Équipe 107

DrawHub

Plan de projet

Version 2.0

Historique des révisions

Date	Version	Description	Auteurs
2020-10-02	1.0	Première version remise pour évaluation	Bruno Curzi-Laliberté William Glazer-Cavanagh Thomas Houtart Antoine Martin Simon Robatto-Simard Vincent Tessier
2020-12-02	2.0	Version révisée suite aux commentaires de la réponse à l'appel d'offres.	Bruno Curzi-Laliberté William Glazer-Cavanagh Thomas Houtart Antoine Martin Simon Robatto-Simard Vincent Tessier

Table des matières

1. Introduction **4**

2. Énoncé des travaux **4**

2.1. Solution proposée 4

2.2. Hypothèses et contraintes 4

2.3. Biens livrables du projet 4

3. Gestion et suivi de l'avancement **6**

3.1. Gestion des exigences 6

3.2. Contrôle de la qualité 7

3.3. Gestion de risque 8

3.4. Gestion de configuration 11

4. Échéancier du projet **13**

5. Équipe de développement **19**

6. Entente contractuelle proposée **21**

Plan de projet

1. Introduction

Le présent document décrit la demande de proposition de l'application DrawHub réalisée dans le cadre de l'appel d'offres lancé par PolyApps. La section *énoncée des travaux* décrit brièvement notre solution et les livrables. La section *gestion et suivi de l'avancement* décrit nos méthodes de développement ainsi que d'assurance qualité. La section *échéancier du projet* contient nos lots de travail sous forme de calendrier accompagné des jalons importants. La section *équipe de développement* présente les 6 étudiants membres de l'équipe 107 en charge du prototype final. La section *entente contractuelle proposée* explicite le contrat offert par l'équipe 107 en réponse à l'appel d'offre de PolyApps.

2. Énoncé des travaux

2.1. Solution proposée

L'équipe 107 propose une continuation de l'application développée dans le cadre du cours de LOG2990, qui sera utilisable sur Windows 10 et Android accompagné d'un hébergement infonuagique et d'une base de données. Le code source du client léger et du serveur ainsi que l'exécutable du client lourd seront livrés à Polytechnique pour le 2 décembre dans le cadre d'un contrat à terme. Le code source du serveur est en Typescript et sera livré avec des explications des commandes à exécuter pour le partir.

2.2. Hypothèses et contraintes

Ce projet se déroule sur les 15 semaines d'un trimestre universitaire et est réalisé par 6 étudiants de l'école Polytechnique. L'avancement de ce projet est étroitement lié au bon déroulement du trimestre. Ce plan considère donc que le trimestre se déroule sans trouble majeur, c'est-à-dire sans interruption et sans perte de coéquipier.

Le développement de l'application devra être réalisé sur les ordinateurs personnels de l'équipe 107.

Les services infonuagiques utilisés pour ce projet seront des services gratuits ou bien payés par les membres de l'équipe 107.

La communication de l'équipe 107 sera faite à distance et la majorité des rencontres seront virtuelles dû à la situation causée par le COVID-19.

Chaque membre devra consacrer aux alentours de 180 heures au projet au cours de la session.

2.3. Biens livrables du projet

2 octobre 2020

- Spécifications des requis du système (SRS)
- Liste d'exigences

- Document d’architecture logicielle
- Plan de projet (document présent)
- Protocole de communication client-serveur
- Application initiale du client lourd
- Application initiale du client léger
- Application initiale du serveur

2 décembre 2020

- Prototype final client lourd (exécutable .exe)
- Prototype final client léger (code source)
- Code source Typescript du serveur
- Plan de tests
- Résultats des tests

3. Gestion et suivi de l’avancement

3.1. Gestion des exigences

Changement de requis de l’équipe de développement

Lorsqu’un développeur croit qu’une exigence ne pourra être respectée, ce dernier devra suivre les étapes suivantes:

1. Créer un ticket Jira dans le sprint courant comprenant une description du problème.
2. Notifier l’équipe du nouveau ticket.
3. Expliquer la situation à la prochaine rencontre bihebdomadaire.

Un temps sera alloué durant la rencontre pour analyser le problème. Si les exigences doivent être modifiées, l’équipe suivra les étapes suivantes:

1. Rédiger la ou les nouvelles exigences.
2. Au besoin, rédiger une documentation démontrant la nécessité du changement.
3. Notifier le client en lui fournissant la documentation produite.
4. Selon la réponse du client, appliquer ou non le changement.

Changement de requis du client

Lorsque le client demande un changement à une ou plusieurs exigences, l’équipe suivra les étapes suivantes:

1. Organiser une rencontre pour mieux comprendre la demande du client et son besoin.
2. Analyser l’impact du changement sur l’échéancier ainsi que sur l’architecture logicielle courante.
    a. Refuser le changement en cas d’impact trop important.
3. Travailler avec le client sur la rédaction de la ou les nouvelles exigences.
4. Apporter la modification à la documentation.
5. Ajuster les tickets Jira et inclure la solution au moment opportun.

3.2. Contrôle de la qualité

Les méthodes suivantes seront utilisées pour assurer la qualité du système :

- Revue de code par les pairs.
- Linting, soit l'utilisation d'un logiciel qui vérifie l'application du guide de codage.
- Tests utilisateurs.
- Tests unitaires.
- Tests d'intégration.
- Tests système.
- Système de gestion de versions de code source.

Si l'une de ces méthodes détecte un manque de qualité, des actions correctives seront enclenchées. De plus, lorsqu'un développeur jugera durant le développement qu'une exigence ne pourra être respectée selon le délai de l'échéancier, un deuxième développeur devra être assigné sur la même exigence afin de travailler en binôme dans l'optique d'accélérer le processus de développement.

Les **versions du projet** seront contrôlées à l'aide de git. Nous utiliserons GitLab pour héberger notre répertoire de code source. L'équipe de développement utilisera la nomenclature de commit git *"Conventional Commits 1.0.0"* disponible sur leur site éponyme. Voici le format:

<type>[optional scope]: <description>

[optional body]

[optional footer (s)]

Sur Git, les branches de fonctionnalités seront utilisées par les membres de l'équipe pour développer et tester leur code répondant aux exigences. La branche de développement contiendra les fonctionnalités testées et validées. La branche master contiendra les versions liées aux jalons du projet.

Pour expliquer la **révision par les pairs**, nous devons définir notre processus de sprint (lot de travail). À chaque sprint, les tickets de fonctionnalités complétés auront un ticket de validation associé, à compléter lors du prochain sprint. Un deuxième développeur vérifiera alors le code ajouté. Ainsi, les tests unitaires, le code et les pratiques de codages seront vérifiés. Des tests système seront faits (chacun des types de tests est défini plus bas). Au terme de cette validation, le développeur en charge du ticket de développement apportera les modifications jugées nécessaires par le valideur (le temps associé aux potentielles modifications est déjà estimé dans la création de l'échéancier au point 4). Une fois notifiés des changements apportés selon ses commentaires, le valideur peut voir les changements apportés et approuver le nouveau code sur GitLab pour recommencer le processus de validation et intégrer la branche de fonctionnalité à la branche de développement si tout est correct.

Des **tests utilisateurs** seront réalisés avec des participants qui ne sont pas en lien au projet. Les candidats principaux seront des personnes âgées entre 20-30 ans et ayant une familiarité avec les

technologies utilisées, mais n'ayant jamais utilisé ou vu DrawHub. Les commentaires seront recueillis au sprint 8 de l'échéancier. Les commentaires seront analysés et des tickets de modification d'expérience utilisateur seront ajoutés au prochain sprint en conséquence.

Des **tests unitaires** seront écrits pour toutes les parties logiques de l'application tout au long du projet. La couverture de branches et lignes de code doit être de 80% et la couverture des fonctions doit être de 95% au minimum pour chaque fichier avant de pouvoir intégrer la branche de fonctionnalité à la branche de développement. La partie *front end* sera testée lors des tests système.

Des **tests d'intégration** assureront que les modules de l'application seront testés en tant qu'ensemble et non séparément. Ils seront faits durant la partie validation d'un ticket de développement.

Des **tests systèmes** seront effectués lors de chaque fin de ticket afin d'assurer que l'ensemble des requis du document *Software System Requirements* sont respectés. Les exigences seront testées séquentiellement afin d'assurer leur conformité.

3.3. Gestion de risque

La description des risques suit la convention suivante :

- Ampleur : sur une échelle de 1 à 10, 10 étant un risque inadmissible et 5 étant un risque qui mérite l'attention de l'équipe au complet. Cette analyse est basée sur la probabilité d'occurrence du risque multiplié par ses impacts.
- Description : une description du risque ainsi que les problèmes engendrés.
- Impact : échelle définissant la portée du risque
    - C – critique (affecte le projet en entier)
    - E – élevé (affecte une fonctionnalité principale 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 - Fuite de données				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
3	Fuite de données personnelles des utilisateurs ou du système à la suite d'une injection SQL	M	**Sécurité**: L'utilisateur pourrait se procurer des cosmétiques, trophée et changer son score de manière illégitime. Implication au niveau légal si notre application n'est pas sécuritaire.	L'encryption permet de réduire le risque de fuite de mots de passe à 0. La vérification des entrées de texte de l'application pour empêcher les requêtes SQL non désirées.

2 - Dépassement de la limite de crédits nuagique				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
6	Le service nuagique Azure offre un temps de processeur d'une heure par jour, avec possibilité de plus lors de l'utilisation de crédit payant. Le serveur commencera ainsi à consommer nos 100$ de crédits Azure. À l'épuisement de ces 100$, le système attend une méthode de paiement avant de fonctionner.	C	**Disponibilité**: La gestion des sockets et requêtes http ne serait plus traitée par le serveur. L'application ne serait plus utilisable dans ce cas.	Gestion intelligente des crédits étudiants pour les services d'Azure et planification de la consommation du temps de processeur. Chaque développeur sera attribué un temps de processeur selon son besoin. Développement des fonctionnalités en local et non avec le serveur hébergé sur Azure. Utilisation du compte Azure d'un autre membre de l'équipe si nécessaire.

3 - Dépassement de la capacité de la base de données				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
3	Notre base de données a une capacité limitée de 1 Gigaoctet.	C	**Disponibilité**: La création d'un nouveau compte serait impossible. Il ne serait pas possible à de nouveaux utilisateurs de se connecter. De plus, les profils utilisateurs, leaderboard et historiques de message ne seraient plus à jour.	Optimiser nos données pour limiter l'espace qu'elles occupent. Utilisation du compte Azure d'un autre membre de l'équipe si nécessaire.

4 - Différence architecturale des clients				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
1	Les clients utilisent des architectures différentes. Le client lourd utilise une combinaison d'Angular et d'Electron alors que le client léger utilise Android Studio avec Kotlin. La conception se fait donc de manière différente, que ce soit au niveau de son fonctionnement, de ses librairies, de leur format de données, etc.	F	**Fiabilité**: Reproduire une interface utilisateur similaire entre les deux clients pourrait s'avérer impossible dans certains cas. De plus, le développement et fonctionnement de certaines fonctionnalités (EX: dessin svg), pourrait s'avérer très différent. L'expérience utilisateur pourrait différer.	Utiliser des librairies qui reproduisent un comportement similaire ou identique entre les deux clients. Développer la logique des fonctionnalités des clients principalement dans le serveur. Communiquer régulièrement avec les autres équipes de développement

3.4. Gestion de configuration

Résolution d'un problème

La procédure à suivre pour la résolution d'un problème est la suivante:

1. La personne ayant découvert le problème doit d'abord créer un ticket sur la plateforme Jira sous le ticket *Bug Reports* du sprint courant
2. Il communique ensuite le problème au reste de l'équipe ainsi que son degré d'importance qui déterminera le moment où l'équipe se réunira à ce sujet
3. L'équipe analyse le problème et trouve une solution
4. La solution est ajoutée aux tickets Jira

Le titre du ticket doit suivre la nomenclature suivante :

(<Type de ticket><Degré d'importance>) <Titre du problème>

Ex: (PS2) Erreur lors d'envoi de message dans les canaux de communication

Les étapes de résolution sont les suivantes :

- Proposition d'une solution ; nomenclature SP, "Solution Proposition";
- Révision de la solution ; nomenclature SR, "Solution Revision";
- Acceptation de la solution ; nomenclature SA, "Solution Accepted";

Un passage de l'étape de révision à l'étape de proposition doit être envisagé si la solution ne semble pas adéquate.

Les degrés d'importance sont les suivants :

- Degré 1 : Le problème n'est pas urgent. Il devrait être réglé avant la fin du sprint, mais il n'y a aucun impact s'il est réglé après ce sprint.
- Degré 2: Le problème est moyennement urgent. Il doit être réglé avant la fin du sprint, car sinon il empêche le départ d'une tâche du prochain sprint.
- Degré 3: Le problème est urgent. Il doit être réglé le plus tôt possible, car il empêche l'avancement ou la complétion du sprint.

Les étapes de résolution de problème selon le degré d'importance sont les suivantes:

- Dans cas d'un problème de degré 1:
    1. La personne ayant trouvé le problème communique, au minimum, avec la personne connexe se spécialisant dans le domaine lié au problème (Ex: client lourd, client léger, serveur, artefact/organisation).
    2. La personne peut aussi choisir de ne pas communiquer avec l'équipe et de résoudre le problème par lui-même. Cette approche n'est pas recommandée, mais elle est acceptée

si le problème est de degré 1.

- Dans le cas d'un problème de degré 2:
    1. La personne ayant trouvé le problème doit communiquer avec l'équipe de développement ou une personne connexe dans le plus bref délai.
    2. Dans le cas où le problème touche d'autres coéquipiers, celui-ci doit les informer directement et les intégrer dans la discussion.
    3. L'équipe de développement dirige rapidement ses efforts pour trouver une solution au problème.
    4. Lorsqu'un consensus est atteint, une partie des membres de l'équipe doivent implémenter et valider la solution.
    5. Il est recommandé de discuter de ce type de problème entre 3 à 4 personnes.
- Dans le cas d'un problème de degré 3:
    1. La personne ayant trouvé le problème communique avec l'ensemble de l'équipe dans le plus bref délai.
    2. La résolution du problème doit commencer avec la présence de l'équipe entière.
    3. L'équipe dirige immédiatement ses efforts pour trouver une solution au problème.
    4. Lorsqu'un consensus est atteint, une partie des membres de l'équipe doivent implémenter et valider la solution.

4. Échéancier du projet

L'échéancier est découpé en sprints, représentant un lot de travail, d'une semaine sauf exception. L'échéancier a été complété durant le troisième sprint et donc les deux premiers ne sont pas une prévision, mais plutôt un compte rendu. Le temps total du projet est estimé à 1080 heures. Avec 12 sprint ceci représente 90 heures-personne par sprint, donc 15 heures par personne pour notre équipe de six membres. Chaque sprint contient 12 heures de rencontres et 12 heures de gestion d'équipe. Il reste donc 66 heures dédiées au développement qui sont divisées suivant le tableau 1, dont 6 de révision par les pairs non inscrits au tableau. On peut aussi y voir les 3 jalons importants du projet : la première remise, les premiers tests utilisateurs et la deuxième remise.

Tableau 1 : Échéancier

Sprint	Travaux	HP**	Date début	Date fin
1	Lire et commenter la documentation de projet 3 ● Complément pédagogique ● Appel d'offre ● Document de vision ● Aide à la rédaction des artéfacts	30	5 septembre	13 septembre
	Enlever le code non nécessaire du projet 2	5		
	Suivre des tutoriels sur les nouvelles technologies	15		
	S'informer sur les technologies reliées au serveur	10		
2	Choisir la liste d'exigences	10	14 septembre	20 septembre
	Compléter le SRS	20		
	Débuter la documentation pour l'appel d'offre ● Plan de projet ● Protocole de communication ● Architecture logicielle	15		
	Recherche sur les websockets versus les requêtes HTTP	5		
	Tutoriel Kotlin pour l'équipe travaillant sur la tablette	10		

3	Terminer la documentation pour l'appel d'offre ● Plan de projet ● Protocole de communication ● Architecture logicielle ● Correction SRS	25	21 septembre	4 octobre
	Prototype initial client léger	15		
	Prototype initial client lourd	10		
	Prototype initial serveur	10		
Jalon 1 : Remise de l'appel d'offres (2 octobre)				
4	Clavardage intégration et canaux de discussion ● Développement des interfaces (intégré et détaché) ● Enregistrement des données sur la base de données ● Communication entre client lourd et léger	40	5 octobre	11 octobre
	Système et norme de communication définitive client-serveur (route et API du serveur utilisé par le client)	10		
	Refactorisation des prototypes initiaux et implémentation de l'architecture finale	10		
5	Création d'une paire mot-image ● Interface pour la création ● Enregistrement sur la base de données	20	12 octobre	18 octobre
	Dessin automatique du serveur à partir d'une paire mot-image	20		
	Logique de dessin SVG pour mobile (affichage et enregistrement)	20		
6	Mode de jeu (Mêlée générale) ● Interface et option dans le menu principal ● Interface de jeu ● Enregistrement sur la base de données des statistiques de partie	40	19 octobre	25 octobre
	Joueur virtuel (tous la même personnalité) - *début* ● Conseil dans le chat ● Intégration dessin automatique client lourd et client léger	20		

7	Joueur virtuel (tous la même personnalité) - *fin* ● Conseil dans le chat ● Possibilité d'ajout dans le lobby	30	26 octobre	1 Novembre
	Profil utilisateur et historique ● Interface profil pour modifier les paramètres de l'application ● Enregistrement des changements sur la base de données	15		
	Tutoriel, bouton du menu principal pour accéder à une explication visuelle	15		
8	Profil utilisateur et historique - *fin* ● Interface profil pour modifier les paramètres de l'application ● Enregistrement des changements sur la base de données	20	2 novembre	8 novembre
	Effet visuel et sonore (1 groupe d'effet) ● Emote durant la partie	20		
	Test utilisateur ● Recrutement ● Développement de test ● Réalisation des tests ● Récolte de commentaire et suggestion ● Planification des changements nécessaires	20		
Jalon 2 : Fin des exigences essentielles, premiers tests utilisateur et révision plan de projet (12 novembre)				
9	Mode spectateur	30	9 novembre	15 novembre
	Effet visuel et sonore (3 groupes d'effet) ● Musique et animation dans le menu principal ● Animation dans le profil	10		
	Changement nécessaire des tests utilisateurs ● Modification ou ajout d'éléments suite aux commentaires des premiers utilisateurs	10		
	Système anti-triche - *début* ● Logique serveur pour détecter une triche ● Notification du chat pour alerter le tricheur et les autres joueurs	10		

10	Système anti-triche - *fin* ● Logique serveur pour détecter une triche ● Notification du chat pour alerter le tricheur et les autres joueurs	15	16 novembre	22 novembre
	Tableau de classement ● Interface accessible du menu principal ● Logique serveur pour présenter les données intéressantes	15		
	Trophées ● Interface accessible du menu principal ● Logique serveur pour effectuer le suivi du joueur ● Création de trophées intéressants	20		
	Confirmation de la création du compte par courriel (ou recouvrement de mot de passe) - *début* ● Recherche de service de courriel ● Implémentation du service	10		
11	Confirmation de la création du compte par courriel (ou recouvrement de mot de passe) - *fin* ● Recherche de service de courriel ● Implémentation du service	10	23 novembre	29 novembre
	Mode de jeu bataille royale ● Ajout de l'option dans le menu ● Logique serveur pour contrôler le déroulement de la partie ● Interface différente que le mode mêlé générale	30		
	Personnalité des joueurs virtuels	20		
	Documentation ● Modification des artefacts ● Plan de tests ● Résultats de tests	10		
	Travail supplémentaire non prévu	20		

<table>
<tr><td rowspan="2">12*</td><td>Préparation à la présentation orale</td><td>10</td><td rowspan="2">30 novembre</td><td rowspan="2">2 décembre</td></tr>
<tr><td>Documentation
Modification des artefacts
Plan de tests
Résultats de tests</td><td>20</td></tr>
<tr><td colspan="5">Jalon 3 (2 décembre) :
Remise des prototypes finaux (.exe et code source)
Mise à jour des artefacts remis précédemment
Plan de tests
Résultats de tests</td></tr>
<tr><td colspan="5">Jalon 4 : Présentation orale (3 décembre)</td></tr>
</table>

**Certaines heures du sprint 12 ont été transférées au 11e sprint, car il est plus court.*

***HP représente des heures-personnes, les heures cumulées de toutes personnes assignées à la tâche*

Le tableau 2 présente les exigences essentielles et souhaitables choisies pour le projet.

Tableau 2. Exigences essentielles et souhaitables

	Exigences	**Point client lourd**	**Point client léger**
Essentielles	Clavardage - Intégration	4	5
	Clavardage - Canaux de discussion	4	4
	Profil utilisateur et historique	7	2
	Modes de jeu - Mêlée Générale	20	20
	Création d'une paire mot-image (Manuelle I)	15	X
	Personnalité des joueurs virtuels	1 (Tier 1)	X
	Effets visuels et sonores	1 (Tier 1)	2 (Tier 2)
	Tutoriel non interactif	2	2
	Total	54	35
Souhaitables	Personnalité des joueurs virtuels	4 (Tier 3)	0 (Tier 3)
	Effets visuels et sonores	2 (Tier 3)	1 (Tier 3)
	Mode Spectateur	3	2
	Classement	3	2
	Trophées	3	2
	Mêlée générale - Bataille Royale	3	2
	Personnalisation du profil	4 (Tier 1)	4 (Tier 1)
	Système Anti Triche	2 (Tier 1)	2 (Tier 1)
	Confirmation par courriel	2	X
	Total	26	15

5. Équipe de développement

Tableau 3. Équipe de développement

Membre	Plateforme principale	Responsabilité	Expertise
Thomas Houtart	Client lourd	Communication avec serveur et opération logique	Expert en Angular.
Vincent Tessier		Éléments visuels et interactifs en HTML	Expert en HTML et CSS.
Simon Robatto Simard	Client Léger	Création des maquettes pour la cohérence entre les clients et implémentation de l'interface graphique dans le client léger.	Expert en UI/UX.
Antoine Martin		Communication avec serveur et opération logique	Expert en gestion d'équipe et résolution de conflits.
William Glazer-Cavanagh	Serveur	Communication avec les clients	Expert en base de données et intégration serveur - client lourd.
Bruno Curzi-Laliberté		Algorithme et architecture logicielle	Expert en architecture logicielle et patrons de conception.

Tableau 4. Équipe de tests et d'assurance qualité

Membre	Plateforme	Responsabilité
Thomas Houtart	Client lourd	Assurance qualité pour les fonctionnalités sur le client lourd.
Vincent Tessier		Assurance qualité UI/UX pour le client lourd.
Simon Robatto-Simard	Client Léger	Assurance qualité UI/UX pour le client léger et intégrateur avec le serveur et le client léger.
Antoine Martin		Testeur et gestionnaire d'assurance qualité pour l'intégration serveur et client léger.
William Glazer-Cavanagh	Serveur	Testeur et gestionnaire d'assurance qualité pour l'intégration serveur et client lourd.
Bruno Curzi-Laliberté		Écriture des tests d'intégration automatiques sur le serveur et testeur d'intégration multi-plateforme entre les clients.

Note: Chaque membre de l'équipe est responsable du réglage des bogues de l'application à laquelle il est lié (Voir le tableau 3) comme explicité dans la section 3.4.

En plus de leur rôle principal, en cas de problème majeur, certains des membres de l'équipe se sont vu attribuer un rôle secondaire qui permettra d'aider une équipe de développement dans le besoin. Le tableau 3 résume les deux rôles de chacun.

Tableau 5. Rôles principal et secondaire pour chaque membre de l'équipe

	Client lourd	Client léger	Serveur
Antoine Martin	●	X	
Simon Robato-Simard	●	X	
Bruno Curzi-Laliberté		●	X
William Glazer-Cavanagh			X
Vincent Tessier	X		
Thomas Houtart	X		●

X = Rôle principal

● = *Rôle secondaire*

6. Entente contractuelle proposée

L’équipe 107 propose un contrat à terme de 1080 heures pour l'application DrawHub avec une date de livraison prévue le 2 décembre. Ce type de contrat est choisi car il n’y aura aucune modification possible suite au 2 décembre dû à la fin du cours. De plus, l’application ne sera peut-être pas complète à la date d’échéance et les heures établies ne comptent pas le surplus nécessaire pour compléter l’application après cette date. Avec un taux horaire de 125$/h pour un gestionnaire et 100$/h pour un développeur, l’équipe 107 demande une rémunération de 115 200$ pour le budget décrit au tableau 4.

Tableau 6. Budget

Dépense	Heures	Montant
Gestion d’équipe	288 h	36 000 $
Développement de l’application	792 h	79 200 $
Total	1080 h	115 200 $

Il y aura 66 heures par sprint dédiées au développement de DrawHub, avec 6 heures de révision par les pairs, d'où les 792 heures. Les 24 heures restantes du sprint, 288 heures pour tous les sprints, seront dédiées à la gestion d’équipe, qui inclut les rencontres d’équipes, le suivi des tâches et la gestion des tickets Jira.