POLYTECHNIQUE MONTRÉAL

LE GÉNIE EN PREMIÈRE CLASSE

Département de génie informatique et génie logiciel

INF3995

Projet de conception d'un système informatique

Proposition répondant à l'appel d'offres
no. A2018-INF3995 du département GIGL.

Système audio pour café Internet

Équipe No < ***03*** >

Soukaina Moussaoui
Ibrahima-Sega Sangare
Wassim Guellati
Benjamin Heinen

Septembre 2018

Table des matières

1. Vue d'ensemble du projet

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

Le but de ce projet consiste à concevoir et développer une solution technique adéquate à l'appel d'offre proposé par le propriétaire du Café-Bistro Élévation Inc. dans un projet orienté vers le divertissement musical. Les membres de l'équipe en charge auront donc pour objectifs de pouvoir développer cette solution pendant le trimestre d'automne en s'appuyant sur des concepts clés introduits en projet intégrateur. Parmi ceux-ci, la gestion de projet représente une notion centrale de ce projet, car la planification ne sera contrainte que par la remise de deux livrables en termes d'échéancier et sans vraiment considérer les contraintes techniques, pour l'instant. Un autre objectif de ce projet sera d'acquérir et approfondir les connaissances sur des technologies nouvelles et familières pour être en mesure d'implémenter les fonctionnalités demandées par le client. Ce projet aura aussi pour but de professionnaliser le travail qui sera effectué ainsi que le produit final de la collaboration. En plaçant le projet dans un contexte d'appel d'offre, les exigences concernant la documentation du travail, l'organisation du projet, l'établissement d'un calendrier, l'estimation de coûts ainsi que les autres aspects implicites au contexte, auront pour but de former les étudiant sur le processus de développement dans le monde du travail.

En ce qui concerne les biens livrables attendus, le client désire avoir deux livrables constituant, dans un premier temps, une version partielle du produit avec des quelques fonctionnalités, et dans un deuxième temps, un autre livrable qui sera accompagné de la version finale du produit avec toutes les fonctionnalités implémentées. Pour ce qui est du produit lui-même, le client, propriétaire de café-bistro, souhaite être en mesure de mettre à disposition de sa clientèle une application Android qui soit en mesure de construire une liste de lecture basée sur les propositions des utilisateurs dans le but de rendre l'ambiance du café-bistro plus agréable. Ces chansons sont acheminées aux haut-parleurs dans le bâtiment pour le divertissement collectif. Entre autres, les utilisateurs pourront soumettre ou retirer leurs propositions de chansons par l'intermédiaire d'une file d'attente. Celle-ci sera sous la surveillance d'un administrateur qui pourra choisir de manipuler l'ordre des chansons, supprimer des propositions ou encore fournir ajuster le son. Concrètement, les livrables seront récupérés à travers l'outil de gestion de version Git et contiendront tous les fichiers et code sources pertinentes provenant du serveur et de l'application Android, sans oublier le rapport technique du projet.

1.2 Hypothèse et contraintes (Q3.1)

On peut compter plusieurs hypothèses et contraintes liées à ce projet. D'abord, il est établi que les étudiants seront en mesure de développer une solution satisfaisante aux yeux du client en dépit de leurs possibles lacunes et niveau de formation actuel. Ils seront donc face à une situation particulièrement nouvelle dont la réussite dépend en grande majorité de leur capacité d'apprentissage et d'adaptation. Les caractéristiques de la solution demandée par le client doivent être suffisamment précises et dépourvues d'ambiguïtés. Ces spécifications sont rassemblées dans un document en bonne et due forme pour éviter les complications par la suite. Le temps accordé à l'équipe pour remplir son contrat a été jugé correct. Les étudiant vont s'organiser pour travailler en équipe en dehors des heures de cours et laboratoires pour répondre aux attentes en termes de nombres d'heures nécessaires.

En ce qui concerne les contraintes, elles sont nombreuses. Le travail lié à l'implémentation du serveur est contraint à l'utilisation du langage C++ et une carte SD insérée dans une carte FPGA qui simulera l'environnement Linux (Arch Linux). L'application Android doit être développé en Kotlin ou Java et compatible avec les tablettes possédant Android 6 ou plus. L'utilisation d'une interface REST et du protocole HTTP est imposée pour la communication client-serveur. Le client impose de légères contraintes quant à l'esthétique l'application Android (thèmes et couleurs par exemple). Il est attendu d'avoir de mode différent pour le contrôle de l'application (utilisateur et administrateur) et chacun d'entre eux doit avoir des limites sur les actions qui peuvent être réalisées. Le produit doit naturellement assurer la présence de plusieurs utilisateurs simultanément, et ce, sans avoir à s'enregistrer (uniquement pour les utilisateurs lambda). La possibilité d'utiliser des adresses IP différentes pour la connexion au serveur doit être supportée. La communication client-serveur doit supporter le transfert de plusieurs types de fichiers comme JSON et MP3. Le décodage de fichiers MP3 est fondamental dans le projet. La validation de logiciel supplémentaire est laissée à la discrétion du promoteur.

1.3 Biens livrables du projet (Q2.3)

Le projet sera réparti en deux livrables selon les directives du café-bistro Élévation Inc. En premier lieu, le premier livrable fonctionnel devra être prêt pour évaluation le 13 novembre 2018. D'une part, le serveur doit être en mesure d'assurer le décodage complet du son sans tenir compte des ajustements liés au volume. De plus, la file de chansons doit être présente sans pouvoir être capable de retirer des chansons ou en inverser l'ordre. D'autre part, l'application Android doit contenir le mode usager ordinaire avec toutes ces fonctionnalités, hormis le retrait de chansons. Le mode administrateur ne fera pas partie de ce livrable.

En dernier lieu, le second livrable sera évalué le 29 novembre 2018 et marquera la fin du projet. Du côté du serveur, la liste noire contenant les adresses IP ou

MAC des utilisateurs bloqués sera ajoutée. Les fonctionnalités manquantes telles que les retraits et inversions et ajustement de volume viendront compléter le serveur. Du côté de l'application, le mode administrateur sera complètement implémenté avec tous ses rôles qui sont décrits plus en détails dans les spécifications du projet.

2. Organisation du projet

2.1 Structure d'organisation

La structure organisationnelle que nous avons choisie est de type décentralisé permettant aux membres de l'équipe d'être autonomes et responsables de l'ensemble des tâches à effectuer. D'un point de vue technique, étant donné que le nombre de membres est assez restreint par rapport à la complexité du projet, nous avons décidé que chacun devait avoir plus d'une fonction. Le tableau ci-dessous présente les rôles de chaque membre en fonction des deux architectures définies : client et serveur ainsi que les rôles concernant la gestion de projet.

Tableau I : Rôles et fonctions

	Benjamin Heinen	*Ibrahima Séga*	*Soukaina Moussaoui*	*Wassim Guellati*
Architecture serveur				
Développement	**x**	**x**	**x**	**x**
Vérification assurance qualité	**X**		**x**	
Vérification tests		**x**		**x**
Architecture client				
Développement	**x**	**x**	**x**	**x**
Vérification assurance qualité		**x**		**x**
Vérification tests	**X**		**x**	
Gestion				
Product manager		**x**	**x**	
Scrum master				**x**

2.2 Entente contractuelle

Une entente contractuelle présente l'ensemble des responsabilités et les engagements d'une équipe pour un projet donné. Pour donner suite à l'appel d'offre A2018-INF3995 du département GIGL, notre équipe propose un contrat de type clé en main. En effet, ce contrat engage l'équipe à livrer un projet prêt à être utilisé une fois finalisé, et au client d'effectuer le paiement total une fois le produit final est accepté.

Nous avons fait le choix de ce type d'entente étant donné que toutes les spécifications concernant la qualité du projet à livrer sont établies de manière très détaillée et vérifiable. Ceci prouve que le client détient une connaissance rigoureuse de la demande. Par ailleurs, notre équipe est consciente que les spécifications peuvent être sujettes à changements après négociation. Finalement, le projet requiert un faible niveau de suivi par le promoteur. En l'occurrence, nous considérons que ce le contrat clé en main s'adapte le mieux à ce type de projet.

3. Solution proposée

3.1 Architecture logicielle sur serveur (Q4.2)

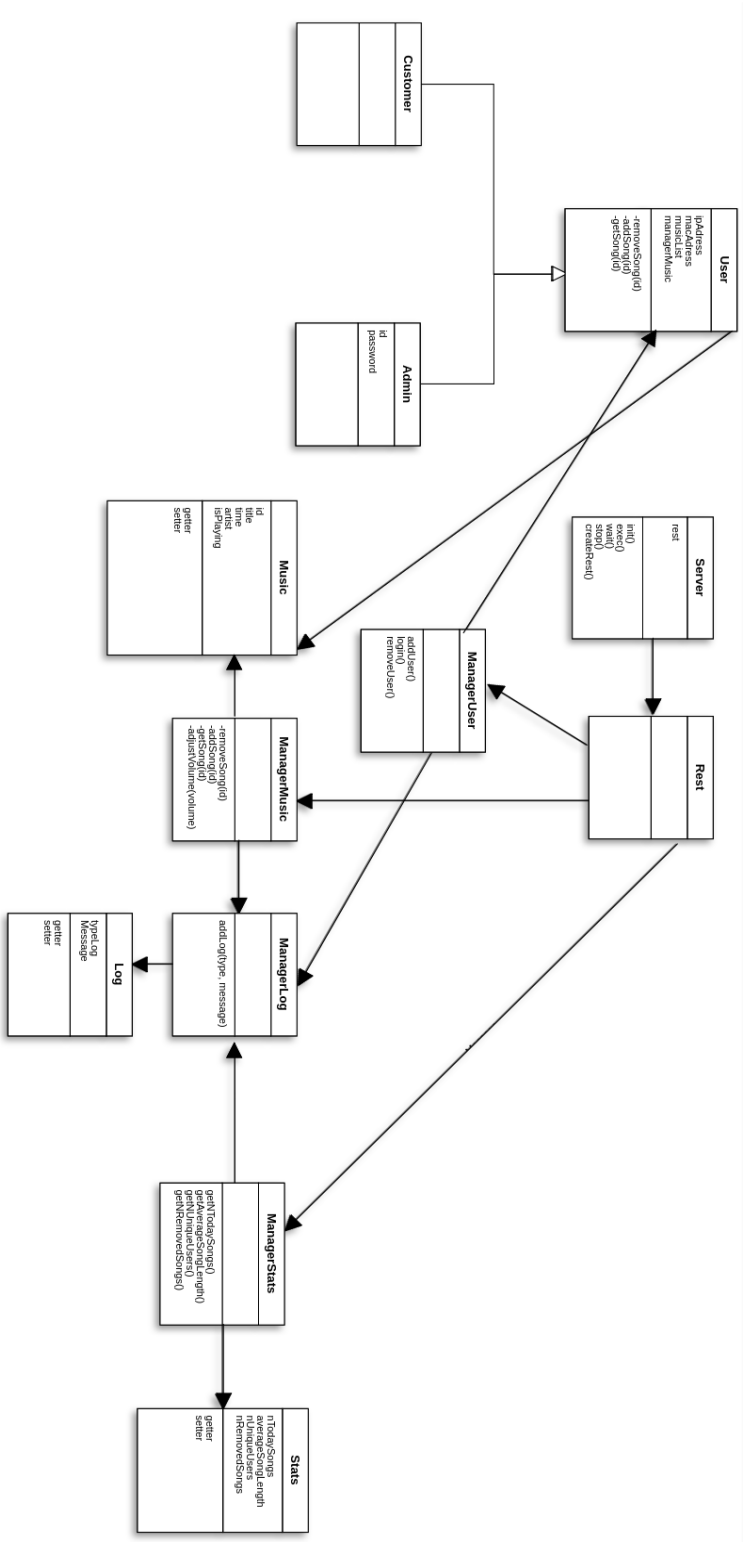

Figure 1 : Diagramme de classe du serveur

Comme vous pouvez le constatez dans la figure 1, « Diagramme de classe du serveur » qui se trouve ci-dessus, nous avons déjà préparé tout ce qui était lié à la conception de serveur.
Ainsi, on aura une classe permettant de gérer les interactions du serveur. Pour se faire, il appellera une classe chargée de gérer les différents appels REST. C'est celle-ci qui sera chargé d'effectuer les différentes redirections vers les managers correspondant au type de requête exécuté. Par exemple, je si veux bannir un utilisateur, la classe REST appellera la classe ManagerUser. C'est elle qui se chargera alors de bannir l'utilisateur sélectionner.

3.2 Architecture logicielle sur tablette (Q4.3)

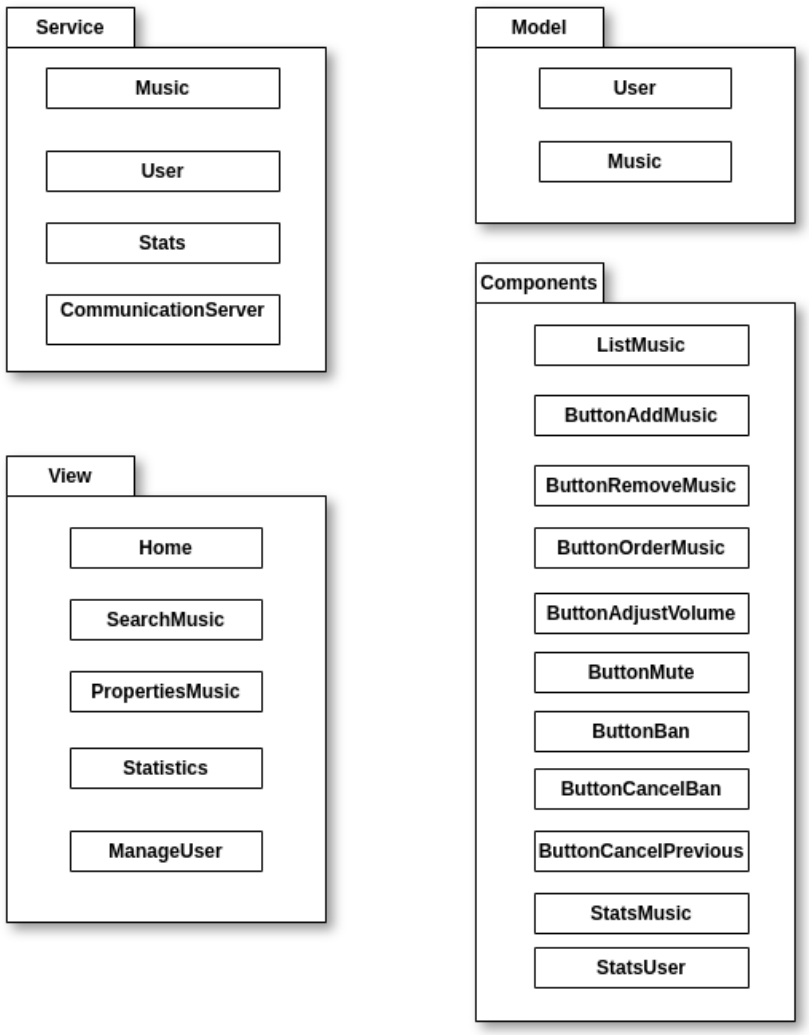

Figure 2 : Diagramme de packages de l'application Android

Dans la figure 2 « Diagramme de packages de l'application Android » que vous pouvez trouver ci-dessus, on vous explique les différents modules dont sera composé l'application Android.
Elle sera divisée selon le modèle MVC (Model View Controller) ». Il s'agit d'un modèle particulièrement bien connu par les ingénieurs, qui n'a plus à prouver son utilité au sein des applications.
Ainsi, on retrouve un package « View », qui contient les différentes pages que contiendra l'application. Ce package sera soutenu par le package « Components » qui nous permettra de dessiner et de gérer les interactions des différents composants graphique de l'application, tel que les boutons.
Par la suite, le package « Model » permettra de contenir les informations liées aux utilisateurs ou aux musiques.
Enfin, le package « Service » permettra de gérer les différentes actions que souhaitent exécuter les utilisateurs. Par exemple, lorsqu'un l'administrateur clique sur le bouton permettant de bannir un utilisateur, le components « ButtonBan » appellerai la classe du package service « User ».

4. Processus de gestion

4.1 Estimations des coûts du projet :

Notre équipe de projet est constituée de quatre membres. Trois de ces membres occuperont les fonctions d'un développeur-analyste et le dernier membre assumera les fonctions d'un coordonnateur de projet.

Après avoir effectué le diagramme de Gantt, nous sommes parvenus à une estimation de la charge de travail de ce projet intégrateur. Cette dernière est équivalente à 218 heures. Il a donc été convenu, en conséquence de cette estimation, d'accorder en moyenne un nombre de 3 heures et demi par jour aux différentes tâches du calendrier, ceci dans le cas où tout se présenterait bien et que les conditions resteraient favorable au cours du projet intégrateur. Dans la mesure où d'éventuels problèmes devaient survenir, nous avons prévu d'accorder une heure supplémentaire chaque jour, ce qui équivaut à un total de 66 heures supplémentaires. C'est l'équivalent du temps nécessaires à réaliser 5 tâches des plus complexes du projet intégrateur, en entier.
Vous pouvez retrouver les détails à propos des coûts du projet dans le tableau 2 « Calcul du coût estimé et du coût plafond pour le projet » qui se trouve ci-dessous.

Tableau II : Calcul du coût estimé et du coût plafond pour le projet

Ressources	Coût estimé	Coût Plafond
Coordonateur de projet (145$/heure)	1	1
Développeur-analyste (130$/heure)	3	3
Heures estimée (meilleur cas)	218	X
Heures estimées (pire cas)	X	264
Total (en $)	116630	141240

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

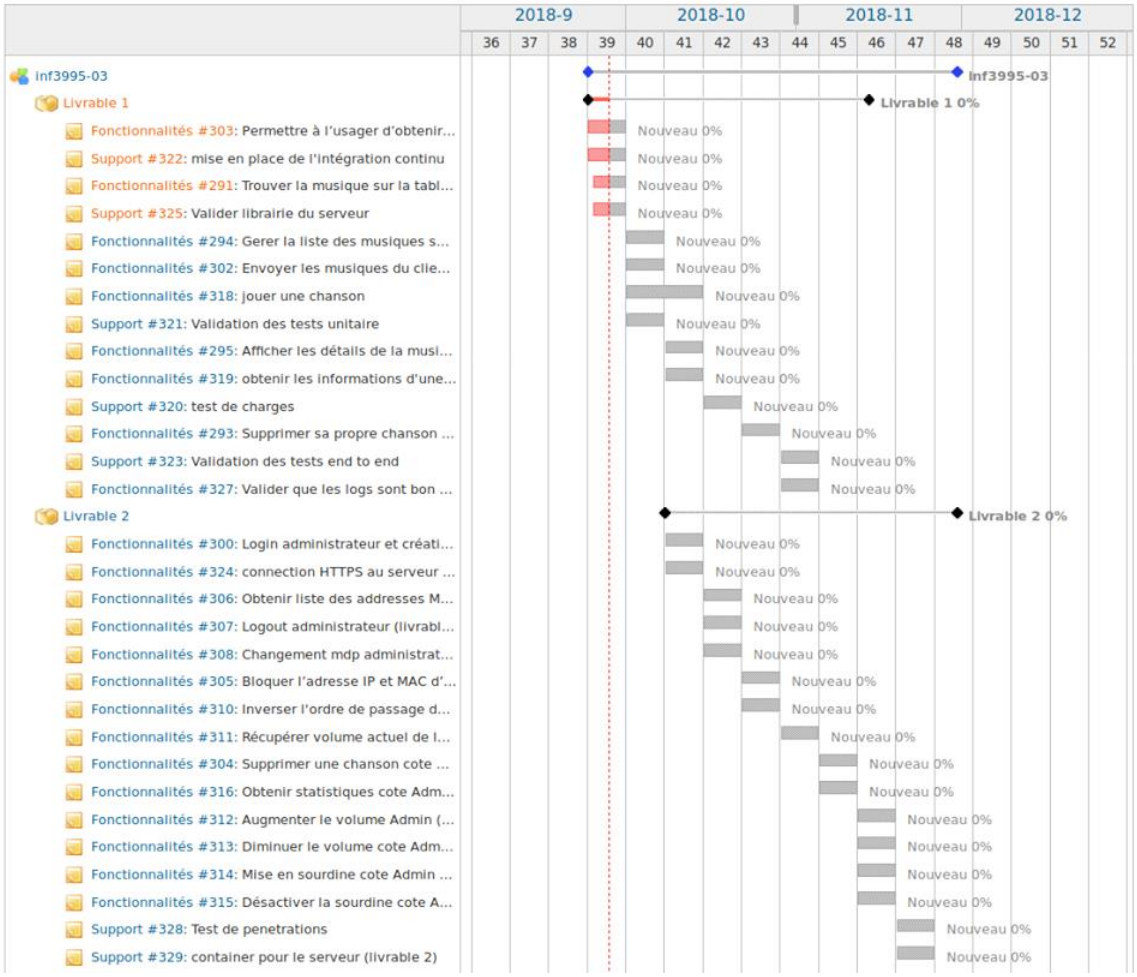

Figure 3 : Diagramme de Gantt

Dans le figure 3 « Diagramme de Gantt » qui se trouve ci-dessus, nous vous présentons comment nous comptons répartir les tâches de ce projet. Comme vous pouvez le constater, on commence certaines tâches du deuxième livrable avant de finir le premier livrable. La raison de ce choix est purement stratégique. Nous estimons que le deuxième livrable nous demandera plus de temps de travail que le temps dont l'on disposerait en commençant après le premier livrable, puisque l'on ne disposera que de deux semaines. Ainsi, en faisant ce choix, nous nous assurons de ne pas prendre de retard. De plus, toujours dans l'optique de ne pas prendre de retard, nous disposons d'une semaine vide avant la remise des chacun des livrables. Cela nous permettra de vérifier que chaque élément du livrable en cours correspond bien à vos attentes.

4.3 Calendrier de projet (Q3.3)

Afin de faciliter le suivi du projet, que ce soit pour nous ou pour vous, nous avons effectué un calendrier des différentes phases de développement permettant d'assurer facilement le suivi du projet. Vous pouvez le retrouvez dans le tableau 3 « Calendrier prévu pour le projet « qui se trouve ci-dessous.

Tableau III : Calendrier prévu pour le projet

Phases de développement	Date ciblée
Serveur : décodage du son 1ere version conforme au livrable 1.	14 Octobre 2018
Application : mode usager ordinaire 1ere version conforme au livrable 1.	28 Octobre 2018
Serveur : gestion de la file 1ere version conforme au livrable 1.	28 Octobre 2018
Livrable 1 : remise du système avec fonctionnement partiel.	13 Novembre 2018
Serveur : gestion de la liste noire.	28 Octobre 2018
Serveur : modification de la file (retraits, inversions).	28 Octobre 2018
Application : mode usager ordinaire (retrait des chansons opérationnel).	11 Novembre 2018
Serveur : ajustements du volume et de la sourdine.	18 Novembre 2018
Application : mode superviseur opérationnel.	18 Novembre 2018
Le livrable final complet avec toutes les fonctionnalités du système.	29 Novembre 2018

4.4 Ressources humaines du projet

Le nombre de ressources humaines nécessaires à la réalisation du projet est de 4 personnes possédant chacune les compétences de développeur analyste, dont une personne possédant également les compétences de coordonnateur de projet.

Voici les qualifications spéciales et expériences nécessaires aux membres de l'équipe :

Aptitudes personnelles et relationnelles :

- Posséder un sens de l'analyse et de l'organisation.
- Posséder un sens de la communication en équipe.
- Posséder le sens du travail en équipe.
- Sens de la pédagogie dans la gestion d'une équipe.

Aptitudes professionnelles :

- Maîtrise de la programmation orienté objet Java.
- Maîtrise du langage de programmation C/C++.
- Connaissances en SDK.
- Expériences avec le système d'exploitation Android.
- Connaissances en design d'applications mobiles.
- Connaissances en architecture logicielle.
- Connaissances en Web service REST.
- Connaissances en HTTP.
- Maîtrise des données du format XML, JSON.

5. Suivi de projet et contrôle

5.1 Contrôle de la qualité

Il est très important pour nous de fournir un livrable dont la qualité vous semblera irréprochable. Pour ce faire, il nous semble indispensable de mettre en place un processus de révision rigoureux, avec une liste de points à vérifier claire.

Voici la liste des points qui semblent particulièrement important à vérifier :

- Maintenabilité du code
- Valider le fonctionnement de chaque élément de la tablette.
- Valider que le serveur réagisse correctement.
- Qualité visuelle de l'application Android et ergonomie.

La maintenabilité du code est un élément particulièrement important à nos yeux. Il se peut que dans le futur, vous souhaitiez effectuer certaines améliorations sur le projet. Par exemple, vous pourriez vouloir que vos clients puissent regarder des séries sur votre télévision. Si le code est maintenable, il sera très aisé pour les futurs ingénieurs de rajouter cette fonctionnalité. Par conséquent, ce point peut vous permettre d'économiser sur des coûts futurs. De plus, comme il est

plus rapide de développer avec un code maintenable, cela devrait vous assurer que l'on vous remettra les livrables dans les temps.

Notre équipe a à cœur de vous remettre un livrable complet, qui satisfera vos attentes. Nous nous sentons donc dans l'obligation de vous remettre un produit qui fonctionne complètement, que ce soit sur la tablette Android ou bien sur le serveur.

Enfin, il est indispensable aujourd'hui d'avoir une application qui soit visuellement bien conçu. C'est un point indispensable pour les utilisateurs de l'application. En plus de cela, il est important que l'application soit le plus ergonomique possible, afin de faciliter la prise en main des utilisateurs.

Afin de mettre tout cela en place, nous utiliserons plusieurs méthodes qui sont réputés pour leur apport au contrôle de la qualité d'un projet.

Tout d'abord, chaque membre de l'équipe devra approuver qu'une tâche est bien rempli. S'il l'un des membres pense qu'une tâche peut être amélioré, il pourra expliquer comment. Cela peut autant être par rapport à la qualité du code que par rapport aux fonctionnalités. En outre, cela permet de gagner en flexibilité. En effet, en agissant de la sorte, chaque membre de l'équipe pourra travailler sur n'importe quelle fonctionnalité. Cela permet également de réduire significativement les délais de livraisons.

Le bon fonctionnement de l'application et du serveur sera validé par des tests. Afin de s'assurer que les tests fonctionnent continuellement, nous utiliserons des outils spécifiques au test d'intégration, tel que Circle CI.

Enfin, pour s'assurer de l'ergonomie et de la qualité visuelle de l'application, nous définirons, sur papier, l'apparence et le positionnement des différents éléments visuel de l'application. Si l'on pense qu'il peut y avoir une amélioration à faire intéressante par rapport à la maquette construite initialement, nous prendrons le temps de faire une réunion, afin de considérer les avantages et les inconvénients de cette modification. En utilisant cette méthode, nous nous assurons d'avoir un ensemble graphique cohérent.

5.2 Gestion de risque (Q2.6 et 11.3)

Les projets informatiques encourent généralement de nombreux risque. Voici les différents risques qui risquent d'apparaître pour ce produit, ainsi que les dispositions que nous emploierons afin d'éviter qu'ils ne nuisent au projet.

Tout d'abord, l'application Android, tout comme le serveur, sont susceptible de subir une perte de qualité au niveau de la maintenabilité de son code source. En effet, la réalisation d'un code dit propre peut parfois s'apparenter à un défi de

taille. Afin de résoudre celui-là, nous nous engageons à utiliser des patrons de conceptions connus et approuvés par de nombreux développeur.

De plus, il se peut qu'il y ait une fonctionnalité manquante lors d'un livrable. Bien que nous espérons fortement que cela n'arrive pas, il arrive parfois que les ingénieurs ne comprennent pas la volonté du client vis-à-vis d'une fonctionnalité. Afin d'éviter cela, nous communiquerons avec vous à chaque fois que l'on aura une hésitation à propos de ce que vous souhaitez réellement obtenir comme résultat.

Il arrive parfois également que le délai prévu pour la réalisation du produit ne soit pas assez long. Cela peut engendrer une augmentation des coûts. Afin de limiter les risques, nous effectuerons une réunion hebdomadaire afin de valider l'avancer de chacun des ingénieurs travaillant sur le projet. De plus, les ingénieurs devront effectuer une démonstration des fonctionnalités développés à chaque semaine. En agissant de la sorte, nous pensons limiter fortement les risques de nous écarter des délais convenus. Dans le cas où nous nous rendons comptes que l'avancée du projet est tout de même plus lente que ce qui était prévu au départ, nous vous en avertirons.

Il est très important pour nous que les utilisateurs de nos produisent aient une bonne expérience en tant qu'usager. Néanmoins, on encourt toujours le risque que l'interface graphique ne satisfasse pas les clients. Afin d'éviter que cela ne se produise, nous ferons extrêmement attention à l'expérience que pourront avoir les usagers. Pour ce faire, nous pensons tester régulièrement l'ergonomie de l'application. De plus, nous pourrons demander à des testeurs n'ayant jamais utilisé l'application de nous donner une opinion vis-à-vis de celle-ci, ou de certains modules.

Enfin, la sécurité des applications informatiques est toujours un facteur à risque important. Néanmoins, la majorité de nos ingénieurs ont suivi une formation en informatique, ce qui leur permet d'avoir conscience des différents risquent, ainsi que des solutions à employer afin de les résoudre. Ils feront donc tout leur possible afin de vous proposer une structure limitant les risques de vulnérabilités.

5.3 Tests (Q4.4)

Afin de s'assurer de la qualité du produit que nous vous proposerons, nous le soumettrons à de nombreux tests.

Tout d'abord, nous utiliserons de nombreux tests unitaires afin de nous assurer que les fonctionnalités aient toutes le comportement souhaité. Ainsi, ces tests seront faits aussi sur l'application Android que sur le serveur. Néanmoins, ils seront nettement plus présents sur le serveur. En effet, celui-ci a pour

responsabilité de traiter la majorité des données, c'est pourquoi notre équipe sera particulièrement rigoureuse sur ses tests unitaires.

Par la suite, nous effectuerons des tests end-to-end. Il s'agit de tester l'interface graphique de l'application Android. Ainsi, nous pourrons valider que chacun des composants graphiques disposent du comportement adéquat.

Comme nous le précisions dans la partie précédente, il y a aujourd'hui de nombreuses failles de sécurités dans les applications. Afin d'éviter que ses failles ne s'introduisent dans nos produits, nous effectuons nous même de nombreux tests de pénétration. Ils permettent d'essayer d'accéder à des données ou à des fonctionnalités auquel on ne devrait pas pouvoir disposer, ou bien de parvenir à changer le comportement prévu d'une fonctionnalité. En effectuant ses tests, nous diminuons fortement les risques que quelqu'un de mentionner puisse vous nuire par l'intermédiaire de notre produit. Cela nous semble d'autant plus important dans votre situation puisque l'application va interagir avec des personnes autres que les usagers.

Étant donné que plusieurs personnes pourront interagir avec le serveur en simultanée, il nous semble vital d'intégrer un test de charge. Il permettra de vérifier que le serveur soit capable de fonctionner correctement malgré un nombre conséquent d'utilisateurs.

Enfin, il nous semble vital d'effectuer des tests d'intégrations. Ceux-ci nous permettent de valider que lorsque nos ingénieurs travaillent sur une fonctionnalité, ils ne risquent pas de nuire à une autre fonctionnalité déjà développé. Ainsi, on limite également les risques de sortir des délais que nous nous sommes fixés au début du projet.

5.4 Gestion de configuration

Afin de nous assurer de démarrer nos projets sur de bonnes bases, nous utilisons Git comme système de contrôle des versions. Il s'agit d'un outil qui permet de gérer efficacement l'évolution d'un projet informatique dans le temps. De plus, afin d'éviter toute utilisation chaotique de cet outil, nous utilisons quotidiennement un modèle d'utilisation de cet outil s'appelant GitFlow. Il nous permet de savoir exactement le type de tâche effectué par les ingénieurs, ainsi que comment celle-ci influe sur le projet.

De plus, l'organisation du code source est particulièrement important pour les développeurs. C'est un facteur qui influe fortement sur la vitesse de développement ainsi que sur la facilité à maintenir le code dans le futur.
Le serveur sera développé en C++. C'est pourquoi nous utiliserons la norme ISO/IEC 14882:2017. Cette norme est la dernière qui est sortie pour le C++, en 2017. Elle est sortie en même temps que la dernière version du C++. En outre,

elle sera probablement utilisée pour de nombreuses années encore, étant donné que la prochaine norme devrait sortir en 2020.

Les tests d'intégrations seront testés continuellement à l'aide d'un outil appelé Circle CI. Il permet de s'assurer, à chaque fois que l'on envoi des fichiers sur le serveur git, que les fonctionnalités de l'application et du serveur fonctionnent toujours correctement si elles ont déjà été développé auparavant.

Il est important pour tout ingénieur en informatique d'écrire de la documentation en rapport avec le code qu'il développe. Pour se faire, nos ingénieurs mettront un effort tout particulier d'écrire continuellement les en-têtes de leur fonction. Cela permettra notamment, à l'aide de l'outil Doxygen, de mettre à jour une documentation sous forme de page web.

L'application Android et le serveur seront également accompagné de fichier texte intitulé README, qui permettront de décrire leur fonctionnalité, ainsi que d'expliquer comment les utiliser correctement. Ce fichier agira comme un manuel d'utilisation.

6. Références (Q3.2)

Circle CI : https://circleci.com/
Doxygen : http://www.doxygen.nl/
GitFlow: https://datasift.github.io/gitflow/IntroducingGitFlow.html

ANNEXES

Norme C++17 : ISO/IEC 14882:2017 de l'organisation internationale de Normalisation (ISO)