Spécification Composant I

Lecture fichiers et blocs

Le composant I est la représentation de la blockchain. Dans cette spécification, la structure du fichier et les fonctionnalités attendues seront détaillées.

1. Structure du fichier

Format et enregistrement

Le fichier sera un document au format plat : extension .txt ou éventuellement sans extension (sous Linux). Le fichier sera composé d'un ensemble de blocs et chaque bloc contiendra un ensemble de transactions.

Format des transactions

Une transaction est un message structuré. Une transaction enregistre une opération de transfert d'avoirs entre les utilisateurs. Une transaction possède un identifiant unique, une date de réception, le numéro du bloc dans lequel elle a été validée, identifiant de l'auteur de la transaction, le destinataire, les entrées et les sorties et éventuellement une taxe. Le format de représentation dans le fichier peut être mis sous format plat. Dans ce cas, chaque enregistrement est séparé par un point-virgule.

Ci-dessous, la représentation d'une transaction au format JSON.

```
{
  "Identifiant": "5442h",
  "DateDeCreation": "2017-02-27 13:25:09",
  "Bloc": 45484,
  "SenderId": 153,
  "ReceiverId": 158,
  "TotalEntrees": 583.56,
  "TotalSortiess": 583.56
  }
```

Les transactions sont irréversibles par leur conception donc elles ne peuvent pas être modifiées ou supprimées.

Format des blocs

Les transactions sont enregistrées dans des blocs. Les blocs sont eux-mêmes chainés les uns aux autres pour former la blockchain. Un bloc est structuré en deux parties : un en-tête et un corps. L'en-tête permet de restituer le jeton d'horodatage. Le jeton d'horodatage est une empreinte numérique qui authentifie un bloc comme un chainon de la blockchain. Le corps contient un ensemble de transactions.

Un bloc possède un **identifiant**, un **hash** ou profondeur de la chaine à partir de ce bloc (s'il y a 400 blocs alors le suivant aura pour id 401), le **nombre de transactions**, **son code de hachage** et **celui du bloc précédent**, **une date de réception** ou de création, **la version du hash**, **la somme des transactions** et **une liste des transactions**. Dans le cas d'un fichier plat, les blocs seront séparés par : ou @ ou #.

```
{
  "Identifiant ou profondeur": "5442b",
  "NombreDeTransaction": 45484,
  "Hashes":{
                        "Hash": "b94d27b93e08a5380ee9088f7ace2efcde9",
                        "BlocPrecedent": "4d2e5abfac48a5380ee9088f7ace2"
                },
  "DateDeCreation": "2017-02-27 13:25:09",
  "Version": "0x2000",
  "Transactions»: [
            {

            },
            {

            }
    ]
}
```

2. Fonctionnalités

Lecture de fichier : Cette première fonction a pour but d'effectuer une lecture totale du fichier. Ainsi l'ensemble du fichier sera parcouru et chaque bloc sera identifié.

Cette fonction prendra en entrée le chemin d'accès absolu du fichier et retournera une liste des blocs.

Insertion d'un bloc : Cette fonction a pour but d'insérer un bloc dans le fichier.

Elle prend en entrée un bloc et on l'insère sur la chaine la plus longue (chaine principale).

Recherche d'un bloc : Cette fonction permet de retrouver un bloc à partir d'un hash. Cette fonction prend en entrée un hash et retourne un bloc. Si le hash n'a pas de bloc qui lui est associé, on ne retourne rien.

Recherche d'une transaction: Cette fonction permet de retrouver d'une transaction à partir de son identifiant.

Elle prend en argument un identifiant de transaction. Si l'identifiant n'est pas associé à une transaction, on ne retourne rien.

Recherche des transactions d'un utilisateur: Cette fonction prend en entrée un identifiant d'un utilisateur et retourne toutes ses transactions.