UNIVERSITE ASSANE SECK DE ZIGUINCHOR

UFR des Sciences et Technologies
Département d'Informatique

Mémoire de fin d'étude

Pour l'obtention du diplôme de
Master en Informatique mention **Génie Logiciel**
spécialité Conduite de projet

Sujet: La Blockchain pour la Sécurisation des E-livrets scolaires

Réalisé par :

Mme Ana BAKHOUM

Sous la direction de :

Dr Ibrahima DIOP

Dr Youssou FAYE

Avec la supervision de :

Pr Salomon SAMBOU

Mémoire soutenu le 02 décembre 2019 devant le jury composé de :

Prénom et nom	**Qualité**
Pr Salomon SAMBOU	Président de jury
M. Gorgoumak SAMBE	Examinateur
Dr Papa Alioune CISSE	Rapporteur
Dr Ibrahima DIOP	Encadreur
Dr Youssou FAYE	Encadreur

Année universitaire : 2018-2019

Résumé

Le livret scolaire est un document administratif au format papier permettant de répertorier les notes des élèves dans les cycles moyen et secondaire (de la classe de sixième à la classe de terminal). Comme dans tout document physique partagé entre plusieurs acteurs, ces derniers rencontrent d'énormes difficultés dans la manipulation, le remplissage, le stockage et la sécurité des livrets scolaires. En 2017, l'entreprise YAKAARTIC, dans le cadre de ses activités, a proposé la dématérialisation du livret scolaire. Cette dématérialisation a constitué à la mise en place d'un système de recueil et de stockage des informations qui étaient dans le livret en papier. On parle ainsi de livrets électroniques (E-livrets).

Pour offrir à ce système le niveau de sécurité qui est requis par les livrets scolaires électroniques nous proposons de mettre en place une application décentralisée pour la sauvegarde des E-livrets. Cette application permet de renforcer la sécurité du système de gestion des livrets électroniques (SGLE) en s'appuyant sur la technologie Blockchain particulièrement sur Ethereum.

En effet, la Blockchain est caractérisée par la transparence, la non répudiation, l'immuabilité des données mais aussi son réseau distribué et les mécanismes de sécurité comme la cryptographie et les fonctions de hachages qu'elle incorpore dans son protocole. En ce sens, l'application décentralisée pour la sécurisation des E-livrets Scolaires permet de protéger les livrets électroniques contre les modifications inattendues et sert de vérifications et de validations de livrets électroniques imprimés depuis le système SGLE.

Abstract

The school booklet is a paper-based administrative document that records student grades in the middle and high school grades (from the sixth class to the terminal class). As in any physical document shared between several actors, they face enormous difficulties in the handling, filling, storage and safety of school booklets. In 2017, the company YAKAARTIC, as part of its activities, proposed the dematerialization of the school booklet. This dematerialization constituted the establishment of a system for collecting and storing information that was in the paper school booklet. This is called electronic booklets (E-booklets).

To offer this system the level of security that is required by electronic school booklets we propose to implement a decentralized application for the backup of E-booklets. This application makes it possible to reinforce the security of the electronic booklet management system (EBMS) by relying on Blockchain technology, particularly on Ethereum.

Indeed, Blockchain is characterized by transparency, non-repudiation, data immutability but also its distributed network and security mechanisms like cryptography and hash functions that it incorporates in its protocol. In this sense, the decentralized application for the securing of school E-booklets makes it possible to protect electronic booklets against unexpected changes and serves as verifications and validations of printed electronic booklets from the EBMS system.

Dédicaces

Je dédie ce travail

À mon père Babacar BAKHOUM

Autant de phrases et d'expressions aussi éloquentes que soient-elles ne sauraient exprimer ma gratitude et ma reconnaissance. Tu as su m'inculquer le sens de la responsabilité, de l'optimisme et de la confiance en soi face aux difficultés de la vie. Tes conseils ont toujours guidé mes pas vers la réussite. Ta patience sans fin, ta compréhension et ton encouragement sont pour moi le soutien indispensable que tu as toujours su m'apporter. Je te dois ce que je suis aujourd'hui et ce que je serai demain et je ferai toujours de mon mieux pour rester ta fierté et ne jamais te décevoir. Que Dieu le tout puissant te préserve, t'accorde santé, bonheur, quiétude de l'esprit, longévité et te protège de tout mal.

À la mémoire de ma grand-mère maternelle Maïmouna LEYE,

À la mémoire de ma mère Fatou SARR

Elles seront toujours présentes dans mon esprit et occuperont toujours une place importante cœur. J'espère tout simplement qu'elles seront fières de moi où qu'elles puissent être. Que Dieu, le miséricordieux, les accueille dans son éternel paradis.

Remerciements

La réalisation de ce mémoire a été possible grâce au concours de plusieurs personnes à qui je voudrais témoigner toute ma gratitude.

Je tiens à exprimer toute ma reconnaissance à **mes encadreurs** Docteur Ibrahima DIOP et Docteur Youssou FAYE pour la patience, la disponibilité et surtout les judicieux conseils et orientation dont ils ont fait preuve tout au long de ce mémoire.

Ma gratitude et ma reconnaissance vont à l'endroit de mon **superviseur**, le Professeur Salomon SAMBOU pour toute l'importance qu'il a accordé à ce travail. Mes remerciements vont aussi à l'endroit du Docteur Papa Alioune CISSE et Monsieur Gorgoumack SAMBE pour avoir accepté de faire partie du **jury d'évaluation** de ce mémoire : Vos critiques, suggestions et conseils m'ont été d'une grande importance.

Je remercie également tous **les enseignants-chercheurs** de l'Université de Assane Seck de Ziguinchor qui ont participé à ma formation et ayant assuré la partie théorique de celle-ci. Vous m'avez fourni les outils nécessaires à la réussite de mes études universitaires. L'enseignement de qualité dispensé a également su nourrir mes réflexions et a représenté une profonde satisfaction intellectuelle, merci donc à tous.

Toute ma gratitude va à l'endroit de **mes collaborateurs**, amis et camarades à YAKAARTIC. Mention spécial à notre Directeur Général Monsieur Sidiya DIENG, son sens de responsabilité et ses qualités de dirigeant sont pour moi une source de motivation. Merci pour les critiques, suggestions et encouragements. À M. Modou DIOP, M. Rodrigue Eloi MANGA sans oublier M. Aliou DIEME et M. Ibrahiama DIAVE, c'est un plaisir pour moi de travailler avec vous.

Je remercie mon **très cher père** qui a toujours été là pour moi. Merci pour votre éducation, vos soutien et encouragements. À **mes sœurs** Mariama et Mbéré, mes cousin(e)s spécialement Ousmane FAYE, mes tantes, et à tous les **membres de ma famille** au sens large, je les dis merci pour tout ce qu'ils ont fait pour moi. Vous m'avez toujours soutenu dans tous les moments de la vie. Vos inconditionnels encouragements m'ont été d'une grande aide.

À mon **époux** Ousmane DIANGAR, qui me comble d'affection et qui m'a soutenu et encouragé d'une manière incommensurable. Merci d'être là pour moi.

A **la famille FALL**, ma famille d'accueil depuis mon entrée à l'université, Alousseynou FALL, son épouse Siga DIOUF et leurs enfants je vous témoigne toute ma gratitude pour votre amour,

votre affection, votre hospitalité, votre soutien et vos encouragements. Vous avez fait que je n'ai jamais eu ce sentiment d'être loin de mes parents.

Je tiens à témoigner toute ma reconnaissance à **mes camarades du département informatique** et spécialement de la promotion de 2016. Nos moments de partages et de solidarité resteront gravés dans ma mémoire.

À tous ceux qui ont, de près ou de loin, participé à la réussite de ce mémoire; Je vous dis merci.

SOMMAIRE

LISTE DES FIGURES

LISTE DES TABLEAUX

ACRONYMES ET ABRÉVIATIONS

ANSD : Agence Nationale de la Statistique et de la Démographie

API : application programming interface

BFEM : Brevet de Fin d'Etude Moyen

BTC : La monnaie Bitcoin

CFEE : Certificat de Fin d'Etude Elémentaire

DApp : Decentralized Applications

ECC : Elliptic Curve Cryptography

ECDSA : Elliptic Curve Digital Signature Algorithm

EOA Externally Owned Accounts

ETH : Une abréviation d'Ether qui est la monnaie électronique de la blockchain Ethereum

EVM : Ethereum Virtual Machine

IDE : Integrated Development Environment

IPFS : InterPlanetary File System

JSON : JavaScript Object Notation

P2P : « Peer-to-Peer »

PoW : Proof of Work

RPC : Remote Procedure Call

RSA : Rivest Shamir Adleman

SDK : Software development kit

SDSEL : Système Décentralisé pour la Sécurisation des E-Livrets Scolaires

SGLE : Système de Gestion des Livrets Electroniques

UTXO : Unspent Transaction Output (sorties de transaction non dépensées)

INTRODUCTION GÉNÉRALE

Au Sénégal, l'enseignement moyen et secondaire ou l'éducation en général est régi par un ensemble de règles et de normes. Étant sous la tutelle du ministère de l'éducation nationale, les établissements publics et privés disposent d'un système de partage d'informations sur les parcours des élèves, basé sur des livrets scolaires. Un livret scolaire est un document administratif obligatoire pour tout élève du cycle moyen et secondaire. Sous format papier, il permet de retracer le parcours ainsi que le comportement des élèves de la sixième à la terminale.

Cependant le livret scolaire est frappé par plusieurs problèmes liés à son caractère papier. Parmi ces problèmes on peut citer :

- ✓ la lourdeur dans le remplissage ;
- ✓ le délabrement de lieu de stockage ;
- ✓ la difficulté de transfert des livrets ;
- ✓ la recherche fastidieuse ;
- ✓ la difficulté d'établir des statistiques ;
- ✓ etc.

A ces problèmes s'ajoutent la non-intégrité des informations saisies dans les livrets, la falsification ou la recréation de livrets et les pertes d'informations.

Pour pallier à ces problèmes, l'entreprise YAKAARTIC[1] a mis en place un système informatique permettant la dématérialisation du livret scolaire papier. Pour ce faire, elle a réalisé un système de gestion des livrets électroniques (SGLE), permettant d'alléger la manipulation et le partage des informations sur les élèves mieux que le système basé sur les livrets scolaires en papier. Le SGLE permet aux collèges et lycées, publics comme privés de gérer les élèves et leurs notes, d'établir des statistiques en interne mais aussi de faciliter le transfert des livrets scolaires d'un établissement à un autre. Ce système fonctionne sur un serveur et stocke les données des livrets dans une base de données relationnelle MySQL.

Cependant, comme tout système central, le SGLE ne reste pas sans failles liées à la sécurité et à la protection des données. En effet la centralisation des données des livrets dans un seul serveur est un facteur de risque de perte d'informations. Vue le rôle important qu'ils

[1] YAKAARTIC est une entreprise crée en 2017 par de jeunes étudiants (dont nous faisons partie) de formation génie logiciel et informatique de gestion. Sa création est motivée par la victoire (premier prix) de l'équipe de ses membres fondateurs au CHALLENGE GAINDE STARTUP ENTREPRENDRE, organisé par l'entreprise GAINDE 2000. YAKAATIC évolue dans le domaine informatique en offrant des services de développement d'applications sur mesure. Pour plus d'information, consulter son site web www.yakaartic.com

peuvent jouer dans le cursus des élèves, les livrets électroniques nécessitent un niveau de sécurité élevé. Ainsi, il est opportun de mettre en place une application décentralisée communiquant avec le SGLE et fonctionnant sur la Blockchain. Ce qui rentre dans le cadre de ce présent mémoire.

En effet, la Blockchain (« chaîne de blocs », en Français) est une technologie émergente, qui permet de stocker des données numériques, de manière décentralisée, chronologique, immuable et sécurisée. Il s'agit d'une sorte de livre de compte ou de registre qui contient la liste de tous les échanges effectués entre les utilisateurs du réseau. Elle a été initialement conçue, avec Bitcoin, pour des transactions monétaires jusqu'à l'ajout d'un cadre d'exécution des contrats intelligents qui donna naissance à la Blockchain Ethereum. Cette dernière permet le développement d'applications décentralisées avec des fonctionnalités outre que le transfert de crypto-monnaie. En ce sens, nous allons mettre en place un système décentralisé de sécurisation des E-livrets scolaires (nommé SDSEL) qui aura pour vocation de sauvegarder et de sécuriser les E-livrets scolaires enregistrés par le SGLE, à la fin de l'année. En outre, SDSEL permettra de valider ou vérifier les E-livrets scolaires des élèves.

Ainsi, pour une bonne organisation du document, nous l'avons scindé en quatre chapitres. Le premier chapitre concerne la contextualisation du sujet. Il nous permet de présenter en premier lieu le livret scolaire papier et ses problèmes. Ensuite nous allons montrer le SGLE avec la problématique qui tourne autour de celui-ci. Enfin nous définirons les objectifs généraux et spécifiques de ce mémoire.

Au deuxième chapitre, nous ferons un état de l'art sur la Blockchain afin de mieux appréhender cette technologie et de comprendre la sécurité qu'elle implémente. À cet effet, nous expliquerons les notions fondamentales à la Blockchain, l'historique et la terminologie de cette technologie. En outre, nous détaillerons les techniques de sécurité mises en évidence dans les protocoles des Blockchain.

Le troisième chapitre est réservé à la Blockchain Ethereum dans lequel nous montrerons les particularités d'Ethereum par rapport à Bitcoin qui est la Blockchain mère. Ce chapitre mettra en exergue les contrats intelligents, qu'on considère comme la pièce maitresse de la poussée d'Ethereum. Nous parlerons également des applications décentralisées et leurs fonctionnements. Par ailleurs, dans ce chapitre, nous montrerons comment Ethereum nous permet d'accéder à nos objectifs.

Et en fin, vient le chapitre quatre que nous consacrons à la conception, à l'implémentation et à la présentation de notre solution SDSEL. Dans la première partie, nous présenterons les éléments de conception et d'implémentation de l'application notamment les contrats

intelligents et leurs interactions mais aussi les outils de développement utilisés. Dans la deuxième partie, nous présenterons quelques captures des interfaces de l'application.

CHAPITRE I : E-LIVRET SCOLAIRE

Au Sénégal, la fin des études primaires est sanctionnée par la réussite au Certificat de Fin d'Études Élémentaires (CFEE). Tout élève ayant réussi cet examen peut s'inscrire en classe de Sixième dans un collège public ou privé du pays. L'entrée au collège nécessite, entre autre, l'achat d'un livret scolaire communément appelé « **livret de bac** » qui joue un rôle important dans le cursus des élèves. En effet, le livret scolaire est un papier (sous forme de livre) contenant le parcours d'un élève du moyen au secondaire et aide à la prise de décision (pour le repêchage) lors des examens de baccalauréat et de BFEM. Il est au centre d'un nombre important d'acteurs tels que, les surveillants, les enseignants, les présidents de jury d'examen etc. Cependant, ces acteurs rencontrent d'énormes difficultés dans le remplissage, la recherche ou les déplacements des livrets vers les centres d'examen. Ainsi, YAKAARTIC, une entreprise s'activant dans le développement de solutions informatiques dont nous faisons partie des membres fondateurs, a mis en place une solution de dématérialisation des livrets scolaires pour pallier à ces problèmes. Cette solution offre des fonctionnalités sur la gestion des livrets des élèves pour les établissements mais ne garantit pas toujours l'authenticité des notes saisies dans le livret.

Dans ce chapitre, nous présentons d'abord le livret scolaire dans sa version papier en montrant ses acteurs ainsi que les problèmes qu'ils rencontrent dans l'utilisation. Ensuite nous présentons le système de gestion du livret électronique (SGLE) avec ses fonctionnalités et ses limites. Puis, nous montrons les problématiques que ce mémoire tente de résoudre. Ce qui nous permettra de définir les objectifs du sujet.

1.1. Le livret scolaire papier

Une présentation générale du livret scolaire permet de comprendre l'importance accordée à ce document dans le domaine de l'éducation au Sénégal notamment dans l'enseignement moyen et secondaire. Pour cela nous avons identifié les acteurs qui interviennent dans la manipulation des livrets et les problèmes qu'ils rencontrent.

1.1.1. Présentation Générale du livret scolaire

Homogène et normé par le décret N° 78/691 du 12 Juillet 1978, le livret scolaire en papier est le seul outil utilisé au Sénégal pour rendre compte de l'évolution d'un élève durant tout son parcours aux cycles moyen et secondaire. Il est constitué d'une quinzaine de pages et renseigne, année par année, sur la scolarité de l'élève. Chaque page contient les informations sur l'établissement fréquenté durant l'année scolaire, le niveau d'étude, le développement physique, les notes et les appréciations obtenues dans chaque matière, l'avis du professeur principal, l'avis des conseils de discipline sur son comportement après chaque tenu de celui-ci,

le visa et l'observation du chef d'établissement. Pour les classes d'examen (troisième et terminale), la décision du jury s'ajoute à ces informations. Sur la page de garde du livret sont écrites les informations d'identification de l'élève tel que son nom, son prénom, sa date et son lieu de naissance, son adresse de résidence complète. Toutefois, les informations d'identité sont rappelées dans chaque nouvelle page avec les informations sur le parent ou le tuteur de l'élève. En effet l'adresse et les tuteurs peuvent être changés d'une page à une autre.

L'utilisation du livret scolaire est obligatoire et tout élève s'inscrivant en classe de sixième, pour la première fois, dans une école privée ou publique est tenu d'acheter un livret scolaire qui sera rendu à l'administration de l'école avec un extrait de naissance. Son remplissage est effectué au fur des années, après chaque nouvelle inscription pour une année académique, après les passages des compositions ou examens et après les conseils de discipline de l'établissement. Le livret scolaire permet d'avoir un suivi continuel sur l'état d'évolution de l'élève au fur des années. Ainsi il donne une appréciation sur l'évolution, d'une part, de son comportement social vis-à-vis de son environnement éducatif et d'autre part de son état psychologique de par ses résultats. En outre, le livret constitue, pour les examens, un des éléments d'appréciation dont dispose le jury lors des délibérations. Il lui permet de prendre en compte le travail effectué par l'élève dans les classes précédentes (sixième, cinquième, quatrième, troisième pour le BFEM et seconde, première et terminale pour le baccalauréat) pour mieux orienter les repêchages.

1.1.2. Les acteurs du livret scolaire

Le livret scolaire est un point central entre plusieurs acteurs. Depuis son achat, il passe par les mains de plusieurs personnes au sein de l'école où il est déposé. Par ailleurs, il peut passer d'une école à une autre. Dans le tableau qui suit, nous avons répertorié les acteurs qui sont en relation avec le livret scolaire ainsi que leurs descriptions.

Acteurs	Descriptions
L'élève	Il achète le livret et le met à la disposition de l'administration de son école lors de son inscription en classe de sixième dans un établissement d'enseignement moyen privé ou public. Il ne peut pas voir son livret durant tout son cursus scolaire jusqu'à l'obtention du baccalauréat.
L'enseignant	Il renseigne les notes de l'élève ainsi que les appréciations sur les résultats obtenus lors des évaluations dans les matières qu'il

	dispense. Il peut aussi léguer ce travail aux surveillants pour qu'ils le fassent à sa place.
Le surveillant	Il est chargé de remplir les informations personnelles de l'élève (nom, prénom, photo après chaque année, changement d'adresse de tuteur etc.) lors de l'inscription de l'élève. En outre, avec l'aval de l'enseignant, peut renseigner les notes des élèves à l'issu des évaluations.
Le surveillant général ou le censeur	Il notifie les appréciations faites sur l'élève après chaque conseil de discipline (Autoriser à redoubler, passe en classe supérieur, élève sérieux, élève indiscipliné…). Il peut aussi jouer le rôle d'un surveillant simple.
Le chef d'établissement (principal ou proviseur)	Il appose son cachet et signe sur livret de chaque élève de son établissement après la tenue du conseil de classe du semestre. Il doit mentionner son nom et prénom ainsi que la date.
Ministère de l'éducation	Définit le format standard du livret, son mode de fonctionnement et sa politique de confidentialité.
Président de jury Examen (BFEM, BAC)	Consulte le livret pour les repêchages des candidats (voir si l'élève a le mérite d'être repêché ou pas grâce aux notes qu'il a eu dans les classes précédentes). Il renseigne le résultat de l'élève à l'examen (Ajourné, autoriser à passer les épreuves du second groupe…) et met son visa (cachet et signature).

Tableau 1: les acteurs qui interagissent avec livret scolaire papier

Avec ce nombre d'acteur assez important et des rôles divers, il y a lieu de noter que des problèmes sont souvent rencontrés par certains de ces acteurs.

1.1.3. Les problèmes liés au livret scolaire

La version papier du livret scolaire engendre un certain nombre de problèmes que ce soit dans le stockage, le remplissage ou les déplacements des livrets. Ces problèmes peuvent affecter l'existence du livret mais aussi sa manipulation et son utilisation. Ils sont décrits dans le tableau qui suit ainsi que leurs conséquences.

Problèmes	Descriptions	Impact sur les livrets
Insécurité dans les lieux de stockage	Les livrets scolaires sont gardés dans les écoles. Ce qui entraine des risques d'insécurité sur ces derniers. Ceci est souvent dû aux inondations pouvant frapper les écoles, aux cas rares d'incendies. En outre, des personnes non habilitées peuvent accéder aux lieux de stockage. Par ailleurs, les cas de vols, le délabrement des lieux de stockage et l'ordonnancement laborieux ne sont pas épargnés.	+ Pertes de livrets + Fraudes + falsifications
La recherche difficile	Rechercher le livret d'un élève est hyper fastidieux surtout pour un élève qui a quitté l'école depuis des années ou pour les écoles avec un grand effectif.	+Perte de temps
Difficulté dans le remplissage	Pour remplir le livret l'enseignant ou le surveillant doit rester pendant des heures à l'école en dehors de ces heures de travail sinon il devra user des heures de travail pour le remplissage des livrets. D'autre part deux enseignants ne peuvent remplir les notes d'un même élève simultanément. En plus rien interdit la modification de note (par rature). Par exemple, un acteur peut facilement changer note d'un élève même s'il ne s'agit de sa matière.	+ Perte de temps dans la saisie des livrets + pas de validations des contenus des livrets
Voyage matériel des livrets	À chaque transfert d'un élève, son établissement d'origine est obligé de transmettre le livret vers son établissement de destination après une demande notifiée par cette dernière. Ce qui amène très souvent des problèmes de commutation des livrets.	+ Perte de livrets + Livrets incomplets + Fraudes ou recréation de livret

Manque de statistiques	Une grande difficulté d'établir des statistiques à partir des livrets en papier. Par exemple, comment trouver le nombre de fille ayant la moyenne supérieure à 15 en mathématiques.	+ Pas de statistiques sur l'enseignement
Non Authenticité des données remplies	Aucun dispositif ne permet de vérifier si le livret a été bien rempli lors d'une opération d'écriture par l'un des acteurs habilités (exemple : lors du report des notes d'un élève, un surveillant se trompe et met 05 à la place de 15).	+ non intégrité des notes saisies + non fiabilité des livrets
Non Authentification des acteurs	Pas de contrôle sécurisé sur les acteurs surveillant et enseignant qui ont des droits de modification sur le livret. En lisant le livret aucune information ne permet d'affirmer que c'est tel utilisateur qui a effectué l'action.	+ accès libre aux données + modification des informations du livret

Tableau 2: liste des problèmes liés à la version papiers du livret

L'ensemble des problèmes qui frappent l'existence du livret entraine le plus souvent des pertes de livrets et des cas de fraudes et de falsifications de notes. Or le caractère tangible du livret ne permet pas, en cas de perte, de retrouver les informations que le livret contenait. Ce qui amène à racheter un nouveau et à le remplir avec des informations inexactes.

A ces problèmes s'ajoute la croissance du nombre d'apprenants au fur des années. En effet, en 2013, dans son rapport sur la situation économique et sociale du Sénégal [1], l'ANSD (l'agence Nationale de la statistique et la démographie) montre que 74 215 candidats sont admis au CFEE sur 219 020 candidats aux épreuves soit un taux d'admission de 33,9%. Par ailleurs, l'accès à l'enseignement moyen est mesuré à travers le taux de transition du CM2 (dernier niveau du cycle primaire) à la sixième (première classe du cycle moyen). En 2014, le taux de transition est estimé à 87,0% d'après les statistiques de l'ANSD de la dite année [2]. Tout ceci montre la tendance évolutive de l'effectif des élèves du moyen et du secondaire au fur des années. Cette augmentation est fonction de l'accroissement de la population, de l'augmentation du taux d'alphabétisation et du renforcement du nombre d'établissements. A ce nombre pléthorique s'ajoute la durée de vie d'un livret. Un livret est stocké et manipulé pendant sept ans au minimum. Donc le problème de l'archivage des livrets des élèves sortant du cycle secondaire (ayant le Bac) se pose à cet effet.

On note que le caractère tangible du livret engendre d'énormes difficultés dans la manipulation et le stockage. Avec l'ère du numérique où n'importe quel processus peut être

simplifié avec la dématérialisation, nous ne pouvons pas rester sans rien faire face ces problèmes. En ce sens que l'idée de mettre en place une application permettant de rendre le livret intangible en le dématérialisant est émise. Cette application permet d'automatiser la gestion des livrets scolaires au Sénégal.

1.2. Dématérialisation des livrets scolaires dans les collèges et lycées

Le système de gestion des livrets électroniques (SGLE) a pour vocation de faire face aux problèmes rencontrés par les acteurs du livret scolaire et de rendre, ainsi, les livrets scolaires des élèves intangibles. Son objectif est de fluidifier la manipulation des livrets d'une part et d'autre part, d'accélérer les recherches de statistiques (les nombres d'élèves, suivant les moyennes, les localités, le genre etc.) et faciliter l'archivage des livrets qui étaient tellement pénibles voire même impossibles avec les livrets papiers. Tout comme le livret en version papier, le livret électronique renferme les informations sur l'identité de l'élève, ses notes dans toutes matières, son comportement (appréciations des professeurs ; les décisions à l'issu des conseils de classes et de discipline) ; pour chaque année de la classe de sixième à la classe de terminal. Le livret électronique est consultable lors des examens par les présidents de jury.

1.2.1. Fonctionnalités du SGLE

Le SGLE permet aux établissements de générer des livrets électroniques (E-Livret) qu'on considère comme un ensemble d'informations (identité, notes, décisions de conseil …) relatives à un élève et accessible en ligne avec les privilèges requis. À cet effet le système de gestion des livrets électroniques assure les fonctionnalités suivantes :

Fonctionnalité	Description
L'enregistrement des établissements :	Chaque établissement (public ou privé) est stocké dans la base de données ainsi que son personnel enseignant et administratif. Un établissement a plusieurs niveaux et chaque niveau a plusieurs classes.
La saisie des élèves :	Les nouveaux élèves qui viennent d'entrer au collège sont stockés dans la base de données. Leurs informations personnelles ne sont pas modifiées tandis que d'autre comme l'adresse ou la photo de profil peuvent être modifiées à chaque nouvelle inscription.

L'inscription des élèves	À chaque début d'année, les élèves s'inscrivent dans une classe ou se réinscrivent dans la même classe pour les redoublants.
La saisie des notes semestrielles des élèves	Les enseignants renseignent les notes des élèves de leurs classes à chaque évaluation. Les moyennes semestrielles sont calculées automatiquement et enregistrées dans les livrets de ces élèves.
La saisie des avis des conseils de classe :	À la fin du semestre, les enseignants se réunissent en conseil de classe pour évaluer les comportements des élèves (absences, indiscipline,…) durant le semestre. Les avis du conseil vis-à-vis des élèves, qui représentent une partie du livret, sont enregistrés dans le système.
La saisie des appréciations du professeur principal	Chaque classe est affectée à un professeur principal, c'est le représentant de la classe dans les instances de décisions (conseil de classe par exemple) au sein de l'établissement. À la fin de l'année, le professeur principal donne une appréciation global sur la dite classe. Cette appréciation est enregistrée dans le système.
Consultation (affichage) des livrets	Les élèves, les parents d'élèves, les enseignants et l'administration des établissements notamment les chefs d'établissement peuvent consulter les livrets. Cependant, l'affichage n'est pas la même pour ces profils. Pour l'élève et le parent d'élève, seules les informations sur l'identité de l'élève et les notes obtenues dans les différentes matières sont visibles (autrement dit le bulletin de notes). La consultation du livret est protégée par une authentification forte.
Le transfert de livrets d'un établissement vers un autre	Le transfert d'un élève vers un autre établissement se matérialise dans le système par la révocation du droit de modification du livret par l'établissement quitté et l'attribution de ce droit à l'établissement d'accueil. Le livret ne pourra plus être accessible depuis l'ancien établissement.
Envoi des livrets aux centres d'examens (BAC et BFEM)	Après l'enregistrement des centres et jurys d'examen, chaque président de jury peut consulter les livrets de ses candidats.
La saisie des résultats d'examen	Les résultats des examens sont saisis dans le système par le président de jury.

Recherche de statistique et archivage	L'obtention de statistiques est l'un des problèmes majeurs que l'E-Livret a réglé. On peut rechercher tous les statistiques sur l'enseignement moyen et secondaire au Sénégal dans le système SGLE. À partir des critères de recherche, les utilisateurs peuvent facilement trouver un livret.

Tableau 3: liste des fonctionnalités du SGLE

La figure suivante correspond à une capture d'écran de la saisie des notes de devoir par un enseignant dans le système SGLE.

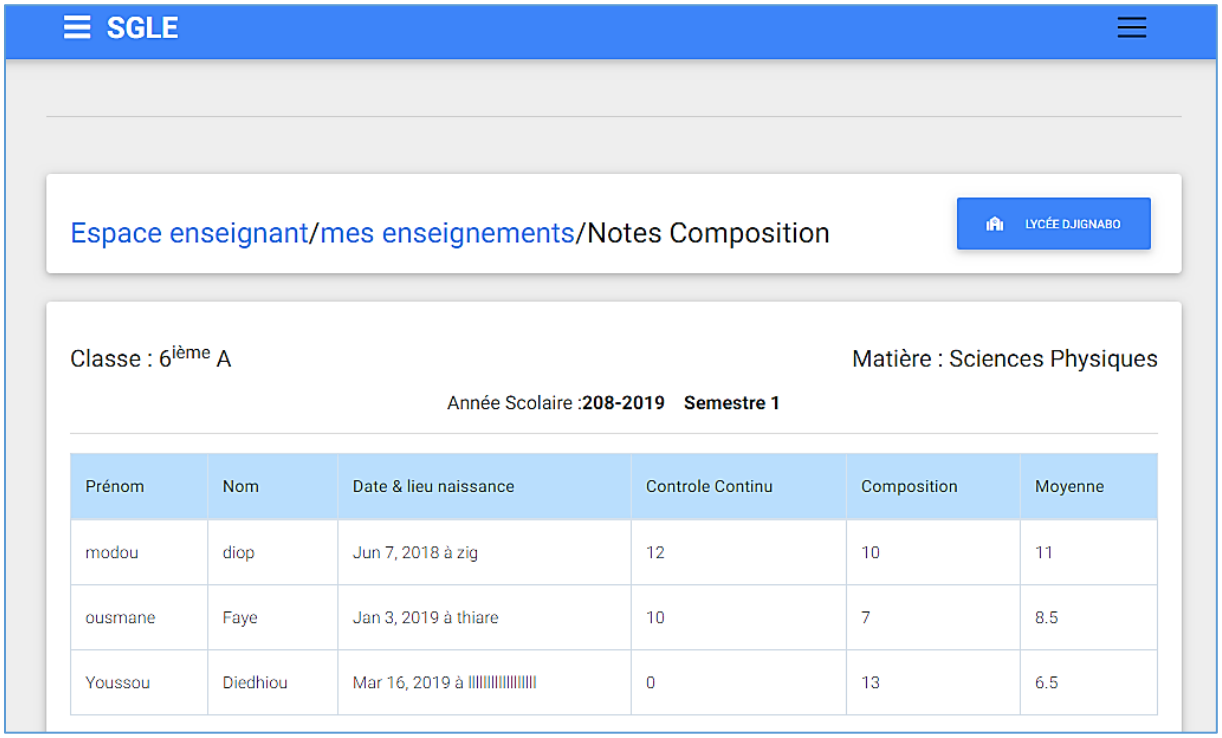


Figure 1: Interface de saisie des notes des devoirs proposée dans E-livret Scolaire

La réalisation de cette application de gestion des livrets électroniques (E-livret) a résolu l'ensemble de problèmes qui frappent le livret papier.

1.2.2. Les problèmes résolus par le SGLE

L'application de gestion des livrets électroniques a sans doute atteint ses objectifs de dématérialisation du livret scolaire papier et l'automatisation dans sa gestion. À cet effet, son utilisation dans les établissements moyens et secondaires permettra de :

- ✓ libérer les espaces de stockage dans les établissements car le livret est stocké dans une base de données. Ce qui règle ainsi le problème d'insécurité des livrets;
- ✓ assurer l'intégrité des notes en offrant la possibilité aux élèves de consulter uniquement leurs notes et de faire leurs revendications en cas d'erreur sur une note saisie par un enseignant ;

- ✓ garantir l'authentification des acteurs en définissant des niveaux d'accès par type d'utilisateur. Ce qui assure le respect des politiques d'accès aux informations contenues dans le livret selon les amendements de l'organisme compétent ;
- ✓ faciliter et accélérer la recherche et la consultation des livrets avec des options de recherches détaillées ;
- ✓ les enseignants peuvent accéder aux livrets et remplir les notes simultanément à partir d'un ordinateur ou d'un téléphone : plus besoin d'attendre qu'un enseignant termine ;
- ✓ élimer les difficultés liées aux voyages des livrets entre les établissements d'enseignements, lors des transferts d'élèves ou des examens de BFEM et BAC ;
- ✓ Disposer de statistiques fiables ;
- ✓ Faciliter la consultation des livrets archivés
- ✓ …

L'application de gestion des livrets électroniques a apporté des solutions aux problèmes qui touchent notre système scolaire basé sur le livret papier. Cependant, elle présente des limites dans la sécurité des livrets. Ainsi, la problématique de ce mémoire est d'analyser ces limites pour y apporter des solutions.

1.3. Problématique du sujet

Le SGLE fonctionne sur une architecture client-serveur à trois niveaux : couche présentation, métier et d'accès aux données, appelée architecture 3-tiers. Les utilisateurs accèdent au système à partir de la couche de présentation. Cette dernière interroge la couche métier pour offrir des réponses aux requêtes des utilisateurs. Ainsi les utilisateurs n'ont pas directement accès au stockage des données, ils passent toujours par le deuxième niveau qui est la couche métier. Même si des mécanismes permettant de protéger les requêtes des utilisateurs sont inclus dans le système de gestion des E-Livrets (livrets électroniques), il y a lieu de noter que la sécurité de la couche de stockage des données est d'une grande importance. En effet, le système est hébergé dans un serveur distant. En d'autres termes, les données (les informations concernant les établissements, les élèves, les notes etc.) de l'application sont stockées dans une base de données hébergée par un serveur. Ainsi, la sécurité de l'application dépend de la sécurité des données qu'elle produit et stocke dans sa base de données. D'où le disfonctionnement de la base de données entraine l'inaccessibilité des E-livrets des élèves.

En outre, l'accès au serveur (par un programme malveillant comme les virus ou par une personne mal intentionnée) peut entrainer une altération ou corruption des données. Ceci est possible car le contenu de base de données est stocké dans le disque du serveur. Or les livrets

scolaires jouent un rôle important dans le cursus de l'élève. Par conséquent une haute protection de ces derniers, surtout contre des modifications est requise. Compte tenu de tout cela, le SGLE devrait être beaucoup plus fiable, robuste et tolérante aux pannes avec la décentralisation et un stockage sécurisé des données.

Afin de répondre aux attentes d'une dématérialisation d'un papier si important que soit le livret scolaire, le SGLE doit garantir un niveau de sécurité assez élevé ainsi que la disponibilité des livrets et leur authenticité. Pour ce faire, une base de données relationnelle seulement ne suffirait pas pour assurer la fiabilité, l'authenticité, la transparence et la sécurité des E-livrets. Comment le système devra-t-il prendre en compte tous ces facteurs ? Quelle architecture sera idéale ? Et quel type de stockage devons mettre en place ? Les réponses à ces questions font l'objet de ce présent sujet de mémoire.

1.4. Objectifs du sujet

Après avoir analysé les risques de perte de données qui frappent le système de gestion des livrets électroniques et mesuré l'impact que cela peut avoir sur les livrets électroniques générés, il est important d'évaluer les solutions à en envisager pour faire face à ces problèmes.

L'objectif général de ce sujet est de mettre en place un système de stockage décentralisé des livrets électroniques générés à partir du SGLE afin de garantir la sécurité des E-livrets. Concrètement nous avons mis en place une application décentralisée sécurisée et robuste en utilisant la Blockchain Ethereum.

Les objectifs spécifiques sont :

- ✓ Mettre en place les contrats intelligents qui permettent d'importer les informations sur les établissements qui utilisent SGLE dans la Blockchain Ethereum.
- ✓ Développer les contrats pour la sauvegarde des données des livrets des élèves de chaque établissement. Pour ce faire, une importation des données des livrets est effectuée à la fin du semestre.
- ✓ Permettre aux établissements d'accéder à l'historique des livrets authentiques et d'effectuer des recherches facilement. En outre, leur donner la possibilité de générer des statistiques fiables sur les élèves
- ✓ Grace aux techniques de sécurité de blockchain Ethereum, tous les livrets importés ne peuvent plus subir de modification. Ainsi on assure l'inviolabilité des livrets afin d'avoir un niveau de sécurité qui rend quasiment impossible les fraudes et altérations des notes des élèves.

- ✓ Les établissements vont pouvoir générer les livrets électroniques de leurs élèves en format PDF et le confronter aux livrets qui sont générés par le SGLE pour valider l'intégrité des informations qui y figurent.

Pour arriver à terme de cette réalisation, il est opportun de connaitre la technologie de la Blockchain en général et le fonctionnement d'Ethereum en particulier. Ainsi le chapitre suivant, fait un état de l'art sur la Blockchain.

CHAPITRE II : ÉTAT DE L'ART SUR LA BLOCKCHAIN

Depuis toujours, une transaction monétaire entre personnes ou sociétés nécessite le contrôle d'un organisme tiers à l'image des banques. Outre le rôle d'intermédiaire qu'elle joue, la banque (ou une société de carte de crédit) reçoit les frais issus de cette transaction. Par ailleurs ce même processus est observé dans plusieurs autres domaines, tels que les jeux, la musique, les logiciels, etc. Ainsi, le système de transaction est généralement centralisé et toutes les données et informations sont contrôlées et gérées par une organisation tierce, plutôt que par les deux entités principales impliquées dans la transaction. C'est en ce sens que la technologie Blockchain a été développée pour résoudre ce problème en créant un environnement décentralisé dans lequel aucun tiers ne contrôle les transactions et les données.

La Blockchain est une sorte de base de données distribuée qui gère une liste croissante d'enregistrements de données confirmés, par consensus, par les nœuds qui y participent. Les enregistrements sont distribuées dans tous les nœuds du réseau ; ce qui rend le système plus transparent. De plus, les nœuds de la blockchain sont tous anonymes, ce qui la rend plus sécurisée. Bitcoin fut la première application à introduire la technologie Blockchain et qui crée un environnement décentralisé pour la crypto-monnaie. Cependant, des défis et limitations techniques ont été résolus plus tard pour donner d'autres types d'applications.

Dans ce chapitre nous mettrons en exergue les notions fondamentales de la technologie Blockain partant de l'historique aux applications dérivées passant par les définitions et les types de blockchain. Ensuite, nous ferons le point sur la structure (architecture) globale des blockchains. En enfin nous montrerons la sécurité implémentée dans le protocole des blockchains.

2.1. Qu'est-ce que la Blockchain ?

2.1.1. Historique

La crypto-monnaie [3] est une monnaie virtuelle qui n'a aucun lien avec une politique monétaire ou une banque. Son implémentation repose sur des algorithmes cryptographiques permettant de générer de la monnaie et de faire des transactions anonymes entre des paires sur internet. L'idée de crypto-monnaie a été émise pour la première fois vers les années 1998 lorsque Wei Dai a publié une description de b-money [4], un système de payement électronique sans interface bancaire. En revanche ce système manquait de détails sur la mise en œuvre effective du consensus décentralisé. C'est en ce sens que bitcoin [5] fut inventé en 2008 par une

personne mystérieuse[2] appelé **Satoshi Nakamoto** dans le but de palier aux obstacles notés dans les protocoles monétaires antérieurs. Selon Nakamoto dans [6], « Bitcoin est une solution au problème de double-dépense en utilisant un serveur horodaté, distribué dans un réseau utilisant une preuve de travail pour enregistrer un historique public des transactions. Le système est sécurisé tant que des nœuds honnêtes contrôlent ensemble plus de puissance de calcul qu'un groupe de nœuds qui coopéreraient pour réaliser une attaque. ».

Bitcoin [7] a permis une plate-forme innovante pour les transactions électronique entre pairs **sans autorité centrale.** Pour assurer la sécurité et la confiance, des protocoles sont implémentés pour la **validation**, la **vérification** et le **consensus** : une telle infrastructure est appelé la **blockchain**.

2.1.2. Définitions

Il n'existe pas qu'une seule définition de la Blockchain. Chaîne de blocs en Français, la blockchain est une technologie à la même échelle qu'internet, qui permet de stocker des données numériques, de manière décentralisée et sécurisée.

Techniquement, il s'agit d'une base de données distribuée dont les informations envoyées par les utilisateurs et les liens internes à la base sont vérifiés et groupés à intervalles de temps réguliers en blocs. Les blocs sont sécurisés par cryptographie et forment ainsi une chaîne [8]. Par extension, une blockchain est une base de données distribuée qui gère une liste d'enregistrements protégés contre la falsification ou la modification par les nœuds de stockage ; c'est donc un registre distribué et sécurisé de toutes les transactions effectuées entre ses utilisateurs depuis sa création. Il s'agit donc d'une sorte de livre de compte ou de registre qui contient la liste de tous les échanges effectués entre utilisateurs. Ce registre 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. La blockchain est alors comparable à un grand registre immuable (*ledger*) [5] distribué dans un réseau peer-to-peer dont les pages ne sont rien d'autre que les blocs et les lignes sont les transactions.

Sous sa forme générique, la technologie blockchain fait référence à un système cryptographique entièrement distribué qui capture et stocke un journal des événements **cohérents**, **immuables** et **linéaires** des transactions entre acteurs en réseau. Dans un tel réseau, la technologie Blockchain renforce la transparence et garantit un consensus éventuel à l'échelle du système sur la validité de tout l'historique des transactions. Selon Tschorsch et Scheuermann

[2] Une personne connue que par son pseudonyme, sa véritable identité n'a toujours pas été divulguée.

[9], la technologie Blockchain peut non seulement traiter les transactions monétaires, mais également garantir que celles-ci sont conformes aux règles programmables sous la forme de «contrats intelligents». Comme énoncé dans [10], les transactions dans la Blockchain sont regroupées en blocs et chaque bloc peut contenir un nombre quelconques de transactions tout en respectant une limite de taille. Les nœuds du réseau se synchronisent pour avoir une copie exacte de la blockchain sur l'ensemble du réseau. Un consensus c'est-à-dire un accord entre les pairs du réseau est la procédure de mise à jour de la blockchain.
Une blockchain est alors :

- ✓ **Un réseau distribué :** Des copies identiques de tous les enregistrements sont partagées dans la blockchain. Les participants peuvent vérifier les informations de manière indépendante. Les processus de vérification ne dépendent d'aucune autorité centralisée. Si un nœud tombe en panne, les autres peuvent continuer à fonctionner, garantissant la disponibilité et la fiabilité.
- ✓ **Numérisée :** Presque tous les types d'informations peuvent être exprimés sous forme numérique puis référencés via une entrée du grand livre.
- ✓ **Basée sur le consensus :** Les participants du réseau authentifient et approuvent collectivement les transactions dans la blockchain. Il existe différentes méthodes pour atteindre le consensus. De manière générale, la majorité des participants au réseau doivent accepter l'exactitude de la transaction et les règles peuvent être adaptées aux circonstances.
- ✓ **Mise à jour chronologique :** La blockchain est horodatée en permanence, chaque bloc fait référence aux données stockées dans le bloc précédent de la chaîne, de sorte que tous les blocs sont liés les uns aux autres.
- ✓ **Scellé par la cryptographie :** Scellés dans la chaîne, les blocs ne peuvent plus être changés: la prévention de la suppression, de la modification ou de la copie crée de véritables actifs numériques.

Ces processus de blockchain multipliés et décentralisés conduisent à un niveau élevé de robustesse et de confiance. Chaque participant au réseau a la possibilité de vérifier l'exactitude des transactions. Des méthodes de consensus de réseau et une technologie cryptographique sont utilisées pour valider les transactions. Ainsi, la confiance n'est pas établie de manière externe par une autorité centrale ou un auditeur mais en permanence dans le réseau. De plus, le stockage décentralisé dans une blockchain est connu pour sa grande résistance aux pannes. Même en cas

de défaillance d'un grand nombre de participants au réseau, la blockchain reste disponible, éliminant ainsi le point de défaillance unique. Les nouvelles informations stockées dans une blockchain sont immuables. Sa méthode d'archivage empêche la suppression ou l'inversion des transactions une fois ajoutées à la blockchain, une fois que d'autres blocs ont été ajoutés.

2.1.3. La blockchain en couches

Les composants technologiques sous-jacents aux couches de la Blockchain comprennent les transactions, les blocs, les consensus, les applications et les contrats intelligents. Tous ces composants sont séparés en différentes couches, ce qui équivaut à l'écosystème de blockchain. Les principaux aspects de la blockchain peuvent être divisés en six couches répertoriées comme suit: couches de réseau, de transaction, de blockchain, de confiance, d'application et de sécurité. Chacune de ces couches a des propriétés et des caractéristiques différentes, comme le montre la *Figure 2*.

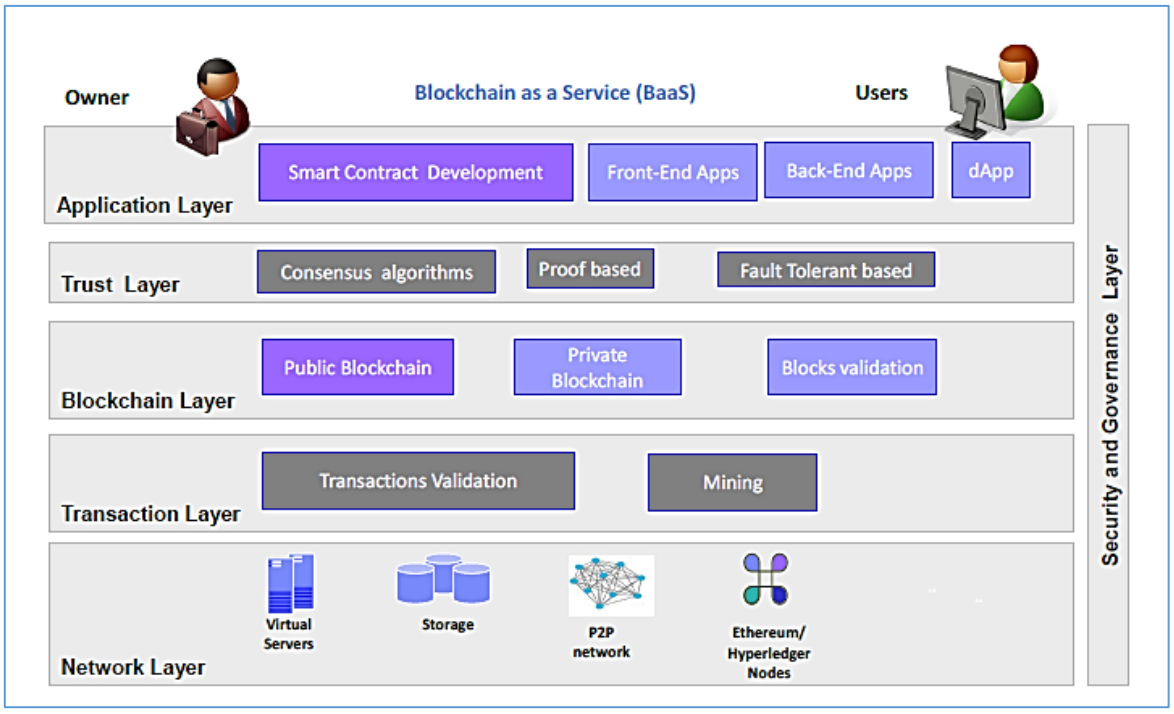


Figure 2: Illustration des différentes couches de la blockchain

La couche réseau fait référence à un réseau P2P avec des nœuds Ethereum ou Hyperledger. La couche de transaction fait référence aux transactions déclenchées par les utilisateurs ou le contrat intelligent. La couche Blockchain fait référence au statut de bloc contenant toutes les informations nécessaires, tandis que la couche de confiance se réfère au protocole de consensus pour la validation de bloc et de transactions. La couche d'application englobe les applications, la machine à états et le contrat intelligent. La couche de sécurité est essentielle pour la

technologie Blockchain. La technologie de la blockchain est vulnérable à de nombreux types d'attaques (l'attaque de 51% est la plus connue).

2.2. Types de blockchain

L'innovation perpétuelle et l'évolution de la technologie blockchain donnent naissance à plusieurs types de blockchain. D'autre part, la blockchain qui était initialement conçue pour les applications de crypto-monnaies est rapidement étendue aux contrats intelligents et de nombreuses nouvelles applications sont déjà envisagées.

2.2.1. Classification basé sur le réseau

La blockchain est un protocole permettant la transmission d'informations de manière sécurisée dans un réseau décentralisé. On distingue alors trois types de blockchains selon les droits d'accès au réseau.

3.1.1.1. Blockchain publique

Une blockchain publique, comme son nom l'indique, est accessibles à n'importe qui: tout le monde peut rejoindre le réseau et partir comme il le souhaite. Ainsi les blocs de transaction et la blockchain peuvent être observés publiquement, même si les participants sont anonymes. Par ailleurs, chacun peut envoyer des transactions et s'attendre à ce qu'elles soient incluses dans le registre ; de toute évidence si ces transactions respectent les règles de cette blockchain. Dans les blockchains publiques, n'importe qui peut participer librement au processus d'approbation c'est-à-dire le processus qui décide le bloc à ajouter à la chaîne et qui définit l'état du système. La Blockchain **Bitcoin** et la blockchain **Ethereum** sont deux principaux exemples de blockchain publiques.

3.1.1.2. Blockchain privée

Dans une **blockchain privée**, l'accès à la blockchain est limité aux participants sélectionnés, par exemple les participants d'une organisation. Cette restriction aide à simplifier les opérations normales telles que la création de blocs et le modèle de contingence. Ainsi le processus d'approbation est contrôlé par un nombre restreint et choisi de nœuds. Ce qui entraine une double modification au système originel, puisque non seulement les participants au processus d'approbation sont limités et sélectionnés, mais aussi ce n'est plus la règle de la majorité qui s'impose. Quant à l'autorisation de lecture de la blockchain, c'est-à-dire l'accès au registre, elle peut être, soit public, soit réservé aux participants du réseau. On peut rencontrer des cas de

blockchains privées où le processus d'approbation est limité à un unique acteur, bien que les autorisations de lecture par exemple puissent être publiques.

3.1.1.3. Blockchain autorisée (« permisionned »)

La troisième catégorie de blockchain est la blockchain autorisée, également appelée ***blockchain consortium***. Il est destiné à un consortium de parties collaborant la transaction sur une chaîne de blocs pour la facilité de gouvernance, de provenance et de responsabilité, par exemple, un consortium de toutes les sociétés automobiles ou organisations de santé. La blockchain autorisée a les avantages d'une blockchain publique en permettant uniquement aux utilisateurs autorisés à collaborer et à effectuer des transactions. L'existence d'une crypto-monnaie n'est pas nécessaire pour ce type de blockchain : ces dernières n'ont en effet pas besoin de rémunérer leurs membres pour la validation des transactions.

Les blockchains privées présentent certains avantages notamment gouvernance simplifiée, acteurs connus, coûts réduits, rapidité, confidentialité, mise en conformité facilitée par les possibilités d'audit y compris par le régulateur... Mais celle-ci réintroduisent des acteurs humains dans la gestion du réseau (gérant l'accès et le fonctionnement) alors que le concept central d'une blockchain (publique) est de supprimer le tiers de confiance.

En dehors de la classification des blockchains en fonction du réseau (public, privé ou autorisé) ; on peut les classer en trois génération selon les types d'applications.

2.2.2. Classification basée sur les applications dérivées

L'infrastructure de la Blockchain a connu une évolution rapide. La blockchain initialement conçue pour les applications de crypto-monnaies, elle est rapidement étendue aux contrats intelligents et de nombreuses nouvelles applications sont déjà envisagées. À cet effet, on ne parle pas d'une blockchain, mais de plusieurs blockchains qui existent, cohabitent, voire interagissent. Ainsi, une blockchain peut posséder des spécificités techniques pour des utilisations ou des applications particulières qui conduisent à trois générations de Blockchain [11], à savoir Blockchain 1.0, Blockchain 2.0 et Blockchain 3.0.

3.1.1.4. Blockchain 1.0 : Les crypto-monnaies

La blockchain 1.0 fait références à la toute première blockchain et son orientation initiale qui était les crypto-monnaies. En effet, la Blockchain a, avant tout, été inventée et développée dans le contexte de la mise en œuvre du Bitcoin qui permet d'effectuer des transactions en ligne sans intermédiaire [12]. La gestion des transactions et la création de monnaie électronique bitcoins (BTC) est prise en charge collectivement par le réseau. La blockchain de Bitcoin est open-

source et tout le code est disponible sur le GitHub [13]. Au cours des premières années, ce code source ouvert a été étendu à la libération de différentes crypto-devises. Environ 300 « **altcoins**[3] » et plus ont été introduites: chacune a ses spécificités mais toutes fonctionnent globalement sur le même principe. On peut recenser, par exemple, les monnaies suivantes :

- ✓ **Litecoin** [14] [15] : créée en 2011, Litecoin est une application qui permet des paiements instantanés, à coûts quasi nuls, à n'importe qui dans le monde. C'est un réseau de paiement mondial open source [16] entièrement décentralisé sans aucune autorité centrale. La blockchain de Litecoin est capable de gérer un volume de transaction plus élevé que son homologue - Bitcoin. En raison de la génération plus fréquente de blocs, le réseau prend en charge davantage de transactions sans qu'il soit nécessaire de modifier le logiciel à l'avenir
- ✓ **Dash** [17]: créée en 2014, elle s'appuie sur le code source de Bitcoin pour le rendre plus sûr et complètement anonyme. Dash permet des payements rapides en rendant privées les informations financières.
- ✓ **Conscoin** [18]: il s'agit d'une crypto-monnaie éthique, qui comporte une forme d'intelligence artificielle pour surveiller l'environnement et garantir des transactions éthiques.

En outre de la libération d'autres crypto-monnaie, la fonctionnalité de script de bitcoin est étendu à un cadre d'exécution de code complet appelé contrat intelligent. Ce qui donne naissance à la deuxième génération de blockchain.

3.1.1.5. Blockchain 2.0 : les contrats intelligents

La deuxième génération, Blockchain 2.0, est marquée par les contrats intelligents (smart contract), et représentée par le succès grandissant de la blockchain d'**Ethereum** [19]. Contrairement au Bitcoin, Ethereum ne se limite pas à des transactions financières mais permet de créer n'importe quelle application décentralisée grâce aux contrats intelligents, qui sont de petits logiciels capables, de manière autonome, d'exécuter automatiquement certaines actions en fonction de conditions prédéfinies. Un contrat intelligent fournit la capacité très puissante d'exécution de code pour intégrer la logique métier de la blockchain. Cette architecture permet de supprimer n'importe quel intermédiaire entre deux acteurs de cette blockchain, l'un étant l'utilisateur et l'autre le fournisseur d'un service. Inscrits dans la blockchain, ces contrats ne peuvent être affectés par aucun changement. Nous réservons un chapitre à ethereum car notre solution est basée sur cette blockchain.

[3] Abréviation de « Alternative coin », un altcoin est une crypto-monnaie autre que bitcoin.

Parmi les applications déjà existantes sur Ethereum, on peut citer les contributions suivantes :

- ✓ **Slock.it** [20] a pour ambition de devenir la future infrastructure de l'économie collaborative. La startup Slock.it, sous le slogan « Rent, sell or share anything – without middleman », développe son offre en combinant les avantages de la Blockchain (absence d'intermédiaire) et ceux de l'Internet des objets. Ainsi, elle propose par exemple des verrous spéciaux pour les portes qui peuvent être débloqués par une personne qui aurait loué un appartement, via son téléphone mobile.
- ✓ **Arcade City** [8] envisage de détrôner Uber et de devenir ainsi la première « *killer app* » de la blockchain. Son but est d'offrir un service de partage de voiture (« *car sharing* ») équivalent, sans la contrainte de l'intermédiaire. L'ensemble des courses réalisées sont contenues dans la blockchain d'Ethereum. L'avantage pour le consommateur se situe au niveau du coût puisque en diminuant le nombre d'intermédiaires, les tarifs baissent également.
- ✓ **The DAO** souhaite devenir la première organisation décentralisée sur la Blockchain. Les membres de « The DAO » auront par exemple la possibilité de voter pour déterminer à quels projets seront distribués des éthers pour encourager le développement de certaines startups.

Les concepts et les caractéristiques de la Blockchain peuvent être étendus à de nombreux domaines d'activités. Ainsi on converge vers la troisième génération de blockchain qui n'implique aucune devise (monnaie) mais prend en charge l'exécution logicielle pour la logique d'entreprise.

3.1.1.6. Blockchain 3.0 : Applications futures

La blockchain 3.0 est assimilé aux applications futures de la blockchain avec un tournant totalement éloigné de l'aspect financier. Ainsi nombreuses expériences existent déjà prouvant l'énorme potentiel de **disruption** de la Blockchain. Nous énumérons brièvement ci-après quelques exemples :

- ✓ **Namecoin** [21] est l'une des premières applications non-financière de la Blockchain. C'est une technologie open source expérimentale qui améliore la décentralisation, la sécurité, la résistance à la censure, la confidentialité et la rapidité de certains composants de l'infrastructure Internet, tels que le DNS et les identités. Namecoin [22] a été inspiré après des discussions sur un protocole BitDNS utilisant une chaîne de blocs pour gérer un service de recherche de nom de domaine. La motivation était qu'une autorité centrale gérant des noms de domaine, telle que l'ICANN, avait trop confiance dans une seule entité et représentait un point de défaillance unique.

- ✓ **Hyperledger Fabric** [23] **:** est un système open-source modulaire et extensible permettant de déployer et d'utiliser des blockchains autorisées et l'un des projets Hyperledger hébergé par Linux Foundation [24]. Il prend en charge les protocoles de consensus modulaires, ce qui permet d'adapter le système à des cas d'utilisation et à des modèles de confiance particuliers. Fabric est également le premier système de chaîne de blocs qui exécute des applications distribuées écrites dans des langages de programmation standard à usage général, sans dépendance systémique à une crypto-monnaie native. Cela contraste vivement avec les plates-formes blockchain existantes qui exigent que les «contrats intelligents» soient écrits dans des langages spécifiques à un domaine ou qu'ils reposent sur une crypto-monnaie.
- ✓ Des initiatives visant l'introduction de la Blockchain dans le cadre des **MOOCs** mais également de la publication académique sont également en cours. Par extension, la Blockchain pourrait aussi se développer au sein des gouvernements. En effet, elle permettrait de proposer certains services étatiques de manière décentralisée, tels que l'enregistrement des mariages, l'émission de passeports, etc… Au final, l'organisation d'élections [25] à l'aide de la Blockchain pourrait également assurer une plus grande transparence aux Etats.

Les blockchains privée et autorisée permettent un accès contrôlé à la blockchain. Par ailleurs les innovations importantes, telles que les contrats intelligents, ont ouvert de nouvelles applications pour la technologie blockchain. Pour toutes ces applications, la technique utilisée pour assurer la sécurité des transactions reste quasiment la même.

2.3. Structure de la blockchain

Bitcoin est la mère de toutes les blockchain. En ce sens, la structure globale des autres blockchain est étroitement similaire à celle de bitcoin. Ainsi pour comprendre la structure des Blockchains, nous présentons la structure de Bitcoin.

La **transaction** est l'élément de base de la blockchain Bitcoin. Les transactions sont validées et diffusées. De nombreuses transactions forment un **bloc**. Beaucoup de blocs forment une **chaîne** via un lien de données numériques [26].

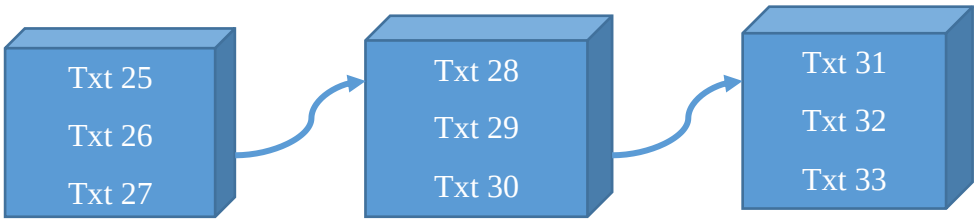


Figure 3: Chaine de blocs

Les blocs passent par un processus de consensus permettant de sélectionner le prochain bloc à ajouter à la chaîne. Le bloc choisi est vérifié et ajouté à la chaîne en cours. Le processus de validation et de consensus est effectué par des nœuds spéciaux appelés **mineurs**. Ce sont des ordinateurs dotés de grosses capacités de traitement, directement utiles au fonctionnement de la Blockchain et capables d'émettre des transactions au même titre qu'un nœud régulier.

2.3.1. La transaction

Une transaction représente un paiement en Bitcoins, en d'autre terme un transfert d'actifs (de bitcoins). Elle contient des données techniques et caractérisée par des entrées et sorties.

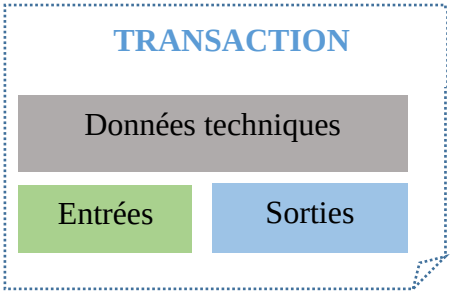


Figure 4: Forme simplifiée d'une transaction Bitcoin [26]

Les **données techniques** sont :

- Le **numéro de version** : utilisé pour spécifier à quel ensemble de règle du protocole cette transaction se réfère,
- Transaction à terme (**nLockTime**) : l'horodatage ne rendant valide une transaction qu'à partir d'une date futur, similaire à un chèque postdaté,
- Le **décompte des entrées** : le nombre d'entrées contenu dans cette transaction,
- Et **le décompte des sorties** : combien de sortie cette transaction crée.

Les entrées sont les « **coins** » dépensés. Techniquement, elles sont caractérisées par la longueur du script d'entrée (le nombre de données d'entrée) et du numéro de séquence. Par ailleurs les entrées de la transaction contiennent d'une part, le **hash de la précédente transaction et l'index** qui permet d'identifier d'où proviennent les **coins** en spécifiant une valeur de sortie d'une précédente transaction et d'autre part, les données du script où l'on prouve qu'on possède les **coins** et qu'on est autorisé à les dépenser, en signant avec la clé privée de l'adresse contenant les bitcoins.

Les sorties sont les coins reçus par le destinataire de la transaction. Elles contiennent trois propriétés : la **longueur du script** de sortie (nombre de données de sortie), le **montant** c'est à dire le nombre de bitcoins envoyés et **le script de sortie** qui détermine à qui (quelle/s adresse/s) ils sont envoyés et quelles sont les signatures requises pour ré-dépenser ces bitcoins

Les entrées et les sorties font apparaitre un concept fondamental du réseau bitcoin : les sorties de transaction non dépensées (« Unspent Transaction Output ») noté **UTXO** [27]. L'ensemble de tous les UTXO d'un réseau Bitcoin définit collectivement l'état de la chaîne de blocs Bitcoin. Les UTXO sont référencés en tant qu'entrées dans une transaction et sont également des sorties générées par une transaction. Tous ces UTXO se trouvent dans un système et sont stockés par les nœuds participants dans une base de données. La transaction utilise la quantité spécifiée par un ou plusieurs UTXO et la transmet à un ou plusieurs UTXO de sortie nouvellement créés, en fonction de la demande initiée par l'expéditeur. On exprime en paradigme le mécanisme de validation d'une transaction par un nœud:

- Pour chaque entrée dans la transaction :
  - Si l'UTXO référencé n'est pas dans la base de données des UTXOs, retourner une erreur
  - Si la signature fournie ne correspond pas à celle du propriétaire de l'UTXO (i.e. pas la même adresse), retourner une erreur
- Si la somme des montants des UTXOs référencés en entrée n'est pas égale à la somme des montants des UTXOs en sortie, retourner une erreur
- Mettre à jour la base de données des UTXOs.

Un exemple de transaction bitcoin (disponible sur blockchain.com) est donné à la Figure 5.

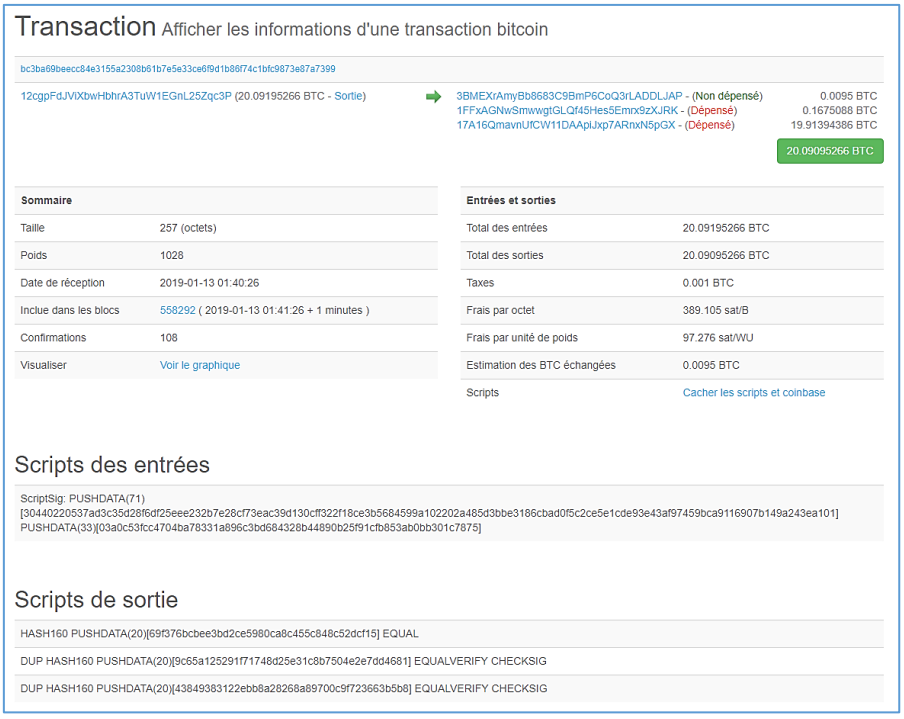


Figure 5: Exemple de transaction Bitcoin

2.3.1. Le bloc

Les blocs sont des unités de la Blockchain comparables aux pages de transactions dans un livre de comptes. Un bloc est composé d'un **en-tête** et d'un **contenu** dans lequel les transactions sont regroupées.

L'en-tête du bloc est haché deux fois pour créer l'empreinte numérique à laquelle on se réfère dans le bloc suivant. Il contient :

- ✓ Les **données techniques** qui incluent un ID magique, un numéro de version (spécifier à quel ensemble de règle du protocole ce bloc est conforme) et la taille du bloc.
- ✓ Le **hash du bloc précédent** : le hash de l'en-tête du précédent bloc (excluant l'ID magique et la taille de bloc). C'est le lien créant la chaine de blocs, et ce depuis le bloc de genèse (« *Genesis block* »). Ainsi, à la Figure 6, le bloc 2 est bien situé entre les blocs 1 et 3, ce qui peut être vérifié en s'assurant que le haché du bloc 1 est correctement renseigné dans l'entête du bloc 2 et de même pour le hash du bloc 2 dans l'entête du bloc 3.
- ✓ La **racine de Merkle** : toutes les transactions du bloc sont distillées en un hash unique. En effet un identifiant (TxID) qui est le hash du contenu de la transaction est calculé pour chaque transaction. L'arbre de Merkle permet ensuite de solidifier l'ensemble des transactions par calcul de hashs successifs, et ce jusqu'à trouver la racine de l'arbre.
- ✓ **L'horodatage** de la création du bloc
- ✓ **La difficulté ciblée :** liée au minage et à la difficulté de miner le bloc avec succès.
- ✓ **Et le nonce :** qui est nombre aléatoire. C'est l'une des choses que l'on peut modifier en minant, pour créer différents hashs et trouver le hash adéquat.

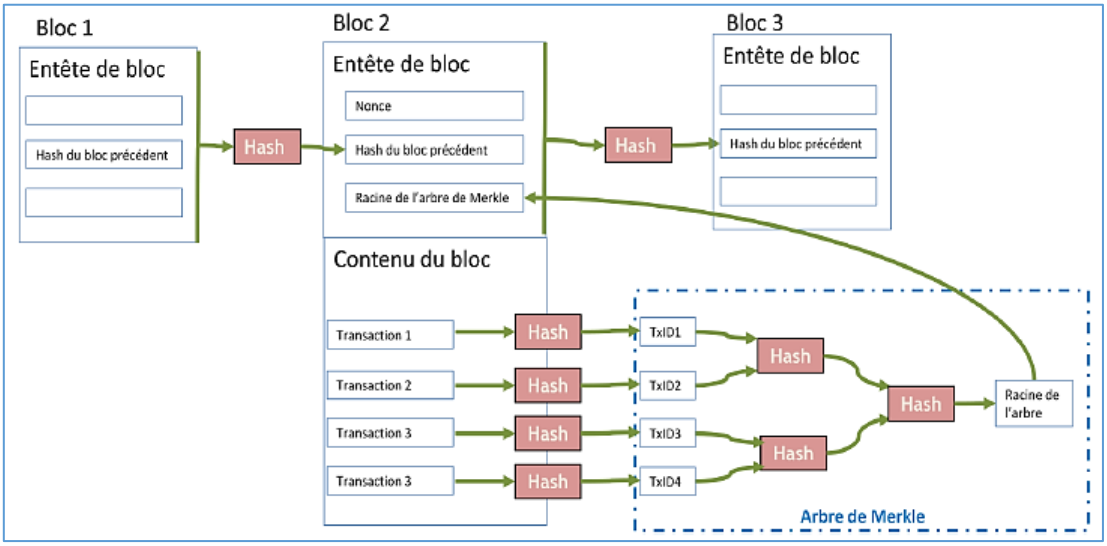


Figure 6: Forme simplifiée d'un bloc Bitcoin [28]

Le contenu du bloc est constitué de la transaction **coinbase** qui est transaction spéciale où il n'y a ni entrées, ni adresse d'expédition et l'ensemble des transactions du bloc qui représente la charge principale.

Concrètement, à la Figure 7, nous montrons l'exemple du bloc de genèse (visible sur blockchain.com). C'est là que tout a commencé. Satoshi Nakamoto a lancé la Bitcoin Blockchain avec une transaction de 50 Bitcoins ou 50 BTC, la date de la création du Bloc est le 3 janvier 2009. Il n'y avait pas de bloc précédent. La récompense de bloc est de 50 BTC.

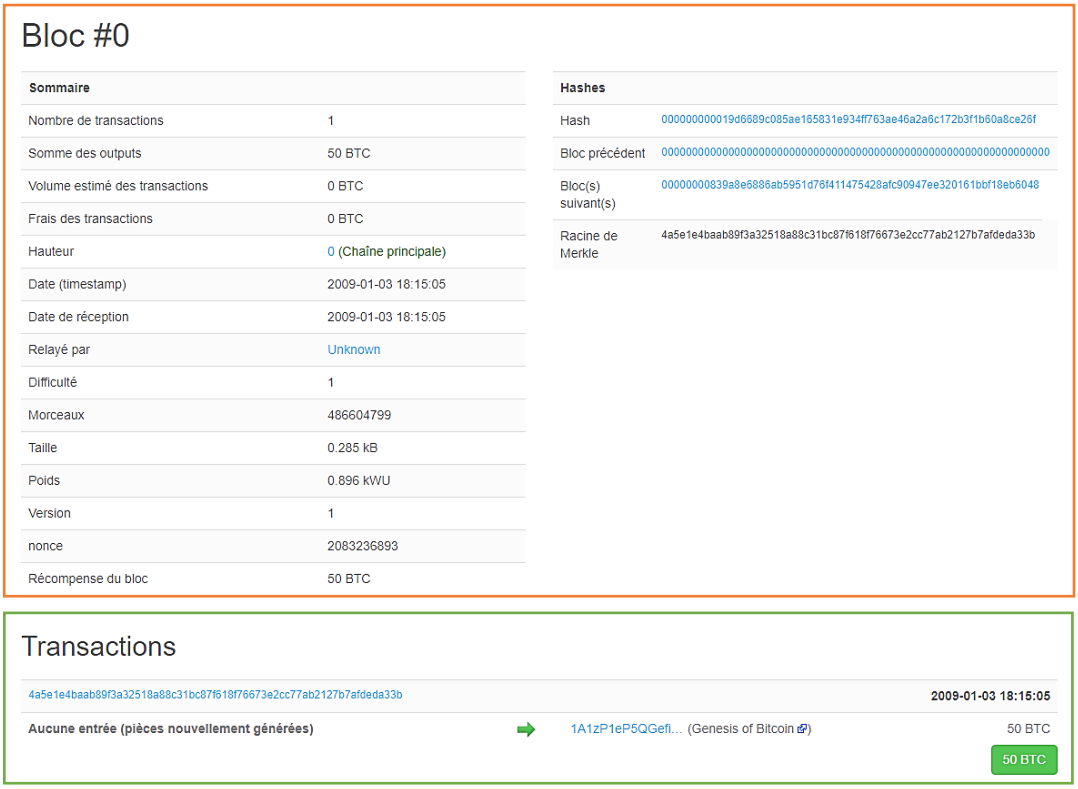


Figure 7: Bloc de genèse

La blockchain est constituée d'un ensemble de transactions regroupées sous forme de blocs liés conjointement en une chaine. Les processus validations et de création de nouveau bloc représentent ainsi les opérations effectuées dans le réseau.

2.1. La sécurité

La sécurité, dans le domaine de l'informatique, consiste à garantir que les ressources matérielles ou logicielles d'une organisation sont uniquement utilisées dans le cadre prévu. Il convient à toute organisation de protéger son patrimoine qui est représenté, essentiellement, par son système d'information.

Un système informatique est sécurisé s'il incorpore des mécanismes permettant de faire face aux vulnérabilités, menaces, risques et attaques. En effet, une vulnérabilité ou une faille d'un système est une faiblesse ou un logiciel permettant à un attaquant de porter atteinte à la sécurité

d'une information ou d'un système d'information. Les menaces, quant à elles, sont les actions potentiellement nuisibles au système. Un risque désigne la probabilité d'un événement dommageable ainsi que les coûts qui s'ensuivent. Le risque dépend également des montants des valeurs à protéger. Une attaque est l'exploitation d'une faille d'un système informatique à des fins non connues par l'exploitant des systèmes et généralement préjudiciables. Pour sécuriser un système, on doit assurer :

- ✓ **L'authentification** : L'identification des utilisateurs est primordiale pour gérer les accès aux espaces de travail pertinents et maintenir la confiance dans les relations d'échange.
- ✓ **L'intégrité** : S'assurer que les données ne sont pas altérées de façon fortuite, les éléments considérés doivent être exacts et complets
- ✓ **La confidentialité** : Limiter l'accès aux informations qu'aux personnes autorisées.
- ✓ **La disponibilité** : Garantir le fonctionnement du système sans faille, et l'accès aux services et ressources installés avec le temps de réponse voulu.
- ✓ **La non-répudiation** : C'est la propriété qui assure la preuve de l'authenticité d'un acte, c'est-à-dire qu'aucun utilisateur ne peut ensuite contester les opérations qu'il a réalisées, et qu'aucun tiers ne pourra s'attribuer les actions d'un autre utilisateur.

Le stockage d'information dans un système distribué quelconque, et donc aussi dans une blockchain, est sujet à des limitations fondamentales, connues sous le nom de « théorème CAP » : il est impossible de garantir à la fois la cohérence (C) et la disponibilité (A) des données au fil du temps dans un réseau de partitions (P). Les propriétés de sécurité que les blockchains souhaitent assurer sur un réseau dont on se méfie car non sécurisé (engageant donc cohérence et résistance aux partitions) sont obtenues aux dépens de la disponibilité d'une information récente qui a été objet d'accord, ou « consensus », par l'ensemble du système.

Dans une blockchain, faute d'une autorité centrale qui détermine l'état correct, le consensus est déterminé de façon distribuée par application de techniques probabilistes (telles que la preuve de travail) pour l'élection d'un nœud, qui sera responsable pendant un temps limité de l'évolution de l'état du système (typiquement, pour la durée d'un seul bloc).

2.1.1. Les vulnérabilités, menaces, risques et attaques dans la blockchain

La technologie blockchain est un « simple » composant d'un système, d'une application, ou d'un service, tout comme une base de données dans une application métier. Elle fournit des

propriétés de sécurité, mais les vulnérabilités et les risques doivent également être évalués. Pourtant, dans la pratique, plusieurs attaques existent qui manipulent directement ou indirectement le mécanisme de récompense, donnant d'injustes avantages aux mineurs de large taille aux frais des petits mineurs. Si la sécurité de Bitcoin était censée initialement être compromise par une attaque de type « 50%+1 », on reconnait aujourd'hui que la taille critique (en termes de puissance de calcul) d'un acteur malveillant qui souhaiterait manipuler le système d'incitation en sa faveur pourrait être bien moins large : on estime ce seuil critique pour lancer une attaque connue comme « selfish mining » à 30% ou moins [29]. D'autres travaux concluent que tout arbitrage dont les mineurs jouissent, tel que le choix des transactions qui rentrent dans un bloc de la blockchain, permet de manipuler le comportement des autres mineurs [30]. Et comme la valeur intrinsèque d'une unité de crypto-monnaie est difficile à estimer et sujette à une forte spéculation, tout phénomène de 'bulle' de prix de marché risque à la fois d'intensifier l'engagement des mineurs, en augmentant la consommation totale d'énergie, et de décourager l'utilisation de la crypto-monnaie pour exécuter des transactions légitimes, que ce soit par des frais élevés [31], [32] ou par une volatilité des prix accrue. Ceci tout en attisant la vigueur des attaques contre le système, qui risquent à tout moment de pulvériser la confiance des utilisateurs et des mineurs si une faille sérieuse au niveau des algorithmes ou des protocoles était soudainement exploitée.

Tel que décrit par le groupe Sécurité du comité ISO TC307, il existe plusieurs propriétés de sécurité fournies par les systèmes blockhain, appelés dans le contexte de l'ISO TC307 DLTs (*Distributed Ledger Technologies*). Ce qui suit est une liste de ces propriétés de sécurité. Certaines d'entre elles sont des propriétés de sécurité communes à toutes les applications basées sur DLT tandis que d'autres sont facultatives et dépendent de la nature de l'application. Cette liste est provisoire et est susceptible de changer :

- ✓ L'intégrité : les enregistrements dans le *ledger* sont protégés contre toute modification après leur création. Aussi connu comme l'immutabilité ou la résistance à l'altération.
- ✓ L'Authenticité: toute personne (ou un ensemble certifié d'entités) peut vérifier l'entité qui crée une transaction enregistrée dans le registre.
- ✓ La confidentialité : les enregistrements dans le registre ne peuvent être consultés que par une entité autorisée.
- ✓ L'ordre des événements : l'ordre des enregistrements dans le *ledger* ne peut pas être modifié.
- ✓ La disponibilité : les transactions enregistrées dans le *ledger* et la fonctionnalité pour enregistrer et récupérer les données sont toujours disponibles pour les utilisateurs.

- ✓ La résilience : le système DLT continue de fonctionner comme prévu en cas d'échecs et d'autres incidents. C'est une sorte d'exigence de disponibilité.
- ✓ « *Trusted-server less* » : même s'il n'y a pas d'entité (serveur de confiance), la Blockchain (DLT) continue de fonctionner comme prévu.
- ✓ L'assurance à long terme : certains cas d'utilisation (par exemple les registres fonciers) impliquent un maintien à long terme du grand livre et un transfert en toute sécurité de ses dossiers à un autre système en cas de déclassement.

Les risques liés aux composants qui interagissent avec une Blockchain telles que des personnes physiques ou morales qui utilisent des interfaces, des applications, des objets intelligents comme des compteurs électriques ou des capteurs, sont multiples. Il faut par exemple que tous ces composants aient un identifiant unique et sécurisé sur la blokchain afin d'éviter toute usurpation d'identité, et de pouvoir associer une responsabilité à une action dans la Blockchain.

Les risques liés aux données doivent également être analysés. Comment s'assurer par exemple qu'une transaction soumise au système sera bien validée et qu'il n'y a pas eu d'altération des données par un *smart contact*, ou le système lui-même ? L'intégrité des données injectées et manipulées est un point clef de la blockchain. Comment respecter des cas d'usage ou des régulations en place qui exigent une confidentialité des données, comme les données personnelles ?

Du point de vue de la norme ISO, les risques et les vulnérabilités des systèmes DLT comprennent des risques et vulnérabilités communs aux systèmes d'information tels que :

1. Mauvaise gestion de l'information (altération, suppression, destruction non autorisée, divulgation, etc.).
2. Vulnérabilités de mise en œuvre (y compris les mécanismes cryptographiques, vulnérabilités de mise en œuvre, fuites d'informations au moment de l'exécution, etc.).
3. Mauvaise gestion des mécanismes cryptographiques (y compris l'utilisation d'algorithmes faibles, la divulgation de clé).
4. Mauvaise gestion des privilèges de l'utilisateur.

L'identification des vulnérabilités et/ou menace des blockchains permet de mieux comprendre les mécanismes de sécurité incorporés dans leurs protocoles.

2.1.2. Mécanismes de sécurité

Pour faire face aux menaces et vulnérabilités d'un système, plusieurs mécanismes de sécurité doivent être mis en place. Chaque mécanisme est spécifique et peut même présenter des faiblesses ou des avantages par rapport à un autre.

3.1.1.7. Cryptographie

La cryptographie est la science qui utilise les mathématiques pour le cryptage et le décryptage de données. Elle nous permet ainsi de stocker des informations confidentielles ou de les transmettre sur des réseaux non sécurisés (tels que l'Internet), afin qu'aucune personne autre que le destinataire ne puisse les lire. Il existe deux type de cryptographie : symétrique (ou à clé secrète) et asymétrique (ou à clé publique).

2.1.2.1.1. Cryptographie Symétrique

En cryptographie conventionnelle, également appelée cryptage de clé secrète ou de clé symétrique, une seule clé suffit pour le cryptage et le décryptage. Toute la sécurité cette cryptographie est directement liée au fait que la clé de chiffrement n'est connue que par l'expéditeur et le destinataire. La cryptographie symétrique est très utilisée et se caractérise par une grande rapidité (cryptage à la volée, "on-the-fly"), des implémentations aussi bien software (Krypto Zone, firewalls logiciels type Firewall-1 et VPN-1 de Checkpoint) que hardware (cartes dédiées, processeurs cryptos 8 à 32 bits, algorithmes câblés...) ce qui accélère nettement les débits et autorise son utilisation massive. Cependant, elle présente des inconvénients. En effet la distribution des clés est le principal problème de cette méthode de chiffrement. Si la même clé est utilisée par plus de deux personnes, elle doit être abandonnée lorsqu'une copie est interceptée. Elle ne peut pas être authentifiée car elle est connue de plus d'une personne. D'autre part, la sécurité incertaine lors du transfert de la clé et la nécessité de générer autant de clés que de couples de correspondants sont à soulignés.

2.1.2.1.2. Cryptographie Asymétrique

La cryptographie à clé publique [33], encore appelée cryptographie asymétrique, est une méthode de chiffrement qui utilise deux clés qui se ressemblent mathématiquement mais qui ne sont pas identiques : une **clé publique** et une **clé privée**. À l'inverse des algorithmes de cryptographie symétrique qui dépendent d'une seule clé pour le chiffrement et le déchiffrement, les clés de la cryptographie asymétrique ont chacune une fonction bien spécifique : la clé publique sert à chiffrer et la clé privée sert à déchiffrer [34]. Le fait qu'il est impossible de deviner la clé privée à partir de la clé publique lui confère un niveau de sécurité élevé. En ce

sens les clés publiques peuvent être partagées sans danger, permettant ainsi aux utilisateurs de bénéficier d'une méthode facile et pratique de chiffrement de contenu et de vérification de signature numérique. Quant aux clés privées, elles restent secrètes. Ainsi seul leur propriétaire peut déchiffrer du contenu et créer des signatures numériques.

Par exemple, considérons une communication entre deux entités A et B à la Figure 8.

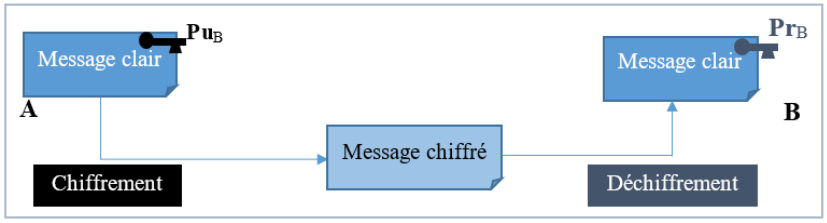


Figure 8: Principe de la cryptographie asymétrique

A et **B** génère leurs paires de clés. **A** chiffre son message avec la clé publique de **B** afin de garantir que seul **B** pourra déchiffrer le message. Ainsi pour lire le massage de **A**, **B** décrypte le message chiffré à l'aide de sa clé privée.

Il existe plusieurs crypto-systèmes qui implémentent la cryptographie asymétrique. Puisque la paire de clés publique privée est fréquemment utilisée dans de nombreuses opérations différentes du protocole de la blockchain, il est important de choisir un algorithme efficace et puissant. C'est à ce sens que le crypto-système ECC (Elliptic Curve Cryptography) a été retenu dans la blockchain bitcoin et dans la blockchain Ethereum pour générer la paire de clés, avec l'algorithme de signatures **ECDSA** [35] (Elliptic Curve Digital Signature Algorithm). L'algorithme ECDSA s'appuie sur les courbes elliptiques qui ont l'avantage de garantir, pour un même niveau de sécurité, des tailles de clés raisonnables par rapport à d'autres crypto-systèmes à clés publiques plus classiques tels que le RSA **(Rivest Shamir Adleman)**. En effet, pour un niveau de sécurité 112[4], là où le RSA exige des clés de 3072 bits, ECC exige seulement 256 bits [36].

3.1.1.8. Signature numérique

Les signatures numériques permettent au destinataire de vérifier l'authenticité des données, leur origine, mais également de s'assurer qu'elles sont intactes. Ainsi, les signatures numériques garantissent l'authentification et l'intégrité des données. Elles fournissent également une fonctionnalité de non répudiation. Ces fonctions jouent un rôle tout aussi important pour la

[4] Un niveau de sécurité 112 correspond à un attaquant à qui on confère la capacité de réaliser 2^{112} opérations pour réaliser une attaque

cryptographie que la confidentialité, sinon plus. Une signature numérique a la même utilité qu'une signature manuscrite. Cependant, une signature manuscrite peut être facilement imitée, alors qu'une signature numérique est pratiquement infalsifiable. De plus, elle atteste du contenu des informations, ainsi que de l'identification du signataire.

3.1.1.9. Hachage

Les fonctions de hachage [28] sont des fonctions mathématiques qui permettent de transformer une chaine de caractères de longueur indifférente en une autre chaine de longueur fixe. Elles sont caractérisées par :

- ✓ L'irréversibilité : Connaissant le résultat de la fonction, il est très difficile[5] de trouver le message fourni en entrée.
- ✓ La résistance aux collisions : Il est très difficile de trouver deux messages qui aboutissent au même haché.
- ✓ Effet avalanche : La modification d'un bit du message en entrée aboutit à au moins la moitié des bits modifiés en sortie. Cette propriété est intéressante pour garantir la propriété d'intégrité.

On distingue le hachage simple et le hachage arborescent appelé l'arbre de Merkle**.**

Dans l'approche de hachage simple, toutes les données sont arrangées et hachées de manière linéaire. Le hachage simple est utilisé pour un nombre fixe d'éléments à hacher, tels que les éléments d'un en-tête de bloc mais aussi pour vérifier l'intégrité du bloc composite et non l'intégrité de chaque élément. La fonction de hachage **SHA-256** est utilisée dans bitcoin.

Dans une approche arborescente, un arbre de Merkle est un arbre binaire, composé d'un ensemble de nœuds avec un grand nombre de nœuds feuilles au bas de l'arborescence contenant les données sous-jacentes (si le nombre est impair, le dernier élément sera dupliqué), un ensemble de nœuds intermédiaires où chaque nœud est le hachage de ses deux enfants. Et enfin un seul nœud racine, également formé à partir du hachage de ses deux enfants, représentant la "racine" de l'arbre. Un exemple d'un arbre de Merkle est donné à la Figure 9.

[5] « Très difficile » signifie que les outils algorithmiques et informatiques actuels ne le permettent pas en un temps raisonnable.

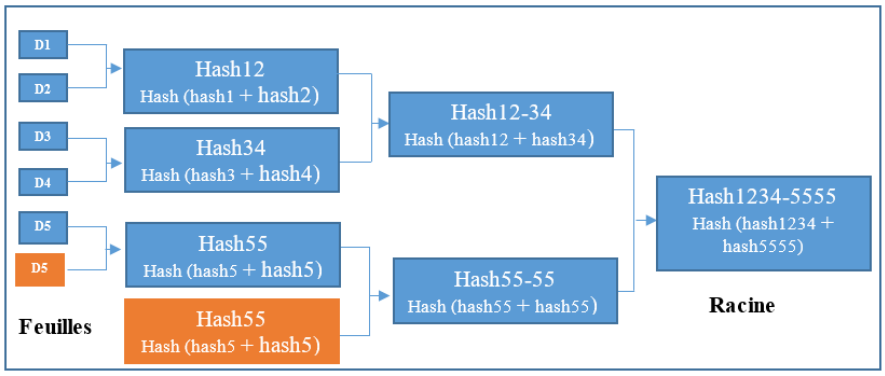


Figure 9: Arbre de Merkle

On a cinq données à regroupés sous forme d'arbre de Merkle. La donnée D5 est dupliquée pour former les nœuds intermédiaires. La racine de l'arbre est donc le hachage de toutes les données combinées par paires.

Le hachage par arborescence est utilisé pour un nombre d'éléments variable d'un bloc à l'autre, par exemple, le nombre de transactions. En effet, chaque bloc contient des milliers et des milliers de transactions. Ce qui rend difficile le stockage de toutes les données à l'intérieur de chaque bloc en série. Par conséquent la recherche d'une transaction particulière extrêmement fastidieuse et longue. L'utilisation de l'arbre de Merkle réduit considérablement le temps nécessaire pour déterminer si une transaction particulière appartient ou non à un bloc.

Les fonctions de hachage sont utilisées pour générer les adresses de compte, les signatures numériques, le hachage de transaction, le hachage d'état et le hachage d'en-tête de bloc.

Le système décrit précédemment comporte certains problèmes. Il est lent et produit un volume important de données (au moins le double de la taille des informations d'origine). L'ajout d'une fonction de hachage à sens unique dans le processus permet d'améliorer ce système. Cette fonction traite une entrée de longueur variable afin d'obtenir en sortie un élément de longueur fixe, à savoir 160 bits. En cas de modification des données même d'un seul bit, la fonction de hachage garantit la production d'une valeur de sortie complètement différente.

Deux techniques sont principalement utilisées pour assurer la sécurité de la blockchain ainsi que l'efficacité de la validation et de la vérification des transactions: la cryptographie à clés publiques et des fonctions de hachage.

2.1.3. L'intégrité des données

Un système distribué tel qu'une blockchain publique demande la participation d'un grand nombre d'acteurs pour fournir des propriétés de sécurité adéquates à la sauvegarde de l'intégrité des données conservées par le registre (*ledger*). L'utilisation de la preuve de travail (Proof of Work – PoW) en tant que mécanisme de sécurisation d'une blockchain telle que Bitcoin

entraine une consommation énergétique importante, dont le coût est absorbé entièrement par les mineurs.

La combinaison des fonctions de hachage et de la cryptographie à clé publique permet de gérer l'intégrité des transactions en **sécurisant les adresses uniques des comptes**[6], en **autorisant la transaction** par l'expéditeur via la **signature numérique** et en **vérifiant** que le contenu de cette transaction n'est pas modifié.

Les adresses des comptes sont générées à l'aide des paires de clé publique, clé privée. Pour se faire, un nombre aléatoire de 256 bits, est généré et désigné comme **clé privée**. Il est gardé en sécurité et verrouillé à l'aide d'une phrase secrète. Ensuite un algorithme ECC est appliqué à la clé privée pour obtenir une **clé publique** unique. En fin une fonction de hachage est appliquée à la clé publique pour obtenir l'adresse du compte. L'adresse est de taille plus courte, seulement 20 octets ou 160 bits.

La clé privée permet signer les transactions et la clé publique correspondante permet de vérifier que l'expéditeur coïncide avec le signataire des transactions. La signature numérique comporte deux phases:

- ✓ Phase de signature: l'expéditeur hache les données et les signe avec la signature numérique générée avec sa clé privée. Ensuite, le hachage signé est envoyé avec les données originales au destinataire.
- ✓ Phase de vérification: les données signées sont déchiffrées avec la clé publique de l'expéditeur et comparées à la valeur de hachage des données originales.

On note que, dans les deux phases, la fonction de hachage utilisée doit être la même (par exemple, SHA256 pour la blockchain Bitcoin). La Figure 10 illustre ces deux phases.

Pour signer numériquement une transaction, on calcule le hachage des champs de données de cette transaction. Ce hachage est ensuite chiffré à l'aide de la clé privée de l'expéditeur de la transaction afin de l'autoriser et la rendre non répudiable. La transaction est vérifié par d'autres personnes en la décryptant à l'aide de la clé publique de son expéditeur et en recalculent le hachage de la transaction: si le hachage calculé et le hachage reçu à la signature numérique correspondent, la transaction est acceptée ; sinon, elle est rejetée. Ainsi, la signature et la vérification permettent d'**autoriser** la transaction et la rendre **non répudiable** et **non modifiable**.

[6] Une approche standard pour identifier de manière unique les participants au réseau décentralisé

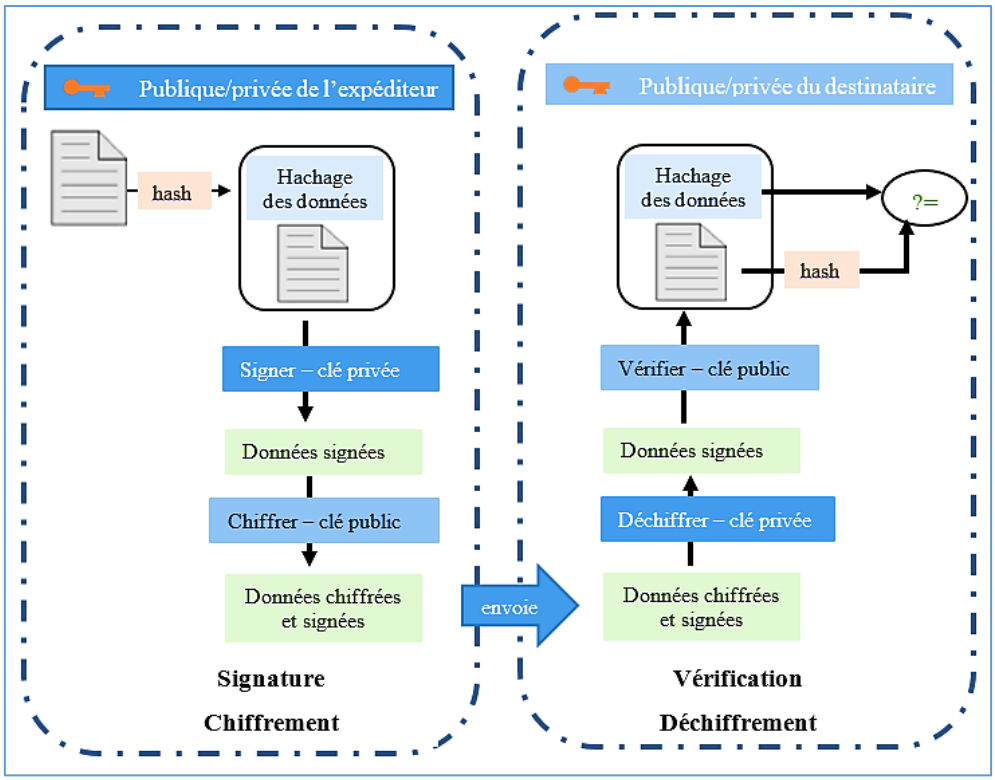


Figure 10: Cryptographie asymétrique: phases de signature-chiffrement et de vérification-déchiffrement

L'intégrité du bloc est gérée en s'assurant que le contenu de l'en-tête de bloc n'est pas falsifié, que les transactions ne sont pas tempérées et que les transitions d'état sont efficacement calculées, hachées et vérifiées. Ainsi les transactions dans un bloc sont hachées par l'arbre de Mekle. Chaque changement d'état nécessite un nouveau calcul du hachage de la racine d'état. Au lieu de calculer le hachage pour l'ensemble des états, seul le chemin affecté dans l'arbre de Merkle doit être recalculé. D'autre part, le hachage de bloc est obtenu en calculant d'abord le hachage de l'état de la racine, le hachage de la transaction racine, puis le hachage de la réception racine. Ces racines et tous les autres éléments de l'en-tête sont des hachages avec les nœuds variables pour résoudre le puzzle de la preuve de travail.

Le hachage du bloc a deux objectifs principaux : la vérification de l'intégrité du bloc et des transactions et la formation du maillon en incorporant le hachage du bloc précédent dans l'en-tête du bloc actuel. Si un nœud participant altère le bloc, les valeurs de hachage changent, ce qui entraîne une incompatibilité entre les valeurs de hachage et rend la chaîne locale du nœud dans un état non valide. Tout futur bloc initié par le nœud serait rejeté par d'autres mineurs en raison de la non-concordance de hachage. On parle de **l'immuabilité de la blockchain**.

2.2. Les opérations de base sur la blockchain

La blockchain est un réseau décentralisé peer-to-peer avec deux principaux types de nœuds participatifs. D'une part, les participants qui initient le transfert de valeur en créant une transaction et d'autre part, les participants appelés **mineurs**.

Les mineurs effectuent un travail ou des calculs supplémentaires pour vérifier les transactions, diffuser une transaction. Ils se concurrencent pour revendiquer le droit de création d'un nouveau bloc et réalisent un consensus pour valider un bloc. Les mineurs assurent aussi la diffusion du bloc nouvellement créé et la confirmation des transactions. Ils sont encouragés par les bitcoins récompensés pour gérer la blockchain.

2.6.1. Validation d'une transaction

La validation de la transaction est effectuée indépendamment par tous les mineurs. Le processus dans bitcoin implique la validation de plus de 20 critères dont:

- la taille,
- la syntaxe,
- les références de l'UTXO d'entrée et de sortie,
- la validité des UTXO,
- les références du montant en entrée et du montant en sortie sont suffisamment appariées.

Les transactions non valides sont rejetées et ne seront pas diffusées. Toutes les transactions valides sont ajoutées à un pool de transactions. Les mineurs sélectionnent un ensemble de transactions dans ce pool pour créer un bloc. La transaction zéro (l'index zéro du bloc confirmé) est créé par le mineur du bloc. Il a un UTXO spécial et n'a pas d'entrée UTXO. C'est ce qu'on appelle la transaction « **coinbase »** qui génère les honoraires d'un mineur pour la création du bloc.

2.6.2. Création d'un nouveau bloc (minage)

La blockchain est une chaîne de flux liée unique et cohérente. Cependant, si chaque mineur ajoute le bloc à la chaîne, la chaîne comportera de nombreuses branches, ce qui entraînera un état incohérent. Ainsi un système pour surmonter ce défi serait indispensable. Les mineurs se concourent pour créer un nouveau block et obtenir la récompense. En effet, le minage est une opération qui permet au mineur de trouver un bloc valide par la résolution d'un problème mathématique complexe et de s'octroyer un gain.

Une fois qu'un mineur ait résolu le problème, l'annonce est diffusée sur le réseau et le bloc est également diffusé sur le réseau. Ensuite, un autre mineur vérifie le nouveau bloc. Ils parviennent à un consensus pour ajouter un nouveau bloc à la chaîne. Ce nouveau bloc est ajouté à leur copie locale de la blockchain. Ainsi, un nouvel ensemble de transactions est enregistré et confirmé. L'algorithme de consensus est appelé protocole de preuve de travail (proof-of-work), puisqu'il nécessite beaucoup de calculs de la part du mineur et consiste à trouver la valeur du champ Nonce de 32 bits à renseigner dans l'entête du bloc pour que le hachage de l'entête du bloc aboutisse à un résultat inférieur à une certaine valeur. Plus cette valeur est petite, plus le problème est difficile à résoudre. Pour conserver une même complexité calculatoire au fil du temps, il est intéressant que la Blockchain ajuste le niveau de difficulté. C'est le cas pour le projet Bitcoin qui s'appuie sur une **moyenne de minage d'un bloc de 10 minutes**. Ainsi, tous les 2016 blocs qui correspondent à une période théorique de deux semaines, une moyenne est calculée ; si le temps moyen est trop court, la difficulté est alors revue à la hausse ; s'il est trop long, la difficulté est revue à la baisse.

La nature probabiliste de cette sélection donne également lieu à la création de divergences (ou « forks ») dans l'histoire de l'état, qui doivent être résolues de façon univoque pour qu'un consensus stable soit rapidement achevé. Par exemple, les systèmes utilisant PoW considèrent la longueur de la chaine pour décider : la chaine la plus longue, qui correspond aux transactions validées (avec haute probabilité) par la majorité de la capacité de calcul du système, définit à tout moment l'état du consensus. La convergence de ce processus nécessite un temps de latence important : ce dernier ne peut pas être réduit au-delà d'un certain seuil en accélérant la fréquence de génération des blocs, car cela engendrerait une augmentation de l'occurrence des forks. Cette limitation fondamentale découle du théorème CAP et constitue donc un obstacle à la scalabilité des blockchains.

Ainsi, les opérations principales dans une blockchain sont la **validation de transaction** et la **création de bloc** avec le consensus des nœuds participants. En outre, il existe également de nombreuses opérations implicites sous-jacentes dans la blockchain bitcoin.

En synthèse de ce chapitre, le projet Bitcoin créé en 2008 dans le but d'échanger des crypto-monnaie (des BTC) sur un réseau décentralisé sans faire appel à un tiers de confiance est à l'origine de la technologie Blockchain. La Blockchain est souvent comparée à un gros livre de comptes publiquement accessible et vérifiable. Ses membres (les nœuds) peuvent y ajouter des écritures, mais cette opération nécessite une validation par plusieurs membres du groupe, voire la majorité du groupe.

L'ouverture du code source de Bitcoin a permis la génération de plusieurs « Altcoin » offrant une diversité d'application de la technologie Blockchain. L'évolution de l'architecture et l'ajout de fonction permet également de converger vers plusieurs types de Blockchain notamment les blockchains publique, privée et autorisée.

Des mécanismes de sécurité tels que la cryptographie asymétrique et les fonctions de hachage sont utilisé dans plusieurs processus de la Blockchain afin de garantir l'intégrité des transactions et rendre inviolables les données.

En 2014, la fondation à but non lucratif Ethereum se lance dans le projet d'étendre le principe de la Blockchain à une Blockchain programmable, ouvrant ainsi le champ à tout type de transactions (smart contracts) et à de nouveaux services. Dans le chapitre suivant, nous verrons en détail cette blockchain.

CHAPITRE III : LA BLOCKCHAIN ETHEREUM

ethereum

Bitcoin est la mère de toutes les blockchains. Elle était destinée au transfert de valeur de pair à pair sans aucun intermédiaire. Vers 2013, un cadre pour l'exécution de code a été introduit pour donner naissance à une nouvelle blockchain, appelée Ethereum, autorisant un grand nombre d'applications qui ne se limitent pas seulement sur le transfert monétaire. De ce fait, elle ouvre de nombreuses perspectives dans le domaine décentralisé.

En effet, dans la blockchain Bitcoin, la base de données distribuée est conçue comme un tableau des soldes des comptes considéré comme un grand livre et les transactions sont des transferts de devise facilitant la gestion financière sans aucune confiance entre les individus. Cependant, au moment où les développeurs et les technologues donnaient une importance à bitcoin, de nouveaux projets ont commencé à utiliser le réseau de Bitcoins pour d'autres fins que le transfert de valeur. Si un bon nombre d'entre eux se sont penchés vers les «*alt-coins*» (des blockchains distinctes avec des crypto-monnaies propres, qui améliorent le protocole Bitcoin d'origine pour ajouter de nouvelles fonctionnalités ou capacités), l'inventeur d'Ethereum, Vitalik Buterin est motivé par le souci d'offrir une blockchain reprogrammable pour effectuer des calculs arbitrairement complexes qui puisse englober de nombreux autres projets. Ainsi il publie le livre blanc d'Ethereum [37] en novembre 2013, dans lequel il décrit en détail la conception technique et la raison d'être du protocole Ethereum mais aussi l'architecture des contrats intelligents. Selon Buterin, Ethereum fusionne et améliore les concepts de scripts (comme dans bitcoin) et d'«*alt-coins*» afin de permettre aux développeurs de créer des applications basées sur un consensus arbitraire offrant l'évolutivité, la standardisation, la facilité de développement et l'interopérabilité offerte par ces différents paradigmes. Et pour ce faire, **Ethereum intègre l'ultime couche fondamentale abstraite: une Blockchain avec un langage de programmation complet, permettant à chacun d'écrire des contrats intelligents et des applications décentralisées dans lesquelles ils peuvent créer leurs propres règles de propriété, leurs propres formats de fonctions de transition d'état.** Plus tard en janvier 2014, Etherik a officiellement annoncé Ethereum lors d'une conférence sur le bitcoin à Miami, en Floride, aux États-Unis. Par ailleurs, Vitalik a également commencé à travailler avec le Dr Gavin Wood qui publia, en avril 2014, le livre jaune d'Ethereum [38], le livre de spécification technique de la machine virtuelle Ethereum (EVM). Les détails de ces spécifications laissent voir plusieurs implémentations du client Ethereum, dans environ sept langages de programmation (C ++, Go, Python, Java, JavaScript, Haskell, Rust).

En s'inspirant de Bitcoin, Ethereum fonctionne sur un réseau décentralisé peer-to-peer autorisant des communications basées sur le consensus tout éliminant les tiers de confiance.

Mais quelques éléments techniques les différencient notamment les notions de compte, de message, un nouveau cadre d'exécution basé sur les contrats intelligents plutôt que sur des scripts et un modèle d'incitation minière particulière. Ainsi, nous allons voir en détails ces différents concepts dans ce chapitre.

3.2. Les comptes, messages et transactions

À la différence de Bitcoin, Ethereum est comme une machine à état où une nouvelle transaction permet de passer d'un état à un autre. L'état global d'Ethereum est composé de nombreux comptes capables d'interagir les uns avec les autres via des transactions ou des messages.

3.2.1. Les comptes

Ethereum introduit formellement le concept de **compte** dans le cadre de son protocole. Le compte est l'initiateur et la cible d'une transaction. Une transaction met directement à jour les soldes des comptes, par opposition au maintien de l'état, comme dans les UTXO en bitcoins et permet la transmission de valeur, de messages et de données entre les comptes pouvant entraîner des transitions d'état. Chaque compte est associé à un état et à une adresse de 20 octets (c'est-à-dire 160 bits) qui permet de l'identifier. Il existe deux types de comptes :

- **Les comptes de propriété externe ou EOA (« Externally Owned Accounts »)** qui sont contrôlés par des clés privées et ne sont associés à aucun code. Un compte externe est nécessaire pour participer au réseau Ethereum. Il interagit avec la blockchain en créant et en signant une transaction à l'aide de sa clé privée. Une transaction entre deux comptes externes est simplement un transfert de valeur. Mais une transaction entre un compte externe et un compte de contrat active le code du compte de contrat, lui permettant d'effectuer diverses actions (par exemple, transférer des jetons, écrire dans la mémoire interne, créer de nouveaux jetons, effectuer des calculs, créer de nouveaux contrats, etc.).
- **Les comptes de contrat** ou **CA (« Contract Account »)**, qui sont contrôlés par leur code de contrat et sont associés à un code. Ils représentent un contrat intelligent (à voir dans la section ci-dessous) et ne peuvent être activés que par un EOA. Les comptes de contrat ne peuvent pas initier de nouvelles transactions par eux-mêmes. Par contre, ils peuvent uniquement déclencher des transactions pour répondre à d'autres transactions qu'ils ont reçues (d'un compte appartenant externe ou d'un autre compte de contrat).

3.2.1.1. État du compte

L'état d'un compte Ethereum est composé (quelques soit son type) de quatre champs [38] (voir Figure 11):

- Le **nonce** : Si le compte est un compte externe, le nonce représente le nombre de transactions envoyées à partir de l'adresse du compte. S'il s'agit d'un compte de contrat, le nonce est le nombre de contrats créés par le compte.
- Le **solde actuel** du compte exprimé en Wei (**1 Ether = 10^{18} Wei**[7]),
- Le **storageRoot** : c'est le hachage de la racine de l'arbre de Merkle. Cette racine est obtenue à partir du hachage du contenu de stockage du compte. StorageRoot est vide par défaut.
- Et enfin le **codeHash** qui représente le hachage du code du compte dans la machine virtuelle d'Ethereum (EVM). Pour un compte de contrat, le code du contrat est haché et stocké sous le nom codeHash et pour un compte externe, le codeHash est le hachage de la chaîne vide.

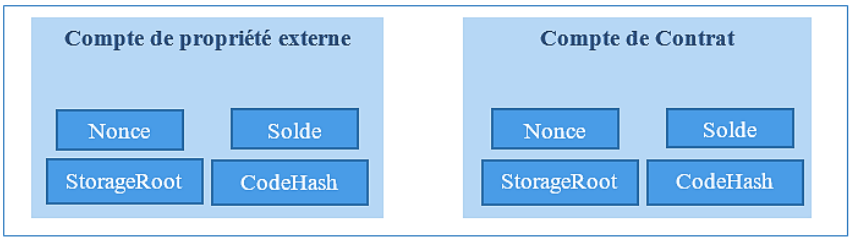


Figure 11: L'état d'un compte dans Ethereum

3.2.1.2. L'état global

L'état global de la blockchain Ethereum est considéré comme un mappage entre les adresses de compte et les états de compte. Ce mappage est stocké dans une structure de données appelée **arbre Merkle.** Cet arbre doit avoir une clé pour chaque valeur stockée à l'intérieur. À partir du nœud racine de l'arbre, la clé doit vous indiquer quel nœud enfant suivre pour obtenir la valeur correspondante, stockée dans les nœuds terminaux. Dans le cas d'Ethereum, le mappage clé / valeur de l'arbre d'état se situe entre les adresses et leurs comptes associés, y compris les éléments solde, nonce, codeHash et storageRoot pour chaque compte (où storageRoot est lui-même la racine d'un autre arbre).

Comme dans Bitcoin, la Blockchain Ethereum est gérée par un groupe de nœuds. De manière générale, il existe deux types de nœuds: les nœuds complets (appelés aussi mineurs) et les nœuds légers (nœuds simples). **Un nœud complet synchronise la blockchain en**

[7] L'Ether est la principale crypto-monnaie interne d'Ethereum et qui sert à payer les frais de transaction

téléchargeant la chaîne complète, du bloc de genèse au bloc actuel, en exécutant toutes les transactions qui y sont contenues. Généralement, les mineurs stockent toute l'archive complète, car ils sont tenus de le faire pour le processus d'extraction. Un nœud complet peut ne pas exécuter chaque transaction. Mais à moins qu'un nœud n'ait besoin d'exécuter chaque transaction ou d'interroger facilement des données historiques, il n'est vraiment pas nécessaire de stocker la chaîne entière. C'est là que le concept de nœud léger entre en jeu. **Au lieu de télécharger et de stocker la chaîne complète et d'exécuter toutes les transactions, les nœuds légers ne téléchargent que les en-têtes de la chaine, depuis le bloc de genèse jusqu'à l'état actuel, sans exécuter aucune transaction ou opération.** Ces nœuds ayant accès aux en-têtes de bloc, ils peuvent toujours générer et obtenir facilement des réponses vérifiables concernant les transactions, les événements, les soldes, etc.

Ceci est possible car dans l'arbre de Merkle les données se propagent vers le haut. En effet, si un utilisateur malveillant tente de permuter une fausse transaction au bas de l'arbre, cette modification entraînera une modification du hachage du nœud qui est au-dessus, ce qui modifiera le hachage du nœud au-dessus de ce dernier, et ainsi de suite, jusqu'à ce qu'il change finalement la racine de l'arbre. **Par conséquent, l'utilisation d'un arbre Merkle présente l'avantage que le nœud racine de cette structure est, de façon cryptographique, dépendant des données stockées dans l'arbre. Le hachage du nœud racine peut donc être utilisé comme une identité sécurisée pour ces données.**

Les comptes régissent l'état de la Blockchain Ethereum. Ils se communiquent via des transactions. La nature de l'initiateur de la transaction impacte sur le type de transaction.

3.2.2. Message et Transaction

L'intervention d'une transaction permet de changer l'état global de la blockchain Ethereum. En ce sens, une transaction **est une instruction signée avec des fonctions cryptographiques qui est générée par un compte externe, sérialisée, puis soumise à la blockchain.** Il existe deux types de transactions: les **appels de message** et **les créations de contrat** (c'est-à-dire les transactions qui créent de nouveaux contrats Ethereum). Toutes les transactions contiennent les composants suivants, quel que soit leur type (Figure 12):

- **Le nonce** : nombre de transactions envoyées par l'expéditeur.
- **Le prix du « gas » (gasPrice)** : Le nombre de Wei que l'expéditeur est prêt à payer par unité de « gas » nécessaire pour exécuter la transaction.

- La **limite de « gas » (gasLimit)** : quantité maximale de « gas » que l'expéditeur est prêt à payer pour l'exécution de cette transaction. Ce montant est défini et payé d'emblée, avant tout calcul.
- **L'adresse du destinataire**. Dans une transaction de création de contrat, l'adresse du compte du contrat n'existe pas encore et une valeur vide est donc utilisée.
- **Un champ valeur (Value)**: le montant (en Wei) à transférer de l'expéditeur au destinataire. Dans une transaction de création de contrat, cette valeur sert de solde de départ dans le compte de contrat nouvellement créé.
- **v, r, s** : trois variables utilisées pour générer la signature identifiant l'expéditeur de la transaction.
- Un champ **init** (uniquement pour les transactions créant un contrat) qui est un fragment de code utilisé pour initialiser le nouveau compte de contrat. Il est exécuté une seule fois, puis ignoré. Lorsque « **init** » est exécuté pour la première fois, il renvoie le corps du code de compte, qui est la partie de code associée en permanence au compte du contrat.
- **Les données** (champ facultatif qui existe uniquement pour les appels de message): ce sont les données d'entrée (c.-à-d. les paramètres) de l'appel de message. Par exemple, si un contrat intelligent sert de service d'enregistrement d'un élève, un appel à ce contrat peut s'attendre à des champs de saisie tels que le nom, le prénom etc.

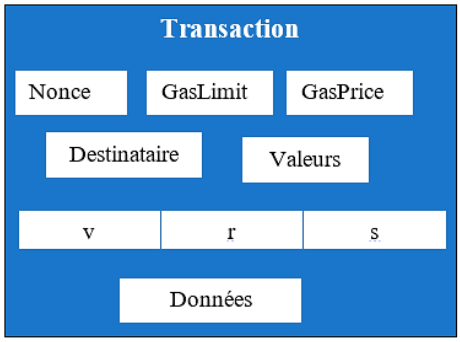


Figure 12: Composants d'une transaction Ethereum

Généralement une transaction provient d'un compte externe du réseau. Mais cela ne signifie pas que les contrats ne peuvent pas communiquer entre eux. **Un contrat envoie des «messages» (ou transactions internes) à d'autres contrats.** Ainsi un message peut être considéré comme une transaction, à la différence près qu'il n'est pas généré par un compte appartenant à un tiers. Au lieu de cela, les messages sont générés par un contrat. Ce sont des objets virtuels qui, contrairement aux transactions, ne sont pas sérialisés et n'existent que dans l'environnement d'exécution Ethereum. **Lorsqu'un contrat envoie une transaction interne à un autre contrat, le code associé au compte de contrat du destinataire est exécuté.** Précisons donc que les transactions internes ne contiennent pas de **gasLimit.** En effet, la limite

de « gas » est déterminée par le créateur externe de la transaction d'origine (c'est-à-dire un compte externe). La limite de « gas » de l'ensemble des comptes externes doit être suffisamment élevée pour exécuter la transaction.

3.2.3. Exécution d'une transaction

L'exécution d'une transaction dans Ethereum passe par un processus très complexe avant d'inclure la transaction dans la chaine. Ainsi un certain nombre d'exigences doit être fourni pour qu'une transaction soit exécutée. Ces exigences sont, entre autre :

- La transaction doit avoir un préfixe de longueur récursive (**PLR** ou **RLP** - Recursive Length Prefix – en anglais) correctement formaté. Un PLR est un format de données utilisé pour coder des tableaux imbriqués de données binaires. Il est utilisé dans Ethereum pour sérialiser des objets.
- La transaction doit avoir une signature valide
- Le « **nonce** » de la transaction doit lui aussi être valide. Pour rappel le nonce d'un compte est le nombre de transactions envoyées à partir de ce compte. Le nonce d'une transaction est valide s'il est égal au nonce du compte de l'expéditeur.
- La limite de « gas » de la transaction doit être égale ou supérieure au « **gas** » **intrinsèque** utilisé par la transaction. Le gaz intrinsèque se calcule comme suit:

Gas intrinsèque = X + 4 Y + 68 Z + U

Avec :

X : le prix de « gas » prédéfini = **21000**

Y : le nombre d'octets de données ou code égal à zéro

Z : le nombre d'octets de données ou code différent de zéro

U : frais supplémentaire { **32000** USD si c'est une transaction de création de contrat ; **0** sinon

- Le solde du compte de l'expéditeur doit comporter suffisamment d'Ether pour couvrir les **coûts de gas initiaux** que l'expéditeur doit payer. Le calcul du coût initial du gas est simple: premièrement, la limite de gas **(GasLimit)** de la transaction est multipliée par le prix du gas **(GasPrice)** de la transaction afin de déterminer le coût maximal du gas. Ensuite, ce coût maximum est ajouté à la valeur totale transférée de l'expéditeur au destinataire.

Une fois toutes ces conditions vérifiées et validées, le cout initial d'exécution est déduit du solde de l'expéditeur et son nonce est incrémenté de 1 pour prendre en compte la transaction en cours. Ainsi le « **gas restant** » peut être obtenu en diminuant le « **gas** » intrinsèque du **GasLimit** total de la transaction. C'est là que la transaction commence à s'exécuter.

D'une part, Ethereum assure le suivi du sous-état tout au long de l'exécution de la transaction. Ce sous-état est caractérisé par :

- ✓ **Un ensemble d'autodestruction** : l'ensemble des comptes (le cas échéant) qui seront supprimés une fois la transaction terminée.
- ✓ **Une série de journaux** : les points de contrôle archivés de l'exécution du code dans de la machine virtuelle.
- ✓ **Un Solde de remboursement** : le montant à rembourser sur le compte de l'expéditeur après la transaction. En effet, le stockage dans Ethereum est coûteux et donc son nettoyage est rémunéré (l'expéditeur est remboursé pour le nettoyage du stockage). Ainsi un compteur de remboursement est créé, initialisé à zéro et incrémenté à chaque fois que le contrat supprime quelque chose stocké.

D'autre part, les différentes instructions de la transaction sont traitées. Une fois que toutes les étapes requises par la transaction ont été traitées et en supposant qu'il n'y ait pas d'état invalide, l'état est finalisé en déterminant la quantité de « **gas »** non utilisé à restituer à l'expéditeur. En plus du gaz non utilisé, l'expéditeur se voit également rembourser une partie de l'allocation du «solde de remboursement». Après avoir effectué le remboursement :

- L'Ether pour le **gas** est donné au mineur,
- Le **gas** utilisé par la transaction est ajouté au compteur de **gas** du bloc (qui garde une trace du **gas** total utilisé par toutes les transactions dans le bloc et est utile lors de la validation d'un bloc),
- Et tous les comptes de l'ensemble d'autodestruction (le cas échéant) sont supprimés.

Voilà les différentes étapes de l'exécution des transactions. Cependant ce processus diffère selon le type de transaction (les transactions créant des contrats et les appels de message).

3.2.3.1. Exécution d'une transaction de création de contrat

La transaction créatrice de contrat permet de créer un nouveau compte de contrat. Pour ce faire, l'adresse du compte est déclarée à l'aide d'une formule spéciale. L'initialisation du compte est effectuée en suivant ces étapes:

- ✓ Mettre le nonce à zéro,

- ✓ Si l'expéditeur a envoyé une certaine quantité d'Ether en tant que ***valeur*** avec la transaction, définir le solde du nouveau compte avec cette valeur,
- ✓ Déduire la valeur ajoutée au solde de ce nouveau compte du solde de l'expéditeur,
- ✓ Définir le stockage comme vide,
- ✓ Définir le **codeHash** du contrat comme hachage d'une chaîne vide,

Une fois le compte initialisé, le compte est réellement créé en exécutant le **code init** envoyé avec la transaction. Ce qui se passe pendant l'exécution de ce **code init** est varié. Selon le constructeur du contrat, il peut mettre à jour le stockage du compte, créer d'autres comptes de contrat, faire d'autres appels de message, etc.

Lorsque le code pour initialiser un contrat est exécuté, il utilise du **gas**. La transaction n'est pas autorisée à utiliser un **gas** supérieur au **gas** restant. Si tel est le cas, l'exécution se heurte à une exception de manque de **gas** et se termine.

Si la transaction se termine en raison d'une exception de non-consommation, l'état est rétabli au point immédiatement avant la transaction. L'expéditeur n'est *pas* remboursé du **gas** qui a été dépensé avant de s'épuiser. Toutefois, si l'expéditeur a envoyé une valeur d'Ether avec la transaction, la valeur sera remboursée même si la création du contrat échoue.

Si le code d'initialisation s'exécute correctement, un coût final de création de contrat est payé. Il s'agit d'un coût de stockage, proportionnel à la taille du code du contrat créé. S'il ne reste plus assez de **gas** pour payer ce coût final, la transaction déclare à nouveau une exception de manque de **gas** et se termine.

Si tout se passe bien et que nous allons jusque-là sans exception, tout gaz non utilisé restant est restitué à l'expéditeur d'origine de la transaction et l'état modifié est maintenant autorisé à persister.

3.2.3.2. Exécution d'une transaction d'appel de message

L'exécution d'un appel de message est similaire à celle d'une création de contrat, à quelques différences près. Cette exécution n'inclut aucun code d'initialisation, car aucun nouveau compte n'est créé. Cependant, il peut contenir des données d'entrée, si ces données ont été fournies par l'expéditeur de la transaction. Une fois exécutés, les appels de message ont également un composant supplémentaire contenant les données de sortie, qui est utilisé si une exécution ultérieure a besoin de ces données.

Comme c'est le cas avec la création de contrat, si l'exécution d'un appel de message se termine parce qu'elle manque de « gas » ou que la transaction est invalide (par exemple, débordement de pile, destination de saut invalide ou instruction non valide), les « gas » utilisés

ne sont restitué à l'expéditeur. Au lieu de cela, tout le gaz non utilisé restant est consommé et l'état est réinitialisé au point immédiatement avant le transfert de solde.

Les frais de « gas » mis en jeux pour l'exécution des transactions est une motivation pour les mineurs afin qu'ils maintiennent la cohérence de l'état global d'Ethereum.

3.3. Le modèle d'incitation minière.

La blockchain Ethereum est à bien des égards similaire à la blockchain Bitcoin, bien qu'elle présente quelques différences. La principale différence entre Ethereum et Bitcoin en ce qui concerne l'architecture blockchain est que, contrairement à Bitcoin, les blocs Ethereum contiennent une copie de la liste de transactions et de l'état le plus récent. En plus de cela, le bloc contient deux autres valeurs, le numéro de bloc et la difficulté. L'algorithme qui donne un sens au concept de difficulté d'un bloc s'appelle Preuve de travail (PoW – Prof Of Work). L'algorithme de preuve de travail utilisé s'appelle Ethash [39] et consiste à rechercher une entrée **nonce** dans l'algorithme afin que le résultat soit inférieur à un certain seuil de difficulté. Cet algorithme permet, en quelque sorte, de calculer le mixHash et le nonce de bloc. En effet, ces deux composants qu'on retrouve dans un bloc Ethereum sont techniquement liés :

- **Le mixHash** d'un bloc est un hash qui, combiné avec le nonce, prouve que ce bloc a effectué suffisamment de calculs
- **Le nonce** d'un bloc est un hash qui, combiné avec le mixHash, prouve que ce bloc a effectué suffisamment de calculs.

Le calcul de ces deux composant par l'algorithme de PoW est assez complexe mais peux se résumer, à un niveau élevé, de la manière suivante :

Une «**graine**» est calculée pour chaque bloc. Cette graine est différente pour chaque «**époque**», où chaque époque compte **30 000 blocs**. Pour la première époque, la graine est le hachage d'une série de **32 octets de zéros**. Pour chaque époque ultérieure, il s'agit du hash du hash précédent. En utilisant cette graine, un nœud peut calculer un «**cache**» pseudo-aléatoire. Ce cache est utilisé par les nœuds legers pour la vérification efficace d'une transaction sans la charge de stockage de l'ensemble du jeu de données de la blockchain. Ainsi, un nœud léger peut vérifier la validité d'une transaction basée uniquement sur ce cache, car celui-ci peut régénérer le bloc spécifique à vérifier.

Les mineurs peuvent ensuite prendre des tranches aléatoires de l'ensemble de données et les soumettre à une fonction mathématique pour les regrouper en un « **mixHash ».** Un mineur va générer à plusieurs reprises un **mixHash** jusqu'à ce que la sortie soit en dessous

du **nonce** cible souhaité. Lorsque la sortie répond à cette exigence, ce nonce est considéré comme valide et le bloc peut être ajouté à la chaîne.

La preuve de travail que les mineurs effectuent garantie la sécurité du protocole d'Ethereum mais aussi permet au mineurs de s'enrichir.

3.3.1. L'exploitation minière et sécurité

Globalement, l'objectif de la PoW est de prouver, de manière cryptographique et sécurisée, qu'une quantité de calcul particulière a été utilisée pour générer une sortie (c'est-à-dire le **nonce**). En effet, il n'existe pas de meilleur moyen de rechercher un nonce inférieur au seuil requis, à part d'énumérer toutes les possibilités. Les sorties de l'application répétée de la fonction de hachage ont une distribution uniforme, et nous pouvons donc être assurés que, en moyenne, **le temps nécessaire pour trouver un tel élément dépend du seuil de difficulté.** Plus la difficulté est élevée, plus la résolution du problème est longue. De cette manière, **l'algorithme PoW donne un sens à la notion de difficulté, qui est utilisé pour appliquer la sécurité des blockchains.**

En effet la sécurité de la blockchain repose sur la confiance des utilisateurs (membres du réseau). S'il existait plus d'une chaîne, les utilisateurs perdraient la confiance, car ils ne pourraient pas déterminer de manière raisonnable quelle chaîne était la chaîne «valide». En ce sens, pour qu'un groupe d'utilisateurs accepte l'état sous-jacent qui est stocké dans une blockchain, il est nécessaire d'avoir une seule chaine canonique en laquelle tout le monde a confiance. **C'est exactement ce que fait l'algorithme PoW: il garantit qu'une chaîne de blocs particulière restera canonique dans le futur, ce qui rend extrêmement difficile pour un attaquant de créer de nouveaux blocs qui écrasent une partie de l'historique (par exemple, en effaçant des transactions ou en créant de fausses transactions) ou maintenir un « fork ».** Pour que leur bloc soit d'abord validé, un attaquant doit résoudre systématiquement le problème plus rapidement que quiconque sur le réseau, de sorte que le réseau estime que sa chaîne est la chaîne la plus lourde. Cela serait impossible à moins que l'attaquant dispose de plus de la moitié de la puissance d'extraction du réseau, un scénario connu sous le nom d'attaque à 51%.

3.3.2. L'exploitation minière et richesse

En plus de fournir une blockchain sécurisée, la preuve de travail est également un moyen de distribuer de la richesse à ceux qui utilisent leurs calculs pour fournir cette sécurité. Rappelez-vous qu'un mineur reçoit une récompense pour l'extraction d'un bloc, notamment:

- une *prime* de *bloc statique* de **5 Ether** pour le bloc «gagnant» (qui est réduit à **3Ether** avec les mis à jour de Byzantium de 2017),
- un coût du gaz dépensé dans le bloc par les transactions incluses dans le bloc,
- une récompense supplémentaire pour l'inclusion de membres dans le bloc.

Afin de garantir la pérennité de l'utilisation du mécanisme de consensus de la preuve de travail pour la sécurité et la répartition des richesses, Ethereum s'efforce d'instiller ces deux propriétés:

- Rendez-le accessible au plus grand nombre de personnes possible. En d'autres termes, les utilisateurs ne devraient pas avoir besoin de matériel spécialisé ou inhabituel pour exécuter l'algorithme. L'objectif est de rendre le modèle de répartition de la richesse le plus ouvert possible, afin que tout le monde puisse fournir n'importe quelle quantité de puissance de calcul en retour d'Ether.
- Réduisez la possibilité pour un seul nœud (ou petit groupe) de générer un profit disproportionné. Tout nœud pouvant générer un profit disproportionné signifie que le nœud a une grande influence sur la détermination de la blockchain canonique. C'est gênant car cela réduit la sécurité du réseau.

Dans le réseau de la Blockchain Bitcoin, l'un des problèmes liés aux deux propriétés ci-dessus est que l'algorithme PoW est une fonction de hachage SHA256. La faiblesse de ce type de fonction est qu'il peut être résolu beaucoup plus efficacement à l'aide de matériel spécialisé, également appelé ASIC. Afin d'atténuer ce problème, Ethereum a choisi de rendre son algorithme **Ethash** séquentiellement difficile en mémoire. Cela signifie que l'algorithme est conçu de sorte que le calcul du nonce nécessite beaucoup de mémoire et de bande passante. **Les besoins importants en mémoire font qu'un ordinateur a du mal à utiliser sa mémoire en parallèle pour détecter plusieurs nonces simultanément, et les exigences élevées en bande passante rendent difficile, même pour un ordinateur super rapide, de détecter plusieurs nonce simultanément.** Cela réduit les risques de centralisation et crée des conditions de concurrence plus équitables pour les nœuds effectuant la vérification.

Basé sur la blockchain Bitcoin, Ethereum ressemble à cette dernière mais à ses propres spécifiés. Ethereum permet des transactions pouvant mener à des opérations plus sophistiquées. Par exemple, une transaction peut nécessiter un transfert conditionnel, une évaluation, une signature etc. Ceci est possible grâce aux contrats intelligents.

3.4. Les contrats intelligents

Le terme contrat intelligent (ou smarts contracts, en Anglais) a été introduite depuis 1997 par Nick Szabo [40]. Un contrat intelligent est un programme de code identifié par une adresse du réseau Blockchain. Ethereum est l'une des technologies privilégiées pour le développement des contrats intelligents. Les principaux composants des transactions sont basés sur la machine à états et les fonctions. Il s'agit d'une plate-forme de traitement et d'exécution de contrat *Turing-complete* basée sur un grand livre partagé décentralisé Blockchain. La conception et la mise en œuvre de l'Ethereum sont totalement indépendantes de la crypto-monnaie Bitcoin. Chaque transaction a des paramètres d'entrée qui sont requis par une fonction dans le contrat. Lors de l'exécution d'une fonction, l'état des variables d'état est modifié en fonction de la mise en œuvre logique. Le code de contrat intelligent est écrit dans des langages de haut niveau tels que **Solidity** [41], Serpent ou LLL (List Like Language). Le code est compilé en bytecode à l'aide de compilateurs tels que Solidity ou Serpent. Le programmeur peut créer les formats de transaction, les transitions d'état, les fonctions d'événement et les règles de propriété. Le code du logiciel est exécuté sur une machine virtuelle appelée machine virtuelle Ethereum [42].

Les contrats intelligents permettent aux contreparties d'automatiser des tâches de transaction généralement effectuées manuellement et nécessitant l'intervention d'intermédiaires tiers. Une technologie de contrat intelligent peut aboutir à des processus plus rapides, plus précis et plus économiques. Ainsi les contrats intelligents couvrent un grand nombre de domaines d'application contractuels pouvant tirer parti d'une fiabilité accrue, d'un traitement des transactions plus rapide, d'une réduction des coûts et d'un nombre réduit d'étapes de processus manuels via des intermédiaires.

C'est en ce sens que nous l'adoptons pour la sécurisation des livrets scolaires des élèves du Sénégal. Un tel choix est motivé, par ailleurs, par les caractéristiques uniques de la blockchain en tant que facilitateur d'applications plus fiables, inviolables et résistantes aux pannes. En outre, grâce aux contrats intelligents, la blockchain Ethereum incorpore dans son infrastructure de confiance une couche de logique et de calcul. Ce qui lui confère une ouverture vers un nombre potentiel d'applications décentralisées.

3.4.1. Structure d'un contrat intelligent

Pouvant être considéré comme la pièce maîtresse de la poussée de la blockchain Ethereum, un contrat est un ensemble de code (**ses fonctions**) et de données (**son état**) résidant à une

adresse spécifique de la blockchain Ethereum. Cet ensemble est illustré à la Figure 13 et laisse apparaitre clairement la différence entre le message et la transaction.

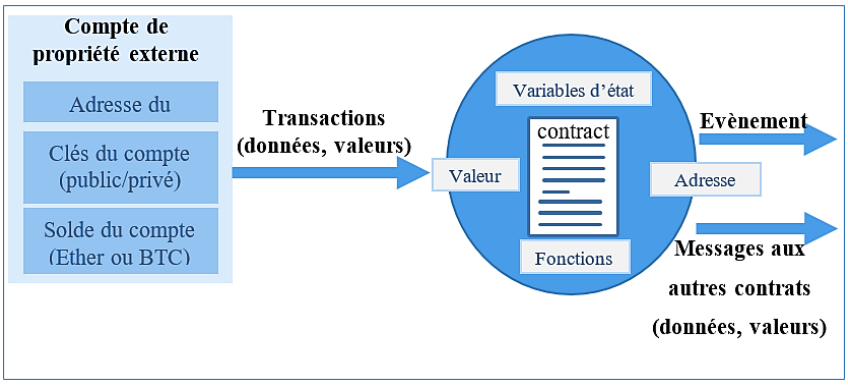


Figure 13: Structure globale d'un contrat intelligent

En effet, les comptes de contrat sont capables de passer des messages entre eux et d'effectuer pratiquement le calcul complet de Turing. Les composants principaux du contrat intelligent sont les variables d'état, les fonctions, les modificateurs et les événements. Les contrats régissent le comportement des comptes au sein de l'état Ethereum.

Solidity est un langage de haut niveau orienté objet pour la mise en œuvre de contrats intelligents. C'est un langage typé de manière statique qui prend en charge l'héritage, les bibliothèques et les types complexes définis par l'utilisateur, entre autres fonctionnalités. Structurellement, un contrat intelligent ressemble à une **définition de classe dans la conception orientée objet** [41]. Il contient des données, des fonctions ou des méthodes avec des modificateurs publics ou privés, ainsi que des fonctions getter et setter. Examinons un simple contrat intelligent Solidity dans la figure qui suit pour en comprendre la structure.

```
1  pragma solidity ^0.4.0;
2
3  contract Livret {
4
5      uint moyenne;
6      string prenomEleve;
7      string nomEleve;
8
9      function setEleve(string prenom,string nom)
10     public {
11         nomEleve = nom;
12         prenomEleve = prenom;
13     }
14
15     function getMoyenne (uint n)
16     public returns (uint)  {
17         return moyenne;
18     }
19
20 }
```

Figure 14 : Exemple d'un contrat intelligent

La première ligne avec **pragma** indique la version du langage de solidity 0.4.0 ou toute autre version plus récente (jusqu'à la version 0.6.0). Cela garantit que le contrat n'est pas compilable avec une nouvelle version du compilateur, où il pourrait se comporter différemment. Le pragma est l'instruction qui dit aux compilateurs la façon de traiter le code source du contrat.

Le nom du contrat **Livret** est précédé du mot clé **contract**. Ce contrat particulier concerne le stockage d'un élève. Les données pour le prénom et le nom sont définies avec le type **string** et pour la moyenne avec le type **uint**. Deux fonctions sont définies pour renseigner le prénom et le nom de l'élève **setEleve** et pour lire sa moyenne **getMoyenne**.

3.4.2. Exécution et déploiement d'un contrat intelligent

L'exécution d'un contrat intelligent est initiée par un message intégré à la transaction par exemple une demande de transfert de devise numérique, une simple addition et soustraction. Chaque nœud du réseau Ethereum doit pouvoir exécuter le code indépendamment du type de matériel sous-jacent ou du système d'exploitation sous-jacent lancé dans **la machine virtuel d'ethereum** (**EVM**). Un contrat intelligent, écrit dans un langage de programmation de haut niveau, est traduit en bytecode, puis déployé sur l'EVM. Chaque nœud hébergera les mêmes codes de contrat intelligents que sur l'EVM. Un nœud Ethereum est un système informatique représentant une entité commerciale ou un participant individuel. Un nœud complet Ethereum héberge le logiciel nécessaire à l'initiation de transaction, à la validation, à l'extraction, à la création de blocs et à l'exécution de contrat intelligent. La figure suivante illustre le déploiement d'un contrat intelligent et l'invocation d'un contrat intelligent.

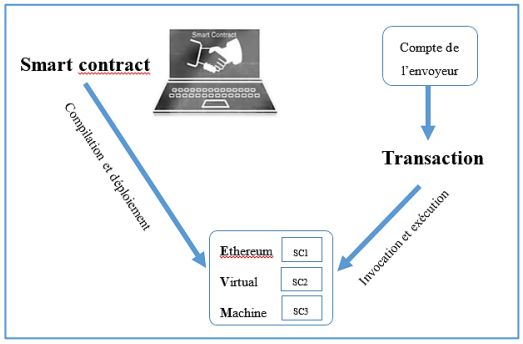


Figure 15: Déploiement d'une transaction Ethereum

Le contrat intelligent est conçu, développé, compilé et déployé sur l'EVM ; il peut avoir plus d'un contrat intelligent dans un EVM. Lorsque l'adresse cible dans une transaction est un contrat intelligent, le code d'exécution correspondant au contrat intelligent est activé et exécuté sur le EVM. Les données nécessaires à cette exécution sont extraites du champ de données utiles de la transaction. L'état actuel du contrat intelligent correspond aux valeurs des variables qui y sont définies. L'état du contrat intelligent peut être mis à jour par cette exécution. Une blockchain conserve à la fois le hachage d'état et le hachage de réception. Toutes les transactions générées sont validées. La validation de la transaction implique la vérification de la validité de la combinaison d l'horodatage, du nonce et de la disponibilité de frais d'exécution suffisants. Les nœuds mineurs du réseau reçoivent, vérifient, rassemblent et

exécutent des transactions. Le code du contrat intelligent en cours d'exécution est exécuté par tous les mineurs.

3.5. Les Dapps : applications décentralisées

L'émergence de l'**Ethereum** a changé la façon dont la blockchain est perçue. Ethereum est la blockchain faite pour le développement des **applications décentralisées DApps** (Decentralized Applications). Une DApp est un service qui permet une interaction directe entre les utilisateurs finaux et les fournisseurs (par exemple, la connexion des acheteurs et des vendeurs sur certains marchés). Les applications décentralisées d'Ethereum interagissent généralement avec les utilisateurs via une application Web (HTML, Javascript, CSS) utilisant une API Javascript, Web3 afin de communiquer avec la blockchain. Le front-end est ensuite déployé publiquement et l'application est accessible à tous les utilisateurs d'Ethereum. Par contre le back-end (les Smart Contracts) est déployé sur la Blockchain.

Une application décentralisée (DApp) est une application qui utilise des contrats intelligents fournissant une interface utilisateur conviviale pour les contrats intelligents. Un exemple typique de DApp est une application de crypto-monnaie qui s'exécute sur un réseau blockchain. La structure d'une application décentralisée est composée d'une interface frontale (navigateur Web, HTML, CSS) et d'une interface back-end (Web3 JavaScript). Comme décrit à la Figure 19, l'application DApp interagit avec le nœud Ethereum (EVM) à l'aide de JSON RPC. JSON RPC est un protocole d'appel de procédure distante sans état et léger utilisé par les clients Ethereum pour interagir avec un nœud Ethereum.

Une DApp n'a pas besoin d'autorité centrale pour fonctionner : elle rend ainsi possible des interactions directes, pair-à-pair, entre utilisateurs, via des contrats intelligent. **Elles montrent le potentiel des applications sur la blockchain d'Ethereum.** La plupart des DApps nécessitent l'installation d'un client Ethereum ou l'utilisation de MetaMask [44], un portefeuille d'Ether léger, sous forme d'extension Google Chrome.

En résumé, le projet de création d'Ethereum est inspiré de Bitcoin. Ainsi Ethereum reprend les concepts de base de Bitcoin mais avec un mode de fonctionnement assez diffèrent. Outre les comptes et les messages, Ethereum intègre formellement les contrats intelligents. En ce sens, les contrats intelligents ajoutent une couche de logique et de calcul à l'infrastructure de confiance prise en charge par la blockchain Ethereum. Les contrats intelligents permettent l'exécution de codes et l'amélioration de la capacité de transfert de valeur de base de la blockchain Bitcoin. Grace aux contrats, Ethereum permet l'ouverture d'une large gamme d'applications décentralisées.

Ce qui rend possible la conception et la réalisation d'une application décentralisée des livrets scolaires. Une telle application permettra un stockage sécurisé des livrets, la validation ou la vérification des livrets de même que la décentralisation du système (accessible par tous).

CHAPITRE IV:

RÉALISATION DU SYSTÈME DÉCENTRALISÉ DE SÉCURISATION DES E-LIVRETS SCOLAIRES (SDSEL)

Nous sommes arrivés maintenant à la partie la plus importante qui constitue le cœur du travail. Dans le chapitre précédant nous avons vu quelques éléments techniques du fonctionnement de la Blockchain Ethereum ainsi que ses caractéristiques (même si les notions générales ont été élaborées dans le chapitre II). En effet les contrats ont permis d'implémenter n'importe quel type d'application décentralisée utilisant la plateforme d'Ethereum. Ainsi, nous avons choisi Ethereum pour offrir aux livrets scolaires un système **autonome, sécurisé et robuste** afin de garantir leur **authenticité** et leur **fiabilité**.

L'utilisation d'une blockchain publique servant de stockage, de vérification des livrets prend place. La réalisation d'un système décentralisé répondant au besoin de sécurisation des livrets n'est rien d'autre que l'implémentation des contrats intelligents qui permettent d'importer, dans la Blockchain Ethereum, les données des livrets scolaires stockées dans la base de données du SGLE. En ce sens, il est vu comme un système de sauvegarde et de protecteur des livrets contre les modifications et les pertes d'informations.

Dans ce chapitre, nous allons, en première lieu, montrer les spécifications des besoins fonctionnels, le découpage et l'architecture du système. Deuxièmement nous allons implémenter le système en montrant quelques programmations. En fin nous présenterons les interfaces de l'application.

3.6. Analyse et conception du système

La mise en œuvre de notre application décentralisée suit un enchaînement logique de phases et d'étapes, depuis la spécification des besoins fonctionnels jusqu'à l'implémentation ou la réalisation (écriture et tests des programmes). Cette conception laisse apparaitre les contrats qui entrent en jeu et leurs comportements.

1. Spécification des besoins fonctionnels

Cette étape est marquée par l'identification des acteurs du système et leurs rôles. En outre il y a lieu de faire une conception des contrats décrivant le fonctionnement du système afin de faciliter la réalisation de l'application.

4.1.1.1. Identification des acteurs et leurs rôles

Au cours de la phase d'analyse, un ensemble d'exigences de différentes personnalités interagissant avec le système est requis. Ensuite, une série d'ateliers est développée pour comprendre comment la technologie de la blockchain et les contrats intelligents peuvent apporter des avantages dans la gestion des livrets scolaires et identifier les acteurs, les rôles et les responsabilités. Il existe trois types d'acteurs :

✓ **Les comptes de propriété externe (EOA):** les établissements et l'entreprise YAKAARTIC sont considérés comme des comptes de propriété externes. Ces comptes sont contrôlés par des clés privées (voir Section 3.2.1 du chapitre précédent). Ces acteurs peuvent créer des contrats intelligents, appeler des fonctions de contrat. Les rôles de ces comptes sont illustrés dans le Tableau 4.

✓ **Les comptes des contrats (CA):** les contrats ont chacun un compte caractérisé par une adresse. À chaque fois que le contrat reçoit un message, son code s'exécute, lui permettant de lire et d'écrire dans la mémoire interne et d'envoyer des messages à d'autres contrats ou de créer des contrats en retour. Nous avons deux principaux contrats pour assurer les rôles des comptes externes.

✓ **Les mineurs:** Ils valident les transactions et les blocs. Les transactions sont regroupées dans un bloc et une preuve de travail sera fournie pour ce bloc. Après validation de la transaction dans le bloc, un montant est fourni aux mineurs en guise de récompense. Puisse que nous utilisons la Blochain public Ethereum, **on ne crée pas nos propres comptes de Mineurs: ça sera réservé au fonctionnement d'Ethereum**.

Acteurs	Rôles
Etablissements	- Enregistrer les informations de l'école - Importer les élèves inscrits dans l'année scolaire courante depuis la base de données de l'application de gestion des notes afin de les enregistrer dans la Blockchain - Afficher les livrets des élèves. - Afficher l'historique des élèves
YAKAARTIC	- peut enregistrer un établissement - peut enregistrer la liste des élèves d'une école donnée dans la blockchain - peut afficher les élèves et leurs livrets - peut voir les statistiques par établissement ou globalement

Tableau 4: les comptes externes pour la DAPP des livrets

Une fois les entités identifiées et les comptes établis, la conception des contrats intelligents sera développée. Les principales composantes du contrat intelligent, telles que décrites précédemment, sont les fonctions, les processus, les variables d'état, les événements et les transactions. La conception de notre Dapp fait l'objectif de la conception des contrats intelligents qui décrivent le fonctionnement de l'application.

4.1.1.2. Diagramme de conception des contrats intelligents

Nos deux contrats ont spécifiquement pour objectif de stocker les livrets scolaires des élèves. Chaque élève a un livret qui le suivra tout au long de son cursus. Ainsi le contrat « Eleves » est caractérisé par un **mappage de livrets** et un nombre incrémental à chaque enregistrement d'un nouveau livret. Comme illustré dans la Figure 16, un Livret contient les informations personnelles (nom, prénom, sexe, …) de l'élève et un tableau d'inscriptions. Une Inscription est caractérisée par : l'année scolaire, le niveau, l'établissement (l'adresse du compte de l'établissement), les avis du conseil de classe, du professeur principal de la classe et du chef d'établissement, le tableau des semestres qui est généralement de taille deux. Quant au semestre, il contient le tableau des notes obtenues dans les matières.

Le contrat « Etablissements » permet d'enregistrer les établissements. Il est caractérisé par un **mot clé**, qui sert de mot passe d'authentification des chefs d'établissements, et d'un **tableau d'établissements**. Chaque établissement possède une **adresse de compte** unique (cette adresse est associée à l'inscription de ses élèves) et ses informations d'identification.

Les données stockées pas ces contrats permettent de construire le livret scolaire d'un élève sous forme de fichier PDF imprimable. Ce fichier est utilisé pour vérifier (par comparaison) la conformité d'un livret issu du SGLE.

4.1.1.3. Diagramme d'interaction des contrats intelligents

Les EOAs communiquent avec les contrats en appelant leurs fonctions (voir Figure 17). En ce sens, le contrat « Etablissements » dispose de ces fonctions ayant une visibilité publique (accessible depuis l'extérieur) :

- **addEtablissement()** : cette fonction est appelée lorsqu'on veut ajouter un nouveau établissement. Elle prend en entrée les références du dit établissement et l'adresse du compte. Cette adresse est le plus souvent l'adresse qui envoie la transaction[8]. L'exécution complète de la transaction déclenche alors l'évènement **addEtablissementEvent.** Une telle transaction coute environ **0.004023** ETH[9].
- **getEtablissement()** : permet de retourner un établissement (sous forme json) à partir de son identifiant (ID) ou son matricule. Cette fonction n'est pas couteuse car ne modifie pas l'état de la blockchain.

[8] Cependant si le compte de YAKAARTIC initie la transaction, l'adresse de l'établissement est explicitement renseignée.

[9] Dans une Blockchain Ethereum locale lancée avec **GANACHE** v2.0.0-beta.2. Où le prix de gas = **20 GWei** et gasLimit = **08160 unité de gas.**

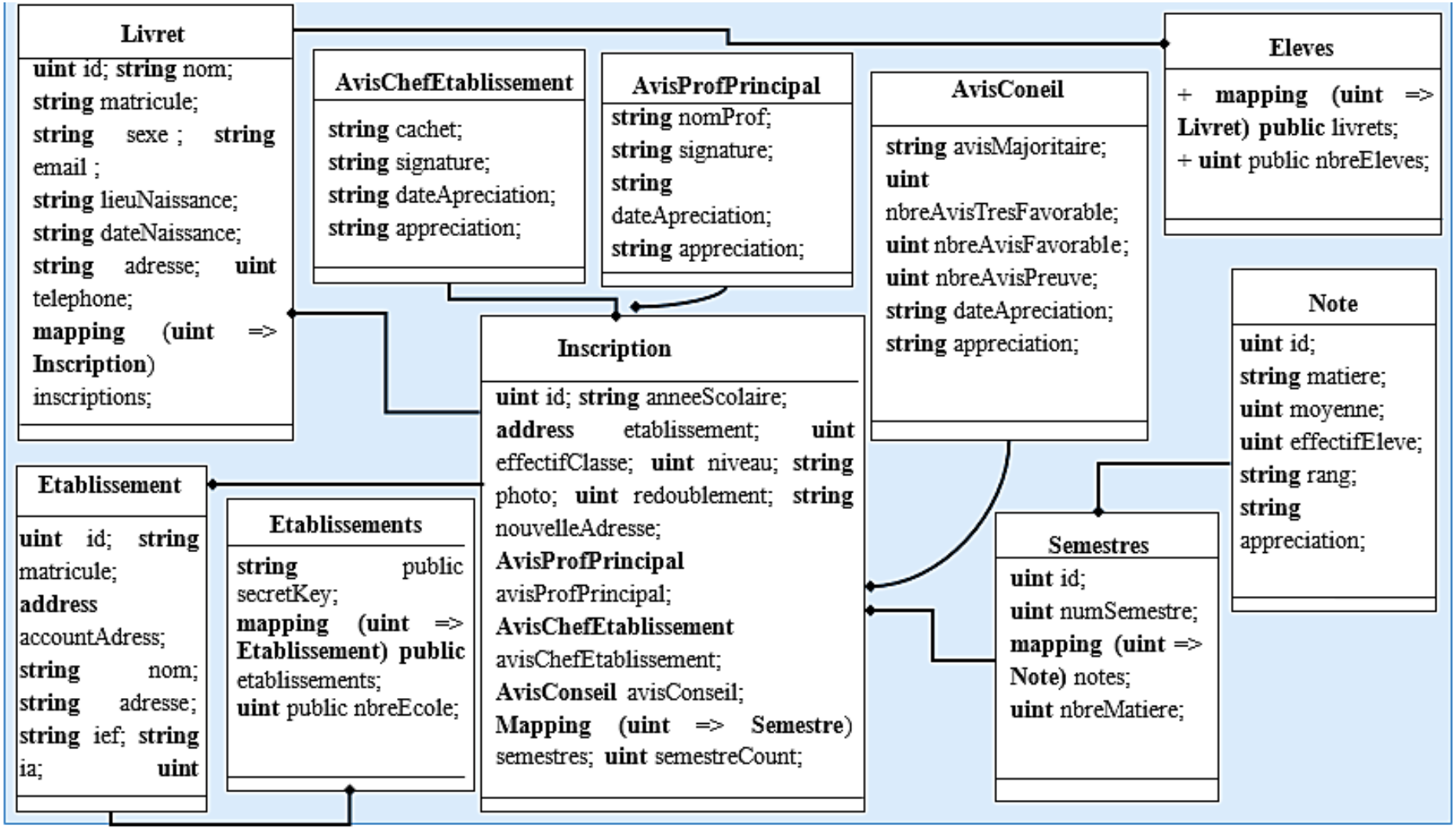


Figure 16: Diagramme de conception des contrats

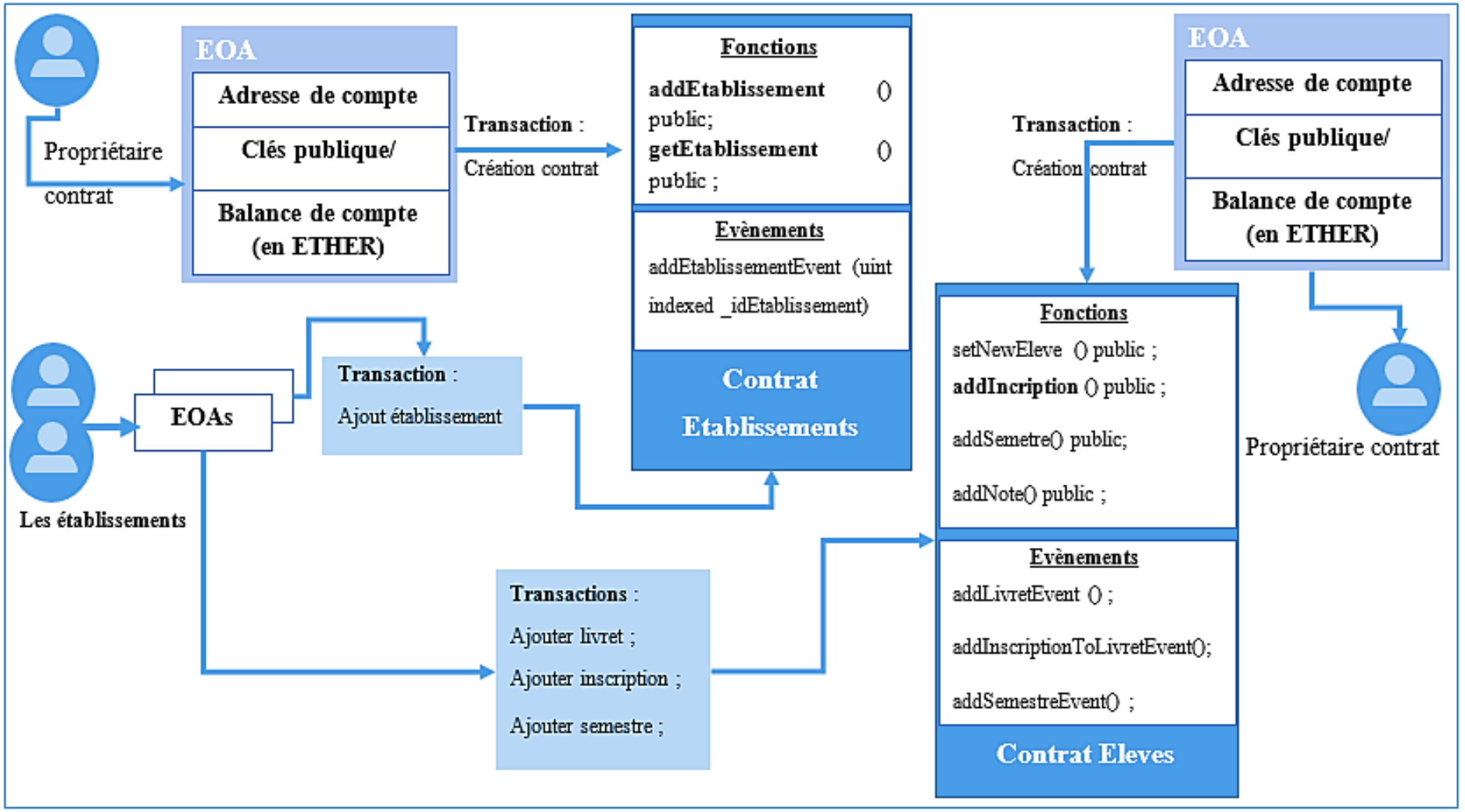


Figure 17: Diagramme d'interaction avec les contrats

Le contrat « Eleves » ouvre publiquement les fonctions suivantes :

- **setNewEleve()** : permet d'enregistrer un nouveau élève qui s'inscrit pour la première fois dans le SGLE. L'exécution complète de la transaction renvoi la position de cet élève dans le tableau. Ce numéro sera utilisé pour compléter son livret. L'évènement **addLivretEvent** découle de cette transaction**.**
- **addInscription()** : permet d'ajouter une nouvelle inscription dans le livret d'un élève. L'exécution complète de la transaction renvoi numéro d'inscription dans le tableau. Ce numéro sera utilisé pour ajouter les deux semestres de l'inscription. L'évènement **addInscriptionToLivretEvent** découle de cette transaction.
- **addSemestre()** : ajoute un semestre à une inscription. Elle prend en entrée le numéro du semestre et un tableau de matières et de notes. Pour enregistrer les notes elle fait appel à la fonction AddNote. Cette fonction est appelée sous forme de transaction qui déclenche l'évènement **AddSemestreEvent**.
- **addNote()** : étant une fonction interne, elle est appelée par d'autre fonction du contrat. Elle permet d'enregistrer les notes obtenues au cours d'un semestre.

4.1.2. Découpage du système

La configuration des nœuds Ethereum constitue l'étape initiale dans la conception des contrats intelligents. Ensuite la définition des services métiers prend place. Enfin, les processus entre les utilisateurs sont décrits. Ceci laisse apparaitre les trois composants (voir Figure 18) essentiels pour le fonctionnement d'une application décentralisée.

Les nœuds ethereum constituent les comptes interagissant avec les contrats. On a quatre types de nœuds :

- ✓ Le compte de contrat
- ✓ Le compte d'établissement ;
- ✓ Le compte de l'entreprise YAKAARTIC,
- ✓ Et le compte de mineur.

Les services et fonctions métiers offerts par le contrat intelligent sont les suivants:

- ✓ Créer des transactions : par exemple l'importation, sous forme de transaction des élèves depuis la base de données du système central des livrets ;
- ✓ Créer des contrats intelligents : le chargement des livrets nécessite l'initialisation du contrat « Livret » ;
- ✓ Envoyer des messages : qui est matérialisé par la communication entre deux contrats ; ça peut être un appel de fonction;

- ✓ Miner de l'Ether : réservé au mineur, cette fonctionnalité permet aux mineurs conquérir la validation des blocs pour recevoir la récompense (en Ether). Ce qui permet au réseau d'exister.

Ces services permettent de déclencher des processus de la plateforme Ethereum. Nous en avons quatre principaux que sont:

- ✓ **Validation**: ce processus permet de valider un bloc.
- ✓ **Découverte du réseau:** ce processus est nécessaire pour qu'un nouveau nœud rejoigne le réseau P2P de la Blockchain.
- ✓ **Création de transaction:** permet aux utilisateurs de créer des transactions et aux contrats intelligents de créer des événements et des messages.
- ✓ **Exploitation minière:** ce processus décrit le processus d'exploitation minière et la diffusion d'un nouveau bloc sur le réseau.

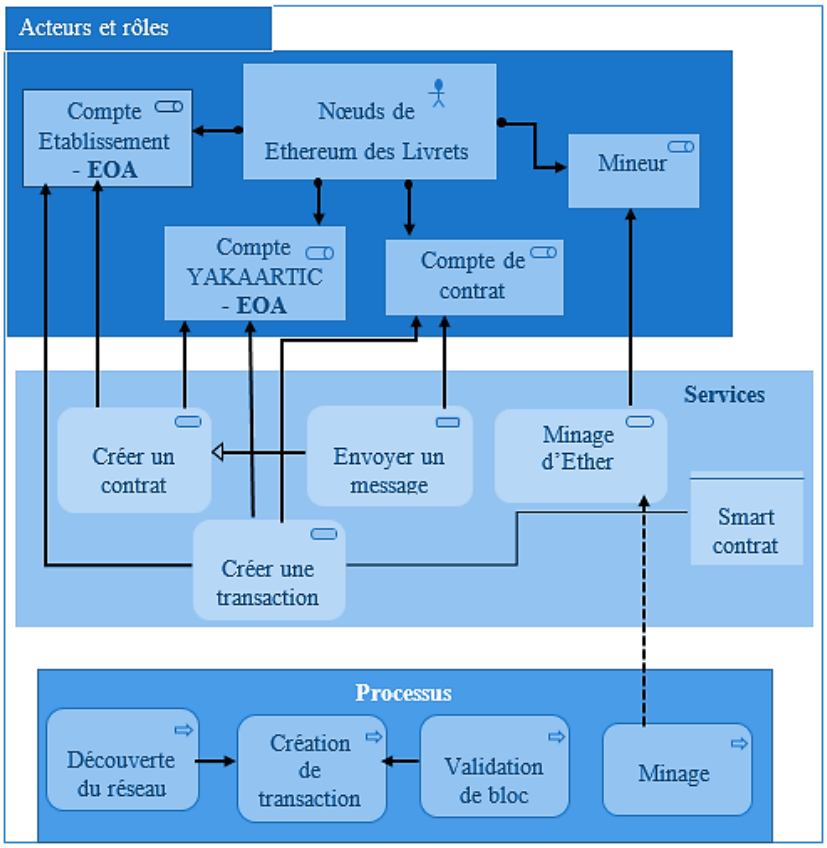


Figure 18: Composants proposés pour le cas d'utilisation du système des livrets

4.1.3. Architecture du système

Les applications décentralisées fonctionnant sur la Blockchain, sont généralement structurées comme l'illustre la Figure 19 ci-dessous. Une application décentralisée (DApp) est composée d'applications front-end et back-end. Les utilisateurs interagissent avec la DAPP à partir la

front-end qui, elle-même communique avec la partie back-end de l'application. Ce dernier entre en contact avec les nœuds d'Ethereum (la machine virtuelle) grâce à l'interface RPC.

Cette architecture montre les composants logiciels de notre système décentralisé pour la sécurisation des E-livrets (SDSEL). Mais le fonctionnement de ce système est étroitement lié au comportement du SGLE (Système de gestion des livrets Electroniques). En effet, pour fournir les entrées des fonctions des contrats, l'application décentralisée dans sa couche métier doit interroger la base de données du système centralisé SGLE. Ainsi nous avons décidé de **l'importation des données soit faite à chaque fin d'année** (Si l'établissement clôture l'année scolaire dans l'application de gestion des livrets électroniques). Une telle décision est prise dans le but d'optimiser le nombre de transactions afin de réduire les coûts. En ce sens, il y a belle et bien un compromis entre la sécurité et cout : « il n'existe pas sécurité gratuit ».

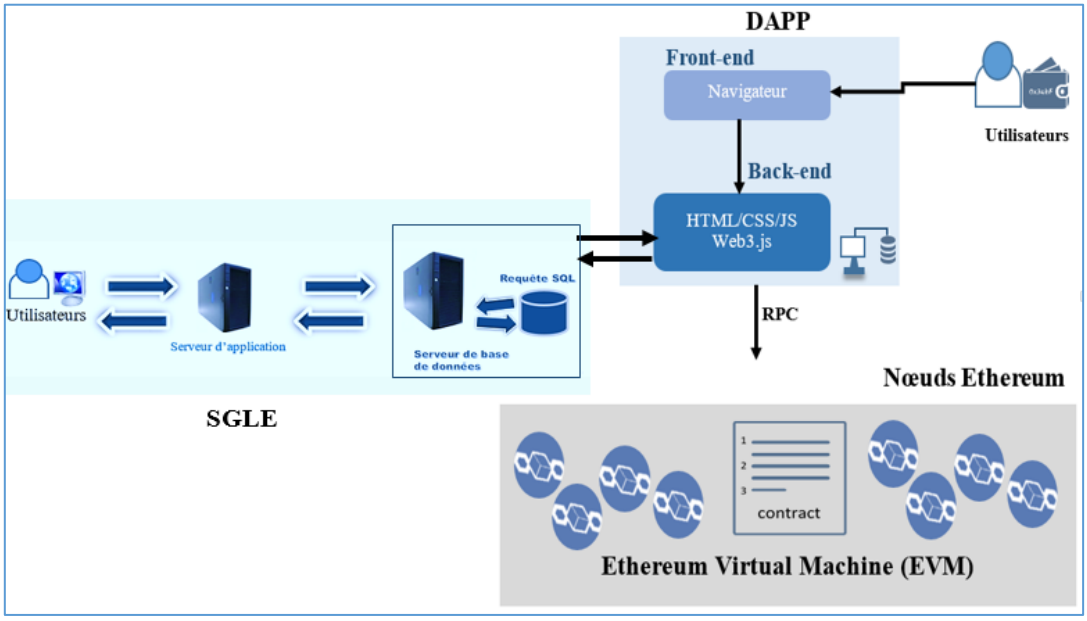


Figure 19: Architecture l'application

3.7. Implémentation du système

Les fonctions et processus qui sont définis lors de la phase de conception sont traduits en programme de code. L'implémentation du système est en ce sens le codage des spécifications techniques de la conception. Ainsi nous présentons dans cette partie les codes clés pour fonctionnement du système ; les détails serons données annexes. Mais avant cela, voyons les technologies et langages de programmations utilisés pour le codage.

4.1.4. Outils et technologies de développement utilisés

Pour parvenir à terme du développement de notre DApp des livrets, des recherches complémentaires sont indispensable pour comprendre l'API JavaScript **Web3**, l'API **JSON RPC** et le langage de programmation **Solidity**.

En effet l'API Web3Js, un SDK JavaScript est utilisé pour interagir avec un nœud Ethereum. L'API JSON RPC quant à lui sert d'interface entre les utilisateurs et les nœuds; elle est utilisée par le Web3. Cette interface permet d'accéder aux données stockées par les *Smarts Contracts* et de réagir à des événements survenus dans la Blockchain. L'interface JSON-RPC qui est une interface RPC qui utilise JSON comme structure de données pour modéliser les données à envoyer à la blockchain. Solidity est le langage de programmation pour coder les contrats intelligents.

Il existe des outils de développement qui aide à développer, tester et déployer des applications de manière à utiliser automatiquement les ressources de ces APIs. Parmi les IDE (environnements de développement intégrés) ou Framework qui existent jusque-là on peut citer :

- **Remix-IDE :** un remplacement de Mix-IDE, remix [45] est un environnement de développement de contrats intelligents en Solidity. Il intègre un compilateur, un environnement d'exécution, de débogage et de test.
- **Embark Framework** : permet de développer et de déployer facilement des applications décentralisées (DApps). Il intègre actuellement la machine virtuelle Ethereum (EVM), des stockages décentralisés (IPFS) et des plateformes de communication décentralisées (Whisper et Orbit).
- **Truffle Framework** [46] : est un environnement de développement intégré, une infrastructure de test et un portefeuille d'actifs de classe mondiale pour les blockchains utilisant la machine virtuelle Ethereum (EVM), dans le but de faciliter la vie des développeurs. Il génère le squelette d'une application Ethereum et d'un smart contract ainsi qu'un front-end. Truffle permet le déploiement des Dapps sur le client qui s'exécute sur la machine (la blockchain locale). Cette blockchain locale peut être un client Ethereum ou celle lancée par **GANACHE** (qui fait partie de la suite Truffle). Ganache permet d'avoir une blockchain Ethereum sur sa machine pour l'exécution des tests, des commandes et l'inspection des états de la blockchain. Par ailleurs, Truffle framework assure les **Test RPC** en exécutant en local la Dapp. Ces **packages NPM** offrent un outillage classique pour les développeurs : l'automatisation de la compilation **Solidity**, l'intégration aux outils de **tests**

unitaires, l'automatisation du déploiement et la communication avec la blockchain. Ces nombreuses caractéristiques ont motivé notre choix pour ce Framework.

Par ailleurs, la front-end du système est développé grâce au Framework **Angular** (version 7.2.14).

Pour assurer la communication entre la partie front-end et la base de données des E-livrets scolaire, nous avons utilisé java à travers le Framework **Spring**.

4.1.5. Configurations requises

La Figure 20 montre la structure de notre projet :

- le dossier **build** contient les fichiers Json des contrats qui sont générés lors du déploiement (avec la commande « **Truffle migrate** »).
- Le dossier **contracts** comme son nom l'indique contient les contrats intelligents
- **Src** est le dossier principal du projet ; il contient les classes, les services, les components repartis en modules. Les classes servent de facilitateurs de la manipulation de données reçues soit de la base de données soit de la blockchain par les services.

Figure 20: Structure du projet

Par ailleurs, le composant du pack Truffle Suite, Ganache permet de faire tourner notre blockchain Ethereum personnelle. Il génère les comptes avec un montant initial de 100 ETH. Le fichier **truffle-config.js** assure la liaison entre ganache et notre application. La Figure 21 montre quelques interfaces du fonctionnement de ganache.

4.1.6. Codage des services

✓ **Instanciation de Web3**

Le code qui suit permet de créer une instance web3 et définir un fournisseur. Un navigateur compatible avec Ethereum (dans notre cas, on utilise l'extension **MetaMask**) aura **web3.currentProvide**r disponible.

```
if (typeof window.web3 !== 'undefined') {
    this.web3Provider = window.web3.currentProvider;
    this.compatible = true;
} else {
    console.log('Sorry you are not connect');
        Web3.providers.HttpProvider.prototype.sendAsync =
Web3.providers.HttpProvider.prototype.send;
    this.web3Provider = new
Web3.providers.HttpProvider('http://ropsten.infura.io/v3/ca2b12f732ef47fcb6b76223b39f8c
f3');
    // Change to your own private key in infura.io
    this.compatible = false;
}
```

Cette instance de Web3 permet la récupération du compte connecté et sert de Provider aux ABI des contrats. Le code de récupération des comptes est donné en annexe 1.

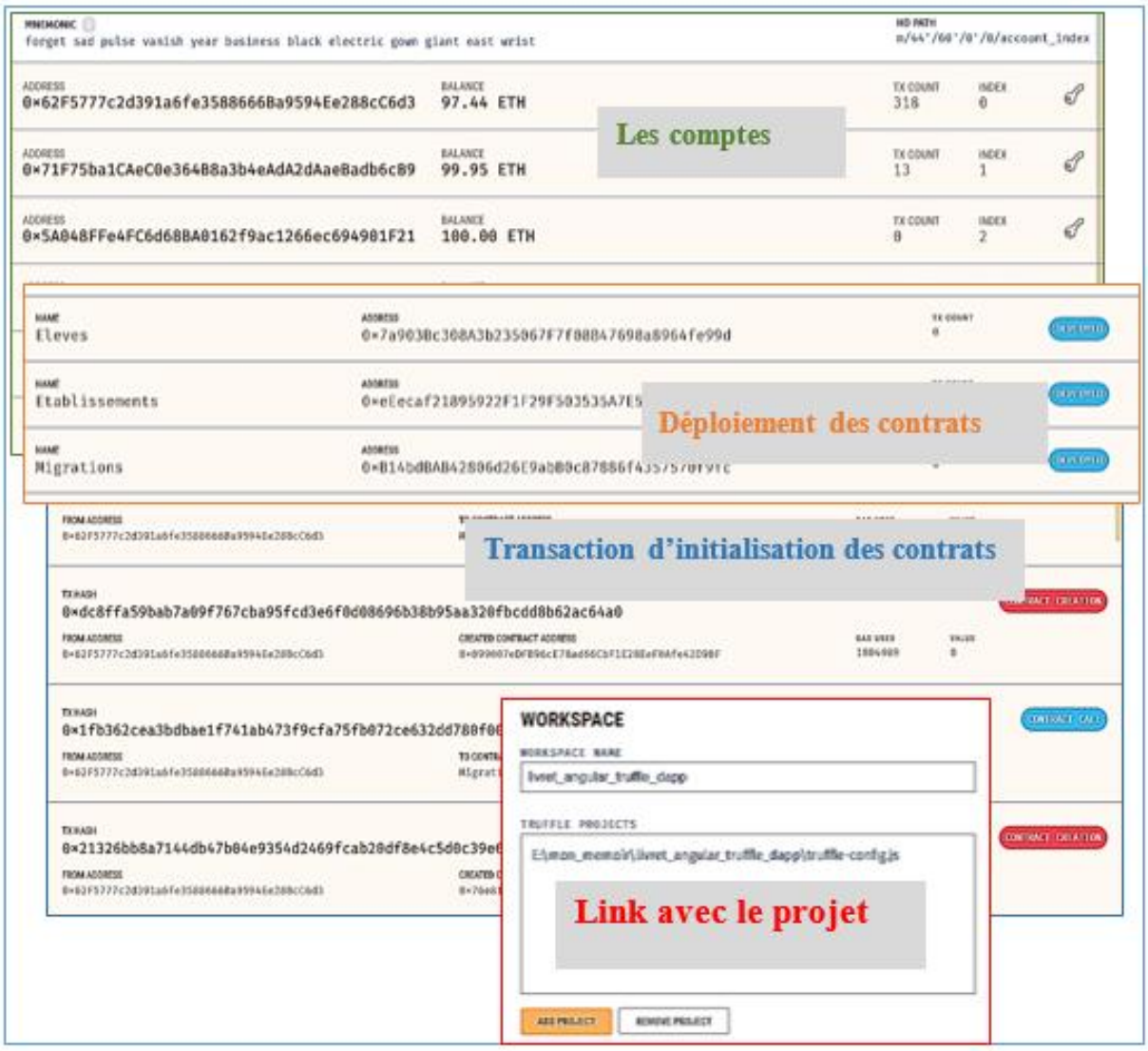


Figure 21 : Interfaces de Ganache

- ✓ Le service qui permet d'envoyer la transaction **addEtablissement**

```
addEtablissement(etab: Etablissement, originAccount: string) {
    const that = this;
    return new Promise((resolve, reject) => {
      const etablissementContract = contract(tokenAbiEtablissement);
      etablissementContract.setProvider(that.web3Provider);
      etablissementContract.deployed().then((instance) => {
        return instance.addEtablissement(
          etab.nom,
          etab.matricule,
          etab.adresse,
          etab.ief,
          etab.ia,
          { from: originAccount }
        );
      }).then((secretKey) => {
        return resolve({secretKey: secretKey});
      }).catch((error) => {
        return reject('Une erreur est survenue ...');
      });
    });
  }
```

3.8. Présentation de l'application

Dans cette partie, nous présentons les interfaces de l'application. L'application comporte trois principaux modules : le site visible par tout le monde, l'espace réservé aux établissements et l'espace réservé à YAKAARTIC.

4.1.7. Interfaces du site

La page d'accueil de notre Dapp se présente comme suit (Figure 22). Cinq menus de navigation sont observés sur cette page :

- Le menu **« Accueil »** : permet de revenir sur la page d'accueil qui explique brièvement la plateforme ;
- Le menu **« A propos »** : permet de naviguer à la page « à propos » (Figure 23) qui, comme son nom l'indique, montre le contexte et les objectifs visé par l'application ;
- L'onglet **« Fonctionnalités »**: qui nous mène vers la liste des fonctionnalités (Figure 24) offertes par l'application ;
- Le menu **« Support »** : cette menu nous conduit vers les informations de contact qui sont utiles en cas de besoin d'un support technique comme illustré à la Figure 25

- Et en fin **« Mon compte »** : qui récapitule les informations du compte tel que l'adresse et le solde connecté sur la blockchain (à travers l'extension **MetaMask**). On peut accéder à page d'authentification, montrée à la
- Figure 26, en cliquant sur « **Me connecter** ».

Figure 22: Interface de la page d'accueil du site

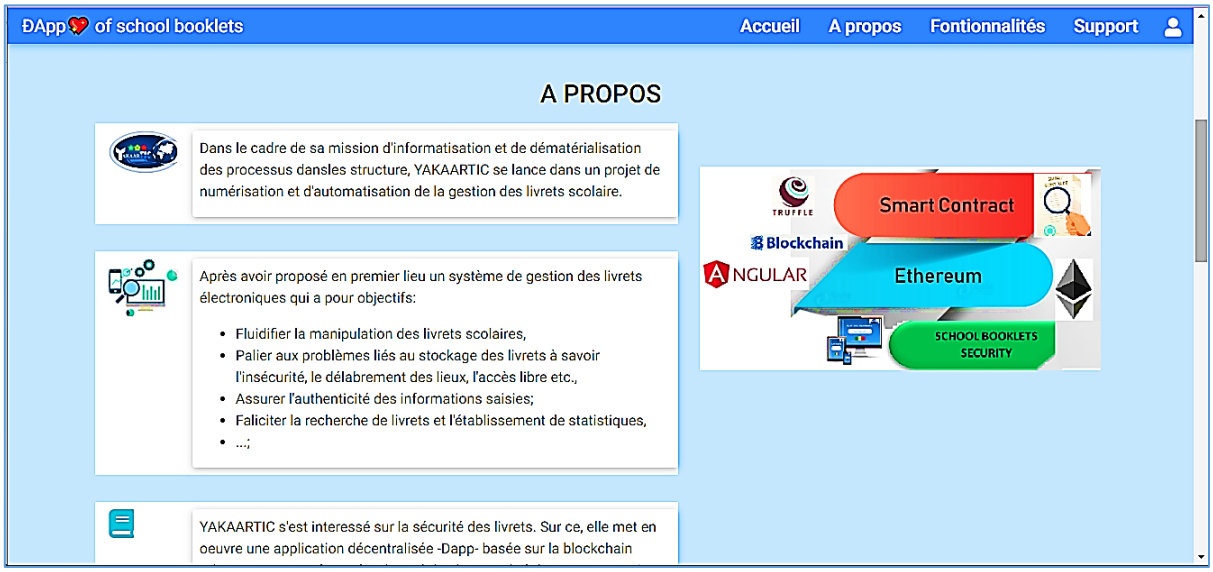


Figure 23: Interface de la page "A propos"

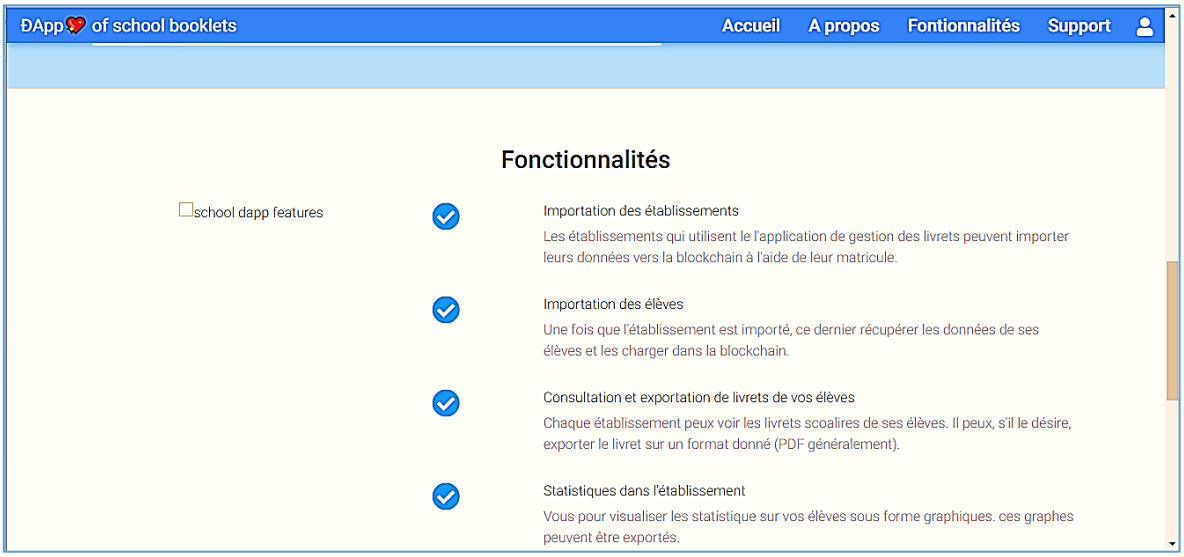


Figure 24: Interface de la page "Fonctionnalités"

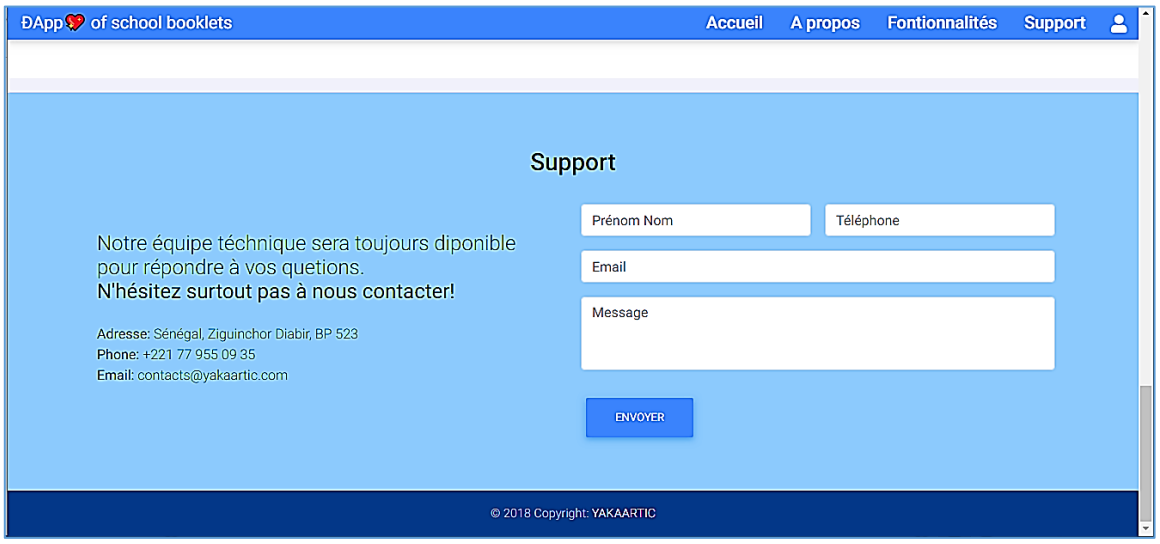


Figure 25: Interface de la page "Support"

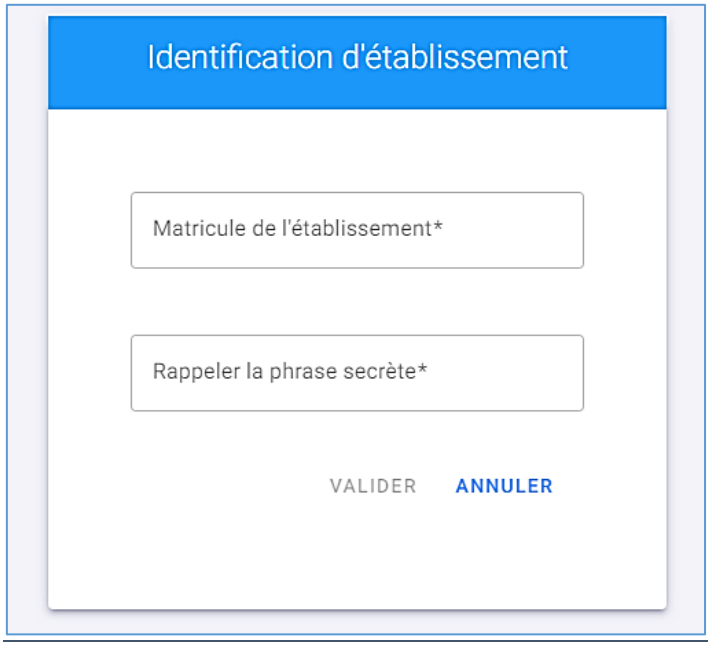


Figure 26: Interface d'authentification

Les informations demandées pour authentifier un établissement sont :

- ✓ Le matricule : généré à l'ajout de l'établissement dans la base de données du SGLE, ce matricule est unique pour tout établissement.
- ✓ La phrase sécrète. Vue que l'adresse de compte est assez long, nous avons fixé une phase sécrète dans la blockchain (ne pourra pas être modifié) qui servira de mot de passe aux établissements. Cette information doit être considérée top secret car la seule information qui garantit que l'utilisateur qui se connecte est autorisé à accéder aux espaces protégés.

Une fois que les informations de connexion sont vérifiées[10] et sont correctes, le système vérifie si le matricule saisie existe déjà dans les données des contrats :

- ✓ Si oui ; il nous redirige vers la page d'accueil de l'espace établissement (Figure 29).
- ✓ Le cas contraire le système lance la transaction de création de l'établissement qui doit être confirmé par l'utilisateur (Figure 27). Le succès de cette transaction est capturé dans Ganache à la Figure 28. La transaction échoue si solde du compte est insuffisant pour payer les frais de transaction.

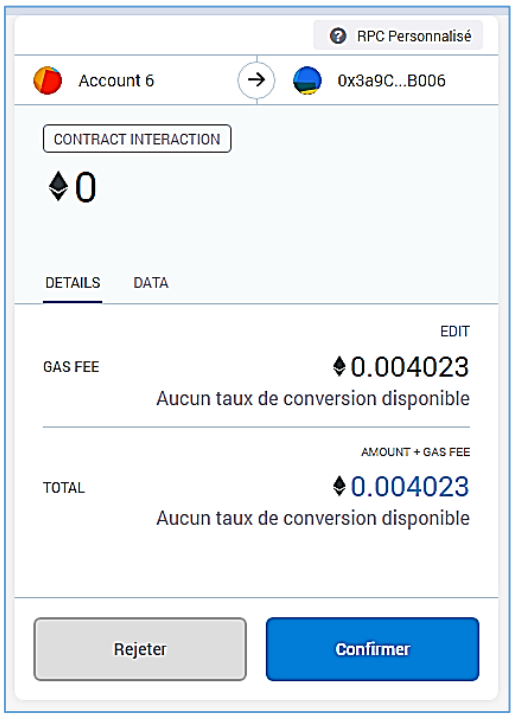


Figure 27: Envoi de la transaction AjoutEtablissement

[10] Pour le matricule, sa vérification est assuré par les services ouverts par le SGLE ; et pour la phase secret, et est vérifié à partir des données du contrat établissement.

Figure 28: Exécution du contrat "Etablissement" : appel de la fonction AddEtablissement()

4.1.8. Interfaces de l'espace réservé aux établissements

Une fois que l'authentification ait réussit, le système nous redirige vers la page d'accueil des établissements. Cette page qui est montrée à la … contient la barre des menus et le contenu de la page. Les différents menus sont :

- **Tableau de bord** : qui est aussi la page d'accueil permet de faire un bref rappel sur l'établissement connecté, le nombre d'année d'adhésion à la plateforme et le nombre d'élèves inscrit chaque année. La Figure 29 illustre cette page.
- **Elèves** : cette page (Figure 30) montre la liste des élèves de l'année courante. Si les élèves ne sont pas encore importés dans la blockchain, on verra le bouton « **importer** » à coté l'année scolaire. C'est partir de cette liste qu'on pourra visualiser le livret d'un élève en cliquant sur « détails ».
    - ✓ Les détails du livret s'affichent avec un sous-menu contenant la page de couverture, page de garde (Figure 31) du livret, la page des renseignements sanitaires (Figure 32) et les pages de parcours de toutes les classes de la sixième à la terminale. La Figure 33 montre le parcours des classes sixième et cinquième.
- **Historique** : en cliquant sur ce menu, nous aurons une page avec la liste des années scolaire importées dans la blockchain par l'établissement connecté. Pour chaque année on peut observer les statistiques sur les élèves suivant le sexe, les moyennes, par niveau ou de façon globale. On peut également visualiser les livrets des élèves pour chaque année.
- **Mon compte** : récapitule les informations du compte, tel que l'adresse et le solde, connecté sur la blockchain (à travers l'extension MetaMask). Le bouton « **déconnexion** » permet de supprimer les sessions de retourner à la page d'accueil du site présentée ci-dessus.

4.1.9. Interfaces de l'espace réservé à YAKAARTIC

L'espace YAKAARTIC lui permet d'avoir un suivi sur l'évolution des données de l'application c'est-à-dire les élèves et les établissements. En outre, l'entreprise YAKAARTIC peut prendre le rôle d'un établissement (importer les élèves) dans le cas où l'établissement concerné ne parvient pas à accéder à l'application pour une quelconque raison. Cependant le développement de cet espace n'a pas encore abouti.

Figure 29: Page d'accueil des établissements

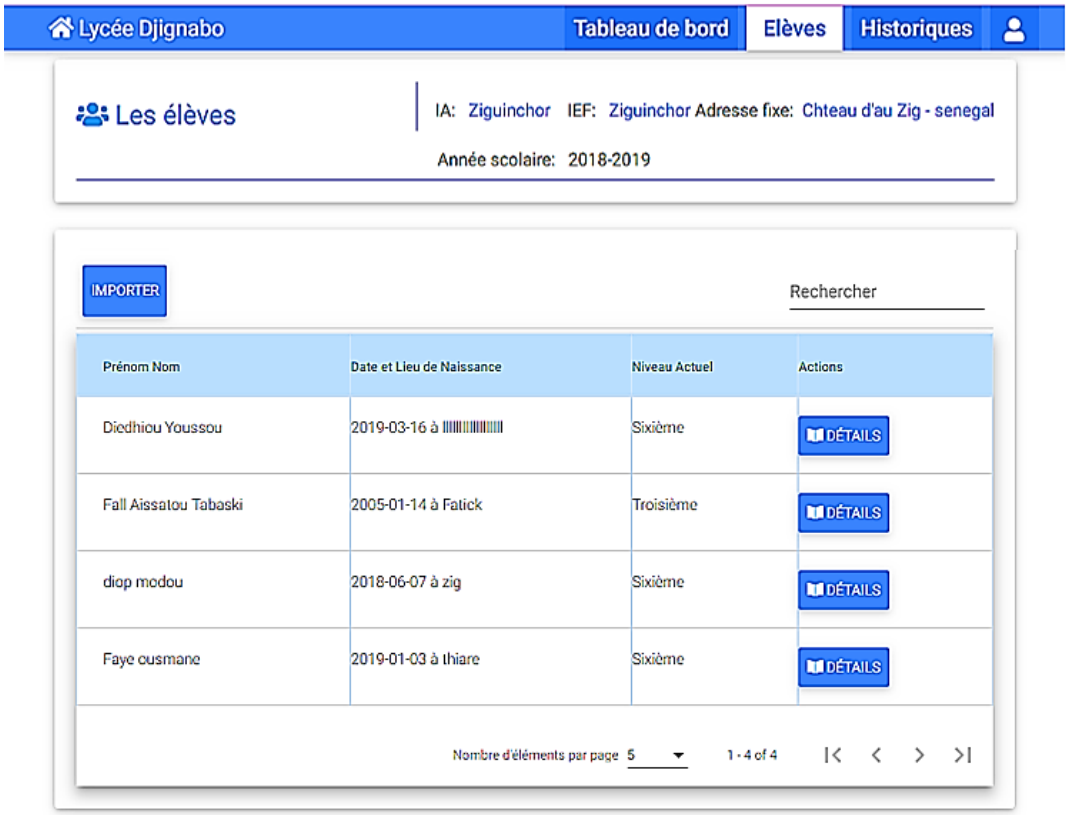


Figure 30: Pages des élèves (espace établissement)

ENSEIGNEMENTS GENERAL ET TECHNIQUE

Decret n° 78/691 du 12 juillet 1978

DOSSIER SCOLAIRE ET LIVRET SCOLAIRE POUR LE BACCALAUREAT

NOM DE L'ELEVE : DIEDHIOU

Prénoms : Youssou

Date de naissance : 16/03/2019 **à** IIIIIIIIIIIIIIII

Adresse : **Pays :** Sénégal

Nationalité : Sénégalaise

SCOLARITE DE PREMIER ET SECOND CYCLES

Classe	Année Scolaire	Etablissement	Commune
6e	2018-2019	Lycée Djignabo	
		C|	
5e		C|	
4e		C|	
3e		C|	
2nd		C|	
1ère		C|	
Terminale		C|	

C: redoublement pour - M: raison medical- R: résultats scolaires - X: autres raisons

Figure 31: page de couverture du livret

PRENOM NOM DE L'ELEVE : Youssou DIEDHIOU

RENSEIGNEMENTS SUR L'ETAT DE SANTE DE L'ELEVE

Classe de 6ème

Groupe d'éducation physique : ____ Acuité auditive : ____ Acuité visuelle : ____

Repercusions de l'état sur le travail de l'élève

Pas de répercusion [X] répercusion probable [ ]

répercusion certaine légère [X] importante [ ]

avec interruption de scolarité : courte [X] prolongée [ ] répétée [ ]

OBSERVATIONS:

Date : _____ Nom et Signature du médecin scolaire :

Classe de 2nd

Groupe d'éducation physique : ____ Acuité auditive : ____ Acuité visuelle : ____

Repercusions de l'état sur le travail de l'élève

Pas de répercusion [X] répercusion probable [ ]

répercusion certaine légère [X] importante [ ]

avec interruption de scolarité : courte [X] prolongée [ ] répétée [ ]

OBSERVATIONS:

Date : _____ Nom et Signature du médecin scolaire :

Classe de 5ème

Groupe d'éducation physique : ____ Acuité auditive : ____ Acuité visuelle : ____

Repercusions de l'état sur le travail de l'élève

Pas de répercusion [X] répercusion probable [ ]

répercusion certaine légère [X] importante [ ]

avec interruption de scolarité : courte [X] prolongée [ ] répétée [ ]

OBSERVATIONS:

Date : _____ Nom et Signature du médecin scolaire :

Classe de 1ère

Groupe d'éducation physique : ____ Acuité auditive : ____ Acuité visuelle : ____

Repercusions de l'état sur le travail de l'élève

Pas de répercusion [X] répercusion probable [ ]

répercusion certaine légère [X] importante [ ]

avec interruption de scolarité : courte [X] prolongée [ ] répétée [ ]

OBSERVATIONS:

Date : _____ Nom et Signature du médecin scolaire :

Classe de 4ème

Groupe d'éducation physique : _____ Acuité auditive : ____ Acuité visuelle : ____

Repercusions de l'état sur le travail de l'élève

Pas de répercusion [X] répercusion probable [ ]

répercusion certaine légère [X] importante [ ]

avec interruption de scolarité : courte [X] prolongée [ ] répétée [ ]

OBSERVATIONS:

Date : _____ Nom et Signature du médecin scolaire :

Terminale

Groupe d'éducation physique : _____ Acuité auditive : ____ Acuité visuelle : ____

Repercusions de l'état sur le travail de l'élève

Pas de répercusion [X] répercusion probable [ ]

répercusion certaine légère [X] importante [ ]

avec interruption de scolarité : courte [X] prolongée [ ] répétée [ ]

OBSERVATIONS:

Date : _____ Nom et Signature du médecin scolaire :

Classe de 3ème

Groupe d'éducation physique : ____ Acuité auditive : ____ Acuité visuelle : ____

Repercusions de l'état sur le travail de l'élève

Pas de répercusion [X] répercusion probable [ ]

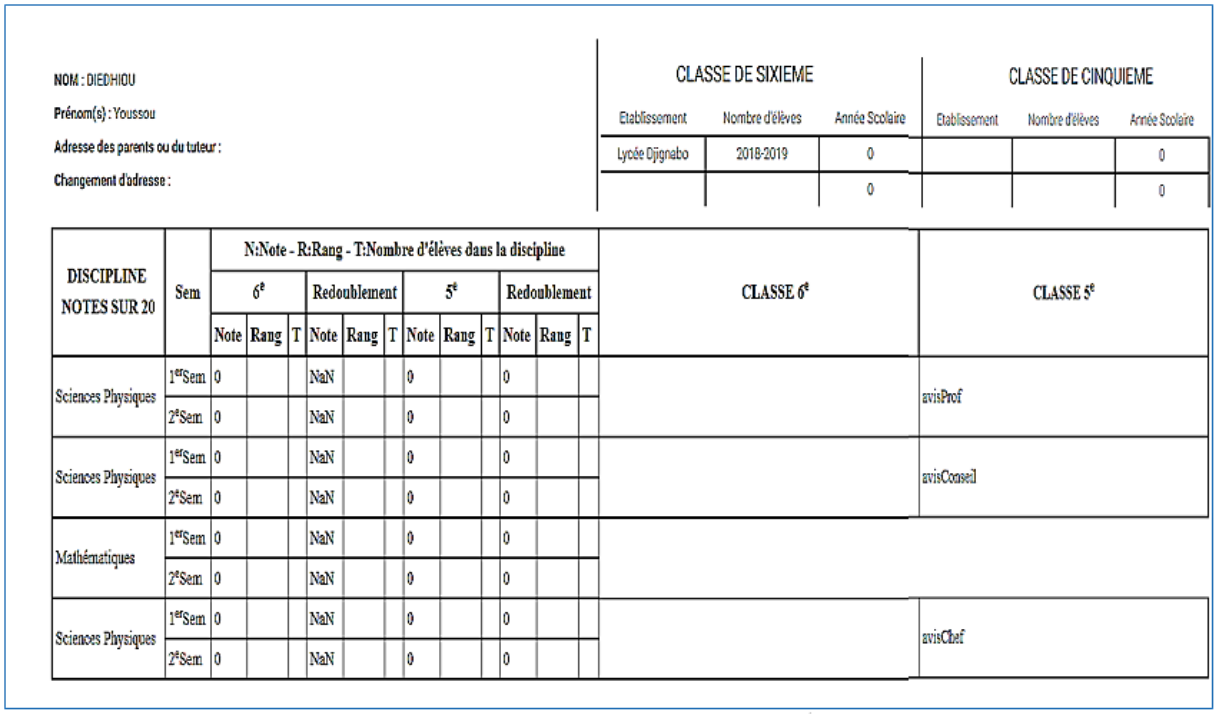
répercusion certaine légère [X] importante [ ]

avec interruption de scolarité : courte [X] prolongée [ ] répétée [ ]

OBSERVATIONS:

Date : _____ Nom et Signature du médecin scolaire :

Figure 32: Page des renseignements sanitaire



NOM : DIEDHIOU
Prénom(s) : Youssou
Adresse des parents ou du tuteur :
Changement d'adresse :

CLASSE DE SIXIEME			CLASSE DE CINQUIEME		
Etablissement	Nombre d'élèves	Année Scolaire	Etablissement	Nombre d'élèves	Année Scolaire
Lycée Djignabo	2018-2019	0			0
		0			0

DISCIPLINE NOTES SUR 20	Sem	N:Note - R:Rang - T:Nombre d'élèves dans la discipline												CLASSE 6e	CLASSE 5e
		6e			Redoublement			5e			Redoublement				
		Note	Rang	T	Note	Rang	T	Note	Rang	T	Note	Rang	T		
Sciences Physiques	1erSem	0			NaN			0			0				avisProf
	2eSem	0			NaN			0			0				
Sciences Physiques	1erSem	0			NaN			0			0				avisConseil
	2eSem	0			NaN			0			0				
Mathématiques	1erSem	0			NaN			0			0				
	2eSem	0			NaN			0			0				
Sciences Physiques	1erSem	0			NaN			0			0				avisChef
	2eSem	0			NaN			0			0				

Figure 33: Page du parcours 6e - 5e

La réalisation de l'application qui a fait l'objet de ce mémoire passe par un processus bien définit. D'abord l'analyse du système nous permet de concevoir les contrats intelligents qui décrivent le fonctionnement et les orientations de l'application décentralisée. Ensuite l'implémentions de cette dernière qui n'est rien d'autre que le codage des composants du système. Ce qui nous a permis de présenter les interfaces de l'application.

CONCLUSION GÉNÉRALE ET PERSPECTIVES

Nous voilà au terme ce travail de mémoire qui consistait en la réalisation d'une application décentralisée pour la sécurisation des E-Livrets Scolaires. En effet, la gestion des livrets scolaire des élèves au Sénégal est frappée par beaucoup de problèmes. A cause de son caractère papier, il est très difficile de stocker les livrets dans des lieux physiques sans failles.

L'entreprise YAKAARTIC a proposé une solution informatique pour pallier aux problèmes qui gangrène l'utilisation du livret scolaire papier en la dématérialisant. Mais cette dématérialisation ne garantit pas totalement un haut niveau de sécurité requis par le livret scolaire de par son importance dans la prise de décision au sein des établissements et dans les examens de BEFEM et de BAC. C'est en ce sens qui nous avons proposé dans ce mémoire une application décentralisée pour renforcer la sécurité des livrets électroniques fournis par cette solution.

En effet, des travaux recherches ont été effectués pour mieux appréhendé le domaine et le fonctionnement des établissements dans ce qui concerne les livrets scolaires. Ce qui a permis d'écrire le premier chapitre de ce rapport de mémoire.

Par ailleurs, une connaissance parfaite de la technologie des blockchains est nécessaire pour atteindre les objectifs que nous nous sommes fixés à l'entame de ce mémoire. Ainsi les deuxième et troisième chapitres sont réservés à cet effet.

Au quatrième et dernier chapitre, nous avons montré les éléments techniques qui rentrent dans le cadre de la réalisation de l'application.

A ce stade de développement ; l'application décentralisée pour la sécurisation des livrets scolaires permet aux établissements :

- D'importer les informations caractérisant leurs écoles depuis la base de données du SGLE vers la Blockchain des livrets scolaires
- D'importer également les élèves inscrits dans l'année cours, leurs notes et les appréciations du conseil, des professeurs, bref toutes les données de leurs livrets scolaires.
- De visualiser la liste de ses élèves, leurs livrets scolaires sous format PDF ou de les télécharger. Ce qui permet de vérifier la conformité avec le livret généré avec le SGLE.
- De voir les statistiques sur ses élèves et les exporter sous format PDF.

Les perspectives d'ouverture et les travaux futurs, qui doivent compléter ou étendre le travail effectué, sont :

- Compléter le développement de l'espace réservé à YAKAARTIC (B) pour l'offrir la possibilité de voir les statistiques globaux des différents établissements ;

- Développer les contrats qui permettront à l'office du bac de pouvoir générer automatiquement la liste de tous les élèves inscrits en terminale pour les besoin nécessaire à l'organisation de l'examen de baccalauréat.
- Déployer notre Dapp dans la blockchain publique Ethereum pour qu'elle soit accessible par tous les établissements.

RÉFÉRENCES

[1] S. Sene *et al.*, « SES 2013 par ANSD : Chapitre III : EDUCATION », p. 26, 2013.

[2] S. Sene *et al.*, « SES 2014 par ANSD : Chapitre III : EDUCATION ». 2014.

[3] R. Subramanian et T. Chino, « The State of Cryptocurrencies, Their Issues and Policy Interactions », *J. Int. Technol. Inf. Manag.*, vol. 24, n° 3, janv. 2015.

[4] W. Dai, « b-money: A scheme for a group of untraceable digital pseudonyms to pay each other with money and to enforce contracts amongst themselves without outside help. », 1998. [En ligne]. Disponible sur: http://www.weidai.com/bmoney.txt. [Consulté le: 01-déc-2018].

[5] J. DAVIS, « The crypto-currency: Bitcoin and its mysterious inventor. », vol. 10, p. 6, 2011.

[6] S. Nakamoto, « Bitcoin: A Peer-to-Peer Electronic Cash System », p. 9.

[7] A. Narayanan et J. Clark, « Bitcoin's Academic Pedigree The concept of cryptocurrencies is built from forgotten ideas in research literature », p. 30.

[8] Blockchain France, *La Blockchain décryptée: les clefs d'une révolution*. Paris: Observatoire Netexplo, 2016.

[9] F. Tschorsch et B. Scheuermann, « Bitcoin and Beyond: A Technical Survey on Decentralized Digital Currencies », *IEEE Commun. Surv. Tutor.*, vol. 18, n° 3, p. 2084-2123, thirdquarter 2016.

[10] M. Belotti, N. Bozic, G. Pujolle, et S. Secci, « A Vademecum on Blockchain Technologies: When, Which and How ». 04-oct-2018.

[11] M. Swan, *Blockchain: Blueprint for a New Economy*. O'Reilly Media, Inc., 2015.

[12] « Bitcoin - Argent P2P libre et ouvert ». [En ligne]. Disponible sur: https://bitcoin.org/fr/. [Consulté le: 26-nov-2018].

[13] *Bitcoin Core integration/staging tree. Contribute to bitcoin/bitcoin development by creating an account on GitHub*. Bitcoin, 2019.

[14] « Litecoin - Monnaie numérique P2P Open Source ». [En ligne]. Disponible sur: https://litecoin.org/. [Consulté le: 11-janv-2019].

[15] T. Gibbs et S. Yordchim, « Thai Perception on Litecoin Value », *Int. J. Econ. Manag. Eng.*, vol. 8, n° 8, p. 3, 2014.

[16] « Litecoin Project », *GitHub*. [En ligne]. Disponible sur: https://github.com/litecoin-project. [Consulté le: 11-janv-2019].

[17] « Dash Site Officiel | Dash Crypto Currency - Dash ». [En ligne]. Disponible sur: https://www.dash.org/. [Consulté le: 11-janv-2019].

[18] M. E. Gladden, « Cryptocurrency with a Conscience: Using Artificial Intelligence to Develop Money that Advances Human Ethical Values », déc. 2015.

[19] « Ethereum Project ». [En ligne]. Disponible sur: https://www.ethereum.org/. [Consulté le: 28-nov-2018].

[20] « Slock.it - Landing ». [En ligne]. Disponible sur: https://slock.it/. [Consulté le: 11-janv-2019].

[21] « Namecoin ». [En ligne]. Disponible sur: https://namecoin.org/. [Consulté le: 04-déc-2018].

[22] *An empirical study of Namecoin and lessons for decentralized namespace design*. .

[23] E. Androulaki *et al.*, « Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains », in *Proceedings of the Thirteenth EuroSys Conference*, New York, NY, USA, 2018, p. 30:1–30:15.

[24] « Hyperledger – Open Source Blockchain Technologies ». [En ligne]. Disponible sur: https://www.hyperledger.org/. [Consulté le: 12-janv-2019].

[25] « Blockchain Technology in Online Voting », *Follow My Vote*. .

[26] « bitcoin-blockchain-infographic-fr.jpg (1457×1030) ». [En ligne]. Disponible sur: https://bitconseil.fr/wp-content/uploads/2016/03/bitcoin-blockchain-infographic-fr.jpg. [Consulté le: 10-déc-2018].

[27] G. MARIN-DAGANNAUD, « Comprendre la blockchain Ethereum – Article 1 : Bitcoin, première implémentation de la blockchain (1/2) », *Ethereum France*, 03-juin-2016. [En ligne]. Disponible sur: https://www.ethereum-france.com/comprendre-la-blockchain-ethereum-article-1-bitcoin-premiere-implementation-de-la-blockchain-12/. [Consulté le: 10-déc-2018].

[28] M. Laurent, « La blockchain est-elle une technologie de confiance », in *Signes de confiance : l'impact des labels sur la gestion des données personnelles*, Institut Mines-Télécom, 2018, p. 179-198.

[29] A. Kiayias, E. Koutsoupias, M. Kyropoulou, et Y. Tselekounis, « Blockchain Mining Games », in *Proceedings of the 2016 ACM Conference on Economics and Computation*, New York, NY, USA, 2016, p. 365–382.

[30] M. Carlsten, H. Kalodner, S. M. Weinberg, et A. Narayanan, « On the Instability of Bitcoin Without the Block Reward », in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, New York, NY, USA, 2016, p. 154–167.

[31] F. Pianese, M. Signorini, et S. Sarkar, « Small Transactions with Sustainable Incentives », in *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, 2018, p. 1-5.

[32] Ö. Gürcan, A. Del Pozzo, et S. Tucci-Piergiovanni, « On the Bitcoin Limitations to Deliver Fairness to Users », in *On the Move to Meaningful Internet Systems. OTM 2017 Conferences*, 2017, p. 589-606.

[33] G. Florin et S. Natkin, « LES TECHNIQUES DE CRYPTOGRAPHIE », p. 79, 2001.

[34] « Qu'est-ce que la cryptographie à clé publique ? » [En ligne]. Disponible sur: https://www.globalsign.fr/fr/centre-information-ssl/cryptographie-cle-publique/. [Consulté le: 10-déc-2018].

[35] « The Elliptic Curve Digital Signature Algorithm (ECDSA) | SpringerLink ». [En ligne]. Disponible sur: https://link.springer.com/article/10.1007/s102070100002. [Consulté le: 12-janv-2019].

[36] « Cheneau et al. - Amélioration des performances des adresses CGA et .pdf ». .

[37] V. Buterin, « A NEXT GENERATION SMART CONTRACT & DECENTRALIZED APPLICATION PLATFORM », p. 36.

[38] D. G. Wood, « ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER », p. 32.

[39] *The Ethereum Wiki - Ethash*. ethereum, 2019.

[40] N. Szabo, « Formalizing and Securing Relationships on Public Networks », *First Monday*, vol. 2, nº 9, sept. 1997.

[41] « Solidity — Solidity 0.5.4 documentation ». [En ligne]. Disponible sur: https://solidity.readthedocs.io/en/latest/. [Consulté le: 25-janv-2019].

[42] V. Buterin, *On public and private blockchains (2015)*. 2015.

[43] N. Atzei, M. Bartoletti, et T. Cimoli, « A Survey of Attacks on Ethereum Smart Contracts (SoK) », in *Principles of Security and Trust*, vol. 10204, M. Maffei et M. Ryan, Éd. Berlin, Heidelberg: Springer Berlin Heidelberg, 2017, p. 164-186.

[44] « MetaMask ». [En ligne]. Disponible sur: https://metamask.io/. [Consulté le: 25-janv-2019].

[45] « Remix - Solidity IDE ». [En ligne]. Disponible sur: http://remix.ethereum.org/#optimize=false&version=soljson-v0.5.1+commit.c8a2cb62.js. [Consulté le: 25-janv-2019].

[46] « Truffle Suite | Documentation ». [En ligne]. Disponible sur: https://truffleframework.com/docs. [Consulté le: 25-janv-2019].

ANNEXES

```
seeAccountInfo() {
  return new Promise((resolve, reject) => {
    window.web3.eth.getCoinbase((err, account) => {
        if (account === true) {
          console.log('dondt work' + account);
          return reject({name: 'account'});
        } else {
          window.web3.eth.getBalance(account, (error, balance) => {
            if (error === false) {
              return resolve({
                originAccount: account,
                balance: (window.web3.utils.fromWei(balance, 'ether'))
              });
            } else {
              console.log(balance);
              return reject({name: 'balance'});
            }
          });
        }
    });
  });
}
```

L'appel dans nos component

```
this.etabService.seeAccountInfo()
.then((value: any) => {
     this.direction = value.originAccount;
     this.balance = value.balance;
     console.log(this.direction);
     this.getEtabs();
}).catch((error: any) => {
     console.log(error);
});
```

Annexe 1: Web3Js pour la communication avec nos comptes dans la Blockchain

La fonction suivante permet de récupérer un établissement à partir de son adresse de compte ainsi la liste de ses élèves.

```
getEtabContractByAdresse(direction: string, anneeScolaire: string): Promise<any> {
  const that = this;
  return new Promise(async (resolve, reject) => {
    const etablissementContract = contract(tokenAbiEtablissement);
    etablissementContract.setProvider(that.web3Provider);
    let etablissementInstance: any;
    let etab = new Etablissement();
    const eleveContract = contract(tokenAbi);
    eleveContract.setProvider(that.web3Provider);
    let eleves: Array<Livret> = [];

    await etablissementContract.deployed().then((instance) => {
      etablissementInstance = instance;
      return instance.nbreEcole();
    }).then(async (nbreEcole) => {
      for (let i = 1; i <= nbreEcole; i++) {
        await etablissementInstance.etablissements(i).then(async (etablissement: any) => {
          if (etablissement[2].toLowerCase() === direction.toLowerCase()) {
            etab.id = etablissement[0];
            etab.matricule = etablissement[1];
```

```
            etab.nom = etablissement[3];
            etab.adresse = etablissement[4];
            etab.ief = etablissement[5];
            etab.ia = etablissement[6];
            etab.accountAdresse = etablissement[2];
          }
          if (nbreEcole == i) {
            await this.getInscriptionsAnneeByEtab(direction, anneeScolaire).then(async (ins) => {
              await ins.forEach(async (inscription: Inscription,indexIns) => {
                await this.getEleveById(inscription.idEleve).then(async (eleve: Livret) => {
                  await this.getSemestresByIncription(inscription.id).then(async (semestres: any) => {
                    await semestres.forEach((sem: Semestre, index) => {
                      //console.log(sem);
                      inscription.semestres.push(sem);
                      if (index == semestres.length - 1) {
                        eleve.inscriptions.push(inscription);
                        let indexEleve = this.existeEleve(eleves, eleve.id);
                        if (indexEleve == -1) {
                          eleves.push(eleve);
                        } else {
                          eleves[indexEleve] = eleve;
                        }
                      }
                    });
                    if (semestres.length == 0) {
                      eleve.inscriptions.push(inscription);
                      let indexEleve = this.existeEleve(eleves, eleve.id);
                      if (indexEleve == -1) {
                        eleves.push(eleve);
                      } else {
                        eleves[indexEleve] = eleve;
                      }
                    }
                  }).catch((error) => {
                    console.log(error);
                    return reject('Une erreur est survenue ...');
                  });
                }).catch((error) => {
                  console.log(error);
                  return reject('Une erreur est survenue ...');
                });
                if (indexIns == ins.length - 1) {
                  return resolve({ etablissement: etab, livrets: eleves });
                }
              });
              if (ins.length == 0) {
                return resolve({ etablissement: etab, livrets: eleves });
              }
            }).catch((error) => {
              console.log(error);
              return reject('Une erreur est survenue ...');
            });
          }
        });
      }
    }).catch((error) => {
      console.log(error);
      return reject('Une erreur est survenue ...');
    });
  });
}
```