Les dindons

PAINseau

Plan de projet

Version 1.2

Historique des révisions

Date	Version	Description	Auteur
2021-02-02	1.0	Première rédaction du plan de projet	Équipe 107
2021-02-09	1.1	Première révision en équipe du plan de projet	Équipe 107
2021-02-16	1.2	Modifications finales pour la remise de la réponse à l'appel d'offres	Équipe 107

Table des matières

Plan de projet

1. Introduction

Ce document présente la planification autour du développement du logiciel *Fais-moi un dessin*. La section 2 se porte sur la solution proposée, les hypothèses et les contraintes sur lesquelles repose notre plan, et les biens livrables se rattachant au projet. Par la suite, la section 3 décrit la gestion des exigences, le contrôle de la qualité, la gestion de risque et de configuration. La section 4 présente l'échéancier à suivre tout au long, et puis la section 5 décrit brièvement l'expertise des membres de l'équipe. Finalement, la section 6 suggère une entente contractuelle en réponse à l'appel d'offres.

2. Énoncé des travaux

2.1. Solution proposée

La solution que nous proposons est *PainSeau*, une application de dessin et un jeu ayant pour but de deviner ce qu'un dessin illustre. Ce logiciel permettra aux usagers de se connecter à un serveur, et d'organiser des parties pour jouer entre eux, ou avec des joueurs virtuels. Ainsi, une interface de communication sera implémentée afin de permettre l'échange de messages entre les joueurs durant et hors d'une partie.

*Pain*Seau sera séparée en deux clients, soit un client lourd et un client léger. Le client lourd doit être fonctionnel sur un PC avec l'environnement Windows 10, puis le client léger, sur une tablette Android. Peu importe la plateforme de jeu, les joueurs doivent se créer un profil et s'y connecter afin d'accéder à *PainSeau*. Ces derniers seront redirigés vers un menu principal qui affichera les parties en cours et en attente de joueurs. De cette page, ils pourront également créer et configurer une partie, ou bien joindre une partie. Ils pourront notamment sélectionner le mode de jeu désiré, soit le mode *Classique* et possiblement le mode *Sprint solo*. Par ailleurs, au moment de la connexion du joueur au serveur, la fonctionnalité de clavardage lui sera accessible. Puis, dans le cas où l'usager ne maîtriserait pas l'utilisation et les fonctionnalités de l'application, un tutoriel lui sera disponible en tout temps. Pour plus de détails sur les exigences et les caractéristiques de l'application, veuillez-vous référer au document de spécification des requis du système (SRS).

Finalement, *PainSeau* sera développée à partir du logiciel *PolyDessin*, et contiendra donc certaines de ses fonctionnalités. Par exemple, les joueurs pourront accéder à l'outil du crayon et ses paramètres, à l'outil de l'efface, la grille ainsi que la fonctionnalité annuler-refaire.

2.2. Hypothèses et contraintes

2.2.1 Ressources humaines

L'équipe de développement est formée de six étudiants, chacun devant fournir un minimum de six heures de travail par semaine, et la charge de travail requise pour la livraison du projet est de 1008 heures-personnes. Puisque ce projet introduit de nouvelles technologies, les membres de l'équipe nécessiteront un temps d'introduction et d'apprentissage pour s'équiper avec une connaissance de base.

2.2.2 Équipement

Pour le développement du client léger, il nous sera nécessaire d'installer une machine virtuelle ou bien un émulateur d'une tablette Android respectant certaines spécifications, afin de s'assurer qu'il se comporte comme attendu. Pour le développement du client lourd, cela va s'effectuer sur un PC régulier. Pour l'hébergement de notre serveur, nous avions décidé d'utiliser *Heroku* et pour la base de données, nous avons repris *MongoDB*; son l'hébergement s'effectuera sur *MongoDB Atlas.*

2.2.3 Échéancier

- Réponse à l'appel d'offres: 19 février;

- Produit final:

- Suivi chaque semaine

Un premier ensemble d'artefacts doit être remis en guise de réponse à l'appel d'offres le 19 février. Puis, une seconde remise d'artefacts s'effectuera le 19 avril avec la livraison du produit final. Afin de garantir le bon déroulement du projet, une rencontre hebdomadaire avec les responsables du cours est tenue. Des rencontres entre les membres de l'équipe sont également planifiées pour suivre de près le progrès de tous et de toutes. La section 2.3 décrit plus en détail les biens livrables.

2.3. Biens livrables du projet

- *19 février 2021* - Réponse à l'appel d'offres
    - Document d'architecture logicielle;
    - Plan de projet;
    - Protocole de communication;
    - Liste d'exigences;
    - Spécification des requis du système (SRS).

- *19 avril 2021* - Produit final
    - Mise à jour des artefacts remis précédemment ;
    - Plan de test logiciels;
    - Résultat de tests logiciels.

3. Gestion et suivi de l'avancement

3.1. Gestion des exigences

L'équipe a établi des exigences initiales dans le SRS qui seront transmises au client le 19 février 2021. Ces exigences vont guider le développement des fonctionnalités. En effet, les tâches et les sous-tâches seront créées à partir de ces exigences. Dans le cas où les exigences venaient à changer au cours du développement, une demande de changement devra être envoyée au client et être approuvée avant de pouvoir modifier le SRS. La plateforme utilisée pour la gestion du projet est Jira ce qui permettra de faire le suivi des exigences à l'aide des tâches et de facilement pouvoir modifier des tâches existantes ou encore ajouter des tâches de correction.

3.2. Contrôle de la qualité

Plusieurs mesures ont été prises afin d'assurer la qualité des biens livrables. En ce qui concerne les artefacts, leurs contenues seront discutées en équipe pendant les rencontres avant d'assigner les tâches de rédaction aux membres de l'équipe. Plus encore, au moins une révision générale d'équipe sera réalisée pour chaque artefact avant la remise.

Pour ce qui est de la qualité du code, la fonctionnalité de *Merge request* sera utilisée avant de fusionner vers dev ou master. Le code devra respecter trois conditions avant d'être fusionné. Il devra être révisé par les pairs c'est-à-dire au moins 2 autres personnes autres que l'auteur du code. Le code devra également passer l'analyse TSLint dans le client lourd et il devra également être dépourvu d'avertissement dans le client léger. Il devra aussi avoir une couverture de test d'au moins 90%. Un code ne respectant pas ces trois conditions devra être corrigé avant d'être fusionné.

Dans le cas où un bogue est détecté concernant une tâche considérée terminée, une tâche de bogue sera créée et assignée à un membre de l'équipe qui se chargera de faire une branche hotfix connectée à la branche master pour la correction du bogue.

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> - Nouvelles technologies				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
5	L'équipe va travailler avec de nouvelles technologies avec lesquelles elle a peu d'expérience en particulier pour le client léger où il faudra utiliser Android Studio et Kotlin. La technologie pour la communication client-serveur (Socket) pour le système de messagerie est également une nouvelle technologie à apprendre.	E	Temps nécessaire à l'implémentation des fonctionnalités Nombre de bogues dans le système Maintenabilité du code	Commencer le plus tôt possible à se familiariser avec les nouvelles technologies notamment en réalisant le prototype. Aussi, il faudra communiquer avec les membres de l'équipe le plus rapidement possible lorsqu'un bloquant survient.

<2> - Cohérence entre le client lourd et le client léger				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
3	Au fur et à mesure du développement, le client lourd et le client léger pourraient prendre de différentes directions en termes de fonctionnalité ou encore d'interface graphique.	C	Qualité des biens livrables Temps passé à corriger les problèmes	Il faudra assurer une bonne communication entre les développeurs du client lourd et du client léger principalement durant les rencontres d'équipe ainsi que sur Discord. Aussi, il sera nécessaire de tester l'application de façon hebdomadaire afin d'assurer une cohérence ainsi que le respect des exigences.

<3> - Absence d'un coéquipier				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
3	Un coéquipier pourrait ne plus pouvoir travailler sur le projet de façon temporaire ou permanente à cause d'un événement imprévu.	M	Temps de travail par coéquipier	Les membres de l'équipe devront constamment mettre à jour les tâches sur Jira pour ainsi pouvoir plus facilement suivre le travail de chaque coéquipier.Aussi, si une telle situation se produit, il faudra se rencontrer le plus rapidement possible en équipe afin de discuter des changements à apporter pour assurer la relève du travail du coéquipier manquant.

<4> - Charge de travail d'une fonctionnalité				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
5	L'équipe pourrait sous-estimer le temps et l'effort nécessaire pour implémenter une fonctionnalité. Cela pourrait créer un retard dans l'échéancier.	M	Temps nécessaire à l'implémentation des fonctionnalités Nombres de fonctionnalités implémentées	Il faudra toujours choisir une durée estimée à la hausse afin de garder une marge de manœuvre. Il faudra également prioriser les tâches que l'équipe considère les plus complexes.

<5> - Intégration des différentes parties				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
6	Lors de l'intégration entre le client lourd, le client léger et le serveur, il pourrait y avoir de nombreux bogues et de fonctionnalités déficientes.	M	Temps nécessaire à corriger les problèmes	Il faudra tester chaque semaine l'intégration des différentes parties du système. Il faudra également se donner une marge d'au moins une semaine avant la remise des livrables afin de s'assurer du bon fonctionnement du système.

<6> - Travail d’équipe efficace				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
3	Lors des différentes rédactions et implémentations de code, l’équipe pourrait avoir tendance à travailler chacun de leur côté et un manque de communication pourrait arriver et nuire à l’intégration et au développement du projet.	E	Qualité des biens livrables Cohérence des interfaces	En plus des rencontres d'équipes, il faudra organiser des séances de révision du code. Il faudra que tout le monde ait une idée commune de ce qui se passe et de ce qui a été réalisé pour ne pas prendre du retard et que tout le monde soit à jour.

3.4. Gestion de configuration

Lorsqu'un problème est découvert sur la branche *master*, il faudra tout d'abord créer une tâche de type problème sur Jira, assigner une priorité et une personne à ce bogue. Il sera également important de détailler le problème et de décrire la procédure pour reproduire le bogue en description au besoin. La personne responsable de travailler sur le problème devra créer une branche nommée *hotfix-[nom du problème]* attachée à la branche *master*.

En ce qui concerne les artefacts, le nom des artefacts sera tout simplement le nom du document suivi du numéro de notre équipe comme par exemple, le nom du SRS sera "SRS-equipe107". Pour ce qui est de la numérotation, l'équipe rédige une version initiale qui sera considérée comme la version 1.0. Ensuite de cela, pour chaque légère modification, l'auteur des changements devra ajouter un incrément de 0.1 dans la version ainsi que documenter les changements dans l'historique des versions. Dans le cas où une révision majeure de l'artefact est nécessaire, un incrément de 1.0 sera fait.

4. Échéancier du projet

Dans le cadre de ce projet, nous respectons un ratio de 42h par personne par crédit. Comme ce cours a une valeur de 4 crédits et que notre équipe est composée de 6 membres, le nombre d'heures total correspond à 42h x 4 crédits x 6 membres = 1008 heures-personnes. Notre échéancier sera donc séparé en lots et en jalons, où chaque ligne de notre tableau correspond à un lot de travail et où les jalons marquent la fin de chaque phase (surligné en vert), signifiant que les objectifs ont été atteints et qu'il est temps de passer à la phase suivante.

Notre échéancier est séparé en deux phases principales : la réponse à l'appel d'offres et la remise du produit final, qui sera divisé en plusieurs sprints.

Phase	Tâches/fonctionnalités	Effort estimé (heures-personnes)	Date de début	Date de fin
Réponse à l'appel d'offres		200h	2021-01-19	2021-02-19
	Première rédaction du document SRS (Spécifications des Requis du Système)	30h	2021-01-19	2021-02-01
	Révision du SRS en équipe	25h	2021-02-02	2021-02-02
	Modifications finales du SRS suite aux révisions et remise	10h	2021-02-02	2021-02-04
	Première rédaction du plan de projet	10h	2021-02-04	2021-02-08
	Révision du plan de projet en équipe	5h	2021-02-08	2021-02-08
	Modifications finales du plan de projet suite aux révisions	5h	2021-02-08	2021-02-15
	Première rédaction du document d'architecture	10h	2021-02-04	2021-02-11

	logicielle			
	Révision du document d'architecture logicielle en équipe	10h	2021-02-11	2021-02-11
	Modifications finales du document d'architecture logicielle suite aux révisions	5h	2021-02-11	2021-02-15
	Première rédaction du protocole de communication	10h	2021-02-04	2021-02-11
	Révision du protocole de communication en équipe	5h	2021-02-11	2021-02-11
	Modifications finales du protocole de communication suite aux révisions	5h	2021-02-11	2021-02-15
	Première version du prototype du client léger	15h	2021-02-04	2021-02-11
	Finaliser l'interface et l'envoi de données du client léger	10h	2021-02-11	2021-02-15
	Première version du prototype du client lourd	15h	2021-02-04	2021-02-11
	Finaliser l'interface et l'envoi de données du client léger	10h	2021-02-11	2021-02-15
	Première version du prototype du serveur	15h	2021-02-04	2021-02-11
	Finaliser la gestion de la réception et de l'envoi des données du serveur	5h	2021-02-11	2021-02-15
Remise de la réponse à l'appel d'offres				2021-02-19
Produit final		808h	2021-02-19	2021-04-19

Sprint 1		260h	2021-02-19	2021-03-12
	Client lourd : Clavardage - Intégration (mode intégré)	15h	2021-02-19	2021-02-26
	Client léger : Clavardage - Intégration	15h	2021-02-19	2021-02-26
	Client lourd : Clavardage - Canaux de discussion (historique)	10h	2021-02-19	2021-02-26
	Client léger : Clavardage - Canaux de discussion (historique)	10h	2021-02-19	2021-02-26
	Serveur : Sauvegarder et gérer l'historique des messages des canaux	25h	2021-02-19	2021-02-26
	Client lourd : Profil utilisateur et historique (statistiques)	20h	2021-02-26	2021-03-05
	Client léger : Profil utilisateur et historique (statistiques)	20h	2021-02-26	2021-03-05
	Serveur : Gérer et sauvegarder les informations publiques et privées des utilisateurs	25h	2021-02-26	2021-03-05
	Client lourd : Modes de jeu (Classique)	60h	2021-02-26	2021-03-12
	Client léger : Modes de jeu (Classique)	60h	2021-02-26	2021-03-12
Fin de sprint 1				2021-03-12
Sprint 2		185h	2021-03-12	2021-03-26

	Client lourd : Création d'une paire mot-image (Assistée I)	55h	2021-03-12	2021-03-26
	Serveur : Sauvegarde et gestion des paires de mot-image	40h	2021-03-12	2021-03-26
	Client lourd : Personnalité des joueurs virtuels	25h	2021-03-12	2021-03-19
	Client lourd : Thèmes de couleurs différentes	15h	2021-03-12	2021-03-19
	Client léger : Thèmes de couleurs différentes	15h	2021-03-12	2021-03-19
	Client lourd : Système de points	10h	2021-03-19	2021-03-26
	Client léger : Système de points	10h	2021-03-19	2021-03-26
	Serveur : Système de points	15h	2021-03-19	2021-03-26
Fin de sprint 2				2021-03-26
Sprint 3		363h	2021-03-26	2021-04-19
	Client lourd : Leaderboard	10h	2021-03-26	2021-04-02
	Client léger : Leaderboard	10h	2021-03-26	2021-04-02
	Serveur : Leaderboard	15h	2021-03-26	2021-04-02
	Client lourd : Album de dessins	20h	2021-03-26	2021-04-02
	Serveur : Album de dessins	30h	2021-03-26	2021-04-02
	Client lourd : Création de paire mot-image (Assistée II)	50h	2021-03-26	2021-04-09
	Client lourd : Langues (allemande)	5h	2021-03-26	2021-04-02
	Client lourd : Thèmes de couleurs différentes	15h	2021-04-02	2021-04-09

	(5)			
	Client léger : Thèmes de couleurs différentes (5)	15h	2021-04-02	2021-04-09
	Client lourd : Système de niveaux	20h	2021-04-02	2021-04-09
	Client léger : Système de niveaux	20h	2021-04-02	2021-04-09
	Serveur : Système de niveaux	23h	2021-04-02	2021-04-09
	Client léger : Profil utilisateur et historique (historique détaillé)	20h	2021-04-02	2021-04-09
	Serveur : Profil utilisateur et historique (historique détaillé)	20h	2021-04-02	2021-04-09
	Client lourd : Langues	10h	2021-04-09	2021-04-16
	Client léger : Langues	10h	2021-04-09	2021-04-16
	Client lourd : Effets visuels et sonores (3)	15h	2021-04-09	2021-04-16
	Client léger : Effets visuels et sonores (3)	15h	2021-04-09	2021-04-16
	Client lourd : Tutoriel (Interactif)	20h	2021-04-09	2021-04-16
	Client léger : Tutoriel (Interactif)	20h	2021-04-09	2021-04-16
Fin de sprint 3				2021-04-19
Remise du produit final				2021-04-19

5. Équipe de développement

Alice Gong

Possédant de l'intérêt pour le UI et le UX, Alice est à sa troisième année en génie logiciel à l'École Polytechnique Montréal. Elle considère avoir une expertise particulière dans le développement *frontend*, surtout suite à son projet intégrateur de deuxième année (développement d'une application web) et à son stage l'été dernier (amélioration d'interface utilisateur, planification avec une équipe de conception web, etc). Elle est notamment familière avec Typescript, Angular, HTML/CSS, MongoDB, Node.js et Express. Bien sûr, ses cours académiques l'ont rendue habile avec des technologies comme C++, Java, Python, Javascript et SQL.

Responsabilité principale : Développement du client lourd
Autres responsabilités :

- Planification des sprints et répartition des tâches
- Aide supplémentaire dans les tâches liées au UI et au UX

Oliver Jean

Étudiant de troisième année en génie logiciel à Polytechnique Montréal. Suivant ses projets d'école, il maîtrise de nombreux langages de programmation comme C++, Java, Python, Javascript et SQL. Oliver a également réalisé le projet intégrateur de deuxième année, où il s'est familiarisé avec des technologies liées à la programmation frontale, comme Typescript, HTML/CSS et le cadriciel Angular. Enfin, il se qualifie comme étant une personne organisée qui apprend rapidement. Il a surtout remarqué ça suite à son dernier stage, où il a vite appris de nouvelles techniques en gestion de projet.

Responsabilité principale : Développement du client léger
Autres responsabilités :

- Planification des sprints et répartition des tâches
- Gestionnaire du temps (autant dans les rencontres d'équipe que dans les sprints)

Charles Jiang

Étudiant de troisième année en génie logiciel à Polytechnique Montréal. Grâce aux divers travaux de son baccalauréat, il se considère très à l'aise avec les technologies suivantes: C++, Java, Python, Javascript, Typescript, Angular, Node.js, SQL et Git. Lors de son stage chez Matrox, il a conçu une application console avec interface graphique en utilisant le cadriciel .Net ainsi que le langage C#. C'est pourquoi, Charles se considère habile autant dans le développement *backend* que *frontend*.

Responsabilité principale : Développement du serveur et de la base de données
Autres responsabilités :

- Planification des sprints et répartition des tâches
- Aide supplémentaire auprès des tâches du côté client (lourd et léger)

Yuhan Li

Elle en est à sa troisième année en génie logiciel à l'École Polytechnique Montréal. Elle est habile dans l'utilisation du cadriciel Angular et du langage Typescript, puisqu'elle a implémenté le côté client tout au long de son projet intégrateur de deuxième année. De plus, elle possède de bonnes connaissances en programmation *full stack* dans le domaine du développement web, surtout suite à son stage en développement web l'été dernier. Suite à ce stage, Yuhan a acquis de l'expérience dans la conception d'interface ergonomique, sachant qu'elle a abordé de près les concepts de UI et UX.

Responsabilité principale : Développement du client lourd
Autres responsabilités :

- Planification des sprints et répartition des tâches
- Aide supplémentaire dans les tâches liées au UI et au UX

Nu Chan Nhien Ton

Étudiante de 3e année en génie logiciel à Polytechnique Montréal. Nu Chan Nhien a déjà été stagiaire à la Banque Nationale où elle était chargée de concevoir et développer des solutions d'intégrations et de consommations de

données au moyen de Jobs ETL. Elle considère donc avoir de la facilité à apprendre de nouvelles technologies puisqu'elle a dû se familiariser avec de nouveaux logiciels tout au long de son stage. Enfin, elle est familière avec C++, Java, Python, JavaScript, TypeScript, Angular, MongoDB, Node.js et Express grâce aux différents projets d'école.

Responsabilité : Développement du serveur et de la base de données
Autres responsabilités :

- Planification des sprints et répartition des tâches
- Secrétaire et animatrice principale lors des réunions d'équipe
- Aide supplémentaire auprès des tâches du côté client (lourd et léger)

Yanis Toubal
Étant rendu à sa troisième année universitaire en génie logiciel à Polytechnique Montréal, Yanis possède de fortes habiletés en résolution de problèmes algorithmiques et il maîtrise les langages comme C/C++, Java et Python. Il a également de l'expérience en tant que tuteur à Polytechnique, ce qui le rend habile à expliquer et comprendre des notions selon les besoins d'autrui. Enfin, lors de son stage au Ministère de la Justice du Québec, il a travaillé en tant que développeur full stack sur un projet web en Angular selon les principes SCRUM. Puis, dans ce même stage, il a travaillé sur un projet ASP.Net MVC en programmation frontale.

Responsabilité principale : Développement du client léger
Autres responsabilités :

- Planification des sprints et répartition des tâches
- Aide dans la revue et le réusinage (*refactoring*) global du code

6. Entente contractuelle proposée

Notre équipe propose une entente contractuelle de type clé en main puisque ce type de contrat correspond le mieux avec notre projet. En effet, ce type de contrat a un prix fixe déterminé lors de l'entente contractuelle et un produit final doit être remis au client à la fin. Les avantages sont qu'on procure une assurance des coûts finaux et que le promoteur n'a pas à avoir besoin de faire beaucoup de suivis des travaux. Les désavantages sont qu'il faut avoir une connaissance exacte des demandes pour pouvoir déterminer toutes les exigences et les spécifications d'avance et que des frais et du temps sont associés à cette longue étape.

1. Les exigences du projet sont détaillées dans la section 3 du document de spécification des requis du système (SRS).

    1.1 Le projet va répondre à l'entièreté des exigences essentielles.

    1.2 Le projet va répondre à au moins 50% des exigences souhaitées.

2. Le projet sera livré au plus tard le 19 avril 2021 à 23h59.

3. Le nombre total d'heures estimées pour le projet est de 1008h.

    3.1. 180 heures sont prévues pour la production des artefacts.

    3.2. 828 heures sont prévues pour le développement logiciel.

4. L'équipe sera composée de 6 ingénieurs qui vont prendre le rôle de développeur et de gestionnaire.

    4.1 Le taux horaire par gestionnaire pour la conception des artefacts est de 125$ par heure.

    4.2 Le taux horaire par programmeur pour le développement logiciel est de 100$ par heure.

    4.3 Le prix total du projet est de 105 300 $.