Équipe 6

Fais-moi un dessin
Plan de projet

Version 3.0

Historique des révisions

Date	Version	Description	Auteur
2019-09-08	1.0	Rédaction de la partie 1, 2 et 3	Mohamed Amine Kamal
2019-09-08	1.0	Rédaction de la partie 4, 5	Georges Louis
2019-09-09	1.0	Correction de la partie 4 Rédaction de la partie 6	Georges Louis
2019-09-11	1.9	Correction de la partie 4	Georges Louis
2019-09-24	2.0	Échéancier révisé	Sami T. Arar
2019-11-23	3.0	Mise-à-jour du plan de projet	Équipe 6

Table des matières

Plan de projet

1. Introduction

Ce document contient le plan de projet. Le plan de projet est le mécanisme principal pour établir les objectifs du projet. Ce dernier est divisé en plusieurs sections, nous avons dans un premier temps l'énoncé des travaux qui explique brièvement la solution proposée, les hypothèses et contraintes ainsi que les biens livrables du projet. Dans un deuxième temps, nous avons la gestion et le suivi de l'avancement. Dans cette section, nous détaillons la gestion des exigences, le contrôle de la qualité, la gestion des risques et la gestion de configuration. Dans un troisième temps, nous avons l'échéancier du projet où nous décrivons l'effort nécessaire pour les principaux lots de travail ainsi que les dates de début et fin associées à chacun des lots et des livrables. Dans un quatrième temps, nous avons l'équipe de développement où nous décrivons l'expertise et les responsabilités de chaque membre de l'équipe. Finalement, nous avons l'entente contractuelle proposée où nous suggérons une entente qui répond à l'appel d'offre.

2. Énoncé des travaux

2.1. Solution proposée

Notre mandat était de transformer le logiciel PolyPaint en un jeu « Fais-moi un dessin » avec la possibilité de jouer en réseau. Notre solution est constituée de trois composantes: le client léger (une application Android), le client lourd (une application basée sur le logiciel fourni (PolyPaint)) et un serveur permettant différentes actions, dont le jeu en réseau. Le but du jeu est de faire un dessin et de le faire deviner par son coéquipier. Différents modes de jeu seront jouables. Le jeu sera donc accessible depuis 2 plateformes différentes, un ordinateur (Windows) et une application Android avec des fonctionnalités en moins.

L'application permettra à des utilisateurs de tablettes Android et à des utilisateurs d'ordinateurs Windows de pouvoir jouer à différents modes de jeu. Ces modes de jeux seront basés sur le dessin en temps réel d'un mot à deviner et la capacité à celui qui regarde le dessin se dessiner de pouvoir le deviner dans des contraintes de temps et de tentatives permises. Ces dessins seront soit dessinés par un autre joueur ou par un joueur virtuel. L'application offrira aussi une interface pour créer des jeux, mais seulement sur Windows. Ces jeux seront alors utilisés lors de vraies parties. L'application offrira aussi une fonctionnalité de gestion de profil et de clavardage.

2.2. Hypothèses et contraintes

Ce plan repose sur plusieurs hypothèses, la coopération entre tous les membres de l'équipe. Les membres de l'équipe doivent être responsables et faire un effort pour respecter l'échéancier au meilleur de leur capacité. Le projet présente aussi plusieurs contraintes. Tout d'abord, nous sommes une équipe de 5, sachant qu'il est possible d'être en équipe de 6, notre charge de travail est alors plus élevée. Ensuite, aucun membre de l'équipe ne possède des connaissances de la technologie WPF, cela rend la tâche du développement du client lourd plus difficile. Finalement, tous les membres de l'équipe ont d'autres cours qui se font parallèlement à ce cours ci-présent. Cela rendra plus difficile le maintien de l'équilibre au niveau de la charge de travail avec les autres cours.

Une première contrainte serait la vitesse à laquelle le serveur pourra délivrer des données aux 2 clients. Le serveur sera une importante partie de notre application. On ne doit donc pas le surcharger de logique pour qu'il ne soit pas ralenti. De plus, il ne faut absolument pas que le serveur puisse planter. Si le serveur arrive à planter, nos 2 clients deviennent alors inutilisables. Une autre contrainte est le temps alloué au développement de toutes les fonctionnalités de l'application. Il faudra s'organiser de façon qu'on ne soit pas en retard.

On suppose aussi que le serveur basé sur le « cloud » Heroku est assez puissant pour gérer toutes les communications avec les clients qui y seront connectés, et qu'il n'y aura pas de latence ou d'arrêt de service.

2.3. Biens livrables du projet

- Réponse à l'appel d'offres - 27 septembre 2019
    - Plan de projet

- SRS
- Liste d'exigences
- Document d'architecture logicielle
- Protocole de communication
- Prototype de communication client lourd-serveur
- Prototype de communication client léger-serveur

- Produit final - 28 novembre 2019
  - Mise à jour des artefacts remis précédemment
  - Plan de tests
  - Résultats de tests
  - Code source et exécutable du produit (client lourd, client léger, serveur)

3. Gestion et suivi de l'avancement

3.1. Gestion des exigences

3.1.1 Mesurer les changements aux exigences du produit

Pour mesurer les changements aux exigences du produit, un membre de l'équipe va tout d'abord estimer les composants de code (modules, classes, lignes de codes) affectés par le changement en question. Ensuite, l'estimation devra être confirmé par un ou plusieurs coéquipiers. Finalement, il faudra transformer l'estimation trouver en nombres d'heures à allouer au changement.

Des changements pourront être apportés aux exigences. Le client pourrait changer d'idée sur une des fonctionnalités de l'application à tout moment. Aussi, lorsque le développement d'une des exigences aura débuté à être développé, le client pourrait s'apercevoir que ce n'était pas ça qu'il voulait. De plus, lorsque la date d'échéance sera proche, il serait nécessaire d'ajuster la sélection d'exigences pour celles qui pourront être terminées à temps et de laisser tomber celles qui ne pourront pas être terminées pour se concentrer sur l'atteinte des points exigés par le client. Il faudra alors aller piger dans les « exigences souhaitables ».

3.1.2 Rapporter les changements aux exigences du produit

Après avoir mesuré le nombre d'heures allouées, il faudra créer les tâches nécessaires sur Redmine en allouant les heures définies lors de la phase de mesure et ajouter les nouvelles tâches au sprint suivant. Ces tâches seront alors associées à des branches sur Gitlab.

3.1.3 Contrôler les changements aux exigences du produit

Le contrôle des changements devient alors simple, puisqu'il faut tout simplement faire le suivi sur Redmine et sur Gitlab. Si nécessaire, nous pourrions aussi créer des rencontres d'équipes pour faire un suivi plus approfondi.

3.2. Contrôle de la qualité

Plusieurs méthodes seront utilisées pour contrôler la qualité du code. Tout d'abord, chaque demande sur Redmine concernant du code sera associée à une branche sur Gitlab. Lorsque la demande est complétée, l'auteur de la branche demandera un *pull request* sur la branche Master, le code sera révisé par les autres membres de l'équipe. De cette façon tout le code sera contrôlé. La branche master étant protégé aucun membre de l'équipe ne pourra directement (push) du code sur cette branche, et, il sera obligatoire de passer par le *pull request*. Ensuite, nous utiliserons du *CI*/*CD* (*continuous integration* / *continuous deployment*) pour tester chaque commit. De cette façon, nous serons assurés que tous les tests passent en tout temps. Finalement, nous aurons des rencontres régulières où nous pourrons revoir sommairement l'avancement du projet et donc nous assurer de la qualité de code en même temps.

Lorsqu'une défaillance est trouvée, nous devrions aller sur Redmine pour ouvrir une demande « Anomalie » et de l'assigner à la personne responsable qui pourrait résoudre le bogue technique. De plus, il est important de ne jamais

« push » sur la branche « master » de git. Les fonctionnalités sont développées sur des branches séparées. Lorsque des tests ont été effectués sur cette branche, cette dernière pourra alors être « merge » sur le master. Le master doit toujours avoir une version stable du produit.

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 - Mauvais fonctionnement du serveur lors de la remise				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
5	Mauvais fonctionnement du serveur lors de la remise malgré le bon fonctionnement sur notre machine local.	C	Toutes les fonctionnalités du système qui interagissent avec le serveur.	Tester le serveur à plusieurs reprises pour s'assurer que celui-ci marche lors de l'évaluation. Prévoir une copie du serveur en local pour prévenir toute nuisance du serveur sur le « cloud ».

2 - Retard par rapport à l'échéancier initial				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
7	Prise de retard sur l'avancement des fonctionnalités, par rapport à ce qui avait été prévu initialement dans l'échéancier.	M	Heures consacrées à la réalisation du système.	Réévaluation de l'échéancier et de la source du problème en équipe et ajout de plus d'heures de travail pour compenser le retard.

3 - Absence d'un membre de l'équipe...				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
5	Membre de l'équipe s'absente lors d'un cours ou d'une rencontre.	M	Heures consacrées à la réalisation du système.	Un travail non réalisé ne doit pas être pris à la légère. Dans ce cas, si cela se reproduit à plusieurs reprises, il en faudra informer le professeur ou la personne à charge pour effectuer une intervention dans l'équipe.

3.4. Gestion de configuration

Le processus qui permet de soumettre, revoir et disposer des problèmes et des changements se fera en plusieurs étapes.

3.4.1 Problèmes techniques

Premièrement, lors de l'apparition d'un problème, celui-ci devra être reporté sur Redmine avec comme tracker « anomalie ». Deuxièmement, le membre de l'équipe qui aura détecter le problème devra notifier les autres membres de l'équipe sur le système de chat défini par l'équipe (Discord dans notre cas). Troisièmement, lorsque l'équipe est notifiée, la tâche sera assignée à un ou plusieurs membres de l'équipe. Finalement, le ou les membres de l'équipe responsable de résoudre le problème créeront une branche sur Gitlab du nom bugfix/<nom du bug>. Un suivi pourra alors être fait pour assurer la résolution du problème.

3.4.2 Problèmes non techniques

Les problèmes non techniques seront résolus lors des rencontres d'équipes. Les rencontres auront lieu, au minimum, deux fois par semaine.

3.4.3 Nomenclature

Les artefacts du projet (SRS, protocole de communication, plan de projet, architecture logicielle et les résultats des tests) garderont les mêmes nom suivis de leurs versions respectives. Les demandes sur Redmine auront la nomenclature suivante : <Nom de la demande>. Les branches sur Gitlab auront la nomenclature suivante: <Type de la demande>/<ID de la demande>-<Nom de la demande>. Les *commits* auront la nomenclature suivante: <ID de la demande> <Description>.

4. Échéancier du projet

Tout d'abord on a regardé le travail qui est à remettre pour le 27 septembre, c'est la réponse à l'appel d'offres. Cette réponse constitue de plusieurs parties:

- Architecture logicielle.
- Plan de projet.
- SRS.
- Protocole de communication.

Ensuite, on a calculé le nombre d'heures qu'il fallait mettre pour délivrer le projet avant la date limite. Chaque point est équivalent à 4 heures de travail. Ainsi, on devrait atteindre 900 heures-personne puisqu'on est une équipe de 5 personnes.

Dates de début et de fin des principaux lots de travail:

Semaine 1 (29 août - 4 septembre)	
Tâches	**Effort** (en heure-personne)
Lecture de l'appel d'offres	8
Lecture du document de vision	8
Lecture du complément pédagogique	8
Total	**24**

Semaine 2 (5 septembre - 11 septembre)	
Tâches	**Effort** (en heure-personne)
Rédaction du SRS	20
Rédaction du plan de projet	15
Total	**35**

Semaine 3 (12 septembre - 18 septembre)	
Tâches	**Effort** (en heure-personne)
Début de l'architecture logicielle	12
Début du prototype (Client léger)	12
Début du prototype (Client lourd)	12
Familiarisation avec les technologies	20
Serveur	15
Total	**71**

Semaine 4 (19 septembre - 25 septembre)	
Tâches	**Effort** (en heure-personne)
Finir prototype (Client léger)	20
Finir prototype (Client lourd)	20
Finir serveur	20
Finir l'architecture logicielle	10
Finir SRS deuxième version	10
Finir plan de projet	10
Finir document protocole de communication	10
Total	**100**

Semaine 5 (26 septembre - 2 octobre)	
Tâches	**Effort** (en heure-personne)
Clavardage - Intégration (Client lourd)	20
Clavardage - Canaux de discussion (Client lourd)	20
Clavardage - Intégration (Client léger) -Avec notifications	20
Clavardage - Canaux de discussion (Client léger)	16
Tests sur fonctionnalités	6
Total	**82**

Semaine 6 (3 octobre - 9 octobre)	
Tâches	**Effort** (en heure-personne)
Construction d'un jeu (Client lourd - Serveur) -Manuelle I	23
Construction d'un jeu (Client lourd - Serveur) -Manuelle II	23
Mode de jeu – Classique (Client lourd) -Lobby -Groupe	20
Mode de jeu – Classique (Client léger) -Lobby -Groupe	20
Tests sur fonctionnalités	6
Total	**92**

Semaine 7 (10 octobre - 16 octobre)	
Tâches	**Effort** (en heure-personne)
Construction d'un jeu (Client lourd - Serveur) -Assistée I	23
Construction d'un jeu (Client lourd - Serveur) -Assistée II	23
Mode de jeu – Classique (Client lourd) -Joueur virtuel intégration	20
Mode de jeu – Classique (Client léger) -Joueur virtuel intégration	20
Tests sur fonctionnalités	6
Total	**92**

Semaine 8 (17 octobre - 23 octobre)	
Tâches	**Effort** (en heure-personne)
Mode de jeu – Classique (Client lourd) -Mode classique complété	32
Mode de jeu – Classique (Client léger) - Mode classique complété	40
Personnalité des joueurs virtuels (Serveur)	4
Tests sur fonctionnalités	6
Total	**76**

Semaine 9 (24 octobre - 30 octobre)	
Tâches	**Effort** (en heure-personne)
Mode de jeu – Sprint (Client lourd) -Sprint solo complété	10
Mode de jeu – Sprint (Client lourd) -Sprint coopératif complété	6
Mode de jeu – Sprint (Client léger) -Sprint solo complété	10
Mode de jeu – Sprint (Client léger) -Sprint coopératif complété	6
Profil utilisateur et historique (Client lourd) -Profil utilisateur fonctionnel -Statistique d'utilisation du jeu	20
Profil utilisateur et historique (Client léger) -Profil utilisateur fonctionnel -Statistique d'utilisation du jeu	16
Tests sur fonctionnalités	6
Total	**74**

Semaine 10 (31 octobre - 6 novembre)	
Tâches	**Effort** (en heure-personne)
Profil utilisateur et historique (Client lourd) -Profil utilisateur fonctionnel -Statistique d'utilisation du jeu	8
Profil utilisateur et historique (Client léger) -Profil utilisateur fonctionnel -Statistique d'utilisation du jeu	4
Profil utilisateur – Avatar (Client lourd) -Téléverser une image personnalisée pour avatar	12
Profil utilisateur – Avatar (Client léger) -Téléverser une image personnalisée pour avatar	8
Profil utilisateur – Désactiver un compte (Client lourd)	8
Profil utilisateur – Désactiver un compte (Client léger)	4
Effet sonores (Client lourd) -Effet de particule -Son de rapprochement de fin de tour -Son quand l'utilisateur clique	12
Effet sonores (Client léger) -Effet de particule -Son de rapprochement de fin de tour -Son quand l'utilisateur « clique »	12
Tests sur fonctionnalités	6
Total	**74**

Semaine 11 (7 novembre - 13 novembre)	
Tâches	**Effort** (en heure-personne)
Tutoriel (Client lourd) -Interactif	20
Tutoriel (Client léger) -Interactif	20
Mode de jeu – Défi (Client lourd) -fonctionnel	16
Mode de jeu – Défi (Client léger) -fonctionnel	12
Tests sur fonctionnalités	6
Total	**74**

Semaine 12 (14 novembre - 20 novembre)	
Tâches	**Effort** (en heure-personne)
Finir les fonctionnalités non terminées du client lourd.	20
Finir les fonctionnalités non terminées du client léger.	20
Tests sur fonctionnalités	6
Total	**46**

Semaine 13 (21 novembre - 27 novembre)	
Tâches	**Effort** (en heure-personne)
Fin plan de tests	10
Fin résultats de tests	10
Correction du code	25
Correction des artefacts	15
Total	**60**

5. Équipe de développement

Georges Louis:

Étudiant en 3e année, il a déjà fait plusieurs projets universitaires, soit les projets 1 et 2 du génie logiciel. Il connaît plusieurs langages de programmation comme C, C++, Java, ainsi que des langages de développement web comme le HTML/CSS/JavaScript/Angular/Express/NodeJS. Il maîtrise le langage de base de données PostgreSQL. Il aime travailler en équipe et il donne de son mieux pour s'améliorer et être bénéficiaire pour son équipe. Ses responsabilités principales durant le projet seront:

- Client lourd

Bassam Ajam:

Étudiant en 3e année, Bassam a fait son stage à "la Ville de Montréal" durant 2 sessions consécutives (Hiver 2019 et Été 2019). Il maîtrise HTML/JavaScript/CSS, car il a travaillé dans le développement web, et possède une bonne expérience de la gestion d'équipe selon l'approche Agile. Il a également d'autres capacités techniques en C/C++, Java, Angular, NodeJs, MongoDB etc. Ses connaissances seront utiles dans le développement du projet et dans la gestion d'équipe. Ses responsabilités principales durant le projet seront:

- Client lourd

Syphax Ait-Yahia:

Étudiant en 3e année, Syphax a de l'expérience en JavaScript/ Angular/ Express/ NodeJS/ MongoDB/ C++/ PYTHON. Il a fait un stage en tant que développeur *full stack* en été 2019. Son expérience sera utile dans notre projet vu qu'on utilisera la même technologie pour construire notre application. Son expérience en C++ nous sera aussi très utile. Ses responsabilités principales durant le projet seront:
Ses responsabilités seront:

- Serveur
- Client lourd

Sami Tamer Arar:

Étudiant en 3e année qui maîtrise l'orienté objet en C++ et en Java. Il a fait un stage en tant qu'Analyste d'affaires, donc maîtrise également la gestion de projet. Il a un niveau intermédiaire en C# et en Kotlin. Ses responsabilités principales durant le projet seront:

- Client léger
- Gestion des échéanciers

Mohamed-Amine Kamal:

Étudiant en 4e année. Maîtrise l'orienté objet. Langages de programmation pertinent : TypeScript, Java, Kotlin. Ses responsabilités principales durant le projet seront:

- Serveur
- Client léger

6. Entente contractuelle proposée

Le type de contrat qu'on a choisi est le contrat livraison clé en main - Prix ferme. Ce dernier a été choisi puisque nous, en tant que contracteur, on est obligé de livrer le produit final qui sera l'application "*Fais-moi un dessin"* le 28 novembre 2019 à 23h59. Donc, on a une date limite, ainsi qu'une liste d'exigences à respecter dans notre réponse à l'appel d'offre.

Le paiement final sera émis lorsque notre client accepte le produit final livré par l'équipe. Le détail du calcul est le suivant:

Équipe de 5 développeurs.
On estime que le temps de gestion du projet prendra 20% de notre temps.
La charge de travail est estimé à 900 heures pour le 28 novembre.
1 heure = 100$ pour le développeur.
1 heure = 125$ pour le gestionnaire de projet.
180 heures de gestion de projet et 720 heures de développement.

On aura 180 * 125 + 720 * 100 = 92 500$.

Cette somme peut être modifiée durant la durée de développement du projet dans le cas où le client modifie les requis. Mais, il faudra tenir compte de la nature des requis ajoutés. En effet, si les nouveaux requis causent un délai, le client devrait accepter et accorder le temps nécessaire à l'équipe de développement pour finir le travail des requis demandés. De plus, le client devrait s'acquitter de la nouvelle somme d'argent demandé.