PIERRE PEREZ

Application

Rapport sur l'application

12/05/2017

Prévu initialement

J'avais prévu initialement qu'une application Android communique avec un web service.

Ce web Service devait comporter une base de données qui devait stocker des utilisateurs avec playlists. Le web service devait aussi communiquer avec un serveur de reconnaissance vocal pour traiter la requête du client.

Après tout cela le web service devait ensuite parler avec un serveur de répartition de charges (IceGrid) via Ice pour répartir le trafic entre différents serveurs de stream. Ces serveurs de stream auraient toute la même bibliothèque de fichier. Cela afin de contrer la panne d'un des serveurs et de permettre de répartir au mieux la charge.

Delta entre ce qui était prévu et ce qui est fait

Le delta entre ce qui était prévu est assez important. En effet, je n'ai pas réussi à implémenter IceGrid pour commencer, du coup le web service communique via ICE avec un serveur de stream.

Il n'y a pas de base de données ni de serveur spécifique de reconnaissance vocale. La reconnaissance est directement faite sur le web service via des énumérations de regex.

Les regex ont les inconvénients d'être statiques. Donc l'ordre des mots est important, c'est-à-dire, on est obligé de dire « Action » + « MUSIQUE » pour pouvoir lire la musique. On ne peut pas de faire de phrase complexe tel que « j'aimerais écouter la musique hotel california».

De même si nous disons « stop ColdPlay » il peut ne pas comprendre et renvoyer une erreur à cause du « play » dans le mot « coldPlay ».

Ce qui est fait

L'application Android est fonctionnelle, on peut la démarrer, mettre l'adresse IP du web service et faire la reconnaissance vocale du type « play hotel california ». Il y a ici un délai, puis la musique se lance en streaming.

Le web service reçois la requête du client Android, celui-ci détecte l'action a effectuer entre play et stop puis essai de trouver via ICE et le serveur de stream le nom de la musique pour vérifier si on peut lancer un stream et continuer ou pas.

Une fois que la musique est trouvée, et que nous avons une action de lecture, alors le web serveur communique avec le serveur de stream en ICE, le serveur de stream lance le stream sur une URL unique avec le clientID (qui a été envoyé de bout en bout) et renvoie cette URL au web service. Celui-ci transmet l'URL au client Android et le client Android lance la musique.

Difficulté rencontrée

Durant le projet, j'ai eu des difficultés à avoir un bon plan d'action. Mon organisation a été mauvaise. Je me suis éparpillé dès le début sur toutes les fonctionnalités en même temps, et je n'ai rien construit de valide.

J'ai perdu notamment énormément de temps à vouloir un serveur IceGrid sans avoir les éléments à mettre dedans. Et essayer de déployer se serveur a été un échec…

À ce moment précis, j'ai décidé de recommencer en organisant cette fois-ci ce que je faisais, j'ai alors développé chaque module (serveur de stream / web service / client Android) indépendamment et un a un.

De plus une des difficultés que j'ai rencontrées est sur le web service. J'ai commencé par faire un WSDL « à la main » et j'ai fait du code associé qui était très brouillon. Puis nous avons eu les cours et les TP sur les web service (qui devrait arriver plus tôt dans le cursus) et j'ai pu faire un web service via glassfish et générer automatiquement. Ce fut beaucoup plus rapide à développer et plus efficace.

Axe d'amélioration

Pour mon projet, les axes d'amélioration pourraient être :

- Optimiser l'interface du client Android,
- Améliorer la reconnaissance vocale des regex,
- Mettre en place IceGrid avec la possibilité de rajouter à chaud des serveurs de stream,
- améliorer la gestion de la bibliothèque de musique (avec une copie sur tous les serveurs),
- mettre en place une base de données avec des comptes utilisateurs ainsi que des playlists
- Ajouter des méthodes de reconnaissance telle que la pause, musique suivante, musique aléatoire…