Équipe EOS Marketplace

EOS Marketplace V1

Plan de développement logiciel

Version 1.1

Historique des révisions

Date	Version	Description	Auteur
2021-09-10	0.1	Ajout introduction et biens livrables	Guilhem Dubois
2021-09-12	0.2	Vue d'ensemble et organisation du projet	Arthur Garnier
2021-09-15	0.3	Ajout des processus de gestion	Nicolas Hirab
2021-09-15	0.4	Ajout suivi de projet et contrôle	Guilhem Dubois
2021-09-15	0.5	Ajout du calendrier du projet et des objectifs	Gabriel Dambreville
2021-09-16	1.0	Finalisation et correction du document	Jeremy Boulet
2021-09-20	1.1	M.A.J des SRS ID + ajout d'une itération pour le livrable de mi-session + recalcul de h-p	Arthur Garnier
2021-10-07	1.2	M.A.J des sections 4.1.1 et 4.1.2 suite aux changements du calendrier	Guilhem Dubois
2021-12-04	2.0	Correction de mi-session & ajustement par rapport au développement du projet	Arthur Garnier

Table des matières

1. Introduction

Ce document vise à décrire les différentes phases de cycle de vie du produit, les itérations et les activités de notre processus. La première section détaille le projet et les livrables associés. Ensuite, l'organisation du projet décrit la structure de notre équipe et nos interactions avec des groupes externes. Finalement, la section sur le processus de gestion permet de définir les phases et itérations que nous allons suivre en estimant leur effort en termes de durée. Nous prévoyons aussi le suivi du projet, notamment en cas de changements d'exigences ou les risques possibles.

2. Vue d'ensemble du projet

2.1 But du projet, portée et objectifs

Le but de ce projet est de développer un prototype d'une application qui permettra à l'utilisateur d'annoncer un bien ou service et de l'afficher sur la plateforme, sous forme de liste ou de carte. Ce type d'application s'inscrit dans la prochaine évolution des plateformes d'échanges en favorisant l'achat et l'interaction locale et en facilitant davantage le processus d'octroi de services locaux, tout en étant facile d'utilisation.

De plus, la plateforme se présente comme une solution de transfert de bien et de service avec l'avantage de la technologie blockchain. La blockchain permet entre autres, d'assurer l'authenticité de chacune des parties d'une transaction, d'assurer la traçabilité, l'immuabilité et la sécurité des transactions, de programmer les taxes et leur envoi au gouvernement tout en enlevant l'intermédiaire de paiement tel les virements Interac, Visa, Master, PayPal qui charge des frais variants de 1$ jusqu'à 3% du montant de la transaction.

L'application sera donc constituée d'un front-end sur mobile déployé sur Android et IOS, communiquant avec un serveur hébergé sur Google Cloud Engine. Le serveur s'occupera de communiquer avec les différents protocoles blockchain et d'authentification et de transmettre la réponse au client.

2.2 Hypothèses et contraintes

2.2.1. Ressources humaines

L'équipe de développement est composée de 5 étudiants en génie logiciel qui peuvent fournir entre 1000 et 1500 heures-personnes au projet. Ces étudiants possèdent des expériences variables avec les technologies utilisées, et une période de découverte et d'apprentissage est à prendre en compte. On suppose que le projet prendra environ 1100 heures-personnes.

2.2.2. Équipement

Chaque membre à accès aux ressources nécessaire pour le développement du projet, c'est-à-dire un ordinateur et une connexion internet. Pour le développement mobile, des appareils IOS et Android seront disponibles pour tester la capacité cross-plateforme de l'application. Pour le serveur on prévoit d'utiliser un service d'Amazon Web Services permettant le scaling automatique afin de pouvoir répondre à différent type de charge d'utilisateur. On suppose aussi que l'accès à la blockchain EOS se fera sans contrainte matérielle et de manière illimitée.

2.2.3. Échéancier

Le livrable bêta est à remettre le 23 novembre et le livrable final le 7 décembre. Ces livrables correspondent aux remises majeures du projet, ou un prototype fonctionnel est attendu. Entre-temps, l'avancement du projet sera contrôlé par des rencontres hebdomadaires avec le superviseur et des remises au cours de la session comme le livrable de mi-session et le prototype.

2.3 Biens livrables du projet

Ce document, le plan de développement logiciel devra être remis le 16 septembre.

Un livrable de mi-session devra être remis le 8 octobre. Ce livrable doit contenir le processus, le SRS, le document d'architecture logicielle, le plan de développement logiciel et le plan de tests. Le processus, SRS et plan de développement logiciel seront mis à jour par rapport aux remises

précédentes si nécessaire. Le livrable de mi-session doit aussi inclure un prototype de l'application selon les besoins du client.

Un livrable bêta devra être remis le 23 novembre. Une version bêta du produit final devra être remise au client.

Le livrable final devra être remis le 7 décembre. Ce livrable devra contenir le processus, le SRS, le document d'architecture logicielle, le plan de développement logiciel, le plan de tests et le résultat des tests. La version finale du produit devra aussi être remise, soit le code source et la documentation usager.

3. Organisation du projet

3.1 Structure d'organisation

La structure d'organisation pour ce projet se compose de l'équipe de développement, du client, ainsi que du superviseur pédagogique. Ainsi, au niveau du rendu et de la qualité du projet, l'équipe de développement répond au client et sur l'aspect du processus de développement et conformité du projet, au responsable du cours.

3.2 Interfaces externes

Un groupe externe important est le client, EOS Nation Inc. Notre correspondant dans ce groupe est Vincent Grenier, CFO et cofondateur de la compagnie. En tant que responsable du système client, Arthur Garnier fera la plupart des communications par courriel avec le client. Un groupe Telegram permet aussi à toute l'équipe de communiquer avec plusieurs personnes techniques de EOS Nation.

Une autre groupe externe est le responsable du cours du projet final, LOG8970. Olivier Gendreau est notre correspondant dans ce groupe. Les canaux de communications privilégiées seront l'email et le discord pour communiquer avec ce partie.

3.3 Responsabilités

Pour assurer le bon fonctionnement du projet, notre équipe a décidé de répartir les responsabilités principales de manière suivante :

Pour la coordination et la planification : Nicolas Hirab serait le responsable. Il effectuera les tâches de planification et d'organisation des différentes activités. Il s'occupera aussi de s'assurer que les échéanciers sont bien établis et que le plan d'exécution est clair. Dernièrement, il s'occupera de la coordination des activités, notamment des rencontres.

Pour le volet systèmes client : Arthur Garnier serait le responsable. Il s'assurera de la communication avec le client que cela soit par courriel ou lors des rencontres. Il s'assurera aussi que les tâches à effectuer par l'équipe sont bien comprises et mises en accord pour le client et l'équipe afin de bâtir un système convenant aux attentes des deux parties.

Pour le volet processus et méthodologie : Nicolas Hirab serait le responsable. Il s'assurera qu'un processus logiciel est établi afin d'avoir une claire représentation de l'enchaînement des activités que cela soit l'établissement du SRS, tâches de développement, d'analyse ou d'assurance qualité. À travers le projet, il surveillera que notre équipe respecte le processus établi et ajustera le processus au besoin.

Pour le volet technique: Jérémy Boulet serait le responsable. Il s'assurera que les outils technologiques que l'équipe utilisera sont pertinents pour le projet et ont été réfléchis. Bien qu' une liste des technologies ait été fournie, ce rôle serait pertinent au long du développement pour assurer de rester à la fine pointe de technologies et d'utiliser les librairies qui nous conviennent.

Pour le volet produit: Gabriel Dambreville en serait le responsable. Il s'assurera de la gestion des configurations dont le volet intégration continue et le développement continu (CI/CD) du projet. Il sera la personne ressource pour établir les technologies et méthodiques utilisés durant et post développement afin de surveiller le cycle de vie du projet et d'assurer que le projet fonctionne bien en tout temps.

Pour le volet assurance qualité : Guilhem Dubois en serait le responsable. Il s'assurera qu'un plan de validation et des activités de la validation du logiciel sont établis au début du projet. Il devra aussi s'assurer que le logiciel est conforme aux attentes établies avec le client et qu'il ne présente pas d'erreurs de conception majeures.

L'ensemble de ces participants sont aussi responsable de la conception et du développement de l'application.

4. Processus de gestion

4.1 Plan de projet

4.1.1 Planification des phases

Le cycle de vie du produit peut être décomposé en plusieurs phases. Tout d'abord, il y a la phase d'identification des besoins. Lors de cette phase, nous allons spécifier de manière générale les besoins ainsi que les objectifs visés par le client. Vient ensuite la phase de définition du projet. Le but de cette phase est de préparer notre solution à la problématique. Nous prévoyons alors le déroulement du projet et les aspects techniques utiles à la réalisation de notre système, tel que l'architecture du logiciel. Nous allons développer des prototypes pour certaines des composantes du projet tel que l'interface usager. Une fois que nous aurons une idée plus précise de notre solution, nous pourrons entamer la phase de réalisation du produit. C'est lors de cette étape que nous allons implémenter les différentes solutions élaborées précédemment. Le développement logiciel est fait en fonction des recherches faites au préalables et avec un suivi du client et du responsable du cours. Cette phase finira avec la livraison d'une version bêta du produit. Tout au long du développement des composantes, il sera aussi important de tester et de valider la qualité du code. Lors de la dernière phase, le déploiement, les tests d'acceptation seront faits et les résultats des tests seront remis. C'est aussi lors de cette phase que le produit passe de l'environnement de développement à l'environnement de production, où le produit sera remis au client.

4.1.2 Objectifs d'itération

Itération 1: Prise de connaissance du projet et remise des artefacts

Itération 2: Prise de connaissance des technologies (tutoriels, documents d'information...) et conception de l'architecture logicielle

Itération 3: Implémentation des prototypes

Itération 4: Complétion des documents, préparation de l'oral et remise du livrable de mi-session

Itération 5: Début de la création et l'exécution de contrats intelligents, implémentation des fonctionnalités de comptes (voir ses annonces, envoyer des messages, etc.), implémentation des annonces, de la carte et d'anchorWallet

Itération 6: Implémentation du chat. Terminer l'intégration de la carte interactive Montréal/Sherbrooke

Itération 7: Intégration de Moonpay, paiement par crypto et implémentation du contrat intelligent. Processus de transactions

Itération 8: Historique des annonces + System de référencement + récompense & Exécution des tests et remise du livrable final

4.1.3 Calendrier du projet

It.	Date de début et de fin	Tâche	Effort estimé (h-p)	Traçabilité (ID SRS)
1 (70h)	30 août au 16 septembre	Première rencontre avec le client	5	n/a
		Prise de connaissance du projet	15	n/a
		Complétion et livraison de la première version du SRS	25	n/a
		Complétion et livraison du plan de développement	25	n/a
2 (80h)	17 au 23 septembre	Prise de connaissance des technologies requises au projet	35	n/a
		Conception de l'architecture logicielle	45	n/a
3 (110h)	24 au 30 septembre	Développement de l'interface d'authentification (UI)	20	3.1.1
		Conception de la maquette pour l'interface d'authentification (UI)	15	n/a

		Authentification avec firebase + user database	20	3.1.1 3.1.2
		Création de compte	15	3.1.1 3.1.3
		Avancement du document d'architecture	15	n/a
		Avancement du plan de tests	15	n/a
4 (110h)	1er au 8 octobre	Prise de connaissance des technologies de AnchorWallet (Anchor-link,UAL)	15	3.1.5
		Complétion et révision du document d'architecture	15	n/a
		Complétion et révision du plan de tests	15	n/a
		Préparation de la présentation oral	15	n/a
		Implémentation de la base de données et persistance des annonces (ajout en BD)	10	3.2.1
		Intégration de l'authentification avec EOS Wallet (Anchor)	15	3.1.4
		Complétion de la maquette pour toute l'application	15	n/a
		Complétion du plan de développement	10	n/a

5 (200h)	8 au 22 octobre	Création d'un contrat intelligent à partir du backend ("hello world")	40	3.10.1
		Intégration de AnchorWallet (UI,Anchor-Link)	30	3.1.5
		Développement de l'interface pour la création des annonces	25	3.2.1
		Développement de l'interface de la liste des annonces crées par l'utilisateur, groupés par état (open, active, done)	10	3.4.4
		Développement de la vue détaillée d'une annonce	15	3.3.4
		Permettre la modification d'une annonce	15	3.2.6
		Implémentation de compte/profil utilisateur publique	20	3.4
		Implémentation de compte/profil utilisateur privé	15	3.4.2
		Intégration de la carte interactive openStreet	10	3.3
		Intégration des cadastres de Montréal	20	3.3.3
6 (145h)	22 octobre au 5 novembre	~~Implémentation du SWAP~~ Gestion des images sur le serveur	~~35~~ 15	~~3.6~~ 3.2.1.3 3.11.5 3.5.4

		Implémentation de l'interface du chat	10	3.11
		Implémentation des notifications	2	3.11.3
		Implémentation du partage de contrat à travers la messageries	10	3.11.2
		Implémentation du partage de photo à travers la messagerie	8	3.11.5
		Implémentation du chat et de son historique	10	3.11.4
		Intégration des cadastres de Montréal	20	3.3.3
		Afficher les annonces sur la carte dans leur cadastre respectif	25	3.3.3
		Implémentation de la vue et logique permettant d'accepter une annonce + sockets	25	3.5
		Intégration de la vue des détails d'une annonce sur la carte	10	3.3.4
		Implémentation d'un filtre sur les annonces	10	3.3.5
7		Intégration de Moonpay	35	3.7
		Intégration paiement par cryptomonnaie	35	3.8
		Intégration de ApplePay, GooglePay dans Moonpay.	15	3.7.1.3
		Création d'un contrat intelligent sur la blockchain associé à une annonce.	15	3.5.3

(180h)	5 au 23 novembre	Création du code du contrat intelligent EOS permettant le transfert de crypto	35	3.10.2
		Implémentation des services de confirmation de transaction	20	3.5.7
		Mise en place du système d'escrow	15	3.10.2
		Livraison d'un produit bêta	10	n/a
8 (150h)	23 novembre au 7 décembre	Création de la vue de l'historique d'une annonce	10	3.4.4
		Création du système de récompense	20	3.9.3
		Implémentation du système de référencement ~~automatique~~	5	3.9.1 3.9.2
		Ajout de la conversion EOS/CAD	10	3.2.3.5
		Création d'une vidéo pitch présentant le projet pour le client	5	n/a
		Nettoyer et commenter le code	5	n/a
		Exécution des cas de tests	25	n/a
		Rédaction des résultats de tests	15	n/a
		Rédaction de la documentation usager	15	n/a
		Préparation de la présentation oral	15	n/a

		Livraison finale (Processus, SRS, document d'architecture logicielle, plan de développement logiciel, plan de tests, résultats de tests, code source et documentation usager)	25	n/a

4.2 Suivi de projet et contrôle

4.2.1 Gestion des exigences

L'objectif de ce projet est de démontrer une preuve de concept avec un prototype de magasin utilisant EOS. Nous avons donc choisi les exigences essentielles et souhaitables en fonction des fonctionnalités les plus importantes à développer pour ce prototype.

Les exigences essentielles devraient toutes être satisfaites si possible. Elles seront quand même travaillées en ordre d'importance afin de prioriser les exigences les plus importantes. En cas de problème ou de blocage sur une des exigences, l'équipe est responsable d'aider le(s) développeur(s) travaillant sur celle-ci. Si le problème persiste, il est discuté avec le client lors des réunions bihebdomadaires et une décision est prise selon le cas, par exemple rendre l'exigence souhaitable. Les exigences souhaitables seront travaillées si le temps le permet. Lorsque toutes les exigences essentielles seront finies, une réunion avec le client permettra de déterminer quelles exigences souhaitables sont les plus prioritaires. Lors d'un changement d'exigence avec le client, l'équipe se réunira pour ajuster le calendrier de développement et attribuer les nouvelles tâches.

4.2.2 Contrôle de la qualité

Chaque fonctionnalité est développée dans une branche Git qui lui correspond. Avant d'appliquer les changements sur la branche "dev", les changements doivent être confirmés par un autre membre de l'équipe à l'aide d'un "pull request", afin d'avoir une vérification additionnelle sur le code. Lors du "push" sur la branche "dev", les "commits" sont combinés (squashed) pour en former un seul, ceci permet de facilement et rapidement enlever les

changements s'ils posent un problème. Lors du développement dans une branche quelconque, le développeur doit souvent tirer les modifications de la branche "dev" afin de s'assurer d'avoir la dernière version des fonctionnalités, ce qui évite de devoir régler un gros conflit peu de temps avant la remise d'un livrable.

Si un problème se retrouve sur la branche "dev", la personne ayant introduit le changement causant le problème doit s'occuper de créer une nouvelle branche qui offre une solution au problème; la priorité doit alors être de fusionner cette branche avec "dev". Cette nouvelle branche doit cependant aussi passer par un pull request pour éviter d'introduire encore davantage de problèmes. Le code de "dev" doit être fusionné sur "master" (la branche des livrables) au moins deux jours avant la date du livrable si possible. De cette manière, le build du livrable peut être testé une dernière fois et il reste un peu de temps pour entreprendre des actions correctives si nécessaire. Aucune grande fonctionnalité ne devrait être ajoutée à master moins de deux jours avant la remise d'un livrable.

Grâce à cette procédure, les correctifs seront faits au fur et à mesure du développement logiciel, ce qui garantit une meilleure qualité des biens livrables

4.2.3 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 : 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.

R1 - Risque concernant la confidentialité				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
6	En tant que service de vente et achat de biens et services, il est important que les informations d'un utilisateur restent confidentielles.	M	Confidentialité	S'assurer que toute information envoyée au serveur et stockée dans la base de données, soit encryptée.

R2 - Risque lié à une nouvelle technologie				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
8	L'équipe devra apprendre à utiliser plusieurs nouvelles technologies pour le projet (EOS, SWAP, contrats intelligents…). Nos estimations d'effort aux tâches associées à ces nouvelles technologies sont donc approximatives et des problèmes que nous n'avons pas prévus d'utilisation de ces technologies pourraient survenir.	M	Temps	Faire toute recherche nécessaire avant d'entamer une tâche. Répartir les membres de l'équipe sur les tâches dépendamment de l'effort demandé. En cas où le travail sur une technologie est sous-estimé, il est possible de séparer la tâche et de la partager entre plusieurs membres de l'équipe.

R3 - Disponibilité du client				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
5	Ce projet étant fait à partir de la vision du client, il est possible que celui-ci soit très occupé pendant la session et n'ait pas toujours le temps de communiquer avec notre équipe.	M	Temps	S'assurer de comprendre la tâche en communiquant avec le client tôt dans l'avancement du projet, afin d'éviter de garder des questions pour la fin du projet.

R4 - Manque de documentation				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
4	Bien qu'il existe d'autres applications utilisant la blockchain EOS, peu sont similaires à ce projet. De plus, EOS n'est pas aussi connu que Ethereum et il est possible qu'il soit difficile de trouver de la documentation pour certaines de ces fonctionnalités.	M	Temps	Partager toute documentation utile avec l'équipe et trouver des exemples d'applications utilisant les mêmes technologies. Demander de l'aide au client si besoin, qui a généralement une bonne base de connaissance sur le sujet.

4.2.4 Gestion de configuration

Les problèmes détectés dans le code doivent être fixés dès que possible, tel que décrit dans la section 4.2.2 de ce document. En cas de changement dans un document à remettre, celui qui fait le changement est responsable de mettre à jour le numéro de version du document et d'ajouter le changement dans l'historique de révision. Si le changement est majeur, il est aussi responsable de notifier le reste de l'équipe du changement.

La numérotation des artéfacts suit le format <majeur>.<mineur> où la version majeure est associée au livrable (initial, mi-session, bêta, final…) en commençant par 1 à la première remise et en étant incrémentée pour chaque nouvelle remise. La version mineure fait référence au numéro de changement dans un même livrable et doit donc être incrémentée à chaque changement dans le document, puis retourne à 0 entre chaque livrable.

Pour ce qui est du code, chaque code pushed sur "master" sera associé à une version à l'aide de "git tag". Lorsque du nouveau code est pushed pour un nouveau livrable, le tag Git sera <majeur>.<mineur> avec le majeur dépendant du prochain livrable et mineur étant 0 au départ. Si un correctif doit être fait avant la remise, le commit associé au correctif sur "master" garde la même version majeure, mais incrémente sa version mineure. Pour chaque livrable, le commit final associé sera donc la dernière version mineure disponible pour le majeur du livrable.

Cela permet de se rappeler exactement chaque commit correspond à quelle remise sur la branche "master".