

COMPTE RENDU

TP1



CHAIMAE EL MATTARI

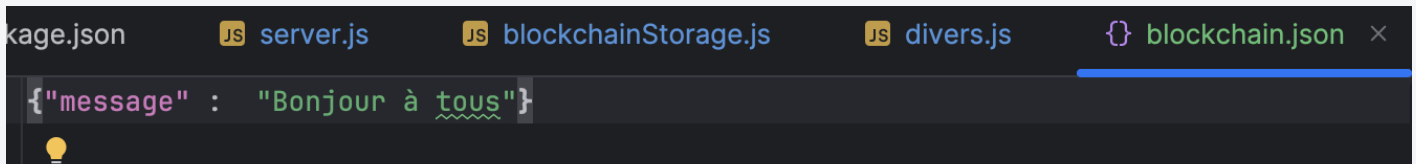
304

ÉTAPE 2 – APPRENONS À LIRE

Objectif de la Première Fonction findBlocks(): récupérer l'ensemble de la blockchain et la retourner au client sous forme de Json

Comment ? :

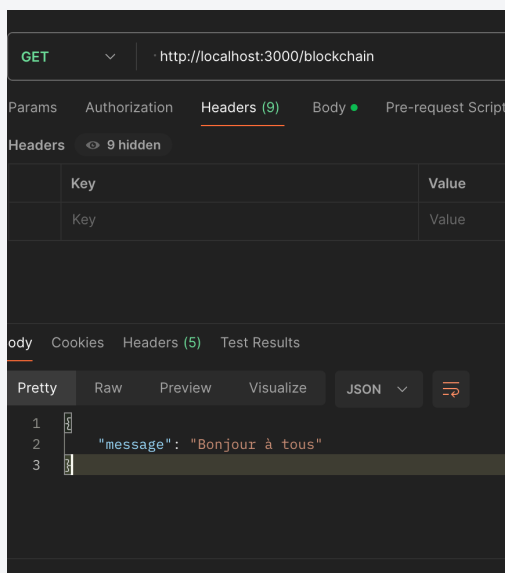
Pour ce faire j'ai commencé par créer un fichier blockchain.js que voici :



```
["message" : "Bonjour à tous"]
```

Ensuite, j'ai implémenté la fonction findBlock permettant de lire le fichier blockchain.js, et de retourner le contenu de ce fichier au format JSON. Ainsi, lors d'un appel GET /blockchain, le serveur renvoi ce contenu JSON.

Voici ce que retourne la requête sur Postman et dans la console :



```
JSON : {"message" : "Bonjour à tous"}

Get : {"message" : "Bonjour à tous"}
```

La fonction findBlock est déclarée avec le 'async', ce qui permet l'utilisation du 'await' et qui permet par conséquent de gérer les opérations asynchrones. J'utilise ensuite un try-catch afin d'encapsuler le code et de gérer les erreurs. Ensuite, la fonction readFile du module 'node:fs/promises' permet la lecture du fichier, dont le chemin a été défini dans une variable path. Puis, je convertis le contenu du fichier en JSON, et j'affiche dans la console ce contenu d'une part, et d'autre part je retourne le JSON en tant que réponse à la requête GET.

Facilités : L'utilisation du module 'node:fs/promises' facilite grandement la manipulation de fichier en asynchrones, et la conversion en JSON est directe.

Difficultés : J'ai eu quelques difficultés avec l'utilisation de Postman que je n'ai jamais utilisé auparavant. Et j'ai dans un premier temps eu du mal à retourner correctement les données, notamment car le chemin de mon fichier n'était pas correct.

Surmonter les difficultés : La gestion des erreurs mis en place par les try catch m'a permis de repérer les erreurs plus facilement et de les traiter efficacement.

ÉTAPE 3 – UNE BRIQUE APRÈS L'AUTRE ET ÉTAPE 4 – « VERS L'INFINI ET AU-DELÀ ! »

Objectif de la fonction createBlock(): créer un nouveau block en ajoutant le champs hash, ajouter ce block à la chaîne existante et enregistrer cette chaîne dans le fichier blockchain.json.
Objectif de la fonction findLastBlock(): trouver le dernier bloc de la chaîne existante.

Comment ? :

J'ai commencé par compléter la fonction getDate du fichier divers.js, permettant d'obtenir un timestamp au format spécifié :

```
/**
 * @description Retourne un timestamp au format aaaammj-jj-hh:mm:ss
 * @return {string}
 */
1+ usages  ▲ Laurent Giustignano *
export function getDate():string {
  // A coder exo 3
  ⚡ const now :Date = new Date();
  const year = now.getFullYear();
  const month = (now.getMonth() + 1).toString().padStart(2, '0');
  const day = now.getDate().toString().padStart(2, '0');
  const hours = now.getHours().toString().padStart(2, '0');
  const minutes = now.getMinutes().toString().padStart(2, '0');
  const seconds = now.getSeconds().toString().padStart(2, '0');

  return `${year}${month}${day}-${hours}:${minutes}:${seconds}`;
}
```

Ensuite, dans la fonction createBlock, je commence par générer un identifiant unique pour chaque nouveau block grâce à la fonction 'uuidv4'.

Ensuite, je crée un nouveau block avec les informations fournies dans le contenu de la requête POST, à savoir :

```
const newBlock :{...} = {
  id,
  nom: contenu.nom,
  don: contenu.don,
  date,
  hash: "",
};
```

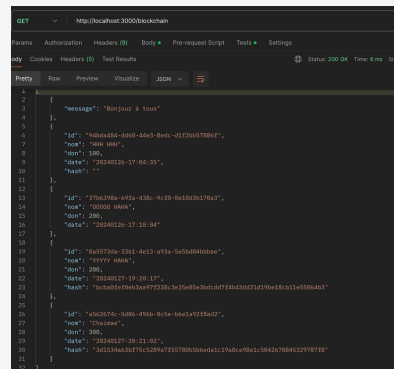
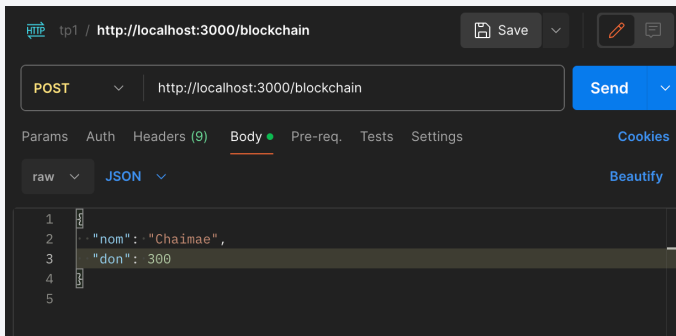
Grâce à la fonction 'findBlocks', on obtient la chaîne existante depuis le fichier blockchain.json. Puis le nouveau bloc est ajouté à la fin de cette chaîne. Le hash du dernier bloc est ensuite calculé grâce à l'algorithme SHA-256, et ce hash est attribué au nouveau block. Enfin, la chaîne est mise à jour et enregistrée dans le fichier blockchain.json et elle est retournée comme réponse.

Voici le fichier blockchain.json avant l'ajout via la requête POST d'un nouveau block, et le fichier après l'ajout d'un block contenant 'Chaimae'.

```
package.json  server.js  blockchainStorage.js  divers.js  blockchain.json
{
  "message": "Bonjour à tous"
},
[
  {
    "id": "94bda484-dd60-44e3-8edc-d1f2b57886f",
    "nom": "HHH HHH",
    "don": 100,
    "date": "20240126-17:04:35",
    "hash": ""
  },
  {
    "id": "37b6398a-69fa-438c-9cf0-0e10d3b170a3",
    "nom": "00000 HAHA",
    "don": 200,
    "date": "20240126-17:10:04"
  },
  {
    "id": "0a5573da-3361-4e13-a93a-5e5bd04bbbee",
    "nom": "YYYYY HAHA",
    "don": 200,
    "date": "20240127-19:20:17",
    "hash": "bcba0fef0eb3aa97f238c3e25e85e3bdcdd7f4b43dd21d19be18cb11e55064b3"
  }
]
```

```
{
  "id": "37b6398a-69fa-438c-9cf0-0e10d3b170a3",
  "nom": "00000 HAHA",
  "don": 200,
  "date": "20240126-17:10:04"
},
{
  "id": "0a5573da-3361-4e13-a93a-5e5bd04bbbee",
  "nom": "YYYYY HAHA",
  "don": 200,
  "date": "20240127-19:20:17",
  "hash": "bcba0fef0eb3aa97f238c3e25e85e3bdcdd7f4b43dd21d19be18cb11e55064b3"
},
{
  "id": "a562674c-5d86-496b-8c5e-b6e1a92f8ad2",
  "nom": "Chaimae",
  "don": 300,
  "date": "20240127-20:21:02",
  "hash": "3d1534a63bf75c5289a7f55780b5b6eda1c19a0ce98e1c5042670845329787f8"
}
]
```

Voici, les requêtes POST et GET de l'ajout d'un block contenant le nom 'Chaimae' sur Postman



Concernant la fonction `findLastBlock()`, elle permet d'obtenir tous les blocs existants de la chaîne. Pour ce faire avec on vérifie que `existingBlock` est un tableau et qu'il contient bien des éléments, s'il contient ces éléments alors on retourne le dernier bloc de la chaîne.

Facilités :

- L'utilisation de `uuidv4` facilite la génération d'identifiants uniques.
- La fonction `findLastBlock` simplifie la manipulation des données dans le fichiers JSON.

Difficultés :

- J'ai eu du mal à comprendre la logique derrière l'ajout du champs hash en fonction du bloc précédent.
- Le calcul du hash SHA-256 est compliqué à comprendre.

Surmonter les difficultés :

- J'ai veillé à implémenter ces fonctions étapes par étapes, soit progressivement.
- J'ai utilisé la documentation avec des exemples, pour comprendre le calcul du hash.