

## Développement Avancé : TP1

Etape 1 : état des lieux

Lors d'une requête GET, je vérifie la réception du message "1ère log". Ensuite, je procède à un GET dans Postman, et je confirme la réception du message dans ma console.

```
server.js x package-lock.json package.json block
1      let results
2
3      try {
4        switch (endpoint) {
5          case 'GET:/blockchain':
6            results = await liste(req, res, un
7            console.log("1ère log ",results);
8            break
9          case 'POST:/blockchain':
10           results = await create(req, res)
11           console.log("2ème log", results)
12           break
13         default :
14           res.writeHead(404)
```

HTTP http://localhost:3000/blockchain

GET http://localhost:3000/blockchain

```
(node:24596) ExperimentalWarning:
(Use `node --trace-warnings ...`
{ message: 'Bonjour à tous' }
1ère log
█
```

Etape 2 : apprenons à lire

J'ai créé un dossier nommé "Data" dans lequel j'ai également généré le fichier "blockchain.json".

screen 1 : fichier block.json

screen 2 : path

screen 3 : fonction findBlocks()

screen 4 : message dans la console

```
server.js package-lock.json package.json block.json x
1 {
2   "message" : "Bonjour à tous"
3 }
```

```
7
8 /* Chemin de stockage des blocks */
9 const path :URL = new URL('data/block.json', import.meta.url) // './src/data/blockchain.json'
```

```
export async function findBlocks() :Promise<any> {
  try {
    const blockchainData = await readFile(path, 'utf-8');
    const blockchain = JSON.parse(blockchainData);

    console.log(blockchain);
    return blockchain;
  } catch (error) {
    console.error('Erreur lors de la lecture de la blockchain :', error);
    throw error;
  }
}
```

```
> node --watch src/server.js

(node:26024) ExperimentalWarning: Watch
(Use `node --trace-warnings ...` to show
{ message: 'Bonjour à tous' }
```

### Etape 3 : Une brique après l'autre

```
server.js  block.json  blockchain.js  blockchainStorage.js  divers.js

1+ usages  Laurent GUSTIGNANO *
57 export async function createBlock(contenu) : Promise<...> {
58   try {
59     const blockchainData = await readFile(path, 'utf-8');
60     const blockchain = JSON.parse(blockchainData);
61     const newId : any | string = uuidv4();
62     const currentDate : string = getDate();
63     const newBlock : {} = {
64       id: newId,
65       nom: contenu.nom,
66       don: contenu.don,
67       date: currentDate,
68       hash: '',
69     };
70     blockchain.push(newBlock);
71     await writeFile(path, JSON.stringify(blockchain, {replacer: null, space: 2}, 'utf-8'));
72     return blockchain;
73   } catch (error) {
74     console.error('Erreur lors de la création du bloc :', error);
75     throw error;
76   }
77 }
```

```
JS server.js  {} block.json  JS blockchain.js

1 [
2   {
3     "message": "Bonjour à tous"
4   }
5 ]
```

```
POST  http://localhost:3000/blockchain

Params  Authorization  Headers (8)  Body  Pre-req

none  form-data  x-www-form-urlencoded  raw

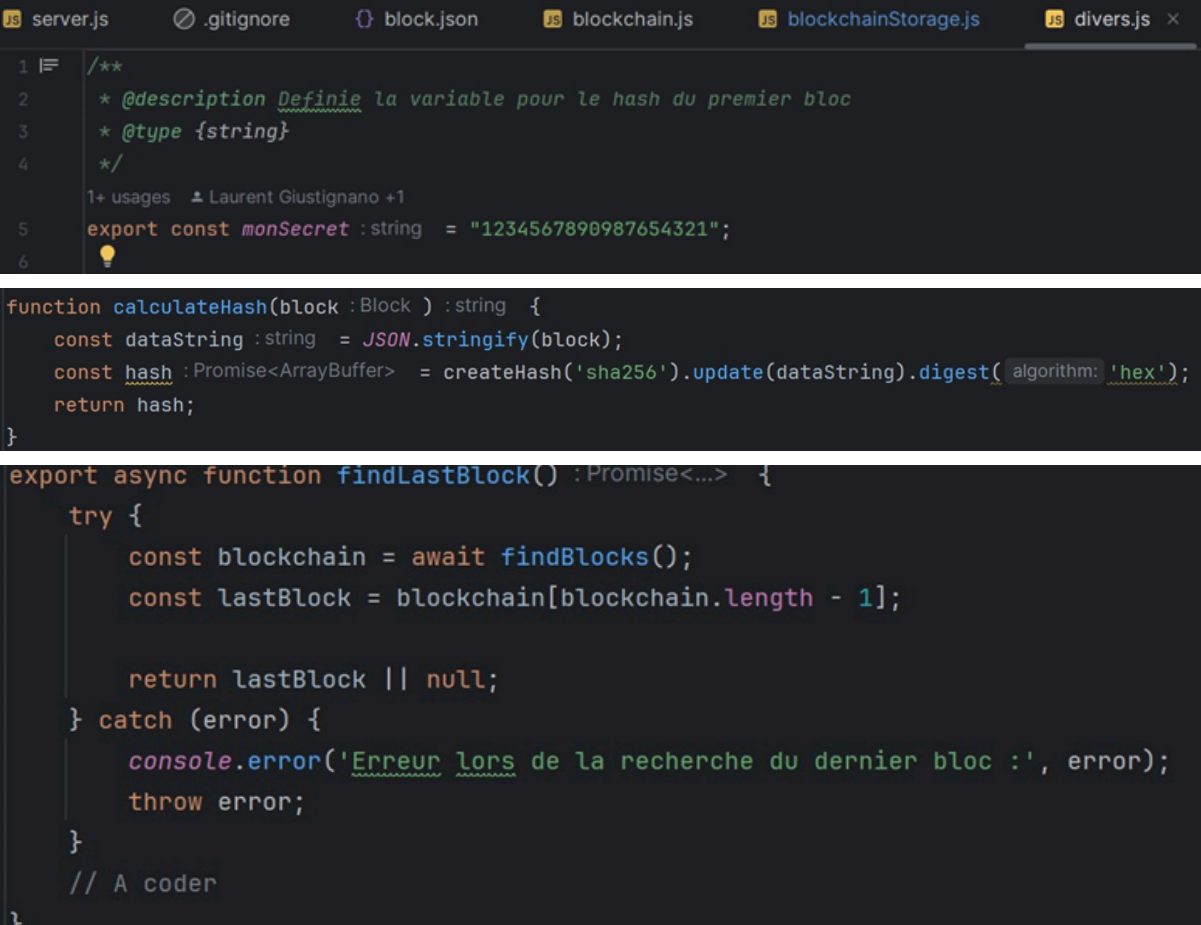
1 {
2   "nom": "brunot",
3   "don": 50
4 }
```

```
server.js  block.json  blockchain.js  blockchainStorage.js  divers.js

1 [
2   {
3     "message": "Bonjour à tous"
4   },
5   {
6     "id": "b101e28e-64bc-41f1-993a-87be89945b8f",
7     "nom": "brunot",
8     "don": 50,
9     "date": "20240126-12:18:14",
10    "hash": ""
11  }
12 ]
```

La fonction `CreatBlock()` récupère la variable `blockchainData` en utilisant la fonction `await readFile()`, suivie d'un `JSON.parse()`. Cependant, plus tard, j'ai réalisé que j'avais déjà effectué cette opération dans `FindBlocks` en utilisant simplement `const blockchain = await findBlocks();`. J'ai ajouté des crochets `[]` dans le fichier `block.json`, car on ne peut pas utiliser `push` tant que ce n'est pas un tableau. Ensuite, j'ai effectué la requête dans le document du TP sur Postman, et j'ai obtenu la réponse dans mon fichier, comme illustré dans la dernière capture d'écran.

Étape 4 : “ Vers l'infini et au-delà ! “



```
1 /**
2  * @description Définie la variable pour le hash du premier bloc
3  * @type {string}
4  */
5 1+ usages  Laurent Giustignano +1
6  export const monSecret : string = "1234567890987654321";

function calculateHash(block : Block ) : string {
  const dataString : string = JSON.stringify(block);
  const hash : Promise<ArrayBuffer> = createHash('sha256').update(dataString).digest( algorithm: 'hex' );
  return hash;
}

export async function findLastBlock() : Promise<...> {
  try {
    const blockchain = await findBlocks();
    const lastBlock = blockchain[blockchain.length - 1];

    return lastBlock || null;
  } catch (error) {
    console.error('Erreur lors de la recherche du dernier bloc :', error);
    throw error;
  }
  // A coder
}
```

```

export async function createBlock(contenu) : Promise<...> {
  try {
    const blockchain = await findBlocks();
    const newId : any | string = uuidv4();
    const currentDate : string = getDate();
    // On trouve le dernier bloc
    const lastBlock : {id: string, nom: string, don: ... | null} = await findLastBlock();
    const newBlock : {...} = {
      id: newId,
      nom: contenu.nom,
      don: contenu.don,
      date: currentDate,
      hash: lastBlock ? calculateHash(lastBlock) : monSecret,
    };
    blockchain.push(newBlock);
    await writeFile(path, JSON.stringify(blockchain, {replacer: null, space: 2}, 'utf-8'));
    return blockchain;
  } catch (error) {
    console.error('Erreur lors de la création du bloc :', error);
    throw error;
  }
}

```

```

[
  {
    "id": "b101e28e-64bc-41f1-993a-87be89945b8f",
    "nom": "brunot",
    "don": 50,
    "date": "20240126-12:18:14",
    "hash": ""
  },
  {
    "id": "24c245ba-43c2-4419-951b-887cbfee1435",
    "nom": "babi",
    "don": 20,
    "date": "20240126-17:16:23",
    "hash": "30bc306b1f749ccc594e33c9d09583c9e6fd3f3d7da4242e1e4db5002c73963e"
  }
]

```

On attribue une chaîne de caractères aléatoire comme valeur de hachage pour le premier bloc. Ensuite, nous créons une fonction qui va hasher le bloc qu'on lui envoie, dans notre cas, ce sera le dernier bloc de la chaîne. Pour obtenir le dernier bloc, nous programmons la fonction `findLastBlock`. Nous modifions ensuite la fonction `createBlock` pour récupérer le bloc renvoyé par `findLastBlock`, le hasher, puis le placer dans la propriété "hash". Une fois que j'ai effectué ma requête POST dans Postman, je vérifie le contenu de mon fichier `block.json`. Je constate que le hachage du dernier bloc a bien été mis à jour.