ÉNONCE TRAVAUX PRATIQUE – SUJET 1

Charité et Blockchain

Exercice 1 :

```
switch (endpoint) {
    case 'GET:/blockchain':
        results = await liste(req, res, url)
        console.log("Fonction GET")
        break
    case 'POST:/blockchain':
        results = await create(req, res)
        console.log("Fonction POST")
        break
    default :
        res.writeHead(404)
}
```

De simple console.log pour vérifier que le serveur http marche correctement. S'il y a un problème, le message n'est pas affiché.

Exercice 2 :

```
export async function findBlocks() {
    try{
        const filePath = new URL(path, import.meta.url);
        const contents = await readFile(filePath, {encoding: 'utf8'});
        return JSON.parse(contents)
    }
    catch (e) {
        console.error("Erreur lors de la lecture :", e);
    }
}
```

Je lis d'abord le fichier JSON et je convertie ensuite le message JSON en JS pour que cela soit lisible.

Exercice 3 :

```
export async function createBlock(contenu) {
    const block = {
        "id": uuidv4(),
        "nom":contenu.nom,
        "don":contenu.don,
        "date": getDate()
    }
    const res = await findBlocks();
    res.push(block);
    console.log(res);
    try{
        const filePath = new URL(path, import.meta.url);
        await writeFile(filePath, JSON.stringify(res), 'utf-8');
        return res;
```

```
    }
    catch (error) {
        console.error("erreur lors de l'écriture du fichier :", error);
    }
}
```

On crée le nouveau block qu'on va ajouter avec les informations demandées. Je récupère les valeurs dans le fichier grâce à la fonction findBlocks() que j'ai codé précédemment. J'ajoute ensuite à la liste mon nouveau block et je l'écris dans le fichier. J'ai des valeurs JS donc je dois le convertir en JSON pour l'écrire dans le fichier JSON. Et ensuite un console.log pour voir que j'ai les bons résultats.


```
[
  {
    id: '59df366e-bfd0-46ff-b387-75eb8729fd4c',
    nom: 'Nom de la personne',
    don: 1234,
    date: '20240129-20:08:41'
  },
  {
    id: '7e44d881-56f8-4f95-83fb-2b09bbb62d11',
    nom: 'Nom de la personne',
    don: 1234,
    date: '20240129-20:08:50'
  },
  {
    id: 'faecf9e3-3414-4f1f-8ef0-9fde7fc0d5e1',
    nom: 'Nom de la personne',
    don: 1234,
    date: '20240129-20:24:10'
  }
]
```


Exercice 4 :

```
export async function createBlock(contenu) {
    const dernierBlock = await findLastBlock();
    const hash =
createHash('sha256').update(JSON.stringify(dernierBlock)).digest('hex');

    const block = {
        "id": uuidv4(),
        "nom":contenu.nom,
        "don":contenu.don,
        "date": getDate(),
        "previousBlockHash": dernierBlock ? hash : null
    }
    const res = await findBlocks();
    res.push(block);
    console.log(res);

    try{
        const filePath = new URL(path, import.meta.url);
        await writeFile(filePath, JSON.stringify(res), 'utf-8');
        return res;
```

```
    }
    catch (error) {
        console.error("erreur lors de l'écriture du fichier :", error);
    }
}
```

```
export async function findLastBlock() {
    let tmp = await findBlocks();
    let res;
    if(tmp.length > 0){
        res = tmp[tmp.length-1];
    }
    else{
        res = null;
    }
    return res;
}
```

Je récupère le dernier block grâce à la fonction findLastBlock() et je le hash pour l'écrire dans mon nouveau block. S'il n'y a pas de dernier block, le hashage n'est pas possible donc le résultat sera nul.