**Équipe 102**

**PolyScrabble Plan de projet**

**Version 1.2**

**Historique des révisions**

| **Date** | **Version** | **Description** | **Auteur** |
| --- | --- | --- | --- |
| 2022-09- 19 | 1.0 | Énoncé des travaux partie 2.3  Gestion et suivi de l’avancement parties 3.1 et 3.2 | Aghilès  Maximiliano |
| 2022-09- 26 | 1.1 | Échéancier du projet et finalisation des autres parties | Équipe 102 |
| 2022-09- 28 | 1.2 | Modifications et ajouts d’éléments manquant | Équipe 102 |
| 2022-09- 30 | 1.3 | Ajustement des heures et des tâches dans l’échéancier | Stéphane |

2

**Table des matières**

**1. Introduction 4**

**2. Énoncé des travaux 4** 2.1. Solution proposée 4 2.2. Hypothèses et contraintes 4 2.3. Biens livrables du projet 5

**3. Gestion et suivi de l’avancement 5** 3.1. Gestion des exigences 5 3.2. Contrôle de la qualité 6 3.3. Gestion de risque 6 3.4. Gestion de configuration 8

**4. Échéancier du projet 9 5. Équipe de développement 11 6. Entente contractuelle proposée 12**

3

**Plan de projet**

**1. Introduction**

Dans ce document, nous parlerons en premier lieu de ce que contiendra notre projet, des contraintes associées à celui-ci ainsi que des différents livrables qui seront effectués. Dans la deuxième partie de ce document, nous parlerons des mesures qui seront mises en place pour gérer notre projet. Avec cette partie de gestion viendra ensuite un échéancier du projet.

Nous finirons par présenter notre équipe de développement et parlerons de l'entente contractuelle proposée.

**2. Énoncé des travaux**

**2.1. Solution proposée**

Pour répondre au projet PolyScrabble, nous implémenterons une application pour client léger sous android 9.0 (.apk), une application windows 10 (.exe) ainsi qu’un serveur permettant de faire communiquer les clients entre eux et de gérer la logique derrière chaque action effectuées par nos client. De plus, notre serveur utilisera une base de données MongoDb pour pouvoir stocker des informations qui doivent être persistantes entre parties du client. Ces informations contiennent les informations du compte du client ainsi que les dictionnaires de parties. Pour la communication entre client et serveur nous utiliserons la librairie SocketIO du protocole WebSocket et des requêtes HTTP. Notre solution contiendra deux livrables, un premier contenant la réponse à l’appel d’offre et un prototype de communication entre le client(léger/lourd) avec le serveur pour le 28 septembre 2022 ainsi que la remise de notre projet final (exécutable et code source) pour le 2 décembre 2022.

**2.2. Hypothèses et contraintes**

Lors de ce projet nous serons une équipe de six personnes. Ces six personnes seront disponibles tout au long de la session et travailleront chacun environ cinq heures par semaine. De manière générale, les heures investies tout au long de la session doivent être d’environ 1080h.

L'équipe devra travailler sous Windows pour tout développement du client lourd à cause d'Électron.

L'équipe devra s’appareiller d'une tablette Samsung étant identique à celle utilisée pour définir les exigences du projet.

L'équipe part du principe que les exigences définies dans l’appel d’offre ne changeront pas au cours du projet.

L'équipe devra organiser et planifier des réunions toutes les semaines pour s’informer de l’avancement du projet.

4

L'équipe aura deux remises de travaux importantes; le 28 septembre 2022 pour la réponse à l’appel d’offre et le prototype de communication entre client et serveur. Le 2 décembre 2022 pour la remise finale du projet. Ces dates de remise devront être respectées pour respecter l’appel d'offres.

**2.3. Biens livrables du projet**

Nous avons deux livrables dans ce projet:

Le premier livrable sera remis le 30 septembre 2022 et contiendra notre réponse à l’appel d'offres de notre client. Cette réponse contient les artefacts suivants : - Le plan du projet

- Le SRS

- La liste d’exigence

- Le document d’architecture logicielle

- Le protocole de communication

- Un prototype de communication du client-lourd au serveur et du client léger au serveur

Le deuxième livrable sera remis le 2 décembre 2022 et contiendra le produit final rendu au client.

Ce produit final contiendra les artefacts suivants:

- Une mise à jour des artefact remis lors du premier livrable

- Le plan de test

- Le résultat des tests

- Le code source du projet (client lourd, client léger et serveur)

- Les exécutables pour l’application (.apk/.exe)

**3. Gestion et suivi de l’avancement**

**3.1. Gestion des exigences**

Les exigences pour notre système sont déclarées dans le document de spécifications des requis (voir document SRS). Les exigences furent rédigées par l'équipe en commun et approuvées par les chargés du cours. Par la suite nous avons développé en détail les tâches et de quoi elles consistaient en détail. Ceci nous a permis de diviser les tâches en sous tâche ou même de les séparer en des tâches distinctes. Le logiciel JIRA nous permet de faire le suivi des tâches à un niveau hebdomadaire. Nous pouvons inscrire les tâches et sous tâches et les répartir aux membres de l'équipe ainsi que faire un journal de travail pour cette tâches. Dans le cas où nous devons apporter des modifications aux exigences, nous allons en discuter en équipe pour voir la meilleure approche possible. Nous en discuterons par la suite avec le client pour avoir son approbation, puis on modifierait le document SRS et notre planification JIRA en conséquence

5

**3.2. Contrôle de la qualité**

Afin de s’assurer que notre application garde un certain niveau de qualité tout à travers le processus de développement, nous allons mettre en place certains outils qui seront obligatoires pour les membres de notre équipe. Premièrement, on utilise la fonctionnalité de gitlab pour l'intégration continue, Gitlab CI. Le CI serait mis en place pour les merge requests, et sur chaque merge sur notre branche dev et master. Nous allons vérifier que le serveur, le client lourd et le client léger compilent et passent les vérifications du linter. Aussi on aura les artefacts générés a la fin que l’on peut par la suite tester manuellement si besoin.

Un autre aspect important est la vérification des Merge Requests. Chaque fois que l’on veut merger une branche sur dev, cela requiert au minimum un reviewer obligatoire. Cela permet de bien contrôler ce qui est mis dans notre base de code et de s’assurer que le code est de bonne qualité et qu’il soit compréhensible et modifiable sans casser la fonctionnalité si besoin.

Pour la rédaction des artefacts, nous travaillons à plusieurs sur le même document en tout temps et nous faisons une relecture collective afin d'être sûr qu’il soit cohérent.

**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

o C – critique (affecte le projet en entier)

o E – élevé (affecte les fonctionnalités principales du système)

o M – moyen (devrait être maîtrisable en appliquant une stratégie d’atténuation adéquate) o 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.

6

| **01 - Utilisation d’une nouvelle Technologie** | | |
| --- | --- | --- |
| **Ampleur** | **Description Impact** | **Facteurs Stratégie de gestion** |
| **6** | Dans le cadre du projet nous allons  utiliser une nouvelle technologie que  peut d’entre nous avons utilisé, le  développement mobile avec Flutter.  M  A ce fait, il va falloir apprendre cette  nouvelle technologie au fur et à  mesure de l’avancement du projet. | Temps  consacré à  Nous allons suivre des tutoriels, et  une tâche.  adapter les estimations des tâches  pour le client léger en fonction de  Mauvaises  l’expertise de développeur et de sa  pratiques  compétence avec la technologie.  utilise. |

| **02 - Fonctionnalitées client lourd et leger différentes** | | |
| --- | --- | --- |
| **Ampleur** | **Description Impact** | **Facteurs Stratégie de gestion** |
| **4** | Nous voulons garder les plateformes  client-lourd et client-léger le plus  proche possible en termes de  M  fonctionnalité car on ne veut pas  exclure nos clients dû à leurs choix  de client. | Nous allons avoir des spécialistes  client lourd et léger, puis pour  Temps de  affronter une tâche nous allons  développe  arriver à une solution commune et  ment  applicable aux deux clients afin  qu’ils puissent se réaliser sur les  deux plateformes. |

| **03 - Compatibilité cross-platform** | | |
| --- | --- | --- |
| **Ampleur** | **Description Impact** | **Facteurs Stratégie de gestion** |
| **4** | Le fait de développer sur différentes  plateformes peut faire ressortir des  différences au niveau des  implémentations qui pourraient  causer des problèmes, notamment  M  en réseautique avec les websockets.  Nous voulons qu’un utilisateur puisse  passer facilement du client lourd au  client léger sans devoir ré-apprendre  comment utiliser l’application. | Nous allons devoir faire des tests de  stress du système afin de trouver s'  Temps de  il existe des problèmes au niveau  débogage  des clients légers et lourds.  Nous allons faire des maquettes des  Temps de  interfaces utilisateurs pour le client  préparatio  lourd et léger afin d’avoir une  n de tâche  interface cohérente entre les deux  plateformes. |

| **04 - Risque de sécurités des identifiants** | | |
| --- | --- | --- |
| **Ampleur** | **Description Impact** | **Facteurs Stratégie de gestion** |
| **8** | Étant des débutants dans le domaine  de la sécurité informatique, il se peut  que des personnes mal intentionnées  C  profite de failles qui n’ont pas  forcement ete gérer (failles sql, brute  force, etc). | Essayer de se documenter le plus  possible de manière à pouvoir  Disponibilit  implémenter des solutions efficaces  és/Sécurit  de prévention et de remédiation  é  dans le cas d’une attaque.  Tester nos routes protégées du  serveur. |

7

| **05 - Planification du projet sur 3 mois** | | |
| --- | --- | --- |
| **Ampleur** | **Description Impact** | **Facteurs Stratégie de gestion** |
| **3** | N’ayant pas de remise du mois de  septembre au mois de décembre,  l'équipe risque de se perdre dans les  F  tâches à faire, de ne pas respecter le  planning ou de ralentir son travail. | Faire des réunions fréquentes  chaque semaine avec une  planification résultante de la  Organisati  réunion. Se tenir au courant entre  on/Remise  chaque membre de l'équipe pour  de travail  voir les points qui avancent ou qui  n’avancent pas. Se motiver entre  nous pour garder en tête notre  objectif et notre but commun. |

| **06 - Intuitivité du UI** | | |
| --- | --- | --- |
| **Ampleur** | **Description Impact** | **Facteurs Stratégie de gestion** |
| **5** | Les nombreuses fonctionnalités dans  notre application font que notre UI  pourrait devenir sale et trop remplie a  certain endroit. L’utilisateur pourrait  alors avoir des difficultés à se  M  retrouver et à trouver les contrôles  sur l’application ce qui pourrait le  pousser à ne plus utiliser  l’application. | Faire des réunions pour parler  uniquement du design, appointer  une personne responsable du  Esthétique  design pour vérifier l'intuitivité de  /Opérabilit  l’application. Faire des tests sur des  é  personnes extérieures à l'équipe  pour vérifier l'utilisabilité et  l'intuitivité de l’application. |

**3.4. Gestion de configuration**

Lorsque un bug est soulevé, ou trouvé par un membre de l'équipe, ce dernier doit ouvrir un ticket sur JIRA en décrivant le problème et comment le reproduire, il devrait par la suite mettre une annonce sur notre discord d'équipe en mettant le link du issue afin de voir si quelqu'un peut avoir une idée de la source du problème. Par la suite on assigner une personne à régler ce ticket sur une branche /git/\*. Le développeur va régler le bug puis il va soumettre un Merge Request. Le Merge Request va devoir être approuvé par un reviewer qui s'assurera que le bug est bien réglé puis approuverait le merge request.

Si un artefact doit être modifié, nous allons incrémenter la version du document et mettre dans le tableau de modifications les informations concernant le ou les changements apportés.

8

**4. Échéancier du projet**

| Numéro du sprint | Nom de l’exigence | Temps attribué Date début | Date Fin |
| --- | --- | --- | --- |
| Sprint 1 | Rédaction et correction du SRS | 20h 05-09-2022 | 15-09-2022 |
| Document  D’architecture logicielle | 25h 15-09-2022 | 30-09-2022 |
| Plan de Projet | 15h 15-09-2022 | 30-09-2022 |
| Protocole de Communication | 20h 15-09-2022 | 30-09-2022 |
| Prototype client lourd | 30h 22-09-2022 | 30-09-2022 |
| Prototype client leger | 30h 22-09-2022 | 30-09-2022 |
| Sprint 2 | Mode de jeu classique  (Client Léger) | 70h 30-09-2022 | 17-10-2022 |
| Compte  utilisateurs et Historique  (Client lourd) | 40h 30-09-2022 | 10-10-2022 |
| Créer les avatars (Client lourd) | 20h 30-09-2022 | 10-10-2022 |
| Visibilité des parties (client lourd) | 10h 30-09-2022 | 10-10-2022 |
| Clavardage - Canaux de discussion | 40h 30-09-2022 | 17-10-2022 |

9

| Sprint 3 | (client lourd) |  |  |
| --- | --- | --- | --- |
| Presentation  des resultats de fin de la partie (Client lourd) | 25h 10-10-2022 | 17-10-2022 |
| Mode de jeu avec  rankings(client lourd) | 30h 30-09-2022 | 17-10-2022 |
| Mode de jeu carte de pouvoir  configurable  (client lourd)  Rétrospective de sprint  Synchronisatio n en continue  Clavardage  Intégration  Mode de jeu avec rankings (client  léger)Système de elo | 30h 30-09-2022  18h 17-10-2022 50h 17-10-2022 50h 17-10-2022 25h 17-10-2022 | 17-10-2022  17-10-2022  31-10-2022  31-10-2022  24-10-2022 |
| Compte  utilisateurs et Historique  (Client léger) | 40h 17-10-2022 | 24-10-2022 |
| Créer les avatars (Client léger) | 25h 17-10-2022 | 24-10-2022 |
| Presentation  des resultats de fin de la | 25h 17-10-2022 | 24-10-2022 |

10

|  | partie (Client léger) |  |  |
| --- | --- | --- | --- |
| Mode de jeu carte de pouvoir  configurable  (client léger) | 30h 24-10-2022 | 31-10-2022 |
| Rétrospective de sprint | 18h 01-11-2022 | 01-11-2022 |
| Sprint 4 | Persistance  des  configurations  Barre de recherche  (client léger)  Sons sur actions  Retravailler  l’interface et l’expérience  utilisateur  Plan de test | 40h 07-11-2022 35h 07-11-2022  25h 07-11-2022 50h 14-11-2022  40h 07-11-2022 | 14-11-2022  21-11-2022  21-11-2022  21-11-2022  20-11-2022 |
| Rétrospective de sprint | 18h 20-11-2022 | 20-11-2022 |
| Sprint 5 | Résultat des tests | 40h 21-11-2022 | 02-12-2022 |
| Mise à jour des artefacts de départ | 50h 21-11-2022 | 02-12-2022 |
| création des artefacts finaux (exécutables) | 10h 21-11-2022 | 02-12-2022 |
| Refactoring/rev ue de code côté serveur et | 8h 21-11-2022 | 26-11-2022 |

11

|  | client |  |  |
| --- | --- | --- | --- |
| Revue de toutes les fonctionnalités du client léger | 30h 26-11-2022 | 02-12-2022 |
| Revue de toutes les fonctionnalités du client lourd | 30h 26-11-2022 | 02-12-2022 |
| Rétrospective de sprint | 18h 02-12-2022 | 02-12-2022 |

**5. Équipe de développement**

**Aghiles Gasselin**

Aghiles Gasselin est un étudiant de troisième année en génie logiciel spécialisé dans la logique de jeu et la structure de l’application côté client lourd et serveur.

**Maximiliano Falicoff**

Maximiliano Falicoff est un étudiant de troisième année en génie logiciel spécialisé dans la programmation serveur, gestion de la base de données, de l’automatisation du CI/CD.

**Corentin Glaus**

Corentin Glaus est un étudiant de troisième année en génie logiciel spécialisé dans l’interface graphique du client léger.

**Stéphane Toyo Demanou**

Stéphane Toyo Demanou est un étudiant de troisième année en génie logiciel spécialisé dans l’interface graphique des clients lourd et léger ainsi que de la gestion de base de données.

**Marc-antoine Baillargeon**

Marc-Antoine est un étudiant de troisième année en génie logiciel spécialisé dans la logique de jeu et la structure de l’application côté client lourd et serveur.

12

**Mohamed Fenjiro**

Mohamed est un étudiant de troisième année en génie logiciel spécialisé dans la programmation du côté serveur et implémenter la logique du jeu.

**6. Entente contractuelle proposée**

Le type du contrat choisi par l'équipe 102 est celle d’un contrat fixe selon les exigences et ententes mentionnées ci-dessus et dans les autres artefacts. L'équipe 102 doit livrer PolyScrabble, ainsi que les artefacts à la date du 2 décembre. On estime le projet a 1080 heures de travail pour l'entièreté du projet et de l'équipe. L'équipe de développement étant de six personnes, cela fait un total de 712 800$. Le client s’engage a payer Equipe102.inc cette somme a la fin du contrat.

13