetteur.

PrintWriter : envoyer le texte saisi.

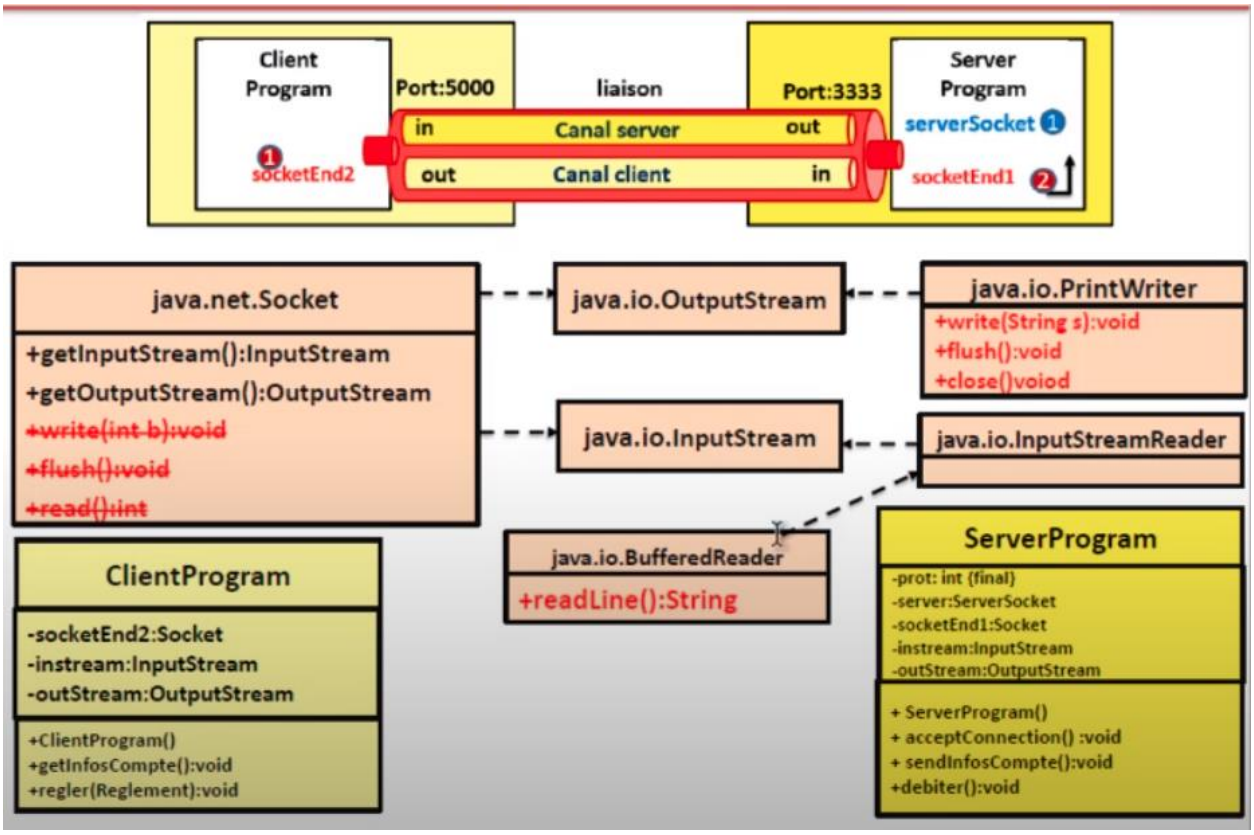
Après la création du socket serveur qui porte le numéro de port 5000, le serveur attend les connexions entrantes et dès que une est détectée, il l'accepte. Les deux variables qui gèrent le flux de lecture et d'écriture in et out sont initialisées pour qu'elles soient reliées directement avec le flux d'envoi et de réception.

Pour nos cas la connexion entre client et serveur se fait infiniment avec la boucle while(), et plus presicement la boucle while permet de determiner si la connexion est toujours etablie.

Le Client

Le client n'a besoin qu'à la classe Socket pour établir la connexion serveur, le constructeur prend en entrée l'adresse IP du serveur et le numéro de port. Le reste du code est identique a celui du serveur.

Voici le schema des echanges de donnees entre le serveur et le client


Client Program
Port:5000
liaison
Port:3333
Server Program
in
Canal server
out
serverSocket 1
socketEnd2
out
Canal client
in
socketEnd1 2
java.net.Socket
+getInputStream():InputStream
+getOutputStream():OutputStream
+write(int b):void
+flush():void
+read():int
java.io.OutputStream
java.io.PrintWriter
+write(String s):void
+flush():void
+close()voiod
java.io.InputStream
java.io.InputStreamReader
java.io.BufferedReader
+readLine():String
ClientProgram
-socketEnd2:Socket
-instream:InputStream
-outStream:OutputStream
+ClientProgram()
+getInfosCompte():void
+regler(Reglement):void
ServerProgram
-prot: int (final)
-server:ServerSocket
-socketEnd1:Socket
-instream:InputStream
-outStream:OutputStream
+ ServerProgram()
+ acceptConnection() :void
+ sendInfosCompte():void
+debiter():void

Fonctionnalité de l’application

Notre application dispose de plusieurs fonctionnalités qui sont les suivantes :

- L’utilisateur peut choisir les musiques
- Affichage des informations sur le client
- Envoie des mises a jours sur le client
- Le client a la possibilité de quitter l’application
- La possibilité d’écouter plusieurs musiques
- Envoie des informations entre le serveur et le client
- Création de musique sur le serveur
- Affichage de la liste des playlists sur le serveur
- Un logo afficher lorsque le client et en marche
- La possibilité d’envoyer des messages depuis le client (messagerie instantanée entre le client et le serveur)
- Connexion avec un identifiant pour accéder aux services de l’application
- Possibilité de rechercher une musique (serveur)
- Le client et informer de ce qui passe sur l’application
- Affichage des chansons disponibles

Notice d’emplois de l’application

Pour faire fonctionner l’application, compiler en premier le serveur et par la suite le client.

Le serveur proposera de chercher, ajouter, une musique, et aussi de quitter la phase d’initialisation et de mise à jour, d’âpres la capture ci-dessous :

```
C:\Users\Admin\Desktop\PROJET GLPOO 2>java Serveur
Welcome to the JMusicHub server!
1 - Add song data
2 - List all songs
3 - Search for song by Artist
4 - Search for song by Album
5 - Quit the Program
```

Pour l’ajout ou la recherche, saisir le numéro de l’action voulue, une fois cette phase terminer le serveur va attendre la connexion du client pour qu’il puisse interagir en temps réels.

```
1 - Add song data
2 - List all songs
3 - Search for song by Artist
4 - Search for song by Album
5 - Quit the Program
5
waiting for the connection.....
```

Une fois le client compiler, on a une interface dans laquelle un logo apparait et avec un message de bienvenue :

Le client et toute suite informer des musiques mis à jour depuis le serveur.

```
update done
Jam,Michael Jackson,Dangerous
titanium,david guetta,nothing but the beat
taki taki,dj snake, carte blanche
beat it,Micheal Jackson,thriller
Billie Jean,Micheal Jackson ,thriller
```

Par la suite le client demandera de vous identifier avec un identifiant et un mot de passe :

Voici l'identifiant et le mot de passe pour accéder aux services :

IDENTIFIANT: prof

MOT DE PASSE: esiea

```
Veuillez saisir votre identifiant : prof
Veuillez saisir votre mot de passe : esiea
Bonjour  Proffesseur, Authentification Accepter
```

Apres taper s pour continuer

```
Taper s (start) pour continuer
s
Enjoy the service!
```

Le client recevra une proposition de musique, et il devra choisir une musique qui veut écouter.

Le client et informer de la chanson qu'il écoute preuve :

```
selectionner un numero pour jouer la musique
1- weekend save your tears
2- weekend Bliding light
3- ninho
1
weekend en ecoute
```

Une fois l'écoute terminer le client pourra choisir de quitter l'application ou de rester connecter

Pour se déconnecter saisir DISCONNECT et pour rester connecter saisir CONTINUE.

```
ecoute terminer
diconnected or conected
```

Si on continue, on découvre d'autre fonctionnalité de l'application.

Et pendant ce temps le serveur et connecter au client.

```
waiting for the connection.....
connected
Client : lecteur a l'arret
Client : appuyer sur sur touche a partir du serveur pour les chansons disponibles
Client : s
s
```

```
diconnected or conected
continue
connected
s
Serveur : [Alone Again - The Weeknd , Too Late - The Weeknd , Hardest to Love - The Weeknd , Scared to Liv
e - The Weeknd , Snowchild - The Weeknd , Escape from LA- The Weeknd , Heartless -The Weeknd , Faith - The
 Weeknd , Blinding Lights - The Weeknd , In Your Eyes - The Weeknd , Save Your Tears - The Weeknd ]
```

Le serveur et bien informer que le client a fini d'écouter une chanson avec la phrase suivante :

« Lecteur à l'arrêt ».

Par la suite suivre l'instruction indique sur le serveur.

On peut observer que le serveur a transmis au client les chansons qui étaient disponible, d'après la capture précédente.

Sur le client :

```
Serveur : [Alone Again - The Weeknd , Too Late - The Weeknd , Hardest to Love - The Weeknd , Scared to Liv
e - The Weeknd , Snowchild - The Weeknd , Escape from LA- The Weeknd , Heartless -The Weeknd , Faith - The
 Weeknd , Blinding Lights - The Weeknd , In Your Eyes - The Weeknd , Save Your Tears - The Weeknd ]
Serveur : music available
Serveur : update done
```

Le client reçoit de la part du serveur les musiques disponibles, et aussi informer par message des musiques disponible et aussi que la mise à jour a était effectuer avec les phrases suivantes : « music available » et « update done »

Et la dernière fonctionnalité que dispose l'application c'est la messagerie instantanée :

```
bonjour
Client : bonjour
```

```
Serveur : bonjour
bonjour
```

On voit bien que le serveur et le client communique en continue.

Sachant que l'espace de stockage et limiter sur Moodle, télécharger les autres chansons sur le lien suivant :

Chanson 2 :
https://drive.google.com/file/d/1ppxH5E1KdUNRaSjMAbic9c1KFWUxXJHM/view?usp=sharing

Chanson 3 :

https://drive.google.com/file/d/18SU02Wj2HUH827zkU7aXBKhzcd5plfc0/view?usp=sharing

Développement de l'application

L'application a était développer à partir du cahier des charges qui nous a était imposées.

Le logo de l'application se trouve dans un fichier txt développer a partir des caractères ASCII.

Pour l'écoute de la musique se fait grâce à la classe sourcedataline.

L'application dispose de deux design pattern, les deux pattern design utilisées sont les singletons.

Définition patron singleton d'après le cours :

Le patron Singleton permet la création d'un objet unique en son

Genre pour lequel il n'y a qu'une seule instance

• Utile pour :

• Boîtes de dialogue, objets qui gèrent des préférences, des réglages

Registres

• Les traceurs, des pilotes de périphériques pour imprimantes ou cartes

Graphiques

• Les pilotes d'équipements matériels

• Point d'accès unique à une base de données

L'application peut toujours être implémenter si on veut mettre d'autre fonctionnalité, l'application et ouvert à l'extension et fermer à la modification avec les interfaces que dispose l'application.

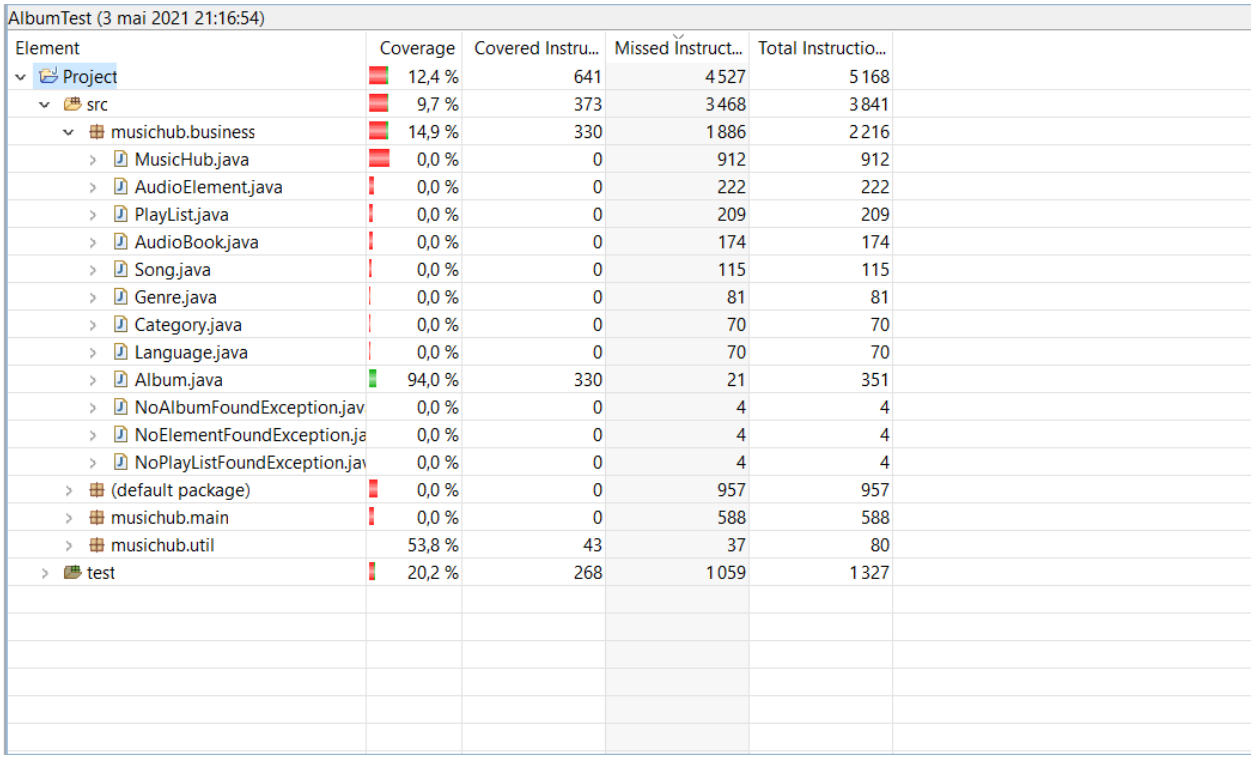
Les tests unitaires :

Mohamed Wahabi :

Réalisation des tests unitaires de AlbumTest et MusicHubTest sous junit5 avec repérage et correction des différentes erreurs trouver vis-à-vis du code avec création de nouvelles méthodes pour alléger le code.

-familiarisation avec de nouveau concept comme iterateur en java ainsi que les Nodes.

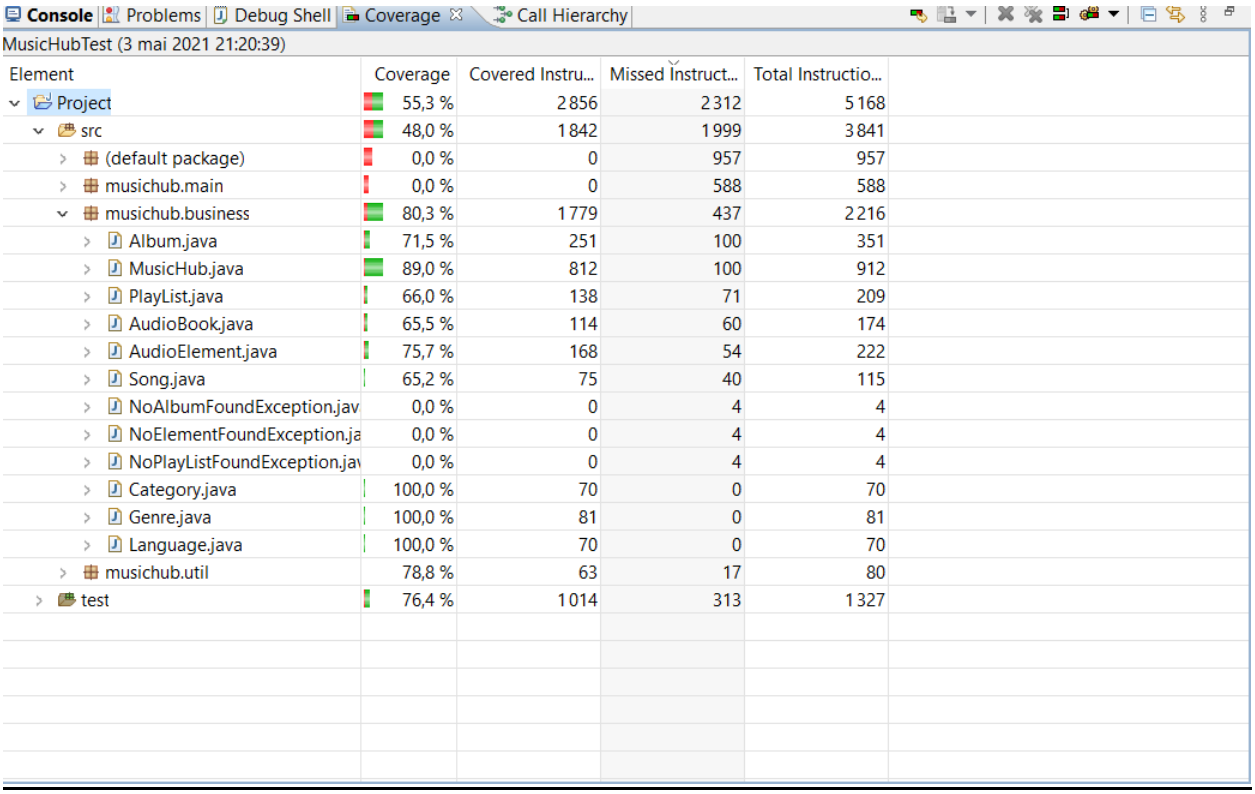
ALBUMTEST :



AlbumTest (3 mai 2021 21:16:54)

Element	Coverage	Covered Instru...	Missed Instruct...	Total Instructio...
Project	12,4 %	641	4 527	5 168
src	9,7 %	373	3 468	3 841
musichub.business	14,9 %	330	1 886	2 216
MusicHub.java	0,0 %	0	912	912
AudioElement.java	0,0 %	0	222	222
PlayList.java	0,0 %	0	209	209
AudioBook.java	0,0 %	0	174	174
Song.java	0,0 %	0	115	115
Genre.java	0,0 %	0	81	81
Category.java	0,0 %	0	70	70
Language.java	0,0 %	0	70	70
Album.java	94,0 %	330	21	351
NoAlbumFoundException.jav	0,0 %	0	4	4
NoElementFoundException.ja	0,0 %	0	4	4
NoPlayListFoundException.jav	0,0 %	0	4	4
(default package)	0,0 %	0	957	957
musichub.main	0,0 %	0	588	588
musichub.util	53,8 %	43	37	80
test	20,2 %	268	1 059	1 327

MUSICHUBTEST :



Element	Coverage	Covered Instru...	Missed Instruct...	Total Instructio...
Project	55,3 %	2856	2312	5168
src	48,0 %	1842	1999	3841
(default package)	0,0 %	0	957	957
musichub.main	0,0 %	0	588	588
musichub.business	80,3 %	1779	437	2216
Album.java	71,5 %	251	100	351
MusicHub.java	89,0 %	812	100	912
PlayList.java	66,0 %	138	71	209
AudioBook.java	65,5 %	114	60	174
AudioElement.java	75,7 %	168	54	222
Song.java	65,2 %	75	40	115
NoAlbumFoundException.jav	0,0 %	0	4	4
NoElementFoundException.ja	0,0 %	0	4	4
NoPlayListFoundException.jav	0,0 %	0	4	4
Category.java	100,0 %	70	0	70
Genre.java	100,0 %	81	0	81
Language.java	100,0 %	70	0	70
musichub.util	78,8 %	63	17	80
test	76,4 %	1014	313	1327

Sources

https://docs.oracle.com/javase/7/docs/api/javax/sound/sampled/DataLine.html

https://www.enseignement.polytechnique.fr/informatique/Java/1.8/javax/sound/sampled/DataLine.Info.html
https://docs.oracle.com/javase/7/docs/api/java/net/Socket.html
https://www.baeldung.com/a-guide-to-java-sockets
https://miashs-www.u-ga.fr/prevert/LicenceMIASS/Documents/B-API-socket-java.pdf
https://www.jmdoudoux.fr/java/dej/chap-junit.htm

Annexes

Tous les diagrammes sont en ligne sur le lien suivant :

https://github.com/srikanthsc/projet.git