Équipe 102

PolyScrabble
Plan de projet

Version 1.6

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
2022-11-21	1.4	Mise à jour suite à la correction	Maximiliano
2022-11-28	1.5	Mise à jour de la section 5	Equipe 102
2022-11-29	1.6	Mis à jour de l'échéancier du projet	Maximiliano

Table des matières

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 client léger pour mobile, et une application desktop, ainsi qu'un serveur permettant de faire communiquer les clients entre eux et de gérer la logique derrière chaque action effectuée par nos clients. De plus, notre serveur utilisera une base de données MongoDb pour pouvoir stocker des données qui doivent être persistantes entre parties du client. Ces informations contiennent les informations du compte du client, des parties 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.

Nous proposons une application Scrabble avec un système d'utilisateurs qui peuvent jouer dans différents modes de jeu. Nous avons le mode classique avec deux joueurs réels minimum, un mode classé où l'utilisateur peut gagner ou perdre son niveau de classement. On a aussi un mode de jeu avec des cartes de pouvoir ou l'utilisateur

possède certaines cartes qu'il peut utiliser à son avantage dans le jeu.

L’application comprend un système de canaux de clavardage accessible à travers toute l'application. Ils pourront créer, joindre et supprimer des canaux de discussion.

En termes de fonctionnalités différentes entre le client léger et lourd, la seule différence concrète est que le client léger possède une fonctionnalité de recherche d’utilisateurs. Les deux clients ont des différences en termes de fonctionnalité dû à la plateforme tel que le mode fenêtre du clavardage n'est pas présent sur le client léger.

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 douze heures par semaine. De manière générale, les heures investies tout au long de la session doivent être d’environ 1080h.

L'équipe peut travailler sous Windows, LInux ou MacOS pour tout développement du client lourd à cause que electron est multiplateforme.

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.

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.

L'équipe et leurs membres devront s’adapter à l'expertise des autres membres et prendre cela en considération lors de l’assignation et la planification des tâches.

L'équipe devra planifier des sprints qui datent d'un lundi au dimanche, avec des réunions prévues le lundi et le mercredi. La création des tâches se fera le dimanche et le lundi ainsi que leurs planification.

L'équipe devra apporter son matériel.

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 le client. 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 et de suivre l'avancement de la tâche en parallèle avec le SRS. 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 tâche. 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, par exemple en terme de modification de son échéancier (plus court ou long).

Prenons le cycle de vie d'une fonctionnalité que l'on doit implémenter. Cette fonctionnalité est tirée du document SRS et on va créer une tâche JIRA pour le sprint actuel ou prochain. On doit mettre le nom de la tâche, un responsable, le type de tâche et l'échéancier, on peut aussi l'associer à un sprint. Lorsqu' un développeur prend cette tâche, il la bouge dans In Progress. Le développeur va travailler sur cette tâche et au fur et à mesure va mettre à jour le JIRA avec le temps consacré et ce qu'il a réalisé. Une fois sa tâche terminée, il va ouvrir un Merge Request sur Gitlab et va bouger la tâche dans In Review en attendant l'approbation des membres de son équipe. Une fois mergé, il va bouger la tâche dans donc en ajoutant les modification nécessaire s'il a effectuer des changement dû à la rétroaction du merge request.

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 de l'analyseur statique. Aussi on aura les artefacts générés a la fin que l'on peut par la suite tester manuellement si besoin.

Dans le cas où une personne trouve un bug dans l'application, il va créer un ticket sur JIRA avec une capture d'écran, étapes pour reproduire le bogue. Ce bug va être assigné à une personne que ca soit une personne familière avec cette section spécifique. Par la suite on suit les processus décrits ci dessus comme pour un tickets normal.

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. Pour assurer la traçabilité de la version des documents, chaque modification du document entraîne une augmentation de la version du document.

Nous avons un document portant spécifiquement sur les tests que l'on va effectuer sur notre système afin de s'assurer du bon fonctionnement du système.

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.

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 en réseautique avec les websockets. Le risque principal est l'utilisation de certaines libraires sur le client lourd et si leurs fonctionnalités est pareil ou pas sur le client léger. Nous voulons qu'un utilisateur puisse passer facilement du client lourd au client léger sans devoir ré-apprendre comment utiliser l'application.	E	Temps de préparation de tâche Cohésion du système.	Nous allons devoir faire des tests de stress du système afin de trouver s' il existe des problèmes au niveau des clients légers et lourds. Nous allons tester les librairies et observer leur comportement sur les différents clients. Nous allons faire des maquettes des interfaces utilisateurs pour le client lourd et léger afin d'avoir une 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 profite de failles qui n'ont pas forcement ete gérer (failles sql, brute force, etc).	C	Intégrité du système et de leurs données.	Développer un système d'authentification pour protéger certaines fonctionnalités derrière. Hasher les mots de passe sur la base de données. Tester nos routes protégées du serveur.

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 retrouver et à trouver les contrôles sur l'application ce qui pourrait le pousser à ne plus utiliser l'application.	M	Esthétique/ Opérabilité	Faire des réunions pour parler uniquement du design, appointer une personne responsable du design pour vérifier l'intuitivité de l'application. Faire des tests sur des 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 de type bogue 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.

Pour notre environnement de développement, nous avons une branche master, une branche dev, on peut alors considérer notre branche master comme étant notre branche de release et notre branche dev comme notre branche principale. Dev n'est que merged sur master lors d'une remise. La branche dev contient le code le plus à jour dans notre processus de développement. Nous avons certains critères sur comment nommer nos branches. Si on développe une fonctionnalité pour le client lourd on la nomme: feature/heavy_client/feature, pour le client léger: feature/light_client/feature. Si on a un bugfix, on le nomme bugfix/light_client/bugtofix, ou bugfix/heavy_client/bugtofix. Chaque semaine nous faisons un deploy hebdomadaire à partir de notre branche dev chaque lundi.

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	Réunions: Lundi et Mercredi	15h		
	Création Des classes client léger	10h	30-09-2022	17-10-2022
	Creation page game-list client leger	5h	30-09-2022	17-10-2022
	Creation page game client leger	5h	30-09-2022	17-10-2022
	Création du board client léger	10h	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 (client lourd)	40h	30-09-2022	17-10-2022
	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)	30h	30-09-2022	17-10-2022
	Rétrospective de sprint	18h	17-10-2022	17-10-2022
Sprint 3	Réunions: Lundi et Mercredi	15h		
	Synchronisation en continue	50h	17-10-2022	31-10-2022
	Clavardage Intégration	50h	17-10-2022	31-10-2022

	Mode de jeu avec rankings (client léger)Système de elo	25h	17-10-2022	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 partie (Client léger)	25h	17-10-2022	24-10-2022
	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	Réunions: Lundi et Mercredi	15h		
	Persistance des configurations	40h	07-11-2022	14-11-2022
	Barre de recherche (client léger)	35h	07-11-2022	21-11-2022
	Sons sur actions	25h	07-11-2022	21-11-2022
	Retravailler	50h	14-11-2022	21-11-2022

	l'interface et l'expérience utilisateur			
	Plan de test	40h	07-11-2022	20-11-2022
	Rétrospective de sprint	18h	20-11-2022	20-11-2022
Sprint 5	Réunions: Lundi et Mercredi	15h		
	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) et vérification	5h	21-11-2022	02-12-2022
	Refactoring/revue de code côté serveur et client	8h	21-11-2022	26-11-2022
	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 ayant travaillé sur le projet scrabble lors de la session d'hiver 2022. Il connaît très bien les fonctionnalités touchant à la logique de jeu sur le client lourd et sur le serveur (gestions des joueurs ou spectateurs, gestion des tours et gestion de la synchronisation entre chaque utilisateurs). C'est un étudiant ayant cinq mois d'expérience avec Angular et deux ans d'expérience avec nodeJS.

Lors de ce projet il va s'occuper des fonctions des nouveaux modes de jeux de l'application (classique à 4 joueurs et carte de pouvoir configurable) avec le système de spectateurs. Il sera aussi contributeur et instituteur de la planification sur jira. De plus, il s'occupera d'écrire des commentaires et des revues du travail des autres tout au long du projet.

Maximiliano Falicoff

Maximiliano Falicoff est un étudiant de troisième année en génie logiciel. Il possède de l'expérience en développement de serveurs basé sur nodeJS. Il a aussi de l'expérience avec mongoDB. Il a réalisé un projet Angular dans le cadre de projet 2.

Il va principalement s'occuper du développement des features qui nécessitent le développement des routes et des REST API's. Il va aussi s'occuper du client léger et de le mettre à parité avec le client lourd.

Il va aussi contribuer à la planification du JIRA a chaque semaine en créant des tickets selon les normes décrites.

Corentin Glaus

Corentin Glaus est un étudiant de troisième année en génie logiciel spécialisé dans l'interface graphique du client léger. Il possède de l'expérience avec angular et nodeJs. Il sera chargé de la création du game dans le client léger ainsi que dans l'implémentation des langues, des thèmes ainsi que de leur sauvegarde dans la base de donnée.

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 MongoDB. Il a également de l'expérience en développement d'applications Web avec le *framework Angular* en *frontend* et un serveur *NodeJS* dans le cadre du projet 2 et d'un stage en entreprise.

Il aura comme responsabilités principales la création de la page de présentation des résultats de fin partie ainsi que du processus de sauvegarde des parties et de l'ajout de certaines parties dans les favoris de l'utilisateur. Tout ceci implique la création de routes et d'une REST API ainsi que des requêtes vers la base de données sans oublier l'interface graphique qui coordonnera tout ça.

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. Il a de l'expérence avec angular et avec NodeJS. Il a aussi de acquis de l'expérience en typescript lors d'un stage en entreprise et lors de son cours de projet 2. Il sera chargé du mode de partie classé sur le client lourd et léger ainsi que de la rédaction de test et leur exécution.

Mohamed Fenjiro

Mohamed est un étudiant de troisième année en génie logiciel. Suite à la session d'hiver 2022, il a pu se travailler avec le programme Scrabble de base. Il est donc familier avec les méthodes de base, les librairies et protocoles de communication tout autant du côté client que serveur.

Il possède 1 an d'expérience avec Angular, MongoDb et node.js. Mohamed sera chargé de travailler sur la fonctionnalité des canaux de discussions, leurs multiples

communications et intégrations. Il sera aussi chargé de la révision du code ainsi que contributeur sur les tâches de design architectural et de documentation.

6. Entente contractuelle proposée

Le type du contrat choisi par l'équipe 102 est celle d’un contrat fixe clé en main, selon les exigences et ententes mentionnées ci-dessus et dans les autres artefacts.

L'équipe 102 doit livrer PolyScrabble a la date du 2 décembre 2022 à 23h59 au plus tard.

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 avec les développeurs, Chacun seras payé 140$ de l’heure pour son travail, 140 * 1080 = 151200 $ pour le projet . Le client s’engage a payer Equipe102.inc cette somme a la fin du contrat.