

**Département de génie logiciel**

Rapport de labo 3(T3) : Réingénierie Mobile

**Réingénierie de GSRemote**

|  |  |
| --- | --- |
| Cours | LOG530 |
| Session | Été 2013 |
| Groupe | 01 |
| Numéro de votre équipe |  |
| Chargé(e) de laboratoire | Francis Cardinal |
| Étudiant(s) | Binette Mathieu  Pelletier, François |
| Adresse(s) de courriel | mbinette91@gmail.com  f.pelletier0@gmail.com |
| Code(s) permanent(s) | BINM27029100  PELF28118907 |
| Date | 30 juillet 2013 |

**GRILLE DE CORRECTION**

|  |  |
| --- | --- |
| 1 Introduction /0.25 |  |
| 2 Planification du travail /0.25 |  |
| 3 Rétro-ingénierie du système source /4 |  |
| 4 Ingénierie vers le système cible /4 |  |
| 5 Conclusion /0.5 |  |
| 6 Démonstration du fonctionnement /4 |  |
| 7 Présentation orale /2 |  |
| Total partiel /15 |  |
| **Points négatifs – rapport** |  |
| Références (-10% max) |  |
| Orthographe et grammaire (-10% max) |  |
| Présentation (-10% max) |  |
| Retard (-10% par jour) |  |
| **Note du rapport / 15** |  |

**1. Introduction**

Le système source qui a été choisi pour ce laboratoire est l’application de GSRemote[1] sur Android[2] (captures d’écrans en Annexe I). Cette application permet de contrôler une liste de lecture Grooveshark à partir d’un téléphone Android (ou iPhone) qui agit essentiellement comme une télécommande. L’ordinateur contrôlé doit avoir installé l’extension de navigateur GSRemote[3], puis entrer un code pour connecter le téléphone intelligent. Le système cible sera un port de GSRemote, mais sous la forme d’une application web mobile générique (captures d’écrans en Annexe II).

Dans le cadre du laboratoire, on doit faire une réingénierie du système dans le but de le reproduire en application web mobile générique. Étant donné que le code source de l’application GSRemote n’est pas disponible, il est plutôt difficile de faire une rétro-ingénierie du système à partir de celui-ci. Grooveshark a cependant plusieurs API de disponibles, ce qui rend possible la conception d’un nouveau système basé sur celles-ci. On aura tout de même à faire un travail de rétro-ingénierie en examinant le fonctionnement de l’application ainsi que l’API disponible. Il sera alors possible de spéculer sur l’architecture de l’application ainsi que sur la structure de certaines composantes du système. Ces spéculations permettront d’orienter la conception du système cible.

# 2. Planification du travail

Voici la planification des tâches à réaliser pour compléter ce laboratoire.

|  |  |  |
| --- | --- | --- |
| **Date** | **Tâche** | **Responsable** |
| 12 juillet | Faire le coeur du logiciel et les appels (prototype de base). | Mathieu Binette |
| 12 juillet | Faire l'application côté serveur. | Francois Pelletier |
| 16 juillet | Terminer tous les appels à l'API client de Grooveshark. | Mathieu Binette |
| 16 juillet | Recherche de documentation de l’API Grooveshark. | Francois Pelletier |
| 19 juillet | Intégration de l'API serveur de recherche de Grooveshark. | François Pelletier |
| 19 juillet | Rapport, section « Rétro-ingénierie ». | Mathieu Binette |
| 23 juillet | Interfaces HTML et compagnie. | François Pelletier |
| 23 juillet | Rapport, section « Ingénierie vers Systeme Cible ». | Mathieu Binette |
| 26 juillet | Évaluation du fonctionnement en laboratoire. | - |
| 26 juillet | Préparation de la présentation pour l'exposé. | - |
| 30 juillet | Exposé oral. | - |
| 5 aout | Remise finale du rapport. | - |

# 3. Rétro-ingénierie du système source

Étant donné que le code source de l’application initiale n’était pas disponible, il était difficile de faire un travail complet de rétro-ingénierie. On a donc tenté d’établir une compréhension du logiciel original, sans le code source.

On a commencé par utiliser certains patrons de réingénierie qui semblaient pertinents dans le contexte du laboratoire, en prenant compte du fait qu’on n’a pas accès au code source du logiciel d’origine.

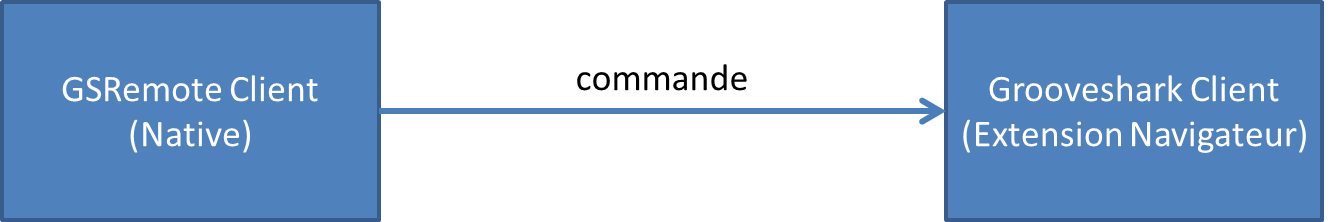
Tout d’abord, tel que recommandé dans le patron « Appoint a Navigator », on a décidé de nommer un membre de l’équipe comme responsable de la vision architecturale du projet, soit François Pelletier. De plus, on a choisi le patron « Speak to the Round Table », qu’on met en place facilement en s’assurant de bien communiquer durant les périodes prévues pour le laboratoire (même si elles étaient souvent effectuées via Google Hangouts).

On a également utilisé quelques patrons de réingénierie du chapitre trois, soit « Most Valuable First », « Skim the Documentation » et « Do a Mock Installation ». Pour « Most Valuable First », on s’est assuré de se concentrer sur les fonctionnalités importantes de l’application et de garder l’implémentation des moins importantes pour la fin. On a également décidé de laisser tomber certaines fonctionnalités, comme la visualisation d’une fiche d’artiste, au profit de certaines fonctionnalités qu’on a jugées importantes et qui n’étaient pas dans le système source (la version gratuite), par exemple, la gestion du volume.

Dans le cas du patron « Skim the Documentation », on recommande de lire la documentation disponible afin de distinguer des éléments importants du logiciel, ce qu’on a fait dans le cas de la documentation des API disponibles.

Suivant les recommandations du patron « Do a Mock Installation », on a installé l’application sur un téléphone intelligent. Ensuite, on a observé le fonctionnement de l’application. Grâce à ces observations, on a tenté de deviner comment fonctionne l’architecture de l’application initiale. On a ébauché le schéma de la figure 3.1.

**Figure 3.1 : Première ébauche de l’architecture du système source**



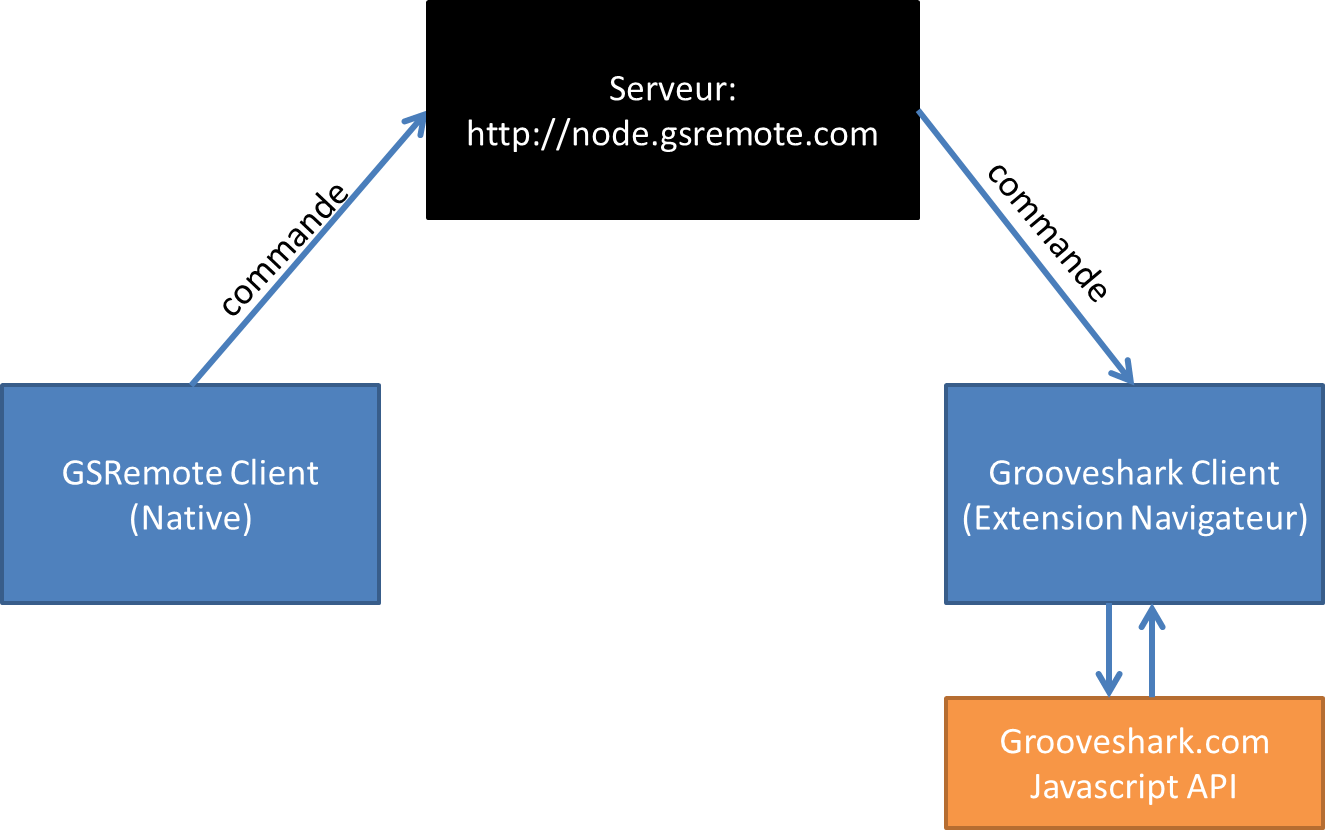
Une des premières choses qu’on a effectuée par la suite fut d’essayer de décompiler l’application Android. Étant donné que le code était obscurci, la valeur de celui-ci était réduite. Puisque le temps à investir sur le nettoyage du code était considérable, on a décidé de s’en tenir à tenter de comprendre le fonctionnement interne de l’application en l’utilisant et en observant son comportement.

Par la suite, on a remarqué que bien que le code de l’extension client était également obscurci, il existait une version obsolète de l’extension dont le code n’était pas obscurci. Cela a fait en sorte que l’on comprenne le fonctionnement interne de l’extension, même si ses interactions avec l’API JavaScript de Grooveshark (qui avait été modifié depuis) étaient obsolètes. En effet, ces observations on permis de conclure qu’une composante serveur gérait les connexions entre les manettes et les sessions grooveshark et s’occupait de relayer les commandes.

De plus, après une étude plus en profondeur de cette ancienne extension, on a découvert qu’elle interagissait avec un API JavaScript de Grooveshark.

On a donc amélioré le schéma de la figure 3.1 pour obtenir une version plus raffinée de l’architecture du système source,, soit la figure 3.2.

**Figure 3.2 : Ébauche rafinée de l’architecture du système source**

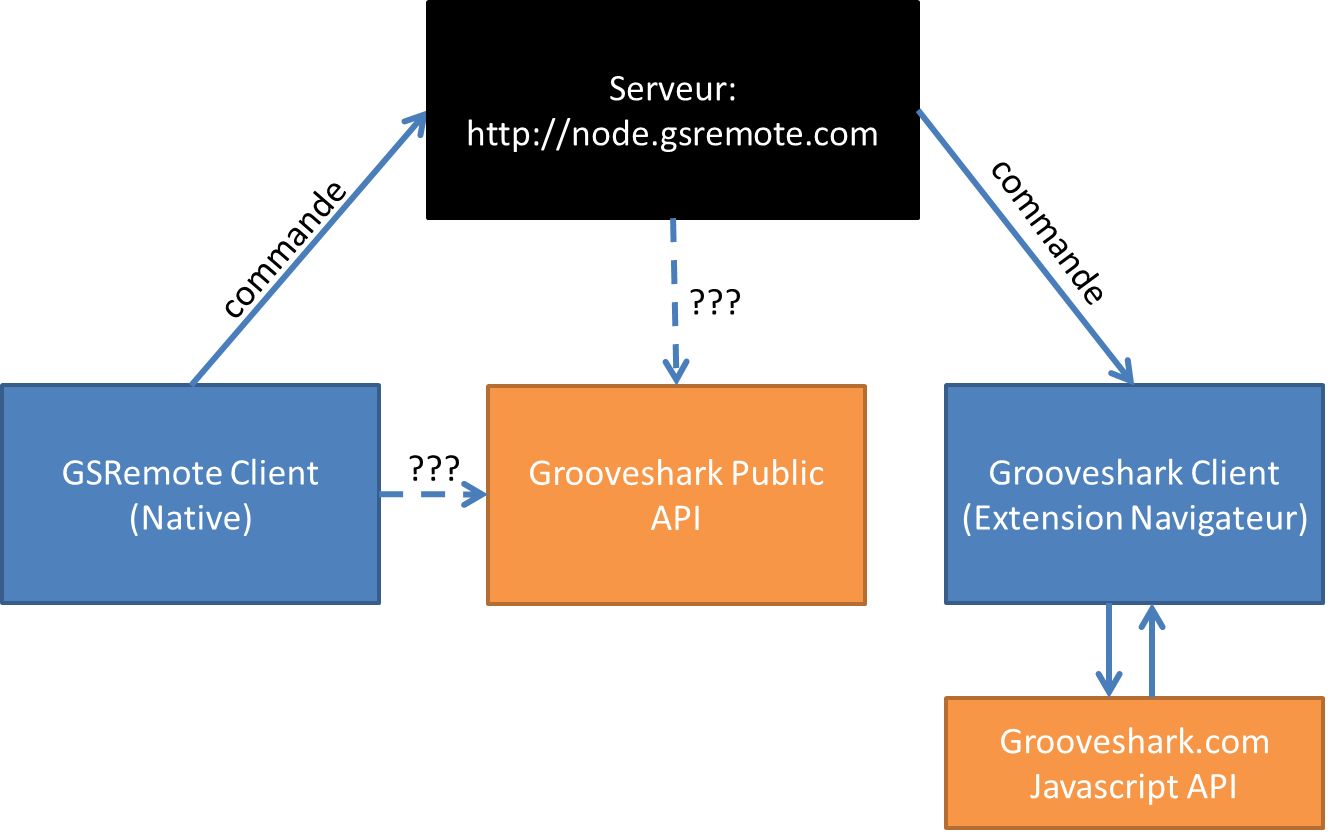


Cependant, aucune trace dans l’ancienne extension du code pour la fonctionnalité de recherche de chanson disponible dans l’application originale. De plus, même après un examen approfondi de l’API JavaScript de Grooveshark, aucune trace de cette fonctionnalité.

Après quelques recherche, on a cependant découvert un autre API de Grooveshark, soit le Grooveshark Public API, qui offre une interface riche pour la recherche d’artistes, d’albums et de chansons. Bien qu’il soit impossible de savoir avec certitude où cette interface est utilisée dans le système source, on sait que ce n’est pas dans l’extension du navigateur.

Bref, cette approche itérative a permise de conclure que le système source était divisé en cinq (5) composantes clés, soit un serveur, une extension de navigateur, l’application Android, l’API Grooveshark publique et l’API Grooveshark JavaScript (voir Figure 3.3).

**Figure 3.3 : Architecture du système source**



On a donc segmenté la suite du travail de rétro-ingénierie en quatre parties: la compréhension du serveur, de l’application Android, de l’extension client (navigateur) et finalement une recherche sur les interfaces des différentes API de Grooveshark.

Pour la composante serveur, il n’y avait pas grand chose à faire. À partir des observations faites plus tôt, on sait qu’il agit en grande partie comme intermédiaire entre les manettes et les clients Grooveshark (GS). Il a également un rôle à jouer dans la gestion des liaisons entre les manettes et les clients GS. Bref, tout ce qui est connu sur cette composante provient des interactions de celle-ci avec l’extension obsolete, ainsi que de l’expérience au travers de l’utilisation de l’application.

Pour l’application Android, tel que mentionné plus tôt, bien qu’on ait essayé de décompiler l’application, le code obscurci réduit la valeur que l’on peut en tirer en peu de temps. On a donc décidé de s’en tenir à l’observation de celle-ci afin d’en tirer des conclusions sur son fonctionnement.

Pour l’extension du navigateur, on a eu plus de chance. Même si l’extension trouvée était obsolète, la structure de celle-ci était très fort probablement représentative de la nouvelle structure. Seuls les appels à l’API JavaScript de Grooveshark semblent obsolètes.

Finalement, après une recherche sur les différents API de Grooveshark, on a constaté que l’API Publique nécessite une clé de développement pour l’utiliser, et permettra de faire des recherches riches à multiples critères. L’API JavaScript s’occupe principalement de la liste de lecture courante et offre une interface pour ajouter/retirer des chansons, ainsi que d’avancer, de jouer, d’arrêter et de modifier le volume de la chanson courante.

Les deux API sont bien documentées en ligne [4][5] et de la documentation additionnelle se trouvait dans l’extension originale de GSRemote, dans laquelle il y avait le code pour les appels à l’ancien API client de Grooveshark, qui était pas trop différent du nouvel API JavaScript.

Pour le fonctionnement de la fonctionnalité de recherche, celle-ci se trouvait soit dans l’application Android, soit dans la composante serveur, pour lesquelles le code source n’est pas disponible. On a donc comme information seulement celle qui est disponible dans la documentation (les interfaces).

Après une utilisation exhaustive de l’application dans le but de comprendre son fonctionnement, on a remarqué que la convivialité de l’application était déficiente. Par exemple, la flèche de retour en arrière est partiellement fonctionnelle (elle ferme parfois complètement l’application) et il est impossible de changer le volume (cette fonctionnalité est cependant disponible dans la version payante de l’application).

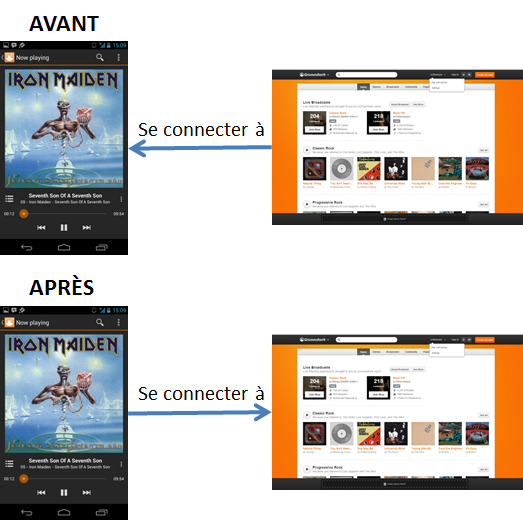
# Au niveau des performances, le client Grooveshark répondait très rapidement aux commandes de la manette. On a donc déterminé qu’il y a une connexion persistante entre les deux composantes et le serveur de Grooveshark pour avoir de tels temps de réponse.

# 4. Ingénierie vers le système cible

On a choisi de faire une réingénierie du système présenté plus tôt vers une application mobile générique puisque la nouvelle application serait alors supportée sous tous les types de mobiles, en plus de la nouvelle plateforme de Firefox OS, qui ne supporte que les applications ouvertes en HTML5. De plus, cela permet de n’avoir qu’à effectuer des changements futurs une seule fois au lieu de plusieurs fois sur plusieurs plateformes. Étant donné qu’aucun problème relié au type d’application choisi n’a été rencontré, on referait le même choix si on devait recommencer le projet.

Une des premières choses qu’on a modifiées fut d’inverser le sens de contrôle de l’application originale. Au départ, le client Grooveshark devait se connecter à une manette et maintenant, c’est la manette qui doit se connecter au client Grooveshark (Figure 4.1), ce choix semblait plus approprié vu la nature d’une télécommande. De plus, si l’on est assis sur notre divan avec notre téléphone, la dernière chose que l’on veut faire est de se lever pour aller à l’ordinateur pour entrer le code de notre télécommande.

**Figure 4.1 : Inversion du sens de connexion**



De plus, on a réglé quelques bogues au niveau de l’interface. En fait, il est difficile de dire si ce sont des bogues ou des fonctionnalités de l’ancienne interface, mais le système de navigation n’était pas très intuitif. Particulièrement le bouton de retour en arrière de l’interface (en haut à gauche), qui fermait parfois l’application au complet. On a donc revu le système de navigation du UI dans l’application.

On a également ajouté une fonctionnalité pour contrôler le son. Cette fonctionnalité n’était pas disponible sur l’application source qu’on a choisie, mais elle l’était sur la version payante.

Le système original était déjà mobile, donc pour la grande majorité, était déjà adapté à la technologie mobile. Cependant, tel qu’expliqué plus tôt, on a tout de même effectué des modifications au niveau de l’interface pour le rendre plus intuitif pour les utilisateurs mobiles qui sont habitués à certains standards, par exemple le bouton de retour en arrière dans la barre de navigation en haut ne ferme pas l’application.

Étant donné que le système est maintenant une application web mobile générique, il fonctionne de la même façon sur tous les systèmes mobiles. Cela permet de faciliter la maintenance puisque chaque modification doit être faite dans un seul système, plutôt qu’une fois par système, pour chaque plateforme mobile. On a également rajouté un contrôle pour le son, qui est une fonctionnalité très utile.

Cependant, la réingénierie du système source vers notre système cible ne s’est pas effectuée sans aucun problème!

Pour commencer, on n’avait pas accès au code source de l’application originale. Cependant, on a quand même pu déterminer les composantes du système original et les refaire lorsque nécessaire, par exemple le serveur. On a donc fait plusieurs hypothèses à propos du fonctionnement de l’application originale. Heureusement, on a pu récupérer le code de l’extension de navigateur client et cela a permis d’analyser une partie du code composant l’application. Cependant, pour ce qui est du code serveur, on n’avait pas de moyens d’obtenir le code de cette composante et on a donc dû faire de nouvelles hypothèses concernant le fonctionnement, afin de le reproduire.

De plus, pour la partie de l’application mobile, soit la manette, on a essayé de décompiler l’application Android, mais sans grand succès. Le code était plutôt illisible et beaucoup trop difficile à comprendre. Le code était obscurci et aurait demandé un travail massif afin de pouvoir le rendre lisible et compréhensible. On l’a donc refait à partir des observations de l’application originale seulement.

Pour terminer, lors de l’implémentation de la fonctionnalité de recherche, on a dû remplir une demande d’accès à l’API publique de Grooveshark, qui offre une interface de recherche très riche (plusieurs critères de recherche). Cependant, étant donné que la requête a été plusieurs semaines sans réponse, on a décidé de se servir de leur API non officiel, qui offre une fonctionnalité de recherche beaucoup moins riche, soit Tinysong [6]. Cela a forcé à revoir l’interface de recherche et la simplifier puisqu’il était impossible de faire des recherches complexes, tel qu’une recherche par album.

Bref, la réingénierie a été un succès, l’application fonctionne aussi bien sur Android, en plus de maintenant être disponible sur toutes les autres plateformes mobiles.

De plus, la nouvelle version n’a qu’à être modifiée une seule fois pour que toutes les plateformes mobiles aient accès à cette nouvelle version. Cela élimine donc les problèmes possibles au niveau de la synchronisation entre différentes versions de l’application, par exemple un changement sur Android non implémenté dans la version IOS.

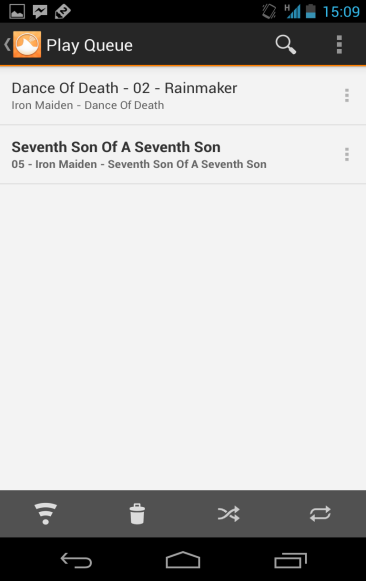
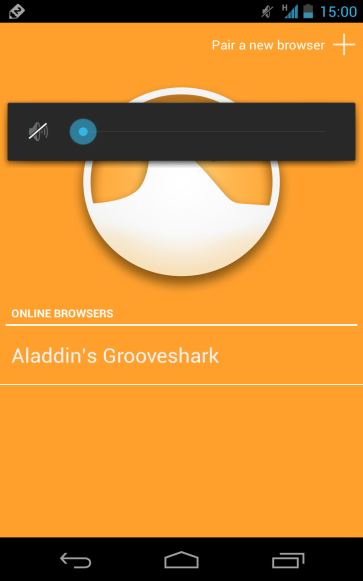
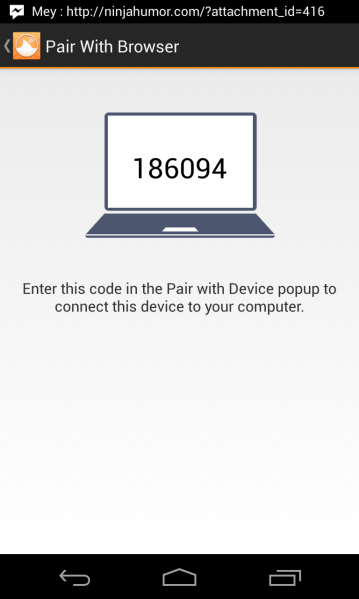
# 5. Conclusion

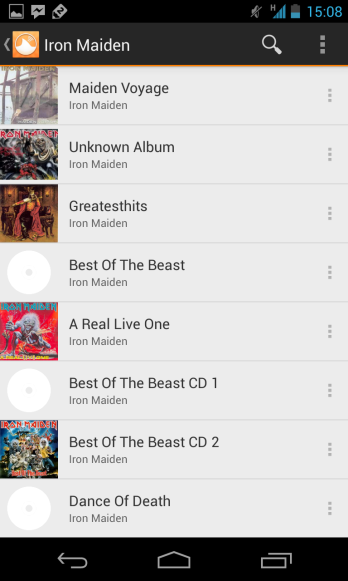
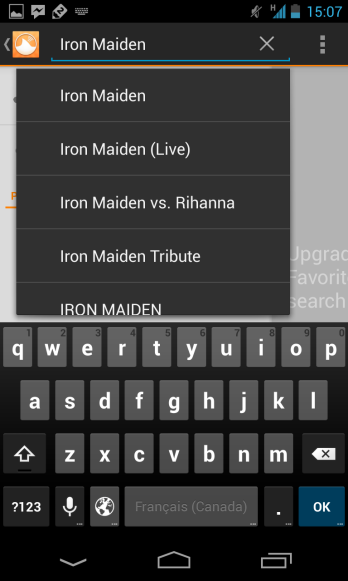
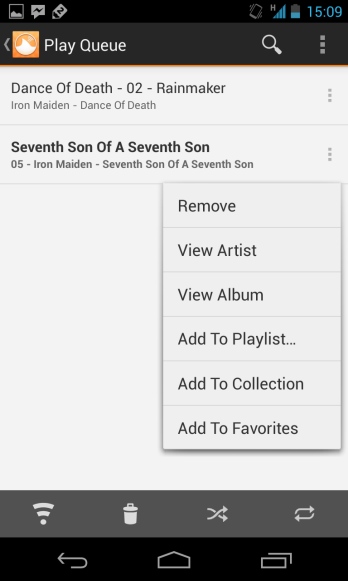
Dans le cadre du laboratoire, on devait choisir un système et en faire un port pour mobile. Le système qui a été choisi est une application Android pour contrôler une liste de lecture Grooveshark. Il a été choisi de prendre cette application et de faire un port pour web mobile, en se concentrant sur Firefox OS.

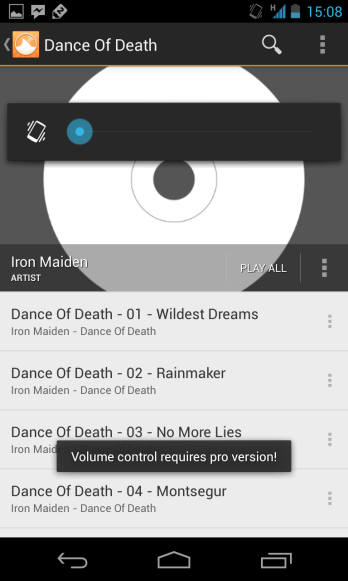
Ce système est particulier pour un travail de réingénierie, car il comporte plusieurs API, mais le code source de l’application n’est pas accessible. En utilisant plusieurs patrons de réingénierie, on a été en mesure de faire une réingénierie efficace, en se concentrant sur les aspects les plus importants du projet et en gardant une vision unique tout au long du travail.

On a remarqué que même sans avoir accès au code source, il est possible de faire un bon travail de réingénierie. Bien que plusieurs patrons de réingénierie ne s’appliquent tout simplement plus, d’autres sont très efficaces et s’il sont respectés, augmentent les chances de succès du projet. On croit qu’il serait intéressant qu’un travail de réingénierie sur un projet dont le code source n’est pas disponible soit donné chaque session. Il s’agit d’une expérience différente de ce qui est donné en ce moment, mais qui est tout à fait possible de rencontrer dans une véritable expérience de travail.

# Annexe I

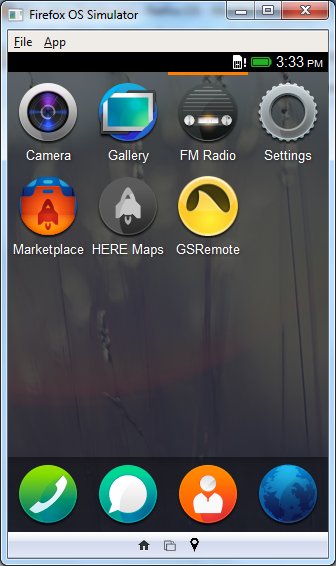
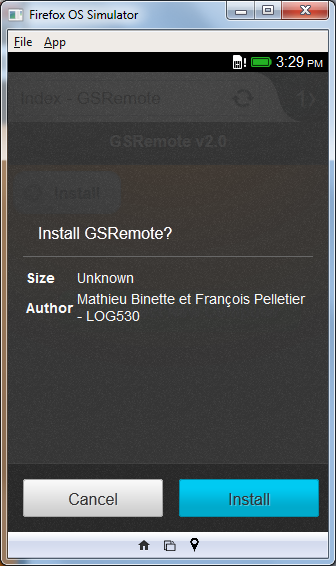
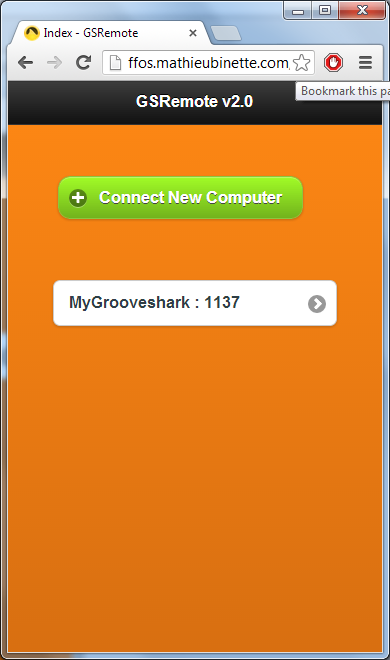
**Application source**

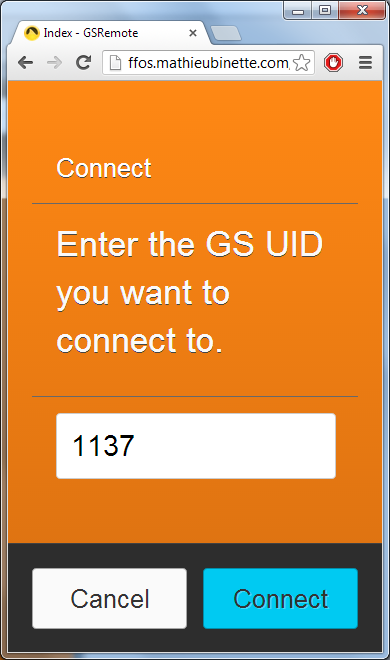
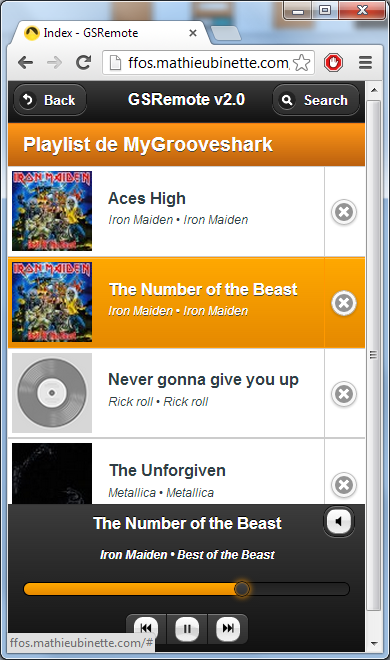
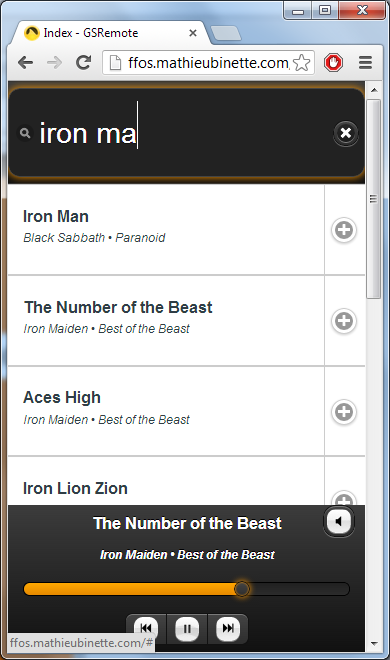


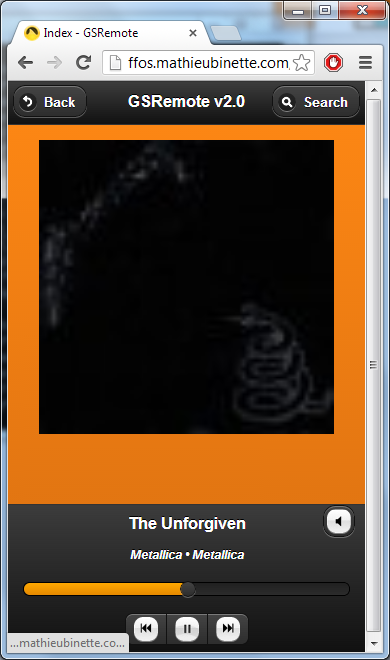
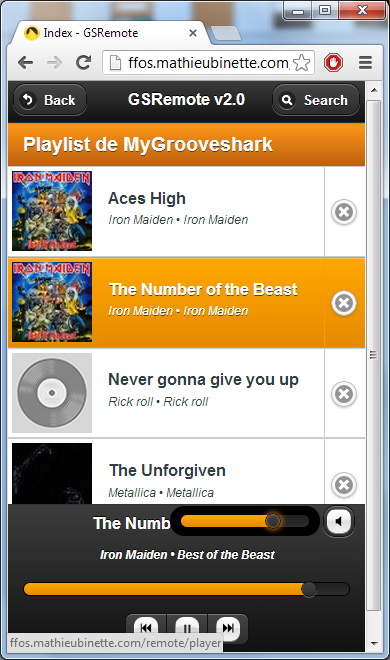
 

**Annexe II**

**Application cible**

**** **** 

# Références

*[1] Grooveshark Remote for iPhone and Android, [En ligne].*

[*http://gsremote.com/*](http://gsremote.com/) *(Consulté le 30 juillet 2013).*

*[2] Remote pour Grooveshark - Applications Android sur Google Play, [En ligne].*

[*https://play.google.com/store/apps/details?id=uk.co.awesomedigital.gsremote*](https://play.google.com/store/apps/details?id=uk.co.awesomedigital.gsremote)

*(Consulté le 30 juillet 2013).*

*[3] Chrome Web Store - Grooveshark Remote, [En ligne].*

[*https://chrome.google.com/webstore/detail/grooveshark-remote/fbpifhknilaflibiifjhhofddbbchmhh*](https://chrome.google.com/webstore/detail/grooveshark-remote/fbpifhknilaflibiifjhhofddbbchmhh) *(Consulté le 30 juillet 2013).*

*[4] Grooveshark Developers, Javascript API [En ligne].*

[*http://developers.grooveshark.com/docs/js\_api/*](http://developers.grooveshark.com/docs/js_api/) *(Consulté le 30 juillet 2013).*

*[5] Grooveshark Developers, Public API [En ligne].*

[*http://developers.grooveshark.com/*](http://developers.grooveshark.com/) *(Consulté le 30 juillet 2013).*

*[6] Tinysong | Type in a song and make a free music link to share music with friends, [En ligne].*

[*http://tinysong.com/*](http://tinysong.com/) *(Consulté le 30 juillet 2013).*