esiea
INGÉNIEUR·E·S D'UN NUMÉRIQUE UTILE

2020-2021

Rapport de projet GLPOO

Génie Logiciel 3133

GENOVESE Melissa | LOUKAKOU-BATAILLE Eden | OZOUF Jonathan | SCEMAMA Johanne | TCHAN Ariitea
ESIEA | 74BIS AVENUE MAURICE THOREZ 94200 IVRY-SUR-SEINE

Project organization

We firstly brainstormed altogether to identify the requirements of our project, the interesting features that we wanted and felt able to implement as well as the corresponding constraints.
Melissa has been in charge to write down the final expression of the requirements and classify them as functional or non-functional. She also did the sequence diagram.
Johanne was responsible of the Trello organization and properly define user stories and their acceptance criteria. She determined the different level of progression for the user stories. She drew the use case diagram too.
Ariitea has been the one setting up the maven project in Eclipse and in charge of guiding us about the pom.xml file completion when needed (for the Javadoc for example) and the project structure. He then did the class UML diagram.
Eden worked on finding the best design patterns to use. He made the package, component, and deployment diagrams.
Jonathan did the analysis of the design according to the solid principle as well as the choice of the unit tests with the right-BICEP criteria. He hosts the GitHub repository and managed it too.
Johanne did the report about the acceptance tests in the end and this project report layout has finally been done by Melissa.

Our organization to implement the code was mostly Agile. Each one of us chose 2 user stories. The details are observable in Trello. Basically, we coded on a separate branch each part, merging when we had the occasion. One someone thought he or she was done with his/her part someone else was checking the respect of the requirements. It could also be the same person validating his/her own work. We commented the code in a common.
Jonathan coded the unit tests, and the feature to play the music. Eden improved the media player and user experience with lecture and queue management functionalities. Johanne implemented the server console with the automated data update. Melissa started the organization of the server-client structure based on the project of the first semester and developed it to be able to display the albums, playlists, and elements on a client. Ariitea carried out the possibility to update the data manually at any time on the client. He was also meant to deal with the error logging system, to write time stamped information but unfortunately, he went through deep personal troubles at the end of the year and was not able to pursue until the end of the project with us. (We are still wondering if he will have the possibility to stay in France and pass the year.)

Spécification

Le Cahier des charges :

- **La liste des exigences fonctionnelles et non-fonctionnelles**

ID	Nom	Type	Description
REQ-1	Fonctionnalité de base	F	Créer des chansons, des albums ou des playlists sur le serveur
REQ-2	Connexion	NF	Au minimum un client peut accéder au service via le serveur
REQ-3	Affichage	F	Afficher sur le client les chansons, les albums et les playlists du serveur
REQ-4	Écouter	F	Écouter les morceaux de musique sur le client
REQ-5	Automatisation	NF	Mise à jour automatique quand du nouveau contenu est disponible sur le serveur
REQ-6	Actualisation	NF	Mise à jour manuelle des données côté client
REQ-7	Maintenabilité	NF	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)
REQ-8	Disponibilité	F	Choix du prochain morceau.
REQ-9	Organisation / Gestion	F	Trier dans l'ordre alphabétique les morceaux musicaux
REQ-10	Contrôle	F	Mettre en pause un morceau
REQ-11	Flexibilité	F	Passer à la musique suivante sans terminer l'écoute de la musique en cours

Notre organisation sous Trello est accessible via ce lien :
https://trello.com/b/9jxBv9ff/projet-poo

On y retrouve les User et Constraint Stories agrémentée de critères de vérifications ainsi que leurs parcours dans notre processus.

- **Les diagrammes UML de cas d'utilisation et de séquence**

Diagramme de cas d'utilisation :

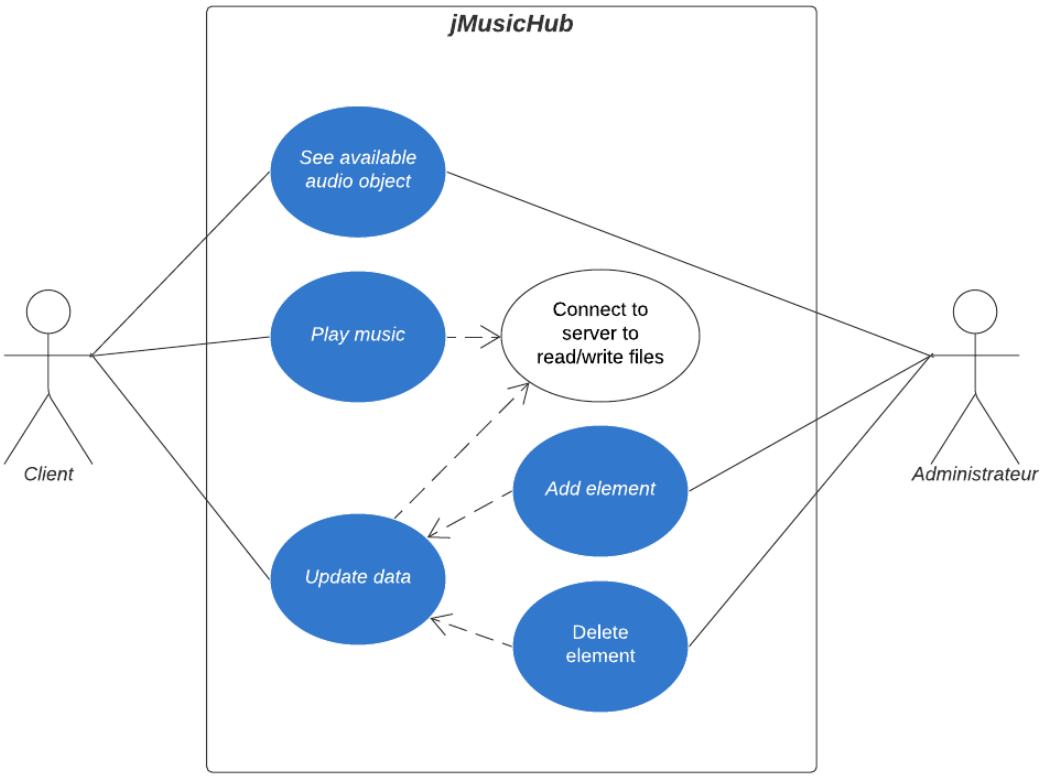
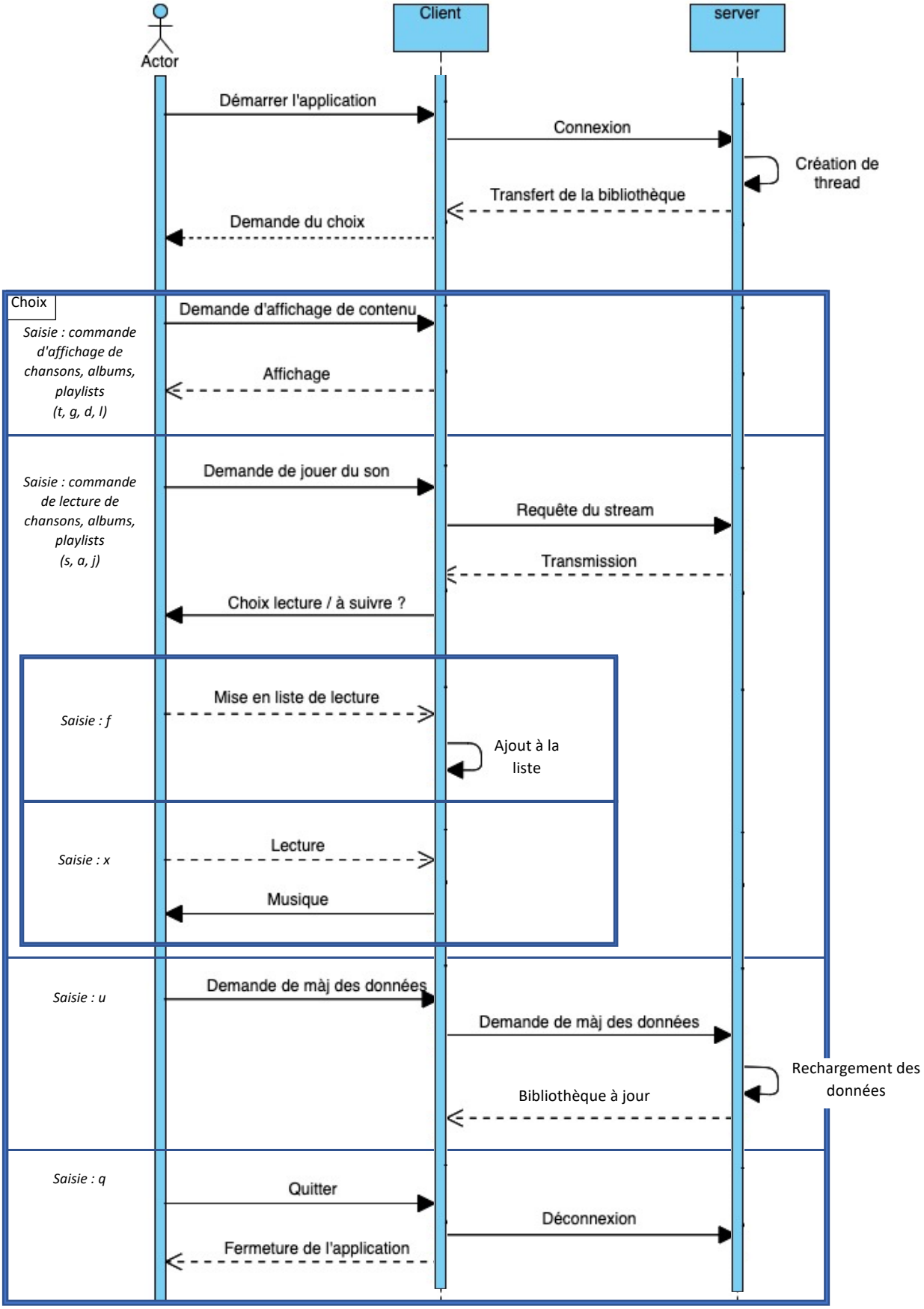


Diagramme de séquence :



- **Le backlog des User Stories et Constraint Stories**

User stories:

User Story 1
En tant qu'utilisateur je veux afficher sur le client les chansons, les albums et les playlists du serveur afin de choisir ce que je veux écouter.
Test d'acceptation : • Lorsque le serveur démarre alors j'ai le choix d'afficher les chansons, les albums ou les playlists. • Je peux sélectionner l'une de ces options et les chansons, albums ou playlists s'affichent.

User Story 2
En tant qu'utilisateur je veux écouter les morceaux de musique sur le client afin de me divertir.
Test d'acceptation : • Je peux sélectionner un morceau et cette musique démarre.

User Story 3
En tant qu'utilisateur je veux choisir le prochain morceau afin de l'écouter directement après le morceau en cours.
Test d'acceptation : • Lorsque je choisis le prochain morceau alors il est sauvegardé en tant que suivant. • Lorsque le morceau est fini, alors ce morceau débute.

User Story 4
En tant qu'utilisateur je veux que les musiques puissent être triées de différentes façons afin de faciliter l'utilisation de l'application.
Test d'acceptation : • Quand je clique "t" les noms des albums sont triés et affichés par date. • Quand je clique "g" les noms des chansons d'un album sont triés et affichés par genre. • J'ai le choix d'afficher les noms des albums triés par date ou les chansons d'un album triées par genre.

User Story 5
En tant qu'utilisateur je veux pouvoir mettre en pause un morceau afin de pouvoir choisir les moments d'écoute.
Test d'acceptation : • Lorsqu'une musique est en cours, des commandes sont disponibles pour arrêter la lecture et la reprendre

User Story 6
En tant qu'utilisateur je veux passer à la musique suivante sans terminer l'écoute de la musique en cours afin d'écouter ce que je souhaite instantanément.
Test d'acceptation : • Lorsque j'entre "n" alors la musique en cours est arrêtée et une autre musique commence automatiquement. • Lorsqu'une musique est en cours, une commande est disponible pour écouter la musique suivante.

User Story 7
En tant que fournisseur, je veux créer des chansons, des albums ou des playlists sur le serveur afin que les clients puissent les visualiser/écouter.
Test d'acceptation : • Lorsque je crée des chansons, albums, playlists alors ils peuvent apparaître sur la machine du client • Une interface est disponible côté serveur proposant des commandes pour créer et supprimer des chansons, albums, playlists

Constraint stories:

Contraint story 1
Minimum un client peut se connecter à la plateforme

Contraint story 2
Mise à jour automatique quand du nouveau contenu est disponible sur le serveur

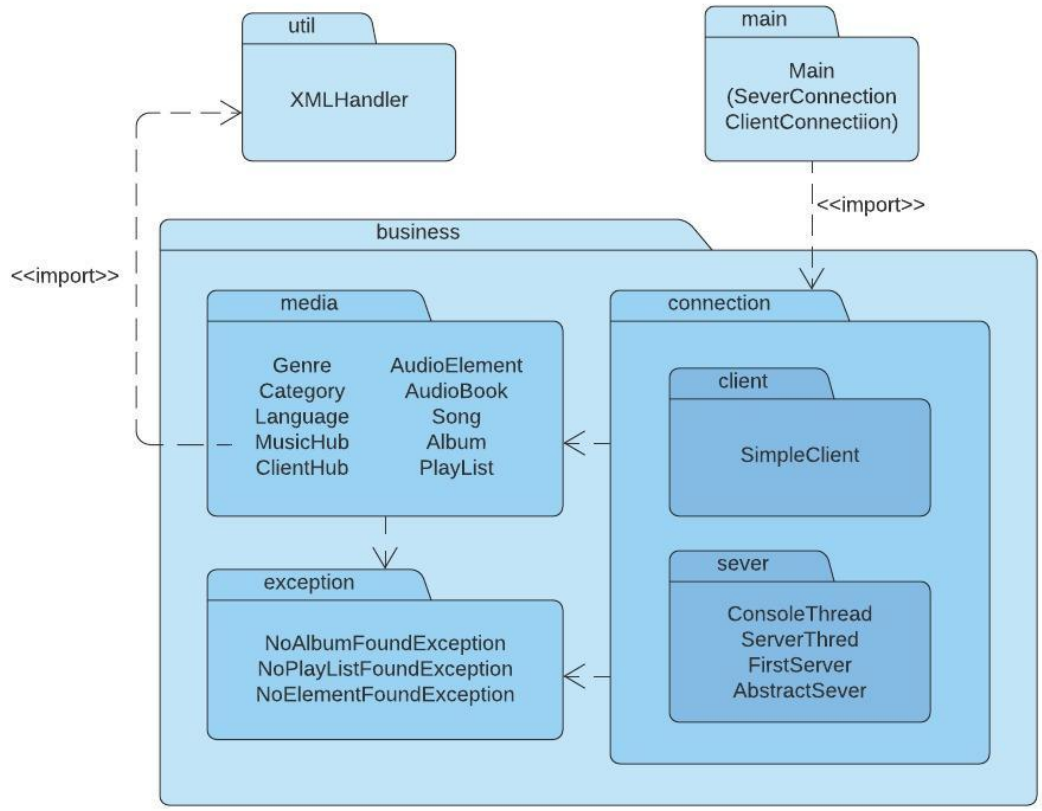
Contraint story 3
Mise à jour manuelle des données côté client afin de perfectionner mon espace.

Contraint story 4
Système de journalisation des erreurs qui permettra l'écriture des erreurs, avertissements ou inf ormations survenues lors de l'exécution.

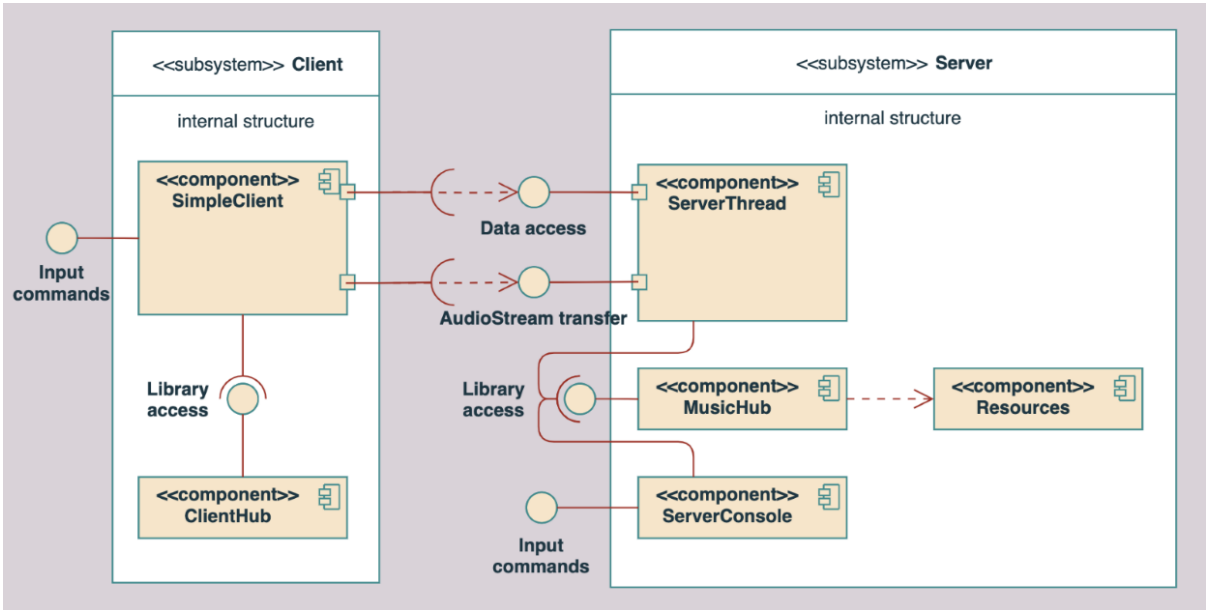
Conception

- **Diagrammes UML**

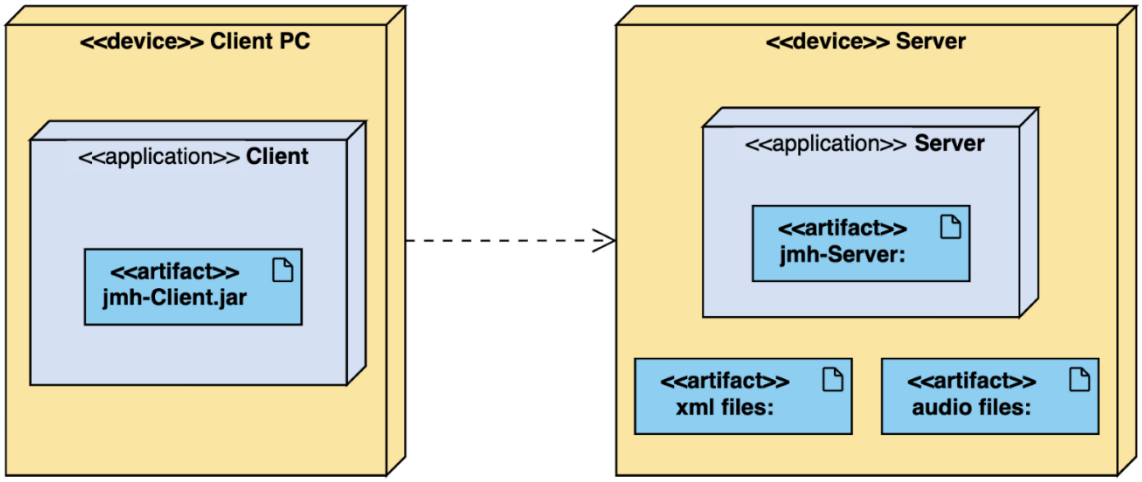
Paquets :



Composants :

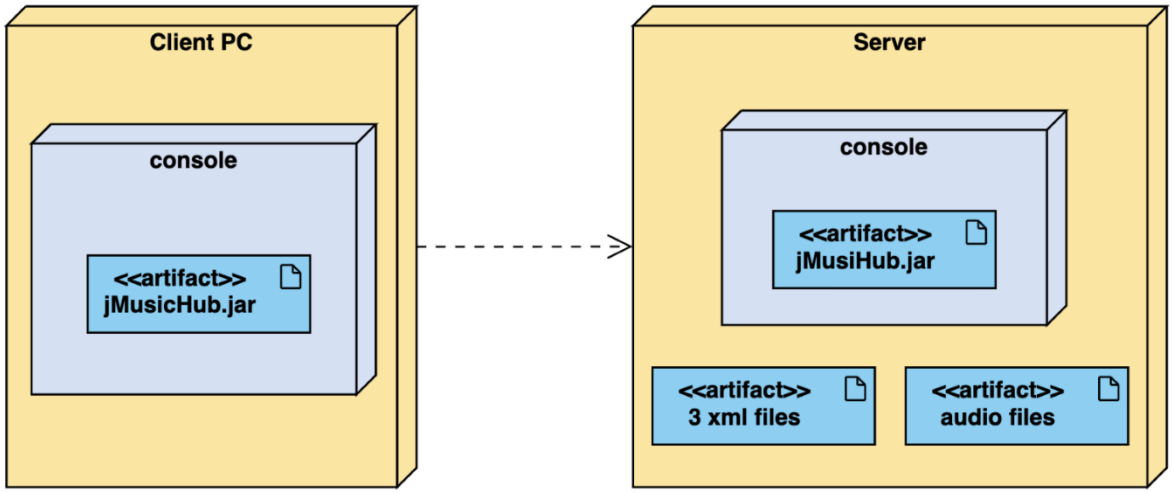


Déploiement :



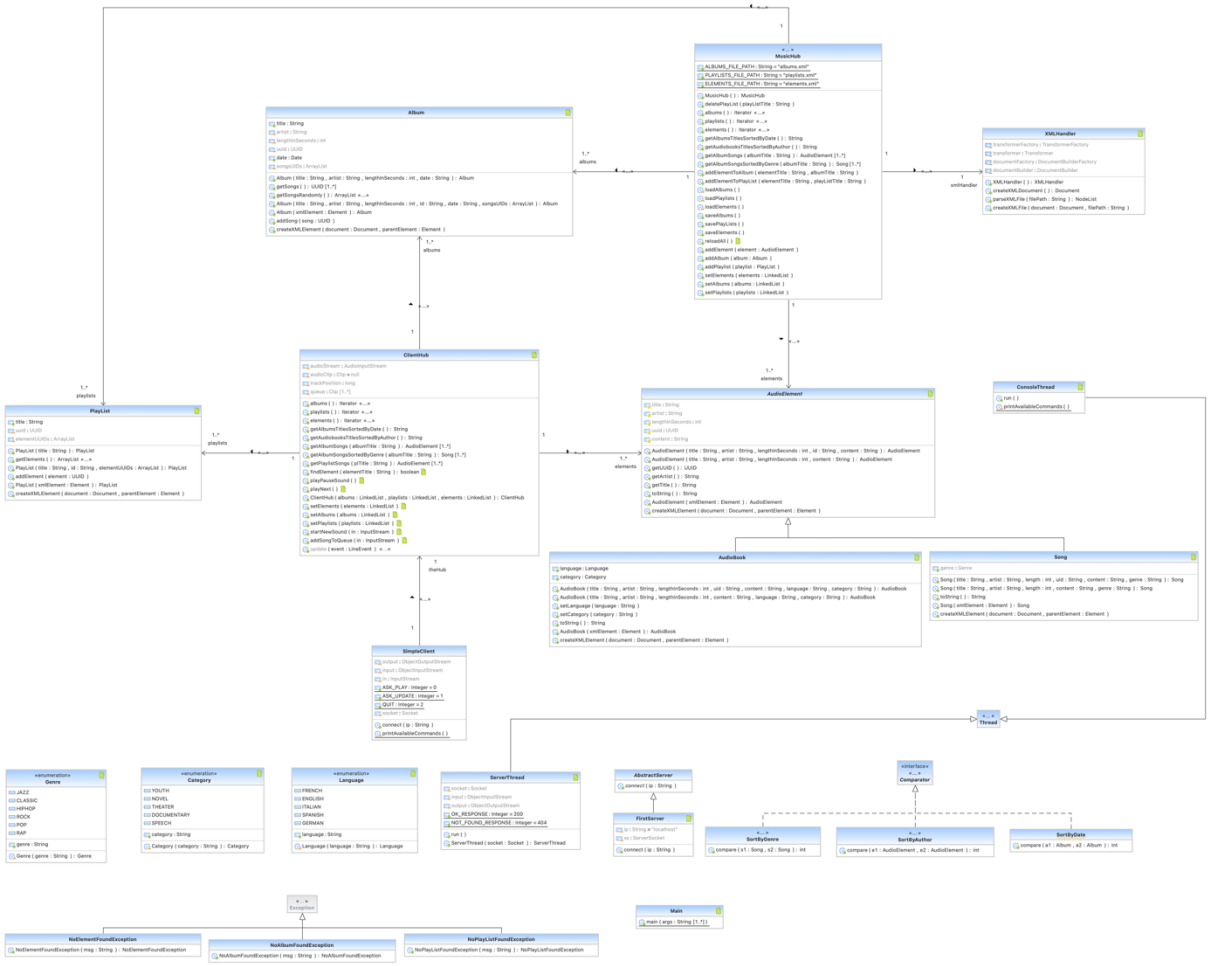
The above diagram shows how we planned to deploy our project with the example of only one client machine, where we can deploy the jar file and run it with java. It will connect the server where the files (classes, audio, and xml) should be well organized.

Actually, it is currently usable as the following diagram because we did not yet configure our pom file to create 2 separate artifacts.



Classes :

L'image est disponible sur le GitHub du projet pour pouvoir voir le diagramme en entier



- **Analyse de la conception selon les principes SOLID**

S: Notre projet implémente bien le principe de responsabilité unique. En effet les modules classes et méthodes sont responsable d'une seule fonctionnalité que le logiciel fournit. Nous avons tout d'abord les interfaces utilisateurs permettant de réceptionner les choix de des utilisateurs, puis le MusicHub qui gère la bibliothèque musicale. L'accès aux fichiers est dépendant du XMLHandler uniquement. De même pour la décomposition de la connexion entre serveur et client. Finalement, chaque fonction comme les tris, affichages, ou lectures sont assez nombreuses et précise pour ne servir qu'à leur tâche de manière indépendante et modulable. Un point ne respecte pas complètement ce principe.

O: Notre structure client server utilisant une classe abstraite, elle respecte le principe ouvert/fermé. Chaque server pourrait implémenter sa version avec par exemple une adresse différente. Cependant il s'agit de la seule classe abstraite et il n'est donc par exemple pas possible d'ajouter des fonctionnalités musicales à notre projet par la suite sans modifier la classe correspondante. On ne peut donc pas dire que ce principe soit respecté.

L: Le principe de substitution de Liskov est respecté dans notre projet car les classes mères peuvent être substituées par leurs classe filles. Par exemple la classe *SortByAuthor* et la fonction *getAudiobooksTitlesSortedByAuthor* manipulent des AudioElement. En réalité elles peuvent manipuler aussi bien des Song que des AudioBook qui sont ses classes filles sans que cela ne pose de problème au fonctionnement. De plus, ces fonctions n'utilisent que des attributs et méthodes d'AudioElement. On pourrait donc en utiliser d'autres sans problème.

I: Notre projet contenant très peu d'interface, nous respectons logiquement de principe de ségrégation des interfaces. Il n'y a en effet pas d'obligation pour un élément d'implémenter une interface inutile. Toutefois, nous pouvons constater qu'il serait intéressant d'ajouter des interfaces. Dans ce cas, il faudrait faire attention à valider ce principe en les discernant bien.

D: Le principe d'inversion des dépendances ne semble pas réellement respecté dans notre projet, en partie car nous n'implémentons quasiment pas d'interface. Par exemple, les classes SimpleClient ou ServerThread paraissent dépendantes des MusicHub. L'utilisation d'une interface aurait permis de pouvoir remplacer MusicHub par une autre classe gestionnaire de bibliothèque musicale, comme notre ClientHub par exemple, ce qui nous a obligé à ajouter de fonctions manquantes et ne nous garantit pas de compatibilité entre deux objets pourtant similaires.

Nous avons réalisé plusieurs tests unitaires qui vérifient dans l'ensemble les critères du right-BICEP. On constate qu'ils sont tous exécutables et corrects, vérifiant ainsi la première condition, corrects aux limites définies. Ils peuvent se vérifier de manière "inverse" et l'on peut faire apparaître les messages d'erreur et exceptions. Nous n'avions pas d'exigences de performances précises définies en amont mais on ne constate aucune anti-performance.

Lien vers le projet sous GitHub

Lien : https://github.com/jonthnoz/esiea-3A-jmusichub

Identifiants :

- Melissa GENOVESE : melissagenovese
- Eden LOUKAKOU-BATAILLE : Eden309
- Jonathan OZOUF : jonthnoz
- Johanne SCEMAMA : jojo0712001
- Ariitea TCHAN :

Le projet est un projet Eclipse et Maven. Un fichier pom.xml est inclus. Lors de l'exécution de la phase package avec Maven, la documentation est générée sous le dossier apidocs, les tests JUnit sont exécutés et le rapport Surefire est enregistré. Un jar nommé Projet-{version}.jar est créé. Tous les résultats sont sous le dossier target.

Le fichier jar ne permettant pas d'utiliser la fonctionnalité d'écriture puis lecture de fichier, il faut exécuter la classe Main depuis le dossier classes (musichub.main.Main) en ajoutant un argument pour lancer :

- Un simple server : server
- Un client : client
- Une interface d'administration du server : serverConsole

Les diagrammes UML sont disponibles dans le dossier complementary.

Le test

Liste des tests :

US 1: En tant que fournisseur, je veux créer des chansons, des albums ou des playlists sur le serveur afin que les clients puissent les visualiser/écouter.

- Test1 : Lorsque je crée des chansons, albums, playlists alors ils apparaissent sur la machine du client.
  Description du test :
  - On crée une nouvelle chanson sur le serveur.

  Dans le client, on affiche la liste des chansons, la chanson créée n'apparaît pas.
  Une commande est disponible pour mettre à jour le contenu. On clique “u”.
  On réaffiche la liste des chansons à l’aide de la commande “s”. On peut voir que la nouvelle chanson est dans la liste.
  - On crée un nouvel album sur le serveur.

  Dans le client, on affiche la liste des albums, l’album créé n'apparaît pas.
  Une commande est disponible pour mettre à jour le contenu. On clique “u”.
  On réaffiche la liste des albums à l’aide de la commande “a”. On peut voir que le nouvel album est dans la liste.
  - On crée une nouvelle playlist sur le serveur.

  Dans le client, on affiche la liste des playlists, la playlist créée n'apparaît pas.
  Une commande est disponible pour mettre à jour le contenu. On clique “u”.
  On réaffiche la liste des playlists à l’aide de la commande “j”. On peut voir que la nouvelle playlist est ajoutée.
  Résultat :
  Album : Succès
  Chansons : Succès
  Playlists : Succès

- Test2 : Une interface est disponible côté serveur proposant des commandes pour créer et supprimer des chansons, albums, playlists.
  Description du test :
  Sur le serveur des commandes sont disponibles pour créer chansons, albums ou playlists. Lorsque j’utilise l’une des commandes, les informations nécessaires à la création d’un album, playlist, ou chanson sont demandées. Une fois entrées, la chanson, album ou playlist est créée dans le serveur.
  Résultat:
  Album : Succès
  Chansons : Succès
  Playlists : Succès
  (voir bugs 1 & 2)

US 2: En tant qu'utilisateur je veux afficher sur le client les chansons, les albums et les playlists du serveur afin de choisir ce que je veux écouter.

- Test1 : Lorsque le client démarre alors j'ai le choix d'afficher les chansons, les albums ou les playlists.
  Description du test :
  Lorsque le client démarre, je clique "h" pour afficher les commandes disponibles.
  Sur le client des commandes sont disponibles pour afficher les listes de chansons, albums ou playlists.
  Résultat : Succès

- Test2 : Je peux sélectionner l'une de ces options et les chansons, albums ou playlists s'affichent.
  Description du test :
  Je peux afficher la liste des chansons à l'aide de la commande "s".
  Je peux afficher la liste des albums à l'aide de la commande "a".
  Je peux afficher la liste des playlists à l'aide de la commande "j".
  Résultat:
  Album : Succès
  Chansons : Succès
  Playlists : Succès

US 3: En tant qu'utilisateur je veux écouter les morceaux de musique sur le client afin de me divertir.

- Nom Test1 : Lorsque je sélectionne un morceau, alors cette musique démarre.
  Description du test :
  Sur le client je peux utiliser la commande "s" pour afficher la liste des musiques. Ensuite j'entre le nom du morceau que je veux écouter. J'ai le choix d'écouter le morceau directement en pressant "x". Lorsque je clique "x" la chanson démarre.
  Résultat: Succès

US 4: En tant qu'utilisateur je veux choisir le prochain morceau afin de l'écouter directement après le morceau en cours.

- Nom Test1 : Lorsque je choisis le prochain morceau alors il est sauvegardé en tant que prochain morceau. Lorsque le morceau est fini, alors ce morceau débute.
  Description du test : Lorsqu'un morceau est en cours. je peux choisir le morceau qui va suivre à l'aide de la commande "s" afin d'afficher le nom des chansons. J'entre le nom de la chanson que j'aimerais écouter à la suite de la musique en cours. J'utilise la commande "f" pour mettre la chanson en question dans la liste d'attente. Lorsque la musique en cours est terminée, celle en liste d'attente commence.
  Résultat : Succès

US 5: En tant qu'utilisateur je veux que les musiques puissent être triées de différentes façons afin de faciliter l'utilisation de l'application.

- Nom Test1 : Quand je clique "t" les noms des albums sont triés et affichés par date.
  Description du test : Sur le client une commande est disponible pour afficher les noms des albums par date. On clique "t". Les albums sont affichés du plus ancien au plus récent.
  Résultat : Succès

- Nom Test2 : Quand je clique "g" les noms des chansons d'un album sont triés et affichés par genre.
  Description du test : Sur le client une commande est disponible pour afficher les noms des chansons d'un album par genre. On clique "g". Les noms des albums sont affichés. J'entre le nom de l'album que je veux afficher. Les chansons de celui-ci sont triées par genre.
  Résultat : Succès

- Nom Test3 : J'ai le choix d'afficher les noms des albums triés par date ou les chansons d'un album triées par genre.
  Description du test : Sur le client les commandes "g" et "t" sont disponibles pour afficher les noms des albums triés par date ou les chansons d'un album triées par genre.
  Résultat: Succès

US 6: En tant qu'utilisateur je veux pouvoir mettre en pause un morceau afin de pouvoir choisir les moments d'écoute.

- Nom Test1 : Lorsqu'une musique est en cours, une commande est disponible pour arrêter la lecture et la reprendre.
  Description du test : Sur le client la commande "p" est disponible pour arrêter et reprendre la lecture. Lorsqu'une musique est en cours je clique sur "p", elle est en pause. Je re-clique sur "p", la musique reprend.
  Résultat : Succès

US 7: En tant qu'utilisateur je veux passer à la musique suivante sans terminer l'écoute de la musique en cours afin d'écouter ce que je souhaite.

- Nom Test1 : Lorsqu'une musique est en cours, une commande est disponible pour écouter la musique suivante.
  Description du test : Sur le client la commande "n" est disponible pour écouter la musique suivante dans la liste d'attente.
  Résultat : Succès

- Nom Test2 : Lorsque j'entre "n" alors la musique en cours est arrêtée et une autre musique commence automatiquement.
  Description du test : Lorsque l'écoute d'une musique est en cours, si une ou plusieurs musiques sont dans la liste d'attente, je clique "n", la première musique sur la liste d'attente commence.
  Lorsqu'aucune musique est en cours, si je clique "n", rien ne se passe, le client affiche le menu des commandes.
  Résultat : Succès (voir bug 6)

CS 1: Au minimum un client peut accéder au service via le serveur.

Observation de ports, processus, et du simple comportement du programme
Description du test : Lorsqu'on démarre un client il se connecte correctement sur le port 6666.
Résultat : Plusieurs connexions simultanées sont possibles. Succès

CS 2: Mise à jour automatique quand du nouveau contenu est disponible sur le serveur

Lorsqu'on modifie le contenu depuis la console server les modifications sont immédiatement effectives (si un client demande une mise à jour il verra les modifications). Si le fichier xml est modifié de manières externes les changements de sont pas détectés mais comme les fichiers sont recharger à la demande de mise à jour les changements seront effectifs pour le client.
Succès

CS 3: Mise à jour manuelle des données du côté client.

Test similaire à ceux des user stories : une fonction est disponible et elle fonctionne (voir US 1)
Succès

Liste des bugs :

ID	BG_1
Description	Lors de la création d'un album, si la date de celui-ci n'est pas entrée le programme s'arrête.
Tracés vers	REQ_1
Comment reproduire	Créer un album sur le serveur et ne pas entrer de date de création lors de la requête.
Etat	New
Trouvé dans	V.1.0.0
Sévérité	S1
Priorité	P3
Analyse	
Corrigé dans	
Trouvé par	Johanne

ID	BG_2
Description	Lors de la création d'un album, si la durée de celui-ci entrée est sous forme alphabétique le programme s'arrête avec une erreur.
Tracés vers	REQ_1
Comment reproduire	Créer un album sur le serveur et entrer des lettres pour la durée.
Etat	New
Trouvé dans	V.1.0.0
Sévérité	S1
Priorité	P3
Analyse	
Corrigé dans	
Trouvé par	Johanne

ID	BG_3
Description	Sur le client, lorsqu'on entre une commande qui n'est pas dans le menu, le programme s'arrête.
Tracés vers	
Comment reproduire	Entre une lettre qui ne correspond à aucune des commandes du menu.
Etat	New
Trouvé dans	V.1.0.0
Sévérité	S1
Priorité	P3
Analyse	
Corrigé dans	V.1.0.1
Trouvé par	Johanne

ID	BG_4
Description	Sur le client, lorsqu'on affiche la liste des chansons disponibles et tape le nom d'une chanson pour l'écouter. Si on entre quelque chose différent de "x" et "f". Le client commence la chanson sélectionnée instantanément.
Tracés vers	REQ_4
Comment reproduire	Dans le menu du client, cliquer "s", entrer le nom d'une chanson, entrer une lettre différente de "f" et "x"
Etat	New
Trouvé dans	V.1.0.0
Sévérité	S4
Priorité	P4
Analyse	
Corrigé dans	
Trouvé par	Johanne

ID	BG_5
Description	Lors du lancement du client, entrer h est nécessaire pour afficher les commandes disponibles. Si quelque chose autre que "h" est entré alors le programme est bloqué.
Tracés vers	
Comment reproduire	Au tout début du programme client, taper autre chose que "h".
Etat	New
Trouvé dans	V.1.0.0
Sévérité	S1
Priorité	P3
Analyse	
Corrigé dans	V 1.0.1
Trouvé par	Johanne

ID	BG_6
Description	Lorsqu'une musique est en cours, si aucun morceau est sur la liste d'attente, quand on presse "n" rien ne se passe. Le morceau devrait se terminer.
Tracés vers	REQ_11
Comment reproduire	Lancer une musique et cliquer sur "n".
Etat	New
Trouvé dans	V.1.0.0
Sévérité	S3
Priorité	P4
Analyse	
Corrigé dans	
Trouvé par	Johanne