Université de Tunis El Manar

Faculté des Sciences de Tunis

RAPPORT DE STAGE

Elaboré par :

Ben Kedim Mohamed

Sujet : Implémentation d'une Blockchain pour l'optimisation de la Traçabilité des Produits

---

Organisme d'accueil :

I.T Grow

Superviseur du stage :

Arif Sami

Période :

Début : 15/07/2023 Fin :15/09/2023

Table de matières

Remerciements :

Je tiens à exprimer ma sincère gratitude envers l'équipe d'IT Grow et son CEO, M. Sami Arif, pour leur accueil chaleureux, leur encouragement constant, et leur précieuse contribution à mon développement professionnel. Leur expertise, leur créativité, et leur engagement envers l'excellence ont enrichi mon expérience de stage de manière significative, et je suis reconnaissant d'avoir eu l'opportunité de contribuer à l'excellence digitale de cette entreprise innovante.

Introduction générale

Selon plusieurs études, l'émergence de la pandémie de COVID-19 a mis en évidence l'importance cruciale de la gestion efficace de la chaîne d'approvisionnement. Les perturbations massives dans la distribution de produits essentiels, les retards dans les livraisons et l'incertitude quant à l'origine des produits ont montré à quel point la transparence et la traçabilité de la chaîne d'approvisionnement étaient essentielles pour garantir la continuité des opérations et la sécurité des consommateurs.

Dans ce contexte, de plus en plus d'entreprises ont commencé à explorer des technologies avancées telles que la blockchain pour améliorer leur gestion de la chaîne d'approvisionnement. Un exemple notable est le projet "Walmart Food Safety Solution".

L'apprentissage de la blockchain est devenu essentiel de nos jours, car elle représente une pierre angulaire du futur numérique, souvent considérée comme la base du Web 3.0. La blockchain offre des solutions innovantes pour résoudre des problèmes complexes, allant de la traçabilité dans la chaîne d'approvisionnement à la sécurisation des transactions financières et à la préservation de la vie privée des utilisateurs.

Dans ce projet, j'ai cherché à développer un système de blockchain d'infrastructure simple dans le but de simuler à quoi peut ressembler un système de suivi dans un exemple plus vaste.

- Le premier chapitre expose la technologie blockchain, son utilisation dans le domaine du supply-chain, et pourquoi l'intégration du blockchain en supply-chain.

- Le deuxième chapitre prend en compte un aperçu sur le projet de traçabilité.

- Le troisième chapitre présentera pratiquement l'intégration de l'idée de traçabilité en blockchain, et l'algorithmique du système.

- Le quatrième donnera une idée sur la partie JavaScript dans quelques pages web développées, et sur la partie de l'utilisation de la Framework Flask.

CHAPITRE 1: ÉTUDE PRÉLIMINAIRE

1. Introduction

Ce chapitre vise à mettre dans son cadre général l'idée du projet on va introduire les principales informations pour montrer l'importance du choix du sujet de ce projet.

2. Présentation de l'entreprise

Pendant mon stage au sein de la startup IT Grow, j'ai eu l'opportunité de plonger au cœur d'une culture d'entreprise véritablement exceptionnelle. IT Grow transcende le simple concept de prestation de services pour incarner une véritable philosophie, redéfinissant les relations entre client et prestataire en offrant un parcours personnalisé, optimisé et efficient. En tant que stagiaire au sein de cette entreprise innovante, j'ai pu observer comment IT Grow croit en chaque rêve, en chaque entrepreneur, et en chaque entreprise, et existe pour les accompagner dans leur transformation, éliminant ainsi les obstacles qui limitent leur croissance. Dans un monde où les limites semblent apparentes, IT Grow envisage un horizon d'opportunités illimitées. L'entreprise ne se contente pas d'offrir un simple service, mais tisse des partenariats, bâtit des connexions et insuffle le changement. L'engagement d'IT Grow va bien au-delà de la technologie et de l'innovation, il réside dans l'esprit humain, la volonté d'exceller, et le courage de poursuivre l'extraordinaire. Mon expérience au sein de cette entreprise m'a permis de comprendre que chez IT Grow, l'excellence digitale est bien plus qu'un objectif, c'est une passion intemporelle.

I.T Grow

3. Contexte et motivation

Le contexte et la motivation pour ce projet sont profondément enracinés dans la nécessité croissante de transparence et de visibilité dans le monde des affaires et de l'alimentation. Ces impératifs imposent une nouvelle stratégie visant à établir et à renforcer la confiance entre les producteurs et les consommateurs, tout en répondant à un besoin essentiel de rapidité et d'efficacité pour rendre les transactions et les services plus rentables.

L'exigence de rapidité et d'efficacité est devenue un élément clé dans la manière dont les entreprises mènent leurs activités aujourd'hui. Dans un environnement commercial en constante évolution, les entreprises sont confrontées à des défis pour rester compétitives. Les attentes des consommateurs en matière de rapidité de livraison, de suivi des produits et de réponses aux problèmes sont les majeurs concernes. Pour prospérer dans cet environnement, il est impératif de mettre en place des processus qui sont à la fois rapides et efficaces.

4. Objectifs et Portée du Stage

L'objectif du stage, en premier lieu, est de permettre une immersion profonde dans les aspects novateurs de la blockchain, en mettant l'accent sur la compréhension des algorithmes prises en charge lors du parcours, aussi de se familiariser avec cette technologie prometteuse.

Au cours du stage, l'accent a été mis sur la construction d'une infrastructure blockchain de base. Cette infrastructure constitue la fondation sur laquelle repose la capacité à répondre à la question cruciale de la traçabilité des produits. Et pour effectuer ce dernier point il fallait une compréhension des principaux titres de la supply-chain comme un métier à lequel se base la blockchain conçu.

Puisque le développement d'une solution de toute une chaine d'approvisionnement est aussi compliqué et demande beaucoup de connaissances métier, on a mis l'accent sur la traçabilité des produit en utilisant un QR code unique.

La transparence de la traçabilité est le point le plus important pour que les consommateurs prennent confiance en constructeurs ou producteurs dans le cas général.

L'objectif est donc de combiner les connaissances acquises dans la blockchain et la supply-chain pour pouvoir rependre au question de la transparence.

Chapitre 2 : Description du Système de Traçabilité à base blockchain

Partie 1 : La blockchain : la technologie prometteuse

1. Présentation de la Technologie Blockchain

a- Vue général :

En termes simples, une blockchain est un type de registre distribué ou de base de données décentralisée qui maintient en permanence des enregistrements numériques de la propriété des actifs. Au lieu d'avoir un administrateur central comme une base de données traditionnelle (pensez aux banques, aux gouvernements et aux comptables), un registre distribué fonctionne avec un réseau de bases de données répliquées, synchronisées via Internet et visibles par tous les membres du réseau. Les réseaux blockchain peuvent être privés avec une adhésion restreinte, similaire à un intranet, ou publics, comme Internet, accessibles par toute personne dans le monde.

b- Transaction :

Une transaction signifie un transfert de données entre deux nœuds de la blockchain, dans notre cas, une transaction contient principalement les adresses publiques de l'émetteur et du récepteur, la signature de la transaction, le Product Unicode, et le hashed privacy.
La sémantique du transaction c'est le transfert physique d'un produit d'une partie à une autre dans la blockchain. Donc on effectue une transaction quand un produit est transféré d'une partie à une autre.

c- Bloc :

Un bloc est une unité de base dans une blockchain. Il s'agit d'une structure de données qui contient un ensemble de transactions ou d'informations liées qui sont enregistrées de manière permanente dans la blockchain.

d- Spécificité

La nature décentralisée, ouverte et cryptographique de la blockchain permet aux individus de se faire mutuellement confiance et de réaliser des transactions de pair à pair, éliminant ainsi le besoin d'intermédiaires. Cela offre également d'énormes avantages en termes de sécurité. Les attaques de piratage qui touchent fréquemment les intermédiaires centralisés tels que les banques seraient pratiquement impossibles à réaliser sur la blockchain.

e- Advantages du blockchain:

1. Un contrôle total sur la valeur que vous possédez ; il n'y a pas de tiers qui détient votre valeur ou qui peut limiter votre accès à celle-ci.
2. Le coût pour effectuer une transaction est très faible.
3. La valeur peut être transférée en quelques minutes.
4. Tout le monde peut à tout moment vérifier chaque transaction effectuée sur la blockchain, ce qui assure une transparence totale.

2. La chaine d’approvisionnement :

a- Vue général :

Une chaîne d'approvisionnement, également appelée chaîne logistique, est un réseau complexe d'organisations, de ressources, d'activités et d'informations impliquées dans le processus de production, de distribution et de livraison de biens ou de services à partir du point de fabrication ou de production jusqu'au consommateur final. Elle englobe toutes les étapes nécessaires pour mettre un produit ou un service à disposition du marché.

Une chaîne d'approvisionnement typique peut inclure les éléments suivants :

- **Fournisseurs** : Les entreprises ou les individus qui fournissent les matières premières, les composants ou les produits nécessaires à la fabrication d'un produit.
- **Fabricants** : Les entreprises qui transforment les matières premières en produits finis.
- **Distributeurs** : Les entreprises ou les canaux de distribution qui stockent, transportent et livrent les produits aux points de vente ou directement aux consommateurs.
- **Points de ventes** : Les points de vente où les produits sont vendus aux consommateurs finaux.
- **Logistique et Transport** : Les services de transport et de gestion des stocks qui permettent de déplacer les produits tout au long de la chaîne.
- **Gestion de l'Information** : Les systèmes et les technologies utilisés pour suivre, gérer et partager des informations essentielles sur les produits, les commandes, les inventaires, etc.

b- Paysage actuel de la chaîne d’approvisionnement :

Les chaînes d'approvisionnement sont souvent vulnérables à toute une série de facteurs, notamment les tensions géopolitiques, les cyberattaques, l'inflation, les sécheresses qui perturbent le transport en abaissant le niveau de l'eau, les ruptures de stocks de produits essentiels, ainsi que les nombreux effets imprévus du réchauffement climatique.
Face à toutes ces perturbations, de nombreuses entreprises, ainsi que les responsables de l'efficacité de la chaîne d'approvisionnement, repensent leurs stratégies axées sur la simplicité et la planification en temps réel, ainsi que les problématiques liées aux processus et aux systèmes de source, de fabrication, de livraison et de retour. De plus, les dirigeants de la chaîne d'approvisionnement sont de plus en plus tenus de prévoir et de prévenir de manière proactive les vulnérabilités de la chaîne d'approvisionnement. Pour cette raison, ces dirigeants concentrent leurs investissements stratégiques sur trois principaux moteurs de l'efficacité :

- **Prédire les risques** de la chaîne d'approvisionnement.
- Permettre **le suivi grâce à la traçabilité** de la chaîne d'approvisionnement.
- **Renforcer la confiance** dans un environnement complexe impliquant de multiples parties prenantes.

3. Comment la blockchain peut aider ?

Il est important de se rappeler que les chaînes d'approvisionnement sont, fondamentalement, un réseau de sociétés interconnectées. Dans ce réseau, chaque entreprise ajoute de la valeur à un produit ou à un service avant qu'il n'atteigne l'utilisateur final. Cet échange et cette accumulation de valeur sont enregistrés à travers une série de transactions, ou flux, d'informations, de biens, de services et de finances. Une "blockchain forte" offre la possibilité d'enregistrer ces transactions (à la fois physiques et virtuelles) sur un registre partagé et immuable, ce qui permet la capture, la validation et le partage de données entre ces sociétés interconnectées.

- Amélioration de la visibilité :

Le problème : De nombreuses entreprises éprouvent des difficultés en matière de visibilité et de transparence de bout en bout au sein de leurs chaînes d'approvisionnement.

Intégration du blockchain : En mettant en place une chaîne d'approvisionnement basée sur la blockchain, les entreprises peuvent efficacement numériser les actifs physiques et créer un registre décentralisé et immuable de toutes les transactions à travers l'ensemble du processus de création de valeur de bout en bout.

- Amélioration de confiance :

Le problème : Dans une grande chaîne d'approvisionnement mondiale qui englobe de nombreuses parties prenantes, les entreprises principales peuvent avoir du mal à faire confiance à leurs partenaires. La visibilité et la confiance diminuent rapidement pour la plupart des entreprises, même après la deuxième couche de relations.

Intégration du blockchain : En enregistrant toutes les transactions de la chaîne d'approvisionnement sur un registre partagé et immuable, la blockchain offre un niveau de confiance qui était auparavant impossible à atteindre.

Partie 2 : Système de Traçabilité

a- Introduction :

Commençant le pratique du projet, j'ai vécu des problèmes fondamentales en reliant l'esprit de la supply-chain avec celui du blockchain, le premier problème que j'ai confronté été l'algorithme de consensus qui assure l'acceptation de la transaction et l'ajout d'un nouveau bloc, on ne peut pas utiliser le même principe que cela du Bitcoin ou des applications financières dans le cas général. Il me fallait un algorithme de consensus dans une blockchain hybride (où tout le monde peut lire, mais que des personne connus peuvent écrire)

b- Problématique et résolution :

- L'algorithme de consensus :
  - Définition : L'algorithme de consensus est un protocole ou un ensemble de règles utilisés dans les réseaux informatiques décentralisés, tels que les blockchains, pour parvenir à un accord sur l'état du système ou sur une transaction donnée. L'objectif principal de l'algorithme de consensus est de garantir que toutes les parties du réseau parviennent à un consensus, c'est-à-dire qu'elles acceptent et valident la même version de la réalité, même en présence de défaillances ou de comportements malveillants.

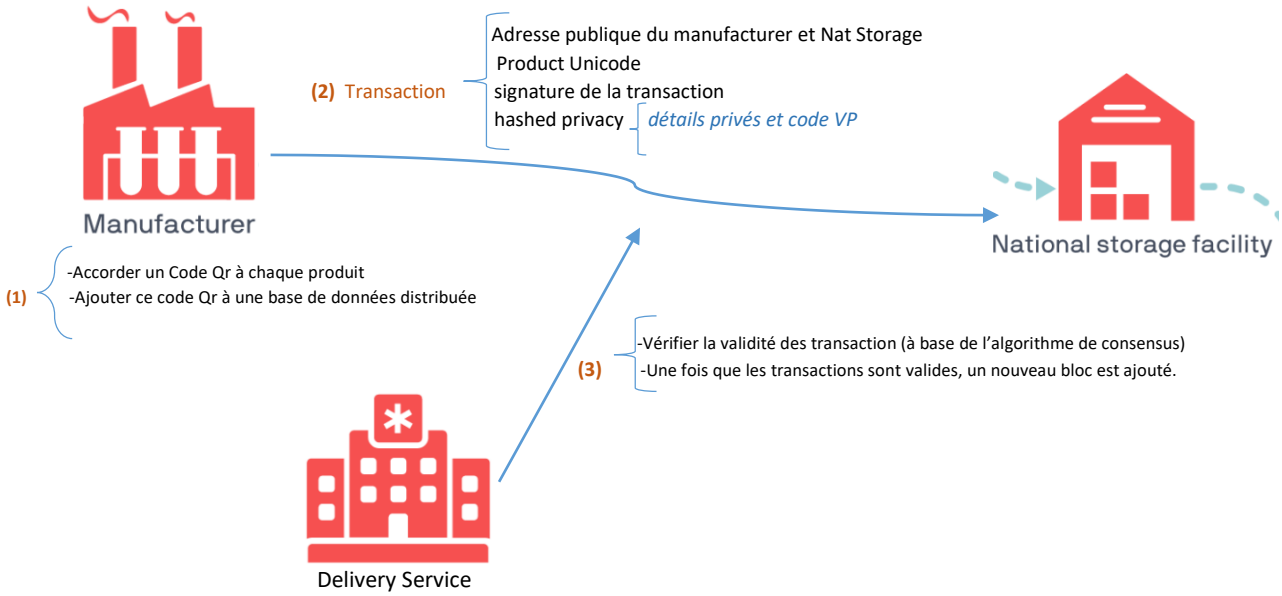
  - Problématique : Après avoir basculer partout pour trouver un algorithme qui satisfait l'esprit de la traçabilité j'ai fini par créer un que je pense appropriée et qui prend en charge ces principes :
    1- Seuls les participants connus et autorisés ont le droit d'ajouter un bloc.
    2- Toute personne autorisée peut lire la blockchain à tout moment afin d'accroître la transparence
    3-La confirmation sur le bloc et les transactions doit être plus souple dans un environnement fermé donc on a dû développer un autre algorithme de consensus.
    "Les applications de la blockchain réussies pour les chaînes d'approvisionnement nécessiteront de nouvelles blockchains permissionnées, de nouvelles normes pour représenter les transactions sur un bloc, et de nouvelles règles pour gouverner le système, qui sont toutes en cours de développement à différents stades." *Harvard Business Review article in 2020*

  - Solution : l'algorithme qu'on a mis en œuvre vise à vérifier la crédibilité de la partie qui effectue la transaction (seuls les parties de la supply-chain ont le droit).
    'Proof of privacy' : la partie qui confirme la transaction en vrai va faire un test sur les données privées de la partie qui a effectué la transaction (tout se fait avec la cryptographie et en backend).
    L'input 'hashed_privacy' dans la plateforme sert à effectuer ça.
    Après la confirmation, la valeur VP dans les données privées du 'Sender' va changer et va être broadcasté vers toutes les nœuds.
    La valeur VP (verified privacy) sert à assurer que si une personne malveillante a eu accès aux données privés et former le hashed_privacy nécessaire pour effectuer une transaction

d’un nœud de la supply-chain, alors elle ne peut pas car cette valeur change toujours d’une façon arbitraire. Chaque nœud a son propre VP.

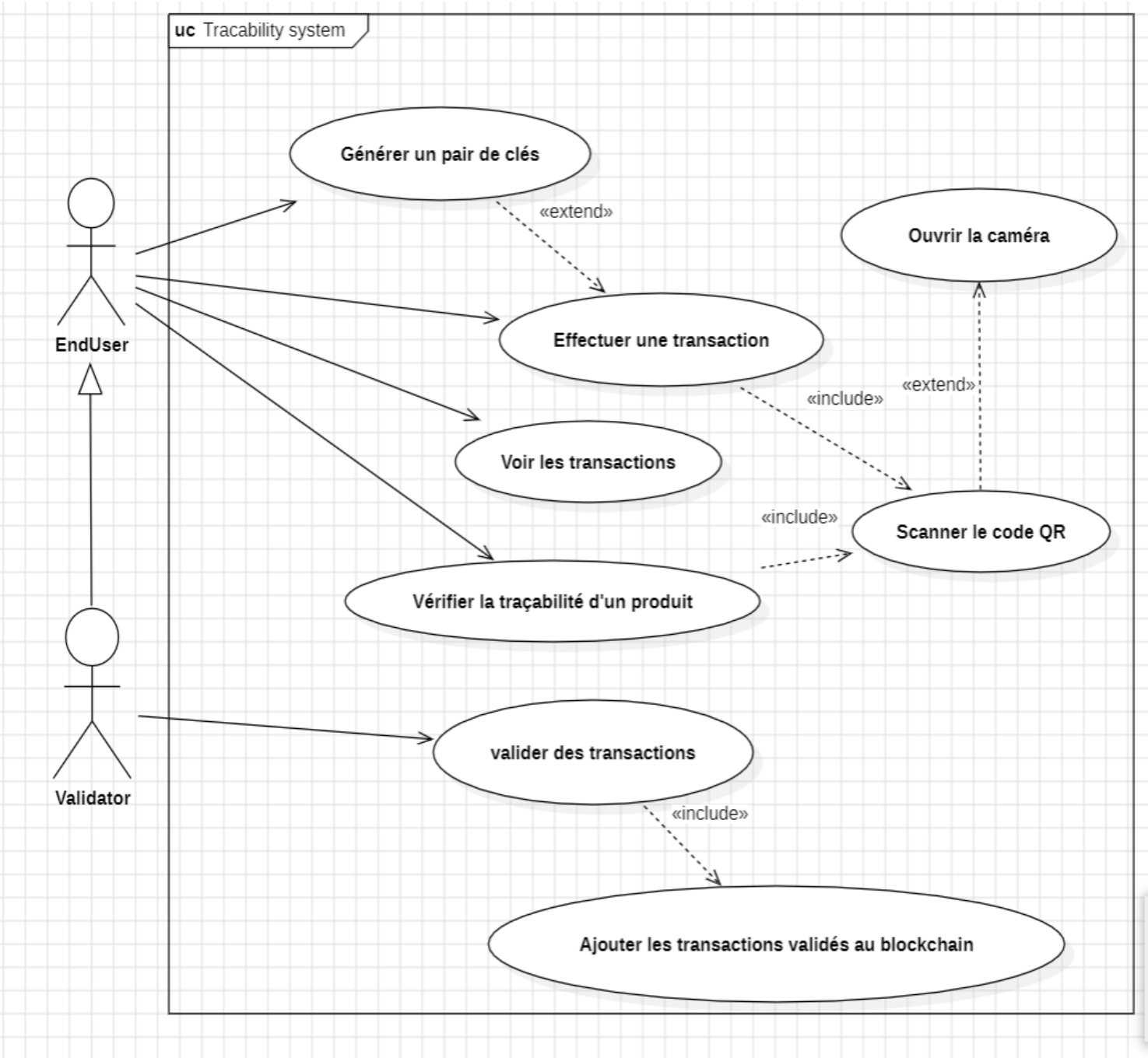
- Le suivi des produits :
  - Problématique :
    Comment assurer la traçabilité des produits pour que la chaine d’approvisionnement atteint ses objectifs d’un côté et la visibilité pour tous les end-user d’un autre côté.
  - Solution :
    On accorde à chaque produit un code unique dans la chaine, la première partie de la chaine y est responsable, après à chaque déplacement du produit on effectue une transaction et cette transaction représente le déplacement physique du produit.

c- Figure explicative :



Chapitre 3 : Implémentation du système de Traçabilité à base blockchain

1. Diagramme UML : Conception général du système de traçabilité :



2. Vue basique du développement du blockchain :

```
class Blockchain:

    def __init__(self):

        self.transactions = []
        self.chain = []
        self.nodes = set()
        self.node_id ="Man-ID1100"
        self.create_block('00')

    def verify_transaction_signature(self, sender_address, signature, transaction):
    def submit_transaction(self, sender_address, recipient_address, value,privacy, signature):
    def create_block(self, previous_hash):
    def proof_of_privacy(self,transaction):
    def valid_proof(self):
    def valid_chain(self, chain):
    def resolve_conflicts(self):
```

La blockchain comme on l'a décrit dans notre système est un class contenant ces attributs :

- Self.transactions : Liste qui contient toutes les transactions mis en attente pour la confirmation et l'ajout dans un block.
- Self.chain : la chaine des blocks
- Self.nodes : la liste des nœuds équivaut à mineurs dans le bitcoin.
- Self.node_id : identifient unique du nœud se composant de deux champs :
    1) Type de participant : Man veut dire 'manifacturer ', STR (store), SHP(shop), DLV(delivery)
    2) Identifiant unique selon le type commençant par ID.

Les méthodes utilisés pour la manipulation de la blockchain sont principalement :

- Verify_transaction_signature : permet de vérifier la signature de la transaction.
- Submit_transaction : permet de soumettre une transaction dans la blockchain.
- Create_block : permet de créer un block.
- Proof_of_privacy : cette méthode implémente l'algorithme de consensus pris en compte.
- Valid_proof : permet de vérifier la validité des transactions.
- Valid_chain : cela vérifie la validité de toute la blockchain.
- Resolve_conflicts : fait à tolérer contre les pannes possibles.

3. Vue basique du développement des transactions :

```python
class Transaction:

    def __init__(self, sender_address, sender_private_key, recipient_address, Product_UniCode):
        self.sender_address = sender_address
        self.sender_private_key = sender_private_key
        self.recipient_address = recipient_address
        self.Product_UniCode = Product_UniCode

    def sign_transaction(self):
    def get_sender_details_base64(self):
```

La transaction dans notre système est caractérisée par ces attributs :

- Sender_adress : adresse publique de celui qui effectue la transaction.
- Sender_private_key : adresse privée de celui qui effectue la transaction
- Recipient_adress : adresse public de celui qui va recevoir la transaction
- Product_unicode : identifiant unique du code de produit.

Les méthodes utilisées au sein du class transaction sont les suivantes :

- Sign_transaction : permet de signer une transaction.
- Get_sender_details_base64 : permet d'extraire les données d'un fichier local, ces données sont l'adresse privée et publique de l'utilisateur.

4. Génération de Portefeuilles pour les acteurs de la Supply Chain :

➢ Backend :

```python

@app.route('/wallet/new', methods=['GET'])
def new_wallet():
    random_gen = Crypto.Random.new().read
    private_key = RSA.generate(1024, random_gen)
    public_key = private_key.publickey()
    response = {
        'private_key': binascii.hexlify(private_key.exportKey(format='DER')).decode('ascii'),
        'public_key': binascii.hexlify(public_key.exportKey(format='DER')).decode('ascii')
    }
    return jsonify(response), 200
```

-La fonctionnalité de la route Flask **new_wallet** consiste à générer une nouvelle paire de clés RSA. Cette paire de clés comprend une clé privée et une clé publique correspondante.

➢ Front end :

-En cliquant sur 'Generate Wallet', on fait appel à la fonctionnalité du backend de génération des clés privé et publique.

5. Effectuer une transaction :

Une transaction est définie par 4 attributs

1- sender_address 2- sender_private_key 3- recipient_address 4- Product_UniCode

➢ Signature :

-Pour générer une transaction il fallait passer par un algorithme de signature, qui vise à indiquer d'une façon permanente que la transaction est émis par le propriétaire du 'private' et 'public' adresses.

-Cette signature est après utilisée pour vérifier la validé par les mineurs.

-La signature est en fait effectuer en utilisant la clé privé de celui qui a effectuer la transaction.

- Code:

```
def sign_transaction(self):
    private_key = RSA.importKey(binascii.unhexlify(self.sender_private_key))
    signer = PKCS1_v1_5.new(private_key)
    h = SHA.new(str(self.to_dict()).encode('utf8'))
    return binascii.hexlify(signer.sign(h)).decode('ascii')
```

private_key = RSA.importKey(binascii.unhexlify(self.sender_private_key))

Cette ligne importe la clé privée à partir d'une représentation hexadécimale (self.sender_private_key) et la stocke dans la variable private_key.

signer = PKCS1_v1_5.new(private_key)

Ici, un objet de signature est créé en utilisant le schéma de rembourrage PKCS1_v1_5 et la private_key précédemment importée. Ce signataire sera utilisé pour créer une signature numérique pour la transaction.

h = SHA.new(str(self.to_dict()).encode('utf8'))

Cette ligne calcule un hachage SHA-1 des détails du transaction renvoyé par la méthode to_dict().

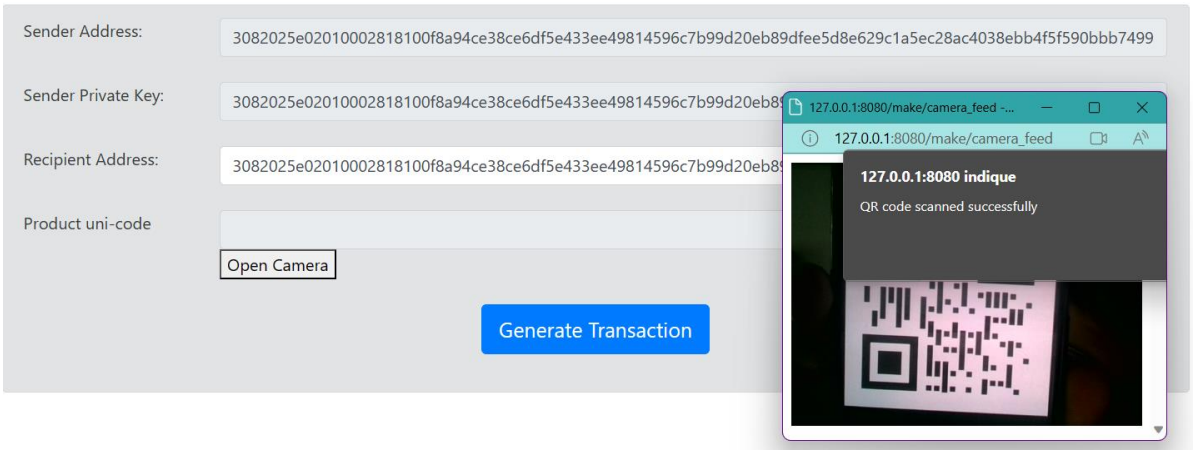
return binascii.hexlify(signer.sign(h)).decode('ascii')

Cette ligne utilise le signer pour signer le hachage SHA-1 (h) et renvoie la signature numérique sous forme d'une chaîne hexadécimale après encodage et décodage.

- Le Product UniCode :
  -Pour faire entrer le Unicode du produit, il fallait faire le scanning d'un QR code unique généré précédemment par le constructeur(ou l'initiateur de la chaine d'approvisionnement)

  -Pour effectuer ça on passe par

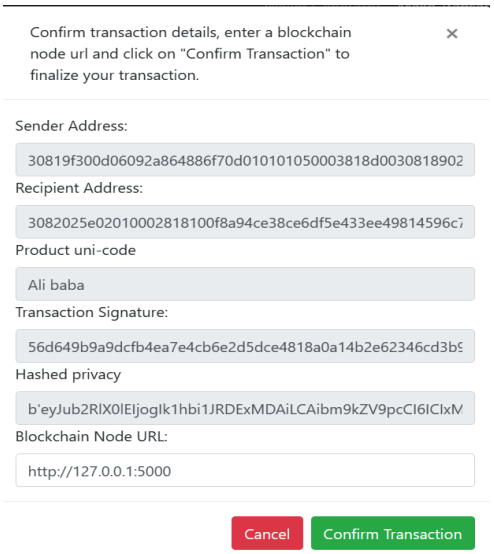
  1- Scanning du QR code en JavaScript (on ouvre la camera du pc et on fait le scanning dans la partie front-end).
  2- Vérification de l'existence de ce QR dans des anciens blocs (sauf pour l'initiateur, il ne vérifie rien).



A la fermeture de la fenêtre du camera le QR code scanné s'écrit automatiquement dans la zone Product Unicode dans la page parente.

➢ Génération de la transaction :

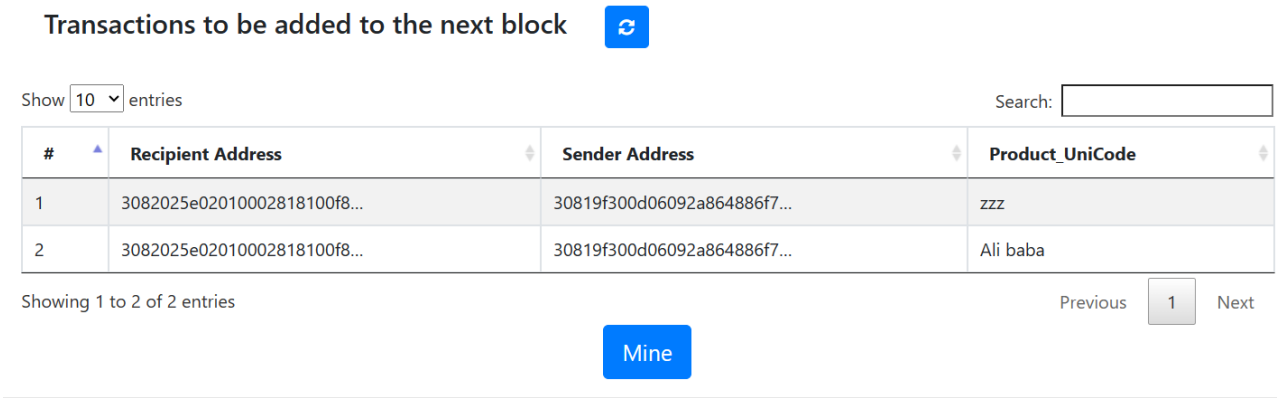
Après avoir remplir tous les champs et scanner le code QR du produit, on passe à la génération de la transaction, cela est fait en cliquant un bouton qui initie un code en backend qui stock les données de la transaction de manière bien structurée pour les prochains processus du blockchain.



➢ Ou part la transaction ?

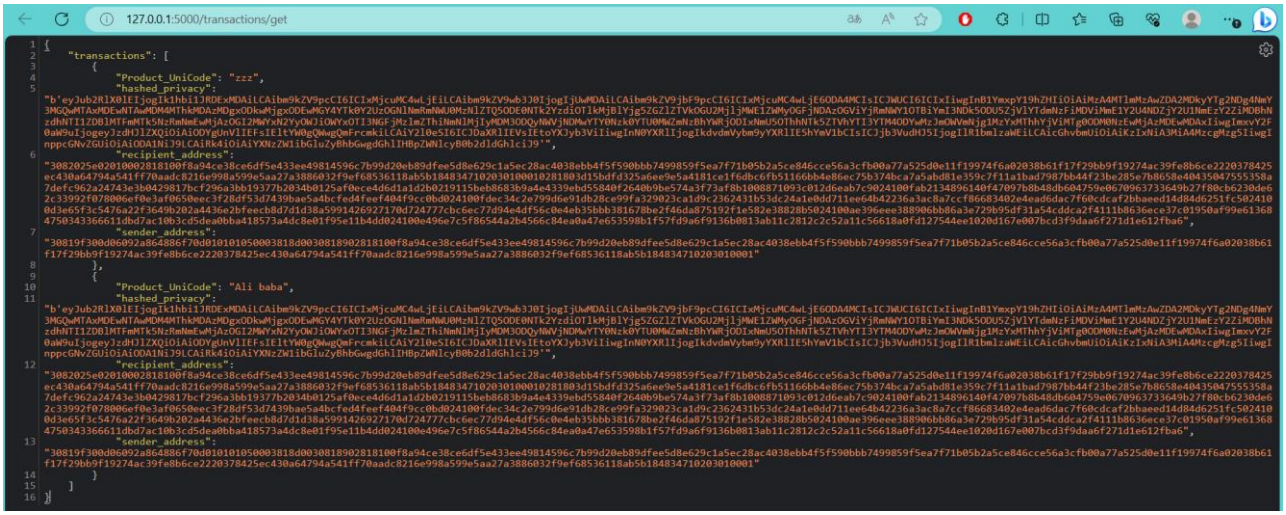
Les transactions se regroupent dans une listes d'attente, cette liste d'attente est dans un nœud où seuls les mineurs peuvent vérifier et ajouter la transaction à un bloc.

-Vue en front end : Un tableau trié selon l'antériorité



-Vue en backend :

Un fichier Json, déployé sur un le endpoint /transaction/get.



- Code de l'ajout d'une nouvelle transaction :

```
@app.route('/transactions/new', methods=['POST'])
def new_transaction():
    values = request.form
    print(values)
    # Check that the required fields are in the POST'ed data
    required = ['sender_address', 'recipient_address', 'Product_UniCode', 'signature','hashed_privacy']
    if not all(k in values for k in required):
        return 'Missing values', 400
    # Create a new Transaction
    transaction_result = blockchain.submit_transaction(values['sender_address'], values['recipient_address'],
    print(values['Product_UniCode'])
    if transaction_result == False:
        response = {'message': 'Invalid Transaction!'}
        return jsonify(response), 406
    else:
        response = {'message': 'Transaction will be added to Block '+ str(transaction_result)}
        return jsonify(response), 201
```

6. Mining : le procès de confirmation et ajout des blocs

➢ L'idée :

-La méthodologie adopté par notre système est de faire une vérification des données personnelles de l'émetteur, ce qu'on fait en vrai c'est d'encrypter ces données lors de la création de la transaction, décrypter ces données par le mineur et vérifier si elles sont valides, si elles y sont donc on crée un bloc qui contiendra les transactions mises en attente.

On a utilisé un simple algorithme de cryptage juste pour le test, c'est le cryptage à base64, et pour une meilleure sécurité on utilisera des algorithmes de cryptage dans un seul sens (hashing) dans des versions plus améliorés.

- Le previous hash :

-Pour l'ajout d'un block on a besoin du 'previous_hash' c'est le hash code du dernier block, la sémantique du 'previous_hash' est un concept important pour assurer la sécurité :

- Le previous hash est un code fait après passer tout le contenu du dernier bloc sur une fonction de hashing. Si une seule modification se fait sur un bloc N, le previous hash du block N+1 ne correspondra plus au hash actuelle du block N, on constate qu'une modification s'est passé et que le chainage n'est plus valide.

```
def valid_chain(self, chain)
```

Dans le code cette fonction est responsable de tester la validité de la chaine comme décrit.

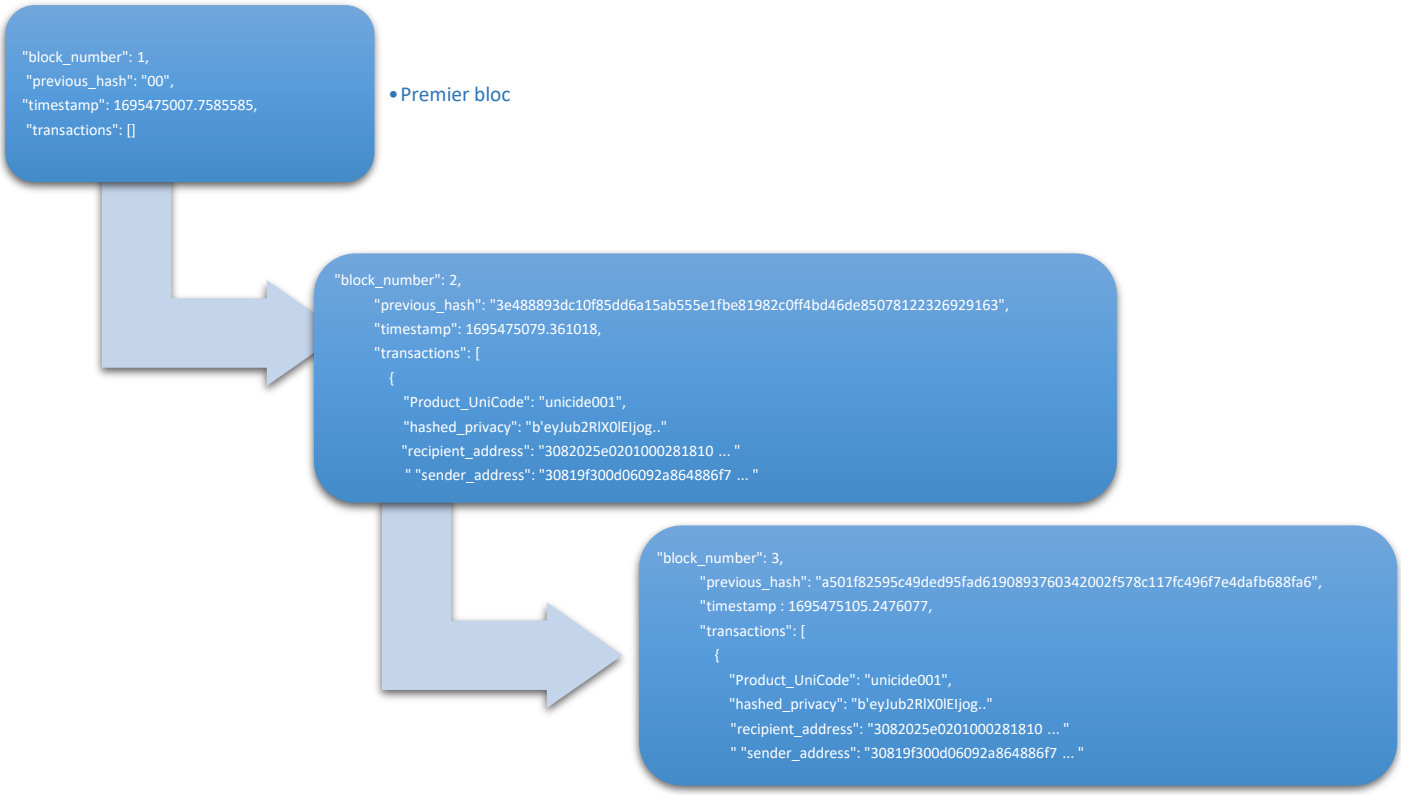
- Le premier Bloc :

-Ce bloc n'admet pas de previous hash, ni de transaction, il n'a comme rôle que commencer la chaine.

- Le timestamp :

-La sémantique derrière cet attribut c'est de dater et trié les transaction en ordre chronologique.

- Aperçu sur à quoi ressemble réellement les blocs dans notre système :



➢ La validation des transactions :

```python
    def proof_of_privacy(self,transaction):
        private_data = str(base64.b64decode(transaction["hashed_privacy"][2:]))
        component = self.get_component(transaction)
        if private_data == component:
            return True

    def valid_proof(self):
        for transaction in self.transactions:
            print(transaction)
            if not self.proof_of_privacy(transaction) :
                return False
        return True
```

Ce bout de code fait le procès inverse de la création du hashed privacy. Si on obtient True, donc les données privées y compris VP sont bien correctes et l'algorithme de consensus est validé donc on aura le droit d'ajouter un nouveau bloc.

7. Soumettre une transaction :

Dans la fonction du mining on fait appelle à une fonction d'ajout d'un bloc qui contient les transactions confirmés une par une.

```python
@app.route('/mine', methods=['GET'])
def mine():
    # We run the proof of work algorithm to get the next proof...
    transactions = blockchain.transactions
    last_block = blockchain.chain[-1]
    for trans in transactions:
        if blockchain.proof_of_privacy(trans) :
            print(trans)
            sender_adress = trans["transaction"]["sender_address"]
            recipient_address=trans["transaction"]["recipient_address"]
            Product_UniCode=trans["transaction"]["Product_UniCode"]
            blockchain.submit_transaction(sender_address=sender_adress, recipient_address=blockchain.node_id,
        # Forge the new Block by adding it to the chain'''
    previous_hash = blockchain.hash(last_block)
    block = blockchain.create_block(previous_hash)

    response = {
            'message': "New Block Forged",
            'block_number': block['block_number'],
            'transactions': block['transactions'],
            'previous_hash': block['previous_hash'],
        }
    return jsonify(response), 200
```

-Cette fonction initie la partie de vérification et confirmation de la liste des transactions.
-On fait une itération sur les transactions mises en attente et on commence par tester si elle vérifient bien l'algorithme de consensus pris en charge.

-Si c'est bien vérifier alors on part à l'ajout des transactions dans un bloque, et on 'append' le bloque à la blockchain : 1) on fait un hashing du dernier bloque pour faire le chainage

2) on crée le nouveau bloque et on l'ajoute à la fin de la chaine.

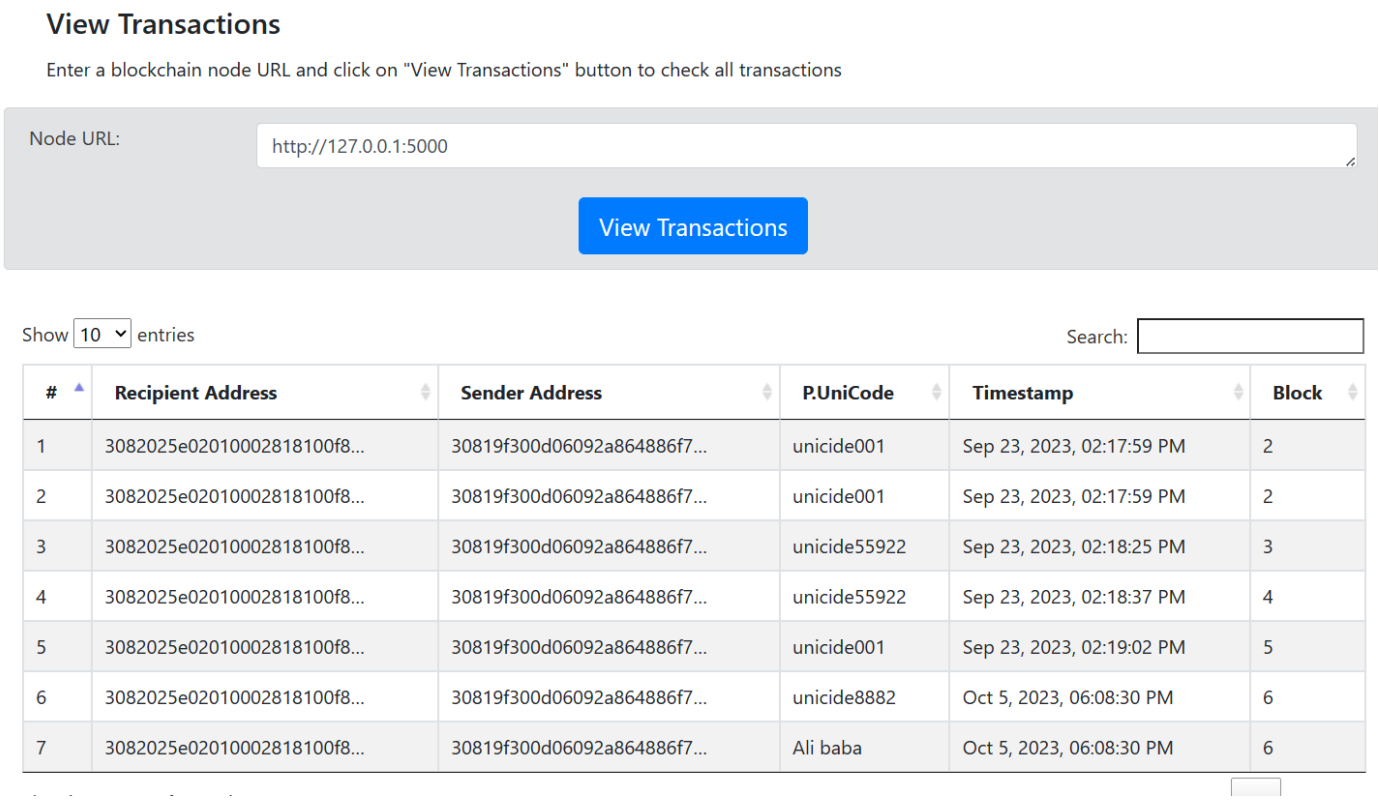
- La fonction 'Submit_transaction' :

```
def submit_transaction(self, sender_address, recipient_address, value,privacy, signature):
    transaction = OrderedDict({'sender_address': sender_address,
                               'recipient_address': recipient_address,
                               'Product_UniCode': value,
                               'hashed_privacy': privacy})
    transaction_verification = self.verify_transaction_signature(sender_address, signature, transaction)
    if transaction_verification:
        self.transactions.append(transaction)
        return len(self.chain) + 1
    else:
        return False
```

-Dans le procès de confirmation de la transaction, il fallait passer par la vérification de la signature qui assure la possession de la transaction par la personne qui l'a transmis.
-Si la signature est valide alors on ajoute la transaction à une liste de transactions qui formeront ensuite la liste des transaction dans le bloque à créer.

8. Consultation des transactions et vérification dans la blockchain : la transparence

-L'un des critères basiques qui définissent une blockchain c'est la transparence.
-Toute les transactions sont afficher en pleine visibilité par tous types d'utilisateurs.



#	Recipient Address	Sender Address	P.UniCode	Timestamp	Block
1	3082025e0201000281810 0f8...	30819f300d06092a864886f7...	unicide001	Sep 23, 2023, 02:17:59 PM	2
2	3082025e0201000281810 0f8...	30819f300d06092a864886f7...	unicide001	Sep 23, 2023, 02:17:59 PM	2
3	3082025e0201000281810 0f8...	30819f300d06092a864886f7...	unicide55922	Sep 23, 2023, 02:18:25 PM	3
4	3082025e0201000281810 0f8...	30819f300d06092a864886f7...	unicide55922	Sep 23, 2023, 02:18:37 PM	4
5	3082025e0201000281810 0f8...	30819f300d06092a864886f7...	unicide001	Sep 23, 2023, 02:19:02 PM	5
6	3082025e0201000281810 0f8...	30819f300d06092a864886f7...	unicide8882	Oct 5, 2023, 06:08:30 PM	6
7	3082025e0201000281810 0f8...	30819f300d06092a864886f7...	Ali baba	Oct 5, 2023, 06:08:30 PM	6

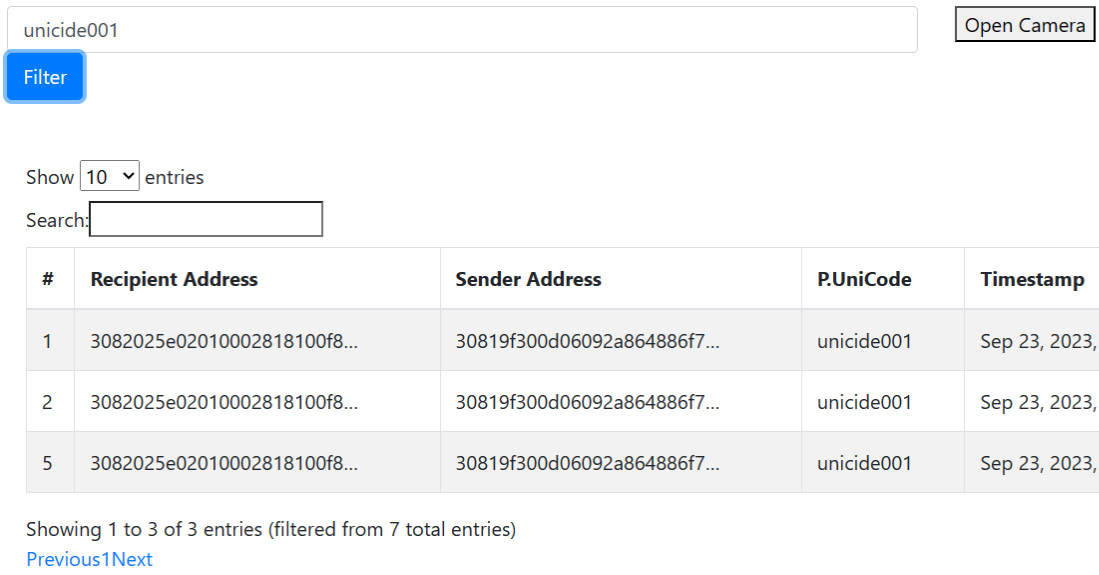
Les transactions validées sont mises en ordre décroissant selon le timestamp, c'est la date exacte du moment de la génération de la transaction.

Le nombre de block doit être visible pour donner une idée sur la structuration de la blockchain.

9. Aspect de traçabilité :

La fonctionnalité principale du système développé c'est d'effectuer un traçage des produits identifiés par leurs codes uniques. La traçabilité doit s'effectuer sur toute la chaine d'approvisionnement du début de la fabrication jusqu'à le produit atteint le consommateur final.
Le logique derrière, c'est de donner une vision bien clair sur les étapes faites pour transformer, fabriquer, emballer, déplacer et finalement tester le produit.
Toute ces étapes doivent accompagner la transparence le long du processus de traçage.
Pour cela une page web de traçabilité est mis en place pour effectuer cette tâche.

- Vue en front end :



Dans cette page un tableau qui regroupe toutes les transactions du blockchain peut être filtrer en utilisant le code unique d'un tel produit, on pourra donc savoir la personne qui a effectué la transaction et celle qui l'a reçue, le temps et le block contenant la transaction.

Chapitre 4 : Intégration Front-End et Back-End : JavaScript et Flask

1. Flask :

a. Importation :

```
from flask import Flask, jsonify, request, render_template , Response
```

- **Flask** : C'est la classe principale de Flask utilisée pour créer et configurer une application web.
- **jsonify** : Ce module permet de renvoyer des réponses au format JSON à partir de l'application Flask, utile pour la construction d'API RESTful.
- **request** : Ce module donne accès aux données de la requête entrante, y compris les paramètres de requête, les en-têtes HTTP, et plus encore, facilitant le traitement des informations envoyées par les clients.
- **render_template** : Ce module permet de générer des modèles HTML en utilisant un moteur de modèle tel que Jinja2, pratique pour générer des pages web dynamiques.
- **Response** : Ce module permet de créer des réponses HTTP personnalisées pour l'application Flask, utile pour les situations où il est nécessaire de contrôler précisément le contenu de la réponse.

b. Initiation :

```
app = Flask(__name__)
```

- En exécutant cette ligne de code, une instance de l'application Flask est créée, ce qui signifie que l'application est démarrée. Cela permet ensuite d'ajouter des routes, des vues et de configurer le comportement de l'application en fonction des besoins.

c. Routes :

- Les routes en Flask sont des chemins URL qu'on définisse dans l'application web pour spécifier comment l'application doit réagir aux différentes URL. Chaque route est associée à une fonction particulière qui sera exécutée lorsque l'URL correspondante est accédée.

  Exemple :
  @app.route('/view/transactions')

  Cette route correspond à l'URL "/view/transactions".
  Elle est configurée pour répondre aux requêtes HTTP de type GET par défaut.
  Lorsque l'URL "/view/transactions" est accédée, la fonction associée à cette route sera exécutée pour afficher des transactions.

@app.route('/make/get_qr_code', methods=['POST'])
Cette route correspond à l'URL "/make/get_qr_code".
Elle est configurée pour répondre uniquement aux requêtes HTTP de type POST.
Lorsque l'URL "/make/get_qr_code" est accédée via une requête POST, la fonction associée à cette route sera exécutée. Cela peut être utilisé pour générer un code QR en réponse à une requête POST.

@app.route('/wallet/new', methods=['GET'])
Cette route correspond à l'URL "/wallet/new".
Elle est configurée pour répondre uniquement aux requêtes HTTP de type GET.
Lorsque l'URL "/wallet/new" est accédée via une requête GET, la fonction associée à cette route sera exécutée. Cela peut être utilisé pour afficher un formulaire de création de portefeuille

d. Fonctions usuelles :

app.run(host='127.0.0.1', port=port)
Cette ligne est utilisée pour lancer l'application Flask. La méthode run() est appelée sur l'objet Flask nommé app.
Le paramètre **host** spécifie l'adresse IP sur laquelle l'application sera exécutée. Dans cet exemple, l'application sera accessible uniquement depuis la machine locale (127.0.0.1).
Le paramètre **port** spécifie le numéro de port sur lequel l'application écoutera les requêtes HTTP. La valeur de port est attendue d'être définie ailleurs dans votre code.

app = Flask(__name__)
Cette ligne crée une instance de l'objet Flask.
__name__ est une variable spéciale en Python qui représente le nom du module actuel. Dans ce contexte, cela permet à Flask de savoir où se trouvent les modèles et les fichiers statiques associés à votre application.

return render_template('./index.html')
Cette ligne renvoie une réponse HTML en utilisant le mécanisme de rendu de modèles de Flask. Elle suppose que vous avez un fichier HTML nommé "index.html" dans le répertoire du modèle de votre application.
La fonction render_template est une fonction de Flask qui prend en charge la gestion des modèles et renvoie le contenu HTML généré à partir du modèle spécifié. Elle est utilisée pour générer une page web en utilisant des modèles HTML.

return jsonify(response), 200
Cette ligne renvoie une réponse JSON à une requête HTTP avec un code de statut HTTP 200 (OK). jsonify est une fonction de Flask qui convertit un objet Python en une réponse JSON valide. Dans cet exemple, elle prend l'objet response et le renvoie sous forme de réponse HTTP JSON.

2. JavaScript :

On va juste attaquer brièvement la page web du scanning du QR code, Partie JavaScript :

```
<script src="{{ url_for('static', filename='jsQR.js') }}"></script>
    <script>
        const video = document.getElementById('qr-video');
        const canvas = document.getElementById('qr-canvas').getContext('2d');

        async function startCamera() {
            try {
                const stream = await navigator.mediaDevices.getUserMedia({ video: true });
                video.srcObject = stream;
            } catch (error) {
                console.error('Error accessing camera:', error);
            }
        }

        startCamera();
        var qrCodeData = "";
        video.addEventListener('play', function () {
            const canvasElement = document.getElementById('qr-canvas');
            canvasElement.width = video.videoWidth;
            canvasElement.height = video.videoHeight;

            const canvasContext = canvasElement.getContext('2d');
            const qrCanvas = document.getElementById('qr-canvas');

            (function scan() {
                if (video.paused || video.ended) {
                    return setTimeout(scan, 100);
                }

                canvasContext.drawImage(video, 0, 0, canvasElement.width, canvasElement.height);
                const imageData = canvasContext.getImageData(0, 0, canvasElement.width,
canvasElement.height);
                const code = jsQR(imageData.data, imageData.width, imageData.height);

                if (code) {
                    console.log('QR Code detected:', code.data);
                    qrCodeData = code.data;

                    // Create a JSON object with the QR code data
                    const jsonData = {
                        'code': qrCodeData
                    };
                    fetch('/make/get_qr_code', {
                        method: 'POST',
                        headers: {
                            'Content-Type': 'application/json'
                        },
                        body: JSON.stringify(jsonData)
                    })
                    .then(response => {
                        if (response.ok) {
                            console.log('QR code data sent to the server.');
                            console.log(JSON.stringify(jsonData));
                            alert("QR code scanned successfully");
                            window.opener.postMessage(code.data, "*");
                            window.close();
                        } else {
                            console.error('Error sending QR code data to the server.');
                        }
                    })
                    .catch(error => {
                        console.error('Error sending QR code data:', error);
                    });
                }
                requestAnimationFrame(scan);
            })();
            });


    </script>
```

Ce code JavaScript réalise les étapes suivantes :

1. Il inclut un script externe, jsQR.js, qui est utilisé pour le décodage de QR codes. Ce script est chargé depuis un fichier statique du projet.
2. Le code crée une interface pour accéder à la caméra de l'appareil. Une fois la caméra activée, il commence à capturer des images vidéo.
3. Il capture constamment des images de la caméra et les dessine sur un élément HTML canvas (qr-canvas). Le code utilise jsQR pour analyser ces images à la recherche de QR codes.
4. Lorsqu'un QR code est détecté dans la vidéo en direct, le code JavaScript extrait les données du QR code.
5. Il crée un objet JSON contenant les données du QR code.
6. Ensuite, il envoie ces données au serveur en utilisant une requête POST via fetch(). Le serveur est supposé être accessible à l'URL '/make/get_qr_code'.
7. Si la requête est réussie (c'est-à-dire, si le serveur répond avec un code HTTP 200 OK), le code affiche un message de succès et envoie les données du QR code à la fenêtre parente à l'aide de window.opener.postMessage(). Il ferme ensuite la fenêtre actuelle avec window.close().

Conclusion général

En conclusion, mon stage au sein d'IT Grow a été une expérience inestimable qui m'a permis de développer et d'approfondir mes compétences et connaissances dans des domaines clés de l'industrie, notamment la blockchain et le développement front-end et back-end. Travailler aux côtés de l'équipe exceptionnelle d'IT Grow, m'a offert une opportunité unique d'explorer ces technologies de pointe et de contribuer à des projets innovants. Cette expérience a non seulement renforcé mes compétences professionnelles, mais elle m'a également enrichi sur le plan personnel. Je tiens à exprimer ma gratitude envers toute l'équipe d'IT Grow pour cette opportunité précieuse, et je suis impatient d'appliquer les connaissances acquises dans ma future carrière.