Équipe 210

Draw It Up!
Plan de projet

Version 1.5

Historique des révisions

Date	Version	Description	Auteur
2021-02-14	1.0	Écriture du de quelques lots dans le l'échéancier	Gérard Akkerhuis
2021-02-15	1.1	Écriture de l'énoncé de travail Ajout de lots dans l'échéancier	Gérard Akkerhuis Vincenzo Pucci-B
2021-02-17	1.2	Ajout de tableaux (section 3.3) Ajout d'entente contractuelle	Marc-André Filion Bouchra Nour El Yakin Dahamni
2021-02-19	1.3	Dernières modifications et révision finale en équipe avant la remise.	Équipe 210
2021-04-18	1.4	Révision des sections 2 et 3	Marc-André Filion
2021-04-19	1.5	Révision finale de l'équipe avant la remise	Équipe 210

Table des matières

Plan de projet

1. Introduction

Dans ce document, nous discuterons du plan de projet du projet logiciel “Fais-moi un dessin”. Tout d’abord, nous allons décrire notre proposition de projet et les biens livrables. Ensuite, nous décrirons comment nous comptons gérer les exigences, la qualité et les risques associés au projet. Après, nous présenterons les échéances fixées par notre équipe. Puis, nous allons décrire pour chaque membre de notre équipe ses compétences et ses responsabilités au sein du projet. Finalement, nous présenterons notre proposition d’entente contractuelle pour le projet.

2. Énoncé des travaux

2.1. Solution proposée

Nous allons créer une application, disponible sur Windows et Android, de type Pictionary. La version Windows consistera en une extension d’une application servant uniquement à dessiner qui a été conçue l’an passé lors du projet 2. Nous allons donc reprendre cette base et lui ajouter les fonctionnalités en ligne, tandis que la version Android serait entièrement construite de A à Z. Cette application va permettre aux utilisateurs de jouer seuls ou en équipe dans plusieurs modes de jeu. Le premier mode de jeu sera un mode classique deux contre deux, où un des membres de l’équipe dessinera un mot et l’autre tentera de le deviner. Ainsi, l’équipe avec le plus de points à la fin des manches gagnera la partie. Le deuxième mode de jeu est nommé le sprint solo. Dans celui-ci, un joueur virtuel dessine des mots alors qu’un utilisateur tente de deviner ceux-ci dans un temps prédéterminé. En plus des modes de jeux, les utilisateurs pourront clavarder en temps réel en ayant chacun un profil qui leur sera associé, car le jeu nécessitera que les joueurs se créent un compte pour y accéder.

2.2. Hypothèses et contraintes

Pour ce plan, notre équipe doit développer deux applications, une pour ordinateur et une pour tablette Android, ainsi qu’un serveur et une base de données. Ainsi, nous devons permettre à deux clients de communiquer avec le serveur sans problème. De plus, nous devons développer un prototype de système de clavardage entre les clients pour le 19 février 2021. Nous estimons que lorsque ce prototype sera créé, nos sous-équipes seront plus à l’aise avec les communications client-serveur et que le développement du reste des applications se fera plus rapidement, et ce, jusqu’à la remise finale établie au 19 avril 2021.

Nous avons une contrainte de ressources humaines : l’équipe est constituée de 6 personnes, il faudra donc que l’énergie et le temps de chacun soient utilisés à bon escient.

Ce plan de projet considère également que les outils développés lors de l’ancien projet sont fonctionnels et complets, et donc ne nécessiteront pas de temps de développement.

Nous estimons que la base de données utilisée ainsi que le service de déploiement de Amazon Web Service sont tous deux très stables et ne vont pas entraîner de conséquences sur la fiabilité du serveur et l’intégrité des données. Notre calendrier tient donc compte de cela en ne planifiant pas de tâches d’entretien de services de mise en ligne.

Le service de gestion de version GitLab est également estimé comme très fiable et en mesure de répondre à nos besoins sur cet aspect du projet. C’est pourquoi notre projet reposera sur ce site et sera utilisé pour la gestion de configuration tel que discuté dans la section 3.4.

2.3. Biens livrables du projet

Pour ce projet, 6 artefacts majeurs seront créés et séparés en deux remises. La première remise, pour le 19 février 2021, contient le SRS, le document d’architecture logicielle, le protocole de communication, le plan de projet ainsi qu’un prototype de l’application. La deuxième remise, pour le 19 avril 2021, contient le plan de tests et le document de résultats de tests logiciels et le produit final.

3. Gestion et suivi de l'avancement

3.1. Gestion des exigences

Les changements aux exigences sont gérés grâce au processus de développement que nous employons pour le projet. En effet, nous utilisons une approche Agile, alors nous sommes en mesure d'offrir un produit évaluable après chacun des sprints que nous avons établis. Contrairement à une approche Cascade, nous demeurons flexibles à ce que des modifications dans les exigences soient apportées, puisque nous développerons les fonctionnalités au fur et à mesure et nous ne nous retrouverons pas dans une phase de non-retour en milieu de développement.

Les exigences souhaitables ont le niveau de priorité le plus bas. Advenant qu'il y ait un changement des priorités au sein de l'équipe, ce seront ces exigences qui écoperont des conséquences. Il se peut donc que celles-ci ne figurent pas dans le produit final, mais en aucun cas le produit sera livré en respectant des exigences souhaitables et pas certaines exigences essentielles.

Si, à la demande du client, les exigences changent, notre équipe continuera le sprint en cours pour terminer tout ce qui touche les exigences non modifiées. Au sprint subséquent, nous planifierons les tâches en fonction des nouveautés selon leur degré d'importance aux yeux du client, surtout si ce sont des nouvelles exigences qu'il juge essentielles. Le retrait d'exigences liées à des tâches du sprint courant implique potentiellement la complétion des tâches restantes avant la fin du sprint. Il y aura donc une coupure du sprint au moment de complétion des tâches encore nécessaires, ce qui entraînera un décalage et mise à jour du calendrier à la section 4. Des mesures pour effectuer du travail à temps supplémentaire à nos frais seront déployées, advenant le rejet d'exigences auxquelles les tâches auront déjà été complétées.

3.2. Contrôle de la qualité

Pour effectuer le contrôle de la qualité des artefacts, toutes les sections des documents sont revues par l'ensemble des membres de l'équipe avant leur remise, à un moment choisi par l'équipe lors de la planification des tâches. Cette révision permet de détecter les sections incomplètes et de les remplir, d'autant plus que le contenu aura pris compte des suggestions de tous et sera accepté unanimement.

En ce qui a trait au projet lui-même, nous utiliserons tout au long de celui-ci l'outil de gestion de version Git, accompagné du service web d'hébergement et de gestion de développement logiciel GitLab. Le processus employé explicitant leur utilisation est disponible plus en détail dans la section 3.4, mais ces outils serviront notamment à entreprendre des actions correctives en cas de problème. Dans un premier temps, GitLab va nous permettre de faire passer le code à partager à travers une brique logicielle (*pipeline*) qui effectuera une suite de tâches servant à vérifier la qualité du code, c'est-à-dire un système d'intégration continue. Ce système vérifiera d'abord s'il n'y a pas de problèmes en lien avec les dépendances (*Install*), si les bonnes pratiques de codage établies sont respectées (analyse statique, *Lint*), s'il n'entraîne pas d'erreurs de compilation (*Build*) et enfin que les tests passent (*Test*). De plus, notre méthode de travail tient compte des *merge requests*, où les portions de codes à partager sur la branche principale sont mises en attente et doivent être vérifiées et acceptées par un membre de l'équipe au préalable. Si des bogues réussissent à se faufiler malgré tout, des références aux problèmes (*issues*) seront créées sur GitLab afin de faire part d'une section problématique et que la personne ou le groupe concerné amène des correctifs imminents. De plus, un échec du *pipeline* entraînera automatiquement la notification de la personne ayant créé la *merge request* afin qu'elle corrige les anomalies.

Un plan de tests logiciels est également disponible afin de voir quelles sont les exigences à tester. Celui-ci nous permettra notamment de voir les objectifs visés par les différents types de tests, les critères permettant de confirmer l'atteinte de ces objectifs ainsi que les ressources nécessaires. Ce document servira donc à nous assister lorsqu'il sera question de faire des activités d'assurance-qualité (tests) pour le produit afin qu'il soit conforme aux attentes du client.

3.3. Gestion de risque

La description des risques suit la convention suivante :

- Ampleur : sur une échelle de 1 à 10, 10 étant le risque le plus élevé. Cette analyse est basée sur la probabilité d'occurrence du risque, ainsi que ses impacts.
- Description : une description textuelle du risque ainsi que les problèmes attendus.
- Impact : échelle définissant la portée du risque
  - C – critique (affecte le projet en entier)
  - E – élevé (affecte les fonctionnalités principales du système)
  - M – moyen (devrait être maîtrisable en appliquant une stratégie d'atténuation adéquate)
  - F – faible (l'acceptation du risque est une stratégie envisageable)
- Facteurs : aspects (**métriques**) du système pouvant être compromis.
- Stratégie de gestion : mesures à prendre afin de gérer le risque.

1 - Mauvaise planification				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
6	Mauvaise planification qui peut entraîner des retards sur l'avancement des tâches et des fonctionnalités essentielles manquantes.	E	Conformité	Subdiviser généreusement les tâches en plusieurs sous-tâches simples et focaliser sur ce qui est prioritaire.

2 - Perte d'effectif				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
2	Un membre de l'équipe abandonne le cours en milieu de projet et les tâches doivent être transférées aux autres membres.	F	Conformité	Acceptation.

3 - Comportements inattendus du système (serveur et/ou client lourd et/ou client léger)				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
5	Comportements inattendus auprès d'une ou plusieurs composantes du système qui affectent l'expérience de l'utilisateur (c.-à-d. bogues mineurs subtils).	M	Conformité	Mitiger : établir une stratégie pour l'implémentation de tests unitaires et de convivialité.

4 - Code de mauvaise qualité				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
5	Code de mauvaise qualité non commenté, difficile à comprendre et mal organisé qui le rend difficile à maintenir. Ce qui est d'ailleurs important dans le cas présent, puisque nous utilisons des langages de programmation différents pour chaque partie où le nombre de spécialistes est limité.	M	Maintenabilité	Mitiger : effectuer de la revue de code par un pair avant de le fusionner avec du code fonctionnel commun à l'ensemble de l'équipe (branche principale Git).

5 - Expiration du nom d'hôte No-IP				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
10	Expiration du nom d'hôte en ligne attribué à notre projet permettant à tous les clients de rejoindre le serveur. Le service gratuit expirant un nom après 30 jours, ce risque est inévitable.	C	Disponibilité	Enregistrer un rappel afin de veiller à garder en vie le nom d'hôte sur No-IP en le prolongeant de 30 jours quelque temps avant son expiration.

6 - Nouveaux bogues liés à l'intégration				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
8	L'intégration du code entre les différentes sous-équipes entraîne des nouveaux bogues qui n'étaient pas présents dans les portions séparées du travail. Les comportements inattendus pourraient faire planter le serveur	M	Disponibilité, Fiabilité	Effectuer de petites activités d'intégration fréquentes pour éviter un gros impact soudain.

3.4. Gestion de configuration

Tel que mentionné dans la section 3.2, nous allons utiliser Git et Gitlab afin d'effectuer la gestion de version. Pour apporter des nouveaux changements, nous allons d'abord utiliser la dénomination suivante pour indiquer une branche à travers une nouvelle fonctionnalité est travaillée :

NOM DE LA SOUS-ÉQUIPE/feature/NOM DE LA FONCTIONNALITÉ ;

où le nom de la sous-équipe est soit *angular* (client lourd), *android* (client léger) ou *server* (serveur), et le nom de la fonctionnalité est associée à une des exigences du système.

Dans le cas d'un correctif :

NOM DE LA SOUS-ÉQUIPE/(hot)fix/NOM DE LA FONCTIONNALITÉ ;

où l'on va indiquer *fix* ou *hotfix* tout dépendant de la taille du correctif à apporter.

Une fois qu'une fonctionnalité où un correctif est complété, un *merge request* est créé afin de fusionner le contenu de la branche courante vers une branche nommée *develop* qui sert de référence pour garder des versions fonctionnelles du projet à jour. Bien entendu, le *merge request* est vérifié par la ou les personnes qui est/sont attribuée(s), en plus de passer à travers une brique logicielle telle que décrite en 3.2. Une fois le *merge request* accepté, la branche source est supprimée et les étapes se répètent en parallèle jusqu'à l'arrivée d'une remise, où le contenu de *develop* est envoyé sur la branche *master,* dont le contenu se veut une version livrable qui sera présentée au client.

Nous aurons une rencontre de planification par semaine en lien avec le projet. Tous les mardi, nous nous réunirons pour parler de l'avancement du projet. Nous discuterons de ce qui a été fait par chacun au courant de la semaine et de ce qui est prévu jusqu'à la prochaine rencontre. À chaque fin de *sprint*, soit aux deux semaines, nous ferons un retour sur le déroulement et l'atteinte des objectifs, puis terminerons en parlant des prochaines fonctionnalités à implémenter, de sorte qu'après la rencontre, nous nous divisons en binômes pour planifier les tâches à réaliser par rapport aux fonctionnalités nouvellement prévues.

Les artefacts seront nommés par leur titre respectif accompagné de leur numéro de version selon les révisions apportées. Toutes les modifications apportées au sein d'une même journée constituent en une nouvelle version.

4. Échéancier du projet

Tâche	Effort estimé (heures-personne)	Date de début	Date de fin
Janvier	90	2021-01-20	2021-01-31
Réponse à l'appel d'offres	593	2021-01-20	2021-02-19
Sprint 1	90	2021-01-20	2021-02-03
Liste d'exigences	12	2021-01-20	2021-01-27
Apprentissage de Go	10	2021-01-25	2021-01-30
Spécification des requis (SRS): Rédaction de l'introduction	1	2021-01-27	2021-02-04

Spécification des requis (SRS): Rédaction de la description globale	5	2021-01-27	2021-02-04
Spécification des requis (SRS): Rédaction des exigences non fonctionnelles	14	2021-01-27	2021-02-04
Spécification des requis (SRS): Rédaction des exigences fonctionnelles	14	2021-01-27	2021-02-04
Familiarisation avec Electron	12	2021-01-30	2021-02-03
Familiarisation avec l'architecture MVVM	10	2021-01-30	2021-02-03
Apprentissage de la programmation Android	12	2021-01-31	2021-02-10
Février	778	2021-02-01	2021-02-28
Sprint 2	107	2021-02-04	2021-02-17
Prototype de communication client lourd-serveur	15	2021-02-07	2021-02-17
Prototype de communication client lourd-serveur	15	2021-02-07	2021-02-17
Protocole de communication: Rédaction de l'introduction	1	2021-02-12	2021-02-19
Protocole de communication: Rédaction de la communication client-serveur	2	2021-02-12	2021-02-19
Protocole de communication: Rédaction de la description des paquets	2	2021-02-12	2021-02-19
Document d'architecture logicielle: Rédaction de la vue logique	6	2021-02-13	2021-02-19
Document d'architecture logicielle: Rédaction de l'introduction	1	2021-02-14	2021-02-19
Document d'architecture logicielle: Rédaction des objectifs et contraintes architecturaux	2	2021-02-14	2021-02-19

Plan de projet: Rédaction de l'échéancier du projet	6	2021-02-14	2021-02-19
Plan de projet: Rédaction de l'énoncé des travaux	2	2021-02-15	2021-02-19
Déploiement du serveur sur AWS	8	2021-02-16	2021-02-17
Document d'architecture logicielle: Rédaction de la vue des cas d'utilisation	8	2021-02-16	2021-02-19
Document d'architecture logicielle: Rédaction de la vue des processus	8	2021-02-16	2021-02-19
Document d'architecture logicielle: Rédaction de la vue de déploiement	6	2021-02-16	2021-02-19
Document d'architecture logicielle: Rédaction de la taille et de la performance	5	2021-02-16	2021-02-19
Protocole de communication: Rédaction de la communication serveur à base de données	5	2021-02-16	2021-02-19
Plan de projet: Rédaction de l'introduction	1	2021-02-16	2021-02-19
Plan de projet: Rédaction de la gestion et suivi de l'avancement	4	2021-02-16	2021-02-19
Spécification des requis (SRS) : Modification et correction suite à la rétroaction	10	2021-02-16	2021-02-19
Sprint 3	388	2021-02-18	2021-03-03
Test et intégration des prototypes	8	2021-02-18	2021-02-19
Remise de la réponse à l'appel d'offres			2021-02-19
Produit final	1146	2021-02-20	2021-04-19
Client lourd : Mode intégré de l'interface de clavardage	10	2021-02-20	2021-02-24
Client lourd : Rejoindre un ou plusieurs canaux de discussion	15	2021-02-20	2021-02-24
Client léger : Rejoindre un ou plusieurs canaux de discussion	15	2021-02-20	2021-02-24

Client lourd : Historique de conversation	15	2021-02-20	2021-02-27
Client léger : Historique de conversation	15	2021-02-20	2021-02-27
Client léger : Fenêtre de clavardage intégrée	20	2021-02-20	2021-02-27
Client lourd : Mode classique : Création d'une partie multijoueur avec et sans joueur virtuel (Matchmaking system)	20	2021-02-24	2021-03-06
Client lourd : Mode classique : Système de transmission d'images à tous les joueurs (Broadcasting system)	20	2021-02-24	2021-03-06
Client lourd : Mode classique : Système de résultat du dénouement d'une partie selon une victoire ou une défaite (Result system)	15	2021-02-24	2021-03-03
Client lourd : Interface d'utilisateur d'une partie (UI et UX)	10	2021-02-24	2021-03-03
Client lourd : Mode classique : Création d'une partie multijoueur avec et sans joueur virtuel (system matchmaking)	20	2021-02-24	2021-03-06
Client léger : Mode classique : Système de transmission d'images à tous les joueurs (Broadcasting system)	20	2021-02-24	2021-03-06
Client léger : Mode classique : Système de résultat du dénouement d'une partie selon une victoire ou une défaite (Result system)	20	2021-02-24	2021-03-06
Client léger : Interface d'utilisateur d'une partie (UI et UX)	15	2021-02-24	2021-03-03
Client lourd : Associer un mot, un ou des indices et un niveau de difficulté à un	10	2021-02-24	2021-03-03

dessin			
Client lourd : Informations de base de l'utilisateur sur son profil (informations privées et publiques)	20	2021-02-24	2021-03-03
Client léger : Informations de base de l'utilisateur sur son profil (informations privées et publiques)	15	2021-02-24	2021-03-03
Mars	647	2021-03-01	2021-03-31
Client lourd : Aperçu accéléré de l'animation d'un dessin	15	2021-03-02	2021-03-09
Client lourd : Mode fenêtre de l'interface de clavardage et possibilité d'alterner	20	2021-03-02	2021-03-16
Client léger : Ajout du mode sprint solo	20	2021-03-02	2021-03-16
Client léger : Ajout du mode sprint solo	20	2021-03-02	2021-03-16
Client lourd : Ajout du mode sprint coopératif	15	2021-03-02	2021-03-09
Client léger : Ajout du mode sprint coopératif	15	2021-03-02	2021-03-09
Sprint 4	165	2021-03-04	2021-03-17
Client lourd : Mode conventionnel du joueur virtuel	20	2021-03-10	2021-03-17
Client lourd : Mode aléatoire du joueur virtuel	20	2021-03-10	2021-03-17
Client lourd : Mode panoramique du joueur virtuel	20	2021-03-10	2021-03-20
Client lourd : Mode centré du joueur virtuel	20	2021-03-10	2021-03-20
Client lourd : Personnalités des joueurs virtuels	15	2021-03-10	2021-03-20
Client lourd : Statistiques d'utilisation du jeu sur le profil de l'utilisateur (nombre de parties jouées, pourcentage de victoires, temps moyen d'une partie et temps total passé à jouer)	20	2021-03-16	2021-03-26

Client léger : Statistiques d'utilisation du jeu sur le profil de l'utilisateur (nombre de parties jouées, pourcentage de victoires, temps moyen d'une partie et temps total passé à jouer)	15	2021-03-16	2021-03-26
Client lourd : Historique détaillé de l'utilisateur sur son profil (dates et heures de connexion / déconnexion et historique des parties jouées)	20	2021-03-16	2021-03-26
Client léger : Historique détaillé de l'utilisateur sur son profil (dates et heures de connexion / déconnexion et historique des parties jouées)	15	2021-03-16	2021-03-26
Sprint 5	106	2021-03-18	2021-03-31
Client lourd : Système de trophées	20	2021-03-23	2021-03-26
Client léger : Système de trophées	10	2021-03-23	2021-03-26
Client lourd : Système d'équipe dans le jeu	15	2021-03-26	2021-03-31
Client léger : Système d'équipe dans le jeu	15	2021-03-26	2021-03-31
Client lourd : Canal de discussion et création de partie propre à une équipe	15	2021-03-27	2021-03-31
Client léger : Canal de discussion et création de partie propre à une équipe	15	2021-03-27	2021-03-31
Client lourd : Création d'autocollants	16	2021-03-28	2021-03-31
Avril	224	2021-04-01	2021-04-19
Sprint 6	224	2021-04-01	2021-04-19
Client lourd : Enregistrement d'autocollants	20	2021-04-01	2021-04-15
Client lourd : Téléversement d'autocollants	20	2021-04-01	2021-04-15

Client léger : Téléversement d'autocollants	10	2021-04-01	2021-04-08
Client lourd : Classement	20	2021-04-02	2021-04-16
Client léger : Classement	15	2021-04-02	2021-04-08
Client lourd : Mot de passe oublié	15	2021-04-02	2021-04-08
Client léger : Mot de passe oublié	6	2021-04-02	2021-04-08
Client lourd : Effets sonores et visuels pour un nouveau trophée	6	2021-04-02	2021-04-08
Client léger : Effets sonores et visuels pour un nouveau trophée	6	2021-04-02	2021-04-08
Client lourd : Effets sonores et visuels pour une victoire ou défaite	6	2021-04-02	2021-04-08
Client léger : Effets sonores et visuels pour une victoire ou défaite	6	2021-04-02	2021-04-08
Client lourd : Compteur de temps, son de compte à rebours et rétroaction sonore pour une tentative de résolution	6	2021-04-02	2021-04-08
Client léger : Compteur de temps, son de compte à rebours et rétroaction sonore pour une tentative de résolution	6	2021-04-02	2021-04-08
Client léger : Notification sonore et visuelle d'un nouveau message	6	2021-04-02	2021-04-08
Client lourd : Tutoriel	15	2021-04-09	2021-04-12
Client léger : Tutoriel	15	2021-04-09	2021-04-12
Client lourd : Filtre de blasphème	6	2021-04-09	2021-04-12
Rédaction du plan de tests logiciels	20	2021-04-12	2021-04-19
Rédaction des résultats de tests logiciels	20	2021-04-12	2021-04-19
Remise du produit final			2021-04-19
Total du projet	1739	2021-01-20	2021-04-19

5. Équipe de développement

5.1 Pucci-Barbeau, Vincenzo :

- Étudiant de 3e année au baccalauréat de génie logiciel à l'école Polytechnique de Montréal
- Maîtrise des langages Typescript, C++ et du cadriciel Angular
- Expérience en développement de jeu avec Unity
- Responsable du client lourd

5.2 Akkerhuis, Gérard Mugisha :

- Étudiant de 3e année au baccalauréat de génie logiciel à l'école Polytechnique de Montréal
- Maîtrise des langages Java, C++ et Javascript
- Expérience en développement Web (MEAN Stack)
- Responsable du client léger

5.3 Turcot, Charles-Édouard :

- Étudiant de 4e année au baccalauréat de génie logiciel à l'école Polytechnique de Montréal
- Maîtrise des langages Java et Python.
- Expérience en industrie en développement d'applications en Java avec support et assurance-qualité. (DevOps)
- Responsable du client léger

5.4 Filion, Marc-André :

- Étudiant de 3e année au baccalauréat de génie logiciel à l'école Polytechnique de Montréal
- Maîtrise des langages Java, Typescript et C++
- Expérience en développement Web (MEAN Stack)
- Responsable du serveur

5.5 Saint-Cyr, Benjamin:

- Étudiant de 3e année au baccalauréat de génie logiciel à l'école Polytechnique de Montréal
- Maîtrise des langages Java, C#, Typescript, Python et C++
- Expérience en développement Web (MEAN Stack)
- Responsable du serveur

5.6 El Yakin Dahamni Bouchra Nour :

- Étudiante en 3ème année génie logiciel à l'école Polytechnique de Montréal.
- Maîtrise des langages de C++ , TypeScript , Python.
- Expérience en développement Web .
- Responsable du client lourd

6. Entente contractuelle proposée

En réponse à l'appel d'offres émis par Polytechnique, nous suggérons une entente contractuelle de type livraison clé en main. Nous choisissons cette entente contractuelle, car l'ensemble des exigences essentielles est connu lorsque nous recevons l'appel d'offres. Notre équipe s'engage à livrer le produit final pour le 19 avril 2021 respectant toutes les exigences essentielles. En ce qui concerne les exigences souhaitables, l'équipe 210 s'engage à livrer 50 % de ce qui a été proposé à l'appel d'offres. Comme mentionné plus haut dans la section 5, notre équipe est composée de 5 développeurs prenant un salaire horaire de 100$/heure. Il faudra aussi ajouter le montant chargé par le gestionnaire de projet qui prend 125$/heure. La gestion de projet prend un temps approximatif de 185 heures et est incluse dans les lots de travail de l'échéancier présenté. À chaque deux semaines, un membre de l'équipe laissera son rôle de développeur de côté pour prendre le rôle de gestionnaire et celui qui avait le rôle de gestionnaire, prendra le rôle de développeur. Le paiement doit être obligatoirement effectué à la date de livraison du projet, soit le 19 avril 2021. Le montant demandé pour la livraison du produit est 112 625$, ce montant est le résultat du calcul suivant (895 heures X 100$) + (185 heures X 125$).