ANDROID APP.

Cahier des charges

Application Architectures Distribuees

DELVAUX JULIEN

INDEX

Introduction

Ce projet a pour but de développer une application distribuée android permettant de lire des musiques mp3 en streaming depuis un smartphone.
Il s'agira de mettre en oeuvre les différentes technologies vues dans l'UCE pour implémenter les modules nécessaires pour faire marcher l'application.

Description des modules

Acquisition du signal (Client-Side)

Contexte:
Ce module doit être capable de capter le signal émis à travers une entrée comme un microphone, puis le numériser pour pouvoir le transmettre au module de transcription automatique du message.
Une fois le message capté sous forme de signal analogique, il faut le convertir en signal numérique tel qu'une séquence FLAC pour pouvoir l'envoyer au module de transcription.

Solutions possibles:
Plusieurs façons d'enregistrer la voix sont disponibles, parmis les plus utilisées :

- "Media Recorder" : https://developer.android.com/reference/android/media/MediaRecorder.html

- "AudioRecord" : https://www.newventuresoftware.com/blog/record-play-and-visualize-raw-audio-data-in-android
- https://github.com/mmig/speech-to-flac

Solution choisie:
On utilisera un module de l'API Android "AudioRecord" pour réaliser ce procédé car c'est un module de l'API android, très bien documenté et beaucoup utilisé. De plus il permet de sortir un fichier raw (FLAC) qui peut être interprété par un appareil android: https://developer.android.com/guide/topics/media/media-formats.html

Transcription automatique du message (Server-Side)

Contexte:
Ce module a pour but de récupérer le signal numérique créé par le module d'acquisition et de le traduire en langage naturel.

Solutions possibles:
Encore une fois, plusieurs choix s'offrent à nous.

- La librairie du LIA pour réaliser cette opération.

Le module doit prendre un tableau de réels en entrée pour pouvoir sortir une chaîne de caractères correspondant à la phrase dite par l'utilisateur.

- L'API de google "speech" de reconnaissance vocale.
- L'API android "SpeechRecognizer".

Solution choisie:
Nous préférons utiliser L'API de google, très utilisée et facile à mettre en place. Le format des données à envoyer étant en JSON :

```
{
  "config": {
      "encoding":"FLAC",
      "sampleRateHertz": 16000,
      "languageCode": "en-US",
      "enableWordTimeOffsets": false
  },
  "audio": {
      "uri":"gs://cloud-samples-tests/speech/brooklyn.flac"
  }
}
```

Tout comme le format de retour. De plus, il nous sera fourni, si souhaité, plusieurs estimations de ce qui a été dit pour pouvoir sélectionner la meilleure reconnaissance possible avec un facteur de "confidence" allant de 0 a 1 :

```
"hypotheses": [
      {
      "utterance": "bonjour et bienvenue",
      "confidence": 0.7474387
      }
      ]
}
```

Nous utiliserons aussi cURL pour pouvoir envoyer des ces requêtes et ainsi poster et récupérer les données.

Analyseur de requêtes (Server-Side/Web Service)

Contexte:
Ce module permet d'analyser les requêtes retournées par le module de transcription et ainsi exécuter les actions requises en commande vocale par l'utilisateur.

Différentes actions seront disponibles en premier lieu :

- Jouer : permet de lancer un morceau en lecture.
- Stop : permet d'arrêter instantanément la lecture du morceau.
- Morceau/Musique/Titre suivant : permet d'accéder à la lecture du morceau suivant de la playlist
- Morceau/Musique/Titre précédent : permet d'accéder à la lecture du morceau précédent de la playlist

On aura donc des modèles type <Jouer, California Love>.

Il sera aussi implémenté d'autres fonctionnalitées comme la possibilité de "pause" et "unpause" la lecture en cours, ou de reconnaître plusieurs façons d'arrêter ou de commencer la lecture d'un morceau.

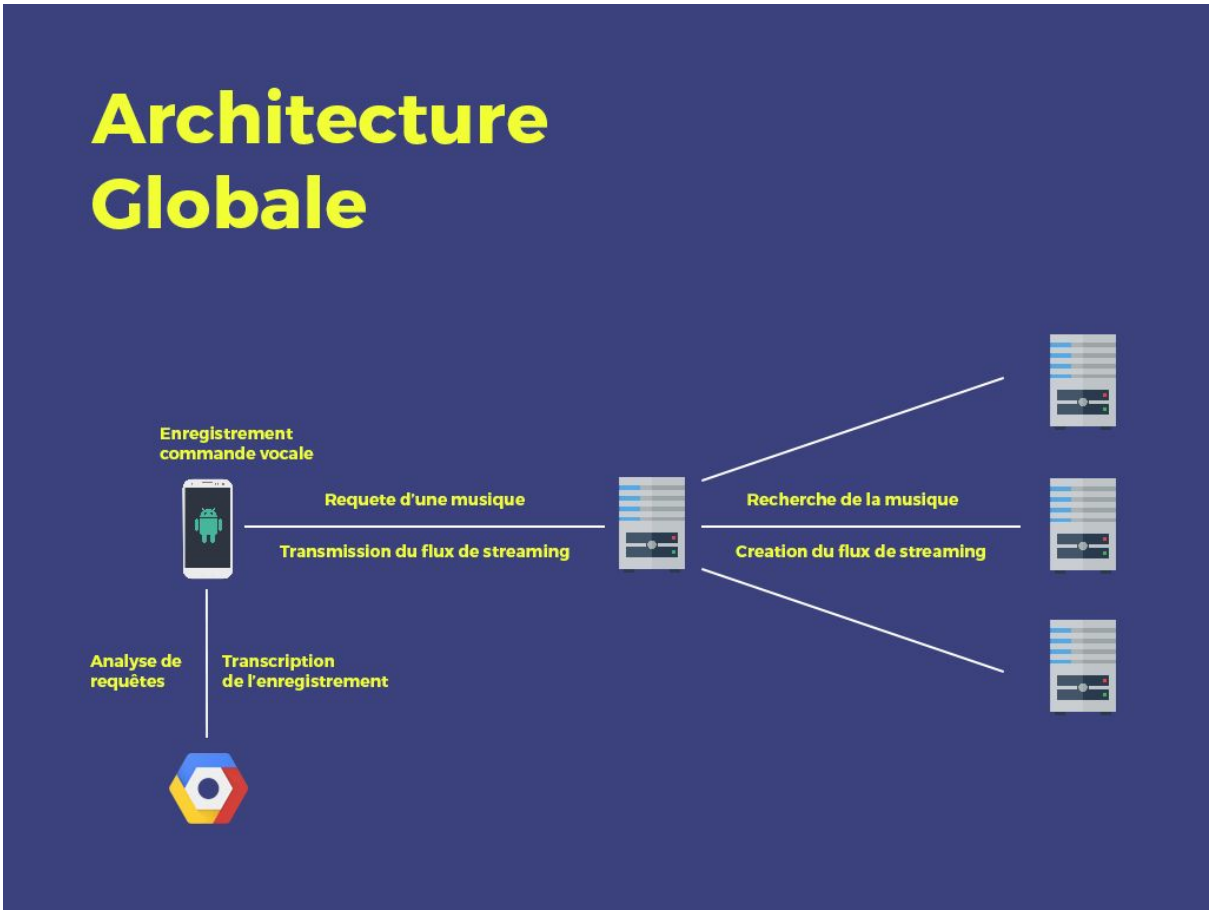
Solution possibles:
Le module sera développé côté serveur via un web service implémentée en java EJB et sera capable d'identifier un couple <Action, Objet>
On pourrait aussi utiliser une librairie python "Flask" qui permet de créer un serveur qui fera office de web service.

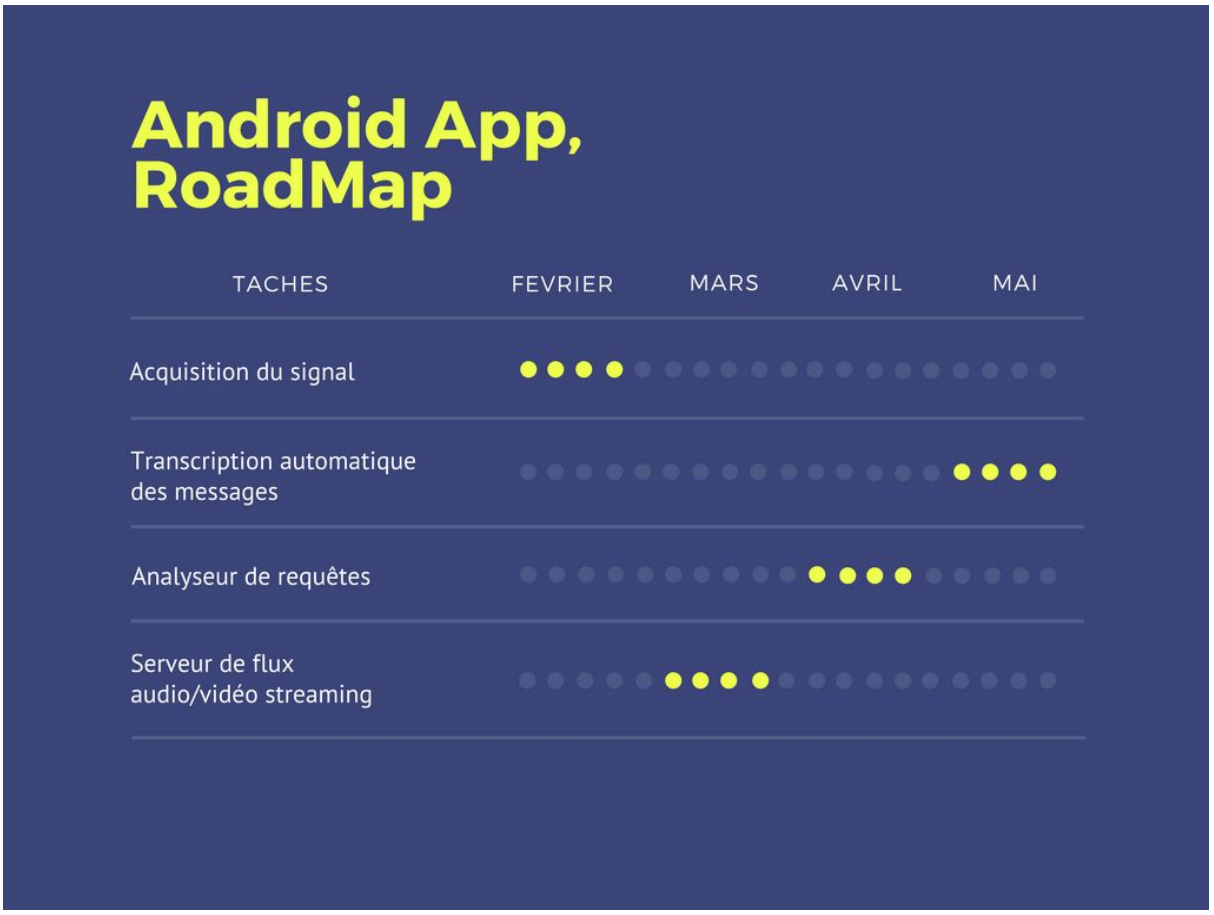
Solution choisie:
Etant donné notre avancement dans les TPs d'EJB, je ne sais pas encore quelle technologie est la plus adaptée, même si j'ai déjà utilisé flask sur un autre projet et que cela a été simple à implémenter.

Serveur de flux audio-vidéo en streaming (Client-Server)

Contexte:
Ce module permet de jouer les morceaux demandés par l'utilisateur sur l'application android. Il a été en partie développé au préalable lors du 1er semestre de Master et sera fini lors du 2nd semestre.
Il permet de créer un méta-serveur recensant tous les serveurs disponibles, avec de la musique, et de les jouer en streaming.

Solution possibles:
Il s'agit d'une application codée en Java coté client et python coté serveur avec une interface Ice permettant de faire le lien entre le client et le serveur. La communication pourra se faire grâce à IceStorm ou JMS.
Pour le serveur de streaming on utilisera la librairie recommandée vlclib qui permet de lire en streaming un fichier mp3, sur un client depuis un serveur distant.





Conclusion

Grâce à la réalisation de ce projet nous pourrons ainsi gagner de l'expérience dans le domaine du développement d'applications mobiles distribuées et des méthodes d’organisation ce qui est non négligeable dans notre parcours, particulièrement en tant qu’étudiant non alternant.