Équipe 102

ColorImage
Plan de projet

Version 2.0

Historique des révisions

Date	Version	Description	Auteur
2021-09-29	1.0	Rédaction initiale	Équipe 102
2021-12-04	2.0	Révision du plan de projet pour la remise du produit finale	Équipe 102

Table des matières

Plan de projet

1. Introduction

Ce document représente notre planification qui fait le tour du développement de notre logiciel ColorImage. On décrira ainsi plusieurs aspects du développement dans différentes sections distinctes. Tout d'abord, dans la section 2, on énoncera l'énoncé des travaux où l'on va expliquer la solution qui a été proposée, les hypothèses et contraintes que notre projet doit respecter, ainsi que les biens livrables de ce dernier. Par la suite, dans la section 3, nous entamerons la gestion et suivi de l'avancement qui portera plus précisément sur la gestion des exigences, le contrôle de la qualité, la gestion des risques et la gestion de configuration. Ensuite, la section 4 présentera l'échéancier du projet en précisant le lot de travail, l'effort estimé, la date de début et de fin de chaque sprint, ainsi que le total d'heures que chacun de ces sprints aura comme effort estimé en heures. La section 5 quant à elle décrira notre équipe de développement où chaque membre fera une brève description de ses connaissances et ses expériences acquises tout au long de son parcours en tant qu'étudiant, en plus d'énumérer ses responsabilités principales et secondaires. Finalement, la section 6 portera sur l'entente contractuelle proposée entre l'équipe 102 et l'entreprise PolyApps en réponse à l'appel d'offres.

2. Énoncé des travaux

2.1. Solution proposée

Nous proposons comme solution de développer une application multiplateforme supportée par le système Android sur tablette et Windows 10 sur ordinateur. L'application se nommant ColorImage sera une plateforme de dessin connectée à un serveur et permettant à plusieurs utilisateurs, tant sur tablette que sur ordinateur, de dessiner sur le même dessin et de voir les modifications des autres utilisateurs en temps réel.

Le client lourd est une évolution du logiciel de dessin PolyDessin. Cette évolution va englober la plupart des fonctionnalités de dessin majeures présentes dans le logiciel PolyDessin (trait, efface, sélection…) et ajoutera des fonctionnalités permettant le dessin collaboratif, comme un système de clavardage pour faciliter la communication entre membres d'une équipe de dessin. Le client léger quant à lui va être développé à partir de zéro, et en général, il inclura les mêmes fonctionnalités de dessin et de clavardage que le client lourd. Ainsi, le système permettra aux utilisateurs de créer un nouveau dessin, d'y inviter des collaborateurs, de sauvegarder leur dessin afin de le continuer à une date ultérieure et de partager leur dessin à une galerie de dessin visible par tous les autres utilisateurs du système.De plus, la solution intégrera la possibilité de former des équipes de collaboration afin de faire des dessins en groupe, de ce fait, les utilisateurs auront la possibilité de créer et de rejoindre des équipes, ainsi que d'initier des dessins propres à leurs équipes. Les spécifications détaillées sont écrites dans le document de spécification des requis (SRS). Veuillez vous y référer pour plus de détails.

Pour accéder à l'application, le système imposera aux utilisateurs de faire un compte afin de se connecter. Le système de compte permettra aux utilisateurs d'être identifiables par un pseudonyme unique, permettant ainsi de mieux gérer les fonctionnalités de création d'équipe et de clavardage.

2.2. Hypothèses et contraintes

L'équipe de développement doit être composée de six membres, tous étudiants en génie logiciel à l'école Polytechnique. Chacun d'eux devra fournir un minimum de 12 heures par semaine pour un total de développement de 1080 heures-personnes. Nous supposons qu'il n'y aura aucun membre de l'équipe qui abandonnera le projet en cours de développement et qu'aucun conflit ne causera de retard ou de problèmes de développement.

Concernant le développement, l'équipe aura accès à leurs ordinateurs personnels où ils effectueront le développement. Ils auront également accès aux locaux de l'université pour effectuer les rencontres et les suivis hebdomadaires, ainsi qu'à une connexion internet fiable. Le serveur sera hébergé à l'aide de l'hébergeur Heroku. Quant à la base de données, elle sera hébergée sur MongoDB. Le développement de client lourd se basera sur le logiciel PolyDessin, nous supposons qu'il n'y aura aucun problème majeur d'adaptation des fonctionnalités de ce logiciel à nos besoins.

Une réponse à l'appel d'offres sera soumise le 1er octobre 2021. Cette dernière contiendra tous les documents relatifs au développement, ainsi qu'un prototype initial de communication client-serveur. Le produit final sera livré le 6 décembre 2021. Il y aura des suivis d'avancement chaque semaine en personne entre les membres de l'équipe, ainsi que sur la plateforme de collaboration Jira. On suppose que l'échéancier est adapté aux membres de l'équipe et est respectable.

2.3. Biens livrables du projet

Réponse à l'appel d'offre (1er octobre 2021)

- Plan de projet.
- Liste d'exigences
- Spécification des requis (SRS).
- Architecture logicielle.
- Protocole de communication client - serveur
- Prototype de communication client lourd - serveur.
- Prototype de communication client léger - serveur.

Produit final (6 décembre 2021)

- Artéfacts mis à jour.
- Plan de tests.
- Résultats des tests.
- Code source du client léger.
- Code source du client lourd.
- Code source du serveur.
- Client léger, compilé en fichier .apk prêt à être installé sur système Android.
- Client lourd, compilé en fichier .exe prêt à être exécuté sur un système Windows 10.

3. Gestion et suivi de l'avancement

3.1. Gestion des exigences

L'équipe a dressé une liste d'exigences qui a été validée par les chargés, puis un document de spécification des requis (SRS) a été écrit, en se basant sur la liste des exigences. Les tâches de chaque membre seront créées sur la plateforme JIRA en suivant le document de spécification des requis à la lettre.

Si une modification aux exigences venait à avoir lieu en cours de développement, une rencontre d'équipe aura lieu afin d'en discuter et vérifier qu'elle répond encore aux exigences du client, si c'est le cas, une mise à jour du

document de spécification des requis sera effectuée et si nous estimons que cette dernière aura un gros impact sur la planification, une replanification des échéances aura lieu en conséquence. Si un changement d'exigence affecte des fonctionnalités qui ont déjà été implémentées, nous créerons de nouvelles tâches sur le projet du Jira. Elles seront planifiées pour un sprint futur. Si le changement affecte des fonctionnalités qui n'ont pas été implémentées, nous modifierons seulement les tâches affectées dans le Jira.

3.2. Contrôle de la qualité

Premièrement, les artefacts seront relus par tous les membres de l'équipe et le logiciel Antidote sera utilisé afin de s'assurer qu'il n'y ait aucune faute de syntaxe ou de grammaire.

Ensuite, afin d'assurer un bon contrôle de la qualité du code, nous aurons recours à plusieurs procédures visant à optimiser les revues d'ajouts de fonctionnalités. Premièrement, nous utiliserons la plateforme Gitlab, et nous utiliserons la fonctionnalité *merge request* qui fait en sorte qu'un ajout ne peut être intégré que s'il est révisé et validé par d'autres membres de l'équipe. Ainsi, si un problème ou défaut n'est pas remarqué par l'initiateur du *merge request*, il sera très probablement remarqué par d'autres membres de l'équipe. Deuxièmement, le développement s'effectuera en parallèle à l'aide d'une fonctionnalité de branches, ainsi les fonctionnalités ne seront ajoutées que lorsqu'elles seront complètement prêtes et testées. Troisièmement, si un bogue ou problème est trouvé par un membre de l'équipe, nous créeront une tâche Jira en conséquence, s'il s'agit d'un bogue majeur qui affecte nos ressources trop longtemps, une replanification de l'échéancier aura lieu. Finalement, nous développerons un plan de tests qui couvrira tous les cas d'utilisations et qui veillera ainsi à ce que tout s'exécute normalement et qu'il y ait le moins de bogues possibles.

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.

01 - Hébergeur du serveur (Heroku)				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
1	Le serveur est déployé sur la plateforme infonuagique Heroku. il est probable que cette dernière soit inaccessible.	C	Temps d'indisponi--bilité du système	L'utilisation des pipelines gitlab résout en partie ce problème. En effet, notre pipeline de déploiement automatique va tenter de déployer le serveur sur l'hébergeur Heroku, si c'est impossible, elle tentera de déployer sur l'hébergeur AWS.

02 - Intégration serveur - client				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
3	La plupart des fonctionnalités nécessitent d'avoir une communication entre le serveur et les clients.	E	Nombre de bogues Temps de développement de l'application	Lors de l'intégration entre un client et le serveur, nous allons faire une réunion avec au moins une personne travaillant sur le serveur et au moins une personne travaillant sur le client en question. Cela nous permettra d'éviter plusieurs bogues. À la découverte d'un bogue lié à une mauvaise intégration, nous allons planifier une rencontre avec un membre du serveur et un membre du client concerné afin d'y résoudre rapidement le problème.

03 - Capacité du serveur à gérer toutes les requêtes				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
3	Lors de l'édition d'un dessin, l'utilisateur doit voir en temps réel les changements que les autres utilisateurs effectuent. Il y aura donc plusieurs requêtes qui devront être envoyées au serveur simultanément.	M	Temps de réponse du serveur	Avant d'implémenter une nouvelle fonctionnalité nécessitant une communication au serveur, nous allons vérifier que le client envoie seulement les informations essentielles à la fonctionnalité au serveur. Dans le cas où cette problématique arrive après que nous nous soyons assurés d'envoyer un minimum d'informations au serveur, nous pourrions ajouter des limites à notre application afin de réduire le temps de réponse du serveur à un niveau acceptable.

04 - Incompatibilité entre les librairies utilisées pour le client léger et le client lourd				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
8	Les communications avec le serveur sont faites en sorte que le dessin doit être fait en SVG. Il y a un risque d'incompatibilité avec la librairie VectorDrawable en Kotlin qui ne supportera pas toutes les balises SVG.	M	Temps de développement de l'application	Avant de commencer l'implémentation du dessin collaboratif, il faudra faire une recherche pour s'assurer qu'il existe une librairie qui sera compatible autant pour le client lourd que le client léger. Il faudra alors évaluer que la librairie qui répliquera le client lourd sera possible sans ajouter trop d'efforts.

06 - Cohérence entre le client léger et le client lourd				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
4	On pourrait avoir des différences au niveau de l'interface de l'utilisateur ou même dans le comportement de certaines fonctionnalités nouvellement implémentées entre client lourd et client léger.	C	Nombre d'éléments incohérents entre l'interface du client lourd et celle de léger Temps consacré à corriger les incohérences entre les interfaces.	Il faudra impérativement que les développeurs du client lourd et du client léger partagent continuellement le développement de leurs fonctionnalités entre eux pour vérifier la similitude de leurs applications. Ainsi, il faudra qu'à chaque rencontre hebdomadaire, on teste les 2 clients pour s'assurer de leurs cohérences. Ainsi, lorsqu'il y aura des différences remarquées, les développeurs des deux clients pourront se rencontrer de nouveau pour parler de la stratégie à entamer quant à la résolution de ses incohérences.

3.4. Gestion de configuration

Lorsqu'un changement est demandé ou qu'un problème est découvert, nous allons créer une tâche Jira qui sera assignée à un membre de l'équipe. Une description et une priorité seront ajoutées dans cette tâche. Si c'est un changement qu'on doit faire, le membre responsable de cette tâche crée une branche avec un nom débutant par « edited » et terminant par un titre descriptif du changement. Quant à un bogue, le nom de la branche débutera par « bug-fix » et elle se terminera par un titre descriptif du bogue.

Pour les artéfacts, nous allons nommer chacun des documents avec son nom de l'artéfact. Par exemple, le document du plan de projet sera nommé « Plan_de_projet_équipe_102 ». Suite à la rédaction initiale d'un artéfact, nous commençons à la version 1.0. Par la suite, à chaque modification majeure de l'artéfact, le premier chiffre de version sera incrémenté de 1 et le second chiffre reviendra à 0. Si on est à la version 1.3 d'un artéfact et qu'on effectue une modification majeure sur ce même document, nous passerons à la version 2.0. Par contre, si on ne fait qu'une modification qui n'a pas un grand impact sur l'artéfact, on incrémente le second chiffre de 1. L'historique des versions de chaque artéfact se trouve sur la page suivante la première page et contiendra les descriptions associées à chaque version ainsi que le détails des ajouts et des modifications, et les auteurs de chaque révision.

4. Échéancier du projet

Pour ce qui est de ce projet, nous allons respecter un ratio de 45 par crédit par personne. Le cours étant de 4 crédits avec notre équipe composée de 6 camarades, notre nombre total d'heures est de 45*4*6 = 1080 heures-personnes. Notre échéancier serait séparé en lots et en jalons. Chaque ligne du tableau correspond à un lot de travail et les jalons sont la fin de chaque phase (surligné en vert), signifiant la soumission d'un livrable. Notre échéancier a deux jalons principaux : la réponse à l'appel d'offres et la remise du produit final. Ceux-ci sont divisés en plusieurs sprints, soit 2 sprints de 2 semaines pour la réponse à l'appel d'offres et 3 sprints de 3 semaines pour la remise du produit final.

Pour faciliter la lecture et la composition du second livrable, soit le produit final, nous avons sectionné les sprints en deux parties: client lourd et client léger. Le dernier sprint fait exception d'une troisième section, soit les artefacts, dus à la rédaction du plan de tests et de la révision des documents reliés à l'appel d'offres, si nécessaire.

Lot de travail	Effort estimé (heure-personne)	Date de début	Date de Fin
Sprint 1		06-09-2021	19-09-2021
Rédaction de la première version des requis du système	40h	06-09-2021	17-09-2021
Choix de la liste d'exigences	18h	06-09-2021	09-09-2021
Corriger la liste des exigences	12h	09-09-2021	13-09-2021
Création de l'interface de la page de connexion du client lourd	20h	10-09-2021	01-10-2021
Création de l'interface de la page de connexion du client lourd	20h	10-09-2021	01-10-2021
Développement de la fonctionnalité d'envoi de message, de connection et de déconnection, côté serveur	30h	10-09-2021	01-10-2021
Total	140h		
Sprint 2		20-09-2021	04-10-2021
Intégration du serveur avec le prototype du client lourd	22h	20-09-2021	01-10-2021
Intégration du serveur avec le prototype du client léger	22h	20-09-2021	01-10-2021
Création de l'interface de la messagerie pour le prototype du client léger	20h	20-09-2021	01-10-2021
Rédaction du plan de projet	24h	21-09-2021	01-10-2021

Création de l'interface de la messagerie pour le prototype du client lourd	20h	22-09-2021	01-10-2021
Rédaction du document d'architecture	40h	22-09-2021	01-10-2021
Rédaction du protocole de communication	26h	23-09-2021	01-10-2021
Rédaction de la deuxième version des requis du système	10h	27-09-2021	01-10-2021
Total	184h		
Remise de la réponse à l'appel d'offre			01-10-2021
Sprint 3		04-10-2021	25-10-2021
Client Lourd			
Profil utilisateur - Ajouter la création de compte	8h	04-10-2021	08-10-2021
Profil utilisateur - Implémenter l'authentification au compte	8h	08-10-2021	11-10-2021
Clavardage - Possibilité de clavarder en mode fenêtré et intégré.	10h	08-10-2021	11-10-2021
Clavardage - Ajouter canaux de discussion et historiques	8h	08-10-2021	13-10-2021
Outil - Revoir l'interface de dessinage de PolyDessin pour pouvoir supporter les outils à implémenter	6h	08-10-2021	13-10-2021
Outils - Intégrer l'outil main libre avec le serveur.	8h	08-10-2021	13-10-2021
Outils - Intégrer l'outil Forme avec le serveur.	8h	13-10-2021	15-10-2021
Outils - Intégrer la sélection avec le serveur.	8h	15-10-2021	18-10-2021
Outils - Implémenter l'outil Annuler/Refaire	5h	15-10-2021	18-10-2021
Outils - Intégrer l'outil Annuler/Refaire avec le serveur	12h	15-10-2021	18-10-2021
Outils - Implémenter l'outil Sélection	12h	18-10-2021	20-10-2021
Outils - Intégrer l'outil Sélection avec le serveur	15h	18-10-2021	23-10-2021

Outils - Implémenter l'outil Suppression	4h	23-10-2021	25-10-2021
Outils - Intégrer l'outil Suppression avec le serveur	10h	23-10-2021	25-10-2021
Client Léger			
Profil utilisateur - Ajouter la création de compte	12h	04-10-2021	11-10-2021
Profil utilisateur - Implémenter l'authentification au compte	13h	04-10-2021	11-10-2021
Clavardage - Ajouter canaux de discussion et historiques	12h	04-10-2021	11-10-2021
Clavardage - Intégrer la fenêtre de clavardage	6h	11-10-2021	13-10-2021
Créer une interface de dessin et une vue de dessin	6h	11-10-2021	13-10-2021
Outils - Implémenter l'outil Main Libre	6h	11-10-2021	13-10-2021
Outils - intégrer l'outil Main Libre avec le serveur	7h	13-10-2021	15-10-2021
Outils - Implémenter l'outil Forme (Ellipse et Rectangle)	6h	15-10-2021	18-10-2021
Outils - Intégrer l'outil forme (Ellipse et Rectangle), avec le serveur	10h	15-10-2021	18-10-2021
Outils - Implémenter l'outil Annuler/Refaire	6h	15-10-2021	18-10-2021
Outils - Intégrer l'outil Annuler/Refaire avec le serveur	12h	18-10-2021	20-10-2021
Outils - Implémenter l'outil Sélection	10h	20-10-2021	23-10-2021
Outils - Intégrer l'outil Sélection avec le serveur	12h	20-10-2021	23-10-2021
Outils - Implémenter l'outil Suppression	7h	23-10-2021	25-10-2021
Outils - Intégrer l'outil Suppression avec le serveur	5h	23-10-2021	25-10-2021
Total	252h		
Sprint 4		25-10-2021	15-11-2021
Client Lourd			
Galerie de dessins - Créer l'interface de la galerie de dessin	16h	25-10-2021	1-11-2021

Galerie de dessins - Afficher les dessins publics, protégés et privés propre à l'utilisateur.	10h	25-10-2021	1-11-2021
Galerie de dessins - Intégrer le clavardage à la galerie de dessin.	10h	25-10-2021	1-11-2021
Galerie de dessins - Intégrer l'édition de dessin à la galerie de dessins.	20h	25-10-2021	1-11-2021
Galerie de dessins - Permettre la création d'un nouveau dessin collaboratif à partir de la galerie de dessin.	5h	25-10-2021	27-11-2021
Galerie de dessins - Permettre le filtrage de la liste de dessins	5h	27-10-2021	1-11-2021
Équipe de Collaboration - Implémenter la création d'équipe de collaboration	5h	1-11-2021	3-11-2021
Équipe de collaboration - Implémenter l'affichage du statut des membres de l'équipe de collaboration	3h	1-11-2021	4-11-2021
Outils - intégrer l'outil Translation avec le serveur	20h	3-11-2021	10-11-2021
Outils - Intégrer l'outil Redimensionnement avec le serveur	20h	10-11-2021	15-11-2021
Historique de dessins - Implémenter la fonctionnalité	16h	10-11-2021	15-11-2021
Client Léger			
Galerie de dessins - Créer l'interface de la galerie de dessin	20h	25-10-2021	1-11-2021
Galerie de dessins - Afficher les dessins publics, protégés et privés propre à l'utilisateur.	10h	25-10-2021	28-10-2021
Équipe de collaboration - Créer l'interface pour visionner les équipes de collaboration liées au compte connecté	16h	28-10-2021	1-11-2021
Équipe de collaboration - Créer l'interface pour visionner les dessins reliés à une équipe de collaboration	10h	1-11-2021	4-11-2021
Outils - Implémenter l'outil Translation	10h	1-11-2021	4-11-2021
Outils - intégrer l'outil Translation avec le serveur	10h	4-11-2021	8-11-2021

Outils - Implémenter l'outil Redimensionnement	10h	4-11-2021	8-11-2021
Outils - Intégrer l'outil Redimensionnement avec le serveur	10h	8-11-2021	15-11-2021
Outils - Implémenter l'outil Texte	16h	8-11-2021	15-11-2021
Outils - Intégrer l'outil Texte avec le serveur	10h	8-11-2021	15-11-2021
Total	252h		
Sprint 5		15-11-2021	06-12-2021
Client Lourd			
Outils - Implémenter l'outil Rotation	8h	15-11-2021	16-11-2021
Outils - Implémenter l'outil Texte	10h	16-11-2021	18-11-2021
Outils - Intégrer l'outil Texte avec le serveur	10h	16-11-2021	18-11-2021
Outils - Implémenter la fonctionnalité couche	10h	18-11-2021	20-11-2021
Outils - Intégrer la fonctionnalité couche avec le serveur	10h	18-11-2021	20-11-2021
Profil utilisateur et historique - Implémenter la page "Paramètre de compte"	20h	20-11-2021	27-11-2021
Profil utilisateur et historique - Afficher l'historique de connexion/déconnexion et l'historique d'édition	10h	27-11-2021	30-11-2021
Avatar - Implémenter le téléversement de l'avatar d'un utilisateur.	8h	30-11-2021	03-12-2021
Effets sonores - implémenter un son lorsque l'utilisateur reçoit un message dans la messagerie	8h	30-11-2021	03-12-2021
Apparence - Ajout de 4 thèmes de couleurs différents	6h	03-12-2021	06-12-2021
Client Léger			
Outils - Implémenter l'outil Rotation	8h	15-11-2021	17-11-2021
Outils - Intégrer l'outil Rotation avec le serveur	8h	15-11-2021	17-11-2021
Outils - Implémenter l'outil Couche	16h	17-11-2021	22-11-2021
Outils - intégrer l'outil Couche avec le serveur	10h	17-11-2021	22-11-2021
Gallerie de dessin - Filtrer les dessins par les	4h	22-11-2021	23-11-2021

attributs			
Historique des versions - Implémenter le changement de version de dessin fait par un client lourd	6h	23-11-2021	24-11-2021
Avatar - Ajouter une liste prédéfinie d'images pour le choix d'avatar	6h	23-11-2021	24-11-2021
Avatar - Permettre de choisir un avatar à l'aide de l'appareil photo intégré	6h	24-11-2021	25-11-2021
Profil utilisateur et historique - Implémenter la page "Paramètre de compte"	20h	25-11-2021	2-12-2021
Profil utilisateur et historique - Afficher l'historique de connexion/déconnexion et l'historique d'édition	10h	2-12-2021	4-12-2021
Clavardage - Ajouter la notification à la réception du nouveau messages	5h	4-12-2021	6-12-2021
Artéfacts			
Rédiger le plan de tests	20h	04-10-2021	15-11-2021
Produire les résultats de tests	10h	04-10-2021	6-12-2021
Revoir et corriger le document SRS	8h	4-12-2021	6-12-2021
Revoir et corriger l'architecture logicielle	8h	4-12-2021	6-12-2021
Revoir et corriger le plan de projet	2h	4-12-2021	6-12-2021
Revoir et corriger le protocole de communication	5h	4-12-2021	6-12-2021
Total	252h		
Grand total	1080h		
Remise du produit final			06-12-2021

5. Équipe de développement

Aleksandar Stijelja

Étudiant de troisième année en génie logiciel à Polytechnique de Montréal. Il a acquis de l'expérience dans les langages de programmation tels que C++, Java, JavaScript et Python, à travers ses années d'études. De plus, il acquis bagage soutenu en développement Web frontal, particulièrement avec les technologies comme HTML/CSS, TypeScript et le cadriciel Angular, étant donné qu'il s'est largement occupé du développement frontal du projet intégrateur de deuxième année. Son stage de 8 mois accompli chez la compagnie Haivision lui a permis de raffiner sa compétence en gestion de projet.

Responsabilité principale:

- Développement du client lourd

Responsabilités secondaires:

- Animateur principal durant les conférences de groupe
- Maintenir l'assurance qualité dans le client lourd
- Création des tests logiciels chez le client lourd

Andy Lam

Étudiant de troisième année en génie logiciel à Polytechnique Montréal et diplômé en Sciences Informatiques et Mathématiques au Collège de Maisonneuve. Il possède une bonne maîtrise des langages Java, C++ et Typescript. Ce dernier possède de l'expérience dans le développement d'applications Web. Au cours de son stage chez Desjardins, il a pu développer des connaissances sur le développement d'APIs avec Spring Boot.

Responsabilité principale:

- Développement du client léger

Responsabilités secondaires:

- Planification des sprints avec la répartition des tâches
- Rédaction de tests pour le client léger
- Maintenir l'assurance qualité dans le client léger

Anis Zouatene

Étudiant diplômé en sciences pures et appliquées au Collège de Maisonneuve. En ce moment, il finit sa troisième année en génie logiciel à Polytechnique de Montréal. Il a pu acquérir de l'expérience dans différents langages de programmation comme Python, C++, Java, JavaScript, HTML, CSS pendant son stage et durant ces années d'études. Son stage de 8 mois accompli chez la compagnie d'IBM lui a permis d'approfondir ses compétences en Python et web frontale (HTML et CSS).

Responsabilité principale:

- Développement du client lourd

Responsabilités secondaires:

- Maintenir l'assurance qualité dans le client lourd
- Rédaction de test cases pour le client lourd

Ismail Bakkouri

Diplômé en Sciences Informatiques et Mathématiques au collège de Bois-de-Boulogne et étudiant de troisième année en génie logiciel à Polytechnique de Montréal. Il a acquis beaucoup d'expérience en backend grâce à ses 3 stages, ou il faisait principalement du développement fullstack. Il maîtrise les langages Typescript, Java et C++. Il pourra donc aider au développement du client lourd comme du client léger en plus de ses tâches de développement de serveur.

Responsabilités principales:

- Développement du serveur.
- Aide à l'intégration des fonctionnalités des clients avec le serveur.

Responsabilités secondaires:

- Apporter de l'aide supplémentaire aux tâches des clients (lourd et léger)
- Rédaction des tests chez le serveur

Jason Thai
Étudiant diplômé en sciences pures et appliquées au collège de Bois-de-Boulogne et étudiant de troisième année en génie logiciel à Polytechnique Montréal. Il a pu acquérir de l'expérience sur différents rôles durant ses trois stages. Tout d'abord, son stage chez Matrox lui a permis d'approfondir ses connaissances en REST API et en protocole de communication. Ensuite, son second stage chez GIRO lui a donné des connaissances en interfaces utilisateurs implémentées en C++. Finalement, son stage chez la Banque Nationale du Canada lui a permis d'approfondir ses connaissances sur les pipelines de déploiements. Dans son parcours, il a pu se développer dans nombreux langages de programmation tels que C++, python et C#.

Responsabilités principales:

- Développement du serveur.
- Aide à l'intégration des fonctionnalités des clients avec le serveur.

Responsabilités secondaires:

- Planification des sprints avec la répartition des tâches
- Apporter de l'aide supplémentaire aux tâches des clients (lourd et léger)
- Maintenir l'assurance qualité chez le serveur
- Gestionnaire de temps dans les sprints

Mohammed Ariful Islam
Diplômé en sciences informatiques et mathématiques au collège de Maisonneuve. Présentement étudiant de troisième année en génie logiciel à Polytechnique de Montréal. Il possède une bonne connaissance en des langages orientés objet tels que Java, C++ et Python. Il possède une bonne connaissance en développement de jeu grâce à son stage chez Behaviour Interactive. Lors de ce stage, il a également acquis une petite base en développement de logiciel mobile.

Responsabilité principale:

- Développement du client léger

Responsabilité secondaire:

- Rédaction de tests pour le client léger
- Maintenir l'assurance qualité du client léger

6. Entente contractuelle proposée

La présente entente contractuelle contient l'ensemble des clauses proposées pour une entente entre l'entreprise PolyApps et l'équipe 102.

L'entente contractuelle proposée est un contrat clé en main à prix ferme, car toutes les exigences sont déjà bien définies. L'équipe 102 s'engage à réaliser toutes les exigences essentielles et au moins 50% des exigences souhaitables. Le temps de développement du projet estimé est d'environ 1080 heures-personnes pour une équipe composée de 6 ingénieurs. Les tarifs applicables sont les suivants:

- 110$ pour une heure de développement.

- 145$ pour une heure de gestion de projet.

En considérant que 20% du temps est passé à la gestion de projet, et 80% du temps est passé au développement, le prix de revient du projet sera de 31,320$ pour la gestion et de 95,040$ pour le développement, pour un prix total de 126,360$. L'équipe 102 s'engage à respecter les clauses suivantes:

- Livrer le logiciel Colorimage au plus tard le 6 décembre 2021.
- Respecter les droits de la propriété intellectuelle.

Le client peut demander une modification des exigences à condition que l'équipe le valide, à noter que cela peut entraîner des coûts supplémentaires et/ou des retards de livraison. Le présent contrat ne peut être modifié, à moins d'une entente mutuelle entre les deux parties pour établir un changement des spécifications.