Institut National des Sciences Appliquées et de Technologie

UNIVERSITÉ DE CARTHAGE

Projet de Fin d'année

Filičre : **GL**

E-voting on blockchain

Présenté par

Yasmine Sidhom
Marouane Elkamel

Encadrant : **Mr YOUSFI Souheib**

Présenté le : **24/05/2018**

JURY

Mme. Lilia Sfaxi (Présidente)

Année Universitaire : 2017/2018

Remerciements

Nous adressons nos remerciements aux personnes qui nous ont soutenus dans la réalisation de notre projet de fin d'année et de notre rapport.

Nous tenons à remercier particulièrement Mr Souheib Youssfi, professeur encadrant, qui a assisté notre travail tout au long du deuxième semestre. Nous le remercions pour ses conseils et pour le temps qu'il nous a consacré.

Merci à Mme Marwa Chaieb, doctorante à l'INSAT, pour sa précieuse collaboration en acceptant de répondre à nos questions et en nous aidant à nous documenter sur le domaine de la BlockChain.

Merci à Mme Lilia Sfaxi qui nous a fourni des modèles et des extraits sur lesquels nous nous sommes appuyés dans l'élaboration du présent rapport.

Nous remercions aussi toutes les personnes, qui ont donné leur avis et également celles qui ont corrigé et qui ont aidé à la relecture de notre rapport.

Table des Matières

Liste des Figures

Liste des Tableaux

Résumé

L'application « *InsatVote* »a été mise en oeuvre dans le cadre d'un projet de fin d'année. L'étude, ainsi que la réalisation de ce projet s'est déroulée du mois de janvier à mai 2018. Elle a été menéé par « Yasmine Sidhom »et « Marouane Elkamel », auteurs de ce rapport et étudiants en deuxième année du cycle ingénieur en Génie Logiciel et ce, sous la supervision du professeur-encadrant Mr Souheib Youssfi.

« *InsatVote* »est une application web de vote électronique, destinée à être utilisée au sein de l'institut (INSAT) pour l'élection des responsables des différents départements.
Elle a été pensée pour être un outil de vote à la fois décentralisé et sécurisé, intégrant un maximum de fonctionnalités et digitalisant la procédure utilisée actuellement.

L'étude conceptuelle de l'application « *InsatVote* »a porté sur les 3 axes suivants :

— L'enregistrement des votant sur le site,
— Le processus de vote en ligne intégré à la BlockChain,
— La facillité d'administration du vote (côté administrateur).

Le protocole sous-jacent repose sur des primitives cryptographiques (dont notamment, la cryptographie à base de courbes elliptiques) et sur le réseau Blockchain Ethereum [1], offrant ainsi une sécurité accrue et plus de transparence dans les élections.
Ce protocole fait intervenir :

— Un administrateur
— Un ensemble de candidats
— Un ensemble de votants
— Un contrat de vote

Où chaque votant représente un compte sur le réseau Ethereum. Le Smart Contract (englobant toute la logique métier du processus de vote) est déployé sur un réseau Ethereum de test.

Le présent rapport est une synthèse des différentes étapes développées dans la conception et l'élaboration de ce projet. Ce travail a été effectué en appliquant les principes de la méthodologie Agile et en adoptant le principe de génie logiciel « Intégration Continue ».

1. Les courbes elliptiques sont utilisées en Blockchain pour des opérations asymétriques comme la génération de clés publiques à partir de clés privées via un algorithme cryptographique unilatéral à courbe elliptique (Bitcoin). Le réseau Ethereum utilise la courbe ECDSA (Elliptic Curve Digital Signature Algorithm) pour valider l'origine et l'intégrité des messages échangés.

Introduction Générale

Comme son nom l'indique, « *InsatVote* »est une application de vote électronique qui repose sur la technologie Blockchain. Plus précisément, « *InsatVote* »est destinée à être utilisée dans les élections des chefs de départements au sein de l'institut.

Nous avons choisi le sujet de notre application parmi ceux qui ont été proposés comme projets de fin d'année pour les étudiants en 4ème année Génie Logiciel. L'idée derrière ce travail est de révolutionner entièrement le processus de vote en proposant une solution digitale comme alternative au systèmes actuels. Cette solution en ligne vient remédier aux différentes limitations imposées par le vote manuel et qui portent notamment sur :

- la nécessité de se déplacer pour voter.
- les ressources à consacrer spécialement pour la mise en oeuvre des élections (temps, argent ainsi que la mobilisation de nombreuses personnes physiques).
- la présence d'une autorité centrale qui gère tout le processus (mise en place, supervision et comptage des voix) : la sécurité et l'intégrité des élections ne sont ainsi pas toujours au rendez-vous.

Le présent rapport, scindé en 4 principales sections citées ci-dessous est une synthèse du travail de recherche, de spécification, de conception et d'implémentation qui a été réalisé tout au long du projet.

Le présent document est organisé comme suit :

1. Une étude théorique du projet : C'est le premier chapitre du rapport. Il contient une étude de l'art.
2. Spécifications du système : Dans cette partie nous nous sommes intéressés à l'étude des exigences requises de notre système, du protocole de vote ainsi qu'à l'environnement destiné à accueillir notre application.
3. Conception : C'est la phase la plus importante dans la réalisation du projet. Elle comporte deux grandes lignes :
   - L'analyse : qui modélise les fonctionnalités de l'application
   - La conception : qui est à une fois une réalisation de l'analyse et une représentation de la structure du code à générer.
4. Réalisation : Ce chapitre spécifie l'environnement technologique de l'application et présente les résultats de l'implémentation de celle-ci.
5. Conclusion : Ce chapitre résume le travail décrit dans le rapport et dresse les perspectives du projet.

Chapitre I

Etude Théorique

Plan

Introduction

Dans ce chapitre, nous présentons l'état de l'art où nous faisons une étude comparative des différentes méthodes de vote et de notre application.

1 Les méthodes de vote existantes

1.1 Le vote sur papier

Ces systèmes largement utilisés présentent plusieurs déficiences :

— Des coûts importants liés à la mobilisation de milliers de personnes.

— Le vote manuel dure longtemps en raison des différents actes à poser.

— Les résultats des élections ne peuvent être connus rapidement, mais dans le meilleur des cas le lendemain du vote.

— Le vote sur papier est sensible à la fraude.

— La taille du bulletin de vote peut également constituer un inconvénient pratique.

— Les mesures de sécurité préventives sont souvent insuffisantes pour maintenir la confiance.

1.2 Les systèmes électoraux électroniques (e-voting)[1]

Il existe dans la littérature une panoplie de systèmes de vote électronique. Il peut s'agir de :

Simples ordinateurs de vote : Les électeurs doivent se rendre dans leur bureau de vote habituel où ils votent à l'aide d'ordinateurs.

Kiosques à voter : Les électeurs peuvent voter depuis n'importe quel bureau de vote puisque la gestion des émargements est réalisée par un serveur central. Ce serveur recueille également les choix des électeurs au fur et à mesure du déroulement de la journée.

Vote par internet (i-voting) : Les électeurs se connectent sur un site officiel de vote depuis n'importe quel ordinateur. Ils doivent alors s'identifier (donner leur identité) et s'authentifier (prouver leur identité), avant de voter. Un serveur recueille les votes et les stocke jusqu'à la clôture du scrutin. Il produit les résultats du vote à la clôture du scrutin.

À première vue, le vote électronique semble :
— Nécessiter un nombre moins élevé d'assesseurs, les frais en sont alors plus réduits.
— Permettre un dépouillement plus rapide et plus fiable.
— Permettre d'éliminer les bulletins de vote trop imposants.
— Simplifier le processus et le rendre plus pratique.
— Accroître l'accessibilité du processus et favoriser la participation électorale.

Mais tous les systèmes de e-voting présentent des vulnérabilités. Nous avons examiné les solution précitées et repéré des problèmes apparents à différents niveaux :

Fiabilité : La grande faiblesse des ordinateurs de vote reste que la plupart ne produisent pas de preuve de vote vérifiée par l'électeur, leurs résultats sont donc invérifiables.

Anonymat : Coté serveur, il est délicat de garantir l'anonymat puisque chaque bulletin voyage accompagné de l'identité du votant et que ces informations parviennent ensemble sur un serveur de vote.

Transparence et confiance : Il faut prouver que le programme en service est exactement identique à celui qui a été publié, et qu'il n'y a pas d'intrusion d'autres programmes. Or, actuellement, la surveillance de l'application de vote est, dans le meilleur des cas, confiée à des tiers, ce qui amoindrit la confiance des électeurs.

Confidentialité : Dans certains systèmes, des standards cryptographiques dépassés sont encore utilisés. Dans d'autres cas encore, les identifiants et mots de passe sont envoyés au

domicile des électeurs par courrier non recommandé, ce qui ne garantit pas que les informations nécessaires à la connexion soient délivrées à leur destinataire.

Sécurité des systèmes : Des fraudes peuvent être menées par une ou plusieurs personnes impliquées dans l'organisation du vote. Il peut s'agir d'un programmeur ou d'un technicien chargé de la maintenance et des mises à jour. Les attaques internes sont les plus graves car elles sont faciles à mener et elles peuvent rester complètement invisibles.

Qualité des logiciels de vote : Le code du logiciel peut compromettre le processus électoral s'il est de mauvaise qualité. Pour exemple, le fameux bug des élections bruxelloises en 2014 trouve une partie de son origine dans les lacunes du logiciel *Jites*.

Sûreté du matériel : Tout ordinateur peut connaître des pannes ou des dysfonctionnements matériels. Des pannes de courant par exemple ont perturbé le dépouillement à Bruxelles en 2014.

2 Notre application

Nous avons développé une solution de vote par internet (ou i-voting) qui répond aux caractéristiques cités ci-dessous (ces caractéristiques seront détaillées dans le chapitre suivant) :

- Transparence
- Unicité
- Secret de vote
- Authentification
- Vérifiabilité individuelle
- Anonymat
- Intégrité
- Pas de résultat partiel
- Non-répudiation

Conclusion

Tout au long de ce chapitre, nous avons pu donner une présentation claire du cadre du projet. Dans le chapitre suivant, nous nous intéressons au protocole régissant notre application.

Chapitre II

Spécifications du système

Plan

Introduction

Dans ce chapitre, nous présentons le diagramme des cas d'utilisation et les différentes étapes du processus de i-vote. Ensuite, nous abordons le contexte matériel, logiciel et humain de notre projet.

1 Diagramme des cas d'utilisation

Notre système fait intervenir 3 acteurs :

— Un administrateur

— Un internaute anonyme.

— Un votant : un votant est un internaute anonyme qui s'est enregistré sur notre site.

— Un acteur secondaire : Le Smart Contract de vote

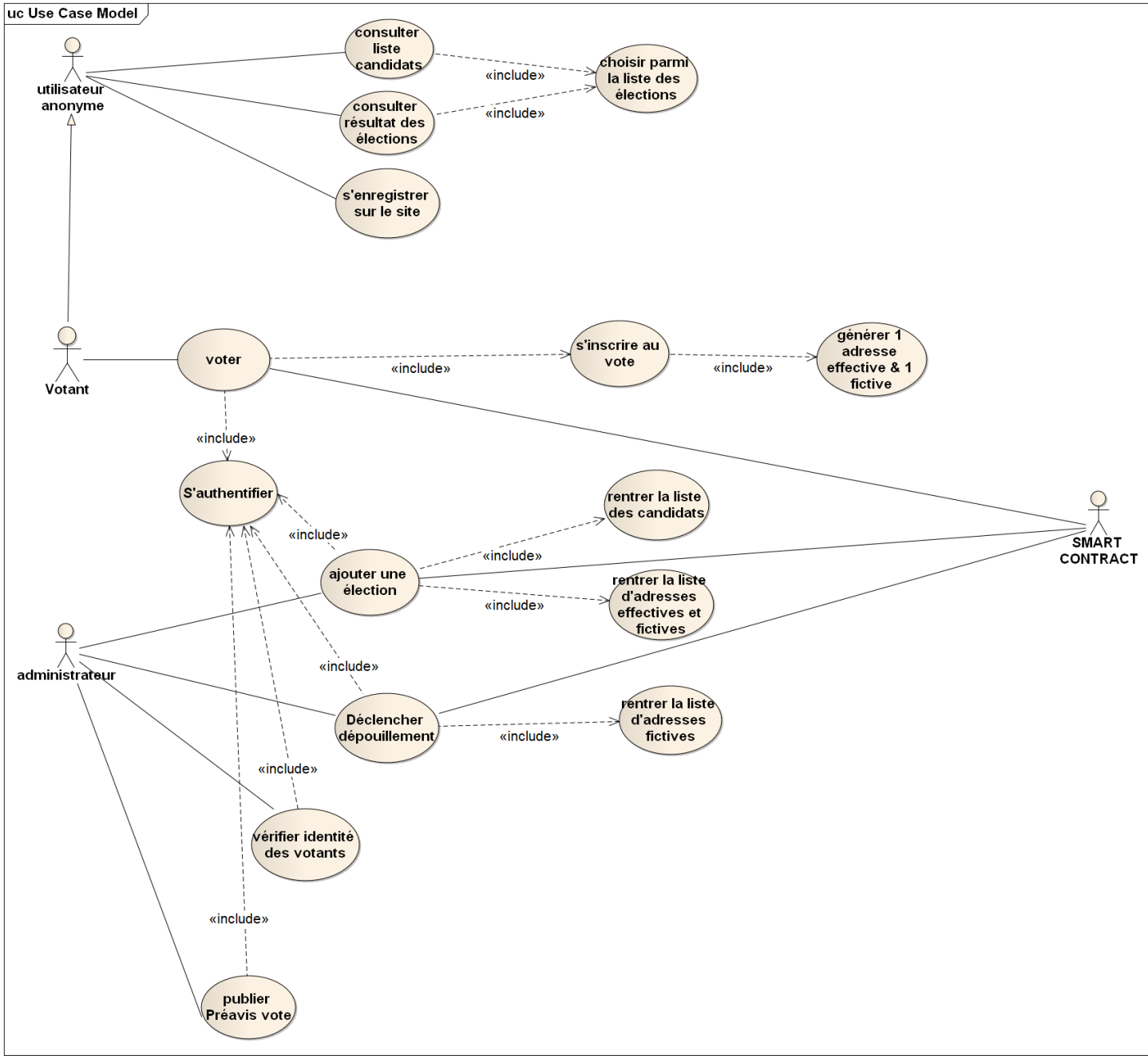


Figure II.1 – Diagramme des cas d'utilisation

Chaque cas d'utilisation modélise une fonctionnalité du système. Dans le paragraphe suivant, nous allons présenter le protocole qui permet d'assuer ces fonctionnalités.

2 Protocole

Nous avons créé une application décentralisée (ou « DApp » [1]) de vote électronique déployée sur le réseau Ethereum et permettant de participer à plusieurs élections de chefs de départements de l'INSAT. Ces élections sont créées par l'administrateur de la plateforme.

Voici, de manière générale et non spécifique à notre application, une infographie modélisant les étapes et éléments de base pour un processus de e-voting déployé sur Ethereum :

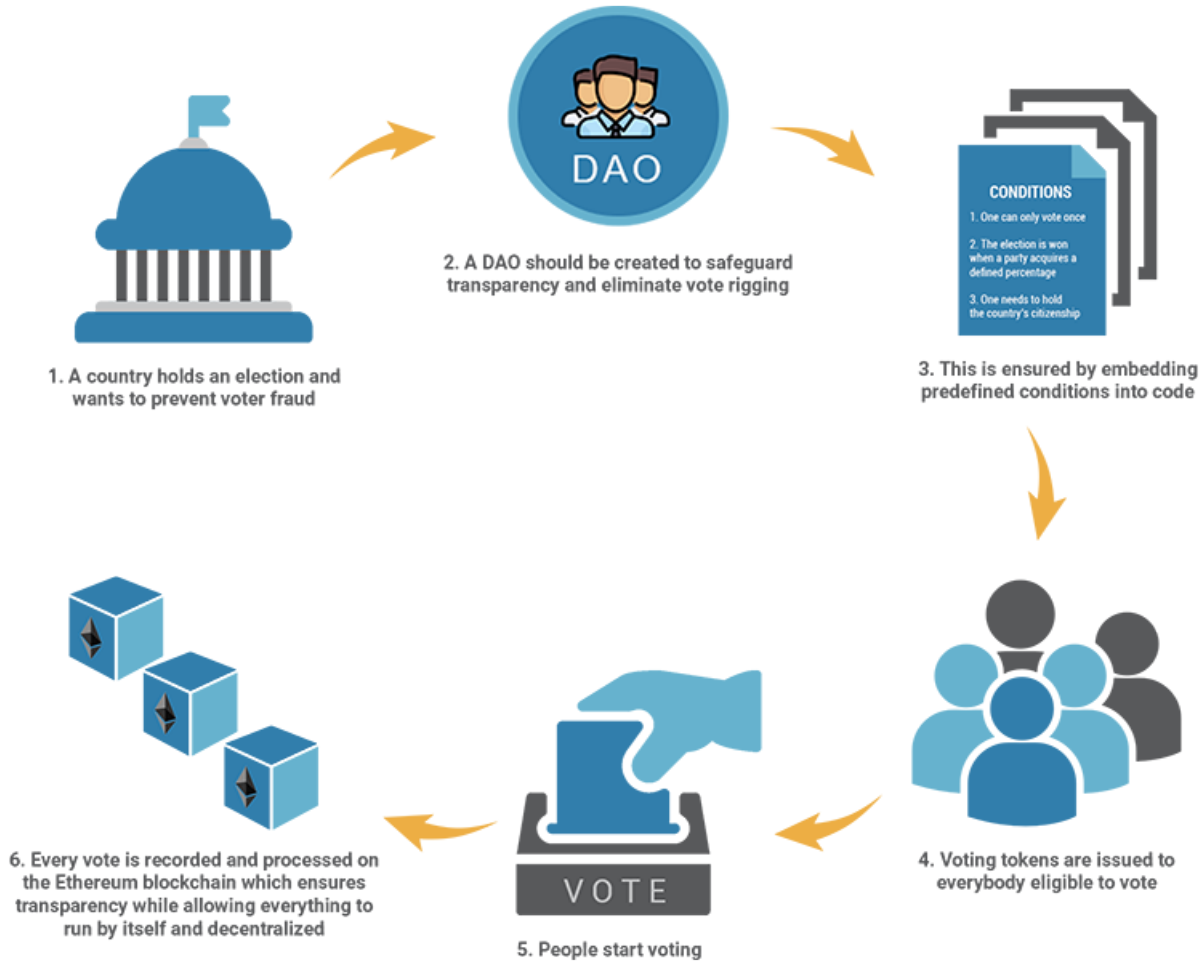


Figure II.2 – Protocole général [5]

Le protocole que l'on a élaboré est scindé en 7 phases. Nous vous présentons ci-dessous les

1. Est considérée comme DApp toute application complètement open-source, qui fonctionne de manière autonome et sans entité centrale qui la contrôle et dont les données et les enregistrements sont stockés cryptographiquement dans une BlockChain décentralisée

différentes phases et le cas d'utilisation (UC) correspondant.
Vous pouvez ainsi vous référer au diagramme des cas d'utilisation représenté plus haut.

2.1 Enregistrement

UC [2] : « S'enregistrer sur le site »

Cette phase consiste en la vérification de l'identité de l'utilisateur.
Un utilisateur anonyme peut s'enregistrer sur notre plateforme afin de pouvoir participer en tant que votant aux élections qui y seront créées dans le futur.
Un courriel lui sera envoyé par la suite comportant un nombre aléatoire qui a été automatiquement généré par notre plateforme, appelé « *code de vérification* ».
L'utilisateur doit se déplacer pour présenter ce code à l'administrateur du site. Celui-ci se chargera alors de vérifier manuellement ses informations personnelles et d'activer son compte.
L'utilisateur dispose désormais d'un compte vérifié sur notre plateforme. Il s'en servira à chaque fois qu'il voudra participer à un vote pour lequel son compte votant a été sélectionné comme « *votant éligible*».
Autrement dit :

- L'enregistrement se fait en amont. C'est une phase préalable essentielle pour que le votant puisse intégrer le système de vote que l'on a créé.
- Le compte activé est un compte permanent, qui permet à son propriétaire de participer à tous les votes qui le concernent à partir du même compte (avec la même identité).

2.2 Réunion et mise en place d'un vote

UC : « Publier préavis vote »

Cette phase se fait en grande partie hors ligne.
Suite à une réunion des personnes concernées par la création du vote avec l'administrateur du site, le sujet du vote ainsi qu'une liste de candidats et de votants éligibles est élaborée. (Dans notre cas : les enseignants membres d'un département A qui souhaitent élire un chef de département se réunissent avec l'administrateur. La liste des votants éligibles comprend tous les membres du département A enregistrés sur le site).
Suite à cela, l'administrateur est tenu de publier le sujet du vote, la liste des candidats et celle des comptes votants éligibles sur la plateforme.

2. Cas d'utilisation correspondant

2.3 Génération d'adresses publiques pour les votants

UC : « générer 1 adresse effective et 1 adresse fictive »

Les personnes qui ont été ajoutées à la liste des votants éligibles sont notifiées sur l'interface utilisateur du site. Elles sont invitées à créer deux adresses publiques qui leur serviront pour voter de manière anonyme et d'intégrer la BlockChain.
L'une des deux adresses servira comme adresse «Effective» et l'autre comme adresse «Fictive» (voir plus de détails dans le paragraphe 6).
Il est à savoir que :

- Ces adresses sont en principe [3] valides pour le vote courant uniquement.
- La génération d'adresses publiques se fait avec l'outil MetaMask, et ce, en créant 2 comptes Ethereum indépendants.

2.4 Inscription des votants

UC : « S'inscrire au vote »

À l'inverse de la phase d'enregistrement qui est durable, la phase d'inscription ne concerne que le vote auquel l'utilisateur a été invité à participer.
Cette inscription n'est accessible que depuis le volet « Administrateur » du site. Ainsi, pour accomplir cette phase, un votant doit se déplacer à l'INSAT où il pourra rentrer ses deux adresses dans les cases correspondantes : la case adresse « Effective » et la case adresse « Fictive ». Seuls l'administrateur et le votant connaissent la nature de chaque adresse.
À la suite de cette inscription, deux fichiers seront générés côté administrateur :

- Le fichier des adresses (effectives et fictives) éligibles à voter.
- Le fichier des adresses fictives qui servira pour le comptage des voix (plus de détails seront fournis dans le paragraphe 7 consacré à la phase de dépouillement).

2.5 Création du vote et déploiement du Smart Contract

UC : « ajouter une élection »

Une fois les votants inscrits, l'administrateur crée le vote sur la plateforme.

3. Un votant peut décider d'utiliser d'anciennes adresses générées lors d'un vote précédent. Rien ne l'en en empêche. Mais rien ne l'y oblige non plus.

La création d'un vote se fait sur l'interface administrateur et donne lieu au déploiement d'un SMART CONTRACT (dont la structure sera expliquée plus tard) avec pour paramètres :

— La liste des candidats telle qu'établie depuis la phase «réunion et mise en place du vote»
— La liste des adresses publiques éligibles pour le vote (effectives et fictives).

2.6 Vote

UC : « voter »

À l'approche ainsi qu'au commencement de la session de vote, les votants éligibles sont notifiés sur le site (interface utilisateur).
Le vote se fait directement sur le site ou à travers l'outil Metamask. Les deux interfaces permettent de transmettre les choix des votants de manière transparente au Smart Contract.
Pour voter, un votant doit sélectionner l'adresse du réseau Ethereum surlequel est déployée notre application et utiliser ses deux comptes comme suit :

— Avec le compte liée à son adresse effective, il vote pour le candidat A de son choix.
— Avec le compte lié à son adresse fictive, il vote pour un candidat B quelconque.

Ainsi, le protocole impose de voter à deux candidats à partir de deux comptes différents, tout en sachant qu'un vote ne peut être modifié et que seul le vote effectué à partir du compte de l'adresse effective sera retenu.

2.7 Dépouillement

UC : « Déclencher dépouillement »

C'est la phase finale qui va permettre de calculer et d'afficher les résultats du vote.
Les fonctions « RevokeRightToVote » et« WinnerName» du Smart Contract, activées explicitement par l'administrateur à la fin de la session de vote, sont responsables du calcul des résultats.
La fonction « RevokeRightToVote » prend en paramètres la liste des adresses fictives. Comme son nom l'indique, elle annule les votes qui ont été transmis à partir des comptes liés aux adresses fictives. Les résultats du vote (désormais effectifs) seront par comptabilisés à l'aide de la fonction « WinnerName ».
L'annulation des votes fictifs se fait une fois que la session de vote a été clôturée. Ainsi, en observant les transactions effectuées sur la BlockChain et en arrivant éventuellement à obtenir la liste des candidats avec le nombre de voix associés et les adresses qui leur ont voté, personne

(sinon l'administrateur) ne peut prédire les résultats. Il est donc impossible pour toute autre personne de déduire un résultat partiel.

Caractéristiques du processus de vote :

Nous avons développé une DApp (application décentralisée déployée sur la BlochChain Ethereum) qui permet le vote en ligne tout en veillant à garantir les spécificités de sécurité et les services énoncés au cahier des charges :

- **Transparence** : chaque électeur a le droit et la possibilité effective de contrôler toutes les étapes d'un scrutin puisque le contrat de vote est déployé sur la blockchain (toute les transactions effectuées depuis et vers ce contrat sont donc traçables).
- **Unicité** : un seul vote par électeur sera retenu.
- **Secret de vote** : chaque électeur peut effectuer son choix en secret. Même s'il ne se trouve pas seul devant l'écran, personne d'autre que lui ne connait la nature de chaque compte qu'il va utiliser (compte d'adresse effective ou fictive). Même pour l'administrateur qui dispose d'une information sur la nature de chaque adresse publique, il ne peut pas relier ces adresses aux comptes Ethereum d'où elles ont été générées.
- **Authentification** : Notre système garantit qu'il n'y ait pas d'usurpation d'identité (grâce au code de vérification).
- **Vérifiabilité individuelle** : Chaque candidat peut, en supervisant les transactions émises en direction du Smart Contract, vérifier que son adresse fictive et non son adresse effective a été utilisée pour supprimer son vote fictif.
- **Anonymat** : il est impossible de relier un candidat à l'électeur qui l'a choisi. La manière avec laquelle les votes sont cryptés sur la Blockchain rend impossible la violation du secret du vote. Ce critère est garantit sous réserve de faire confiance à l'administrateur.
- **Intégrité** : les résultats du scrutin reflètent fidèlement la volonté des électeurs puisque :
  - Les votes sont envoyés sur la Blockchain.
  - Ainsi est écrit le code du Smart Contract et il le restera toujours (les Smart-Contracts sont immuables une fois déployés sur la Blockchain).
- **Pas de résultat partiel** : il est impossible de procéder à un comptage partiel des voix ou de prédire qui sera élu puisque la méthode « RevokeRightToVote »ne peut être appelée qu'à la fin du vote et par l'administrateur uniquement.
- **Non-répudiation** : un votant ne peut pas nier avoir participé à un vote puisque :
  1. Toutes les adresses publiques possibles ont été sauvegardées au moment de l'inscription. Il ne peut donc pas participer à partir d'un autre compte non inscri.

2. Toutes les transactions de vote ainsi que l'espace de stockage du Smart Contract sont visibles sur la blockchain.
3. Dans le code du Smart Contract, chaque fois qu'un vote est effectué à partir d'un compte Ethereum, la variable "voted" de l'entité "Voter" associée est mise à "true".

3 Contexte du projet

3.1 Environnement matériel

Du point de vue matériel, la plateforme à mettre en œuvre est simple : mis à part le serveur web et les appareils électroniques ayant accès à internet (ordinateur, tablette ou même Smartphone pour les votants et ordinateur pour l'administrateur), aucun autre équipement n'est requis.

3.2 Environnement logiciel

Du point de vue logiciel, notre application est hébergée sur le serveur Web « Tomcat »de SpringBoot, et le Smart Contract de vote qu'elle utilise est déployé sur la Blockchain Ethereum (plus précisément, sur le réseau de test RPC local). Le fonctionnement de « *InsatVote* »exige que l'extension « MetaMask » soit installée sur le navigateur.
Cependant, les informations publiques telles que la liste des candidats et les résultats de vote sont accessibles pour tout internaute normal (il n'a pas besoin d'installer « Metamask »).
La présence d'un administrateur est requise pour l'administration de la plateforme et la supervision des différentes étapes du protocole.

3.3 Environnement humain

Les seuls utilisateurs et acteurs de notre plate-forme sont l'administrateur, les votants qui disposent de comptes activés et les internautes anonymes qui peuvent y consulter les informations publiques sur les votes ouverts.

Conclusion

Dans ce chapitre, nous avons spécifié les fonctionnalités attendues de notre système. Cette spécification va servir dans le prochain chapitre à trouver le modèle conceptuel le plus adapté à notre protocole.

Chapitre III

Conception

Plan

Introduction

Dans ce chapitre nous vous présentons une première section détaillant l'analyse de notre application « *InsatVote* », une seconde qui constitue la conception proprement dite de celle-ci et une troisième qui représente l'architecture du système ainsi conçu. Nous abordons également les méthodologies suivies dans son développement.

1 Principes et méthodologie

Dans la conception et le développement de notre projet, nous avons adopté la méthodologie agile et le principe d'intégration continue.

1.1 L'intégration continue [2]

L'intégration continue est un ensemble de pratiques utilisées en génie logiciel dont le principal but est de détecter les problèmes d'intégration au plus tôt lors du développement.

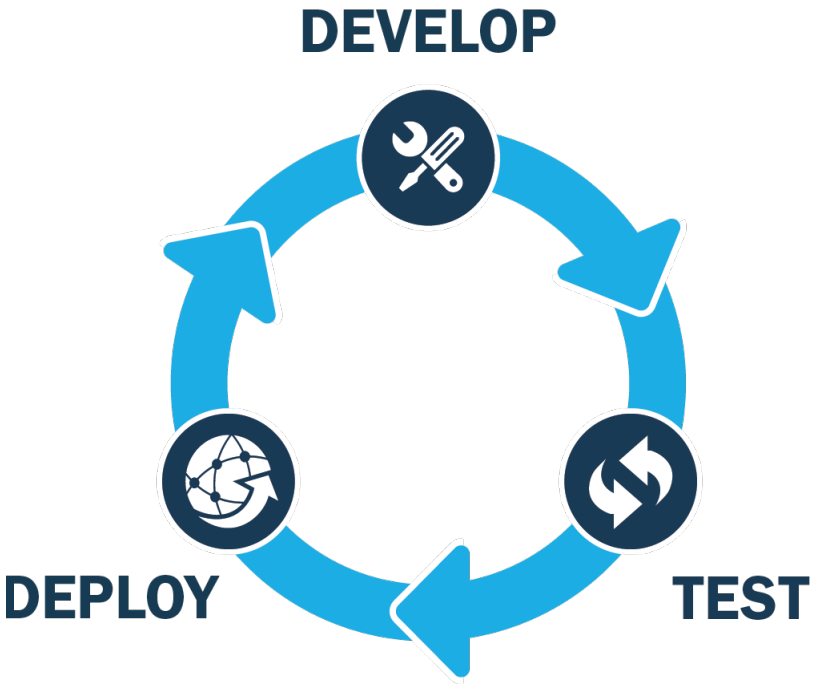


Figure III.1 – Boucle de l'Intégration Continue [6]

L'intégration continue repose souvent sur la mise en place d'une brique logicielle permettant l'automatisation de tâches : compilation, tests unitaires et fonctionnels, validation produit, tests de performances... Pour cela, il faut d'abord que :

- Le code source soit partagé (en utilisant des logiciels de gestion de versions : **Git** dans notre cas).
- Les développeurs intègrent (commit) quotidiennement leurs modifications.
- Des tests d'intégration soient développés pour valider l'application.
- Un outil d'intégration continue soit utilisé, tel que **Travis CI** dans notre cas.
- Des outils, comme **SonarQube** soient mis en place afin de superviser la qualité du code.

Tous les outils d'intégration continue précités seront présentés en détail dans la section dédiée du chapitre « Réalisation »

1.2 La méthodologie Agile [3]

La méthodologie Agile est définie par opposition aux approches traditionnelles prédictives et séquentielles (de type cycle en V ou en cascade).
Elle vise à accélérer le développement du logiciel en développant une version minimale, puis en intégrant les fonctionnalités une à une par un processus itératif basé sur une écoute client et

sur des tests tout au long du cycle de développement.

1.3 L'approche Scrum

Nous avons choisi de travailler avec la méthode SCRUM qui est la méthode agile la plus utilisée et la mieux documentée parmi toutes celles existantes.
Un **product backlog** nous a été fourni en début de projet. Les **sprints** que nous avons élaboré ont inclu des travaux de conception, de spécification fonctionnelle et technique, de développement et de test. L'attribution des tâches et l'organisation des travaux ont été assurées à travers l'outil de gestion de projet **« Meister Task »**. Nous reviendrons sur cet outil dans le chapitre « Réalisation »

2 Analyse : Diagrammes de séquence

L'analyse a pour but de se doter d'une vision claire et rigoureuse du système à réaliser en déterminant ses éléments et leurs interactions. Nous allons modéliser cela à travers les diagrammes de séquence système (DSS).

2.1 Interactions Administrateur-Systèmes

Ci-dessous le diagramme de séquence correspondant au cas d'utilisation *« Publier Préavis Vote»* :

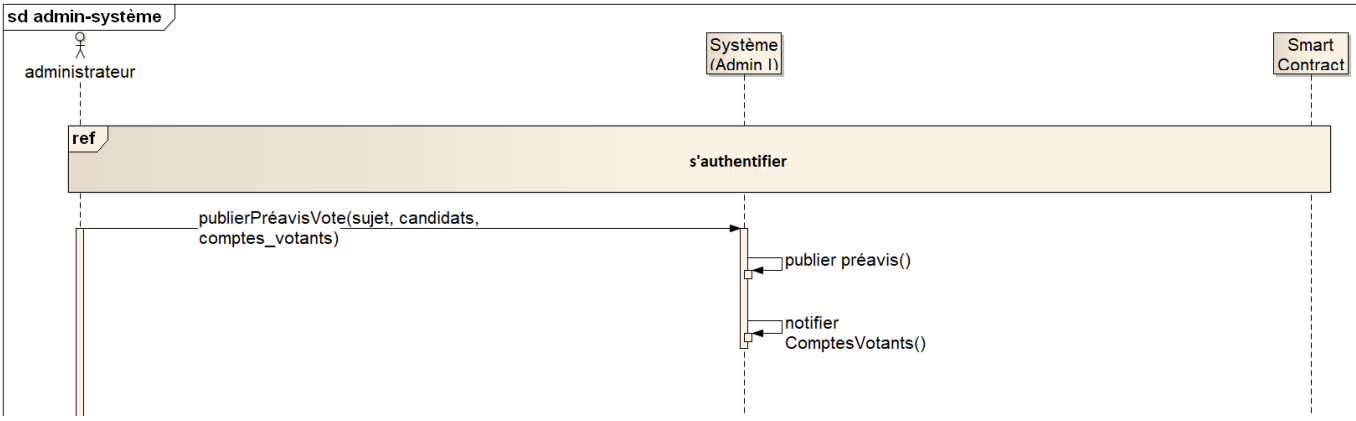


Figure III.2 – DSS Publier Préavis Vote

Publier un préavis de vote est une manière d'inviter les votants à inscrire leurs adresses publiques. C'est une étape essentielle du processus de vote.

Après s'être authentifié, l'administrateur publie à travers l'interface du site un préavis comportant le sujet du vote (exemple : « élection du chef du département Mathématiques ») ainsi que les candidats qui se sont présentés. Il y mentionne les comptes votants concernés et qui pourraient participer. (Ici en l'occurrence : les comptes des membres du département Mathématiques).
Le système se chargera de notifier les comptes votants par la suite.

Ci-dessous le diagramme de séquence correspondant au cas d'utilisation *« Ajouter une élection »* :

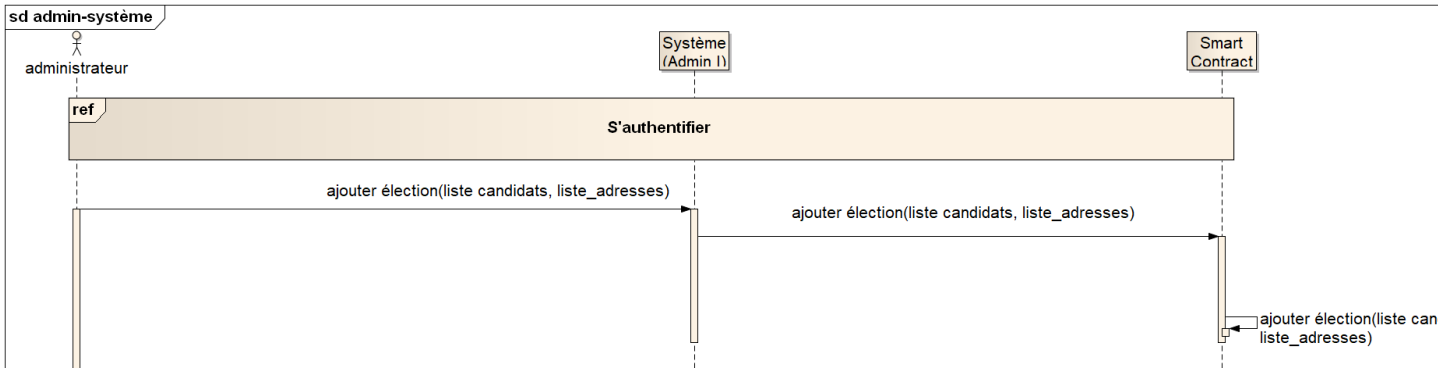


Figure III.3 – DSS Ajouter une élection

L'administrateur doit s'authentifier en premier lieu. Il peut alors, sur l'interface du site, ajouter un vote (ou une élection) en rentrant la liste des candidats et la liste des adresses publiques des votants éligibles (liste globale comprenant adresses effectives et fictives). Celles-ci ont été rentrées préalablement par les votants (voir UC « S'inscrire au vote »). L'interface d'administration communique d'une manière implicite avec la BlockChain. Elle y déploie le contrat de vote avec, en entrée du constructeur, les mêmes données que celles saisies sur l'interface.

2.2 Interactions Votant-Système

Dans tous les types d'interactions qu'on va présenter, le votant doit d'abord s'authentifier pour accéder à l'interface utilisateur.

Ci-dessous le diagramme de séquence correspondant au cas d'utilisation *«S'inscrire au Vote»* :

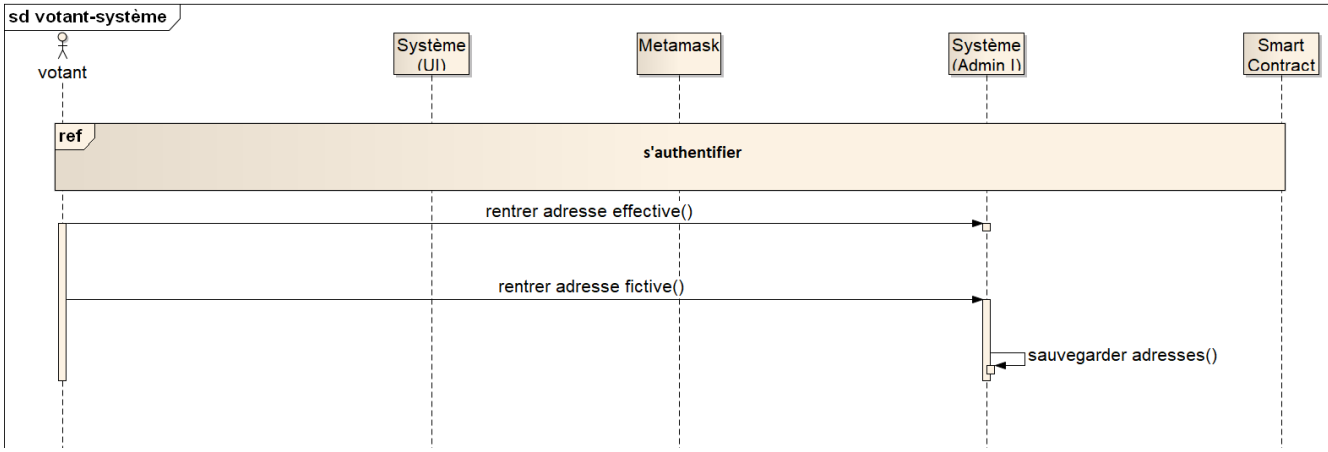


Figure III.4 – DSS S'inscrire au Vote

L'inscription se fait sur l'interface administrateur. Le votant doit donc se déplacer à l'INSAT où il pourra faire rentrer ses deux adresses publiques dans le système. Ce dernier chargera de sauvegarder automatiquement chaque adresse dans le fichier adéquat. Cette opération s'effectue, bien entendu, en la présence de l'administrateur.

Ci-dessous le diagramme de séquence correspondant aux cas d'utilisation *«Voter»* :

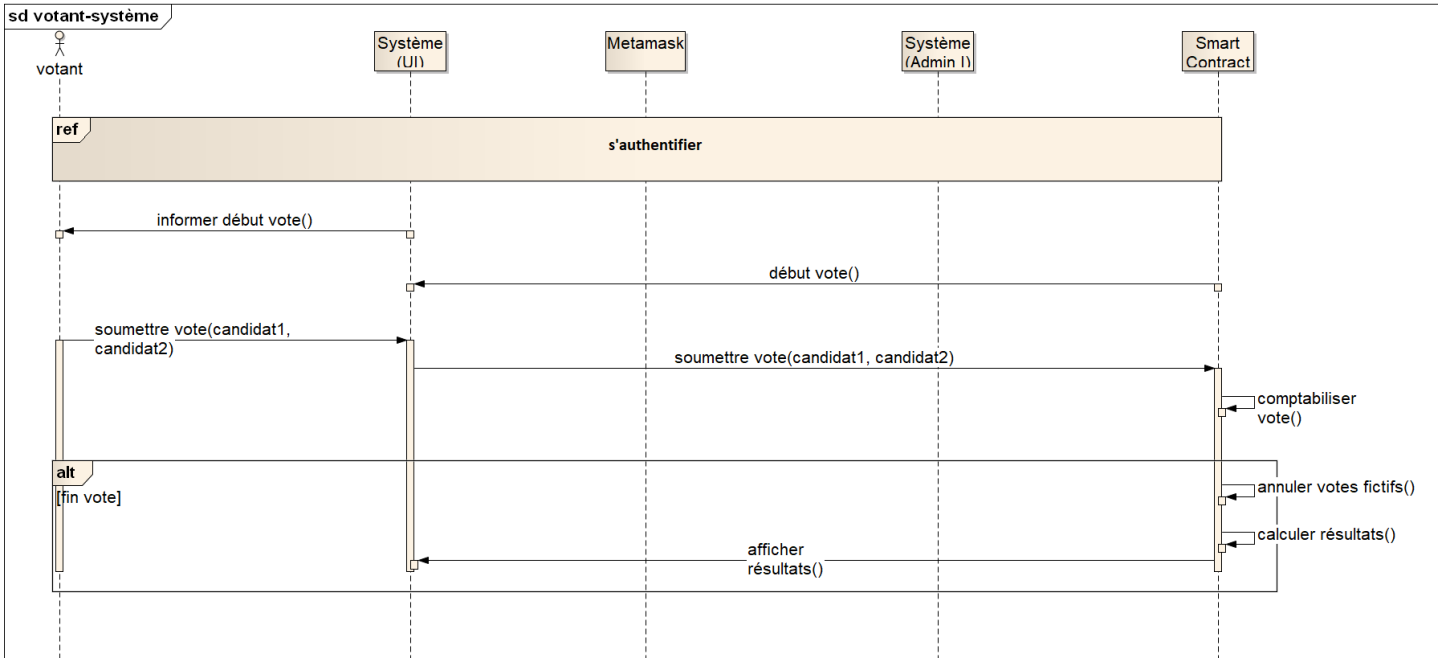


Figure III.5 – DSS Voter

Lorsque le vote est ouvert, le Smart Contract déclenche une notification qui vise la liste des comptes votants qui se sont inscris. Ceux-ci peuvent alors voter.
L'opération de vote se déroule sur 2 étapes consécutives : Dans la première, le votant doit utiliser son compte Ethereum rattaché à son adresse **effective** et voter pour le candidat de son choix.
Dans la deuxième, il doit utiliser son compte lié à son adresse **fictive** et voter pour un autre candidat. Les choix du votant sont automatiquement transmis au Smart Contract qui comptabilise chaque vote en incrémentant de 1 le nombre de voix du candidat concerné.

Puis vient l'étape d'annulation des votes fictifs. Cette action n'est déclenchée que par l'administrateur et qu'à la clôture du vote. Elle s'exécute sur le Smart Contract. (chaque vote fictif décrémente de 1 le nombre de voix du candidat en question).
À la suite de cette action, les résultats du vote sont calculés et transmis au site.

2.3 Interactions Internaute Anonyme-Système

Ci-dessous le diagramme de séquence correspondant au cas d'utilisation *«S'enregistrer sur le site»* :

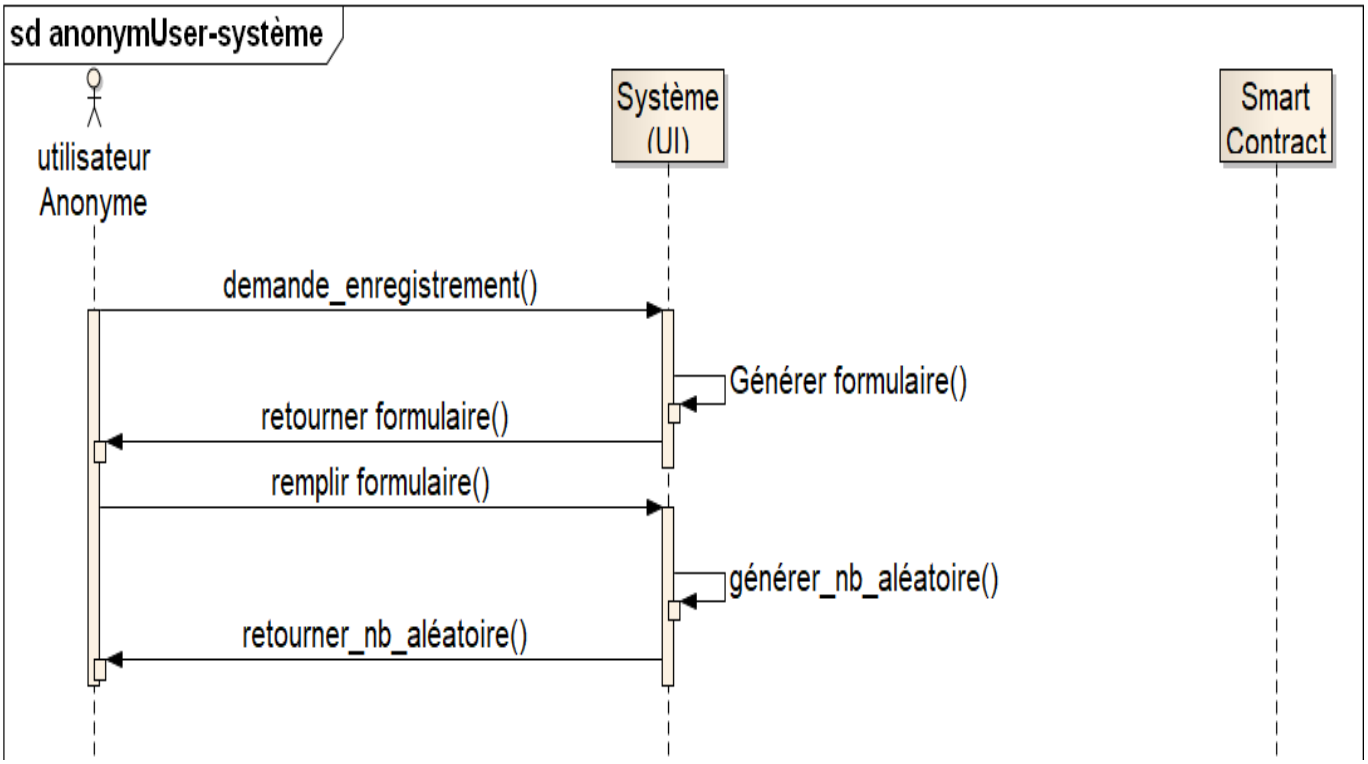


Figure III.6 – DSS S'enregistrer sur le site

Un internaute normal peut demander à s'enregistrer sur le site. Il a alors à remplir un formulaire en indiquant ses informations personnelles et ses informations de connexion. Une fois le formulaire soumis, il reçoit par mail un code de vérification qu'il doit remettre en mains propres à l'administrateur du site.
Le compte qu'il a créé n'est considéré valide qu'après vérification de l'identité du votant et de l'authenticité du code auprès de l'administrateur.

Ci-dessous le diagramme de séquence correspondant aux cas d'utilisation *« Consulter liste candidats »* et *« Consulter Résultats Elections »* :

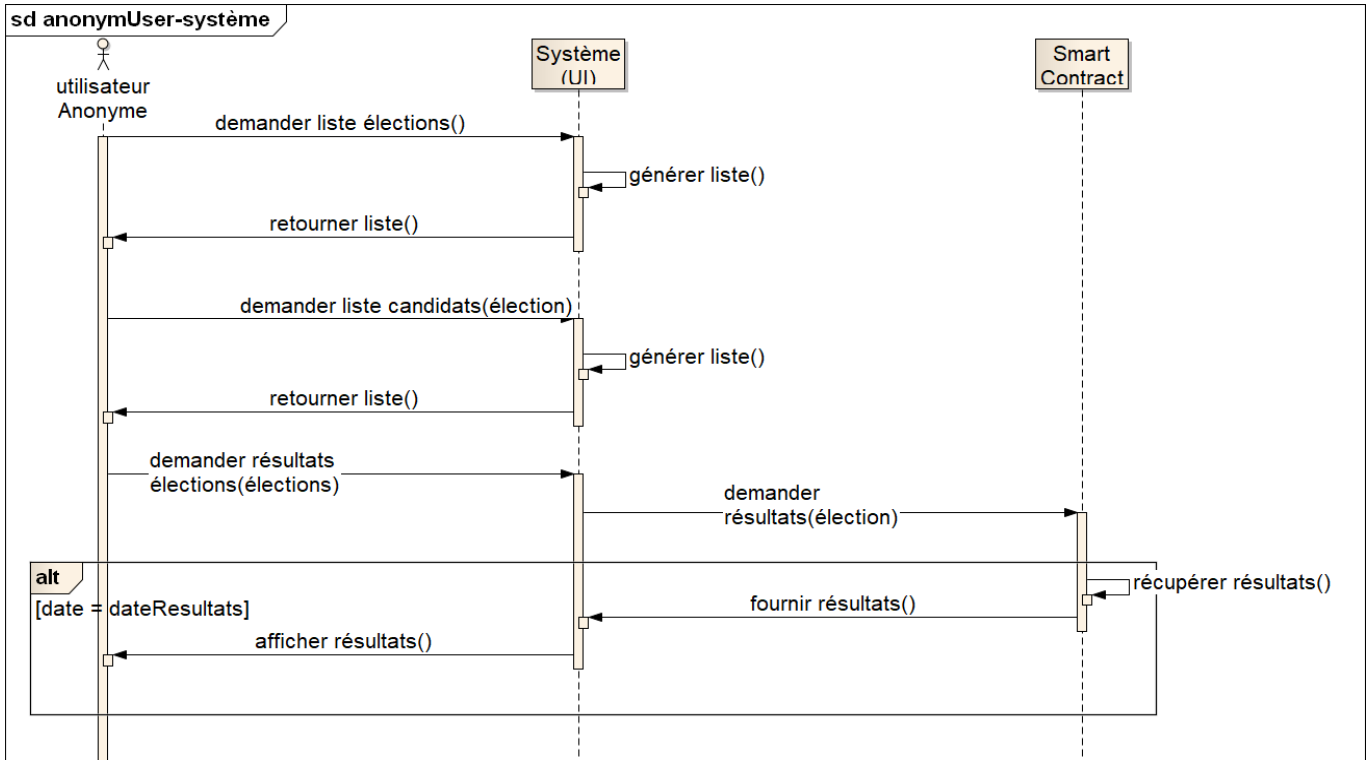


Figure III.7 – DSS Consulter résultats élection

Tout internaute a accès au site. Il n'a pas besoin de s'enregistrer ou de s'authentifier pour consulter la liste des candidats ou les résultat d'un vote donné.

Alors que la liste des candidats est récupérée directement sur la base de données du site et affichée en tout temps, les résultats sont récupérés à partir du Smart Contract et affichés seulement s'ils sont disponibles.

3 Conception : Domaine du projet

Cette phase propose une réalisation de l'analyse et des cas d'utilisation en prenant en compte toutes leurs exigences.
L'élaboration du modèle du domaine permet d'opérer une transition vers une véritable modélisation objet :

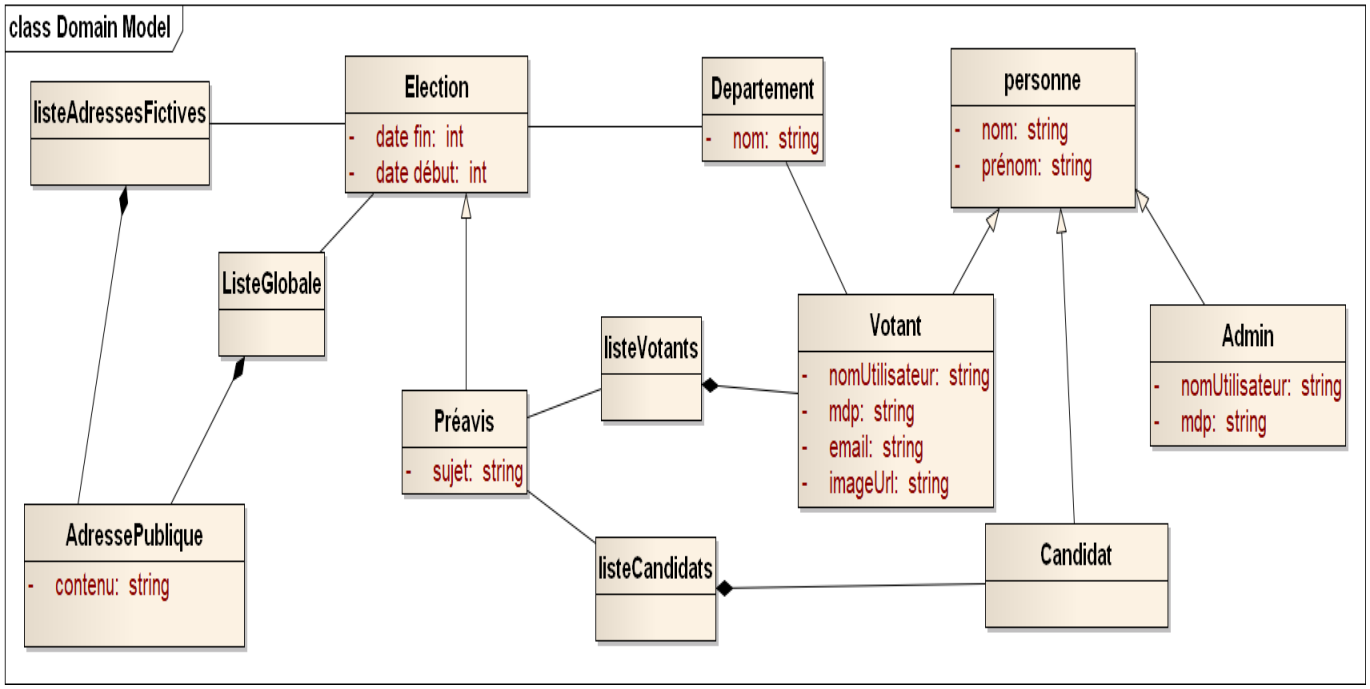


Figure III.8 – Modèle du domaine

Nous avons établi les entités suivantes :

Les utilisateurs du site :

— *« Votant »* : représente un compte votant enregistré sur le site et vérifié par l'administrateur (compte activé).

— *« Admin »* : représente le compte administrateur.

Autres :

— *« Candidat »* : représente un candidat identifié tout simplement par son nom et son prénom. Il ne possède pas de compte sur le site.

— *« Préavis »* : représente le préavis de vote, publié sur le site par l'administrateur afin d'inviter les votants à inscrire leurs adresses. Un préavis de vote inclut une liste de comptes votants éligibles et une liste de candidats et porte sur un sujet et un département précis.

— *« ListeAdressesFictives »* et *« ListeGlobale »* : représentent les adresses publiques saisies par les votants au moment de l'inscription. Ces listes seront passées au Smart Contract et le vote pourra alors commencer. (L'entité Election présente sur le diagramme sera matérialisée par le Smart Contract).
À partir de ce moment-là, toutes les données seront persistées dans l'espace de stockage du Smart Contract (listes adresses, liste candidats, nombre de voix...).

4 Architecture de l'application

L'architecture de notre application dé-centralisée (DApp) est modélisée par l'infographie suivante :

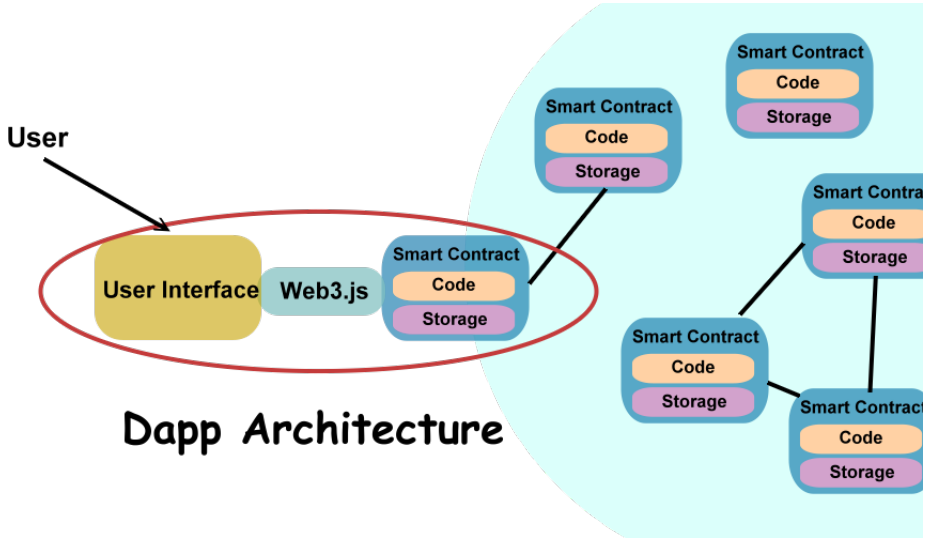


Figure III.9 – Architecture DApp [7]

On y voit clairement que toute DApp communique avec la BlockChain au travers de « **Web3.js** »qui est une collection de bibliothèques permettant d'intéragir avec un noeud Ethereum local ou distant, en utilisant une connexion HTTP ou IPC.

La figure suivante montre plus en détails le déroulement de cette communication entre la partie de la DApp hébérgée sur le serveur web (off-Chain) et la partie hébergée sur la BlochChain (On-Chain).

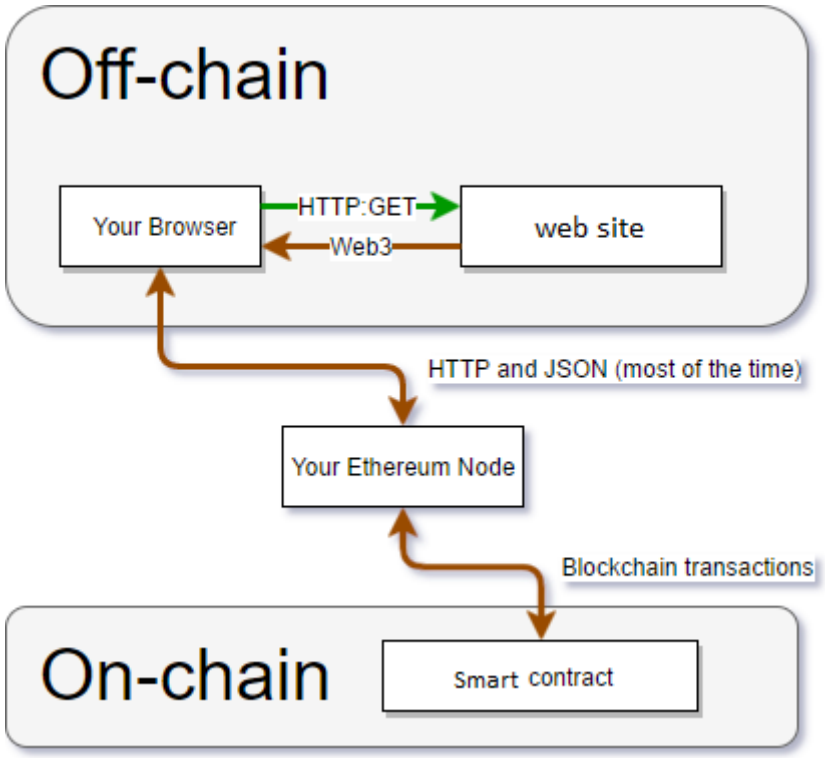


Figure III.10 – Architecture DApp (2) [8]

Conclusion

Ce chapitre nous a permis de présenter la démarche qu'on a suivie pour apporter une solution de vote électronique déployée sur la BlockChain, que ce soit par la description de la hiérarchie du site ou même par le diagramme des classes du domaine. Mais, il reste tout de même l'aspect concret que l'on n'a pas encore ébauché ; ce sera le sujet du chapitre suivant.

Chapitre IV

Réalisation

Plan

Introduction

Ce chapitre est consacré à la présentation de l'environnement logiciel nécessaire pour implémenter notre protocole de vote et aux étapes de l'implémentation.

1 Outils et langages utilisés

La réussite du projet est conditionnée par une multitude de choix essentiellement techniques :

1.1 Choix technologiques de développement

Nous avons choisi de développer une application de vote décentralisée (DApp) avec un front-end « Angular » et un back-end « Spring Boot », qui intègre un Smart Contract déployé sur la BlockChain « Ethereum ».
Dans la génération de notre application, nous avons utilisé l'outil « JHispter ».
Nous avons utilisé l'extension « Metamask » pour l'appel et l'exécution du Smart Contract de vote à partir du navigateur.

Angular : Angular est un framework web qui permet de réaliser des applications web en mode *Single Page Application*. C'est à dire une seule page qui ne se recharge jamais.
L'idée de base est d'étendre le langage HTML afin de :

- Pouvoir créer ses propres balises et attributs
- Permettre la représentation des données métiers, qui sont traitées et gérées avec le langage Typescript.

Spring Boot : [11] Spring Boot est un framework conçu pour simplifier le démarrage et le développement des applications Spring. Le framework propose une approche dogmatique de la configuration, qui permet d'éviter aux développeurs de redéfinir la même configuration à plusieurs endroits du code et de se soulager de certains imports fastidieux.

JHipster : JHipster (Java Hipster) est un générateur Yeoman utilisé pour créer un projet Spring Boot + Angular en quelques minutes. Il permet de générer une application Web moderne, responsive et performante, basée sur un socle technique innovant.

Ethereum : [12] Ethereum est considérée comme la blockchain la plus prometteuse en dehors de Bitcoin.
Ethereum vise à créer un nouveau web, décentralisé, plutôt qu'une nouvelle monnaie.
En pratique, les participants du réseau d'Ethereum ne se contentent pas de valider des transactions monétaires : ils exécutent du code provenant d'applications décentralisées, dites «Dapps». Ce code permet en particulier la mise en place de « smart contracts », qui constituent le cœur du potentiel d'Ethereum.
Dans l'infrastructure d'Ethereum, un Smart Contract n'est pas modifiable ou falsifiable puisque sa sécurité est garantie par un protocole cryptographique.

Metamask : [13] L'extension MetaMask est comme un pont qui permet d'exécuter des DApps directement dans son navigateur sans avoir à télécharger un nœud Ethereum complet. MetaMask inclut un coffre-fort d'identité sécurisé et une interface utilisateur pour gérer les identités sur différents sites et signer des transactions dans la blockchain.

MySQL : MySQL est la base de données open source la plus plébiscitée au monde, de par sa performance, sa fiabilité et sa facilité d'utilisation éprouvées. Elle est avant tout connue pour son utilisation par des sociétés Web, telles que Google, Facebook et Yahoo.

TruffeSuite et Ganache-CLI : TruffleSuite un framework qui inclut l'outil Ganache-CLI. Ensemble, ils permettent de simuler une blockchain Ethereum privée. Ils incluent également toutes les fonctions et caractéristiques RPC qui rendent le développement et le déploiement des applications Ethereum plus rapide, plus facile et plus sûr.

1.2 Langages de programmation :

Compte tenu des choix technologiques présentés ci-dessus, les langages de programmation utilisés étaient les suivants :

- **Langage de programmation du smart Contract :** Solidity
- **Langage de programmation du front-end :** TypeScript
- **Langage de programmation du back-end :** Java

1.3 Outils d'intégration continue [4]

Les outils suivants constituent des outils complémentaires utilisés dans le cadre de l'application du principe d'intégration continue :

Travis CI : Travis CI est un logiciel libre d'intégration continue. Il fournit un service en ligne utilisé pour compiler, tester et déployer le code source des logiciels développés.
Dans le cadre de notre application, nous avons utilisé Travis CI en association avec le service d'hébergement et de versionnement du code source **GitHub** et de l'outil d'analyse de la qualité du code **SonarQube**.

Git : Git permet de collaborer avec des développeurs sur d'autres ordinateurs, chacun pouvant extraire et envoyer les fichiers sources de et vers le dépôt central. Cette forme d'assistance s'intègre parfaitement dans le cadre de la méthodologie **agile** que nous avons adoptée et offre une sécurité et une facilité dans le développement itératif et incrémental.

SonarQube : SonarQube est un logiciel libre permettant de mesurer la qualité du code source en continu. Il permet essentiellement de générer un reporting sur :

- L'identification des duplications de code
- Le respect des règles de programmation
- La détection des bugs potentiels
- L'évaluation de la couverture de code par les tests unitaires

JUnit : JUnit est un framework de test unitaire pour le langage de programmation Java. Il définit deux types de fichiers de tests :

- Les *TestCase* (cas de test) : ce sont des classes contenant un certain nombre de méthodes de tests. Chaque TestCase sert généralement à tester le bon fonctionnement d'une classe.
- Une *TestSuite* : permet d'exécuter un certain nombre de TestCase déjà définis.

MeisterTask : [14] C'est une plateforme web qui permet de coordonner le travail au sein d'une équipe et de suivre le progrès de chaque membre et plus généralement, du projet en lui-même.
L'image que vous voyez ci-dessous est une capture d'écran du tableau de bord de notre projet où sont regroupées les différentes tâches à venir, en cours et terminées et où chaque tâche est assignée à un coéquipier. Ce fil d'activités est partagé entre les membres de l'équipe et permet ainsi une traçabilité plus facile.

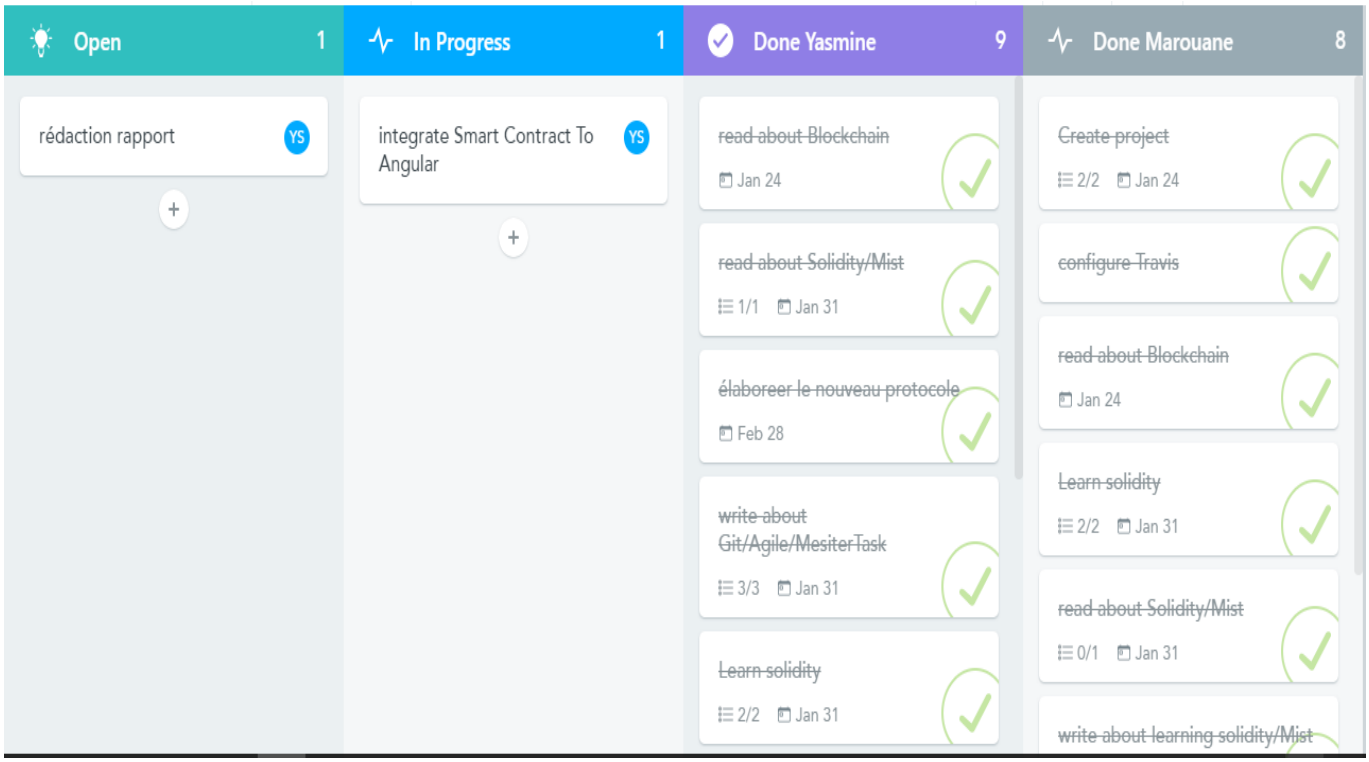


Figure IV.1 – Tableau de bord MeisterTask

Outre ses services en matière de planification et de suivi, et en plus de ses caractéristiques en termes de flexibilité et de simplicité, MeisterTask est un outil qui s'intègre facilement à d'autres application dont notamment **Git**, permettant d'automatiser le flux de travail.

La figure ci-dessous regroupe les différents outils cités plus haut et montre les interactions entre eux. C'est la manière avec laquelle nous avons implémenté une plate-forme d'intégration continue pour notre application :

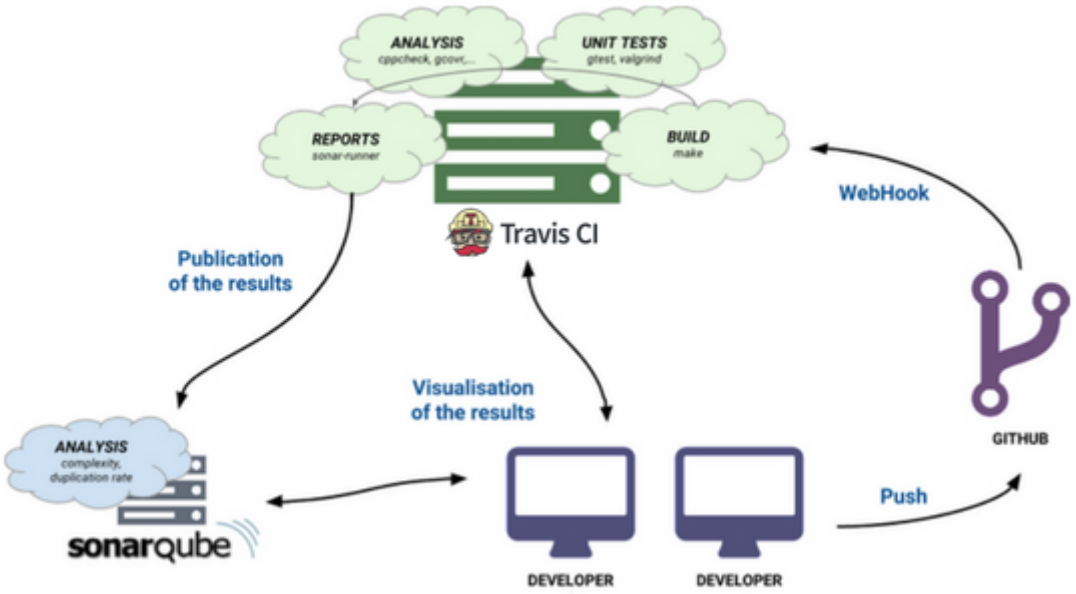


Figure IV.2 – Intégration Continue : outils et mise en oeuvre [4]

2 Présentation de l'application

Dans cette partie, nous vous présontons le Smart Contract de vote qu'on a développé ainsi que les étapes de l'implémentation de notre projet.

2.1 Smart Contract

Toute la logique métier du protocole de vote est encapsulée dans le Smart Contract que nous vous présentons ci-dessous :

```
pragma solidity ^0.4.16;

contract Ballot {

    struct Voter {
        bool voted;  // if true, that person already voted
        bool revoked;
        bytes32 vote;   // index of the voted proposal
```

```
    address publicAdress;
}

struct Proposal {
    bytes32 name;   // short name (up to 32 bytes)
    uint voteCount; // number of accumulated votes
}

address public chairperson;
Proposal[] public proposals;
Voter[] public voters;

bool voteStarted;
bool voteEnded;


function Ballot(bytes32[] proposalNames,address[] haveRightToVote) public
    {
    chairperson = msg.sender;
    for (uint i = 0; i < proposalNames.length; i++) {
        proposals.push(Proposal({
            name: proposalNames[i],
            voteCount: 0
        }));
    }

     for (uint j = 0; j < haveRightToVote.length; j++) {
        voters.push(Voter({
            voted: false,
            revoked: false,
            publicAdress: haveRightToVote[j],
            vote: 0
        }));
    }

voteStarted = false;
voteEnded = false;

}


function RevokeRightToVote(address[] haveRightToVote) public {
```

```
    require(
        (msg.sender == chairperson)
    );

    for (uint j = 0; j < haveRightToVote.length; j++) {
         for (uint k = 0; k < voters.length; k++)
         {
            if (voters[k].publicAdress == haveRightToVote[j])
            {
                        for (uint i = 0; i < proposals.length; i++) {
       if (voters[k].vote == proposals[i].name && !voters[k].revoked)
       {
           voters[k].revoked= true;
       proposals[i].voteCount -= 1;
       }
    }  }}}}



function vote(bytes32 tovote)
{
    require(
        (voteStarted && !voteEnded)
    );

    address voteradress = msg.sender;
    uint voterRank = 500;
     for (uint j = 0; j < voters.length; j++) {
        if (voters[j].publicAdress == voteradress)
        {
            voterRank = j;
        }}


    if (voters[voterRank].voted == false)
    {
        voters[voterRank].voted = true;
     voters[voterRank].vote = tovote;

     for (uint t = 0; t < proposals.length; t++) {
        if (proposals[t].name == tovote)
        {
           proposals[t].voteCount += 1;
```

```
            }}}}

    function winnerName() public view
            returns (bytes32 winnerName_)
    {
        require(
            (voteEnded)
        );

        bytes32 name ;
        uint mostCount = 0;
        for (uint t = 0; t < proposals.length; t++) {
            if (proposals[t].voteCount > mostCount)
            {
                mostCount = proposals[t].voteCount;
               name = proposals[0].name;
            }
        }
        winnerName_ = name;
    }


    function startVote()
    { require(
            (msg.sender == chairperson)
        );

        if (voteStarted == false)
        {
            voteStarted = true; }
        }

    function EndVote()
    {
        require(
            (msg.sender == chairperson)
        );

        if (voteStarted == true && voteEnded == false) {
            voteEnded = true; }
        }
}
```

Ce contrat de vote comporte différentes parties :

Les entités

a. Voter : Représente l'entité « votant » et contient les champs suivants :
- Voted : indique si le votant a soumis ou non son choix
- Revoked : indique si le vote a été ou non révoqué
- Vote : ce champ indique la référence du candidat choisi (son indice dans le tableau « propasals » : à voir plus tard)
- Publicaddress : indique l'adresse publique à partir de laquelle le vote a été soumis

b. Proposal : Représente l'entité « candidat » et contient les champs suivants :
- Name : nom du candidat
- VoteCount : nombre de voix accumulés

c. ChairPerson : représente l'adresse publique de l'administrateur (celui qui crée le Smart Contract)

d. proposals : tableau regroupant les entités Proposal référencés par leurs indices dans le tableau

e. voters : tableau regroupant les entités de type Voter

f. voteStarted, voteEnded : booléens indiquant le début et la clôture des élections

Les méthodes

a. Le constructeur : Prend en paramètres :
- Un tableau de strings (noms des candidats)
- Un tableau d'adresses : adresses éligibles au vote (adresses effectives et fictives)

Cette méthode :
- Définit l'adresse de l'administrateur.
- Initialise la liste des candidats (le tableau Proposals) et celle des votants (le tableau Voters).
- Met les variables « voteStarted » et « voteEnded » à « faux ».

b. RevokeVote() : Prend en paramètres une liste d'adresses : ce sont des adresses fictives. La méthode annule tous les votes effectués par des comptes votants rattachés à ces adresses publiques fictives.
La définition de la méthode exige que l'appelant ne soit autre que l'administrateur.

c. Vote() : Prend en paramètre un entier correspondant à l'indice du candidat visé dans le tableau Proposals.
Cette méthode est accessible à tout votant et lui permet tout simplement de soumettre un vote. Elle incrément de 1 le champ « voteCount » du candidat choisi et elle ne peut être appelée que lorsque le vote a déjà commencé et n'est pas encore terminé.

d. WinnerName() : Permet de récupérer tout simplement les résultats du vote : elle parcoure la liste des candidats et retourne celui avec le nombre de voix (« voteCount ») le plus élevé. Elle ne peut être appelée que lorsque le vote est terminé.

e. StartVote() et EndVote() : Ne sont acccessibles que par l'administrateur pour commencer ou clôturer les élections.

2.2 Etapes de l'implémentation

Développement du back-end et du front-end de l'application :

Le frontal l'arrière plan du site ont été générés initialement avec l'outils de scaffolding « JHipster ».

En nous appuyant sur ce socle de départ, nous avons pu implementer notre site web en en travaillant à chaque fois sur une fonctionnalité nouvelle et parallélisant les tâches de développement du back-end et front-end pour l'implémentation de la tâche en question.

À ce stade, nous avons pu développer les fonctionnalités de la partie Off-Chain du site (sécurisation du siteà l'aide de **Spring Security, JWT Token** et le protocole https, gestion de l'authentification, de enregistrement...) :

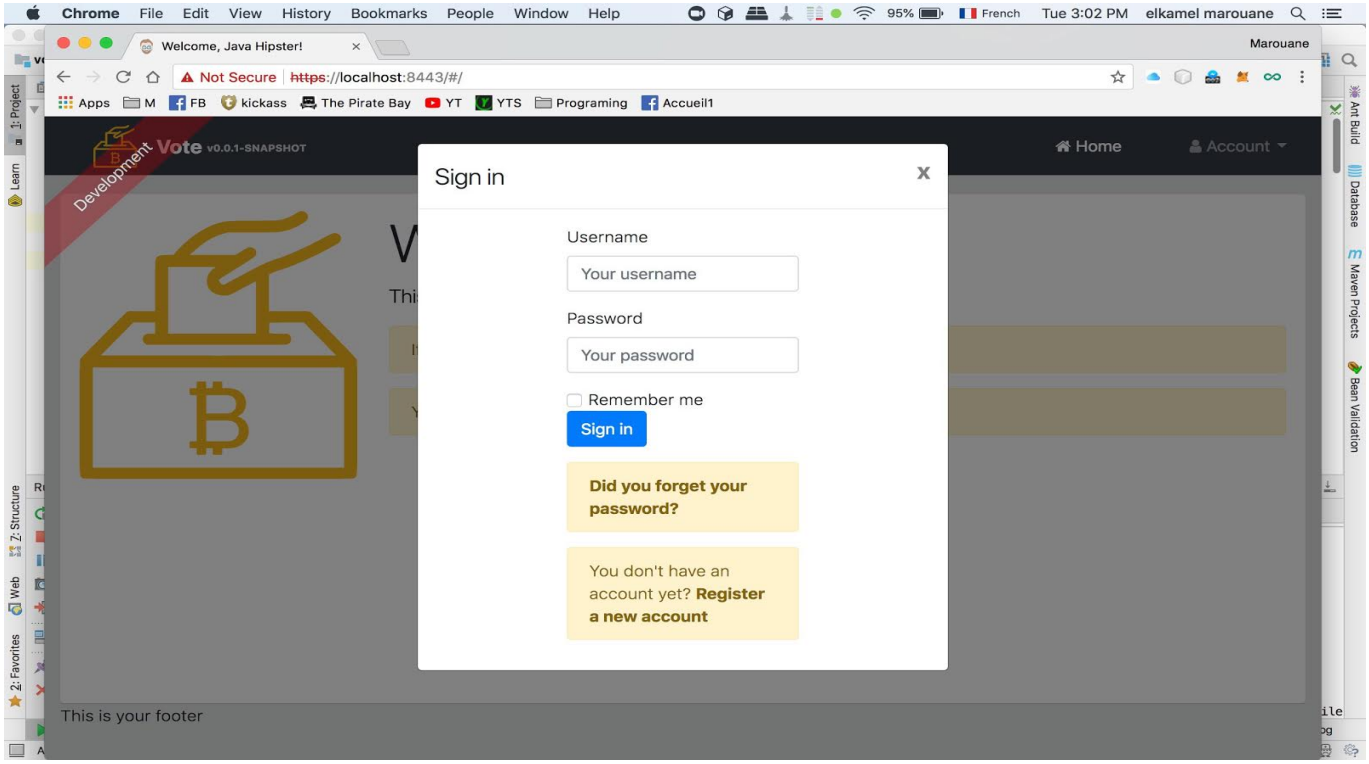


Figure IV.3 – gestion de l'authentification

Implémentation et test du Smart Contract de vote :

Nous nous sommes inspirés du contrat de vote développé par la communauté Solidity [15] que nous avons modifié pour l'adapter à notre protocole. Nous avons ensuite pu le tester directement sur **Remix** qui est un IDE Ethereum pour le langage Solidity. Il dispose d'un débogueur intégré et d'un environnement de test des Smart Contracts. Cela nous a permis de nous assurer du bon fonctionnement de notre contract de vote et de sa conformité au protocole et aux exigences du cahier des charges.

La figure suivante montre l'appel de la fonction « startVote() »de notre Smart Contract effectué à partir de l'adresse de l'administrateur (ChairPerson) et le hash de la transaction correspondante.

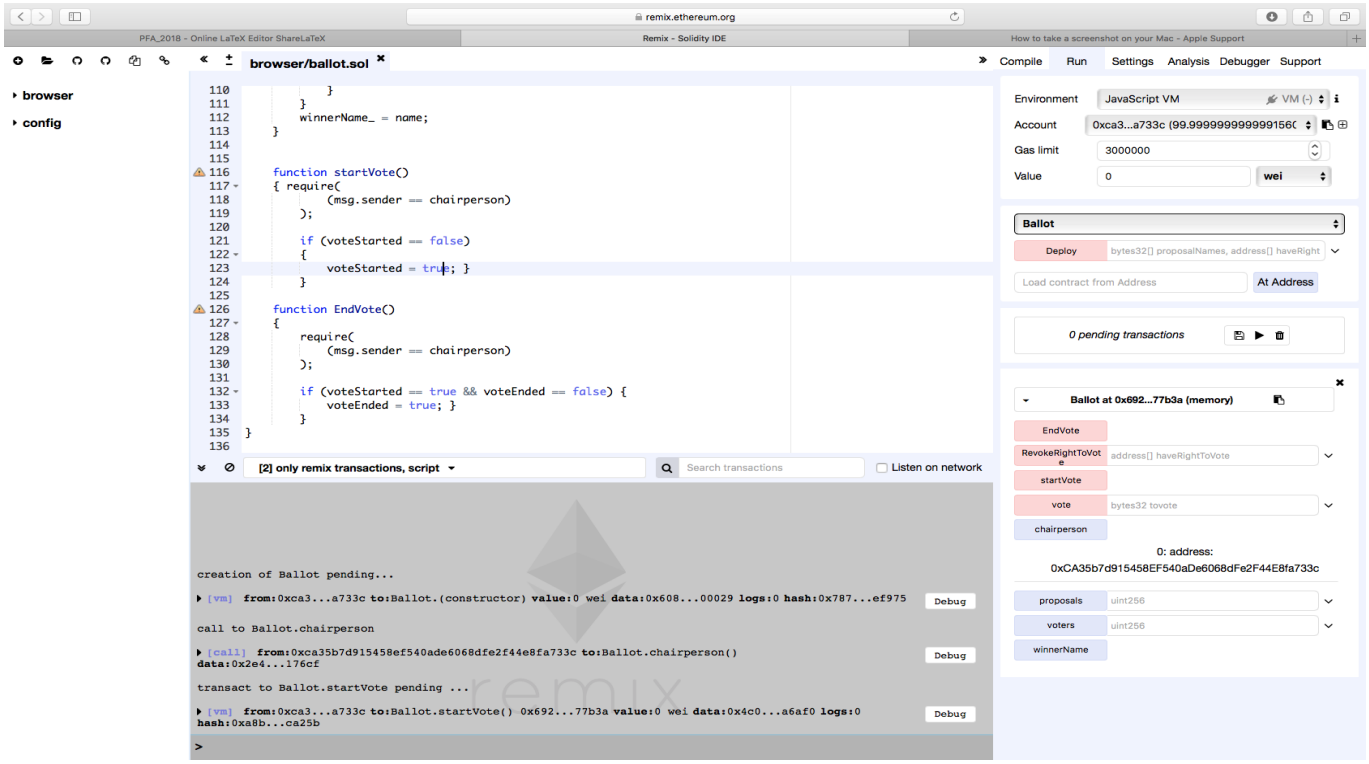


Figure IV.4 – déploiement du Smart contract sur Remix

Intégration du Smart Contract au front-end Angular :

L'étape d'intégration a été facilitée grâce aux outils **TruffeSuite** et **Ganache-CLI**. Après l'intégration du Smart Contract, nous avons continué à developer le front-end Angular afin de l'adapter aux fonctionnalités du Smart Contract qui y ont été ajoutée.

Cette dernière étape a permis l'assemblage de la partie On-chain et la partie Off-chain en une seule DApp fonctionnelle.

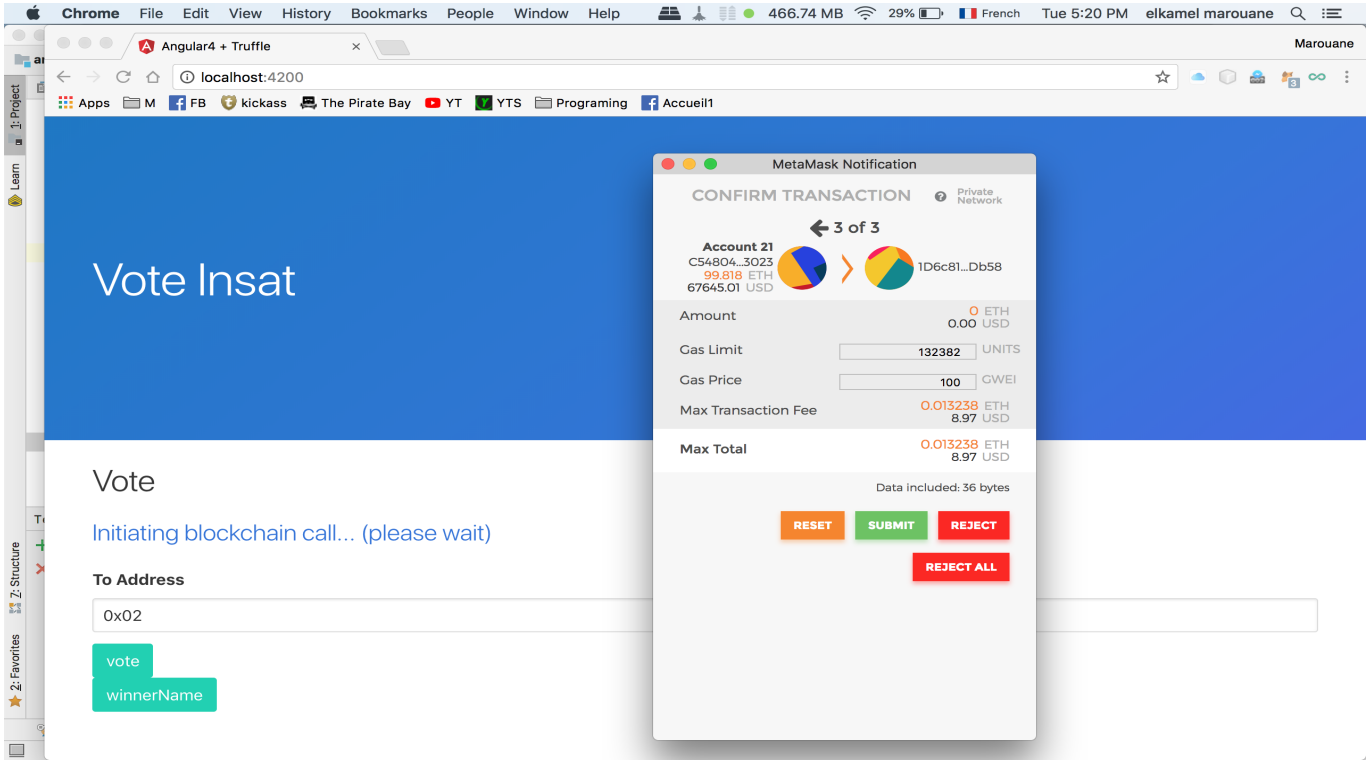


Figure IV.5 – opération de vote

Conclusion

Dans ce chapitre, nous avons présenté les différentes techniques utilisées pour implémenter le système. Nous avons également abordé les principes généraux d'intégration continue et d'ergonomie qui nous ont aidés à fournir des interfaces conviviales qui permettent aux utilisateurs de réaliser leurs tâches d'une manière efficace.

Conclusion Générale et Perspectives

Nous avons développé une application de vote décentralisée destinée à être utilisée dans le cadre de l'élection des chefs de départements à l'INSAT.

Une application décentralisée ou « DApp » est une application déployée sur la Blokhchain dont la logique métier est régie par un Smart Contract. La décentralisation garantit que la confiance des électeurs ne soit plus placée en une autorité unique puisque les transactions de vote sont surveillées et validées par tous les nœuds participants sur le réseau Ethereum. Nous sommes partis de cette idée et avons conçu notre application en utilisant :

- la méthode SCRUM qui est la méthodologie agile la plus répandue et la mieux adaptée aux petits projets travaillés en un temps limité.
- Le principe d'« Intégration Continue » durant toute la phase d'implémentation. C'est l'un des grands principes du génie logiciel. Il permet d'impliquer l'ensemble des collaborateurs, d'orchestrer, de contrôler et de consigner en continu le travail d'une équipe et de gagner ainsi en temps et en productivité.

La durée limitée du projet a considérablement influé sur la réalisation de celui-ci. Nous avons focalisé sur les fonctionnalités essentielles mais d'autres pistes peuvent être explorées dans le futur :

- Les candidatures doivent pouvoir être soumises en ligne.
- Tout utilisateur disposant d'un compte sur notre plate-forme doit pouvoir créer un vote sur un sujet de son choix (généralisation de l'application).
- La résistance à la coercition [1] et la vérifiabilité universelle [2] sont des critères qui doivent être pris en compte dans une version améliorée du protocole.

1. Coercition : Fait de contraindre. Un votant ne doit pas pouvoir prouver qu'un vote est le sien.
2. Vérifiabilité universelle : Chaque votant peut vérifier que que le résultat final de l'élection correspond bien à la somme de tous les votes.

Bibliographique

[1] Chantal Enguehard. Vote par internet : failles techniques et recul démocratique. https://hal.archives-ouvertes.fr/hal-00181335/document, (2007). [En ligne ; consulté le 09-avril-2018]. ii, 2

[2] Wikipédia. Intégration continue. https://fr.wikipedia.org/wiki/Intégration_continue, (2018). [En ligne ; consulté le 05-mai-2018]. ii, 13

[3] Méthodes agiles (RAD, XP). https://www.commentcamarche.com/contents/477-methodes-agiles-rad-xp, (2017). [En ligne ; consulté le 05-mai-2018]. ii, 13, 14

[4] Aurelien Maigret. Continuous Integration tools : How we built our CI Platform. http://skies.io/2016/01/05/Continuous-Integration-tools/, (2016). [En ligne ; consulté le 11-mai 2018]. iii, iv, 23, 25, 27

[5] Vincent Camiré. Blockchain : 4 transformations pour un monde prospère. http://vincentcamire.com/blockchain-4-transformations-monde-prospere/, (2017). [En ligne ; consulté le 22-avril-2018]. iv, 7

[6] Bhupendra Joshi. Continuous integration for Salesforce lightning development. https://medium.com/salesforce-developers/continuous-integration-for-salesforce-lightning-development-a59adf1a21b
(2017). [En ligne ; consulté le 05-mai-2018]. iv, 14

[7] Karan Ahuja. Programming an Ethereum based dApp. https://medium.com/coinmonks/programming-an-ethereum-based-dapp-part-1-ebdff09ec0f6, (2018). [En ligne ; consulté le 12-avril-2018]. iv, 21

[8] Olaf Tomalka. Case Study : Ethereum Dapp For Crowdfunded Loans. https://medium.com/tomalka-me/ethereum-dapp-for-crowdfunded-loans-case-study-7903155c780d, (2018). [En ligne ; consulté le 15-avril-2018]. iv, 22

[9] Mahesh Murthy. Full Stack Hello World Voting Ethereum Dapp Tutorial. https://medium.com/@mvmurthy/full-stack-hello-world-voting-ethereum-dapp-tutorial-part-1-40d2d0d807c
(2018). [En ligne ; consulté le 07-avril 2018]. iv, 39

[10] ConsenSys. A 101 Noob Intro to Programming Smart Contracts on Ethereum. https://medium.com/@ConsenSys/a-101-noob-intro-to-programming-smart-contracts-on-ethereum-695d15c1dab
(2015). [En ligne ; consulté le 10-avril 2018]. iv, 40

[11] BDan Woods. A la découverte des Micro-frameworks : Spring Boot. https://www.infoq.com/fr/articles/microframeworks1-spring-boot, (2014). [En ligne ; consulté le 13-mai 2018]. 24

[12] Blockchain France. Qu'est-ce qu'Ethereum ? https://blockchainfrance.net/2016/03/04/comprendre-ethereum/, (2018). [En ligne ; consulté le 17-mars 2018]. 24

[13] Metamask.io. About Metamask. https://metamask.io, (2018). [En ligne ; consulté le 13-mai 2018]. 24

[14] Meistertask.com. About MeisterTask. https://www.meistertask.com/fr, (2018). [En ligne ; consulté le 11-mai 2018]. 26

[15] Solidity. Solidity By Example : Voting. http://solidity.readthedocs.io/en/v0.4.21/solidity-by-example.html, (2017). [En ligne ; consulté le 28-mars 2018]. 33

Annexe

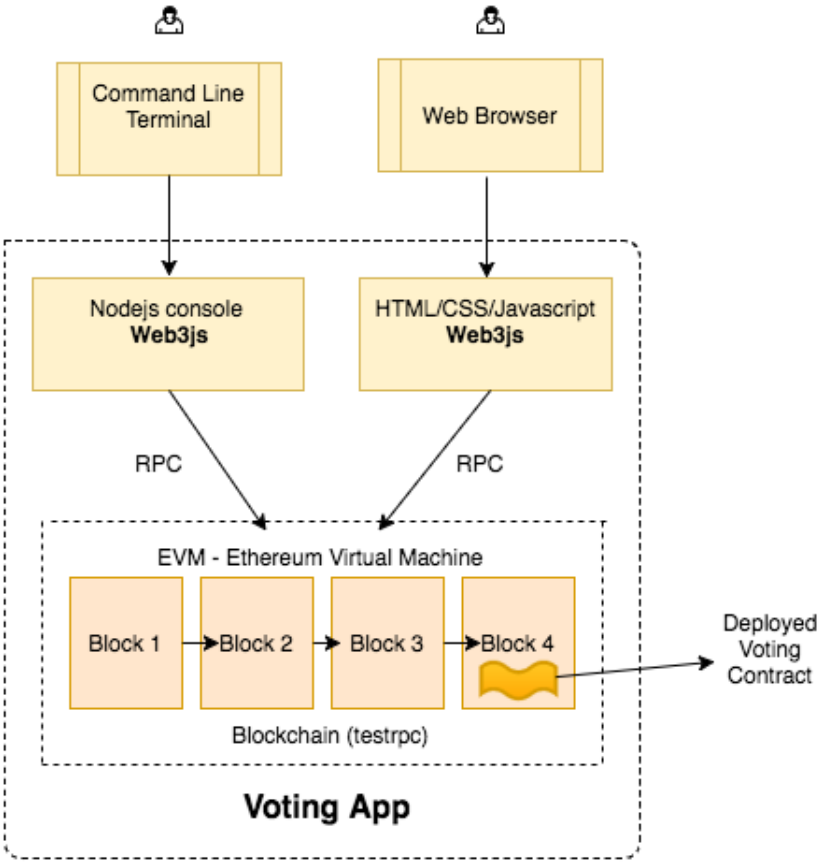
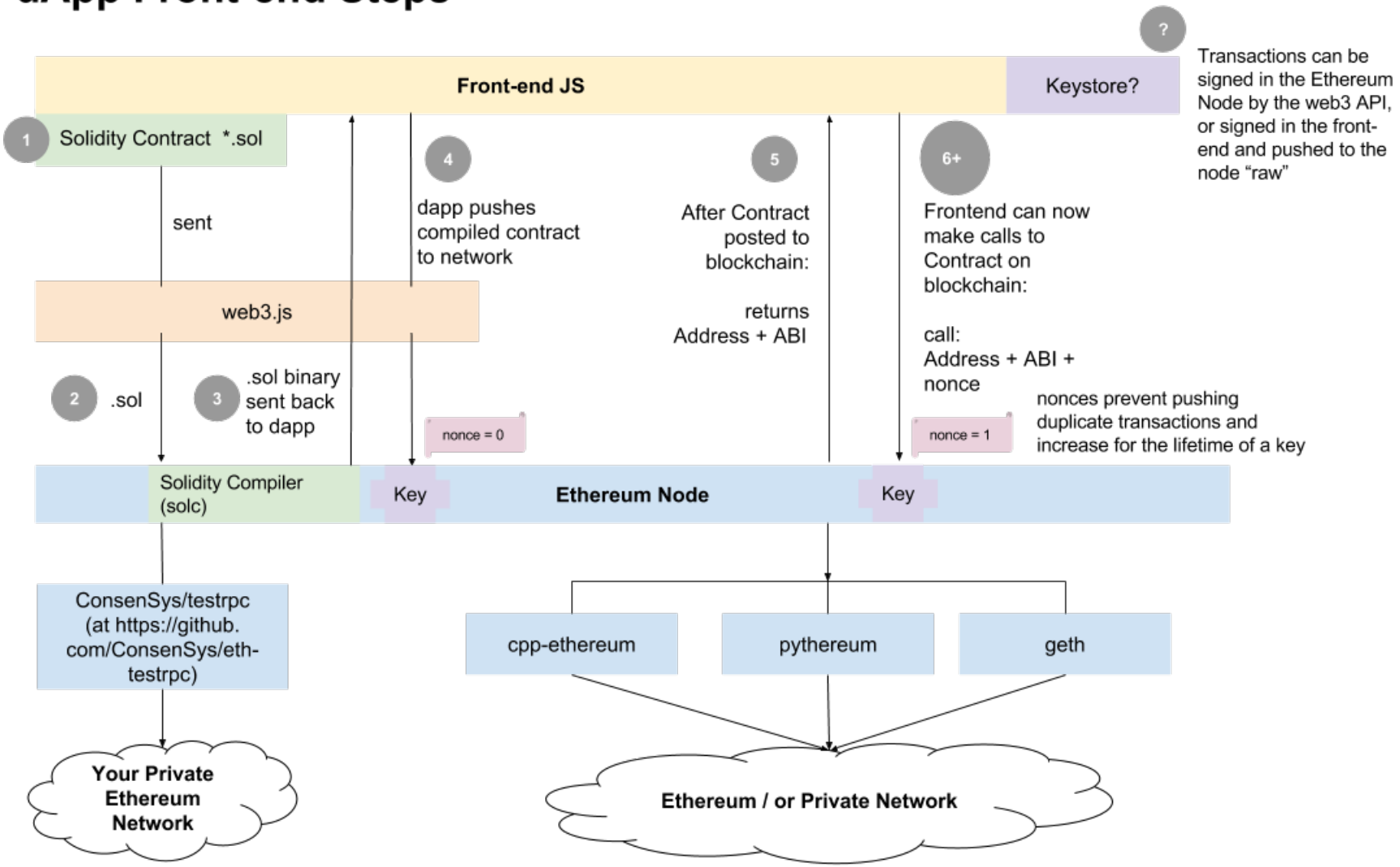


Figure A.1 – DApp front end steps simplified [9]



A **Contract Creation Transaction** is shown in steps 1-5 at above.

An **Ether Transfer** or **Function Call Transaction** is assumed in step 6.

Figure A.2 – DApp front end steps [10]