AVIGNON UNIVERSITÉ

CENTRE D'ENSEIGNEMENT ET DE RECHERCHE EN INFORMATIQUE

Master Informatique

ILSEN

UE Architectures distribuées UCE Application architectures distribuees

>>>

Rapport Application d'architecture distribuée

Monday 20th May, 2019

Groupe M1-ILSEN ALT

Delfeil CASANOVA

CERI - LIA
339 chemin des Meinajariès
BP 1228
84911 AVIGNON Cedex 9
France

Tél. +33 (0)4 90 84 35 00
Fax +33 (0)4 90 84 35 01

http://ceri.univ-avignon.fr

Contents

1 Travail à réaliser

Lors de ce projet il est demandé de développer une application mobile permettant de créer un lecteur de musique commandé par la voix. Il sera nécessaire d'implémenter 4 applications distinctes communiquant entre-elles.

- une application mobile centrale servant de base pour le lecteur de musiques,
- une application de retranscription de la parole, qui transforme des commandes vocales en texte,
- un analyseur d'entités nommées, qui doit permettre d'interpréter le texte généré par l'application de retranscription et en extraire des actions à réaliser.
- Une quatrième application permet de gérer une base de données de musiques et de faire du streaming de ces musique à la demande des utilisateurs.

Chacune de ces applications doivent être implémentées sur des serveurs et accessibles depuis l'application mobile.

2 Travail réalisé

2.1 Application mobile Android

J'ai développé l'application mobile en Java sous Android, celle-ci joue un rôle central pour jouer de la musique, en faisant interagir tous les modules entres eux. Je peux enregistrer une commande vocale à envoyer au serveur de transcription, qui sera stockée temporairement dans la mémoire du téléphone, lors de l'appui sur un bouton.

J'ai mis en place un système d'options, qui permet à l'utilisateur de faire afficher un formulaire avec un champ texte pour lui permettre de saisir la commande qu'il souhaite réaliser, sans passer par le système de retranscription de la parole. De plus, ces paramètres me permettent de saisir l'adresse ip des différents serveurs accueillant les différents modules.

J'ai personnalisé l'affichage des éléments d'une liste , pour faire apparaître les informations concernant une liste de musiques et d'albums, ainsi que l'image correspondant à l'album.

2.2 Transcripteur de la parole

J'ai développé sur un serveur nodeJs, en utilisant le framework ExpresssJs, l'application de retranscription des commandes audio vers du texte. Je fais appel à l'api "Speech-to-Text" de Google, qui prend en entrée un fichier audio au format AMR et fournit en retour de requête, le texte correspondant à la transcription de la commande audio passée en paramètre.

Pour faire communiquer l'application Android avec ce serveur, j'utilise Volley, avec lequel je fais une requête http, de type Post à l'url suivante, avec en paramètre le fichier audio à retranscrire.

http://@ip:3101/transcribe

Une fois la transcription effectuée, le serveur renvoie la transcription au format Json, comme suit:

```
{
    "transcription": "texte"
}
```

2.3 Analyseur d'entités nommées

J'ai développé en java un serveur Rest, déployé sur un serveur Glassfish pour gérer l'analyseur d'entités nommées, permettant d'analyser les commandes demandées par l'utilisateur. En passant un texte correspondant à une commande en entrée, cette application identifie l'action à réaliser en conséquence, ainsi que certains paramètres en relation avec cette commande. Plusieurs commandes sont reconnues:

- Jouer une musique
- Jouer les musiques d'un artiste
- Jouer les musiques d'un album
- Afficher une musique
- Afficher les musiques disponibles
- Afficher les musiques d'un album
- Afficher les musiques d'un artiste
- Afficher un album
- Afficher les albums disponibles
- Afficher les albums d'un artiste
- Musique suivante
- Musique précédente
- Augmenter le son
- Diminuer le son
- Couper le son
- Remettre le son
- Arrêter la musique
- Mettre la musique en pause
- Reprendre la musique
- Recommencer la musique

Cette reconnaissance est effectuée à l'aide d'expressions régulières.

Pour faire communiquer le serveur Rest avec l'application Android, j'utilise la librairie Volley qui permet de gérer le client Rest, en faisant une requête au serveur, et en passant en paramètre le texte à analyser.

> http://@ip:8080/correspondance/rest/action?text=Joue la musique le petit bonhome en mousse

Une fois l'analyse effectuée par le serveur, les informations sur l'action à réaliser ainsi que les paramètres correspondants, sont retournés au format Json suivant:

```
{
    "action": "play_musique",
    "param": {
        "musique": "le petit bonhomme en mousse"
    }
}
```

Ce format peut-être modulé selon les différentes actions demandées au travers de la requête. L'attribut "action" peux prendre plusieurs valeurs, en fonction de la commande analysée: (play_musique, affiche_musique, next_musique,...).
De plus les éléments dans "param" diffèrent si l'on veut des informations concernant un album, une musique ou un artiste

```
"param": {
    "musique": "get lucky"
}
"param": {
    "album": "random access memories"
}
"param": {
    "artiste": "daft punk"
}
```

Pour améliorer cet analyseur d'entités nommées, j'ai utilisé des EJB ainsi que des Entity Bean, proposées par Java Entreprise Edition, pour communiquer avec une base de données PostgreSql, contenant des correspondances entre un texte et une catégorie (musique, artiste et album). Ainsi, il est possible de faire des demandes comme "Joue le petit bonhomme en mousse", le système reconnaîtra "le petit bonhomme en mousse" comme étant une musique. Il est aussi possible d'ajouter de nouvelles correspondances dans la base de données, en faisant une requête sur l'url suivante, avec en paramètre la catégorie et le texte. Cet ajout est géré par l'application Android, qui fait appel au serveur Rest, dès lors que des informations concernant un artiste, un album ou une musique ont pu être récupérées du serveur Ice.

http://@ip:8080/correspondance/rest/validCorrespondance?type=artiste&text=daft punk

2.4 Serveur de streaming de musiques

Un serveur Ice développé en python me permet de gérer le serveur de musiques, en interagissant avec une base de données MongoDb, dans laquelle les informations sur les musiques sont au format Json:

```
{
    "titre": "on melancholy hill",
    "artiste": "gorillaz",
    "album": "plastic beach",
    "path_musique":"~/gorillaz/plastic_beach/on_melancholy_hill.mp3",
    "path_image":"~/gorillaz/plastic_beach/cover.jpg"
}
```

L'application Andoid joue le rôle du client Ice et fait appel aux fonctions proposées par le serveur Ice. Celui-ci permet de jouer une musique, les musiques d'un artiste et les musiques d'un album passés en paramètre, il peut aussi retourner les informations concernant une musique, les musiques d'un artiste et d'un album, ainsi que les informations concernant un album et des albums d'un artiste. Il prends aussi en compte le fait d'arrêter la lecture d'une musique, de mettre en pause et de reprendre la lecture, ainsi que de passer à la musique suivante et revenir à la précédente.

Le serveur Ice se charge de récupérer la liste des musiques correspondantes dans la base de données et lance un stream sur une url et un port précis du serveur, en utilisant la librairie vlc. Il retourne ainsi au client l'url d'accès au stream (si celui-ci a demandé de jouer de la musique), ainsi que la liste des informations sur les musiques, au format Json suivant.

```
{
  "url": "http://192.168.1.6:8083/stream.mp3",
  "info": [{
    "titre": "clint eastwood",
    "artiste": "gorillaz",
    "album": "gorillaz",
    "cover": "Image encodee"
  }, {
    "titre": "on melancholy hill",
    "artiste": "gorillaz",
    "album": "plastic beach",
    "cover": "Image encodee"
  }]
}
```

Si une musique ne possède pas d'image liée, une image par défaut lui est attribuée.
Pour pouvoir passer l'image au client Ice, je l'encode en base64. L'application Android se charge de la reconvertir en BitMap pour pouvoir l'afficher dans la vue correspondant aux information des musiques et des albums.

2.5 Représentation

Une représentation globale des différents modules présentés précédemment, ainsi que leurs différentes interactions, est présente en annexe.1

3 Améliorations possibles

Pour améliorer cette application, il serait intéressant d'améliorer le serveur de lecture de musique, pour permettre à plusieurs utilisateurs de lancer un stream de musique en même temps, et d'offrir à ces utilisateurs la possibilité de gérer leur propre base de données de musiques, en ajoutant et en enlevant des musiques à celle-ci.
Une autre amélioration, serait de développer une plateforme de gestion back office, pour pouvoir analyser les différentes requêtes demandées par les utilisateurs, ainsi que la base de données des correspondances.
Au niveau de l'application Android, il serait possible d'avoir plus d'informations sur la musique en train d'être jouée, en communiquant avec le serveur Ice, pour notamment afficher un timer.
Au niveau de l'analyseur d'entités nommées, il serait intéressant de permettre la demande de requêtes plus précises, tel que la réduction du son de x%, avancer une musique de x minutes, ou encore demander de jouer une playlist de musiques qui n'ont aucun lien entre elles: "joue les musiques get lucky et le petit bonhomme en mousse et on melancholy hill" ou "ajoute à la lecture la musique le petit bonhomme en mousse".

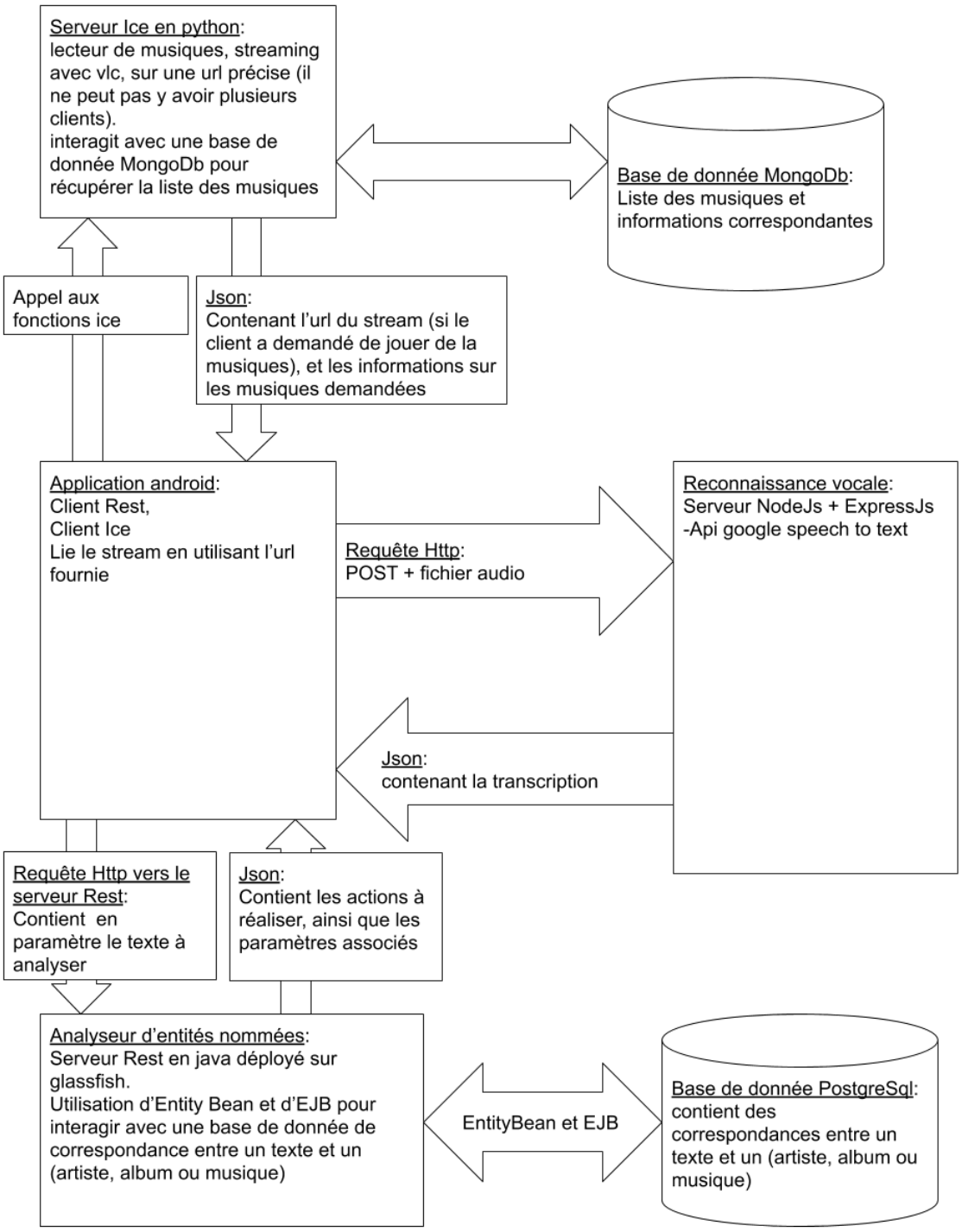

Figure 1. Représentation schématique des modules et de leurs interactions