RAPPORT TP1 – Développement avancé

Étape 1 : Mise en place du serveur HTTP

La première étape consistait à vérifier le bon fonctionnement du serveur HTTP. Dans le fichier `server.js`, un message de confirmation a été ajouté pour s'afficher dans la console au démarrage du serveur. Cela permet d'identifier facilement que tout fonctionne correctement.

Les tests ont été effectués avec les console.log

Étape 2 : Lecture des blocs

Pour cette étape, un fichier `blockchain.json` a été créé dans un dossier nommé `data`. Ce fichier contient un simple message de test initial

Ensuite, la fonction `findBlocks` a été créer. Elle permet de lire le contenu de ce fichier JSON et de renvoyer les blocs au serveur. En cas d'erreur ou si le fichier est vide, elle retourne un tableau vide, ce qui évite tout blocage.

Une fois cette fonction prête, un test a été effectué avec une requête `GET /blockchain` via Postman. Le serveur répond en renvoyant le contenu du fichier `blockchain.json`, validant ainsi le fonctionnement de la lecture.

Étape 3 : Ajout de blocs

L'objectif suivant était de permettre l'ajout de nouveaux blocs dans la blockchain. La fonction `createBlock` a été codée pour accomplir cette tâche. Elle prend en entrée les informations du bloc, comme le nom du donateur et le montant du don

Le bloc est ensuite ajouté à la liste des blocs existants, et l'ensemble est sauvegardé dans le fichier `blockchain.json`.

Les tests réalisés avec Postman montrent que les nouveaux blocs s'ajoutent correctement. Après chaque requête `POST /blockchain`, le fichier est mis à jour et contient bien les nouveaux blocs ajoutés.

Étape 4 : Hachage et intégrité des blocs

Pour sécuriser les blocs et garantir leur intégrité, le champ `hash` a été ajouté à chaque bloc. Ce champ est calculé à partir des données du bloc précédent, en utilisant l'algorithme SHA-256. La fonction `findLastBlock` a été développée pour récupérer facilement le dernier bloc de la chaîne, ce qui est nécessaire pour calculer le hash du nouveau bloc.

Une fonction `verifBlocks` a également été écrite pour parcourir la blockchain et vérifier l'intégrité des blocs. Elle compare les hashes des blocs avec les valeurs recalculées. Si une modification non autorisée est détectée, la fonction retourne `false`.

Pour tester cette étape, plusieurs blocs ont été ajoutés à la blockchain, et leur intégrité a été vérifiée avec une requête `GET /blockchain/verify`. Une modification manuelle dans le fichier `blockchain.json` a été effectuée pour simuler une altération, et la fonction de vérification a bien détecté l'anomalie.