Équipe 102

Fais-moi un dessin
Plan de projet

Version 1.5

Historique des révisions

Date	Version	Description	Auteur
2021-01-24	1.0	Première version de la section 3	Ming Xiao Yuan
2021-01-30	1.1	Première version de la section 1	Andi Podgorica
2021-02-10	1.2	Première version de la section 4	Vlad Drelciuc
2021-02-14	1.3	Révision du document au complet	Vlad Drelciuc
2021-02-15	1.4	Révision de l'échéancier	Laura Beaudoin et Vlad Drelciuc
2021-04-14	1.5	Corrections suite aux commentaires de l'appel d'offre	Vlad Drelciuc

Table des matières

Plan de projet

1. Introduction

Tout d'abord, ce document contient le plan de projet pour le logiciel *Fais-moi un dessin*. Premièrement, la section 2 décrit en détails le projet ainsi que les biens livrables. Par la suite, la section 3 décrit nos outils de gestion de projet qui seront utilisés tout au long de celui-ci. En outre, la section 4 présente l'échéancier du projet en détail. Pour ce qui est de la section 5, cette dernière présentera l'expérience et le rôle des membres de l'équipe de développement. Enfin, la section 6 contiendra notre proposition d'entente contractuelle en réponse à l'appel d'offres.

2. Énoncé des travaux

2.1. Solution proposée

Notre projet consiste à faire évoluer *PolyDessin*, une application de dessin en ligne. Nous transformons l'application en un jeu en réseau multijoueur (un à quatre joueurs réels ou virtuels) et multi-plateforme (Windows et Android) de type *Fais-moi un dessin*. Le principe est que les joueurs forment des équipes et doivent dessiner et deviner divers mots en une période de temps limité.

Nous proposons ensuite une section de clavardage texte pour faciliter la communication, c'est-à-dire qu'une interface implémentée en client lourd (Windows) et en léger (Android) sera accessible à tout moment dès que le client se connectera au serveur et les joueurs pourront échanger des messages avant et pendant une partie. Par ailleurs, à la création d'une partie, nous proposons aux utilisateurs de pouvoir choisir un joueur virtuel comme coéquipier. Afin de permettre l'identification des utilisateurs, nous proposons un système d'authentification et de gestion des profils complet. De plus, nous proposons de rajouter un tutoriel pour permettre aux nouveaux utilisateurs d'apprendre rapidement à utiliser les nouvelles fonctionnalités de l'application. Les utilisateurs du client lourd auront aussi la possibilité de créer des paires de mots-images que le système utilisera pendant les parties. La grande majorité des fonctionnalités du client lourd seront disponibles aussi sur le client léger Android, mais dans une interface plus conviviale afin d'être adaptées à un écran tactile.

2.2. Hypothèses et contraintes

D'abord, ce plan repose sur nos hypothèses au niveau des compétences de chaque membre de l'équipe (décrites dans la section 5), ainsi qu'au temps que chaque développeur est prêt à investir au projet (environ 180h/personne). L'échéancier se base sur nos estimations du temps nécessaire pour chaque fonctionnalité ainsi qu'aux contraintes dues aux autres cours suivis par les membres de l'équipe. Alors, une semaine pour laquelle nous prévoyons être surchargés d'examens ou de remises de laboratoires devrait prévoir moins de fonctionnalités à implémenter. C'est par exemple le cas durant la semaine du 8 mars (à cause des intra). Nous la compenserons par un effort supplémentaire durant la semaine de relâche (semaine du 1 mars).

Notre échéancier nous contraint à livrer le produit final avant le 19 avril, date où aucune modification de projet ne sera acceptée. Nos ressources humaines sont limitées à 6 membres dans l'équipe. Le budget pour ce projet doit être de 0$ car il est réalisé dans un cadre pédagogique.

2.3. Biens livrables du projet

19 février 2021:

- Remise des artefacts liés à la réponse de l'appel d'offres, soit le SRS, le document d'architecture logicielle, le protocole de communication ainsi que le plan de projet
- Remise d'un prototype de communication pour le client léger et lourd

19 avril 2021:

- Remise d'un produit final fonctionnel (client lourd, léger et serveur)
- Remise du code source du produit final (client lourd, léger et serveur)
- Remise du plan de tests, des résultats de tests logiciels ainsi que de la correction des artefacts du livrable du 19 février (SRS, architecture logicielle, protocole de communication et le document ci-présent)

3. Gestion et suivi de l'avancement

3.1. Gestion des exigences

Afin de mesurer et évaluer les changements aux exigences qui pourraient potentiellement survenir durant le développement du produit, plusieurs mesures sont employées durant le développement afin de minimiser les impacts potentiels. Une des mesures dites est l'utilisation de l'outil de gestion Jira. Cet outil permet à l'équipe entière de créer et/ou modifier librement des tâches jugées nécessaires et permet aux individus de mettre à jour l'avancement de ses tâches. Cette transparence permet donc une meilleure organisation de notre équipe et d'ainsi augmenter la productivité et la conformité aux exigences changeantes.

Le processus de gestion des exigences va comme suit :

- Nous sélectionnons d'abord la liste d'exigences essentielles et souhaitables que nous décidons d'implémenter dès le début du projet. Nous évaluons également la difficulté et le niveau de risque des tâches liées à ces exigences. Nous priorisons de terminer les tâches les plus risquées en début de projet, au cas où des difficultés techniques nous obligeraient à déborder de la planification initiale.
- Notre équipe organise deux à trois rencontres chaque semaine afin que tous les membres de notre équipe soient à jour avec l'avancement du produit. Cette pratique encourage également plus de communication des membres en cas de blocage et permet d'adresser rapidement tout changement des requis. Si une tâche est en retard comparativement à l'échéancier, que ce soit une sous-estimation de temps ou autres, le point sera adressé à la prochaine rencontre hebdomadaire et la production d'une solution deviendra l'ordre du jour prioritaire.
- Nous rencontrons aussi le client chaque semaine pour présenter notre avancement et pour lui faire des démonstrations du produit. Advenant le cas où une modification aux exigences requiert des changements majeurs de notre produit, celle-ci sera traitée en priorité dans la prochaine semaine, et les tâches jugées moins importantes ou risquées pourraient être déplacées au sprint suivant. Nous avons prévu un "buffer" d'environ une semaine en fin de projet pour terminer toutes les petites tâches peu risquées qui auront été dépriorisées durant le projet.
- Nous souhaitons autant que possible finir toutes les fonctionnalités prévues pour le produit final, mais si nous devions couper quelque chose en fin de projet, ce serait dans les fonctionnalités souhaitables de faible pointage.

Les artéfacts corrigés à la remise de la réponse à l'appel d'offre seront mis à jour durant la dernière semaine du projet étant donné qu'ils devraient refléter le plus possible l'architecture et le fonctionnement du produit final. Le plan de tests et les résultats des tests seront entamés plus tôt.

3.2. Contrôle de la qualité

Pour s'assurer et contrôler la qualité des biens livrables du projet, des mesures correctives seront appliquées aux éléments concernés aux moments pertinents.

En premier lieu, les artefacts, rédigés sur l'outil *Google Drive,* seront relus et réévalués par nos membres avant la remise de notre prototype. Chaque membre s'en chargera de relire et corriger des parties rédigées par d'autres membres et ainsi d'en proposer potentiellement des changements constructifs. Après la finalisation du contenu, deux

membres s'en chargeront de corriger tous les artefacts avec l'outil *Antidote*. Cela permet ainsi de corriger toutes fautes grammaticales qui pourraient potentiellement nuire aux plaisirs des lecteurs.

En deuxième lieu, le projet sera testé et évalué tout au long de son développement. Au niveau intra-fonctionnalité, des tests unitaires seront rédigés en parallèle avec le code afin de valider la qualité et comportements voulus de ce dernier. Des tests d'intégration et d'acceptation auront aussi lieu dans les dernières semaines du projet. Notre stratégie de tests sera détaillée davantage dans le *Plan de tests*.

Après la complétion d'une tâche, des *Merge Request* de *Gitlab* sont utilisés afin de valider la fonctionnalité et l'assurance qualité par un autre membre. Seules les branches acceptées seront fusionnées sur la branche de développement. Les branches refusées recevront des commentaires sur les éléments à retravailler. Une fois les corrections apportées, le responsable du *Merge Request* demandera une réévaluation au réviseur assigné. Ce processus pourrait prendre plus d'une itération avant que le résultat soit satisfaisant. Nous nous assurerons que les *Merge Request* soient révisés le plus tôt possible, idéalement en moins de 12h. Les principaux réviseurs seront Benjamin pour le code du serveur et du client lourd et Justin pour le code du client léger.

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 - Chute de la performance de l'application				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
3	L'application réagit lentement aux actions de l'utilisateur.	C	Performance	Minimiser le nombre de requêtes entre les clients et le serveur. Stocker localement les assets les plus lourds pour éviter les temps de chargement et économiser la bande passante. Utilisation de plusieurs threads lorsque possible pour éviter que la vue ne bloque. Utilisation d'indicateurs visuels pour l'utilisateur lors du chargement.

2 - Confusion provenant de l'interface utilisateur

Ampleur	Description	Impact	Facteurs	Stratégie de gestion
2	L'utilisateur est confus durant l'utilisation de l'application	M	Utilisabilité	Planifier l'apparence UI/UX lors de l'étape de conception. Faire valider le design et la cohérence d'utilisation par le client à chaque semaine. Accepter que le design ne puisse plaire à tout le monde.

3 - Serveur lent ou inutilisable				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
6	L'application n'est pas capable d'acquérir des données à partir du serveur.	C	Performance / Stabilité	Tester le serveur au fur et à mesure de l'introduction des nouvelles fonctionnalités. Utiliser les outils de débogage de Google Cloud Functions en cas de panne pour identifier la source du problème. Au besoin, ré-envoyer les requêtes qui ne donnent pas de réponse.

4 - Base de données lente ou inutilisable				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
4	L'application perd temporairement accès à la base de données ou celle-ci timeout.	E	Performance / Utilisabilité	Minimiser le nombre de requêtes entre les clients et le serveur. Utiliser une base de données de type "scalable" et redondante comme Firebase.

3.4. Gestion de configuration

Afin de revoir les problèmes et/ou les changements apportés, des mesures de gestion de configuration y sont apportées tout au long du développement du projet.

Premièrement, chaque membre met à jour son journal de travail relié à chaque ticket qui lui est associé sur la plateforme *Jira*. Chaque entrée du journal inclut le nombre de temps (m/h) que l'individu a passé sur la tâche et une brève description de l'avancement ou des difficultés rencontrées. Par ailleurs, si une anomalie est détectée dans l'application, un membre de l'équipe pourra le soulever durant une de nos rencontres hebdomadaires et le gestionnaire de produit ouvrira un ticket Jira. Ce ticket détaillera la manière de reproduire le bug et sera assigné à la personne qui a programmé la fonctionnalité défectueuse. Si la résolution du bug est urgente, il est possible qu'une autre tâche moins risquée soit dépriorisée et poussée au sprint suivant.

Deuxièmement, l'environnement *GitLab* est utilisé pour gérer les différentes versions et changements de notre répertoire de projet. Chaque membre doit obligatoirement ajouter un commentaire pertinent à chaque changement qui sont poussé sur leurs branches afin de faciliter la gestion des progressions. Lors de l'achèvement d'une fonctionnalité, la branche concernée, si acceptée, est fusionnée sur la branche de développement pour ainsi garder en tout temps un code fonctionnel. Nous utiliserons une adaptation de *Gitflow* avec les branches *master*, *dev*, *feature*/ et *hotfix*/. Nous produirons une *release* sur la branche *master* à la fin de chaque sprint. Ces releases auront le tag "sprint-x", avec x le numéro du sprint terminé.

Enfin, les artefacts ont également une historique de révision incluant les changements, la version, la date et les auteurs. La première version des artefacts débutera avec 1.0 et augmentera de 0.1 jusqu'à un maximum de 0.9 chaque modification importante. (Version 2.0 pour la version suivie de 1.9)

4. Échéancier du projet

Légende : W = Web, M = Mobile, S = Serveur

SPRINT 0						
Tâche	**Effort total**	**Effort Web**	**Effort Mobile**	**Effort Serveur**	**Date de début**	**Date de fin**
Liste des exigences	10	4	3	3	2021-01-26	2021-02-04
Spécification des requis du système (SRS)	40	13	13	14	2021-01-26	2021-02-04
Plan de projet	20	7	7	6	2021-01-29	2021-02-19
Document d'architecture logicielle	35	12	11	12	2021-02-01	2021-02-19
Protocole de communication	30	10	10	10	2021-02-13	2021-02-19
Prototype - interface de messagerie instantanée (W+M+S)	60	20	20	20	2021-02-05	2021-02-19
Prototype - Compte, authentification et menus principaux (W+M+S)	60	20	20	20	2021-02-05	2021-02-19
Révision du SRS à la suite de la rétroaction	5	1	3	1	2021-02-16	2021-02-19

Livrable #1 : réponse à l'appel d'offre (2021-02-19)						
TOTAL SPRINT	**260**	**87**	**87**	**86**	**2021-01-26**	**2021-02-19**

SPRINT 1						
Tâche	**Effort total**	**Effort Web**	**Effort Mobile**	**Effort Serveur**	**Date de début**	**Date de fin**
Gestion des compte, authentification et menus principaux (W+M+S)	30	10	10	10	2021-02-19	2021-02-26
Profil utilisateur et historique (W+M+S)	60	20	20	20	2021-02-19	2021-03-04
Canevas et interface de dessin (M)	15		15		2021-02-19	2021-02-26
Implémentation du crayon (M)	20		20		2021-02-26	2021-03-04
Implémentation de l'efface (M)	20		20		2021-02-26	2021-03-04
Implémentation de la grille (M)	15		15		2021-02-26	2021-03-04
Mode de jeu - Classique (W+S)	65	45		20	2021-02-26	2021-03-04
Plan de tests				10		
Jalon #1 : fin du sprint 1 (2021-03-04)						

TOTAL SPRINT	**235**	**75**	**100**	**60**	**2021-02-19**	**2021-03-04**

SPRINT 2						
Tâche	**Effort total**	**Effort Web**	**Effort Mobile**	**Effort Serveur**	**Date de début**	**Date de fin**
Implémentation du undo-redo (M)	20		20		2021-03-04	2021-03-08
Implémentation des options de couleur (M)	20		20		2021-03-08	2021-03-11
Mode de jeu - Classique (M)	65		65		2021-03-04	2021-03-17
Mode de jeu - Sprint solo (W)	20	20			2021-03-04	2021-03-08
Choix de mots (W)	5	5			2021-03-04	2021-03-08
Lobby (W+S)	25	15		10	2021-03-08	2021-03-12
Création d'une paire mot-image - Manuelle 1 (W)	50	50			2021-03-08	2021-03-17
Jalon #2 : fin du sprint 2 (2021-03-17)						
TOTAL SPRINT	**205**	**90**	**105**	**10**	**2021-03-04**	**2021-03-17**

SPRINT 3						
Tâche	**Effort total**	**Effort Web**	**Effort Mobile**	**Effort Serveur**	**Date de début**	**Date de fin**

Mode de jeu - Sprint solo (M)	20		20		2021-03-17	2021-03-22
Choix de mots (M)	5		5		2021-03-17	2021-03-22
Lobby (M)	15		15		2021-03-17	2021-03-24
Clavardage - intégration (W+M)	35	20	15		2021-03-24	2021-04-02
Clavardage - canaux de discussion (W+M)	40	20	20		2021-03-24	2021-04-02
Personnalité des joueurs virtuels (W+S)	40	10		30	2021-03-17	2021-03-24
Classement des joueurs (W+S)	15	10		5	2021-03-17	2021-03-24
Badge / trophées (W+S)	20	10		10	2021-03-17	2021-03-24
Tutoriel (W)	5	5			2021-03-17	2021-03-24
Musique d'arrière-plan (W)	5	5			2021-03-24	2021-04-02
Effets visuels et sonores (W)	15	15			2021-03-24	2021-04-02
Jalon #3 : fin du sprint 3 (2021-04-02)						
TOTAL SPRINT	**215**	**95**	**75**	**45**	**2021-03-17**	**2021-04-02**

SPRINT 4						
Tâche	**Effort total**	**Effort Web**	**Effort Mobile**	**Effort Serveur**	**Date de début**	**Date de fin**
Personnalité des joueurs virtuels (M)	10		10		2021-04-02	2021-04-09
Classement des joueurs (M)	10		10		2021-04-02	2021-04-09
Tutoriel (M)	5		5		2021-04-02	2021-04-09
Musique d'arrière-plan (M)	5		5		2021-04-02	2021-04-09
Effets visuels et sonores (M)	15		15		2021-04-09	2021-04-16
Progression / niveaux (W+M+S)	35	15	15	5	2021-04-09	2021-04-16
Monnaie virtuelle (W+S)	20	15		5	2021-04-02	2021-04-09
Magasin (W+S)	25	20		5	2021-04-04	2021-04-16
Évaluation des tests				10		
Jalon #4 : fin du sprint 4 (2021-04-16)						
Livrable #2 : remise du projet final (2021-04-19)						
TOTAL SPRINT	**135**	**50**	**50**	**25**	**2021-04-02**	**2021-04-16**

RÉSUMÉ				
	Effort total	**Effort Web**	**Effort Mobile**	**Effort Serveur**
Sprint 0	260	87	87	86
Sprint 1	235	75	100	60
Sprint 2	205	90	105	10
Sprint 3	215	95	75	45
Sprint 4	135	50	50	25
TOTAL	1050	407	427	206

5. Équipe de développement

5.1 Beaudoin, Laura
Expérience :
- Étudiante en troisième année au baccalauréat en génie logiciel à l'école Polytechnique de Montréal
- Stagiaire en assurance qualité chez Bombardier
- Maîtrise des langages : C++, Python, C#, Java et Matlab
- Expérience de développement web (Angular, TypeScript, Javascript, HTML, CSS)

Responsabilités :
- Membre de l'équipe de développement du client léger
- En charge de l'UX/UI du client léger

5.2 Boucher-Charest, Benjamin
Expérience :
- Étudiant en troisième année au baccalauréat en génie logiciel à l'école Polytechnique de Montréal
- Stagiaire programmeur-analyste chez GIRO
- Maîtrise des langages : C#, C++, C, Java, Scala, Chisel, Python
- Expérience de développement web (Angular, TypeScript, HTML, CSS)
- Expériences avec Google Cloud Platform & Firebase

Responsabilités :
- Membre de l'équipe de développement du client lourd
- En charge d'intégrer les fonctionnalités du serveur avec le client lourd
- En charge du QA (tests + revue de code) pour le client lourd et le serveur

5.3 Caisse, Justin
Expérience :
- Étudiant en troisième année au baccalauréat en génie logiciel à l'école Polytechnique de Montréal
- Membre de l'équipe Computer Vision de la division Robomaster dans la société technique PolySTAR
- Maîtrise des langages : C++, Java, C#, Python
- Expérience de développement web (Angular, TypeScript, HTML, CSS)

Responsabilités :

- Membre de l'équipe de développement du client léger
- En charge d'intégrer les fonctionnalités du serveur avec le client léger
- En charge du QA (tests + revue de code) pour le client léger

5.4 Drelciuc, Vlad
Expérience :

- Étudiant en troisième année au baccalauréat en génie logiciel à l'école Polytechnique de Montréal
- Stagiaire en développement à la Banque Nationale
- Maîtrise des langages : Java, JavaScript, Python
- Expérience de développement web (Angular, TypeScript, HTML, CSS)
- Expérience avec Firebase

Responsabilités :

- Membre de l'équipe de développement du serveur
- Gestionnaire du projet

5.5 Podgorica, Andi
Expérience :

- Étudiant en troisième année au baccalauréat en génie logiciel à l'école Polytechnique de Montréal
- Maîtrise des langages : JavaScript, C++, Java
- Expérience de développement web (Angular, TypeScript, HTML, CSS)
- Membre de l'équipe Partenariat dans la société technique PolySTAR

Responsabilités :

- Membre de l'équipe de développement du client léger
- En charge du développement des services du client léger

5.6 Yuan, Ming Xiao
Expérience :

- Étudiant en troisième année au baccalauréat en génie logiciel à l'école Polytechnique de Montréal
- Membre de l'équipe Computer Vision de la division Robomaster dans la société technique PolySTAR
- Stagiaire R&D chez OSSimTech
- Maîtrise des langages : C, C++, Java, Python, JavaScript
- Expérience de développement web (Angular, TypeScript, HTML, CSS)

Responsabilités :

- Membre de l'équipe de développement du client lourd
- En charge de l'UX/UI du client lourd

6. Entente contractuelle proposée

Nous proposerons une entente de contrat de type livraison clé en main. Nous livrerons le projet avec toutes les exigences essentielles incluses ainsi que la moitié des exigences souhaitables incluses pour le 19 avril 2021. Les exigences sont détaillées dans le document de spécification des requis au point 3. Le temps requis pour le projet est estimé à 1050 heures :

- 140 heures seront admises à la rédaction des artéfacts par le gestionnaire de projet. Ces heures seront rémunérées à un taux horaire de 125 $/h pour un sous-total de 17 500 $ (CAD).
- 910 heures seront admises au développement de l'application par les développeurs. Ces heures seront rémunérées à un taux horaire de 100 $/h pour un sous-total de 91 000 $ (CAD).

Le coût total du projet sera donc de 108 500 $ (CAD).