Table des matières

Liste des tableaux

Liste des figures

1. Introduction

Depuis quelques années, les blockchains ont vu une accélération de leur développement. Les avancées dans ce milieu ont engendré une certaine frénésie et de nombreuses crypto-monnaies et applications décentralisées ont vu le jour. (Fairview Capital, [sans date]). Cette technologie, souvent perçue comme révolutionnaire, permet de réaliser une nouvelle classe d'applications et de supprimer bon nombre d'intermédiaires. Cependant, bien qu'elle soit très ingénieuse et très prometteuse, elle ne permet pas encore d'égaler les performances des infrastructures centralisées. En effet, les caractéristiques fondamentales qui font la force de cette technologie sont également un obstacle à son adoption à grande échelle. En particulier pour les cas d'utilisations où un nombre conséquent de transactions doivent être effectuées instantanément ou presque.

Dans le cadre de ce travail, nous nous intéresserons aux plateformes de trading, qui doivent être capables de traiter plusieurs milliers d'écritures à la seconde (https://www.thetradenews.com/nyse-cuts-order-latency-to-five-milliseconds/). A l'heure actuelle, les plateformes existantes ont une infrastructure très centralisée induisant une forte dépendance à l'égard d'une figure d'autorité centrale. Malheureusement ce type d'architecture est exposé à des risques et des contraintes qui sont de moins en moins favorables pour les utilisateurs (Werbach, 2017). Notamment, la centralisation d'un point unique de défaillance, les frais de transaction élevés et le manque de contrôle sur ses propres fonds.

Nous verrons que la technologie dont les caractéristiques sont orthogonales à celles évoqués précédemment est la blockchain. Est-il donc possible d'utiliser cette technologie pour mettre en œuvre une plateforme de trading à moindre frais ayant un débit suffisamment grand, tout en conservant un contrôle total de ses fonds ? Est-il possible de s'inspirer des plateformes de trading de crypto-monnaies pour en réaliser une qui permettrait d'échanger des titres ? Est-il possible de bénéficier des avantages des crypto-monnaies tout en réduisant les risques liés à leur volatilité ?

L'objectif de ce travail est de tenter d'apporter réponse à ces questions. Pour y parvenir, nous allons tout d'abord étudier le fonctionnement d'un registre décentralisé. Nous verrons comment plusieurs technologies et concepts utilisés dans la « cypherspace » ont été réunis pour créer la première monnaie virtuelle–le Bitcoin–dont la technologie sous-jacente est la blockchain.

Deuxièmement, nous découvrirons l'univers d'Ethereum, la plateforme qui permet de réaliser des applications décentralisées sophistiquées. Nous approfondirons aussi nos connaissances des Smart Contracts, de leurs avantages ainsi que de leurs limitations.

Ensuite, nous verrons comment il est possible d'augmenter le nombre de transaction traitées à la seconde, tout en bénéficiant des avantages d'une blockchain, en utilisant un mécanisme connu sous le nom de *State Channel*. Nous étudierons son fonctionnement ainsi que son utilisation dans des plateformes existantes. Cette partie nous permettra de mieux comprendre quels sont les compromis qui doivent être fait à l'heure actuelle pour qu'un débit raisonnable soit atteint.

Finalement, grâce à la recherche effectuée, nous serons en mesure de mettre en place une plateforme de trading décentralisé permettant de tester nos hypothèses. Le prototype qui sera créé utilisera des *Smart Contracts* qui seront déployés sur la blockchain Ethereum. Des *State Channels* seront utilisés pour augmenter la vitesse d'exécution des ordres d'achat et de vente d'instrument financier. Une crypto-monnaie, permettant de représenter une monnaie fiduciaire stable, sera également implémentée et utilisée comme moyen d'échange des titres en circulation sur la plateforme.

2. Blockchain

2.1 Définition générale

Une Blockchain est un registre décentralisé qui est géré entre pairs et sans l'intervention d'une autorité centrale, garantissant ainsi la transparence et la sécurité du système. Ceci est possible grâce à un mécanisme qui permet d'atteindre un consensus entre plusieurs parties distribuées (peer-to-peer) qui n'ont pas besoin de se faire confiances les unes aux autres mais qui doivent uniquement avoir confiance dans le mécanisme qui permet d'atteindre ce consensus. Grâce à ce mécanisme, aucun acteur dans le réseau de paires ne peut forcer le registre à accepter une entrée particulière qui n'a pas été validée par la majorité des autres nœuds présents sur le réseau.

Une Blockchain est donc une base de données distribuée contenant l'historique de tous les échanges réalisés entre ses utilisateurs depuis sa création. Chaque participant possède une copie de la blockchain, permettant au système de fonctionner de pair à pair (peer to peer) et donc d'éliminer l'autorité centralisée chargé de valider chacune des transactions. L'absence d'intermédiaire et la distribution de l'information sont les garants de la sécurité de la chaîne. (Blockchain France, 2016)

Comme son nom l'indique, une blockchain est composée d'une série de « blocs ». L'algorithme qui régit la blockchain enregistre chaque transaction dans un bloc sans l'aide d'un tiers. L'algorithme blockchain chiffre et authentifie automatiquement la transaction, qui est immédiatement visible par tous les utilisateurs, ce qui réduit fortement les risques de fraude. Une fois enregistrées, les données d'un bloc ne peuvent être modifiées rétroactivement sans modification de tous les blocs suivants, ce qui nécessite un consensus de la majorité du réseau.

La technologie Blockchain a été inventée pour régir Bitcoin, la première et la plus répandue des crypto-monnaies. Certaines plates-formes plus récentes, comme Ethereum, utilisent une Blockchain pour fournir un écosystème pour l'informatique distribuée, en utilisant efficacement une crypto-monnaie pour permettre d'exécuter un programme (connu sous le nom de *Smart Contrat)* garantissant que certaines conditions sont remplies avant qu'un service ne soit rendu ou qu'une transaction n'ai lieu. (Bankrate, [sans date])

2.2 Historique

Pour comprendre le fonctionnement de la Blockchain, il est intéressant de voir comment plusieurs technologies et concepts utilisés dans la « cypherspace » ont été réunis pour créer la première blockchain.

2.2.1 La liste de diffusion décentralisée

En 1997, un réseau de nœuds indépendants de liste de diffusion d'email à vue le jour. Ce réseau, appelé *Cypherpunks Distributed Remailer* (CDR), était l'œuvre des Cypherpunks, un groupe d'activistes plaident en faveur d'un recours généralisé à l'utilisation une cryptographie puissante et à des technologies renforçant la protection de la vie privée comme voie de changement social et politique. (BitcoinWiki, 2014). Ils communiquaient à l'origine par le biais de la « liste de diffusion électronique des Cypherpunks » (Cypherpunk's mailing list). Cette liste acceptait tous les emails, supprimait les informations de l'expéditeur et les acheminait au destinataire. Elle a cependant fait l'objet de modération et de censure, ce qui a poussé ce groupe à mettre en place le CDR.

Le CDR mis en place se basait sur le même concept utilisé par la liste de diffusion des Cypherpunks. La différence majeure entre ces deux technologies est l'aspect distribué du CDR. Chacun des nœuds du réseau détenaient une copie de liste de diffusion, empêchant ainsi tout type de censure ou de modération. Dans le cas où un des nœuds serait défaillant ou bien censuré, la liste continuerait de fonctionner et de relayer les emails. Peu importe à quel nœud on enverrait un email, tous les autres recevraient une copie et la transmettrait au destinataire. (Diedrich, 2016, p.262)

2.2.2 Spammeurs

Comme toutes les infrastructures d'envoi et de réception d'email, la liste de distribution des Cypherpunks a rapidement dû faire face aux Spams. La pensée générale à cette période était que les emails ne devraient tout simplement pas être (à ce point) gratuits pour l'envoyeur. Sans cette gratuité, les spammeurs perdraient intérêt à spammer.

De cette pensée, deux idées majeures ont été mises en avant. La première était d'exiger une sorte de micropaiement pour chaque envoi d'email. La seconde était d'exiger une sorte de preuve de travail (proof-of-work). Pour qu'un email puisse être diffusé à l'entièreté du réseau, il fallait que l'envoyeur prouve qu'il avait occupé son processeur (CPU) pendant un certain moment. Cette approche suffirait à ralentir les spammeurs et rendre le réseau inintéressant en vue de leurs objectifs, tout en étant « économique » financièrement (bien que pas du tout économique en matière de consommation d'énergie). (Diedrich, 2016, p.262)

2.2.3 Hashcash

En mars 1997, Adam Back, un britannique, diffuse un email sur la liste des Cypherpunks intitulé « [ANNOUNCE] hash cash postage implementation ». L'email comprend une description et une mise en œuvre précoce de ce qu'il décrit comme un « "partial hash

collision based postage scheme" » - une sorte d'équivalent de timbre postal pour les courriers électroniques, basé sur une astuce cryptographique. (Back, 1997)

Le principe était le suivant. Un email serait haché autant de fois que nécessaire avec un nonce aléatoire (nombre arbitraire ne pouvant être utilisé qu'une seule fois lors de communications cryptographique) (Wikipedia, 2013), jusqu'à obtenir un hash commençant par le bon nombre de zéros (leading zeros). Le nonce permettant d'avoir un hash acceptable serait rajouté aux métadonnées de l'email lors de son envoi. Du côté du destinataire, un rapide calcul suffirait à vérifier que le nonce est valide et de ce fait avoir une preuve mathématique qu'un certain travail a été effectué par l'envoyeur. (Diedrich, 2016, p.263-264)

Il est intéressant de noter que c'est exactement ce que font les Blockchains, ou plus précisément ce qui motive l'algorithme de consensus de Proof of Work. En particulier, Bitcoin et Ethereum utilisent l'algorithme original de Hashcash dans leur mécanisme de consensus. (BitcoinWiki, [sans date])

Cette preuve de travail crée une rareté et par extension une valeur. Avec cet algorithme, on arrive donc à faire en sorte que les emails coûtent quelque chose. Ce qui était initialement l'idée évoquée.

2.2.4 B-money

En 1998, le concept de *B-money*, a été décrit par Wei Dai sur la liste des Cypherpunk. D'après la logique selon laquelle l'envoi d'emails devrait coûter quelque chose et comment Hashcash était une solution à cela. Dai a proposé d'ancrer la valeur d'une monnaie numérique dans la rareté des nonces. Le travail pourrait être considéré comme l'acte de frapper une monnaie. Les nonces eux-mêmes seraient les objets de valeur. Le problème était de savoir comment régler la question de propriété. (BitcointWiki, 2016)

Pour pallier ce problème, Dai propose que chaque participant reçoive une copie des comptes de tous les autres participants du réseau. B-money serait transféré en diffusant les transactions à tout le monde. En cas de désaccord, chaque participant corrigerait manuellement les comptes concernés. À ce stade, il n'y avait aucun algorithme de consensus, mais on peut déjà voir le Bitcoin s'esquisser gentiment. (Diedrich, 2016, p.264)

2.2.5 Bit Gold

L'idée d'une chaîne de blocs remonte à une idée émise dans un document de Haber et Stornetta datant de 1991. Ils essayaient de trouver un moyen fiable de dater (time-stamping) des documents et ont proposé un service centralisé qui intégrerait le hash de

son dernier document daté dans le prochain document à dater. Avec ce système, toute personne détenant un document de ce service pourrait attester de la validité du document daté immédiatement avant et après sans l'aide de ce service. Cela donnerait à chacun trois options pour valider l'exactitude du document en question, tout en n'ayant pas recours au service central. (Haber, Sorneta, 1991)

C'est de ce concept que c'est inspiré Nick Szabo et qui lui a permis d'émettre sa proposition de *Bit Gold*. Son idée était d'enchaîner des hashs pour créer une chaîne de plus en plus longue et infalsifiable ou, d'après Szabo « unforgeably costly bits ». (Szabo, 2008)

Avec Big Gold, la question restait de savoir comment faire en sorte que les gens restent honnêtes et ne copie pas leurs Bit Gold lors de leur transfert à quelqu'un d'autre. Pour régler ce problème, Szabo émet l'idée qu'un registre garderait une trace de la dernière chaîne de hashs (voir ci-dessus). Les utilisateurs pourraient transférer leurs Bit Gold à quelqu'un d'autre en ajoutant un nouveau nonce et un nouveau hash incorporant l'ancien hash de la plus longue chaîne précédente, ainsi que le transfert de propriété du propriétaire précédent. Le registre identifierait la propriété par une clé publique et le détenteur de la clé privée correspondante pourrait contrôler la propriété (les Bit Gold). Donc, « dépenser » était l'acte de signer une chaîne de hash précédemment valide et signée par une multitude d'autres détenteurs. (Diedrich, 2016, p.265)

2.3 Bitcoin

La seule chose qui n'avait pas été trouvée en 1997 était de savoir comment décentraliser correctement ce registre de comptes et d'assurer leur synchronicité. Szabo, Dai et Back avaient contribués significativement en apportant tous les éléments principaux. En particulier, les nonces, les chaînes de hash et l'arbitrage indépendant. Éléments qui constituent le noyau de l'algorithme de consensus de Bitcoin. (Diedrich, 2016)

2.3.1 Proof of work

Proof-of-work (POW) est le protocole de consensus introduit par Bitcoin qui rend possible pour des milliers de nœuds de se mettre d'accord sur l'état global du réseau, en particulier sur l'état du registre des transactions. POW est utilisé pour valider les transactions, ayant lieu sur le réseau, qui sont regroupées dans des blocs et qui sont ensuite liés les uns aux autres pour former une chaine de blocs (une blockchain).

Les acteurs responsables de la vérification des transactions et de la création de nouveaux blocs sont appelés des « mineurs ». Chaque fois que des transactions ont lieu, tous les nœuds du réseau les reçoivent et vérifient leurs validités. Les mineurs, des

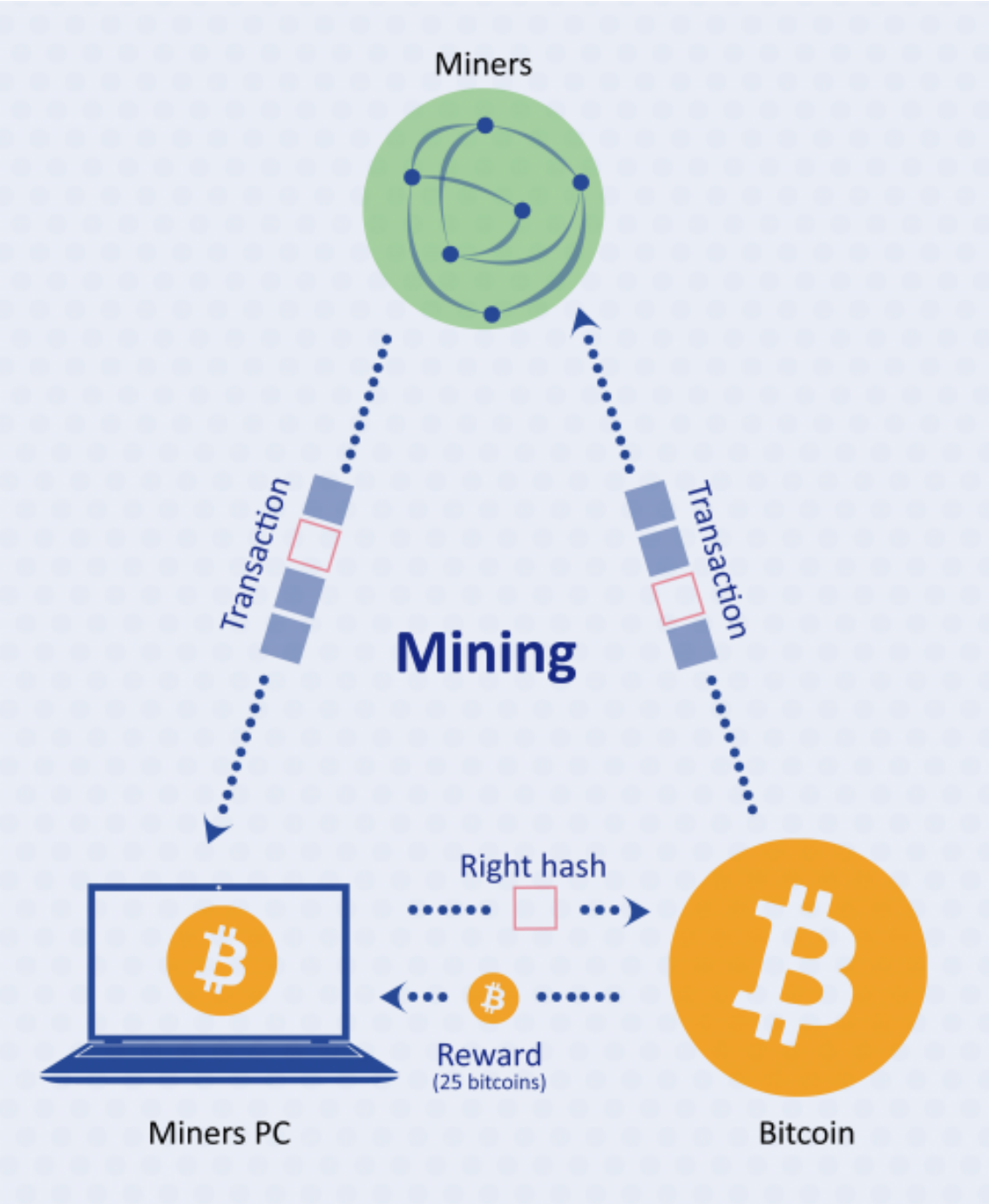
nœuds sur le réseau, collectent ensuite les transactions et les organisent en bloc. Ce processus est généralement appelé « minage » (Binance, [sans date]).

La première étape du minage d'un bloc est de hacher individuellement chacune des transactions reçues. Une fois les hashs calculés, ils sont organisés dans un arbre de Merkle qui est formé en organisant les divers hashs des transactions par pairs et à les re-hacher jusqu'à ce que le sommet de l'arbre soit atteint. Le sommet de l'arbre est également appelé hash-racine (root hash en anglais) et est tout simplement un hash unique qui représente tous les hashs précèdent utilisés pour le générer. (Binance, [sans date] b)

Une fois le hash-racine obtenu, il est placé dans l'en-tête (header) d'un bloc avec le hash du bloc précédent et un nonce aléatoire. L'en-tête est ensuite haché et le résultat est utilisé comme identifiant unique du bloc généré (appelé bloc candidat). Afin d'être valide, ce nouveau hash obtenu doit répondre à certaines conditions. Notamment, il doit commencer avec un certain nombre de zéros. Ceci indiquant qu'il est effectivement inférieur à la valeur limite définie par le protocole. Cette valeur limite représente la difficulté de hachage et est régulièrement mise à jour par l'algorithme POW afin de garantir une vitesse de création de blocs constante et proportionnelle au nombre de mineurs (Bitcoinwiki, 2019).

Le processus de minage requiert donc aux mineurs d'hacher l'en-tête d'un bloc avec un nouveau nonce tant qu'un hash valide n'a pas été générer. Lorsqu'un hash valide est découvert, le nœud l'ayant découvert va diffuser le bloc sur le réseau. Tous les autres nœuds vont ensuite vérifier la validité du hash, et donc du bloc, par une simple opération. Si le hash est confirmé, ils vont ajouter le bloc à leur copie de la blockchain. Le mineur ayant produit le bloc approuvé sera ensuite récompensé par une somme de monnaie virtuelle propre à la blockchain en question, dans ce cas-là, en Bitcoin (Binance, [sans date] b). Ce processus est illustré dans la figure 1 en page suivante.

Figure 1: Bitcoin Mining



(Binance, [sans date] b)

3. Ethereum

Le Bitcoin est relativement simple et facile à comprendre du point de vue d'un utilisateur standard. Il y'a des portefeuilles, tout le monde peut transférer de la monnaie virtuelle d'un portemonnaie à un autre. Le réseau est assez sophistiqué et permet d'éviter d'avoir une entité centrale ayant le contrôle, tout en résolvant les problèmes classiques liés aux échanges virtuels. Cependant, les interactions monétaires et les transferts de propriété sont très souvent plus complexes qu'un simple échange unidirectionnel.

En 2013, Vitalik Buterin propose l'idée d'Ethereum. Il soutenait que Bitcoin avait besoin d'un langage de script pour le développement d'applications plus sophistiquées. Ne parvenant pas à ces fins avec la communauté Bitcoin, il propose alors le développement d'une nouvelle plateforme avec un langage de script plus général. L'idée de Buterin, est de mettre en œuvre un système de paiement qui permette à n'importe qui de créer des algorithmes qui agissent sur les comptes de la blockchain. Ces algorithmes peuvent recevoir de la monnaie virtuelle et décider de combien et à qui transférer cette dernière. Pour que cela soit possible et corresponde aux principes fondamentaux du protocole Bitcoin, les algorithmes doivent répondre aux conditions suivantes : ils doivent être transparents, ils doivent être prévisibles (déterministes) et personne ne doit pouvoir les modifier une fois qu'ils sont déployés sur le réseau. Contrairement au Bitcoin, il existe deux types de portemonnaie dans la chaine Ethereum. Ceux gérés par des humains et ceux gérés par des algorithmes (Podolan, 2017).

3.1 Smart Contracts

Ces algorithmes précédemment évoqués sont appelés « Smart Contract » (contrat intelligent en français). D'après la documentation officielle de Solidity, le langage utilisé pour réaliser des applications décentralisées, les Smart Contracts (SC) sont définis comme une collection de code (des fonctions) et de données (un état) qui réside sur la blockchain Ethereum (Solidity, 2019)

Une autre définition intéressante est celle de la Fondation Ethereum :

> *« Smart contracts are applications that run exactly as programmed without any possibility of downtime, censorship, fraud or third-party interference. »*

Les caractéristiques évoquées étant inhérentes à la blockchain, elles sont donc forcément mises en œuvre pour chaque application décentralisée. Analysons les points évoqués pour mieux comprendre les avantages que les SC assurent (Consensys, 2019).

Tout d'abord, l'indisponibilité (downtime). Chacun des nœuds (les ordinateurs exécutant le protocole) du réseau ayant une copie de la chaine et des contrats qui y sont déployés,

l'exécution d'un SC sera donc toujours identique pour tous les acteurs du réseau. Chaque nœud est donc interchangeable et peut garantir une exécution déterministe d'un contrat. Les applications ne peuvent donc jamais s'éteindre soudainement, ou même être éteintes soudainement. Elles restent donc en permanence disponible.

Ensuite, la censure (censorship). Étant donné que les nœuds sur la blockchain Ethereum sont distribués à travers le monde, cela élimine tout risque de censure provenant d'une autorité centrale.

Puis, la fraude (fraud). Les contrats ne peuvent pas être modifiés, piratés ou bien manipulés sans un consensus de la majorité du réseau. Il n'est donc pas possible pour un acteur ou même un groupe d'acteurs de saboter l'exécution d'un contrat sans se faire remarquer par le reste du réseau.

Finalement, les tierce parties (third parties). Le déclenchement d'un contrat étant autonome, il ne nécessite donc d'aucun intermédiaire pour assurer ou assister une transaction. L'unique responsable de l'exécution d'une transaction et donc le contrat lui-même (Daniel, 2019).

3.1.1 Token

Dans l'écosystème de la blockchain, ce qui est appelé « token » fait référence à tout type d'actifs pouvant être transféré sur le réseau entre deux acteurs. (Coinhouse, [sans date]) Un token est mis en place à l'aide d'un Smart Contract et répond généralement au standard d'échangeabilité ERC-20 (qui sera discuté plus en détail dans la partie sur l'implémentation).

3.1.2 Limitations

3.1.2.1 Réseaux partagés et débit limité

La plateforme Ethereum n'étant supportée que par une blockchain (autrement dit un seul réseau partagé) cela introduit une limite stricte sur le nombre de transactions que le réseau peut traiter à un instant donné. En l'occurrence, la blockchain Ethereum ne peut traiter que 15 transactions par seconde (Hertig, 2018).

Bien que cela puisse sembler être un nombre suffisant pour un réseau fonctionnel, ce n'est malheureusement pas le cas. En particulier pour des applications complexes qui ont besoin d'exécuter des centaines, voire des milliers, de transactions par seconde pour pouvoir fonctionner correctement. Notamment, pour les plateformes de trading qui ont besoin d'une exécution quasi instantanée pour un ordre et qui doivent traiter plusieurs centaines de milliers de transactions par seconde (Fisher, 2008).

Plus grave encore, il y a actuellement plus de 264'000 tokens déployés sur la blockchain Ethereum (https://etherscan.io/tokens). Si l'on part du principe qu'uniquement 40% des contrats ERC20 sont actifs, le réseau ne parviendrait même pas à traiter une transaction toutes les deux heures pour chaque application active (Kiffer, Levin, Mislove, 2018).

3.1.2.2 Sécurité des Smart Contracts

Une autre conséquence liée à la décentralisation est le manque d'audits de sécurité des SC qui sont exécutés sur la plateforme. En effet, n'importe qui peut exécuter n'importe quel contrat sur la plateforme tant que les frais de transactions sont payés. Ce qui implique que les développeurs peuvent déployer et utiliser des contrats comportant des bugs et des failles de sécurité importantes.

La majorité des SC sur la blockchain effectuent des opérations sur les comptes des utilisateurs. Une erreur de programmation dans un contrat peut exposer à un risque toutes les adresses associées avec ce dernier. Dans le cas où un acteur malveillant découvrirait une vulnérabilité, il pourrait l'exploiter et siphonner les fonds provenant de chacune des adresses aillant effectuer une transaction avec le Smart Contract défaillant (Daniel, 2019)

Pour des investisseurs ordinaires et des utilisateurs basiques, s'assurer qu'un contrat n'est pas défectueux dépasse de loin leurs capacités techniques. Ce qui conduit à des investissements mal informés dans des contrats défaillants qui peuvent engendrer des pertes importantes. C'est d'ailleurs ce qui s'est produit à plusieurs reprises. Comme, par exemple, le vol de l'équivalent de 50 millions de dollars en Ether du projet DAO (un fond de capital-risque mis en place sur la blockchain Ethereum) (Finley, 2016)

3.1.2.3 Le problème des oracles

Dans l'univers de la blockchain, le terme « oracle » fait référence à un nœud ou un petit groupe de nœuds du réseau ayant le pouvoir d'envoyer sur la blockchain des données qui proviennent d'une source externe à cette dernière. Par exemple, un tel type de données pourraient être le taux de change entre deux crypto-monnaies. Le contrat appellerait une fonction lui permettant de recevoir des données en temps réel provenant de l'oracle (Mou, [sans date]).

Cela présente un paradoxe. Alors que le principe de la blockchain est d'être complètement décentralisé et de ne pas dépendre d'une entité centralisée, l'utilisation d'un oracle est, par sa nature même, centralisée et dépendante d'un tiers de confiance. Les oracles, étant des sources d'informations centralisées, sont donc exposés aux

mêmes risques d'attaques que toutes les structures de ce type, ainsi qu'a des risques de manipulation ou de collusion (Daniel, 2019).

3.2 State Channel

Pour faire face au faible débit de transaction tout en bénéficiant des avantages d'une blockchain, il est possible d'utiliser certains mécanismes. Un de ces mécanismes est l'utilisation de canaux de paiement (Payement Channel). Fondamentalement, ces canaux permettent d'effectuer des transactions de crypto-monnaie en dehors de la blockchain sur laquelle elles sont déployées (Harding, 2019). Le principe des canaux de paiement est relativement simple et peut être expliqué simplement à l'aide d'une analogie.

Le fonctionnement des canaux de payement peut être comparé à celui des chèques. Admettons qu'un tiers souhaite acheter régulièrement des produits chez un marchand. À chaque fois qu'il finit de faire ses courses, il signe un chèque avec le montant qu'il souhaite transmettre au vendeur. Le vendeur peut ensuite décider d'encaisser le chèque à la banque. Cependant, chaque fois qu'il en encaisse un, il paye des frais de transaction fixe. Le vendeur a donc intérêt d'encaisser l'argent moins fréquemment et avec de plus gros montant. Le client peut donc conclure un accord avec le marchand et lui transmettre un nouveau chèque lors de chaque nouvelle transaction. Ce dernier comportant le montant de tous les achats antérieurs ainsi que celui en cours, la date et la signature du client. Périodiquement, ou bien une fois que le client a effectué tous ses achats, le vendeur peut aller encaisser le dernier chèque qu'il aura reçu.

En résumé, nous avons donc deux acteurs qui souhaitent effectuer des transactions entre eux, des chèques qui donnent aux détenteurs le droit de retirer une certaine somme sur le compte du signataire et une banque qui permet d'échanger des chèques contre de l'argent. Dans un canal de paiement, la banque serait représentée par un contrat sur la blockchain et les chèques par des messages signés cryptographiquement et échangés hors-blockchain. Le processus d'échange est expliqué en détails dans la section qui suit.

3.2.1 Canaux de payement

Deux parties désireuses d'échanger des petites sommes d'argent régulièrement et rapidement vont envoyer une portion de leurs fonds vers un Smart Contract afin qu'ils servent de garantie pour leurs échanges en dehors de la blockchain ; une caution en quelque sorte.

Dès que les fonds sont bloqués, ils peuvent s'échanger des messages hors-blockchain. Pour qu'un message soit accepté par le SC il faut qu'il soit signé par les deux parties. Chaque transaction doit également contenir le nonce courant de la blockchain afin que le SC puisse connaitre l'ordre chronologique des transactions.

Une fois que les deux parties ont terminé leurs échanges, elles peuvent débloquer leurs fonds en soumettant la dernière transaction datée et signée par les deux au SC. Celui-ci, démarre une période de challenge dans laquelle il attend que la seconde partie envoie une transaction identique afin d'avoir la certitude que les deux parties sont d'accord sur cette finalité. Le SC se charge ensuite de vérifier la validité de la transaction reçue en comparant les nonces, les signatures ainsi que les montants. Une fois validé, le contrat envoie transfert à chaque partie le solde correspondant. (Tremback, 2015)

Dans le cas où l'un des deux acteurs enverrait une transaction n'étant pas la dernière en date afin de tenter d'arnaquer l'autre (ex : une transaction étant plus favorable pour un des acteurs). L'autre peut envoyer une transaction plus récente durant la période de challenge initié par le SC afin de contester la première soumission frauduleuse. Tant que les deux parties ne se mettent pas d'accord sur un état final, le SC relance la période de challenge. Si la période de challenge est dépassée, le SC transfert les fonds d'après la dernière transaction reçue. (Bitcoin.org, 2017)

La blockchain est donc utilisée comme arbitre et comme registre final plutôt que comme plateforme de payement direct. Bien que le mécanisme de challenge soit ingénieux, il finit par avoir peu d'importance car, la théorie des jeux (game theory) de cette situation est suffisante pour dissuader les acteurs d'avoir un comportement malhonnête. Tant que ce mécanisme est théoriquement valable, il ne sera probablement jamais utilisé. De plus, avoir recours au processus de minuterie peut entraîner des frais supplémentaires et donc amoindrir davantage le gain qui aurait pu être fait en réalisant le retrait de manière honnête. Dans certains cas, il est également possible de mettre en place un mécanisme de pénalité afin de dissuader d'avantage les acteurs frauduleux. Ceci est suffisant à pousser les parties prenantes à ne soumettre au SC qu'uniquement des transactions représentants un accord mutuel. (Coleman, 2015)

3.2.2 Canaux d'état

Les canaux d'état (State Channel) améliorent les performances de la blockchain en supprimant les opérations de modification d'état d'une blockchain et en les exécutant directement entre des ensembles définis de participants. Les canaux de paiement ont été le premier type de State Channel à être décrit. Ils utilisent des interactions off-chain pour modifier la propriété des bitcoins verrouillés on-chaine, permettant ainsi aux

utilisateurs de procéder à des paiements instantanés hors-chaîne. Le terme « State Channel » généralise cette approche, au-delà des paiements, englobant tous les types de modification d'état possible sur la blockchain. (Coleman, 2015)

Les State Channels font donc référence au processus par lequel les utilisateurs négocient ou échange directement entre eux des changements d'état sans passer par la blockchain et minimisent considérablement le nombre d'opérations sur la blockchain. C'est l'une des principales solutions d'évolutivité (scalability) d'Ethereum et la plus avancée à l'heure actuelle, pouvant être utilisée en production. (District0x, [sans date])

Elles sont particulièrement avantageuses et intéressantes dans les cas d'utilisation qui demandent un grand nombre de changements d'états instantanés avant l'obtention d'un résultat final. Par exemple, pour un jeu de poker en ligne. En passant uniquement par la blockchain, l'expérience d'utilisation serait vraiment très mauvaise dû aux temps de latence entre une action en jeu et la synchronisation de la blockchain. De plus, les gains obtenus par les joueurs serait fortement réduit dû au coût élevé de chaque transaction placée sur la blockchain et ne permettrait pas aux joueurs de parier des *micro-montants*. En revanche, avec les canaux d'état, les joueurs n'ont qu'à payer les frais de transaction au début de leur session de jeu, ainsi qu'à la fin, tout en ayant la possibilité de s'échanger autant de « changement d'états » que désiré, de manière instantanée et gratuite.

4. Plateformes de trading

Il existe deux types de plateforme d'échange d'actifs ; centralisé et décentralisé. Comme leur nom l'indique, les échanges décentralisés existent sans avoir besoin d'un intermédiaire. Cela signifie que l'utilisateur contrôle ses fonds. Permettant aux acheteurs et aux vendeurs de traiter directement les uns avec les autres.

En gérant les transactions par eux-mêmes, ils peuvent effectuer des échanges avec plus de confiance car ils ne partagent pas de données personnelles ou financières avec une organisation tierce. De plus, les échanges décentralisés bénéficient d'une sécurité plus forte, car les données des utilisateurs ne sont pas stockées dans un seul système (point de défaillance centralisé), mais via une variété de nœuds, ce qui les rend plus résistants au piratage et aux attaques (Blockbasis, [sans date]).

4.1 Architecture des échanges décentralisés

Le terme « échange décentralisé » fait généralement référence aux applications permettant à deux parties de conclure un échange sans devoir passer par un tiers de confiance, mais plutôt en utilisant des protocoles se reposant sur un registre décentralisé. Dans notre cas de figure la blockchain.

Ces échanges décentralisés vont donc bien évidement hériter des avantages et inconvénients de la technologie sous-jacents. Par extension, ils héritent également du *trilemme* (Buterin, 2018) auquel les blockchain (et les registres décentralisés de manière plus générale) sont confrontés. Ce trilemme affirme que les blockchains ne peuvent garantir au plus que deux des trois propriétés suivantes:

- Décentralisation
- Extensivité (scalability)
- Sécurité

Chacune de ces caractéristiques implique également d'autres aspects. Par exemple, plus un système sera décentralisé, plus il sera « trustless ». C'est-à-dire qu'il existe des mécanismes permettant à toutes les parties du système de parvenir à un consensus sur l'état du système à un moment donné. Le pouvoir et la confiance sont distribués (ou partagés) entre les parties prenantes du réseau (les nœuds), plutôt que concentré dans une seule personne ou entité (banques, gouvernements et institutions financières, par exemple(Lisk, [sans date]). En revanche, plus il est décentralisé, moins il est extensible. En prenant le mécanisme de consensus Proof-of-work, par exemple, on se rend vite compte que plus il y a d'acteurs sur le réseau, plus le temps requis pour confirmer une transaction est élevé. Il est donc très difficile, voire impossible d'augmenter le débit de

transaction sans utiliser des astuces ou des techniques d'évolutivité (*scalability)* empruntée de l'informatique distribué, tout en maximisant la décentralisation et la sécurité à la fois (Gilbert, Lynch, 2011).

Outre ces contraintes, les échanges décentralisés offrent un certain nombre d'avantages importants, notamment un risque moindre lié au faille potentiel de sécurité d'une plateforme ou système centralisé, la possibilité de réduire les frais de transaction et d'avoir une gamme d'actif inter-échangeable plus diversifiées.

Suivant les cas d'utilisations, différentes architectures peuvent être adoptée pour tenter de maximiser l'une des propriétés précédemment évoquées. Toutefois, les échanges décentralisés s'appuient, en règle générale, sur un panel de composants semblables que l'on retrouve aux travers de tous les échanges. Certains tenterons de maximiser décentralisation et extensivité, d'autre de maximiser sécurité et performances. Peu importe l'approche choisie, ce type d'échange est globalement composé des composants suivants :

- La blockchain et les standards de compatibilité choisis
- Le mécanisme de découverte d'ordres
- L'algorithme d'allocation d'ordres
- Le protocole de règlement (ou d'établissement) de transactions

Pour répondre à certains critères de performance ou à certain attribut de qualité, les composants d'une application d'échange décentralisée peuvent ne pas être totalement décentralisés. Un ou plusieurs de ces composants peuvent être « sorti » de la blockchain et centralisés.

Pour mieux comprendre les différentes approches possibles et surtout les avantages et inconvénients de chacune, il est important d'éclaircir l'utilité et le rôle qu'on ces composants dans un tel système.

4.1.1 La blockchain et les standards de compatibilité choisis

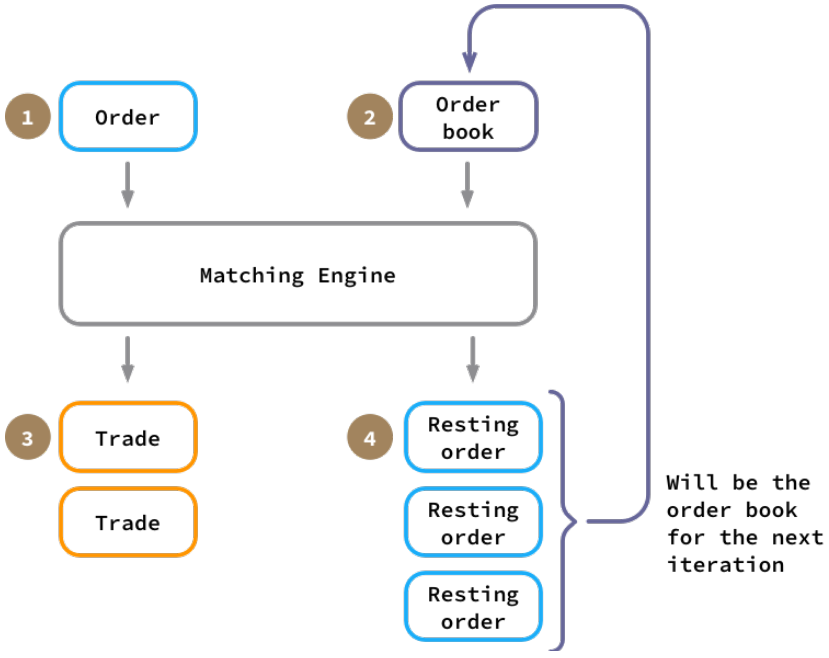
Pour que deux actifs, ou tokens, puissent être échangés, il faut qu'ils répondent à un standard technique afin d'être aussi interchangeable que possible et pouvoir être utilisé par un certain protocole. Différents standards existent pour diffèrentes blockchain et cas d'utilisation. Le standard de tokens le plus connu est probablement celui d'Ethereum pour l'implémentation de tokens transférable ; le standard ERC-20. Cette interface fournit les fonctionnalités minimales requises pour rendre possible le transfert de ces tokens. Ce standard permet également de les rendre approuvables et dépensables par

une partie tierce autorisée (Pirapira, 2018). Un standard semblable, le NEP-5, a été mis en place pour les tokens créés sur la blockchain NEO (Zhang, 2017). Il en existe bien évidement tout une multitude pour des cas de figure bien particulier, comme le ERC-721 pour les tokens non fongibles sur Ethereum (Fulldecent, 2018) Grâce à ces normes, il est facile de mettre en circulation de nouveaux tokens sur un échange utilisant l'interface en question.

4.1.2 Matching engine

Un « matching engine » est un algorithme d'allocation d'ordres qui a pour but de trouver une correspondance entre un ordre ouvert d'achat ou de ventes avec un ordre fraîchement reçu. Pour avoir une idée générale de son fonctionnement, nous pouvons le considérer comme une fonction (voir image ci-dessous) prenant comme paramètres d'entrée un ordre et un carnet d'ordres. Cette fonction renvoie une liste des transactions, ainsi que tous les ordres restants n'ayant pas été exécutés. Les ordres restants deviendront le carnet d'ordres du prochain ordre reçu par le *matching engine. (CME Group, [sans date])*

Figure 2: Algorithme d'appariement d'ordres



(Ghazi, 2018)

4.1.3 Le mécanisme de découverte d'ordres

Les mécanismes de découverte d'ordres permettent aux acheteurs de découvrir des vendeurs disposés à exécuter des transactions à des conditions mutuellement acceptables. Sur les plateformes de trading centralisées de crypto-monnaie, telles que Binance, Bitfinex et Kraken, les traders ont la possibilité de soumettre à la fois des ordres

de type *limit* ou des ordres de type *market*. Ces ordres sont automatiquement mis en correspondance (par le matching engine) avec des contreparties non identifiées par le biais du carnet d'ordres qui regroupe tous les ordres placés par les utilisateurs. Sur ces échanges, l'allocation d'ordre est effectué quasi-instantanément (pourvu qu'il y ait de la liquidité au niveau de prix donné).

Du côté des échanges décentralisés le concept des carnets d'ordre est également présent. Ces carnets peuvent exister *on-chain (implémenté sur la blockchain par un smart contract)* ou bien *off-chain* (hébergé par des tiers sur un serveur centrale). La plupart des carnets d'ordres décentralisés affichent les ordres placés par chaque contrepartie, plutôt que d'afficher une liste d'ordre agrégées de toutes les contreparties. Pour qu'un échange ait lieu, il faut donc qu'un utilisateur identifie une contrepartie ayant émise un ordre particulier avant de pouvoir effectuer des opérations (Lin, 2019).

Certain échange décentralisé propose une alternative aux carnets d'ordres *on-chain* en mettant en place un *smart contract* faisant office de « automated market maker » (AMM). Ces systèmes adoptent un modèle d'ajustement de prix dans lequel toutes les parties négocient avec la AMM directement. Le prix de marché d'un actif est déterminé par l'offre et la demande de ce marché. Les AMM offrent une disponibilité, une meilleure expérience d'utilisation et des performances accrues par rapport aux carnets de commandes *on-chain*. Ils sont cependant encore beaucoup plus lents que les échanges centralisés. Aussi, ils doivent imposer des contraintes artificielles à l'offre pour éviter que leurs réserves ne soient épuisées par les potentiels traders pratiquent de l'arbitrage et limiter l'impact, dû à la taille d'un ordre, sur les fluctuations de prix (Warren, Bandeali, 2017).

4.1.4 Carnet d'ordres on-chain

Certains échanges décentralisés utilisent des carnets d'ordres placé directement sur la blockchain (dans un smart contract). Dans ces systèmes, les utilisateurs doivent effectuer des transactions sur la blockchain à chaque fois qu'ils souhaitent passer, modifier ou annuler un ordre et donc payer des frais lors de chaque action. Bien que le coût d'une transaction unique soit peu élevé, la modification fréquente des ordres pour s'adapter à l'évolution des conditions du marché devient vite couteuse. De plus, au fur et à mesure que de nouveaux ordres sont passées, un smart contract doit exécuter un mécanisme d'allocation des ordres qui s'exécute lentement (et de manière redondante) sur toutes les machines virtuelles du réseau. En plus de ça, le fait de tenir à jour le carnet d'ordre sur la blockchain consomme une quantité de bande passante non négligeable et l'alourdit sans forcément entrainer un transfert de valeur (Warren, Bandeali, 2017, p.2)

4.1.4.1 Avantages d'un carnet d'ordre on-chain

Comme évoqué plusieurs fois au préalable, un mécanisme placé directement sur la blockchain et implémenter à l'aide d'un Smart Contract, va hériter directement des propriétés techniques fondamentales de la blockchain en question. Par conséquent, un order book *on-chain* sera moins facilement censurable que sa contrepartie *off-chain.* Comme le carnet d'ordre sera distribué et répliqué à travers tout le réseau, il n'y aura pas de point d'attaque centralisé. La probabilité qu'un carnet d'ordre décentralisé se fasse abattre ou bien compromettre sera donc quasiment nul.

Également, le carnet d'ordre héritera de la propriété de « trustlessness » et les acteurs utilisant celui-ci n'auront pas à avoir confiance en une entité centralisé pour publier, diffuser et réaliser leurs ordres de manière équitable, juste et fiable. Ils n'auront qu'à avoir confiance dans le processus, normalement transparent, publique et théoriquement déterministe, implémenter par le smart contrat.

4.1.4.2 Inconvénients d'un carnet d'ordre on-chain

De manière orthogonal au dernier point évoqué, le carnet d'ordre *on-chain* héritera des inconvénients et des problèmes de la blockchain sous-jacente. La rapidité et le coût de soumission ou de suppression d'un ordre dans un carnet d'ordres *on-chain* sont limités par la vitesse et le coût d'interaction avec la blockchain. Les utilisateurs devront payer pour chaque mise à jour du carnet d'ordres sur le réseau, attendre que le réseau parvienne à un consensus sur leurs mises à jour, puis attendre une confirmation des mises à jour. Par conséquent, les blockchains avec des frais plus élevés et des temps de transactions plus grands sont peu favorables pour l'hébergement d'un carnet d'ordre décentralisé.

En fin, contrairement aux carnets d'ordre centralisé, ils sont soumis à une latence provenant du temps de création de block. En effet, les order books *on-chain* sont généralement mis à jour en fonction des informations contenues dans le dernier bloc validé de la chaîne. Il se peut donc qu'entre le moment où un ordre a été alloué et le moment où il est validé par l'ensemble de la blockchain, plusieurs minutes se soient écoulées. Ceci peut être fortement désavantageux dans un marché très volatile où des changements de positions rapides sont un atout clé pour qu'une stratégie de trading soit profitable.

4.1.5 Carnet d'ordres off-chain

Les carnets d'ordres *off-chain* sont hébergés sur un serveur centralisé (hors de la blockchain) et sont détenus par une un tiers de confiance. Cette entité centralisée aide les acheteurs et vendeurs à trouver une correspondance à leurs ordres et permet de

créer un marché plus liquide et plus transparent (à condition que ce tiers de confiance soit réellement honnête et fair-play et qu'il puisse le prouver).

Le choix d'utiliser un carnet d'ordre implémenté directement sur une blockchain ou bien en dehors de celle-ci dépend fortement des performances de la blockchain en question. Les échanges décentralisés suffisamment performant (et rentable) pour être en production n'utilisent généralement pas de carnet d'ordres *on-chain* étant donné que chaque émission ou modification d'ordre déclenche une mise à jour de la blockchain entrainant des frais de transaction ainsi que des délais. En revanche, sur certaine blockchain les frais de transaction sont négligeables et les temps d'attente sont raisonnablement court (de l'ordre de quelque seconde). Bien que ces avantages soit intéressant, il ne faut pas oublier qu'ils existent au détriment d'autres attributs qualités. Dans de telles circonstances, un carnet d'ordres *on-chain* peut être intéressant à mettre en œuvre pour un volume modérés d'ordres.

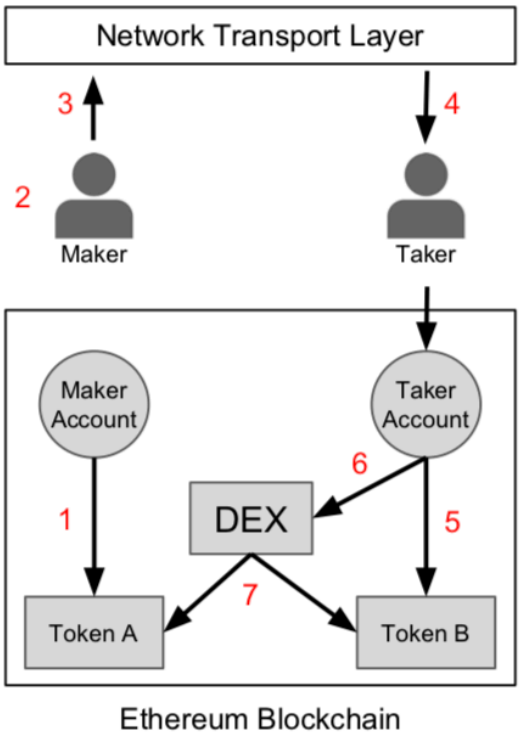
Malheureusement, dans la blockchain Ethereum, les frais de transaction ne sont pas négligeables et les temps d'attente sont de l'ordre de quelques minutes. Par conséquent, l'utilisation d'un carnet d'ordre implémenté directement sur cette blockchain entraînerait probablement des frais de transaction élevés et des temps de latence exagérément long pour une application de ce type. Pour cette raison, les plateformes de trading décentralisées les plus importantes basée sur Ethereum utilisent des carnets d'ordres *off-chain.* Les plateformes 0x, EtherDelta et IDEX utilisent chacune une approche différente qu'il est utile de comparer afin de voir les compromis devant être fait pour trouver un équilibre viable s'accordant aux attributs qualités désirés.

4.1.5.1 0x

0x utilise une implémentation hybride qui associe l'efficacité des State Channels à la transparence et la décentralisation d'un carnet d'ordres *on-chain*. D'après leur approche, les ordres, signés de manière cryptographique, sont diffusés en dehors de la blockchain, vers ce qu'ils appellent un « relayer ». Leur vision du problème est que pour qu'un marché liquide se créé au sein d'échanges décentralisés, il faut qu'il y ait un endroit public où les acheteurs et les vendeurs puissent poster leurs intentions d'achat et de vente pour qu'elles puissent être agrégées sous la forme de carnet d'ordres. Ces « relayers » remplissent ce rôle, hébergent et maintiennent des carnets d'ordres. Grâce à ces tierce parties qui permettent d'avoir une vue d'ensemble des offres et demandes sur un marché donné, les traders n'ont plus cas sélectionner les ordres qui les intéresses, les signer cryptographiquement et les envoyer dans un smart contract prévu

à cet effet (DEX dans la figure 3). Ce dernier est capable d'authentifier les signatures des deux parties, de vérifier la validité des ordres reçus, de vérifier qu'aucun des ordres n'a pas déjà été exécutés, puis transférer les fonds entre acheteur et vendeur aux taux de change spécifié (Warren, Bandeali, 2017).

Figure 3: Représentation du fonctionnement de 0x



(Warren, Bandeali, 2017, p.5)

Dans leur whitepaper, ils se réfèrent à leur mécanisme comme étant un « off-chain order relay with on-chain settlement », ce qui décris bien l'infrastructure mise en place (Warren, Bandeali, 2017, p.2). La figure 3, représente la séquence des étapes à suivre ainsi que les éléments présents dans leur protocole. Elle est décrite plus en détails la figure ci-dessous provenant de leur whitepaper :

Figure 4: Séquence des étapes du protocole 0x

1. Maker approves the decentralized exchange (DEX) contract to access their balance of Token A[2].
2. Maker creates an order to exchange Token A for Token B, specifying a desired exchange rate, expiration time (beyond which the order cannot be filled), and signs the order with their private key.
3. Maker broadcasts the order over any arbitrary communication medium.
4. Taker intercepts the order and decides that they would like to fill it.
5. Taker approves the DEX contract to access their balance of Token B.
6. Taker submits the makers signed order to the DEX contract.
7. The DEX contract authenticates makers signature, verifies that the order has not expired, verifies that the order has not already been filled, then transfers tokens between the two parties at the specified exchange rate.

(Warren, Bandeali, 2017, p.2)

Ce type d'approche hybride requiert donc d'effectuer beaucoup moins de transactions sur la blockchain et permet d'augmenter la liquidité d'un marché et la qualité de l'expérience utilisateur. En revanche, cela ne permet toujours pas d'égaler les performances des échanges centralisés, pas même de loin. Ceci est principalement dû au fait que les ordres doivent être manuellement assorti par les utilisateurs, plutôt que par un système automatisé. Aussi, cela représente des opportunités d'arbitrage contre les utilisateurs qui serait lent à annuler leurs ordres en attente. Également, l'absence d'allocation automatique des ordres ne permet pas de remplir totalement un ordre dans le cas où aucun autre ordre disponible n'aurait une quantité suffisante. Du moins, il faudrait que la partie intéressée s'occupe de trouver plusieurs ordres, à des niveaux de prix différents, correspondant à son ordre initial.

Finalement, la conception de 0x permet deux types d'ordres, les ordres étant diffusés vers les « relayers » et les ordres directes entre deux parties (sans utiliser le « relayer » comme intermédiaire). Les « relayers » se rémunèrent en facturant des frais en récompense de leurs listings d'ordres. Lorsqu'une commande est remplie, le smart contract transfert les frais que les acheteurs ou vendeurs ont inclus au moment de la transaction. Le problème de cette architecture est que les traders peuvent choisir d'uniquement consulter les carnets d'ordre maintenus par les « relayers » pour ensuite contacter directement l'émetteur d'une offre qui les intéresse. En procédant ainsi, ils peuvent échapper aux frais des « relayers ». Ce qui est avantageux pour les traders, mais très problématique pour la plateforme (Bentov et al., 2017).

4.1.5.2 EtherDelta

EtherDelta, s'inspire fortement du protocole mis en place par 0x pour son mécanisme et son architecture. L'apport majeure de cette plateforme est l'intégration de « relayers » directement à la plateforme. Contrairement à 0x qui permet à n'importe quelle tierce partie d'agir en tant que « relayer » et de toucher une commission pour sa contribution,

EtherDelta se place comme seul « relayer » sur sa plateforme. L'expérience de trading en est donc d'autant plus améliorée et agréable. Afin de passer un ordre ou d'en remplir un émis par un autre acteur, un trader va déposer une partie de ses tokens (ceux qu'il souhaite « mettre en jeu ») dans le smart contrat d'EtherDelta prévu à cet effet, sur la blockchain. Une fois le dépôt effectué, il sera en mesure de placer des ordres sur la plateforme qui les publiera sur son carnet d'ordre *off-chain*. Ce dernier consultera le montant déposé par le trader, en appelant le smart contract sur la blockchain, afin de vérifier qu'il détient suffisamment de tokens pour remplir l'ordre émit. Les autres traders verront ensuite l'ordre émis par le premier trader et pourront choisir de le remplir en signant un ordre correspondant l'envoyer vers le Smart Contract d'EtherDelta (Bentov et al., 2017).

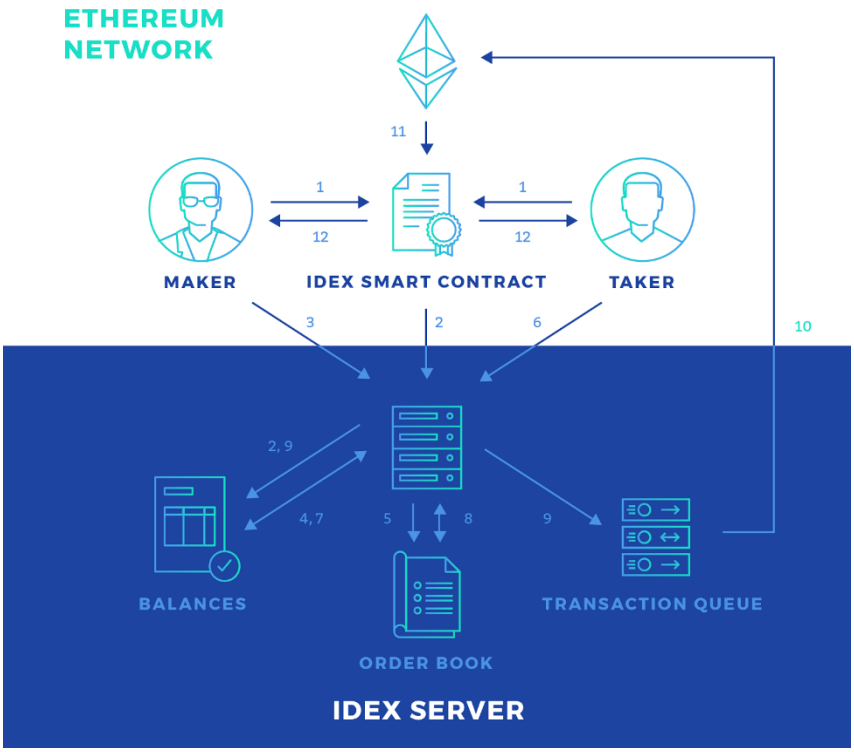
4.1.5.3 IDEX

IDEX, se base sur l'approche de EtherDelta tout en conservant l'idée d'un carnet d'ordre *off-chain* de 0x. Le processus est quasiment identique que celui de ses prédécesseurs. Afin de passer ou d'exécuter un ordre, les utilisateurs déposent des tokens de leur portefeuille Ethereum dans un smart contrat se trouvant sur la blockchain. Les traders utilisent ensuite l'interface de l'application IDEX pour passer des ordres d'achat et de vente qui seront publiés sur un carnet d'ordres *off-chain*. IDEX et EtherDelta ont des structures similaires dans la mesure où ils intègrent tous deux un carnet d'ordres *off-chain* avec un smart contract *on-cha*in utilisé pour le règlement des transactions. Cependant, IDEX ajoute deux éléments supplémentaires (Lin, 2019)

Le premier, qu'ils appellent un « transaction processing arbiter» (arbitre de traitement des transaction) qui permet de gérer l'ordre de publication (sur la blockchain) des transactions en attente de manière à ce qu'elles soient confirmées dans le bon ordre (Aurora Labs, 2019, p.2) Par conséquent, à mesure que les utilisateurs font des échanges, l'interface de l'application IDEX met à jour les montants détenus par les traders en temps réel. Ceci, de manière asynchrone avec le *mining* des transactions sur la blockchain qui a généralement lieu avec une certaine latence. En contrôlant l'ordre des transactions, IDEX sépare l'exécution des ordres du règlement des transaction, améliorant ainsi l'expérience utilisateur. Aussi, il est important de noter que seule la plateforme IDEX peut envoyer des transactions signées au Smart Contract. En empêchant les utilisateurs de soumettre des échanges au contrat, la plateforme est capable de contrôler l'ordre dans lequel les transactions sont traité sur la blockchain. Par la même occasion, les tradeurs ne sont plus en mesure d'éviter de payer les frais de commission de la plateforme en concluant des échanges hors-plateforme.

Le deuxième élément qui diffère des autres plateformes de trading décentralisé et qui permet à IDEX de se rapprocher de très près des performances des échanges centralisés est l'ajout d'un « matching engine » (algorithme ou mécanisme d'allocation des ordres). Grâce à cet apport, les traders n'ont plus à se préoccuper de trouver un ordre correspondant à leurs besoins.

Figure 5: Représentation du fonctionnement de IDEX



(Aurora Labs, 2019, p.3)

La figure 5, représente la séquence des étapes dans le protocole de IDEX. La séquence, est décrite plus en détails dans leur whitepaper et est composée des étapes décrites dans la figure 6 ci-dessous :

Figure 6: Séquence des étapes du protocole IDEX

1) The maker and taker deposit their tokens into the IDEX contract.
2) The IDEX database is updated to include the customer addresses and token balances.
3) Maker creates and submits a signed order that includes the relevant trade data.
4) IDEX confirms that the maker's account has sufficient funds and that the signed transaction matches what was submitted to IDEX.
5) If all checks in part 4 pass, the order is added to the orderbook.
6) The taker submits a matching order, signing a transaction with the same price as the target order and an amount less than or equal to it.
7) IDEX confirms that the maker's account has sufficient funds and that the signed transaction matches what was submitted to IDEX.
8) If all checks in part 7 pass, the trade is marked as matched and the orderbook is updated.
9) The IDEX database is updated to reflect the new balances, and both traders can continue to make new trades based these updates. Simultaneously, the signed order is added to the queue to be broadcast to the Ethereum network for processing.
10) After all dependent trades have mined, the transaction is dispatched to the blockchain.
11) The transaction is mined and the contract balances update to reflect the trade.
12) Once the transaction has mined, the maker and taker are able to withdraw their funds.

(Aurora Labs, 2019, p.3)

On remarque, bien évidemment, une forte similitude avec le fonctionnement d'EtherDelta et de 0x. Les étapes sont toutes quasiment identiques à l'exception de la numéro 7, dans laquelle l'algorithme de correspondance rentre en jeu. Également et surtout, la numéro 9 dans laquelle on peut voir le mécanisme particulier de IDEX.

4.1.5.4 Avantages d'un carnet d'ordres off-chain

Premièrement, on constate une amélioration des performances. Les carnets d'ordres *off-chain* permettent d'avoir un débit de transactions bien plus élevée que pour leurs contreparties *on-chain*. Au lieu d'attendre qu'un bloc soit « miner » et confirmé pour mettre à jour le carnet d'ordres et les comptes des utilisateurs, les services *off-chain* peuvent mettre à jour ces informations presque instantanément. (Lin, 2019)

D'autre part, on constate une amélioration des coûts. En effet, il n'est plus nécessaire de payer des frais de transaction pour la soumission, la mise à jour ou l'annulation d'un ordre. Des frais sont encourus uniquement lors du dépôt initial de tokens dans le Smart Contract, ainsi que durant le transfert initié par l'algorithme d'allocation (Aurora Labs, 2019, p.3-4)

4.1.5.5 Inconvénients d'un carnet d'ordres off-chain

On retrouve encore une fois le problème de confiance. Les utilisateurs doivent s'appuyer sur les hébergeurs (centralisés) du carnet d'ordres se trouvant en dehors de la blockchain afin de pouvoir diffuser correctement leurs ordres. Ces hôtes pourraient être défaillant à tout moment et ne plus afficher et mettre à jour les ordres avec précision, de sorte que les utilisateurs ne pourraient plus trouver de correspondance à leurs ordres. Phénomène qui peut avoir un impact conséquent sur le prix du marché (Lin, 2019). Dans

le pire des cas, ces hôtes pourraient choisir de censurer arbitrairement des ordres valides ou de manipuler le marché en affichant de manière stratégique des ordres inexacts ou périmés.

De plus, des hackers pourraient infiltrer le serveur centralisé et modifier l'interface du carnet d'ordres afin de tromper les utilisateurs et faire en sorte qu'il transfert les fonds dans les comptes des hackers. Ces risques ne sont pas uniquement théoriques, mais des attaques de ce genre se sont déjà produites. En décembre 2017, des pirates ont réussis à s'emparer du serveur DNS de EtherDelta et à servir une fausse version du site aux visiteurs (Schroeder, 2017).

Bien que les échanges décentralisés utilisent un carnet d'ordres décentralisé, ou « relayer », ils sont toujours susceptibles au « front-running » du fait que le règlement des transactions a lieu sur la blockchain. En effet, comme les *mineurs* ont le contrôle ultime sur l'ordre d'inclusion des transactions dans la blockchain, ils peuvent décider d'inclure la leur en premier et bénéficier d'un avantage de prix (ConsenSys Diligence, 2017).

4.1.6 L'algorithme d'allocation d'ordres

L'appariement d'ordres (order matching) est le processus par lequel les ordres d'achat sont jumelés à des ordres de vente comportant des conditions mutuellement acceptables. Les échanges décentralisés peuvent avoir un algorithme de correspondance automatique–comme IDEX–ou obliger les traders à identifier et à exécuter manuellement un ordre–comme EtherDelta ou 0x.

4.1.6.1 Exécution manuelle des ordres

Avec l'exécution manuelle des ordres, les traders doivent rechercher, dans le carnet d'ordres mis à disposition par la plateforme ou par les « relayers », de manière proactive un ordre correspondant à leurs besoins et leurs critères. Cette manière de procéder introduit plus de latence dans l'exécution des ordres, mais requiert généralement d'avoir moins de confiance dans une entité tierce centralisée. Les utilisateurs ont donc davantage de contrôle au détriment de la rapidité de la correspondance et de l'exécution.

4.1.6.2 Exécution automatisée des ordres

Avec un traitement automatisé des ordres, un algorithme a la responsabilité de faire correspondre les ordres comportant des conditions mutuellement acceptables. L'automatisation de ce processus réduit le temps et les efforts nécessaires pour identifier les ordres appropriés, réduisant ainsi la latence du traitement des transactions.

Cependant, cette approche nécessite que les utilisateurs fassent confiance au mécanisme de mise en correspondance pour exécuter les ordres de manière fair-play.

5. Implémentation

5.1 Concept

Une plateforme de trading se basant sur les projets réalisés permettant d'augmenter la vitesse d'exécution des ordres tout en bénéficiant des propriétés de la blockchain sous-jacente. Bien que l'externalisation du carnet d'ordre et l'utilisation de State Channel permet d'obtenir des performances intéressantes, un problème subsiste. En effet, la volatilité des crypto monnaie est telle qu'il n'est pas envisageable de les utiliser comme monnaie d'échange sur la plateforme. Pour cette raison, il est nécessaire d'avoir une monnaie virtuelle stable et échangeable contre des instruments financiers. Les traders devront donc acheter cette monnaie afin de pouvoir bénéficier de la plateforme. Chacun des contrats représentant un actif ne pourront être acheté qu'uniquement contre cette dernière. Aussi, pour que cette plateforme respecte les principes fondamentaux de la blockchain, il est impératif que le seul acteur étant autorisés à signer une transaction soit le trader. Les clés privées des traders ne devront donc jamais être stocké sur les serveurs centralisés de la plateforme. Les fonds des traders ne devront également jamais être bloqués par la plateforme. Les traders doivent pouvoir retirer leurs actifs à leur entière discrétion (sauf pour les transactions en cours requérant un dépôt).

5.1.1 Objectifs de la solution

L'objectif principale de ce travail est de démontrer qu'il est possible de réaliser une plateforme de trading basée sur la blockchain Ethereum. L'infrastructure doit permettre d'égaler les performances des échanges centralisés en utilisant des State Channels permettant d'externaliser l'allocation d'ordres. L'implémentation d'un tel system, nous permettra également de vérifier s'il est possible d'amoindrir les commissions prélevées sur chaque transaction. Une comparaison pourra alors être effectuée avec les plateformes existantes.

5.1.2 Portée de la solution

Le but de ce travail n'est pas de présenter toutes les optimisations possibles pour un tel système. En effet, chaque partie du système pourrait être modifier afin d'obtenir de meilleures performances. Rappelons que l'objectif principal de ce travail est de démontrer qu'il est possible de mettre en place un système de trading reposant sur une blockchain. Comme évoqué précédemment, ceci est possible grâce à la séparation du mécanisme lourd de système de trading–la logique de correspondance d'ordre–de la blockchain, en conservant les avantages de la blockchain sous-jacente. C'est même grâce à cette approche qu'il est possible d'optimiser l'architecture qui va être présenter. Par exemple, en dupliquant le service de traitement des ordres ou encore en augmentant

la puissance allouée aux services d'allocation des ordres (développé les approches possibles).

5.2 Spécifications fonctionnelles

Ci-dessous sont listées les fonctionnalités nécessaires permettant de réaliser une plateforme de trading décentralisée. Elles sont exprimées sous forme de « user story » afin de mieux comprendre qu'elles sont les acteurs qui rentrent en jeux et de limiter le jargon technique.

- En tant qu'entreprise/institution, je veux pouvoir enregistrer un instrument financier (étant déployé au préalable sur la blockchain) sur la plateforme.
- En tant que trader, je veux pouvoir me créer un compte de trading, afin de pouvoir acheter et vendre des titres sur la plateforme.
- En tant que trader, je veux voire la liste des instruments financier en circulation sur la plateforme.
- En tant que trader, je veux effectuer un dépôt de tokens, afin de pouvoir garantir mes transactions avec d'autres traders.
- En tant que trader, je veux placer des ordres d'achat ou de vente, afin de pouvoir échanger ou obtenir un instrument financier spécifique.

5.3 Spécifications non-fonctionnelles

Les spécifications non-fonctionnelles sont les contraintes qui doivent être respecter afin de mettre un tel système en place. Elles vont permettent de justifier le choix d'une architecture distribuée et des outils utilisé pour y parvenir. Elles sont listées ci-dessous :

- Une transaction doit être effectué en dix secondes ou moins.
- Les frais de transactions doivent être moins élevés que les plateformes de trading centralisées.
- Les fonds ne doivent pas être bloqué par la plateforme.
- La monnaie servant à échanger des instruments financiers ne doit pas être sujette aux volatilités extrêmes des crypto monnaies (doit être stable).
- La plateforme ne doit pas connaitre les clés privées ou les informations personnelle des traders.

5.4 Spécifications techniques

Dans cette section sont listées tous les outils, librairies et frameworks utilisés pour mettre en place l'échange décentralisé

5.4.1 React.js

React est une librairie JavaScript permettant de créer des interfaces utilisateurs. Cette librairie peut être utilisée comme base dans le développement d'applications web ou mobiles (voir l'annexe 5 pour le lien vers le site officiel).

5.4.2 Solidty

Solidity est un langage haut-niveau, orienté objet dédié à l'implémentation de smart contracts (voir l'annexe 5 pour le lien vers le site officiel).

5.4.3 Ganache

Une blockchain Ethereum personnelle (hébergée en local) utilisée pour exécuter des tests, exécuter des commandes et inspecter l'état tout en contrôlant le fonctionnement de la blockchain (voir l'annexe 5 pour le lien vers le site officiel).

5.4.4 Web3.js

Web3.js est une collection de librairies qui permettent d'interagir avec une blockchain Ethereum locale (par exemple Ganache) ou distant (par exemple Ropsten, le réseau de test de la blockchain Ethereum), en utilisant une connexion HTTP ou IPC (voir l'annexe 5 pour le lien vers le site officiel).

5.4.5 Express.js

Express, est un framework pour Node.js. Il est conçu pour créer des applications web et des API (notamment des API REST) (voir l'annexe 5 pour le lien vers le site officiel).

5.4.6 OpenZeppelin

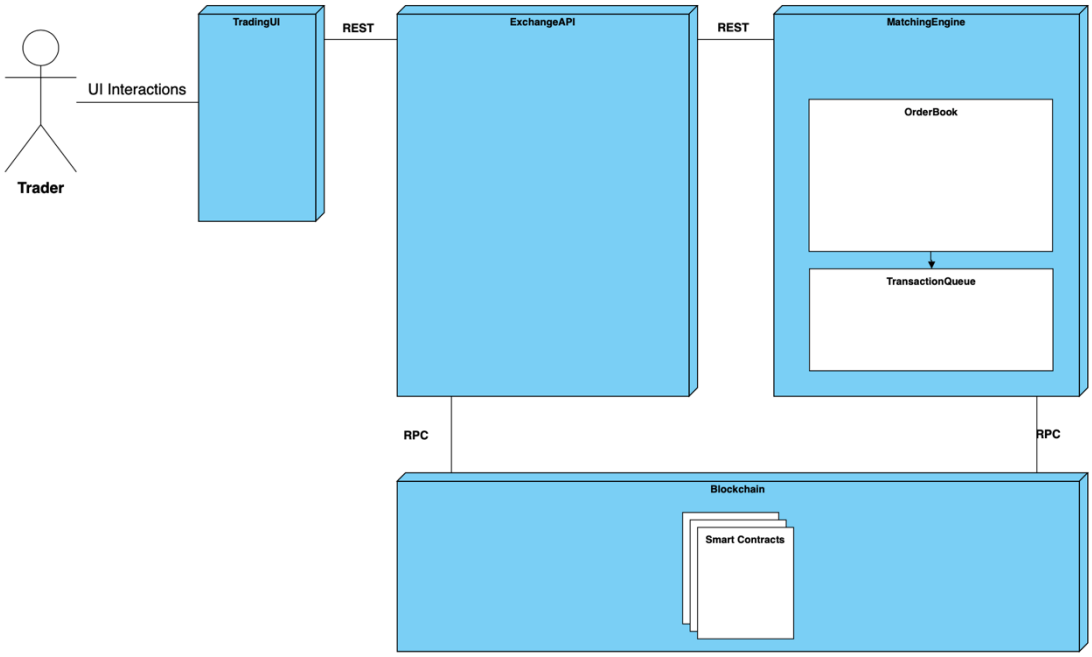
Une librairie de Smart Contracts modulaires, réutilisables et sécurisés pour le réseau Ethereum, écrite en Solidity (voir l'annexe 5 pour le lien vers le site officiel).

5.5 Implémentation des use-cases

Dans cette section, nous allons décrire l'approche d'ingénierie choisi pour mettre en place la plateforme de trading basée sur la blockchain Ethereum. Cette partie du travail, représente le produit des recherches et des préparations effectuées.

5.5.1 Architecture de la solution

Figure 7: Architecture globale de l'infrastructure



Fait à l'aide du logiciel Visual Paradigm

Le diagramme ci-dessus représente les éléments majeurs qui compose le système réalisé. Les composants principaux sont décrits dans les sections qui suivent. Leur fonctionnement et leurs interactions sont expliquées plus en profondeur dans la partie concernant l'implémentation des spécifications fonctionnelles.

5.5.1.1 Trading UI

Ce service est tout simplement l'interface graphique par lequel les traders vont pouvoir s'inscrire et placer des ordres sur l'échange décentralisé. Il n'y a absolument aucune logique métier ou aucune interaction directe avec la blockchain depuis ce composant.

Code source : https://bit.ly/dextr_trading-ui

5.5.1.2 ExchangeAPI

C'est dans ce service que se trouve toute la gestion des comptes de trading, du placement d'ordre et des interactions principales avec la blockchain.

Code source : https://bit.ly/dextr_exchange-api

5.5.1.3 MatchingEngine

Ce service est uniquement responsable du traitement des ordres. Il s'occupe de l'allocation des ordres et de la publication des ordres sur la blockchain. Le système a été découpé de cette manière, afin de garantir une exécution des ordres en moins de dix secondes, comme stipulé dans les spécifications non-fonctionnelles. L'allocation d'ordres étant une opération intensive qui requiert d'être lancée constamment, elle se doit d'être isolée afin de ne pas impacter ou retarder le fonctionnement des autres opérations critiques, tel que la gestion des balances des utilisateurs.

Code source : https://bit.ly/dextr_matching-engine

5.5.1.4 Order Book

Dans un système de trading–qu'il soit centralisé ou bien distribué–un des composants les plus important est le Limit Order Book (LOB). Étant donné que la structure de données choisie pour représenter le LOB sera la principale source d'informations sur le marché pour les clients (modèles de trading ou trader directement), il est important de la rendre à la fois totalement correcte et aussi rapide que possible.

Il existe trois opérations principales qu'un LOB doit implémenter: ajouter, annuler et exécuter. L'objectif est de mettre en œuvre ces opérations en un temps O(1) tout en permettant au modèle de trading de poser efficacement des questions telles que «quelle est le meilleur prix d'achat ou de vente ?», «Quel est le volume à un certain prix ?» ou "quel est le prix d'achat actuelle ?".

Une opération d'ajout place un ordre à la fin d'une liste d'ordres à exécuter à un prix limite particulier. Une opération d'annulation supprime un ordre de n'importe où dans le livre d'ordre. Une exécution supprime un ordre de l'intérieur du livre (l'intérieur du livre est défini comme l'ordre d'achat le plus ancien au prix d'achat le plus élevé et l'ordre de vente le plus ancien au prix de vente le plus bas) (Halfelf, 2019).

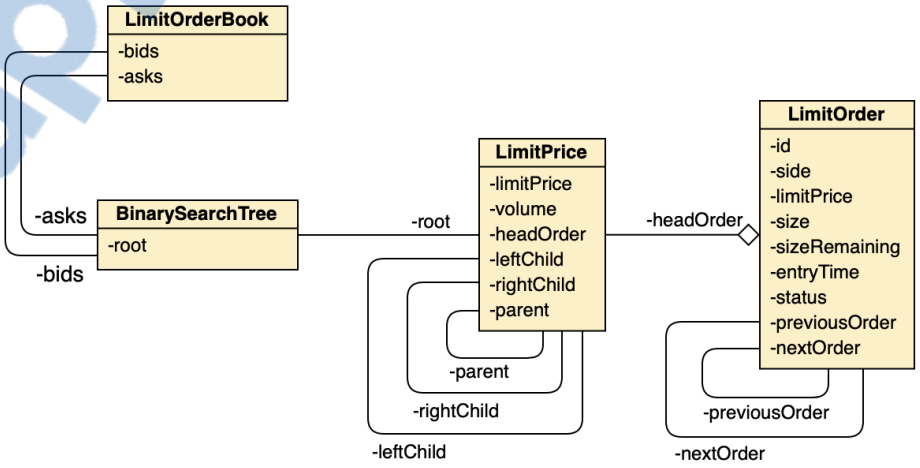
Un carnet d'ordres limites, comme mentionné ci-dessus, est représenter par le volume présent à chaque niveau de prix. Plus précisément, chaque niveau de prix contient tous les ordres qui ont été émis à ce niveau de prix respectif et qui n'ont pas encore été exécutés ou bien annuler dans leur entièreté. Une caractéristique essentielle des LOBs

à prendre en compte est le fait qu'une majorité substantielle des niveaux de prix sont vides et ne contiennent aucun volume. Cette caractéristique importante mène à la nécessité d'utiliser des nœuds pour représenter chaque niveau de prix, tout en trouvant un moyen de les connecter les uns aux autres, sans ne jamais devoir considérer les niveaux de prix inutilisés. Ce dernier facteur permet tout de suite d'éliminer l'idée d'utiliser une simple liste pour représenter le LOB.

Afin de pouvoir mettre en place ces nœuds, il est tout d'abord important de comprendre quelles informations ils vont devoir contenir. Premièrement, chaque nœud (niveau de prix) contient la liste des tous les ordres encore ouverts à ce même niveau de prix. Deuxièmement, un autre attribut non-négligeable doit être conservé dans le nœud : le volume agrégé des ordres ouvert à ce niveau de prix. Bien que cette valeur puisse être calculé directement en parcourant la liste des ordres, il est plus judicieux de la maintenir à jour séparément pour ne pas devoir la recalculé lors de chaque lecture et économiser de la puissance de calcul. Finalement, chaque nœud doit contenir des liens le connectant aux autres niveaux de prix (aux autres nœuds) (Schroeter, 2014).

L'idée est d'utiliser un arbre binaire de recherche (Binary Search Tree) dans lequel chaque nœud représenterait un niveau de prix. Les nœuds seraient naturellement ordonnés grâce aux propriétés d'une telle structure. Chaque côté du LOB, les ordres d'achats et les ordres de ventes, doivent se trouver dans des arbres séparés de sorte que l'intérieur du livre corresponde respectivement à la fin et au début des arbres. Chaque nœud serait également une liste doublement liée (Doubly Linked List) d'ordres ouvert. Le diagramme de classe ci-dessous représente les éléments qui rentre en jeu dans l'implémentation choisie.

Figure 8: Diagramme de classe du Limit Order Book



Fait avec l'outil Mermaid Live Editor

Avec la structure adoptée, la complexité algorithmique des trois opérations principales est de O(n) dans le pire des cas et de O(h), ou « h » est la hauteur de l'arbre, dans la majorité des cas.

Cependant, la complexité pourrait être réduite en utilisant une stratégie pour maintenir l'arbre équilibré (AVL ou Height Balanced Tree). En effet, dans un arbre auto-balancé la complexité pour les trois opérations est de $O(\log_2 n)$ dans le pire des cas. D'autant plus que la nature des marchés est telle que les ordres seront supprimés d'un côté de l'arbre au fur et à mesure qu'ils seront ajoutés de l'autre (Tuteja, [sans date]).

D'autre part, chacun de ces ordres étant associés à un numéro d'identification unique, il serait très avantageux de les suivre dans une table de hachage. Chaque ordre et chaque nœud pour un niveau de prix serait également une entrée dans une table de hachage. Cette approche permettrais d'améliorer la vitesse d'exécution des opérations principales et d'atteindre les performances escomptées : une complexité de O(1) (Halflef, 2019).

Bien qu'il aurait été intéressant d'approfondir ces deux possibilités d'optimisation, une structure simple composée de deux arbres binaires non-balancés semble amplement suffisante pour l'objectif de ce travail et sera donc celle implémentée.

Code source : https://bit.ly/dextr_limit-order-book

5.5.1.5 Remarques

La logique de placement d'ordre nécessitant une interaction avec la blockchain aurait également pu se trouver sur le client TradingUI. Cependant, elle a été placée du coté serveur pour deux raisons. La première étant la séparation des responsabilités, il va de soi qu'un client (une vue) ne devrait pas contenir de logique métier. La seconde raison est que nous souhaitons éventuellement pouvoir mettre en place un système (un bot) permettant d'envoyer des ordres d'achat et de vente afin de pouvoir tester la charge du système. En déplaçant la logique de traitement d'ordre du coté serveur, il suffit simplement d'envoyer des requêtes sur le serveur depuis le bot, sans avoir à implémenter (ou à connaitre l'implémentation) de la logique de traitement d'ordre.

Dans les services ExchangeAPI et MatchingEngine la logique métier est fortement couplée à la logique des contrôleurs. Ce travail n'ayant pas pour but de démontrer la meilleure architecture possible, tant en termes de performance qu'en terme de robustesse, la logique métier restera couplé de la sorte. Dans un système plus avancé, on pourrait envisager l'utilisation d'un objet « Orchestrateur » ou bien d'utiliser le pattern « Publish/Subscribe » pour que lorsqu'un ordre soit placé, les services ou composants

ayant besoin de l'information puisse s'abonner et la recevoir sans que l'émetteur de l'information ne connaisse les abonnées.

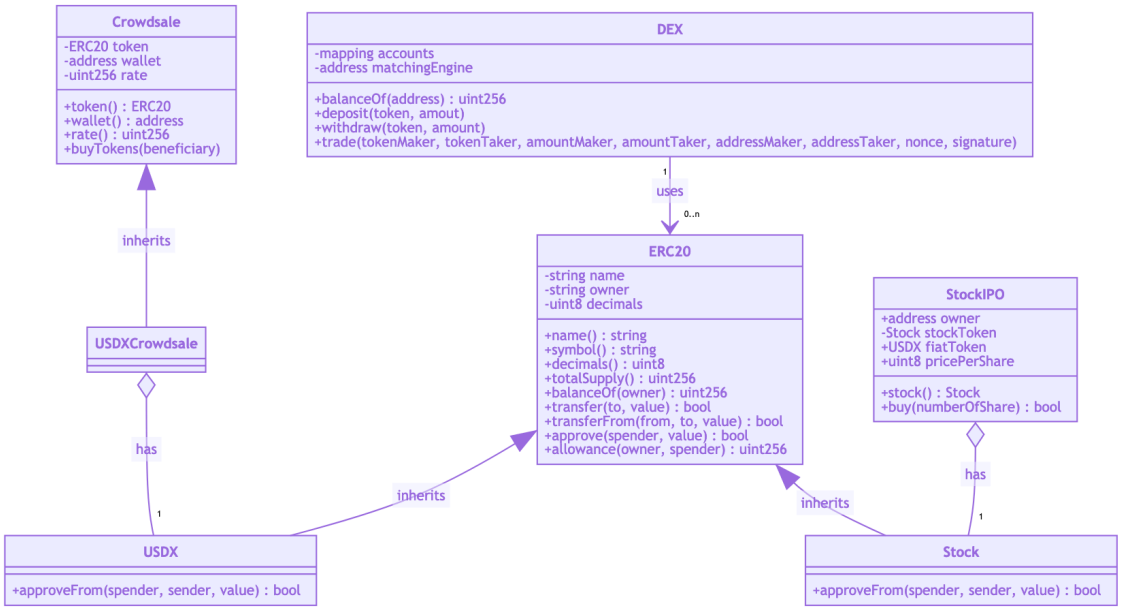
Le processus de mise en circulation initiale (IPO, crowdsale, etc) n'est pas géré par ce système. Ce système sert de plateforme d'échange et pas de plateforme de création/mise en circulation de token/instruments financiers.

Afin de garantir que le code réponde aux critères d'acceptations et de pouvoir ajouter ou modifier les fonctionnalités sans craintes, des tests d'acceptation sont rédiger avant le développement d'une fonctionnalité (voire l'annexe 4 pour plus d'information).

5.5.2 Architecture des Smart Contracts

Pour mettre en place la plateforme de trading décentralisé, des Smart Contracts (SC) ont dû être déployés sur la blockchain Ethereum. Leurs rôles et leurs implémentations sont détaillés dans les sections qui suivent. Un diagramme représentant l'architecture globale des SC est exposé dans la figure ci-dessous. Le code source est disponible à cette adresse : https://github.com/samyabouseda/dex-contracts.

Figure 9: Architecture des Smart Contracts



Fait avec l'outil Mermaid Live Editor

5.5.2.1 ERC20

ERC20 est une norme qui permet la mise en œuvre d'une API standard pour les token fongible dans les Smart Contracts (Vogelsteller, Buterin, 2015). Elle permet d'assurer que certaines fonctionnalités de base soient présentes pour pouvoir transférer et approuver des tokens afin qu'ils puissent être dépensés par un tiers sur la même blockchain (en l'occurrence Ethereum).

Dans le cadre de ce travail, les implémentations de ce standard de la librairie OpenZeppelin ont été utilisés pour réaliser les autres Smart Contracts.

5.5.2.2 USDX

Comme évoqué précédemment, un des problèmes principaux de l'utilisation d'une crypto-monnaie comme monnaie principale d'échange sur une plateforme de trading est sa forte volatilité. Il n'est donc pas envisageable d'utiliser la crypto-monnaie native (ETH) de la blockchain Ethereum à cette fin. Il est alors nécessaire d'avoir un token dont le prix est stable. Ce type de token, appelé « Stablecoin » obtient sa stabilité grâce à un ou plusieurs actifs garantissant sa valeur à tout moment (Stablecoin Wikipedia, 2018). Il existe plusieurs approches afin d'y parvenir ; les deux plus communes sont listées ci-dessous.

5.5.2.2.1 Stablecoin garanti par une commodité (Commodity-backed)

Ce type de Stablecoin étant garanti par des commodités telles que les métaux précieux (or, argent, etc.), ils sont beaucoup moins susceptibles de subir une inflation que les token dont le cours est uniquement régulé par un algorithme se basant sur l'offre et la demande. Il bénéficie alors d'une garanti semblable à la monnaie fiduciaire que nous utilisons tous les jour. Une des caractéristiques essentielles à ce type de Stablecoin est que la quantité de commodité utilisée pour soutenir le token doit refléter l'offre en circulation de ce dernier.

Un exemple d'un tel token est le DGX (Digix Gold Tokens) qui représente un gramme d'or par DGX mis en circulation (Digix, [sans date]).

5.5.2.2.2 Stablecoin garanti par une monnaie (Fiat-backed)

La valeur des Stablecoin de ce type est basée sur la valeur de la devise qui sert de garanti. La monnaie fiduciaire en question doit être détenue par une entité financière réglementée. Dans ce contexte, la confiance dans le dépositaire de l'actif est cruciale pour la stabilité du prix du token. Tout comme pour les token garantis par des commodités, la quantité utilisée pour soutenir le token doit refléter l'offre en circulation.

Un exemple d'un tel token est le USDT (Tether) qui représente un dollar américain par token mis en circulation. Le cours de ce token est également toujours fixé sur celui de l'actif sous-jacent (en l'occurrence 1 USDT = 1 USD).

Pour que l'échange décentralisé puisse garantir une certaine stabilité de ses actifs, il est alors crucial d'avoir une monnaie d'échange qui est, elle aussi, stable. L'utilisation d'un Stablecoin est donc tout à fait propice à notre cas de figure. En revanche, la création d'un tel token sort du cadre de ce travail. Aussi, l'utilisation d'un token de ce type déjà

existant (par exemple le Tether) est compliqué et couteux à mettre en place dans un environnement de développement et n'est pas d'une importance majeure pour démontrer la preuve de concept de ce projet.

Un Smart Contract nommé « USDX » a donc été mis en place pour ce travail. Il représente le dollar américain. Il répond au standard ERC20 et hérite de toutes les fonctionnalités qui permettent son échange entre plusieurs acteurs. Il peut être acheter avec de l'Ether à un taux fixe et échangé contre des token de type stock (détaillés dans les sections suivantes). Il sert à représenter un Stablecoin et démontrer comment un token de ce type pourrait être utiliser dans ce système.

Il est important de noter que les tokens USDX répondent au standard ERC20, mais qu'ils sont également « mintable » et non « minable ». La différence entre ces deux termes est expliquée ci-dessous.

Les tokens mintables sont des token compatibles ERC20 avec une fonctionnalité supplémentaire : de nouveaux tokens peuvent être créés à tout moment et ajoutés au montant totale en circulation. Les tokens ERC20 standard n'ont pas cette fonctionnalité, ce qui en fait des tokens avec une offre fixe.

5.5.2.3 USDXCrowdsale

Les tokens mintables sont souvent utilisés en combinaison avec des Smart Contract de mise en circulation, qui sont appelées Crowdsales. Ces contrats sont utilisés pour créer une ICO (Initial Coin Offering) où les tokens ERC20 sont le plus souvent vendus contre de l'Ether (Merkle Blue, [sans date]). Il y a toute une multitude de manière de mettre des tokens en circulation. Les trois plus fréquentes sont les suivantes.

Une Crowdsale simple : Le Smart Contract possède des tokens et les transfère simplement de son propre portefeuille aux utilisateurs qui les achètent.

Une MintedCrowdsale : Le Smart Contract va « minter » les tokens lorsqu'un achat est effectué (par un transfert d'Ether de l'acheteur) et va les attribués à ce dernier.

Une AllowanceCrowdsale : Le Smart Contract reçoit un droit de transfert sur un autre portefeuille qui possède déjà les tokens pouvant être vendu lors de la mise en circulation.

Pour ce travail, l'approche MintedCrowdsale a été choisi afin de pouvoir générer des tokens au fur et à mesure sans devoir se soucier d'une limite à ne pas dépasser. Les utilisateurs désireux d'obtenir des tokens USDX peuvent donc envoyer de l'Ether à un contrat nommé USDXCrowdsale et recevoir des tokens en retour.

5.5.2.4 Stock

Pour représenter une action au porteur, un Smart Contract nommé « Stock » a été créé. Il répond au standard ERC20 et hérite de toutes les fonctionnalités qui permettent son échange entre plusieurs acteurs. Ces tokens sont échangeable contre des tokens de type monétaire, comme par exemple le token USDX mentionné plus haut. Ils sont semblables à des titres qu'une entreprise pourrait émettre lors de son entrée en bourse.

Il suffit à cette entreprise ou institution de déployer un contrat de type stock comportant toutes les informations relatives à l'actif en question (nom, symbole, offre initiale) sur la blockchain pour créer des actions. Par exemple, la multinationale Apple Inc. pourrait déployer un contrat nommé Apple Inc, avec pour symbole AAPL et une offre initiale de 4.6 millions. En revanche, pour mettre en circulation ces tokens, gérer la distribution et définir un prix, il est nécessaire de passer par un contrat supplémentaire.

5.5.2.5 StockICO

Le fonctionnement et le but de ce contrat sont très similaires aux contrats de type Crowdsale. Toutefois, il reste fondamentalement différant car les tokens qu'il détient ne sont pas échangeable contre de l'Ether, mais uniquement contre des tokens de type monétaire, tel que l'USDX.

C'est donc dans ce contrat qu'est définit le prix par token exprimé en unité de token monétaire. En reprenant l'exemple précèdent, nous aurions donc un contrat StockICO détenant des token AAPL, disponible au prix de 231 USDX. Les investisseurs désireux d'obtenir des parts de la société Apple Inc. peuvent donc envoyer des USDX à ce Smart Contrat et obtenir des tokens APPL en retour.

5.5.2.6 DEX (pour decentralized exchange)

Ce Smart Contrat est responsable de gérer toutes les actions relatives aux transactions sur la plateforme. Contrairement aux autres contrats, il n'y aura jamais plus d'un DEX déployé sur la blockchain. Chaque méthode étant importante et assez conséquente, elles seront détaillées ci-dessous, dans une section spécifique à chacune.

5.5.2.6.1 Constructeur

Lors du déploiement du contrat, l'adresse publique du portefeuille détenu par le matching engine sur la blockchain doit y être enregistrer. Cette étape est cruciale afin de garantir que seul matching engine soit capable d'exécuter une transaction dans ce Smart Contrat.

5.5.2.6.2 Dépôt de tokens

Lorsqu'un trader veut lancer une session de trading, il doit en premier lieu déposer un certain montant de tokens sur ce contrat qui servira à garantir ses échanges qui seront effectué hors-blockchain grâce au mécanisme de State Channel. Le processus et les choix qui ont été fait vis-à-vis de ce dernier sont discutés plus en détails dans la partie concernant l'implémentation du dépôt dans l'ensemble du système.

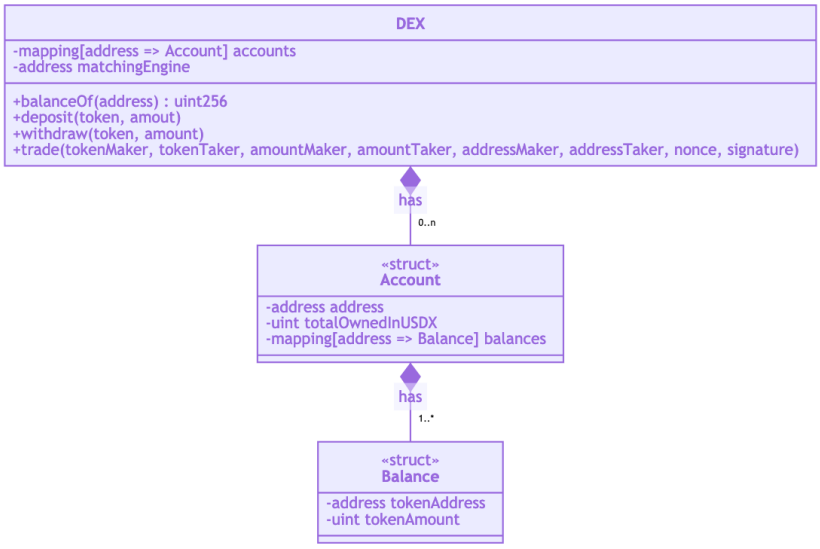
5.5.2.6.3 Retrait de tokens

La méthode propre au retrait de token, étant tout simplement l'inverse de la méthode de dépôt et n'étant pas indispensable à la réalisation de ce travail, n'a pas été mise en place. En revanche, il est important de relevé que c'est grâce à cette méthode qu'il est possible de garantir que les fonds ne seront jamais bloqués par la plateforme. Chaque trader est libre de retirer son dépôt à tout moment et le seul délai encouru sera celui de la propagation de la transaction sur la blockchain. Ce mécanisme est mis en place pour faire face au cas de figure éventuelle d'un problème de fonctionnement ou d'une attaque de la plateforme (surtout de ses composants centralisés).

5.5.2.6.4 Montant du dépôt

Il est important que chaque partie prenante puisse connaitre le montant déposé par les autres traders. Par exemple, le service ExchangeAPI doit pouvoir vérifier si un trader a suffisamment de tokens déposé pour pouvoir placer un nouvel ordre. Il a donc fallu mettre en place une méthode publique qui permet d'obtenir cette valeur pour une adresse donnée. La valeur retournée est toujours exprimée en USDX, peu importe les tokens déposés. Ce, pour des raisons d'uniformisation des State Channel crées à chaque session de trading. Ce dernier point est détaillé dans la partie implémentation concernant les dépôts de tokens.

Figure 10: Architecture de la strucutre Account dans le DEX

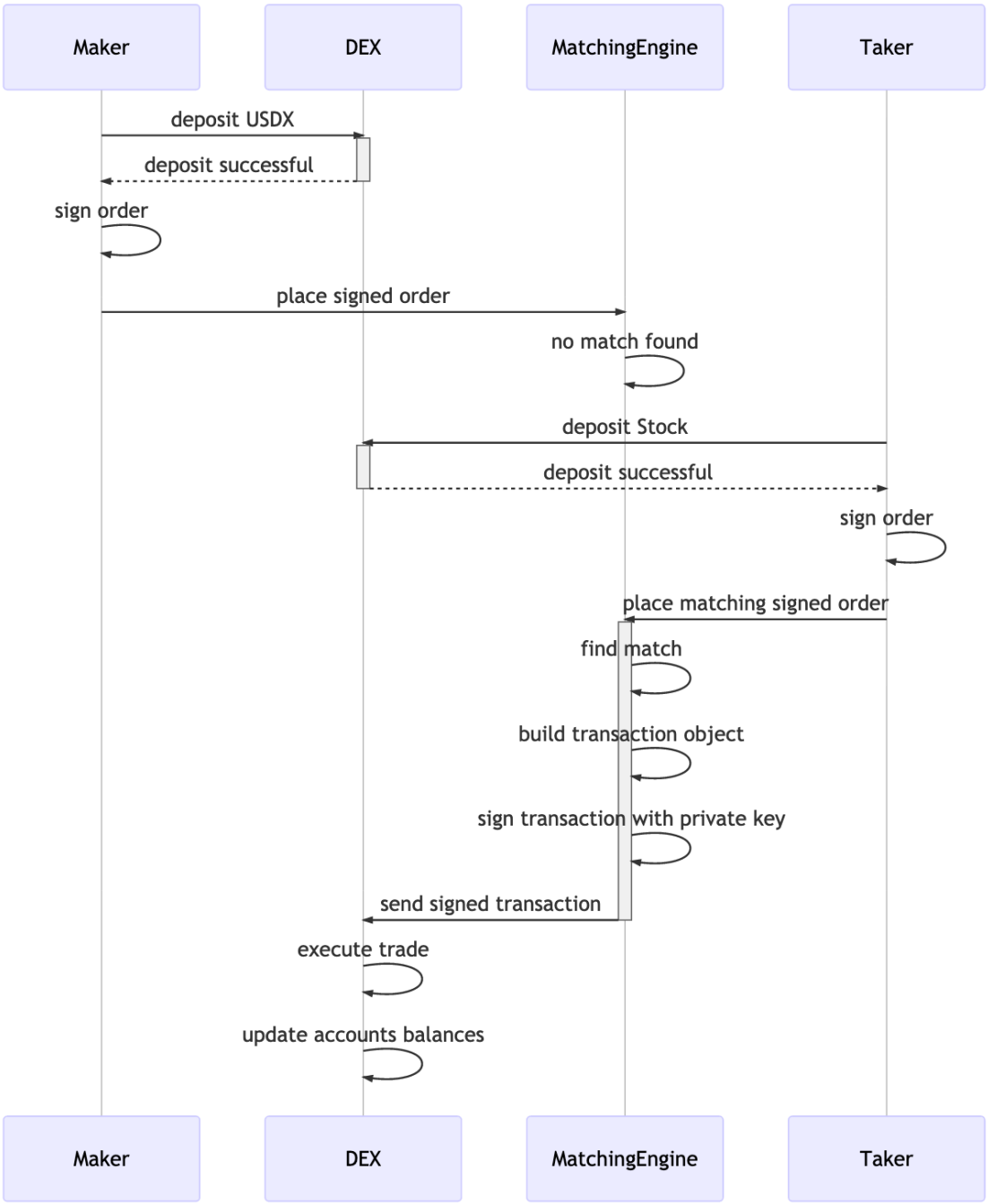


Fait avec l'outil Mermaid Live Editor

5.5.2.6.5 Transactions

Le processus d'exécution d'ordres est illustré dans le diagramme de séquence ci-dessous. Il représente les interactions entre les traders (le maker et le taker) et le contrat DEX, ainsi que les interactions entre le service MatchingEngine et le contrat DEX. Le processus de signature et de placement d'ordre hors-blockchain est décrit de manière plus approfondie dans la partie concernant l'implémentation de la gestion des ordres dans l'ensemble du système.

Figure 11: Diagramme de séquence d'un dépôt sur le DEX



Fait avec l'outil Mermaid Live Editor

5.5.3 Blockchain Interface

Afin de communiqué avec la blockchain Ethereum, il est indispensable d'utiliser une librairie : Web3.js. Cette libraire offre plusieurs packages permettant de simplifier les interactions avec la blockchain. En revanche, pour ne pas introduire un couplage trop fort avec cette librairie et pour augmenter la lisibilité et la simplicité des interactions avec la blockchain, un composant de type « Facade » (GOF, Design Pattern) a été introduit.

Ce composant « BlockchainInterface » permet d'encapsuler la complexité des interactions avec la blockchain et fournis une interface simple et unique aux clients (autres composants du système). Ce composant pourra par la suite être réutilisé dans un autre service qui nécessite également d'avoir un accès à la blockchain. Notamment un simulateur de transactions.

```
import Web3 from 'web3'
import { Transaction } from 'ethereumjs-tx'

class BlockchainInterface {
  constructor() { ... }

  async createAccount() { ... }

  async getAddressFrom(privateKey) { ... }

  async buyFiat(amount, privateKey, fiat) { ... }

  async deposit(amount, privateKey, token, dex) { ... }

  async sendTransaction(from, to, value, data, web3) { ... }

  async buildTxObject(from, to, value, data, web3) { ... }

  async signTransaction(txData, privateKey) { ... }

  async sendSignedTransaction(tx) { ... }

  // rest of implementation here...
}
```

La méthode constructor de cette classe va d'abord créer une instance de WebsocketProvider. L'objet WebSocketProvider fournit l'API pour créer et gérer une connexion WebSocket à la blockchain Ethereum, ainsi que pour envoyer et recevoir des données via cette connexion. Pour créer cette instance, il faut passer en paramètre l'url de la blockchain. Afin d'avoir une certaine modularité et de pouvoir se connecter à différentes blockchains, il est nécessaire d'utiliser des variables d'environnements. Par exemple, pour pouvoir se connecter à Ganache lors du développement et se connecter à un testnet d'Ethereum lors de la phase de déploiement dans un environnement de

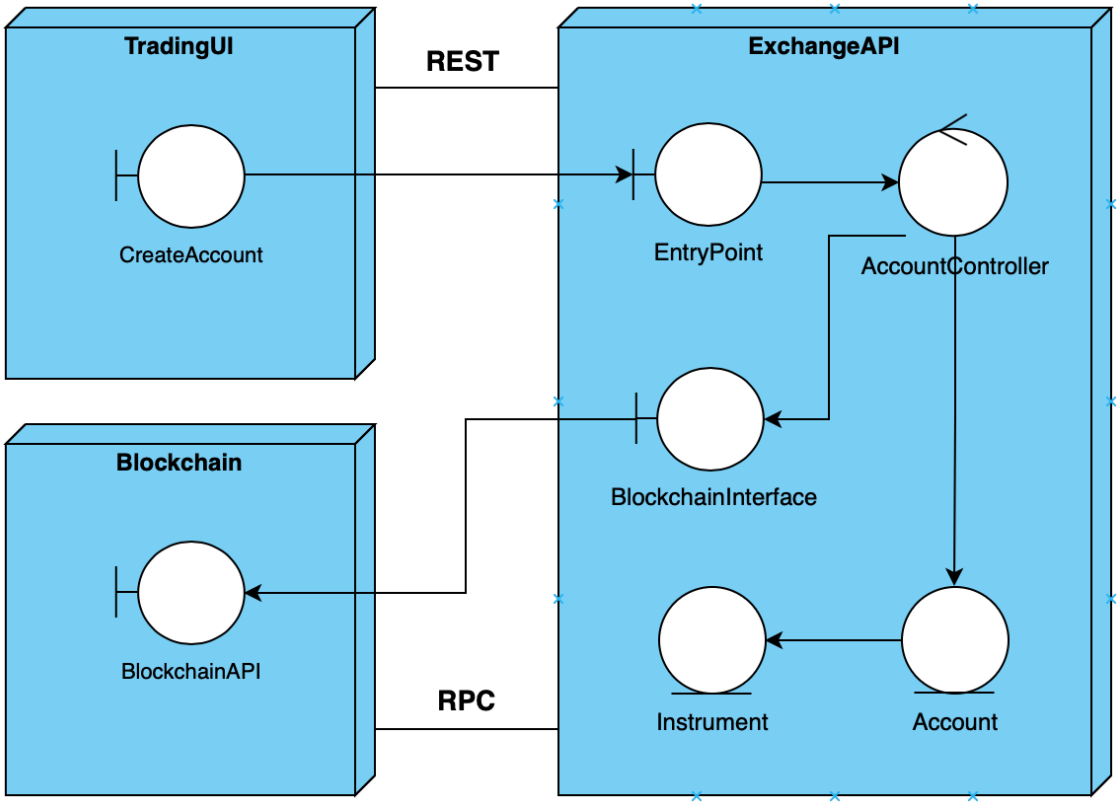
production. Ensuite, une instance de la classe Web3 est initialisée avec le WebsocketProvider et est conservée en mémoire.

Les méthodes de l'interface présentée ci-dessus sont utilisées et documentées dans les sections respectives de l'implémentation des fonctionnalités. Elles ont été nommées de manière très compréhensible afin de garantir le plus de transparence sur leur utilisation sans pour autant exposer leur implémentation. Le code source de l'interface en question est disponible à l'adresse suivante : https://bit.ly/dextr_blockchain-interface

5.5.4 Création d'un compte de trading

Afin de pouvoir accéder à la plateforme de trading pour acheter et vendre des instruments financiers, un trader a besoin d'un compte. Pour pouvoir bénéficier des fonctionnalités de la blockchain sur laquelle repose l'échange, il faut qu'il ait un compte enregistré sur la blockchain. Il faut également qu'il puisse déposer des tokens sur la plateforme afin qu'il puisse garantir ses transactions. Pour ce faire, il est indispensable de suivre et de mettre à jour un registre des transactions qu'il effectue sur la plateforme.

Figure 12: Diagramme des microservices impliqués dans la création d'un compte de trading

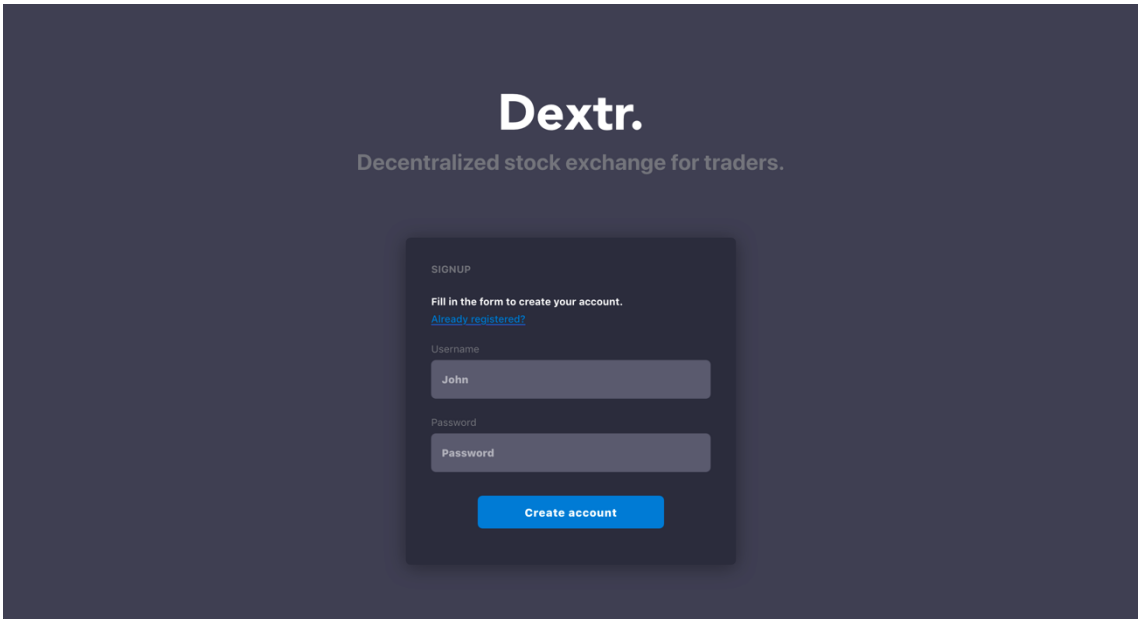


Fait à l'aide du logiciel Visual Paradigm

Le diagramme ci-dessus représentent les services et composants qui interagisse afin de réaliser la fonctionnalité de création de compte.

Pour simplifier l'expérience du trader, il peut accéder à une interface graphique (application web) lui permettant d'utiliser toutes les fonctionnalités de l'échange décentralisé. Celui-ci, est représenté sur le schéma ci-dessus par le service nommé « TradingUI ». Pour qu'il puisse se créer un compte, il doit fournir un identifiant ainsi qu'un mot de passe.

Figure 13: Interface de création de compte



Capture d'écran de la plateforme

L'application web va ensuite envoyer ses informations vers le service web nommé « ExchangeAPI » sur le schéma de l'architecture des services. Ce service, contient des informations en rapport aux comptes de trading. Il sert à maintenir les informations concernant les transactions propres à chaque compte de trading, en particulier les montants détenus en temps réel par les traders.

Pour réaliser cette fonctionnalité, le model de donné comporte deux entités. Premièrement, l'entité « User » qui va contenir les informations relatives aux comptes de trading :

```
User Model (represented as MongoDB document)

{
  _id: <ObjectId>,
  username: <String>,
  address: <String>,
  balances: [
    {
      instrumentId: <ObjectId>,
      amount: <Number>
    }
  ]
}
```

Ce model comporte plusieurs attributs qui sont expliqués dans le tableau ci-dessous :

Table 1: Attributs du modèle User

Attribut	Description	Type
_id	Le numéro d'identification unique du trader (auto-générer par Mongodb).	ObjectId
username	L'identifiant du trader	String
address	L'adresse du compte du trader sur la blockchain.	String
balances	Une liste qui contient le montant de chaque instrument détenu par un trader.	Array
instrumentId	Le numéro d'identification unique pointant vers un instrument enregistré.	ObjectId
amount	Le nombre total de titres détenu d'instrument par le trader.	Number

Les balances d'un trader sont enregistrées directement dans le document de son compte et pas dans une table à part (table d'association) pour des raisons de simplicité. Dans la version actuelle du système, il n'existe pas beaucoup d'instruments sur la plateforme. Il est donc peu probable que le nombre d'instruments détenus par un seul trader dépasse la taille maximum autorisé d'un fichier dans MongoDB.

Ensuite, l'entité « Instrument » qui va contenir les informations relatives aux instruments financier disponible sur la plateforme. Cette dernière est décrite en détails dans la section relative à l'enregistrement d'un instrument financier.

Interface blockchain

Afin de mettre en place la création de compte, il faut générer un compte à l'aide d'une des méthodes de Web3. Comme évoqué précédemment, pour ne pas introduire de dépendances trop fortes à cette librairie, une interface ont été mis en place et c'est avec lui que les composants du système vont dialoguer. Pour ce cas d'utilisation, les composants interagiront avec la méthode createAccount. Pour plus d'info sur les méthodes dipsonibles, se référer à la section Blockchain Interface (plus haut).

5.5.4.1 Remarques

Les clés publiques et privées ne sont jamais enregistrées sur le serveur. Après leur génération, elles sont directement envoyées au client. Le client ne peut les consultés qu'une seule fois. Il est donc important que l'utilisateur du client les conservent précieusement. Pour simplifier l'utilisation, l'utilisateur pourrait avoir la possibilité de télécharger un fichier contenant les clés publique et privée. Cependant, cette fonctionnalité ne sera pas implémentée car elle n'apporte pas de valeur à la version minimal du système. Le fait que les clés privées ne soient jamais stockées sur le serveur permet de suivre les principes fondamentaux de la blockchain en garantissant ne pas avoir de point de faille centralisée. De cette manière, même lors d'une éventuelle corruption du système centralisé ExchangeAPI aucune donnée critique ne pourrait être obtenues.

La logique de création de compte sur la blockchain aurait également pu se trouver sur le client TradingUI. Cependant, elle a été placée du coté serveur pour deux raisons. La première étant la séparation des responsabilités. Il va de soi qu'un client (une vue) ne devrait pas contenir de logique métier. La deuxième raison est que nous souhaitons par la suite pouvoir mettre en place un système (un bot) permettant d'envoyer des ordres d'achat et de vente afin de pouvoir tester la charge du système. En déplaçant la logique d'interaction avec la blockchain du coté serveur, il suffit simplement d'envoyer des requêtes sur le serveur depuis le bot, sans avoir à implémenter (ou à connaitre l'implémentation) de la logique de traitement d'ordre.

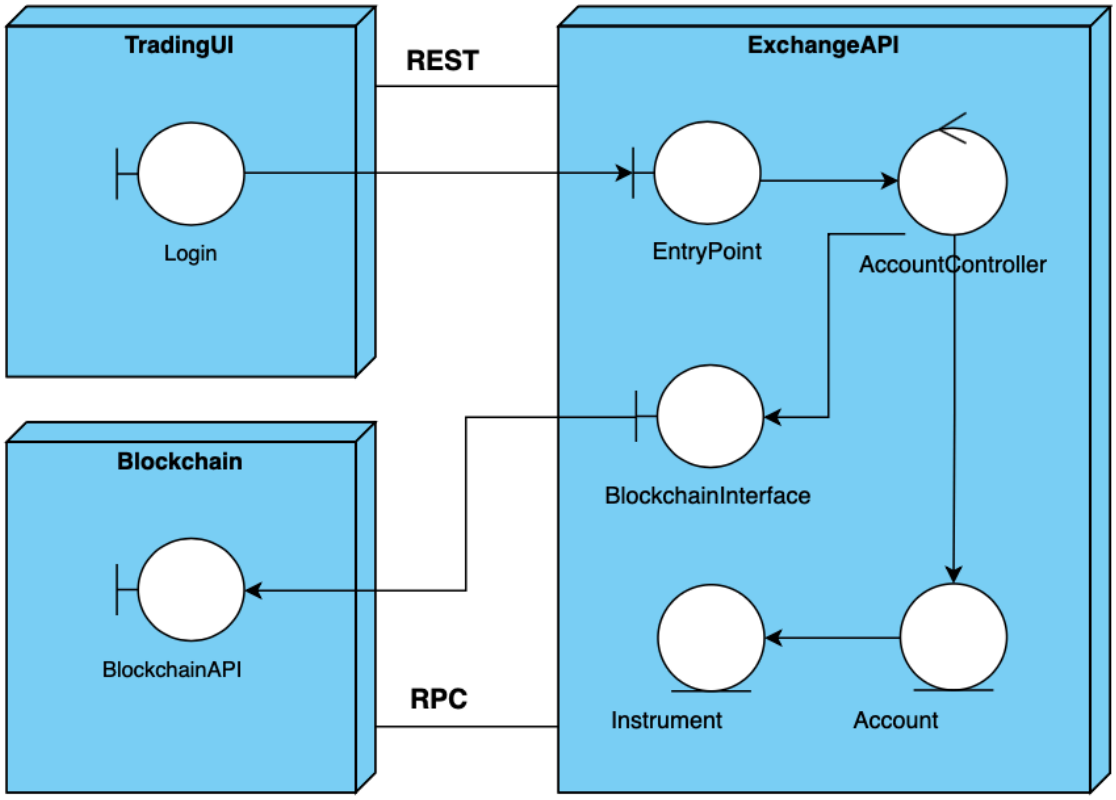
Nous pourrions nous passer des numéros d'identification unique auto-générer par MongoDB et utiliser les adresses de blockchain. Elles sont également uniques et permettrait d'arriver aux mêmes fins. En revanche, en supposant que l'on souhaite par

la suite donner la possibilité aux utilisateurs d'avoir plusieurs prote-monnaie (wallet) lié à leur compte ou bien de pouvoir changer de porte-monnaie, il est important d'avoir une certaine flexibilité et de ne pas coupler trop fortement une adresse blockchain à un compte de trading. Les ids auto-généré par MongoDB ont donc été conservés afin de bénéficier de cette modularité.

5.5.5 Login

Un trader doit pouvoir accéder à la plateforme de trading grâce aux identifiants qu'il aura préalablement créé grâce au formulaire de création de compte. Une fois enregistrer, il a accès à son tableau de bord et aux fonctionnalités d'achat et de vente de titres. Seuls les traders enregistrés peuvent accéder au système de trading.

Figure 14: Diagramme des microservices impliqués dans la connexion à la plateform



Fait à l'aide du logiciel Visual Paradigm

Dans un système standard, les utilisateurs s'authentifient en général grâce à un nom de compte unique et un mot de passe (MDP). Ces deux informations sont stockées dans une base de données. Pour des raisons de sécurité le MDP et normalement conservé sous forme de hash. Lorsqu'un utilisateur s'identifie, son MDP envoyés au serveur qui va calculer un hash à partir de ce dernier. Si le hash correspond à celui se trouvant dans la BDD, alors cela signifie qu'il est correct et l'utilisateur est authentifié. De cette manière,

même si des pirates parviennent à obtenir les mots de passe des utilisateurs, il ne pourrait rien en faire (ne peuvent pas trouver le MDP depuis le hash) grâce aux propriétés unidirectionnel des hashs.

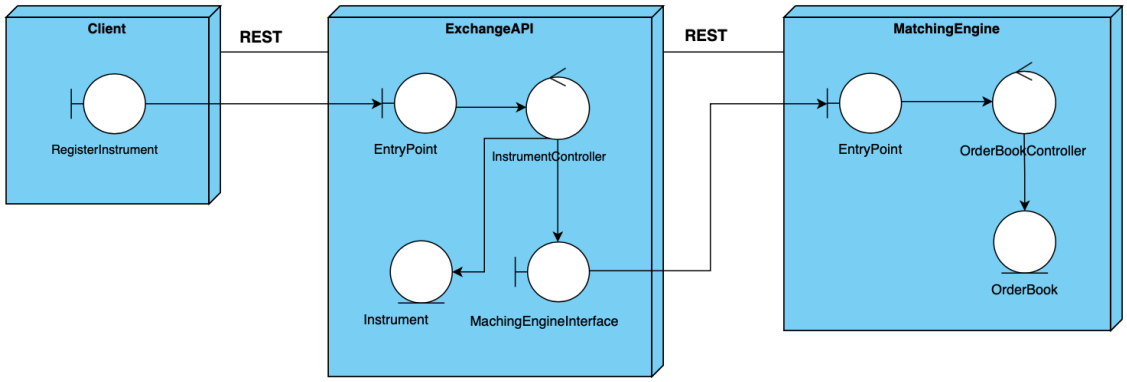
Pour suivre la philosophie de la blockchain, un utilisateur doit être le seul à pouvoir géré et autoriser ses transactions. Aussi, le caractère distribué de la blockchain permet de ne pas avoir un point de faille centralisé. Sa clé privée ne doit donc jamais être stocké sur le serveur. Il est alors important d'adopter une approche qui permette de conserver ses valeurs et de maximiser la sécurité tout en ayant une plateforme ergonomique. Il n'est donc pas envisageable de conserver les clés privées des utilisateurs dans la base de données. Pour cette raison, chaque utilisateur est obligé de conserver sa clé privée, ainsi que sont MDP, en lieu sûr.

Ce travail ne portant pas sur la gestion de comptes sécurisés sur un serveur, ni sur l'optimisation de l'expérience d'utilisation d'une telle plateforme. La gestion des MDPs et des clés privées a été simplifier afin de pouvoir nous concentrer sur d'autres aspects plus importants. Pour s'identifier, un utilisateur doit rentrer son identifiant, ainsi que sa clé privée qui sera envoyée au serveur. Grâce aux propriétés des clés privés de la blockchain, le serveur va dériver la clé publique et l'adresse correspondante à cette clé privée (Tayvano, 2016). Puis, il va s'assurer qu'un utilisateur ayant cette adresse soit bien enregistrer dans le système. Bien que cette dernière ne soit pas stockée, il n'est tout de même pas judicieux de l'envoyer sur le réseau sans être protégée d'une certaine manière.

5.5.6 Enregistrement d'un instrument financier

Pour que les traders puissent acheter et vendre des titres, il est indispensable pour leurs émetteurs de pouvoir les enregistrer sur la plateforme. Un instrument financier est représenté sur la blockchain Ethereum par un smart contract de type « Stock ». Ces contrats sont mis en circulation par les détenteurs initiaux de manière indépendante de la plateforme de trading.

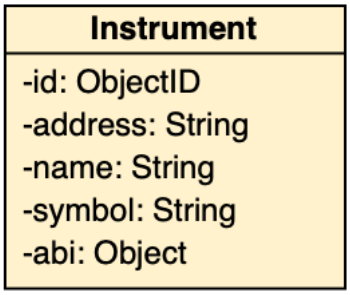
Figure 15: Diagramme des microservices impliqués dans l'enregistrement d'un instrument financier



Fait à l'aide du logiciel Visual Paradigm

Chaque instrument doit être enregistrer sur deux services : ExchangeAPI et MatchingEngine. Sur le premier, sont enregistrés les informations qui permettent d'accéder au contrat sur la blockchain et de simplifier l'affichage des informations sur l'interface (notamment ne pas devoir acceder à la blockchain simplement pour obtenir le symbole ou le nom d'un contrat).

Figure 16: Schéma UML d'un instrument sur ExchangeAPI



Fait à l'aide de l'outil Mermaid Live Editor

Ce model comporte plusieurs attributs qui sont expliqués dans le tableau ci-dessous :

Table 2: Attributs du modèle Instrument

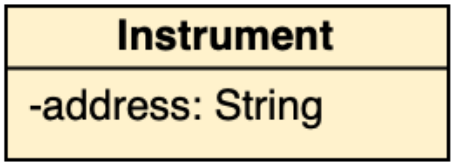
Attribut	Description	Type
id	Le numéro d'identification unique de l'instrument (auto-générer par Mongodb).	ObjectId
address	L'adresse du Smart Contract sur la blockchain.	String
name	Le nom de l'instrument financier.	String
symbol	Le symbole (une suite de 3 lettres) de l'instrument financier.	Strings
abi	La représentation ABI du Smart Contract pour un instrument financier.	Object (on veut stocker l'abi dans le format json pour éviter de devoir le reconvertir à chaque utilisation).

5.5.6.1 Représentation ABI

Un ABI (application binary interface) dans le domaine informatique définit le mode d'accès aux structures de données et aux routines dans un code machine de bas niveau. Dans l'ecosystème Ethereum, c'est le moyen standard d'interagir avec les Smart Contracts. A la fois depuis l'extérieur, mais également depuis l'intérieur (interaction contract-à-contract). Les données sont encodées en fonction de leur type (comme décrit dans la spécification de Solidity). Une ABI est nécessaire pour permettre de spécifier la fonction à appeler dans le contrat et d'obtenir la garantie que la fonction renverra les données dans le format attendu (Solidity, 2019b).

Sur le second service, une seule information relative au contrat est conservée : son adresse. Cette dernière sert de clé pour identifier un LOB.

Figure 17: Schéma UML d'un instrument sur MatchingEngine



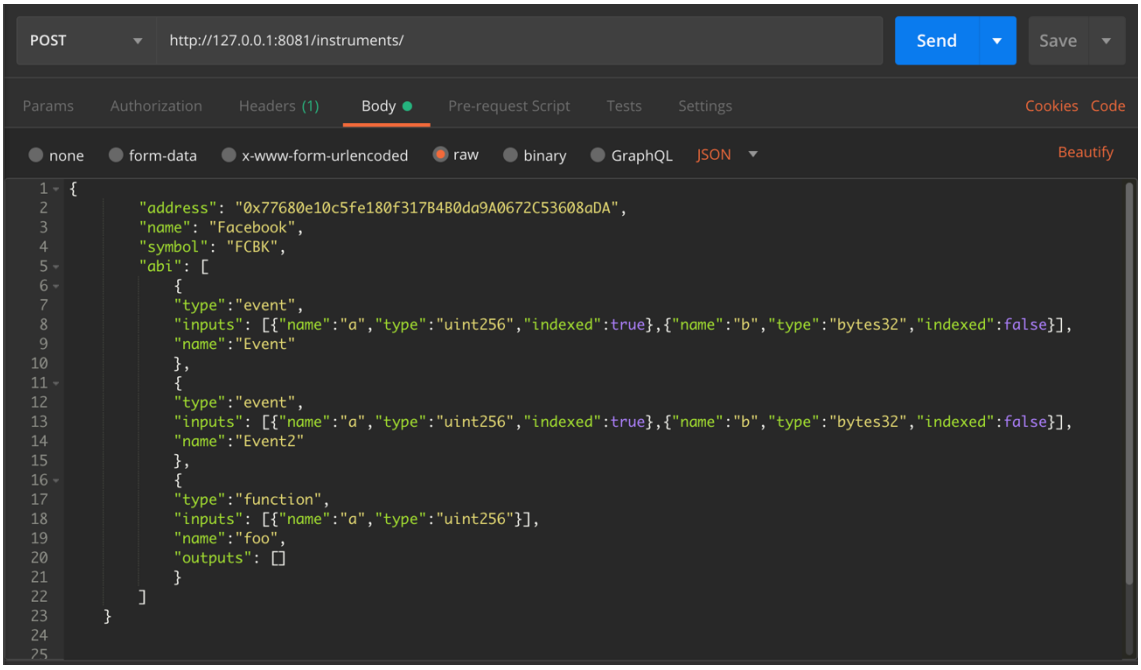
Fait à l'aide de l'outil Mermaid Live Editor

L'API « ExchangeAPI » permet donc l'enregistrement d'un instrument financier. Étant donné que les titres sont enregistrés par des institutions financières ou des entreprises,

les traders ne doivent pas avoir accès à cette fonctionnalité. Un interface graphique spécialisé pour ces acteurs aurait pu être réaliser mais il n'est pas crucial pour ce travail et sera donc omis. Afin d'inscrire un nouvel instrument, il faudra donc passer par un client REST pour envoyer la requête au service « ExchangeAPI ». Ce service s'occuper de gérer l'enregistrement au niveau du service « MatchingEngine » pour assurer qu'il n'y ait pas d'incohérence entre les services.

5.5.6.2 Exemple de requête

Figure 18: Exemple de requête d'enregistrement d'un instrument



Capture d'écran de l'outil Postman

5.5.6.3 Exemple de réponse

```
Status          : 201 Created
Content-Type    : application/json; charset=utf-8

{
  "instrument": {
      "id": "5e5797fbaa3a3104124dfef7",
      "address": "0x77680e10c5fe180f317B4B0da9A0672C53608aDA",
      "name": "Facebook",
      "symbol": "FCBK",
      "abi": [
          {
              "type": "event",
              "inputs": [
                  {
                      "name": "a",
                      "type": "uint256",
                      "indexed": true
                  },
                  {
                      "name": "b",
                      "type": "bytes32",
                      "indexed": false
                  }
              ],
              "name": "Event"
          },
        // Rest of the abi…
      ]
  }
}
```

5.5.6.4 Remarques

Nous pourrions utiliser les numéros d'identification unique auto-générer par MongoDB plutôt que d'utiliser les adresses de blockchain. Elles sont également uniques et permettrait d'arriver aux mêmes fins. En revanche, en supposant que l'on souhaite par la suite donner la possibilité aux smart contracts représentant les instruments d'avoir différentes versions (par exemple en cas de défaillance ou d'amélioration), il faut que le modèle de donné soit le plus normalisé possible. Utilisé les adresses comme numéro d'identification ne permettrait donc pas d'obtenir cette modularité. En revanche, pour le but de ce travail elles sont largement suffisantes.

5.5.7 Modification et suppression d'un instrument financier

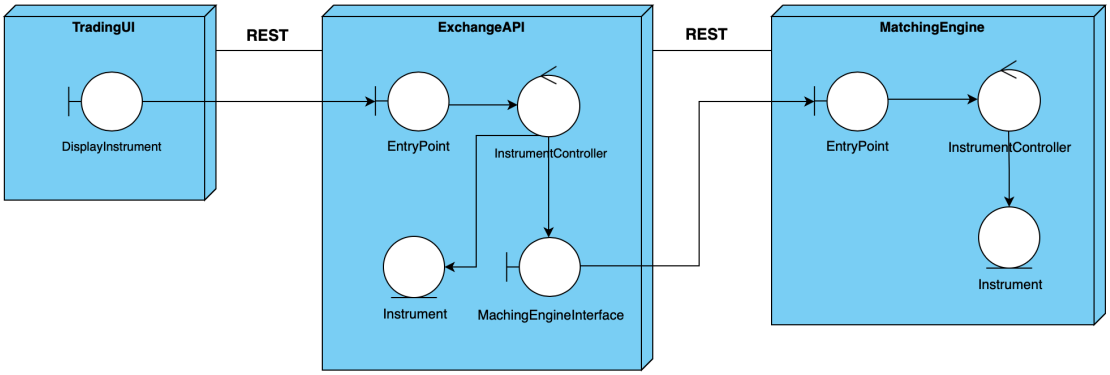
La mise à jour et la suppression d'instruments ne sont pas mises en place pour des raisons de sécurité. Dans le cas où l'adresse d'un instrument serait mise à jour, les traders possédant cet instrument ne serait plus en mesure de l'échangé et perdrait donc la totalité de sa valeur sur la plateforme d'échange. Le problème est le même dans un cas de suppression d'instrument. On pourrait cependant imaginer un cas de figure où une institution, ayant mis un instrument en circulation, se verrait dans l'obligation de redéployer une version comportant un patch de sécurité ou une modification du comportement du contrat. Ce redéploiement engendrerait la modification de l'adresse initial du contrat. Il serait donc utile de pouvoir modifier l'adresse ou bien l'ABI du contrat.

En revanche, cela demande d'avoir une validation et une supervision beaucoup plus complexe des contrats en circulation qui sont acceptés sur la plateforme. Il serait donc nécessaire d'ajouter un mécanisme de consensus (manuel ou automatisé) à la plateforme afin de pouvoir prévenir les modifications frauduleuses de contrats se trouvant sur l'échange décentralisé. Pour cette raison, la modification et la suppression d'instrument ne sera pas couvert dans ce travail, que ce soit au niveau conceptuel ou pratique.

5.5.8 Affichage des instruments financier disponible sur la plateforme

Une fois enregistrer, un trader doit pouvoir connaitre et rechercher les instruments disponibles sur la plateforme. Pour que l'expérience d'utilisation soit semblable à celle des applications de trading centralisé, il est primordial que l'utilisateur puisse chercher les titres par leur nom ou par leur symbole plutôt que par leurs adresses.

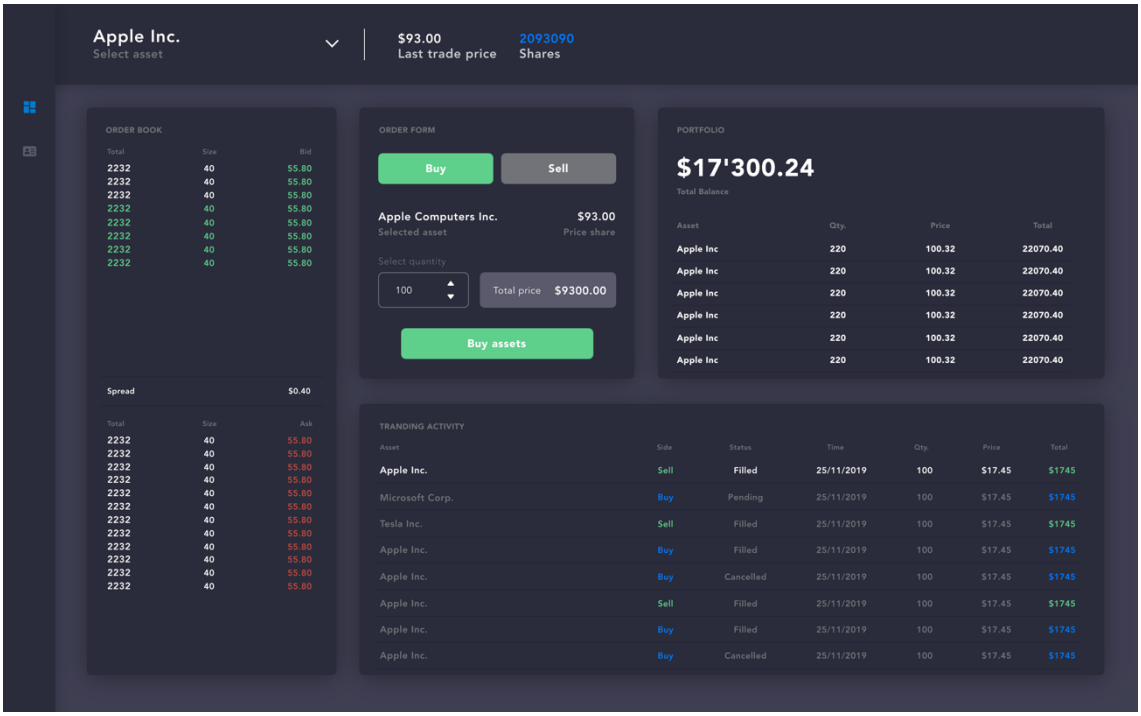
Figure 19: Diagramme des microservices impliqués dans l'affichage des instruments financier



Fait à l'aide du logiciel Visual Paradigm

Lorsqu'un utilisateur se connecte, il est directement redirigé sur son tableau de bord. Depuis ce dernier, il peut sélectionner l'instrument qui l'intéresse dans une liste déroulante (en haut à gauche sur la figure ci-dessous). En fonction de l'élément choisi, le carnet d'ordre est automatiquement mis à jour avec les informations relatives au titre.

Figure 20: Tableau de bord de la plateforme



Capture d'écran de la plateforme

Les prix affichés dans le carnet d'ordres sont directement récupérés depuis le service « MatchingEngine » et sont mis à jour à chaque fois qu'un nouvel ordre est reçu sur ce dernier. L'implémentation de la logique d'allocation d'ordre est détaillée dans la section « Order book ».

5.5.9 Dépôt de tokens pour lancer la session de trading

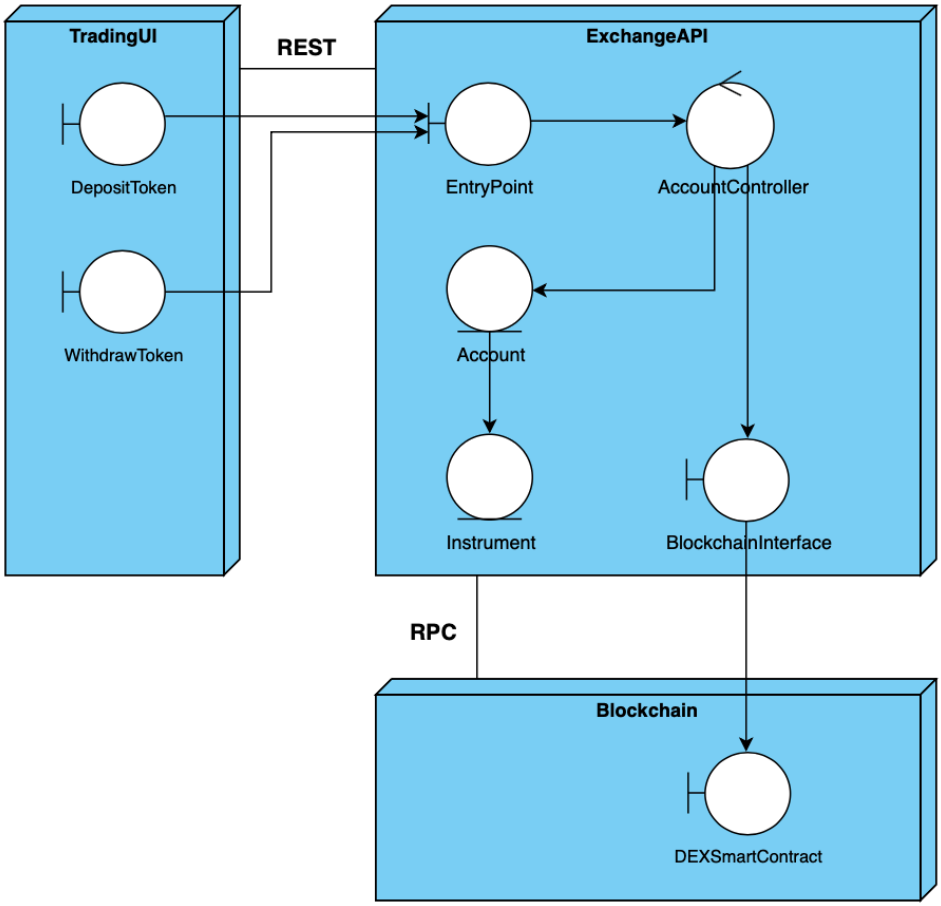
Comme évoqué précédemment, pour que les ordres soient exécutés instantanément, il n'est pas possible de traiter l'allocation d'ordre directement sur la blockchain. En revanche, grâce à une technique connue sous le nom de « State Channel » (SC), il est possible de garantir un échange hors-blockchain (HB) par un dépôt sur un smart contract se trouvant sur la blockchain (pour plus d'information se référer à la section State Channel). En déplaçant la gestion des ordres en dehors de la blockchain tout en garantissant chacun des échanges par un dépôt, il est alors possible de gagner en vitesse d'exécution tout en réduisant fortement le risque de conflit lié à des échanges HB.

Il est cependant important de noter que le principe des SC est généralement appliqué entre deux acteurs qui se connaissent mutuellement. Or, dans un système de trading, il n'est ni nécessaire, ni possible pour un acteur de connaitre chacun des traders présents sur la plateforme. Bien qu'un échange HB puisse être effectué instantanément, ce n'est pas le cas du dépôt initial qui doit être effectué sur la blockchain et qui est limité par la vitesse de transaction de cette dernière. Il n'est donc pas envisageable que pour chaque échange entre deux acteurs il faille créer une nouvelle SC entre les deux afin de pouvoir procéder à l'échange HB.

Il a donc fallu trouver une solution qui permette à un trader de garantir tous ses échanges lors d'une session de trading, sans pour autant devoir créer des nouvelles SCs en permanence. Donc plutôt que de définir un montant à déposer pour un échange entre deux traders, il faut définir un montant pour l'entièreté de la session de trading. Avant de pouvoir placer un ordre, le système va vérifier que le montant total des ordres du trader ne dépasse pas le montant total du dépôt.

Le problème des multiples SC est résolu, mais cela en crée un autre. Sur une SC simple, les échanges d'un token A sont garantis par un dépôt de token A. Or, sur l'échange nous voulons pouvoir échanger un token de type monétaire pour un token de type actif financier (une action par exemple). Les différents types n'ayant pas le même prix à l'unité, il n'est donc pas possible de garantir un échange uniquement par une certaine quantité de token. Il est donc nécessaire de prendre en compte le prix à l'unité de chaque token et d'exprimer le montant total des actifs bloqués pour une session dans un unité de mesure unique. En l'occurrence, l'unité de mesure choisi est le « USDX ». Il est également important de prendre en compte que les prix des titres fluctuent en permanence et donc que les montants des dépôts également. Il faut donc actualiser périodiquement le prix sur chaque smart contract d'instrument financier pour refléter ces fluctuations.

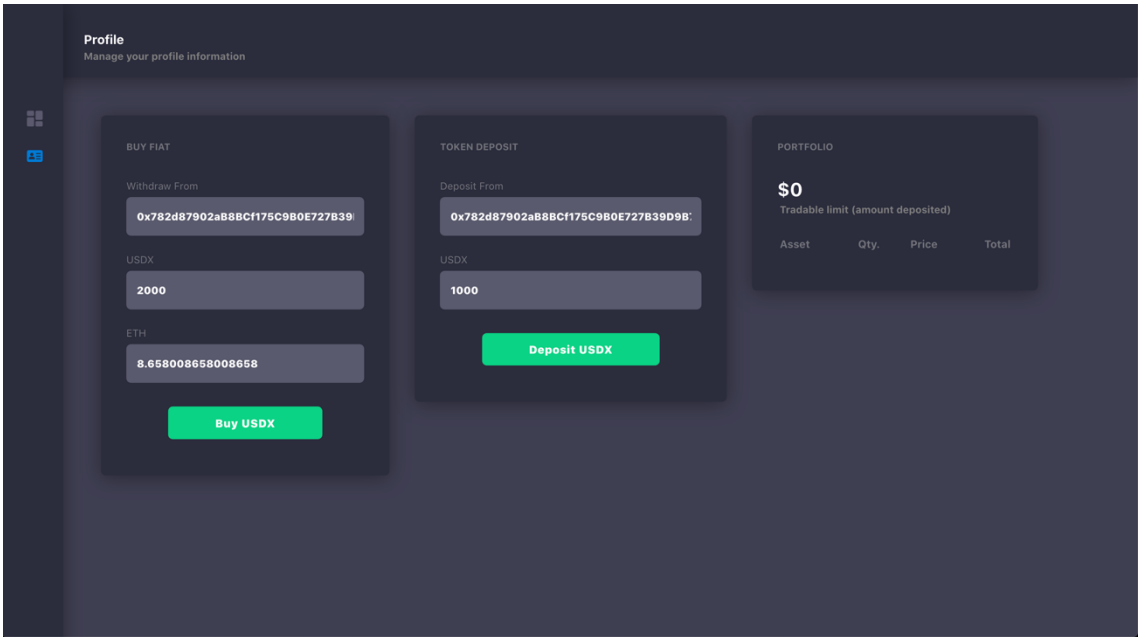
Figure 21: Diagramme des microservices impliqués dans le dépôt de tokens



Fait à l'aide du logiciel Visual Paradigm

Pour pouvoir effectuer un dépôt et placer des ordres, le trader doit donc avoir en sa possession, des tokens de type « USDX ». Il peut en acheter directement sur la plateforme contre des ETH. Dans la partie « Profile » de la plateforme, il lui suffit de rentrer le montant qu'il désire obtenir dans le formulaire « Buy Fiat » (voir figure ci-dessous). Il pourra ensuite utiliser le formulaire de dépôt qui se trouve au même endroit pour pouvoir initier une session de trading avec la limite qu'il souhaite s'accorder.

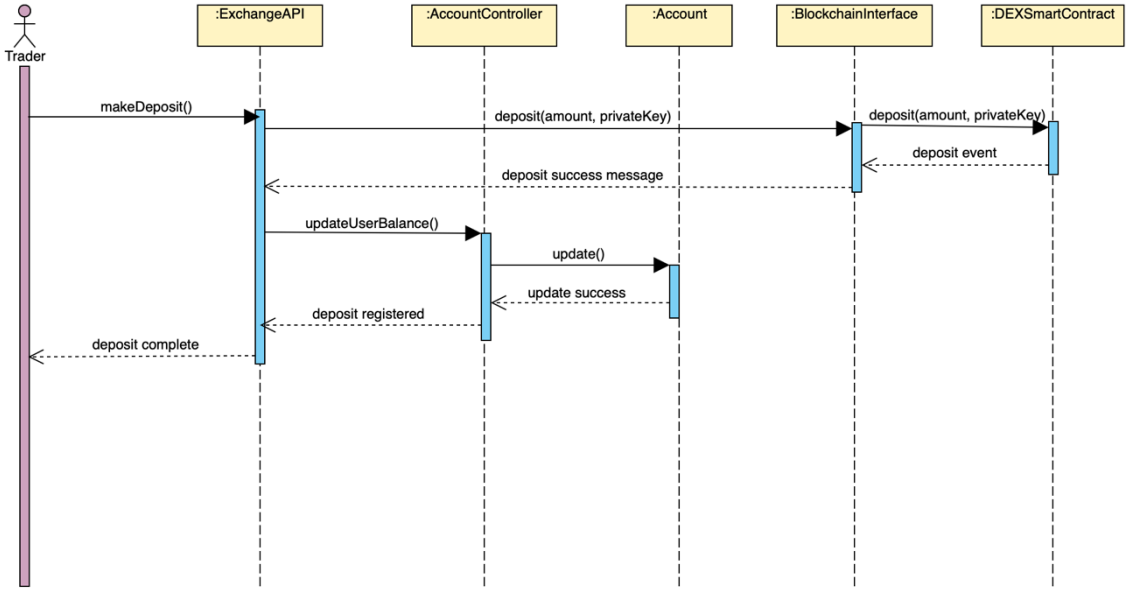
Figure 22: Interface pour la fonctionnalité de dépôt



Capture d'écran de la plateforme

Les interactions entre les composants du système qui réalisent la fonctionnalité de dépôt de token sont représentées ci-dessous par un diagramme de séquence. Pour comprendre comment le Smart Contract nommé « DEX » implémente le dépôt de token, se référer à la section portant sur l'architecture des Smart Contracts.

Figure 23: Diagramme de séquence des composants impliqués dans le dépôt de tokens



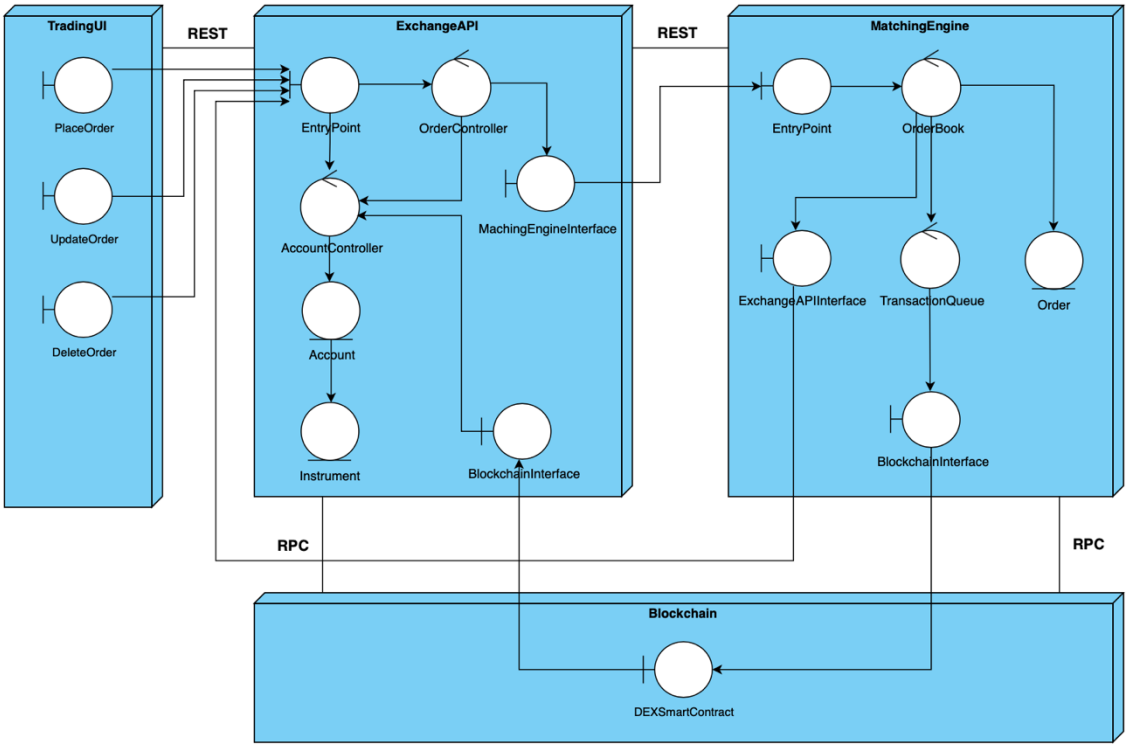
Fait à l'aide de l'outil Mermaid Live Editor

5.5.10 Placement d'un ordre d'achat ou de vente pour un instrument financier spécifique

Après avoir déposé des fonds sur le DEX en passant par l'interface graphique, un trader doit pouvoir placer des ordres d'achat ou de ventes pour chaque actif disponible sur la plateforme. Il doit également pouvoir modifier ou annuler un ordre envoyé et consulter l'état d'un ordre à tout moment.

Avec l'approche hybride choisi, contrairement aux échanges décentralisés avec un carnet d'ordre on-chain, aucune commission n'est prélevée lors d'une annulation ou d'une modification. En effet, n'ayant pas besoin d'interagir avec la blockchain pour effectuer ce type d'opération, aucun frais n'est engendré.

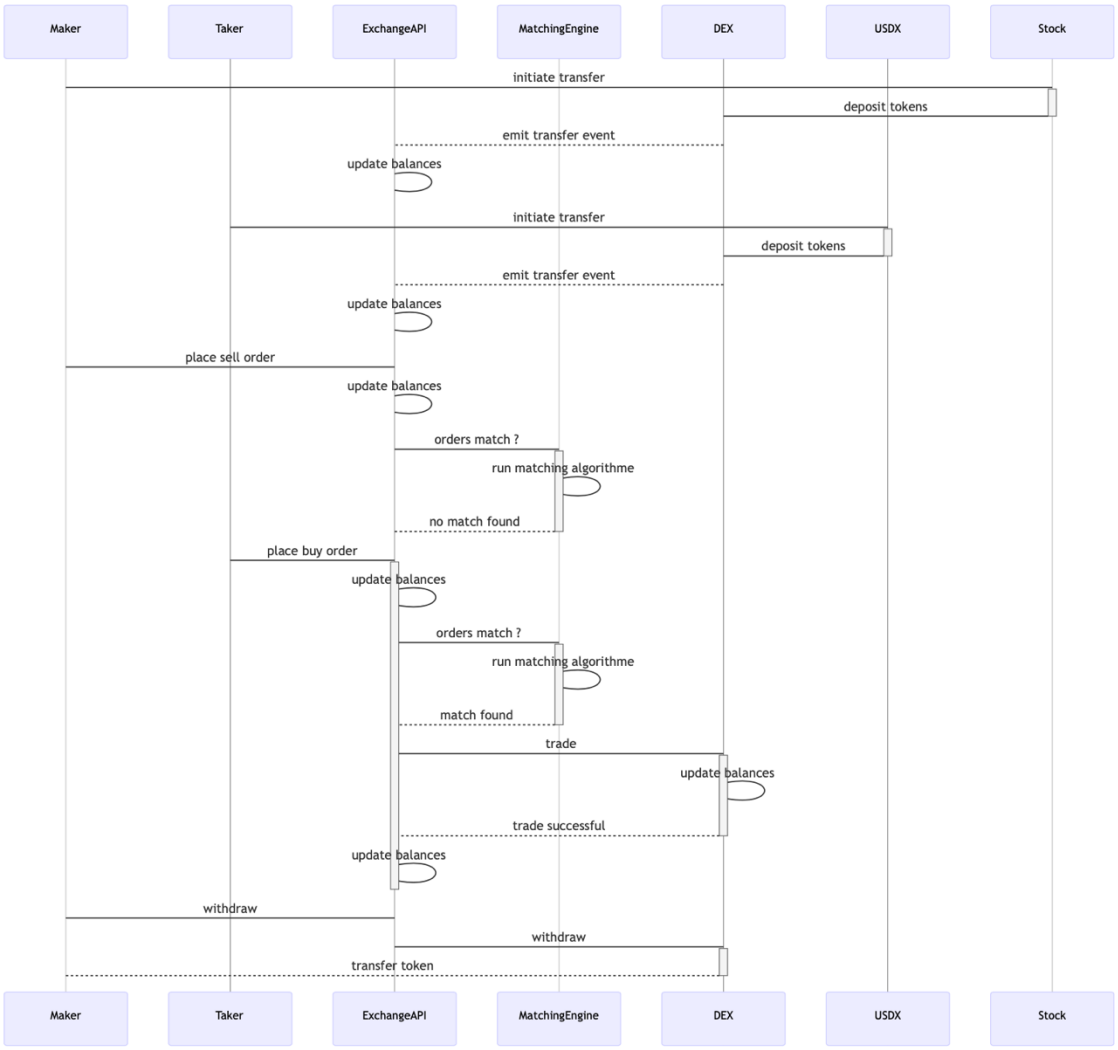
Figure 24: Diagramme des microservices impliqués dans le placement et l'exécution d'un ordre



Fait à l'aide du logiciel Visual Paradigm

Le processus de placement d'ordres est décrit dans le diagramme de séquence ci-dessous. Il met en avant les actions effectuer par chacun des composants de haut-niveau impliqués dans ce cas d'utilisation ainsi que les messages et informations échangés.

Figure 25: Diagramme de séquence du placement d'un ordre de trading



Fait avec l'outil Mermaid Live Editor

Une information essentielle n'est cependant pas présente sur le schéma : la signature des messages envoyés off-chain. Dans le cadre standard de l'utilisation de canaux d'état, le processus est simple. Chacun des acteurs voulant s'échanger des montants sans devoir passer par la blockchain déposent une partie de leurs fonds dans un Smart Contrat. Ils peuvent ensuite procéder à des échanges off-chain qui doivent être inférieur à la somme déposée. Chaque fois qu'ils s'envoient une transaction, les deux parties doivent signer le message cryptographiquement. Lorsqu'ils ont terminé leurs échanges,

ils peuvent envoyer la dernière transaction signée représentant l'état final de leurs comptes.

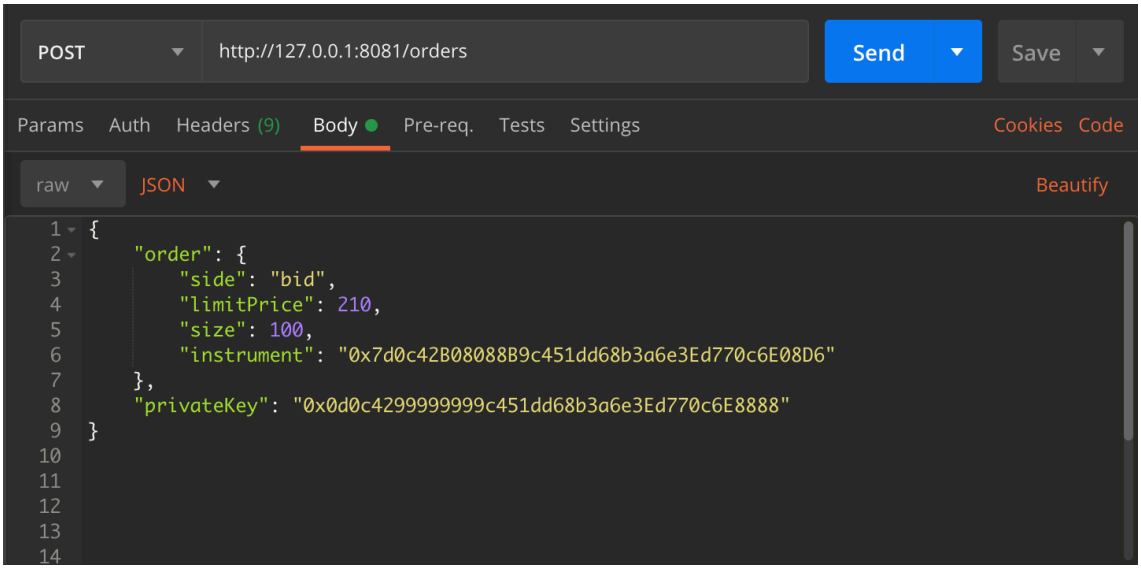
Dans le cadre d'une plateforme de trading en revanche, les acteurs ne communiquent pas directement et ne se connaissent pas et ils doivent passer par la plateforme pour conclure des échanges. Ils ne sont donc pas en charge de conserver les messages signés, ni de connaitre l'état des comptes des autres utilisateurs. C'est le rôle de l'ExchangeAPI. Toute la logique de signature et de conservation des messages est donc effectuée sur ce service. Le processus est le suivant :

1. Réception de l'ordre d'un trader
2. L'API vérifie que le trader a suffisamment de fonds pour placer un ordre d'un certain montant
3. L'API vérifie que le trader a suffisamment de fonds déposé dans le DEX
4. LAPI signe l'ordre avec la clé privé envoyé par le trader (sans ne jamais la conserver)
5. L'API conserve l'ordre signé en mémoire pour référence
6. L'API met à jour l'état du compte du trader pour refléter la transaction
7. L'API met à jour l'historique d'activité du trader pour refléter la transaction
8. L'API envoie l'ordre signé au service MatchingEngine
9. L'API met à jour le statut de l'ordre en fonction du résultat d'exécution retourné par le MatchingEngine (le statut peut être un des suivants : pending, cancelled, filled, partially-filled)
10. L'API met à jour l'état du compte du trader pour refléter le résultat d'exécution
11. L'API met à jour l'historique d'activité du trader pour refléter le résultat d'exécution

Le service d'appariement d'ordre à deux responsabilités. La première étant bien évidement de trouver des correspondances aux ordres qu'il reçoit et de les exécuter. Ce processus est expliqué en détail dans la section concernant l'implémentation du carnet d'ordres. Sa deuxième responsabilité est de grouper les ordres pour lesquels il a trouvé une correspondance et de les envoyer sur la blockchain. Ceci, afin que le contrat DEX puissent mettre à jour les balances des utilisateurs et refléter l'état réel des comptes. Pour que les transactions soient envoyées dans l'ordre réel d'exécutions, elles sont placées dans une queue. Un mécanisme va ensuite les prendre une par une et par ordre d'arriver. Pour pouvoir être accepter par le DEX, il est impératif que le service MatchingEngine signe cryptographiquement chaque pair d'ordres avant de les diffuser. Le DEX est conçu pour n'accepter qu'uniquement des pairs d'ordre signés par la clé privée du MatchingEngine. Chaque ordre d'une paire doit également comporter une signature valide provenant de l'adresse du signataire, autrement dit du trader.

5.5.10.1 Exemple de requête

Figure 26: Exemple de requête d'un placement d'ordre



Capture d'écran de l'outil Postman

5.5.10.2 Exemple de réponse d'un ordre sans correspondance

```
Status          : 201 Created
Content-Type    : application/json; charset=utf-8

{
    "order": {
        "id": "5ed381a8fd3f3503f149fc31",
        "account": "0x3d088960898540017ABeCEcAf6017246899495e4",
        "side": "bid",
        "instrument": "0x7d0c42B08088B9c451dd68b3a6e3Ed770c6E08D6",
        "limitPrice": "210",
        "size": "100",
        "sizeRemaining": "100",
        "status": "pending",
        "entryTime": 1590919592914
    },
    "user": {
        "id": "0x3d088960898540017ABeCEcAf6017246899495e4",
        "balances": [...]
    }
}
```

5.5.10.3 Remarques

La modification d'un ordre correspond à la suppression d'un ordre existant et la création d'un nouvel ordre et non à la simple modification d'un ordre existant. Cette distinction est importante vis-à-vis de l'algorithme d'allocation d'ordre. En effet, comme la liste des ordres est triée par ordre d'arrivée et par prix, il se pourrait qu'un trader utilise la modification à son avantage. Il pourrait placer des ordres arbitraires avec un prix ou une quantité aberrante, puis modifier un de ses ordres pour qu'il soit exécuté en priorité en fonction des nouvelles conditions du marché.

Pour atteindre une vitesse d'exécution raisonnable, il n'est pas possible de stocker les ordres dans une base de données sur le service MatchingEngine. Leurs traitements (CRUD) ralentiraient fortement le système dans son ensemble. Il est donc nécessaire de stocker les ordres en mémoire (in-memory) et de ne pas persister les données. Cependant, il est important de considérer l'éventualité où le serveur tomberait en panne ou qu'il ait besoin d'être arrêté pour des raisons de maintenance. Dans ces cas de figures, les ordres maintenus en mémoire serraient perdus. Il est donc nécessaire de stocker les ordres de manière plus persistante. Nous pourrions envisager d'utiliser le pattern « Event Sourcing » pour y parvenir (Fowler, 2005). Nous pourrions alors utiliser une base de données et y stocker toutes les modifications apportées (création, modification, suppression d'ordres) aux carnets d'ordre (LOB) sous la forme d'une séquence d'événements. Avec cette approche, lors d'un redémarrage forcé du service MatchingEngine, il suffirait de reproduire la séquence d'évènement pour reconstruire le carnet d'ordre. Cette optimisation, bien que nécessaire, sort du cadre de ce travail et ne sera donc pas mise en place dans l'application.

Il est également intéressant de noter que dans le cas de figure d'un échange décentralisé avec un carnet d'ordre on-chain, le seul moyen d'augmenter la vitesse de traitement est d'augmenter les frais par transaction pour obtenir une priorité plus élevée. En revanche dans le cadre d'un échange décentralisé avec un carnet d'ordres off-chain, la vitesse de traitement d'ordres au niveau de la blockchain perd de l'importance étant donné que les transactions sont conclues off-chain, puis persister on-chain par la suite. Les seules actions, dont la vitesse reste dépendante de la blockchain, sont alors les dépôts et retraits qui doivent forcement s'effectuer on-chain. On peut néanmoins envisager la possibilité d'augmenter les frais pour ces actions afin de bénéficier d'une priorité maximale pour pouvoir déposer ou retirer ses fonds le plus rapidement possible. L'utilisateur serait donc libre de choisir entre rapidité et économie.

5.6 Limitations et risques

5.6.1 Matching engine off-chain et centralisé

Dans l'architecture proposée, le plus gros point de défaillance potentiel est bien évidement le server centralisé où les correspondances d'ordres sont effectuées. Même si les échanges entres les traders sont garanti par le dépôt de tokens dans le Smart Contract et que les éventuels conflits entre trader peuvent être régler en se tournant vers la blockchain qui agira en tant que juge, ce serveur reste sujet à divers types d'attaque. Attaques qui peuvent être difficilement traçable et donc par extension difficilement contestables.

5.6.1.1 Attaque par front-running

Par exemple, un acteur malveillant pourrait infiltrer le serveur et avoir connaissance des nouveaux ordres entrants et injecter des ordres qui jouerait en sa faveur. Cette technique de placer un ordre en aillant connaissance d'information qui ne sont pas censés être connue du public et qui ont suffisamment d'ampleur pour faire fluctuer les prix du marché s'apparente au « front-running ».

5.6.1.2 Attaque par censure

Un pirate pourrait également modifier l'heure ou la date de création d'un ordre pour se placer les siens en tête de liste. Cette liste, ou plutôt cette queue d'exécution, qui est normalement sensée exécutée les transactions dans un ordre « First-in-first-out » serait trompé et exécuterais des ordres plus récents en premier. L'hacker pourrait donc placer des ordres qui serait fortement avantageux et bénéficierais d'une priorité synthétique. Il pourrait également supprimer les ordres de certains traders en créant une sorte de censure lui étant favorable.

5.6.1.3 Défaillance

Dans une autre optique, le serveur centralisé de correspondances pourrait tout simplement être défaillant. Il pourrait par exemple ne plus être déterministe et donc ne plus exécuter les ordres de manière équitable et juste.

6. Résultats

Grâce à la recherche effectuée, nous avons donc été en mesure de mette en place une plateforme de trading décentralisée permettant de tester nos hypothèses. Le prototype réalisé utilise des *Smart Contracts* déployés sur la blockchain Ethereum, ainsi que des *State Channels* permettant d'augmenter la vitesse d'exécution des ordres. Une crypto-monnaie représentant le dollar américain a été implémentée et est utilisée comme moyen d'échange des titres en circulation sur la plateforme. L'objectif de ce chapitre est de tirer des conclusions basées sur la partie pratique et théorique du travail et de répondre aux questions posées au départ.

Premièrement, en ce qui concerne les frais de transaction, la réponse est quelque peu ambiguë. En effet, on constate que l'approche choisi permet d'utiliser la blockchain comme registre final et comme garant des échanges entre acteurs hors-blockchain. Ne nécessitant pas de faire appel à une partie tierces, comme c'est actuellement le cas pour tout type d'échange centralisés, la plateforme n'est donc pas soumise à des frais externes pour la validation effectuée. Nous avons également observé qu'en utilisant des canaux d'état, on peut éviter de devoir recourir à la blockchain lors de chaque transaction et ne payer des frais que lorsqu'une modification finale d'état à lieu sur la chaîne.

De cette perspective, nous pouvons alors dire que les frais de transactions sont moindres, en comparaisons des plateformes de trading ayant des caractéristiques semblables et un débit du même ordre de grandeur. Cependant, il ne faut pas oublier de prendre en compte que, l'infrastructure n'étant plus totalement décentralisée, les services centralisés qui la compose doivent être développés, maintenus et gérés. L'entité qui en a la charge doit se rémunérer d'une certaine manière. Plusieurs alternatives de rémunération sont envisageables. La plateforme Robinhood, par exemple, permet aux traders d'effectuer de transactions illimitées sans commissions. Elle se rémunère majoritairement par des arrangements en vertu desquels elle est rémunérée pour acheminer les ordres des clients à un teneur de marché donné (« payement for order flow » est le terme exacte). L'option la plus standard et la moins controversée reste néanmoins le prélèvement de frais lors de chaque transaction.

A ce stade, il est alors nécessaire de faire une distinction entre la capacité d'une infrastructure à fonctionner à moindre coûts et sa capacité à se rémunérer. La hauteur des frais est donc fonction des charges que doit amortir l'entité centrale et de la méthode de rémunération choisie. Il est par conséquent difficile de donner une estimation globale et une étude beaucoup plus poussée du modèle d'affaires d'une telle structure serait requise pour y parvenir.

Ensuite, nous avons pu observer une exécution des ordres instantanée. Bien que nous puissions théoriquement garantir un débit suffisamment grand, il serait toutefois nécessaire d'effectuer des tests de charges plus approfondis pour valider de manière plus scientifique cette constatation. Comme évoqué lors de l'implémentation, l'architecture du système a été conçue de manière à ce que ce type de tests puissent être réalisés assez simplement en envoyant automatiquement des requêtes vers l'API. Dans tous les cas, nous pouvons affirmer qu'un débit d'un ordre de grandeur plus élevé peut être atteint en utilisant le mécanisme de canaux d'état.

Troisièmement, nous avons pu voir qu'il est possible de garantir aux utilisateurs un contrôle total sur leurs fonds tout en utilisant une infrastructure hybride. Effectivement, en centralisant l'appariement d'ordre et en garantissant les échanges par des dépôt sur la blockchain, on peut garantir un débit suffisamment grand, sans pour autant devoir dépendre d'une figure d'autorité centrale. Bien que le service d'appariement soit en charge de l'envoie final d'une transaction sur le réseau Ethereum, il ne peut y envoyer que des transactions signées cryptographiquement par les deux parties. Comme les seuls détenteurs des clés privés sont les traders, leurs signatures sont reconnues d'office comme valides et authentiques.

Avec cette approche, on constate également que malgré le manque de contrôle sur les composants centralisés du système, les utilisateurs gardent une maîtrise total sur les fonds qu'ils déposent dans les divers Smart Contracts de l'échange. Ils sont en mesure d'initier un retrait total ou partiel de leurs tokens à tout moment, sans devoir passer par la plateforme. Ceci, leur garantissant un contrôle même en cas de panne ou d'attaques d'un des services centralisés.

Finalement, nous avons pu voir qu'il est possible de remédier au problème de volatilité des crypto-monnaies dans le cadre d'un échange de titres. Ceci, en utilisant une monnaie virtuelle dont la stabilité est assurée par un ou plusieurs actifs physiques, tels que de la monnaie fiduciaire ou des commodités. Les Stable Coins sont cependant assez controversés à cause de la centralisation de leurs réserves. En les utilisant, bien qu'ils permettent d'assurer une stabilité, nous nous éloignons encore un peu plus des principes fondamentaux d'une blockchain.

7. Améliorations possibles

7.1 Transparence d'exécution

Pour faire face aux types d'attaques évoqué dans les limitations de la plateforme, il est donc envisageable (voire même requis) dans un premier temps, de rendre l'exécution des ordres entièrement transparente. En rendant public la correspondance d'ordres et l'exécution des transaction, on peut prouver le déterminisme de l'algorithme d'appariement ainsi qu'une exécution en règles. Grâce au caractère pseudonyme des comptes de la blockchain, la publication des ordres et l'exécution des échanges n'aurait aucun impact sur la confidentialité des données des utilisateurs de la plateforme. Au contraire, cela permettrait de répondre d'avantage aux critères de transparence et de publicité de la blockchain. Avec un tel mécanisme, n'importe quel acteur pourrait donc comparer les logs de correspondances et d'exécution et comparer les transactions avec celle publiées sur la blockchain pour avoir la garantie d'une exécution franche et équitable (fair-play).

7.2 Échange décentralisé basé sur un réseau de State Channel

Une approche plus complexe pourrait éventuellement être utilisée afin d'avoir un mécanisme d'appariement d'ordres totalement décentralisé, composé de canaux d'état bidirectionnel. Ce réseau aurait l'avantage de ne pas avoir recours à un algorithme de consensus et n'aurait pas non plus besoin de validateurs comme dans le cas d'une blockchain « privée ».

Le protocole utilisé serait semblable aux protocoles de routage bien connus tel que EIGRP ou OSPF. Chaque nœud aurait une copie du carnet d'ordre avec, en plus de ça, l'adresse des nœuds permettant de router la transaction vers le bon destinataire. Cette approche peut s'avérer intéressante pour son aspect décentralisé et son respect plus strict des principes fondamentaux d'une blockchain. Cependant, sa faisabilité et surtout sa rapidité reste à prouver.

En effet, ce type de réseau, par exemple Raiden ou Lightning, sont généralement prévu pour des simple payement unilatéraux (par exemple payement d'un service, ou placement d'un pari) et pas pour des échanges. Ces derniers étant par définition multilatéraux et, dans le cas d'une plateforme de trading, réalisés entre deux parties ne se connaissant pas. Ce qui ajoute de la complexité dans l'algorithme de correspondance. Il faut donc trouver deux chemins différents qui possède suffisamment de tokens transférables disponibles. Un chemin pour l'acheteur, un chemin pour le vendeur. Comme cet échange implique un transfert de chaque partie prenante, il faut pouvoir

garantir la réception de chacun et donc employé soit un intermédiaire (comme un Smart Contract) qui transfèrerait les fonds lors de leur réception (de chaque côtés). Soit un protocole qui permette de verrouiller les actifs échangés, comme pour les « hash time locked transaction » du réseau Raiden par exemple (Raiden, 2018), avant de recevoir l'accusé de réception des deux parties.

8. Conclusion

Dans ce travail de recherche, nous avons pu démontrer qu'il est possible de réaliser une plateforme de trading basée sur une blockchain qui offre une expérience d'échange en temps réel et à haut débit tout en conservant le contrôle de ses propres fonds.

La réalisation d'un prototype a permis de prouver qu'en centralisant l'appariement des ordres et la diffusion des transactions sur la blockchain, il est possible pour les traders d'échanger des actifs en continue, sans devoir attendre qu'un consensus soit atteint par tout le réseau sur la validé d'une écriture. Nous pouvons affirmer qu'un débit d'un ordre de grandeur supérieur à la vitesse de résolution propre à la blockchain Ethereum peut être atteint en utilisant le mécanisme de canaux d'état. Même si la centralisation du système d'appariement nous éloigne une peu plus des principes fondamentaux de la blockchain, nous sommes en mesure de valider les transactions sans dépendre d'une tierce partie.

D'autre part, nous avons pu remarquer que la commission prélevée sur chaque ordre dépend en fin de compte du modèle d'affaires choisi. Néanmoins, nous avons pu démontrer qu'en adoptant une architecture hybride, il est possible de faire baisser les coûts de transactions tout en offrant une expérience de trading semblables à celle des échanges de titres centralisés.

En conclusion, une plateforme de trading basée sur une blockchain est réalisable, mais des compromis doivent être fait sur l'aspect de décentralisation. Pour qu'une telle classe d'application ait une architecture entièrement distribuée, les technologies de registre décentralisés ont encore du chemin à parcourir.

Annexe 1 : Lexique des termes financier

8.1.1 Broker

De manière générale, le terme broker représente une entité agissant comme intermédiaire entre un acheteur et un vendeur. Dans le domaine de la finance, un broker est une entreprise qui facture des frais ou une commission pour l'exécution des ordres d'achat et de vente de titres passés par un investisseur. (Smith, 2019)

8.1.2 Market order

Un « market order », ou « ordre au marché » en français, est une demande d'un investisseur (généralement faite par l'intermédiaire d'un broker) d'acheter ou de vendre un titre au meilleur prix disponible sur le marché actuel. Un *market order* est considéré comme le plus simple et le plus fondamental de tous les types d'ordres. Il est destiné à être exécuté le plus rapidement possible au prix demandé actuel d'un titre (ask price). (Dhir, 2019)

8.1.3 Limit order

Un « limit order », ou ordre à cours limité en français, est une demande d'un investisseur (généralement faite par l'intermédiaire d'un broker) d'acheter ou de vendre un titre à un prix spécifique. Par exemple, s'il souhaite acheter des actions d'une valeur de 80$ à 78$ ou moins, il peut placer un *limit order* qui ne sera pas exécuté à moins que le prix qu'il a spécifié (ou un prix inférieur) ne soit disponible. Il n'est cependant pas possible de définir un ordre de ce type pour acheter une action au-dessus du prix du marché, car un meilleur prix est déjà disponible. Il en est de même pour les ordres de ventes. Tant que le prix spécifié par le trader (ou un prix plus élevé) n'est pas disponible sur le marché, alors l'ordre ne sera pas exécuté. Symétriquement, il n'est pas possible de définir un ordre à cours limité pour vendre en dessous du prix du marché actuel car de meilleurs prix sont disponibles. (Majaski, 2019)

8.1.4 Order book

Un carnet d'ordres, ou « order book » en anglais, est utilisé pour enregistrer publiquement l'intérêt des acheteurs et des vendeurs pour un instrument financier particulier. Chaque entrée comprend le nombre d'actions et le prix que l'acheteur ou le vendeur offre ou demande pour le titre en question.

Le carnet d'ordres aide les traders à prendre des décisions de trading informées. Ils peuvent voir si le marché est influencé par les investisseurs particuliers ou par les institutions en se basant sur la taille des ordres placés sur un marché donné. Le carnet

d’ordres permet également de mettre en avant les déséquilibres d’ordre susceptibles de fournir une indication quant à l’orientation du marché à très court terme.

Un carnet d'ordres se compose généralement de trois parties : les ordres d'achat, les ordres de vente ainsi que l’historique des ordres. Les ordres d'achat contiennent toutes les informations sur les acheteurs présents sur le marché, notamment toutes les offres, le montant qu'ils souhaitent acheter et le prix demandé. Les ordres de vente sont semblables aux ordres d'achat, mais bien évidement dans la direction opposée. L’historique des ordres liste toutes les transactions qui ont eu lieu dans le passé. (Kenton, 2019)

8.1.5 Bid et Ask

Les termes « bid » et « ask » font référence à une offre de prix bidirectionnelle indiquant le meilleur prix auquel un titre peut être vendu et acheté à un moment donné. Le « bid price », ou prix offert en français, représente le prix maximum qu'un acheteur est prêt à payer pour un titre. Le « ask price », ou prix demandé en français, représente le prix minimum qu'un vendeur est disposé à recevoir. Une transaction a lieu dès lors que l'acheteur et le vendeur se sont mis d'accord sur un prix d’échange pour un titre donné.

La différence entre les cours acheteur et vendeur–le *spread* (écart entre ces deux prix)–est un indicateur clé de la liquidité de l'actif en question. En général, plus le *spread* est faible, meilleure est la liquidité. (Chen, 2018)

8.1.6 Profondeur du marché

En finance, la profondeur du marché est une liste en temps réel affichant la quantité à vendre par rapport au prix unitaire. La liste est organisée par niveau de prix et reflète l'activité du marché en temps réel. Mathématiquement, il s'agit de la taille d'une commande nécessaire pour déplacer le prix du marché d'un instrument financier donné. (Wikipedia, 2016)

8.1.7 Front-running

Le « front-running » est une pratique illégale (dans la majeure partie des cas) dans laquelle un courtier ou une autre entité conclut une transaction parce qu’il a une connaissance préalable d’une grande transaction non annoncée qui influera sur le prix de l’actif. Ce qui engendra un gain financier pour le courtier. Cela se produit également lorsqu'un courtier ou un analyste achète ou vend des actions pour son compte avant d’avoir émis la recommandation d'achat ou de vente à ses clients (Mitchel, Scott, 2020).

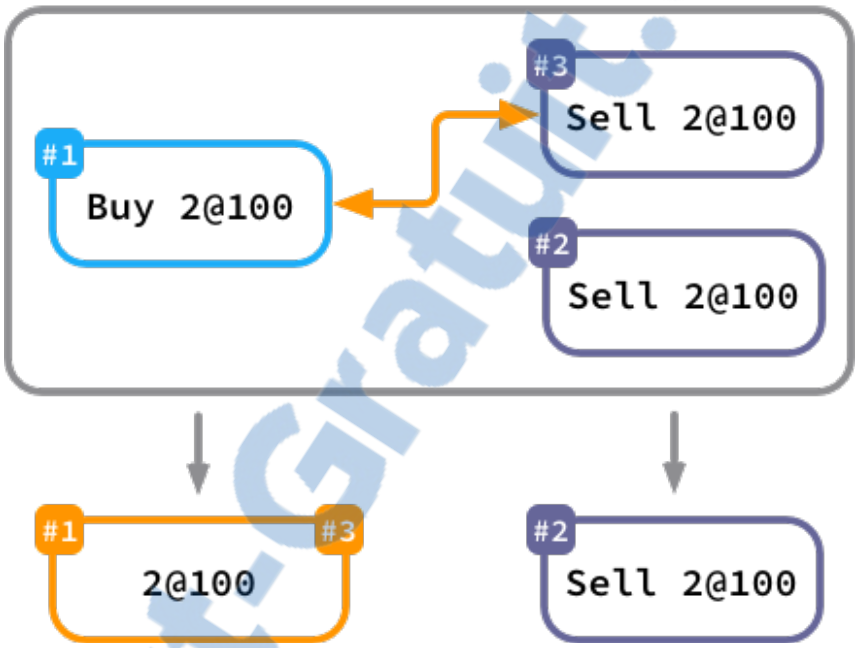
8.1.8 Arbitrage

L'arbitrage est l'achat et la vente simultanés d'un actif pour bénéficier d'un déséquilibre de prix. Ce type de transaction profite des différences de prix d'instruments financiers identiques ou similaires sur différents marchés. Ce genre de stratégie de trading est possible grâce aux inefficiences du marché (Chen, 2020).

Annexe 2 : Algorithme de correspondance d'ordres

First-in-first-out (FIFO)

L'algorithme FIFO utilise le prix et l'heure comme seul critère pour exécuter un ordre. Dans cet algorithme, tous les ordres du même niveau de prix sont exécutés en fonction de la priorité temporelle. Le premier ordre à un niveau de prix est le premier ordre à être assorti (matched). (CME Group, [sans date]).



(Ghazi, 2018)

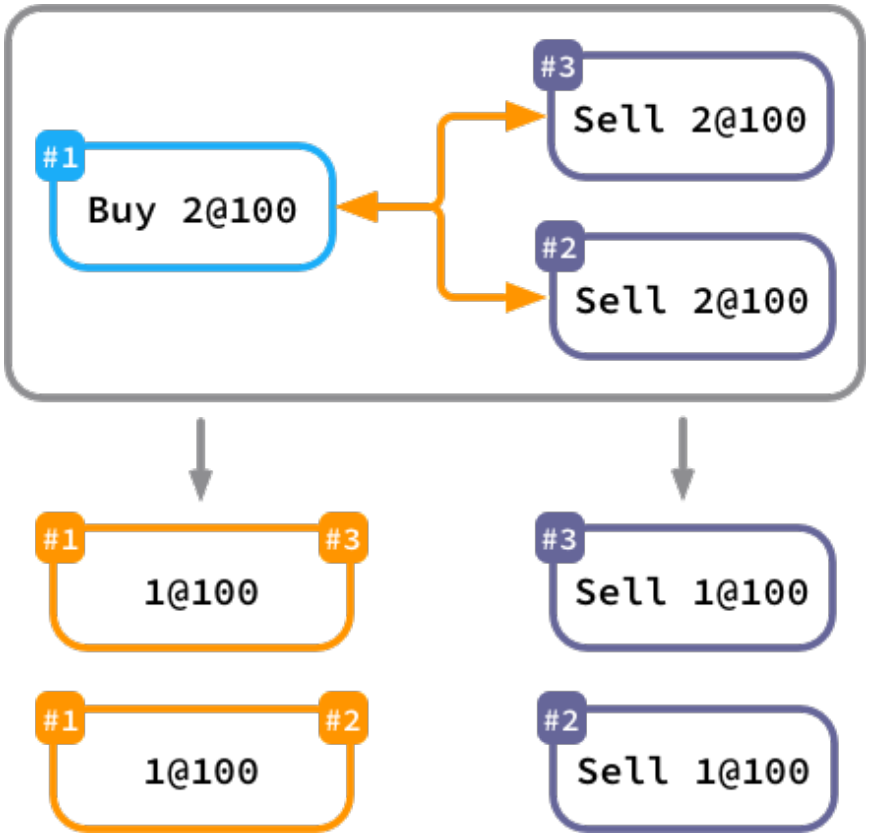
Il est important de noter qu'un ordre perd sa priorité et qu'il est remis en file d'attente (comme s'il venait d'être créé) lorsqu'il est modifié de l'une des manières suivantes (CME Group, [sans date]) :

- Augmentation de la quantité
- Changement de prix
- Changement du numéro de compte

8.1.8.1 Pure Pro-rata

L'algorithme au prorata exécute les commandes en fonction du prix, de la taille du lot et du temps. La quantité allouée de l'ordre entrant est partagée entre tous les ordres en attente du carnet d'ordres au meilleur prix. L'affectation est proportionnelle à la taille de chaque ordre (par taille on entend quantité échangée). Tous les ordres au meilleur prix (correspondant) sont pris en compte dans la répartition. La méthode de répartition trie

d'abord les ordres éligibles en fonction de leur quantité. Les commandes avec une quantité de titres plus importante sont placées en tête de liste. S'il existe des ordres avec la même quantité, ceux-ci sont ensuite triées par ordre chronologique. (Eurexchange, [sans date])



(Ghazi, 2018

Annexe 3 : Sharding

Le concept de « sharding » (fragmentation) existait bien avant de faire partie du lexique de l'industrie de la Blockchain. Il est traditionnellement utilisé dans la gestion de base de données pour assurer le fonctionnement optimal et rapide de la base de données.

Le partage traditionnel effectue cette opération en séparant une base de données en une série de bases de données plus petites qui restent connectées, mais en répartissant la charge de travail sur un grand nombre de serveurs ou de nœuds sur ce réseau de bases de données. (Beedham, 2019)

Le concept est semblable lorsqu'il est appliqué à la Blockchain. Le réseau de la Blockchain est considéré comme étant la base de données avec les nœuds représentant les serveurs de données individuels. « Sharder » (ou fragmenter) une Blockchain impliquerait de diviser le réseau de la Blockchain en segments individuels (ou fragments). Chaque fragment détiendrait une partie de la plus longue chaine existante sur le réseau et devrait être maintenu en quasi-isolation des autres fragments.

Les nœuds seraient ensuite attribués à des fragments pour vérifier les transactions et les opérations, au lieu que chacun soit responsable de la vérification de chaque transaction sur l'ensemble du réseau.

L'idée est qu'en divisant la Blockchain en segments plus faciles à gérer, elle devrait permettre d'accroître le débit des transactions et permettrait de surmonter les problèmes d'extensibilité (scalability) auxquels sont confrontés la plupart des principales Blockchains. (Pauw, 2019)

En revanche, la manière dont une blockchain peut être fragmentée dépend fortement du mécanisme de consensus sous-jacent de celle-ci. Une blockchain basée sur le mécanisme de consensus de Proof-of-Work est très difficile à fragmenter. En effet, cela impliquerait de devoir valider des transactions sans avoir accès à l'historique complet (à la plus grande chaine de bloque validés) des transactions. De nouvelles transactions devraient donc être validées sans que son historique ne soit connu ce qui pose un réel problème et va à l'encontre des principes fondamentaux qui régissent une blockchain. Par contre, dans le cas d'une blockchain qui utiliserait mécanisme de Proof-of-Stake, la fragmentation en blocs serait un peu plus simple à mettre en place. (Beedham, 2019)

Annexe 4 : Tests d'acceptation

Pour réaliser les tests d'acceptation, les fonctionnalités sont d'abord exprimées sous forme de scénarios qui mettent en avant les actions principales effectués pour arriver à un résultat.

Scénario : Un nouveau trader s'enregistre sur la plateforme

- ETANT DONEE que le trader se trouve sur la page de création de compte
- LORSQUE le trader remplis les champs « username » et « password » avec ses informations
- ET click sur le bouton « Create account »
- ALORS il doit voir un message de confirmation avec son username et sa clé privée

Afin de garantir que le code réponde aux critères d'acceptations, il suffit de les traduire en test d'acceptation (end-to-end tests). Pour le scénario précédemment évoqué, on aura donc la suite de tests suivante :

```
describe('Scénario : Un nouveau trader s'enregistre sur la plateforme', async () => {
    it('ETANT DONEE que le trader se trouve sur la page de création de compte', async () => {
        await page.goto(ROUTES.SIGNUP)
        await expect(page).toMatchElement(sel('signup-screen'))
    })

    it('LORSQUE le trader remplis les champs "username" et "password" avec ses info', async ()
    => {
        const values = {
            username: 'James',
            password: 'Bond',
        }
        await expect(page).toFillForm(sel('signup-form'), values)
    })

    it('ET click sur le bouton "Create account"', async () => {
        await expect(page).toClick('button', {
            text: 'Create account',
        })
    })

    it('ALORS il doit voir un message de confirmation avec son username et sa clé privée', async
    () => {
        await expect(page).toMatchElement(
            sel('signup-success-dialog'),
        )
        const message = await page.waitForSelector(
            sel('signup-success-message'),
        )
        await expect(message).toMatch(
            'Your account has been created successfully!',
        )
        await expect(page).toMatchElement(sel('user-address-field'))
        await expect(page).toMatchElement(
            sel('user-private-key-field'),
        )
    })
})
```

Avec cette approche, on peut avoir la certitude que la fonctionnalité est remplie. Également, on peut ajouter ou modifier du code en nous assurant que tout fonctionne comme avant la modification. Lorsque tous les tests passent, on a un output comme ceci :

```
PASS src/tests/e2e/trader-signup.test.js
  Scénario : Un nouveau trader s'enregistre sur la plateforme
    ✓ ETANT DONEE que le trader se trouve sur la page de création de compte (959ms)
    ✓ LORSQUE le trader remplis les champs « username » et « password » avec ses informations (88ms)
    ✓ ET click sur le bouton "Create account" (31ms)
    ✓ ALORS il doit voir un message de confirmation avec son username et sa clé privée (132ms)

Test Suites: 1 passed, 1 total
Tests:       4 passed, 4 total
Snapshots:   0 total
Time:        4.838s
Ran all test suites related to changed files.

Watch Usage: Press w to show more.
```

Lorsqu'un des tests ne passe plus, l'output change et nous indique où se trouve l'erreur :

```
FAIL src/tests/e2e/trader-signup.test.js
  Scénario : Un nouveau trader s'enregistre sur la plateforme
    ✓ ETANT DONEE que le trader se trouve sur la page de création de compte (836ms)
    ✓ LORSQUE le trader remplis les champs « username » et « password » avec ses informations (175ms)
    × ET click sur le bouton "Create account" (505ms)
    × ALORS il doit voir un message de confirmation avec son username et sa clé privée (508ms)
```

Annexe 5 : Librairies et frameworks utilisés

- https://solidity.readthedocs.io/en/v0.6.6/
- https://reactjs.org
- https://www.trufflesuite.com/ganache
- https://web3js.readthedocs.io/en/v1.2.7/
- https://expressjs.com
- https://openzeppelin.com

Bibliographie

AURORA LABS, 2019. *IDEX: A Real-Time and High-Throughput Ethereum Smart Contract Exchange* [en ligne]. 23 janvier 2019. [Consulté le 15 août 2019]. Disponible à l'adresse : https://idex.market/static/IDEX-Whitepaper-V0.7.6.pdf

BACK, adam, 1997. *[ANNOUNCE] hash cash postage implementation*. [en ligne]. 28 mars 1997. [Consulté le 19 août 2019]. Disponible à l'adresse : https://cypherpunks.venona.com/date/1997/03/msg00774.html

BANKRATE, [sans date]. *What is blockchain ?* [sans date]. [Consulté le 13 août 2019]. Disponible à l'adresse : https://www.bankrate.com/glossary/b/blockchain

BEEDHAM, Matthew, 2019. *Blockchain sharding made so simple your dog would understand. In : TheNextWeb.com* [en ligne] 18 janvier 2019. [Consulté le 12 août 2019]. Disponible à l'adresse : https://thenextweb.com/hardfork/2019/01/18/explainer-blockchain-sharding-beginners

BITCOINWIKI, 2014. *Cypherpunk.* [en ligne]. [sans date]. Mis à jour le 25 mars 2019. [Consulté le 11 août 2019]. Disponible à l'adresse : https://en.bitcoinwiki.org/wiki/Cypherpunk

BITCOINWIKI, 2016. *B-money.* [en ligne]. Mis à jour le 26 janvier 2013. [Consulté le 15 août 2016]. Disponible à l'adresse : https://en.bitcoin.it/wiki/B-money

BITCOINWIKI, [sans date]. *Block hashing algorithm.* [en ligne]. Mis à jour le 13 mai 2019. [Consulté le 14 août 2019]. Disponible à l'adresse : https://en.bitcoin.it/wiki/Block_hashing_algorithm

BLOCKCHAIN France, 2016. *Qu'est-ce que la blockchain ?* [en ligne]. 2016. [consulté le 15 août 2019]. Disponible à l'adresse : https://blockchainfrance.net/decouvrir-la-blockchain/c-est-quoi-la-blockchain/

DIEDRICH, Henning, 2016a. *Ethereum: blockchains, digital assets, smart contracts, decentralized autonomous organizations.* Grande Bretagne: Wildfire Publishing, 2016. ISBN 9781523930470.

HABER, Stuart, STORNETTA, W., Scott, 1991. [en ligne] [consulté le 15 août 2019]. Disponible à l'adresse : https://www.anf.es/pdf/Haber_Stornetta.pdf

SZABO, Nick, 2008. *Bit Gold.* In : Unenumerated.Blogspot.com. [en ligne] 27 décembre 2008. [Consulté le 14 août 2019] Disponible à l'adresse : http://unenumerated.blogspot.com/2005/12/bit-gold.html

WIKIPEDIA, 2013. *Cryptographic Nonce.* [en ligne] [Consulté le 15 août 2019]. Disponible à l'adresse : https://en.wikipedia.org/wiki/Cryptographic_nonce

FAIRVIEW CAPITAL, [sans date]. *The rise of blockchain.* [en ligne] [Consult ; le 31 mai 2020] Disponible à l'adresse : http://fairviewcapital.com/ourview/the-rise-of-blockchain/

ARTICLE, [date]. *Titre.* [en ligne]. [Consulté le 31 mai 2020]. Disponible à l'adresse suivante :

WERBACH, Kevin, 2017. *Blockchain, The Rise of Trustless Trust ?* [en ligne]. [Consulté le 31 mai 2020]. Disponible à l'adresse suivante : https://publicpolicy.wharton.upenn.edu/live/files/280-summary-blockchain-the-rise-of-the-trustless-trust

BINANCE, [sans date]. *What is proof of work?* [en ligne]. [Consulté le 31 mai 2020]. Disponible à l'adresse suivante : https://academy.binance.com/blockchain/proof-of-work-explained

BINANCE, [sans date] b. *What is cryptocurrency mining?* [en ligne]. [Consulté le 31 mai 2020]. Disponible à l'adresse suivante : https://www.binance.vision/blockchain/what-is-cryptocurrency-mining

BITCOINWIKI, 2019. *Bitcoin Mining.* [en ligne]. [Consulté le 31 mai 2020]. Disponible à l'adresse suivante : https://en.bitcoinwiki.org/wiki/Bitcoin_mining

PODOLAN, Dan, 2017. *The easiest way to understand Ethereum (for 5-Years-Olds).* [en ligne]. [Consulté le 31 mai 2020]. Disponible à l'adresse suivante : https://hackernoon.com/the-easiest-way-to-understand-ethereum-for-5-years-olds-d8c537a7c7b8

SOLIDITY, 2019. *Introduction to Smart Contract.* [en ligne]. Mis à jour le 16 décembre 2019. [Consulté le 31 mai 2020]. Disponible à l'adresse suivante : https://solidity.readthedocs.io/en/v0.6.1/introduction-to-smart-contracts.html

CONSENSYS, 2019. *What is Ethereum ?.* [en ligne]. [Consulté le 31 mai 2020]. Disponible à l'adresse suivante : https://consensys.net/knowledge-base/about-ethereum-eth/

DANIEL [pseudonym], 2019. *Smart Contract Platforms: The Design & Its Limitations.* [en ligne]. [Consulté le 31 mai 2020]. Disponible à l'adresse suivante : https://komodoplatform.com/limitations-of-smart-contract-platforms/

COINHOUSE, [sans date]. *What is a token ?.* [en ligne]. [Consulté le 31 mai 2020]. Disponible à l'adresse suivante : https://www.coinhouse.com/learn/ethereum/what-is-a-token/

HERTIG, Ayssa, [2018]. *Will Ethereum Scale ?.* [en ligne]. Mis à jour le 23 février 2018. [Consulté le 31 mai 2020]. Disponible à l'adresse suivante : https://www.coindesk.com/learn/ethereum-101/will-ethereum-scale

FISHER, Daniel, 2008. *Company of the year: Nasdaq.* [en ligne]. [Consulté le 31 mai 2020]. Disponible à l'adresse suivante : https://www.forbes.com/forbes/2009/0112/056.html#7bf576467cc7

KIFFER, Lucianna, LEVIN, Dave, MISLOVE, Alan. [2018]. *Analyzing Ethereum's Contract Topology.* In 2018 Internet Measurement Conference (IMC '18) *Boston, MA, USA. ACM, New York, NY, USA, 6 pages.* [en ligne]. [Consulté le 31 mai 2020]. Disponible à l'adresse suivante : https://mislove.org/publications/Ethereum-IMC.pdf

FINLEY, Klint, 2016. *A $50 Million Hack Just Showed That the DAO Was All Too Human.* [en ligne]. [Consulté le 31 mai 2020]. Disponible à l'adresse suivante : https://www.wired.com/2016/06/50-million-hack-just-showed-dao-human/

MOU, Vallery, [sans date]. *Blockchain Oracles Explained.* [en ligne]. [Consulté le 31 mai 2020]. Disponible à l'adresse suivante : https://academy.binance.com/blockchain/blockchain-oracles-explained

HARDING [pseudonyme], 2016. *Payement channels.* [en ligne]. Mis à jour le 18 juillet 2019 [Consulté le 31 mai 2020]. Disponible à l'adresse suivante : https://en.bitcoin.it/w/index.php?title=Payment_channels&action=history

BLOCKBASIS, [sans date]. *What is a Decentralised Exchange (DEX)?* [en ligne]. [Consulté le 31 mai 2020]. Disponible à l'adresse suivante : https://blockbasis.com/en/blog/what-is-dex-decentralised-exchange/#pll_switcher

BUTERIN, Vitalik, 2018. *This sounds like there's some kind of scalability trilemma at play. What is this trilemma and can we break through it?* In : *Sharding FAQ.* [en ligne]. [Consulté le 31 mai 2020]. Mis à jour le 18 avril 2019 Disponible à l'adresse suivante : https://github.com/ethereum/wiki/wiki/Sharding-FAQ#this-sounds-like-theres-some-kind-of-scalability-trilemma-at-play-what-is-this-trilemma-and-can-we-break-through-it

LISK, [sans date]. *What is Blockchain.* [en ligne]. [Consulté le 31 mai 2020]. Disponible à l'adresse suivante : https://lisk.io/what-is-blockchain

GILBERT, Seth, LYNCH, Nancy A 2011. *Perspectives on the CAP Theorem.* [en ligne]. [Consulté le 31 mai 2020]. Disponible à l'adresse suivante : https://groups.csail.mit.edu/tds/papers/Gilbert/Brewer2.pdf

PIRAPIRA [pseudonyme], 2018. *EIPS.* [en ligne]. Mis à jour le 8 mars 2019. [Consulté le 31 mai 2020]. Disponible à l'adresse suivante : https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md

ZHANG, Erik, 2017. *NEP-5.* [en ligne]. Mis à jour le 28 juillet 2019. [Consulté le 31 mai 2020]. Disponible à l'adresse suivante : https://github.com/neo-project/proposals/blob/master/nep-5.mediawiki

FULLDECENT [pseudonyme], 2018. *EIP-721.* [en ligne]. Mis à jour le 2 avril 2020. [Consulté le 31 mai 2020]. Disponible à l'adresse suivante : https://github.com/ethereum/EIPs/blob/master/EIPS/eip-721.md

LIN, Lindsay X. 2019. *Deconstructing Decentralized Exchanges.* [en ligne]. [Consulté le 31 mai 2020]. Disponible à l'adresse suivante : https://stanford-jblp.pubpub.org/pub/deconstructing-dex/release/1

WARREN, Will, BANDEALI, Amir, 2017. 0x: An open protocol for decentralized exchange on the Ethereum blockchain. [en ligne]. [Consulté le 31 mai 2020]. Disponible à l'adresse suivante : https://assets.ctfassets.net/sdlntm3tthp6/3dQfrsgcj6CGG2I880omsy/346f2ff3281ee22690986d3c6bc44139/0x_white_paper.pdf

BENTOV, Iddo, BREIDENBACH, Lorenz, DAIAN, Phil, JUELS, Ari, LI, Yunqi, et ZHAO Xueyuan, 2017. *The Cost of Decentralization in 0x and EtherDelta.* [en ligne]. [Consulté le 31 mai 2020]. Disponible à l'adresse suivante : https://hackingdistributed.com/2017/08/13/cost-of-decent/

AURORA LABS, 2019. *IDEX: A Real-Time and High-Throughput*

Ethereum Smart Contract Exchange. [en ligne]. [Consulté le 31 mai 2020]. Disponible à l'adresse suivante : https://idex.market/static/IDEX-Whitepaper-V0.7.6.pdf

SCHROEDER, Stan, 2017. *Cryptocurrency exchange EtherDelta gets hacked and replaced by a fake site that steals your money.* [en ligne]. [Consulté le 31 mai 2020]. Disponible à l'adresse suivante : https://mashable.com/2017/12/21/etherdelta-hacked/#5MF0H2RevqqP

ConsenSys Diligence [pseudonyme], 2017. *General Findings* In : *0x review.* [en ligne]. [Consulté le 31 mai 2020]. Disponible à l'adresse suivante : https://github.com/ConsenSys/0x-review/blob/master/report/3_general_findings.md#front-running

HALFLEF [pseudonyme], 2019. *How to build a fast limit order book.* [en ligne]. [Consulté le 31 mai 2020]. Disponible à l'adresse suivante : https://gist.github.com/halfelf/db1ae032dc34278968f8bf31ee999a25/revisions

SCHROETER, Julien, 2014. *Limit Order Book reconstruction, visualisation and statistical analysis of order flow.* [en ligne]. [Consulté le 31 mai 2020]. Disponible à l'adresse suivante : https://ethz.ch/content/dam/ethz/special-interest/mtec/chair-of-entrepreneurial-risks-dam/documents/dissertation/master%20thesis/thesis_schroeter.pdf

TUTEJA, Sonal, [sans date]. *Complexity of different operations in Binary tree, Binary Search Tree and AVL tree.* [en ligne]. [Consulté le 31 mai 2020]. Disponible à l'adresse suivante : https://www.geeksforgeeks.org/complexity-different-operations-binary-tree-binary-search-tree-avl-tree/

VOGELSTELLER, Fabian, BUTERIN, Vitalik, 2015. *EIP 20: ERC-20 Token Standard.* [en ligne]. [Consulté le 31 mai 2020]. Disponible à l'adresse suivante : https://eips.ethereum.org/EIPS/eip-20

STABLECOIN WIKIPEDIA, 2018. *Stablecoin.* [en ligne]. Mis à jour le 10 mai 2020 [Consulté le 31 mai 2020]. Disponible à l'adresse suivante : https://en.wikipedia.org/wiki/Stablecoin

DIGIX, [sans date]. *Digix.* [en ligne]. [Consulté le 31 mai 2020]. Disponible à l'adresse suivante : https://digix.global/#/

MERKLE BLUE, [sans date]. *MINTABLE ERC20 TOKEN EXPLAINED.* [en ligne]. [Consulté le 31 mai 2020]. Disponible à l'adresse suivante : https://tokenmint.io/blog/mintable-erc20-token-explained.html

TAYVANO [pseudonyme], 2016. *How are ethereum addresses generated? Stackexchange.* [en ligne]. Mis à jour le 2 juillet 2017. [Consulté le 31 mai 2020]. Disponible à l'adresse suivante : https://ethereum.stackexchange.com/questions/3542/how-are-ethereum-addresses-generated

SOLIDITY, 2019b. Contract ABI Specification. [en ligne]. [Consulté le 31 mai 2020]. Disponible à l'adresse suivante : https://solidity.readthedocs.io/en/develop/abi-spec.html

FOWLER, Martin, 2005. *Event Sourcing.* [en ligne]. [Consulté le 31 mai 2020]. Disponible à l'adresse suivante : https://martinfowler.com/eaaDev/EventSourcing.html

RAIDEN, 2018. *Raiden Specification.* [en ligne]. [Consulté le 31 mai 2020]. Disponible à l'adresse suivante : https://raiden-network.readthedocs.io/en/v0.7.0/spec.html

MITCHEL, Cory, SCOTT, Gordon [reviewer] 2020. *Front-running. Investopedia.* [en ligne]. [Consulté le 31 mai 2020]. Disponible à l'adresse suivante : https://www.investopedia.com/terms/f/frontrunning.asp

CHEN, James, 2020. *Arbitrage. Investopedia.* [en ligne]. [Consulté le 31 mai 2020]. Disponible à l'adresse suivante : https://www.investopedia.com/terms/a/arbitrage.asp

EUREXCHANGE, [sans date]. *Matching Principles. Eurexchange.* [en ligne]. [Consulté le 31 mai 2020]. Disponible à l'adresse suivante : https://www.eurexchange.com/exchange-en/trading/order-book-trading/matching-principles

GHAZI, Tawfik, 2018. *A Matching Engine for our values: Part 1. Medium.* [en ligne]. [Consulté le 31 mai 2020]. Disponible à l'adresse suivante : https://medium.com/lgogroup/a-matching-engine-for-our-values-part-1-795a29b400fa

PAUW, Chrisjan, 2019. Sharding Explained. Cointelegraph. [en ligne]. [Consulté le 31 mai 2020]. Disponible à l'adresse suivante : https://cointelegraph.com/explained/sharding-explained

TREMBACK, Jehan, 2015. *Universal Payement Channels.* [en ligne]. [Consulté le 31 mai 2020]. Disponible à l'adresse suivante : https://blog.althea.net/universal-payment-channels/

BITCOIN, 2017. *Micropayement Channel. Bitcoin.org.* [en ligne]. Mis à jour le 7 décembre 2017. [Consulté le 31 mai 2020]. Disponible à l'adresse suivante : https://bitcoin.org/en/contracts-guide#micropayment-channel

COLEMAN, Jeff, 2015. *State Channels.* [en ligne]. [Consulté le 31 mai 2020]. Disponible à l'adresse suivante : https://www.jeffcoleman.ca/state-channels/

NEO, 2020. *Neo Whitepaper.* [en ligne]. [Consulté le 31 mai 2020]. Disponible à l'adresse suivante : https://docs.neo.org/docs/en-us/basic/whitepaper.html

NASH TEAM, 2019a. The heart of nash our off chain matching engine. Medium. [en ligne]. [Consulté le 31 mai 2020]. Disponible à l'adresse suivante : https://medium.com/nashsocial/the-heart-of-nash-our-off-chain-matching-engine-499cf2c23670

NASH TEAM, 2019b. *Connecting Nash to the blockchain: State channel smart contract. Medium.* [en ligne]. [Consulté le 31 mai 2020]. Disponible à l'adresse suivante : https://medium.com/nashsocial/connecting-nash-to-the-blockchain-state-channel-smart-contracts-da51d30b6226