

Interlude

Rapport TPI et documentation technique



Travail personnel individuel (TPI)

Nicolas Ettlin — Mai 2019

Maîtresse d'apprentissage : Jasmina Travnjak

Table des matières

Table des versions.....	4
Introduction	4
Rappel de l'énoncé.....	4
Organisation.....	4
Livrables	4
Matériel et logiciels à disposition.....	5
Description de l'application.....	5
Méthodologie.....	7
Planification	9
Product backlog.....	9
Diagramme de Gantt	14
Généralités concernant l'implémentation.....	16
Base de données	16
Dictionnaire de données.....	16
Structure du projet.....	18
Classes (PHP).....	19
API interne	19
Composants Vue.js	20
Exemples d'utilisation des composants	21
Librairies et outils externes.....	22
Vue.js	22
Axios	22
NPM et Composer	23
Laravel Mix	23
SASS	24
ESLint.....	24
getID3()	25
TwitterOAuth.....	25
PHP dotenv	25
Symfony VarDumper	26
Git	26
Analyse des fonctionnalités majeures.....	27
Connexion avec Twitter	27

Affichage de la bibliothèque musicale	28
Lecteur musical	28
Importation de fichiers audio	28
Recherche de musique	29
Modification de la bibliothèque musicale	30
Plan de test et tests	31
Périmètre des tests.....	31
Équipement de test.....	31
Environnement	31
Fichiers de test.....	31
Scénarios de test.....	32
Évolution des tests.....	44
Conclusion.....	45
Difficultés rencontrées.....	45
Variantes de solutions et choix	45
Améliorations possibles.....	46
Architecture technique	46
Fonctionnalités.....	46
Bilan personnel	47
Remerciements.....	47
Annexes	48
Bibliographie.....	49
Glossaire.....	50
Termes métier	50
Termes techniques	50
Résumé du TPI	
Énoncé	
Journal de bord	
Code source	

Table des versions

Nº de version	Date	Auteur	Changements apportés
1.0	23.05.2019	Nicolas Ettlin <a href="mailto:<nicolas.ettlin@me.com>"><nicolas.ettlin@me.com>	Version finale du document pour le rendu du TPI

Introduction

Ce document est un rapport présentant différents aspects de la conception du projet Interlude. Ce projet a été réalisé dans le cadre du *Travail pratique individuel* (TPI) durant la session de mai 2019. Il a pour but de valider mes compétences acquises pendant la formation *Informaticien CFC* dispensée à l'école d'informatique du CFPT au Petit-Lancy.

Interlude est une application web qui permet aux utilisateurs d'écouter leur musique depuis un navigateur web. Pour ce faire, l'utilisateur peut utiliser diverses fonctionnalités telles que l'importation de musique avec extraction automatique des métadonnées et l'authentification au moyen d'un compte Twitter.

Rappel de l'énoncé

Les informations suivantes sont extraites du cahier de charges du TPI.

Organisation

Élève	Maîtresse d'apprentissage
Nicolas Ettlin <a href="mailto:<nicolas.ettlin@me.com>"><nicolas.ettlin@me.com>	Jasmina Travnjak <a href="mailto:<edu-travnjakj@eduge.ch>"><edu-travnjakj@eduge.ch>
Experts	
Francesco Foti <a href="mailto:<francesco.foti@devinfo.net>"><francesco.foti@devinfo.net>	Carol Quarroz <a href="mailto:<cquarroz@gmail.com>"><cquarroz@gmail.com>

Livrables

Pour les experts et la formatrice par e-mail :

- Planning détaillé du projet
- Rapport du projet contenant le code source au format PDF
- Journal de bord
- Résumé du TPI (1 page A4)

Pour la formatrice uniquement :

- L'accès au repository distant du projet avec les droits de « clone »
- Un readme explicitant l'installation du projet en local
- Un dump de la base de données contenant la structure ainsi qu'un set de données de test

Matériel et logiciels à disposition

- Un PC standard école avec Windows 10, 2 écrans
- Serveur Web et SGBD à choix (EasyPHP, Wamp, Laragon, Adminr, phpMyAdmin, autre)
- IDE à choix (NetBeans, Visual Studio Code, PHPStorm, autre)
- Logiciel de création de schémas (Visio, Gliffy, autre)
- Outil de versionnage de code (Git, avec dépôt distant sur Github / Bitbucket / GitLab)
- Navigateur web (Mozilla Firefox/Google Chrome)
- Logiciel de création de maquettes d'interfaces utilisateur (Sketch, Adobe XD, Proto.io, autre)
- Outil bureautique à choix pour les documents (Google Docs, MSOffice, OpenOffice)
- L'étudiant est autorisé à utiliser son matériel personnel au besoin

Description de l'application

Interlude est une application web destinée à un usage privé de sa propre musique. Seule une personne connectée peut accéder à ses musiques et en ajouter, supprimer, ou modifier. Il n'y a pas de gestion de droits d'auteur ni d'administration/modération puisqu'il s'agit simplement d'écouter sa propre musique sans la partager ni la publier. L'application sera développée à l'aide des langages web suivants : PHP, HTML et Javascript.

Les fonctionnalités suivantes doivent être implémentées :

- **Connexion avec Twitter**

Pour se connecter, l'utilisateur doit utiliser son compte Twitter. Cela permet d'éviter la création de comptes de spam, tout en offrant à l'utilisateur une expérience simple et rapide de connexion, lui évitant la création d'un compte supplémentaire. La communication entre Interlude et Twitter se fait à l'aide du protocole OAuth. Tant que l'utilisateur n'est pas connecté, il ne peut pas accéder à la bibliothèque musicale : un écran d'accueil lui présente le projet et lui demande de se connecter.

- **Affichage de la collection de musique**

Une fois connecté sur le site, l'utilisateur a accès à sa bibliothèque musicale, récupérée depuis le serveur. Ses albums s'affichent, triés par ordre d'importation (le plus récent en premier).

- **Lecture musicale**

Lorsque l'utilisateur clique sur un album, une vue détaillée s'ouvre, affichant les morceaux que cet album contient. Il est possible de planifier la lecture d'un morceau (ou de tous les morceaux de l'album) de trois façons :

- immédiatement : le morceau en cours de lecture est remplacé par le morceau sélectionné ;
- à la suite : les morceaux sont ajoutés au début de la file d'attente ;
- plus tard : les morceaux sont ajoutés à la fin de la file d'attente.

La lecture est planifiée pour la session en cours, si l'utilisateur quitte l'application, aucune trace de sa liste de lecture n'est gardée

- **Importation de fichiers audio**

L'utilisateur peut importer un ou plusieurs fichiers audio en les glissant-déposant dans la fenêtre ou en cliquant sur un bouton qui ouvre un sélecteur de fichiers. Interlude crée un album avec les morceaux ajoutés. Les informations concernant le nom de l'album, le nom de l'artiste, la pochette d'album et les numéros de piste sont extraites automatiquement des métadonnées ID3 contenues dans le fichier. Dans le cas où les morceaux ne contiennent pas de métadonnées, un nom d'album par défaut est donné avec la date du jour. Les titres sont nommés Titre1, Titre2, etc. si les métadonnées ne sont pas présentes.

- **Modification de la bibliothèque**

L'utilisateur peut après l'importation :

- modifier le titre d'un album
- modifier l'auteur d'un album
- modifier le titre d'un morceau
- modifier le numéro de piste d'un morceau
- supprimer un morceau
- supprimer un album entier et tous les morceaux qu'il contient.

- **Recherche**

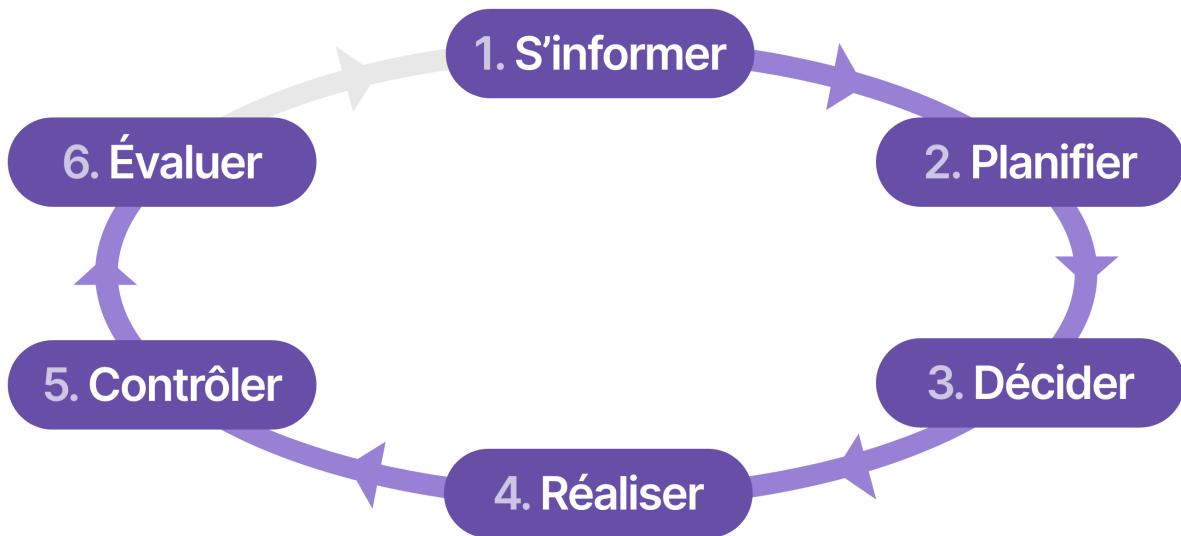
Interlude contient un moteur de recherche permettant de rechercher un morceau ou un album à partir de son nom.

- **Conception web adaptative**

Interlude est conçu pour fonctionner dans un navigateur web sur un ordinateur de bureau. Le site n'a pas besoin de s'adapter aux téléphones portables et tablettes.

Méthodologie

Pour planifier mon projet de TPI, je me suis basé sur la **méthode en 6 étapes** qui nous a été enseignée au CFPT.



1. S'informer

La toute première étape de mon projet était la lecture en profondeur de mon énoncé pour comprendre toutes les fonctionnalités qu'il était nécessaire d'implémenter. J'ai également demandé à ma formatrice des clarifications sur des détails des spécifications lorsque c'était nécessaire.

2. Planifier

Dès le début du projet, j'ai préparé un planning de travail pour savoir ce que j'avais besoin de faire et quand. Afin de pouvoir créer ce planning, j'ai eu besoin de découper mon travail en sous-tâches : j'ai donc divisé les divers points de l'énoncé sous forme d'*user stories*, qui présentent la tâche du point de vue de l'utilisateur final.

Pour chaque story, j'ai fixé une priorité pour m'aider ensuite à savoir ce qui doit être réalisé au plus vite (mes niveaux de priorité sont **🚫 P0 : Bloquant**, **❗ P1 : Critique**, **❗ P2 : Important** et **❓ P3 : Secondaire**).

Une fois les stories créées, je les ai stockées dans un **product backlog**. Ce fonctionnement est inspiré par la méthodologie Scrum (cependant, je n'ai pas appliqué entièrement cette méthodologie, car elle perd son utilité lorsqu'on travaille seul sur un projet).

J'ai ensuite créé un **diagramme de Gantt**, outil qui m'a permis de visualiser au quotidien ma progression dans le travail ainsi que les différences entre ma planification et mon avancement effectif.

3. Décider

Au cours de l'avancement de mon travail, j'ai dû prendre de nombreuses décisions sur la manière de le réaliser. Lorsque je faisais des choix que je jugeais importants pour le projet, j'en ai parlé dans mon journal de bord en expliquant les raisons qui m'ont poussé à choisir ces options d'implémentation.

4. Réaliser

Une fois les bonnes décisions prises, j'ai pu continuer sur l'implémentation de la fonctionnalité dans le code (ou la rédaction pour tout ce qui concerne la documentation).

5. Contrôler

À chaque fois que je terminais une fonctionnalité, je l'ai testée dans différents cas d'usage pour être sûr qu'elle fonctionne comme prévu. J'ai documenté mes tests dans mon rapport TPI pour m'assurer qu'ils soient reproductibles par une personne externe au besoin.

J'ai aussi effectué à nouveau tous mes tests à la fin de chaque journée de travail pour éviter les régressions. J'ai ajouté à mon rapport TPI un tableau qui permet de visualiser l'avancement de l'état des tests au cours du projet.

Enfin, j'ai testé de fond en comble le programme à la fin de son implémentation sur d'autres plateformes (navigateurs et systèmes d'exploitation) pour m'assurer qu'il puisse bien fonctionner chez tout le monde.

6. Évaluer

La dernière étape de la méthodologie en 6 étapes est l'évaluation, pour faire une rétrospective de ce que j'ai fait et avoir un regard critique sur ce qui pourrait être amélioré. Pour ce faire, j'ai écrit des bilans journaliers à chaque fin de journée de travail dans mon journal de bord. Pour avoir une auto-évaluation du travail en entier, j'ai également rédigé la partie *Conclusion* du rapport TPI.

Planification

Product backlog

Nom	S1 : Connexion avec Twitter
Description (user story)	En tant qu'utilisateur non connecté, je peux utiliser mon compte Twitter pour me connecter au site.
Critère d'acceptation	Les tests 1.1 à 1.5 passent.
Priorité	🚫 P0 : Bloquant

Nom	S2 : Importation de musique
Description (user story)	En tant qu'utilisateur connecté, je peux importer les différents fichiers audio qui composent un album pour l'ajouter dans ma bibliothèque.
Critère d'acceptation	Les tests 2.1 et 2.2 passent.
Priorité	❗ P1 : Critique

Nom	S3 : Extraction des métadonnées ID3
Description (user story)	En tant qu'utilisateur connecté, les métadonnées ID3 (nom de l'album, nom de l'artiste, pochette d'album et numéro de piste) des fichiers audio que j'importe sont automatiquement extraites et sont visibles dans ma bibliothèque.
Critère d'acceptation	Le test 2.1 passe.
Priorité	❓ P3 : Secondaire

Nom	S4 : Affichage des albums
Description (user story)	En tant qu'utilisateur connecté, mes albums s'affichent sur la page principale, dans l'ordre dans lequel je les ai importés.
Critère d'acceptation	Le test 3.1 passe.
Priorité	❗ P1 : Critique

Nom	S5 : Recherche de musique
Description (user story)	En tant qu'utilisateur connecté, je peux utiliser un champ de recherche pour trouver un album ou un morceau.
Critère d'acceptation	Les tests 3.2 et 3.3 passent.
Priorité	❗ P2 : Important

Nom	S6 : Affichage d'un album
Description (user story)	En tant qu'utilisateur connecté, je peux cliquer sur un album depuis la vue principale ou des résultats de recherche pour ouvrir la vue d'un album. Cette vue me permet de voir tous ces morceaux (ainsi que leurs numéros de piste).
Critère d'acceptation	Le test 4.1 passe.
Priorité	!! P1 : Critique

Nom	S7 : Modification d'un album
Description (user story)	En tant qu'utilisateur connecté, je peux cliquer sur le bouton "modifier" de la page d'un album pour modifier son titre et son auteur.
Critère d'acceptation	Le test 4.2 passe.
Priorité	! P2 : Important

Nom	S8 : Modification d'un morceau
Description (user story)	En tant qu'utilisateur connecté, je peux modifier pour chaque morceau son titre et son numéro de piste dans la page de modification d'un album.
Critère d'acceptation	Le test 4.3 passe.
Priorité	? P3 : Secondaire

Nom	S9 : Suppression d'un morceau
Description (user story)	En tant qu'utilisateur connecté, j'ai à ma disposition un bouton qui me permet de supprimer un morceau dans le formulaire de modification de son album.
Critère d'acceptation	Le test 4.4 passe.
Priorité	? P3 : Secondaire

Nom	S10 : Suppression d'un album
Description (user story)	En tant qu'utilisateur connecté, j'ai à ma disposition un bouton de suppression sur la page de modification d'un album qui me permet de supprimer l'album en entier. Cette action supprime également tous les morceaux qui le composent.
Critère d'acceptation	Le test 4.5 passe.
Priorité	? P3 : Secondaire

Nom	S11 : Lancement de la lecture d'un album
Description (user story)	En tant qu'utilisateur connecté, je peux cliquer sur un bouton sur la page d'un album pour lancer immédiatement sa lecture. Le premier morceau de l'album est lancé immédiatement, et les morceaux suivants sont ajoutés au début de la liste d'attente.
Critère d'acceptation	Le test 6.1 passe.
Priorité	! P2 : Important

Nom	S12 : Lancement de la lecture d'un morceau
Description (user story)	En tant qu'utilisateur connecté, je peux cliquer sur un bouton "Jouer" à côté d'un morceau dans un album ou dans les résultats de recherche pour lancer immédiatement sa lecture. Ce morceau remplace le morceau final.
Critère d'acceptation	Le test 5.1 passe.
Priorité	!! P1 : Critique

Nom	S13 : Mise en liste d'attente d'un morceau (à la suite et plus tard)
Description (user story)	En tant qu'utilisateur connecté, je peux choisir de mettre un morceau dans la liste d'attente. Lorsque la lecture d'un morceau se termine, le premier morceau de la liste d'attente est retiré de la liste et sa lecture commence. Je peux choisir de mettre mon morceau soit au tout début de la liste d'attente, ou à la fin après les morceaux qui y sont déjà.
Critère d'acceptation	Les tests 7.1 et 7.2 passent.
Priorité	! P2 : Important

Nom	S14 : Mise en liste d'attente d'un album (à la suite et plus tard)
Description (user story)	En tant qu'utilisateur connecté, je peux mettre tous les morceaux d'un album dans la liste d'attente en un clic depuis la page de l'album. Je peux choisir de mettre les morceaux au début de la liste ou à la fin, de la même manière que la story 13 le décrit pour un morceau individuel.
Critère d'acceptation	Les tests 8.1 et 8.2 passent.
Priorité	? P3 : Secondaire

Nom	S15 : Utilisation du lecteur musical
Description (user story)	En tant qu'utilisateur connecté, je vois tout le temps une interface me permettant de contrôler la lecture de ma musique. Je peux mettre ma musique en pause, passer à la suivante et voir quels sont les prochains morceaux dans la liste d'attente. J'ai également un indicateur de progression qui m'indique quelle est la position actuelle de l'écoute dans le morceau, et qui me permet de la redéfinir.
Critère d'acceptation	Les tests 9.1 à 11.2 passent.
Priorité	!! P1 : Critique

Nom	S16 : Affichage de la landing page
Description (user story)	En tant qu'utilisateur non connecté, je ne peux pas encore accéder à la bibliothèque musicale et au lecteur : le site m'affiche une page présentant brièvement le projet et qui contient un bouton "Se connecter avec Twitter".
Critère d'acceptation	Le test 12.1 passe.
Priorité	!! P1 : Critique

Nom	S17 : Vérification du guide de style Airbnb
Description (user story)	En tant que développeur du projet, je peux lancer une commande (npm run lint) qui me permet de vérifier la conformité du code JavaScript du projet avec le guide de style d'Airbnb .
Critère d'acceptation	La documentation du projet m'explique comment utiliser le linter : lorsque j'exécute la commande, les infractions au guide de style me sont signalées s'il y en a.
Priorité	!! P1 : Critique

Nom	S18 : Compilation des ressources JavaScript et CSS
Description (user story)	En tant que développeur du projet, je peux lancer une commande qui compile tous les fichiers source *.js, *.vue et *.css en deux fichiers minifiés (un fichier .js et un fichier .css).
Critère d'acceptation	La documentation du projet m'explique comment lancer la commande de compilation. Lorsque je l'exécute, deux fichiers minifiés (nommés par exemple build.min.js et build.min.css) sont générés à partir des fichiers source.
Priorité	!! P1 : Critique

Nom	S19 : Utilisation d'un dépôt Git
Description (user story)	En tant que développeur, je peux pouvoir utiliser Git sur le code source du projet. Je peux également accéder à un dépôt distant hébergé sur un site comme GitHub.
Critère d'acceptation	Un dépôt Git est configuré dans le dossier contenant le code source. Le code source du projet est accessible depuis le web à l'adresse indiquée dans la documentation.
Priorité	🚫 P0 : Bloquant

Nom	S20 : Utilisation de Composer
Description (user story)	En tant que développeur, je peux pouvoir utiliser Composer pour installer des dépendances PHP externes au projet. Je peux également utiliser les classes propres à Interlude au sein du projet sans avoir à inclure manuellement les fichiers dans lesquels elles se trouvent, en utilisant l'autoloader fourni par Composer.
Critère d'acceptation	Le projet contient un fichier composer.json qui permet à Composer de fonctionner correctement. Dans le code PHP, les classes contenues dans le dossier src du projet sont accessibles via le namespace Interlude.
Priorité	🚫 P0 : Bloquant

Nom	S21 : Configuration de la base de données
Description (user story)	En tant que développeur, je peux utiliser une base de données MySQL respectant le modèle proposé par l'énoncé du TPI. Pour ce faire, j'ai accès à une classe côté PHP qui me permet d'effectuer les opérations de base (CRUD : <i>create, read, update, delete</i>) sur la base de données. Je souhaite également avoir dans le dépôt un <i>dump</i> de la structure de base de données qui me permettrait de la recréer sur un autre environnement.
Critère d'acceptation	Les tables songs, albums et users ont été créées et sont utilisables au moyen d'une classe Database.
Priorité	🚫 P0 : Bloquant

Diagramme de Gantt

Sur ce diagramme, vous pouvez trouver ma planification prévue pour les tâches du projet, ainsi que la façon dont la réalisation des tâches s'est déroulée.

Il n'y a eu qu'une seule différence concernant l'ordre effectif des tâches. Lors du deuxième jour (J1), j'ai fait la *landing page* avant la connexion à Twitter. J'étais en avance mais je n'avais pas pour autant le temps de commencer la connexion à Twitter ce jour-ci. Ce changement d'ordre de causait pas de problème car les tâches étaient indépendantes.

De manière globale, j'ai été en avance sur le planning. Il n'y a pas eu de tâche qui a pris plus longtemps à réaliser que ce qui était prévu (certaines ont simplement été réalisées sur deux journées différentes pour terminer le travail commencé en avance la veille).

Tâche/user story	J0 ma. 7	J1 me. 8	J2 je. 9	J3 lu. 13	J4 ma. 14	J5 me. 15	J6 je. 16	J7 lu. 20	J8 ma. 21	J9 me. 22	J10 je. 23
Lecture de l'énoncé	■										
Rédaction du product backlog	■	■									
Planification des tâches		■									
Rédaction des scénarios de tests		■	■								
S19 Utilisation d'un dépôt Git			■								
S20 Utilisation de Composer			■								
S21 Configuration de la base de données				■							
S18 Compilation des ressources JS & CSS				■							
S17 Vérification du guide de style Airbnb				■							
S1 Connexion avec Twitter					■						
S16 Affichage de la landing page						■					
S4 Affichage des albums							■				

Tâche/user story	J0 ma. 7	J1 me. 8	J2 je. 9	J3 lu. 13	J4 ma. 14	J5 me. 15	J6 je. 16	J7 lu. 20	J8 ma. 21	J9 me. 22	J10 je. 23
S5 Recherche de musique											
S6 Affichage d'un album											
S15 Utilisation du lecteur musical											
S12 Lancement de la lecture d'un morceau											
S11 Lancement de la lecture d'un album											
S13 Mise en liste d'attente d'un morceau											
S14 Mise en liste d'attente d'un album											
S2 Importation de musique											
S3 Extraction des métadonnées ID3											
S7 Modification d'un album											
S8 Modification d'un morceau											
S9 Suppression d'un morceau											
S10 Suppression d'un album											
Tests en profondeur et corrections de bugs											
Finalisation de la documentation											
Tenue du journal de bord											

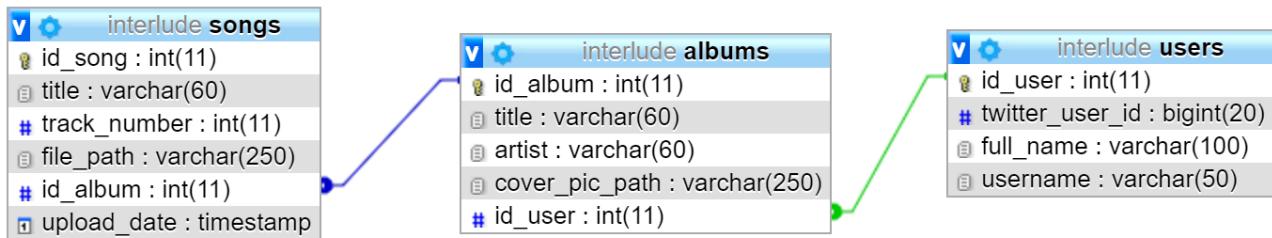
● Planification prévue ● Planification effective

Généralités concernant l'implémentation

Base de données

Interlude utilise une base de données MySQL pour stocker les utilisateurs et le contenu de leurs bibliothèques musicales. La version utilisée pendant le développement est MySQL 14.14. J'ai utilisé InnoDB comme moteur de stockage et utf8_general_ci comme interclassement.

Le modèle de données à utiliser était fourni dans l'énoncé.



Le modèle de données fourni dans l'énoncé

Un script SQL permettant de créer les tables nécessaires m'a également été donné par ma formatrice le premier jour du TPI. J'ai dû y effectuer des changements mineurs en accord avec ma formatrice pour pouvoir réaliser mon projet, car certains champs y avaient le type par défaut au lieu du type adéquat (int(11) à la place de varchar(250)).

Dictionnaire de données

Voici le dictionnaire de données de mon application, tel que créé par phpMyAdmin :

albums

Colonne	Type	Null	Défaut
id_album (<i>Primaire</i>)	int(11)	Non	
title	varchar(60)	Non	
artist	varchar(60)	Non	
cover_pic_path	varchar(250)	Oui	NULL
id_user	int(11)	Non	

Index

Nom de l'index	Type	Unique	Compressé	Colonne	Cardinalité	Interclassement	Null
PRIMARY	BTREE	Oui	Non	id_album	16	A	Non
id_user	BTREE	Non	Non	id_user	2	A	Non

songs

Colonne	Type	Null	Défaut
id_song (<i>Primaire</i>)	int(11)	Non	
title	varchar(60)	Non	
track_number	int(11)	Non	
file_path	varchar(250)	Non	
id_album	int(11)	Non	
upload_date	timestamp	Non	CURRENT_TIMESTAMP

Index

Nom de l'index	Type	Unique	Compressé	Colonne	Cardinalité	Interclassement	Null
PRIMARY	BTREE	Oui	Non	id_song	70	A	Non
id_album	BTREE	Non	Non	id_album	15	A	Non

users

Colonne	Type	Null	Défaut
id_user (<i>Primaire</i>)	int(11)	Non	
twitter_user_id	bigint(20)	Non	
full_name	varchar(100)	Non	
username	varchar(50)	Non	

Index

Nom de l'index	Type	Unique	Compressé	Colonne	Cardinalité	Interclassement	Null
PRIMARY	BTREE	Oui	Non	id_user	2	A	Non

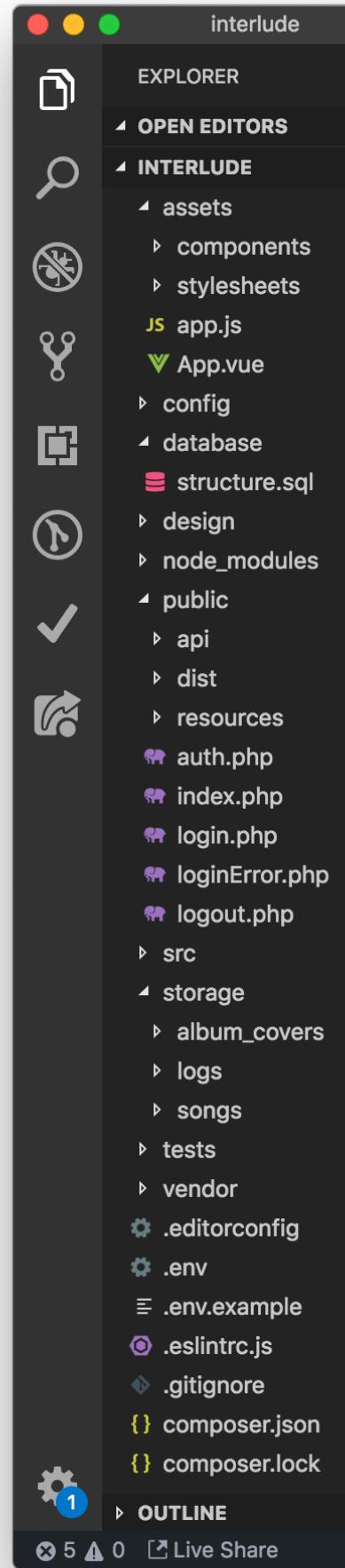
Structure du projet

La structure que j'ai utilisé pour l'arborescence de mon projet est similaire à celle utilisée par des frameworks PHP répandus comme Laravel. Les dossiers employés sont les suivants :

- **/assets/** : Contient les fichiers sources front-end (JavaScript et CSS).
 - /assets/components/ : Composants Vue.js
 - /assets/stylesheets/ : Feuilles de styles non liées à un composant Vue.js en particulier
- **/config/** : Contient le fichier config.inc.php, qui est inclus dans tous les fichiers PHP accessibles par les utilisateurs.
- **/database/** : Contient un *dump* de la structure de la base de données.
- **/design/** : Contient les fichiers source Adobe XD, Illustrator et Sketch utilisés pour créer les éléments visuels du projet.
- **/node_modules/** : *Dépendances JavaScript externes (géré par NPM)*
- **/public/** : Racine du serveur web. Tous les fichiers entreposés dans ce dossier sont accessibles directement par les utilisateurs.
 - **/public/api/** : API interne
 - **/public/dist/** : *Fichiers JavaScript et CSS générés par Laravel Mix*
 - **/public/resources/** : Images et autres ressources statiques
- **/src/** : Classes PHP du projet Interlude
- **/storage/** : Contient les données stockées par l'application elle-même : données des utilisateurs (fichiers audio et couvertures d'album) et logs.
- **/tests/** : Ressources (données de test) pour tester l'application
- **/vendor/** : *Dépendances PHP externes (géré par Composer)*

Les dossiers en gras sont les plus importants.

Les dossiers en italique contiennent des fichiers générés automatiquement.



Classes (PHP)

Pour implémenter les diverses fonctionnalités d'Interlude, j'ai créé les classes suivantes :

\Interlude\Album

Modèle représentant un album

\Interlude\Auth

Gère l'authentification d'un utilisateur

\Interlude\Database

Classe permettant d'effectuer des opérations sur la base de données MySQL

\Interlude\Song

Modèle représentant un morceau

\Interlude\User

Modèle représentant un utilisateur

\Interlude\Exceptions\UploadException

Exception émise lorsqu'une erreur se produit pendant l'import d'un fichier audio

\Interlude\Util\Files

Classe utilitaire contenant des fonctions utiles pour travailler avec l'import de fichiers

\Interlude\Util\Metadata

Classe permettant d'extraire le métadonnées ID3 d'un fichier

Le fonctionnement des méthodes de ces classes est expliqué en détail dans le code source.

API interne

Interlude dispose d'une API interne qui est utilisée depuis le front-end pour interagir avec les données de l'application. Les différents points d'entrée disponibles sont :

/api/delete.php

Permet de supprimer un album et les morceaux qu'il contient

/api/edit.php

Permet de modifier un album et ses morceaux

/api/library.php

Permet d'obtenir la bibliothèque musicale (album et morceaux) de l'utilisateur connecté

/api/upload.php

Importe des fichiers audio dans la bibliothèque musicale de l'utilisateur connecté

/api/user.php

Permet d'obtenir l'utilisateur actuellement connecté.

Les paramètres et types de retour de l'API sont expliqués en détail dans le code source.

Composants Vue.js

Pour réaliser l'interface utilisateur d'Interlude, j'ai créé des composants. Les composants sont une fonctionnalité que le framework Vue.js propose pour avoir un élément d'interface (appelé *composant*) réutilisable qui possède son propre état (ses propres données).

Un composant peut contenir des instances d'autres composants. Un composant parent peut passer des propriétés (*props*) à ses composants enfants, et un composant enfant peut émettre des événements (*events*) pour communiquer des informations à son composant parent.

Les composants que j'ai créés sont les suivants :

App.vue

Composant principal de l'application. Il affiche la structure de base de l'interface, ainsi que la liste des albums et la recherche.

AlbumPage.vue

Page où l'on voit les détails d'un album. Ce composant gère aussi la modification de l'album.

DraggingFileOverlay.vue

Élément d'interface qui s'affiche lorsqu'un fichier est en train d'être glissé sur la fenêtre d'Interlude. Affiche le texte « *Déposez les fichiers pour les importer dans votre bibliothèque musicale.* ».

LandingPage.vue

Page qui s'affiche lorsque l'utilisateur n'est pas connecté. Contient le bouton « *Se connecter avec Twitter* ».

Player.vue

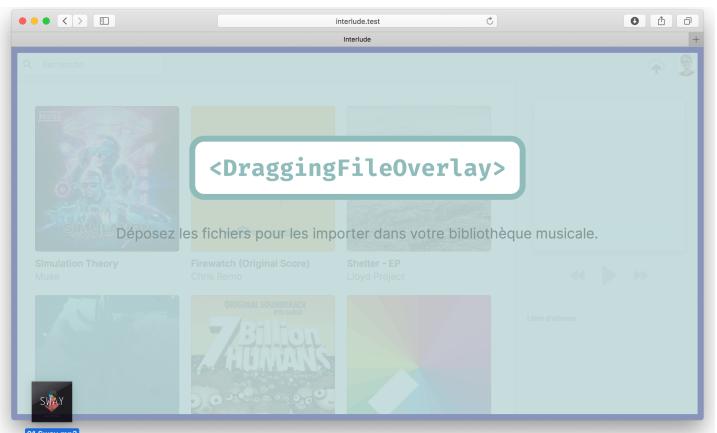
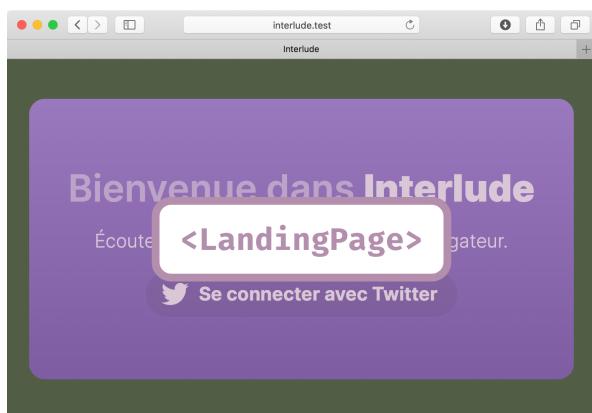
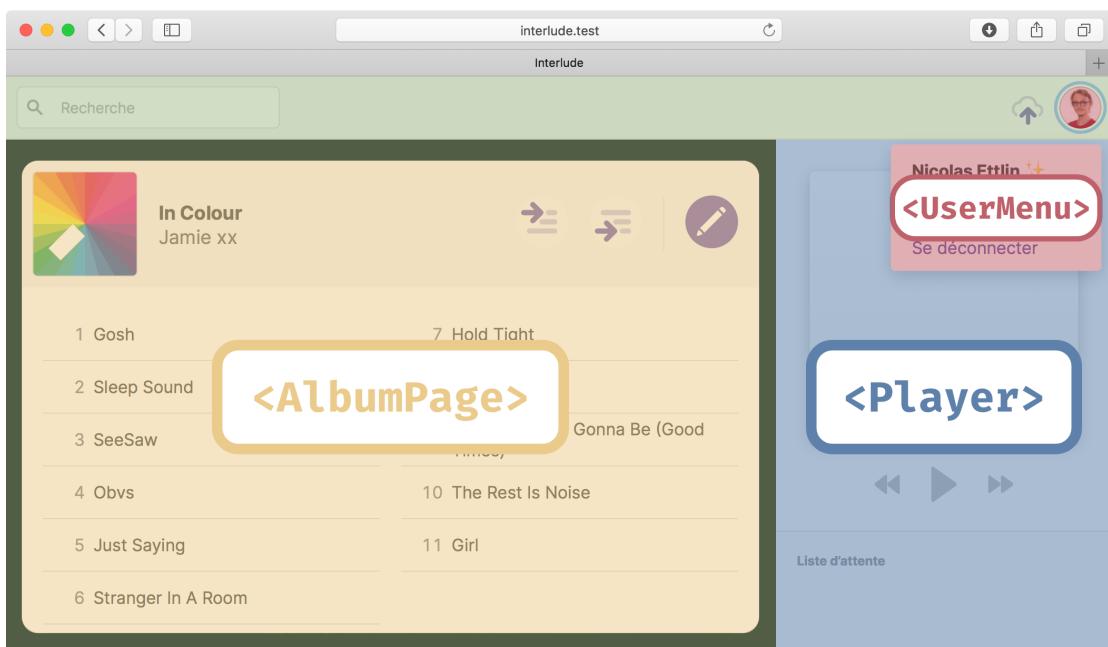
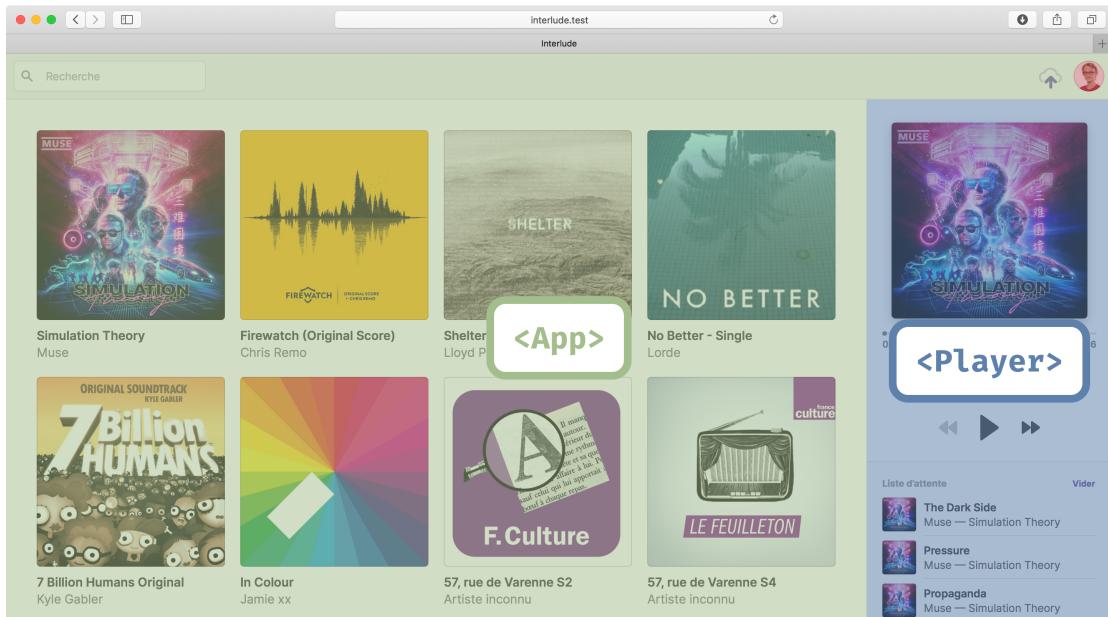
Contient le lecteur audio et l'affichage de la liste d'attente.

UserMenu.vue

Affiche l'avatar de l'utilisateur connecté. Lorsqu'on clique dessus, le composant affiche un menu avec le nom complet de l'utilisateur, son @nom Twitter et un bouton de déconnexion.

Exemples d'utilisation des composants

Voici des exemples d'utilisation de mes composants Vue.js dans l'application :



Librairies et outils externes

Vue.js

Vue.js est un framework JavaScript utilisé pour la création d'interfaces utilisateur réactives. Il a été créé en 2014 par Evan You, un ancien employé de Google, et fait aujourd'hui partie des frameworks *front-end* JavaScript les plus populaires aux côtés de React et d'Angular¹. J'ai choisi d'utiliser ce framework pour mon TPI car il m'a permis d'aisément :



- Gérer mes données sur toute l'étendue de l'application, en m'assurant que toutes les vues sont immédiatement mises à jour. Un bon exemple de l'utilité de Vue.js est la fonctionnalité de modification d'un album : lorsque je change le titre d'un album, cette information est directement mise à jour dans toutes les parties de l'application où elle s'affiche (bibliothèque musicale, page de l'album, résultats de recherche et lecteur musical).
- Gérer plusieurs pages (bibliothèque musicale, album, modification d'album...) sans qu'un nouveau document HTML n'ait besoin d'être chargé. Cette partie était indispensable afin que le morceau en cours de lecture dans le lecteur ne soit pas interrompu lorsque l'utilisateur se rend sur une nouvelle page.
- Gérer certaines transitions visuelles entre plusieurs états (Vue.js propose des outils qui aident le développeur à le faire).

Axios

Axios est une librairie JavaScript qui permet de faire des requêtes AJAX. Cette librairie est fréquemment utilisée en conjonction par les développeurs avec Vue.js. Elle présente les avantages suivants par rapport aux APIs natives des navigateurs :

- Syntaxe d'utilisation moderne (basé sur l'objet Promise) et plus simple à utiliser pour la création d'applications web que l'API native XMLHttpRequest.
- Comptabilité assurée avec l'intégralité des navigateurs web fréquemment utilisés (ainsi que certains navigateurs plus anciens comme Internet Explorer 11 qui ne supportent pas les APIs natives les plus récentes, comme fetch)
- Il est possible de configurer Axios de manière globale pour envoyer à chaque requête certains en-têtes, pour l'authentification par exemple. Cependant, je n'ai pas utilisé cette fonctionnalité.

¹ <https://2018.stateofjs.com/front-end-frameworks/overview/overview>

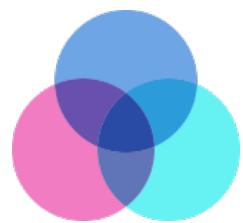
NPM et Composer

NPM et Composer sont deux gestionnaires de dépendances que j'ai utilisés dans mon projet. Le premier permet de gérer les dépendances JavaScript tandis que le second est son équivalent pour PHP. Ces deux outils me permettent de télécharger automatiquement tout le code externe dont j'ai besoin dans mon projet : il ne m'est donc pas nécessaire de récupérer manuellement les sources nécessaires et de les copier dans mon dossier de projet. L'utilisation de gestionnaires de dépendances tels que NPM et Composer permettent donc de faciliter le développement du projet et d'utiliser des dépendances à jour simplement.



Laravel Mix

Laravel Mix est l'une des briques logicielles qui composent le célèbre framework PHP Laravel. Je n'ai pas utilisé Laravel dans mon projet, car la complexité en terme de fonctionnalités à implémenter était réduite et un outil aussi puissant que Laravel n'était pas nécessaire. Cependant, j'ai pu utiliser Mix à part. C'est un outil qui permet de compiler automatiquement les ressources JavaScript et CSS. Mix m'a offert les avantages suivants :



- Configuration simple et installation rapide (par rapport à d'autres outils plus complexes comme Webpack, librairie sur laquelle Laravel Mix est basée par ailleurs).
- Commande `npm run watch` qui recompile les ressources à chaque modification pendant le développement, sans avoir à lancer une commande soi-même.
- Commande `npm run production` (ou `prod`) qui crée les mêmes fichiers que la commande `npm run watch`, mais en une version *minifiée* (taille réduite).
- Gestion native des fichiers `.vue`, qui contiennent à la fois le code HTML, JavaScript et CSS d'un composant Vue.js.
- Gestion native du préprocesseur SASS (voir plus bas).

```

1. fish /Users/nicolapps/Documents/interlude (fish)
fish /Users/nicolapps... #1
~/.local/share/fish/functions/npm.fish:1: master ±
~/.local/share/fish/functions/npm.fish:1: npm run production

> interlude@1.0.0 production /Users/nicolapps/Documents/interlude
> cross-env NODE_ENV=production node_modules/webpack/bin/webpack.js --no-progress --hide-modules --config=node_modules/laravel-mix/setup/webpack.config.js

 DONE Compiled successfully in 2492ms
9:22:11 AM

      Asset      Size  Chunks      Chunk Names
/public/dist/app.js    131 KiB     0  [emitted]  /public/dist/app
/public/dist/app.css   24.6 KiB     0  [emitted]  /public/dist/app
~/.local/share/fish/functions/npm.fish:1: master ±

```

Utilisation de Mix : deux fichiers, `app.js` et `app.css` sont automatiquement générés.

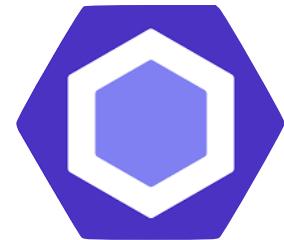
SASS

SASS est un préprocesseur CSS : cet outil permet d'étendre les possibilités du langage CSS pour y ajouter de nombreuses fonctionnalités qui sont utile pour la création de feuilles de style avancées. SASS supporte notamment un système de variables (pour spécifier à un seul endroit des informations comme les couleurs utilisées dans l'application) et un système de fonctions (les *mixins*) qui permet de réutiliser des styles à différents endroits.



ESLint

ESLint est un outil qui permet de vérifier automatiquement que le code d'un projet est conforme à des règles de style précédemment établies. Ce projet m'a été utile pour m'assurer que mon code JavaScript respectait le guide de style d'Airbnb, ce qui était l'un des critères de la grille d'évaluation spécifiques à mon projet.



```
1. fish /Users/nicolapps/Documents/interlude (fish)
x fish /Users/nicolapp... %1
~/D/interlude ➜ master ✘ npm run lint
> interlude@1.0.0 lint /Users/nicolapps/Documents/interlude
> eslint --ext .js,.vue assets

/Users/nicolapps/Documents/interlude/assets/App.vue
  367:24  error  Strings must use singlequote  quotes

✖ 1 problem (1 error, 0 warnings)
  1 error and 0 warnings potentially fixable with the `--fix` option.

npm ERR! code ELIFECYCLE
npm ERR! errno 1
npm ERR! interlude@1.0.0 lint: `eslint --ext .js,.vue assets`
npm ERR! Exit status 1
npm ERR!
npm ERR! Failed at the interlude@1.0.0 lint script.
npm ERR! This is probably not a problem with npm. There is likely additional logging output above.

npm ERR! A complete log of this run can be found in:
npm ERR!     /Users/nicolapps/.npm/_logs/2019-05-20T06_39_07_407Z-debug.log
✖ ~/D/interlude ➜ master ✘
```

Exemple d'utilisation de la commande `npm run lint` fonctionnant grâce à ESLint. Ici, ESLint me signale que je dois utiliser des apostrophes au lieu de guillemets pour délimiter mes chaînes de caractères.

getID3()

getID3() est une librairie PHP qui permet d'extraire les métadonnées ID3 de fichiers. J'en ai eu besoin pour pouvoir récupérer automatiquement les informations des fichiers audio importés par l'utilisateur dans Interlude.

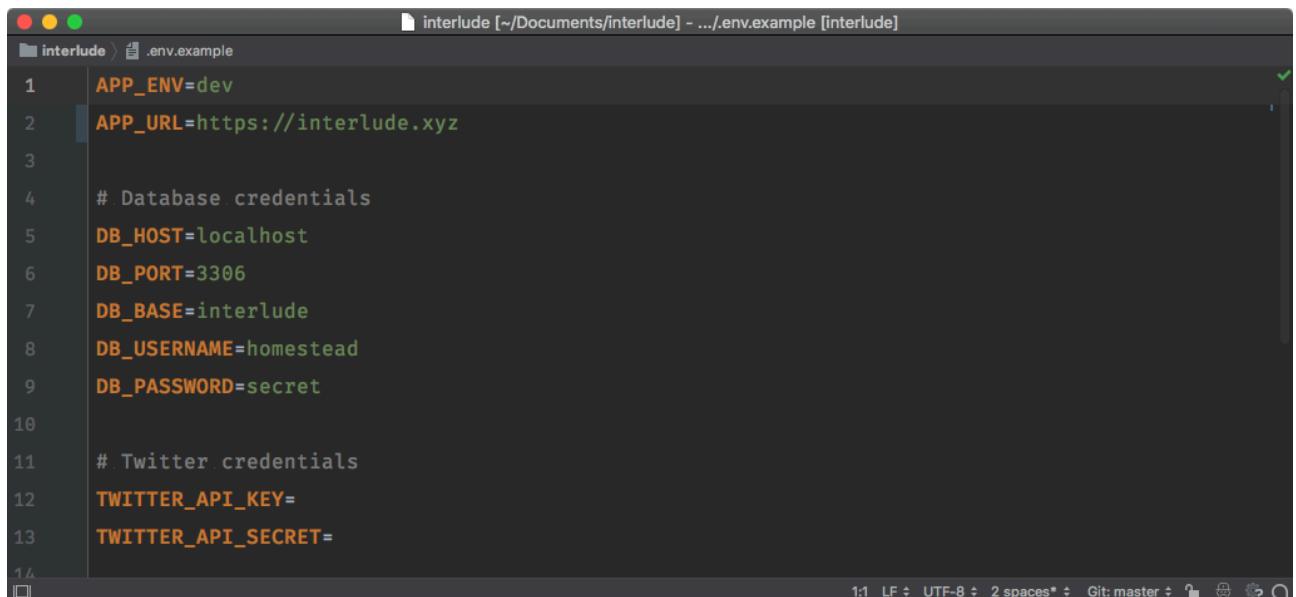
J'utilise cette librairie sous licence LGPL v3. Cette licence m'autorise à utiliser *getID3()* dans Interlude à condition que mon projet soit indépendant de la librairie : pour satisfaire cette obligation légale, j'ai fait en sorte que si la librairie n'est pas installée, Interlude se comporte comme si aucune métadonnée n'avait été trouvée dans le fichier importé. La licence LGPL m'oblige également à publier toute modification que j'aurais faite au code source de *getID3()* sous la même licence ; cependant, je n'ai pas eu besoin de la modifier.

TwitterOAuth

TwitterOAuth est une librairie qui permet d'utiliser les APIs que Twitter propose depuis PHP. Je l'ai utilisée pour la fonctionnalité de connexion avec Twitter, en gérant les différentes erreurs que la librairie pourrait émettre (par exemple, indisponibilité du service Twitter). Cette librairie n'est pas maintenue par Twitter mais par un développeur indépendant, Abraham Williams.

PHP dotenv

PHP dotenv est une librairie PHP qui me permet de stocker mes variables d'environnements (comme mes identifiants d'accès à la base de données et mes clés d'accès aux APIs Twitter) dans des fichiers .env qui ne sont pas versionnés, et ainsi pouvoir passer plus aisément d'une instance d'Interlude à une autre (environnement de développement, serveur de production...), tout en m'assurant que les identifiants secrets restent secrets.



```
interlude [~/Documents/interlude] - .../.env.example [interlude]
1 APP_ENV=dev
2 APP_URL=https://interlude.xyz
3
4 # Database credentials
5 DB_HOST=localhost
6 DB_PORT=3306
7 DB_BASE=interlude
8 DB_USERNAME=homestead
9 DB_PASSWORD=secret
10
11 # Twitter credentials
12 TWITTER_API_KEY=
13 TWITTER_API_SECRET=
```

Fichier .env d'exemple du projet Interlude

Symfony VarDumper

Le composant VarDumper fait partie du framework PHP Symfony (mais est aussi utilisé par d'autres frameworks comme Laravel, ou dans des projets qui n'utilisent pas de frameworks PHP comme Interlude). J'ai utilisé cette librairie pendant le développement d'Interlude pour trouver l'origine de bugs mais aussi pour tester les différentes parties de mon logiciel. VarDumper propose une fonction appelée `dump` qui fonctionne comme la fonction `var_dump` de PHP, mais dont les résultats sont plus faciles à utiliser depuis le navigateur.



 A screenshot of a terminal window titled "interlude [~/Documents/interlude] - .../public/test.php [Interlude]". The command "dump(Auth :: user()->albums());" is run at line 5. The output shows a detailed dump of an array of 11 Album objects, each with properties like id, title, and artist. The output is color-coded for readability.

```

array:11 [▼
  0 => Album {#14 ▼
    +id: 17
    +title: "In Colour"
    +artist: "Jamie xx"
  }
  1 => Album {#15 ▼
    +id: 16
    +title: "Simulation Theory"
    +artist: "Muse"
  }
  2 => Album {#16 ▶}
  3 => Album {#17 ▶}
  4 => Album {#18 ▶}
  5 => Album {#19 ▶}
  6 => Album {#20 ▶}
  7 => Album {#21 ▶}
  8 => Album {#22 ▶}
  9 => Album {#23 ▶}
  10 => Album {#24 ▶}
]
  
```

Exemple d'utilisation de Symfony VarDumper

Git

Le logiciel de gestion de versions Git a été utilisé tout au long de la création du projet afin de garder un historique des modifications effectuées, de synchroniser en permanence les dernières versions du code source et d'avoir une sauvegarde externe en cas de défaillance technique.

Un dépôt distant contenant le code source du projet est disponible sur GitHub (droits d'accès nécessaires) :

<https://github.com/Nicolapps/Interlude>



Analyse des fonctionnalités majeures

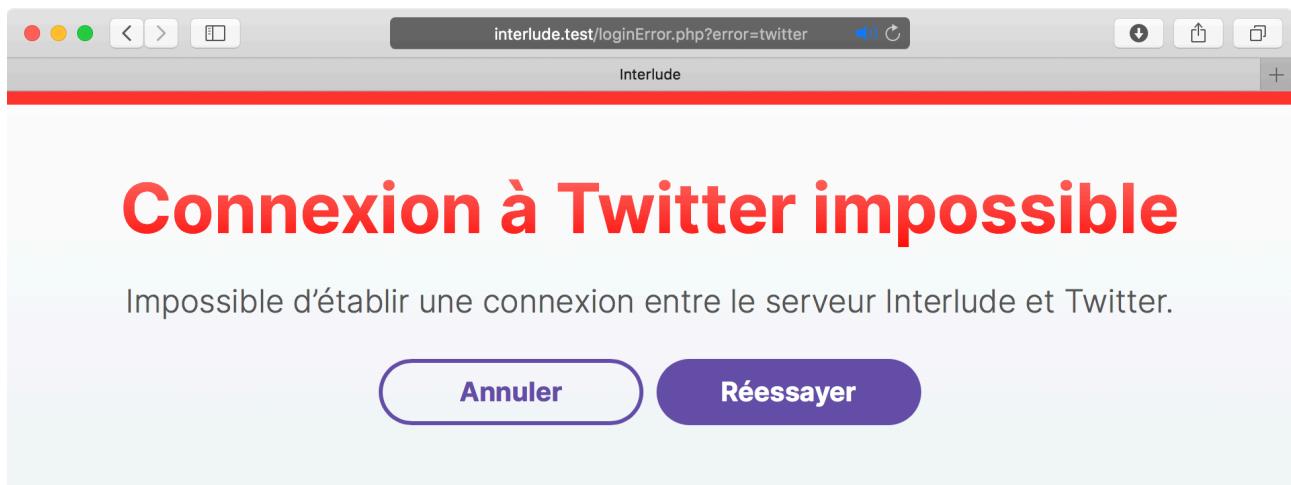
Connexion avec Twitter

La première fonctionnalité majeure de mon application est la connexion avec Twitter. Celle-ci est un peu différente des fonctionnalités d'inscription/connexion qu'on a l'habitude d'implémenter, car elle doit interagir avec un service externe pour fonctionner. Comme expliqué dans la partie *Librairies et outils externes* du rapport, j'ai utilisé une librairie PHP disponible sur Composer qui me permet d'interagir avec le service Twitter, nommée TwitterOAuth.

Mon implémentation de la connexion avec Twitter fonctionne grâce à trois pages :

- public/**login.php** génère un lien unique de connexion sur le service Twitter (vers l'URL <https://api.twitter.com/oauth/authorize>, mais avec des jetons uniques) et y redirige l'utilisateur.
- public/**auth.php** est la page vers laquelle Twitter redirige l'utilisateur (c'est la *callback URL*) après que l'utilisateur accepte ou refuse la connexion. Si l'utilisateur accepte, cette page vérifie le jeton que Twitter a généré puis connecte l'utilisateur à son compte.
- public/**loginError.php** est la page qui affiche les messages d'erreur en cas de problème durant la connexion. L'utilisateur est redirigé vers cette page avec un code d'erreur lorsqu'un problème quelconque se produit. Il peut alors retourner vers la *landing page* ou recommencer le processus de connexion.

Étant donné que j'interagis avec un service externe, je dois m'assurer que je gère les différents problèmes qui peuvent se produire entre mon serveur et celui de Twitter. Pour ce faire, je récupère à chaque fois que j'utilise TwitterOAuth les TwitterOAuthException que cette librairie émet. Cela me permet d'afficher un message clair à l'utilisateur sur ce qui s'est passé.



Le message affiché lors d'un problème de connexion (par exemple réseau coupé).

Affichage de la bibliothèque musicale

Lorsqu'un utilisateur connecté ouvre Interlude, ses albums et ses morceaux sont récupérés depuis l'API. La vue contenant les albums de l'utilisateur affiche de grandes photos de couverture qui permettent de facilement visualiser sa musique et identifier les albums.

J'utilise la fonctionnalité CSS Grid, aujourd'hui très répandue dans les navigateurs modernes, pour afficher cette vue : la fonction `minmax` me permet d'optimiser l'affichage en utilisant toute la place disponible pour afficher des albums.

Dans l'affichage de la vue des albums (mais aussi dans la plupart des éléments d'interface), j'utilise la déclaration CSS `text-overflow: ellipsis` pour éviter que les données d'utilisateurs débordent de l'interface dans certains cas (si elles sont plus longues que ce qui serait affichable avec la place disponible).

Lecteur musical

Mon lecteur personnalisé est implémenté à l'aide d'un élément `<audio>` caché : les éléments d'interface (bouton play, pause, suivant, barre de progression...) ne font que répercuter les actions de l'utilisateur sur l'élément `<audio>`. Je récupère l'accès à cet élément grâce au système de référence de Vue.js : en y ajoutant l'attribut `ref="audio"`, je peux accéder à l'élément depuis mon code en utilisant l'identificateur `this.$refs.audio`.

Importation de fichiers audio

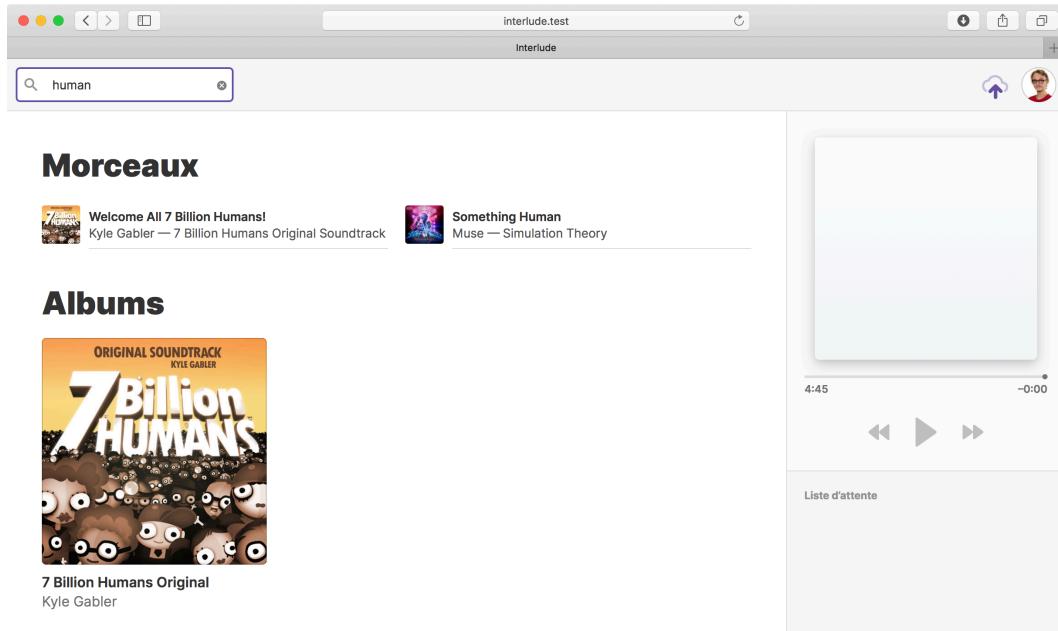
Les spécifications du projet indiquaient deux manières pour l'utilisateur de téléverser un fichier : par glisser/déposer ou au moyen d'un bouton. Pour la première façon, j'ai utilisé l'événement `drop` que le navigateur émet pour détecter lorsqu'un fichier est déposé. Pour la deuxième façon, j'ai utilisé un élément `<input type="file">` caché. Lorsque je reçois l'événement `drop` ou l'événement `change` du sélecteur de fichiers, j'envoie les fichiers à la fonction `upload` qui les envoie via AJAX.

Côté serveur, je commence par vérifier que tous les fichiers audio sont valides (je vérifie le type MIME pour m'assurer qu'il s'agit bien de fichiers audio). Si c'est le cas, je stocke les fichiers et j'extrais leurs métadonnées avec la librairie `getID3()`. Ensuite, je répartis les fichiers par nom d'album et je choisis pour chaque album quel morceau sera utilisé pour le métadonnées de l'album : par exemple, j'utilise le nom d'artiste le plus fréquent comme nom d'artiste de l'album.

Une fois que j'ai récupéré toutes les métadonnées, j'insère les albums et les morceaux en base de données et je les renvoie au client pour qu'ils soient ajoutés à son interface.

Recherche de musique

J'ai choisi de faire fonctionner le système de recherche entièrement depuis le navigateur de l'utilisateur : cela me permet d'avoir un système très rapide car il n'y a pas de requête à faire vers le serveur.



L'affichage des résultats de recherche

Pour récupérer les éléments à afficher depuis mon code JavaScript, je prends tous les éléments et je filtre ceux qui contiennent la recherche dans leur titre ou artiste :

```
/**
 * Returns the search results to display
 * @returns {{albums: Array, songs: Array}}
 */
searchResults() {
    if (!this.songs || !this.albums) { // Not loaded yet
        return { songs: [], albums: [] };
    }

    const query = this.searchQuery.toLowerCase();

    // This builds a search filter for the attributes passed
    const byAttributes = (...attributesToCheck) => (
        itemToCheck => attributesToCheck
            .some(attribute => itemToCheck[attribute].toLowerCase().includes(query))
    );

    const songs = this.songs.filter(byAttributes('title'));
    const albums = this.albums.filter(byAttributes('title', 'artist'));

    return { songs, albums };
},
```

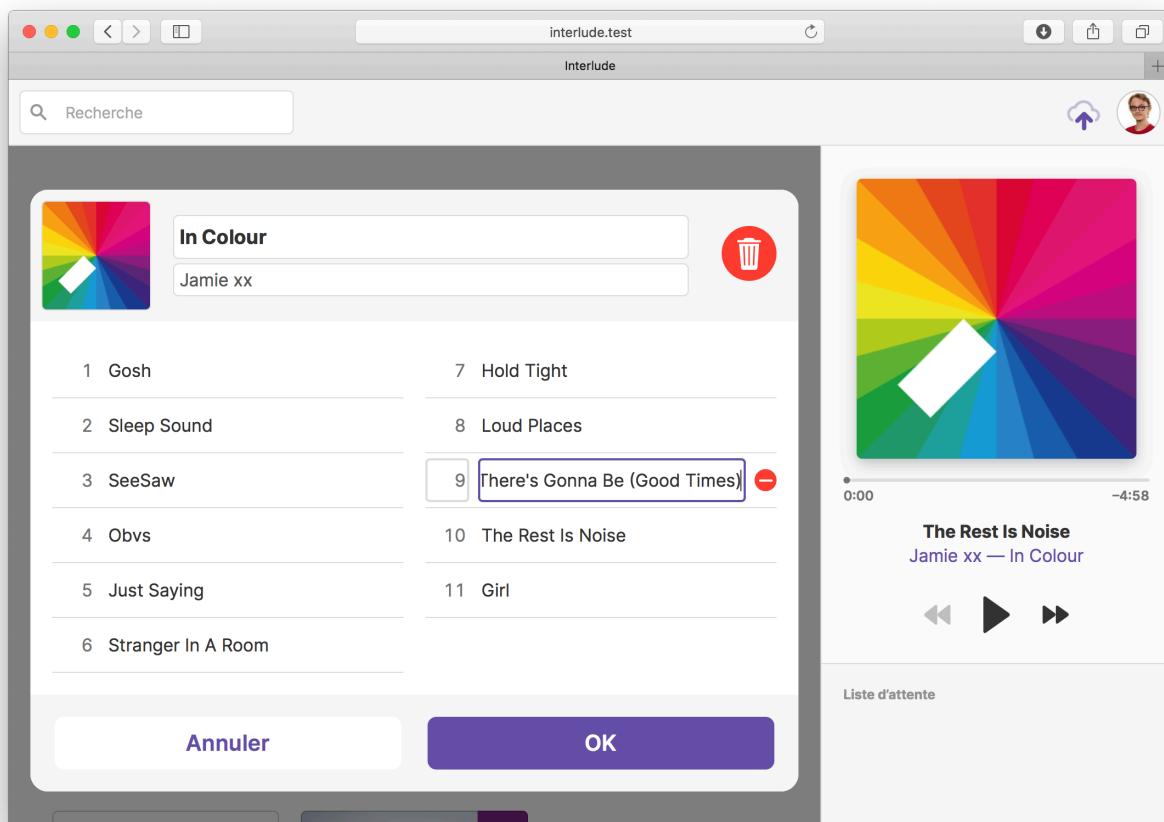
Modification de la bibliothèque musicale

Lorsque l'utilisateur clique sur le bouton *Modifier* d'un album, l'interface de modification s'ouvre. Les zones contenant du texte sont remplacées par des champs de texte qui sont modifiables par l'utilisateur. Lors de l'ouverture de l'interface de modification, une copie des données de l'album est effectuée pour la modification, afin que les données originales ne soient pas écrasées tant que l'utilisateur n'a pas terminé la modification. Un bouton permet également de supprimer un morceau dans l'album.

Côté serveur, je mets à jour les informations pour l'album et pour chaque piste. Si un morceau est présent dans l'album en base de données mais pas dans les données envoyées par le client, ce morceau est effacé du serveur : cela signifie que l'utilisateur l'a supprimé.

Ensuite, le serveur renvoie l'album modifié et les titres pour que l'interface utilisateur puisse se mettre à jour avec les nouvelles données. Étant donné que les données sur les morceaux et albums ne sont stockées qu'à un seul endroit (dans App.vue), toute l'interface est mise à jour pour prendre en compte les modifications.

La fonctionnalité de suppression d'un album fonctionne avec une API séparée (*edit.php*), car elle ne fait que supprimer l'album et toutes ses pistes.



Plan de test et tests

Périmètre des tests

Pour le projet Interlude, j'ai choisi d'écrire des protocoles de test pour les actions qu'un utilisateur lambda utilisant un navigateur moderne et répandu pourrait raisonnablement effectuer. J'ai aussi pris en compte l'éventualité où l'application n'arrive pas à interagir avec le service externe (c'est-à-dire Twitter) comme prévu. De plus, j'ai ajouté la vérification du style de code en tant que test, car c'est également quelque chose que je dois vérifier chaque jour pendant mon travail sur le projet.

Équipement de test

Environnement

Lors du développement du projet, j'ai utilisé l'environnement suivant pour effectuer les tests :

- Mozilla Firefox 66.0.5 (64 bits) sur Windows 10 Pro, version 1809
- Google Chrome 74.0.3729.157 (64 bits) sur Windows 10 Pro, version 1809

Après l'implémentation de toutes les fonctionnalités, j'ai également effectué les tests sous les environnements suivants pour m'assurer du fonctionnement du projet sur macOS :

- Apple Safari 12.1 (14607.1.40.1.4) sur macOS Mojave, version 10.14.4
- Mozilla Firefox 66.0.5 (64 bits) sur macOS Mojave, version 10.14.4
- Google Chrome 74.0.3729.157 (64 bits) sur macOS Mojave, version 10.14.4

Côté serveur, nginx 1.15.0 a été utilisé avec PHP 7.2 et MySQL 14.14.

Le système utilisé comme serveur est la machine virtuelle Laravel Homestead 6.3.0 (à base d'Ubuntu 18.04.1 LTS).

Fichiers de test

Pour tester la fonctionnalité l'extraction des métadonnées ID3, j'ai utilisé les fichiers suivants :

- Bande-son originale du jeu *7 Billion Humans* au format MP3
(fichiers contenant des métadonnées ID3)
- Bande-son originale du jeu *Firewatch* au format MP3
(fichiers ne contenant pas de métadonnées ID3)

Ces fichiers sont disponibles dans le dépôt Git du projet Interlude, dans le dossier tests.

Scénarios de test

Lors du premier jour du TPI, j'ai rédigé les scénarios de tests suivants qui m'ont permis par la suite de m'assurer que les besoins exprimés dans l'énoncé sont remplis. Les scénarios de tests sont précis dans le but d'être facilement reproductibles par une autre personne si c'est nécessaire. Pour ce faire, ils utilisent une structure proche de la **syntaxe Gherkin**.

Nom	1.1 Connexion avec Twitter (nouveau compte)
User story	S1 : Connexion avec Twitter
Situation	<p>Étant donné que je suis un utilisateur non connecté, qui possède un compte Twitter jamais utilisé sur le site</p> <p>Quand je clique sur le bouton "Se connecter avec Twitter" sur la <i>landing page</i> et que je donne l'autorisation à Interlude d'accéder à mon compte Twitter</p> <p>Alors, je suis redirigé vers la page principale du site, en étant connecté à mon compte nouvellement créé. La photo de profil de mon compte Twitter est visible dans la barre de navigation.</p>
Résultats obtenus	Je clique sur <i>Se connecter avec Twitter</i> . Je suis redirigé vers une page <i>Autoriser Interlude à utiliser votre compte ? sur twitter.com</i> . Je clique sur <i>Autoriser l'application</i> . Je suis rapidement redirigé sur Interlude et ma photo de profil est visible dans la barre de navigation.
Statut	✓ OK

Nom	1.2 Connexion avec Twitter (compte existant)
User story	S1 : Connexion avec Twitter
Situation	<p>Étant donné que je suis un utilisateur non connecté, qui possède un compte Twitter qui a déjà été utilisé sur le site</p> <p>Quand je clique sur le bouton "Se connecter avec Twitter" sur la <i>landing page</i> et que je donne l'autorisation à Interlude d'accéder à mon compte Twitter</p> <p>Alors, je suis redirigé vers la page principale du site, en étant connecté à mon compte existant (les albums que j'ai importés auparavant sont visibles). La photo de profil actuelle de mon compte Twitter est visible dans la barre de navigation.</p>
Résultats obtenus	Je change mon nom sur Twitter. Je clique sur <i>Se connecter avec Twitter</i> . Je suis redirigé vers une page <i>Autoriser Interlude à utiliser votre compte ? sur twitter.com</i> . Je clique sur <i>Autoriser l'application</i> . Je suis rapidement redirigé sur Interlude et ma photo de profil est visible dans la barre de navigation. Lorsque je vais dans le menu <i>Utilisateur</i> , mon nouveau nom est visible.
Statut	✓ OK

Nom	1.3 Refus de la connexion Interlude ↔ Twitter
User story	S1 : Connexion avec Twitter
Situation	<p>Étant donné que je suis un utilisateur non connecté</p> <p>Quand je clique sur le bouton "Se connecter avec Twitter" sur la <i>landing page</i>, mais que je refuse ensuite à Interlude l'accès à mon compte Twitter</p> <p>Alors, je suis redirigé vers une page d'erreur du site Interlude m'informant que la connexion à mon compte Twitter n'a pas pu être établie à cause de mon refus.</p>
Résultats obtenus	Twitter m'affiche une page « Vous ne vous êtes pas connecté à Interlude » contenant un bouton « Retourner sur Interlude ». Lorsque je clique sur ce bouton, une page d'Interlude s'affiche : « Accès refusé / Vous avez refusé à Interlude l'accès à votre compte Twitter. Cet accès est nécessaire au fonctionnement du programme » avec des boutons <i>Réessayer</i> et <i>Annuler</i> .
Statut	✓ OK

Nom	1.4 Déconnexion
User story	S1 : Connexion avec Twitter
Situation	<p>Étant donné que je suis un utilisateur connecté au site</p> <p>Quand je clique sur le bouton "Se déconnecter" du site dans le menu <i>Utilisateur</i></p> <p>Alors je deviens un utilisateur non connecté et je suis redirigé vers la <i>landing page</i>.</p>
Résultats obtenus	Je clique sur le bouton <i>Utilisateur</i> puis sur le bouton <i>Se déconnecter</i> . La <i>landing page</i> s'affiche à nouveau et ma photo de profil n'est plus visible dans la barre de navigation.
Statut	✓ OK

Nom	1.5 Connexion au service Twitter impossible
User story	S1 : Connexion avec Twitter
Situation	<p>Étant donné que je suis un utilisateur non connecté et que le serveur d'Interlude est déconnecté d'Internet (et ne peut donc pas accéder au service Twitter)</p> <p>Quand je clique sur le bouton "Se connecter avec Twitter" sur la <i>landing page</i></p> <p>Alors, je suis redirigé vers une page d'erreur du site Interlude m'informant que la connexion à mon compte Twitter n'a pas pu être établie à cause d'un problème de connexion entre Interlude et Twitter.</p>
Résultats obtenus	« Impossible d'établir une connexion entre le serveur Interlude et Twitter. »
Statut	✓ OK

Nom	2.1 Importation de musique via un bouton (avec métadonnées ID3)
User story	S2 : Importation de musique
Situation	<p>Étant donné que je suis un utilisateur connecté</p> <p>Quand je clique sur le bouton "Importer" de la page principale, et que dans le sélecteur du fichier qui s'ouvre je sélectionne tous les fichiers de la bande son du jeu <i>7 Billion Humans</i> (présente dans l'équipement de test), puis que je clique sur « OK »</p> <p>Alors, un nouvel album nommé « <i>7 Billion Humans</i> » ayant comme artiste « <i>Kyle Gabler</i> » est créé. Cet album contient 14 morceaux ayant tous un titre correct (par exemple, la piste n° 5 s'appelle « <i>Long Term Storage</i> »). La photo de couverture qui est affichée correspond à la photo de couverture contenue dans les fichiers MP3.</p>

Résultats obtenus	Le nouvel album importé s'ouvre. On y voit le titre « <i>7 Billion Humans Original Soundtrack</i> », « <i>Kyle Gabler</i> » comme auteur et 14 morceaux bien nommés dans l'ordre correct.
Statut	✓ OK

Nom	2.2 Importation de musique via glisser/déposer (sans métadonnées ID3)
User story	S2 : Importation de musique
Situation	<p>Étant donné que je suis un utilisateur connecté</p> <p>Quand je sélectionne dans l'explorateur de fichiers de mon système d'exploitation les fichiers de la bande son du jeu <i>Firewatch</i> (présente dans l'équipement de test)</p> <p>Alors, un nouvel album nommé « Importé le [date du jour] à [heure actuelle] » ayant comme artiste « <i>Artiste inconnu</i> » est créé. Cet album contient 19 morceaux ayant tous un titre compris entre « <i>Titre1</i> » et « <i>Titre19</i> ». La photo de couverture qui est affichée correspond à une photo de couverture neutre (grise).</p>
Résultats obtenus	Un nouvel album s'ouvre : il s'appelle « Importé le 15/05/2019 à 09:23 ». Il a « <i>Artiste inconnu</i> » comme nom d'artiste et il contient 14 morceaux nommés <i>Titre1</i> à <i>Titre19</i> .
Statut	✓ OK

Nom	3.1 Affichage des albums
User story	S4 : Affichage des albums
Situation	<p>Étant donné que je suis un utilisateur connecté</p> <p>Quand je me rends sur la page d'accueil</p> <p>Alors tous les albums que j'ai importés sur Interlude sont visibles, avec leur photo de couverture, leur titre et le nom d'artiste.</p>
Résultats obtenus	Les albums liés à mon compte s'affichent.
Statut	✓ OK

Nom	3.2 Affichage des albums : recherche par titre de musique
User story	S5 : Recherche de musique
Situation	<p>Étant donné que je suis un utilisateur connecté qui a importé dans sa bibliothèque musicale un titre nommé « Loud Places (feat. Romy) »</p> <p>Quand je me rends sur la page d'accueil et que j'écris « loUd pl » dans la barre de recherche</p> <p>Alors les résultats de la recherche contiennent le titre « Loud Places (feat. Romy) ».</p>
Résultats obtenus	Le morceau « Loud Places (feat. Romy) » est visible dans la section « Morceaux » des résultats de recherche.
Statut	✓ OK

Nom	3.3 Affichage des albums : recherche par titre d'album
User story	S5 : Recherche de musique
Situation	<p>Étant donné que je suis un utilisateur connecté qui a importé dans sa bibliothèque musicale un album nommé « In Colour »</p> <p>Quand je me rends sur la page d'accueil et que j'écris « colo » dans la barre de recherche</p> <p>Alors les résultats de la recherche contiennent l'album « In Colour ».</p>
Résultats obtenus	L'album <i>In Colour</i> apparaît bien dans les résultats de la recherche.
Statut	✓ OK

Nom	4.1 Affichage d'un album
User story	S6 : Affichage d'un album
Situation	<p>Étant donné que je suis un utilisateur connecté qui a importé un album dans sa bibliothèque musicale</p> <p>Quand je clique sur cet album dans la page principale</p> <p>Alors une interface avec le nom de l'album et de l'artiste, la photo de couverture et la liste des titres qui composent l'album s'affiche.</p>
Résultats obtenus	Une fenêtre modale s'ouvre contenant tous les éléments mentionnés plus haut.
Statut	✓ OK

Nom	4.2 Modification d'un album
User story	S7 : Modification d'un album
Situation	<p>Étant donné que je suis un utilisateur connecté sur la page d'un album</p> <p>Quand je clique sur le bouton « Modifier » de cette page, que j'écris « Mon super album » dans le champ <i>Titre de l'album</i> et « The Dø » dans le champ <i>Nom de l'artiste</i> puis que je clique sur « OK »</p> <p>Alors la page de l'album s'affiche avec « Mon super album » comme nom et « The Dø » comme auteur. Si je ferme la fenêtre de l'album, je peux voir que ces informations ont été également mises à jour dans la liste des albums.</p>
Résultats obtenus	On voit la fenêtre d'album avec le nouveau nom et le nouveau titre. Le changement est également visible en arrière-plan, dans la liste des albums.
Statut	✓ OK

Nom	4.3 Modification d'un morceau
User story	S8 : Modification d'un morceau
Situation	<p>Étant donné que je suis un utilisateur connecté sur la page d'un album</p> <p>Quand je clique sur le bouton « Modifier » de cette page, que j'écris « Mon super morceau » dans le champ <i>Nom</i> du premier morceau de cet album et « 42 » dans le champ <i>Nº de piste</i> puis que je clique sur « OK »</p> <p>Alors la page de l'album s'affiche à nouveau. Le premier morceau est déplacé tout à la fin de la liste des morceaux (comme son numéro de piste a changé) et son titre est devenu « Mon super morceau ».</p>
Résultats obtenus	La page d'album s'affiche, avec à la fin de la liste des morceaux « 42 Mon super morceau ».
Statut	✓ OK

Nom	4.4 Suppression d'un morceau
User story	S9 : Suppression d'un morceau
Situation	<p>Étant donné que je suis un utilisateur connecté sur la page de modification d'un album</p> <p>Quand je clique sur le bouton « Supprimer » à côté d'un morceau puis sur « OK »</p> <p>Alors la page de l'album s'affiche à nouveau, mais sans le morceau que je viens de supprimer. Le fichier MP3 est également supprimé du serveur.</p>
Résultats obtenus	Le morceau n'est plus visible sur la page de l'album ni dans les résultats de recherche.
Statut	✓ OK

Nom	4.5 Suppression d'un album
User story	S10 : Suppression d'un album
Situation	<p>Étant donné que je suis un utilisateur connecté sur la page de modification d'un album</p> <p>Quand je clique sur le bouton « Supprimer » à côté du nom de l'album</p> <p>Alors l'album et les morceaux qui le composent disparaissent de ma bibliothèque musicale et les fichiers MP3 des morceaux sont supprimés.</p>
Résultats obtenus	Après avoir cliqué sur le bouton <i>Supprimer</i> dans la fenêtre de confirmation, l'album et ses morceaux disparaissent totalement de l'interface : dans la bibliothèque musicale, les résultats de recherche, le lecteur et la liste d'attente.
Statut	✓ OK

Nom	5.1 Lecture d'un morceau
User story	S12 : Lancement de la lecture d'un morceau
Situation	<p>Étant donné que je suis un utilisateur connecté sur la page d'un album</p> <p>Quand je clique sur le bouton « Jouer » à droite du nom d'un morceau</p> <p>Alors la lecture de ce morceau est lancée et il est visible dans le lecteur.</p>
Résultats obtenus	Le nom, l'artiste, l'album, la couverture et la longueur du morceau sont visibles dans le lecteur et remplacent le morceau qui était alors en cours de lecture.
Statut	✓ OK

Nom	6.1 Lecture d'un album
User story	S11 : Lancement de la lecture d'un album
Situation	<p>Étant donné que je suis un utilisateur connecté sur la page d'un album</p> <p>Quand je clique sur le bouton « Jouer » dans l'en-tête de l'album</p> <p>Alors la lecture du premier morceau de l'album est lancée et il est visible dans le lecteur. Les morceaux suivants sont placés tout en haut de la liste d'attente.</p>
Résultats obtenus	Le premier morceau est directement lancé dans le lecteur. Les morceaux suivants sont visibles tout en haut de la liste d'attente.
Statut	✓ OK

Nom	7.1 Mise en liste d'attente d'un morceau à la suite
User story	S13 : Mise en liste d'attente d'un morceau (à la suite et plus tard)
Situation	<p>Étant donné que je suis un utilisateur connecté et qu'un morceau est en cours de lecture</p> <p>Quand je me rends sur la page d'un album et que j'appuie sur le bouton "jouer à la suite" à droite d'un morceau</p> <p>Alors le morceau en question est ajouté au début de la liste d'attente et est visible tout en haut de celle-ci.</p>
Résultats obtenus	Le morceau est visible tout en haut de la liste d'attente. Lorsque le morceau actuel se termine, c'est le morceau que nous venons de mettre en liste d'attente qui prend sa place.
Statut	✓ OK

Nom	7.2 Mise en liste d'attente d'un morceau plus tard
User story	S13 : Mise en liste d'attente d'un morceau (à la suite et plus tard)
Situation	<p>Étant donné que je suis un utilisateur connecté et qu'un morceau est en cours de lecture</p> <p>Quand je me rends sur la page d'un album et que j'appuie sur le bouton "jouer plus tard" à droite d'un morceau</p> <p>Alors le morceau en question est ajouté à la fin de la liste d'attente et est visible tout en bas de celle-ci.</p>
Résultats obtenus	Le morceau est visible tout en bas de la liste d'attente. Lorsque la lecture du morceau actuel et de tous les morceaux qui étaient précédemment dans la liste d'attente est terminée, la lecture du morceau que nous avons mis en liste d'attente commence.
Statut	✓ OK

Nom	8.1 Mise en liste d'attente d'un album à la suite
User story	S14 : Mise en liste d'attente d'un album (à la suite et plus tard)
Situation	<p>Étant donné que je suis un utilisateur connecté et qu'un morceau est en cours de lecture</p> <p>Quand je me rends sur la page d'un album et que j'appuie sur le bouton "jouer à la suite" à droite du titre de l'album</p> <p>Alors tous les morceaux de l'album sont ajoutés au début de la liste d'attente et sont visibles tout en haut de celle-ci.</p>
Résultats obtenus	Tous les morceaux de l'album sont visibles en haut de la liste d'attente.
Statut	✓ OK

Nom	8.2 Mise en liste d'attente d'un album plus tard
User story	S14 : Mise en liste d'attente d'un album (à la suite et plus tard)
Situation	<p>Étant donné que je suis un utilisateur connecté et qu'un morceau est en cours de lecture</p> <p>Quand je me rends sur la page d'un album et que j'appuie sur le bouton "jouer plus tard" à droite du titre de l'album</p> <p>Alors tous les morceaux de l'album sont ajoutés à la fin de la liste d'attente et sont visibles tout en bas de celle-ci.</p>
Résultats obtenus	Tous les morceaux de l'album sont visibles en bas de la liste d'attente.
Statut	✓ OK

Nom	9.1 Bouton pause
User story	S15 : Utilisation du lecteur musical
Situation	<p>Étant donné que je suis un utilisateur connecté et qu'un morceau est en cours de lecture</p> <p>Quand j'appuie sur le bouton "pause" du lecteur</p> <p>Alors la lecture du morceau s'interrompt. Le bouton "pause" devient un bouton "play".</p>
Résultats obtenus	La lecture s'arrête. Le morceau n'est plus audible et la barre de progression ne bouge plus. Le bouton "play" a maintenant une icône "pause".
Statut	✓ OK

Nom	9.2 Bouton play
User story	S15 : Utilisation du lecteur musical
Situation	<p>Étant donné que je suis un utilisateur connecté et qu'un morceau est présent dans le lecteur mais en pause</p> <p>Quand j'appuie sur le bouton "play" du lecteur</p> <p>Alors la lecture du morceau reprend là où elle s'était interrompue. Le bouton "pause" devient un bouton "play".</p>
Résultats obtenus	<i>Pas encore implémenté</i>
Statut	✓ OK

Nom	10.1 Passage au morceau suivant (automatique)
User story	S15 : Utilisation du lecteur musical
Situation	<p>Étant donné que je suis un utilisateur connecté, qu'un morceau est en cours de lecture et que des morceaux sont présents dans la liste d'attente</p> <p>Quand la lecture du morceau actuelle est finie</p> <p>Alors la lecture du morceau suivant dans la liste d'attente démarre. Ce morceau est alors retiré de la liste d'attente.</p>
Résultats obtenus	Le premier morceau de la liste d'attente disparaît de la liste et sa lecture commence. Ses informations (titre, artiste, album, couverture, durée...) s'affichent dans l'interface du lecteur.
Statut	✓ OK

Nom	10.2 Passage au morceau suivant (automatique) : pas de morceau suivant
User story	S15 : Utilisation du lecteur musical
Situation	<p>Étant donné que je suis un utilisateur connecté, qu'un morceau est en cours de lecture et qu'aucun morceau n'est présent dans la liste d'attente</p> <p>Quand la lecture du morceau actuelle est finie</p> <p>Alors le lecteur se vide (aucun morceau actuel n'est affiché) et la lecture s'arrête.</p>
Résultats obtenus	La partie du lecteur affichant les informations sur le morceau en cours est masquée.
Statut	✓ OK

Nom	10.3 Passage au morceau suivant (manuel)
User story	S15 : Utilisation du lecteur musical
Situation	<p>Étant donné que je suis un utilisateur connecté, qu'un morceau est en cours de lecture et que des morceaux sont présents dans la liste d'attente</p> <p>Quand j'appuie sur le bouton "suivant" du lecteur</p> <p>Alors la lecture du morceau suivant dans la liste d'attente démarre. Ce morceau est alors retiré de la liste d'attente.</p>
Résultats obtenus	Le premier morceau de la liste d'attente disparaît de la liste et sa lecture commence. Ses informations (titre, artiste, album, couverture, durée...) s'affichent dans l'interface du lecteur.
Statut	✓ OK

Nom	10.4 Passage au morceau suivant (manuel) : pas de morceau suivant
User story	S15 : Utilisation du lecteur musical
Situation	<p>Étant donné que je suis un utilisateur connecté, qu'un morceau est en cours de lecture et qu'aucun morceau n'est présent dans la liste d'attente</p> <p>Quand j'appuie sur le bouton "suivant" du lecteur</p> <p>Alors le lecteur se vide (aucun morceau actuel n'est affiché) et la lecture s'arrête.</p>
Résultats obtenus	La partie du lecteur affichant les informations sur le morceau en cours est masquée.
Statut	✓ OK

Nom	11.1 Mise à jour de la barre de progression dans le lecteur
User story	S15 : Utilisation du lecteur musical
Situation	<p>Étant donné que je suis un utilisateur connecté et qu'un morceau est en cours de lecture</p> <p>Quand la progression de la lecture avance au fil du temps</p> <p>Alors la progression dans la barre de progression est mis à jour (la barre est de plus en plus remplie). Les indicateurs textuels indiquant le temps joué et le temps restant sont également mis à jour sur le format "[m]:[ss]".</p>
Résultats obtenus	La barre de progression avance vers la droite. Les indicateurs de progression indiquent "0:09" (partie lue du morceau) et "-1:02" (partie à lire du morceau).
Statut	✓ OK

Nom	11.2 Clic sur la barre de progression dans le lecteur
User story	S15 : Utilisation du lecteur musical
Situation	<p>Étant donné que je suis un utilisateur connecté et qu'un morceau est en cours de lecture</p> <p>Quand je clique au milieu de la barre de progression dans le lecteur</p> <p>Alors la lecture du morceau reprend au milieu de celui-ci et les indicateurs de progression sont mis à jour.</p>
Résultats obtenus	La lecture du morceau reprend au milieu de la musique. Les indicateurs de progression indiquent "0:35" et "-0:35".
Statut	✓ OK

Nom	12.1 Affichage de la landing page
User story	S16 : Affichage de la landing page
Situation	<p>Étant donné que je suis un utilisateur non connecté</p> <p>Quand je me rends sur le site</p> <p>Alors une landing page s'affiche. Cette page me présente brièvement le projet et contient un bouton "Se connecter à Twitter" qui permet de se connecter au site.</p>
Résultats obtenus	La page affiche : « Bienvenue dans Interlude / Écoutez votre musique depuis votre navigateur » et contient un bouton « Se connecter à Twitter ».
Statut	✓ OK

Nom	13.1 Respect du style de guide Airbnb
User story	S17 : Vérification du guide de style Airbnb
Situation	<p>Étant donné que je suis un développeur du projet</p> <p>Quand j'exécute la commande <code>npm run lint</code> dans le répertoire du projet</p> <p>Alors, aucune infraction au guide de style ne m'est signalée.</p>
Résultats obtenus	<pre>> interlude@1.0.0 lint C:\Users\Nicolas\Documents\interlude > eslint —ext .js,.vue assets</pre>
Statut	✓ OK <i>Note sur le jeudi 16 mai : le test n'a pas passé en raison car 3 lignes de codes étaient indentées un niveau trop haut dans le fichier App.vue. Ce problème a été corrigé dès le début du jour suivant.</i>

Évolution des tests

Nº test	J0 ma. 7	J1 me. 8	J2 je. 9	J3 lu. 13	J4 ma. 14	J5 me. 15	J6 je. 16	J7 lu. 20	J8 ma. 21	J9 me. 22	J10 je. 23
1.1	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
1.2	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
1.3	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
1.4	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
1.5	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
2.1	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓	✓
2.2	✗	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓
3.1	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
3.2	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓
3.3	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
4.1	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓
4.2	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓	✓
4.3	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓	✓
4.4	✗	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓
4.5	✗	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓
5.1	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓
6.1	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓
7.1	✗	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓
7.2	✗	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓
8.1	✗	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓
8.2	✗	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓
9.1	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓
9.2	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓
10.1	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓
10.2	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓
10.3	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓
10.4	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓
11.1	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓
11.2	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓
12.1	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
13.1	✗	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓

✗ KO

✓ OK

Conclusion

Difficultés rencontrées

Je n'ai pas rencontré de problème réellement bloquant au cours de mon projet. Les imprévus principaux auxquels j'ai dû faire face ont été :

- *L'indicateur de durée du morceau en cours de lecture affiche Infinity sous Safari, et pas immédiatement la bonne valeur dans les autres navigateurs.*
→ J'ai pu corriger ce bug en implémentant côté PHP les requêtes HTTP Range, qui permettent au navigateur de télécharger des ressources comme des fichiers audio de manière fractionnée. Je suis particulièrement content d'avoir pu faire face à ce problème, car il m'a permis d'en apprendre plus sur les possibilités qu'offre le protocole HTTP.
- *Les utilisateurs peuvent importer des musiques dans certains cas spéciaux : par exemple, il se peut qu'ils sélectionnent des fichiers audio appartenant à des albums différents. Il arrive aussi que certains morceaux du même album aient des noms d'artistes différents, en cas de collaboration entre artistes : dans ce cas, il ne faut pas que cet artiste soit utilisé pour l'album en entier.*
→ Ce problème n'était pas difficile dans le sens où j'ai dû faire beaucoup de recherches pour savoir comment le corriger : la solution était de changer la manière dont le traitement des fichiers importés est fait côté serveur. Cette solution était juste longue à mettre en place, car elle nécessitait un long réusinage de cette fonctionnalité pour classer les morceaux par album, puis trier les noms d'artiste par fréquence d'apparition.
- *La librairie getID3(), que je souhaitais utiliser car elle était la meilleure façon de réaliser l'une des fonctionnalités demandées par l'énoncé, n'est disponible que sous licence GPL, ce qui m'oblige à publier le projet sous licence GPL également. Je ne peux pas prendre ce choix moi-même car je ne suis pas le propriétaire du projet (l'école de métiers l'est).*
→ Ce problème est assez éloigné de ceux que les informaticiens doivent résoudre en temps normal, car il est plus d'ordre juridique que technique. Heureusement, j'ai pu trouver une solution qui convient à toutes les parties : j'utilise getID3() sous licence LGPL (que l'auteur de la librairie propose également aux côtés de la licence GPL), ce qui me permet de l'utiliser, même si le logiciel est propriétaire. J'ai dû adapter mon code afin que le programme puisse fonctionner lorsque getID3() n'est pas installé afin de satisfaire les obligations de la licence.

Variantes de solutions et choix

Lors de la réalisation de mon projet, j'ai dû faire plusieurs choix concernant la manière de le réaliser. Avec le recul nécessaire, je suis satisfait des décisions que j'ai prises : elles ont permis de mener le projet à bon port, en prenant en compte sa taille, le temps à disposition mais aussi l'évolutivité. Les différentes variantes explorées et les choix faits sont expliqués dans mon journal de bord et dans le rapport TPI pour les choix principaux.

Améliorations possibles

Architecture technique

La structure actuelle de l'application fonctionne bien : la structure du projet est déjà organisée de manière professionnelle, et la partie interface est déjà presque totalement séparée de la partie serveur (le travail à faire pour les séparer entièrement serait minime). Cependant, quelques améliorations pourraient être réalisées si l'application venait à grandir en nombre de fonctionnalités et d'utilisateurs.

Côté front-end (JavaScript), l'architecture actuelle où les données principales de l'interface sont toutes stockées dans le composant `App.vue` fonctionne bien car la taille de l'application est modeste. Cependant, si de nombreuses fonctionnalités supplémentaires venaient à être développées, le projet gagnerait à utiliser un gestionnaire d'état central (*state management pattern*) tel que [Vuex](#) : cela permettrait de clarifier où sont stockées les données et comment elles peuvent être modifiées. S'il fallait développer de nouvelles pages, il serait utile d'utiliser un "routeur" tel que [Vue Router](#) qui permettrait de changer automatiquement l'URL affichée en fonction de la page visitée, et ce sans charger de page entièrement nouvelle et donc sans interrompre la lecture audio. Cela permettrait à un utilisateur d'utiliser les boutons *Précédent* et *Suivant* de son navigateur, et de sauvegarder un lien vers une partie précise de l'interface.

Côté back-end (PHP), il serait peut-être nécessaire d'intégrer le site à un système de stockage externe (de type [Amazon S3](#)) si le site venait à avoir beaucoup d'utilisateurs et donc beaucoup de fichiers hébergés.

Fonctionnalités

Il est clair que dans une application qui peut être aussi complexe qu'un lecteur musical, de nombreuses fonctionnalités pourraient être ajoutées. Sur le court terme, les améliorations suivantes concernant des fonctionnalités déjà existantes pourraient être réalisées :

- Ajouter une fonctionnalité permettant de modifier (par exemple réordonner) les éléments existants de la liste d'attente
- Ajouter une fonctionnalité permettant de modifier après importation l'image de couverture d'un album
- Ajouter une fonctionnalité permettant de déplacer un morceau vers un autre album
- Adapter l'interface actuellement existante pour qu'elle puisse fonctionner sur un smartphone (taille d'écran différente), ou une tablette (certains éléments d'interfaces actuels s'affichent lorsque le pointeur passe au dessus d'un élément : ils devraient être adaptés à un mode d'utilisation basé sur le toucher)

Sur le plus long terme, de nouvelles fonctionnalités majeures pourraient être implémentées sur Interlude. Voici les idées que j'ai eues pendant le déroulement du projet :

- Ajouter une fonctionnalité permettant de partager en temps réel la musique que l'on écoute : par exemple, pour pouvoir écouter la même musique que ses collègues dans un bureau
- Rendre Interlude compatible avec d'autres systèmes de connexion (par exemple un système classique basé sur un e-mail et un mot de passe, ou basé sur d'autres services comme Google)
- Ajouter un système permettant de sauvegarder des playlists
- Transformer le projet en une Progressive Web App (PWA) qui permettrait à l'utilisateur d'enregistrer des morceaux pour une écoute hors connexion
- Ajouter un système de conversion de fichiers audio pour qu'un fichier importé dans un navigateur soit utilisable sur toutes les plateformes, même si d'autres plateformes ne supportent pas le format du fichier importé
- Internationaliser l'interface pour permettre à plus de personnes d'utiliser le projet
- On m'a également suggéré que cela pourrait être pratique qu'Interlude puisse également afficher les bibliothèques musicales à partir de services de streaming comme Spotify ou Apple Music, en utilisant leurs APIs. Cela permettrait d'avoir un seul site où l'on pourrait avoir toute notre bibliothèque musicale, y compris les morceaux qui parfois sont exclusifs à une plateforme. Il ne doit pas avoir beaucoup d'utilisateurs qui ont un abonnement sur plusieurs plateformes, mais cela pourrait être un produit de niche intéressant.

Bilan personnel

J'ai beaucoup apprécié travailler sur Interlude : j'adore créer des projets web, et c'est un plaisir d'avoir deux de mes passions (la musique et le développement web) comme sujet pour mon travail final de CFC. Au final, c'est un plaisir de pouvoir utiliser le projet en tant qu'utilisateur depuis son navigateur. Rédiger une documentation technique complète d'un projet était également un nouveau défi à relever, et je suis très satisfait du résultat.

Remerciements

Je tiens à remercier :

- Mme Travnjak pour son accompagnement durant mon TPI
- Mme Mota qui nous a donné des informations précieuses sur la rédaction de documentation et l'organisation de travail pendant ses cours
- Adamo Cannone et William Dubelly pour leurs relectures attentives

Annexes

Bibliographie

Lors du déroulement de mon projet, j'ai utilisé les ressources suivantes pour obtenir de l'aide technique :

- MDN Web Docs, une référence complète pour les technologies web (HTML, CSS et JavaScript) soutenue par la fondation Mozilla : <https://developer.mozilla.org/en-US/>
- La documentation officielle de PHP : <https://www.php.net/docs.php>
- Le site de questions/réponses Stack Overflow : <https://stackoverflow.com>. Les réponses suivantes m'ont particulièrement aidées :
 - Réponse #7324564 par *ddinchev* concernant les en-têtes HTTP à envoyer pour mettre en cache une ressource : <https://stackoverflow.com/a/7324564/4652564>
 - Réponse #9658844 par *Lloyd* concernant la prise en charge de la fonctionnalité *HTTP Range Requests* : <https://stackoverflow.com/a/9658844/4652564>
- Le style de guide JavaScript d'Airbnb : <https://github.com/airbnb/javascript>
- La documentation officielle des différentes librairies que j'ai utilisées :
 - Documentation officielle de Vue.js : <https://vuejs.org/v2/guide/>
 - Exemples d'utilisation de la librairie *getID3()* :
<https://github.com/JamesHeinrich/getID3/tree/master/demos>
 - Tutoriel de TwitterOAuth pour la connexion : <https://twitteroauth.com/redirect.php>
 - Documentation d'ESLint : <https://eslint.org/docs/rules/>
 - Instructions d'installation de Laravel Mix :
<https://laravel-mix.com/docs/4.0/installation>
- Styling Cross-Browser Compatible Range Inputs with Sass / SCSS sur GitHub (code permettant d'ajouter des styles personnalisés à l'élément HTML `<input type="range">`) : <https://github.com/darlanrod/input-range-scss> Annexes

Glossaire

Termes métier

Album (Anglais : <i>album</i>)	Ensemble de morceaux, qui possède aussi un titre, un artiste (nom de l'auteur) et une photo de couverture.
Bibliothèque musicale (Anglais : <i>music library</i>)	L'ensemble des albums et des morceaux appartenant à un utilisateur.
Landing page	Page qui s'affiche lorsqu'un utilisateur non connecté arrive sur le site.
Lecteur (Anglais : <i>player</i>)	Composant d'interface d'Interlude qui permet à l'utilisateur d'écouter de la musique.
Liste d'attente (Anglais : <i>waitlist</i>)	Liste des morceaux qui seront joués à la fin du morceau actuel.
Morceau (Anglais : <i>song</i>)	Une piste musicale, associée à un fichier audio, qui fait partie d'un album. Un morceau a un titre et un numéro de piste dans l'album.
Utilisateur (Anglais : <i>user</i>)	Personne qui utilise Interlude et possède sa propre bibliothèque musicale sur le service. Un utilisateur est authentifié au moyen d'un compte Twitter.

Termes techniques

Back-end	Partie d'un logiciel qui concerne tout ce qui se passe du côté du serveur (par exemple : bases de données, authentification...).
Composant Vue.js	En utilisant du framework JavaScript Vue.js : morceau d'interface utilisateur réutilisable, qui possède notamment un état (données), des paramètres (propriétés) et gère lui-même son apparence visuelle (feuille de style et template HTML).
Framework (ou <i>cadriel</i>)	Ensemble important de briques logicielles sur lequel on peut se baser pour créer un logiciel informatique. Exemple de frameworks : Laravel (framework PHP), Vue.js (framework JavaScript front-end).
Front-end	Partie d'un logiciel qui concerne tout ce qui se passe du côté de l'utilisateur final (par exemple : apparence du logiciel, traitements de données effectués directement sur l'ordinateur du client).
Librairie (ou <i>bibliothèque</i>)	Brique logicielle externe au projet conçue pour être utilisée par d'autres développeurs.
Métadonnées ID3	Données présentes dans un fichier qui permettent d'obtenir plus d'informations sur son contenu. Un fichier MP3 peut par exemple contenir des métadonnées pour le nom du morceau, de l'album et de l'artiste, et l'image qui sert de couverture à l'album.
OAuth	Protocole permettant à un service d'autoriser l'accès aux données de ses utilisateurs à un autre service. Par exemple, le protocole OAuth permet à un utilisateur d'autoriser à Interlude l'accès aux données de son compte Twitter.

Résumé du TPI



Interlude

SITUATION DE DÉPART Pour valider la formation d'Informaticien-ne CFC, les apprentis doivent effectuer un "Travail pratique individuel" (TPI) d'une durée de 88 heures (11 jours), sur la base d'un énoncé imposé. Le projet qui m'a été attribué est une application web permettant aux utilisateurs d'écouter leur musique depuis un navigateur. Les utilisateurs doivent pouvoir se connecter sur le site avec un compte Twitter et importer leur musique sur le serveur. Les métadonnées ID3 présentes dans les fichiers audio téléchargés doivent être automatiquement extraites pour que les informations concernant le morceau et l'album soient remplies sans effort. D'autres fonctionnalités devaient être implémentées pour que les utilisateurs puissent rechercher de la musique et modifier le contenu de leur bibliothèque musicale.

MISE EN ŒUVRE Le projet a été réalisé en employant le framework JavaScript front-end Vue.js, ainsi que PHP et MySQL côté serveur. L'application prend la forme d'une page unique, ce afin que la lecture de la musique ne soit pas interrompue pendant son utilisation : les données sont chargées par des requêtes AJAX. En outre, la méthodologie dite de *planification en 6 étapes* a été utilisée tout au long du projet pour organiser le travail. Les différents points du cahier des charges ont été transformés en *user stories* permettant d'exprimer le besoin du point de vue des utilisateurs finaux, et un protocole de tests précis a été rédigé pour m'assurer de la conformité de l'implémentation par rapport à l'énoncé. Un journal de bord a été quotidiennement tenu à jour pour détailler le déroulement du projet, lister les imprévus et problèmes auxquels j'ai dû faire face, et expliquer les différents choix techniques effectués.

RÉSULTATS Les utilisateurs finaux ont à leur disposition une application web fluide et simple d'utilisation. L'intégralité des spécifications du cahier des charges a pu être implémentée. Le code JavaScript respecte le guide de style d'Airbnb, et cette conformité est vérifiée avec l'analyseur statique (*linter*) ESLint. Une documentation technique complète a également été rédigée.

Énoncé





République et canton de Genève
Département de l'instruction publique, de la culture et du sport
**Office pour l'orientation,
la formation professionnelle et continue**



Travail pratique individuel (TPI)
Informaticien-ne CFC
Dossier d'inscription et description du travail

Candidat : Nom : ETTLIN <hr/> Prénom : Nicolas <hr/> Téléphone : +41 78 759 96 16 <hr/> E-Mail : nicolas.ettlin@icloud.com	Entreprise formatrice : Société : CFPT-Informatique <hr/> Adresse : Ch. Gérard-de-Ternier 10, 1213 Petit-Lancy <hr/> Téléphone : 022 388 87 28 <hr/> Formatrice : Jasmina Travnjak <hr/> Tél direct : +41 76 693 63 65 <hr/> E-Mail : jasmina.travnjak@edu.ge.ch
---	--

Titre du travail : Interlude
Domaine : <input checked="" type="checkbox"/> Développement d'applications <input type="checkbox"/> Informatique d'entreprise <input type="checkbox"/> Technique des systèmes
Durée du travail (comprise entre 70h et 90h) : <u>88</u> Date de début souhaitée : <u>7 mai 2019</u>
Horaire hebdomadaire du travail : horaire : 7h30-11h40 / 12h40-16h45 <input checked="" type="checkbox"/> lundi _____ <input checked="" type="checkbox"/> mardi _____ <input checked="" type="checkbox"/> mercredi _____ <input checked="" type="checkbox"/> jeudi _____ <input type="checkbox"/> vendredi _____
Lieu où se déroule le TPI : CFPT, Bâtiment Rhône, Salle R106

Résumé du travail : Interlude est un lecteur musical fonctionnant à l'intérieur d'un navigateur web. Cette webapp permet d'importer sa musique sur un serveur externe afin de pouvoir y accéder depuis n'importe quel ordinateur.

RAPPEL : Il est interdit au candidat, de prendre connaissance de l'énoncé du travail de TPI, avant le début de celui-ci. L'énoncé lui sera transmis par les experts, par mail, le matin du 1^{er} jour du TPI avant 8h00.	Devoir d'examen défini. L'entreprise formatrice : Lieu : _____ Date : _____ Signature : _____
--	--

En annexe à la présente inscription il sera joint une description du projet. Le dossier sera ensuite validé par le collège des experts qui désignera un (et dans ce cas le chef expert participera à la présentation) ou deux d'entre eux pour le suivi du déroulement du travail. L'acceptation de celui-ci sera confirmée par leurs signatures sur la feuille d'évaluation du TPI.

Rappel : selon l'article 3.1 des « Dispositions d'exécution pour la procédure de qualification avec examen final, du 1er novembre 2013 », Le supérieur élabore *le mandat*. Tous les dossiers incomplets seront automatiquement refusés.

TPI - Cahier des charges

Ce document sera connu du candidat uniquement au commencement du TPI. Il est interdit d'en communiquer le contenu au candidat avant la date de TPI convenue.

1. Titre

Interlude

2. Matériel et logiciels à disposition

- Un PC standard école avec Windows 10, 2 écrans
- Serveur Web et SGBD à choix (EasyPHP, Wamp, Laragon, Adminr, phpMyAdmin, autre)
- IDE à choix (NetBeans, Visual Studio Code, PHPStorm, autre)
- Logiciel de création de schémas (Visio, Gliffy, autre)
- Outil de versionnage de code (Git, avec dépôt distant sur Github / Bitbucket / GitLab)
- Navigateur web (Mozilla Firefox/Google Chrome)
- Logiciel de création de maquettes d'interfaces utilisateur (Sketch, Adobe XD, Proto.io, autre)
- Outil bureautique à choix pour les documents (Google Docs, MSOffice, OpenOffice)
- L'étudiant est autorisé à utiliser son matériel personnel au besoin

3. Descriptif complet du projet

3.1 Planification :

Analyse	12h
Implémentation	50h
Tests	4h
Documentation	22h
Total	88h

3.2 Méthodologie

Vous devez découper votre travail en tâches et essayer de suivre votre planning durant les 11 jours de réalisation de ce projet. Pour chaque tâche réalisée, pensez à faire des tests avant de la considérer comme terminée, et à tester également les autres fonctionnalités pour éviter les régressions.

3.3 Description de l'application

Interlude est une application web destinée à un usage privé de sa propre musique. Seule une personne connectée peut accéder à ses musiques et en ajouter, supprimer, ou modifier. Il n'y a pas de gestion de droits d'auteur ni d'administration/modération puisqu'il s'agit simplement d'écouter sa propre musique sans la partager ni la publier. L'application sera développée à l'aide des langages web suivants : PHP, HTML et Javascript.

Les fonctionnalités suivantes doivent être implémentées :

- **Connexion avec Twitter**

Pour se connecter, l'utilisateur doit utiliser son compte Twitter. Cela permet d'éviter la création de comptes de spam, tout en offrant à l'utilisateur une expérience simple et rapide de connexion, lui évitant la création d'un compte supplémentaire. La communication entre Interlude et Twitter se fait à l'aide du protocole OAuth.

Tant que l'utilisateur n'est pas connecté, il ne peut pas accéder à la bibliothèque musicale : un écran d'accueil lui présente le projet et lui demande de se connecter.

- **Affichage de la collection de musique**

Une fois connecté sur le site, l'utilisateur a accès à sa bibliothèque musicale, récupérée depuis le serveur. Ses albums s'affichent, triés par ordre d'importation (le plus récent en premier).

- **Lecture musicale**

Lorsque l'utilisateur clique sur un album, une vue détaillée s'ouvre, affichant les morceaux que cet album contient. Il est possible de planifier la lecture d'un morceau (ou de tous les morceaux de l'album) de trois façons :

- immédiatement : le morceau en cours de lecture est remplacé par le morceau sélectionné ;
- à la suite : les morceaux sont ajoutés au début de la file d'attente ;
- plus tard : les morceaux sont ajoutés à la fin de la file d'attente.

La lecture est planifiée pour la session en cours, si l'utilisateur quitte l'application, aucune trace de sa liste de lecture n'est gardée

- **Importation de fichiers audio**

L'utilisateur peut importer un ou plusieurs fichiers audio en les glissant-déposant dans la fenêtre ou en cliquant sur un bouton qui ouvre un sélecteur de fichiers. Interlude crée un album avec les morceaux ajoutés. Les informations concernant le nom de l'album, le nom de l'artiste, la pochette d'album et les numéros de piste sont extraites automatiquement des métadonnées ID3 contenues dans le fichier. Dans le cas où les morceaux ne contiennent pas de métadonnées, un nom d'album par défaut est donné avec la date du jour. Les titres sont nommés Titre1, Titre2, etc. si les métadonnées ne sont pas présentes.

- **Modification de la bibliothèque**

L'utilisateur peut après l'importation :

- modifier le titre d'un album
- modifier l'auteur d'un album
- modifier le titre d'un morceau
- modifier le numéro de piste d'un morceau
- supprimer un morceau
- supprimer un album entier et tous les morceaux qu'il contient.

- **Recherche**

Interlude contient un moteur de recherche permettant de rechercher un morceau ou un album à partir de son nom.

- **Conception web adaptative**

Interlude est conçu pour fonctionner dans un navigateur web sur un ordinateur de bureau. Le site n'a pas besoin de s'adapter aux téléphones portables et tablettes.

3.4 Modèle de données

Le candidat se base sur le modèle ci-dessous pour réaliser son travail. S'il l'estime nécessaire, il peut l'adapter avec l'accord de son supérieur. Le script SQL est fourni au candidat le jour du début du TPI par sa formatrice.

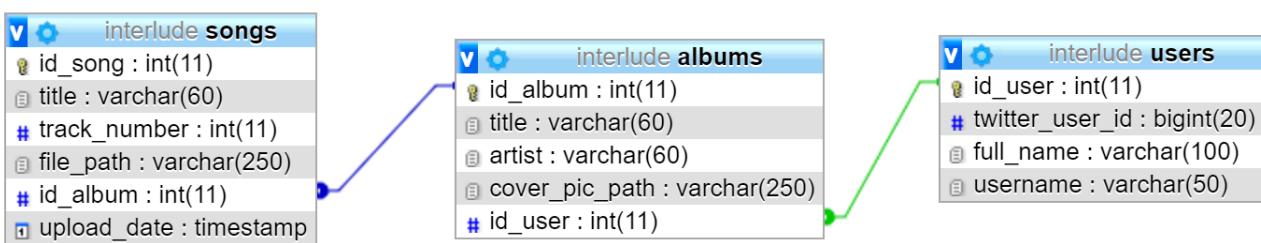


Figure 1 : Modèle physique de données

3.5 Tests de l'application

Un protocole de test est préparé et appliqué pour valider votre application. Les tests sont également réalisés au fur et à mesure du développement afin de s'assurer de la complétude des tâches effectuées et de la non régression de l'application.

4. Livrables

Pour les experts et la formatrice par email :

- Planning détaillé du projet
- Rapport de projet contenant le code source au format PDF
- Journal de bord
- Résumé du TPI (1 page A4)

Pour la formatrice uniquement :

- L'accès au repository distant du projet avec les droits de « clone »
- Un readme explicitant l'installation du projet en local
- Un dump de la base de données contenant la structure ainsi qu'un set de données de test

5. Points techniques évaluées spécifiques au projets

correspondants aux points A14 à A20 du formulaire d'évaluation

- A14. La solution est fonctionnelle (tourne sur un serveur local avec un accès à Internet)
- A15. Respect du guide de style JavaScript d'Airbnb pour le code client (<https://github.com/airbnb/javascript>)
- A16. Un utilisateur non connecté ne peut accéder qu'à la landing page
- A17. Un utilisateur peut se connecter avec son compte Twitter
- A18. Un utilisateur connecté peut parcourir sa musique et la jouer
- A19. Un utilisateur connecté peut importer de la musique
- A20. Un utilisateur connecté peut modifier sa bibliothèque musicale

Journal de bord



Journal de bord TPI • Nicolas Ettlin

J0 : mardi 7 mai 2019

Objectifs

Aujourd’hui, c’est le tout premier jour de mon TPI. L’objectif de cette journée est de prendre connaissance de l’énoncé, de l’analyser pour en extraire les différentes *user stories* qui constitueront mon *product backlog* et de rédiger mes scénarios de test qui me permettront de m’assurer du bon fonctionnement de mon projet.

Déroulement

Je commence le travail à 07:30. J’ai reçu la veille au soir l’énoncé du TPI, que j’ai lu attentivement (ce qui correspond à la première étape de la méthode en 6 étapes, *S’informer*).

J’ai quelques petites questions concernant l’énoncé (voir plus bas), mais ce ne sont que des détails et rien de bloquant. J’en parlerai à ma formatrice lorsque je la verrai aujourd’hui. En attendant, je passe à la deuxième étape, *Planifier*. Je vais maintenant séparer les différents points de l’énoncé en tâches de 4 heures (demi-journées) sur un brouillon (je le retranscrirai plus tard sur un diagramme de Gantt en bonne et due forme).

Brouillon sur Trello : [Brouillon tâches Interlude](#)

07:50 : Pour m’y retrouver de manière plus claire dans les différents jours du TPI, je vais nommer chaque jour de J0 (le jour de la planification) à J10 (dernier jour de travail). On a donc :

- J0 : mardi 7 mai 2019
- J1 : mercredi 8 mai 2019
- J2 : jeudi 9 mai 2019
- J3 : lundi 13 mai 2019
- J4 : mardi 14 mai 2019
- J5 : mercredi 15 mai 2019
- J6 : jeudi 16 mai 2019
- J7 : lundi 20 mai 2019
- J8 : mardi 21 mai 2019
- J9 : mercredi 22 mai 2019
- J10 : jeudi 23 mai 2019

08:10 : Dans la partie *Importation de fichiers audio*, l’énoncé spécifie que :

Les titres sont nommés Titre1, Titre2, etc. si les métadonnées ne sont pas présentes. Cependant, il ne spécifie pas comment il faut nommer les albums créés par défaut. "Album sans nom" ? "Album sans nom 1" ? "Album importé le 07.05.2019" ? Je vais poser la question à ma formatrice lorsque je la verrai.

08:30 : J'ai un brouillon de la planification. Il me faut encore le transformer en un *product backlog* clair et qui contient des tâches priorisées, créer un diagramme de Gantt pour pouvoir mieux visualiser ma progression, et créer un protocole de tests pour m'assurer chaque jour que les attentes ont été validées et que je n'ai pas créé de régression.

08:35 : Ma formatrice invite les élèves qu'elle prend en charge pour le TPI à la rejoindre pour une pause et pour faire un point sur les TPIs.

08:55 : Je suis de retour à ma place de travail. Je vais maintenant rédiger le product backlog à partir des grosses tâches que j'ai identifiées.

Pour la priorité des tâches, je me base sur ces quatre niveaux, issus de la méthode MoSCoW :

- **!! Must** : indispensable au projet, doit impérativement être réalisé
- **! Should** : important pour le projet
- **? Could** : utile pour le projet mais pas indispensable
- **🚫 Won't** : pas nécessaire pour le projet et donc pas prioritaire

10:25 : Ma formatrice est passée dans ma salle pour s'assurer que tout se passe bien. Elle m'a répondu pour la question suivante :

Est-ce que le *product backlog* peut contenir également des stories techniques (par exemple, « En tant que développeur... ») ?

→ Oui, c'est ce qui est régulièrement utilisé dans l'industrie. On peut par exemple en faire pour le critère A15 (respect du guide de style Airbnb).

Elle m'a également adressé une remarque sur la nomenclature que j'ai utilisée pour mes priorités : les mots anglais "could" et "won't" peuvent laisser penser que certaines tâches ne sont pas nécessaires, alors qu'elles sont toutes obligatoires comme elles sont définies dans l'énoncé. Pour remédier à ce problème, j'utilise de nouveaux noms :

- **🚫 P0 : Bloquant** → La tâche doit impérativement être effectuée avant toute autre.
- **!! P1 : Critique** → La tâche est critique pour le bon fonctionnement du projet et doit obligatoirement être réalisée.
- **! P2 : Important** → La tâche est nécessaire pour la complétion d'autres tâches et doit donc être réalisée.

- **? P3 : Secondaire →** La tâche n'est pas liée à d'autres et sa non-complétion n'entrave pas le reste du projet.

Ce système de nommage est inspiré par celui utilisé en interne chez [The Qt Company](#), entreprise dans laquelle j'avais fait un stage l'été dernier.

11:00 : Note sur mon utilisation des outils bureautiques : j'utilise [Bear](#) (un éditeur Markdown pour macOS) pour tenir mon journal de bord. Ce logiciel synchronise automatiquement mon document sur iCloud. Pour la tenue de mes documents plus formels (rapport TPI...), j'ai choisi d'utiliser [Pages](#) (qui fait partie d'iWork, la suite bureautique d'Apple). Je fais des sauvegardes de ces documents à la fin de chaque demi-journée sur Google Drive.

Dans les documents en rapport avec le projet Interlude, j'utilise  #674EA7 comme couleur principale, et la police Inter que je compte également utiliser dans l'interface utilisateur. La ressemblance entre le nom de la police et le nom du projet est totalement fortuite.

11:45 : J'ai terminé la rédaction du *product backlog*. Je prends ma pause de midi.

12:40 : Reprise du travail. Je vais maintenant faire mon diagramme de Gantt. J'ai choisi de le réaliser manuellement sous Pages au lieu d'utiliser un logiciel spécialisé comme GanttProject, car je suis plus à l'aise avec un outil bureautique standard.

14:50 : J'ai terminé le diagramme de Gantt. Je vais maintenant travailler sur le protocole de tests.

16:40 : Je n'ai pas encore terminé la rédaction des scénarios de tests, mais j'aurai le temps de le faire demain matin. J'envoie une version du document sans la partie *Test* à ma formatrice et aux experts.

Bilan

Cette journée s'est bien passée : même si je dois terminer la rédaction des scénarios de test demain, je suis satisfait du résultat. L'écart entre ce que j'avais prévu (*product backlog*, planification des tâches et rédaction des scénarios de test terminés pour la fin de J0) et la réalité (il reste encore des scénarios de test à terminer) est due à la quantité

J1 : mercredi 8 mai 2019

Objectifs

Aujourd’hui, c’est le premier jour où je vais toucher à du code. Je vais commencer par terminer mes scénarios de test, puis je vais configurer Git, Composer, la base de données, la compilation des ressources CSS et JavaScript ainsi que le *linter* pour vérifier que je respecte bien le guide de style d’Airbnb lorsque j’écris du code JavaScript.

Déroulement

07:40 : Reprise du travail. Je me remets sur les scénarios de tests.

08:40 : Pour faire les scénarios de test d’importation de musique, j’ai inclus deux albums de musique : l’un est au fichier MP3 et contient des métadonnées (ce qui me permettra de tester l’extraction des métadonnées ID3), et l’autre est au format WAV et ne contient aucune métadonnée. Pour créer cet album, j’ai utilisé le convertisseur en ligne [CloudConvert](#).

09:00 : En fait, CloudConvert fait trop bien son travail et les métadonnées ID3 sont conservées dans les fichiers convertis. Je vais faire un essai le site [tagmp3.net](#).

09:05 : Finalement, le site susmentionné n’a pas réglé mon problème : il permet bien de modifier les métadonnées, mais on ne peut pas les supprimer (et il faut le faire individuellement pour chaque morceau). J’ai par contre pu trouver une solution (grâce à <https://apple.stackexchange.com/a/16145> et <https://unix.stackexchange.com/a/105270>) avec la bonne vieille ligne de commande et elle marche à merveille :

```
# Installation de id3lib avec le gestionnaire de packages Homebrew
brew install id3lib

# Utilisation de l'option --strip pour retirer toutes les métadonnées
id3convert --strip *.mp3
```

09:30 : C’est tout bon, j’ai terminé les scénarios de test. Je vais maintenant pouvoir attaquer le projet.

10:20 : J’ai terminé la configuration du dépôt Git, de Composer et de l’autoloader (ce qui est donc la fin des tâches initialement prévues pour cette matinée).

10:30 : J’ai reçu une visite de l’un de mes experts, M. Foti. Il a vérifié que tout se passait bien pour le début du projet et m’a posé des questions sur les aspects techniques

spécifiques (connexion à Twitter avec OAuth, extraction des métadonnées ID3) de mon énoncé pour s'assurer que tout est OK pour ces points-là. Je lui ai demandé quelle était la préférence des experts pour la langue employée dans le code source, et il m'a répondu que c'est bon pour eux d'utiliser l'anglais.

10:40 : Je recommence à travailler sur la compilation des assets JavaScript.

11:30 : Tout fonctionne (SASS, Vue et JavaScript). Je profite de mon avance pour ajouter la police Inter au projet.

11:40 : Inter est installée, je pars en pause.

12:40 : Reprise du travail. Je documente le fonctionnement de Laravel Mix (ce que j'ai installé ce matin). J'ai utilisé cette librairie car c'est un moyen très simple et facilement configurable de compiler les ressources JavaScript et CSS. J'ai aussi travaillé avec Gulp avec d'autres projets, et la configuration est beaucoup moins triviale.

13:05 : J'ai terminé la documentation. Je m'attaque au guide de style.

13:25 : La configuration d'ESLint est vraiment simple en suivant la [documentation officielle](#).

13:30 : La story concernant le style de code JavaScript est terminée. Je vais maintenant m'attaquer à la base de données (S21). Pour l'utiliser, je compte utiliser une classe que j'avais créé pour un autre projet et qui permet de facilement effectuer les quatre actions de base CRUD (*create, read, update, delete*).

14:18 : En testant que ma classe fonctionne bien, j'avais rencontré un bug étrange : ma méthode `insert` créait trois enregistrements au lieu d'un seul. Après avoir tenté sans succès de déboguer le code PHP pour y trouver un problème, j'ai compris ce qui n'allait pas : le code PHP de test, que j'avais mis dans mon fichier `index.php`, était exécuté plusieurs fois car mon navigateur tentait de pré-charger la page pour rendre le chargement plus rapide. Limiter l'exécution du code aux requêtes `POST` a réglé le problème.

14:35 : Les tâches de la journée sont terminées. Je commence la *landing page*.

15:11 : J'ai téléchargé le logo de Twitter grâce à [simpleicons.org](#), qui est un répertoire en

ligne de logos de marques au format SVG.

15:45 : Derrière la *landing page*, je crée les composants de base de l'interface (structure de la page).

16:30 : J'ai terminé la structure de base de la page et la landing page. Demain, je ferai le système de login via Twitter.

Bilan

Bien que j'aie commencé la journée en terminant les scénarios de tests débutés hier, j'ai terminé les tâches prévues avec une large avance.

Lorsque j'ai fini les tâches initialement prévues, j'ai avancé sur la landing page et la structure de la page. Si cela m'arrive à nouveau de terminer en avance, je pense que cela pourrait être mieux de continuer directement sur le travail du lendemain, ce afin de m'assurer que les tâches les plus prioritaires sont rapidement réalisées.

J2 : jeudi 9 mai 2019

Objectifs

Aujourd'hui, je dois créer le système de connexion via Twitter. J'ai déjà fait l'apparence de la *landing page* hier.

Déroulement

07:30 : Je me mets au travail. Je vais commencer le système de connexion. Pour implémenter la connexion avec le service Twitter par le protocole OAuth, je vais utiliser la librairie [TwitterOAuth](#) disponible sur Composer. J'ai déjà utilisé cette librairie pour un projet de chatbot réalisé au sein d'un atelier en école de métiers.

J'ai envisagé au début d'utiliser une autre librairie, [oauth2-client](#), qui a l'avantage de fonctionner avec une [multitude de services différents](#) (dont Facebook et Google), mais celle-ci ne fonctionne malheureusement pas avec Twitter, ce qui est donc pour moi éliminatoire. Ce manque de support semble venir du fait que Twitter fonctionne avec OAuth 1.0 et non pas OAuth 2.0 ([source](#)), protocole que la librairie supporte.

08:05 : [Ce tutoriel de Tutsplus](#) semble expliquer précisément comment atteindre mon objectif, je vais donc le suivre.

11:05 : J'ai terminé tout ce qui est en rapport avec la connexion/déconnexion. J'ai juste eu un petit problème pour l'affichage de la photo de profil du compte. Twitter fournit une URL

de ce type qui renvoie la photo de profil de n'importe quel compte à partir de son nom d'utilisateur :

```
https://twitter.com/[screen_name]/profile_image?size=original
```

Cependant, les serveurs de Twitter semblent avoir été configurés pour refuser toute connexion à cette URL depuis un site extérieur, la rendant inutilisable depuis l'interface d'Interlude. J'aurais pu utiliser les APIs de Twitter pour récupérer l'URL de l'avatar et la stocker en base de données au login, mais j'ai trouvé une solution plus simple.

En effet, le site [avatars.io](#) permet de faire exactement la même chose mais sans avoir de problème lors de l'intégration à des sites externes. On se retrouve alors avec une URL de ce type :

```
https://avatars.io/twitter/[screen_name]/medium
```

[avatars.io](#) fonctionne aussi avec Facebook, Instagram et Gravatar. C'est un outil intéressant, je vais enregistrer son adresse en favoris pour d'autres projets !

12:40 : J'ai fait passer tous les tests relatifs à la connexion Twitter.

11:40 : Reprise du travail. Je vais faire l'affichage des albums. Je vais commencer par travailler sur les APIs nécessaires pour la vue.

13:53 : Le script SQL d'importation de la structure de base de données fourni par ma formatrice ne correspond pas à l'énoncé sur les types de données (certains champs, comme `songs.title` sont des `INT` au lieu de `VARCHAR`). Il manque aussi le champ `songs.upload_date`. J'ai informé ma formatrice et je modifie mes tables en conséquence.

14:15 : J'ai terminé l'API JSON qui permet de récupérer les albums de musique de l'utilisateur connecté. Je vais maintenant l'intégrer à Vue.

15:30 : CSS Grid est un outil vraiment fantastique pour certaines dispositions comme la vue de tous les albums sur la page principale d'Interlude. Le code suivant permet au navigateur de déterminer automatiquement combien d'albums il peut faire tenir sur une ligne, et d'utiliser toute la place disponible sur cette ligne pour leurs couvertures :

```
.albums-list {  
    display: grid;  
    grid-template-columns: repeat(auto-fill, minmax(200px, 1fr));  
}
```

15:40 : J'ai choisi de stocker les images des albums dans un dossier `storage` qui n'est pas dans la partie publiquement accessible du site web, et ce pour les raisons suivantes :

- Cela permet de m'assurer à 100% qu'on ne peut pas accéder aux fichiers d'un autre utilisateur, même en ayant l'URL
- Cela me permet aussi d'améliorer la sécurité du site, en me protégeant contre toutes les attaques où un utilisateur mal intentionné souhaiterait exécuter du code PHP qu'il a téléchargé en le faisant passer pour une image. Comme les fichiers réellement stockés ne sont pas accessibles par les utilisateurs via une requête HTTP, le pirate ne pourrait pas lancer son script.

i Autre note : Une fois que ce système fonctionnait, j'ai constaté qu'à chaque chargement de la page principale, toutes les couvertures se chargeaient une par une, même après plusieurs rafraîchissements. J'ai pu améliorer cette situation en indiquant au navigateur de mettre en cache la ressource. J'ai pu être aidé par [un post de StackOverflow](#) qui m'a donné la bonne séquence d'en-têtes HTTP à envoyer.

16:40 : La recherche par nom d'album fonctionne.

Bilan

Je suis très satisfait de mon travail aujourd'hui. La connexion à Twitter fonctionne très bien et c'est agréable de déjà pouvoir voir la page principale fonctionner.

J3 : lundi 13 mai 2019

Objectifs

Aujourd'hui, je vais terminer la recherche de musique que j'avais commencée jeudi dernier. La recherche d'albums fonctionne déjà, et la logique (récupération des données) derrière la recherche par titre de morceau est déjà implémentée mais il faut que je m'occupe de l'affichage des résultats. Lorsque ce sera terminé, je vais m'occuper de la tâche S6, à savoir l'affichage de la page d'un album, contenant notamment ses titres.

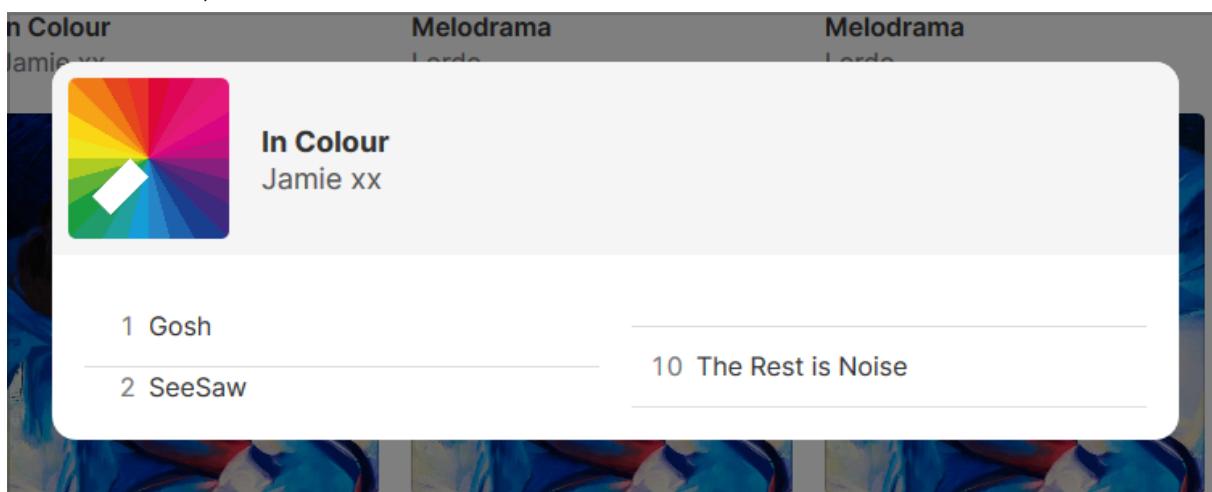
Déroulement

07:45 : Début du travail. Je vais terminer la recherche par morceaux.

09:30 : J'ai terminé l'affichage de la recherche pour les morceaux, et j'en ai profité pour créer un écran qui s'affiche lorsqu'aucun résultat n'est trouvé par la recherche pour en informer clairement l'utilisateur.

Je vais maintenant travailler sur l'affichage de la page d'un album

10:50 : Pour afficher les morceaux à l'intérieur de l'album, j'ai voulu utiliser la propriété `columns` de CSS. Celle-ci m'aurait permis de diviser ma liste de morceaux en deux colonnes facilement. Cependant, le navigateur choisit parfois de diviser un élément sur deux colonnes, comme dans le cas suivant :



Normalement, j'aurais dû pouvoir utiliser `break-inside: avoid` pour indiquer au navigateur que je souhaite que mes éléments soient gardés entier, mais malheureusement, cette propriété n'est pas encore supportée par Firefox et Chrome (en revanche, cela fonctionne sous Safari et Edge). Je vais rechercher une autre solution, probablement en utilisant CSS Grid.

Il est intéressant de noter que j'ai pu utiliser pour la première une fonctionnalité assez avancée de CSS, `font-feature-settings`. Celle-ci me permet d'activer ou de désactiver certaines options qu'une police de caractères propose. Ici, j'ai pu activer l'option `tnum` (pour *tabular numbers*) que la police Inter propose : il s'agit d'une variante dans laquelle tous les chiffres ont la même largeur. C'est utile dans mon cas pour afficher une liste où les chiffres (numéros de piste) sont alignés sur les mêmes colonnes. Voici comment j'ai pu activer l'option :

```
.track-number {  
    /* ... */
```

```
font-feature-settings: 'tnum' 1;  
}
```

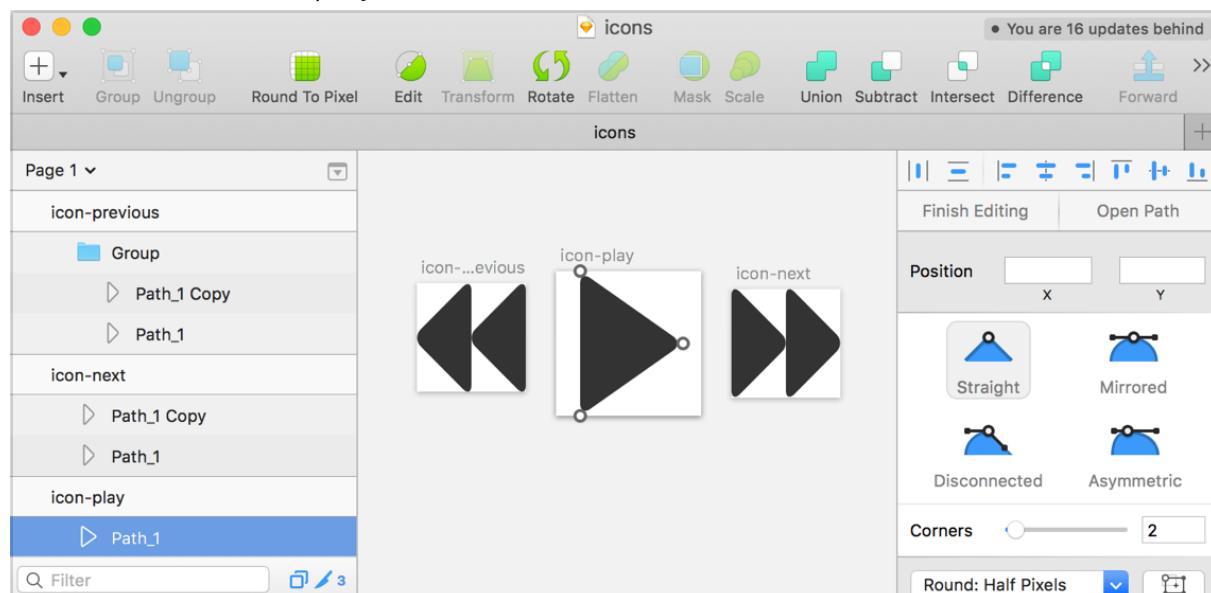
11:25 : Finalement, je suis retourné à la solution initiale, qui est d'utiliser `columns`, car je ne peux pas paramétriser Grid pour me créer automatiquement des colonnes égales en nombre d'éléments. J'ai découvert que mon code avec `columns` fonctionne si mon élément est en `block` ou en `grid` au lieu de `flex`. Je vais le passer en `grid`.

11:40 : L'affichage de la page album fonctionne. Je prends ma pause de midi.

12:40 : Je reprends le travail. Je vais passer à la tâche suivante, c'est à dire le lecteur musical.

13:05 : J'ai reçu une visite de l'une de mes experts, Mme Quarroz. Elle a vérifié mon avancement sur le travail et m'a conseillé d'être très attentif aux points relatifs à la documentation et de relier la grille. Elle m'a aussi demandé d'envoyer aux experts une version provisoire du journal de bord jeudi prochain.

13:15 : Je suis en train de faire des icônes (pause, suivant, précédent...) pour le lecteur musical. Jusqu'à présent, j'ai utilisé Adobe XD pour créer toutes les icônes que j'ai utilisées (notamment l'icône de la barre de recherche). Pour les icônes du lecteur musical, j'utilise Sketch (uniquement disponible sous macOS), car Adobe XD ne me permet pas d'arrondir les coins d'une forme que j'ai créée.



14:10 : La prise jack de mon ordinateur école ne semble pas fonctionner. Lorsque je branche mes écouteurs pour tester la lecture de la musique, le son sort toujours des hauts-

parleurs de mon ordinateur. Mes drivers sont pourtant à jour. Je vais tester au volume minimal pour l'instant.

16:45 : J'ai terminé l'affichage du lecteur, le fonctionnement de la liste d'attente et le passage au morceau d'après. On peut appuyer sur un bouton pour lancer un morceau individuellement et un album. Il me reste à implémenter :

- Les boutons pour mettre un morceau / un album en file d'attente (la logique est déjà implémentée, il ne me reste plus qu'à m'occuper de l'affichage des boutons).
- La barre de progression de lecture a encore son style de `<input type="range">` par défaut, il faut que je la modifie pour que ça ressemble plus à une barre de volume. Je vais aussi devoir faire attention à ce que les styles personnalisés fonctionnent sous tous les navigateurs.
- Peut-être ajouter un bouton pour vider la liste d'attente en cas de besoin, c'est à la fois rapide à faire mais aussi utile pour l'utilisation.

Bilan

Cette journée s'est très bien passé et je suis satisfait de mon travail. Je n'ai pas eu de souci (mis à part le détail concernant `columns`) et je vais pouvoir finaliser les dernières touches du lecteur demain. Je suis encore en avance sur le planning (j'aurais dû faire le lecteur en entier demain), mais je n'ai pas dû faire de compromis sur la documentation et la qualité du code/de l'interface pour avoir cette avance.

J4 : jeudi 9 mai 2019

Objectifs

Ce matin, je vais commencer par terminer les fonctionnalités que j'ai citées plus haut : je dois ajouter les boutons d'interface pour la mise en liste d'attente et améliorer l'apparence visuelle de la barre de progression. Ensuite, je vais poursuivre sur l'importation de musique.

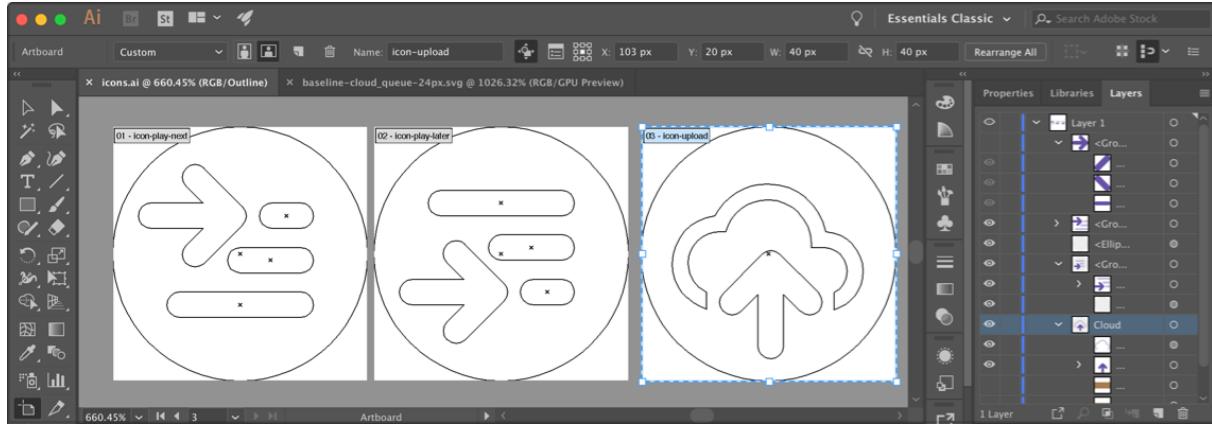
Déroulement

07:40 : c'est reparti pour une journée de travail de TPI. Je vais me mettre à faire les boutons pour mettre une musique ou un album en liste d'attente. Je dois les ajouter :

- Sur la page d'album, dans l'en-tête (pour mettre tout l'album en attente)
- Sur la page d'album, à côté de chaque morceau
- Dans les résultats de recherche, à côté de chaque morceau

Je vais commencer par créer les deux icônes pour les boutons de mise en attente.

08:05 : Pour créer les icônes dont j'avais besoin, j'ai utilisé Illustrator, ses fonctionnalités de création de formes étant plus complètes que celles disponibles sous Adobe XD et Sketch. J'en ai eu besoin pour créer l'icône de flèche nécessaire. J'en ai aussi profité pour faire l'icône *Téléverser une musique*.



09:30 : Tout ce qui est en rapport avec la liste d'attente fonctionne. J'ai aussi ajouté le bouton pour vider la liste d'attente. Je vais maintenant faire le style personnalisé pour la barre de progression.

10:00 : En installant une instance d'Interlude sur mon ordinateur personnel pour tester, je me suis rendu compte que c'est utile d'avoir des logs en cas d'erreur de connexion entre Twitter et l'instance Interlude.

11:20 : J'ai terminé le style de la barre de progression du lecteur. Pour m'aider à faire des styles qui fonctionnent sur toutes les plateformes, j'ai adapté un code que j'ai pu trouver [sur GitHub](#). Je vais commencer à travailler sur l'upload de musique.

11:35 : La colonne `albums.cover_pic_path` n'autorise pas de valeurs nulles dans la structure de base de données qui m'a été fournie. Que faut-il faire si aucune couverture n'est trouvée dans les métadonnées ? Je vais demander à ma formatrice si c'est possible de changer le modèle de données pour autoriser les valeurs nulles.

11:40 : Je prends ma pause. Je reprends mon travail sur l'importation cet après-midi.

12:40 : Je vais continuer la tâche sur l'importation de musique.

13:30 : J'ai eu une réponse de ma formatrice et c'est OK pour elle, je peux mettre à jour le modèle de données pour autoriser des valeurs nulles dans `albums.cover_pic_path`.

15:10 : J'ai aidé William Dubelly (un camarade) à corriger des problèmes liés à la gestion

de dates et heures sur son TPI.

16:40 : L'upload fonctionne, avec le bouton *Importer* et le glisser/déposer. Je vais m'occuper demain de l'extraction des métadonnées ID3.

Bilan

La fin du développement des fonctionnalités de mon TPI avance à grands pas (il ne me reste plus que l'extraction de métadonnées ID3 et la modification/suppression des albums/morceaux). J'ai de l'avance, cela va me laisser pas mal de temps pour tester en profondeur et écrire une bonne documentation.

J5 : mercredi 15 mai 2019

Objectifs

Aujourd'hui, je vais m'occuper de l'extraction des métadonnées ID3 depuis mes fichiers audio. C'est la dernière inconnue qu'il me reste dans le projet (les autres grosses inconnues étaient la connexion à Twitter, et dans une moindre mesure la gestion du lecteur audio). Ensuite, je devrai encore m'occuper de la modification et suppression dans la bibliothèque musicale, mais ça ressemble plus à du CRUD basique.

Déroulement

07:40 : C'est reparti pour une journée de travail. Je vais faire des essais d'extraction de métadonnées ID3 afin de trouver une méthode qui fonctionne bien.

08:30 : Je suis en train de faire des recherches pour trouver une bonne façon d'extraire les métadonnées. Je suis tombé sur les solutions suivantes :

- PHP propose [une série de fonctions](#) en rapport avec ID3 (comme `id3_get_tag`, mais celles-ci sont fournies par une extension, ce qui fait que je ne peux pas être sûr que cela fonctionne sur un autre serveur que le mien).
- La librairie [getID3\(\)](#) semble être très complète et répandue (plus de 600 000 téléchargements sur Packagist), mais elle est uniquement distribuée sous les licences GPL, LGPL et Mozilla MPL, ce qui semble restreindre l'utilisation de la librairie si Interlude n'est pas non plus distribuée sous une licence comme GPL.

08:50 : Je vais partir sur l'utilisation de **getID3()** sous la licence GPLv3. Contrairement à la licence GPLv3, celle-ci autorise l'utilisation de getID3() comme librairie externe (sans avoir besoin de faire passer Interlude sous licence LGPL), à condition qu'Interlude puisse fonctionner sans getID3() si nécessaire.

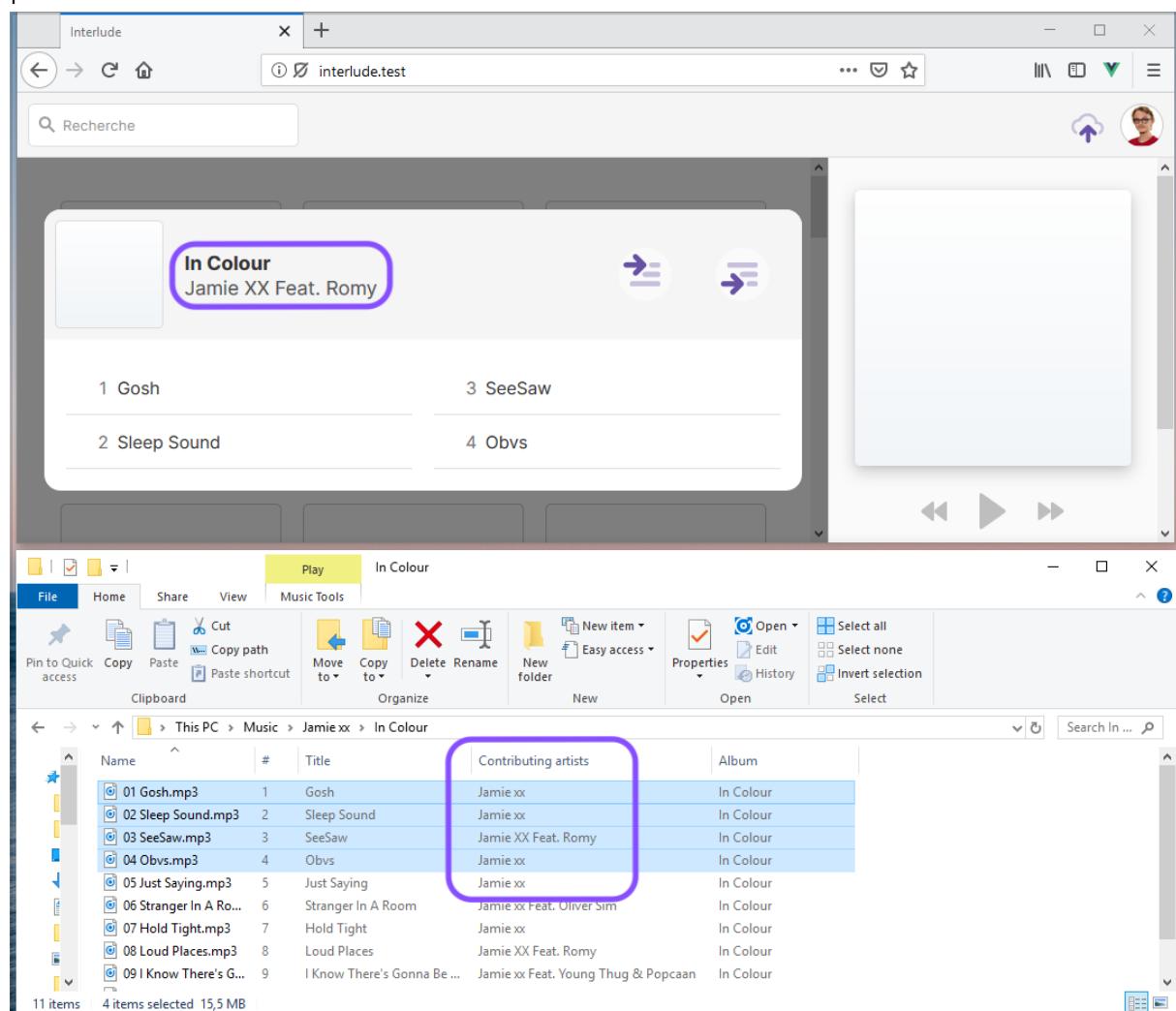
Mes sources concernant cette affirmation :

- [GNU Lesser General Public License v3.0 | Choose a License](#)
- [Can I use an LGPL-licenced library in my commercial app? - Software Engineering Stack Exchange](#)
- [Le site officiel de getID3\(\)](#)

10:00 : J'ai fait un essai de ma classe `Metadata` (celle qui va interagir avec `getID3()`) et tout est fonctionnel. J'ai ajouté une vérification qui fait échouer silencieusement la découverte de métadonnées si `getID3()` n'est pas installé, sans qu'une erreur fatale ne soit déclenchée car la classe est manquante.

Je vais maintenant intégrer ce processus à l'importation de musique sur le site.

11:20 : Un problème mineur que j'ai détecté est que dans certains albums qui ont des pistes avec des auteurs différents (par exemple en cas de collaboration entre plusieurs artistes, ou *featuring*), il arrive que mon système prenne l'auteur du *featuring* comme auteur de l'album. Je vais voir avec ma formatrice pour lui demander s'il est nécessaire de corriger ce problème, par exemple en prenant le nom d'artiste qui apparaît sur le plus de pistes.



11:40 : Je pushe le code pour les métadonnées (qui fonctionne malgré le problème reporté) sur le dépôt distant et je pars en pause.

12:40 : Reprise du travail. Comme l'importation de musique fonctionne, je vais me concentrer sur les fonctionnalités de modification de la bibliothèque musicale (*user stories S7 à S10*). Celles-ci sont indispensables au TPI, comme elles sont l'un des 7 critères spécifiques à mon énoncé :

A20 : Un utilisateur connecté peut modifier sa bibliothèque musicale.

15:30 : Pour gérer le formulaire de modification, j'ai choisi de créer une API qui s'occupe de toute la page *Modifier* : elle met à jour l'album et ses pistes. Comme je ne dois pas modifier une piste individuellement autre part dans l'application, ce choix m'a paru judicieux car c'est plus simple de faire toutes les modifications nécessaires avec une seule requête.

15:50 : Une fois que le formulaire a été envoyé et que le serveur a traité la requête, comment mettre à jour les morceaux dans l'interface côté client avec les nouvelles données ? Je prends chaque morceau de l'album, et je regarde si ce morceau existe dans la nouvelle *tracklist* que le serveur m'a retournée. Si c'est le cas, je remplace l'ancien morceau avec le nouveau. Dans le cas contraire, cela veut dire qu'il n'existe plus : je le supprime de l'interface.

i `this.$set` et `this.$delete` sont des méthodes spéciales fournies par Vue qui permet de mettre à jour les données dans l'interface (elles sont aussi connues sous le nom de `Vue.set` et `Vue.delete`).

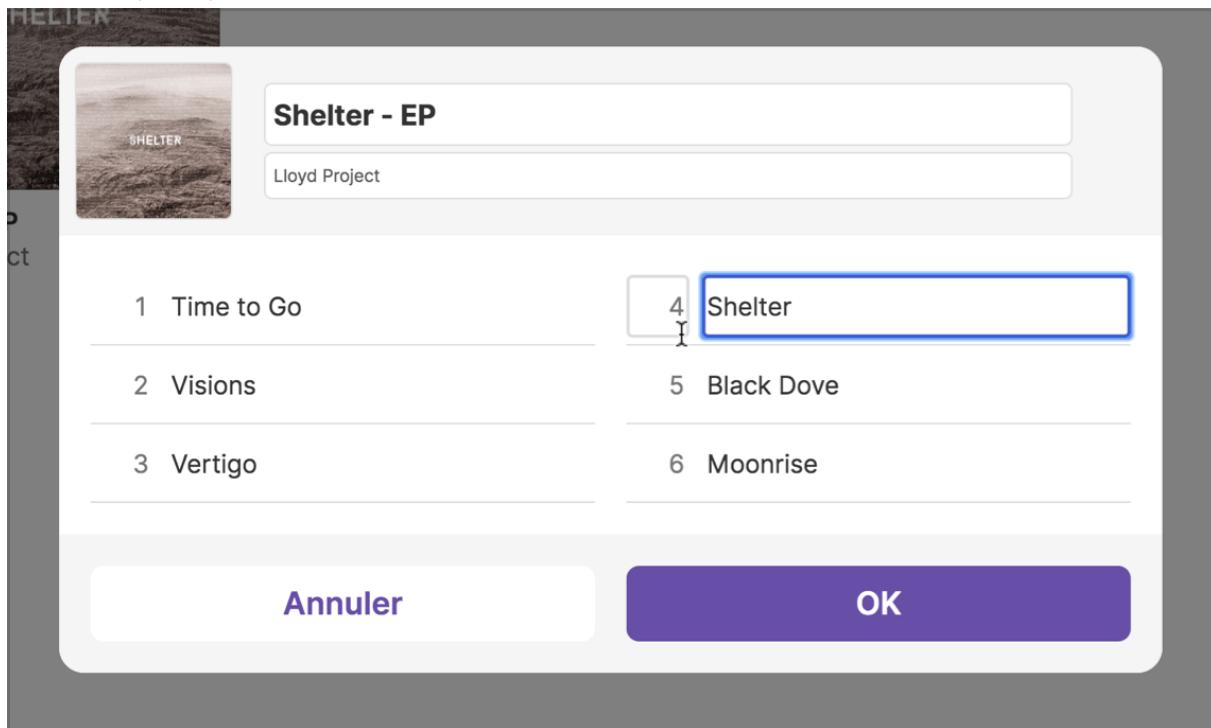
Par exemple, `this.$set(this.tableau, cle, valeur)` et équivalent à `this.tableau[cle] = valeur` en JavaScript pur, sauf qu'en utilisant les méthodes que Vue propose, on informe le framework que les données ont été mises à jour. Sans cela, Vue ne pourrait pas savoir que le tableau a été modifié (à cause de limitations de JavaScript).

```
existingSongs.forEach((existingSong) => {
  const existingSongIndex = this.songs.indexOf(existingSong);

  const newMatchingSong = newSongsFormatted.find(song => song.id ===
  existingSong.id);
  if (newMatchingSong) { // The song was updated
    this.$set(this.songs, existingSongIndex, newMatchingSong);
  } else { // The song was deleted
```

```
this.$delete(this.songs, existingSongIndex);  
}  
});
```

16:20 : Pour la modification des pistes, j'ai choisi de faire une interface comme ceci, où les champs de texte pour les pistes ne sont pas visibles tant qu'on ne passe pas la souris. Cela permet d'avoir une interface plus "naturelle" à mon goût que si l'on avait des champs de textes empilés par douzaines.



16:45 : J'ai terminé la modification. Mon code côté PHP est prêt pour la suppression de morceaux, mais je dois encore m'occuper d'afficher les boutons dans l'interface. Il faut aussi que je fasse la suppression d'album en entier.

Bilan

Une fois n'est pas coutume, la plus grosse difficulté n'était pas technique, mais juridique (la question liée à la licence LGPL). Cependant, je suis quand même satisfait d'avoir pu trouver une solution qui fonctionne pour mon projet et qui extrait les métadonnées ID3 avec succès.

J6 : jeudi 16 mai 2019

Objectifs

Aujourd'hui, je vais terminer la suppression de morceaux et d'albums. Ensuite, je vais pouvoir commencer à tester intensivement mon application pour pouvoir corriger les

éventuels problèmes que j'aurai rencontré.

Déroulement

07:40 : Je reprends le travail. Je vais m'occuper d'ajouter l'interface pour supprimer un morceau.

08:10 : J'ai implémenté et testé le bouton de suppression de morceau. Je vais maintenant faire le bouton de suppression d'un album.

09:15 : J'ai reçu une visite de Mme Travnjak, ma formatrice. Elle a pu me répondre sur les questions suivantes :

Est-ce qu'il faut gérer le cas spécial où certains morceaux d'un album ont un auteur différent (par exemple en cas de collaboration avec d'autres artistes) ?

→ Non, ce n'est pas nécessaire de gérer ce cas spécifique. On peut le faire mais ce n'est pas une priorité.

Si l'utilisateur importe en même temps plusieurs morceaux d'albums différents, doit-on tous les mettre dans le même album ou en créer plusieurs ?

→ Oui, ça serait mieux que plusieurs albums soient créés.

J'ai aussi pu vérifier ma bonne compréhension de l'énoncé avec la question suivante :

Si j'ai bien compris l'énoncé, il n'y a pas besoin d'implémenter de fonctionnalité pour la modification de la photo de couverture d'un album, et on n'a pas besoin non plus de permettre à l'utilisateur de déplacer un morceau d'un album à l'autre. Est-ce correct ?

→ Oui, c'est exact, ces fonctionnalités sont hors de la portée de l'énoncé.

09:30 : J'ai pu corriger mon problème de sortie jack sur mon ordinateur école en réinstallant les pilotes Realtek.

11:00 : En faisant la fonctionnalité de suppression d'un album, je me rends compte que je dois aussi gérer le cas où un morceau supprimé était dans le lecteur ou en liste d'attente pour ne pas qu'un morceau non-existant s'y retrouve.

11:35 : J'ai terminé la fonctionnalité de suppression d'album, ce qui signifie que l'intégralité de mes fonctionnalités est implémentée. Je vais concentrer mon après-midi sur les tests intensifs et les corrections de bugs que j'aurai trouvés.

12:40 : C'est parti pour une grosse chasse au bugs : objectif, trouver des points

d'amélioration et les réaliser. Je vais aussi tester mon application sous des environnements (navigateurs et systèmes d'exploitation) différents pour vérifier la compatibilité multi-plateformes d'Interlude.

12:45 : Je profite du début de cette phase d'optimisations pour ajouter mes fichiers source Illustrator et Sketch à mon dépôt Git.

12:50 : Je commence avec un bug en lien avec l'affichage de la durée du morceau. Sous Firefox (Windows), il arrive que pendant les premières secondes de lecture du morceau, une durée incorrecte (par exemple 4 secondes) est affichée, puis la durée réelle s'affiche. Sous Safari (macOS), la longueur affichée est tout le temps `Infinity`.

13:20 : J'ai trouvé la solution à ce problème. Pour que le navigateur puisse lire le fichier de la manière la plus optimale possible, il faut que mon script qui renvoie le fichier audio gère les requêtes *Range* que le navigateur envoie. Ces requêtes permettent au navigateur de télécharger partiellement le fichier. Par exemple, si j'importe un fichier audio de 2 heures, le navigateur peut ainsi choisir de ne charger que la partie qui est proche de l'utilisateur (et ne pas télécharger des données qui ne seront normalement lues que dans une heure). Le post suivant m'a beaucoup aidé dans la résolution de ce problème :

<https://groups.google.com/d/msg/jplayer/nSM2UmnSKKA/bC-l3k0pCPMJ>

En mettant en place le nouveau code, j'ai dû corriger un bug très étrange : une sorte de "toc" sonore était audible pendant la première seconde de chaque morceau. En fait, c'était une variable (`$cursor`) que j'avais mal renommée, ce qui causait un problème lors de la lecture du fichier.

14:00 : J'ai corrigé l'apparence de la zone de recherche sous Safari (où les styles personnalisés n'étaient pas visibles) en utilisant la déclaration `-webkit-appearance: textfield;`. J'ai aussi modifié la couleur de certains styles de focus pour les rendre plus visibles.

14:20 : Je travaille maintenant sur une amélioration du système d'uploads pour gérer des uploads d'albums multiples en même temps.

14:50 : M. Foti est passé me voir pendant mon travail. Il m'a fait une remarque concernant l'envoi final des documents (il faut bien veiller à envoyer les documents sources par e-mail directement, et pas en passant par un service tiers d'hébergement comme Google Drive). Il m'a posé quelques questions sur mon travail mon environnement de travail (serveur utilisé).

16:35 : J'ai terminé mes modifications sur l'importation de nouveaux fichiers audio qui permettent :

- de gérer les cas où des fichiers venant d'albums différents sont importés dans Interlude (auparavant, ils étaient tous créés dans un album unique).
- de corriger le bug où dans certains cas, le nom d'une collaboration sur un seul morceau pouvait se retrouver en nom d'artiste pour un album entier.

Bilan

Après de nombreux jours de travail, j'ai enfin pu terminer toutes les fonctionnalités demandées par le cahier des charges de mon TPI. L'après-midi, j'ai été agréablement surpris par la correction de bug en rapport avec la durée du morceau : alors que je le prenais à l'origine pour un simple écart de fonctionnement entre deux navigateurs différents sur les APIs permettant d'accéder à l'élément `<audio>`, ce bug m'a au final permis d'en apprendre plus sur la façon dont le protocole HTTP permet un chargement efficace des ressources audio et vidéo. Je me suis ensuite concentré sur un long refactor qui m'a permis de gérer l'importation de plusieurs albums en même temps, qui avait été jugé nécessaire par ma formatrice et était donc prioritaire.

J'ai maintenant corrigé tous les compatibilité multiplateforme que j'ai trouvé. Je pense me concentrer sur la documentation les prochains jours, car c'est ma priorité numéro un pour mon TPI actuellement.

J7 : lundi 20 mai 2019

Objectifs

Aujourd'hui, je vais commencer à rédiger ma documentation technique. J'ai déjà toute la partie concernant la planification, les plans et scénarios de test et les fonctionnalités à implémenter. Je vais maintenant commencer toute la partie qui décrit après coup ce que j'ai réalisé.

Déroulement

07:30 : Je reprends le travail. Je vais commencer par créer la structure de base de la documentation.

08:00 : Je vais partir sur la structure suivante pour ma documentation :

Interlude

Table des matières

Table des versions

Introduction

Cahier des charges

Objectif

Fonctionnalités

Méthodologie

Planification en 6 étapes

Développement agile / backlog

Planification

Product backlog

Diagramme de Gantt

Analyse organique

Base de données

Description

MLD

Dictionnaire de données

Analyse des fonctionnalités majeures

Frameworks et outils externes

Vue.js

Axios

Laravel Mix

SASS

ESLint

getID3()

TwitterOAuth

Symfony VarDumper

Plan de test et tests

Équipement de test

Scénarios de test

Évolution des tests

Conclusion

Difficultés rapportées

Variantes de solutions et choix

Améliorations possibles

Remerciements

Bibliographie

Annexes

Glossaire

Résumé du TPI

Code source

Énoncé

Journal de bord

Celle-ci va peut-être être légèrement modifiée au cours de la rédaction de ma documentation en fonction des besoins.

Je vais commencer par rédiger la partie *Frameworks et outils externes*.

09:30 : La partie est terminée. J'ai ajouté NPM, Composer et PHP dotenv que j'avais initialement oublié. Je vais maintenant m'occuper de la bibliographie et du glossaire.

10:30 : La bibliographie et le glossaire sont faits. Je vais maintenant passer à la description des tests.

10:55 : La description des tests est bonne, je me lance sur le commentaire de la planification.

11:15 : Le commentaire pour la planification est fait.

Je pose une question à ma formatrice concernant le critère 2 de la question B6 de la grille d'évaluation du TPI. Celui-ci indique :

Le rapport est séparé en deux parties distinctes. **Partie 1 :**

- Énoncé
- Planning
- Journal de bord

Partie 2 :

- Résumé du rapport TPI
- Documentation du TPI (manuel technique)
- Manuel utilisateur
- Le code source (si applicable)

Je demande s'il y a des informations complémentaires sur ce critère, car je ne suis pas sûr de voir la logique derrière cette séparation.

→ Sa réponse : Comme j'ai déjà fait un rendu intermédiaire du planning (le premier jour) et du journal de bord (jeudi dernier) et que ceux-ci forment avec l'énoncé la première partie, c'est OK pour elle de ne pas avoir de division telle dans le rapport TPI et de mettre le journal de bord avec les autres annexes.

11:35 : Je pars en pause. Mes prochaines étapes sont le rappel de l'énoncé, la rédaction de l'introduction et la partie concernant la base de données.

12:40 : Reprise du travail. Je commence le rappel de l'énoncé.

13:50 : J'ai terminé le rappel de l'énoncé, l'introduction et la partie sur les bases de données. Je vais maintenant travailler sur la partie *Structure du projet* de la section *Analyse organique*.

15:40 : J'ai rédigé les parties *Structure du projet*, *Classes (PHP)* et *API interne*. Je vais maintenant faire la partie *Composants Vue.js*.

16:40 : J'ai rédigé la présentation des composants et leur liste. Je suis en train de faire des diagrammes pour voir les composants et leur emplacement dans l'interface. Je vais reprendre ce travail demain matin.

Bilan

Je suis très satisfait de mon travail aujourd'hui : j'ai pu réaliser de nombreuses parties essentielles de la documentation. Je suis aussi content du rendu visuel de ces parties dans le document.

J8 : mardi 21 mai 2019

Objectifs

Aujourd'hui, mon but est d'avancer au maximum la documentation. Je vais commencer par finir la partie *Composants Vue.js*, puis je vais continuer sur la partie *Méthodologie* et ensuite commencer la partie *Analyse des fonctionnalités majeures*.

Déroulement

07:40 : C'est reparti pour une journée de travail. Je reprends la conception des images montrant les composants Vue là où je m'étais interrompu hier.

08:20 : Mes images sont terminées. Je commence la partie *Méthodologie*.

09:30 : J'ai terminé ma partie *Méthodologie*. Je profite de l'avance que j'ai pour corriger un petit bug que j'ai remarqué en utilisant Interlude sous Safari : comme ce navigateur n'active pas le focus sur un élément lorsqu'on clique dessus, il est pour le moment impossible d'ouvrir le menu avec la souris.

10:20 : Je reprends le travail sur la documentation. Je vais maintenant m'intéresser à la partie *Analyse des fonctionnalités majeures*.

11:30 : J'ai terminé mon analyse des fonctionnalités majeures de mon projet.

12:40 : Reprise du travail. En attendant une visite prévue de M. Foti (que j'ai croisé dans le couloir il y a quelques instants), je vais travailler sur l'identité visuelle de mon rapport TPI.

14:30 : J'ai terminé les pages de garde et fait les en-têtes pour toutes les pièces jointes. Je vais commencer à rédiger mon résumé de rapport TPI.

16:40 : J'ai terminé le résumé du rapport TPI. Je vais m'occuper demain de la conclusion de mon rapport TPI.

Bilan

Je suis satisfait de mon travail aujourd'hui : j'ai pu continuer à avancer les points qui me manquaient dans ma documentation. Je suis aussi content d'avoir une identité visuelle qui met en valeur mon travail.

J9 : mercredi 22 mai 2019

Objectifs

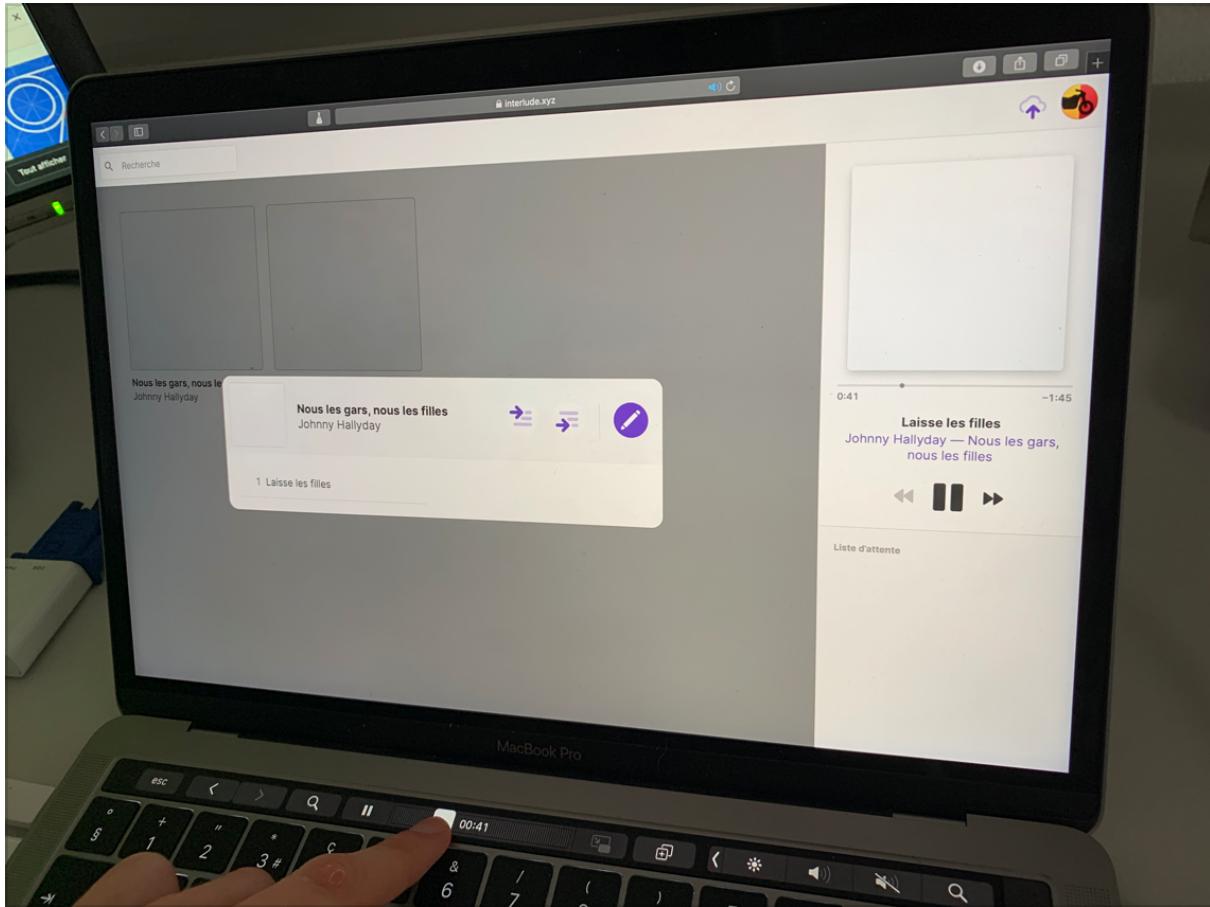
Mercredi 22 mai : veille du rendu. Je vais terminer le dernier point de doc qui me manque, à avoir la conclusion. Ensuite, je vais relire la grille d'évaluation du TPI pour vérifier qu'aucune information nécessaire ne manque. Une fois cette vérification faite, je vais pouvoir retourner sur l'implémentation pour tester une dernière fois que tout fonctionne, et éventuellement faire des améliorations mineures.

Déroulement

07:40 : Je commence ma journée. Je me mets à travailler sur la partie *Conclusion*.

11:15 : J'ai terminé la partie *Conclusion*. Je vais héberger une instance d'Interlude sur un serveur Infomaniak pour tester le projet dans ces conditions.

11:35 : Bonne découverte du jour : en utilisant Interlude sur un MacBook Pro récent, la barre de progression de lecture est visible dans la Touch Bar et on peut interagir avec. Safari détecte l'élément `<audio>` et affiche ses contrôles dans la Touch Bar.



11:40 : Je prends ma pause.

12:40 : Reprise du travail. Je vais relire ma documentation technique et la confronter à la grille d'évaluation.

14:20 : J'ai terminé de relire la documentation et j'ai corrigé les erreurs trouvées. J'ai aussi pu passer mon document à la moulinette d'*Antidote*, un correcteur automatique avancé.

15:00 : J'ai complété la documentation de mes composants Vue et de mes points d'accès à l'API dans mon code.

16:10 : J'ai pu comparer ma documentation avec la grille d'évaluation.

16:33 : J'ai ajouté un message qui rend plus clair le cas où l'utilisateur n'a aucune musique importée dans sa bibliothèque. Au lieu d'avoir une vue vide, l'utilisateur voit un message lui indiquant comment importer de la musique.

Bilan

Je me rapproche de plus en plus de la fin du TPI. À ce point là, il ne me reste plus que des petits détails à corriger. Je me réjouis de voir le résultat final !

J10 : jeudi 23 mai 2019

Objectifs

Aujourd’hui, c’est le tout dernier jour de mon TPI ! Mon premier objectif est d’ajouter des dernières améliorations à l’interface pendant la première partie du matin, pour que l’utilisateur puisse voir lorsque quelque chose est en cours de chargement. Ensuite, je vais faire mes derniers tests et relire intégralement ma documentation en vue du rendu.

Déroulement

07:40 : Je reprends le travail commencé hier sur le chargement des albums.

08:05 : L’indicateur de chargement pour les albums fonctionne. Je passe à la fonctionnalité suivante, c’est à dire les uploads.

09:35 : L’indicateur de progression sur les uploads marche. C’est le tout dernier ajout que je fais, je vais faire maintenant de la recherche de bug pour la fin de la matinée.

11:40 : Mes corrections de bugs réalisées, je pars en pause. Le détail des modifications effectuées est disponible sur GitHub.

12:40 : Reprise du travail.

14:30 : J’ai corrigé un bug que j’ai trouvé qui pouvait supprimer un morceau lors de l’édition si l’on entrait des données incorrectes. Ce bug est maintenant corrigé (j’empêche l’utilisateur de saisir les données incorrectes).

15:45 : Je suis en pleine phrase de rendu du TPI. J’ai exporté mon code avec la fonction *Print* de PhpStorm, et fusionné mon rapport TPI avec les annexes avec *Preview.app* (application native de macOS).

Ici s’achève mon rapport TPI. Heure de fin du travail prévue : 16:45

Bilan

Et voilà, mon TPI est terminé ! Je suis très satisfait du travail que j’ai pu produire pendant ces 11 jours très intenses. Une auto-évaluation sur l’ensemble de mon travail est disponible dans la partie *Conclusion* de mon rapport TPI.

Code source



```
1 # Interlude
2 *Une [**version française**](#fr) est disponible plus bas.*
3
4 Interlude is an online music player built with Vue.js on the front end,
   and PHP/MySQL on the back end.
5
6 ## Getting started
7
8 ### Prerequisites
9 You will need:
10 - A local installation of [PHP](https://www.php.net/downloads.php) (
    version 7.2 or newer)
11 - [Composer](https://getcomposer.org/download/)
12 - A [MySQL](https://www.mysql.com) database
13 - A recent version of [Node.js](https://nodejs.org/en/) and NPM (both are
    installed through the Node.js installer)
14
15 ### Installation
16 1. Install all the PHP dependencies using Composer with this command:
17   ```bash
18 composer install
19   ```
20
21 2. Install all the JavaScript dependencies using NPM with this command:
22   ```bash
23 npm install
24   ```
25
26 3. Import the database structure from the `./database/structure.sql` file:
27   ```sql
28 $ cd interlude/database/
29 $ mysql
30 mysql> use {your table name};
31 mysql> source structure.sql;
32   ```
33
34 4. Make a copy of the `.`env.example` file in the project root named `.`env` and fill in your instance URL (`APP_URL`) and your database credentials.
35
36 5. Set up your local PHP server to have the `public` folder as its document root. This is necessary so the environment configuration files, external dependencies and application storage are hidden from the website users.
37
38 6. Make sure the `storage` folder is writable by the server.
39
40 7. Configure a Twitter app for your instance (see below).
41
42 ### Create a Twitter app for Interlude
43 Since Twitter login is an essential feature of Interlude, you'll need to create a new app in the Twitter developer dashboard to run an Interlude instance. To do so:
44
45 1. If you don't have a Twitter account yet, create one. Otherwise, log in
```

45 on [twitter.com](<https://twitter.com>).
46 2. Open the *Apps* page on the Twitter Developer website: <https://developer.twitter.com/en/apps>.
47 3. Click the *Create an app* button.
48 ➤ **Note:** Twitter may ask you to apply to a developer account. If so, you will need to confirm your account using a phone number, and fill a form asking you why you want to use the Twitter API. Once you have sent the form, you will be able to create the Twitter app (you don't need to wait for an answer from the Twitter Developer team).
49 4. Fill in the *App name* and *Application description* fields. This data will be visible by your users on the Twitter login confirmation page when they have to confirm Interlude's access to their account.
50 5. In the *Website URL* field, write your Interlude instance's URL (e.g. `https://interlude.test`)
51 6. Check the *Enable Sign in with Twitter* checkbox and add `{instance URL}/auth.php` as a callback URL (e.g. `https://interlude.test/auth.php`)
52 7. Fill in the last required field, called *Tell us how this app will be used*, with some text explaining to Twitter employees what the project does.
53 8. Click the *Create* button.
54 9. On your app's page in the Twitter Developer dashboard, go to the *Keys and tokens* section.
55 10. Copy the *API key* in the `TWITTER_API_KEY` field of your `.env` file, and the *API secret key* in the `TWITTER_API_SECRET` field.
56
57 You're all set now!
58
59 ➤ **Note:** By default, the permission level for a Twitter app is set to *Read and write*. Interlude only needs a *Read-only* access, so you might want to change this parameter as well in the *Permissions* tab of the app dashboard. The requested level of permission is shown to the user during the login process.
60
61 ➤ **Pro-tip:** At this point, your Twitter app uses the default Twitter app icon. You can change it to an other icon so it's visible on the Twitter website during the login process, and in the Twitter settings of Interlude users.
62
63 #### Building JavaScript and CSS assets
64 The Interlude project uses [Laravel Mix](<https://laravel-mix.com>) to build the bundled JavaScript and CSS files.
65
66 During development, you can use the `npm run watch` command. It builds automatically the assets when you edit a source file. When you are done, press CTRL-C to stop the command.
67
68 To build the minified assets for production, use `npm run prod`.
69
70 #### Linting JavaScript code
71 You can verify that the JavaScript code of the project is compliant with the [Airbnb JavaScript style guide](<https://github.com/airbnb/javascript>) with the following command:
72 ````bash`
73 `npm run lint`

```

74 ````
75
76 This command will use [ESLint](https://eslint.org) to find issues in the
code saved in the `assets` folder.
77
78 ## Authors
79 - **Nicolas Ettlin** ([nicolas.ettlin@me.com](mailto:nicolas.ettlin@me.
com))
80
81 # <a name="fr"></a> Interlude *(version française)*
82 Interlude est un lecteur musical en ligne créé avec Vue.js côté
navigateur, et PHP/MySQL du côté du serveur.
83
84 ## Pour débuter
85
86 #### Pré-requis
87 Vous avez besoin de :
88 - Une installation locale de [PHP](https://www.php.net/downloads.php) (
version 7.2 ou plus récente)
89 - [Composer](https://getcomposer.org/download/)
90 - Une base de données [MySQL](https://www.mysql.com)
91 - Une version récente de [Node.js](https://nodejs.org/en/) et NPM (les
deux logiciels sont installés par l'installateur Node.js)
92
93 #### Installation
94 1. Installez toutes les dépendances PHP en utilisant Composer avec cette
commande :
95 ````bash
96 composer install
97 `````
98
99 2. Installez toutes les dépendances JavaScript en utilisant NPM avec
cette commande :
100 ````bash
101 npm install
102 `````
103
104 3. Importez la structure de la base de données depuis le fichier `/`/
database/structure.sql` :
105 `````
106 $ cd interlude/database/
107 $ mysql
108 mysql> use {nom de la table};
109 mysql> source structure.sql;
110 `````
111
112 4. Faites une copie du fichier `.`.env.example` à la racine du projet
nommée `.`.env` et remplissez l'URL de votre instance (`APP_URL`) et vos
identifiants de connexion à la base de données.
113
114 5. Configurez votre serveur PHP local pour qu'il utilise le dossier ``
public` comme racine (*document root*). Cette opération est nécessaire
afin que les fichiers de variables d'environnement, les dépendances
externes et le stockage de l'application restent cachés des utilisateurs

```

114 du site.

115

116 6. Assurez-vous que le serveur a les droits d'écrire dans le dossier `storage`.

117

118 7. Configurez une app Twitter pour votre instance (voir plus bas).

119

120 **###** Créer une app Twitter pour Interlude

121 Comme le login avec Twitter est une fonctionnalité essentielle d' Interlude, vous aurez besoin de créer une nouvelle application dans le * Twitter developer dashboard* pour faire fonctionner une instance d' Interlude. Pour ce faire :

122

123 1. Si vous n'avez pas de compte Twitter, créez-en un. Sinon, connectez-vous sur [twitter.com](<https://twitter.com>).

124 2. Ouvrez la page *Apps* sur le site *Twitter Developer*: <https://developer.twitter.com/en/apps>.

125 3. Cliquez sur le bouton *Create an app* (créer une application).

126 ****Note**:** Twitter pourrait vous demander d'envoyer une demande pour avoir un compte développeur. Si c'est le cas, vous aurez besoin de confirmer votre compte en utilisant un numéro de téléphone, et de remplir un formulaire vous demandant pourquoi vous voulez utiliser l'API Twitter . Une fois que vous avez envoyé le formulaire, vous pourrez créer l'app Twitter (vous n'avez pas besoin d'attendre une réponse de Twitter).

127 4. Remplissez les champs *App name* (nom de l'application) et * Application description* (description de l'application). Ces données seront visibles publiquement par vos utilisateurs sur la page de connexion lorsqu'ils doivent confirmer l'accès d'Interlude à leur compte.

128 5. Dans le champ *Website URL*, écrivez l'URL de votre instance d' Interlude (par exemple `https://interlude.test`)

129 6. Cochez la case *Enable Sign in with Twitter* et ajoutez `{URL de l' instance}/auth.php` comme *callback URL* (par exemple `https://interlude. test/auth.php`)

130 7. Remplissez le dernier champ requis, appelé *Tell us how this app will be used* (dites-nous comment cette application sera utilisée), avec du texte expliquant aux employés de Twitter ce que fait le projet.

131 8. Cliquez sur le bouton *Create*.

132 9. Sur la page de votre application dans le *Twitter Developer dashboard* , allez dans la section *Keys and tokens*.

133 10. Copiez le contenu du champ *API key* dans le champ `TWITTER_API_KEY` de votre fichier `.env` file, et le contenu du champ *API secret key* dans le champ `TWITTER_API_SECRET` .

134

135 C'est tout bon à présent !

136

137 ****Note**:** Par défaut, le niveau de permission d'une app Twitter est réglé sur *Read and write* (lecture et écriture). Interlude n'aura besoin que d'un accès *Read-only* (lecture simple), donc vous pourriez vouloir changer ce paramètre également dans la section *Permissions* des réglages de votre application. Le niveau d'accès demandé par l'application est affiché à l'utilisateur pendant la connexion.

138

139 ****Astuce de pro !**** Pour l'instant, votre application Twitter utilisera l'icône par défaut. Vous pouvez la changer pour une vraie icône d'

139 application dans les paramètres, pour que cette icône s'affiche aux utilisateurs d'Interlude pendant la connexion à leur compte Twitter, et dans les réglages Twitter.

140

141 **### Compiler les ressources JavaScript et CSS**

142 Le projet Interlude utilise [Laravel Mix](<https://laravel-mix.com>) pour compiler les ressources JavaScript et CSS nécessaires.

143

144 Pendant le développement, vous pouvez utiliser la commande `npm run watch`. Elle compile les ressources automatiquement lorsque vous modifiez un fichier source. Lorsque vous avez terminé, pressez CTRL+C pour arrêter la commande.

145

146 Pour compiler les ressources minifiées pour la production, utilisez `npm run prod`.

147

148 **### *Linting* de code JavaScript**

149 Vous pouvez vérifier que le code JavaScript du projet est conforme au [guide de style JavaScript d'Airbnb](<https://github.com/airbnb/javascript>) avec la commande `npm run lint`.

150

151 Cette commande utilisera [ESLint](<https://eslint.org>) pour trouver des problèmes dans les fichiers dans le dossier `assets`.

152

153 **## Auteurs**

154 - ****Nicolas Ettlin**** ([nicolas.ettlin@me.com](mailto:nicolas.ettlin@me.com))

155

```
1 *~  
2 .env  
3 .DS_Store  
4 .idea/  
5 .vscode/  
6 /vendor/  
7 /public/dist  
8 /node_modules/  
9
```

```
1 APP_ENV=dev
2 APP_URL=https://interlude.test
3
4 # Database credentials
5 DB_HOST=localhost
6 DB_PORT=3306
7 DB_BASE=interlude
8 DB_USERNAME=homestead
9 DB_PASSWORD=secret
10
11 # Twitter credentials
12 TWITTER_API_KEY=
13 TWITTER_API_SECRET=
14
```

```
1 module.exports = {
2   env: {
3     browser: true,
4     es6: true,
5   },
6   extends: ['airbnb-base', 'plugin:vue/base'],
7   globals: {
8     Atomics: 'readonly',
9     SharedArrayBuffer: 'readonly',
10 },
11 parserOptions: {
12   ecmaVersion: 2018,
13   sourceType: 'module',
14 },
15 plugins: [
16   'vue',
17 ],
18 rules: {},
19 };
20
```

```
1 {
2   "name": "interlude",
3   "version": "1.0.0",
4   "description": "Web music player",
5   "scripts": {
6     "dev": "npm run development",
7     "development": "cross-env NODE_ENV=development node_modules/webpack/bin/webpack.js --progress --hide-modules --config=node_modules/laravel-mix/setup/webpack.config.js",
8     "watch": "npm run development -- --watch",
9     "hot": "cross-env NODE_ENV=development node_modules/webpack-dev-server/bin/webpack-dev-server.js --inline --hot --config=node_modules/laravel-mix/setup/webpack.config.js",
10    "prod": "npm run production",
11    "production": "cross-env NODE_ENV=production node_modules/webpack/bin/webpack.js --no-progress --hide-modules --config=node_modules/laravel-mix/setup/webpack.config.js",
12    "lint": "node_modules/.bin/eslint --ext .js,.vue assets"
13  },
14  "repository": {
15    "type": "git",
16    "url": "git+https://github.com/Nicolapps/Interlude.git"
17  },
18  "author": "Nicolas Ettlin <nicolas.ettlin@me.com>",
19  "license": "UNLICENSED",
20  "bugs": {
21    "url": "https://github.com/Nicolapps/Interlude/issues"
22  },
23  "homepage": "https://github.com/Nicolapps/Interlude#readme",
24  "devDependencies": {
25    "eslint": "^5.16.0",
26    "eslint-config-airbnb-base": "^13.1.0",
27    "eslint-plugin-import": "^2.17.2",
28    "eslint-plugin-vue": "^5.2.2",
29    "laravel-mix": "^4.0.15",
30    "resolve-url-loader": "2.3.1",
31    "sass": "^1.20.1",
32    "sass-loader": "7.*",
33    "sass-resources-loader": "^2.0.0",
34    "vue-template-compiler": "^2.6.10"
35  },
36  "dependencies": {
37    "axios": "^0.18.0",
38    "cross-env": "^5.2.0",
39    "vue": "^2.6.10"
40  }
41 }
42 }
```

```
1 root = true
2
3 [*]
4 charset = utf-8
5 indent_style = space
6 indent_size = 2
7 end_of_line = lf
8 insert_final_newline = true
9 trim_trailing whitespace = true
10
```

```
1 {
2     "name": "nicolapps/interlude",
3     "description": "Web music player",
4     "type": "project",
5     "authors": [
6         {
7             "name": "Nicolas Ettlin",
8             "email": "nicolas.ettlin@me.com"
9         }
10    ],
11    "autoload": {
12        "psr-4": {
13            "Interlude\\": "src"
14        }
15    },
16    "require": {
17        "ext-pdo": "*",
18        "ext-json": "*",
19        "ext-fileinfo": "*",
20        "vlucas/phpdotenv": "^3.3",
21        "abraham/twitteroauth": "^1.0",
22        "james-heinrich/getid3": "^1.9"
23    },
24    "require-dev": {
25        "symfony/var-dumper": "^4.2"
26    }
27 }
28 }
```

```
1 /**
2 * Mix asset management
3 *
4 * Laravel Mix builds the JavaScript and CSS assets for the project.
5 * It bundles all the JavaScript modules with Webpack and processes the
6 * Vue component files.
7 *
8 */
9
10 const mix = require('laravel-mix');
11
12 mix.js('assets/app.js', 'public/dist/app.js')
13 .options({
14     extractVueStyles: 'public/dist/app.css',
15     globalVueStyles: 'assets/stylesheets/variables.scss',
16 });
17
```

```
1 <?php
2 namespace Interlude;
3
4 /**
5  * Manages authentication (login)
6  */
7 class Auth
8 {
9 /**
10  * Know if the user is currently logged in.
11 *
12 * @return bool
13 */
14 public static function isLoggedIn(): bool
15 {
16     return isset($_SESSION['interlude_user_id']);
17 }
18
19 /**
20  * Logs in as an user.
21 *
22 * @param User $user The user to log in as
23 */
24 public static function logInAs(User $user)
25 {
26     $_SESSION['interlude_user_id'] = $user->getId();
27 }
28
29 /**
30  * Disconnects the logged in user.
31 */
32 public static function logOut(): void
33 {
34     unset($_SESSION['interlude_user_id']);
35 }
36
37 /**
38  * Returns the currently logged in user.
39 *
40 * @return User|null
41 */
42 public static function user()
43 {
44     if (!self::isLoggedIn()) {
45         return null;
46     }
47
48     static $user = null;
49
50     if (is_null($user)) { // Retrieve the user form the database
51         $userRow = Database::find('users', 'id_user = :id', [
52             'id' => $_SESSION['interlude_user_id'],
53         ]);
54 }
```

```
55     if (is_null($userRow)) { // Unknown (not existing) user logged in
56         self::logOut();
57         return null;
58     }
59
60     $user = new User(
61         $userRow['id_user'],
62         $userRow['twitter_user_id'],
63         $userRow['full_name'],
64         $userRow['username']
65     );
66 }
67
68     return $user;
69 }
70
71 /**
72 * Returns the ID of the logged in user (or null if no one is logged)
73 *
74 * @return int|null
75 */
76 public static function id()
77 {
78     $user = self::user();
79     return $user ? $user->getId() : null;
80 }
81 }
82 }
```

```

1 <?php
2 namespace Interlude;
3
4 /**
5  * A track in an album.
6 */
7 class Song implements \JsonSerializable
8 {
9     /**
10      * The directory where audio are stored
11     */
12     public const AUDIO_FILES_STORAGE_DIRECTORY = __DIR__ . '/../storage/
songs';
13
14     public const ALLOWED_MIME_TYPES = [
15         'audio/aac',
16         'audio/mpeg',
17         'audio/ogg',
18         'audio/wav',
19         'audio/webm',
20         'audio/x-m4a',
21         'audio/x-aiff',
22     ];
23
24     /**
25      * The song ID in database
26      *
27      * @var int
28     */
29     public $id;
30
31     /**
32      * The song name
33      *
34      * @var string
35     */
36     public $title;
37
38     /**
39      * The track number (in the album)
40      *
41      * @var int
42     */
43     public $trackNumber;
44
45     /**
46      * The name of the audio file
47      *
48      * @var string
49     */
50     public $filePath;
51
52     /**
53      * ID of the album that the song belongs to

```

```

54     *
55     * @var int
56     */
57     public $idAlbum;
58
59     /**
60      * Creates a new Song instance.
61      *
62      * @param int    $id
63      * @param string $title
64      * @param int    $trackNumber
65      * @param string $filePath
66      * @param int    $idAlbum
67      */
68     public function __construct(int $id, string $title, int $trackNumber,
69     string $filePath, int $idAlbum)
70     {
71         $this->id = $id;
72         $this->title = $title;
73         $this->trackNumber = $trackNumber;
74         $this->filePath = $filePath;
75         $this->idAlbum = $idAlbum;
76     }
77
78     /**
79      * Saves the modifications made to the song in the database.
80      */
81     public function save()
82     {
83         Database::update('songs', [
84             'title' => $this->title,
85             'track_number' => $this->trackNumber,
86         ], 'id_song = :id', ['id' => $this->id]);
87     }
88
89     /**
90      * Deletes the song.
91      */
92     public function delete()
93     {
94         $filePath = static::AUDIO_FILES_STORAGE_DIRECTORY . '/' . $this->
95         filePath;
96         @unlink($filePath);
97
98         Database::delete('songs', 'id_song = :id', [
99             'id' => $this->id,
100        ]);
101    }
102
103    /**
104     * Returns the JSON representation of the object (that will be sent in
105     * Interlude APIs).
106     */
107     public function jsonSerialize()

```

```
105  {
106      return [
107          'id' => $this->id,
108          'title' => $this->title,
109          'trackNumber' => $this->trackNumber,
110          'idAlbum' => $this->idAlbum,
111      ];
112  }
113 }
114
```

```
1 <?php
2 namespace Interlude;
3
4 /**
5  * An Interlude user
6 */
7 class User implements \JsonSerializable
8 {
9     /**
10      * The user ID in database
11      *
12      * @var int
13      */
14     private $id;
15
16     /**
17      * The Twitter user ID (a.k.a. snowflake)
18      *
19      * @var string
20      */
21     private $twitterUserId;
22
23     /**
24      * The full name of the user
25      *
26      * @var string
27      */
28     private $fullName;
29
30     /**
31      * The username (handle on Twitter) of the user, without a leading '@'
32      *
33      * @var string
34      */
35     private $username;
36
37     /**
38      * Creates a User instance.
39      *
40      * @param int    $id          The user ID
41      * @param string $twitterUserId The Twitter user ID
42      * @param string $fullName    The Twitter full name
43      * @param string $username    The Twitter handle (with no leading '@')
44      */
45     public function __construct(int $id, string $twitterUserId, string
        $fullName, string $username)
46     {
47         $this->id = $id;
48         $this->twitterUserId = $twitterUserId;
49         $this->fullName = $fullName;
50         $this->username = $username;
51     }
52
53     /**
```

```

54     * Returns the user associated with a Twitter account.
55     * If it doesn't exist, a new user will be created.
56     *
57     * @param string $twitterUserId The Twitter user ID
58     * @param string $fullName      The Twitter full name
59     * @param string $handle        The Twitter handle (a.k.a. username)
60     *
61     * @return User
62     */
63     public static function fromTwitter(string $twitterUserId, string
64     $fullName, string $handle): User
64     {
65         $existingUser = Database::find('users', 'twitter_user_id = ?',
66         [$twitterUserId], 'id_user');
66
67         if (is_null($existingUser)) {
68             $userId = Database::insert('users', [
69                 'twitter_user_id' => $twitterUserId,
70                 'full_name' => $fullName,
71                 'username' => $handle,
72             ]);
73         } else {
74             $userId = $existingUser['id_user'];
75
76             // Update the user name to match the current one on Twitter
77             Database::update('users', [
78                 'full_name' => $fullName,
79                 'username' => $handle,
80             ], 'id_user = :id', ['id' => $userId]);
81         }
82
83         return new User($userId, $twitterUserId, $fullName, $handle);
84     }
85
86     /**
87     * Fetches all the albums in a user's music library.
88     * The albums are sorted by the date of the last uploaded song.
89     *
90     * @return Album[]
91     */
92     public function albums()
93     {
94         $rows = Database::run('SELECT albums.*'
95             . ' FROM albums'
96             . ' WHERE id_user = :id_user'
97             . ' ORDER BY'
98             . ' ('
99                 . ' -- Upload date of the last uploaded song'
100                . ' SELECT upload_date'
101                . ' FROM songs'
102                . ' WHERE songs.id_album = albums.id_album'
103                . ' ORDER BY upload_date DESC'
104                . ' LIMIT 1'
105            . ') DESC,

```

```

106         albums.id_album DESC
107     ', ['id_user' => $this->id]);
108
109     // Return an array of Album instances
110     return array_map(function ($row) {
111         return new Album($row['id_album'], $row['title'], $row['artist']);
112     }, $rows);
113 }
114
115 /**
116 * Fetches all the songs in a user's music library.
117 *
118 * @return Song[]
119 */
120 public function songs() {
121     $rows = Database::run('SELECT songs.*, albums.artist
122         FROM songs
123         LEFT JOIN albums ON albums.id_album = songs.id_album
124         WHERE albums.id_user = :id_user
125         ORDER BY id_song
126     ', ['id_user' => $this->id]);
127
128     // Return an array of Song instances
129     return array_map(function ($row) {
130         return new Song(
131             $row['id_song'],
132             $row['title'],
133             $row['track_number'],
134             $row['file_path'],
135             $row['id_album']
136         );
137     }, $rows);
138 }
139
140 /**
141 * Returns the user ID in database.
142 *
143 * @return int
144 */
145 public function getId(): int
146 {
147     return $this->id;
148 }
149
150 /**
151 * Returns the Twitter user ID.
152 *
153 * @return string
154 */
155 public function getTwitterUserId(): string
156 {
157     return $this->twitterUserId;
158 }
159

```

```
160  /**
161   * Returns the full name.
162   *
163   * @return string
164   */
165  public function getFullName(): string
166  {
167      return $this->fullName;
168  }
169
170 /**
171  * Returns the username (Twitter handle).
172  *
173  * @return string
174  */
175  public function getUsername(): string
176  {
177      return $this->username;
178  }
179
180 /**
181  * Get the JSON representation of the object (that will be sent in
182  * Interlude APIs)
183  */
184  public function jsonSerialize()
185  {
186      return [
187          'id' => $this->id,
188          'fullName' => $this->fullName,
189          'username' => $this->username,
190      ];
191  }
192 }
```

```
1 <?php
2 namespace Interlude;
3
4 use Interlude\Exceptions\UploadException;
5 use Interlude\Util\{Files, Metadata};
6
7 /**
8  * An album that contains some songs.
9 */
10 class Album implements \JsonSerializable
11 {
12     /**
13      * The directory where cover images are stored
14      */
15     public const COVERS_STORAGE_DIRECTORY = __DIR__ . '/../storage/
    album_covers';
16
17     /**
18      * List of all the allowed MIME types for the cover image file
19      */
20     public const ALLOWED_COVER_MIME_TYPES = [
21         'image/jpeg',
22         'image/gif',
23         'image/png',
24     ];
25
26     /**
27      * The album ID in database
28      *
29      * @var int
30      */
31     public $id;
32
33     /**
34      * The title of the album
35      *
36      * @var string
37      */
38     public $title;
39
40     /**
41      * The artist name
42      *
43      * @var string
44      */
45     public $artist;
46
47     /**
48      * Create albums from uploaded audio files
49      *
50      * @param array[] $files Array of PHP file info arrays
51      * @param int      $idUser The ID of the albums owner
52      *
53      * @return Album[]

```

```

54     *
55     * @throws UploadException When an error occurs during the upload
56     */
57     public static function createFromFiles(array $files, int $idUser):
58     array
59     {
60         // Validate all the files
61         foreach ($files as $file) {
62             $mimeType = Files::getMimeType($file['tmp_name']);
63             if (!in_array($mimeType, Song::ALLOWED_MIME_TYPES)) {
64                 throw new UploadException("Le type de fichier $mimeType n'est pas
supporté par Interlude. Veuillez essayer avec un autre fichier.");
65             }
66         }
67         // Split the songs by album name
68         /** @var Metadata[][] $songsByAlbum */
69         $songsByAlbum = [];
70         foreach ($files as $file) {
71             // Save the audio file on the disk
72             $extension = '.' . pathinfo($file['name'], PATHINFO_EXTENSION);
73             $fileName = uniqid('', true) . $extension;
74             $result = move_uploaded_file($file['tmp_name'], Song::
AUDIO_FILES_STORAGE_DIRECTORY . '/' . $fileName);
75             if ($result === false) {
76                 throw new UploadException('Impossible d'enregistrer la musique
sur le disque distant. Ce problème est probablement causé par une
mauvaise configuration du serveur.');
77             }
78             // Get the album name
79             $metadata = Metadata::extract(Song::AUDIO_FILES_STORAGE_DIRECTORY .
'/' . $fileName);
80             $album = $metadata->album ?? null;
81
82             // Store the song's metadata in its album
83             if (!array_key_exists($album, $songsByAlbum)) {
84                 $songsByAlbum[$album] = [];
85             }
86             $songsByAlbum[$album][] = $metadata;
87         }
88     }
89
90     // Save the albums and songs
91     /** @var Album[] $albums */
92     $albums = [];
93     foreach ($songsByAlbum as $albumTitle => $songs) {
94         // Get from all the songs in the album the song that contains the
metadata to use for the whole album
95         $averageMetadata = Metadata::average($songs);
96
97         // Save the cover image on the disk
98         $coverFile = null;
99
100        if (in_array(($averageMetadata->pictureMimeType ?? null), Album::

```

```

100 ALLOWED_COVER_MIME_TYPES)) {
101     $coverFile = uniqid('', true); // no extension
102
103     $result = file_put_contents(Album::COVERS_STORAGE_DIRECTORY . '/' .
104         . $coverFile, $averageMetadata->pictureData);
105     if ($result === false) {
106         throw new UploadException('Impossible d'enregistrer la musique
107             sur le disque distant. Ce problème est probablement causé par une
108             mauvaise configuration du serveur.');
109     }
110 }
111
112 // Insert the album into the database
113 $album = Album::create(
114     $albumTitle ?: ('Importé le ' . date('d/m/Y à H:i')),
115     $averageMetadata->artist ?? 'Artiste inconnu',
116     $coverFile,
117     $idUser
118 );
119
120 // Insert the songs into the database
121 foreach ($songs as $songIndex => $metadata) {
122     Database::insert('songs', [
123         'title' => $metadata->title ?? ('Titre' . ($songIndex + 1)),
124         'track_number' => $metadata->trackNumber ?? ($songIndex + 1),
125         'file_path' => basename($metadata->filePath),
126         'id_album' => $album->id,
127     ]);
128 }
129
130     $albums[] = $album;
131 }
132
133 /**
134 * Creates an album and stores it in the database.
135 *
136 * @param string      $title      The name of the album
137 * @param string      $artist     The name of the album author
138 * @param string|null $coverPicPath The name of the album picture
139 * @param int          $idUser     The ID of the album owner
140 *
141 * @return Album The created album
142 */
143 public static function create(string $title, string $artist,
144     $coverPicPath, int $idUser): Album
145 {
146     $id = Database::insert('albums', [
147         'title' => $title,
148         'artist' => $artist,
149         'cover_pic_path' => $coverPicPath,
150         'id_user' => $idUser,

```

```
150     ]);  
151  
152     return new self($id, $title, $artist);  
153 }  
154  
155 /**  
156 * Creates a new Album instance  
157 *  
158 * @param int    $id  
159 * @param string $title  
160 * @param string $artist  
161 */  
162 public function __construct(int $id, string $title, string $artist)  
163 {  
164     $this->id = $id;  
165     $this->title = $title;  
166     $this->artist = $artist;  
167 }  
168  
169 /**  
170 * Returns all the songs in the album.  
171 *  
172 * @return Song[]  
173 */  
174 public function songs()  
175 {  
176     $rows = Database::select('songs', 'id_album = :id_album', [  
177         'id_album' => $this->id,  
178     ]);  
179  
180     // Return an array of Song instances  
181     return array_map(function ($row) {  
182         return new Song(  
183             $row['id_song'],  
184             $row['title'],  
185             $row['track_number'],  
186             $row['file_path'],  
187             $this->id  
188         );  
189     }, $rows);  
190 }  
191  
192 /**  
193 * Save the modifications applied to the album in the database.  
194 */  
195 public function save()  
196 {  
197     Database::update('albums', [  
198         'title' => $this->title,  
199         'artist' => $this->artist,  
200     ], 'id_album = :id', ['id' => $this->id]);  
201 }  
202  
203 /**
```

```
204     * Deletes the album.
205     */
206     public function delete()
207     {
208         foreach ($this->songs() as $song) {
209             $song->delete();
210         }
211
212         Database::delete('albums', 'id_album = :id', [
213             'id' => $this->id,
214         ]);
215     }
216
217     /**
218      * Get the JSON representation of the object (that will be sent in
219      * Interlude APIs)
220     */
221     public function jsonSerialize()
222     {
223         return [
224             'id' => $this->id,
225             'title' => $this->title,
226             'artist' => $this->artist,
227         ];
228     }
229 }
```

```

1 <?php
2 namespace Interlude;
3
4 use PDO;
5
6 /**
7  * Database wrapper
8 */
9 class Database
10 {
11     /**
12      * The PDO instance
13      *
14      * @var PDO
15      */
16     private static $pdo = null;
17
18     /**
19      * Returns the PDO instance (and creates one if needed).
20      *
21      * @return PDO
22      */
23     public static function getInstance(): PDO
24     {
25         if (is_null(self::$pdo)) {
26             try {
27                 self::$pdo = new PDO(
28                     'mysql:host=' . getenv('DB_HOST') . ':' . getenv('DB_PORT') . '',
29                     getenv('DB_BASE'),
30                     getenv('DB_USERNAME'),
31                     getenv('DB_PASSWORD')
32                 );
33             } catch (\PDOException $e) {
34                 echo 'Unable to connect to the database (' . $e->getMessage() . ')
35             ';
36                 die();
37             }
38
39             self::$pdo->setAttribute(PDO::ATTR_DEFAULT_FETCH_MODE, PDO::
40                                         FETCH_ASSOC);
41
42             if (getenv('APP_ENV')) { // Show errors in the dev environment
43                 self::$pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION
44             );
45                 self::$pdo->setAttribute(PDO::ATTR_EMULATE_PREPARES, false);
46             }
47
48             return self::$pdo;
49         }
50
51         /**
52          * Runs a SQL query.
53          *
54          */
55     }

```

```

51 * @param string $query SQL query
52 * @param array $bind Bound parameters
53 *
54 * @return array Query results
55 */
56 public static function run(string $query, array $bind = [])
57 {
58     $query = self::getInstance()->prepare($query);
59     $query->execute($bind);
60
61     return $query->fetchAll();
62 }
63
64 /**
65 * Selects rows from the database.
66 *
67 * @example
68 * ```php
69 * $rows = Database::select('users', 'id_user = :id_user', ['id_user' =
70 > 1], 'id_user, full_name, username');
71 * // Returns :
72 * // ['id_user' => '1', 'full_name' => 'Nicolas Ettlin', 'username'
73 => 'Nicolapps'],
74 * // ]
75 *
76 * @param string $table Table name
77 * @param string $where WHERE condition
78 * @param array $bind WHERE condition parameter bind
79 * @param string $fields Fields to select
80 *
81 * @return array Rows (associative array form)
82 */
82 public static function select(string $table, string $where = '1', array
$bind = [], string $fields = '*'): array
83 {
84     return self::run("SELECT $fields FROM $table WHERE $where", $bind);
85 }
86
87 /**
88 * Retreives a row from the database.
89 *
90 * @param string $table Table name
91 * @param string $where WHERE condition
92 * @param array $bind WHERE condition parameter bind
93 * @param string $fields Fields to select
94 *
95 * @return array Row (associative array form) or null if no row was
96 * found
97 */
97 public static function find(string $table, string $where = '1', array
$bind = [], string $fields = '*')
98 {
99     $rows = self::run("SELECT $fields FROM $table WHERE $where LIMIT 1",

```

```

99 $bind);
100     return empty($rows) ? null : $rows[0];
101 }
102
103 /**
104 * Inserts a row into the database.
105 *
106 * @example
107 *   ```php
108 * \Interlude\Database::insert('users', [
109 *     'twitter_user_id' => '42',
110 *     'full_name'        => 'Nicolas Ettlin',
111 *     'username'         => 'Nicolapps',
112 * ]); // returns the new user ID
113 *
114 *
115 * @param string $table The table name
116 * @param array $row The fields to insert and their values
117 *
118 * @return int           The inserted row ID
119 */
120 public static function insert(string $table, array $row): int
121 {
122     $fields = '';
123     $values = '';
124
125     foreach (array_keys($row) as $key) {
126         if ($fields !== '') $fields .= ', ';
127         if ($values !== '') $values .= ', ';
128
129         $fields .= "`$key`";
130         $values .= ":$key";
131     }
132
133     self::run("INSERT INTO $table ($fields) VALUES ($values)", $row);
134
135     return self::getInstance()->lastInsertId();
136 }
137
138 /**
139 * Updates some rows in the database.
140 *
141 * @example
142 *   ```php
143 * \Interlude\Database::update('users', [
144 *     'full_name' => 'Nicolas',
145 * ], 'id_user = :id_user', ['id_user' => 1]);
146 *
147 *
148 * @param string $table      Table name
149 * @param array $fields      Fields to update and their new values
150 * @param string $where      WHERE condition
151 * @param array $whereBind  Binds to the condition
152 *

```

```

153     * @return int Updated rows count
154     */
155     public static function update(string $table, array $fields, string
156     $where = '1', array $whereBind = []): int
157     {
158         $sets = '';
159         $fieldsBind = [];
160         foreach ($fields as $column => $value) {
161             if ($sets !== '') $sets .= ',';
162             $sets .= "{$column} = :value_{$column}";
163             $fieldsBind["value_{$column}"] = $value;
164         }
165         $sql = "UPDATE $table SET $sets WHERE $where";
166
167         $query = self::getInstance()->prepare($sql);
168         $query->execute(array_merge($whereBind, $fieldsBind));
169
170         return $query->rowCount();
171     }
172
173 /**
174 * Deletes some rows from the database.
175 *
176 * @param string $table Table name
177 * @param string $where WHERE condition
178 * @param array $bind Binds to the condition
179 *
180 * @return int Deleted rows count
181 */
182     public static function delete(string $table, string $where, array $bind
183     = []): int
184     {
185         $sql = "DELETE FROM $table WHERE $where";
186
187         $query = self::getInstance()->prepare($sql);
188         $query->execute($bind);
189
190         return $query->rowCount();
191     }
192

```

```

1 <?php
2 namespace Interlude\Util;
3
4 /**
5  * Helper class for managing files
6  */
7 class Files
8 {
9     /**
10      * Returns all the files uploaded in the HTTP request with a specific
11      * input key.
12      *
13      * @param string $inputKey The input key name
14      *
15      * @return array[] Array of PHP file info arrays
16      */
17     public static function getUploadedFiles(string $inputKey): array
18     {
19         // If no such key exists in the request
20         if (!isset($_FILES[$inputKey]) || !is_array($_FILES[$inputKey]['name'])) {
21             return [];
22         }
23
24         // Convert $_FILES[$key] to a $files array
25         $files = [];
26         for ($i = 0; $i < count($_FILES[$inputKey]['name']); $i++) {
27             $file = [];
28
29             foreach (array_keys($_FILES[$inputKey]) as $infoKey) {
30                 $file[$infoKey] = $_FILES[$inputKey][$infoKey][$i];
31             }
32
33             $files[] = $file;
34         }
35
36         // Ignore files with errors
37         $files = array_filter($files, function ($file) {
38             return $file['error'] === 0;
39         });
40
41         return array_values($files);
42     }
43
44     /**
45      * Returns the MIME type of a file.
46      *
47      * @param string $filePath The path of the file
48      *
49      * @return string
50      */
51     public static function getMimeType($filePath) {
52         $finfo = new \finfo(FILEINFO_MIME_TYPE);
53         return $finfo->file($filePath);

```

```
53 }  
54 }  
55
```

```
1 <?php
2 namespace Interlude\Util;
3
4 /**
5  * ID3 metadata from an audio file
6  */
7 class Metadata
8 {
9 /**
10  * The title of the track
11  *
12  * @var string|null
13  */
14 public $title = null;
15
16 /**
17  * The artist's name
18  *
19  * @var string|null
20  */
21 public $artist = null;
22
23 /**
24  * The album name
25  *
26  * @var string|null
27  */
28 public $album = null;
29
30 /**
31  * The number of the track in the album
32  *
33  * @var int|null
34  */
35 public $trackNumber = null;
36
37 /**
38  * The contents of the cover picture file
39  *
40  * @var string|null
41  */
42 public $pictureData = null;
43
44 /**
45  * The MIME type of the cover picture
46  *
47  * @var string|null
48  */
49 public $pictureMimeType = null;
50
51 /**
52  * The path where the file is stored
53  *
54  * @var string|null
```

```

55  */
56  public $filePath = null;
57
58 /**
59 * Extracts the ID3 metadata from an audio file
60 *
61 * @param string $filePath The path of the audio file
62 *
63 * @return Metadata A Metadata object containing the metadata (or null
64 if no metadata was found)
65 */
66 {
67     // Handle the case where getID3 is not installed for licensing
68     // reasons.
69     if (!class_exists('\getID3')) {
70         return new static($filePath);
71     }
72
73     // Not existing file
74     if (!file_exists($filePath)) {
75         return new static($filePath);
76     }
77
78     // Use the getID3() lib to parse the ID3 tags
79     try {
80         $getID3 = new \getID3();
81         $fileInfo = $getID3->analyze($filePath);
82         \getid3_lib::CopyTagsToComments($fileInfo);
83     } catch (\getid3_exception $e) {
84         return new static($filePath);
85     }
86
87     // File with no metadata
88     if (isset($fileInfo['error']) || !isset($fileInfo['comments'])) {
89         return new static($filePath);
90     }
91
92     // Store the metadata in a new Metadata object
93     return new static($filePath, $fileInfo['comments']);
94 }
95
96 /**
97 * Get the metadata to use for a whole album from the song's metadata
98 *
99 * @param Metadata[] $tracksMetadata
100 *
101 * @return Metadata
102 */
103 public static function average(array $tracksMetadata): Metadata
104 {
105     $metadata = new static();
106
107     // Use the most frequent artist name

```

```

107     $artistNames = array_map(function ($metadata) {
108         return $metadata->artist ?? null;
109     }, $tracksMetadata);
110     $metadata->artist = self::getMostCommonValue($artistNames);
111
112     // Use the first picture for the album cover
113     $withCovers = array_filter($tracksMetadata, function($metadata) {
114         return !is_null($metadata->pictureMimeType);
115     });
116     $metadata->pictureMimeType = $withCovers[0]->pictureMimeType ?? null;
117     $metadata->pictureData = $withCovers[0]->pictureData ?? null;
118
119     return $metadata;
120 }
121
122 /**
123 * Finds the most common value in an array.
124 *
125 * @param array $values
126 *
127 * @return mixed|null
128 */
129 private static function getMostCommonValue(array $values)
130 {
131     // Remove null values
132     $values = array_filter($values, function ($value) {
133         return !is_null($value);
134     });
135
136     // Sort by frequency
137     $values = array_count_values($values);
138     arsort($values);
139     $values = array_keys($values);
140
141     // Return the most common value
142     return $values[0] ?? null;
143 }
144
145 /**
146 * Creates a new Metadata object from the comments array extracted by
147 * getID3.
148 * @param string|null $filePath The path where the file is stored
149 * @param array $comments The comments array of ID3
150 */
151 protected function __construct(string $filePath = null, array $comments
152 = [])
153 {
154     $this->filePath = $filePath;
155
156     if (isset($comments['title'])) {
157         $this->title = $comments['title'][0];
158     }

```

```
159     if (isset($comments['artist'])) {  
160         $this->artist = $comments['artist'][0];  
161     }  
162  
163     if (isset($comments['album'])) {  
164         $this->album = $comments['album'][0];  
165     }  
166  
167     if (isset($comments['track_number']) && ctype_digit($comments['  
track_number'][0])) {  
168         $this->trackNumber = (int)$comments['track_number'][0];  
169     }  
170  
171     if (isset($comments['picture'])) {  
172         $this->pictureData = $comments['picture'][0]['data'];  
173         $this->pictureMimeType = $comments['picture'][0]['image_mime'];  
174     }  
175 }  
176 }  
177
```

```
1 <?php
2 namespace Interlude\Exceptions;
3
4 /**
5  * Exception thrown when an error occurs during audio files upload.
6  */
7 class UploadException extends \Exception
8 {
9
10 }
11
```

```
1 import Vue from 'vue';
2 import App from './App.vue';
3
4 new Vue({ // eslint-disable-line no-new
5   el: '#app',
6   render: h => h(App),
7 });
8
```

```
1 <!--
2 Main view of the application.
3 Manages the music library view, the search results, and the global state.
4 -->
5
6 <template>
7   <div
8     class="root"
9
10    @dragover.prevent="$event.dataTransfer.types.includes('Files') && (
11      isDraggingFile = true)"
12    @dragleave.prevent.stop="isDraggingFile = false"
13    @drop.prevent="onDrop"
14  >
15    <nav class="navbar">
16      <input
17        class="search-box"
18        type="search"
19        v-model.trim="searchQuery"
20        @keydown.esc="searchQuery = ''"
21        placeholder="Recherche"
22        :disabled="albums === null"
23      >
24
25      <div class="navbar-right">
26        <div class="upload-files-form">
27          <button class="upload-files-button" @click="$refs.filePicker.
28            click()">
29            Importer de la musique
30          </button>
31
32          <input
33            type="file"
34            ref="filePicker"
35            accept="audio/aac, audio/mpeg, audio/ogg, audio/wav, audio/webm"
36            multiple
37            aria-hidden="true"
38            @change="upload($refs.filePicker.files)">
39        </div>
40
41        <UserMenu :user="user" />
42      </div>
43    </nav>
44
45    <div class="main-view">
46      <main class="main-view-content" :class="{ scrollable: albums !==
47        null }">
48        <!-- Loading state -->
49        <div v-if="albums === null" class="albums-list" role="presentation"
50        " title="Chargement...">
51          <li
52            class="album album-fake"
```

```

51          v-for="i in 64"
52          :key="`fake-albums-${i}`"
53      >
54          <div class="album-cover" :class="{ 'loading': user !== null }>
55              <strong :style="{ width: `${Math.random() * 50 + 50}%` }">
56                  <span :style="{ width: `${Math.random() * 50 + 50}%` }"></
57          span>
58          </li>
59      </div>
60
61      <!-- Search results -->
62      <template
63          class="search-results"
64          v-if="searchQuery && (searchResults.albums.length || searchResults.songs.length)">
65          >
66              <template v-if="searchResults.songs.length > 0">
67                  <h2>Morceaux</h2>
68                  <ul class="search-results-songs">
69                      <li
70                          @click="openAlbum(song.album.id, song.id)"
71                          v-for="song in searchResults.songs"
72                          :key="`search-songs-${song.id}`">
73                          
77                      >
78                      <button @click.stop="play(song.id)" class="play-now">
79                          Jouer
80                      </button>
81
82                      <strong>{{ song.title }}</strong>
83                      <span>{{ song.album.artist }} &mdash; {{ song.album.title }}</span>
84                  </li>
85              </ul>
86          </template>
87          <div class="song-waitlist-buttons">
88              <button
89                  class="play-next"
90                  @click.stop="play(song.id, 'next')">
91                  Jouer à la suite
92              </button>
93              <button
94                  class="play-later"
95                  @click.stop="play(song.id, 'later')">
96                  Jouer plus tard
97              </button>
98          </div>
99      </li>

```

```

100          </ul>
101      </template>
102
103      <h2 v-if="searchResults.albums.length > 0">Albums</h2>
104  </template>
105
106      <!-- No search results -->
107  <div class="no-search-results" v-else-if="searchQuery">
108      <div>
109          
110          <p>Aucun résultat pour «&nbsp;{{ searchQuery }}&nbsp;»</p>
111      </div>
112  </div>
113
114      <!-- Albums view -->
115  <ul class="albums-list">
116      <li
117          class="album"
118          v-for="album in (searchQuery ? searchResults.albums : albums)
119          "
120              :key=`albums-${album.id}`
121              @click="openAlbum(album.id)"
122          >
123              <div
124                  class="album-cover"
125                  :style="{ backgroundImage: `url(/resources/img/cover.php?id
126                  =${album.id})` }"
127                  role="img"
128              ></div>
129              <strong>{{ album.title }}</strong>
130              <span>{{ album.artist }}</span>
131          </li>
132      </ul>
133
134      <!-- No music in the library -->
135  <div class="no-music" v-if="!searchQuery && albums && albums.
136  length === 0">
137      <div>
138          
140          <p>Votre bibliothèque musicale est vide.</p>
141          <p class="no-music-help">
142              Pour importer de la musique, cliquez sur le bouton <em>
143              Importer</em> en haut à droite
144              ou glissez des fichiers audio dans la fenêtre.
145          </p>
146      </div>
147  </div>
148  </main>
149
150  <transition name="fade">
151      <AlbumPage
152          v-if="shownAlbumId"
153          :library="{ albums, songs }"

```

```

149          :id="shownAlbumId"
150          :key=`album-page-${shownAlbumId}`
151          :highlightedTrackId="highlightedTrackId"
152          @close="closeAlbum"
153          @play="play($event.music, $event.when)"
154          @saved="onAlbumSaved"
155          @deleted="onAlbumDeleted"
156      />
157    </transition>
158  </div>
159
160  <Player
161    :library="{ albums, songs }"
162    :nowPlaying="nowPlaying"
163    :waitlist="waitlist"
164    :history="history"
165    :uploads="uploads"
166    @next="nextSong"
167    @previous="previousSong"
168    @clearWaitlist="waitlist = []"
169  />
170
171  <LandingPage v-if="user !== undefined && !user" />
172
173  <DraggingFileOverlay v-if="isDraggingFile" />
174 </div>
175 </template>
176
177 <script>
178 import axios from 'axios';
179 import LandingPage from './components/LandingPage.vue';
180 import AlbumPage from './components/AlbumPage.vue';
181 import Player from './components/Player.vue';
182 import UserMenu from './components/UserMenu.vue';
183 import DraggingFileOverlay from './components/DraggingFileOverlay.vue';
184
185 export default {
186   components: {
187     LandingPage,
188     AlbumPage,
189     Player,
190     UserMenu,
191     DraggingFileOverlay,
192   },
193
194   data: () => ({
195     user: undefined, // Logged in user
196
197     // Music library
198     songs: null,
199     albums: null,
200
201     searchQuery: '',
202

```

```

203     // Parameters for the "album" page
204     shownAlbumId: null,
205     highlightedTrackId: null,
206
207     // Player state
208     nowPlaying: null, // ID of the song being played
209     waitlist: [], // IDs of the songs in the waitlist
210     history: [], // IDs of the song previously played
211
212     // File upload
213     isDraggingFile: false, // is a file currently being dragged over the
214     // UI?
215     lastUploadId: -1,
216     uploads: [],
217   },
218
219   created() {
220     // Load the logged in user
221     axios.get('/api/user.php')
222       .then((res) => {
223         this.user = res.data.user;
224         this.loadLibrary();
225       });
226
227   computed: {
228     /**
229      * Returns the search results to display
230      * @returns {{albums: Array, songs: Array}}
231      */
232     searchResults() {
233       if (!this.songs || !this.albums) { // Not loaded yet
234         return { songs: [], albums: [] };
235       }
236
237       const query = this.searchQuery.toLowerCase();
238
239       // This builds a search filter for the attributes passed
240       const byAttributes = (...attributesToCheck) => (
241         itemToCheck => attributesToCheck
242           .some(attribute => itemToCheck[attribute].toLowerCase().
243             includes(query))
244       );
245
246       const songs = this.songs.filter(byAttributes('title'));
247       const albums = this.albums.filter(byAttributes('title', 'artist'));
248
249       return { songs, albums };
250     },
251
252   methods: {
253     /**
254      * Loads the user's music library from the API.

```

```

255      */
256      loadLibrary() {
257          if (!this.user) {
258              return;
259          }
260
261          axios.get('/api/library.php')
262              .then((res) => {
263                  this.albums = res.data.albums;
264
265                  // Add the album info to the song
266                  this.songs = res.data.songs
267                      .map(song => ({
268                          ...song,
269                          album: this.albums.find(a => a.id === song.idAlbum),
270                      }));
271                  });
272          },
273
274      /**
275      * Shows an album.
276      * @param {Number} albumId           The ID of the album to show
277      * @param {Number} highlightedTrackId The ID of the song to highlight
278      */
279      openAlbum(albumId, highlightedTrackId = null) {
280          this.highlightedTrackId = highlightedTrackId;
281          this.shownAlbumId = albumId;
282      },
283
284      /**
285      * Closes the shown album.
286      */
287      closeAlbum() {
288          this.shownAlbumId = null;
289      },
290
291      /**
292      * Sends some music to the player.
293      * @param {Number[]} music ID of the song to play, or array of
294      * IDs
295      * @param {String} when             'now', 'next' or 'later'
296      */
297      play(music, when = 'now') {
298          const trackIds = Array.isArray(music) ? [...music] : [music];
299
300          switch (when) {
301              case 'now':
302                  this.nowPlaying = trackIds.shift() || null;
303                  this.waitlist = [...trackIds, ...this.waitlist];
304                  break;
305              case 'next':
306                  this.waitlist = [...trackIds, ...this.waitlist];
307                  break;
308              case 'later':
309          }
310      }

```

```

308         this.waitlist = [...this.waitlist, ...trackIds];
309         break;
310     default:
311         throw new Error(`Invalid value ("${when}") for the parameter
312         when.`);
313     },
314
315     /**
316      * Goes to the previous song.
317      */
318     previousSong() {
319         if (this.history.length === 0) {
320             return;
321         }
322
323         if (this.nowPlaying !== null) {
324             this.waitlist.unshift(this.nowPlaying);
325         }
326
327         this.nowPlaying = this.history.shift() || null;
328     },
329
330     /**
331      * Goes to the next song.
332      */
333     nextSong() {
334         if (this.nowPlaying !== null) {
335             this.history.unshift(this.nowPlaying);
336         }
337
338         this.nowPlaying = this.waitlist.shift() || null;
339     },
340
341     /**
342      * Updates the data from an album that was just edited.
343      *
344      * @param {Object} album The new album data
345      * @param {Array} songs The new data of the songs in the album
346      */
347     onAlbumSaved({ album, songs }) {
348         // Update the album
349         const albumIndex = this.albums.findIndex(a => a.id === album.id);
350         if (albumIndex === -1) { // Album not found
351             return;
352         }
353
354         this.$set(this.albums, albumIndex, album);
355
356         // Update the songs
357         const existingSongs = this.songs.filter(song => song.album.id ===
358             album.id);
359         const newSongsFormatted = songs.map(song => ({ ...song, album }));

```

```

360         existingSongs.forEach((existingSong) => {
361             const newMatchingSong = newSongsFormatted.find(song => song.id
362 === existingSong.id);
363             if (newMatchingSong) { // The song was updated
364                 const existingSongIndex = this.songs.indexOf(existingSong);
365                 this.$set(this.songs, existingSongIndex, newMatchingSong);
366             } else { // The song was deleted
367                 this.deleteSong(existingSong.id);
368             }
369             // No need to handle a new song in the album.
370         });
371     },
372
373     /**
374      * Removes an album that was just deleted.
375      * @param {Number} id ID of the album
376      */
377     onAlbumDeleted({ id }) {
378         this.shownAlbumId = null;
379
380         this.$delete(this.albums, this.albums.findIndex(album => album.id
381 === id));
382
383         this.songs
384             .filter(song => song.album.id === id) // Songs of the album
385             .forEach((song) => {
386                 this.deleteSong(song.id);
387             });
388
389     /**
390      * Deletes a song from Interlude.
391      * @param {Number} id The song ID
392      */
393     deleteSong(id) {
394         // Remove the song from the player first
395         if (this.nowPlaying === id) {
396             this.nowPlaying = null;
397         }
398
399         // Remove all occurrences of the song in the waitlist and the
400         // history
401         this.waitlist = this.waitlist.filter(item => item !== id);
402         this.history = this.history.filter(item => item !== id);
403
404         // Delete the song
405         this.$delete(this.songs, this.songs.findIndex(song => song.id ===
406             id));
407     },
408
409     /**
410      * Uploads some files to the user's library.
411      * @param {FileList} fileList

```

```

410      */
411      upload(fileList) {
412          const files = Array.from(fileList); // @var files File[]
413          this.$refs.filePicker.value = ''; // Empty the file picker dialog
414          selection
415          this.isDraggingFile = false;
416          if (files.length === 0) {
417              return;
418          }
419
420          this.lastUploadId += 1;
421          const upload = {
422              id: this.lastUploadId,
423              name: files.length === 1
424                  ? (files[0].name || 'Fichier sans nom')
425                  : `${files.length} fichiers`,
426              status: 'progress',
427              progress: 0,
428              error: null,
429          };
430          this.uploads.push(upload);
431
432          const hideAfter = (seconds) => {
433              setTimeout(() => {
434                  const uploadIndex = this.uploads.indexOf(upload);
435                  this.$delete(this.uploads, uploadIndex);
436              }, seconds * 1000);
437          };
438
439          const xhr = new XMLHttpRequest();
440          xhr.open('POST', '/api/upload.php');
441
442          xhr.upload.addEventListener('progress', (e) => {
443              if (e.lengthComputable) {
444                  upload.progress = e.loaded / e.total;
445              }
446          });
447
448          // Handle the request completion
449          xhr.onreadystatechange = () => {
450              if (xhr.readyState !== XMLHttpRequest.DONE) {
451                  return;
452              }
453
454              if (xhr.status === 200) { // OK
455                  const response = JSON.parse(xhr.responseText);
456
457                  if (response.errors.length === 0 && response.albums.length !==
458                      0) {
459                      const { albums, songs } = response;
460
461                      this.albums.unshift(...albums); // Push at the beginning
462                      this.songs.push(...songs.map(song => ({

```

```

462          ...song,
463          album: albums.find(album => album.id === song.idAlbum),
464        })));
465
466        // Open the newly created album
467        this.shownAlbumId = albums[0].id;
468
469        upload.status = 'success';
470        hideAfter(3);
471      } else {
472        // Otherwise, show the errors
473        upload.error = `Erreur : ${response.errors[0]}`;
474        upload.status = 'error';
475        hideAfter(10);
476      }
477    } else if (xhr.status === 413) {
478      upload.error = 'Les fichiers téléversés sont trop volumineux et
        sont refusés par le serveur. \nVeuillez tenter d'envoyer moins de
        fichiers à la fois.';
479      upload.status = 'error';
480      hideAfter(10);
481    } else if (xhr.status !== 0) {
482      upload.error = `Erreur HTTP ${xhr.status}. Veuillez réessayer.`;
483    ;
484      upload.status = 'error';
485      hideAfter(10);
486    } else {
487      upload.error = 'Erreur inconnue. Veuillez réessayer.';
488      upload.status = 'error';
489      hideAfter(10);
490    }
491  };
492
493  // Build the request and send it
494  const formData = new FormData();
495  files.forEach((file) => {
496    formData.append('files[]', file);
497  });
498  xhr.send(formData);
499
500 /**
501   * Handles the event triggered when the user drops some files on the
502   * window.
503   * @param {drop} $event Drop event
504   */
505  onDrop($event) {
506    this.upload($event.dataTransfer.files);
507  },
508};
509</script>
510
511<style lang="scss">
```

```
512  @import 'stylesheets/fonts.scss';
513  @import 'stylesheets/errorPage.scss';
514
515  html, body, .root {
516    height: 100%;
517    margin: 0;
518    padding: 0;
519  }
520
521  * {
522    box-sizing: border-box;
523  }
524
525  body {
526    background-color: #ffffff;
527    color: #333;
528    font-family: 'Inter', -apple-system, BlinkMacSystemFont, 'Segoe UI',
529    Roboto, sans-serif;
530  }
531
532  h2 {
533    margin-top: 0;
534    margin-bottom: .5em;
535    font-size: 36px;
536    font-weight: 900;
537  }
538
539  .root {
540    display: grid;
541    grid-template-rows: auto 1fr;
542    grid-template-columns: 1fr 320px;
543    flex-direction: column;
544  }
545
546  .navbar {
547    grid-column: -1 / 1;
548
549    display: flex;
550    justify-content: space-between;
551    align-items: center;
552    padding: 10px;
553
554    height: 60px;
555    flex-shrink: 0;
556    border-bottom: 1px solid #dedede;
557    background-color: #f5f5f5;
558  }
559
560  .search-box {
561    display: block;
562    width: 250px;
563    height: 40px;
564
565    background-image: url(/resources/img/icon-search.svg);
```

```
565     background-repeat: no-repeat;
566     background-position: 9px center;
567
568     border: 1px solid #dedede;
569     border-radius: 5px;
570
571     color: inherit;
572     font-family: inherit;
573     font-size: 14px;
574     padding-left: 31px;
575
576     transition: .1s ease-in-out background-color;
577
578     -webkit-appearance: textfield; // Fix appearance in Safari
579
580     &:focus {
581         outline: 0;
582         border: 2px solid $primary;
583         padding-left: 30px;
584         background-position: 8px center;
585     }
586
587     &[disabled] {
588         background-color: #f5f5f5;
589         cursor: not-allowed;
590     }
591 }
592
593 .navbar-right {
594     display: flex;
595     align-items: center;
596 }
597
598 .upload-files-form input[type="file"] {
599     // Hide the input element that makes the file selector work
600     position: absolute;
601     left: -9999px;
602     top: -9999px;
603 }
604
605 .upload-files-button {
606     display: block;
607     width: 40px;
608     height: 40px;
609     padding: 0;
610     border: 0;
611
612     background-color: transparent;
613     background-image: url(/resources/img/icon-upload.svg);
614     background-size: cover;
615     font-size: 0;
616
617     &:active {
618         opacity: .8;
```

```
619    }
620  }
621
622  .main-view {
623    position: relative;
624    display: flex;
625    overflow: hidden;
626  }
627
628  .main-view-content {
629    padding: 40px;
630    flex: 1;
631    overflow: hidden;
632
633    &.scrollable {
634      overflow-y: auto;
635    }
636  }
637
638  .main-view-content::after { // Bottom padding
639    display: block;
640    content: '';
641    height: 20px;
642  }
643
644  .albums-list {
645    display: grid;
646    grid-template-columns: repeat(auto-fill, minmax(200px, 1fr));
647    grid-gap: 20px;
648
649    margin: 0;
650    padding: 0;
651    list-style-type: none;
652  }
653
654  .album {
655    display: grid;
656    grid-template-rows: auto 22px 22px;
657
658    .album-cover {
659      display: block;
660      padding-bottom: 100%;
661      margin-bottom: 10px;
662
663      border: 1px solid rgba(0, 0, 0, .2);
664      border-radius: 5px;
665      background-color: #fafafa;
666      background-size: 102%;
667      background-position: center;
668      background-repeat: no-repeat;
669    }
670
671    strong { // Album name
672      display: block;
```

```
673         margin: 0;
674         font-weight: 600;
675     }
676
677     span { // Artist
678         color: #777;
679     }
680
681     strong, span { // Cut too long text
682         overflow: hidden;
683         text-overflow: ellipsis;
684     }
685 }
686
687 .album-fake {
688     strong, span {
689         display: block;
690         border-radius: 50px;
691         margin: 4px 0;
692     }
693
694     strong {
695         background-color: #ccc;
696     }
697
698     span {
699         background-color: #eee;
700     }
701
702     .album-cover.loading {
703         animation: 2s infinite blinkingCover ease-in-out;
704     }
705 }
706
707 @keyframes blinkingCover {
708     0%, 33%, 66%, 100% { background-color: #fafafa }
709     50% { background-color: #fff }
710 }
711
712 .search-results-songs {
713     display: grid;
714     grid-template-columns: repeat(auto-fill, minmax(300px, 1fr));
715     grid-gap: 0 20px;
716
717     padding: 0;
718     margin-bottom: 40px;
719     list-style-type: none;
720
721     font-size: 14px;
722
723     li {
724         position: relative;
725         display: grid;
726         grid-template-columns: auto 1fr;
```

```
727     grid-template-rows: auto auto;
728     align-items: center;
729     grid-gap: 2px 10px;
730     padding: 6px 0;
731
732     img {
733         width: 44px;
734         height: 44px;
735         border: 1px solid rgba(0, 0, 0, .1);
736         border-radius: 3px;
737         grid-row: -1 / 1;
738     }
739
740     &:after { // Separator line
741         position: absolute;
742         bottom: 0;
743         right: 0;
744         width: 100%;
745
746         display: block;
747         content: '';
748         border-bottom: 1px solid #dedede;
749         grid-column: 2;
750     }
751
752     .play-now {
753         position: absolute;
754         top: 13px;
755         left: 6px;
756         width: 30px;
757         height: 30px;
758     }
759
760     button {
761         padding: 0;
762         border: 0;
763
764         background-color: transparent;
765         background-size: cover;
766         font-size: 0;
767         border-radius: 50%;
768
769         &:active {
770             filter: brightness(75%);
771         }
772     }
773
774     .play-now,
775     .song-waitlist-buttons {
776         opacity: 0;
777         transition: .15s ease-in-out opacity;
778     }
779
780     &:hover .play-now,
```

```
781     &:hover .song-waitlist-buttons {  
782         opacity: 1;  
783     }  
784  
785     button.play-now {  
786         background-image: url(/resources/img/icon-play-round.svg);  
787     }  
788  
789     button.play-next {  
790         background-image: url(/resources/img/icon-play-next.svg);  
791     }  
792  
793     button.play-later {  
794         background-image: url(/resources/img/icon-play-later.svg);  
795     }  
796  
797     .song-waitlist-buttons {  
798         position: absolute;  
799         top: 0;  
800         right: 0;  
801         background-color: #ffffff;  
802         padding: 13px 10px;  
803  
804         button {  
805             width: 30px;  
806             height: 30px;  
807             margin-left: 5px;  
808             background-color: #fafafa;  
809         }  
810     }  
811 }  
812  
813     strong {  
814         font-weight: 600;  
815         align-self: flex-end;  
816     }  
817  
818     span {  
819         color: #555;  
820     }  
821  
822     strong, span {  
823         overflow: hidden;  
824         text-overflow: ellipsis;  
825     }  
826 }  
827  
828     .no-search-results,  
829     .no-music {  
830         display: grid;  
831         align-items: center;  
832         justify-content: center;  
833         height: 100%;  
834 }
```

```
835     color: #555;
836     font-weight: 300;
837     font-size: 20px;
838     text-align: center;
839
840     p {
841         hyphens: auto;
842     }
843
844     img {
845         width: 80px;
846         height: 80px;
847     }
848 }
849
850 .no-music-help {
851     font-weight: 400;
852     font-size: 14px;
853 }
854
855 .fade-enter-active, .fade-leave-active {
856     transition: .3s ease-in-out opacity;
857 }
858
859 .fade-enter, .fade-leave-to {
860     opacity: 0;
861 }
862 </style>
863
```

```

1 <!-- Music player and waitlist -->
2
3 <template>
4   <aside>
5     <section class="player">
6       <audio
7         ref="audio"
8         @play="updateProgress"
9         @pause="updateProgress"
10        @error="updateProgress"
11        @ended="$emit('next')"
12        @loadedmetadata="updateProgress"
13      ></audio>
14
15     <div
16       class="cover-art"
17       :style="nowPlaying
18       ? {
19         backgroundImage: `url(/resources/img/cover.php?id=${currentSong.album.id})`
20       }
21       : {}"
22     ></div>
23
24     <div v-if="duration > 0" class="playback-progress">
25       <input
26         type="range"
27         min="0"
28         max="1000"
29         :value="currentTime / duration * 1000"
30         @input="changeProgress"
31         ref="progressSlider"
32       >
33       <div class="playback-progress-indicators">
34         <span>{{ currentTime | time }}</span>
35         <span>&ndash;{{ remainingTime | time }}</span>
36       </div>
37     </div>
38
39     <!-- Current song information -->
40     <div class="now-playing" v-if="currentSong">
41       <strong class="song-title">
42         {{ currentSong.title }}
43       </strong>
44       <p class="song-subtitle">
45         {{ currentSong.album.artist }}
46         &mdash;
47         {{ currentSong.album.title }}
48       </p>
49     </div>
50
51     <!-- Controls -->
52     <div class="controls">
53       <button :disabled="history.length === 0" @click="$emit('previous')"

```

```

53  ">
54      
55  </button>
56  <button
57      class="play-button"
58      @click="togglePlayback"
59      :disabled="waitlist.length === 0 && !nowPlaying"
60  >
61      
62      
63  </button>
64  <button @click="$emit('next')" :disabled="waitlist.length === 0
65      && !nowPlaying">
66      
67  </button>
68  </div>
69  </section>
70
71  <section class="waitlist">
72      <h3>Liste d'attente</h3>
73      <button
74          class="clear-waitlist"
75          v-if="waitlist.length > 0"
76          @click="$emit('clearWaitlist')"
77      >
78          Vider
79      </button>
80
81      <ul class="waitlist-songs">
82          <li
83              v-for="(song, index) in waitlistSongs"
84              :key="`waitlist-${index}`"
85          >
86              
91              <strong>{{ song.title }}</strong>
92              <span>{{ song.album.artist }} &mdash; {{ song.album.title }}<
93          span>
94      </li>
95  </ul>
96  </section>
97
98  <ul class="uploads" v-if="uploads.length > 0">
99      <h3>Importation</h3>
100     <li v-for="upload in uploads" :key="`upload-${upload.id}`">
101         <strong>{{ upload.name }}</strong>
102         <progress
103             v-if="upload.status === 'progress'"
104             :value="upload.progress * 100"
105             max="100"

```

```
104          ></progress>
105          <span class="upload-error" v-if="upload.error">
106              {{ upload.error }}
107          </span>
108          <div class="upload-success" v-if="upload.status === 'success'">
109              Importé
110          </div>
111      </li>
112  </ul>
113 </aside>
114 </template>
115
116 <script>
117 export default {
118     props: {
119         library: Object,
120         nowPlaying: Number,
121         waitlist: Array,
122         uploads: Array,
123         history: Array,
124     },
125
126     data: () => ({
127         currentTime: 0, // seconds played
128         duration: NaN, // total seconds
129
130         updateProgressInterval: null,
131         isPlaying: false,
132     }),
133
134     created() {
135         this.updateProgressInterval = setInterval(() => {
136             this.updateProgress();
137         }, 1000);
138     },
139
140     destroyed() {
141         clearInterval(this.updateProgressInterval);
142     },
143
144     computed: {
145         /**
146         * The song currently playing
147         * @return {Object|null}
148         */
149         currentSong() {
150             if (!this.nowPlaying) {
151                 return null;
152             }
153
154             return this.library.songs.find(song => song.id === this.nowPlaying)
155         || null;
156     },
157 }
```

```

157  /**
158   * The remaining playback time (in seconds)
159   * @returns {Number}
160   */
161   remainingTime() {
162     return this.duration - this.currentTime;
163   },
164
165  /**
166   * The songs in the waitlist
167   * @return {Array}
168   */
169   waitlistSongs() {
170     return this.waitlist
171       .map(songId => this.library.songs.find(song => song.id === songId
172     ));
173   },
174
175   methods: {
176     /**
177      * Play or pause the current song
178      */
179     togglePlayback() {
180       const { audio } = this.$refs;
181
182       if (this.isPlaying) {
183         audio.pause();
184       } else {
185         if (!this.nowPlaying) {
186           // "Play" pressed and no track currently played: go to the next
187           // track.
188           this.$emit('next');
189         }
190
191         audio.play();
192
193       this.updateProgress();
194     },
195
196     /**
197      * Update the playback progress indicators from the audio element
198      */
199     updateProgress() {
200       const { audio } = this.$refs;
201       this.currentTime = audio.currentTime;
202       this.duration = audio.duration;
203       this.isPlaying = !audio.paused && !audio.error;
204     },
205
206     /**
207      * Updates the audio playback progress from the slider value
208      */

```

```

209     changeProgress() {
210         const sliderValue = this.$refs.progressSlider.value;
211         const newProgress = sliderValue / 1000;
212         this.$refs.audio.currentTime = newProgress * this.duration;
213
214         this.updateProgress();
215     },
216 },
217
218 filters: {
219     /**
220      * Formats a time under the {m:ss} format
221      * @param {Number} seconds
222      */
223     time(seconds) {
224         let secs = seconds || 0;
225         secs = Math.round(secs);
226
227         const m = Math.floor(secs / 60);
228         const ss = (secs % 60).toString().padStart(2, '0');
229
230         return `${m}:${ss}`;
231     },
232 },
233
234 watch: {
235     /**
236      * Current track changed
237      */
238     nowPlaying() {
239         const { audio } = this.$refs;
240
241         if (!this.currentSong) {
242             audio.pause();
243             this.playProgress = 0;
244             this.duration = NaN;
245             return;
246         }
247
248         audio.src = `/resources/audio.php?id=${this.currentSong.id}`;
249         audio.play();
250     },
251 },
252 };
253 </script>
254
255 <style scoped lang="scss">
256     @import '../stylesheets/inputrange.scss';
257
258     aside {
259         display: grid;
260         grid-template-rows: auto 1fr auto;
261         overflow-y: auto;
262

```

```
263     background-color: #fafafa;
264     border-left: 1px solid #dedede;
265 }
266
267 .player {
268     padding: 20px;
269     text-align: center;
270 }
271
272 .cover-art {
273     width: 255px;
274     height: 255px;
275     background-color: #fafafa;
276     background-image: linear-gradient(to bottom, #fefefe, #f1f5f8);
277     background-size: contain;
278     background-position: center;
279     border-radius: 5px;
280     box-shadow: 0 5px 15px rgba(0, 0, 0, .2),
281     0 2px 5px rgba(0, 0, 0, .05);
282     margin: 0 auto;
283     margin-top: 10px;
284     margin-bottom: 20px;
285 }
286
287 .waitlist {
288     position: relative;
289     padding: 20px;
290     background-color: #f5f5f5;
291     border-top: 1px solid #dedede;
292     overflow-y: auto;
293
294     .clear-waitlist {
295         position: absolute;
296         top: 19px;
297         right: 20px;
298
299         padding: 0;
300         background: transparent;
301         border: 0;
302
303         color: $primary;
304         font-weight: bold;
305         font-family: inherit;
306
307         &:active {
308             opacity: .8;
309         }
310     }
311 }
312
313 .waitlist h3,
314 .uploads h3 {
315     font-size: 12px;
316     color: #939393;
```

```
317     margin: 0;
318 }
319
320 .waitlist-songs {
321     margin: 5px 0;
322     padding: 0;
323     margin-bottom: 40px;
324     list-style-type: none;
325
326     font-size: 14px;
327
328     li {
329         position: relative;
330         display: grid;
331         grid-template-columns: auto 1fr;
332         grid-template-rows: auto auto;
333         align-items: center;
334         grid-gap: 2px 10px;
335         padding: 6px 0;
336
337         img {
338             width: 44px;
339             height: 44px;
340             border: 1px solid rgba(0, 0, 0, .1);
341             border-radius: 3px;
342             grid-row: -1 / 1;
343         }
344
345         &:after { // Separator line
346             position: absolute;
347             bottom: 0;
348             right: 0;
349             width: 100%;
350
351             display: block;
352             content: '';
353             border-bottom: 1px solid #dedede;
354             grid-column: 2;
355         }
356     }
357
358     strong {
359         font-weight: 600;
360         align-self: flex-end;
361     }
362
363     span {
364         color: #555;
365     }
366
367     strong, span {
368         display: block;
369         overflow: hidden;
370         text-overflow: ellipsis;
```

```
371     }
372 }
373
374 .uploads {
375   margin: 0;
376   padding: 10px 20px;
377   background-color: #f0f0f0;
378   border-top: 1px solid #dedede;
379   list-style-type: none;
380
381   color: #333;
382   font-size: 12px;
383
384   li {
385     margin-top: 10px;
386   }
387
388   strong {
389     display: block;
390     font-weight: 600;
391     margin: 5px 0;
392   }
393
394   progress {
395     width: 100%;
396   }
397 }
398
399 .upload-error,
400 .upload-success {
401   display: block;
402   font-weight: 500;
403
404   padding-left: 20px;
405   background-size: 15px;
406   background-position: 0 0;
407   background-repeat: no-repeat;
408 }
409
410 .upload-success {
411   color: darken($success, 15%);
412   background-image: url(/resources/img/icon-success.svg);
413 }
414
415 .upload-error {
416   color: darken($danger, 15%);
417   background-image: url(/resources/img/icon-error.svg);
418 }
419
420 .playback-progress {
421   margin-top: -15px;
422   margin-bottom: 15px;
423
424 // Custom input[type="range"] styles
```

```
425     input[type="range"] {
426         @extend .custom-range;
427         margin: 5px 0;
428     }
429 }
430
431 .playback-progress-indicators {
432     display: flex;
433     justify-content: space-between;
434
435     color: #777;
436     font-size: 12px;
437     font-weight: 600;
438     font-feature-settings: 'tnum' 1; // fixed width numerals
439 }
440
441 .now-playing {
442     display: grid;
443
444     .song-title, .song-subtitle {
445         display: block;
446         overflow: hidden;
447         text-overflow: ellipsis;
448     }
449
450     .song-subtitle {
451         color: $primary;
452         margin: 2px 0;
453     }
454 }
455
456 .controls {
457     display: flex;
458     align-items: center;
459     justify-content: center;
460     margin-top: 24px;
461     margin-bottom: 6px;
462
463     button {
464         position: relative;
465         width: 24px;
466         height: 24px;
467         padding: 0;
468         background: transparent;
469         border: 0;
470         transition: .15s ease-in-out transform;
471
472         img {
473             width: 100%;
474             height: 100%;
475         }
476
477         &:focus {
478             outline: 0;
```

```
479      }
480
481      &:active {
482          transform: scale(1.1);
483      }
484
485      &[disabled] {
486          opacity: .3;
487          cursor: not-allowed;
488      }
489
490      &::--moz-focus-inner {
491          border: 0;
492      }
493  }
494
495  .play-button {
496      width: 36px;
497      height: 36px;
498      margin: 0 36px;
499  }
500 }
501
502 .controls button {
503     &::after {
504         position: absolute;
505         display: block;
506         content: '';
507         top: 0;
508         left: 0;
509         width: 100%;
510         height: 100%;
511         border-radius: 50%;
512         background-color: rgba(0, 0, 0, .2);
513         pointer-events: none;
514         transform: scale(1.8);
515         opacity: 0;
516         transition: .1s ease-in-out opacity, .1s ease-in-out transform;
517     }
518
519     &:focus::after {
520         opacity: .5;
521     }
522
523     &:active::after {
524         opacity: 1;
525         transform: scale(1.55);
526     }
527 }
528 </style>
529
```

```
1 <!-- Shows the user avatar, name and logout link. -->
2
3 <template>
4   <div class="user-menu">
5     <button
6       class="user-pic"
7       :style="buttonStyles"
8       aria-haspopup="true"
9       tabindex="0"
10    >
11      Ouvrir le menu
12    </button>
13    <div class="user-menu-content-wrapper">
14      <div v-if="user" class="user-menu-content">
15        <header>
16          <h4>{{ user.fullName }}</h4>
17          <p>@{{ user.username }}</p>
18        </header>
19        <form method="POST" action="logout.php">
20          <button class="logout-button" type="submit">
21            Se déconnecter
22          </button>
23        </form>
24      </div>
25    </div>
26  </div>
27 </template>
28
29 <script>
30 export default {
31   props: ['user'],
32
33   computed: {
34     buttonStyles() {
35       return this.user
36         ? { backgroundImage: `url(${this.user.username}/medium)` }
37         : {};
38     },
39   },
40 };
41 </script>
42
43 <style scoped lang="scss">
44   .user-menu {
45     position: relative;
46   }
47
48   .user-pic {
49     display: block;
50     width: 40px;
51     height: 40px;
52     padding: 0;
53     margin-left: 10px;
```

```
54      background: #fafafa;
55      background-size: 102%;
56      background-repeat: no-repeat;
57      border-radius: 50%;
58      border: 1px solid rgba(0, 0, 0, .2);
59
60      font-size: 0;
61  }
62 }
63
64 .user-menu-content-wrapper {
65   position: absolute;
66   top: 35px;
67   right: 0;
68   z-index: 20;
69   padding-top: 20px;
70
71   opacity: 0;
72   pointer-events: none;
73   transition: .2s ease-in-out opacity;
74 }
75
76 .user-menu-content {
77   width: 200px;
78   background-color: #fff;
79   box-shadow: 0 5px 15px rgba(0, 0, 0, .2);
80   border-radius: 5px;
81
82   header {
83     padding: 10px 20px;
84
85     h4, p {
86       margin: 5px 0;
87     }
88   }
89 }
90
91 .user-pic:hover + .user-menu-content-wrapper,
92 .user-pic:focus + .user-menu-content-wrapper,
93 .user-menu:focus-within .user-menu-content-wrapper,
94 .user-menu-content-wrapper:hover { // Open the menu
95   opacity: 1;
96   pointer-events: all;
97 }
98
99 .logout-button {
100   display: block;
101   width: 100%;
102   background: transparent;
103   border: 0;
104   border-top: 1px solid #dedede;
105   padding: 10px 20px;
106
107   color: $primary;
```

```
108     font-family: inherit;  
109     font-size: inherit;  
110     text-align: left;  
111  
112     transition: .15s ease-in-out background-color;  
113     cursor: pointer;  
114  
115     &:hover {  
116         background-color: rgba(51, 51, 51, .05);  
117     }  
118 }  
119 </style>  
120
```

```
1 <!--
2 This page displays the details of an album.
3 The component also manages the album's edit capabilities.
4 -->
5
6 <template>
7   <div class="album-page-overlay" @click.self="close">
8     <form
9       class="album-page"
10      :class="{ 'album-page-editing': !!edit }"
11      @submit.prevent="sendEdit"
12      @keydown.esc="!!edit && cancelEdit()"
13    >
14      <header>
15        <img :src=`/resources/img/cover.php?id=${id}` alt="Couverture d'album">
16        <button
17          type="button"
18          class="play-now-button"
19          v-if="!edit"
20          @click="play({ music: allSongIds })"
21        >
22          Jouer
23        </button>
24
25      <template v-if="!edit">
26        <div class="album-title">
27          <h2>{{ album.title }}</h2>
28          <span>{{ album.artist }}</span>
29        </div>
30
31        <div class="album-action-buttons">
32          <div>
33            <button
34              type="button"
35              class="play-next"
36              aria-describedby="play-next-label"
37              @click="play({ music: allSongIds, when: 'next' })"
38            ></button>
39            <span id="play-next-label">
40              Jouer à la suite
41            </span>
42          </div>
43          <div>
44            <button
45              type="button"
46              class="play-later"
47              aria-describedby="play-later-label"
48              @click="play({ music: allSongIds, when: 'later' })"
49            ></button>
50            <span id="play-later-label">
51              Jouer plus tard
52            </span>
53          </div>
```

```
54          <hr>
55          <div>
56              <button
57                  type="button"
58                  class="edit-album"
59                  aria-describedby="edit-album-label"
60                  @click="startEdit"
61              ></button>
62              <span id="edit-album-label">
63                  Modifier
64              </span>
65          </div>
66      </div>
67  </template>
68  <template v-else>
69      <div class="album-title-edit">
70          <input
71              class="form-input form-input-big"
72              type="text"
73              v-model="edit.title"
74              placeholder="Titre"
75              maxlength="60"
76              required
77          >
78          <input
79              class="form-input"
80              type="text"
81              v-model="edit.artist"
82              placeholder="Artiste"
83              maxlength="60"
84              required
85          >
86      </div>
87      <div class="album-action-buttons">
88          <div>
89              <button
90                  type="button"
91                  class="trash-button"
92                  aria-describedby="trash-label"
93                  @click="edit.deleteModalVisible = true"
94              ></button>
95              <span id="trash-label">
96                  Supprimer
97              </span>
98          </div>
99      </div>
100     </template>
101  </header>
102
103  <div class="album-page-content">
104      <!-- Album songs -->
105      <template v-if="!edit">
106          <ul v-if="songs.length > 0" class="album-songs">
107              <li
```

```

108          :class="{ 'highlighted-song': song.id ===
109            highlightedTrackId }"
110          v-for="song in songs"
111          :key="`song-${song.id}`"
112          >
113          <span class="track-number">
114            {{ song.trackNumber }}
115          </span>
116          <button type="button" @click="play({ music: song.id })"
117            class="play-now-button">
118            Jouer
119          </button>
120          <span class="song-title">{{ song.title }}</span>
121
122          <div class="song-waitlist-buttons">
123            <button
124              type="button"
125              class="play-next"
126              @click="play({ music: song.id, when: 'next' })"
127            >
128              Jouer à la suite
129            </button>
130            <button
131              type="button"
132              class="play-later"
133              @click="play({ music: song.id, when: 'later' })"
134            >
135              Jouer plus tard
136            </button>
137          </div>
138        </li>
139      <p class="album-empty" v-else>
140        Cet album ne contient aucun morceau.
141      </p>
142    </template>
143
144    <!-- Edit album songs -->
145    <template v-else>
146      <ul v-if="edit.songs.length > 0" class="album-songs">
147        <li
148          class="song-edit"
149          v-for="(song, songIndex) in edit.songs"
150          :key="`edit-song-${song.id}`"
151        >
152          <input
153            class="track-number"
154            type="number"
155            v-model.number="song.trackNumber"
156            maxlength="2"
157            placeholder="N° "
158            min="0"
159            max="99"
160            required

```

```

160          >
161          <input
162              class="track-title"
163              type="text"
164              v-model="song.title"
165              placeholder="Titre"
166              maxlength="60"
167              required
168          >
169          <button class="song-delete" type="button" @click="$delete(
  edit.songs, songIndex)">
170              Supprimer
171          </button>
172      </li>
173  </ul>
174  <p class="album-empty" v-else>
175      Cet album ne contient aucun morceau.
176  </p>
177 </template>
178 </div>
179
180 <footer class="edit-action-buttons" v-if="edit">
181     <button type="button" @click="cancelEdit">
182         Annuler
183     </button>
184     <button type="submit" class="primary" :disabled="edit.sent">
185         {{ !edit.sent ? 'OK' : 'Chargement...' }}
186     </button>
187 </footer>
188
189 <transition name="fade">
190     <div class="delete-modal" v-if="edit && edit.deleteModalVisible">
191         <p>
192             Voulez-vous vraiment supprimer
193             <strong>&laquo;&nbsp;{{ album.title }}&nbsp;&raquo;</strong>&
nbsp;?
194         </p>
195
196         <footer class="delete-modal-action-buttons" v-if="edit">
197             <button type="button" @click="edit.deleteModalVisible = false
  ">
198                 Annuler
199             </button>
200             <button
201                 type="button"
202                 class="danger"
203                 :disabled="edit.deleteModalSent"
204                 @click="deleteAlbum"
205             >
206                 {{ !edit.sent ? 'Supprimer' : 'Chargement...' }}
207             </button>
208         </footer>
209     </div>
210 </transition>

```

```
211      </form>
212    </div>
213  </template>
214
215 <script>
216   export default {
217     props: {
218       library: Object,
219       id: Number, // Album ID
220       highlightedTrackId: Number,
221     },
222
223     data: () => {
224       // Contains the value of the edited fields (see startEdit())
225       edit: null,
226     },
227
228     computed: {
229       /**
230        * Metadata of the shown album
231        * @returns {Object}
232        */
233       album() {
234         return this.library.albums
235           .find(album => album.id === this.id);
236       },
237
238       /**
239        * Songs in the shown album
240        * @returns {Array}
241        */
242       songs() {
243         return this.library.songs
244           .filter(song => song.album.id === this.id)
245           .sort((a, b) => a.trackNumber - b.trackNumber);
246       },
247
248       /**
249        * The IDs of all the songs
250        */
251       allSongIds() {
252         return this.songs.map(song => song.id);
253       },
254     },
255
256     methods: {
257       /**
258        * Close the album page
259        */
260       close() {
261         this.$emit('close');
262       },
263
264       /**
```

```

265     * Sends a "play" event to the main view
266     * @param {{test: Number|Number[], when: String?}} params
267     */
268     play(params) {
269         this.$emit('play', params);
270     },
271
272     /**
273      * Toggles on the edit mode
274      */
275     startEdit() {
276         this.edit = {
277             title: this.album.title,
278             artist: this.album.artist,
279             songs: this.songs.map(song => ({
280                 ...song, // Copy the song object to edit it
281             })),
282
283             sent: false,
284
285             deleteModalVisible: false,
286             deleteModalSent: false,
287         };
288     },
289
290     /**
291      * Quits the edit mode without saving changes
292      */
293     cancelEdit() {
294         this.edit = null;
295     },
296
297     /**
298      * Saves the modifications made with the edit mode
299      */
300     sendEdit() {
301         this.edit.sent = true;
302
303         const { title, artist, songs } = this.edit;
304
305         const formData = new FormData();
306         formData.append('id', this.id);
307         formData.append('title', title);
308         formData.append('artist', artist);
309
310         songs.forEach((song, index) => {
311             formData.append(`songs[${index}][id]`, song.id);
312             formData.append(`songs[${index}][title]`, song.title);
313             formData.append(`songs[${index}][trackNumber]`, song.trackNumber)
314         });
315
316         fetch('/api/edit.php', {
317             method: 'POST',

```

```

318         body: formData,
319     })
320         .then(r => r.json())
321         .then((newValues) => {
322             this.$emit('saved', newValues);
323             this.edit = null;
324         });
325     },
326
327     /**
328      * Deletes the album.
329      */
330     deleteAlbum() {
331         const { id } = this;
332
333         const body = new FormData();
334         body.append('id', id);
335
336         fetch('/api/delete.php', { method: 'POST', body })
337             .then((res) => {
338                 if (res.status !== 200) {
339                     return;
340                 }
341
342                 this.$emit('deleted', { id });
343             });
344     },
345 },
346 };
347 </script>
348
349 <style scoped lang="scss">
350     .album-page-overlay {
351         position: absolute;
352         top: 0;
353         left: 0;
354
355         display: grid;
356         align-items: center;
357         justify-content: center;
358         width: 100%;
359         height: 100%;
360         padding: 20px;
361
362         background-color: $modalOverlayBackground;
363     }
364
365     .album-page {
366         position: relative;
367         display: flex;
368         flex-direction: column;
369         width: 700px;
370         max-height: 100%;
371         margin: 20px 0;

```

```
372     background: #ffffff;
373     border-radius: 16px;
374     overflow: hidden;
375   }
376
377   header {
378     position: relative;
379     display: grid;
380     grid-template-columns: auto 1fr auto;
381     grid-gap: 20px;
382     align-items: center;
383     padding: 10px;
384     background-color: #f5f5f5;
385
386     img { // Album cover
387       width: 100px;
388       height: 100px;
389       border-radius: 5px;
390       border: 1px solid rgba(0, 0, 0, .1);
391       object-fit: contain;
392     }
393
394     .play-now-button {
395       position: absolute;
396       left: 30px;
397       top: 30px;
398       width: 60px;
399       height: 60px;
400       background-size: 60px;
401     }
402
403     &:hover .play-now-button {
404       opacity: 1;
405     }
406   }
407
408   .album-title {
409     font-size: 18px;
410
411     h2 {
412       font-size: inherit;
413       font-weight: 700;
414       margin: 2px 0;
415     }
416
417     span {
418       color: #555;
419     }
420
421     h2, span {
422       overflow: hidden;
423       text-overflow: ellipsis;
424     }
425   }
```

```
426
427 .album-page-content {
428   flex: 1;
429   overflow-y: auto;
430   padding: 20px;
431 }
432
433 .album-songs {
434   column-count: 2;
435   column-gap: 20px;
436
437   margin: 0;
438   padding: 0;
439   list-style-type: none;
440
441   li {
442     position: relative;
443     display: grid;
444     grid-template-columns: 2.5em 1fr auto;
445     grid-gap: .5em;
446     align-items: center;
447     height: 50px;
448     border-bottom: 1px solid #dedede;
449
450     break-inside: avoid;
451     page-break-inside: avoid;
452     -webkit-column-break-inside: avoid;
453
454     transition: .15s ease-in-out background-color;
455   }
456
457   .play-now-button {
458     position: absolute;
459     top: 10px;
460     left: 10px;
461     width: 30px;
462     height: 30px;
463   }
464
465   .album-page:not(.album-page-editing) & {
466     li:hover,
467     li:focus,
468     li:focus-within {
469       background-color: #fafafa;
470
471       .track-number {
472         opacity: 0;
473       }
474
475       .song-waitlist-buttons,
476       .play-now-button {
477         opacity: 1;
478       }
479     }
```

```
480    }
481
482    li:hover,
483    li:focus,
484    li:focus-within {
485        .song-delete {
486            opacity: 1;
487        }
488    }
489 }
490
491 .song-edit input {
492     display: block;
493     width: 100%;
494     height: 40px;
495
496     color: inherit;
497     font-family: inherit;
498     font-size: inherit;
499     background: transparent;
500
501     border: 2px solid transparent;
502     border-radius: 4px;
503
504     transition: .15s ease-in-out border-color;
505
506     &:hover {
507         border-color: #dedede;
508     }
509
510     &:focus {
511         outline: 0;
512         border-color: $primary;
513     }
514
515     // Hide the +/- buttons on input[type="number"]
516     -moz-appearance: textfield;
517
518     &::-webkit-outer-spin-button,
519     &::-webkit-inner-spin-button {
520         display: none;
521     }
522 }
523
524 .track-number,
525 .song-edit .track-number {
526     text-align: right;
527     color: #777;
528     font-feature-settings: 'tnum' 1; // fixed width numerals
529     transition: .15s ease-in-out opacity;
530 }
531
532 .song-title {
533     overflow: hidden;
```

```
534     text-overflow: ellipsis;
535 }
536
537 .play-now-button,
538 .album-action-buttons button,
539 .song-waitlist-buttons button {
540     padding: 0;
541     border: 0;
542
543     background-color: transparent;
544     background-size: 30px;
545     background-position: center;
546     font-size: 0;
547     border-radius: 50%;
548
549     @include focusable;
550     @include activable;
551 }
552
553 .play-now-button {
554     background-image: url(/resources/img/icon-play-round.svg);
555 }
556
557 .play-now-button,
558 .song-waitlist-buttons,
559 .song-delete {
560     opacity: 0;
561     transition: .15s ease-in-out opacity;
562 }
563
564 .album-action-buttons {
565     display: flex;
566     align-items: center;
567
568     hr {
569         display: block;
570         width: 1px;
571         height: 50px;
572         margin: 0 10px;
573
574         border: 0;
575         background-color: #dedede;
576     }
577
578     div {
579         position: relative;
580         padding: 0 10px;
581
582         &:hover span,
583         &:focus-within span {
584             opacity: 1;
585             transform: none;
586         }
587     }
588 }
```

```
588     span { // Description label
589         position: absolute;
590         top: 58px;
591         left: -15px;
592         width: 100px;
593
594         color: #555;
595         text-align: center;
596         font-size: 12px;
597
598         transform: translateY(-5px);
599         opacity: 0;
600         transition: .15s ease-in-out opacity, .15s ease-in-out transform;
601     }
602 }
603
604     button {
605         width: 50px;
606         height: 50px;
607         background-color: #fafafa;
608         background-size: 50px;
609     }
610 }
611
612 .album-empty {
613     margin: 0;
614     color: #555;
615     font-style: italic;
616 }
617
618 .highlighted-song {
619     animation: 1.5s highlightedSong;
620 }
621
622 @keyframes highlightedSong {
623     0%, 33.2%, 66%, 100% {
624         background: transparent;
625     }
626
627     16.6%, 50%, 83.3% {
628         background-color: #f5f5f5;
629     }
630 }
631
632 .song-waitlist-buttons {
633     position: absolute;
634     right: 0;
635     top: 2px;
636     padding: 8px 10px;
637     background-color: #fafafa;
638
639     button {
640         width: 30px;
641         height: 30px;
```

```
642         margin-left: 5px;
643         background-color: #fafafa;
644     }
645 }
646
647 button.play-next {
648     background-image: url(/resources/img/icon-play-next.svg);
649 }
650
651 button.play-later {
652     background-image: url(/resources/img/icon-play-later.svg);
653 }
654
655 button.edit-album {
656     background-image: url(/resources/img/icon-edit.svg);
657     @include focusableNeutral;
658 }
659
660 button.trash-button {
661     background-image: url(/resources/img/icon-trash.svg);
662     @include focusableNeutral;
663 }
664
665 .edit-action-buttons {
666     background-color: #f5f5f5;
667     padding: 20px;
668 }
669
670 .edit-action-buttons,
671 .delete-modal-action-buttons {
672     display: grid;
673     grid-template-columns: 1fr 1fr;
674     grid-gap: 20px;
675
676     button {
677         height: 48px;
678         padding: 10px 15px;
679         background-color: white;
680         border: 0;
681         border-radius: 8px;
682
683         color: $primary;
684         font-family: inherit;
685         font-weight: 600;
686         font-size: 20px;
687
688         &[disabled] {
689             filter: brightness(90%);
690         }
691
692         @include focusable;
693         @include activable;
694     }
695 }
```

```
696     .primary {
697         background-color: $primary;
698         color: white;
699         @include focusableNeutral;
700     }
701
702     .danger {
703         background-color: $danger;
704         color: white;
705         @include focusableNeutral;
706     }
707 }
708
709 .album-title-edit .form-input {
710     display: block;
711     width: 100%;
712     height: 30px;
713
714     background-color: white;
715     border: 1px solid #dedede;
716     border-radius: 5px;
717
718     color: #555;
719     font-family: inherit;
720     font-size: inherit;
721     padding: 0 5px;
722     margin: 4px 0;
723
724     transition: .1s ease-in-out background-color;
725
726     @include focusable;
727
728     &:focus {
729         padding: 0 4px;
730     }
731
732     &.form-input-big {
733         color: #333;
734         font-size: 18px;
735         font-weight: 700;
736         height: 40px;
737     }
738 }
739
740 .song-delete {
741     display: block;
742     width: 20px;
743     height: 20px;
744
745     background-image: url(/resources/img/icon-remove.svg);
746     background-size: 20px;
747     background-position: center;
748
749     border: 0;
```

```
750 border-radius: 50%;  
751  
752 font-size: 0;  
753  
754 @include activable;  
755 @include focusableNeutral;  
756 }  
757  
758 .delete-modal {  
759   position: absolute;  
760   top: 0;  
761   left: 0;  
762   width: 100%;  
763   height: 100%;  
764   padding: 20px;  
765  
766   display: flex;  
767   flex-direction: column;  
768   align-items: center;  
769   justify-content: center;  
770  
771   background-color: rgba(255, 255, 255, .9);  
772   text-align: center;  
773  
774   p {  
775     color: #555;  
776     font-size: 20px;  
777     font-weight: 400;  
778     line-height: 1.3;  
779  
780     strong {  
781       color: $primary;  
782       font-weight: 900;  
783     }  
784   }  
785 }  
786 </style>  
787
```

```
1 <!-- Shown when the user is not connected -->
2
3 <template>
4   <div class="landing-page">
5     <main>
6       <h1>
7         Bienvenue dans <strong>Interlude</strong>
8       </h1>
9
10      <p>Écoutez votre musique depuis votre navigateur.</p>
11
12      <a class="twitter-login-button" href="/login.php">
13        Se connecter avec Twitter
14      </a>
15    </main>
16  </div>
17 </template>
18
19 <style scoped lang="scss">
20   .landing-page {
21     position: fixed;
22     top: 0;
23     left: 0;
24     width: 100%;
25     height: 100%;
26
27     display: flex;
28     align-items: center;
29     justify-content: center;
30
31     background-color: $modalOverlayBackground;
32     z-index: 100;
33
34     animation: .2s linear fadeIn;
35   }
36
37   @keyframes fadeIn {
38     from {
39       opacity: 0;
40     }
41     to {
42       opacity: 1;
43     }
44   }
45
46   main {
47     width: 700px;
48
49     margin: 20px auto;
50     padding: 80px 20px;
51     background-color: $primary;
52     background: linear-gradient(to bottom, #8264cf, #4a2b99);
53     border-radius: 20px;
54 }
```

```
55     color: #fff;
56     text-align: center;
57   }
58
59   h1 {
60     font-size: 48px;
61     font-weight: 600;
62     color: rgba(255, 255, 255, .6);
63     margin: 5px 0;
64
65     strong {
66       color: #fff;
67       font-weight: 900;
68     }
69   }
70
71   p {
72     font-size: 24px;
73     font-weight: 300;
74   }
75
76   .twitter-login-button {
77     display: inline-block;
78     padding: 15px 25px;
79     padding-left: 68px;
80     border-radius: 100px;
81     background-color: rgba(0, 0, 0, .15);
82     background-image: url(/resources/img/twitter.svg);
83     background-repeat: no-repeat;
84     background-size: 35px;
85     background-position: 20px center;
86
87     font-size: 24px;
88     font-weight: bold;
89     color: inherit;
90     text-decoration: none;
91
92     transition: .1s ease-in-out background-color,
93     .1s ease-in-out filter;
94
95     &:hover {
96       background-color: rgba(0, 0, 0, .25);
97     }
98
99     &:active {
100       background-color: rgba(0, 0, 0, .5);
101     }
102   }
103 </style>
104
```

```
1 <!-- Overlay shown when a file is currently being dragged into the UI -->
2
3 <template>
4   <div>
5     Déposez les fichiers pour les importer dans votre bibliothèque
6     musicale.
7 </template>
8
9 <script>
10 export default {};
11 </script>
12
13 <style scoped lang="scss">
14   div {
15     position: fixed;
16     top: 0;
17     left: 0;
18     width: 100%;
19     height: 100%;
20     z-index: 90;
21
22     display: flex;
23     justify-content: center;
24     align-items: center;
25     padding: 20px;
26
27     color: #777;
28     background-color: rgba(255, 255, 255, .8);
29     border: 10px solid lighten($primary, 10%);
30
31     font-size: 24px;
32     text-align: center;
33
34     pointer-events: none;
35   }
36 </style>
37
```

```
1 // Inter font family
2
3 @font-face {
4   font-family: 'Inter';
5   font-style: normal;
6   font-weight: 300;
7   src: url("/resources/fonts/Inter-Light-BETA.woff2") format("woff2"),
8     url("/resources/fonts/Inter-Light-BETA.woff") format("woff");
9 }
10
11 @font-face {
12   font-family: 'Inter';
13   font-style: normal;
14   font-weight: 400;
15   src: url("/resources/fonts/Inter-Regular.woff2") format("woff2"),
16     url("/resources/fonts/Inter-Regular.woff") format("woff");
17 }
18
19 @font-face {
20   font-family: 'Inter';
21   font-style: italic;
22   font-weight: 400;
23   src: url("/resources/fonts/Inter-Italic.woff2") format("woff2"),
24     url("/resources/fonts/Inter-Italic.woff") format("woff");
25 }
26
27 @font-face {
28   font-family: 'Inter';
29   font-style: normal;
30   font-weight: 600;
31   src: url("/resources/fonts/Inter-SemiBold.woff2") format("woff2"),
32     url("/resources/fonts/Inter-SemiBold.woff") format("woff");
33 }
34
35 @font-face {
36   font-family: 'Inter';
37   font-style: normal;
38   font-weight: 700;
39   src: url("/resources/fonts/Inter-Bold.woff2") format("woff2"),
40     url("/resources/fonts/Inter-Bold.woff") format("woff");
41 }
42
43 @font-face {
44   font-family: 'Inter';
45   font-style: normal;
46   font-weight: 900;
47   src: url("/resources/fonts/Inter-Black.woff2") format("woff2"),
48     url("/resources/fonts/Inter-Black.woff") format("woff");
49 }
50
```

```
1 .error-page {  
2   display: flex;  
3   flex-direction: column;  
4   align-items: center;  
5   justify-content: center;  
6   padding: 20px;  
7  
8   background: linear-gradient(to bottom, #fff, #f1f5f8);  
9   border-top: 10px solid #ff382e;  
10  
11  text-align: center;  
12  
13  h1 {  
14    margin: 0;  
15    font-size: 52px;  
16    background: linear-gradient(to bottom, hsl(3, 100%, 70%), hsl(3, 100%,  
50%));  
17    -webkit-background-clip: text;  
18    -webkit-text-fill-color: transparent;  
19  }  
20  
21  p {  
22    color: #555;  
23    font-weight: 300;  
24    font-size: 24px;  
25  }  
26  
27  footer {  
28    display: flex;  
29  }  
30  
31  .btn {  
32    display: inline-flex;  
33    justify-content: center;  
34    align-items: center;  
35    width: 200px;  
36    height: 50px;  
37    margin: 5px;  
38  
39    font-weight: bold;  
40    font-size: 20px;  
41    border-radius: 200px;  
42  
43    text-align: center;  
44    text-decoration: none;  
45  
46    transition: .1s ease-in-out opacity;  
47  
48  &.btn-primary {  
49    background-color: $primary;  
50    color: #fff;  
51  }  
52  
53  &.btn-secondary {
```

```
54     border: 3px solid $primary;
55     color: $primary;
56   }
57
58   &:hover {
59     opacity: .8;
60   }
61 }
62 }
63
```

```
1 /**
2 * This SASS file contains some variables that can be
3 * used from all the stylesheets of the project.
4 */
5
6 $primary: #674ea7;
7 $danger: #ff3b30;
8 $success: #4cd964;
9 $modalOverlayBackground: rgba(0, 0, 0, .5);
10
11 @mixin focusable($color: $primary) {
12   &:focus {
13     outline: 0;
14     border: 2px solid $color;
15   }
16
17   &:::moz-focus-inner {
18     border: 0;
19   }
20 }
21
22 // Same as $focusable but where the primary color can't be used
23 @mixin focusableNeutral {
24   @include focusable(rgba(0, 0, 0, .7));
25 }
26
27 @mixin activable {
28   &:not([disabled]):active {
29     filter: brightness(80%);
30   }
31 }
32
```

```
1 /**
2 * Interlude's custom <input type="range"> styling.
3 */
4
5 // Based on:
6 //
7 // Styling Cross-Browser Compatible Range Inputs with Sass
8 // Github: https://github.com/darlanrod/input-range-sass
9 // Author: Darlan Rod https://github.com/darlanrod
10 // Version 1.5.1
11 // MIT License
12
13 $track-color: #ddd !default;
14 $thumb-color: #777 !default;
15
16 $thumb-radius: 50% !default;
17 $thumb-height: 6px !default;
18 $thumb-width: 6px !default;
19 $thumb-shadow-size: 0 !default;
20 $thumb-shadow-blur: 0 !default;
21 $thumb-shadow-color: transparent !default;
22 $thumb-border-width: transparent !default;
23 $thumb-border-color: 0 !default;
24
25 $track-width: 100% !default;
26 $track-height: 3px !default;
27 $track-radius: 6px !default;
28
29 @mixin shadow($shadow-size, $shadow-blur, $shadow-color) {
30   box-shadow: $shadow-size $shadow-size $shadow-blur $shadow-color, 0 0
    $shadow-size lighten($shadow-color, 5%);
31 }
32
33 @mixin track {
34   cursor: default;
35   height: $track-height;
36   transition: all .2s ease;
37   width: $track-width;
38   margin: 0;
39 }
40
41 @mixin thumb {
42   @include shadow($thumb-shadow-size, $thumb-shadow-blur, $thumb-shadow-
color);
43   background: $thumb-color;
44   border: $thumb-border-width solid $thumb-border-color;
45   border-radius: $thumb-radius;
46   cursor: default;
47   height: $thumb-height;
48   width: $thumb-width;
49 }
50
51 @mixin disabled {
52   cursor: not-allowed;
```

```
53 }
54
55 .custom-range {
56   -webkit-appearance: none;
57   background: transparent;
58   margin: $thumb-height / 2 0;
59   width: $track-width;
60
61 &::--moz-focus-outer {
62   border: 0;
63 }
64
65 &::--webkit-slider-runnable-track {
66   @include track;
67   background: $track-color;
68   border-radius: $track-radius;
69 }
70
71 &::--webkit-slider-thumb {
72   @include thumb;
73   -webkit-appearance: none;
74   margin-top: $track-height / 2 - $thumb-height / 2;
75 }
76
77 &::--moz-range-track {
78   @include track;
79   background: $track-color;
80   border-radius: $track-radius;
81 }
82
83 &::--moz-range-thumb {
84   @include thumb;
85 }
86
87 &::--ms-track {
88   @include track;
89   background: transparent;
90   border-color: transparent;
91   border-width: ($thumb-height / 2) 0;
92   color: transparent;
93 }
94
95 &::--ms-fill-lower {
96   border-radius: ($track-radius * 2);
97 }
98
99 &::--ms-fill-upper {
100   background: $track-color;
101   border-radius: ($track-radius * 2);
102 }
103
104 &::--ms-thumb {
105   @include thumb;
106   margin-top: $track-height / 4;
```

```
107 }
108
109 &:disabled {
110   &::-webkit-slider-thumb {
111     @include disabled;
112   }
113
114   &::-moz-range-thumb {
115     @include disabled;
116   }
117
118   &::-ms-thumb {
119     @include disabled;
120   }
121
122   &::-webkit-slider-runnable-track {
123     @include disabled;
124   }
125
126   &::-ms-fill-lower {
127     @include disabled;
128   }
129
130   &::-ms-fill-upper {
131     @include disabled;
132   }
133 }
134
135 &::-moz-range-progress {
136   background-color: $thumb-color;
137   border-radius: 10px 0 0 10px;
138 }
139
140 &:focus {
141   outline: 0;
142 }
143 }
144
```

```
1 <?php
2 /**
3  * Interlude's configuration file
4  *
5  * This file must be included with `require_once` in every page accessible
6  * to the users.
7  * It loads the external dependencies and the class autoloader from
8  * Composer,
9  * and sets up the environment.
10 */
11
12 session_start();
13
14 // Load environment variables
15 $dotenv = Dotenv\Dotenv::create(__DIR__ . '/../vendor/autoload.php');
16 $dotenv->load();
17
```

```

1 <?php
2 /**
3  * OAuth callback page
4  * The user will be redirected to this page from Twitter during the login
5  * process.
6 */
7 use Abraham\TwitterOAuth\{TwitterOAuth, TwitterOAuthException};
8 use Interlude\Auth;
9 use Interlude\User;
10
11 require_once __DIR__ . '/../config/config.inc.php';
12
13 if (isset($_GET['denied'])) { // The user denied the authorization request
14     header('Location: loginError.php?error=denied');
15     die();
16 }
17
18 // Restart the login process if it hasn't started
19 $oauthVerifier = filter_input(INPUT_GET, 'oauth_verifier');
20 if (!$_SESSION['oauth_token'] || !$_SESSION['oauth_token_secret']) {
21     header('Location: login.php');
22     die();
23 }
24
25 try {
26     // Connect to the Twitter API using the token generated for the login
27     // process
28     $applicationApi = new TwitterOAuth(
29         getenv('TWITTER_API_KEY'),
30         getenv('TWITTER_API_SECRET'),
31         $_SESSION['oauth_token'],
32         $_SESSION['oauth_token_secret']
33     );
34
35     // Get the user token (that will allow us to access the user data)
36     $userToken = $applicationApi->oauth('oauth/access_token', [
37         'oauth_verifier' => $oauthVerifier,
38     ]);
39
40     // Connect to the Twitter API using the user token (that allows us to
41     // access their data)
42     $userApi = new TwitterOAuth(
43         getenv('TWITTER_API_KEY'),
44         getenv('TWITTER_API_SECRET'),
45         $userToken['oauth_token'],
46         $userToken['oauth_token_secret']
47     );
48
49     $twitterUserId = $userToken['user_id'];
50     $user = $userApi->get('users/show', [
51         'user_id' => $twitterUserId,
52     ]);

```

```
51
52     Auth::logInAs(
53         User::fromTwitter(
54             $twitterUserId,
55             $user->name,
56             $user->screen_name
57         )
58     );
59
60     header('Location: /');
61 } catch (TwitterOAuthException $e) {
62     header('Location: loginError.php?error=token');
63 }
64
```

```
1 <?php
2 require_once __DIR__ . '/../config/config.inc.php';
3 ?>
4 <!doctype html>
5 <html lang="fr">
6 <head>
7   <meta charset="UTF-8">
8   <meta name="viewport"
9     content="width=device-width, user-scalable=no, initial-scale=1.0,
10    maximum-scale=1.0, minimum-scale=1.0">
10  <meta http-equiv="X-UA-Compatible" content="ie=edge">
11  <title>Interlude</title>
12
13  <link rel="stylesheet" href="/dist/app.css">
14 </head>
15 <body>
16 <div id="app"></div>
17
18 <script src="/dist/app.js"></script>
19 </body>
20 </html>
21
```

```
1 <?php
2 /**
3  * This page redirects to Twitter's login page.
4 */
5 require_once __DIR__.'/../config/config.inc.php';
6
7 use Abraham\TwitterOAuth\{TwitterOAuth, TwitterOAuthException};
8 use Interlude\Auth;
9
10 if (Auth::isLogged()) {
11     header('Location: /');
12     die();
13 }
14
15 try {
16     $twitterOAuth = new TwitterOAuth(getenv('TWITTER_API_KEY'), getenv('
TWITTER_API_SECRET'));
17
18 // Generate a request token for this authentication process
19 $requestToken = $twitterOAuth->oauth('oauth/request_token', [
20     'oauth_callback' => getenv('APP_URL').'/auth.php',
21 ]);
22
23 $_SESSION['oauth_token'] = $requestToken['oauth_token'];
24 $_SESSION['oauth_token_secret'] = $requestToken['oauth_token_secret'];
25
26 $url = $twitterOAuth->url('oauth/authorize', [
27     'oauth_token' => $requestToken['oauth_token'],
28 ]);
29
30 header('Location: '.$url);
31 } catch (TwitterOAuthException $e) {
32     @error_log('['.date('d/m/Y H:i:s').'] Can't connect to the Twitter
server: '.$e->getMessage().PHP_EOL, 3, __DIR__.'/../storage/logs/twitter.
log');
33     header('Location: loginError.php?error=twitter');
34 }
35
```

```
1 <?php
2 /**
3  * Logout page
4 */
5 require_once __DIR__.'../config/config.inc.php';
6
7 // Don't log out unless a POST request was sent
8 // (This isn't as efficient as a proper CSRF protection but it's better
9 // than nothing for now)
9 if ($_SERVER['REQUEST_METHOD'] === 'POST') {
10   \Interlude\Auth::logOut();
11 }
12
13 header('Location: /');
14
```

```
1 <?php
2 require_once __DIR__.'/../config/config.inc.php';
3
4 if (\Interlude\Auth::isLogged()) {
5     header('Location: /');
6     die();
7 }
8
9 $title = null;
10 $description = null;
11
12 $error = filter_input(INPUT_GET, 'error');
13 if ($error === 'token') {
14     $title = 'Jeton d'accès invalide';
15     $description = 'Le jeton d'accès à Twitter fourni n\'est pas ou plus
16 valide. Il se peut qu'il ait déjà été utilisé, ou qu'il ait expiré car il
17 est trop ancien.';
18 } elseif ($error === 'twitter') {
19     $title = 'Connexion à Twitter impossible';
20     $description = 'Impossible d'établir une connexion entre le serveur
21 Interlude et Twitter.';
22 } elseif ($error === 'denied') {
23     $title = 'Accès refusé';
24     $description = 'Vous avez refusé à Interlude l'accès à votre compte
25 Twitter. Cet accès est nécessaire au fonctionnement du programme.';
26 }
27
28 ?>
29 <!doctype html>
30 <html lang="fr">
31 <head>
32     <meta charset="UTF-8">
33     <meta name="viewport"
34         content="width=device-width, user-scalable=no, initial-scale=1.0,
35         maximum-scale=1.0, minimum-scale=1.0">
36     <meta http-equiv="X-UA-Compatible" content="ie=edge">
37     <title>Interlude</title>
38
39     <link rel="stylesheet" href="/dist/app.css">
40 </head>
41 <body class="error-page">
42     <h1><?= $title ?? 'Erreur inconnue' ?></h1>
43     <p><?= $description ?? 'Une erreur inconnue s\'est produite lors de la
44 connexion.' ?></p>
45
46     <footer>
47         <a class="btn btn-secondary" href="/">
48             Annuler
49         </a>
50         <a class="btn btn-primary" href="/login.php">
51             Réessayer
52         </a>
53     </footer>
54 </body>
```

49 </html>

50

```

1 <?php
2 /**
3  * Edits an album and its songs
4 *
5  * Parameters:
6  *   $_POST['id']: The ID of the album to edit.
7  *   $_POST['title']: The new title of the album
8  *   $_POST['artist']: The new artist name of the album
9 *
10 *   $_POST['songs[]'][*]['id'] The ID of the song
11 *   $_POST['songs[]'][*]['title'] The new title of the song
12 *   $_POST['songs[]'][*]['trackNumber'] The new track number of the song
13 *   (Note: any song that is in the database but not in the request
parameters will be deleted.)
14 *
15 * Returns:
16 *   HTTP 200 OK. JSON: { album: <Album>, songs: <Song[]> }
17 *   (The custom types correspond to the jsonSerialize() method of the
homonymous PHP class)
18 *
19 * Errors:
20 *   HTTP 400: Bad request (a parameter is missing)
21 *   HTTP 401: Unauthorized (the user isn't logged in)
22 *   HTTP 404: Not found (the album doesn't exist or doesn't belong to the
user)
23 */
24 use Interlude\{Database, Album, Auth};
25 require_once __DIR__ . '/../../config/config.inc.php';
26
27 if (!Auth::isLogged()) {
28     http_response_code(401); // Unauthorized
29     die('Unauthorized');
30 }
31
32 $id = filter_input(INPUT_POST, 'id', FILTER_VALIDATE_INT);
33 $title = filter_input(INPUT_POST, 'title');
34 $artist = filter_input(INPUT_POST, 'artist');
35
36 // Invalid request (missing parameters)
37 if (!$id || !$title || !$artist) {
38     http_response_code(400);
39     die('Bad request');
40 }
41
42 // Verify that the album exists and is owned by the user
43 if (!Database::find('albums', 'id_album = :id AND id_user = :user', [
44     'id' => $id,
45     'user' => Auth::id(),
46 ], 'id_album')) {
47     http_response_code(404);
48     die('Not found');
49 }
50
51 // Update the album details

```

```

52 $album = new Album($id, $title, $artist);
53 $album->save();
54
55 // Update the songs
56 $existingSongs = $album->songs();
57 /* @var $newSongs array[] */
58 $newSongs = [];
59
60 if (isset($_POST['songs']) && is_array($_POST['songs'])) {
61     $newSongs = array_filter($_POST['songs'], function ($song) {
62         return isset($song['id']) && is_numeric($song['id'])
63             && isset($song['title']) && is_string($song['title'])
64             && isset($song['trackNumber']) && ctype_digit($song['trackNumber'])
65             && strlen($song['trackNumber']) <= 2;
66     });
67 }
68 // For all existing songs, delete the song or update it.
69 foreach ($existingSongs as $existingSong) {
70     $matchingNewSong = null;
71
72     foreach ($newSongs as $song) {
73         if ($existingSong->id == $song['id']) {
74             $matchingNewSong = $song;
75             break;
76         }
77     }
78
79     if (is_null($matchingNewSong)) {
80         $existingSong->delete();
81     } else {
82         $existingSong->title = $matchingNewSong['title'];
83         $existingSong->trackNumber = $matchingNewSong['trackNumber'];
84         $existingSong->save();
85     }
86 }
87
88 header('Content-type: application/json');
89 echo json_encode([
90     'album' => $album,
91     'songs' => $album->songs(),
92 ]);
93

```

```
1 <?php
2 /**
3  * Returns the user that is currently logged in.
4  *
5  * Returns:
6  *   HTTP 200 OK. JSON: { user: <User> },
7  *   (The custom types correspond to the jsonSerialize() method of the
8  *   homonymous PHP class)
9  *
10 * Errors:
11 *   HTTP 200 OK. JSON: { user: null },
12 *   No user is currently logged in.
13 */
14
15 require_once __DIR__ . '/../config/config.inc.php';
16
17 header('Content-type: application/json');
18 echo json_encode([
19     'user' => \Interlude\Auth::user(),
20 ]);
```

```
1 <?php
2 /**
3  * Deletes an album and its songs.
4  *
5  * Parameters:
6  *   $_POST['id']: The ID of the album to delete.
7  *
8  * Returns:
9  *   HTTP 200 OK
10 *
11 * Errors:
12 *   HTTP 400: Bad request (missing parameter id)
13 *   HTTP 401: Unauthorized (the user isn't logged in)
14 *   HTTP 404: Not found (the album doesn't exist or doesn't belong to the
15 * user)
16 */
17 use Interlude\{Database, Album, Auth};
18 require_once __DIR__ . '/../config/config.inc.php';
19
20 if (!Auth::isLogged()) {
21     http_response_code(401); // Unauthorized
22     die('Unauthorized');
23 }
24
25 // Invalid request (missing parameters)
26 $id = filter_input(INPUT_POST, 'id', FILTER_VALIDATE_INT);
27 if (!$id) {
28     http_response_code(400);
29     die('Bad request');
30 }
31
32 // Verify that the album exists and is owned by the user
33 $albumRow = Database::find('albums', 'id_album = :id AND id_user = :user',
34 [
35     'id' => $id,
36     'user' => Auth::id(),
37 ], 'title, artist');
38 if (!$albumRow) {
39     http_response_code(404);
40     die('Not found');
41 }
42
43 $album = new Album($id, $albumRow['title'], $albumRow['artist']);
44 $album->delete();
45
46 http_response_code(200);
```

```

1 <?php
2 /**
3  * Upload some files to the user's music library.
4  *
5  * Parameters:
6  *   $_POST['files[]']: The files to upload
7  *
8  * Returns:
9  *   HTTP 200 OK. JSON: { errors: [], albums: <Album[]>, songs: <Song[]> }
10 *   (The custom types correspond to the jsonSerialize() method of the
homonymous PHP class)
11 *
12 * Errors:
13 *   HTTP 401: Unauthorized (the user isn't logged in)
14 *   HTTP 200 OK. JSON: { errors: String[], albums: null, songs: null }
15 *   An error occurred during the upload. See response.errors for the
error message(s).
16 */
17
18 use Interlude\{Album, Auth};
19 use Interlude\Util\Files;
20 require_once __DIR__ . '/../../config/config.inc.php';
21
22 $errors = [];
23
24 if (!Auth::isLogged()) {
25     http_response_code(401); // Unauthorized
26     die();
27 }
28
29 $uploadedFiles = Files::getUploadedFiles('files');
30 try {
31     $albums = Album::createFromFiles(array_values($uploadedFiles), Auth::id());
32 } catch (\Interlude\Exceptions\UploadException $e) {
33     $errors[] = $e->getMessage();
34     $albums = [];
35 }
36
37 // Load the songs for the response
38 $songs = [];
39 foreach ($albums as $album) {
40     array_push($songs, ...$album->songs());
41 }
42
43 echo json_encode([
44     'errors' => $errors,
45     'albums' => $albums,
46     'songs' => $songs,
47 ]);
48

```

```
1 <?php
2 /**
3  * Returns the contents of the logged user's music library
4  *
5  * Returns:
6  *   HTTP 200 OK. JSON: { albums: <Album[]>, songs: <Song[]> }
7  *   (The custom types correspond to the jsonSerialize() method of the
8  *   homonymous PHP class)
9  *
10 * Errors:
11 *   HTTP 401: Unauthorized (the user isn't logged in)
12 */
13
14 use Interlude\Auth;
15 require_once __DIR__ . '/../../config/config.inc.php';
16
17 if (!Auth::isLogged()) {
18     http_response_code(401); // Unauthorized
19     die();
20 }
21 header('Content-type: application/json');
22 echo json_encode([
23     'albums' => Auth::user()->albums(),
24     'songs' => Auth::user()->songs(),
25 ]);
26
```

```

1 <?php
2 /**
3  * Serves an song's cover image
4  * $_GET['id']: song ID
5 */
6
7 use Interlude\{Database, Song, Auth, Util\Files};
8
9 require_once __DIR__ . '/../config/config.inc.php';
10
11 $id = filter_input(INPUT_GET, 'id', FILTER_VALIDATE_INT);
12
13 $song = null;
14 if ($id && Auth::isLogged()) {
15     $songs = Database::run('SELECT songs.file_path
16         FROM songs
17             LEFT JOIN albums on songs.id_album = albums.id_album
18             WHERE songs.id_song = :song AND albums.id_user = :user
19     ', [
20         'song' => $id,
21         'user' => Auth::id(),
22     ]);
23
24     $song = $songs[0] ?? null;
25 }
26
27 // Song not found, not owned by the logged in user, or with no existing
28 // audio file
29 if (is_null($song) || !file_exists($filePath = Song::
30     AUDIO_FILES_STORAGE_DIRECTORY . '/' . $song['file_path'])) {
31     http_response_code(404);
32     die('Not found');
33 }
34 // Get the stored file's MIME type (that will be used later to send the
35 // right Content-type HTTP header)
36 $mime = Files::getMimeType($filePath);
37 $contents = file_get_contents($filePath);
38 $size = filesize($filePath);
39 $time = date('r', filemtime($filePath));
40 $extension = pathinfo($filePath, PATHINFO_EXTENSION);
41 $fileResource = @fopen($filePath, 'rb');
42 if (!$fileResource) {
43     http_response_code(500);
44     die('Internal server error');
45 }
46 // Allow the browser to read the file partially (to download the file in
47 // multiple requests)
48 $begin = 0;
49 $end = $size - 1;
50 if (isset($_SERVER['HTTP_RANGE'])) {
51     $matches = [];

```

```

51  if (preg_match('/bytes=\h*(\d+)-(\d*)[\D.*]?:?i', $_SERVER['HTTP_RANGE'])
52    , $matches)) {
53    $begin = intval($matches[1]);
54    if (!empty($matches[2])) {
55      $end = intval($matches[2]);
56    }
57
58  http_response_code(206); // Partial Content
59 }
60
61 // Allow the browser to cache this resource
62 header('Pragma: public');
63 header('Cache-Control: max-age=86400');
64 header('Expires: ' . gmdate('D, d M Y H:i:s \G\M\T', time() + 86400));
65
66 // Tell the browser the length of the response
67 header('Content-Length: ' . (($end - $begin) + 1));
68 if (isset($_SERVER['HTTP_RANGE'])) {
69   header("Content-Range: bytes $begin-$end/$size");
70 }
71
72 // Send the resource
73 header('Content-Type: ' . $mime);
74 header('Content-Transfer-Encoding: binary');
75 header('Content-Disposition: inline; filename=interlude-song-' . $id .
76   ' . $extension);
77 header('Last-Modified: ' . $time);
78
79 $cursor = $begin;
80 fseek($fileResource, $begin, SEEK_SET); // Moves the file resource's
pointer
81 const FILE_READ_INCREMENT = 1024 * 16;
82 while (!feof($fileResource) && $cursor <= $end && (connection_status()
83 === CONNECTION_NORMAL)) {
84   echo fread($fileResource, min(FILE_READ_INCREMENT, ($end - $cursor) + 1
));
85   $cursor += FILE_READ_INCREMENT;
86 }
```

```
1 <?php
2 /**
3  * Serves an album's cover image
4  * $_GET['id']: album ID
5 */
6 use Interlude\{Album, Auth, Util\Files};
7 require_once __DIR__ . '/../../../../config/config.inc.php';
8
9 $id = filter_input(INPUT_GET, 'id', FILTER_VALIDATE_INT);
10
11 $album = null;
12 if ($id && Auth::isLogged()) {
13     $album = \Interlude\Database::find('albums', 'id_album = :album AND
14         id_user = :user', [
15             'album' => $id,
16             'user' => Auth::id(),
17         ], 'cover_pic_path');
18 }
19 // Album not found, not owned by the logged in user
20 if (is_null($album)) {
21     http_response_code(404);
22     die('Not found');
23 }
24
25 // The album exists but has no file linked
26 $filePath = Album::COVERS_STORAGE_DIRECTORY . '/' . $album['cover_pic_path'];
27 if (is_null($album['cover_pic_path']) || !file_exists($filePath)) {
28     $filePath = Album::COVERS_STORAGE_DIRECTORY . '/default.png';
29 }
30
31 // Get the stored file's MIME type (that will be used later to send the
32 // right Content-type HTTP header)
33 $mime = Files::getMimeType($filePath);
34 $contents = file_get_contents($filePath);
35
36 // Allow the browser to cache this resource
37 header('Pragma: public');
38 header('Cache-Control: max-age=86400');
39 header('Expires: ' . gdate('D, d M Y H:i:s \G\M\T', time() + 86400));
40
41 // Send the resource
42 header('Content-type: ' . $mime);
43 echo $contents;
```

```
1 <svg role="img" viewBox="0 0 24 24" xmlns="http://www.w3.org/2000/svg"><  
title>Twitter icon</title><path d="M23.954 4.569c-.885.389-1.83.654-2.825.  
775 1.014-.611 1.794-1.574 2.163-2.723-.951.555-2.005.959-3.127 1.184-.896  
-.959-2.173-1.559-3.591-1.559-2.717 0-4.92 2.203-4.92 4.917 0 .39.045.765.  
127 1.124C7.691 8.094 4.066 6.13 1.64 3.161c-.427.722-.666 1.561-.666 2.  
475 0 1.71.87 3.213 2.188 4.096-.807-.026-1.566-.248-2.228-.616v.061c0 2.  
385 1.693 4.374 3.946 4.827-.413.111-.849.171-1.296.171-.314 0-.615-.03-.  
916-.086.631 1.953 2.445 3.377 4.604 3.417-1.68 1.319-3.809 2.105-6.102 2.  
105-.39 0-.779-.023-1.17-.067 2.189 1.394 4.768 2.209 7.557 2.209 9.054 0  
13.999-7.496 13.999-13.986 0-.209 0-.42-.015-.63.961-.689 1.8-1.56 2.46-2.  
548l-.047-.02z" fill="#fff"/></svg>
```

2

```
1 <svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" width="50" height="50" viewBox="0 0 50 50"><defs><clipPath id="b"><rect width="50" height="50"/></clipPath></defs><g id="a" clip-path="url(#b)"><rect width="50" height="50" fill="#674ea7"/><g transform="translate(23.558 -9.865) rotate(45)"><g transform="translate(10.692 5.895) rotate(45)"><path d="M-251.986-242.978l4.243-4.243s2.922,5.275,1.976,6.22a.86.86,0,0,1-.62.188C-248.077-240.812-251.986-242.978-251.986-242.978zm-20.531-20.53a2,2,0,0,1,0-2.829l1.415-1.414a2,2,0,0,1,2.828,0l20.507,20.506L-252.01-243Z" transform="translate(282 257)" fill="#fff"/></g><path d="M9.367,1.126A13.859,13.859,0,0,0,4.645.1,12.86,12.86,0,0,0,.036,1.126" transform="translate(20.335 36.403)" fill="none" stroke="#674ea7" stroke-width="2"/><path d="M7.565,0h0" transform="translate(21.849 10.828)" fill="none" stroke="#674ea7" stroke-width="2"/></g><circle cx="25" cy="25" r="25" fill="none"/></g></svg>
```

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <svg width="18px" height="18px" viewBox="0 0 18 18" version="1.1" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
3     <!-- Generator: Sketch 43.2 (39069) - http://www.bohemiancoding.com/sketch -->
4     <title>icon-next</title>
5     <desc>Created with Sketch.</desc>
6     <defs></defs>
7     <g id="Page-1" stroke="none" stroke-width="1" fill="none" fill-rule="evenodd">
8         <g id="icon-next" fill-rule="nonzero" fill="#333333">
9             <g id="Group" transform="translate(0.000000, 2.000000)">
10                <path d="M1.61561562,1.03220617 C0.72333575,0.388483167 0,0.771196358 0,1.85931546 L0,11.9909198 C0,13.0914426 0.719435095,13.4645661 1.61561562,12.818029 L8.1683057,8.0906812 C9.06058557,7.4469582 9.06448622,6.40609108 8.1683057,5.75955401 L1.61561562,1.03220617 Z" id="Path_1"></path>
11                <path d="M10.6156156,1.03220617 C9.72333575,0.388483167 9,0.771196358 9,1.85931546 L9,11.9909198 C9,13.0914426 9.71943509,13.4645661 10.6156156,12.818029 L17.1683057,8.0906812 C18.0605856,7.4469582 18.0644862,6.40609108 17.1683057,5.75955401 L10.6156156,1.03220617 Z" id="Path_1-Copy"></path>
12            </g>
13        </g>
14    </g>
15 </svg>
```

```
1 <svg width="24px" height="24px" viewBox="0 0 24 24" version="1.1" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
2   <title>Jouer</title>
3   <g stroke="none" stroke-width="1" fill="none" fill-rule="evenodd">
4     <g fill-rule="nonzero" fill="#333333">
5       <path d="M5.63003147,1.15061045 C4.72978995,0.515145844 4,0.897026226 4,2.00494659 L4,21.9950534 C4,23.1023548 4.73046875,23.484375 5.63003147,22.8493896 L19.3699685,13.1506104 C20.2702101,12.5151458 20.2695312,11.484375 19.3699685,10.8493896 L5.63003147,1.15061045 Z"></path>
6     </g>
7   </g>
8 </svg>
9
```

```
1 <svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" width="15" height="15" viewBox="0 0 15 15"><defs><clipPath id="b"><rect width="15" height="15"/></clipPath></defs><g id="a" clip-path="url(#b)"><g transform="translate(17 17)"><circle cx="7.5" cy="7.5" r="7.5" transform="translate(-17 -17)" fill="#ff3b30"/><g transform="translate(0 -0.628)"><g transform="translate(-16.372 -8.597) rotate(-45)"><rect width="2.181" height="9.582" transform="translate(3.701)" fill="#fff"/><rect width="2.181" height="9.582" transform="translate(9.582 3.701) rotate(90)" fill="#fff"/></g></g></g></g></svg>
```

```
1 <svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" width="24" height="24" viewBox="0 0 24 24">
2   <defs>
3     <clipPath id="clip-icon-pause">
4       <rect width="24" height="24"/>
5     </clipPath>
6   </defs>
7   <g id="icon-pause" clip-path="url(#clip-icon-pause)">
8     <rect width="24" height="24" fill="#fff"/>
9     <rect id="Rectangle_3" data-name="Rectangle 3" width="10" height="24" rx="2" fill="#333"/>
10    <rect id="Rectangle_4" data-name="Rectangle 4" width="10" height="24" rx="2" transform="translate(14)" fill="#333"/>
11  </g>
12 </svg>
13
```

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <svg width="60px" height="60px" viewBox="0 0 60 60" version="1.1" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" style="background: #FD3B30;">
3     <!-- Generator: Sketch 43.2 (39069) - http://www.bohemiancoding.com/sketch -->
4     <title>icon-trash</title>
5     <desc>Created with Sketch.</desc>
6     <defs></defs>
7     <g id="Page-1" stroke="none" stroke-width="1" fill="none" fill-rule="evenodd">
8         <g id="icon-trash">
9             <g id="Group-2" transform="translate(17.000000, 13.000000)">
10                <path d="M22.0803336,32.996753 C22.0014488,34.1031158 21.047206,35 19.9336123,35 L6.47263774,35 C5.3659211,35 4.3931276,34.1161604 4.29831853,33.0080759 L2.03125,6.51162791 L23.96875,6.51162791 L22.0803336 ,32.996753 Z" id="Path-2" fill="#FFFFFF"></path>
11                <g id="Group" transform="translate(6.500000, 8.953488)" fill="#FD3B30">
12                    <rect id="Rectangle" x="5.6875" y="0" width="1.625" height="21.1627907"></rect>
13                    <polygon id="Rectangle-Copy" points="0 0 1.625 0 2.23093774 15.7825644 2.4375 21.1627907 0.8125 21.1627907"></polygon>
14                    <polygon id="Rectangle-Copy-2" transform="translate(11.781250, 10.581395) scale(-1, 1) translate(-11.781250, -10.581395)" points="10.5625 0 12.1875 0 12.7934377 15.7825644 13 21.1627907 11.375 21.1627907"></polygon>
15                </g>
16                <path d="M0,3.6627907 C0,2.98848955 0.543046951,2.44186047 1.21916106,2.44186047 L24.7808389,2.44186047 C25.454163,2.44186047 26,2.98378845 26,3.6627907 L26,4.88372093 L0,4.88372093 L0,3.6627907 Z" id="Rectangle-2" fill="#FFFFFF"></path>
17                <path d="M7.41697849,1.24125416 C7.80800797,0.555728418 8.76472473,0 9.54273371,0 L16.4572663,0 C17.240259,0 18.1883163,0.549284469 18.5830215,1.24125416 L19.5,2.84883721 L6.5,2.84883721 L7.41697849,1.24125416 Z" id="Rectangle-2-Copy" fill="#FFFFFF"></path>
18            </g>
19        </g>
20    </g>
21 </svg>
```

```
1 <svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" width="30" height="30" viewBox="0 0 30 30"><defs><clipPath id="b"><rect width="30" height="30"/></clipPath></defs><g id="a" clip-path="url(#b)"><rect width="30" height="30" fill="#ff3b30"/><rect width="19" height="3" rx="1.5" transform="translate(6 14)" fill="#fff"/></g></svg>
```

```
1 <svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" width="15" height="15" viewBox="0 0 15 15">
2   <defs>
3     <clipPath id="clip-icon-search">
4       <rect width="15" height="15"/>
5     </clipPath>
6   </defs>
7   <g id="icon-search" clip-path="url(#clip-icon-search)">
8     <g id="Group_2" data-name="Group 2" transform="translate(1)">
9       <g id="Group_1" data-name="Group 1" transform="translate(-1 0)">
10      <g id="Ellipse_1" data-name="Ellipse 1" transform="translate(0)" fill="none" stroke="#999" stroke-width="2">
11        <circle cx="5.622" cy="5.622" r="5.622" stroke="none"/>
12        <circle cx="5.622" cy="5.622" r="4.622" fill="none"/>
13      </g>
14    </g>
15    <line id="Line_1" data-name="Line 1" x1="5.091" y1="5.091" transform="translate(8.164 9.164)" fill="none" stroke="#999" stroke-width="2"/>
16  </g>
17 </g>
18 </svg>
19
```

```
1 <svg id="Layer_1" data-name="Layer 1" xmlns="http://www.w3.org/2000/svg"
viewBox="0 0 40 40"><defs><style>.cls-1,.cls-2{fill:#6751a5;}.cls-2{
opacity:0.32;}</style></defs><title>icon-upload</title><g id="Cloud"><path
class="cls-1" d="M20.34,19.51a2,2,0,0,0-1.41.59l-.06.06-6,6A2,2,0,1,0,15.
67,29l2.66-2.66v8.15a2,2,0,0,0,2,2,2,2,0,0,0,2-2V26.34L25,29a2,2,0,0,0,2.
83,0,2,2,0,0,0,0-2.83L21.75,20.1A2,2,0,0,0,20.34,19.51Z"/><path class="cls
-2" d="M29,16.55a9.37,9.37,0,0,0-17.5-2.5A7.49,7.49,0,0,0,10.25,28.7V26.
05a5,5,0,0,1,2.08-9.55h.89a6.88,6.88,0,0,1,13.49,1.88V19h1.87a3.75,3.75,0,
0,1,1.67,7.1v2.67A6.23,6.23,0,0,0,29,16.55Z"/></g></svg>
```

```
1 <svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" width="15" height="15" viewBox="0 0 15 15"><defs><clipPath id="b"><rect width="15" height="15"/></clipPath></defs><g id="a" clip-path="url(#b)"><g transform="translate(17 17)"><circle cx="7.5" cy="7.5" r="7.5" transform="translate(-17 -17)" fill="#4cd964"/><g transform="translate(0 -1.628)"><g transform="translate(-15.808 -7.445) rotate(-45)"><rect width="2.181" height="4.582" transform="translate(0.679 1.464)" fill="#fff"/><path d="M0,0H2.181V9.582H0Z" transform="translate(10.261 4.115) rotate(90)" fill="#fff"/></g></g></g></g></svg>
```

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <svg width="18px" height="18px" viewBox="0 0 18 18" version="1.1" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
3     <!-- Generator: Sketch 43.2 (39069) - http://www.bohemiancoding.com/sketch -->
4     <title>icon-previous</title>
5     <desc>Created with Sketch.</desc>
6     <defs></defs>
7     <g id="Page-1" stroke="none" stroke-width="1" fill="none" fill-rule="evenodd">
8         <g id="icon-previous" fill-rule="nonzero" fill="#333333">
9             <g id="Group" transform="translate(9.000000, 9.000000) rotate(180.000000) translate(-9.000000, -9.000000) translate(0.000000, 2.000000)">
10            <path d="M1.57195622,0.483412572 C0.703788774,-0.199347763 0,0.141054652 0,1.24321982 L0,12.6400169 C0,13.7424072 0.703635493,14.082705 1.57195622,13.3998241 L8.21196509,8.17786529 C9.08013254,7.49510496 9.08028582,6.38825228 8.21196509,5.7053714 L1.57195622,0.483412572 Z" id="Path_1"></path>
11            <path d="M10.5719562,0.483412572 C9.70378877,-0.199347763 9,0.141054652 9,1.24321982 L9,12.6400169 C9,13.7424072 9.70363549,14.082705 10.5719562,13.3998241 L17.2119651,8.17786529 C18.0801325,7.49510496 18.0802858,6.38825228 17.2119651,5.7053714 L10.5719562,0.483412572 Z" id="Path_1-Copy"></path>
12        </g>
13    </g>
14 </g>
15 </svg>
```

```
1 <svg id="Layer_1" data-name="Layer 1" xmlns="http://www.w3.org/2000/svg"
viewBox="0 0 40 40"><defs><style>.cls-1{opacity:0.35;}.cls-2{fill:#6751a5
;}</style></defs><title>icon-play-next</title><g class="cls-1"><rect class=
"cls-2" x="23" y="11" width="8.5" height="4" rx="2" ry="2"/><rect class=
"cls-2" x="18" y="18" width="13.5" height="4" rx="2" ry="2"/><rect class=
"cls-2" x="8.5" y="25" width="23" height="4" rx="2" ry="2"/></g><path class=
"cls-2" d="M21,13a2,2,0,0,1-.59,1.41l-.06.06-6,6a2,2,0,1,1-2.83-2.83L14.
15,15H6a2,2,0,0,1-2-2,2,2,0,0,1,2-2h8.17L11.5,8.33a2,2,0,0,1,0-2.83,2,2,0,
0,1,2.83,0l6.08,6.08A2,2,0,0,1,21,13Z"/><path class="cls-2" d="M20.41,11.
57h0l-.06-.06Z"/></svg>
```

```
1 <svg id="Layer_1" data-name="Layer 1" xmlns="http://www.w3.org/2000/svg"
viewBox="0 0 40 40"><defs><style>.cls-1{opacity:0.35;}.cls-2{fill:#6751a5
;}</style></defs><title>icon-play-later</title><g class="cls-1"><rect
class="cls-2" x="23" y="25" width="8.5" height="4" rx="2" ry="2"/><rect
class="cls-2" x="18" y="18" width="13.5" height="4" rx="2" ry="2"/><rect
class="cls-2" x="8.5" y="11" width="23" height="4" rx="2" ry="2"/></g><
path class="cls-2" d="M21,27a2,2,0,0,0-.59-1.41l-.06-.06-6-6a2,2,0,1,0-2.
83,2.83L14.15,25H6a2,2,0,0,0-2,2,2,0,0,0,2,2h8.17L11.5,31.67a2,2,0,0,0,
0,2.83,2,2,0,0,0,2.83,0l6.08-6.08A2,2,0,0,0,21,27Z"/><path class="cls-2" d=
"M20.41,28.43h0l-.06.06Z"/></svg>
```

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <svg width="40px" height="40px" viewBox="0 0 40 40" version="1.1" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
3     <!-- Generator: Sketch 43.2 (39069) - http://www.bohemiancoding.com/sketch -->
4     <title>icon-play-round</title>
5     <desc>Created with Sketch.</desc>
6     <defs></defs>
7     <g id="Page-1" stroke="none" stroke-width="1" fill="none" fill-rule="evenodd">
8         <g id="icon-play-round">
9             <circle id="Oval" fill="#FAFAFA" opacity="0.95" cx="20" cy="20" r="20"></circle>
10            <path d="M16.6307149,10.3074955 C15.7300959,9.67176442 15,10.0533683 15,11.1609035 L15,28.6641349 C15,29.7711902 15.7349523,30.1498459 16.6307149,29.5175429 L28.6071169,21.0636121 C29.5077359,20.427881 29.5028795,19.3937293 28.6071169,18.7614263 L16.6307149,10.3074955 Z" id="Path_1" fill="#674EA7" fill-rule="nonzero"></path>
11        </g>
12    </g>
13 </svg>
```

```
1 <svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" width="150" height="150" viewBox="0 0 150 150">
2   <defs>
3     <clipPath id="clip-no-search-results">
4       <rect width="150" height="150"/>
5     </clipPath>
6   </defs>
7   <g id="no-search-results" clip-path="url(#clip-no-search-results)">
8     <g id="Group_2" data-name="Group 2" transform="translate(12.284 11.284)">
9       <g id="Group_1" data-name="Group 1" transform="translate(-6.112 -5.112)">
10         <g id="Ellipse_1" data-name="Ellipse 1" transform="translate(0)" fill="none" stroke="#b5b4b4" stroke-width="15">
11           <circle cx="52.811" cy="52.811" r="52.811" stroke="none"/>
12           <circle cx="52.811" cy="52.811" r="45.311" fill="none"/>
13         </g>
14       </g>
15       <line id="Line_1" data-name="Line 1" x1="45.512" y1="45.512" transform="translate(80.921 81.921)" fill="none" stroke="#b5b4b4" stroke-width="15"/>
16     </g>
17     <g id="Group_4" data-name="Group 4" transform="translate(-16 6)">
18       <circle id="Ellipse_2" data-name="Ellipse 2" cx="25" cy="25" r="25" transform="translate(22 89)" fill="#ff3b30"/>
19       <g id="Group_3" data-name="Group 3" transform="translate(-61.156 46.856) rotate(-45)">
20         <rect id="Rectangle_1" data-name="Rectangle 1" width="5.689" height="25" transform="translate(26.155 111)" fill="#fff"/>
21         <rect id="Rectangle_2" data-name="Rectangle 2" width="5.689" height="25" transform="translate(41.5 120.655) rotate(90)" fill="#fff"/>
22       </g>
23     </g>
24   </g>
25 </svg>
26
```

```
1 # Ignore all files in the folder, but keep it in the repository structure.  
2 *  
3 !.gitignore  
4
```

```
1 # Ignore all files in the folder, but keep it in the repository structure.  
2 *  
3 !.gitignore  
4
```

```
1 # Ignore all files in the folder, but keep it in the repository structure.  
2 *  
3 !.gitignore  
4 !default.png  
5
```

```

1 -- phpMyAdmin SQL Dump
2 -- version 4.7.0
3 -- https://www.phpmyadmin.net/
4 --
5 -- Hôte : 127.0.0.1
6 -- Généré le : Dim 03 fév. 2019 à 21:05
7 -- Version du serveur : 5.7.17
8 -- Version de PHP : 5.6.30
9
10 SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
11 SET AUTOCOMMIT = 0;
12 START TRANSACTION;
13 SET time_zone = "+00:00";
14
15
16 /*!40101 SET @OLD_CHARACTER_SET_CLIENT=@CHARACTER_SET_CLIENT */;
17 /*!40101 SET @OLD_CHARACTER_SET_RESULTS=@CHARACTER_SET_RESULTS */;
18 /*!40101 SET @OLD_COLLATION_CONNECTION=@COLLATION_CONNECTION */;
19 /*!40101 SET NAMES utf8mb4 */;

20
21 --
22 -- Base de données : `interlude`
23 --
24
25 -----
26
27 --
28 -- Structure de la table `albums`
29 --
30
31 CREATE TABLE `albums` (
32   `id_album` int(11) NOT NULL,
33   `title` VARCHAR(60) NOT NULL,
34   `artist` VARCHAR(60) NOT NULL,
35   `cover_pic_path` VARCHAR(250),
36   `id_user` int(11) NOT NULL
37 ) ENGINE=InnoDB DEFAULT CHARSET=utf8 ROW_FORMAT=COMPACT;
38
39 -----
40
41 --
42 -- Structure de la table `songs`
43 --
44
45 CREATE TABLE `songs` (
46   `id_song` int(11) NOT NULL,
47   `title` VARCHAR(60) NOT NULL,
48   `track_number` int(11) NOT NULL,
49   `file_path` VARCHAR(250) NOT NULL,
50   `id_album` int(11) NOT NULL,
51   `upload_date` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP()
52 ) ENGINE=InnoDB DEFAULT CHARSET=utf8 ROW_FORMAT=COMPACT;
53
54 -----

```

```

55
56  --
57  -- Structure de la table `users`
58  --
59
60  CREATE TABLE `users` (
61      `id_user` int(11) NOT NULL,
62      `twitter_user_id` bigint(20) NOT NULL,
63      `full_name` varchar(100) CHARACTER SET latin1 NOT NULL,
64      `username` varchar(50) CHARACTER SET latin1 NOT NULL
65  ) ENGINE=InnoDB DEFAULT CHARSET=utf8 ROW_FORMAT=COMPACT;
66
67  --
68  -- Index pour les tables déchargées
69  --
70
71  --
72  -- Index pour la table `albums`
73  --
74  ALTER TABLE `albums`
75      ADD PRIMARY KEY (`id_album`),
76      ADD KEY `id_user` (`id_user`);
77
78  --
79  -- Index pour la table `songs`
80  --
81  ALTER TABLE `songs`
82      ADD PRIMARY KEY (`id_song`),
83      ADD KEY `id_album` (`id_album`);
84
85  --
86  -- Index pour la table `users`
87  --
88  ALTER TABLE `users`
89      ADD PRIMARY KEY (`id_user`);
90
91  --
92  -- AUTO_INCREMENT pour les tables déchargées
93  --
94
95  --
96  -- AUTO_INCREMENT pour la table `albums`
97  --
98  ALTER TABLE `albums`
99      MODIFY `id_album` int(11) NOT NULL AUTO_INCREMENT;
100 --
101 -- AUTO_INCREMENT pour la table `songs`
102 --
103 ALTER TABLE `songs`
104     MODIFY `id_song` int(11) NOT NULL AUTO_INCREMENT;
105 --
106 -- AUTO_INCREMENT pour la table `users`
107 --
108 ALTER TABLE `users`

```

```
109  MODIFY `id_user` int(11) NOT NULL AUTO_INCREMENT;
110  --
111  -- Contraintes pour les tables déchargées
112  --
113  --
114  --
115  -- Contraintes pour la table `albums`
116  --
117 ALTER TABLE `albums`
118   ADD CONSTRAINT `albums_ibfk_1` FOREIGN KEY (`id_user`) REFERENCES
119   users(`id_user`);
120  --
121  -- Contraintes pour la table `songs`
122  --
123 ALTER TABLE `songs`
124   ADD CONSTRAINT `songs_ibfk_1` FOREIGN KEY (`id_album`) REFERENCES
125   albums(`id_album`);
126 COMMIT;
127 /*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
128 /*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
129 /*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
130
```