**ECOLE POLYTECHNIQUE MONTREAL**

**LOG8430**

**Travail Pratique 1 : Option2**

**Intégration de services de streaming de musique**

**EQUIPE : Nebil Soumri #1561533**

**Ousmane Dieng #1588059**

**2016-02-03**

**Introduction:**

Ce document décrit l’architecture d’un système intégrant trois systèmes de streaming de musique, en utilisant les APIs disponibles. Les services utilisées ici sont : SoundCloud, Spotify et Google music. La structure du système est présentée à l’aide d’un diagramme de composant et un diagramme de classe, alors que le comportement et les intégrations sont décrits avec un diagramme de séquence.

1. **Définition de l’architecture**

L’architecture choisie dans ce travail et qui répond aux besoins du client est l’architecture la plus simple, celle de 3 niveaux dont vous trouvez le niveau le plus supérieur, celui de présentation qui reflète les fonctionnalités des services fournis à l’utilisateur. Les résultats de ces fonctionnalités comme le jeu de musique et la recherche sont fournis par les algorithmes encapsulés dans le niveau du milieu qui responsable de la logique de l’application. Ce niveau change de comportement en fonction du service dont on accède, et qui a sa propre méthode d’exploit. Pour ce faire on a utilisé le parton stratégie qui va être expliqué dans la section suivante.

Ce dernier niveau communique avec le plus pas à travers les API fournis et qui sont l’interface de communication. Le niveau le plus bas aussi contient une partie des données locales pour faciliter l’accès et garder les données des flux de streaming.

1. **Descriptions des patrons de conceptions utilisés :**

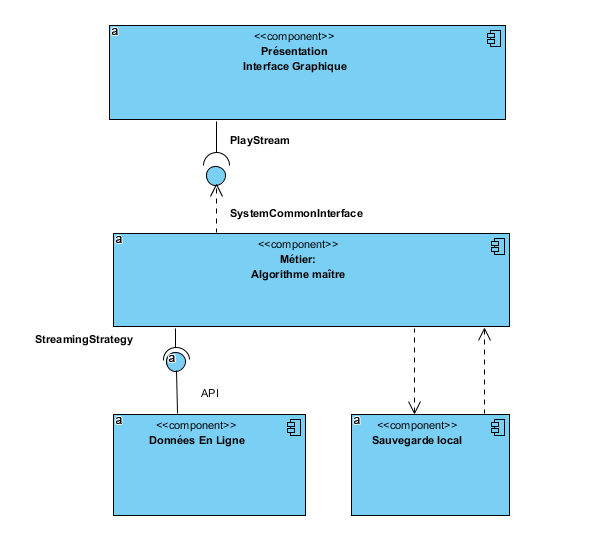
Les patrons de conception utilisés dans cette application sont :

**Patron Stratégie**

Le patron « strategy » intervient dans cette application comme une solution pour l’algorithme maître qui figure dans le deuxième niveau de l’architecture et qui est responsable de la logique de l’application. Cet algorithme doit changer de comportement en fonction des API fournis chaque fois qu’il fait appel à un service de streaming. Donc le patron « strategy » fournit une interface pour l’algorithme ou la méthode utilisée, à partir de laquelle hérite une classe concrète qui encapsule une implémentation de l’algorithme avec le comportement approprié et compatible avec les API du service voulue. De cette façon, Chaque fois qu’on veut intégrer un nouveau service, on crée une classe concrète qui hérite de l’interface du patron « strategy » et qui encapsule le nouveau comportement de u nouveau service.

1. **Vue Structurelle du système:**

**Diagramme de composants**



**Description :**

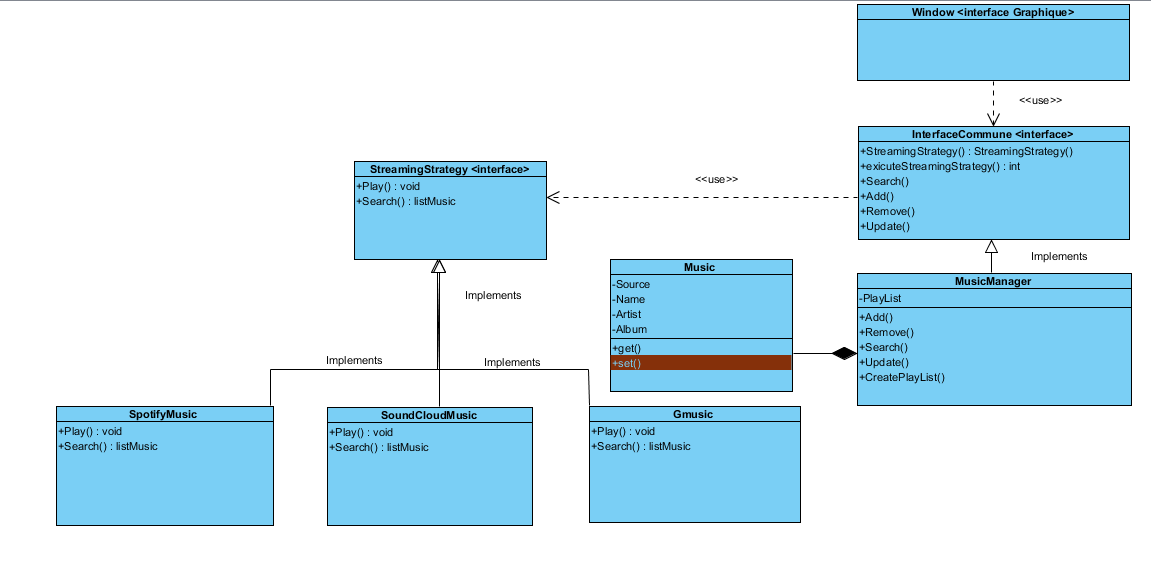
Le diagramme de composants illustre bien l’architecture 3 niveaux choisi. Vous remarquez bien la couche supérieure qui fait sortir les services au client. Cette couche et avec la classe principale « PlayStream » exploite les résultats fournis par la couche logique nommée sur le dessin algorithme maître à travers les méthodes et les fonctions de l’interface « SystemCommonInterface ». Les résultats proviennent de trois services de streaming différents. Chacune a ses propres API et ces méthodes d’accès. Le patron strategy de cette couche permet de résoudre le changement entre l’interface avec les services de la plus basse couche qui sauvegarde la musique en ligne.

**Diagramme de classe**

**Description :**

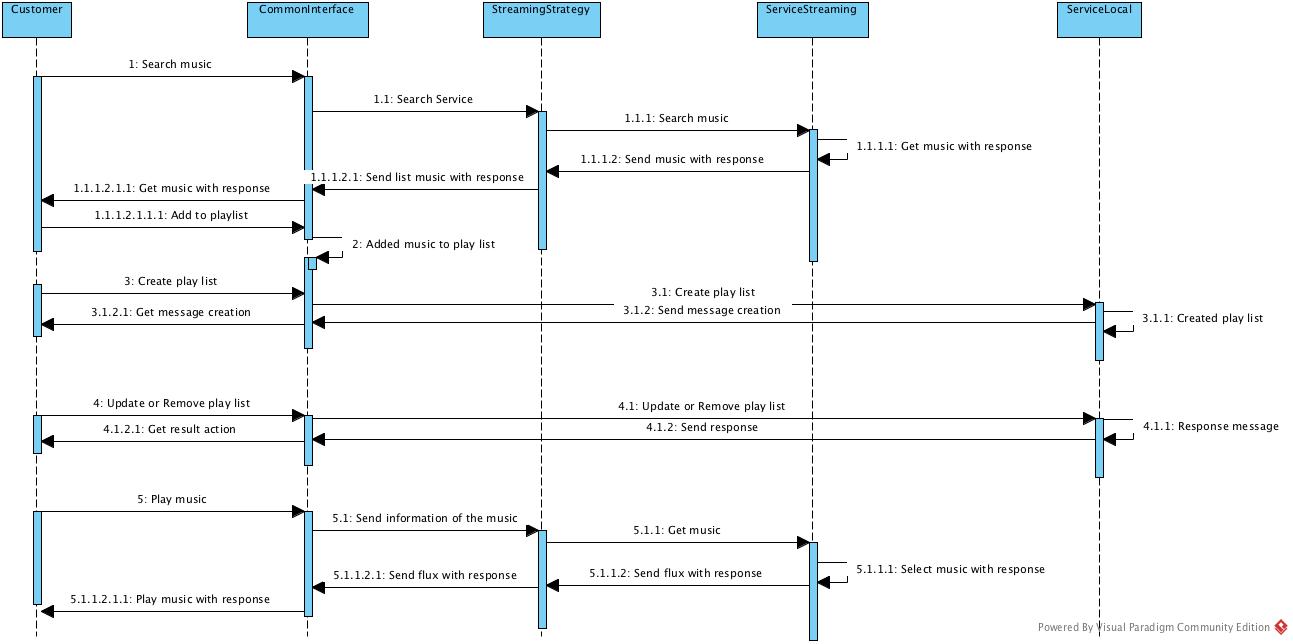
Le diagramme de classe de cette application est simple : Une classe interface « SystemCommonInterface » utilisée par l’interface graphique. Elle contient les méthodes qui fournissent toutes les fonctionnalités au client. C’est une interface pour tout le système. La classe « Music manager » composée de la classe « Music » permet de gérer les musiques localement à l’aide d’un fichier gSon. La classe «Streaming Strategy » Constitue une interface du patron strategy qui permet de communiquer avec les données en ligne et acheminer le flux streaming. Une classe concrète de chaque streaming implémente l’interface a l’aide de ses propres API.

**Diagramme de classe**

****

1. **Vue comportement**

**Diagramme de séquence :**

****

**Description :**

Le diagramme de séquence montre les interactions entre le client et le système.

Pour faire une recherche d’une chanson alors le client utilise l’interface commune dans laquelle il tape un mot à rechercher. Cette interface à son tour va interroger notre interface « strategy » pour que celle-ci sélectionne les services de musique. Ces derniers vont fournir une réponse négative si la recherche ne donne pas de résultats ou une réponse positive dans le cas contraire.

Pour créer une liste de musique le client doit créer d’abord le nom de celle-ci s’il n’existe pas. Ensuite dans cette interface il peut rechercher une chanson et l’ajouter dans sa liste de musique, et ainsi de suite.

Pour jouer une chanson le client doit choisir une chanson et cliquer sur jouer la chanson. Il peut faire son choix selon ses listes de musiques ou faire une recherche et choisir.

Le client peut faire la gestion de ses listes de musiques en les supprimant ou en ajoutant de nouvelle chanson.

1. **Outils utilisés :**

Service web d'hébergement et de gestion de logiciels utilisés est **GitHub.**

<https://github.com/fallouseven/musicStreaming>

Pour les diagrammes UML, on a utilisé **Visual Paradigm**.

**Conclusion**

Nous avons choisie l’option 2 du laboratoire qui a nécessité la mise en place d’un prototype pour savoir comment ça va fonctionner et enlever les nuages qui entourent le sujet avec les API des services de streaming qui ont besoin d’une exploration particulière, et par la suite faire les décisions architecturale. Cette étape a pris plus de temps que prévu. L’avancement dans ce laboratoire était lent, vu les difficultés rencontrés, ce qui nous a empêché d’implanter toutes les fonctionnalités.