

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

Ensuite, la fonction `findBlocks` a été codée. Son rôle est de lire le fichier JSON, convertir son contenu en tableau, et le retourner. Une gestion des erreurs a aussi été mise en place pour éviter que le programme plante si le fichier est vide ou inaccessible.

Test avec **GET /blockchain** : le serveur retourne bien la liste des blocs (vide ou remplie, selon le fichier).

Étape 3 : Ajout de blocs

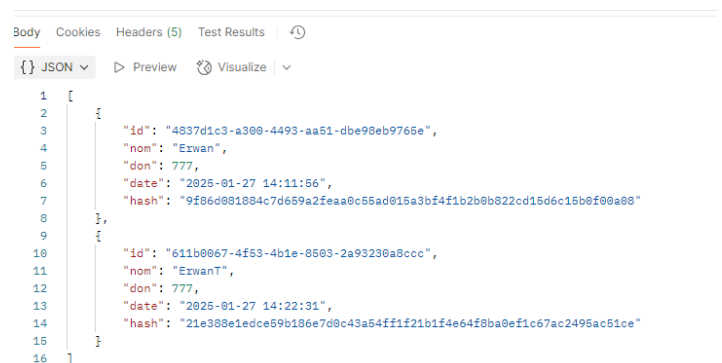
La fonction `createBlock` a été développée pour ajouter de nouveaux blocs dans la blockchain. Chaque bloc contient :

Un **identifiant unique** généré

Un **nom de donateur** et un **montant** reçus depuis la requête POST.

Une **date** générée automatiquement avec `getDate`.

Un **hash** calculé à partir des données du bloc précédent.



```
1 [
2   {
3     "id": "4837d1c3-a308-4493-aa51-dbe98eb9766e",
4     "nom": "Etwan",
5     "don": 777,
6     "date": "2025-01-27 14:11:56",
7     "hash": "9f86d081884c7d659a2feaa0c55ad015a3bf4f1b2b8b822cd15d6c15b0f00a08"
8   },
9   {
10    "id": "611b0067-4f53-4b1e-8503-2a93230a8ccc",
11    "nom": "Etwan",
12    "don": 777,
13    "date": "2025-01-27 14:22:31",
14    "hash": "21e380e1edce59b106e7d0c43a54ff1f21b1f4e64f8ba0ef1c67ac2495ac51ce"
15  }
16 ]
```

Le nouveau bloc est ajouté à la liste existante, et la blockchain mise à jour est sauvegardée dans le fichier `blockchain.json`.

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.