Rapport TP1

Blockchain

Étape 1:

Pour l'étape numéro 1, je n'ai pas eu grand-chose de spécial à faire. J'ai créé mon dépôt Git et j'ai cloné le dépôt qui était mentionné dans l'énoncé du TP1.
Une fois cela fait, dans le fichier "server.js", il a fallu que j'ajoute deux "console.log()" avant chaque break afin d'indiquer que la méthode GET ou POST était bien fonctionnelle, ainsi que d'afficher le résultat en console.
Pour tester les requêtes GET et POST, j'ai utilisé Postman avec l'adresse URL : "http://localhost:3000/blockchain".

Étape 2:

Pour l'étape 2, l'objectif était de développer la méthode findBlocks(), et pour cela, il a fallu :

- Créer un dossier "data" à la racine avec un fichier "blockchain.json" (ce sera le fichier qui contiendra la blockchain).
- Changer la ligne 8 du fichier "blockchainStorage.js" afin d'inclure, dans la variable path, le chemin vers "blockchain.json".
- Écrire un message simple dans "blockchain.json" comme : {"message" : "Bonjour à tous"}.
- Enfin, coder findBlocks() en utilisant await, car readFile(path), qui permet de lire le fichier dans le chemin contenu dans la variable path, est une fonction asynchrone qui retourne une Promise.

Étape 3:

Pour l'étape 3, l'objectif était de développer la méthode createBlock(), et pour cela, il a fallu :

- Installer uuid via la commande npm install uuid, ainsi que l'importer dans blockchainStorage en écrivant : "import { v4 as uuidv4 } from 'uuid';".
- Coder la méthode getDate() dans divers.js en faisant un return new Date().toISOString().
- Finalement, on peut coder createBlock() en récupérant la liste existante avec findBlocks() et en s'assurant que c'est bien un tableau. Une variable message est définie en fonction du dernier bloc, bien que son utilisation ici soit uniquement informative. Ensuite, un nouvel objet newBlock est créé avec un identifiant unique (uuidv4()), des données (nom, don, date), puis ajouté à la liste. Enfin, la liste mise à jour est enregistrée dans un fichier via writeFile(path, JSON.stringify(newBlocks)), et la nouvelle liste est retournée.

- Finalement, on peut tester sur Postman la requête POST avec les champs nom et don au format JSON.

Étape 4:

Pour la dernière étape, il faut ajouter un hash à chaque bloc. Et pour cela, il faut :

- Définir la méthode findLastBlock() en récupérant la liste avec findBlocks(), puis en récupérant le dernier élément.
- Importer la classe Hash avec : “import {createHash} from 'node:crypto'”.
- Intégrer ensuite un mécanisme de hashage basé sur le dernier bloc, récupéré avec findLastBlock().