LOG8430 - Architecture Logicielle et Conception Avancée
TP2: Mise en Œuvre d'une Architecture Logicielle qui fait appel à des services de streaming de musique

Fabio Petrillo et Manel Abdellatif
Département de génie informatique et génie logiciel
École Polytechnique de Montréal, Québec, Canada
fabio@petrillo.com, manel.abdellatif@polymtl.ca

Session	Automne 2017
Pondération	30% de la note finale du TP
Taille des équipes	3-4 étudiants
Date importante (23h55 au plus tard)	06 Novembre 2017 : Remise initiale non notée de votre projet 12 Novembre 2017 : Remise finale de votre projet
Directives particulières	Soumission des travaux par moodle uniquement.
En cas de problème lors de la soumission, veuillez contacter la chargée de laboratoire ou le responsable du cours.	

1 Introduction

L'objectif de cet exercice est de mettre en pratique les connaissances que vous avez acquises lors des cours précédents. Il s'agit de choisir, de justifier et d'implanter la "meilleure" architecture pour les besoins ci-dessous et d'étudier la qualité du logiciel.

1.1 Déroulement du travail pratique

Pour le 06 Novembre 2017: Vous pouvez remettre une première version de votre travail pratique respectant le format des livrables (voir Section 4) et ayant une solution pour les étapes P1 à P6.

Vous recevrez ensuite une rétroaction avec les commentaires sur votre travail. Vous devez utiliser ces commentaires pour éventuellement améliorer votre travail avant la remise finale le 12 Novembre 2017.

2 Requis

Un client vous demande de créer un programme qui intègre au moins **trois** systèmes de streaming de musique, en utilisant les APIs disponibles. Les systèmes sont:

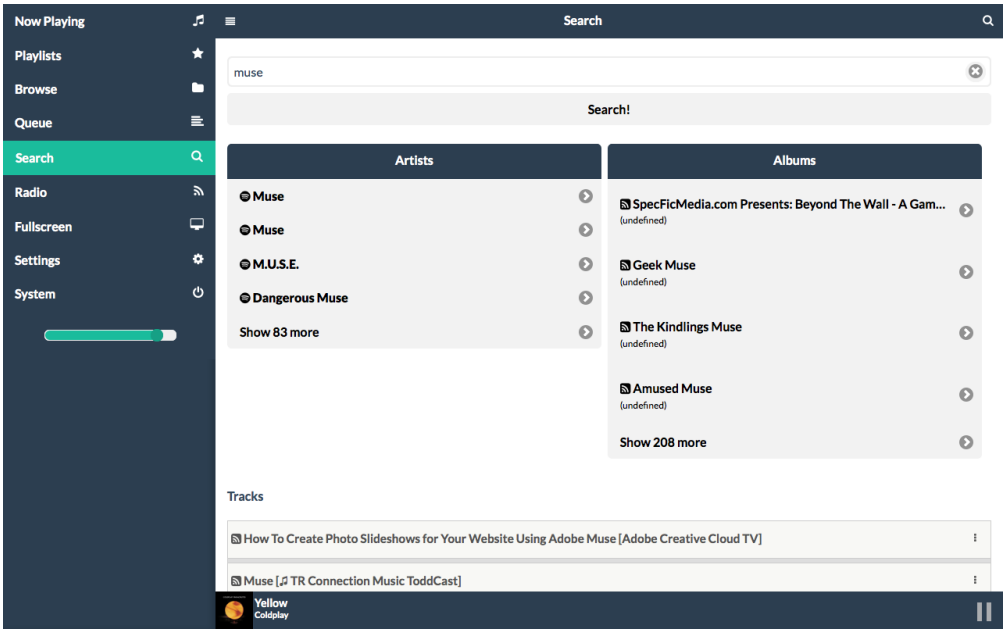

Figure 1: Exemple d'outil de recherche

- Spotify - `https://developer.spotify.com/web-api`
- SoundClound - `https://developers.soundcloud.com/docs/api/guide`
- 8tracks - `https://8tracks.com/developers`
- Deezer - `https://developers.deezer.com`
- Beatport - `https://oauth-api.beatport.com/`
- Google Play Music - `https://github.com/jkiddo/gmusic.api`

Vouz pouvez aussi trouver des informations sur les APIs en `http://www.programmableweb.com`. Le client voudrait une interface usager simple. Un exemple d'application semblable peut être trouvé en `http://www.pimusicbox.com/` (figures 1 et 2).

R1 Avoir un outil de recherche de musiques indépendant de système de streaming.

R2 Être capable de gérer les listes de reproduction.

R3 Reproduire les musiques sélectionnées ou de la liste de reproduction (playlist).

3 Travail à réaliser

Veuillez faire chaque étape du travail dans l'ordre et fournir votre démarche pour chacune d'elles dans votre **guide du développeur**.

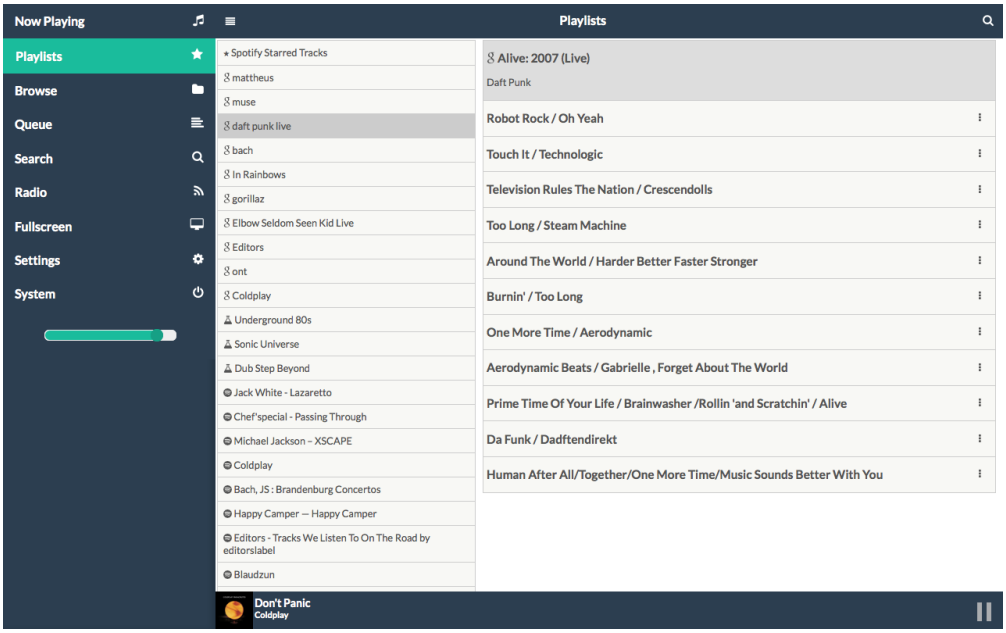

Figure 2: Exemple de liste de reproduction (Playlist)

P1 Choisir/définir une architecture répondant au besoin du client. Décrire le style architectural choisi et modéliser le système selon le modèle 4+1, en utilisant UML ou Archimate.

P2 Décrire le(s) patron(s) de conception utilisés.

P3 Justifier vos choix architecturaux et technologiques

P4 Réaliser cette architecture en implémentant des structures pour garantir une bas couplage et une haute cohésion

P5 Suivre et décrire les bonnes pratiques du "clean code" vues en classe

P6 Implanter des classes de test unitaire JUnit qui permet de tester arriver dans une coverture de 50% du code ou plus.

P7 Étudier la qualité de code de votre architecture comme vous l'avez appris dans le TP1 et critiquer votre solution.

4 Livrables

Chaque groupe doit fournir:

L1 Projet "zippé" avec comme nom du fichier Zip, les noms de famille par ordre alphabétique des étudiants du groupe et " - TP 2".

L2 Un code de qualité, respectant au moins les conventions de code (et de l'école) et réutilisant le plus possible de code d'une partie à l'autre.

L3 Commentaires des classes et méthodes ("Javadoc" pour projet en Java).

L4 Un guide d'utilisateur décrivant comment utiliser votre projet et votre programme.

L5 Des cas de test exécutables pour chaque fonctionnalité importante du projet.

L6 Un guide du développeur contenant le style architectural de votre programme (selon le modèle 4+1), la description du/des patron(s) de conception utilisés, les bonnes pratiques du "clean code" utilisées et une étude critique des choix technologiques, architecturaux et de la qualité de code de votre logiciel.

L7 Utiliser un service web d'hébergement et de gestion de logiciels comme Github ou Bitbucket pendant le développement.