Université de Caen Normandie

Projet Annuel
Smart Contracts Ethereum

Auteurs :
Yacine KERIMI
Mohamed Baha DBEYBIA

Jury :
Tanguy GERNOT
Marc SPANIOL

31 mars 2021

Résumé

Ce projet vise à évaluer Ethereum, une plateforme décentralisée pour le déploiement de Smart Contracts sur la Blockchain. Notre approche est de déployer un réseau de test Ethereum privé, sur lequel nous développerons et déploierons une application Smart Contract décentralisée. Les étapes vers la construction de l'environnement de développement ainsi que l'application développée sont démontrées dans ce rapport. Nos expériences rassemblées ainsi que l'estimation du potentiel du projet Ethereum sont présentées dans notre évaluation. Ce rapport décrit la technologie blockchain et certaines applications de l'Ethereum. Nous examinons ensuite les défis à venir et les opportunités dans cette technologie fondamentale qui est prête à révolutionner notre monde numérique.

Table des matières

Chapitre 1

Introduction

Depuis sa création en 2008, le Bitcoin a réussi à gagner beaucoup de valeur de et de popularité et il est considéré maintenant comme un système monétaire et économique, ainsi qu'un mouvement de sous-culture. La technologie principale du Bitcoin est la blockchain, un modelé de base de données distribuée utilisé pour valider les transactions de manière sécurisée et décentralisée. C'est pour cette raison qu'une approche basée sur la Blockchain peut être utilisée pour d'autres applications, en particulier dans des contextes où l'anonymat la confidentialité sont des aspects importants.

Un nouvel outil appelé Ethereum a été décrit pour la première fois en 2013 et a été officiellement lancé le 30 juillet 2015. L'Ethereum est une plateforme conçue pour exécuter des contrats intelligents sur un réseau décentralisé.

Un contrat intelligent dans la conception d'Ethereum est décrit comme une application qui fonctionne exactement comme programmé sans un temps d'arrêt, sans censure sans fraude ou interférence d'un tiers en fournissant une plateforme sur laquelle tout le monde peut démarrer un service internet décentralisé et sécuriser. L'hypothèse posée est qu'Ethereum facilitera le lancement des applications basées sur la blockchain sans avoir besoin de démarrer un nouveau protocole Blockchain ou cryptomonnaie.

1.1 Sujet

Le sujet interprété dans ce projet est le développement d'une application décentralisé sur la Blockchain Ethereum, à l'aide d'un langage de programmation nommé « Solidity » qui est Turing-Complet, il est dédié à l'écriture des contrats intelligents. Cette application sera testée sur une blockchain privée et locale qui sera sans cout.

Le but de ce projet est de tester la validité de cette hypothèse en construisant une application décentralisée avec Ethereum. De plus, acquérir des connaissances sur le fonctionnement interne d'Ethereum. Cette expérience nous permet d'évaluer empiriquement Ethereum en termes de facilité d'utilisation, temps de développement, performances et sécurité, valeur ajoutée et avenir potentiel.

1.2 Problématique

La principale question à laquelle nous tenterons de répondre dans ce projet est la suivante : Ethereum peut-il être directement utilisé pour déployer rapidement applications fiables et suffisamment performances avec une valeur ajoutée par rapport aux approches traditionnelles ?

Pour rendre la question ci-dessus plus claire, on a posé les questions suivantes.

1. Ethereum, peut-il être utilisé pour déployer des applications non traditionnelles ?
2. Quelle est la valeur ajoutée d'utilisation d'Ethereum par rapport à une approche de développement traditionnelle ?
3. Quel est le temps nécessaire pour déployer une telle solution ?

Chapitre 2

La Blockchain : Cest quoi ?

Nous trouvons utile à ce stade de donner également des définitions de quelques informations générales sur certains concepts clés dont leur compréhension est fondamentale dans les chapitres suivants.

2.1 Blockchain

La blockchain est un grand livre de transactions publiquement vérifiable et distribué qui sécurise le réseau Bitcoin contre la double dépense, la falsification et les transactions annulées.

Abstraitement, la blockchain est une structure de données qui se compose de blocs liés qui sont ordonnés dans le temps et qui contiennent un certain nombre de transactions. Les blocs sont liés d'une manière ou chaque bloc inclut l'identifiant du bloc précédent dans la chaine. L'ID de chaque bloc lui-même est le hachage cryptographique de son contenu.

2.1.1 Blocs

Ainsi ça devient évident que chaque bloc dépend du précédent pour former une chaine. Pour créer cette dernière, tous les blocs menants au bloc actuel devront être recréés avec.

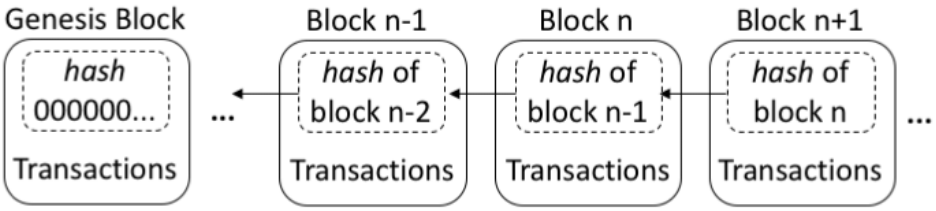


FIGURE 2.1 – Schéma de fonctionnement des blocs.

Les blocs contiennent une référence au bloc précédent, un ensemble de nouvelles transactions plus une valeur aléatoire (nonce). Le contenu du bloc est haché et si la sortie de la fonction de hachage est bien valide alors le bloc est considéré comme miné et diffusé sur le réseau. Ainsi, le mining consiste essentiellement à découvrir le bon nonce qui avec le contenu du bloc produira le hash souhaité. En raison des propriétés des fonctions de hachage utilisées, c'est pratiquement impossible d'essayer de tromper le réseau en précalculant un ensemble de nonces. Le fait que la solution à ce problème cryptographique a été découverte par les nœuds est considéré comme une preuve et pour cette raison, le bloc est accepté par le réseau. Dans l'implémentation actuelle du protocole Bitcoin, les mineurs sont récompensés par 25 BTC pour l'effort qu'ils investissent dans le mining d'un bloc.

2.1.2 Mining et miners

Une fois que les transactions ont atteint les nœuds et ont été vérifiées, il faut une certaine opération qui vise à découvrir un regroupement approprié de ces transactions valides. Cette opération est appelée mining et les nœuds qui participent sont les mineurs. Chacun de ces groupes est appelé un bloc et pointe vers le bloc précédent de transactions groupées, formant ainsi une chaine (d'où le nom Blockchain). Une fois qu'un bloc est

inséré dans la blockchain, les transactions qu'il contient sont considérées comme acceptées par tous les nœuds dans le réseau.

FIGURE 2.2 – Schéma représentatif de vérification du hash lors du mining

2.2 Bitcoin

Le Bitcoin est décrit comme un réseau de consensus qui constitue un système de paiement en ligne ainsi qu'une monnaie numérique. Il est considéré comme le premier réseau de paiements entièrement décentralisé qui est capable de fonctionner sur une base peer-to-peer pure en dehors du contrôle de toute autorité centrale.

Les transactions dans le réseau Bitcoin sont maintenues et vérifiées dans le registre de transactions distribué qui est la blockchain. Le système est protégé contre les doubles dépenses et les fausses transactions.

Chaque transaction est également signée avec la clé privée de l'initiateur de la transaction et adressée à la clé publique du destinataire. En raison de l'utilisation de ces algorithmes cryptographiques, le terme cryptomonnaie est apparu.

2.3 Smart Contracts

Le terme Smart Contract a été introduit par Nick Szabo et fait référence à un ensemble de protocoles informatiques et interfaces destinées à formaliser et sécuriser les relations et les accords sur les réseaux informatiques.

Les Contrats intelligents capturent et traduisent souvent les clauses du contrat traditionnel en un ensemble de règles de logique informatique qui sont exécutées comme n'importe quel programme informatique. Ces contrats sont souvent implémentés pour garantir les clauses vis-à-vis des parties participantes.

Chapitre 3

L'Ethereum

Ethereum est un projet open source introduit pour la première fois en 2013, initialement décrit comme une « plateforme de nouvelle génération décentralisée des Smart Contracts », à première vue Ethereum est un réseau peer-to-peer et une échangeable cryptomonnaie qui permettent aux nœuds participants de partager des ressources informatiques pour l'exécution de contrats intelligents programmables sur la blockchain. Il existe cependant plusieurs façons différentes de décrire Ethereum selon les contextes et les implémentations. En d'autres termes, avec Ethereum, tout utilisateur peut avoir accès à une plateforme mondiale bon marché, sans infrastructure, offrant un ensemble intéressant de fonctionnalités :

- Authentification de l'utilisateur, vérifiée par l'utilisation de signatures cryptographiques.
- Logique de paiement facilement déployable. Un système de paiement peut être déployé sur Ethereum très rapidement sans recours à des tiers.
- Aucune infrastructure de serveur. Ethereum est entièrement construit sur un réseau peer-to-peer sans infrastructure de serveurs centraux . Ainsi, le déploiement d'une application sur la blockchain ne nécessite pas la configuration et les couts d'hébergement et de maintenance des serveurs.

3.1 Histoire Ethereum

Ethereum a été décrit pour la première fois par Vitalik Buterin au début de 2014 dans son article « Ethereum : une cryptomonnaie de nouvelle génération et une plateforme d'application décentralisée ». Sa définition formelle a été donnée plus tard cette année-là dans le « Yellow Paper » de GavinWood. Le développement de la première mise en œuvre a commencé peu de temps après.

Le 22 juillet 2014, la vente publique officielle d'Ethereum (ICO, Initial Coin Offering) a été lancée. Son objectif était de libérer le premier lot d'Ether, la cryptomonnaie interne qui sert de jetons de paiements des couts de transaction. La vente participative s'est achevée le 2 septembre 2014 et a duré 42 jours. Pendant les 14 premiers jours, on pouvait acheter 2.000 Ether pour 1 BTC, un montant qui a été réduit à 1.337 Ether pour 1 BTC par la suite. À la fin de la vente, 60.102.216 Ether ont été vendus à 9.007 acheteurs pour le cout de 31.529 BTC, qui à ce moment-là avaient une valeur de 18.439.000$.

Le 30 juillet 2015, la première version officielle ainsi que le premier réseau en direct, appelé Ethereum Frontier, ont été lancés. Cela peut être considéré comme la phase de test bêta d'Ethereum. Sur ce réseau, tous les Ether achetés peuvent être échangés. Les futures versions du réseau Ethereum sont appelées, dans l'ordre, Homestead, Metropolis et Serenity.

3.1.1 Projets d'Ethereum

Les statistiques concernant la blockchain Ethereum montrent que le projet a attiré beaucoup d'attention. Il existe déjà un certain nombre d'applications et de concepts décentralisés construits sur Ethereum, dont certains présentent un grand potentiel. Nous trouvons utile de présenter trois de ces applications qui, selon nous, sont très prometteuses et démontrent la puissance d'Ethereum :

- Slock.it : Un verrou physique décentralisé. Il s'agit d'un miniordinateur Ethereum ainsi que d'une serrure électronique pouvant écouter la blockchain. Le concept est que l'on peut verrouiller n'importe quel actif (par exemple, un appartement, une voiture, un vélo) derrière le « Slock » et que n'importe qui dans le monde peut louer cet actif moyennant des frais en Ether. Si la transaction est vérifiée, le locataire peut déverrouiller l'actif physique avec sa clé privée et peut l'utiliser pendant la durée du bail. Ce projet montre très bien comment Ethereum peut se connecter au monde physique.
- BlockApps Strato : Une solution technologique Full-Stack qui permet le développement et le déploiement facile d'applications d'entreprise sur la Blockchain. Il fournit un ensemble d'outils tels que des

registres de blockchain privés, des interfaces graphiques de développement de contrats intelligents ainsi que des réseaux P2P semi-privés, c.-à-d. des réseaux qui peuvent être isolés et en même temps pouvoir communiquer avec le réseau Ethereum.
— Colony : une plateforme décentralisée qui permet la création d'entreprises décentralisées à travers le monde. Il aspire à rassembler des personnes possédant des compétences différentes et souhaitant travailler ensemble vers un objectif commun. Une récompense appelée « nectar » est distribuée périodiquement et les gens se disputent en apportant des idées, en prenant des décisions ou en effectuant une sorte de travail. Il existe également un système de réputation communautaire qui reflète le mérite de chaque individu ainsi que son impact dans la communauté.

3.1.2 Implementation

Ethereum a trois différentes implémentations officiellement maintenues écrites en C ++, Go et Python, qui sont toutes parfaitement interopérables sur le réseau Ethereum. Plus précisément :
— Eth : l'implémentation C ++, il est censé fonctionner plus rapidement que les autres implémentations et c'est aussi la base de la boite à outils de développement de contrat de l'IDE Mix. Il est livré avec un certain nombre d'outils d'analyse de réseau comme Alethzero et il est suggéré comme plateforme de développement pour les projets Internet of Things. C'est également le seul client qui prend en charge l'extraction de GPU.
— Geth : l'implémentation Golang ; Geth sera la base du futur navigateur Web Mist. Il est suggéré la plateforme de développement appropriée pour les applications Web.
— Pyethapp : L'implémentions Python, destiné principalement à des fins éducatives. Les utilisateurs souhaitant comprendre le fonctionnement interne d'Ethereum et souhaitant contribuer à son expansion sont invités à utiliser Pyethapp en raison de la lisibilité et de la clarté de son code. Il n'est cependant pas destiné à un usage haut de gamme, car il manque de performances.

3.2 Concepts d Ethereum

Ci-dessous, les principaux concepts d'Ethereum sont expliqués dans une perspective théorique et technique.

3.2.1 Comptes

Dans Ethereum, toute entité qui détient un état interne est associée à un compte, c'est-à-dire une paire de clés privées/public. La clé publique est également considérée comme l'adresse du compte.
Ethereum distingue deux types de comptes, les comptes détenus en externe et les contrats. Un compte externe est un compte personnel contrôlé par une clé privée. Le propriétaire de la clé privée peut envoyer de l'éther ou des messages à d'autres comptes externes. Un contrat est essentiellement un compte qui détient sa propre logique mappée dans le code qui le contrôle.

3.2.2 Contrats et transactions

Comme mentionné dans la section précédente, un contrat est un compte qui contient du code, et c'est essentiellement la manière d'Ethereum d'insérer une logique programmable dans la blockchain. Les contrats, en général, sont destinés à servir les objectifs suivants :
— Stockage de l'état des valeurs significatives pour d'autres contrats ou entités externes. Par exemple, un contrat de cryptomonnaie peut contenir les soldes de compte de toute personne interagissant avec ce contrat.
— Servir de compte externe avec des politiques d'accès spéciales. Il peut s'agir par exemple d'un service de messagerie qui ne transfère les messages que si certaines conditions sont remplies.
— Cartographie et gestion des relations entre un certain nombre d'utilisateurs. Par exemple, on pourrait cartographier la logique d'un contrat financier réel qui sera toujours exécutoire dans l'environnement Ethereum.
— Agissant comme des bibliothèques de logiciels en fournissant des fonctions à d'autres contrats. Les contrats peuvent interagir en passant des messages qui peuvent contenir des quantités d'Ether, des tableaux d'octets ou des adresses de compte. Lorsqu'un contrat reçoit un tel message, il peut renvoyer des données qui sont consommables par l'expéditeur du message, servant ainsi essentiellement d'appel de fonction traditionnel.

L'environnement d'exécution d'Ethereum est inerte jusqu'à ce que quelque chose le mette en mouvement. Les transactions sont le mécanisme par lequel les actions sont déclenchées. Tout utilisateur peut envoyer une transaction de son compte vers un autre compte externe ou vers un contrat. Dans le premier cas, ils peuvent transférer de l'Ether de leur solde au solde d'un autre utilisateur, mais à part cela, ces transactions n'ont pas beaucoup

plus d'intérêt. Dans le second cas, lorsque le destinataire est un contrat, le contrat se réveille et exécute son code.

Un contrat peut lire ou modifier son état interne, consommer le message reçu ou à son tour déclencher l'exécution d'un contrat différent en lui envoyant un message. Lorsque l'exécution d'une action et de toutes ses actions ultérieures s'arrête, l'environnement retourne à un état d'arrêt jusqu'à ce que la transaction suivante soit reçue.

3.2.3 Gas

Le réseau doit être récompensé pour la quantité de calcul qu'il fournit afin d'exécuter les contrats intelligents. Le montant d'Ether payé pour chaque transaction est reflété dans le concept de gas.

Chaque activité de calcul (par exemple, un cycle CPU) coute une certaine quantité d'unités de gas. La complexité totale des calculs au sein d'une transaction définit sa dépense finale en gaz. Chaque transaction spécifie un prix du gaz, c'est-à-dire le prix que l'expéditeur est prêt à payer pour une unité de gaz. Le prix du gaz incite les mineurs à inclure l transaction dans un bloc. Un mineur est libre d'ignorer toute transaction avec un prix gas bas.

Chaque transaction spécifie une limite de gaz, c'est-à-dire la quantité maximale d'unités de gaz que l'expéditeur est prêt à dépenser pour cette transaction. Comme il est impossible de prédire la quantité exacte de gaz que chaque transaction dépensera, l'expéditeur doit fixer une limite supérieure de gaz qu'il est prêt à payer pour la transaction afin de se protéger du manque de fonds (si, par exemple, ils invoquent une boucle infinie). Le concept de limite de gaz s'applique également aux blocs et reflète la somme du gaz dépensé par toutes les transactions d'un bloc. Son but est d'éviter que chaque bloc ne devienne trop volumineux, ce qui aurait un impact sur la création et la propagation des blocs dans le réseau, et par extension sur les performances du système.

Si l'expéditeur d'une transaction n'a pas assez d'Ether dans son compte pour couvrir la transaction, alors l'exécution s'interrompt et tous les changements d'état intermédiaires sont rétablis à leurs valeurs avant l'envoi de la transaction. Les dépenses en gaz jusqu'à ce point sont soustraites du compte de l'expéditeur pour couvrir les couts de calcul survenus avant l'arrêt de l'exécution.

Les mineurs reçoivent une récompense pour l'effort de calcul qu'ils investissent sur le réseau. Le montant qu'ils reçoivent comprend une récompense statique de 5 Ether pour le bloc découvert et tout le gaz dépensé dans le bloc, c'est-à-dire le gaz consommé par toutes les transactions de ce bloc. La récompense statique est émise comme une transaction des mineurs eux-mêmes tandis que les couts de transaction sont payés par chaque expéditeur de transaction. Il est prévu qu'à mesure que le réseau se développe, le gaz de transaction dans chaque bloc dépassera la valeur des récompenses statiques et sera la principale incitation au mining.

3.2.4 Mining

Ethereum, comme la plupart des technologies Blockchain, utilise un processus d'extraction pour sécuriser le réseau. Le processus implique la production de blocs dont la validité sera vérifiée par le réseau. Tout comme le protocole Bitcoin, un bloc n'est valide que s'il contient une preuve de travail d'une certaine difficulté.

Cet algorithme (Proof of Work) utilisé par Ethereum s'appelle Ethash et il consiste à trouver une entrée nonce (un nombre arbitraire destiné à être utilisé une seule fois) à l'algorithme afin que le résultat soit en dessous d'un certain seuil en fonction de la difficulté. La difficulté de mining est automatiquement ajustée pour qu'un bloc soit produit toutes les 12 secondes.

Ethash est conçu pour être dur en mémoire afin d'être impossible à implémenter sur les machines ASIC (Application-Specific Integrated Circuit). En effet, pour que le ce Proof of Work soit calculé, un sous-ensemble d'une ressource fixe dépend de l'entête de bloc et du nonce. Cette ressource, un Directed Acyclic Graph (DAG), a une taille de 1 Go et change tous les 30 000 blocs. Avec un temps de bloc de 12 secondes, 30 000 blocs représentent une période de 100 heures, appelée « Epoch ». Ce DAG dépend uniquement de la hauteur de la blockchain et peut donc être prégénéré. Sa génération prend quelques minutes et aucun bloc ne peut être produit avant la génération du DAG. Cependant, le graphe complet n'est pas nécessaire pour la vérification des blocs, car les sous-ensembles appropriés peuvent être calculés à partir d'un cache de 16 Mo, ce qui nécessite une faible puissance du processeur et de la mémoire.

3.3 Solidity

Solidity est l'un des langages de programmation de haut niveau utilisés pour écrire des contrats intelligents. Solidity est la langue officiellement maintenue par le projet Ethereum et suggérée dans les guides comme langue principale des contrats. Solidity est orienté objet et ressemble à JavaScript. Ci-dessous, nous présenterons quelques-unes de ses principales caractéristiques.

3.3.1 Types

Solidity prend en charge un certain nombre de types de données différents. Comme tout langage traditionnel, il prend en charge les booléens, les entiers et les chaines de caractère. Il n'y a pas encore de support pour les variables à virgule flottante.

Un type de données très intéressant est celui d'une adresse Ethereum. Il contient la représentation de 20 octets d'une adresse de compte Ethereum, qu'il s'agisse d'un compte externe ou d'un contrat. Le type de données d'adresse a des membres prédéfinis internes pour vérifier le solde d'un compte ou transférer de l'Ether via un contrat, ainsi que pour appeler des fonctions à partir d'autres contrats.

Solidity prend également en charge les structures et les énumérations ainsi que les tableaux d'octets pouvant contenir des données de tout type. De plus, les mappages de prise en charge de la solidité qui sont essentiellement des couples clé-valeur qui mappent également les clés de tout type de données à des valeurs de tout type de données. Une fonction d'accesseur est créée pour chaque variable déclarée.

3.3.2 Évènements

Les évènements sont la manière dont Solidity fournit pour accéder aux journaux de transactions, une structure de données spéciale dans la Blockchain. Les fonctions peuvent émettre ces évènements remplis de valeurs de retour, et les messages d'évènements seront diffusés et stockés sur la blockchain. Chaque application doit spécifier certaines fonctions de rappel JavaScript qui écoutent ces messages d'évènement et les impriment sur l'interface utilisateur. Les messages d'évènements ne sont pas accessibles depuis les contrats, pas même les contrats qui les ont créent.

3.3.3 Fonctions

Solidity distingue deux types de fonctions, constantes et transactionnelles. Les fonctions constantes sont les fonctions dont le but est de renvoyer une valeur et ne peuvent pas mettre à jour l'état du contrat. Ils peuvent être appelés directement et ne consomment pas de gas, car ils ne modifient pas la blockchain. Les fonctions transactionnelles permettent de modifier l'état du contrat. Chaque fois que cela est appelé, une quantité de gaz doit être fournie pour couvrir les couts de transaction. Les fonctions constantes doivent être déclarées comme telles à l'aide du mot-clé « constant ».

Il existe quatre niveaux de visibilité pour les fonctions Solidity. Ceux-ci sont :

- **Externe :** les fonctions externes font partie de la spécification du contrat et peuvent être appelées par d'autres contrats, mais elles ne sont pas accessibles par le contrat lui-même.
- **Public :** les fonctions publiques peuvent être appelées par le contrat lui-même ou par tout contrat ou entité externe.
- **Interne :** les fonctions internes ne sont accessibles que par le contrat lui-même et ses contrats dérivés (hérités).
- **Privé :** les fonctions privées ne sont visibles que par le contrat lui-même et ne peuvent être appelées par aucune entité externe ou contrat dérivé.

3.3.4 Modificateurs de fonctions

Les modificateurs de fonction sont des constructions utilisées pour changer le comportement d'une fonction. Ils sont principalement utilisés pour vérifier si une condition est satisfaite avant qu'une fonction puisse être exécutée. Les modificateurs sont des propriétés héritables des fonctions et chaque fonction peut appartenir à plusieurs modificateurs.

Chapitre 4

Notre Application

4.1 Explication de contrat mariage

En général, les contrats de mariage n'ont rien de spécial. Pourquoi ? Parce qu'une fois signé, le contrat de mariage reste généralement intact, jusqu'à un divorce potentiel. En revanche, un contrat de mariage intelligent est flexible et peut être géré activement. La femme et le mari ont leurs propres « wallets Ethereum » (leurs actifs) qui sont directement liés au contrat.

Notre application basée sur Ethereum montre ce qui est possible et comment nous pouvons utiliser la technologie blockchain dans un contexte d'affaires juridiques courantes.

Les avantages de la blockchain dans ce contexte sont la sécurité, la décentralisation et la transparence totale. Chaque activité comme un dépôt ou un paiement est automatiquement enregistrée par l'application dans la blockchain.

L'idée générale était de créer une version numérique d'un contrat de mariage traditionnel écrit qui gère toutes les fonctionnalités requises comme la signature, l'existence d'un solde ainsi que le divorce. Cependant, nous voulons rendre son équivalent numérique non seulement équivalent, mais supérieur, par conséquent, le contrat est déployé sur la blockchain.

4.1.1 Précontrat

Dans un premier temps, un précontrat doit être établi par un notaire. Le régime et les modalités du contrat de mariage sont définis dans ce précontrat, tel que :

- Divorce unilatéral : Dans le divorce unilatéral, le consentement de l'autre partie n'est pas nécessaire pour mener à bien la procédure. Donc si les époux sont d'accord que leur divorce sera unilatéral lors de la signature du contrat de mariage, ça sera pris en compte dans le précontrat, et le divorce nécessitera une seule demande de divorce par l'un des deux époux pour maitre fin au mariage. Dans le cas inverse, un divorce classique, une demande de divorce des deux époux est nécessaire pour signer le divorce.
- Envoi d'argent : Lors de la signature du contrat, on a ajouté une fonctionnalité d'envoi d'Ethereum depuis les comptes des époux au compte commun (adresse du contrat), cette fonctionnalité peut-être activée ou désactivée selon la demande des époux.

4.1.2 Smart Contract Mariage

4.1.2.1 Signature

Le contrat de mariage intelligent doit être signé. Les deux époux confirment leur identité en envoyant une transaction au contrat intelligent à l'aide de Metamask.

4.1.3 Portefeuille numérique

Notre contrat de mariage intelligent peut être utilisé comme un portefeuille numérique. Comme il est convenu dans le contrat écrit, chaque conjoint doit participer — en parts égales — aux frais liés au couple. Par conséquent, le contrat de mariage intelligent fournit les fonctions de dépôt et de paiement envers d'autres adresses.

4.1.4 Divorce

Si un divorce a été déposé et accepté par les deux partenaires. Le solde du compte commun est divisé en 2 parts égales et le paiement est envoyé aux portefeuilles respectifs. D'autres interactions avec le contrat de mariage intelligent ne sont plus possibles.

Dans le cas où un divorce unilatéral est défini dans le précontrat, un divorce peut se faire unilatéralement sans l'acceptation de deux époux.

4.2 Deploiement testnet local

Chaque version d'Ethereum est accompagnée d'un réseau de nœuds. Sur ce réseau, les transactions devront être payées avec l'Ether qui doit être acheté ou miné sur la blockchain.

Si on souhaite essayer la technologie et créer une application Ethereum décentralisée dans un environnement test sans dépenser de l'Ether « réel » ni dépenser beaucoup de ressources de calcul pour le miner, nous pouvons construire un réseau privé (Test Network), isolé du réseau Ethereum (Main Network).

Sur un réseau privé, la difficulté initiale du bloc peut être ajustée en sorte que le mining peut produire des blocs en peu de temps. Tout Ether produit ne peut pas être utilisé dans le réseau Ethereum réel, car il a été extrait sur une blockchain locale, par contre on peut l'utiliser pour couvrir les couts des transactions dans le même réseau local.

4.2.1 Ganache

Pour les besoins de notre projet, nous avons utilisé Ganache pour déployer notre blockchain locale. Ganache permet de modifier les paramètres de la blockchain comme le prix du gas et sa limite, la vitesse des blocs et le solde initial des comptes sur notre réseau. Cela nous a permis un développement plus rapide au cours des premiers stades du développement de l'application.

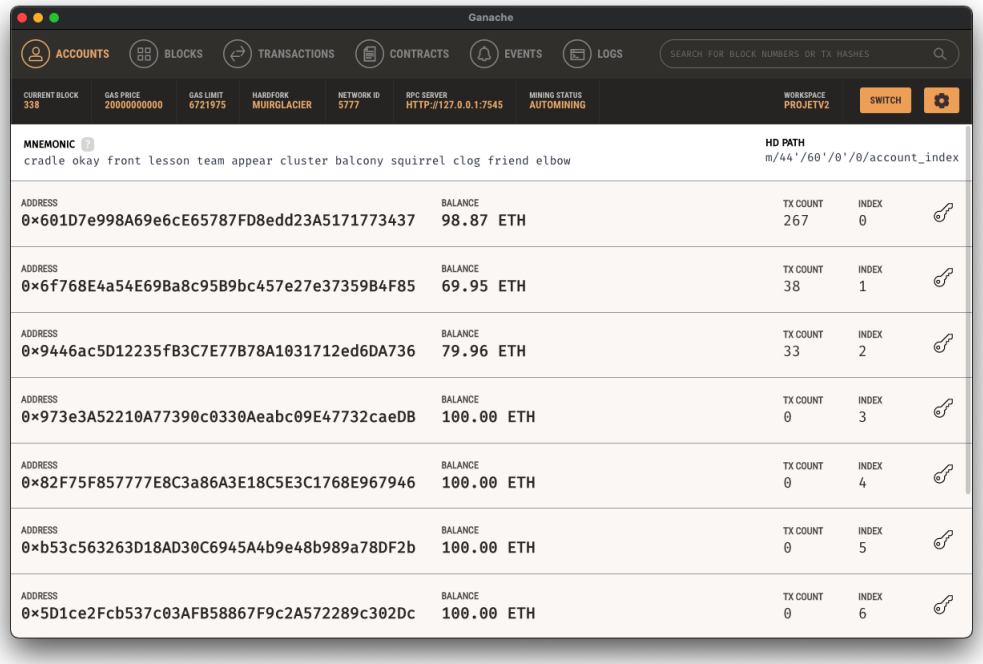


FIGURE 4.1 – Interface Ganache

4.2.2 Truffle

Pour créer une suite de tests automatiques, on a utilisé le framework de test Truffle. Il permet d'écrire des tests pour les contrats écrits en Solidity et avec NodeJS ça nous fournit une grande flexibilité lors du déploiement de nos contrats.

4.2.3 Web3

Le web3 est une collection de bibliothèques qui facilite l'interaction avec les contrats intelligents sur la blockchain Ethereum. Lors de la création d'un nouvel objet de contrat, et en donnant le fichier JSON du contrat intelligent au web3, il convertit automatiquement tous les appels en appels ABI de bas niveau via RPC. Cela vous permet d'interagir avec les contrats intelligents comme s'il s'agissait d'objets JavaScript.

4.2.4 Metamask

MetaMask est une extension pour accéder aux applications distribuées activées par Ethereum, ou « Dapps » dans les navigateurs Web. L'extension injecte l'API Ethereum web3 dans le contexte JavaScript de chaque site Web, afin que les Dapps puissent lire à partir de la blockchain. MetaMask permet également à l'utilisateur de créer et de gérer ses propres identités. Ainsi, lorsqu'une Dapp souhaite effectuer une transaction et écrire dans la blockchain, l'utilisateur dispose d'une interface sécurisée pour examiner la transaction, avant de l'approuver ou de la rejeter. Parce qu'il ajoute des fonctionnalités au contexte normal du navigateur, MetaMask nécessite l'autorisation de lire et d'écrire sur n'importe quelle page Web.

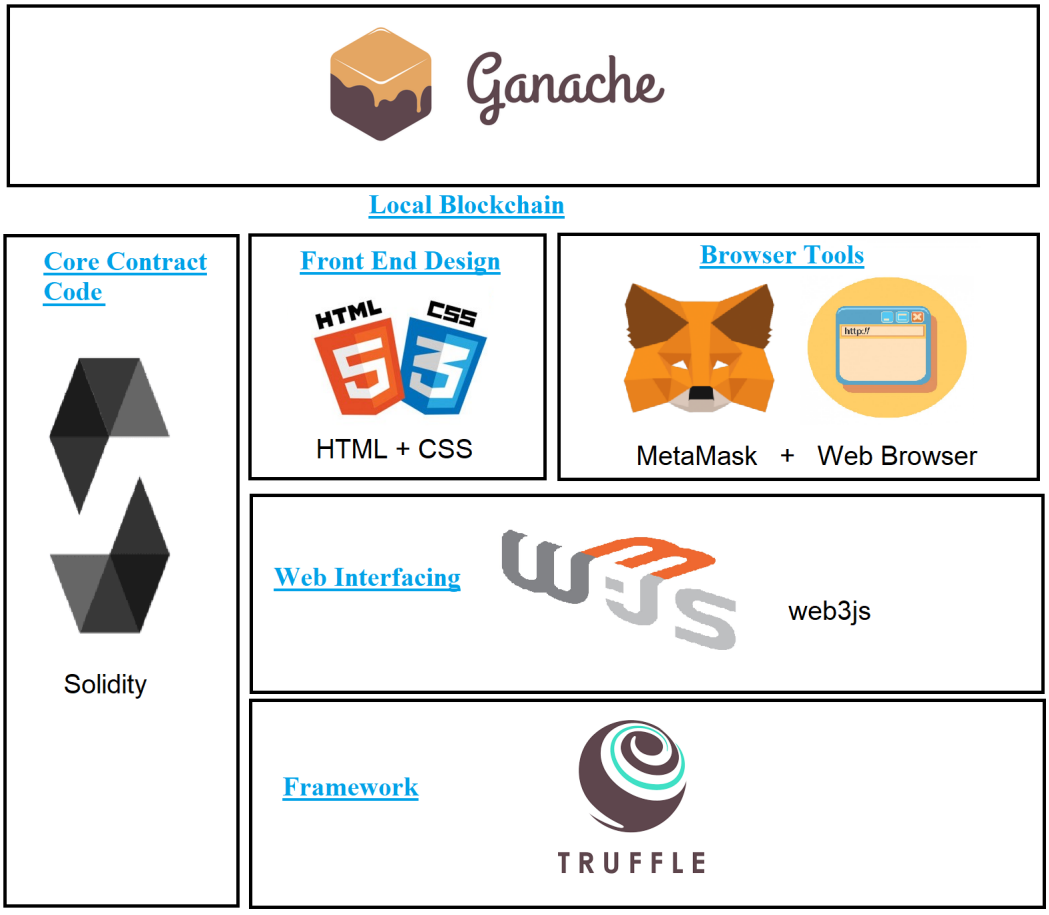


FIGURE 4.2 – Technologies utilisés pour développer notre application

4.2.5 Coût

Dans le troisième chapitre, nous avons décrit le concept de gaz.
La compilation de nos contrats a couté (1720706 + 396713) gas * 10 unités de gas (Gwei).
1 Gwei = 0,00000001 Ether
Ce qui donne un cout total de déploiement de 0.02117419 Ether.

Ainsi compiler deux contrats de taille moyenne coute environ 35,96 $. Chaque transaction qui interagit avec le contrat (i.e : Signature.) Coute une fraction de cette somme.

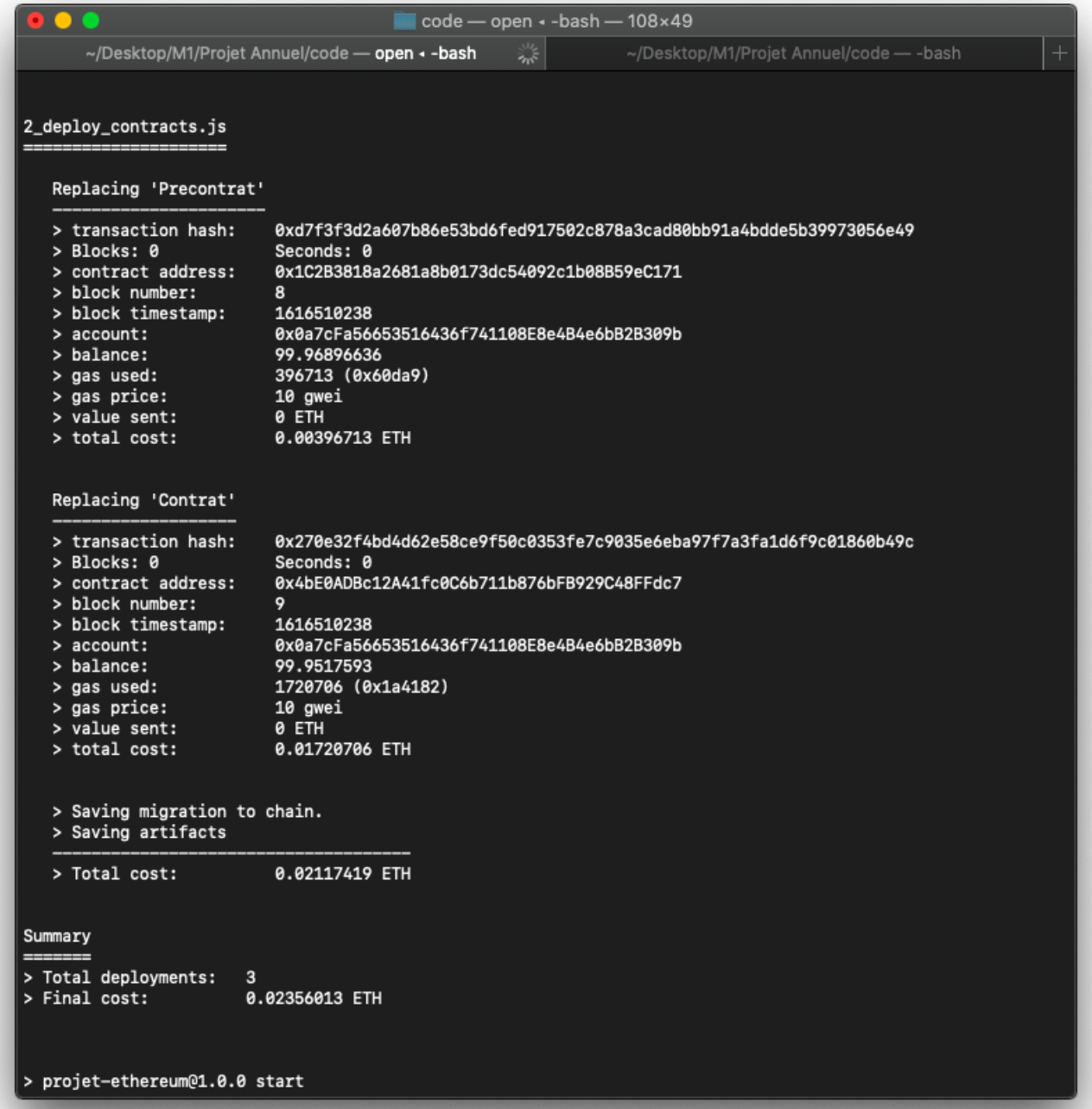


FIGURE 4.3 – Coût déploiement des contrats

4.2.6 Difficulté

Lors de l'évaluation des performances de notre projet, les facteurs pris en compte étaient, le temps nécessaire pour qu'une transaction soit ajoutée à un bloc, et combien de temps on doit passer avant qu'on puisse accéder aux données dans une requête d'écriture ou de lecture avec web3. Le temps nécessaire à une transaction dépend fortement de la difficulté du réseau.

La difficulté est généralement définie dans le fichier « genesis.json» du bloc genesis d'une blockchain et peut varier considérablement entre les différents réseaux. Par conséquent, il n'y a pas eu des mesures faites sur ce point.

Lorsque vous mesurez combien de temps il faut pour déployer un contrat intelligent et combien de temps cela prend pour récupérer les données, une observation a été faite que tant que vous avez l'adresse du contrat, il est possible de récupérer les données. Par conséquent, lors de la mesure du temps que cela prend pour pouvoir récupérer les données d'un contrat déployé, il suffit de mesurer combien de temps il faut pour recevoir l'adresse d'un contrat déployé. Il est important de noter que ces nombres sont fortement influencés par la latence réseau.

4.2.7 Sécurité

Une question de sécurité intéressante pourrait se poser. Les logiciels malveillants peuvent-ils être créés dans la Blockchain ? La réponse intuitive est que ce n'est pas possible. Alors qu'Ethereum peut mapper tout ce qui peut être programmé dans un contrat intelligent Blockchain, les contrats intelligents sont des applications qui appliquent une certaine logique et des relations entre les entités, mais contrairement à d'autres modèles de programmation, les contrats intelligents ne gèrent pas les fichiers ou n'initient pas de connexions réseau.

Cependant, la sécurité de l'ensemble de la plateforme dépend de la sécurité de la machine virtuelle Ethereum (EVM) et de chaque implémentation client. En théorie, s'il y a une faille de mise en œuvre dans le client Ethereum, un contrat pourrait essayer d'imprimer une commande exécutable qui aura réellement un impact sur le client (par exemple, lancer un transfert Ether à partir d'un compte déverrouillé). De plus, si l'implémentation EVM présente des problèmes exploitables, on pourrait tenter de sortir de l'environnement d'exécution du contrat et tenter d'injecter du code dans la mémoire de la machine hôte avec des effets destructeurs.

De plus, il est possible de voir la représentation du bytecode exécutable du contrat, mais le code source de haut niveau n'est pas toujours disponible, donc un contrat peut prétendre effectuer une action alors qu'en réalité il fait autre chose. Même si le code du contrat et la version du compilateur à travers lequel son exécutable a été produit, est connu, il y a toujours un niveau de confiance impliqué.

Chapitre 5

Conclusion

Dans ce chapitre, nous présenterons nos conclusions sur la manière dont elles répondent à nos questions de recherche. Notre principale question de recherche était la suivante : Ethereum peut-il être utilisé pour déployer des applications non traditionnelles ?

Nous pouvons affirmer que la réponse, en général, est oui. Nous avons démontré qu'une application entièrement décentralisée peut être développée et lancée avec un effort minimal sur un réseau Ethereum local. L'application que nous avons développée est bien plus qu'un simple projet « Hello World » et elle est entièrement décentralisée, ce qui ajoute une grande valeur.

Plus précisément :

1. **« Ethereum peut-il être utilisé pour déployer des applications non traditionnelles ? »**
   Comme nous l'avons déjà dit, l'application que nous avons construite implique des fonctionnalités, telles que le transfert de fonds entre les comptes, qui dans les approches traditionnelles sont trop complexes et très difficiles à résoudre. La réponse est donc un oui clair.

2. **« Quelle est la valeur ajoutée de l'utilisation d'Ethereum par rapport à une approche de développement plus traditionnelle ? »**
   La valeur ajoutée d'Ethereum devient évidente par le fait que chaque application est décentralisée et n'est contrôlée par aucune autorité unique. Dans notre exemple de notaire, dans une approche client-serveur traditionnelle, il faudrait faire confiance aux autorités opérationnelles pour assurer la sécurité du système et ne pas se falsifier les données. Dans le cas d'Ethereum, cette confiance est éliminée ou plutôt elle est répartie entre les nœuds.

3. **« Quel est le temps nécessaire pour déployer une telle solution ? »**
   Nous ne sommes pas en mesure de donner une réponse totalement claire à cette question, car nous ne pouvons qu'estimer le temps moyen de déploiement. Une application peut bien sûr être lancée instantanément après son développement. Le temps de développement, cependant, dépend fortement de l'expérience du développeur et de la complexité du contrat en question.

Amélioration

Tout d'abord, nous avons développé qu'un seul service parmi plusieurs que les notaires fournissent tels que d'autres actes de famille comme les testaments et les successions, des contrats immobiliers : signature d'avant-contrat et les contrat de vente, contrat de location... etc.

Pour l'aspect technique, nous n'avons utilisé qu'un sous-ensemble de toutes les fonctionnalités offertes par Ethereum. On pourrait examiner beaucoup plus profondément la mise en œuvre du projet et utiliser d'autres fonctionnalités et d'autres outils (comme les capacités d'interfaçage entre les applications) disponibles pour tirer encore plus parti de la puissance d'Ethereum.

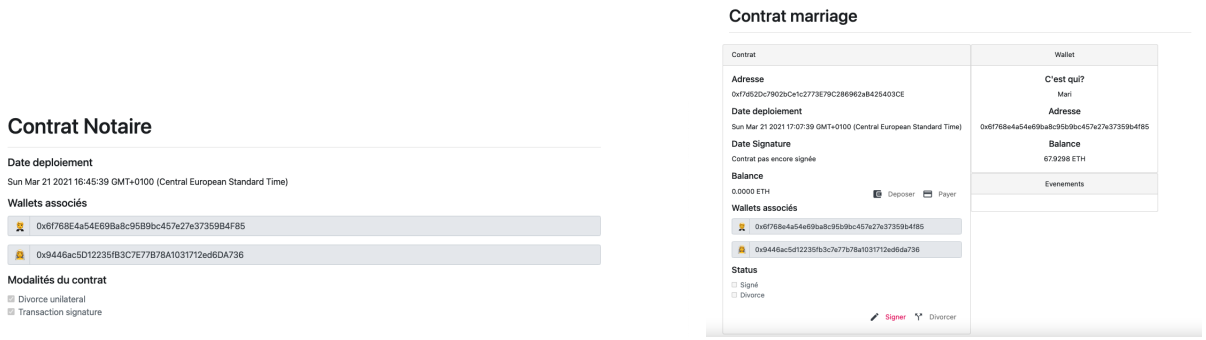
Annexes

Page d'accueil

L'écran d'accueil du contrat de mariage intelligent du point de vue du mari. Le contrat n'est pas encore signé.

Dans le coin gauche, des informations sur le contrat intelligent sont affichées. Dans celui de droite, des détails sur le portefeuille du mari sont affichés.

En bas, les actions possible sur le contrat fait avec MetaMask et les événements seront affichés.



(a) Pre-contrat (b) Contrat de mariage

FIGURE 5.1 – Différents screenshots des interfaces

Signature

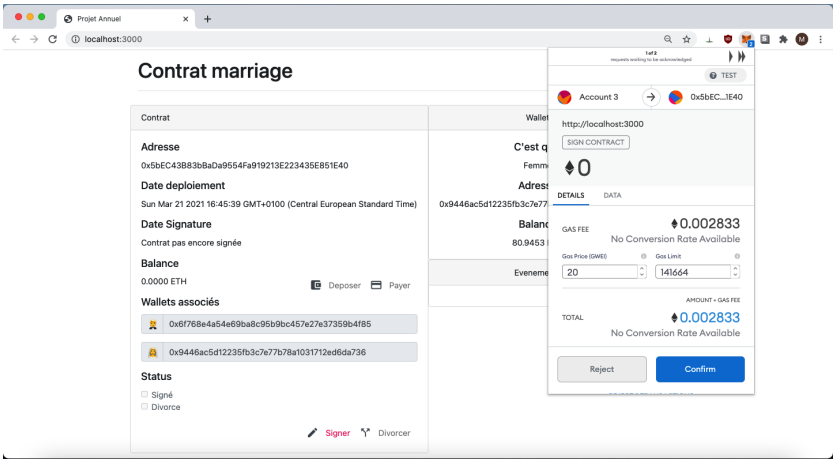
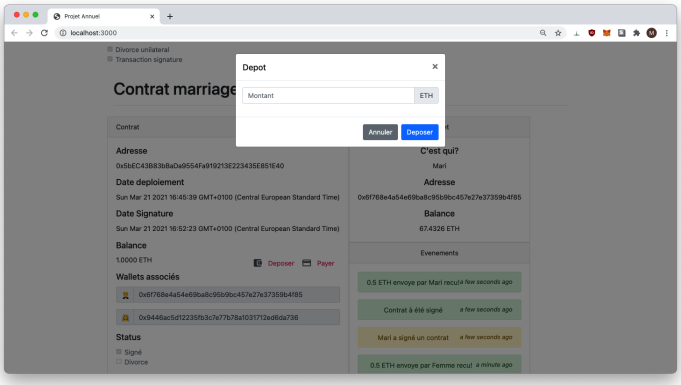
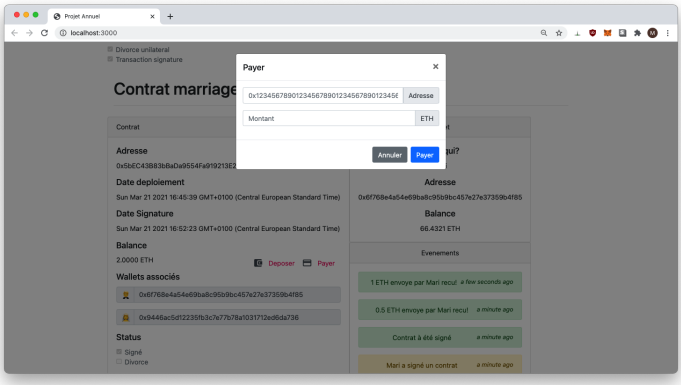


FIGURE 5.2 – Signature avec MetaMask

Fonctionnalité du portefeuille (Dépôt/Paiement)



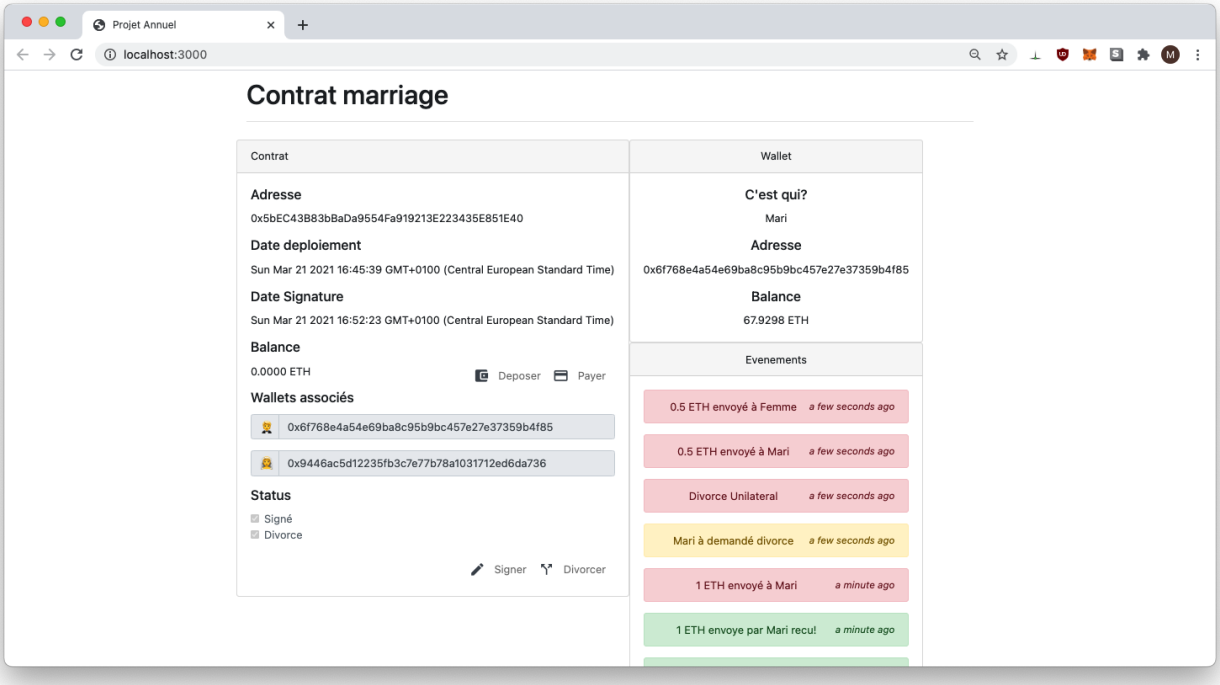
(a) Dépôt



(b) Paiement

FIGURE 5.3 – Fonctionnalité du portefeuille

Divorce



(a) Dépôt

FIGURE 5.4 – Divorce après interaction avec MetaMask

Bibliographie

[1] Ethereum blockchain app using ganache | truffle | solidity | web3.js. https://medium.com/@vshwsnahar3/ethereum-blockchain-app-using-ganache-truffle-solidity-web3-js-40dfc5369c91, 2020. [Online ; Accédé 20-Février-2021].

[2] Qu'est-ce qu'une application décentralisée (dapp) ? https://blockchainfrance.net/2018/09/14/quest-ce-quune-application-decentralisee-dapp/, 2021. [Online ; Accédé 14-Mars-2021].

[3] Truffle migrations. https://www.trufflesuite.com/docs/truffle/getting-started/running-migrations, 2021. [Online ; Accédé 20-Février-2021].

[4] Vitalik Buterin. Ethereum whitepaper. https://ethereum.org/en/whitepaper>, 2013. [Online ; Accédé le 20-Février-2021].

[5] MetaMask.io. Ethereum provider api. https://docs.metamask.io/guide/ethereum-provider.html, 2021. [Online ; Accédé le 10-Mars-2021].

[6] Satoshi Nakamoto. Bitcoin whitepaper. https://bitcoin.org/bitcoin.pdf, 2008. [Online ; Accédé 20-Février-2021].

[7] Shalu S. Step by step guide for installation of node, truffle, ganachecli and ethereum. https://medium.com/@shalu0628/step-by-step-guide-for-installation-of-node-truffle-ganachecli-and-ethereum-4d824f0d74c4 2013. [Online ; Accédé le 20-Février-2021].

[8] Tutorialspoint. Ethereum - ganache for blockchain. https://www.tutorialspoint.com/ethereum/ethereum_ganache_for_blockchain.htm, 2021. [Online ; Accédé le 29-Mars-2021].