FACULTÉ DES SCIENCES
DÉPARTEMENT D'INFORMATIQUE

UNIVERSITÉ LIBRE DE BRUXELLES ULB

INFO-F-307: Génie Logiciel et gestion de projets

2024-2025 – Deezify

Frédéric Pluquet **Hugo Callebaut** **Anthony Cnudde**
Martin Colot **Abel Laval** **Yannick Molinghen**

1 Introduction

Pour ce projet, vous allez développer une application pour la gestion de projets exécutable sous *Linux*, *Windows* et *MacOS*. Pour ce faire, vous emploierez le langage de programmation Java.

Le présent document est composé de deux parties. La première est technique et elle a pour but de vous expliquer la structure du dépôt utilisé pour le projet et de vous aider à configurer l'accès à ce dépôt. La seconde partie reprend les histoires que nous vous demandons d'implémenter lors des itérations.

2 Détails techniques

2.1 Logiciels

IDE

Nous vous suggérons d'utiliser IntelliJ qui offre une expérience de développement Java complète. Vous pouvez cependant utiliser l'environnement de développement que vous désirez.

Java

Vous devez utiliser une version de Java ≥ 18. Veillez à indiquer dans le fichier `README.md` la version minimale à utiliser pour votre projet.

Lorsque vous installez Java, veillez à installer la version JDK (Java Development Kit) qui contient tout le nécessaire pour développer. Si vous installez uniquement le JRE (Java Runtime Environment, vous pourrez lancer des programmes java mais pas les compiler.

Maven

Maven est un système de build qui permet de compiler et de tester des projets écrits en Java. Le projet de base que nous vous fournissons est prévu pour utiliser Maven

Maven permet de

- Tester votre code avec `mvn test`.
- Lancer votre application avec `mvn exec:java`.
- Compiler votre projet avec `mvn clean install`. Vous pouvez ensuite lancer votre projet compilé avec `java -jar target/<nom_du_fichier.jar>`.

La configuration de Maven se fait dans le fichier `pom.xml`.

2.2 Structure du dépôt

Chaque groupe aura accès à son propre dépôt git sur le GitLab de l'ULB, disponible à l'adresse: `https://gitlab.ulb.be/ulb-infof307/2025-groupe-xx.git`.

Voici la structure initiale et la signification des dossiers :

- **src/main** dossier source qui va contenir le code source de votre application sous le package `ulb`. La classe principale qui lancera votre application est la classe `Main` de ce même package. Vous pouvez bien sûr ajouter des sous-packages, des classes et modifier l'implémentation de cette classe `Main`.
- **src/test** est également un dossier source contenant tous vos tests unitaires. Les classes de tests doivent être nommées selon la convention `TestClass` et implémenter les tests unitaires et/ou fonctionnels de la classe `Class`.
- **team** contient la version courante des documents que vous utiliserez pour la gestion de votre équipe pendant le projet, c'est-à-dire uniquement les fichiers .md, .tex, .xls, .ods et .pdf contenant les histoires actualisées (contenu et points), les comptes-rendus des réunions avec le client et avec l'assistant, la répartition des tâches, les burn down charts et tout autre diagramme que vous auriez employé dans la gestion de votre projet.

2.3 Intégration continue

GitLab permet d'automatiser certaines tâches lorsqu'une action en particulier a lieu. Cette configuration se fait dans le fichier `.gitlab-ci.yml`. Par défaut, nous avons configuré ce fichier pour que Maven lance les tests, quelle que soit la branche sur laquelle le `push` a été réalisé. N'hésitez pas à compléter ce document.

Consultez cette page pour configurer GitLab de sorte que vous receviez une notification par e-mail lorsqu'un push a causé une erreur.

2.4 Releases

À la fin de chaque itération, vous devrez fournir un fichier `.jar` exécutable dans une release GitLab. Pour ce faire, créez un tag (par exemple `itération-1`) sur la branche `main` puis créez une release GitLab en y joignant votre fichier compilé. Nous vous encourageons à automatiser le processus via l'intégration continue, comme décrit dans la documentation gitlab.

3 Situation et objectifs

À l'ère du numérique, la musique est devenue une partie intégrante de notre quotidien. Que ce soit pour accompagner nos trajets, animer nos soirées ou simplement se détendre, la musique est un langage universel qui transcende les frontières et les générations. Avec l'explosion des plateformes de streaming et des bibliothèques musicales personnelles, les utilisateurs cherchent de plus en plus à personnaliser leur expérience d'écoute, en fonction de leurs goûts, de leurs humeurs et de leurs besoins.

Dans ce contexte, vous allez développer un lecteur de musique moderne et polyvalent, capable de s'adapter aux attentes variées des utilisateurs. Ce projet vise à offrir une application intuitive et riche en fonctionnalités, permettant aux utilisateurs de gérer, organiser et écouter leur musique de manière fluide et agréable. Que ce soit pour créer des playlists, explorer des morceaux par tags, ou simplement profiter d'une écoute immersive avec des effets audio personnalisés, votre application devra répondre à une multitude de besoins.

L'innovation de ce projet réside dans sa capacité à combiner simplicité d'utilisation et fonctionnalités avancées. En intégrant des éléments tels que la gestion des tags, les suggestions de morceaux similaires, ou encore la possibilité de synchroniser des paroles pour le karaoké, vous offrirez aux utilisateurs une expérience unique et personnalisée. De plus, l'application devra être capable de s'adapter à différents environnements, que ce soit pour une écoute domestique ou pour animer une soirée entre amis.

L'objectif principal est de créer une application qui non seulement répond aux besoins techniques et fonctionnels, mais qui également inspire et accompagne les utilisateurs dans leur découverte musicale. En unifiant des fonctionnalités variées en une seule plateforme, vous favoriserez l'adoption de cette application par un public large et diversifié.

4 Histoires

Histoire 1 : Lecture de morceaux

L'utilisateur peut naviguer parmi les morceaux présent sur son ordinateurs (fichiers .mp3, .flak, ...) enregistrés dans un dossier spécifique. Il peut sélectionner un morceau pour l'écouter. Par ailleurs, l'utilisateur peut également consulter la durée totale du morceau ainsi que le temps déjà écoulé depuis le début du morceau.

Priorité Client: **1**
Risque développeurs: 1 2 3
Introduit dans l'itération:
État:

Histoire 2 : File d'attente

L'utilisateur peut ajouter des morceaux à la file d'attente, c'est-à-dire la liste des morceaux à lire après le morceau en cours (s'il y en a un). Lorsque le morceau en cours se termine, le prochain morceau dans la file d'attente commence automatiquement. Si la file d'attente est vide, la lecture se termine. L'utilisateur peut supprimer des morceaux de la file d'attente ainsi que l'effacer entièrement.

Priorité Client: **1**
Risque développeurs: 1 2 3
Introduit dans l'itération:
État:

Histoire 3 : Boutons de contrôle

Lors de la lecture d'un morceau, un curseur affiche l'avancement de la lecture du morceau et permet de naviguer vers n'importe quel moment du morceau, des boutons permettent d'interagir avec la lecture (pause, suivant, précédent, ...), et un curseur permet de contrôler le volume.

Priorité Client: **1**
Risque développeurs: 1 2 3
Introduit dans l'itération:
État:

Histoire 4 : Modification des données des morceaux

L'utilisateur doit être capable de modifier les méta-données des morceaux telles que les artistes, le nom du morceau, le nom de l'album.

Priorité Client: **1**
Risque développeurs: 1 2 3
Introduit dans l'itération:
État:

Histoire 5 : Autocomplétion des données

Lorsque l'utilisateur modifie les méta-données d'un morceau une autocomplétion des données doit être proposée pour les artistes, albums et tags déjà existants.

Priorité Client: **2**
Risque développeurs: 1 2 3
Introduit dans l'itération:
État:

Histoire 6 : Ajouter des tags aux morceaux

Depuis le menu de navigation, l'utilisateur peut ajouter des tags aux morceaux présents sur son ordinateur. Ces tags sont visibles à côté du nom du morceau dans le menu de navigation. Différents types de tags prédéfinis par l'application sont disponibles pour représenter le style de musique, des émotions ou des événements. L'utilisateur peut aussi ajouter ses propres tags.

Priorité Client: **2**
Risque développeurs: 1 2 3
Introduit dans l'itération:
État:

Histoire 7 : Recherche

Dans le menu de navigation, l'utilisateur peut rechercher un morceau en écrivant son titre, le nom de l'artiste, le nom de l'album ou des tags attribués.

Priorité Client: **1**
Risque développeurs: 1 2 3
Introduit dans l'itération:
État:

Histoire 8 : Listes de lecture

L'utilisateur peut créer des listes de lecture avec un titre, et optionnellement une image. L'utilisateur peut ajouter des morceaux à la liste de lecture et en modifier l'ordre. L'utilisateur peut explorer ses listes de lecture et en lancer la lecture en choisissant soit de remplacer immédiatement la file d'attente par la liste de lecture soit d'ajouter tous les morceaux à la fin de la file d'attente.

Priorité Client: **1**
Risque développeurs: 1 2 3
Introduit dans l'itération:
État:

Histoire 9 : Suggestion de morceaux

Lorsque l'utilisateur crée une liste de lecture, le programme lui suggère des morceaux similaires à ceux déjà présents en fonction des artistes, des tags et des albums.

Priorité Client: **3**
Risque développeurs: 1 2 3
Introduit dans l'itération:
État:

Histoire 10 : Mode « DJ »

Pendant la lecture d'un morceau, l'utilisateur peut ajouter des effets qui modifient le son (par exemple : effet cathédrale, effet « gain », ...)

Priorité Client: **2**
Risque développeurs: 1 2 3
Introduit dans l'itération:
État:

Histoire 11 : Égaliseur

L'utilisateur peut modifier l'équilibrage des fréquences de sorte à augmenter les basses, les aigus, ... dans une amplitude de −20dB à +20 dB à l'aide de curseurs.

Priorité Client: **3**
Risque développeurs: 1 2 3
Introduit dans l'itération:
État:

Histoire 12 : Zôlis dessins

Pendant la lecture d'un morceau, l'utilisateur peut choisir d'afficher un visualiseur graphique du son. Une animation générée aléatoirement et synchronisée avec le rythme de la musique doit alors être affichée.

Priorité Client: **2**
Risque développeurs: 1 2 3
Introduit dans l'itération:
État:

Histoire 13 : Paroles

L'utilisateur peut importer des paroles pour chaque morceau. Lors de la lecture d'un morceau, l'utilisateur peut en consulter les paroles.

Priorité Client: **3**
Risque développeurs: 1 2 3
Introduit dans l'itération:
État:

Histoire 14 : Mode Karaoke

L'utilisateur peut importer les paroles synchronisées d'un morceau. Ensuite, lors de la lecture d'un morceau pour lequel des des paroles synchronisées ont été importées, l'utilisateur peut voir défiler les paroles en accord avec la musique.

Priorité Client: **3**
Risque développeurs: 1 2 3
Introduit dans l'itération:
État:

Histoire 15 : Transition

L'utilisateur peut cocher une option pour faire une transition entre deux morceaux de sorte que le son du morceau se terminant diminue progressivement pendant que le son du morceau suivant augmente. Lorsque l'option est activée, l'utilisateur peut choisir la durée de la transition à l'aide d'un curseur.

Priorité Client: **3**
Risque développeurs: 1 2 3
Introduit dans l'itération:
État:

Histoire 16 : Contrôles divers

Des options plus avancées sont ajoutées : vitesse de lecture d'un morceau, musique aléatoire, balance automatique des différents canaux audio.

Priorité Client: **1**
Risque développeurs: 1 2 3
Introduit dans l'itération:
État:

Histoire 17 : Radio

L'utilisateur peut ajouter un flux de web radio dans sa bibliothèque. L'utilisateur peut écouter une web-radio de la bibliothèque. La lecture d'une radio ne se termine jamais.

Priorité Client: **2**
Risque développeurs: 1 2 3
Introduit dans l'itération:
État:

Histoire 18 : Multilinguisme

L'utilisateur peut choisir la langue de l'application dans un menu d'options. Les trois langues disponibles sont le français, le néerlandais et l'anglais.

Priorité Client: **2**
Risque développeurs: 1 2 3
Introduit dans l'itération:
État:

Histoire 19 : Récupérer les pochettes d'album

L'utilisateur peut ajouter des images (pochettes d'albums) qui sont affichées à coté du nom des morceaux dans les menus de navigation et en grand lors de la lecture d'un morceau si rien d'autre n'est affiché.

Priorité Client: **2**
Risque développeurs: 1 2 3
Introduit dans l'itération:
État:

Histoire 20 : Utiliser une vidéo comme « pochette »

L'utilisateur peut définir une vidéo (au format MP4) associée à un morceau. Le cas échéant, cette vidéo est jouée (sans son) en même temps que le morceau, à la place de la pochette.

Priorité Client: **3**
Risque développeurs: 1 2 3
Introduit dans l'itération:
État:

Histoire 21 : Favoris

Une playlist particulière, appelée « Favoris », doit exister par défaut dans l'application. L'utilisateur peut y ajouter des morceaux facilement.

Priorité Client: **2**
Risque développeurs: 1 2 3
Introduit dans l'itération:
État:

Histoire 22 : Compte utilisateur

Plusieurs utilisateurs peuvent créer un compte (local) sur l'application. On a alors besoin de plusieurs dossiers pour stocker les morceaux : un dossier global, accessible par tous les utilisateurs, ainsi qu'un dossier particulier par utilisateur. Chaque utilisateur a également accès à ses propres playlists.

Un menu d'utilisateur permet de passer d'un utilisateur à un autre, et de créer des nouveaux utilisateurs. Aucun mot de passe n'est nécessaire.

Priorité Client: **3**
Risque développeurs: 1 2 3
Introduit dans l'itération:
État: