# Équipe 112

# Fais-moi un dessin
# Plan de projet

**Version 1.6**

# Historique des révisions

| Date | Version | Description | Auteur |
|---|---|---|---|
| 2020-01-21 | 1.1 | Introduction rédigée | Huyen Trang Dinh |
| 2020-01-22 | 1.2 | Rédaction de ma description dans la partie 5. | Équipe 112 |
| 2020-01-23 | 1.3 | Rédaction de la partie 2, 3 et 6. | Huyen Trang Dinh |
| 2020-01-28 | 1.4 | Complétion de l'échéancier | Huyen Trang Dinh |
| 2020-01-30 | 1.5 | Révision du français | Roger Kazma |
| 2020-04-13 | 1.6 | Révision finale | Huyen Trang Dinh |

# Table des matières

# Plan de projet

## 1. Introduction

Ce document présente le plan de projet du logiciel *Fais-moi un dessin*. La section 2 fera l'objet de l'énoncé des travaux où une solution est proposée. Des hypothèses et contraintes sont émises par rapport à ce plan de projet et les artefacts livrables sont énumérés également dans cette section. Ensuite, la section 3 porte sur la gestion et le suivi de l'avancement. Puis, la section 5 fait état de l'échéancier du projet contenant les efforts prévus et dates importantes. Pour terminer, la section 6 propose une entente contractuelle pour ce projet.

## 2. Énoncé des travaux

### 2.1. Solution proposée

La solution proposée est le logiciel *Fais-moi un dessin* qui consiste en un jeu de devinettes de dessin. La solution sera livrée sous deux formes : une *application de bureau* (client lourd) et une application pour tablette Android (client léger). Les tableaux 1 et 2 démontrent les exigences essentielles et souhaitables pour les deux clients.

Tableau 1. Exigences du client lourd

| **Exigences essentielles** | **Exigences souhaitables** |
|---|---|
| Clavardage en mode fenêtré ou intégré | Profil utilisateur avec historique détaillé, réputation et avatar à partir d'un dessin. |
| Canaux de discussion et historique | Mode de jeu Sprint solo, Sprint coopératif, Duo vs Duo et *Cross-platform* |
| Profil utilisateur et historique | Création de jeu Assistée I, II, et III |
| Mode de jeu Mêlée générale | Personnalité des joueurs virtuels avec mémoire |
| Création de jeu Manuelle I et II | Gestion des amis et invitations |
| Personnalité des joueurs virtuels variés | Options de partage |
| Effets visuels et sonores (un groupe d'effets) | Élimination de joueurs par vote |
| Tutoriel non interactif | Option de langues |

Tableau 2. Exigences du client léger

| **Exigences essentielles** | **Exigences souhaitables** |
|---|---|
| Clavardage intégré | Notifications lors de la réception de nouveaux messages |
| Canaux de discussion | Canaux de discussion avec historique |
| Mode de jeu Mêlée générale | Profil utilisateur avec historique détaillé, réputation, et avatar à partir d'un dessin. |
| Effets visuels et sonores (trois groupes d'effets) | Mode de jeu Sprint solo, Sprint coopératif, Duo vs Duo |

| | |
|---|---|
| | et *Cross-platform* |
| Tutoriel non interactif | Tutoriel interactif |
| | Options de partage |
| | Gestion des amis et invitations |
| | Effet physique de vibration |

Comme les tableaux le démontrent, les exigences sont identiques pour le client lourd et le client léger, à l'exception que le client léger ne permette pas la création de jeux, et ait des effets de vibration sur la tablette.

Il est possible de prendre connaissance des fonctionnalités du logiciel de façon plus approfondie dans le document de spécification des requis du système.

## 2.2. Hypothèses et contraintes

### *2.2.1. Équipement*

Les équipements dont les membres de l'équipe disposent sont les ordinateurs Windows et Linux de Polytechnique Montréal. Chaque membre possède également son ordinateur personnel. Le client lourd sera sur Windows 10 uniquement: les programmeurs de l'équipe travaillant sur leurs ordinateurs roulant iOS peuvent donc connaître des difficultés pour tester le logiciel lors de son développement sur leur machine. Aussi, une tablette Samsung Tab A sera prêtée à l'équipe afin de tester le client léger et devra être retournée à la fin du projet. Enfin, l'application sera développée sur le réseau Wi-Fi de Polytechnique Montréal. Nous supposons que ce réseau ne tombera jamais en panne et fournira une connexion stable en tout temps entre le client et le serveur, avec une vitesse de téléchargement de 10 Mb/s.

### *2.2.2. Ressources humaines et échéancier*

L'équipe de développement est composée de 6 étudiants qui peuvent chacun fournir au minimum 15 heures de travail par semaine. Lors des périodes d'examens à l'école Polytechnique Montréal, ce minimum peut se voir réduire à 10 heures. Ce temps sera compensé par les autres membres de l'équipe pouvant fournir plus d'heures. Ce temps d'effort minimum inclut le temps d'apprentissage et d'adaptation (qui est plus important pour le client léger en Kotlin). Le temps de travail maximum est illimité. Dans le cas d'une catastrophe naturelle, d'une pandémie ou tout autre évènement de ce type où une réorganisation sera nécessaire, ce temps de travail minimum réduira à 7 heures par semaine. Tous les développeurs ont les outils pour travailler à distance, ainsi que les outils de communication nécessaires pour progresser, mais ne seront pas tenus de fournir un effort constant et productif dans cette situation.

### *2.2.3. Échéancier*

L'appel d'offres a été émis le 21 janvier 2020 et la réponse à l'appel d'offres est due le 7 février 2020. Puis, la remise finale du produit ainsi que tous les artefacts du projet sont dus le 13 avril 2020. L'équipe de développement tiendra des réunions à tous les mardis pour une durée minimale de 3 heures afin de s'assurer du suivi de l'avancement. Nous supposons donc que tous les développeurs seront toujours présents pour cette période.

## 2.3. Biens livrables du projet

- 6 février 2020 : Évaluation des coéquipiers sur iPeer
- 7 février 2020 : Réponse à l'appel d'offres incluant les documents suivants :
    - SRS
    - Liste d'exigences

    - Plan de projet
    - Document d'architecture logicielle
    - Protocole de communication
    - Prototype de communication (client lourd - serveur)
    - Prototype de communication (client léger - serveur)
- 13 avril 2020 : Tous les documents corrigés et révisés de la réponse à l'appel d'offres, un plan de tests, des résultats de tests ainsi que le code source final pour les clients lourd, léger et le serveur.

# 3. Gestion et suivi de l'avancement

## 3.1. Gestion des exigences

L'artefact initial contenant toutes les exigences potentiellement sujettes à changement est le SRS, soit le document de spécification des requis. Chaque modification des exigences dans le SRS sera reportée au client et approuvée par chacun des membres de l'équipe. Les changements dans le SRS peuvent potentiellement naître, par exemple, d'une décision du client, d'une réorganisation due à une catastrophe naturelle ou d'une réorganisation due à une pandémie. Chaque demande sur Redmine tient, dans sa description, les numéros associés aux exigences. Lorsqu'une demande est complétée, la demande est fermée et les exigences associées sont marquées comme étant complétées sur le SRS. Chaque semaine, l'équipe se doit de vérifier qu'aucune exigence dans le SRS n'a été oubliée ou modifiée dans les descriptions des demandes. Cela permettra de ne pas perdre de vue le respect obligatoire des exigences vers la fin du projet lors de la validation finale du produit par rapport au SRS. Puisque l'ensemble des exigences sont suivies par Redmine, une modification d'une exigence se verrait modifier la demande qui lui est associée. En ce qui concerne l'identification de ces changements, elles se produisent lors de nos rencontres hebdomadaires afin que chaque membre de l'équipe puisse en être informé. Les propositions de ces modifications se produisent lorsque l'équipe discutera du suivi sur l'avancement des exigences durant cette période.

## 3.2. Contrôle de la qualité

### *3.2.1. Bogues*

Pour maintenir un standard de qualité adéquat dans notre logiciel, il est primordial de considérer les bogues, qui peuvent nuire à l'expérience utilisateur et l'état logique du logiciel. Les bogues seront principalement découverts lors du développement des fonctionnalités. Lors des activités de test, d'autres bogues peuvent survenir. Tous les bogues seront signalés sur Redmine en tant qu'anomalies et des personnes seront ciblées pour assurer leur correction. Chaque bogue sera idéalement résolu avant la fin de la semaine au cours de laquelle il a été trouvé. La résolution du bogue, si nécessaire, se fera sur une nouvelle branche Git issue de la branche consacrée au développement de la fonctionnalité concernée.

### *3.2.2. Intégration continue*

L'intégration continue du code consiste à fusionner les travaux des développeurs dans une branche principale plusieurs fois par jour. L'équipe consacrera un effort à créer des *commits* Git petits et significatifs. Aussi, chaque *push* Git doit impérativement être révisé et vérifié par au moins un autre développeur. Dans le cas où une intégration fautive bloque le développement, toute l'équipe sera notifiée et les personnes concernées devront entreprendre les mesures nécessaires pour corriger ou retirer cette erreur le plus rapidement possible. D'autre part, pour éviter des erreurs d'intégration locales, avant toute activité de développement, chaque développeur devra impérativement mettre à jour la branche Git de son espace de travail pour être à jour avec la branche de développement.

### *3.2.3. Tests*

Des séries de tests seront écrites pour tester des cas limites de fragments de codes et de cas généraux. De cette manière, la qualité du code se verra être augmentée et les risques de défaillances diminuées. Ces tests seront décrits dans le document de plan de tests. Dans le cas où du code serait intégré et verrait ses tests échouer, les développeurs dont les changements ont causé l'échec de tests devront immédiatement apporter un correctif.

### 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 — Échec de délivrance des exigences essentielles** | | | | |
|---|---|---|---|---|
| **Ampleur** | **Description** | **Impact** | **Facteurs** | **Stratégie de gestion** |
| **3** | Délivrance d'un produit qui ne perçoit pas toutes les exigences essentielles | C | Nombre d'exigences essentielles complétées | Un suivi de l'avancement sera effectué à chaque semaine pour s'assurer de la progression normale des exigences essentielles. Les exigences liées à la tâche hebdomadaire du SRS devront toutes être raturées. |

| **2 — Échec de délivrance des exigences souhaitables** | | | | |
|---|---|---|---|---|
| **Ampleur** | **Description** | **Impact** | **Facteurs** | **Stratégie de gestion** |
| **3** | Délivrance d'un produit qui ne perçoit pas au moins 50 % des exigences souhaitables | C | Nombre d'exigences souhaitables complétées | Un suivi de l'avancement sera effectué à chaque semaine pour s'assurer de la progression normale des exigences souhaitables. Les exigences liées à la tâche hebdomadaire du SRS devront toutes être raturées. |

| **3 — Échec de délivrance d'un logiciel fonctionnel** | | | | |
|---|---|---|---|---|
| **Ampleur** | **Description** | **Impact** | **Facteurs** | **Stratégie de gestion** |

| | | | | |
|---|---|---|---|---|
| **2** | Délivrance d'un produit qui ne compile pas, ou non-délivrance d'un produit. | C | Nombre de bogues | Dans le cas où le code sur la branche *master* n'est pas fonctionnel, l'équipe devra restaurer une version antérieure qui est fonctionnelle. L'équipe doit aussi s'assurer d'être à jour avec les versions plus récentes en tout temps pour ne pas pousser du code qui va briser le *master*. Il y aura une vérification hebdomadaire de la santé du *master* afin d'évaluer les risques. |

| **4 — Échec d'écriture des tests** | | | | |
|---|---|---|---|---|
| **Ampleur** | **Description** | **Impact** | **Facteurs** | **Stratégie de gestion** |
| **6** | Délivrance d'un logiciel avec des tests qui ne couvrent pas au moins 90 % du code lié aux fonctionnalités à délivrer. | F | Nombre de lignes non couvertes par les tests | L'équipe de développement devra pratiquer le TDD, c'est-à-dire de développer le code en même temps ou après les tests. |

| **5 — Manque d'expertise avec Kotlin sur le client léger** | | | | |
|---|---|---|---|---|
| **Ampleur** | **Description** | **Impact** | **Facteurs** | **Stratégie de gestion** |
| **10** | Un manque d'expérience en développement Android, WPF ou serveur. | M | Nombre d'heures supplémentaires au développement | Toutes les tâches seront assignées aux gens qui possèdent le plus d'expertise dans ce qui les concerne de la manière la plus équitable possible. Les membres de l'équipe qui connaissent des difficultés doivent immédiatement aviser tout le monde dans l'équipe afin de recueillir des informations pouvant réduire le nombre d'heures supplémentaires au développement. |

| **6 — Non-respect du patron MVVM sur le client lourd** | | | | |
|---|---|---|---|---|
| **Ampleur** | **Description** | **Impact** | **Facteurs** | **Stratégie de gestion** |
| **4** | Certaines parties du client lourd ayant pu être codées avec le patron MVVM n'ont pas été adéquatement divisées en MVVM, résultant en singletons. Cela rendrait le débogage et la lisibilité du code plus difficile. | F | Nombre de singletons non essentiels dans le code | Tous les membres de l'équipe doivent maîtriser le patron MVVM dès le tout début. Dans l'éventualité où les singletons apparaissent, ils doivent être éliminés selon leur priorité. |

| 6 — Non-respect du patron MVVM sur le client léger | | | | |
|---|---|---|---|---|
| **Ampleur** | **Description** | **Impact** | **Facteurs** | **Stratégie de gestion** |
| **5** | Certaines parties du client lourd ayant pu être codées avec le patron MVVM n'ont pas été adéquatement divisées en MVVM, résultant en singletons. Cela rendrait le débogage et la lisibilité du code plus difficile. | F | Nombre de singletons non essentiels dans le code | Tous les membres de l'équipe doivent maîtriser le patron MVVM dès le tout début. Dans l'éventualité où les singletons apparaissent, ils doivent être éliminés selon leur priorité. |

### 3.4. Gestion de configuration

Un système de versionnage pour les artefacts est établi dans tous les documents concernés. Au début de chacun de ceux-ci, une table indiquant les versions, les changements et les auteurs permet de gérer les différents groupes de changements dans les documents. La numérotation débute à 1,0 et est incrémentée de 0,1 à chaque session de travail sur le document. Souvent, la version 1.0 sera le gabarit du document sans modifications.

Le processus visant à soumettre, revoir ou disposer des problèmes et des changements fait intervenir Redmine et Git. Pour tous les bogues découverts en production, une demande sera ouverte sur Redmine sous une anomalie. Pour disposer de ce problème, une branche Git *hotfix* sera créée sur la branche concernée par le bogue et se verra régler le bogue en question, par la personne assignée au bogue. Cette personne recevra un avis par courriel afin de prendre connaissance de la situation. Lorsque le bogue est résolu, la branche *hotfix* disparaît après avoir intégré les changements sur la branche *develop*.

L'outil utilisé pour la gestion de version pour tout le code du projet est Git. La division des branches Git sera effectuée de la manière suivante :

- *master* est la branche principale du projet, contenant toujours une version fonctionnelle du produit ;
- *develop* est la branche de développement contenant une version du code toujours fonctionnelle, où les efforts de développement sont déployés ;
- *hotfix*-fonctionnalité est une branche pour résoudre les bogues signalés ;
- les autres branches sont des branches de développement spécifiques à des fonctionnalités qui, lorsque complètes, sont déployées sur *develop*.

## 4. Échéancier du projet

Les jalons sont différenciés en ***gras et italique*** dans ce tableau.

| **Date de début** | **Tâches** | **Lot de travail** | **Temps (h-personne)** | **Date de fin** |
|---|---|---|---|---|
| 21 janvier 2020 | Réponse à l'appel d'offres (artefacts) | Rédaction du plan de projet | 50 | 3 février 2020 |
| 21 janvier 2020 | | Rédaction du protocole de communication | 50 | |
| 21 janvier 2020 | | Rédaction du document d'architecture | 50 | |

| | | | | |
|---|---|---|---|---|
| 27 janvier 2020 | Prototypes (client lourd, léger et serveur) | Requêtes serveur pour le clavardage | 50 | 7 février 2020 |
| | | Clavardage de base client léger | 50 | |
| | | Clavardage de base client lourd | 50 | |
| ***21 janvier 2020*** | ***Délivrance de la réponse à l'appel d'offres (SRS, Plan de projet, prototype, protocole de communication et document d'architecture logicielle)*** | | ***Total : 300*** | ***7 février 2020*** |
| 4 février 2020 | Dév. du système de clavardage | Mode fenêtré, intégré (lourd) ; Historique de clavardage ; canaux de discussion ; *Souhaitables :* Mode fenêtré, intégré (léger) | 45 | 10 février 2020 |
| | Dév. des profils utilisateurs | Statistiques profil ; *Souhaitables :* Historique détaillé profil ; | 40 | |
| 11 février 2020 | Dév. de la création de jeux (lourd) | Manuelle I et II | 50 | 24 février 2020 |
| | | *Souhaitables :* Assistée I, II et III. | 50 | |
| | Dév. des modes de jeu (léger) | Mêlée générale ; *Souhaitables :* Sprint solo | 50 | |
| 25 février 2020 | Dév. des modes de jeu (lourd) | Mêlée générale ; Sprint solo | 40 | 2 mars 2020 |
| | Dév. des modes de jeu (léger) | *Souhaitables :* Sprint coopératif ; *Cross-platform* | 50 | |
| 3 mars 2020 | Dév. des modes de jeu (lourd) ; évaluation du UI/UX | Sprint solo (suite) ; Sprint coopératif | 45 | 9 mars 2020 |
| | Dév. des modes de jeu (léger) ; Évaluation du UI/UX | *Souhaitables :* Sprint coopératif (suite) ; *Cross-platform* (suite) | 50 | |
| 10 mars 2020 | Dév. des personnalités ; Évaluation du UI/UX (lourd) | Formules génériques ; Personnalités variées ; | 45 | 16 mars 2020 |
| | | *Souhaitables* : Conversations poussées des personnalités | 45 | |
| 17 mars 2020 | Dév. des effets visuels et sonores et physiques (lourd) | Particules sur une victoire ; *Souhaitables :* | 25 | 23 mars 2020 |

| | | | | |
|---|---|---|---|---|
| | | Transition sur changement de vue ;<br>Son sur victoire. | | |
| | Dév. des effets visuels et sonores et physiques (client) | Particules sur une victoire ;<br>Son sur victoire ;<br>Son sur début d'une manche.<br>*Souhaitables* :<br>Vibration de la tablette sur une victoire. | 25 | |
| 24 mars 2020 | Dév. du tutoriel (lourd et léger) | Tutoriel non interactif | 30 | 30 mars 2020 |
| | | *Souhaitables* :<br>Tutoriel interactif | 30 | |
| 31 mars 2020 | Rédaction des tests;à révision des documents finaux ; préparation de la présentation orale | Rédaction des tests | 50 | 6 avril 2020 |
| 7 avril 2020 | | Rédaction du plan des tests et résultats des tests | 40 | 13 avril 2020 |
| | | Révision des documents finaux | 40 | |
| | | Préparation à la présentation orale | 30 | |
| ***20 janvier 2020*** | ***Délivrance du projet final*** | | ***Total : 1080*** | ***13 avril 2020*** |

## 5. Équipe de développement

- Huyen Trang Dinh :
  - est étudiante au baccalauréat en génie logiciel en 3e année dans la concentration *Multimédias* ;
  - a effectué un stage en UI/UX dans les jeux vidéos et a utilisé les technologies Unreal Engine 4, Jira et le langage C++ ;
  - est en partie responsable du client lourd, de la rédaction du plan de projet ainsi que le survol du UX.

- Georges Parfait Djimefo Kapen :
  - est étudiant au baccalauréat en génie logiciel en 3e année dans la concentration *Intelligence artificielle et science des données* ;
  - a effectué un stage dans le domaine du *deep learning* précisément sur la reconnaissance optique des caractères sur la plateforme Windows. Les langages utilisés pour la cause étaient Python, Php, C, JAVA ;
  - est en partie responsable du client lourd, du serveur, du prototype de communication, du document d'architecture.

- Ismael Gbian :
  - est étudiant au baccalauréat en génie logiciel en 3e année dans la concentration *Intelligence artificielle et science des données* ;

    - a effectué un stage dans les systèmes de simulation en aéronautique et a utilisé Jira et les langages C++/C# ;
    - est en partie responsable du client lourd, du serveur et de la rédaction du protocole de communication.

- Ibrahim Choukier :
    - est étudiant au baccalauréat en génie logiciel en 3e année ;
    - a effectué un stage dans le domaine de l'avionique avec un outil de système de gestion de vol et a utilisé les langages C++, Java, Perl, XML et UML ;
    - est en partie responsable du client léger, de la rédaction du protocole de communication et du document d'architecture.

- Talet Kayhan :
    - est étudiant au baccalauréat en génie logiciel en 3e année ;
    - a effectué un stage à la ville de Montréal et y a utilisé les langages C et C++ ;
    - est en partie responsable du client léger, de la rédaction du protocole de communication et du document d'architecture.

- Roger Kazma
    - est étudiant au baccalauréat en génie logiciel en 3e année dans la concentration *Intelligence artificielle et science des données* ;
    - a effectué un stage dans le domaine biomédical et y a utilisé le langage C++ ;
    - est en partie responsable du serveur.

## 6. Entente contractuelle proposée

Les modalités suivantes forment l'entente contractuelle proposée par l'équipe de projet 112.

### 6.1. Type de contrat

Ce projet est lié à un contrat clé en main/prix ferme. L'équipe se doit d'être le plus possible indépendante pour maximiser sa productivité: le client ne devra pas intervenir directement dans le développement de l'application. Le coût que déversera le client pour le projet, sans compter les frais engendrés par les changements imprévus, est convenu à l'avance. Cela permet à l'équipe de négocier un prix monétaire de sorte à s'automotiver à l'atteindre.

### 6.2. Paiement

- Chaque développeur voit un salaire horaire de 100 $ pour toutes activités de développement.
- Chaque développeur voit un salaire horaire de 125 $ pour toutes activités de gestion de projet.
- Les heures projetées consacrées au projet sont de 180 heures par personne dont 15 % sont prévus à la gestion de projet
- L'équipe est composée de 6 développeurs qui assumeront chacun des tâches reliées à la gestion en temps et lieu

Ainsi, le client s'engage à payer la somme de 85 % de 1080 heures-personnes pour le temps de développement à 100$ CAD de l'heure, ainsi que 15% de 1080 heures-personnes pour le temps de gestion à 125$ CAD de l'heure, totalisant 112 050$ CAD à la fin du projet.

### 6.3. Changements aux exigences

Toute demande de changement du client devra être signalée à l'équipe de projet. Le client s'engage à payer les frais supplémentaires que peuvent produire ces changements.