AVIGNON UNIVERSITÉ

Rapport de projetApplication et Architecture Distribuée

Groupe : 15

BOUAZOUNI Mohamed Walid
MERAKEB Ramdane

28 mai 2021

M1 Informatique
ILSEN
UE Application et Architecture Distribuée

Enseignant
M. Mickael Rouvier

UFR
SCIENCES
TECHNOLOGIES
SANTÉ

AVIGNON UNIVERSITÉ
CERI

CENTRE
D'**ENSEIGNEMENT**
ET DE **RECHERCHE**
EN **INFORMATIQUE**
ceri.univ-avignon.fr

Sommaire

Liste des figures

1 Introduction et contexte du projet

Durant ce projet, il nous a été demandé de concevoir et développer une application distribuée sous **Android** permettant de piloter par la parole un lecteur de flux audio-vidéo. L'architecture de l'application devait intégré plusieurs composants décentralisés/distribués au sein de cette dernière, pour cela plusieurs technologies ont été utilisées, certaines de ces technologies nous ont été confiées et leurs utilisations été obligatoires !

Durant ce rapport nous allons détailler tout le processus de réalisation de cette application, c'est-à-dire un diagramme de cas d'utilisation présentant les différentes fonctionnalités que nous avons su intégrés, les différents composants et technologies utilisées dans l'application, et enfin l'architecture de cette dernière.

2 Présentation de l'application et ses fonctionnalités

Durant cette sections nous allons fournir des détails concernant les fonctionnalités que nous avons développé au seins de l'application.

Lors de la présentation du projet, nous avons commencé en premier lieu par identifier les différentes fonctionnalités qu'on devait intégrer, nous avons définies toutes ces fonctionnalités dans un diagramme d'utilisation que nous présentons ci-dessous (voir Figure 1).

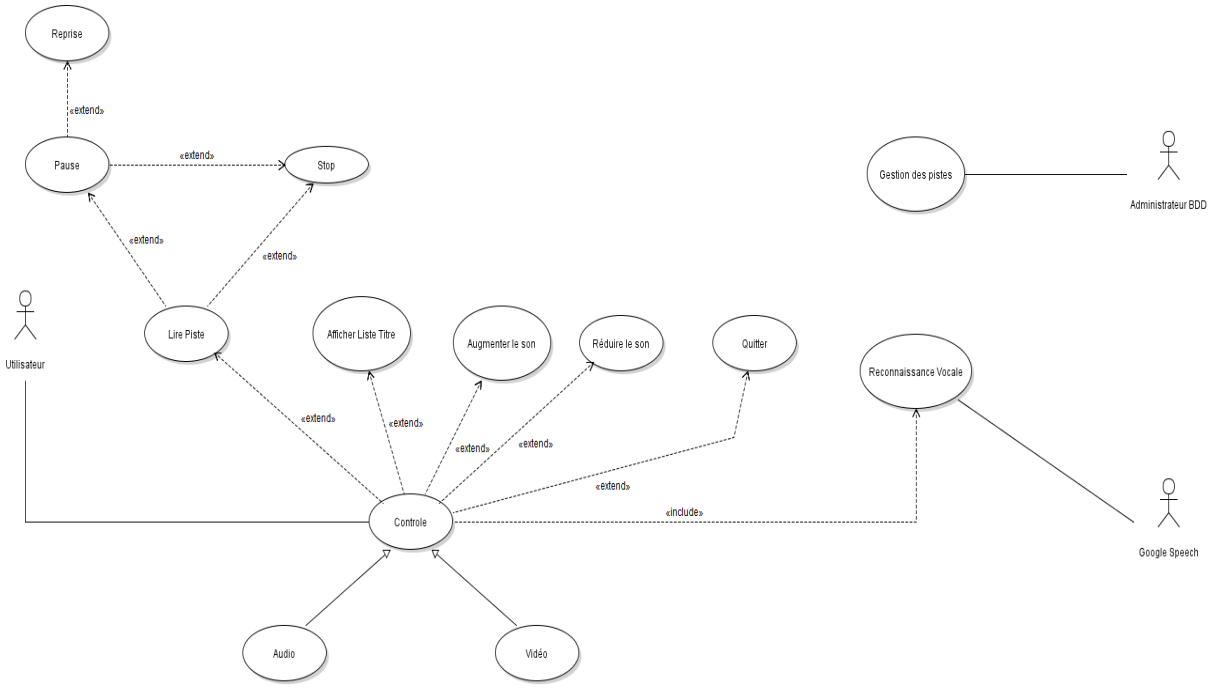

Figure 1. Diagramme de cas d'utilisation

2.1 Fonctionnalités

Après avoir ouvert l'application, l'utilisateur interagit avec l'application via un synthétiseur vocal permettant le contrôle des pistes de musiques et des vidéos.

Nous avons 8 types de fonctionnalités liés à ce contrôle, il faut savoir que d'autres expressions sont prisent en charge du a la mauvaise interprétation du transcripteur audio utilisé (Accents, conjugaisons, etc)! :

2.1.1 Afficher liste musique/vidéo

Permet l'affichage des titres de musiques ou de vidéos dans l'interface via les commandes vocal suivantes :

- **Actions :** Afficher, Voir.
- **Exemple d'expressions :** Afficher liste musiques, Voir liste vidéo, Afficher vidéos, etc.

2.1.2 Play

Permet de jouer une piste (musique ou vidéo) en précisant son titre via les commandes vocal suivantes :

- **Actions :** Lire, Play, Écouter, Jouer.
- **Exemple d'expressions :** Lire *titre musique*, Lire *titre vidéo*, Écouter *titre musique*, etc.

2.1.3 Pause

Permet de mettre en pause une musique ou vidéo déjà en lecture via les commandes vocal suivantes :

- **Actions :** Pause.
- **Exemple d'expressions :** Pause, Pause *titre musique*, Pause *titre vidéo*.

2.1.4 Reprendre

Permet la reprise de lecture d'une piste en pause via les commandes vocal suivantes :

- **Actions :** Reprendre, Reprise.
- **Exemple d'expressions :** Reprendre, Reprise, Reprendre *titre musique*, Reprendre *titre vidéo*, etc.

2.1.5 Stop

Permet l'arrêt complet d'une lecture de piste qu'elle soit en pause ou en cours via les commandes vocal suivantes :

- **Actions :** Stop, Arrêter.
- **Exemple d'expressions :** Stop, Arrêter, Arrêter *titre musique*, Stop *titre vidéo*, etc.

2.1.6 Augmenter le son

Permet d'augmenter le son d'une piste via les commandes vocal suivantes :

- **Actions :** Augmenter.
- **Exemple d'expressions :** Augmenter, Augmenter le son, etc.

2.1.7 Réduire le son

Permet de réduire le son d'une piste via les commandes vocal suivantes :

- **Actions :** Réduire, Diminuer.
- **Exemple d'expressions :** Réduire, Diminuer, Réduire le son, etc.

2.1.8 Quitter l'application

Permet de quitter l'application après confirmation via les commandes vocal suivantes :

- **Actions :** Quitter.
- **Exemple d'expressions :** Quitter, Quitter l'application, etc.

3 Architecture Technique

Durant cette section, nous exposerons l'architecture technique de notre application, en fournissant des détails concernant les technologies utilisées dans chaque composants, ainsi que les liaisons entre ces derniers.

La figure suivante (Voir Figure 2) détaille l'architecture technique de l'application ainsi que les étapes de la circulation de l'information.

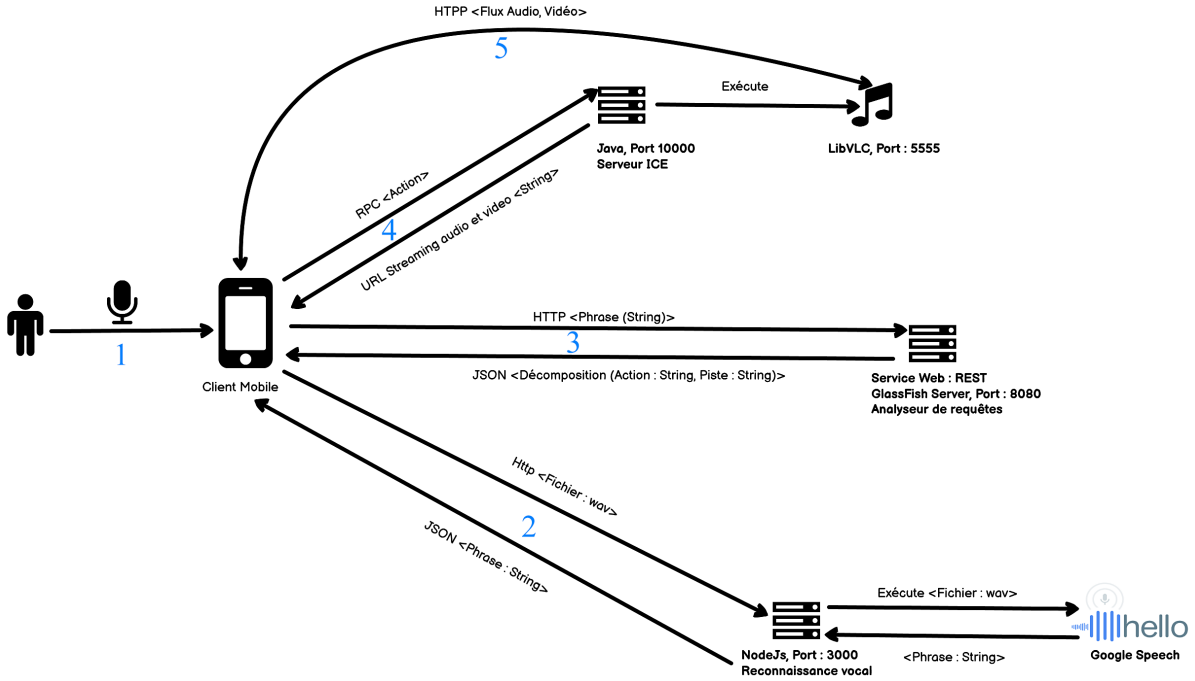

Figure 2. Architecture Technique de l'application

3.1 Technologies utilisées

3.1.1 Implémentation client/serveur : Java

Java présente l'avantage d'être un langage de programmation qui a fait ses preuves. Il possède des librairies implémentées par défaut qui permettent d'effectuer des communications en réseaux. Les sockets en Java sont basés sur le TCP. Cela va permettre de facilement assurer que le message audio envoyé par le smartphone arrive dans son intégralité au serveur. On peut aussi utiliser le protocole TCP pour s'assurer que les données streamées à l'utilisateur, arrivent dans leur intégralité. On ne veut pas lui retirer des bouts du morceau qu'il écoute.

Vis-à-vis de l'application mobile sur Android, Java est approprié : il est pleinement supporté et Android Studio offre une suite d'outils permettant de développer facilement pour cette plateforme.

Java dispose aussi de librairies dédiées à l'hébergement de serveurs qui permettent de

mettre en place des API.

Le fait d'employer le même langage côté serveur et client permet de réduire le champ d'expertise nécessaire des développeurs.

3.1.2 Traitement Automatique du Langage : Regex

Les expressions régulières représentent un standard adopté par la plupart des langages modernes en ce qui concerne la reconnaissance de motifs dans un texte. L'utilisation de Regex trouve parfaitement son sens lorsqu'il est nécessaire de reconnaître dans un discours des patterns. Rechercher des patterns est la solution la plus évidente et simple pour reconnaître les demandes d'un utilisateur.

Par contre les expressions régulières présentent quelques défauts en raison de l'incapacité d'adaptation de manière intelligente car ils ne font qu'obéir à des règles strictes, mais au vu du contexte demandé l'utilisation des ces dernières permettant de réaliser pleinement les objectifs du projet demandés.

3.1.3 Automatic Speech Recognition : Google Speech-To-Text

Permet la reconnaissance vocale pour transcrire des mots et des termes propres en temps réel. Parmi ces gros avantages nous citons : un algorithme d'intelligence artificiel poussé grâce à un réseau de neurones développé par Google pour la reconnaissance automatique de la parole prenant en compte plusieurs millions de paramètres, prise en compte de plus de 125 langues et enfin la facilité d'utilisation.

Nous pouvons citer quelques désavantages, notamment son prix auquel nous avons $1h$ d'essai gratuit puis $0{,}006/15$ secondes et l'obligation de fournir un moyen de paiement pour l'essai gratuit.

3.1.4 Module d'application streaming

libVLC est le moteur principal et l'interface du cadre multimédia sur lequel le lecteur multimédia VLC est basé. libVLC est modulaire en centaines de plugins, qui peuvent être chargés au moment de l'exécution. Cette architecture offre une grande flexibilité aux développeurs (à la fois les développeurs VLC et les développeurs utilisant la bibliothèque). Il permet aux développeurs de créer une large gamme d'applications multimédias en utilisant les fonctionnalités de VLC. elle offre plusieurs avantages :

- Permet de lire tous les formats de fichiers multimédias, tous les codecs et tous les protocoles de streaming
- Matériel et décodage efficace sur toutes les plates-formes, jusqu'à 8K.
- Prise en charge des filtres vidéo et audio.
- Exécutez sur toutes les plates-formes, du bureau (Windows, Linux, Mac) au mobile (Android, iOS) et aux téléviseurs.
- Etc.

3.2 Détails de l'architecture technique

3.2.1 Acquisition du signal

Cette partie est intégrée au client, consiste à capter le signal reçu par le terminal (mobile) via le microphone pour ensuite le numériser en format **.wav** et l'envoyer au serveur de transcription.

3.2.2 Transcription automatique du message

Nous avons mis en place le service de transcription automatique des messages dans un serveur intermédiaire NodeJs, et ce pour plusieurs avantages :

- Faciliter le déploiement et d'apprentissage.
- Faciliter le changement de technologie de reconnaissance vocale sans impacter le client.

La réponse envoyée par le serveur est la phrase transcrit par Google speech sous format JSON.

3.2.3 Analyse des requêtes

Une fois le client Android reçoit la transcription de l'enregistrement, il transmet une requête via le protocole HTTP au serveur GlassFish implémentant le service web REST qui s'occupera d'analyser ce texte et en sortir une commande reconnue par le client Android en utilisant les expressions régulières vu dans les sections précédentes.

Le serveur GlassFish renvoie les commandes sous format JSON, comportant les actions à effectuer et les pistes a joué, etc.

3.2.4 Serveur de Streaming

Une fois que le client reçoit les commandes à exécuter via l'analyseur de requêtes, l'application exécutera la méthode à distance via le middleware ICE qui aura pour rôle de déclencher LibVLC et renvoyer l'URL de streaming au quel le client se placera en écoute.

4 Conclusion

Le projet fut une expérience très enrichissante sur le plan professionnel en termes de compétences acquises, malgré toutes les difficultés que présente le projet dû au manque de documentation de certaines technologies utilisées (Exemple : ICE), mais nous avons su les résoudre au fil du temps grâce à volonté et patience accru pour vous présenter ce projet.