Équipe 11

PolyPaint™
Plan de projet

Version 1.3

Historique des révisions

Date	Version	Description	Auteur
2018-01-17	1.0	Première ébauche du document	Équipe 11
2018-01-29	1.1	Document complété	Équipe 11
2018-01-09	1.2	Derniers correctifs avant la première remise	Équipe 11
2018-01-12	1.3	Correction suite à la rétroaction de la première remise	Équipe 11

Table des matières

Plan de projet

1. Introduction

Ce document contient le plan de projet pour le logiciel PolyPaint. Pour débuter, la section 2 décrira plus en détail le projet ainsi que les biens livrables. Ensuite, la section 3 décrira nos outils de gestion de projet qui seront utilisés tout au long de celui-ci. Par la suite, la section 4 présentera un échéancier du projet en détail. La section 5 présentera l'expérience et le rôle des membres de l'équipe de développement. Finalement, 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 proposition consiste en une évolution de PolyPaint, une application déjà existante. Nous proposons principalement l'ajout d'une fonctionnalité de dessin en réseau, ainsi que le développement d'un client léger sur iOS. Ainsi, l'utilisateur aura la possibilité de dessiner avec d'autres utilisateurs sur le même tableau, et ce, simultanément. Bien évidemment, un mode hors-ligne sera disponible pour les utilisateurs ne souhaitant pas être en réseau, ainsi qu'un mode privé pour les parties entre amis seulement. Nous proposons, de plus, d'ajouter un système de clavardage pour faciliter la communication entre les utilisateurs. En plus des outils déjà présents dans PolyPaint, d'autres outils tels que les formes stylisées ou les calques seront ajoutés. Présentement, PolyPaint utilise un mode par trait ce qui n'est pas pratique pour réaliser certains dessins. Ainsi, nous proposons un mode par pixel, ce qui permet d'augmenter la précision de l'utilisateur. Pour faciliter le tout, nous proposons aussi l'ajout d'une galerie qui permettra de visualiser les images créées dans PolyPaint. Cette galerie sera accessible par un site Web qui sera développé par l'équipe. Finalement, nous proposons de rajouter un tutoriel pour permettre aux nouveaux utilisateurs d'apprendre rapidement à utiliser les nouvelles fonctionnalités de PolyPaint. La grande majorité des fonctionnalités du client lourd seront disponibles sur le client léger iOS, mais dans une interface plus conviviale afin d'être adaptées à un écran tactile.

2.2. Hypothèses et contraintes

Ce plan repose sur nos hypothèses au niveau des compétences de chaque membre de l'équipe, ainsi qu'au temps que chaque développeur est prêt à consacrer au projet. L'échéancier se base sur nos estimations du temps nécessaire pour chaque fonctionnalité ainsi qu'aux contraintes dues aux laboratoires ainsi que les examens des autres cours suivis par les développeurs. Ainsi, une semaine que nous prévoyons être surchargés en examen / laboratoires devrait prévoir moins de fonctionnalités. Chaque développeur possède un ordinateur pour pouvoir développer à la maison, de plus, un membre possède un ordinateur Mac pour développer le client léger. Des solutions alternatives telles qu' avoir une machine virtuelle Mac sont possibles si nécessaire.

Le calendrier est la plus grande contrainte du projet. En effet, l'équipe doit finir le projet avant le 16 avril, date à laquelle aucune autre modification au projet ne sera acceptée. Les ressources du point de vu des développeurs est aussi limiter à notre équipe de 6 personnes tout au plus. Le budget pour le projet doit aussi être de 0$ puisque ceci est un projet réalisé dans un cadre pédagogique. Il faudra donc que nos solutions ne demandent pas des coûts monétaires afin d'être implémentées.

2.3. Biens livrables du projet

- **9 février 2018:**
    - Remise des artefacts liés au projet, soit le SRS, le document d'architecture logicielle, le

protocole de communication ainsi que le document ci-présent.
    - Remise d'un prototype de communication pour le client léger et lourd.

- **16 avril 2018:**
    - Remise d'un produit final fonctionnel du livrable.
    - Remise du plan de tests, des résultats de tests logiciels ainsi que de la correction des artefacts du livrable du 9 février (SRS, architecture logicielle, protocole de communication et le document ci-présent).

3. Gestion et suivi de l'avancement

3.1. Gestion des exigences

Chaque mercredi matin, une réunion de 15 minutes sera réalisée pour analyser et comparer l'avancement des tâches qui devaient être réalisées la semaine précédente avec la situation actuelle. Les tâches seront aussi créées sous forme de demande sur le Redmine du projet avec une date de début et de fin afin de pouvoir mieux gérer l'avancement du projet et des exigences. Si un retard dans l'échéancier est présent, une discussion d'équipe sur les problèmes rencontrés causant ce retard sera entamée par la suite. Une charge de travail légèrement supérieure à ce qui avait été prévu pour la semaine ou un changement dans l'échéancier pourrait ensuite être nécessaire pour rectifier la situation avant que la situation ne devienne trop importante. De possibles solutions pour tenter de pallier les problèmes soulevés durant la rencontre seront aussi à l'ordre du jour afin que la situation ne se reproduise pas dans les futures semaines de travail.

3.2. Contrôle de la qualité

Tous les artefacts liés au projet seront lus par toute l'équipe qui devra discuter des modifications jugées pertinentes sous forme de commentaire sur Google Drive afin que toute l'équipe puisse les voir en temps réel. Au moins deux membres de l'équipe seront aussi responsable de passer les artefacts dans le logiciel Antidote avant la remise afin de s'assurer qu'il n'y reste plus aucune erreur de français. Pour l'application de dessin en tant que telle, avant de terminer une demande développée par un développeur, un autre développeur devra vérifier et tester cette demande. Il devra de plus vérifier si la qualité du code est satisfaisante. De plus, aux deux semaines des tests régressifs seront effectués pour vérifier que les fonctionnalités développées précédemment sont toujours fonctionnelles. Tout au long du projet, les développeurs feront des tests unitaires de base pour s'assurer d'une qualité minimale du code selon le critère EC (Each Choice). Lorsqu'une demande ne respecte pas les conditions de qualité minimales, le dépôt sera refusé et le développeur devra retravailler cette demande. Une fois la correction terminée, la demande sera de nouveau vérifiée et testée. Ce cycle se répète jusqu'à ce que la demande respecte les critères de qualité.

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 - Panne de la base de données				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
1	La base de donnée liée au serveur tombe hors-ligne temporairement.	C	Ralentissement du développement des tâches liées à la base de données	Accepter la panne

2 - Synchronisation défectueuse de la galerie				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
5	Synchronisation défectueuse des images de la galerie d'un ou plusieurs utilisateurs propriétaires.	E	Qualité du code	Renforcement des méthodes de synchronisation des images lors de la création, connexion ou modification de celle-ci. Plusieurs essais de synchronisation en cas d'échec.

3 - Communication défectueuse client lourd-léger				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
3	La communication (envoi d'information, chat, etc.) entre un client lourd et un client léger est défectueuse.	M	Qualité du code	Appliquer une architecture de communication robuste et semblable autant pour le client lourd que pour le client léger. Renvoyer les paquets en cas de communication défectueuse

4 - Perte de performance du système				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
1	Une perte de performance (FPS, feedback serveur, etc.) est remarquée lors de l'utilisation de l'application sous un client lourd ou léger.	C	Performance	Prise en compte d'un niveau de complexité asymptotique adéquat lors de la conception du logiciel.

3.4. Gestion de configuration

La soumission, révision ainsi que disposition des problèmes ou changements sur le produit s'effectuera entièrement par l'utilisation d'un espace Git. Tout dépôt d'un changement dans cet espace Git devra obligatoirement être accompagné d'un commentaire, décrivant clairement le changement appliqué. De plus, ces dépôts doivent préférablement être envoyés sous une branche liée à sa demande respective. Lorsque le développement d'une branche est terminé, la fusion de celle-ci avec la branche principale "*master*" est de mise.

Les noms des demandes et artefacts du projet devraient préférablement respecter le contenu de la section 3 du SRS (Exigences fonctionnelles). Ceci s'applique principalement lors de la création d'une branche et de l'envoi d'un dépôt. Pour ce qui est de la numérotation, toutes demandes seront identifiées par le numéro imposé par le site Redmine.

Pour les artéfacts, chaque modification importante apportée à un document (ajout d'une section, révision globale du document, etc.) sera marquée dans la deuxième page de l'artefact. La numérotation commencera à 1.0 (document initial) et sera incrémentée de 0.1 pour chaque modification importante.

4. Échéancier du projet

Tâche	Effort estimé (heures-personne)	Date de début	Date de fin
Janvier	**210**	**2018-01-08**	**2018-01-31**
Réponse à l'appel d'offres	**220**	**2018-01-08**	**2018-02-09**
Spécification des requis du système (SRS) v.1	**40**	**2018-01-08**	**2018-01-23**
Protocole de communication	**20**	**2018-01-10**	**2018-01-24**
Plan du projet	**30**	**2018-01-17**	**2018-02-01**
Prototype de communication client lourd-serveur	**40**	**2018-01-24**	**2018-02-09**
Prototype de communication client léger-serveur	**40**	**2018-01-24**	**2018-02-09**
Document d'architecture logicielle	**40**	**2018-01-29**	**2018-02-08**

Février	**260**	**2018-02-01**	**2018-02-28**
Spécification des requis (SRS) v.2 suite à la rétroaction	**10**	**2018-02-05**	**2018-02-09**
Remise de la réponse à l'appel d'offres			**2018-02-09**
Produit final	**830**	**2018-02-12**	**2018-04-16**
Base de données	**20**	**2018-02-12**	**2018-02-21**
Finaliser le prototype de communication pour un canal	**30**	**2018-02-12**	**2018-02-21**
Profil utilisateur	**20**	**2018-02-15**	**2018-02-21**
Modifier mot de passe	**10**	**2018-02-21**	**2018-02-25**
Page d'identification	**20**	**2018-02-21**	**2018-02-28**
Lobby multijoueurs	**40**	**2018-02-21**	**2018-03-07**
Clavardage avec plusieurs canaux	**60**	**2018-02-25**	**2018-03-05**
Clavardage mode fenêtré	**20**	**2018-02-25**	**2018-03-05**
Clavardage mode intégré	**20**	**2018-02-25**	**2018-03-05**
Alterner entre les modes de clavardage	**10**	**2018-02-25**	**2018-03-08**
Mars	**460**	**2018-03-01**	**2018-03-31**
Dessiner par pixel (mpp)	**30**	**2018-03-01**	**2018-03-12**
Collaboration simultanée (mpt)	**20**	**2018-03-07**	**2018-03-14**
Collaboration simultanée (mpp)	**20**	**2018-03-07**	**2018-03-14**
Synchronisation automatique	**70**	**2018-03-07**	**2018-04-17**
Effacer par pixel (mpp)	**10**	**2018-03-10**	**2018-03-17**
Galerie d'images	**70**	**2018-03-10**	**2018-03-24**
Formes stylisées	**30**	**2018-03-14**	**2018-03-23**
Outil de rotation	**20**	**2018-03-14**	**2018-03-23**
Insertion de texte	**20**	**2018-03-17**	**2018-03-26**
Sélection et outils d'édition de région par pixel (mpp)	**40**	**2018-03-23**	**2018-04-13**
Insertion/exportation d'images	**30**	**2018-03-26**	**2018-04-05**
Ajout de mot de passe pour une image	**20**	**2018-03-26**	**2018-04-02**
Mode protégé pour les images	**20**	**2018-03-26**	**2018-04-02**

Remplissage par région (mpp)	**20**	**2018-03-28**	**2018-04-02**
Calques (mpt)	**40**	**2018-03-30**	**2018-04-08**
Avril	**120**	**2018-04-01**	**2018-04-16**
Menu tutoriel	**20**	**2018-04-01**	**2018-04-10**
Popup visuel pour tutoriel interactif	**10**	**2018-04-01**	**2018-04-10**
Filtres (mpp)	**50**	**2018-04-05**	**2018-04-16**
Raccourcis clavier pour les outils	**15**	**2018-04-10**	**2018-04-16**
Utilisation du multi-touch sur le client léger	**25**	**2018-04-10**	**2018-04-16**
Remise du produit final			**2018-04-16**
Total du projet	**1050**	**2018-01-08**	**2018-04-16**

5. Équipe de développement

5.1 Agagnier, Félix

- Étudiant de troisième année au baccalauréat en génie logiciel à l'école Polytechnique Montréal.
- Stagiaire programmeur-analyste chez Desjardins en été 2017.
- Maîtrise du langage C#, C++, Java.
- Expérience de développement web (Angular 2, HTML, Socket.io, CSS).
- SCRUM Master.
- Responsable du client lourd.

5.2 Chemam, Youva

- Étudiant de troisième année au baccalauréat en génie logiciel à l'école Polytechnique Montréal.
- Stagiaire intégration continue chez Bombardier Transport en été 2017.
- Membre de l'avionique chez la société technique Oronos de la Polytechnique Montréal.
- Maîtrise du langage Python, C#, C++/C, Java, Shell script.
- Expérience de développement web (Angular 2, HTML, Socket.io, CSS).
- Responsable du client lourd et du serveur.

5.3 Clark, Alexandre

- Étudiant de troisième année au baccalauréat en génie logiciel à l'école Polytechnique Montréal.
- Maîtrise du langage C++, Java, Swift.
- Expérience de développement web (Angular 2, HTML, Socket.io, CSS, Three js).
- Responsable du client léger.

5.4 Felteau, Simon

- Étudiant de troisième année au baccalauréat en génie logiciel à l'école Polytechnique Montréal.
- Maîtrise du langage C++.
- Expérience de développement web (Angular 2, HTML, Socket.io, CSS, Three js).
- Responsable du client léger

5.5 Lemieux, Carl

- Étudiant de troisième année au baccalauréat en génie logiciel à l'école Polytechnique Montréal.
- Maîtrise du langage Python, C#, C++/C, Java, JavaCC.
- Expérience de développement d'interface et de sockets en C#.
- Responsable du client lourd.
- Directeur du département informatique de la société technique Polycortex de la Polytechnique Montréal.

5.6 Tremblay, David

- Étudiant de troisième année au baccalauréat en génie logiciel à l'école Polytechnique Montréal.
- Maîtrise du langage C#, C++, Java, Matlab.
- Expérience de développement web (Angular 2, HTML, Socket.io, CSS).
- Responsable du client lourd et léger.

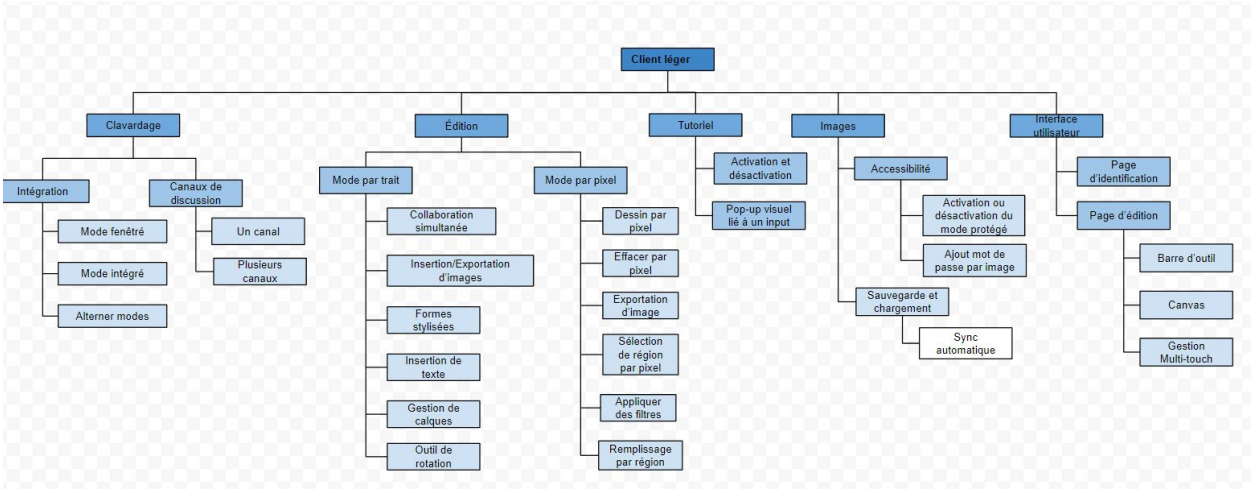
6. Entente contractuelle proposée

Notre équipe propose une entente contractuelle livraison clé en main. Le projet sera livré avec toutes les exigences notées essentielles ainsi qu'au moins 50% des exigences souhaitables qui sont détaillées à la section 3 du document des spécifications des requis système au plus tard le 16 avril 2018 à 23h59. Le projet est estimé à 1050 heures dont 220 heures reliées aux artéfacts et 830 heures reliées au développement du logiciel selon l'échéancier du projet avec une équipe de développement composée de 6 programmeurs au taux horaire de 125$/heure pour la conception des artefacts et 100$/h pour le développement du logiciel.Le prix total demandé pour le projet est donc de 110 500$.

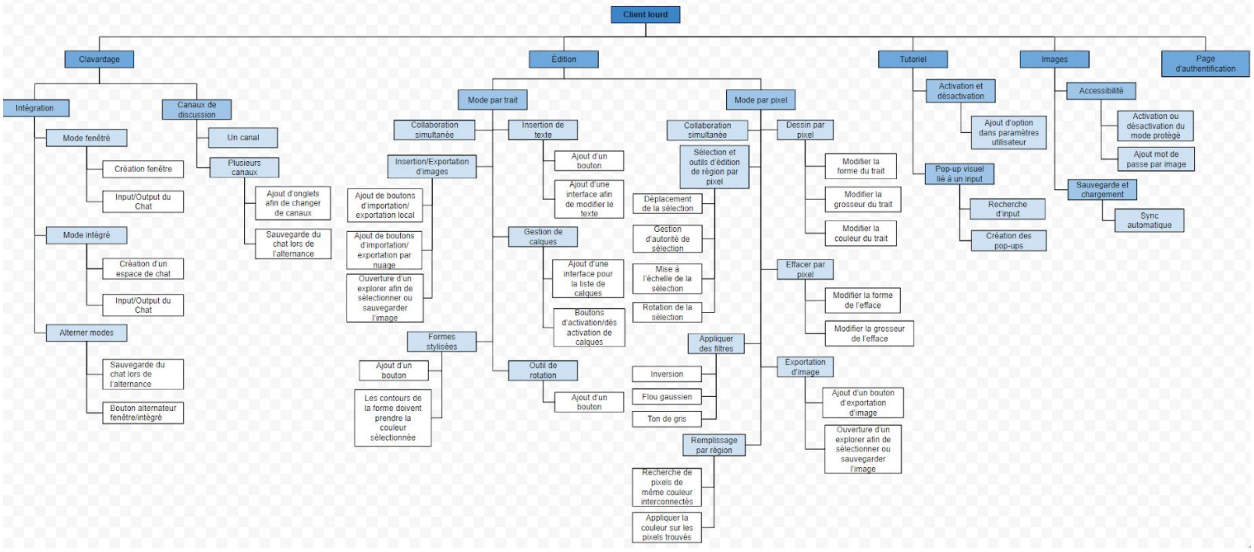
7. Annexes

7.1. Organigrammes des tâches à réaliser

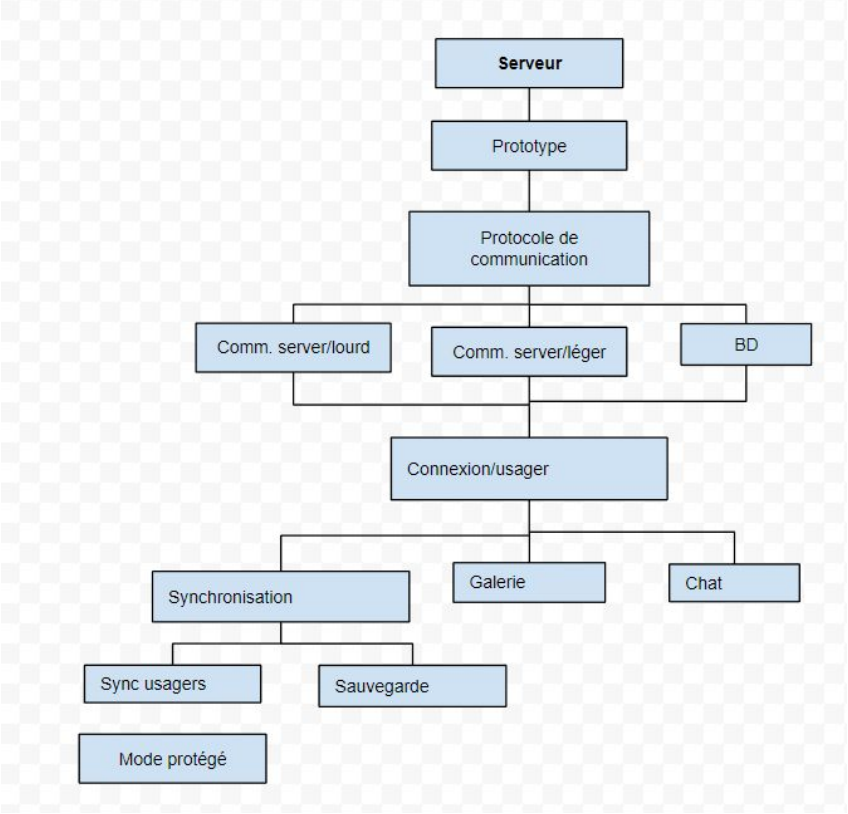
Organigramme du client léger



Organigramme du client lourd



Organigramme du serveur



Organigramme du site Web