UNIVERSITÉ NATIONALE DU VIETNAM À HANOÏ
INSTITUT FRANCOPHONE INTERNATIONAL

Promotion: 23

Option : Réseaux et Systèmes Communicants (RSC)

RAPPORT DE PROJET

Système de paiement sécurisé sur une plateforme de e-commerce basé sur la blockchain

Rédigé par:

Jean-Claude VITOFODJI & VIGAN Silas

Encadrant:
Ho Tuong Vinh

Année académique : 2019 - 2020

CONTENTS

INTRODUCTION GÉNÉRALE

Dans le but de dynamiser les interactions entre les différentes parties prenantes des transactions de tout genre entre diverses catégories d'entités (assurances, finances, commerce etc...), technologie dénommée Blockchain a été envisagée. La blockchain offre la possibilité de mieux gérer les ressources numériques de tout type(transferts de propriétés, contrat d'assurance, gestion des actifs), de manière sécurisée, transparente, sans organe central de gestion. La technologie est ce qu'on peut désigner, le registre distribué et sécurisé par cryptographie, dont les informations envoyées par les utilisateurs sont vérifiées groupées à intervalle de temps régulier en blocs et formant ainsi une chaîne. La technologie blockchain serait donc une chaîne de blocs et son application dans le monde et surtout dans le domaine de la finance ne cesse d'être sollicitée.
Notre travail concerne la sécurité des différentes transactions à grande échelle avec les applications decentralisé ecommerce. D'où notre thème : **Blockchain et crypto-monnaie appliqués à l'e-commerce**. La première partie de ce document énoncera le problème de recherche, l'état de l'art, puis nous présenterons la solution proposée pour faire le travail et présenterons ensuite la démarche de conception et d'implémentation de la solution proposée.

CHAPTER 1

ANALYSE DU PROJET

1.1 Contexte

Vu l'importance que revêt l'argent dans la société actuelle et avec la forte croissance du réseau de commerce international, nous assistons à l'intrusion des malfaiteurs et au manque de transparence dans les opérations. Nous devons donc recourir à des moyens sécurisés et rapides pour aider les acteurs de ce système à atteindre au mieux leurs objectifs et garantir tant aux acheteurs qu'aux vendeurs, une assurance de qualité et de confiance tout au long de la transaction.

De nos jours, nous assistons à l'essor de la digitalisation et des services numériques. C'est sans donc grande surprise que les services de commerce électronique ont pris une dimension bien plus grande que l'on n'espérait à une vitesse fulgurante. Aujourd'hui les achats et ventes s'étant dématérialisés, la blockchain grâce aux différents avantages qu'elle offre en terme de traçabilité et d'immuabilité des opérations s'avère un atout important pour aider dans la gestion efficace des opérations d'achat, vente, assurance livraison des produits commercialisés sur une plateforme.

L'objectif de notre projet est dans un premier temps d'aquérir les connaissances nécessaires sur la blockchain et dans un second temps, les appliquer à renforcer la sécurité des transactions sur un site e-commerce de la commande d'un article par l'acheteur jusqu'à la livraison en passant par la gestion des transactions.

1.2 Problématique

Une analyse du processus de commerce électronique nous a permis de mettre en évidence certains facteurs pouvant être améliorés grâce aux fonctionnalités offertes par la blockchain. En effet, sur le point de vue du paiement, cette étape au niveau des sites e-commerce requiert que l'utilisateur saisisse des données sensibles telles que: ces coordonnées bancaires,

son numéro de téléphone, son adresse. Il est à noter que plusieurs fois ces données ont été volées par des personnes malintentionnées à des buts malveillants. Ces moyens induisent également l'intervention de tiers pour gérer le paiement garantir une certaine assurance aux différentes parties. Cet état de chose à tort ou à raison, entraine des frais supplémentaires mais aussi un certain temps de vérification. Un autre problème souvent rencontré est celui lié à l'assurance de l'identité du client. Il est aussi un impératif de garantir une meilleure gestion des commandes en maitrisant les différents acteurs et leurs différents rôles dans le processus de commande et de livraison des produits.

Dans le souci d'assurer le suivi et la fiabilité de ses produits et lutter ainsi contre les contrefaçons et les arnaques en tous genres sur ses marketplaces, la blockchain fait ses premiers pas dans le eCommerce. Afin d'assurer directement entre le vendeur et l'acheteur une transaction fiable, la blockchain apporte une solution capable de rassurer toutes les parties prenantes d'un processus d'achat en ligne et de permettre à l'e-commerçant d'avoir accès à des données et ainsi prouver l'achat et renforcer la légitimité des prochains fake news pouvant toucher à l'image et la crédibilité des services offerts.

1.3 Objectifs

Ce travail a pour objectif de renforcer la sécurité dans une application e-commerce offrant aux usagers (commerçants, acheteurs, services de livraison) en garantisant, la traçabilité et la confiance grâce à l'unicité et l'immuabilité des informations stockées sur la blockchain. Pour ce faire, nous pouvons répartir les objectifs en deux parties: théorique et pratique.

1.3.1 Objectifs théorique

D'un point de vue théorique, nous avons pour objectif de:

- Etudier la notion de blockchain, ses principes et ses fonctionnalités grâce aux articles disponible sur internet et notions reçus
- Etat de l'art des différentes solutions de e-commerce
- Proposition de solution adaptée
- Etude conceptuelle et fonctionnelle
- Analyse des exigences
- Choix des technologies

1.3.2 Objectifs pratique

Comme objectif Pratique :

- Concevoir le système de commerce électronique simple qui s'intègre à Blockchain

- Valider les opérations d'achat, vente par la blockchain
- Implémenter le suivi et la traçabilité des différentes opérations grâce à la blockchain.

CHAPTER 2

RECHERCHE BIBLIOGRAPHIQUE

Introduction

Après avoir fait une analyse de notre sujet précédemment, nous allons dans cette étapes de notre travail présenter la notion de blockchain, et les outils necessaire qui contribue à sa mise en oeuvre.

2.1 Blockchain

2.1.1 Qu'est ce que c'est que la Blockchain ?

La Blockchain est un protocole informatique qui s'apparente à un registre publique, sécurisé et partagé entre les participants du réseau et certifié de façon cryptographique et en temps réel. Elle constitue une base de données qui enregistre de façon continuelle, toutes les opérations et transactions effectuées entre les utilisateurs. Elle est partagée sans intermédiaire, ce qui permet à chacun de vérifier la validité de la chaîne. Il en existe divers types : les blockchains publiques i.e. ouvertes à tous, et des blockchains privées dont l'accès et l'utilisation sont limitées à un certain nombre d'acteurs.

Partant du principe qu'il y est enregistré des transactions, notons que ces transactions sont regroupées dans des blocs avec des tailles variables. Un bloc est donc un ensemble de transactions mais aussi une somme de contrôle qu'on appelle le hash (cet élément est souvent utilisé comme l'identifiant du bloc), la somme de contrôle du bloc précédent et une mesure de la quantité de travail qui a été nécessaire pour produire le bloc. Lorsqu'un bloc atteint sa taille maximale, il est horodaté et ajouté ensuite à la chaîne de blocs d'où le nom Blockchain. Une fois ajouté à la chaîne, un bloc ne peut plus être ni modifié ni supprimé ce qui garantit l'authenticité et la sécurité du réseau.

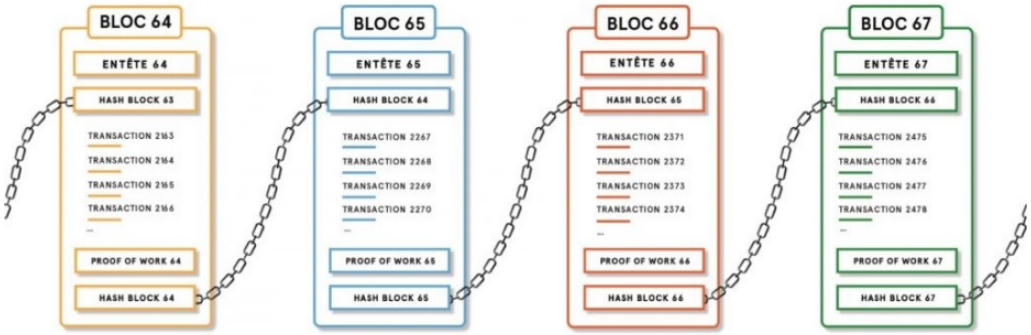


Figure 2.1: chaîne de blocs

2.2 Les technologies Blockchain

2.2.1 Blockchain 1.0 (Bitcoins)

Bitcoin est une monnaie numérique. Il a été créé en 2009 par une entité anonyme utilisant le nom Satoshi Nakamoto. Les paiements utilisant les bitcoins sont enregistrés dans un grand livre public stocké sur des ordinateurs connectés au réseau bitcoin. Le grand livre peut être consulté à tout moment sur Internet. Bitcoin est la première et la plus grande crypto-monnaie décentralisée, tandis que les autres monnaies numériques incluent : Altcoin, Litecoin et Dogecoin. Les utilisateurs peuvent envoyer et recevoir des Bitcoins par voie électronique moyennant des frais de transaction optionnels (ou très minimes) à l'aide d'un portefeuille (un logiciel sur un ordinateur personnel, une application mobile ou une application Web). En réponse à ces transactions, de nouveaux bitcoins sont créés en guise de récompense pour le traitement informatique (appelé extraction), qui est utilisé pour vérifier et enregistrer les transactions en bitcoins dans le grand livre.

2.2.2 Blockchain 2.0 (Ethereum)

Ethereum est un projet open source construit par les développeurs du monde entier, semblable au protocole Bitcoin, mais beaucoup plus adaptable et flexible, car il permet aux utilisateurs de créer et d'utiliser les applications décentralisées exécutées sur la technologie de blockchain sous-jacente. Généralement, il existe deux approches pour créer une application blockchain: démarrer un réseau indépendant ou établir un protocole sur Bitcoin. Ethereum a introduit une blockchain avec un langage de programmation intégré complet de Turing qui permet à quiconque d'écrire des contrats intelligents et des applications décentralisées avec des règles et des paramètres personnalisés.

Les sujets	Bitcoin	Ethereum
Concept	Monnaie numérique	Contrats intelligents
Fondateur	Satoshi Nakamoto	Vitalik Buterin
Méthode de libération	Genesis Block Mined	Prévente
Crypto-monnaie utilisée	Bitcoin (Satoshi)	Éther
Algorithme	SHA-256	Ethash
Minage des Blocs	10 minutes	12-14 secondes
Évolutif	Pas encore	Oui

Figure 2.2: comparaison bitcoin et ethereum

2.3 Quelques travaux sur la technologie Blockchain basées sur le Bigcommerce

De nos recherches, nous avons trouvé plusieurs travaux de la technologies blockchain pour les bigcommerce. Pour dire simple, deux des plates-formes de commerce électronique les plus populaires, les plus faciles à utiliser et disponibles sont le **Omnitude**, **WooCommerce** et **Shopify** .

2.3.1 Omnitude

En tant que gateway basée sur le Blockchain au service de l'e-commerce, l'omnitude est une plateforme qui relie les technologies de la blockchain, les plateformes d'e-commerce et les systèmes des entreprises afin de construire des chaînes d'approvisionnement de bout en bout. Pour palier aux différents problèmes rencontrés dans le marché du e-commerce, le omnitude agit en tant que middleware ce qui permet une intégration simple des technologies de la Blockchain dans les systèmes existants et qui aura pour conséquence de transformer leurs capacités.
Cette intégration permet alors:

- la création de chaînes de production et de fabrication transparentes et responsables;
- une réduction significative de la fraude;
- l'activation d'une identité client unique utilisable sur tout site de commerce électronique connecté à Omnitude;
- intégration de la Blockchain avec les systèmes d'entreprise tels que ERP et WMS.

L'architecture Hyperledger fournit des services de blockchain de base, sur lesquels l'écosystème Omnitude est construit sera composé d'une variété d'entités : les commerçants, les clients, les fournisseurs, les affiliés / référents,etc.

2.3.2 PaySpacelv

PaySpacelv est un fournisseur de services de paiement spécialisé dans le traitement des affaires à haut risque. Compte tenu du fait que ce type d'entreprise préfère accepter les paiements en bitcoin, PaySpacelv a obtenu son expertise en la matière. Pour soutenir les commerçants, PaySpacelv propose un logiciel de prévention de la fraude à la pointe de la technologie. Et les analyses avancées aident à mieux connaître le comportement et les habitudes des clients.

PaySpacelv est un processeur de paiement qui assure une portée internationale à ses marchands. Le règlement des fonds est disponible en crypto-monnaies ou en euros. Le règlement s'effectue via Swift ou le paiement par crypto-monnaie. Les commerçants peuvent intégrer la passerelle via HPP.

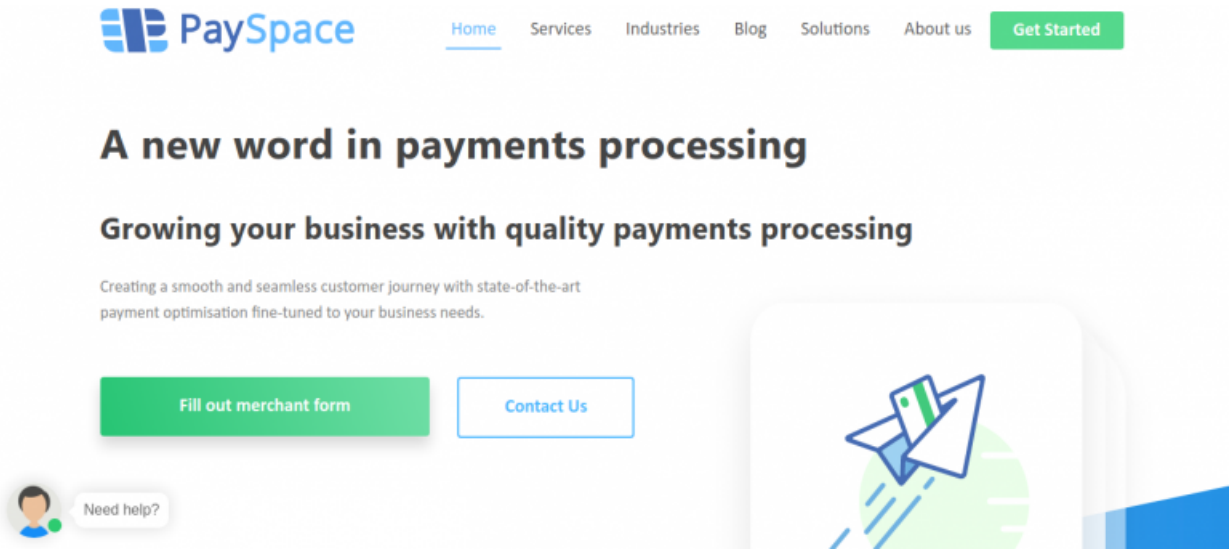


Figure 2.3: PaySpacelv Payment Gateway

2.3.3 GoCoin

GoCoin est un autre fournisseur de services de paiement Bitcoin que vous pouvez considérer. Il fonctionne depuis 2013. Ils offrent une passerelle de paiement mondiale sécurisée et efficace. La société croit en un brillant avenir du commerce blockchain. Ainsi, il aide volontiers les commerçants à accepter une variété de crypto-monnaies. GoCoin permet à une variété d'industries telles que les voyages, l'hébergement, les jeux, etc.

GoCoin offre une protection contre les risques transactionnels pour protéger les fonds du commerçant. Les propriétaires de sites Web peuvent intégrer la passerelle de paiement via HPP. Ainsi, des plugins pour la plateforme comme WooCommerce, Shopify, Magento,

Prestashop, sont également disponibles. Vend une variété de fonctionnalités en ce qui concerne le tableau de bord.

Figure 2.4: interface de la solution woocommerce

2.3.4 Comparaison solution existante

Ci-dessous une comparaison des différents plateformes définies précédemment:

Table 2.1: Tableau de comparaison des plateformes de bigcommerce

Noms	**Avantages**	**Insuffisances**
PaySpacelv	• Excellent systeme de prevention de la fraude • Systeme d'analyses avancées aidant à mieux connaître le comportement et les habitudes des clients.	• Prise en charge uniquement du bitcoin.
GoCoin	• passerelle de paiement mondiale sécurisée et efficace. • Accepte une variété de crypto-monnaies. • Dispose de plugins pour les plateformes: WooCommerce, Shopify, Magento, Prestashop.	• Repose uniquement sur les crypto-monnaies pour le paiement.
Omnitude	• permet aux commerçants de lutter contre le vol d'identité, qui est la forme la plus courante de fraude • l'utilisation de la Blockchain va permettre aux commerçant de se libérer de leur obligation de stocker de façon sécurisée les données personnelles des utilisateurs	• ne fait pas intervenir de façon visible les acteurs secondaires qui sont tout aussi important dans les opérations entrant dans la chaine de bloc .

CHAPTER 3

SOLUTION PROPOSÉE

3.1 Introduction

Le chapitre précédant nous a permis de nous familiariser avec la notion de blockchain et de comprendre ce qu'est la monnaie électronique. Nous allons dans ce chapitre faire l'analyse fonctionnel et technique du système ainsi que la conception du dit système.

3.2 Environement materiel et logiciel

Pour notre implémentation, nous avons travaillé avec:

- Truffle v5.1.39 (core: 5.1.39);
- Solidity v0.5.16 (solc-js);
- Node v12.18.3;
- Web3.js v1.2.1

3.3 Architecture et description du flux de la transaction proposé

Dans le cadre de notre travail, nous nous concentrons plus sur l'aspect de la sécurisation des données personnelles des clients lors des opération de paiement.

Fonctionnement:

Notre système va fonctionner selon le flow qui suit:

- L'utilisateur s'enrégistre et fourni toutes ses informations personnelles
- Les informations du client sont hashées et enregistrées sur la blockchain

- Le client reçoit donc une invitation a faire certifier son appareil

- un certificat est genéré pour identifier les trusted devices du client.

- lors d'une operation d'achat, la commande de l'utilisateur est stockée sur la blockchain

- l'utilisateur saisi le code et le système fait recours au hash sur la blockchain contenant les informations bancaires de l'utilisateur et procède à la validation de l'opération qui génère alors une écriture sur la Blockchain

- La commande est donc transmise au livreur afin qu'il le délivre au client final

3.4 Modèle de Conception architecturale

3.4.1 Diagramme des cas d'utilisation

Les roles des diagrammes de cas d'utilisation sont de recueillir, d'analyser et d'organiser les besoins, ainsi que de recenser les grandes fonctionnalités. IL s'agit donc de la premiere étape UML pour la conception du système:

- les acteurs du projet

Le visiteur : c'est un individu qui est entrain de fouiller sur le net cherchant, cherchant un produit pour l'acheter ou pour avoir une idée sur les modèles et les prix. Jusqu'a ce stade c'est un utilisateur inconnu donc il n'est pas encore le client.
Le client : cet acteur est un visiteur ayant déja crée un compte sur notre site. Il peut donc suivre le processus d'achat des produits en toute sécurité sachant que notre système doit etre l'unique responsable de la confidentialité des données personnelles de ces clients.
L'administrateur : pour les sites web ont l'appelle généralement le webmaster. C'est celui qui assure le dynamisme du site et veille sur les mises à jour des produits, de leurs prix, de leur disponibilités, dela gestion des paiements et de la gestion des livraisons.

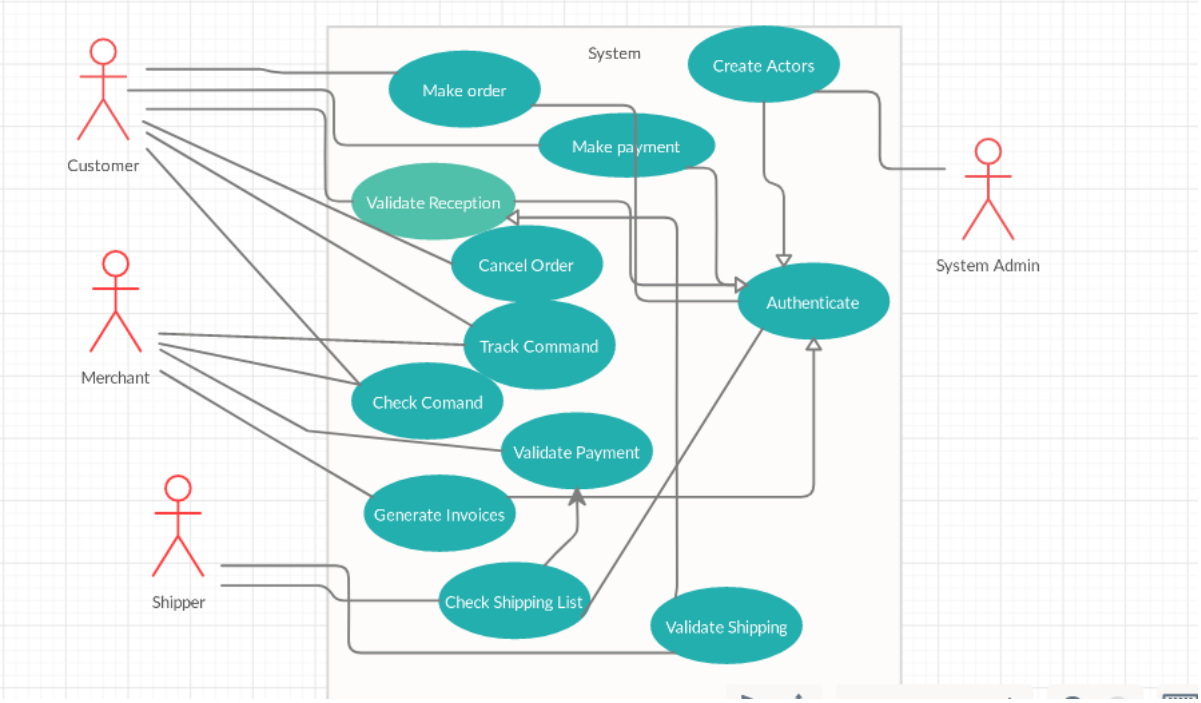


Figure 3.1: Global System Use case Diagram

3.4.2 Diagramme de classe

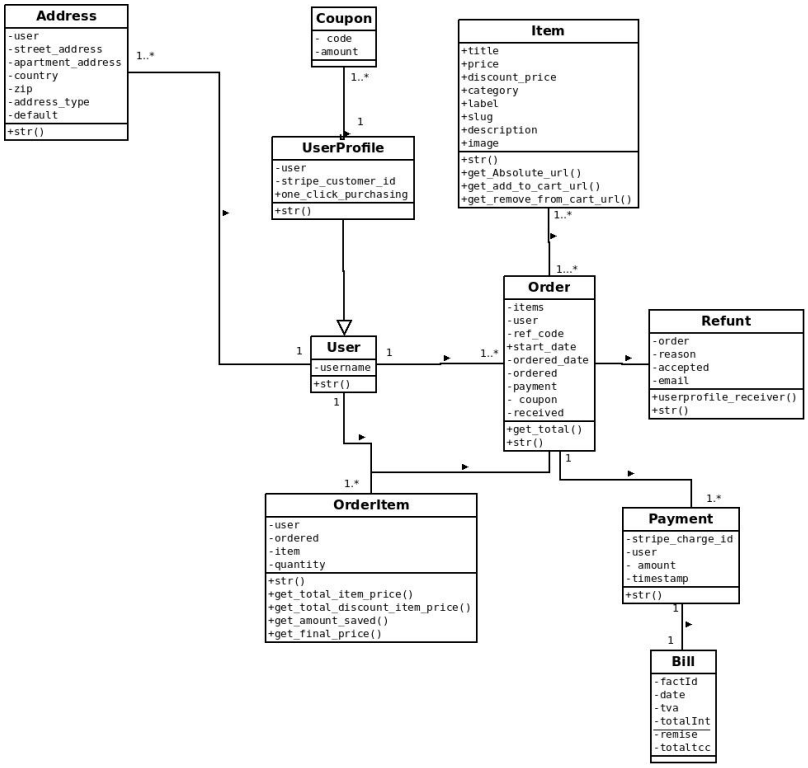


Figure 3.2: Diagramme des classes

Le système est constitué de quatre classes principales, une classe Utilisateur qui sera la classe mère de deux classes filles (Administrateur et client) qu'on n'a pas decrit ici, une classe Item et orderItem pour gerer les produits (c'est a dire les différentes produits

mise en service pour des clients). Il y a une spécificité qu'il faut souligner au niveau de la classe payment, faciliter les tests de validation pour chaque achat, permettre le suivi et rendre souplexe la traçabilité du système.

3.5 Conception de la blockchain

3.5.1 Choix de la technologie blockchain: Ethereum

La truffe, comme le disent les développeurs, est le "Your Ethereum Swiss Army Knife". C'est un couteau pliant qui peut faire beaucoup de choses. Nous avons pour choix de creer un environnement dans lequel nous pouvons développer plus facilement des DApps qui fonctionnent avec Ethereum. Truffle est un framework de développement très populaire car Truffle a de nombreuses fonctionnalités intéressantes et est facile à utiliser.

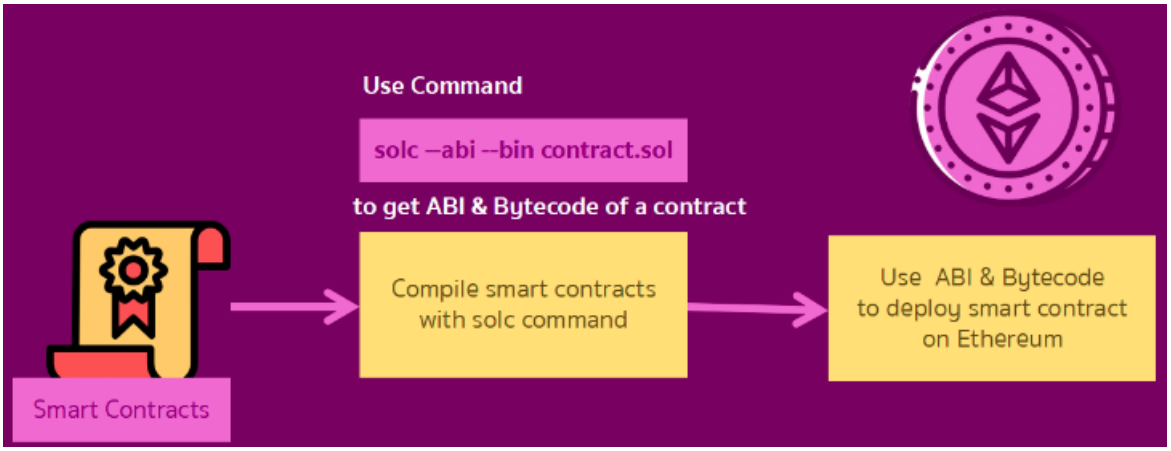


Figure 3.3: Architecture de déploiement du système

3.5.2 Architecture interne de la plateforme de Blockchain

L'image ci dessous nous présente le mécanisme de fonctionnement interne de la blockchain.

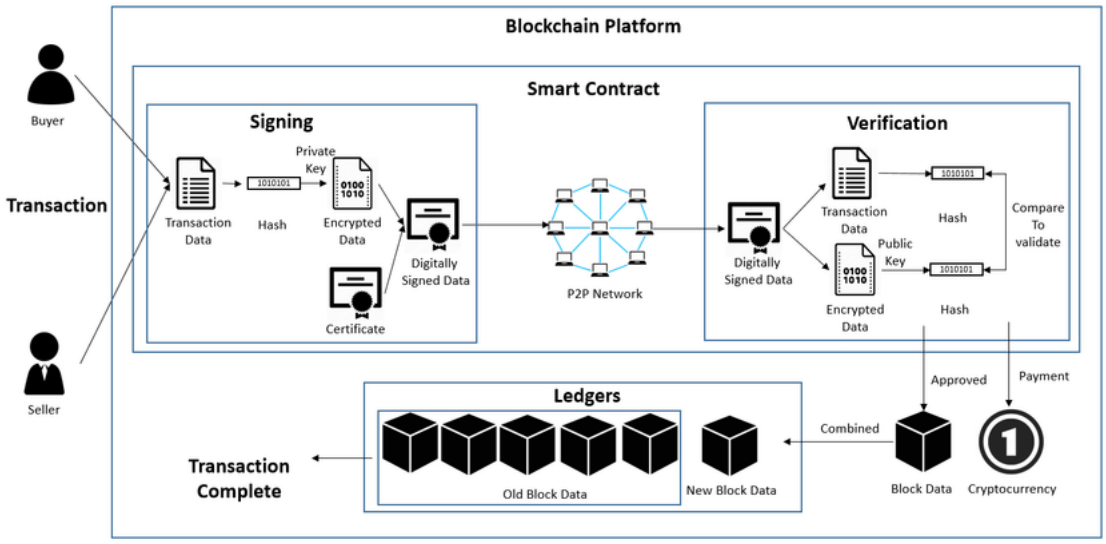


Figure 3.4: Fonctionnement interne de la Blockchain

3.5.3 Plateforme de réalisation et Smart contracts

En fonction des exigences de sécurité et de célérité, nous avons décidé d'utiliser la plateforme Ethereum dont truffle pour la réalisation de la blockchain dans le cadre de ce travail.

Les systèmes de commerce électronique doivent impliquer de l'argent. Par conséquent, nous créons une pièce à utiliser dans notre commerce électronique.

CHAPTER 4

RÉSULTATS ET DISCUSSION

4.1 Introduction

Cette partie dénombre la présentation des scénarios applicatifs de l'application. Nous allons présenter dans ce qui suit, les imprimes-écran des realisations de notre système.

4.2 Implementation du projet

4.2.1 Smart Contracts

créons une pièce à utiliser dans notre commerce électronique puis dans le même dossier un contrat Shop.sol pour notre système de commerce électronique.

- **Token.sol**

contracts > Token.sol

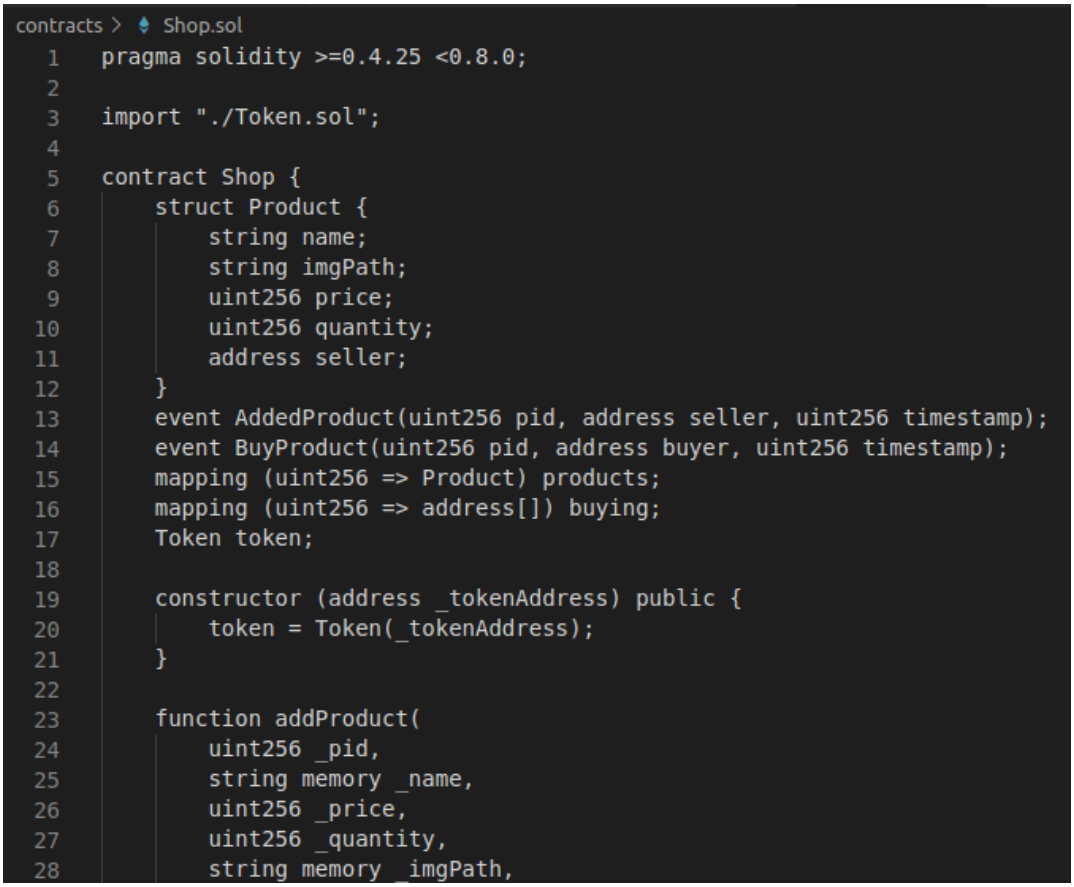
```
1  pragma solidity >=0.4.25 <0.8.0;
2
3  contract Token {
4      mapping (address => uint256) public balanceOf;
5
6      constructor(uint256 initialSupply) public {
7          balanceOf[msg.sender] = initialSupply;
8      }
9
10     function transfer(address _from, address _to, uint256 _value) public returns (bool success)
11         require(balanceOf[_from] >= _value);               // Check if the sender has enough
12         require(balanceOf[_to] + _value >= balanceOf[_to]); // Check for overflows
13         balanceOf[_from] -= _value;
14         balanceOf[_to] += _value;
15         return true;
16     }
17 }
```

Figure 4.1: contract Token.sol

Le code est une génération de pièces très simple, il se compose d'une variable balanceOf et d'une fonction de transfert pour stocker le nombre de pièces de chaque compte et envoyer

des pièces d'un compte à un autre dans l'ordre, juste pour pouvoir les envoyer dans les deux sens.

- **Shop.sol**



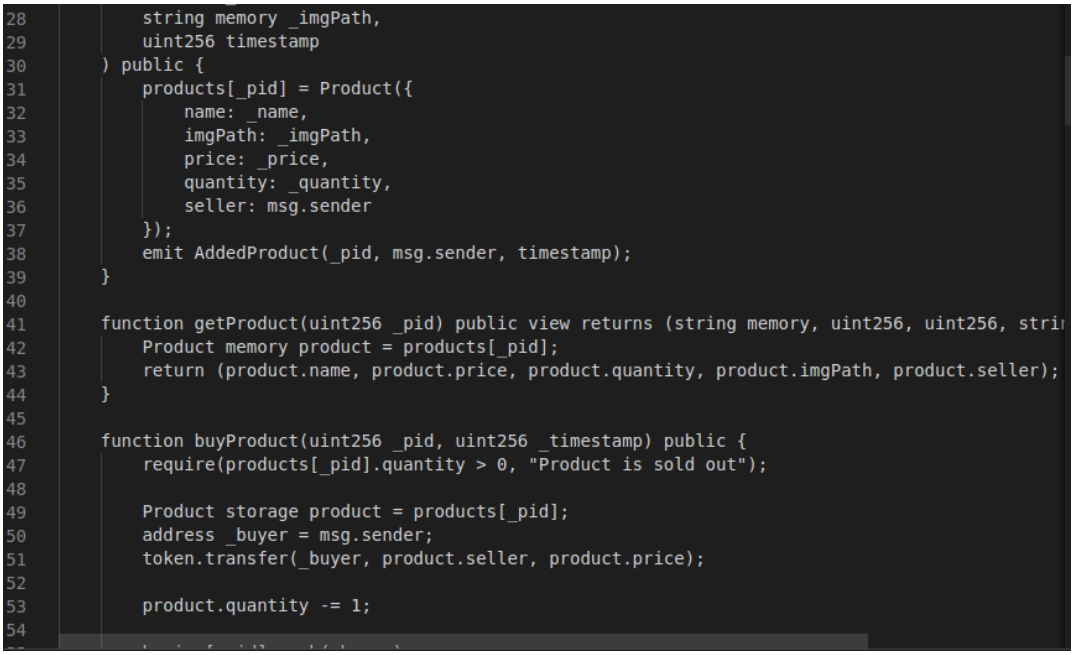
```
contracts > Shop.sol
pragma solidity >=0.4.25 <0.8.0;

import "./Token.sol";

contract Shop {
    struct Product {
        string name;
        string imgPath;
        uint256 price;
        uint256 quantity;
        address seller;
    }
    event AddedProduct(uint256 pid, address seller, uint256 timestamp);
    event BuyProduct(uint256 pid, address buyer, uint256 timestamp);
    mapping (uint256 => Product) products;
    mapping (uint256 => address[]) buying;
    Token token;

    constructor (address _tokenAddress) public {
        token = Token(_tokenAddress);
    }

    function addProduct(
        uint256 _pid,
        string memory _name,
        uint256 _price,
        uint256 _quantity,
        string memory _imgPath,
```



```
        string memory _imgPath,
        uint256 timestamp
    ) public {
        products[_pid] = Product({
            name: _name,
            imgPath: _imgPath,
            price: _price,
            quantity: _quantity,
            seller: msg.sender
        });
        emit AddedProduct(_pid, msg.sender, timestamp);
    }

    function getProduct(uint256 _pid) public view returns (string memory, uint256, uint256, stri
        Product memory product = products[_pid];
        return (product.name, product.price, product.quantity, product.imgPath, product.seller);
    }

    function buyProduct(uint256 _pid, uint256 _timestamp) public {
        require(products[_pid].quantity > 0, "Product is sold out");

        Product storage product = products[_pid];
        address _buyer = msg.sender;
        token.transfer(_buyer, product.seller, product.price);

        product.quantity -= 1;
```

Figure 4.2: contract Shop.sol

Le Smart Contract Shop ci-dessus se compose de 3 fonctions:

1. Fonction d'ajout de produit à la base de données;

2. Obtenez une fonction de description de produit;

3. Fonction de commande.

- **Deploiement des Smarts contract**

Depuis **1_initial_migration.js** On voit que la méthode de Déployer dans Truffle est très simple, il s'agit de charger l'artefact Truffle obtenu en compilant pour le déployer à l'aide de la commande deployer.deploy (...). Nous voyons que le script que nous allons écrire maintenant est beaucoup plus court.
Pour Deploy Smart Contract que nous avons écrit, vous avons écrit Migration comme suit.

migrations > JS 2_deploy_contracts.js > ...

```
1   const Token = artifacts.require('Token');
2   const Shop = artifacts.require('Shop');
3
4   module.exports = function(deployer, network, accounts) {
5     deployer
6       .deploy(Token, 1000000)
7       .then(async () => {
8         const tokenContract = await Token.deployed();
9         return deployer.deploy(Shop, tokenContract.address);
10      })
11
12      .then(async () => {
13          const token = await Token.deployed();
14          const coinbase = accounts[0];
15          const value = 50000;
16          await token.transfer(coinbase, accounts[1], value);
17      });
18  };
```

Figure 4.3: migrate 2deploycontracts

4.2.2 Connectez les réseaux Ethereum

Avant de déployer, nous devons nous assurer que Nos machines sont désormais connectées à Ethereum. Dans notre projet, nous nous sommes connectés à Ethereum qui est un réseau privé (en référence à notre Blockchain locale. Ou la machine que nous configurons uniquement). Truffle a également un programme pour créer un autre programme appelé Ganache. Une fois ouvert, vous aurez Blockchain prêt à être utilisé. Sans avoir à s'asseoir et à écrire ne serait-ce qu'une seule commande.
Nous telechargeons et installons ganache. Ouvrons simplement le programme et cliquons sur Quickstart si vous suivez comme dans l'image ci-dessus.

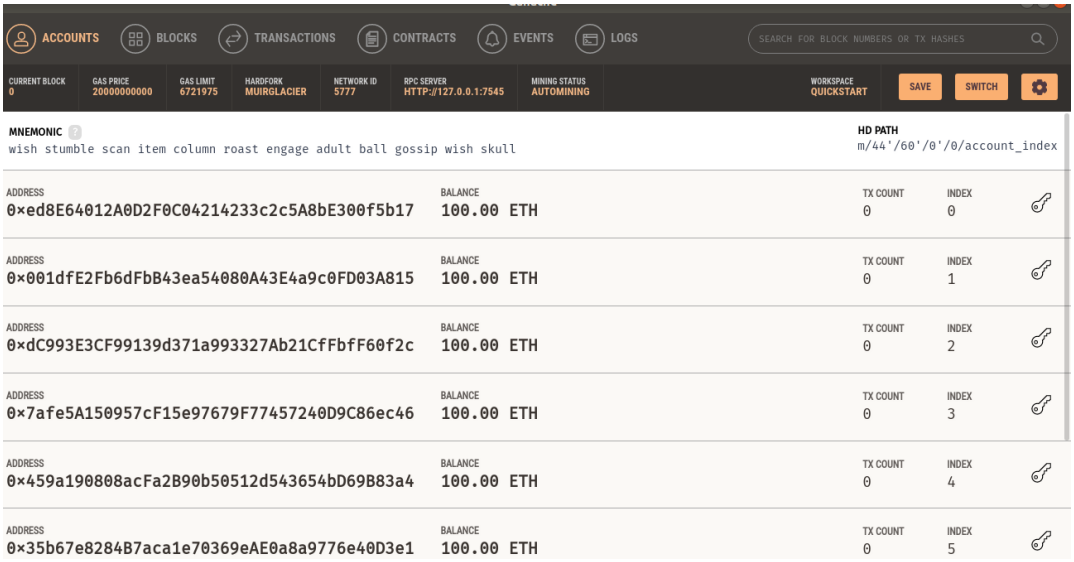


Figure 4.4: blockchain crée

La Blockchain est crée avec succes et attention à ne pas la fermez pendant le déploiement.

4.2.3 Configuration du reseau Ethereum

Après qu'on soit connecté avec succès à Ethereum, nous devons faire savoir à Truffle à quel réseau se connecter en éditant le fichier **truffle-config.js**. Dans la section réseaux, nous pouvons définir l'exemple de Truffle cite dans le fichier. Voici le réseau développement crée comme suit:

```
module.exports = {

  networks: {
    // Useful for testing. The `development` name is special - truffle uses
    // if it's defined here and no other network is specified at the comman
    // You should run a client (like ganache-cli, geth or parity) in a sepa
    // tab if you use this network and you must also set the `host`, `port`
    // options below to some value.
    development: {
      host: "127.0.0.1",     // Localhost (default: none)
      port: 8545,            // Standard Ethereum port (default: none)
      network_id: "*",       // Any network (default: none)
     },
     mainnet: {
      host: "127.0.0.1",
      port:  8545,
      network_id: 1,
     },
     testnet: {
      host: "127.0.0.1",
      port:  8545,
      network_id: 3,
     },
  },
```

Figure 4.5: config ethereum

4.3 Interface utilisateur

Ouvrez le navigateur Web et accédez à l'URL http://localhost:3000.

- **Ajout des produits**

Ici, les différents produits sont ajoutés dans le système, leur prix et aussi la quantié disponible.

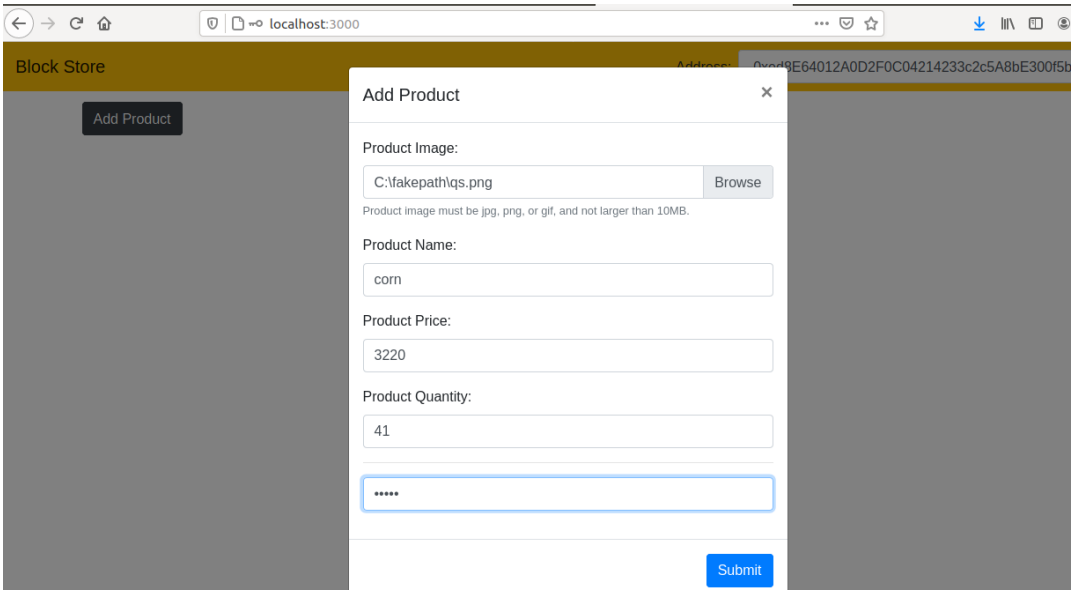


Figure 4.6: ajout produits

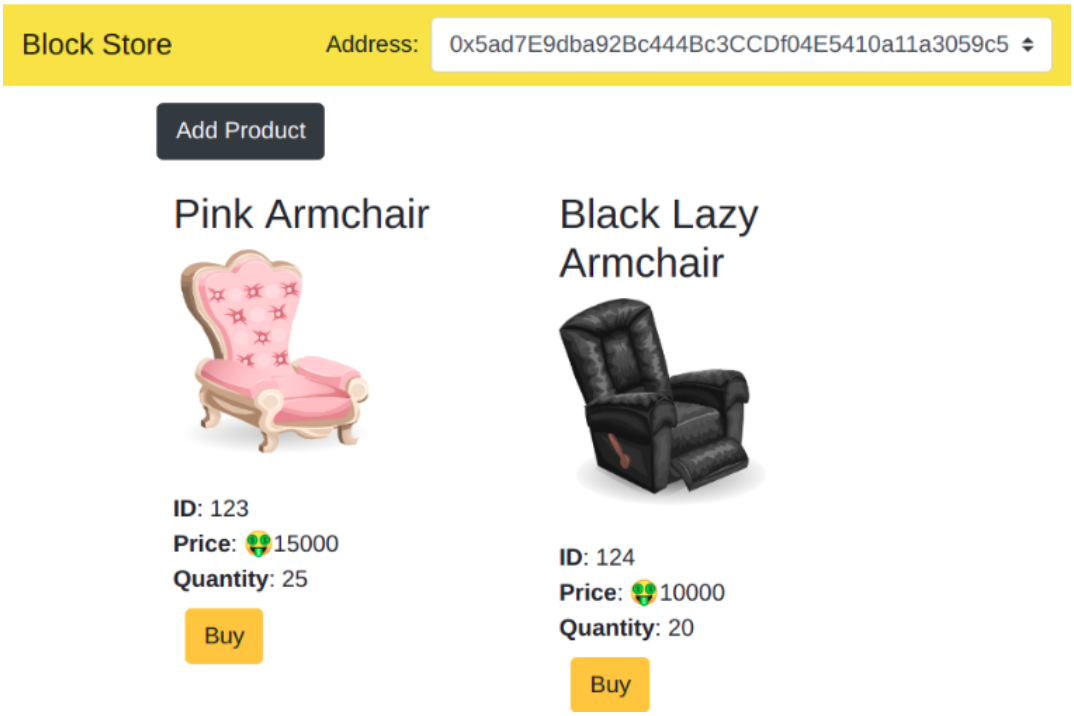


Figure 4.7: produits ajoutés

Chaque produit ajoutés à la blockchain sont hachés à ce qu'on ne puise les modifier à l'avenir.

- **Achat des produits**

L'acheteur peut donc acheter un ou plusieurs produits en selectionnant le bouton buy du produit à acheter puis en entrant son password qui pointe sur son adresse pour pourvoir facilité le paiement sur le site.

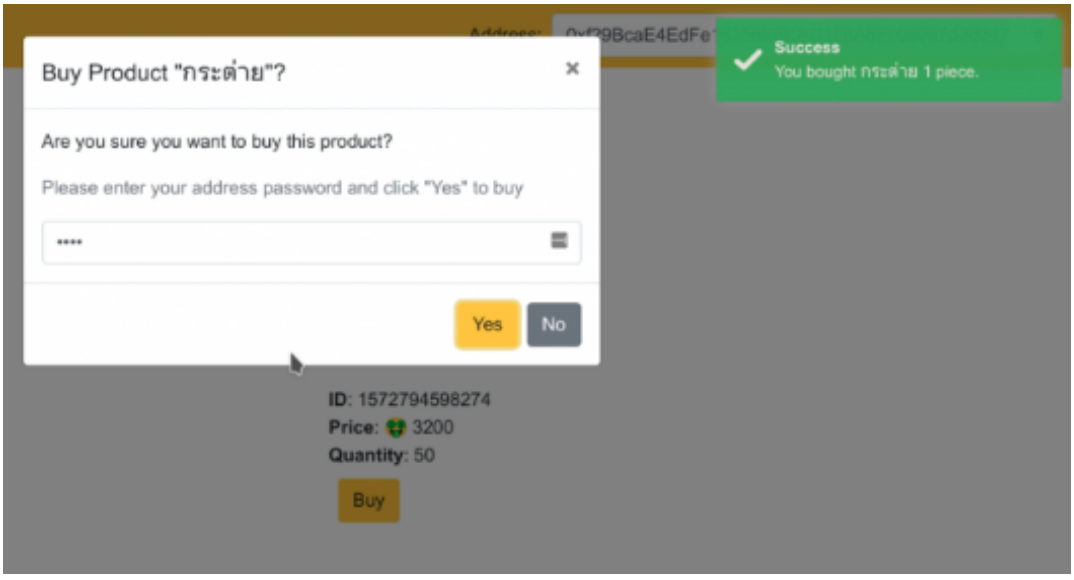


Figure 4.8: achats produit

4.3.1 Tableau Recapitulatif

Ci-dessous un recapitulatif des resultats:

Table 4.1: Tableau recapitulatif

Noms	Fonction ou utilisation
Javascript ou NodeJs	• Dans les applications créées avec JavaScript ou Node.js, nous pouvons communiquer avec Ethereum en utilisant un package nommé web3 après un contact réussi. • Nous pouvons récupérer des informations telles que Compte (web3.eth.accounts()), etc., y compris la création de transactions, l'envoi d'argent et des informations dans les deux sens entre Compte (web3.eth.sendTransaction()).
Smart Contract	• Nous utilisons ici un package appelé @truffle/contract(TruffleContract) pour aider à créer une instance Contract, qui est un objet qui nous permet également d'appeler des fonctions comme annoncé dans Smart Contract. • Ce package nous permet de créer facilement des Instances. Parce qu'il ne nécessite qu'une seule donnée: l'artefact Truffle.
Appel foncton	• l'appel de fonctions dans le smart contract qui ne récupèrent que des données et l'envoi de transaction avec des fonctions qui modifient les données dans le système. S'il ne fonctionne pas correctement Obtiendra des résultats qui ne sont pas ce que vous voulez • En appelant une fonction d'édition d'appel, les données du contrat intelligent ne seront pas modifiées. Et appeler la fonction d'extraction Send Transaction entraînera Transaction, mais nous ne pourrons pas du tout recevoir la valeur.
TruffleContract	• Nous aide également à récupérer automatiquement l'adresse du contrat intelligent de l'artefact truffe. • Dans le cas où nous avons plus d'un contrat de déploiement sur Blockchain, nous pouvons choisir Réseau pour TruffleContract récupére l'adresse correctement en envoyant le fournisseur à partir de la web3fonction.contractInstance.setProvider()

Conclusion

En somme, nous avons essayé de montrer les différents concepts de la blockchain, puis l'utiliser dans le secteur de l'ecommerce. Bien que la technologie de la blockchain fait son apparition peut 'a peut dans le domaine du commerce en ligne, comme toute nouvelle technologie, elle est une idee qui perturbe initialement, et au fil du temps, elle pourrait favoriser le développement d'un ecosystème plus vaste qui comprend à la fois l'ancienne et la nouvelle innovation. Certains exemples historiques sont que l'avenement du commerce en ligne a conduit à une augmentation des ventes et une meilleure tracabilite des achats. La technologie de la blockchain permet don cd' etablir des relations de confiance entre toutes les parties. Decentralisé et inaltérable, elle reduit les risques de litiges concernant les paiements et les details d'unecommande.

Perspectives

Le système mis en place est loin d'avoir la prétention d'être parfait, en science il y a toujours quelque chose à apporter. Dans un avenir très proches, nous comptons utiliser un cas pratique et concret grâce à notre système. Il s'agit de concevoir une application ecommerce qui affiche au niveau de la plateforme toute les transactions effectuer et ceux par ordre croissante ordre.

Bibliographie

- https://www.forbes.fr/technologie/blockchain-13-applications-qui-vont-tout-changer/
- https://www.activestate.com/blog/how-to-build-a-blockchain-in-python/
- https://blockchain.fish/make-dapp-with-truffle-ch-4/
- https://executive-education-online.mit.edu/presentations/lp/mit-blockchain-technologies-online-short-course/
- http://perso.univ-lyon1.fr/jean-patrick.gelas/edu/UE_Blockchain/
- ://cognitiveclass.ai/courses/blockchain-course
- https://cognitiveclass.ai/courses/ibm-blockchain-foundation-dev
- https://cognitiveclass.ai/learn/blockchain-for-developers
- https://www.blockchain.com/fr/
- https://fr.wikipedia.org/wiki/Ethereum
- https://ethereum.org/en/

CHAPTER 5

ANNEXES

5.1 Annexe1 : Contenu des fichiers de configuration et du deploiement

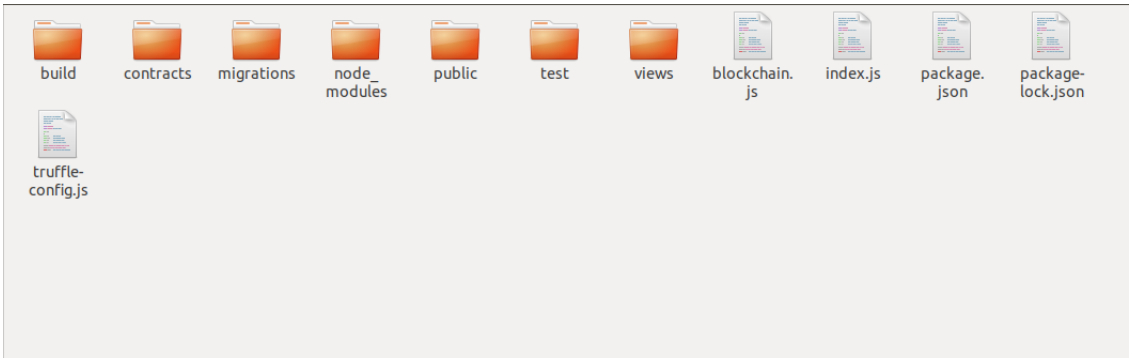


Figure 5.1: Fichiers implementation

5.2 Annexe2 : Deploiement de la blockchain ganache

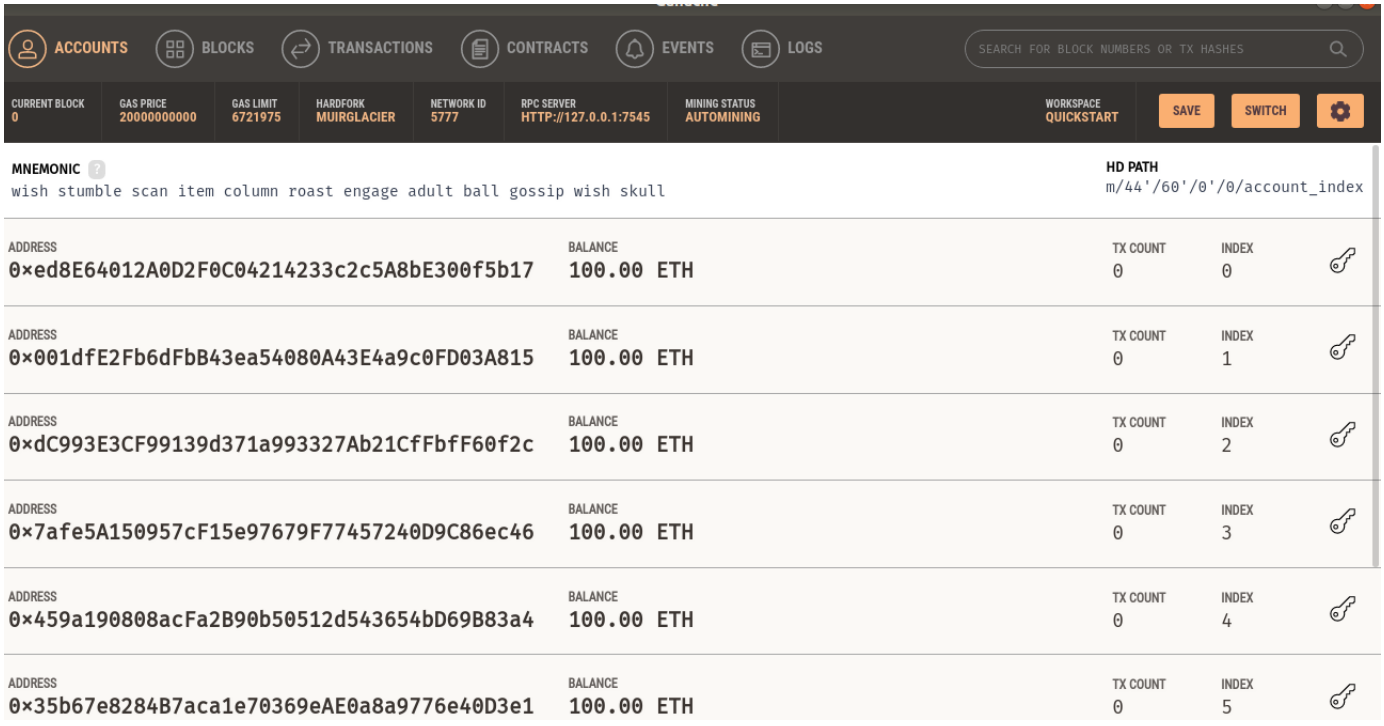


Figure 5.2: Ganache blockchain

5.3 Annexe3 : Données d'événements et d'objets produits

```
(base) jc@jc-HP-635-Notebook-PC:~/truffle_ecommerce_contr$ npm start

> truffle_ecommerce_contr@1.0.0 start /home/jc/truffle_ecommerce_contr
> node index.js

Ecommerce server is running! it is listening on port 3000...
TruffleContract {
  constructor: [Function: TruffleContract] {
    _constructorMethods: {
      configureNetwork: [Function: configureNetwork],
      setProvider: [Function: setProvider],
      new: [Function: new],
      at: [AsyncFunction: at],
      deployed: [AsyncFunction: deployed],
      defaults: [Function: defaults],
      hasNetwork: [Function: hasNetwork],
      isDeployed: [Function: isDeployed],
      detectNetwork: [AsyncFunction: detectNetwork],
      setNetwork: [Function: setNetwork],
      setNetworkType: [Function: setNetworkType],
      setWallet: [Function: setWallet],
      resetAddress: [Function: resetAddress],
      link: [Function: link],
      clone: [Function: clone],
      addProp: [Function: addProp],
      toJSON: [Function: toJSON],
      decodeLogs: [Function: decodeLogs]
    },
    _properties: {
      contract_name: [Object],
      contractName: [Object],
      gasMultiplier: [Object],
      timeoutBlocks: [Object],
      autoGas: [Object],
      numberFormat: [Object],
      abi: [Object],
      metadata: [Function: metadata],
      network: [Function: network],
```

Figure 5.3: Backend Start application

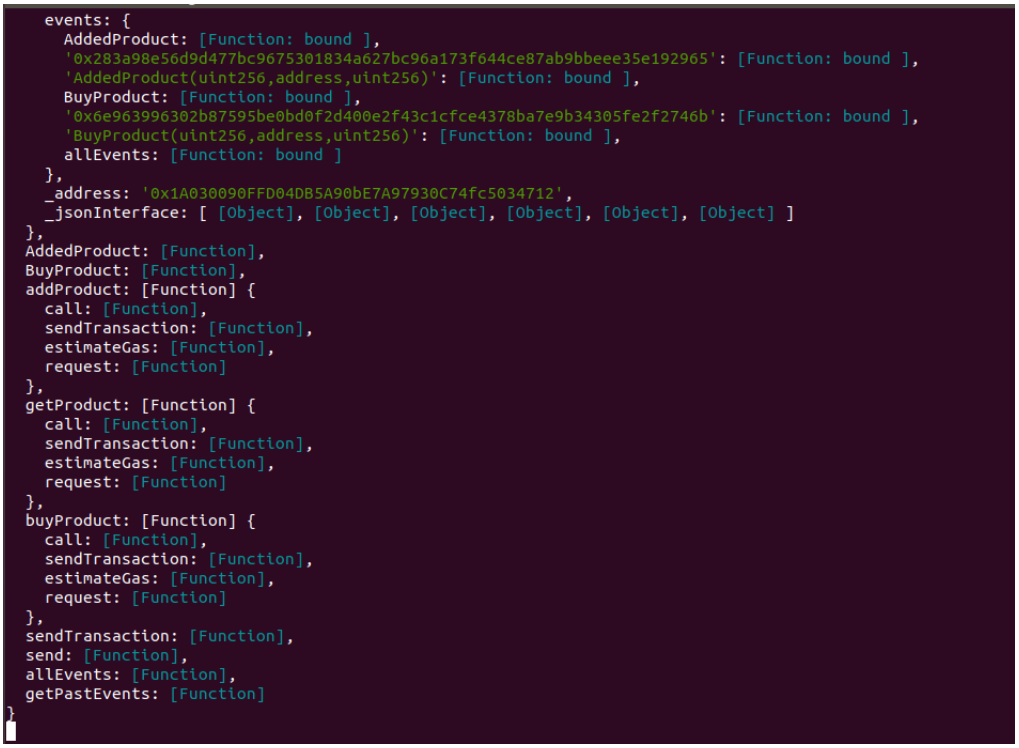


Figure 5.4: Definition des clefs et produits par le système