

Rapport TP1

Babylone ISSHAK - 303

Etape 1 :

Pour cet exercice j'ai seulement ajouté dans server.js `console.log("get")` et `console.log("POST")`

J'étais cependant bloquée dans l'utilisation de Postman car on avait utilisé cela qu'une seule fois en cours de programmation avancé. J'ai donc appelé mon professeur afin d'avoir de l'aide dessus.

Etape 2 :

Pour remplir la fonction findBlocks() j'ai cherché dans la documentation sur devdocs afin d'utiliser la lecture (readFile) mais lire spécifiquement des données JSON.

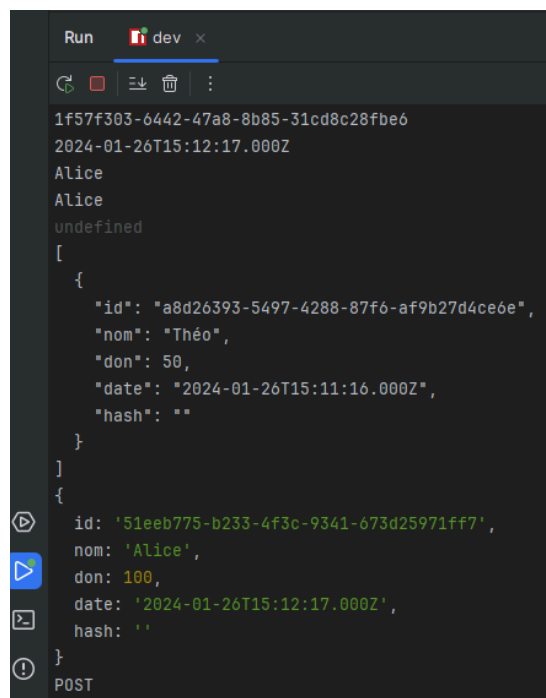
Etape 3 :

Je n'arrivait pas à récupérer les données renvoyées par postman, alors j'ai sollicité l'aide de mon professeur. J'ai alors compris que c'est dans "contenu" (paramètre de la fonction createBlock), que les données sont récupérées. De plus, je me suis aperçue que pour le don, il ne fallait pas appeler contenu.somme, mais contenu.don (c'est pourquoi on peut voir dans la capture d'écran de ma console un endroit où il est écrit "undefined").

Pour l'instant je n'ai uniquement réussi à récupérer et afficher les données que l'on a inscrit dans PostMan, mais il faut mettre ces données dans un block. J'ai les ai donc stockés dans un tableau :

```
79      const newBlock : {date: string, don: ..., hash: string, id: any, nom: ...} = {  
80          id: newId,  
81          nom: contenu.nom,  
82          don: contenu.don,  
83          date: currentDate,  
84          hash: '',  
85      };
```

Résultat console :



```
Run dev x  
1f57f303-6442-47a8-8b85-31cd8c28fbe6  
2024-01-26T15:12:17.000Z  
Alice  
Alice  
undefined  
[  
  {  
    "id": "a8d26393-5497-4288-87f6-af9b27d4ce6e",  
    "nom": "Théo",  
    "don": 50,  
    "date": "2024-01-26T15:11:16.000Z",  
    "hash": ""  
  }  
]  
{  
  id: '51eeb775-b233-4f3c-9341-673d25971ff7',  
  nom: 'Alice',  
  don: 100,  
  date: '2024-01-26T15:12:17.000Z',  
  hash: ''  
}  
POST
```

Afin que ces données soient écrit dans le fichier blockchain, j'ai pensé utilisé findBlocks() que j'ai crée dans l'étape précédente afin de lire le fichier, et ensuite utiliser writeFile(), cependant après avoir longuement lue la documentation, je n'arrivait pas a bien utiliser cette fonction, cela m'affichait des erreurs. J'ai alors copié collé ces erreurs à ChatGPT qui m'a indiquer que les données n'étaient pas parsé en format JSON. De plus il m'a indiqué qu'il fallait utiliser la méthode .push() pour ajouter le nouveau bloc à la fin de la chaîne de blocs existante (récupérée par findBlocks()) :

```
87 console.log(newBlock)
88 blockchain.push(newBlock);
89 await writeFile(path, JSON.stringify(blockchain, {replacer: null, space: 2}), 'utf-8');
90 return blockchain;
```

Le fichier blockchain a maintenant été écrit :

```
package.json  blockchain.json  JS blockchainStorage.js
1  [
2    {
3      "id": "a8d26393-5497-4288-87f6-af9b27d4ce6e",
4      "nom": "Théo",
5      "don": 50,
6      "date": "2024-01-26T15:11:16.000Z",
7      "hash": ""
8    },
9    {
10     "id": "51eeb775-b233-4f3c-9341-673d25971ff7",
11     "nom": "Alice",
12     "don": 100,
13     "date": "2024-01-26T15:12:17.000Z",
14     "hash": ""
15   }
16 ]
```

Etape 4 :

Pour cette étape, je n'ai pas eu le temps de le faire en TP, alors je le fais chez moi ce qui est un peu plus compliqué car j'ai eu du mal à comprendre l'énoncé de cette étape.

J'ai découvert dans la documentation du module hash

<https://nodejs.org/api/crypto.html#class-hash> que pour calculer le hachage, on devait d'abord créer une variable qui nous servira à faire ce calcul en se servant de la méthode createHash('sha256'). Cependant nous devons aussi préciser les données que l'on veut hacher (la string du block précédant), et pour faire cela nous devons utiliser :

.update(donnée)

Enfin, nous devons utiliser : .digest('hex'), car createHash() va nous hacher notre donnée sous forme de bits, or pour pouvoir voir ce hachage dans notre fichier blockchain.json, il faut convertir les bits en hexadécimal.


Ce qui nous donne cette ligne :

```
const hash : Promise<ArrayBuffer> = createHash('sha256').update(previousBlockString).digest({algorithm: 'hex'});
```

Pour ce qui est de récupérer le bloc précédent, j'ai tout simplement utilisé la méthode `findBlocks()` que j'avais déjà créé dans les étapes précédentes, et lue le tableau qu'elle renvoie en sélectionnant la dernière case/ dernier block :

```
return blockchain[blockchain.length - 1];
```

J'avais un problème pour prendre l'attribut "hash" du dernier block, alors j'ai demandé à chatGPT de m'aider dans l'algorithme de ma fonction `createBlock` que j'ai modifié afin d'ajouter la ligne de calcul du hachage. En fait je devais juste faire un `if(lastBlock)` qui utilise la fonction `findLastBlock()` :

```
93 // Pour le hash du bloc précédent s'il existe
94  const lastBlock : {id: string, nom: string, don:... | null} = await findLastBlock();
95 if (lastBlock) {
96     const previousBlockString : string = JSON.stringify(lastBlock);
97     const hash : Promise<ArrayBuffer> = createHash('sha256').update(previousBlockString).digest({ algorithm: 'hex' });
98     newBlock.hash = hash;
99 }
```