Responsable : M. MORCHID Mohamed
Étudiant : HABIB Mohamed

Rapport final

Application architectures distribuées

Année universitaire 2018/2019

OBJECTIFS

L'application à developper est une application distribuée qui utilise différentes technologies (Middleware, service Web, JMS).

L'application doit permettre de piloter par la parole un lecteur de flux audio-vidéo développé sous Android.

L'application devra donner accès à toutes les fonctionnalités du lecteur par des message vocaux formulés en langage naturel (sans contrainte sur la forme des messages). Par exemple, l'application devrait pouvoir lancer la lecture du morceau ''Hotel California'' à partir d'une requête du type ''je veux écouter Hotel California''.

Pour cela nous aurons besoins de trois composants pour pouvoir ,capter le signal du message ,le transcrire puis l'analyser et invoquer le serveur pour obtenir le morceau souhaité.

Les trois composant sont:

1) Composant pour l'acquisition du signal et la transcription automatique du message.
2) un analyseur de requêtesilisé .

3) un serveur de flux audio-vidéo en streaming.

1) Composant pour l'acquisition du signal

Langage utilisé : Java (Android).

Technologie utilisé : API de Google Speech.

2) un analyseur de requêtes

Langage utilisé : Java.

Technologie utilisé : JMS.

3) un serveur de flux audio-vidéo en streaming

Langage utilisé : NodeJS.

Technologie utilisé : mediaserver.

Environnement de travail : Windows 10.

4) un serveur de gestion des musique

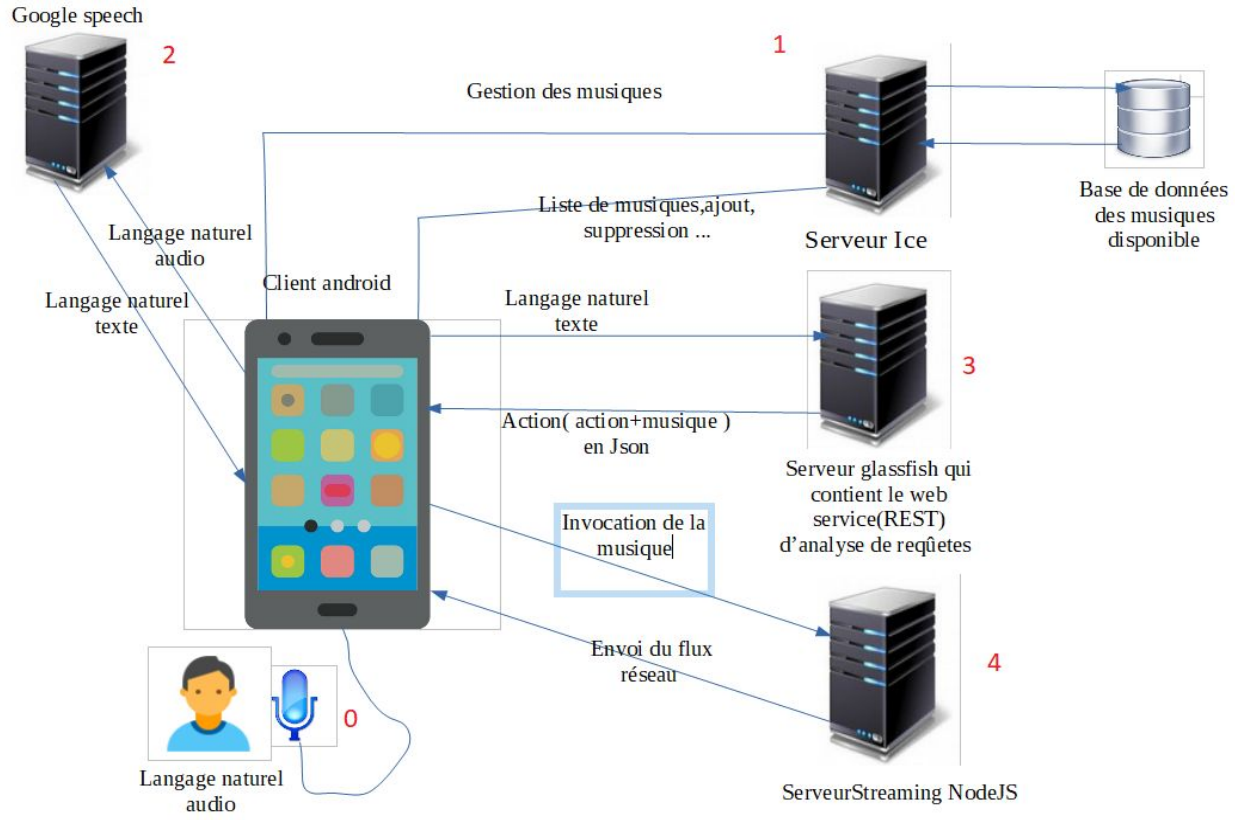
Langage utilisé : Java.

Technologie utilisé : Middleware (ICE).

Environnement de travail : Linux ubuntu 16.04.

DESCRIPTION TECHNIQUE DES COMPOSANTS :

Voici un Schéma illustrant l'architecture applicative du Système :



FONCTIONNEMENT :

1) L’acquisition du signal et la transcription automatique du message en texte :

   Cette partie est intégrée au client, consiste à capter le signal recu par le terminal(Android mobile) et ensuite le numeriser,à partir de ce signal numerique on fait la transcription du message que l’on va ensuite envoyer à notre analyseur de requete.

   Pour cela nous j’ai choisi une technologie parmi plusieurs avec des contraintes telles que:

   -L’API doit supporter la reconnaissance vocale du français.

   -Elle devra traiter correctement le bruit de fond et les accents de prononciation

   Toutes ces contraintes réunies nous ont permis de choisir «Android Speech »

   car c’est une API facile d’utilisation et offre plusieurs des fonctionnalités gratuites contrairement à Google speech qui est payant.

2) L’analyseur de requêtes :

   L’analyseur de requêtes est un service web Rest qui est dans un serveur glassfish.

   Son rôle est trouver dans un texte brut une action (action/musique,volume...).

   C’est a dire quand la requête arrive au niveau de notre analyseur sous format

   texte apres la transcription du message formulé en langage naturel par Android

   speech, Notre analyseur recherche dans le texte un couple action/objet.

   Les actions sont :
   {"augmente","baisse","joue","jouer","pause","reprendre","continue","stopper","stoppé","supprimer","supprime","écouter","arrête","lance"

   ,elles sont stockeés dans un variable au niveau de notre Web service.

   Toute requête dont il n'y a aucune action parmi celles citées en dessus est

   considérée comme invalide autant pour une requête dont l’objet nul.

Une fois le Web service fini son traitement il renvoi une un objet Action sous forme JSON qui a deux deux a attribut action et musique. Par exemple si la requête est « Je veux Jouer Hotel California» ,notre analyseur renvoi la ligne

suivante :

Réponse = {'action': jouer, 'musique' : Hotel California}.

3) Le serveur de flux audio en streaming :

Pour ce composant j'ai utilisé NodeJS avec le module mediaserver car il offre des fonctionnalités de haut niveau,en plus facile à mettre en oeuvre pour le streaming.

Le serveur reçoit un requête GET avec le nom de la musique et commence la transmission de la musique demandé.

4) Serveur Ice :

Le serveur Ice sert a communiquer avec la base de données.Au lancement de l'application il va à la base de données et récupère toute les musiques disponible avec leurs Path

et les envoi au client qui les affiche en ListView, et il sert pour la recherche,l'ajout et la suppression ...

EXEMPLE DE FONCTIONNEMENT :

1) 'je veux écouter Hotel California'
2) 'je veux écouter Hotel California
3) [écouter, Hotel California].
4) Si la chanson existe sur le serveur il envoi le flux audio streaming.
5) La musique se lance.

DÉPLOIEMENT :

Analyseur :

- lancer glassfish.

asadmin start-domain

- Ouvrir dans le dossier ou se trouve le fichier Analyseur.war et lancer la commande

asadmin deploy --force Analyseur.war

Serveur Ice :

Makefile

all: Slice Server

Slice:

slice2java Printer.ice

mv mp3 src

Server:

javac -cp "libs/*" -d classes/ src/*.java src/mp3/*.java

java -cp :Path\libs\ice-3.7.2.jar:.:Path\libs\mysql-connector-java-8.0.15.jar Server

Serveur NodeJS :

- Se mettre au niveau du dossier et lancer la commande :

  node index

CONCLUSION :

Ce projet fut tres benefique car il m’a permit d'approfondir mes connaissance sur le

middleware ICE et les architectures distribuées en generale.

L’application peut être améliorée en améliorant l’interface et en ajoutant plus de fonctionnalités et en y intégrant de serveurs auxiliaires pour pallier les pannes dans serveur quelconque ce qui donnera une haute disponibilité del’architecture.