REPUBLIQUE DE COTE

Union-Discipline -Travail

Ministère de la Communication
de l'Economie Numérique et de la Poste

Ecole Supérieure Africaine des Technologies
de l'Information et de la Communication

ESATIC
ECOLE SUPÉRIEURE AFRICAINE DES TIC

Année académique : 2018/2019

PROJET INTERNE MASTER 2

Etude et mise en œuvre d'une blockchain de traçabilité des produits locaux dans les supermarchés ivoiriens

Présenté par

ASAOUA SODIQ OLAYEMI
Master 2 Sécurité Informatique et Technologie du Web

ASSI ARLETTE AUDREY
Master 2 Sécurité Informatique et Technologie du Web

COULIBALY DOKOUNGO ISSOUF
Master 2 Sécurité Informatique et Technologie du Web

Encadrant académique :

Dr KAMAGATE Beman
Enseignant-Chercheur

DEDICACE

Nous dédions ce mémoire à nos familles respectives

REMERCIEMENT

Avant tout propos, nous tenons à exprimer notre profonde gratitude à des personnes qui, d'une manière ou d'une autre ont contribué à la réalisation du présent document. Sans pour autant être exhaustifs, nous aimerons citer :

- Pr. KONATE Adama, actuel Directeur Général de l'ESATIC, pour tous les efforts faits pour notre réussite ;
- Dr. SORO Etienne, Directeur de la Pédagogie et encadreur à l'ESATIC ;
- Dr KAMAGATE Beman, notre encadrant pédagogique ;
- Tout le corps professoral et administratif de l'ESATIC, pour le savoir transmis.
- Tous ceux qui nous ont soutenus et continuent de nous soutenir par leurs actes, prières et pensées.

SOMMAIRE

LISTE DES FIGURES

LISTE DES TABLEAUX

INTRODUCTION

La Côte d'Ivoire bénéficie de nombreux atouts pour la production agricole. Reconnu 1er producteur mondial de cacao, ce pays a une production agricole variée. Ces produits agricoles ivoiriens sont énormément consommés sur le territoire. Ils sont présents sur la place publique tant sur les grandes surfaces que sur les marchés de quartier. Cependant, depuis un moment, les consommateurs ivoiriens sont de plus en plus méfiants de la qualité des produits consommés. En effet, en nous limitant aux produits locaux alimentaires, les scandales alimentaires et les rumeurs sur la mauvaise qualité des produits consommés n'ont pas manqué en Côte d'Ivoire. Ainsi, pour un produit local acheté, les consommateurs soulèvent les interrogations suivantes : d'où viennent réellement ces produits ? Comment savoir que ces produits ont été cultivés dans les conditions requises de productions ?

Comme toute entreprise désirant satisfaire au mieux sa clientèle, les supermarchés proposant des produits locaux trouvent un intérêt à mettre en place une nouvelle stratégie de fidélisation de leur clientèle. Ainsi, rassurer les consommateurs de la qualité des produits locaux qu'ils consomment à travers une traçabilité de la chaine de production de ces produits est une stratégie efficace de fidélisation de la clientèle

Avec l'explosion du numérique, il existe de nombreuses technologies adaptées à nos différents besoins. Dans un besoin de confiance, la technologie Blockchain décrite comme une technologie assurant la confiance, est une véritable opportunité à saisir pour les supermarchés. C'est pourquoi la traçabilité de la chaine de production se fera au moyen de la technologie Blockchain. D'où le thème « Etude et mise en œuvre d'une blockchain de traçabilité des produits locaux dans les supermarchés ivoiriens ». En d'autres termes, il s'agira de mettre en place une application basée sur une blockchain de traçabilité permettant de rassurer les consommateurs de la qualité des produits locaux consommés.

Pour atteindre cet objectif, nous avons porté réflexion sur les études relatives au projet et subdivisé le travail en trois grandes parties. Premièrement, nous présenterons les informations générales à la bonne compréhension et la nécessité du projet. Ensuite, nous présenterons l'analyse et la conception de notre application. Enfin, nous présenterons le résultat de cette étude tout en insistant sur les limites de celui-ci.

PARTIE I : GENERALITE

CHAPITRE I : PRESENTATION DU PROJET

La présentation d'un projet permet d'avoir une vision claire de celui-ci. C'est pourquoi, dans ce chapitre, nous définissons les notions clés de notre projet. Ensuite, nous présentons une généralité sur la blockchain. Enfin, nous présentons le cahier de charges de notre projet.

I. Définitions des notions clés

1. Blockchain

Une blockchain constitue en fait une base de données qui contient l'historique de tous les échanges effectués entre ses utilisateurs depuis sa création. Cette base de données est sécurisée et distribuée : elle est partagée par ses différents utilisateurs, sans intermédiaire, ce qui permet à chacun de vérifier la validité de la chaîne. [1]

2. Traçabilité

De façon générale, « la traçabilité consiste, pour chaque entreprise d'un secteur, à garder la trace de ses fournisseurs et de ses clients ». [2]

Mise dans un contexte alimentaire, le règlement (CE) n 178/2002 défini la traçabilité comme : « la capacité de retracer, à travers toutes les étapes de la production, de la transformation et de la distribution, le cheminement d'une denrée alimentaire, d'un aliment pour animaux, d'un animal producteur de denrées alimentaires ou d'une substance destinée à être incorporée ou susceptible d'être incorporée dans une denrée alimentaire ou un aliment pour animaux »

3. Produits locaux

Les produits locaux sont des produits cultivés ou fabriques sur le territoire ivoirien. Nous pouvons citer par exemple les tomates, l'aubergine, le gombo, le coton, l'anacarde, etc.

II. Généralité sur la blockchain

Apparue avec « Bitcoin » en 2008, la blockchain implémente une base de données contenant l'historique de tous les échanges effectués entre ses utilisateurs depuis sa création. Les informations stockées sont de toutes natures : transactions, titres de propriétés, contrats, œuvres d'art… Les échanges fonctionnent sans l'intervention d'un organe central de contrôle[3]. Elle est structurellement accessible, partagée et sécurisée grâce à des algorithmes de consensus [4]. La Blockchain découle de la juxtaposition de différentes

innovations informatiques déjà bien connues telles que les réseaux pair-à-pair, la cryptographie asymétrique et les algorithmes de consensus décentralisés. [4]

La cryptographie asymétrique qui repose sur le concept de double clef publique et privée, permet l'envoie de données cryptées à un tiers. Ainsi, l'échange d'informations cryptées entre personnes ne se connaissant pas devient possible. Un réseau pair-à-pair (P2P) est un ensemble d'ordinateurs indépendants – appelés des nœuds de la Blockchain – qui se connectent sur un réseau et peuvent communiquer sans une entité centrale.

Nous distinguons trois catégories de Blockchain :

- Les blockchains publiques **:** Tous les participants du réseau peuvent accéder à la base de données, en héberger une copie et la modifier en participant à un processus de validation des blocs d'informations à enregistrer. L'exemple le plus connu est la Blockchain « Bitcoin »

- Les consortiums **:** Cette catégorie de Blockchain est ouverte au public, mais toutes les informations ne sont pas accessibles. Les utilisateurs ne disposent pas des mêmes droits et la validation des blocs se déroule selon des modalités prédéfinies. La Blockchain consortiums est donc « partiellement décentralisée ».

- Les blockchains privées **:** Les droits d'accès et de modification de la Blockchain sont centralisés auprès d'une organisation. Le système est facilement intégrable au sein des SI et possède l'avantage de disposer d'une ligne d'audit cryptographiée. Ici, le réseau n'a pas besoin d'inciter des mineurs à mettre à disposition leur puissance de calcul pour faire tourner les algorithmes de validation. [4]

1. Les blocs

Littéralement, une Blockchain désigne une chaine de blocs, des conteneurs numériques sur lesquels sont stockés des informations de toutes natures. Il s'agit donc d'un empilement de blocs. Un bloc est représenté par son **en-tête** et **son corps**. Afin d'assurer la sécurité de la Blockchain, l'en-tête d'un bloc contient un identifiant, un pointeur vers le bloc précédent, un horodatage, et un nonce.

Le pointeur n'est rien d'autre que le hash (ou somme de contrôle) du contenu du bloc précédent. Ainsi, les blocs sont liés les uns aux autres de telle sorte que si l'on veut modifier un bloc, on doit modifier tous les blocs présents dans la chaîne. Deux blocs ayant des sommes de contrôle différentes contiennent nécessairement des contenus différents. Donc, la somme de contrôle d'un bloc peut servir d'identifiant unique au bloc qui lui succède.

Le nonce est une partie de l'en-tête qui sert à contenir des valeurs neutres (c'est-à-dire souvent des 0) dans des blocs de hachage, pour qu'ils aient tous la même taille, le même nombre de bits, sans toutefois modifier le corps du bloc. Ainsi, chaque hachage a une certaine quantité de 0 définie par le nonce, que l'ordinateur saura lire comme étant superflue, puisqu'il a en vue le fichier final. [5]

Le corps, quant à lui, contient des transactions, des données, ou des documents. Une telle structuration en chaîne garanti la sécurité de la Blockchain, son intégrité ainsi que l'unicité de chaque élément présent sur le réseau.

La validation et l'enregistrement d'un bloc se fait selon l'algorithme suivant :

1. **Vérifier que le bloc précédent existe et est valide.** Cela revient à vérifier que le hash du dernier bloc correspond bien à l'élément « hash du bloc précédent » (HP (block) sur la figure 1.2) référencé en en-tête du bloc en cours de vérification.

2. **Vérifier que la date du bloc est supérieure à la date du bloc précédent**

3. **Vérifier l'ensemble des transactions**. Si l'une d'entre elles retourne une erreur, stopper et retourner FAUX

4. **Mettre à jour l'ensemble des transactions non vérifiées** et retourner VRAI. Cet algorithme est exécuté par des nœuds.

2. Nœuds

Comme expliqué plus haut, le réseau Blockchain est un système distribué. Un système distribué est un ensemble d'ordinateurs indépendants qui se connectent sur un réseau et peuvent communiquer. Ces ordinateurs représentent les nœuds dans le réseau Blockchain. Chaque nœud possède une copie du registre partagé. Cela assure la réplication des données afin d'assurer une plus grande sécurité au système. Il existe plusieurs types de nœuds en fonction de leurs actions sur le réseau : création de la transaction, validation de la transaction, enregistrement de la transaction et participation à un consensus.

3. Consensus

Le consensus est le mécanisme qui consiste à garantir qu'une transaction n'est pas frauduleuse et qu'un bloc est valide. [6] Afin de parvenir à une validation globale sur le réseau Blockchain, la solution trouvée est de procéder à un vote général entre les nœuds (ceux participants à la validation). Ce choix part de l'hypothèse qu'il aura toujours plus d'utilisateurs honnêtes que d'utilisateurs malveillants sur le réseau. Pour assurer la sécurité du réseau, il est primordial de doter le réseau de mécanismes et/ou des règles reconnus, partagés et validés par tous. Ces dispositions (mécanismes et règles) permettent la validation

des transactions puis des blocs avant qu'ils soient ajoutés dans la chaine de blocs. Une fois la majorité des utilisateurs (les nœuds) du réseau ont confirmé l'exactitude et la validité d'une transaction, le bloc possédant cette transaction sera ajouté à la chaîne de blocs existante.

Pour aboutir à de tels consensus, il existe une diversité de méthodes dont nous énumérons quelques-unes :

- **Proof of Authority (PoA)**

Une preuve d'autorité est le mécanisme de consensus d'une chaîne de blocs privée qui donne essentiellement à un utilisateur ou à un nombre spécifique d'utilisateurs, le droit de miner tous les blocs de la Blockchain.

- **Proof of Stake (PoS)**

Dans le cas du Proof of Stake (ou preuve de participation), les utilisateurs doivent régulièrement prouver la part de monnaie sous-jacente qu'ils détiennent. En effet, la participation au consensus est réservée aux « forgeurs » ayant les montants les plus élevés.

- **Proof of work (PoW)**

Proof of Work (ou preuve de calcul) est un mécanisme de validation d'un bloc qui repose sur la résolution « d'un problème crypto-mathématique complexe ». Le résultat de cette résolution est difficile à obtenir et nécessite beaucoup de puissance de calcul. Le processus de résolution est appelé **mining** et on parle de **miners**. Il existe plusieurs autres méthodes :

  - « Pratical Bizantine Fault tolerance » (PBFT)
  - « Proof of Hold » (PoH)
  - « Proof of Use » (PoU)

4. Smart contracts

Selon Blockchain France, le « *smart contract* est un ensemble de programmes autonomes qui exécutent automatiquement des conditions définies au préalable. Ils sont basés sur les instructions conditionnelles de type "if – then" ».[7] Ce sont donc contrats intelligents autonomes qui sont exécutés automatiquement par la Blockchain sans intervention humaine une fois que les conditions nécessaires à leur réalisation sont réunies. La donnée nécessaire à la vérification d'un contrat peut se trouver en dehors de la Blockchain. Dans de pareilles situations, la notion de « oracle » prend son sens. Un oracle

peut utiliser une base de données classique, un site internet ou autres sources de données qui pourraient servir à l'insertion d'une information à une position et un moment précis dans la Blockchain. Pour s'exécuter ou non, le smart contract lit la donnée et agit en conséquence. Par exemple, dans le cas d'un projet agricole, « l'oracle » rapporte le niveau de précipitation depuis le site de l'agence de météorologie pour ordonner ou non l'exécution de drainages. On se retrouve donc face à une nouvelle forme d'entités centrales.

5. Environnement et outils Blockchain

5.1. Bitcoin

Bitcoin, considéré comme la genèse de la technologie Blockchain, est une monnaie virtuelle cryptographique. En effet, il s'agit de la première monnaie électronique qui décentralise les payements par le biais d'un réseau pair-à-pair, sans avoir recours à un intermédiaire tel qu'une institution financière. L'idée fut présentée pour la première fois en novembre 2008 par une personne, ou un groupe de personnes, sous le pseudonyme de Satoshi Nakamoto. Bitcoin met à la disposition des développeurs un framework open-source pour le développement de solutions au profit du secteur bancaire.

5.2. Ripple

Ripple est un système de paiement instantané lancé en 2012. Le réseau Ripple permet des transactions interbancaires sécurisées, instantanées et à de très faibles coûts partout dans le monde. Il fonctionne comme une base de données publique partagée, avec un échange de monnaie distribuée intégré. Ripple prend en compte n'importe quelle monnaie fiduciaire (Dollars, Euros, ...), les crypto-monnaie ou toute autre unité de valeurs telles que des minutes de téléphone mobile.

5.3. Ethereum

Ethereum est une plate-forme décentralisée qui gère des contrats intelligents. Ces applications fonctionnent sur une Blockchain construite sur mesure, une infrastructure globale partagée extrêmement puissante qui permet des échanges de valeurs. Cela permet aux développeurs de créer des marchés, de stocker des registres, de transférer des fonds conformément à des accords passés. Le projet a été amorcé en août 2014 et est développé par la Fondation Ethereum, un organisme suisse à but non lucratif.

5.4. Hyperledger

Hyperledger est un effort de collaboration open source créé pour faire progresser les technologies Blockchain. Il s'agit d'une collaboration mondiale incluant des leaders dans les domaines de la finance, de la banque, de l'Internet des objets, des chaînes

d'approvisionnement, de la fabrication et de la technologie. Hyperledger est géré par la fondation Linux. [8]

6. Choix de notre environnement Blockchain

L'une des difficultés dans la réalisation d'un projet Blockchain est le choix du framework à utiliser. En fonction des spécificités du projet, la question est de savoir lequel est le meilleur ou lequel répond au mieux à nos besoins. Le tableau 1.1 permet de faire une comparaison rapide entre les quatre principaux frameworks et de voir comment ils se distinguent les uns des autres.

	Hyperledger	Ethereum	Ripple	Bitcoin
Description	Usage général	Usage général	Paiement	Paiement
Instance	Linux Foundation	Développeurs Ethereum	Ripple Labs	Développeurs Bitcoin
Etat	Base de données Clé-valeur	Données des comptes	Pas d'état	Données des transactions
Réseaux de Consensus	Au choix : PBFT	Minage	Protocole de Ripple	Minage
Type de réseau	Privé	Public ou Privé	Public	Public
Confidentialité	Ouvert / Privé	Ouvert	Ouvert	Ouvert
Contrats intelligents	Java, Go, JS	Solidity	Pas de contrats intelligents	Possible, mais pas évident

Figure 1: Comparaison des principaux frameworks de blockchain

Pour numériser une chaîne de Traçabilité alimentaire, un framework à usage général convient le mieux. Aussi, cette numérisation nécessite l'utilisation de smart contracts pour automatiser certaines opérations afin de réduire les failles sécuritaires dues à l'action humaine.

En outre, ces opérations se déroulant dans un contexte agricole, il serait alors très risqué d'opter pour un réseau de type public et ouvert à tous. De ce fait, un framework offrant une confidentialité est le meilleur choix. Aux vues de toutes ces exigences, le framework qui correspond le mieux est Hyperledger. Contrairement aux autres framework Blockchain, Hyperledger remplit les quatre éléments clés d'une Blockchain pour entreprise qui sont :

- Réseau autorisé ;
- Transactions confidentielles ;
- Pas de cryptomonnaie ;
- Programmable ;

Hyperledger offre plusieurs plates-formes de développement Blockchain en fonction de certaines spécificités [8]

- Hyperledger Sawtooth est une plate-forme modulaire pour le développement, le déploiement et l'exécution de registres distribués.

- Hyperledger Sawtooth inclut un nouvel algorithme de consensus, le Proof of Elapsed Time (PoET), qui cible de grandes populations de validateurs distribués avec une consommation minimale de ressources.

- Hyperledger Iroha est une plate-forme conçue pour être simple et facile à intégrer dans des projets d'infrastructure nécessitant une technologie de registres distribués. Hyperledger Iroha dispose d'une structure simple ; un développement en C++. L'accent est mis sur le développement d'applications mobiles.

- Hyperledger Fabric est une plate-forme pour développer des applications ou des solutions avec une architecture modulaire,

- Hyperledger Fabric permet aux composants, tels que les services de consensus et le service d'authentification, d'être plug-and-play. Hyperledger Fabric tire parti de la technologie des conteneurs pour héberger des contrats intelligents appelés "chaincode" qui constituent la logique d'application du système.

- Hyperledger Burrow est une machine de contrat intelligent. Le premier du genre, sorti en décembre 2014, fournit un client Blockchain modulaire avec un interpréteur de contrat intelligent, construit en partie selon les spécifications de la machine virtuelle Ethereum (EVM).

- Hyperledger Indy est un registre distribué, spécialement conçu pour l'identité décentralisée. Il fournit des outils, des bibliothèques et des composants réutilisables pour créer et utiliser des identités numériques indépendantes ancrées sur des chaînes de blocs ou d'autres registres distribués pour l'interopérabilité.

- Hyperledger Composer est un ensemble d'outils de collaboration pour la création de réseaux Blockchain qui simplifient et accélèrent la création de contrats intelligents et d'applications Blockchain pour résoudre les problèmes métier. Conçu avec JavaScript, il s'appuie sur des outils modernes tels que node.js, npm, CLI et des éditeurs populaires (Visual Studio Code, Atom, …). Composer propose des abstractions orientées métier ainsi que des exemples d'applications avec des processus « devops » faciles à tester pour créer des solutions Blockchain robustes.

Ainsi, pour le développement de notre solution, nous opterons pour les plateformes Hyperledger Composer (pour faciliter le développement) et Hyperledger Fabric (pour le déploiement avec son système de conteneurs de smart contracts) car elles fournissent des environnements disposants de composants indispensables au développement de notre solution.

III. Présentation du cahier de charges

1. Contexte général

La population ivoirienne utilise les produits locaux de diverses manières. Certains pour se vêtir, d'autres pour se nourrir ou d'autre tâche. Spécifiquement, dans la consommation alimentaire, de nombreuses interrogations se soulèvent. En effet, suite à de nombreux scandales sur les produits locaux alimentaires présents sur le marché ivoirien, ses consommateurs sont de plus en plus méfiants. En effet, plusieurs ivoiriens doutent de la qualité des produits consommés. L'ère du numérique offre des technologies permettant d'assurer la confiance en un produit notamment la technologie blockchain, considérée comme une révolution numérique. A l'aide de la blockchain, ce projet veut rassurer la population ivoirienne de la qualité des produits locaux en faisant la traçabilité de ces produits. Notre étude se limitera aux produits alimentaires agricoles non transformée.

2. Objectifs

2.1. Objectif général

L'objectif majeur de ce projet consiste à concevoir une application basée sur une blockchain de traçabilité permettant de rassurer les consommateurs de la qualité des produits locaux consommés.

2.2. Objectifs spécifiques

Plusieurs objectifs sont visés à travers ce projet :

- Comprendre la chaine de valeur des produits locaux ;
- Réaliser une étude permettant d' assurer la traçabilité des produits locaux des supermarchés ;
- Faire le choix d'outils et de technologies adaptés ;
- Réaliser une application mobile permettant aux consommateurs d'accéder aux informations sur le produit local voulu ;
- Réaliser une application mobile permettant aux acteurs de la chaine de production de renseigner les informations sur les produits locaux ;

3. Livrables attendus

Au vu des objectifs fixés, plusieurs livrables sont attendus :

- Une application mobile destinée aux consommateurs ;
- Une application mobile destiné aux producteurs, coopératives, aux grossistes, aux distributeurs ;
- Un manuel d'utilisation pour les utilisateurs finaux ;

CHAPITRE II : ETUDE DE L'EXISTANT

Ce projet est né de certaines limites constatées dans notre environnement. C'est pourquoi, dans ce chapitre nous présentons le fonctionnement des supermarchés ivoiriens relatifs aux produits locaux dans le but d'en faire une analyse. Par la suite, nous présentons des études menées sur la traçabilité alimentaire.

I. Fonctionnement actuel de la chaine d'approvisionnement des supermarchés

Le fonctionnement actuel de la chaine d'approvisionnement des supermarchés n'est pas uniforme. Il varie d'un supermarché à un autre. Nous présenterons plusieurs cas qui nous permettront de comprendre le fonctionnement de ces chaînes et les acteurs qui y participent.

1. Présentation des chaines d'approvisionnement

Cas 1 : les acteurs : producteurs, coopératives, grossiste, supermarché

Dans cette chaine nous avons un ou plusieurs planteurs (coopératives) qui produisent différentes fruits et légumes. Ces aliments sont ensuite récupérés par un grossiste qui se charge de les distribuer aux supermarchés.

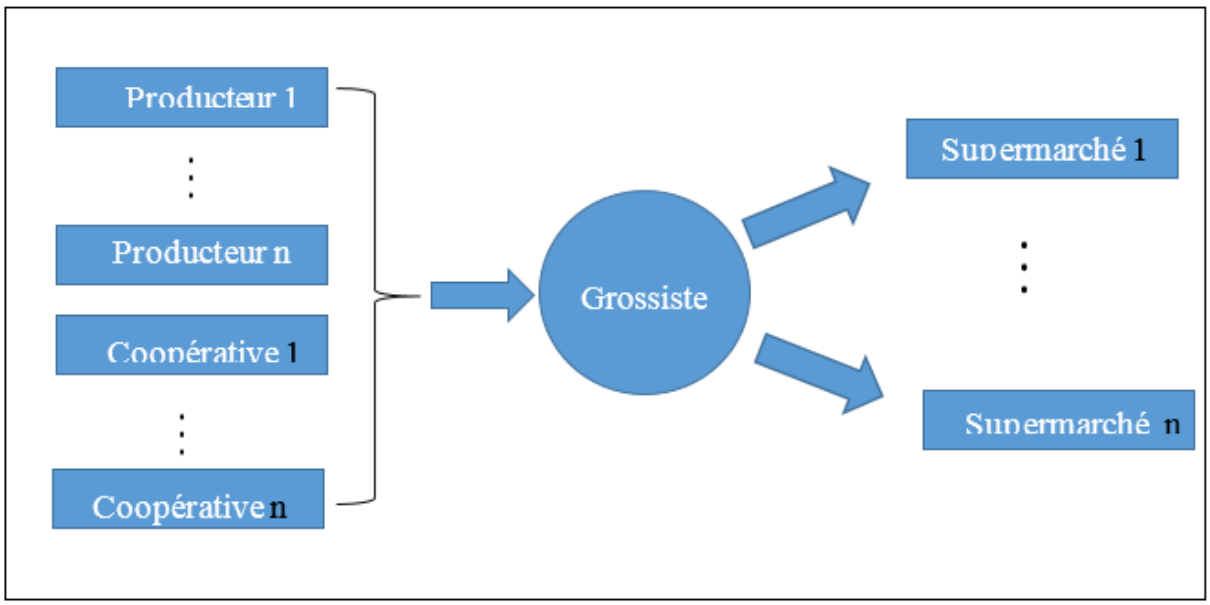

Figure 2: Chaine d'approvisionnement- cas 1

Cas 2 : acteurs : producteurs et supermarchés

Ici le supermarché s'adresse directement aux producteurs pour récupérer leurs produits et les vendre dans ses rayons. Ces planteurs ont généralement une grande capacité de production.

Figure 3: chaine d'approvisionnement- cas 2

Cas 3 : le supermarché est lui-même producteur de la denrée

Un autre cas est celui où le supermarché lui-même est producteur. Il récupère sa production depuis la plantation qu'il va vendre ensuite dans son supermarché.

Figure 4:chaine d'approvisionnement- cas 3

2. La gestion du produit au niveau du supermarché

Que ce soit les petits ou grands supermarchés, la gestion du stock de produits est quasiment pareille. Les produits arrivés aux supermarchés sont stockés. Ensuite un tri est effectué pour enlever les produits de mauvaise qualité. Après cela, le service informatique enregistre le stock restant qui sera exposé dans les différents rayons.

II. Solutions de traçabilité existante

1. Quelques solutions de traçabilité alimentaire existante

- AgriDigital [9]

AgriDigital est projet lancé en Australie par la startup FullProfie, qui permet aux producteurs céréales, acheteurs et grossistes de gérer les contrats, livraisons, paiements et inventaires sur leur plateforme cloud. AgriDigital est donc une plateforme SaaS. Il utilise un réseau privé de la blockchain Ethereum et la plateforme permet un paiement instantané à la livraison par smart contract. En 2016 le projet était encore à sa phase de test, les nœuds du réseau ont été gérés par AgriDigital, qui a joué les rôles d'opérateur, d'acheteur, et de régulateur afin de simuler l'écosystème.

- Provenance [9]

La startup Provenance vise à permettre d'une part de retracer le trajet d'un produit, depuis sa production jusqu'au point de vente, et d'autre part de vérifier que ce produit provient d'une source certifie. Un des premiers proof-of-concept (PoC) entrepris par Provenance s'est centré sur l'industrie de la pêche au thon en Indonésie.

Provenance permet au consommateur final de s'assurer, par le simple fait de scanner un tag NFC que le thon qu'il consomme a été pêché par Lutfi un pêcheur à la ligne indonésien. Les conditions sociales et environnementales du travail du pêcheur sont vérifiées par des ONG locales grâce à des audits réguliers.

En pratique, les pêcheurs, équipés de téléphones portables, enregistrent leurs prises par SMS dans la blockchain Ethereum.

Le transfert du poisson du pêcheur au fournisseur est inscrit dans la blockchain par mobile : le pêcheur envoie une demande de transfert et le fournisseur l'accepte une fois en possession du poisson. Le token représentant ledit poisson (numéro de série unique) est donc transféré à l'adresse du fournisseur. En utilisant un exploreur blockchain tel que morden.ether.camp, l'historique de la possession du produit peut ensuite être vérifié à tout moment de façon transparente.

Une fois livré à l'usine, un ERP, Tally-O, permet l'interfaçage avec la blockchain Ethereum. En parallèle, un code barre vient faire le lien entre le tangible et le digital. Les transformations de la matière première en plusieurs conserves sont également enregistrées. Enfin un tag NFC appliqué sur les produits transformés permet au consommateur final, en utilisant simplement son smartphone, de voir le trajet complet du produit.

- IBM food trust [10]

IBM food trust est une plateforme cloud qui met en relation les producteurs, distributeurs et les détaillants d'une chaine d'approvisionnement. La plateforme propose plusieurs services sous forme de modules qui se présentent comme suit :

IBM food trust Trace permet de tracer de manière sure et transparente l'emplacement et le statut des produits alimentaires dans la chaine d'approvisionnement, en amont et en aval, afin de mieux gérer la demande, le gaspillage alimentaire et la fraicheur.

IBM food trust data entry and acces permet d'envoyer, télécharger, gérer et accéder à des données de transaction liées à votre activité qui seront héberger sur Bluemix.

Onboarding virtual guide c'est un module qui vous aide à maitriser l'utilisation de la solution IBM foot trust.

Les prix de chaque module varient entre 100 dollars et 5000 dollars mensuel selon la taille de votre entreprise.

L'intégration de la plateforme dans votre chaine d'approvisionnement se fait en plusieurs étapes :

1 -constitue une équipe d'expert en chaine d'approvisionnement
2 –identification de vos installations
3 -définition des scenarios possible pour les produits
4 -envoie des données sur la plateforme et configuration

Après le respect de toutes ces étapes vos clients pourront vérifier la fraicheur et la qualité de vos produits grâce à une application mobile via un code QR.

Pour utiliser IBM Food Trust vous aurez besoin d'une connexion internet et d'un navigateur web.

2. Spécificité technique des solutions actuelles

Technologie coté client final	Technologie blockchain	Technologie coté producteur, transporteur, grossiste
• Scanne Code QR • Scanne Code NFC • Application mobile	• Ethereum • Hyperledger	• Envoi de sms • Insertion information grâce interface web • ERP

Tableau 1: Spécificité techniques des solutions actuelles

Les langages et technologies suivants peuvent être utilisés pour l'implémentation des solutions :

• Scanne Code QR • Scanne Code NFC • Application mobile	• Ethereum • Hyperledger
Java, JavaScript, python, PHP	JavaScript, java, PHP, C++

Tableau 2: langage et technologie pour implémentation de solution

3. Les limites des solutions existantes par rapport à notre système

- Le prix élevé des solutions existantes
- Nécessite de mettre en place un système adapter aux réalités ivoiriennes.
- Cout de déploiement élevé dus à la nécessité des équipements agricoles moderne pour le déploiement de certaines solutions.

  Pour le déploiement de certaines solutions comme IBM food trust, il est nécessaire d'installer un système d'agriculture moderne faisant intervenir les capteurs et d'autres équipements modernes pour l'agriculture. Tout cela engendre un coût supplémentaire pour le déploiement de la solution car la majorité des producteurs ivoiriens ne disposent pas de ces équipements.
- Disponibilité de la couverture internet

  La plupart des solutions existante fonctionnent avec la connexion à internet. La mauvaise qualité ou même l'absence totale de couverture internet dans les zones rurale en Côte d'Ivoire est limité à prendre en compte et aussi le coût de la connexion internet.

PARTIE II : ANALYSE ET CONCEPTION

CHAPITRE I : METHODE D'ANALYSE ET DE CONCEPTION

Un projet informatique, quelle que soit sa taille et la portée de ses objectifs, nécessite la mise en place d'un planning organisationnel tout au long de son cycle de vie. C'est ainsi qu'est apparue la notion de méthode. Une méthode, dans le contexte informatique, peut être définie comme une démarche fournissant une méthodologie et des notations standards qui aident à concevoir des logiciels de qualité. Modéliser un système avant sa réalisation permet de mieux comprendre le fonctionnement du système. C'est également un bon moyen de maîtriser sa complexité et d'assurer sa cohérence. Un modèle est un langage commun, précis, qui est connu par tous les membres de l'équipe et il est donc, à ce titre, un vecteur privilégié pour communiquer. Cette communication est essentielle pour aboutir à une compréhension commune aux différentes parties prenantes (notamment entre la maîtrise d'ouvrage et la maîtrise d'œuvre informatique) et précise d'un problème donné.

I. Définition des concepts

1. L'analyse

Correspondant à la phase qui répond à la question « que fait le système », l'analyse est l'une des étapes les plus importantes et les plus difficiles de la modélisation. Elle permet de modéliser le domaine d'application, d'analyser l'existant et les contraintes de réalisation. Elle s'effectue par une abstraction et une séparation des problèmes. Elle peut être découpée en trois phases que sont :

- La définition des besoins : il s'agit d'identifier les acteurs et les cas d'utilisation, de structurer le modèle, et d'identifier les autres exigences.
- La capture des besoins : elle consiste à collecter des informations (interviews, lecture de documentation) et à la compréhension du domaine et du problème posé. A ce niveau il s'agit de restituer les besoins dans un langage compréhensible par le client et de procéder à l'identification, à la structuration et à la définition d'un dictionnaire.
- La spécification des besoins : il sera question d'aller à un niveau de spécification plus détaillé voire même plus formel des besoins. Elle sera d'une grande utilité pour le client mais aussi pour le développeur. A la fin de cette phase d'analyse un modèle conceptuel sera disponible, lequel modèle sera un outil fondamental lors de la phase de conception.

2. La conception

La conception vient à la suite de l'analyse des besoins. Elle met en œuvre tout un ensemble d'activités qui à partir d'une demande d'informatisation d'un processus permettent la conception, l'écriture et la mise au point d'un produit informatique (et donc de programmes

informatiques) jusqu'à sa livraison au demandeur. Elle a comme objectifs de répondre à la question « comment faire le système ?» et de décomposer de façon modulaire le système à mettre en place. La conception définit l'architecture du logiciel. Elle définit par la même occasion chaque constituant du logiciel (Informations traitées, traitements effectués, résultats fournis, contraintes à respecter. A la suite un modèle logique utilisable à la phase d'implémentation est produit.

II. Présentation des méthodes d'analyse et de conception

Pour la conception du système, nous optons pour une méthode orientée objet du fait des avantages qu'elle offre. Une méthode étant un assemblage d'une démarche et d'un langage de modélisation, nous allons choisir le tandem le plus adéquat

1. Rapid Application Development (RAD)

Rapid Application Development est une méthode de conduite de projet permettant de développer rapidement des applications de qualité. Aujourd'hui qualité et réactivité font partie des objectifs généraux de beaucoup d'entreprises. Cela entraine un certain nombre de projets, qui tout en apportant satisfaction aux utilisateurs, doivent être menés dans un délai court.

2. Dynamic Software Development Method(DSDM)

La méthode DSDM (Dynamic Software Development Method) a été mise au point en s'appuyant sur la méthode RAD afin de combler certaines de ses lacunes, notamment en offrant un canevas prenant en compte l'ensemble du cycle de développement.

3. Unified Process (UP)

La méthode du Processus Unifié (UP pour Unified Process) est un processus de développement itératif et incrémental, ce qui signifie que le projet est découpé en phases très courtes à l'issue de chacune desquelles une nouvelle version incrémentée est livrée. Il s'agit d'une démarche s'appuyant sur la modélisation UML pour la description de l'architecture du logiciel (fonctionnelle, logicielle et physique) et la mise au point de cas d'utilisation permettant de décrire les besoins et exigences des utilisateurs.[11]

4. Rational Unified Process (RUP)

Rational Unified Process (RUP) est une méthode de développement par itérations promue par la société Rational Software, rachetée par IBM. RUP propose une méthode spécifiant notamment la composition des équipes et le calendrier ainsi qu'un certain nombre de modèles de documents. RUP est l'une des plus célèbres implémentations de la démarche UP, livrée clé en main, permettant de donner un cadre de développement logiciel, répondant

aux exigences fondamentales préconisées par les créateurs d'UML. RUP est une version commerciale d'UP.

III. Choix d'une méthode d'analyse et de conception

Le choix de la démarche se fera en prenant en compte des critères essentiels de la plateforme à concevoir. Ainsi, pour des raisons d'efficacité, de rapidité et d'analyse complète, nous opterons en effet pour le processus UP (Unified Process).[12]

1. Processus unifié : cadre général

Le processus unifié est une démarche de développement logiciel : il regroupe les activités à mener pour transformer les besoins d'un utilisateur en système logiciel. Ses principales caractéristiques sont :

- Il est à base de composants ;
- Il utilise le langage UML (ensemble d'outils et de diagrammes) ;
- Il est piloté par les cas d'utilisation ;
- Il est centré sur l'architecture ;
- Il est itératif et incrémental ;

2. Vie du processus unifié

L'objectif d'un processus unifié est de maîtriser la complexité des projets informatiques en diminuant les risques. UP est un ensemble de principes génériques adapté en fonctions des spécificités des projets. UP répond aux préoccupations suivantes : QUI participe au projet ? QUOI, qu'est-ce qui est produit durant le projet ? COMMENT doit-il être réalisé ? QUAND est réalisé chaque livrable ? UP gère le processus de développement par deux axes. L'axe vertical représente les principaux enchaînements d'activités, qui regroupent les activités selon leur nature. Cette dimension rend compte l'aspect statique du processus qui s'exprime en termes de composants, de processus, d'activités, d'enchaînements, d'artefacts et de travailleurs. L'axe horizontal représente le temps et montre le déroulement du cycle de vie du processus ; cette dimension rend compte de l'aspect dynamique du processus qui s'exprime en termes de cycles, de phases, d'itérations et de jalons.

UP répète un certain nombre de fois une série de cycle qui s'articule autour de 4 phases qui sont : analyse des besoins, élaboration, construction, transition. Pour mener efficacement un tel cycle, les développeurs ont besoins de toutes les représentations du produit logiciel qui sont : un modèle de cas d'utilisation, un modèle d'analyse détaillant les cas d'utilisation et procéder à une première répartition du comportement, un modèle de conception ,finissant la structure statique du système sous forme de sous-systèmes de classes et interfaces, un modèle d'implémentation, intégrant les composants, un modèle de

déploiement ,définissant les nœuds physiques des ordinateurs, un modèle de test, décrivant les cas de test vérifiant les cas d'utilisation, et une représentation de l'architecture.

3. Les phases

Pour bien mener un projet du début à la fin, UP préconise les phases suivantes :

- Analyse

L'objectif de l'analyse est d'accéder à une compréhension des besoins et des exigences du client. Il s'agit de livrer des spécifications pour permettre de choisir la conception de la solution. Un modèle d'analyse livre une spécification complète des besoins issus des cas d'utilisation et les structure sous une forme qui facilite la compréhension (scénarios), la préparation (définition de l'architecture), la modification et la maintenance du futur système. Il s'écrit dans le langage des développeurs et peut être considéré comme une première ébauche du modèle de conception.

- Conception

La conception permet d'acquérir une compréhension approfondie des contraintes liées au langage de programmation, à l'utilisation des composants et au système d'exploitation. Elle détermine les principales interfaces et les transcrit à l'aide d'une notation commune. Elle constitue un point de départ à l'implémentation : - elle décompose le travail d'implémentation en sous-système - elle crée une abstraction transparente de l'implémentation

- Implémentation

L'implémentation est le résultat de la conception pour implémenter le système sous formes de composants, c'est-à-dire, de code source, de scripts, de binaires, d'exécutables et d'autres éléments du même type. Les objectifs principaux de l'implémentation sont de planifier les intégrations des composants pour chaque itération, et de produire les classes et les sous-systèmes sous formes de codes sources.

- Tests

Les tests permettent de vérifier des résultats de l'implémentation en testant la construction. Pour mener à bien ces tests, il faut les planifier pour chaque itération, les implémenter en créant des cas de tests, effectuer ces tests et prendre en compte le résultat de chacun.

CHAPITRE II : ANALYSE ET CONCEPTION DE NOTRE SYSTEME

Dans ce chapitre, nous présentons le système d'information que nous avons étudié en appliquant les étapes de notre méthode d'analyse et de conception. Nous nous sommes limités à présenter les étapes de l'analyse et de la conception. Ainsi, nous exposons les besoins identifiés, l'analyse et la conception de notre système.

I. Analyse

Il est assez difficile de déterminer les besoins exacts des utilisateurs. C'est pourquoi, pour mieux entamer une discussion avec ceux-ci et mieux connaitre leurs besoins, nous avons utilisé les diagrammes de cas d'utilisation et les diagrammes de séquence.

1. Diagramme de cas d'utilisation

Les diagrammes de cas d'utilisation permettent de représenter les fonctions remplies par le système, du point de vue des acteurs de son environnement. Ces diagrammes sont décrits de manière textuelle par la fiche de description textuelle. Nous notons que cette fiche n'est pas normalisée par UML.

Les acteurs de notre domaine sont :

- Les acteurs principaux : la coopérative, les producteurs, les grossistes, les distributeurs, les consommateurs ;
- Les acteurs secondaires : le système

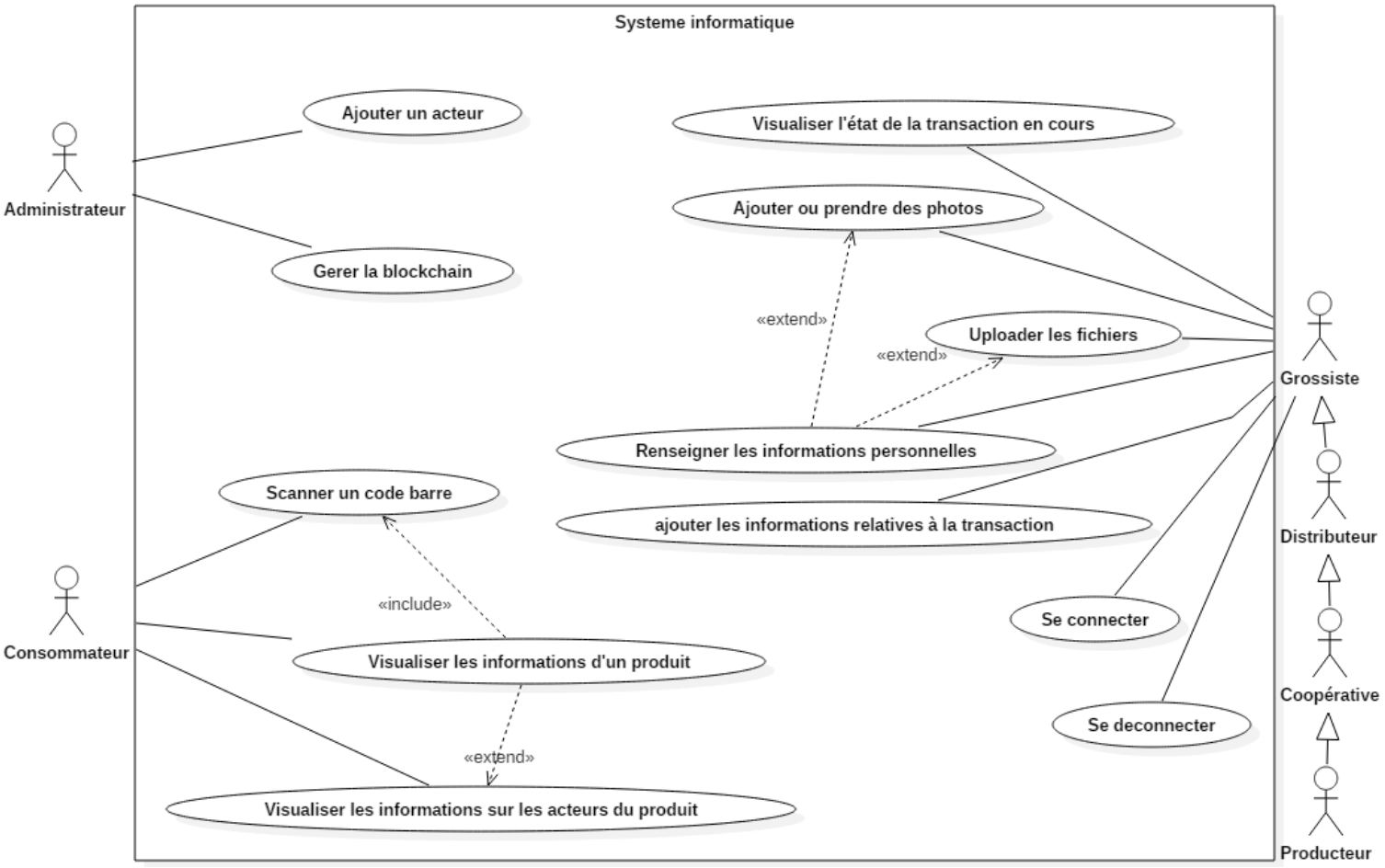

Figure 5: Diagramme de cas d'utilisation

Cas 1: Ajouter d'un acteur

Acteur	Administrateur
Résumé	L'administrateur ajoute un nouvel acteur à la plateforme
Démarrage	Un nouveau partenariat est fait
Description des scenarii	
Précondition	
Scenario nominal	1. Le système affiche la page ajout de nouveaux acteurs 2. L'administrateur sélectionne le type d'acteur à ajouter

	3. L'administrateur renseigne l'identifiant de l'acteur et ses informations de connexion 4. L'administrateur valide les informations 5. Le système notifie l'administrateur du bon déroulement
Fin	Scenario nominal au point 4

Tableau 3: Description textuelle- Cas 1

Cas 2: Renseigner les informations personnelles sur la plateforme

Acteur	Producteurs/coopératives, grossistes, distributeurs
Résumé	Les acteurs concernés enregistrent les informations sur eux
Démarrage	Après être inscrits sur la plateforme
Description des scénarii	
Précondition	Être inscrit sur la plateforme
Scenario nominal	1. Le système affiche la page sur leur compte 2. Les acteurs renseignent leurs informations 3. Les acteurs valident leurs informations 4. Le système enregistre dans le bloc
Fin	Scenario nominal au point 2 après enregistrement des données

Tableau 4: Description textuelle - cas 2

Cas 3: se connecter

Acteurs	Producteurs/coopératives, grossistes, distributeurs
Résumé	Les acteurs concernés renseignent leur login et mot de passe pour accéder à la plateforme
Démarrage	Dès ouverture de l'application/ lancement de la plateforme
Description des scenarii	
Précondition	Être enregistré sur la plateforme
Scenario nominal	1. Le système affiche la page de connexion 2. Les acteurs renseignent leur loin et mot de passe 3. Le système vérifie les informations 4. Le système ouvre l'interface dédiée selon l'acteur
Scenario d'exception	E1 : le login et le mot de passe ne sont pas retrouvés dans la base de données E1 démarre au point 2 3. Le scenario demande de ressaisie le login et le mot de passe Le scenario reprend au point 3 jusqu'à 3 reprise et l'application se ferme
Fin	Scenario nominal au point 3 Scenario d'exception au point 3

Tableau 5:Description textuelle - cas 3

Cas 4 : Se déconnecter

Acteur	Producteurs/coopératives, grossistes, distributeurs
Résumé	Les acteurs se déconnectent de la plateforme
Démarrage	Demande de déconnexion
Description des scenarii	
Précondition	Être connecté
Scenario nominal	1. Les acteurs demandent à se déconnecter 2. Le système affiche l'interface de déconnexion 3. Le système ferme la session de l'acteur 4. Le système affiche la page d'accueil
Fin	Scenario nominal au point 4

Tableau 6: Description textuelle - cas 4

Cas 5 : ajouter les informations relatives à la transaction

Acteur	Producteurs/coopératives, grossiste,
Résumé	Les acteurs renseignent l'un à la suite de l'autre les informations sur la production en cours
Démarrage	Les acteurs demandent à lancer une transaction
Description des scenarii	
Précondition	S'être connecté sur la plateforme

	Avoir initialisé une transaction avec l'acteur
Scenario nominal	1. Saisir les informations relatives à la transaction en cours 2. Présélectionner les acteurs liés à cette transaction 3. Valider la transaction
Fin	Scenario nominal au point 3

Tableau 7: Description textuelle - cas 5

Cas 6 : Ajouter/ prendre les photos

Acteur	Producteur/coopérative, grossiste, distributeur
Résumé	Les acteurs peuvent ajouter / prendre des photos de leur activités
Démarrage	Demande d'ajout / de prise de photos
Description des scenarii	
Précondition	
Scenario nominal	1. Le système ouvre la ressource « Appareil Photo » 2. Les acteurs prennent les photos 3. Les acteurs valident les photos 4. Le système affiche les photos
Scenario d'exception	E1 : la photo n'est pas correctement visible E1 démarre au point 2 3. l'acteur annule la prise de photo Le scenario nominal reprend au point 2

	E2 : un problème lors de l'enregistrement E2 démarre au point 3 4. Le système notifie sur l'erreur d'enregistrement Le scenario reprend au point 2
Fin	Scenario nominal au point 4

Tableau 8: Description textuelle - cas 6

Cas 7 : Uploader les documents de droits (certificat de propreté, de culture)

Acteurs	Producteurs/coopératives, grossistes, distributeurs
Résumé	Les acteurs peuvent uploader des documents permettant de légaliser et de certifier leur exercice
Démarrage	Après avoir renseigné les informations de l'acteur
Description des scenarii	
Précondition	Avoir renseigner les informations personnelles de l'acteur
Scenario nominal	1. L'acteur sélectionne le fichier PDF de ses dossiers 2. Le système vérifie l'extension du fichier 3. Le système vérifie la taille du fichier 4. Le système notifie l'acteur que le fichier a été bien ajouté
Scenario alternatif	

Scenario d'exception	E1: extension n'est pas adéquate, 2. le système notifie que l'extension n'est pas adéquate 3. retour au point 1 E2 : la taille du fichier est trop élevée, 3. le système notifie que la taille du fichier est élevée 4. Retour au point 1
Fin	Scenario nominal au point 4

Tableau 9:Description textuelle - cas 7

Cas 8 : Visualiser l'état de la transaction en cours

Acteur	Producteur/coopérative, grossiste, distributeur
Résumé	Les acteurs peuvent visualiser l'état de la transaction en cours
Démarrage	Les acteurs demandent à voir l'état de la transaction en cours
Description des scenarii	
Précondition	Être une partie prenante de la transaction Être connecté
Scénario nominal	1. Le système affiche la page relative à la transaction 2. L'acteur visualise l'état de la transaction
Fin	Point 2, Scénario nominal

Tableau 10: Description textuelle - cas 8

Cas 9 : Scanner le code barre du produit

Acteur	Consommateur
Résumé	Le consommateur scanne le code barre du produit
Démarrage	Des lancements de l'interface « consommateur »
Description des scenarii	
Précondition	Aucune
Scenario nominal	1. Le système affiche la page scan produit 2. Le système ouvre la ressource « Appareil photo » 3. Le consommateur effleure le code barre à l'aide du système 4. Le système vérifie le code barre du produit 5. Le système notifie le consommateur de la présence du produit dans la base de données
Fin	Point 5

Tableau 11: Description textuelle - cas 9

Cas 10 : Visualiser les informations concernant un produit

Acteur	Consommateur
	Le consommateur visualise toutes les informations du produit depuis le producteur jusqu'au distributeur
Démarrage	Dès lancement de la plateforme
Description des scenarii	
Précondition	Scanner le produit

Scenario nominal	1. Le système affiche les informations sur le produit 2. Le consommateur valide
Fin	Scenario nominal au point 2

Tableau 12:Description textuelle - cas 10

Cas 11 : Visualiser les informations sur les acteurs du produit

Acteur	Consommateur
Résumé	Le consommateur visualise les informations sur les acteurs ayant permis la transaction du produit
Démarrage	Demande les infos sur les acteurs
Description des scenarii	
Précondition	
Scenario nominal	1. Sélectionner l'instance désirée 2. Le système affiche toutes les informations 3. Le consommateur valide
Scenario alternatif	A1 : le consommateur visualise les certificats ou documents légaux A1 démarre au point 2 3. Le système affiche les documents Le scenario nominal reprend au point 3
Fin	Scenario nominal : au point 3

Tableau 13:Description textuelle - cas 11

2. Diagramme de séquence

Ils permettent de représenter les interactions entre objets, acteurs ou instances en précisant la chronologie de l'échange des messages.

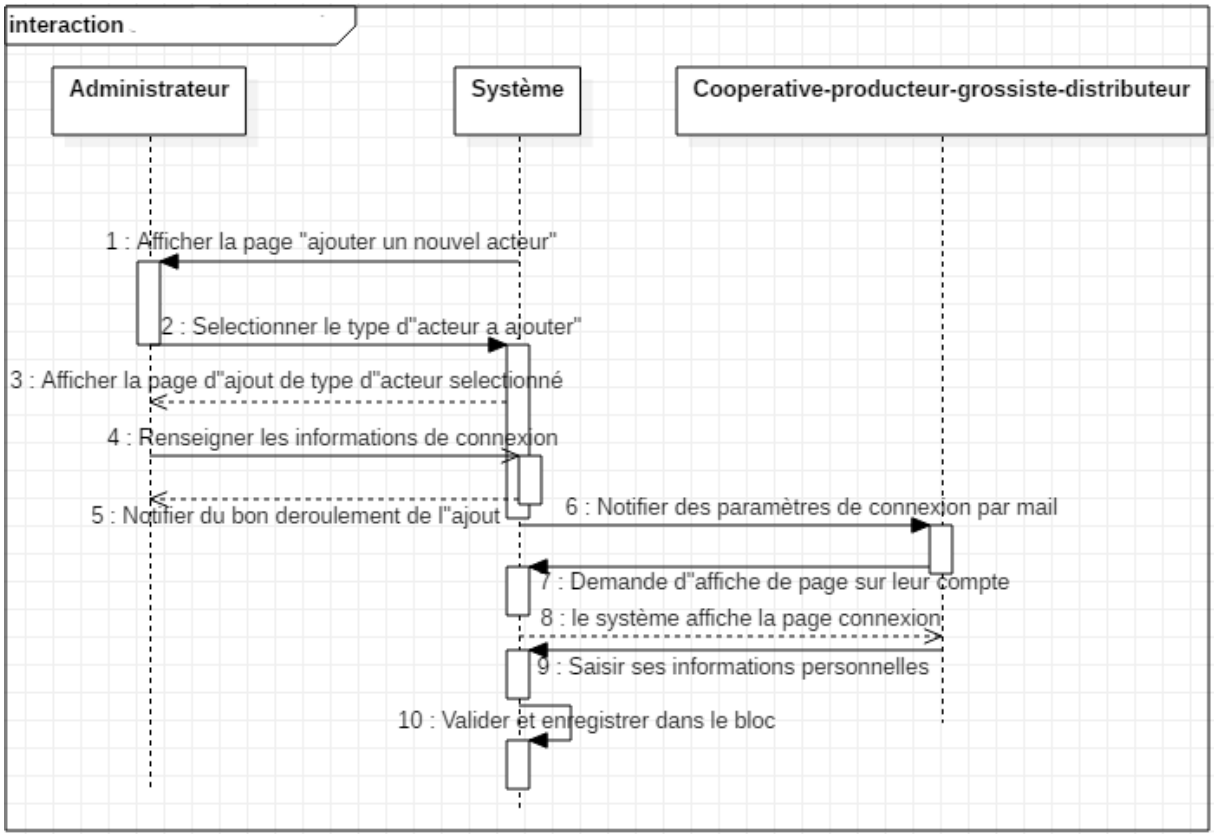

Figure 6: Diagramme de séquence- Ajouter acteur

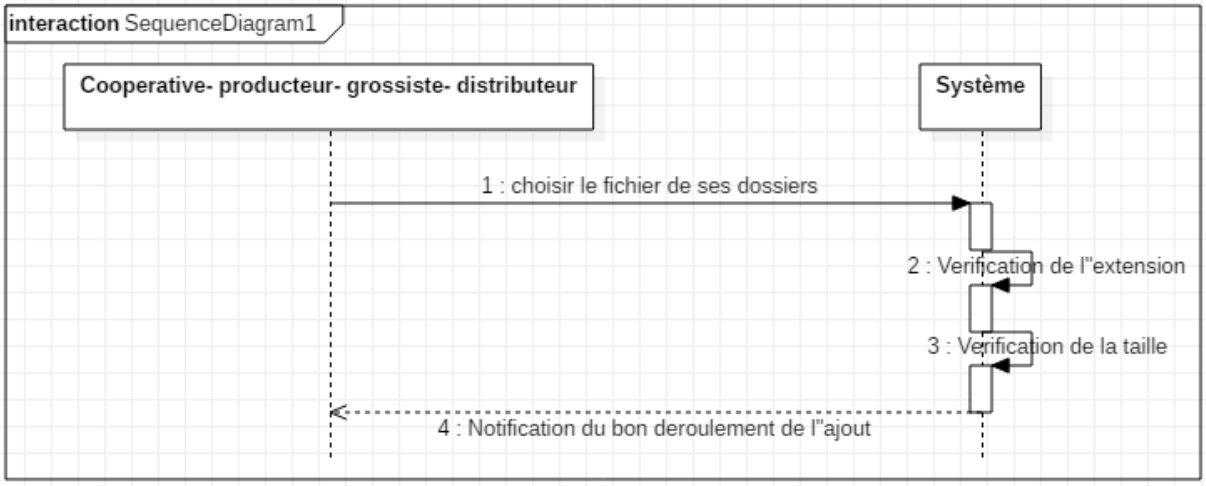

Figure 7: Diagramme de séquence - ajouter document

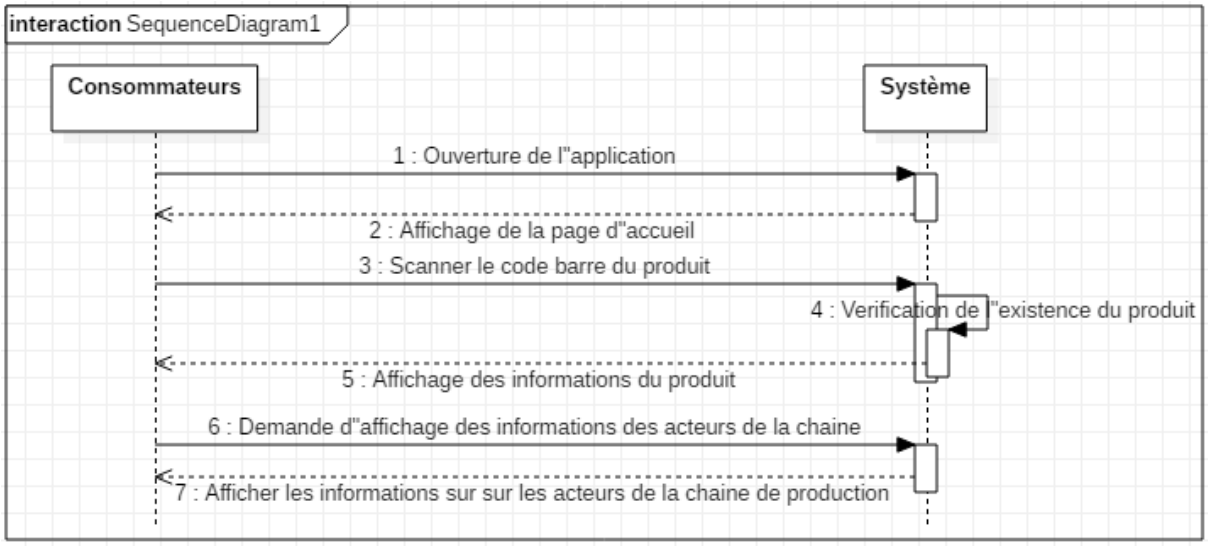

Figure 8: Diagramme de séquence- visualiser information produit

II. Conception

1. Architecture opérationnelle

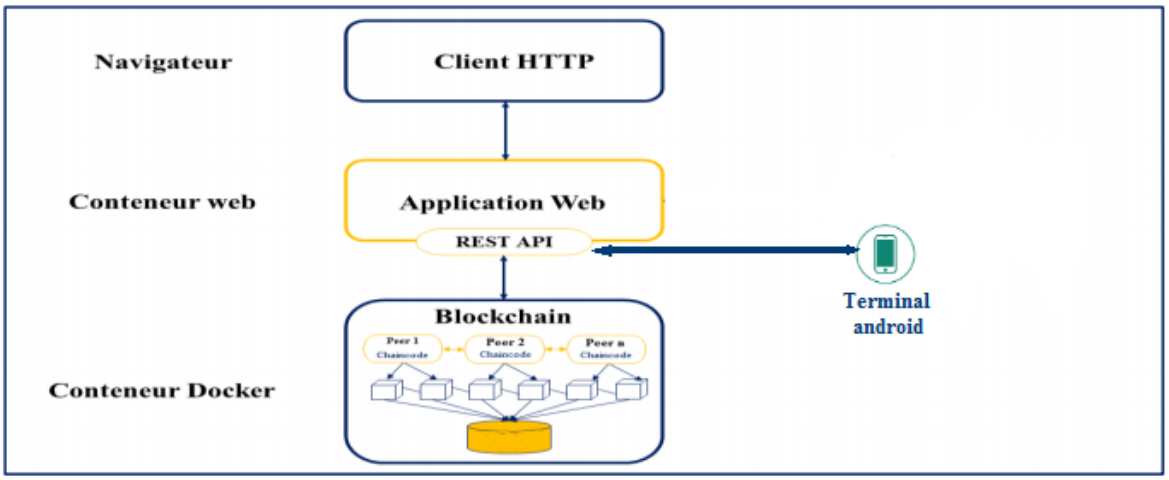

Figure 9: Architecture operationnelle

Cette architecture se compose de quatre principales parties :

• **Un client HTTP :** Cette partie est l'illustration du navigateur web qui favorisera l'accès des interfaces de l'application aux utilisateurs.

• **Un conteneur Web :** Ce conteneur héberge notre application, les contrôleurs et les différentes méthodes d'accès aux données.

• **Un terminal android :** Pour consulter l'historique d'un produit le terminal mobile va interagir avec le réseau blockchain à travers le REST API.

• **Un réseau Blockchain :** Le réseau de Blockchain, hébergé dans un conteneur Docker, permet le déploiement de la logique métier, des smart contracts, du registre distribué (incluant la chaîne de blocs). Cette configuration opérationnelle nous permet d'obtenir une architecture applicative assez détaillée.

Cette configuration opérationnelle nous permet d'obtenir une architecture applicative assez détaillée.

2. Architecture applicative

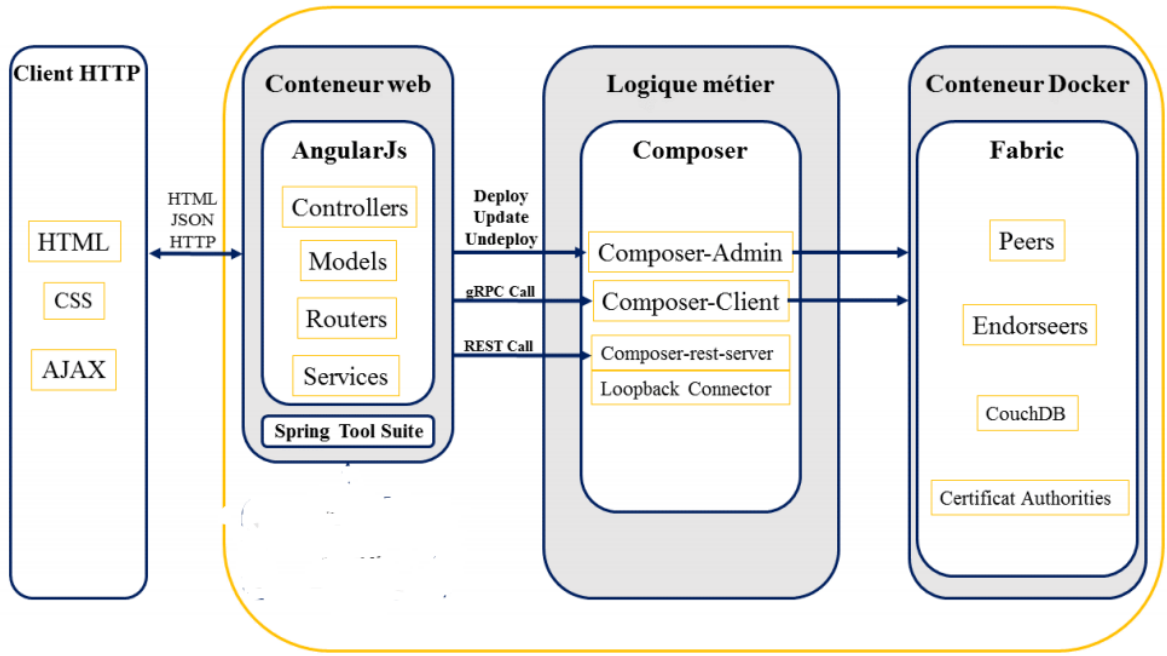

Figure 10: Architecture applicative

Regroupant trois principaux *blocs*, cette architecture est le reflet de l'application de notre système. Les trois principaux blocs sont les suivants :

- **L'application AngularJs** Cette partie de notre architecture se chargera de recueillir et d'afficher à l'utilisateur toutes les actions effectuées sur le réseau (*en fonction de ses droits d'accès*). Elle sera donc connectée à notre réseau déployé dans Hyperledger Fabric à travers la logique métier défini dans Hyperledger Composer.
- **Composer** est la partie applicative du framework Hyperledger. Il nous permet de définir le modèle de données, la logique métier et les accès aux données à partir d'un fichier de type ACL. Il nous fournira aussi un REST API (*composer rest-api*) pour accéder à Fabric.
- **Fabric** est la partie réseau du framework Hyperledger. Après le développement de la logique métier et des smart contracts avec Composer, Fabric servira au

déploiement dans un réseau composé de nœuds, d'autorités de certification (le consensus) et du registre distribué (CouchDB).

3. Diagramme de classe

Le diagramme de classe est la description statique de notre système par l'intégration au niveau de chacune des classes la partie dédiée aux données et celle consacrée aux traitements. C'est le diagramme autour duquel pivotent l'ensemble de la modélisation du système.

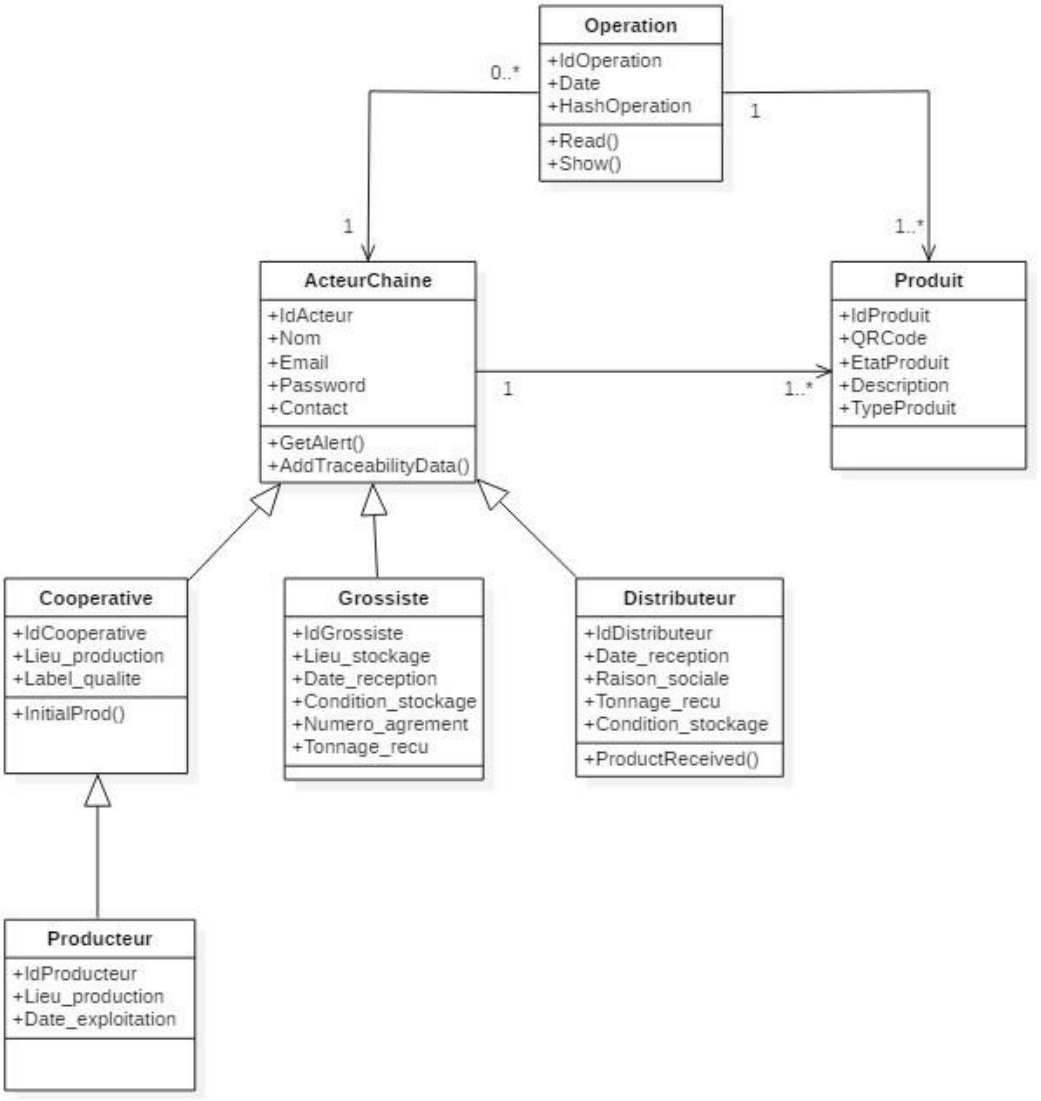

Figure 11: Diagramme de classe

Ce diagramme illustre les différentes relations entre les acteurs et les ressources du système. Il est donc composé de :

- **Produit** : est l'ensemble des produits présents dans notre réseau dont il faut assurer la traçabilité,

- **Opération** : qui regroupe l'ensemble des transactions effectuées par les participants de notre réseau,
- **Coopérative** : regroupe les responsables de la coopérative qui ont un rôle de validateur des produits à l'entrée. C'est d'ailleurs la seule validation humaine possible dans notre processus
- **Producteur** : regroupe l'ensemble des agriculteurs
- **Grossiste** : regroupe les potentiels acheteurs des produits mise en vente par les coopératives et/ou producteurs,
- **Distributeur** : qui regroupe les potentiels acheteurs des produits mise en vente par les grossistes.

En présentant les différentes associations entre les acteurs du système et les différentes cardinalités, ce diagramme de classe nous permet d'appliquer la meilleure des politiques de sécurité à chaque segment du processus.

1. **InitialProd :** Pour la création d'un produit. Cette transaction est exécutée par les cooperatives ou producteurs.

2. **Show :** Cette section permet de consulter le statut d'une transaction. Elle est effectuée uniquement par le participant courant.

3. **Read :** Cette section permet de consulter les différentes opérations (transactions) effectuées par les participants du réseau.

4. **GetAlert :** Après approbation du produit ou de la transaction, les différents participants sont notifiés mais aussi en cas de non-validation.

5. **AddTraceabilityData :** cette transaction est à effectuer par les participants du réseau. Ils renseignent les informations de traçabilité qui les concerne.

6. **ProductReceived :** Cette transaction marque la fin du processus d'acheminement du produit. Elle est exécutée par le distributeur

PARTIE III : REALISATION DE LA SOLUTION

CHAPITRE 1 : ENVIRONNEMENT DE TRAVAIL

Ce chapitre est consacré à la présentation de l'environnement de travail qui a permis le développement du système de Traçabilité alimentaire tel que défini dans le chapitre précédent. Pour un meilleur agencement du chapitre, nous entamerons par la présentation de l'environnement de développement ensuite terminer par la présentation des technologies utilisées

I. ENVIRONNEMENT DE DEVELOPPEMENT

1. Ressources matérielles

Côté matériel, nous avons eu recours à un ordinateur disposant des capacités suivantes :

- Marque : HP Pavillon
- Système d'exploitation : Ubuntu 1.04 x64 / Windows 8.1 Enterprise
- Processeur : Intel Core I3 – 5005U 2.00GHz
- Capacité de disque dur : 500 Go
- RAM : 8.00 GB

2. Ressources logicielles

Côté logiciel, nous avons utilisé un minimum d'outils car des logiciels supplémentaires sont inclut dans les frameworks Blockchain de Hyperledger. De ce fait, nous avons eu recours aux éditeurs :

- VSCode qui simplifie les configurations Backend,
- ATOM pour la partie front-end.

En plus de ces deux éditeurs, nous avons fait usage de Hyperledger Composer Playground. Hyperledger Composer Playground fournit une interface utilisateur pour la configuration, le déploiement et le test d'un réseau Blockchain. Les fonctionnalités avancées de Playground permettent de gérer la sécurité du réseau, d'inviter des participants à des réseaux professionnels et de se connecter à plusieurs réseaux Blockchain.

II. Technologies utilisées

1. Langage de programmation

1.1. JavaScript

Javascript est un langage de programmation dynamique. Il est léger et le plus couramment utilisé dans le cadre de pages Web, dont les implémentations permettent au script côté client d'interagir avec l'utilisateur et de créer des pages dynamiques. C'est un langage de programmation interprété avec des capacités orientées objet. Ce langage nous a servi à coder nos smart contracts[13]

.

1.2. TypeScript

TypeScript est un langage de programmation basé sur JavaScript. Ce n'est pas un langage complet, mais plutôt une couche qui ajoute de nouvelles fonctionnalités à JavaScript telles que les interfaces et les génériques [14] . Le plus important d'entre eux est évidemment le typage fort, d'où le nom est dérivé. Nous avons utilisé ce langage pour le développement de notre front-end.

1.3. Yaml

Yaml est un langage de sérialisation de données conçu pour être lisible par des humains et compatible avec les langages de programmation modernes. Son but est de représenter des informations de façon plus élaborées en gardant une lisibilité presque comparable à un fichier CSV, et bien plus grande en tout cas que du XML. Les fichiers de configurations de Hyperledger Fabric utilisent ce langage.

1.4. Script Shell

Il s'agit d'un programme que tout utilisateur d'Unix utilise. C'est celui qui prend les commandes de l'utilisateur et les passe au système d'exploitation pour être exécutées. Il affiche pour cela un prompt. Le script shell sera utilisé dans la quasi-totalité des étapes de réalisation, cependant nous y mettrons un accent particulier pour la phase de déploiement.

1.5. Langage de modélisation (.cto)

Hyperledger Composer inclut un langage de modélisation orienté objet avec l'extension .cto utilisé pour définir le modèle de domaine pour une définition de réseau. C'est un langage propre à Hyperledger Composer

2. Framework et outils

2.1. Hyperledger Composer

Hyperledger Composer est un ensemble d'outils de collaboration pour la création de réseaux Blockchain qui simplifient et accélèrent la création de contrats intelligents et d'applications de Blockchain pour résoudre les problèmes métier. Conçu avec JavaScript, il s'appuie sur des outils modernes tels que node.js, npm, CLI et des éditeurs populaires (Visual Studio Code, Atom, …). Composer propose des abstractions orientées métier ainsi que des exemples d'applications avec des processus « devops » faciles à tester pour créer des solutions Blockchain robustes.

2.2. Hyperlegder Fabric

Hyperledger Fabric est un framework d'implémentation de Blockchain hébergé par « The Linux Foundation ». Conçu comme une base pour le développement d'applications ou de solutions avec une architecture modulaire, Hyperledger Fabric permet aux composants, tels que les services de consensus et d'appartenance, d'être plug-and-play. Hyperledger Fabric exploite la technologie des conteneurs pour héberger des contrats intelligents appelés "chaincode" qui constituent la logique applicative du système.

2.3. Angular 2

Angular 2 est une plate-forme Web frontale open-source développée en TypeScript. C'est une plate-forme qui facilite la création d'applications Web. Angular 2 combine des modèles déclaratifs, des injections de dépendance, des outils de bout en bout et de meilleures pratiques intégrées pour résoudre les problèmes de développement. Angular 2 permet aux développeurs de créer des applications qui tournent sur le Web et le mobile. [15]

2.4. Docker

Docker est une plateforme qui permet l'exécution d'un code à l'intérieur d'un conteneur indépendamment du support matériel. Un conteneur ressemble à une machine virtuelle sauf qu'il n'embarque pas tout un système d'exploitation avec lui ce qui lui permet de s'exécuter en quelque secondes et d'être beaucoup plus léger. En tirant parti des méthodologies de Docker pour l'envoi, le test et le déploiement rapides du code, on réduit considérablement le délai entre l'écriture du code et son exécution en production. Docker résout donc les problèmes d'environnement, car quelle que soit la machine utilisée, le code s'exécute de la même manière. [16]

2.5. Docker-compose

Docker-Compose est un outil de définition et d'exécution d'applications complexes avec Docker. Avec Docker-Compose, on définit une application multi-conteneurs dans un seul fichier (Dockerfile), puis on fait tourner l'application en une seule commande qui fait tout ce qui doit être fait pour le faire fonctionner. [17]

CHAPITRE II: RESULTATS ET DISCUSSIONS

Dans ce chapitre nous procéderons à l'installation et au déploiement de notre solution. Et nous terminerons par la présentation de celle-ci et ces limites.

I. Résultats

1. Installation de l'environnement

Ce volet regroupe les différentes phases qui ont abouti à la réalisation de notre solution.

1.1. Installation de hyperledger composer, Fabric playground et Yoman

La première phase est la préparation de notre environnement de travail en installant les modules nécessaires au déploiement de notre réseau.

1.1.1 Installation prérequise

Les conditions préalables à l'installation des outils de développement requises sont les suivantes :

Systèmes d'exploitation : Ubuntu Linux 14.04 / 16.04 LTS (64 bits) ou version supérieure

- Moteur Docker: version 17.03 ou supérieure
- Docker-Compose: Version 1.8 ou supérieure
- Node: 8.9 ou supérieur (la version 9 et supérieure n'est pas prise en charge)
- npm: v5.x
- git: 2.9.x ou supérieur
- Python: 2.7.x
- L'éditeur de code VSCode

Sur Ubuntu, on télécharge le fichier *prereqs-ubuntu.sh***,** on modifie les droits d'exécution puis on lance le fichier

```
curl -O https://hyperledger.github.io/composer/latest/prereqs-ubuntu.sh

chmod u+x prereqs-ubuntu.sh
./prereqs-ubuntu.sh
```

1.1.2 Installation des outils essentiels

a.Outils essentiels de la CLI:

```
npm install -g composer-clic@0.20
```

b. Utilitaire permettant d'exécuter un serveur REST sur votre ordinateur afin d'exposer vos réseaux d'entreprise en tant qu'API RESTful:

```
npm install -g composer-rest-server@0.20
```

c. Utilité utile pour générer des actifs d'application :

```
npm install -g generator-hyperledger-composer@0.20
```

d. Yeoman est un outil de génération d'applications qui utilise generator-hyperledger-composer :

```
npm install -g yo
```

1.2. Installation de playground

Hyperledger Composer Playground fournit une interface utilisateur pour la configuration, le déploiement et les tests d'un réseau d'entreprise. Les fonctionnalités de jeu avancées permettent aux utilisateurs de gérer la sécurité du réseau d'entreprise, d'inviter des participants aux réseaux d'entreprise et de se connecter à plusieurs réseaux d'entreprise blockchain.

Commande d'installation :

```
npm install -g composer-playgroung@0.20
```

1.3. Installation et configuration notre IDE

Nous avons téléchargé le fichier *.deb de VSCode* sur leur site officiel et nous l'avons installé avec la commande suivante:

```
sudo dpkg -i
```

a. Installer le VSCode à partir de cette URL : https://code.visualstudio.com/download
b. Rechercher et installer l'Hyperledger Composer extension à partir de Marketplace de VSCode.

1.4. Installation de Hyperledger Fabric

Cette étape nous a donné un environnement d'exécution Hyperledger Fabric local pour le déploiement de notre réseau d'entreprise.

On crée un répertoire Fabric-dev-servers pour installer nos fichiers. Ensuite dans le répertoire *fabric-dev-servers*, récupérez le fichier .tar.gz contenant les outils pour installer Hyperledger Fabric et on décompresse le fichier:

```
mkdir ~/fabric-dev-servers && cd ~/fabric-dev-servers

curl -O https://raw.githubusercontent.com/hyperledger/composer-tools/master/packages/fabric-dev-servers/fabric-dev-servers.tar.gz
tar -xvf fabric-dev-servers.tar.gz
cd ~/fabric-dev-servers
export FABRIC_VERSION=hlfv12
./downloadFabric.sh
```

1.5. Contrôler notre environnement de développement

```
cd ~/fabric-dev-servers
export FABRIC_VERSION=hlfv12
./startFabric.sh
./createPeerAdminCard.sh
```

Nous pouvons démarrer et arrêter notre runtime *avec ~/fabric-dev-servers/stopFabric.sh*, et le redémarrer avec *~/fabric-dev-servers/startFabric.sh*.

1.6. Démarrer l'application Web Playground

Pour démarrer l'application Web, on exécute : *composer-playground*
Il ouvrira généralement votre navigateur automatiquement, à l'adresse suivante : *http://localhost: 8080 / login*

2. Déploiement de la solution

2.1. Développement du système avec Composer

Après installation du Hyperledger Fabric et le lancement de playground voici l'interface de *hyperledger composer*:

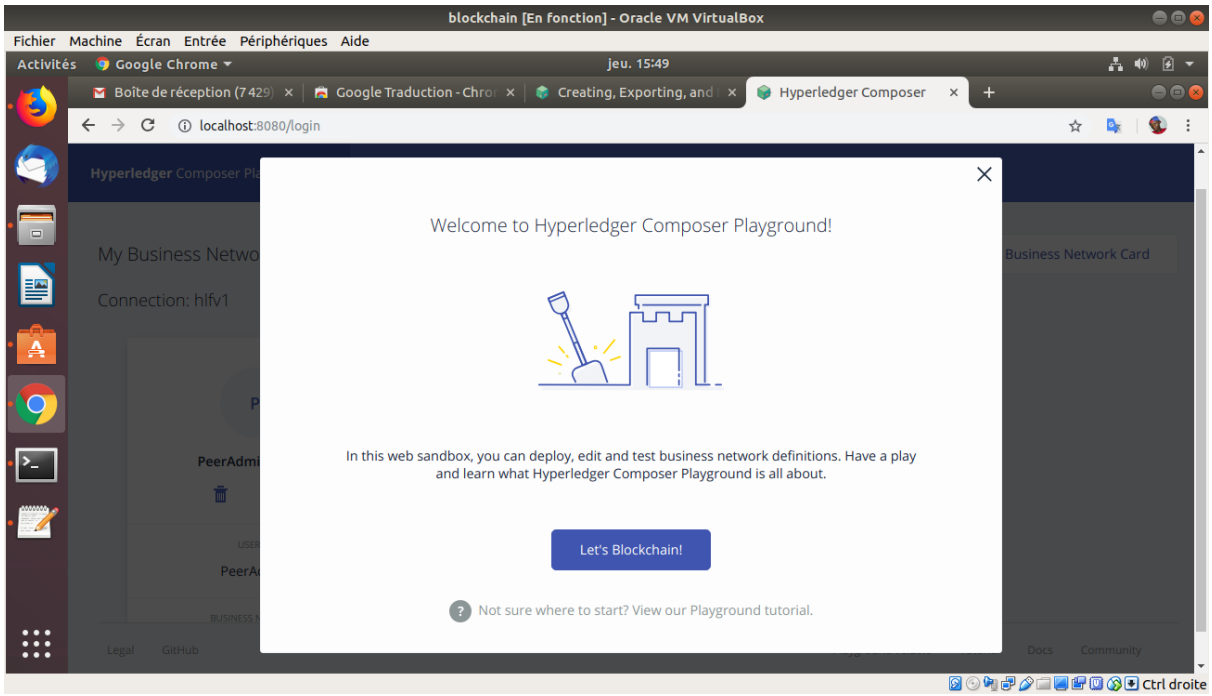

Figure 12: Interface de Playground

2.2. Création de notre réseau de blockchain

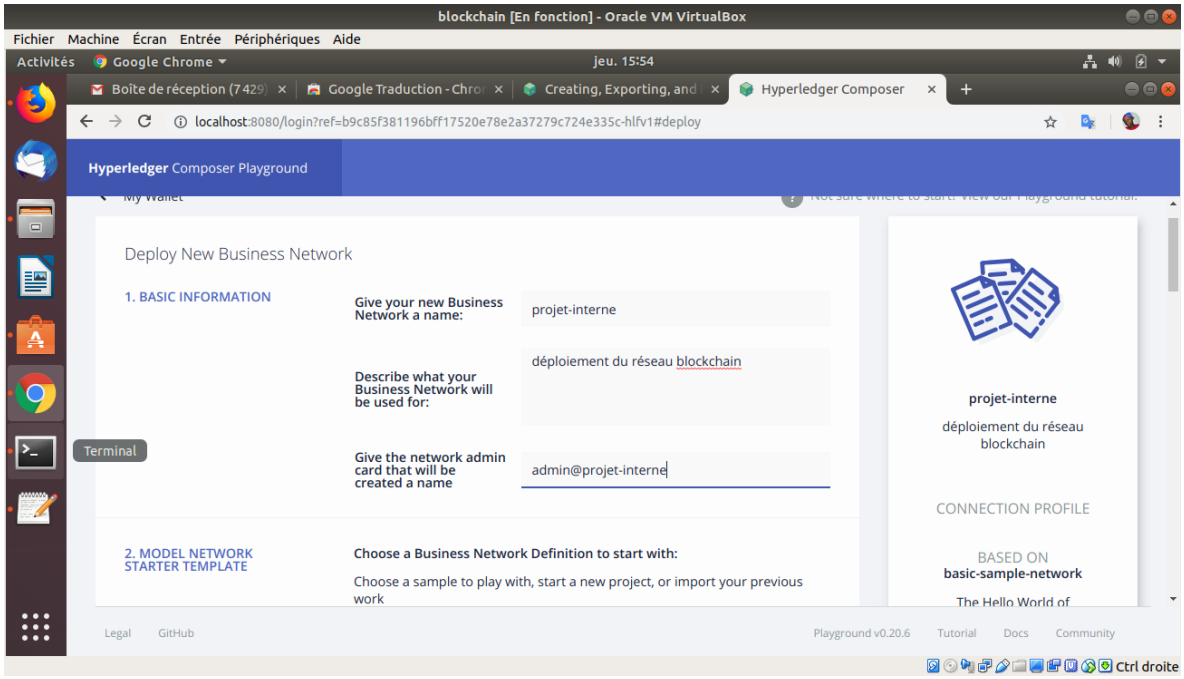

Figure 13: création de nouveaux réseaux

2.3. Différents types de fichiers avec Hyperledger composer

Hyperledger composer fournit différents fichiers qui nous permettent de construire notre réseau Blockchain. Nous vous en présentons les principaux :

- **Model file (.cto)**

Ce fichier permet de décrire les actifs, les participants, les transactions et les événements à l'aide d'un langage de modélisation propre à Hyperledger Composer. Ce langage de modélisation inclut les relations, les tableaux et les règles de validation

- **Script file (.js)**

Ce fichier d'extension JavaScript définit la logique des exécutions de transactions. Il sert au codage des smart contracts qui vont contrôler les transactions définis dans le fichier Model file. Le fichier Script file revêt d'une grande importance dans notre projet car il permettra d'appliquer une politique de sécurisation pour chacune des transactions à effectuer.

- **Access Control (.acl)**

Il s'agit ici d'un fichier de contrôle d'accès d'extension **.acl** qui va définir les droits d'accès de chaque utilisateur à une ressource donnée. Tout comme le Script file, ce fichier prend une part active dans la sécurisation de notre système en appliquant des restrictions sur certaines informations. Ce fichier va donc définir des règles de partage et de confidentialité.

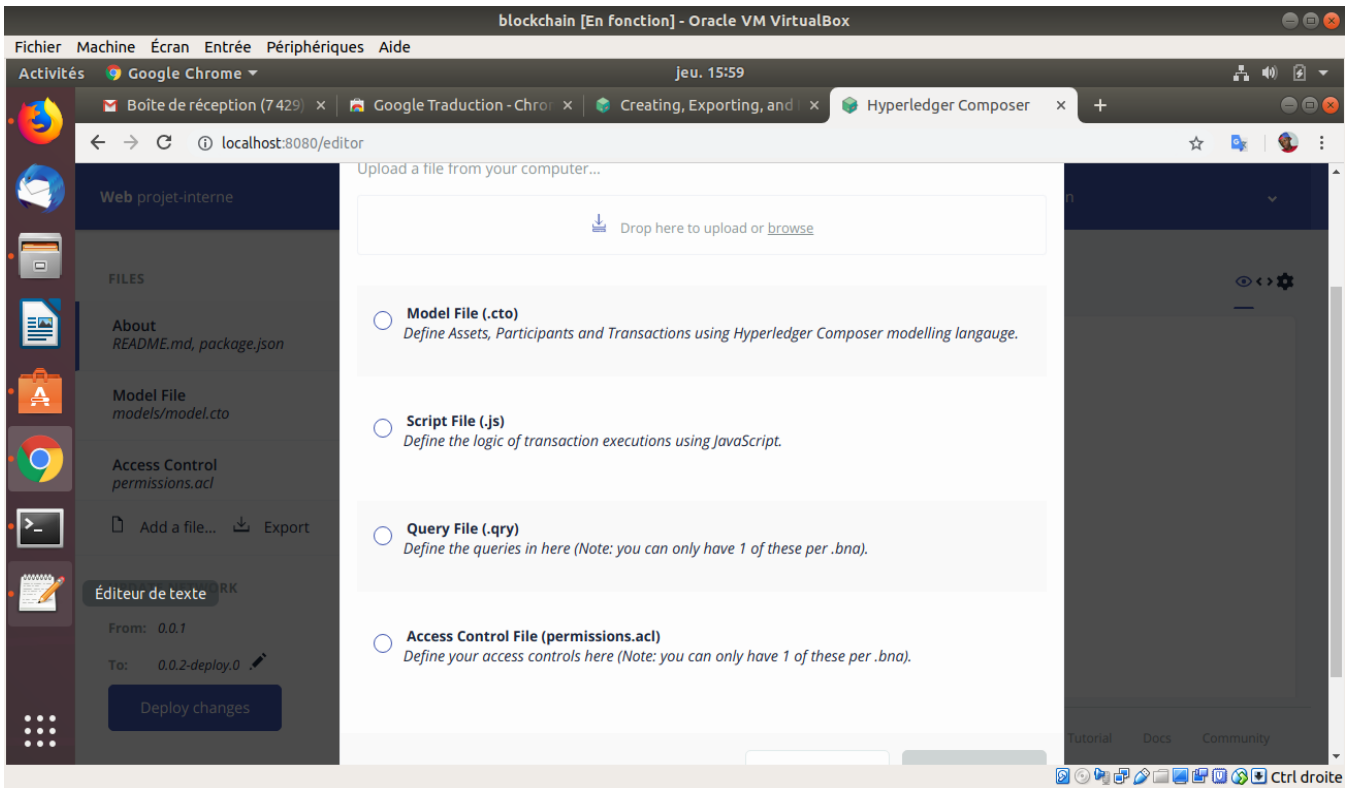

Figure 14: Differents fichiers de Hyperledger Composer

Ce sont là les principaux fichiers que nous avons utilisé pour la construction de notre système de traçabilité. À présent, nous passons aux principaux développements avec ces différents fichiers.

2.4. Développement de la logique « métier »

La première étape est la création du Model file (que nous avons renommé sous le nom *foodtracemodel.cto*) pour la définition des participants, des actifs et des transactions du réseau. Ce fichier contient l'ensemble des transactions de traçabilité. C'est la partie métier de notre développement sur laquelle les contrôleurs (smart contracts) vont agir.

Dans ce fichier, nous avons défini quatre catégories de participants :

- Les producteurs
- Les grossistes
- Les distributeurs
- Les coopératives

Concernant les actifs, nous avons procédé à la création d'un seul fichier : le fichier produit.

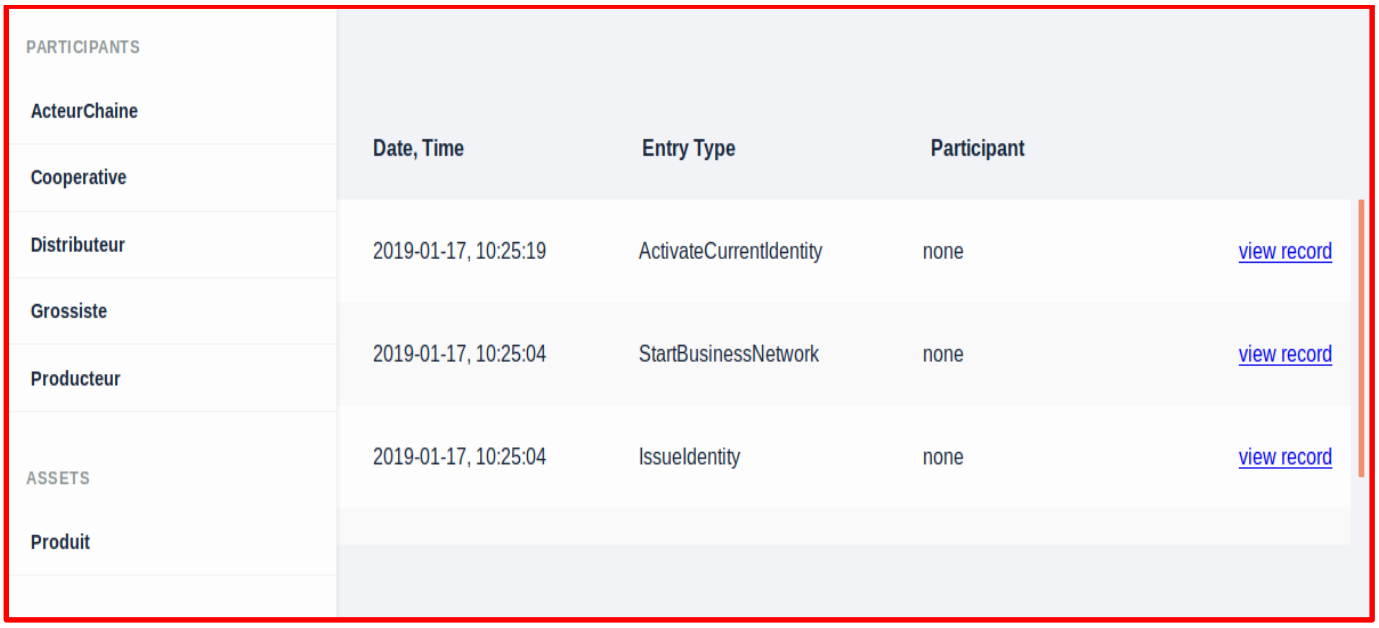

Figure 15: Les participants et les produits

Après la création des participants et des transactions, l'étape suivante est la création des smart contrats qui vont contrôler les transactions afin de les automatiser et mieux les sécuriser. Nous avons réalisé ces smart contracts en JavaScript.

2.5. Développement des « smart contracts »

Pour contrôler toutes les transactions le long de notre chaîne d'approvisionnement, nous avons eu recours à cinq smart contracts. Ces smart contracts agissent sur chacune des principales étapes du processus. Ainsi, on a :

produit.js : pour le contrôle de la création, la validation et la mise en ligne d'un *produit*,

producteur.js : pour la création et le contrôle des comptes des *producteurs*,

cooperative.js : pour la création et le contrôle des comptes des *coopératives*,

distributeur.js : pour la création et le contrôle des comptes des *distributeurs*,

grossiste.js **:** pour la création et le contrôle des comptes des *grossistes*,

2.6. Définition des droits d'accès

La définition des droits d'accès se fait dans le fichier *permissions.acl*. Ce fichier d'extension *.acl* est important dans le cadre de notre projet. En effet, toute la politique de sécurisation externe sera développée dans ce fichier. On y spécifie les droits d'accès de chaque mais aussi les droits de modification, de suppression, d'ajout, … Ce fichier assure la sécurisation à l'entrée avant de laisser le soin aux smart contracts pour la sécurisation interne des processus.

2.7. Création des « wallets »

Après avoir créé les smart contracts et défini les droits d'accès, il nous faut créer des cartes (qui représentent les comptes) associées à chaque participant. Cette opération permettra de tracer les opérations des acteurs du réseau. la figure suivante montre la phase de création du wallet d'un client.

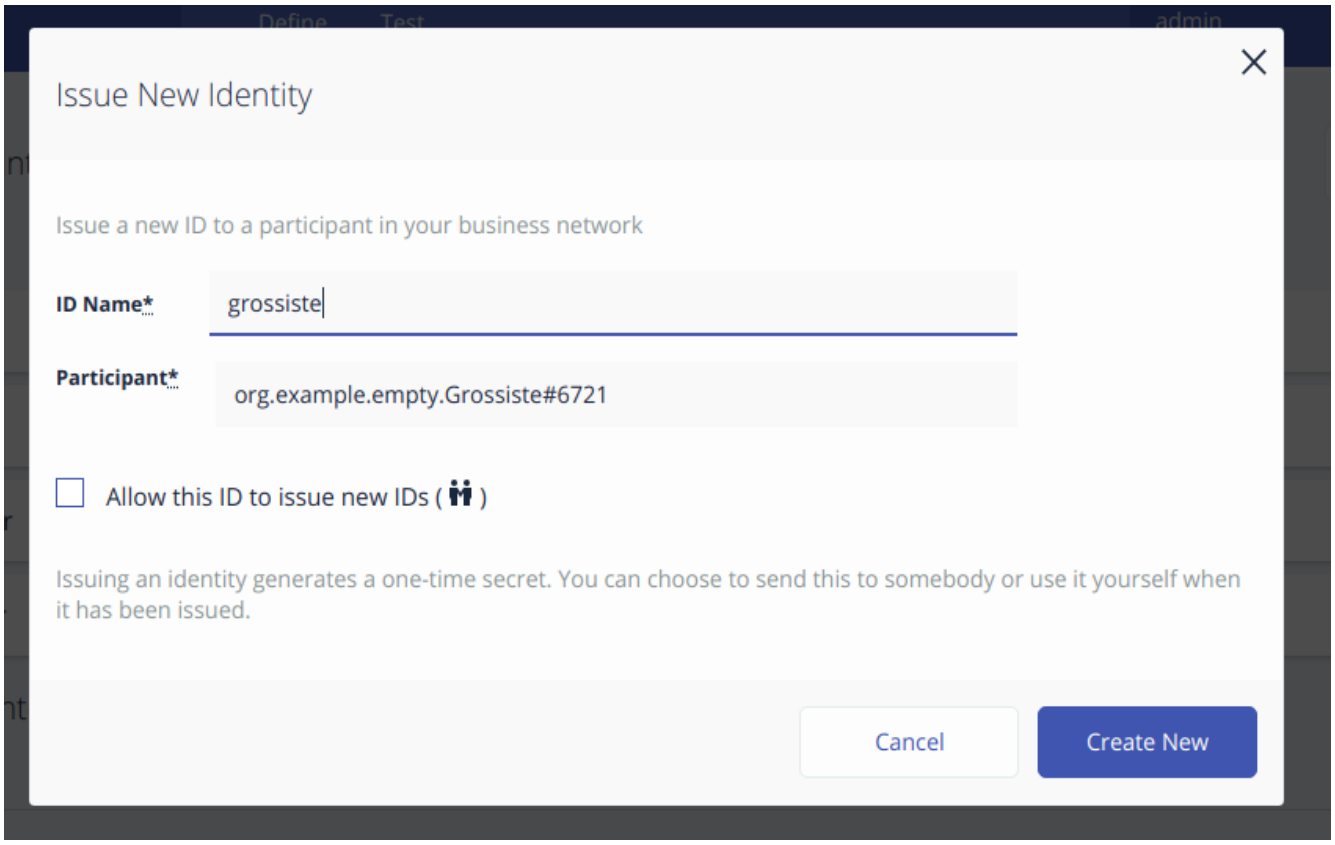

Figure 16: création Wallet

En effet, pour chaque action menée, cet identifiant sera ajouté aux données de la transaction pour constituer un bloc du réseau de Blockchain. À chaque opération, on associe la date, le nom de l'opération et l'acteur l'ayant effectuée. Il est donc impossible d'usurper une identité, ou d'effectuer des transactions sans être repéré.

3. Déploiement dans Hyperledger Fabric

3.1. Configuration des nœuds du réseau Blockchain

Les nœuds de notre réseau sont configurés dans le fichier *docker-compose.yml***.** Ce fichier permet de lancer nos différents conteneurs des composants de Fabric, car cette approche est la plus simple. En effet, dans ce fichier, nous définissons un service multi-conteneur, puis nous transformons ce service en une seule commande qui fait tout ce qui doit être fait pour la faire fonctionner.

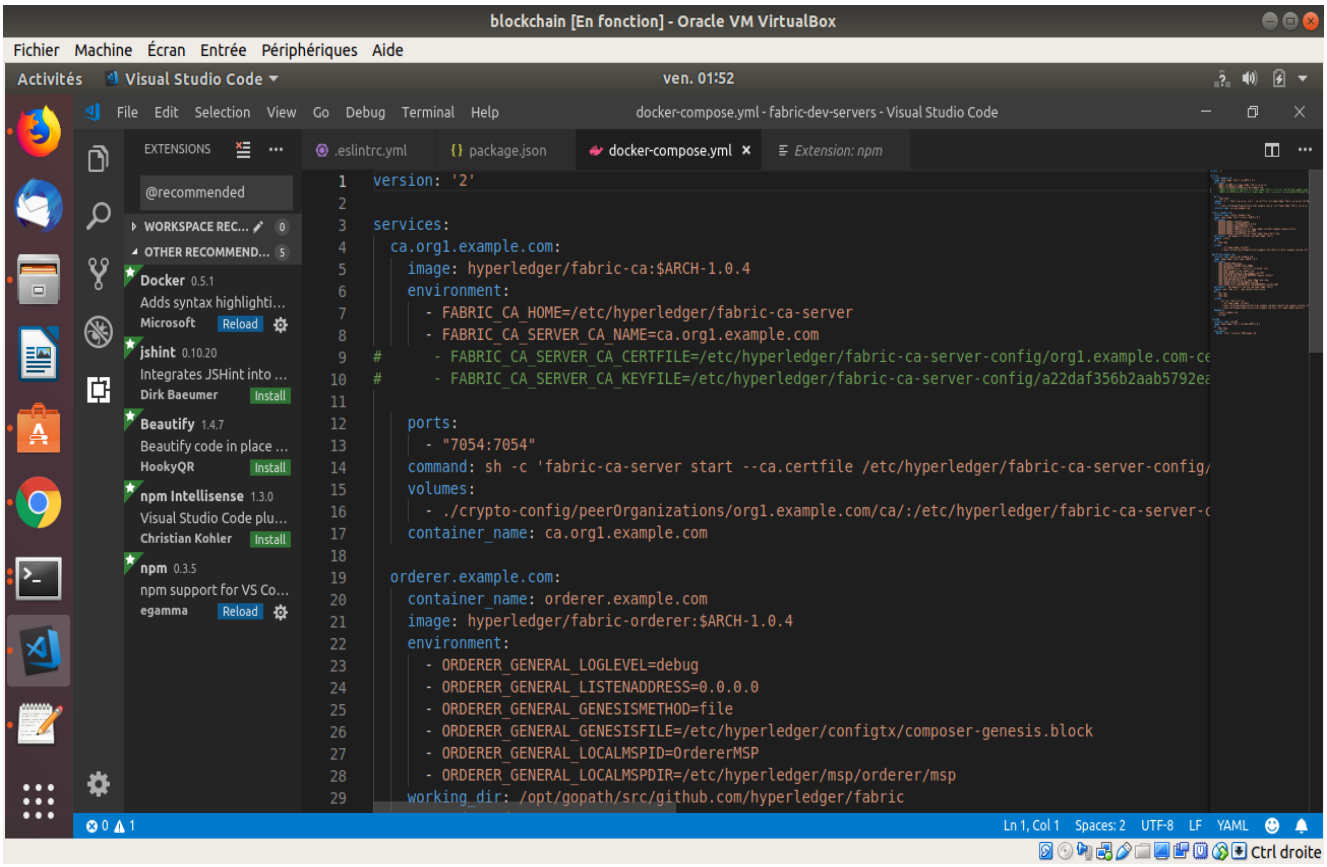

Figure 17: configuration du fichier docker-composer.yml

3.2. Déploiement du « Chaincode »

Le chaincode est l'ensemble de nos smart contracts développés avec Hyperledger composer. Le déploiement va consister à une installation et un démarrage du chaincode dans les nœuds du conteneur Docker configurés précédemment. Pour ce déploiement, nous avons fait usage *des CLI de composer.*

Étape 1 : Installation du réseau dans les nœuds de Hyperledger Fabric

Dans cette étape, nous avons installé notre réseau Blockchain (foodtrace.bna) sur tous les noeuds de votre Hyperledger Fabric. Pour se faire, nous avons eu à exécuter la commande *composer network install* pour installer l'environnement d'exécution Hyperledger Composer sur les noeuds Hyperledger Fabric que nous avons spécifié dans le fichier docker-compose.yml.

```
composer network install --card PeerAdmin@hlfv1 --archiveFile foodtrace.bna
```

Description de la commande :

PeerAdmin@hlv1 : le nom de la carte installée dans Fabric lors de son installation. C'est la carte servant à l'installation de tous les réseaux Blockchain en local.

foodtrace.bna : Une copie archivée de notre réseau développé avec Composer.

Étape 2 : Démarrage du réseau Blockchain0

Dans cette étape, nous avons démarré le réseau Blockchain installé précédemment. Pour se faire, nous avons exécuté la commande composer network start pour démarrer le réseau Blockchain.

```
composer network start --networkName foodtrace --networkVersion 0.0.2-deploy.114 --networkAdmin admin --networkAdminEnrollSecret adminpw --file admin.card
```

admin : l'identifiant de l'administrateur du réseau.

adminpw : le mot de passe de l'administrateur.

admin.card : Cette carte jouera le rôle d'un compte abstrait permettant à l'administrateur d'avoir une vue sur le fonctionnement du réseau. Cette carte n'offre que des droits d'action sur le système et non sur le processus de traçabilité alimentaire mis en place.

Étape 3 : Test de la connexion au réseau Blockchain

Après le démarrage du réseau, il faut donc vérifier son fonctionnement. Pour cette vérification, nous exécutons la commande composer network ping pour tester la connexion au réseau Blockchain:

```
composer network ping --card adminfoodtrace
```

Description de la commande :

adminfoodtrace: le nom de la carte de notre réseau de traçabilité.

Étape 4 : Génération du REST API

Nous exécutons cette dernière étape en prélude à la réalisation de notre front-end. Cette étape va nous permettre de générer l'API par laquelle notre application va interagir avec notre réseau déjà installé dans Hyperledger Fabric. La commande *composer-rest-server* (basé sur la technologie LoopBack) va générer automatiquement une API REST pour notre application.

```
composer-rest-server -c adminfoodtrace -n never -w true
```

4. Développement du front-end

Nous avons, dans cette partie, développé une application avec Angular 2 pour s'interfacer aisément avec notre solution déployée sur Hyperledger Fabric.

Génération des fichiers avec Yeoman

Afin d'accélérer le développement de notre application, nous avons opté pour la solution de génération automatique des principaux composants d'une application Angular 2 qu'offre Yeoman. Commande d'installation de Yeoman

```
npm install -g yo
```

Cette solution est intéressante car elle nous permet de nous passer des fastidieuses tâches de connexion et de développement de services.

Après la génération, nous avons les fichiers de notre application Angular 2 dans le répertoire courant. Nous pouvons ainsi effectuer une personnalisation de l'application selon les besoins de notre projet. Dans notre cas, nous avons dû ajouter des modules et activer des transactions qui ne fonctionnaient pas. Nous vous expliquons cette dernière opération dans le point suivant.

5. Résultats obtenus

À ce niveau, nous allons présenter le résultat obtenu à la suite de nos différentes manipulations. Nous montrerons ainsi le démarrage des nœuds dans Fabric, le lancement de l'application après déploiement, après quoi nous passerons à quelques tests de sécurité.

5.1. Démarrage de Fabric

Pour démarrer les nœuds dans Fabric, il faut, avec le terminal, naviguer dans le répertoire d'installation de Fabric et exécuter la commande *./startFabric.sh*.

5.2. Lancement de l'application

Comme précédemment, il faut naviguer dans le dossier où se trouve notre application Angular 2 puis exécuter la commande suivante :

```
npm start
```

L'application se lance et est accessible par défaut au port *4200* en local.

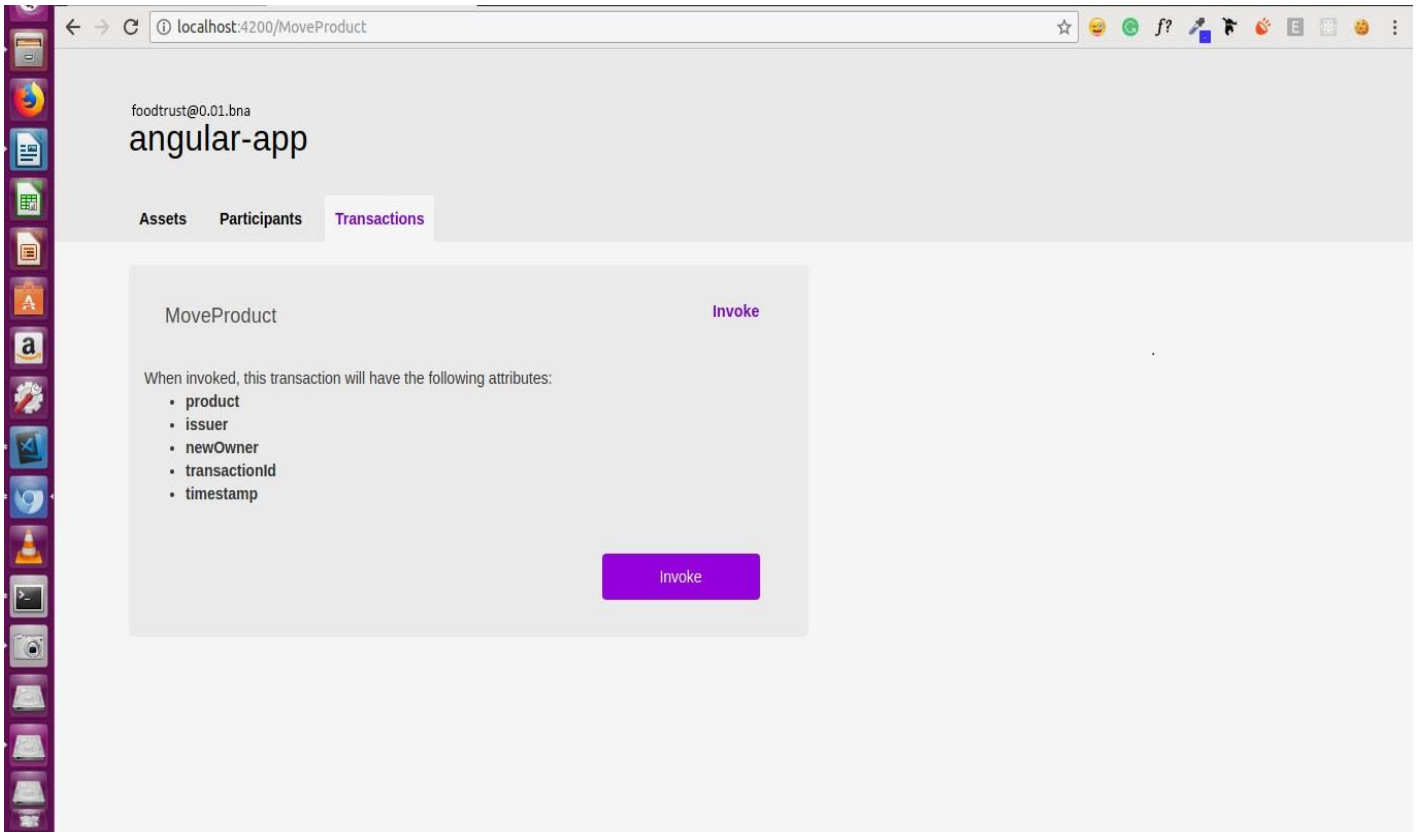

Figure 18: Interface de la solution avec Yeoman

II. Discussion

1. Présentation de la solution obtenue

A la fin des trois mois de travail qui nous a été attribué pour réaliser notre projet interne qui traite de la traçabilité des produits alimentaires dans les supermarchés ivoiriens, nous avons pu réaliser une application web dont le back-end est une blockchain hyperledger composer et le front-end est une application Angular 2 Yeoman qui nous permet d'exploiter notre réseau décentralisé. L'application web permet :

- d'ajouter des participants
- d'ajouter des produit
- d'effectuer des transactions d'échange de produit entre les différents membre du réseau allant du producteur au distributeur.
- de consulter les différentes transaction et block de la blockchain

2. Limite de la solution obtenue

A la fin de notre projet on devrait avoir en plus des fonctionnalités présenté ci-dessus une application mobile qui grâce à un code QR permettrait à un client quelconque d'avoir l'accès à

l'historique de la chaîne d'approvisionnement d'un produit et des informations concernant les différents acteurs de la chaîne, du producteur jusqu'au distributeur.

Au vu des résultats attendus et de ce qui a été réalisé, nous pouvons dit que l'application n'a pas été réalisé entièrement et que notre objectif est atteint partiellement.

3. Evaluation financière

OUTILS	DESCRIPTION	EFFECTIF	MONTANT(Fcfa)
Ordinateur portable(HP)	• Intel Core i3 • 500 Go/ 8Go Ram	1	250000
Téléphone mobile (smartphone)	• Android 7.0	1	70000
Connexion wifi (installation /configuration)	• Flybox Orange	3 mois de connexion	90000
Total			410000

Tableau 14: Evaluation financiere

CONCLUSION

L’étude réalisée a consisté à mettre en place une application mobile basée sur une blockchain pour la traçabilité des produits locaux dans les supermarchés ivoiriens. A travers la méthode UP (unified process) utilisant l’approche orientée objet, nous avons pu réaliser une étude efficace sur notre système afin de réaliser efficacement la traçabilité des produits locaux. Cette étude permettra aux consommateurs ivoiriens d’avoir plus confiance aux produits locaux consommés, en connaissant mieux leur provenance et les conditions de productions et de conservations de ces produits.

Le travail n’étant pas entièrement achevé nous avons mis en place une interface web. L’étude pour accéder à cette interface à partir d’un mobile est en cours de réalisation tant celle des acteurs de la chaine d’approvisionnement que pour le consommateur.

BIBLIOGRAPHIE

[1] LARENCE (Tiana), *La Blockchain pour les nuls*, First Interactive, « Pour les Nuls », 2018, 400 p.

[2] Rapport d'information sur la qualité et la traçabilité des denrées alimentaires

[3] BUFFET Guillaume, *Comprendre la blockchain*, [document électronique], Uchange.co, Janvier 2016, https://www.uchange.co/comprendre-la-blockchain/

[4] Pwc France, « La révolution Blockchain », [document électronique] Juin 2017, https://transformation-digitale.pwc.fr/sites/default/files/decryptage_3_mai_2017.pdf

[5] Qu'y a-t-il dans un « block » de la blockchain [en ligne], http://youcoin.ch/questions-reponses-faq/quy-a-t-il-dans-un-bloc-de-la-blockchain/ [page consultée le 20/12/2018]

[6] ASPROM, blockchain, [document électronique], 2017, http://www.asprom.com/application/blockchain_1.pdf,

[7] **Les 'Smart Contract' dans la blockchain : quézako ? [en ligne]**

http://www.usine-digitale.fr/article/les-smart-contract-dans-la-blockchain-quezako.N494204 [page consultée le 20/12/2018]

[8] Hyperledger, Hyperledger Framework, [en ligne], https://www.hyperledger.org/projects, [page consultée le 20/12/2018]

[9] Blockpartner, Agriculture, agroalimentaire et blockchain, [document électronique], https://blockchainpartner.fr/wp-content/uploads/2017/05/Etude-Agriculture-Agroalimentaire-Blockchain-Partner.pdf

[10] IBM, IBM Food Trust: adding trust and transparency to our food [en ligne]

https://www.ibm.com/blockchain/solutions/food-trust [page consultée le 14/12/2018]

[11] Unfied Process, [en ligne] https://sabricole.developpez.com/uml/tutoriel/unifiedProcess [page consultée le 17/12/2018]

[12] DI GALLO Frédéric, Méthodologie des systèmes d'Information- UML, [document électronique], http://fdigallo.online.fr/cours/uml.pdf

[13] https://www.tutorialspoint.com/javascript/javascript_overview.htm, consulté le 05/01/2019 à 18 : 25

[14] Jani Tarvainen, JavaScript – Overview, [en ligne], https://symfony.fi/entry/what-is-typescript-and-why-should-i-care, , consulté le 05/01/2019 à 19 : 08

[15] Angular, What is Angular?, [en ligne], https://angular.io/docs/ consulté le 05/01/2019 à 14 : 37

[16]Docker, [en ligne], http://putaindecode.io/fr/articles/docker/, consulté le 05/01/2019 à 20 : 57

[17] Docker Compose, [en ligne], https://docs.docker.com/compose/, Consulté le 05/01/2019 à 22 : 06

TABLE DES MATIERES