Département de génie informatique et génie logiciel

INF3995
Projet de conception d'un système informatique

Proposition répondant à l'appel d'offres
no. H2019-INF3995 du département GIGL.

Système audio pour café Internet

Équipe No 3

Nom	Signature
Constantin Bouis	
Christella Cidolit	
Léandre Guertin	
Anis Redjdal	
Issifath Sanni	

04 Février 2019

Table des matières

1. Vue d'ensemble du projet

1.1 But du projet, porté et objectifs (Q2.1et Q4.1)

Le présent document détaille la planification d'un projet de conception d'un système informatique. Ce projet a pour but de permettre au café-bistro Élévation d'offrir à ses clients un système de musique participatif. L'idée serait de permettre aux utilisateurs de la connexion wifi qu'offre le café, de proposer des musiques qui seront transmises à un serveur qui les diffusera, tout en prévoyant la possibilité pour les superviseurs et le gérant d'administrer la diffusion dans le café.

Dans un premier temps, notre objectif est de répondre aux demandes du propriétaire du café-bistro dans la réalisation de ce système. Sa satisfaction et celle des utilisateurs des différentes parties que nous serons amenés à concevoir, en termes de simplicité d'utilisation, d'efficacité et d'aspect esthétique sont donc très importantes dans la réalisation de ce projet. Nous avons aussi pour objectif de fournir un système offrant de bonnes performances (en matière de son, de communication entre les composantes, de rapidité, etc.) et respectant les spécifications techniques requises. Enfin, nous prévoyons respecter au meilleur de nos capacités les échéanciers et le budget qui nous sera alloué.

Afin de répondre à ces objectifs, le projet sera livré en trois composantes principales. La première sera un serveur, s'exécutant sous Linux avec les ressources d'une carte FPGA, qui s'occupera de recevoir et diffuser les musiques des différents utilisateurs. La seconde sera une application Android pour les clients du café-bistro qui se connectera au serveur afin d'envoyer notamment les chansons qu'ils auraient sélectionnées. Cette application offrira également la possibilité aux superviseurs de contrôler une partie du système. La troisième composante de ce projet sera un logiciel sur ordinateur qui offrira à l'administrateur un accès aux activités du serveur et aux options du système en général.

1.2 Hypothèse et contraintes (Q3.1)

En ce qui concerne les hypothèses, notre plan repose sur une interface graphique compilée en Java pour l'application PC. On suppose que cette interface sera relativement facile à coder vu le nombre de libraires Java plutôt variées qui nous faciliteront la tâche de ce point de vue. La fenêtre de l'application reposera donc sur des libraires Java importées pour notre projet.

De plus, on suppose, pour le bon déroulement de notre projet, que le nombre de personnes qui fréquentent le café-bistro et donc qui ont recours à cette application n'est pas trop élevé. On entend par là que si un trop grand nombre de personnes décide d'envoyer des fichiers MP3 à notre serveur, celui-ci ne sera probablement pas en mesure de répondre à toutes les demandes de tous les clients. On pose donc l'hypothèse que le nombre de clients qui utilisent l'application est gérable et approprié pour un café-bistro à fréquentation moyenne.

En dernière hypothèse, on suppose que les clients ont sur leur appareil Android les fichiers audio en format MP3 qu'ils veulent envoyer en étant connectés au wifi du café-bistro Élévation pour être sur le même réseau que le serveur.

Pour ce qui est des contraintes, notre serveur sera codé en C++. Bien que ce ne soit peut-être pas le langage le plus approprié pour coder un tel serveur, l'utilisation d'une FPGA comme support et le fait de vouloir un système performant justifient cette contrainte. En outre, la gestion des interfaces de communications entre nos applications sera basée sur le protocole REST.

Toujours dans le contexte d'une FPGA, la mémoire du serveur repose sur celle de la carte qui a une mémoire RAM de 512 MiB, ce qui restreint l'espace de stockage des fichiers MP3. Pour respecter cette contrainte d'espace, nous nous assurons de limiter la file d'attente du serveur à dix musiques. De plus, nous allons vérifier la taille des fichiers MP3 autorisés pour éviter de surcharger la connexion et le serveur.

Pour en revenir au code de l'application sur PC, nous avons la contrainte d'utiliser un langage natif et de ne pas avoir recours à des interfaces Web. Nous choisissons alors Java afin d'implémenter l'interface graphique de l'application de l'administrateur.

Ces hypothèses et contraintes sont donc les éléments sur lesquelles notre équipe se reposera durant la conception de ce projet.

1.3 Biens livrables du projet (Q2.3)

Dans le but de faciliter la mise en place, l'utilisation et la maintenance du système, nous fournissons plusieurs artéfacts tout au long de la conception de notre projet. L'ensemble des documents que nous livrerons sont classés dans le tableau ci-dessous :

Date	Documents
04 Février 2019	Diagrammes UML du serveur, de l'application Android, du logiciel administrateur
	Diagramme de Gantt
28 Mars 2019	Livrable 1**
	Documentation*
11 Avril 2019	Livrable 2***
	Documentation *
	Rapport technique du projet

Tableau I : Artéfacts classés par date

***Documentation** : elle comprend la documentation relative à l'installation des librairies, à la configuration requise et à l'utilisation du système.

****Livrable 1 :** Les fonctionnalités couvertes sont listées ci-après :

- **Serveur :** gère les connexions clients et les requêtes de ces derniers, en permettant l'authentification des superviseurs et de l'administrateur et une liste noire qui empêche certaines connexions. Il génère des logs et les statistiques, et permet de décoder des fichiers MP3 dans sa file d'attente à laquelle on peut ajouter ou retirer des éléments.
- **Application Android :** les clients ont accès à leur page d'accueil pour sélectionner une de leur musique et l'envoyer au serveur. Il peut aussi afficher les paroles et voir la file du serveur. Le superviseur peut s'authentifier, changer son mot de passe, et voir la liste des musiques du serveur ainsi que leurs informations.

- **Application PC (entière) :** l'application a une interface graphique qui lui permet de s'authentifier de manière sécurisée, changer son mot de passe, gérer les comptes superviseurs et voir les logs ou statistiques en provenance du serveur.

*****Livrable 2 :** Contient le système au complet, c'est-à-dire le livrable 1 ainsi que :

- **Serveur :** la possibilité de changer l'ordre de la file d'attente ainsi que le volume du son. Le serveur sera également lancé au démarrage du système Linux.
- **Application Android :** le client pourra retirer sa musique de la file du serveur, de même pour le superviseur qui pourra en plus gérer le son et changer l'ordre de cette liste. Ce dernier pourra de plus voir et modifier la liste noire des usagers bloqués.

2. Organisation du projet

2.1 Structure d'organisation

Pour développer le système correspondant aux spécifications reçues, nous serons une équipe de 5 développeurs. Étant donné que le produit à concevoir comprend trois grandes parties à savoir l'application mobile Android, le serveur et l'application PC, nous avons décidé d'une répartition globale des tâches qui se présente comme suit :

- Anis Redjdal: Application PC et Application Android
- Christella Cidolit : Serveur et Application PC
- Constantin Bouis: Application Android et Application PC
- Issifath Sanni : Application Android et Serveur
- Léandre Guertin: Application Android et Serveur

Cette première distribution a été faite en fonction des préférences et des domaines de compétence de chacun. Néanmoins elle n'est pas rigide et chaque membre de l'équipe devra s'efforcer de connaître l'entièreté du système à un niveau acceptable. Cela dit, nous comptons faire en sorte qu'il y ait au moins deux personnes-ressources pour une fonctionnalité donnée.
Pour mener à bien le projet, nous avons également jugé nécessaire d'attribuer des rôles secondaires à chacun. Ainsi :

- Anis Redjdal : responsable du dépôt git. Il s'agira pour lui de s'assurer qu'un certain nombre de règles énoncées par l'équipe sur la gestion du répertoire git sont respectées afin que le code source disponible sur la branche principale soit toujours fonctionnelle.
- Christella Cidolit : responsable Redmine. Son rôle sera de veiller à ce que chaque membre de l'équipe remplisse ses rapports d'heures de travail passées sur le projet sur Redmine.
- Constantin Bouis : chargé de communication. En tant qu'expert de la communication, il devra s'assurer que tout le monde ait la bonne information en utilisant Slack comme canal de diffusion.
- Issifath Sanni : supervision des tests. Sa mission sera tout au long du développement, de contrôler la couverture du code qui est produit. Cela suppose, de vérifier que chaque membre de l'équipe a prévu des tests et la qualité de ces derniers.
- Léandre Guertin : responsable de l'avancement et du fichier de suivi. Il sera attendu de lui, de recueillir auprès des membres de l'équipe, leur progression dans les tâches hebdomadaires qui leur seront assignées. Il s'ensuivra une prise de décision de l'équipe par rapport à la planification des tâches ; par exemple, rééquilibrer les tâches s'il le faut.

Par ailleurs, en ce qui concerne les rencontres d'équipe, elles auront une durée totale de 10h par semaine et elles seront animées par chaque membre de l'équipe dans un mécanisme de rotation. De plus, l'équipe fonctionnera en mode Agile et un Scrum réunira les membres tous les jeudis de 8h45 à 9h.

2.2 Entente contractuelle

Dans le cadre de ce projet, nous suggérons un contrat à terme. En effet, avec un tel contrat, nous pourrons commencer à travailler sur le projet au plus tôt ce qui est un avantage puisque le nombre d'heures attendues est très limité. Nous pensons qu'un contrat à terme s'avère un choix judicieux parce qu'il s'agit de concevoir un système de trois applications ayant chacune son niveau de complexité non négligeable et nous apportons notre expertise en tant que développeurs. En outre, étant donné que ce type de contrat permet au moins une soumission avant la livraison du produit final, le promoteur pourrait avoir le loisir d'apprécier la justesse du respect des spécifications ou encore d'apporter plus de précisions, voire des modifications aux applications, comme mentionnées dans l'appel d'offres.

3. Solution proposée

3.1 Architecture logicielle sur serveur (Q4.2)

L'architecture de notre serveur est basée sur une classe serveur qui est reliée à la classe RequestManager qui gère tous les types de requêtes ainsi qu'a la classe ActivityManager qui gère nos fichiers de log et de statistiques. On a également une classe AdministrationManager pour gérer les comptes des superviseurs et une classe SoundManager pour gérer nos musiques. Se référer l'annexe 1 pour le diagramme présentant une vue de l'ensemble du serveur.

3.2 Architecture logicielle sur tablette (Q4.3)

L'application Android est faite avec des patrons de conception MVP, où les modèles sont affichés par la vue et le présentateur s'occupe de la logique du code de la vue puis modifie le modèle. Avec cette configuration il est beaucoup plus facile de séparer ce que les utilisateurs voient versus ce que la logique de code fait. L'application Android peut être séparée en six modules distincts, à savoir la page d'accueil, la vue de la liste de musiques locale, la vue de la liste sur le serveur, la connexion du superviseur, la vue de la liste noire par le superviseur et les paramètres de volume et du compte superviseur.
Se référer l'annexe 2 pour la vue de l'ensemble des modules et de leur interaction.

Puisque chaque module est fait avec le patron de conception Modèle-Vue-Présentateur, il sera beaucoup plus facile d'y intégrer une interface utilisateur conviviale, facilement modifiable à tout moment dans le projet. La distinction de chacune des composantes permettra d'accélérer le développement en parallèle des modules.

3.3 Architecture logicielle de l'application sur PC

Concernant l'application sur PC, nous utiliserons la librairie SwingX en accompagnement avec WindowBuilder. Donc nous emploierons le langage de programmation Java. Cette librairie simple et efficace nous permet de concentrer nos efforts sur la logique et la sécurité de l'application plutôt que sur la vue. Afin de faciliter le développement, nous allons utiliser le modèle de conception MVC. Tout comme l'architecture Android, il sera plus facile de travailler en parallèle sur un module puisque la vue, soit l'interface utilisateur, sera séparée de la logique de code, c'est-à-dire le contrôleur.
L'architecture de l'application suivra le modèle disponible à l'annexe 3.

4. Processus de gestion

4.1 Estimations des coûts du projet

Les développeurs-analystes seront payés à un taux horaire de 130 dollars canadiens. Puisque le projet sera d'une durée estimée totale de 420 heures, le montant total pour payer les développeurs-analystes sera de 54600 dollars canadiens. Dans le but de développer l'application Android, une tablette sera nécessaire afin de réaliser des essais techniques. Le prix de cette tablette est de 200 dollars canadiens [1]. Il faut voir cette tablette comme un investissement puisqu'elle vous reviendra à la fin du projet. Vos superviseurs auront recours à cette tablette afin d'administrer le système de son du café-bistro, ce qui justifie une dépense à l'avance, mais qui sera nécessaire aux développeurs afin d'assurer la qualité du produit qui vous sera remis. Il en sera de même pour un FPGA/SoC Zynq XC7Z020 de Xilinx pour la somme de 449 dollars canadiens [2] ainsi qu'une carte SD à 40 dollars canadiens qui servira d'espace de stockage pour le système d'exploitation. Ce FPGA ainsi que la carte SD serviront de serveur pour nos développeurs, mais sera aussi le serveur qui vous sera fourni, ce qui garantit une qualité de notre produit. La somme des coûts revient donc à 55289 dollars canadiens.

4.2 Planification des tâches (Q2.2 et Q11.2)

Afin de planifier notre projet pour les 3 prochains mois, nous l'avons découpé en tâches réparties sur 4 sprints au cours des 9 prochaines semaines. Pour chaque tâche, nous avons donc estimé le temps de développement qu'elle demande et assigné une personne responsable de son avancement. Le tout nous représente environ 404 heures de développement. L'intégration et une marge d'ajustement de 16 heures nous permettent d'arriver aux 420 heures mentionnées précédemment. Un tableau présentant des informations est disponible en annexe 4.
Dans le diagramme de Gantt disponible sur Redmine et présenté ici à l'annexe 5, on retrouve ces informations, et le Roadmap (annexe 6) permet de pouvoir suivre la répartition des fonctionnalités au long des 4 sprints.

4.3 Calendrier de projet (Q3.3)

Pour fonctionner en mode Agile de manière effective, nous avons planifié les tâches en sprints comme mentionné dans la section de planification ci-dessus. Le tableau qui suit présente les dates auxquelles nous prévoyons terminer nos sprints.

Livrables	Sprints	Date cible de terminaison
Premier Livrable	Sprint I	Lundi 18 Février
	Sprint I	Lundi 11 Mars
	Sprint III	Lundi 25 Mars
Deuxième livrable	Sprint IV	Lundi 8 Avril

Tableau II : Calendrier du projet

4.4 Ressources humaines du projet

Les ressources humaines nécessaires à la réalisation de notre projet selon la planification des tâches est notre équipe de cinq personnes tous étudiants en troisième année à Polytechnique Montréal en génie informatique soit :

- Constantin Bouis : A participé aux LH games 2017 pour un projet IA. Il a également développé un jeu d'arcade en Python d'une balle rebondissante. Sa force repose sur ses connaissances en sécurité informatique.
- Christella Cidolit : Ancienne stagiaire développeur C++ chez CAE, très impliquée dans ses projets intégrateurs, elle est familière avec la méthode agile de développement et ses compétences en débogage et test sont un atout.
- Anis Redjdal : Étudiant international venu d'Algérie, il a fait son stage chez Fluxym en tant développeur Web et Back-end, sa flexibilité et son adaptation sont ses atouts.
- Issifath Sanni : Précédemment analyste qualité chez Intact Assurance, elle a acquis de l'expérience en cycle de développement de logiciel; également tutrice en génie informatique à Polytechnique, ses connaissances en C++ sont à jour.
- Léandre Guertin : Ancien stagiaire chez McKesson développeur d'une application web avec Spring MVC et chargé de laboratoire pour le projet intégrateur 1, les performances de Léandre en Java et C++ nous seront très utiles durant le projet.

5. Suivi de projet et contrôle

5.1 Contrôle de la qualité

Afin d'assurer la qualité de notre produit, la présente liste d'exigences devra être conforme lors de la remise du projet final. De plus, cette liste devra être validée autant pendant qu'à la fin du développement des applications pour garantir la fiabilité de toutes nos fonctionnalités. Nous allons donc faire la vérification:

- des entêtes de fichier, de classe et de fonction sur chaque fichier de code source.
- que la compilation du code source ne fournit aucune erreur et aucun avertissement.
- que l'interface permette de choisir de se connecter en tant que client ou superviseur.
- que le client peut regarder les sons disponibles sur son appareil hors connexion.
- que le client puisse rafraîchir la liste de musique contenue dans sa tablette.
- que sans connexion au serveur, il soit impossible d'appuyer sur le bouton d'envoi de musique au serveur ou de voir la file de musique du serveur
- que nous avons un affichage où le client puisse choisir son nom lorsqu'il se connecte au serveur.
- que le client ne puisse pas avoir plus de 5 chansons dans la file du serveur
- que le client puisse voir les musiques sur le serveur avec une distinction de couleur de ses musiques du reste de la liste
- que le client ne puisse enlever que ses musiques sur la file de musique du serveur.
- qu'il y a un rafraîchissement périodique de la file de musique du serveur sur le mobile du client.
- que le client puisse afficher les paroles d'une musique contenue sur la file de musique du serveur, lorsque disponible.
- qu'un superviseur puisse se connecter seulement si le nom d'utilisateur ainsi que le mot de passe sont contenus sur le serveur.
- qu'il ne soit pas possible pour un utilisateur d'accéder à une page qui peut être vue seulement par les superviseurs si l'utilisateur en question n'en est pas un.
- qu'il soit possible de visualiser la liste de musique du serveur.
- que les superviseurs puissent enlever n'importe quelle musique de la file du serveur.
- que les superviseurs puissent ajouter ou enlever des clients d'une liste noire
- que les clients faisant partie de la liste noire ne puissent pas se connecter au serveur
- que les superviseurs puissent changer leur mot de passe
- que les superviseurs seulement puissent modifier le niveau de son en sortie
- que l'administrateur puisse se connecter si seulement et seulement si son nom d'utilisateur et son mot de passe correspond à celui sur le serveur
- que l'administrateur puisse voir les statistiques des musiques et les logs du serveur
- que l'administrateur puisse créer ou retirer des superviseurs
- que l'administrateur puisse changer son mot de passe
- qu'il ne soit pas possible de faire une requête au serveur sans que l'information soit cryptée par le protocole de communication HTTPS pour l'administrateur et les superviseurs.
- que le serveur puisse s'amorcer dès le lancement du système d'exploitation
- qu'il y ait une présence de documentation suffisante pour installation des librairies, la configuration et l'installation du système.

5.2 Gestion de risque (Q2.6 et 11.3)

Au cours de la réalisation de ce projet, nous sommes conscients que certaines difficultés peuvent survenir. Nous proposons donc ici une liste non exhaustive de situations problématiques, et comment nous prévoyons pour l'instant de les gérer.

Le premier risque considéré se trouve côté matériel, en cas de panne ou défectuosité d'un composant. Puisque notre système repose sur différentes plateformes matérielles, ce risque est tout de même important. Pour essayer de réduire les répercussions néfastes, nous avons prévu une troisième carte SD que nous comptons laisser de côté et prête à servir en cas de problème avec les deux autres que nous utiliserons couramment au cours de la conception. Similairement, avec la carte FPGA, nous avons la possibilité de travailler sur une autre en cas de soucis récurrents, et de même pour la tablette, nous prévoyons d'autres systèmes Android afin de ne pas bloquer le développement en attendant une réparation ou un remplacement, le cas échéant.

Dans cette suite d'idées, nous utilisons un gestionnaire de version qui conserve notre code à jour sur un entrepôt distant, ce qui nous permet d'éviter une perte ou un problème d'accès à notre code. Le risque de perte ou de problème avec l'entrepôt existant toujours, nous allons effectuer des sauvegardes sur nos disques durs de manière régulière.

Un autre type de risque auquel nous pourrions être confrontés est l'absence prolongée d'un membre de l'équipe. Pour faire face à cette situation, l'utilisation de notre entrepôt distant est déjà un bon élément, que nous complèterons avec la mise à jour régulière entre les membres de l'avancement de chacun, afin de pouvoir être en mesure de poursuivre les travaux de quelqu'un.

En outre, nous voulons prendre en compte le risque associé aux décisions de changements importants dans les spécifications du système, ce qui pourrait entraîner des délais dans la conception. Nous essayerons donc de rendre notre code modulaire et réutilisable, ainsi que de maintenir un contact régulier avec le client, de sorte à pouvoir anticiper au maximum et réagir rapidement en cas de changements.

Un autre risque potentiel est celui de la sécurité. Ici on considère l'accès au wifi suffisant pour que les clients puissent utiliser le système. Afin de conserver la simplicité en priorité, on se contentera de demander aux superviseurs et à l'administrateur de s'identifier. Leurs informations relatives à cette connexion seront simplement stockées sur le serveur, ce qui représente un risque potentiel en termes de sécurité. Il sera possible par la suite de concevoir un système de cryptage pour ces données. Il est tout de même à noter que tous les appareils connectés au serveur fourniront leur adresse IP et MAC, ce qui permettra de les identifier. De plus les échanges entre l'administrateur, les superviseurs et le serveur seront cryptés avec le protocole HTTPS.

5.3 Tests (Q4.4)

- Série de tests du côté du serveur :
    - Requêtes : réception, réponse
    - Réalisation correcte des traitements (gestion de file, conflits, limitations …)
    - Son : vérification des fonctionnalités et du décodage
    - Connexion : vérification des autorisations et permissions client/superviseurs/administrateur

- Tests possibles au niveau de l'application Android:
    - Envoi des requêtes
    - Réception des informations du serveur
    - Gestion des erreurs
    - Interface utilisateur / interface avec les fonctionnalités nécessaires du téléphone (accès liste musique)
    - Gestion authentification
- Tests de l'application PC :
    - Envoi des requêtes
    - Réception et traitement des résultats (statistiques, logs…)
    - Authentification
    - Gestions des comptes superviseurs

5.4 Gestion de configuration

En ce qui concerne la gestion des versions de code, nous disposons d'un répertoire git commun à l'équipe sur lequel tout notre code source sera disponible. La dernière version fonctionnelle du code sera sur la branche principale ou master. Pour des raisons de sécurité, nous aurons trois branches majeures de développement correspondant chacune à une application, à savoir Android, serveur ou PC. De ces dernières, nous dériverons des sous-branches sur lesquelles nous écrirons le code devant répondre à une fonctionnalité précise en relation avec l'application concernée. Nous avons donc établi des règles de fonctionnement git pour l'équipe notamment :

- les noms de branches devront être écrits tout en minuscule ;
- tout comme le code, les messages de commit devront être en anglais ;
- une entête pour les messages des commits afin d'en préciser le but (autres que ceux de développement) : [TEST] [QA] [BUG FIX] ;
- une sous branche par fonctionnalité importante en utilisant des "/" pour séparer les mots. Exemple : server/requests ;
- une revue de code par paires obligatoire avant de faire un push sur une branche de développement ou le master;

Pour exécuter nos tests, nous avons prévu utiliser Boost.Test en ce qui concerne les tests du serveur puisque nous comptions déjà avoir librairie Boost. Pour les tests de l'application Android, une librairie extérieure ne sera pas nécessaire, car le logiciel de développement Android Studio fournit un cadre de tests avec JUnit. Enfin, pour l'application PC, nous aurons recours à JUnit pour exécuter nos tests Java.

En outre, nos fichiers de données seront pour la plupart stockés sur le serveur en fichiers textes. En considérant les contraintes de mémoires et vu que le nombre d'utilisateurs de l'application est très limité, nous ne pensons pas avoir besoin de recourir à une base de données. Nous prévoyons fournir avec chaque livrable, une documentation claire pour préciser les conditions d'utilisation de nos produits, autant à l'intérieur des fichiers sources que dans un "Lisez-moi", document accompagnant l'ensemble.

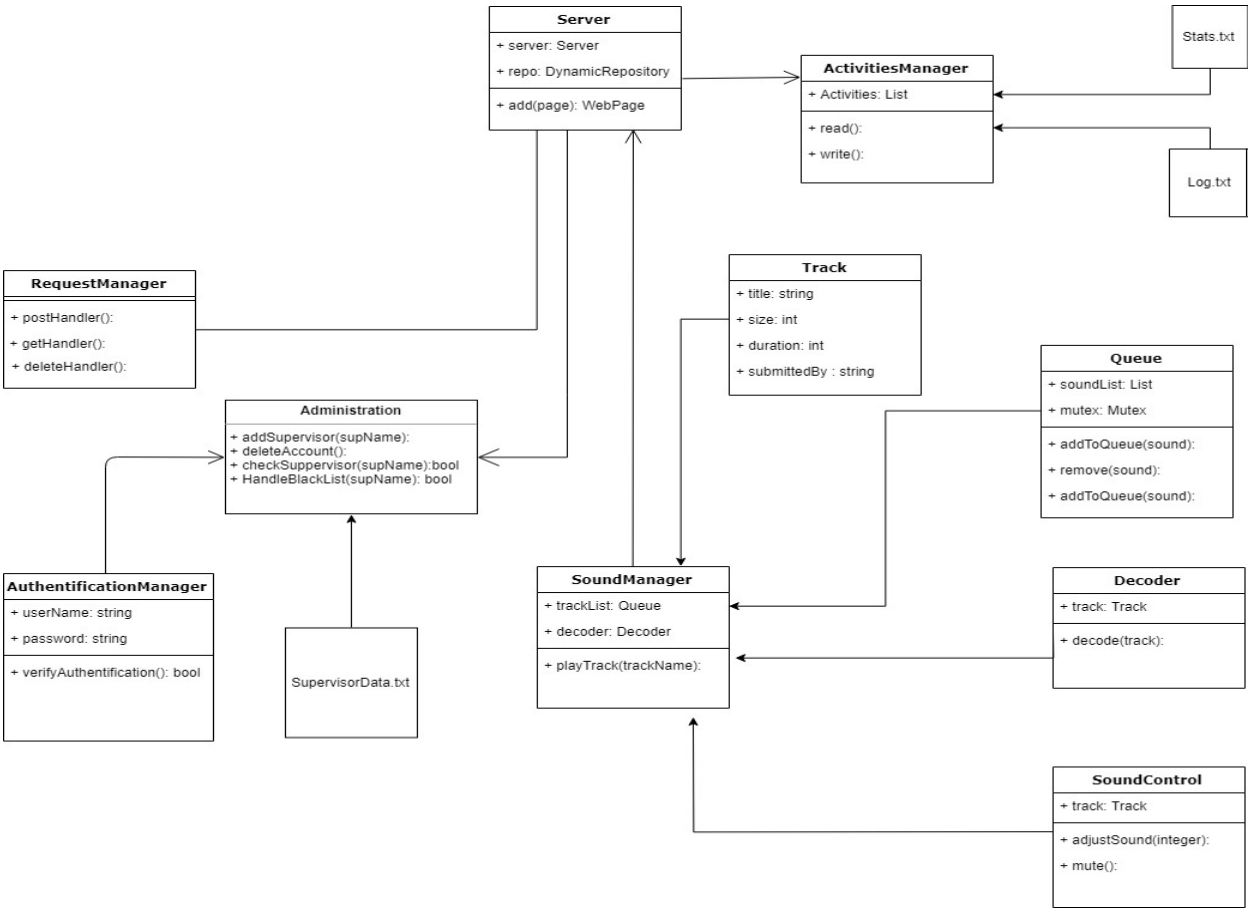
6. Références (Q3.2)

[1]Amazon.com, 2019. [En ligne]. Disponible: https://www.amazon.com/Google-Nexus-Tablet-8-9-Inch-White/dp/B00M6UC5B4?th=1. [Accédé le: 2019-02-01].

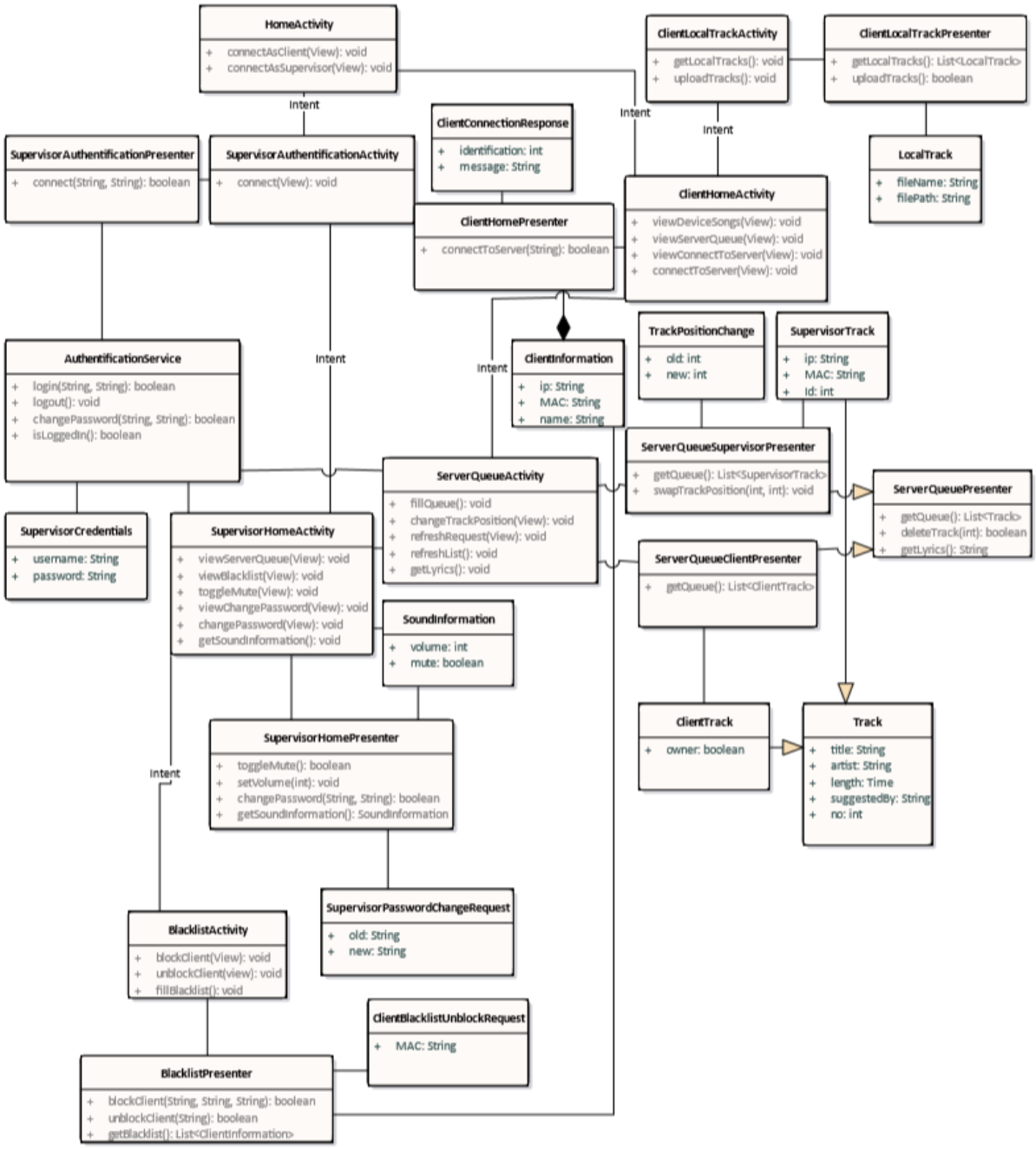
[2] "ZedBoard Zynq-7000 ARM/FPGA SoC Development Board", Digilent, 2019. [En ligne]. Disponible: https://store.digilentinc.com/zedboard-zynq-7000-arm-fpga-soc-development-board/. [Accédé le: 2019-02-01].

ANNEXES

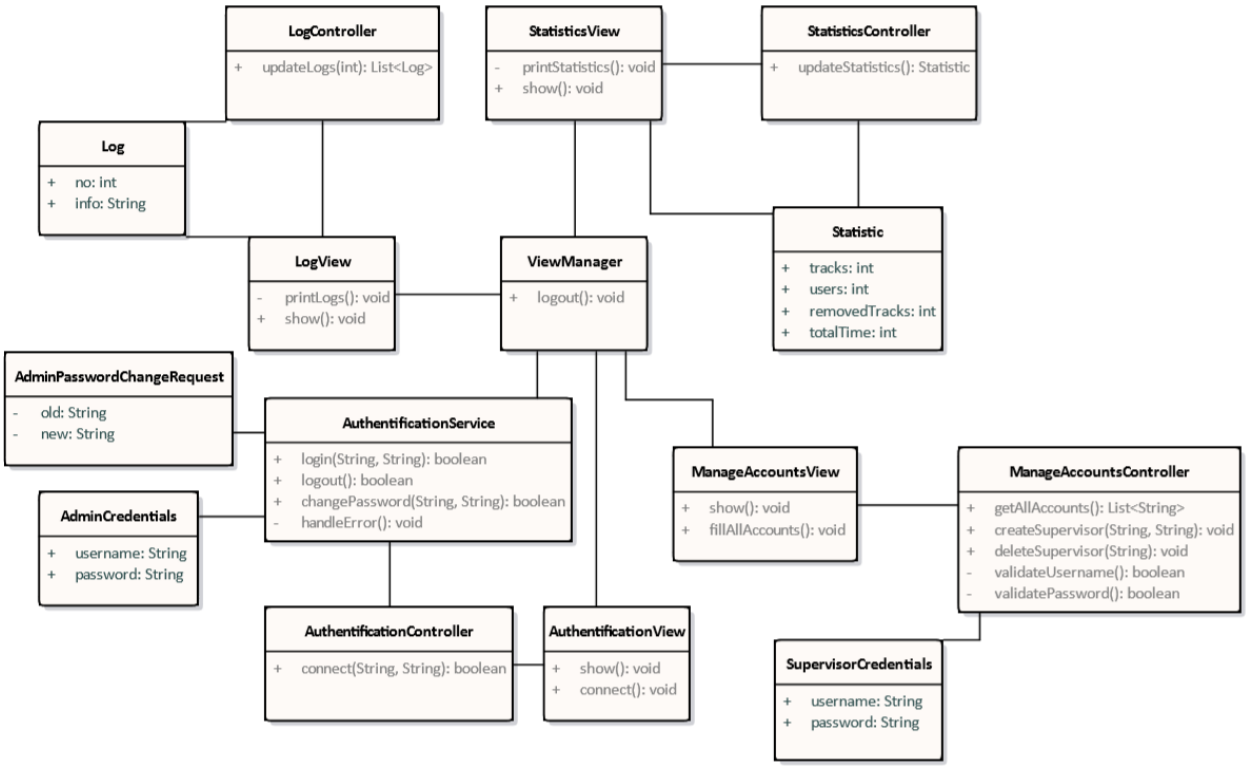
Annexe 1: Diagramme montrant l'architecture du serveur



Annexe 2: Diagramme de classes interagissant sur l'application Android



Annexe 3: Diagramme de classes interagissant sur l'application PC



Annexe 4 : Planification des tâches

Sprint I : Lundi 4 février au Lundi 18 février

Tâche	**Heures**	**Personne**
Application PC	*(24)*	
Interface graphique	15	Anis
Authentification admin	9	Constantin
Serveur	*(55)*	
Accepter connexion client	15	Issifath
Gestion requêtes : Réception	10	Christella
Gestion requêtes : Traitement • venant de l'application Android : 10 • venant de l'application PC : 10	20	Christella, Issifath
Gestion requêtes : Réponse	10	Léandre
Application Android	*(19)*	
Page d'accueil et redirection	2	Léandre
Client : Page d'accueil	3	Léandre
Client : Voir la liste des musiques locales • Affichage liste • Accès au stockage interne	10	Constantin, Léandre
Client : Sélectionner une musique localement	4	Constantin
Total	98	

Sprint II : Lundi 18 février au lundi 11 mars

Tâche	**Heures**	**Personne**
Application PC	*(20)*	
Connexion au serveur	4	Christella
Envoi de requêtes et réception réponse	10	Constantin

Chiffrement des échanges (SSL)	6	Anis
Serveur	*(72)*	
Gestion file d'attente : Ajout et retrait de musiques	20	Christella
Décodage de fichiers MP3 • décodage des entêtes : 12 • exploitation des informations : 13	25	Christella, Issifath
Lecture fichiers audio	15	Issifath
Gestion authentification	12	Léandre
Application Android	*(40)*	
Connexion au serveur • Établissement de la connexion • Interface d'envoi des requêtes	10	Issifath, Anis
Voir la file de musique du serveur • Recevoir la liste • Afficher la liste	20	Léandre, Constantin
Client : Affichage des paroles	10	Anis
Total	132	

Sprint III : Lundi 11 mars au lundi 25 mars

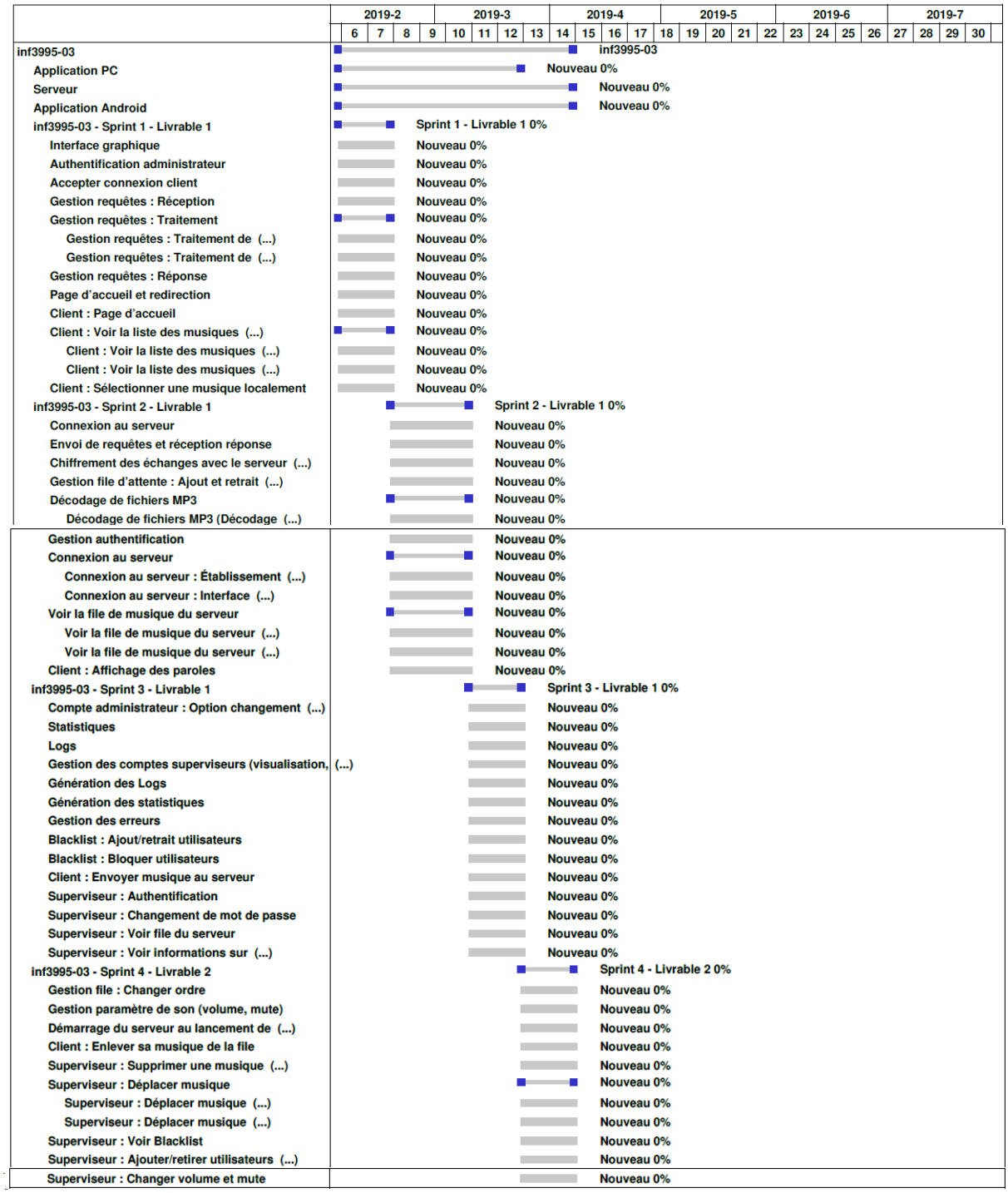
Tâche	**Heures**	**Personne**
Application PC	*(26)*	
Compte administrateur : Option changement de mot de passe	4	Anis
Statistiques	6	Anis
Logs	6	Christella
Gestion des comptes superviseurs (visualisation, ajout, retrait)	10	Constantin
Serveur	*(41)*	
Génération des Logs	5	Christella
Génération des statistiques	8	Christella
Gestion des erreurs	8	Issifath

Blacklist : Ajout/retrait utilisateurs	10	Léandre
Blacklist : Bloquer utilisateurs	10	Issifath
Application Android	*(33)*	
Client : Envoyer musique au serveur	12	Constantin
Superviseur : Authentification	2	Anis
Superviseur : Changement de mot de passe	8	Anis
Superviseur : Voir file du serveur	2	Issifath
Superviseur : Voir informations sur musique	9	Léandre
Total	100	

Sprint IV : Lundi 25 mars au lundi 8 avril

Tâche	**Heures**	**Personne**
Application PC	*()*	
-	-	-
Serveur	*(26)*	
Gestion file : Changer ordre	12	Christella
Gestion paramètre de son (volume, mute)	10	Issifath
Démarrage du serveur au lancement de Archlinux	4	Christella
Application Android	*(48)*	
Client : Enlever sa musique de la file	9	Anis
Superviseur : Supprimer une musique de la file	4	Constantin
Superviseur : Déplacer musique • Réalisation graphique : 10 • implémentation de la logique de déplacement : 6	16	Léandre, Constantin
Superviseur : Voir Blacklist	5	Issifath
Superviseur : Ajouter/retirer utilisateurs de la Blacklist	8	Léandre
Superviseur : Changer volume et mute	6	Anis
Total	74	

Annexe 5 : Diagramme de Gantt



Annexe 6 : Roadmap

Sprint 1 - Livrable 1

Échéance dans 14 jours (18/02/2019)

0%

14 demandes (0 fermée — 14 ouvertes)

Demandes liées

- Fonctionnalités #2339: Interface graphique
- Fonctionnalités #2340: Authentification administrateur
- Fonctionnalités #2341: Accepter connexion client
- Fonctionnalités #2342: Gestion requêtes : Réception
- Fonctionnalités #2343: Gestion requêtes : Traitement
- Fonctionnalités #2345: Gestion requêtes : Réponse
- Fonctionnalités #2346: Page d'accueil et redirection
- Fonctionnalités #2347: Client : Page d'accueil
- Fonctionnalités #2348: Client : Voir la liste des musiques locales
- Fonctionnalités #2349: Client : Sélectionner une musique localement
- Fonctionnalités #2353: Gestion requêtes : Traitement de celles venant de l'application Android
- Fonctionnalités #2354: Gestion requêtes : Traitement de celles venant de l'application PC
- Fonctionnalités #2355: Client : Voir la liste des musiques locales (Affichage)
- Fonctionnalités #2356: Client : Voir la liste des musiques locales (Accès au stockage interne)

Sprint 2 - Livrable 1

Échéance dans 35 jours (11/03/2019)

0%

16 demandes (0 fermée — 16 ouvertes)

Demandes liées

- Fonctionnalités #2350: Connexion au serveur
- Fonctionnalités #2351: Envoi de requêtes et réception réponse
- Fonctionnalités #2357: Chiffrement des échanges avec le serveur (SSL)
- Fonctionnalités #2358: Gestion file d'attente : Ajout et retrait de musiques
- Fonctionnalités #2359: Décodage de fichiers MP3
- Fonctionnalités #2360: Décodage de fichiers MP3 (Décodage des entêtes)
- Fonctionnalités #2361: Décodage de fichiers MP3 (Utilisation de l'informations des entêtes)
- Fonctionnalités #2362: Lecture de fichiers audio
- Fonctionnalités #2363: Gestion authentification
- Fonctionnalités #2364: Connexion au serveur
- Fonctionnalités #2365: Connexion au serveur : Établissement de la connexion
- Fonctionnalités #2366: Connexion au serveur : Interface d'envoi des requêtes
- Fonctionnalités #2367: Voir la file de musique du serveur
- Fonctionnalités #2368: Voir la file de musique du serveur : Recevoir la liste
- Fonctionnalités #2369: Voir la file de musique du serveur : Afficher la liste
- Fonctionnalités #2371: Client : Affichage des paroles

Sprint 3 - Livrable 1

Échéance dans 49 jours (25/03/2019)

0%

14 demandes (0 fermée — 14 ouvertes)

Demandes liées

- Fonctionnalités #2372: Compte administrateur : Option changement de mot de passe
- Fonctionnalités #2373: Statistiques
- Fonctionnalités #2374: Logs
- Fonctionnalités #2375: Gestion des comptes superviseurs (visualisation, ajout, retrait)
- Fonctionnalités #2376: Génération des Logs
- Fonctionnalités #2377: Génération des statistiques
- Fonctionnalités #2378: Gestion des erreurs
- Fonctionnalités #2379: Blacklist : Ajout/retrait utilisateurs
- Fonctionnalités #2380: Blacklist : Bloquer utilisateurs
- Fonctionnalités #2381: Client : Envoyer musique au serveur
- Fonctionnalités #2382: Superviseur : Authentification
- Fonctionnalités #2383: Superviseur : Changement de mot de passe
- Fonctionnalités #2384: Superviseur : Voir file du serveur
- Fonctionnalités #2385: Superviseur : Voir informations sur musique

Sprint 4 - Livrable 2

Échéance dans environ 2 mois (08/04/2019)

0%

11 demandes (0 fermée — 11 ouvertes)

Demandes liées

- Fonctionnalités #2386: Gestion file : Changer ordre
- Fonctionnalités #2388: Gestion paramètre de son (volume, mute)
- Fonctionnalités #2389: Démarrage du serveur au lancement de Archlinux
- Fonctionnalités #2390: Client : Enlever sa musique de la file
- Fonctionnalités #2391: Superviseur : Supprimer une musique de la file
- Fonctionnalités #2392: Superviseur : Déplacer musique
- Fonctionnalités #2399: Superviseur : Déplacer musique (Réalisation graphique)
- Fonctionnalités #2400: Superviseur : Déplacer musique (Implémentation de la logique de déplacement)
- Fonctionnalités #2408: Superviseur : Voir Blacklist
- Fonctionnalités #2410: Superviseur : Ajouter/retirer utilisateurs de la Blacklist
- Fonctionnalités #2411: Superviseur : Changer volume et mute