Équipe 106

Projet 3 : Fais-moi un dessin
Plan de projet

Version 1.5

Historique des révisions

Date	Version	Description	Auteur
2020-01-28	1.0	Échéancier du projet, sections 1, 2.1, 2.3, 3.1 et 3.2	Équipe 6
2020-01-29	1.1	Sections 3.3 et 3.4	Nicole Joyal
2020-01-31	1.2	Section 2.2	Amar Ghaly
2020-02-02	1.4	Sections 4, 5 et 6	Karima Salem
2020-04-13	1.5	Sections 2.2, 3 et 6	Karima Salem

Table des matières

Plan de projet

1. Introduction

Le présent rapport illustre la planification du projet d'application logicielle "Fais-moi un dessin". Tout d'abord, nous débuterons par l'énoncé des travaux. Cette section comporte la solution proposée, les hypothèses et contraintes et les biens livrables du projet. Ensuite, il sera question de la section de gestion et suivi de l'avancement. Dans cette section, nous aborderons les mécanismes que nous emploierons par rapport à la gestion des exigences, le contrôle de la qualité, la gestion de risque et la gestion de configuration. De plus, nous dresserons un échéancier en mettant de l'avant les jalons importants du projet. Pour finir, nous présenterons les membres de l'équipe et l'entente contractuelle.

2. Énoncé des travaux

2.1. Solution proposée

En réponse à l'appel d'offres soumis par la Polytechnique de Montréal, nous proposons la création de l'application de jeu "Fais-moi un dessin". La solution proposée représente le programme PolyPaint bonifié. En effet, nous y ajouterons des éléments plus interactifs.

En premier lieu, l'utilisateur peut s'enregistrer pour accéder au service ce qui mènera à la création d'un profil unique. Le profil contient des informations personnelles et pertinentes à l'utilisateur. Afin d'accéder à une partie de jeu, l'utilisateur a accès à un lobby où il pourra voir et choisir les parties qu'il a envie d'intégrer. Ensuite, plusieurs options s'offrent à l'utilisateur. Ce dernier peut créer un jeu pour faire deviner d'autres joueurs en fonction des paramètres qu'il aura choisis. Il existe différents modes de jeu, notamment le mode mêlé général, solo et coopératif. Le mode de jeu mêlé général, soit de l'anglais *free-for-all*, consiste en l'affrontement de plusieurs joueurs pour deviner un maximum de mots. Quant au mode de jeu solo, l'utilisateur doit deviner un maximum de mots dans un laps de temps prédéfini. Finalement, le mode de jeu coopératif offre la possibilité aux utilisateurs de collaborer pour deviner des mots. En second lieu, l'application logicielle offrira à l'utilisateur l'option de clavarder. Lors de la création d'une partie, nous y retrouverons un canal exclusif aux joueurs de la partie. Aussi, l'utilisateur aura l'option de se joindre à des canaux de discussion et même d'en créer. Pour conclure, afin d'augmenter l'interactivité de l'application, celle-ci sera accessible sur des tablettes Android et manipulée par les fonctions de l'écran tactile.

2.2. Hypothèses et contraintes

Le succès de ce projet repose sur diverses hypothèses. Nous supposons que les artéfacts de ce projet et les dates de l'échéancier ne sont pas finaux. En effet, malgré la rigueur employée lors de cette planification, les artéfacts pourront être modifiés lorsque la vision du client change. Aussi, il est possible qu'au cours des prochains mois certains imprévus surviennent et que les dates de l'échéancier ne soient pas complètement respectées. Nous décidons de promouvoir la valeur de flexibilité afin d'obtenir un produit logiciel à l'image du client. Ensuite, nous nous attendons à ce que chaque membre de l'équipe fournisse une contribution équitable quant à la réalisation de ce projet.

De plus, nous retrouvons plusieurs contraintes imposées dans le cadre de ce projet. Tout d'abord, en ce qui concerne l'équipement, tout membre de l'équipe devra être muni de son propre ordinateur portable. Il s'agit de l'équipement de base qui est requis pour le bon fonctionnement des rencontres et du travail d'équipe. Ensuite, pour les membres de l'équipe s'occupant davantage du client léger, ces derniers devront se servir d'une tablette électronique Android. Quant aux ressources humaines, l'équipe devra fournir un total de 1080 heures de travail. Cela correspond à environ 180 heures par membre. Finalement, les contraintes technologiques à respecter sont : l'usage des langages C# et WPF pour le client lourd, l'usage des langages Kotlin et Java pour le client léger et l'usage du cadriciel NodeJs et du langage Typescript pour le serveur. Pour maintenir une certaine cohérence, nous avons imposé trois environnements de développement intégrés (IDE) pour le client lourd, le client léger et le serveur qui sont respectivement Visual Studio, Android Studio et Visual Studio Code.

Enfin, il est important d'énoncer les diverses contraintes temporelles imposées par le déroulement de la session. Les examens de mi-session et les finaux ont un impact sur la performance, la motivation ainsi que le temps qui peut être accordé à la réalisation du projet. Ainsi, il est important de planifier en conséquence. Pour ce faire, une grande partie du travail peut être réalisée durant la semaine de lecture, car l'équipe ne possède pas d'évaluations dans leurs autres cours. Ce serait aussi le moment idéal pour rattraper tout retard qui a pu être accumulé durant la période des examens intrasemestriels. De plus, pour éviter toute perturbation, il est impératif de planifier la finalisation des fonctionnalités deux semaines avant la remise. De cette façon, une marge de temps suffisante est octroyée afin de réaliser le plan de tests, les tests et les correctifs des bogues.

2.3. Biens livrables du projet

Au cours de ce projet logiciel, nous aurons plusieurs artéfacts à produire. Nous retrouvons deux dates de livraison. Tout d'abord, la réponse à l'appel d'offres est prévue pour le 7 février 2020.

Cette livraison sera composée des artéfacts suivants:

- la liste des exigences,
- les spécifications des requis du système,
- le plan de projet (le présent document),
- le protocole de communication,
- le document d'architecture logicielle.

Afin de consolider la solution proposée, nous produirons des prototypes du client léger et lourd et du serveur. Ensuite, à la date du 13 avril 2020, nous livrerons les nouvelles versions des artéfacts mentionnés ci-dessus. De plus, nous produirons le plan de tests et les résultats des tests. Finalement, le produit logiciel final sera livré, soit le code source et les exécutables du client léger, du client lourd et du serveur.

3. Gestion et suivi de l'avancement

3.1. Gestion des exigences

Les exigences propres au projet sont énumérées dans le document de spécification des requis, autrement appelé le SRS. Au cas d'un changement aux requis de notre produit, il faudra évaluer son impact à la gestion du projet.

Dans le cas d'un empêchement du respect d'une exigence (complexité, parties dépendantes), nous devrons nous réunir en équipe afin d'analyser la situation. Nous procéderons premièrement à une analyse des différentes solutions pour rectifier l'empêchement. Au cas où l'empêchement ne serait pas rectifiable, nous allons procéder à une réorganisation de tâches. Nous avertirons ensuite le client et modifierons nos documents en conséquence. Le cas où un empêchement serait rectifiable, nous procéderons à une réévaluation de l'effort consacré aux tâches reliées et à une réévaluation de la planification.

Dans toutes les situations, nous devons faire des ajustements aux documents d'appel. En premier, il faudra réviser l'échéancier établi dans la section 4, afin de maintenir un équilibre dans la distribution des jalons. Ensuite, dans Redmine, il faudra déplacer les demandes pour prendre en compte la modification des exigences (ex. déplacer les tâches liées à une exigence dans une semaine postdatée). Redmine aidera à planifier les tâches à accomplir chaque semaine. De plus, il permettra de suivre de près la progression et l'avancement du projet. Finalement, si l'exigence est modifiée ou retirée, on mettra à jour le SRS ainsi que la liste d'exigences, toujours avec l'approbation du client.

3.2. Contrôle de la qualité

Le contrôle de la qualité des biens livrables sera effectué de manière continuelle. Pour s'assurer de l'intégrité du système, la réalisation de tests unitaire sera priorisée. Suite à l'implémentation de toutes les fonctionnalités, un plan de test permettra d'évaluer la facilité d'utilisation de l'application et donner une idée de l'expérience utilisateur. L'équipe s'engage à utiliser un contrôle des versions efficace avec Git, avec des messages de modification pertinents. Chaque branche sera associée à une fonctionnalité. Il y aura au moins un autre membre de l'équipe qui révise la nouvelle fonctionnalité à intégrer (en faisant référence au SRS) avant qu'elle soit intégrée dans la version stable.

Les membres de l'équipe s'engagent à faire une démonstration du travail effectué à la fin de chaque semaine. Ceci nous permettra de maintenir une transparence à travers les différentes fonctionnalités, notamment les exigences reliées à chacune. Si une exigence d'un bien livrable est oubliée ou problématique, nous allons ajouter les tâches nécessaires afin de l'intégrer pleinement. À la suite de cela, une présentation de la fonctionnalité entière sera requise encore afin de s'assurer que toutes les exigences sont toujours respectées.

3.3. Gestion de risque

1 - Google Cloud Platform est en panne ou manque de performance				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
3	Le service d'hébergement de notre serveur, Google Cloud Platform, subit un problème.	C	Système non disponible, performance réduite	Fournir une notification aux utilisateurs de la panne ou du ralentissement. Tester les nouvelles fonctionnalités avec un serveur local en attendant le rétablissement de la connexion.

2 - Interface implémentée sans la fonctionnalité associée				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
2	Un membre de l'équipe implémente des vignettes interactives sans fonctionnalité.	M	Page de l'application non fonctionnelle	Il faudra cacher l'outil de l'interface en attendant que la fonctionnalité associée soit implémentée. Il faut prendre l'habitude de ne dévoiler un composant que lorsque sa fonction ait été entièrement implémentée.

3 - Membre de l'équipe quitte ou a un empêchement important				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
1	Un membre de l'équipe doit s'absenter entièrement ou temporairement, mettant de côté toutes ses tâches (prévues et/ou commencées).	M	Implémentation ralentie	L'équipe devra se réunir afin de redistribuer les tâches prévues du coéquipier non disponible. Nous devrons nous organiser pour terminer les tâches incomplètes avant l'échéance de celles-ci.

4 - Suppression des données de la base de données				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
3	Un membre de l'équipe supprime des données de la base de données, créant possiblement des erreurs d'authentification ou la perte de données utilisateur et/ou historique de clavardage.	E	Application plante, non-respect des exigences	Faire des sauvegardes fréquentes de l'état de la base de données. Avec cela, nous pourrons restaurer les données perdues entièrement sinon complètement.

5 - Mauvaise gestion d'erreurs				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
8	Les membres de l'équipe ne gèrent pas assez les cas particuliers d'utilisation, causant ainsi des erreurs non attendues.	F	Application plante, fonctionnalités non disponibles	Nous allons demander à des utilisateurs lambda de tester l'application. Aussi, nous aurons un plan de tests d'utilisation à parcourir pour éviter de briser d'autres fonctions. Le cas d'un bogue trouvé, il faut capturer les étapes de reproduction de l'erreur, faire du débogage localement avec les étapes de reproduction, identifier le type d'erreur et, finalement, la gérer.

6 - Incompatibilité sur différents systèmes				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
5	L'application contient des fonctionnalités qui ne sont pas compatibles avec certaines versions d'Android.	F	Système non disponible	Nous devons fixer une version minimale réaliste et implémenter les fonctionnalités pour les adapter au système cible. Les systèmes en bas de la version ciblée ne pourront pas utiliser l'application. Aussi, nous allons tester avec les appareils fournis par l'école.

7 - Interface manque de fluidité (pas intuitif)				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
4	L'interface est difficile à naviguer. Les utilisateurs doivent effectuer plusieurs étapes avant d'arriver au résultat voulu.	M	Temps de complétion élevé	Premièrement, nous devons établir des cas d'utilisation typiques. Ainsi, à chaque nouvelle fonctionnalité, passer par les cas typiques utilisant la nouvelle fonction et évaluer le temps pris pour les accomplir.

8 - Fonctionnalité incomplète				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
7	Un membre ne parvient pas à compléter une fonctionnalité avant la date d'échéance établie au sein de l'équipe.	E	Fonctionnalités non disponibles	L'équipe fera des suivis de manière hebdomadaire afin d'assurer la bonne continuation des tâches. Nous allons tous nous fixer une limite de temps pour tenter d'accomplir une tâche avant de demander de l'aide. Lors d'un empêchement, il faut prévenir l'équipe pour qu'elle agisse en conséquence.

9 - Application pas facilement extensible				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
3	Les développeurs ont de la difficulté à intégrer une nouvelle fonctionnalité ou étendre une fonctionnalité existante.	E	Implémentation ralentie	En équipe, il faut se familiariser avec la complexité des différentes fonctionnalités et avec la façon de les intégrer. Ensuite, nous pourrons bâtir une architecture qui considère les intégrations futures. Faire des diagrammes de paquetage et/ou de classes pourrait s'avérer utile.

3.4. Gestion de configuration

Lorsque l'équipe trouve un bogue dans le produit, un membre dont l'expertise s'aligne avec le problème (voir section 5) sera convoqué à résoudre le bogue. Il est important que le problème soit abordé aussitôt que possible, pour maintenir une qualité acceptable des biens. Il effectuera la résolution sur une nouvelle branche de Git afin de mieux isoler le problème. Le cas où le membre est incapable de trouver la source du bogue, il sollicitera l'aide des autres membres de l'équipe et, si nécessaire, des chargés de laboratoire.

Par rapport aux normes de l'identification des artéfacts, nous avons établi certaines normes de contrôle *Git*.

- La branche *master* servira comme branche stable et finale. Elle ne doit pas être modifiée sans l'accord de la totalité de l'équipe.
- La branche *dev* sera semblable à la branche *master*, mais servira d'un point d'intégration de plusieurs fonctionnalités. Elle devra être testée et être stable avant d'intégrer dans *master* (elle est d'ailleurs la seule branche qui pourra être intégrée directement dans *master*).
- thin/* : Correspond aux branches de fonctionnalités (et changements d'interface) du client léger.
- fat/*: Correspond aux branches de fonctionnalités (et changements d'interface) du client lourd.
- server/*: Correspond aux branches de fonctionnalités côté serveur.
- hotfix/*: Nomenclature pour les branches qui serviront à résoudre un bogue.

Après la résolution d'un problème fonctionnel, le membre ayant complété la résolution devra soumettre le code pour une revue. Les autres membres de l'équipe devront réviser le code afin d'assurer la qualité de la modification.

Lors d'un changement aux artéfacts écrits (les documents d'appel d'offre par exemple),

nous sommes engagés à prévenir les membres de l'équipe. Les changements seront tous visibles dans Google Drive et nous devons mettre à jour l'historique des changements avec une description pertinente.

4. Échéancier du projet

Date de début	Date de fin	Lot de travail	Description	Nombre d'heures-travail
14 janvier	28 janvier	Rédaction liste des exigences	Consultation du complément pédagogique, choix des tâches essentielles et des tâches souhaitables, propositions de nouvelles fonctionnalités, validation et révision de la liste des exigences.	40
13 janvier	20 janvier	Rédaction du plan de projet	Énoncer la solution proposée, analyser les hypothèses en ce qui a trait au travail, lister les livrables, discuter de la gestion du projet, de la configuration et des risques, proposition de l'échéancier, présentation de l'équipe de travail ainsi que de l'entente contractuelle.	20
13 janvier	23 janvier	Rédaction du SRS	Description globale du jeu et de ses utilisateurs, présentation des exigences fonctionnelles et non fonctionnelles.	40
21 janvier	30 janvier	Prototype-Implémentation du serveur	Implémentation des fonctionnalités de base du côté serveur.	20
23 janvier	29 janvier	Prototype-Connexion	Pour le client lourd et léger: implémentation de l'interface de connexion au jeu.	20
28 janvier	3 février	Prototype-Clavardage	Pour le client lourd et léger: implémentation de l'interface de clavardage du jeu.	40
28 janvier	3 février	Rédaction du document d'architecture	Énoncer les objectifs et les contraintes architecturales du projet, Présentation des différentes vues à l'aide de diagrammes, explication de ces derniers ainsi qu'à énoncer les objectifs au	20

			niveau de la taille et de la performance du logiciel.	
30 janvier	31 février	Prototype-Communication Client Lourd avec Serveur	Liaison du client lourd au serveur afin de permettre la sauvegarde du profil et donner la possibilité à l'utilisateur de discuter avec les autres utilisateurs à travers le serveur.	20
30 janvier	2 février	Prototype-Communication Client Léger avec Serveur	Liaison du client léger au serveur afin de permettre la sauvegarde du profil et de donner la possibilité à l'utilisateur de discuter avec les autres utilisateurs à travers le serveur.	20
3 février	7 février	Prototype-Raffinement et Livraison	Perfectionnement de l'interface utilisateur, correction des bogues, nettoyage du code et vérification du fonctionnement du code à l'aide des tests.	20
11 février	25 février	Intégration du clavardage au jeu	Pour le client lourd et léger: intégrer la fenêtre de connexion et de clavardage au jeu.	20
11 février	25 février	Canaux de discussion	Pour le client lourd et léger: offrir la possibilité à l'utilisateur de rejoindre plusieurs canaux de discussion simultanément et pouvoir interagir dans chacun de ceux-ci; alterner entre le mode fenêtré et le mode intégré; filtration des messages; possibilité d'afficher l'historique de chaque canal de clavardage.	30
11 février	25 février	Sauvegarde du dessin	Pour le client lourd et léger: L'utilisateur peut sauvegarder localement son dessin.	20
11 février	25 février	Développement du serveur	Addition de fonctionnalités au serveur pour permettre les canaux de discussions multiples ainsi que la sauvegarde locale du dessin.	40
11 février	25 février	Mode de jeu: Dessin libre	Pour le client lourd et léger: Implémenter les fonctionnalités de sorte à respecter les exigences concernant le mode de jeu dessin libre.	20
25 février	10 mars	Profil utilisateur et	Pour le client lourd et léger:	30

		historique	Implémenter les fonctionnalités de sorte que l'utilisateur puisse consulter l'historique de son profil ainsi que l'historique de toutes les parties que ce dernier a joué.	
25 février	10 mars	Classement	Pour le client lourd et léger: Implémenter les fonctionnalités de sorte à respecter l'exigence de classement. Mise à jour du classement après chaque partie, afficher la position de l'utilisateur.	30
25 février	10 mars	Création d'un jeu	Pour le client lourd et léger: Implémenter les fonctionnalités de sorte à respecter les exigences de création d'un jeu.	60
25 février	10 mars	Développement du serveur	Addition de fonctionnalités au serveur pour permettre les mises à jour du classement et l'historique du profil utilisateur, supporter la création et l'enregistrement d'un nouveau jeu.	40
10 mars	24 mars	Mode de jeu	Pour le client lourd et léger: Implémenter les fonctionnalités de sorte à respecter les exigences des modes de jeu sprint solo et collectif.	60
10 mars	24 mars	Mode de jeu	Pour le client lourd et léger: Implémenter les fonctionnalités de sorte à respecter les exigences du mode de jeu 1 contre 1	30
10 mars	24 mars	Mode de jeu	Pour le client lourd et léger: Implémenter les fonctionnalités de sorte à respecter les exigences du mode de jeu inversé.	60
10 mars	24 mars	Dévoilement des lettres	Pour le client lourd et léger: implémenter la fonctionnalité qui permet de dévoiler les lettres en fonction du temps.	30
24 mars	31 mars	Personnalité des joueurs	Pour le client lourd et léger: Implémenter les fonctionnalités de sorte à respecter les requis concernant la personnalité des joueurs.	20

24 mars	31 mars	Effets sonores	Pour le client lourd et léger: Implémenter les fonctionnalités de sorte à respecter les requis concernant les effets sonores.	15
24 mars	31 mars	Tutoriel	Faire un tutoriel que l'utilisateur pourra consulter à tout moment.	15
24 mars	31 mars	Développement du serveur	Addition de fonctionnalités au serveur pour permettre aux nouveaux modes de jeu de fonctionner correctement.	60
31 mars	13 avril	Tests-Planification	Planifier les tests et rédiger le plan de planification des tests.	30
31 mars	13 avril	Tests	Vérification des tests et du fonctionnement du code.	40
31 mars	13 avril	Tests-Résultats	Rédiger les résultats des tests effectués.	25
31 mars	13 avril	Produit Final-Révision	Réviser le produit final afin de s'assurer qu'il réponde à l'appel d'offres, aux exigences et aux contraintes.	75
31 mars	13 avril	Produit Final-Raffinement et Livraison	Perfectionnement du code, résolution des bogues, amélioration de l'expérience utilisateur.	70
Total				1080

5. Équipe de développement

L'équipe de développement est composée de six membres ayant presque complété trois ans du programme en ingénierie logicielle. La répartition des tâches est faite de sorte que Zakari et Hubert s'occuperont principalement du serveur, Émilio et Karima effectueront les tâches concernant le client lourd et Amar et Nicole seront responsables du client léger. Tous les membres de l'équipe ont déjà fait du développement Web. Ils ont donc des connaissances poussées en TypeScript, JavaScript en HTML et CSS. Ils ont une bonne maîtrise et compréhension de C++, de Java et de Python. Parmi les membres de l'équipe, nous retrouvons Nicole, Hubert et Amar ont déjà complété un stage en développement Web et Android. Leurs connaissances poussées sont des atouts considérables lors de ce projet. Les tâches concernant la documentation sont équitablement partagées par tous les membres de l'équipe. Les révisions des documents sont elles aussi faites par tous les individus qui composent l'équipe. Pour ce qui concerne la répartition des

tâches de développement du produit, chacun a choisi la partie qui le motivait et qui l'intéressait le plus. Les habiletés et les compétences de chacun ont bien évidemment joué un rôle important dans cette division.

6. Entente contractuelle proposée

L'entente contractuelle la plus favorable dans ce cas est le contrat livraison clé en main. Les exigences sont connues au préalable et il y a peu de chances qu'elles soient changées une fois le début du processus de développement du produit. Il est donc impératif que tout ce qui se retrouve dans la catégorie d'exigences essentielles soit implémenté lors de la livraison du produit. Le temps estimé pour compléter ce projet est de 1080 heures au total. Cela fait en sorte que chaque membre de l'équipe devra consacrer 180 heures au projet. En ce qui concerne le coût des ressources, les cinq développeurs ont un salaire de 100$/heure et le salaire du chargé de projet s'élève à 125$/heure. En effectuant le calcul suivant : (100$/h * 180h) * 5 personnes + (125$/h *180h), nous obtenant un coût total s'élevant à 112 500$.