Génie Logiciel : Projet 2021

2 MAI

FERNANDES Mickaël
RABII Ahmad
NOUARI Achraf

esiea
INGÉNIEUR·E·S D'UN NUMÉRIQUE UTILE

Table des matières

I. Liste des membres : Rôles et description

a) Réalisations

NOUARI Achraf : Rédaction du rapport, liste des exigences fonctionnelles, user stories et critères d'acceptance, diagramme de package, BICEP, réalisation de l'interface utilisateur et implémentation de méthodes dans la classe RequestManager.

RABII Ahmad : Liste des exigences non-fonctionnelles, diagramme de séquence, diagramme de classes, analyse principes SOLID, liste des tests d'acceptance, JavaDOC, ajout des fonctionnalités (charger les éléments, annuler les changements…).

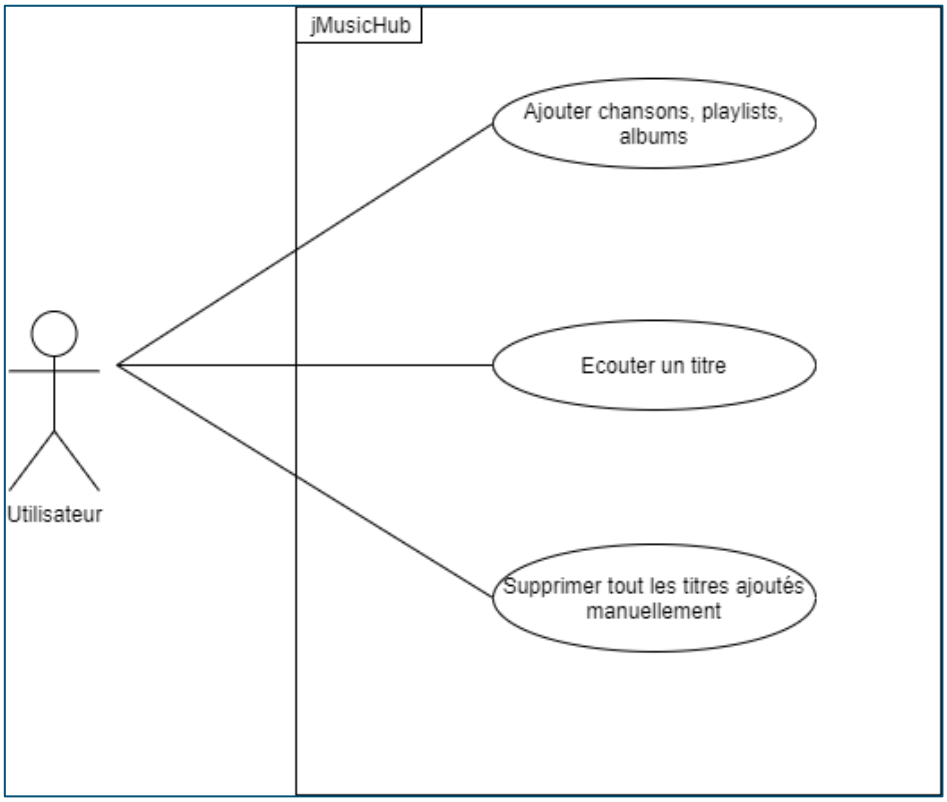
FERNANDES Mickael : Diagramme des cas d'utilisation, constraint stories, diagramme de composants et déploiement, liste des bugs trouvés, réalisation de l'architecture client-serveur ainsi que la lecture des fichiers audio.

II. Liste des exigences fonctionnelles et non fonctionnelles

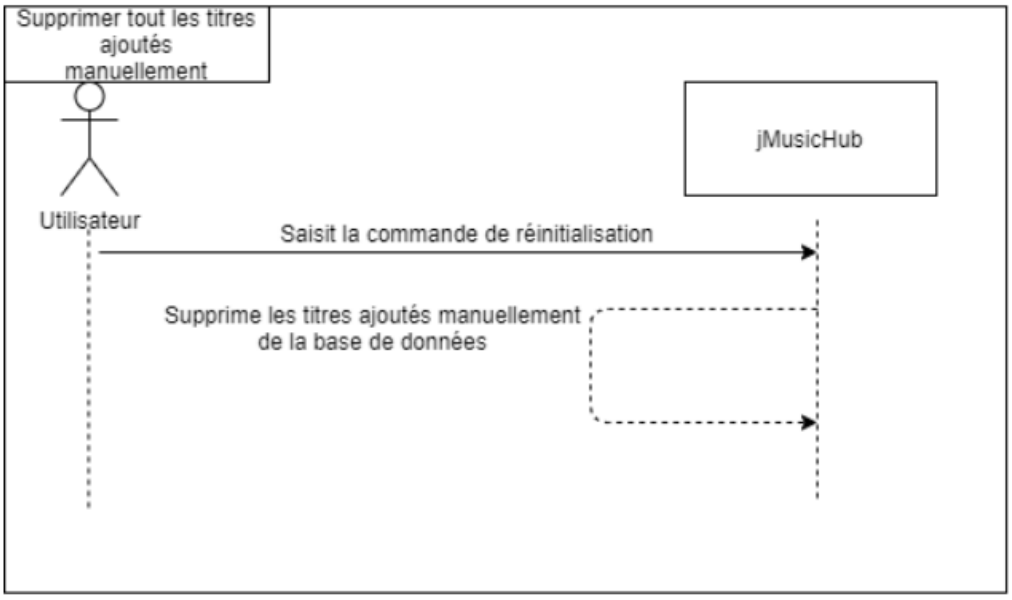
ID	Nom	Type	Description
JMUSIC_REQ_1	Fonctionnalité de base	F	Au minimum un client doit pouvoir se connecter au serveur
JMUSIC_REQ_2	Ajout	F	Le client pourra ajouter des chansons, albums ou playlists sur le serveur
JMUSIC_REQ_3	Affichage	F	Le client aura la possibilité de visualiser le contenu du serveur (chansons, albums, playlists)
JMUSIC_REQ_4	Ecoute	F	Le client aura la possibilité d'écouter les morceaux de musique
JMUSIC_REQ_5	Serveur	F	Le serveur sera mis à jour automatiquement lorsqu'un élément y est ajouté
JMUSIC_REQ_6	Mise à jour	F	Le client peut accéder aux mises à jour du serveur à tout moment.
JMUSIC_REQ_7	Gestion des erreurs	F	L'application disposera d'un système de journalisation des erreurs (avertissements, fichier lu,...) avec horodatage
JMUSIC_REQ_8	(Press to kill)	NF	Le client à la possibilité d'appuyer sur une touche pour quitter l'application (en cas de bugs,...)
JMUSIC_REQ_9	Réinitialisation	NF	On pourra supprimer tout les éléments ajoutés par l'utilisateur en appuyant sur un bouton.
JMUSIC_REQ_10	Unicité	NF	L'application est accessible que pour un client à la fois

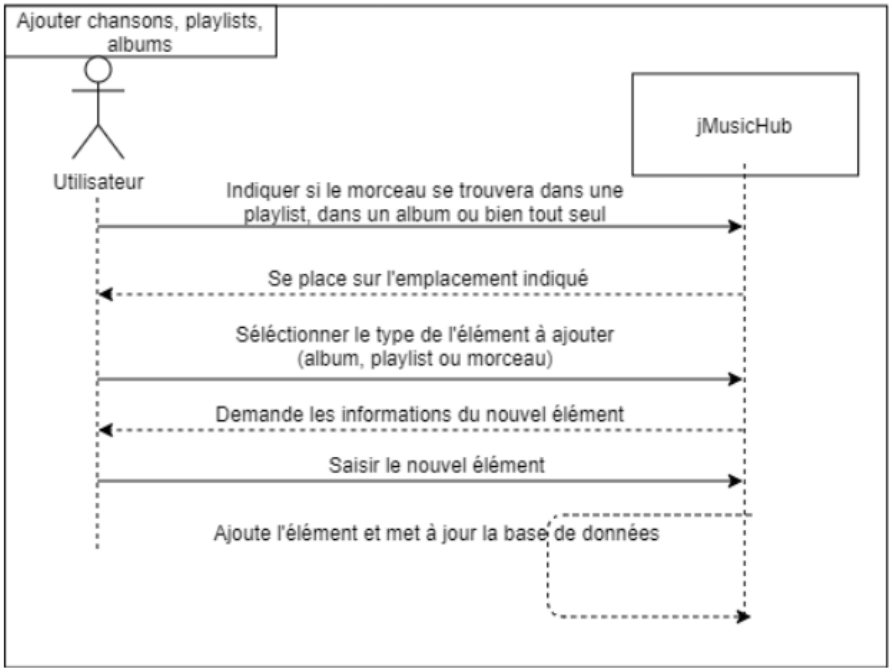
III. Diagramme UML des cas d'utilisation et de séquence

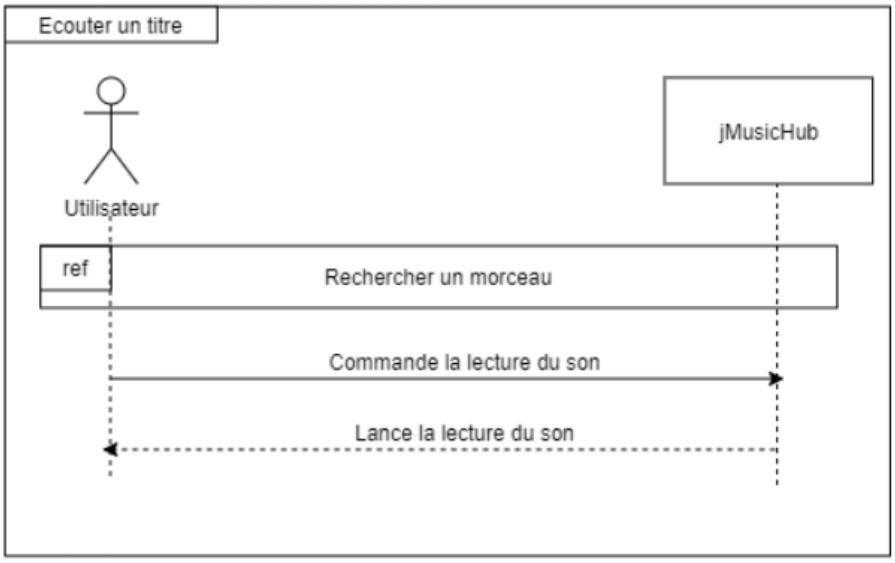
a) Diagramme UML des cas d'utilisations



a) Diagramme UML de séquence




Ajouter chansons, playlists, albums
Utilisateur
jMusicHub
Indiquer si le morceau se trouvera dans une playlist, dans un album ou bien tout seul
Se place sur l'emplacement indiqué
Séléctionner le type de l'élément à ajouter (album, playlist ou morceau)
Demande les informations du nouvel élément
Saisir le nouvel élément
Ajoute l'élément et met à jour la base de données

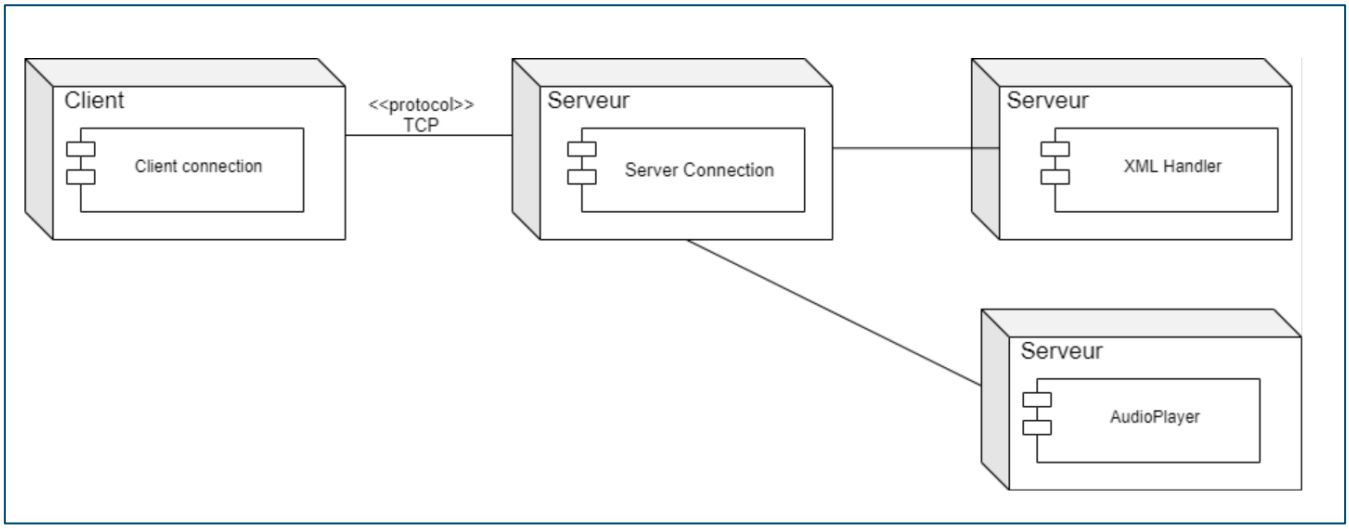

Ecouter un titre
Utilisateur
jMusicHub
ref
Rechercher un morceau
Commande la lecture du son
Lance la lecture du son

IV. Diagramme UML de package, composants, déploiement, classes

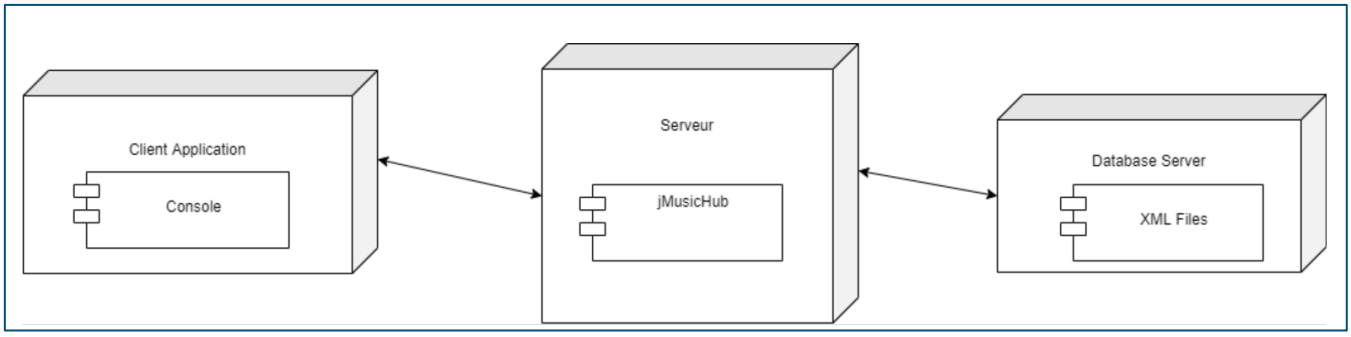
a) Diagramme UML de classes

Voir l'image UML Classe.png dans le fichier .zip

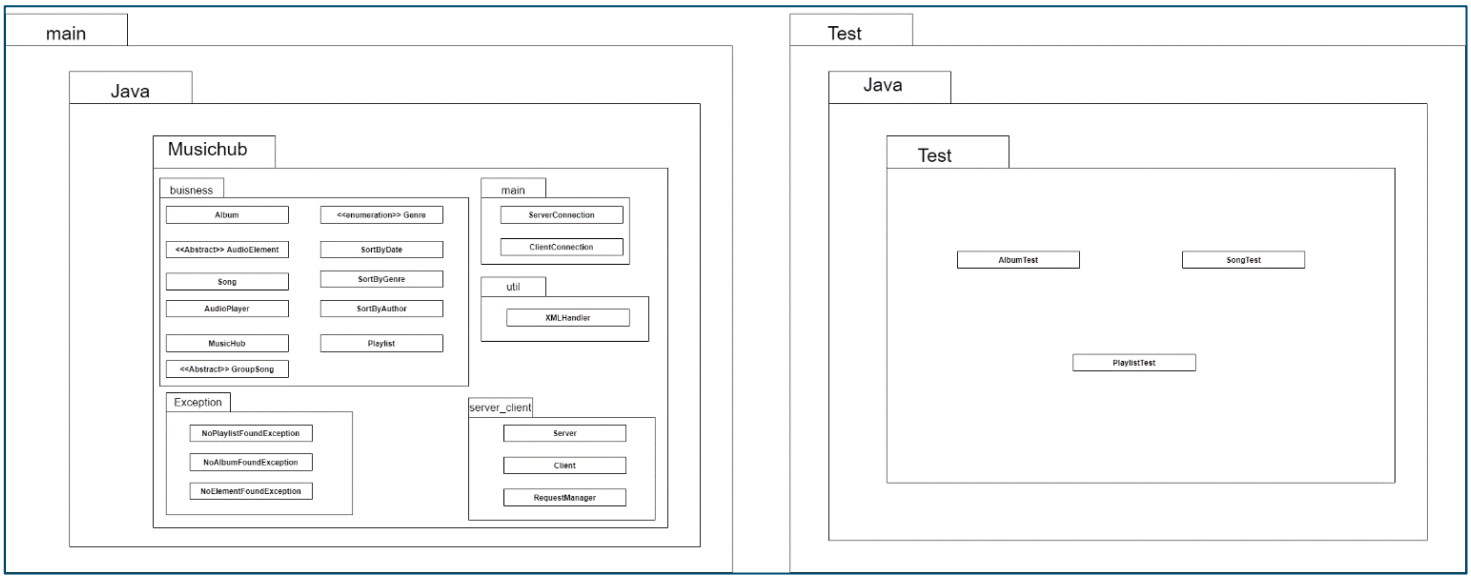
b) Diagramme UML de composants



c) Diagramme UML de déploiement



d) Diagramme UML de package



V. Analyse de la conception selon les principes SOLID

a. Responsabilité unique

« 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 »

Nous pensons que le critère de responsabilité unique est respecté puisque nous avons découpé le projet en fonctionnalités que nous avons implémenté par classe. Nous avons plusieurs éléments musicaux : albums, playlists, morceaux et ils sont chacun gérés par une classe séparée. Lorsqu'on ajoute des morceaux, playlists ou albums au serveur, les méthodes sont respectivement dans les classes de leur type, encapsulés.
De même pour les interactions entre le client et le serveur, les méthodes sont encapsulées par un RequestManager afin de permettre l'envoi des commandes du client au serveur.

⇨ **Critère respecté**

b. Ouvert/Fermé

« Toutes les entités logicielles doivent être ouvertes à l'extension, mais fermées à la modification »

Nous avons défini des classes et méthodes publiques qui peuvent hériter d'autres fonctionnalités mais les attributs sont privés pour ne pas perturber le fonctionnement de notre logiciel.

⇨ **Critère respecté**

c. Substitution de Liskov

« 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 »

Nous n'avons ajouté à l'exemple d'implémentation jMusicHub fourni qu'une seule classe abstraite, nous avons créé une classe abstraite Groupsong regroupant la classe album et playlist, le principe de substitution de Liskov est respecté car une instance de Groupsong peut être remplacé par une instance de playlist ou d'album.

⇨ **Critère respecté**

d. Ségrégation des interfaces

« *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 »*

Pour ce critère, le logiciel IntelliJ gère cela en nous proposant de supprimer les interfaces qui ne sont pas utilisés. Par ailleurs nous n'avons eu recours que très rarement à l'implémentation d'une interface.

⇨ **Critère respecté**

e. Inversion des dépendances

« *Les entités doivent dépendre uniquement des abstractions »*

Notre classe Song comme peut le témoigner le diagramme UML de classes ne dépend que de l'abstraction AudioElement.
Les entités Albums et Playlists aussi dépendent de l'abstraction GroupSong.

⇨ **Critère respecté**

VI. Lien vers GitHub

Lien du projet
https://github.com/MickaelFrds/Projet_MusicHub_GL

Identifiants :
Fernandes Mickaël : MickaelFrds
Rabii Ahmad : a-r-z
Nouari Achraf : nouari1

VII. Critères right-BICEP utilisés

Les critères right-BICEP que nous avons utilisés sont les suivants :

- **Right : Nous avons ajouté des playlists, albums ainsi que des morceaux puis nous avons vérifié qu'ils étaient bien ajoutés à la base de données du serveur.**

- **B : Nous avons tenté de créer des playlists avec un nom vide, un message d'erreur s'est affiché. Même chose lorsqu'on souhaite supprimer une playlist sans nom. Nous avons réalisé le même test pour album et morceaux.**

- **I : Pour vérifier les relations inverses, nous avons ajoutés puis supprimés à tour de rôle une playlist, un album et un morceau et nous avons consulté le contenu de la base de données pour vérifier qu'ils ont bien été supprimés.**

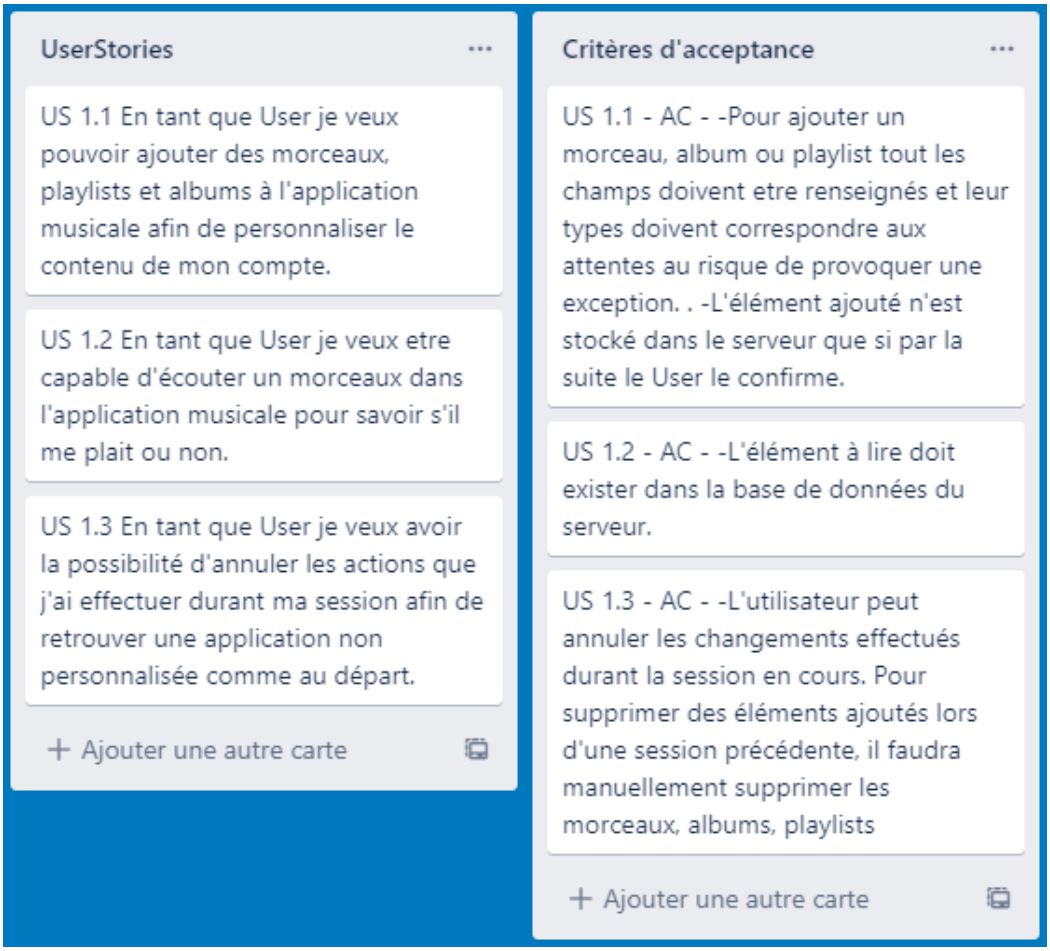
- **C : Afin de vérifier qu'une playlist/album/morceaux a été ajouté(e), il faut s'assurer qu'aucun message d'erreur n'apparaît lors de l'exécution. Le moyen de faire une vérification est d'afficher la liste des playlists/albums/morceaux**

- **E : Pour forcer les apparitions des conditions d'erreur nous avons supprimé des playlists qui n'existaient pas, le message « Playlist <nom_playlist> not found ». Il en est de même pour album et pour les morceaux.**

- **P : Nous n'avons pas pris en compte ce critère de performance dans notre conception.**

VIII. Tests d’acceptances et bugs trouvés

a) Tests d’acceptances



Test n°1 : US 1.1 :

Description : On renseigne un champ vide lors de la création d’un morceau, album, playlist.

Résultat : L’application redemande au Client de renseigner le champ vide

⇨ **Réussi**

Test n°2 US 1.2 :

Description : On commande la lecture d’un élément audio qui n’existe pas.

Résultat : L’application redemande au Client un nom de fichier audio qui existe.

⇨ **Réussi**

Test n°2 US 1.3:

Description : On ajoute un morceau, album, playlist on le charge dans la base de données puis on commande au serveur d'annuler cette modification.
On se charge d'effacer manuellement une playlist crée auparavant.

Résultat : Les données sont bien supprimées, si on renseigne un champ vide on tombe bien sur une exception qui nous demande de saisir un champ non vide

⇨ **Réussi**

b) Bugs trouvés

Bugs rencontrés	Sévérité/Priorité
Lors de la suppression d'une playlist, on peut avoir un message d'erreur qui survient rarement. Il disparait en redémarrant le logiciel.	**Sévérité : Faible** **Priorité : Moyenne**
Lors de la création d'un album et morceau, quand on renseigne une chaine de caractères au lieu d'un entier pour le champ « Lenght » : l'application nous demande de saisir un entier mais peut importe le type de cette donnée que l'on ressaisit, il y'a un message d'erreur.	**Sévérité : Faible** **Priorité : Moyenne**