IUT de Paris - Rives de Seine
Université Paris Cité

BUT 3 INFORMATIQUE

Parcours A FI

Réalisations d'applications

RAPPORT TP1

Charité et Blockchain

Prénom et Nom de l'étudiant : Akash Selvaratnam

Groupe : 303

Promotion : 2023-2024

Sommaire

Table des matières

Étape 1 – état des lieux

Au début de la réalisation de ce TP, j'ai pu récupérer le code source de l'exercice via le lien fourni sur le TP et réalisé un clone avec GitHub desktop, cela fait un moment que je n'ai pas réalisé ces manipulations sur GitHub desktop, j'ai pu donc me remémorer de ces différentes manipulations grâce à cette étape.

Ensuite, j'ai pu analyser le code source, avec la fonction createServer qui permet d'exécuter différentes fonctions sur le port 3000, telles qu'obtenir l'ensemble des blocs présent sur le fichier JSON avec la méthode GET et avec le chemin suivant http://localhost:3000/blockchain par exemple.

Les méthodes liste et create présent dans le fichier server.js sont récupérer dans le fichier blockchain.js et ils font appel aux méthodes présente dans le fichier blockchainStorage.js qui est le côté métier de l'application, l'endroit dont l'ensemble des fonctionnalités de l'application sont codées (Par exemple : créer un bloc, chercher un bloc avec son id …).

Pour finir, j'ai vérifié le bon fonctionnement du serveur http en testant les méthodes GET et POST avec Postman.

Étape 2 – apprenons à lire

Pour l'étape 2, j'ai créé un fichier JSON et placer un texte JSON à l'intérieur, j'ai indiqué le chemin qui mène vers ce fichier JSON dans la variable path du fichier blockchainStorage.js.

J'ai réalisé la méthode findBlocks en transformant ma variable path en url puis j'ai utilisé la fonction readFile proposé par la module promises permettant de lire un fichier, cette fonction prend comme argument le chemin du fichier que nous avons transformé en url et retourne l'ensemble des données qu'il a pu lire sur le fichier, que personnellement, j'ai stocker dans une variable et je retourne cette variable avec un Promise.

Ensuite, j'ai pu tester cette fonction avec l'application Postman qui m'a retourné l'ensemble des blocs du fichier.

Pour finir, avec cette étape, j'ai pu apprendre l'utilisation d'une nouvelle module promises, notamment, la fonction readFile, j'ai également compris comment retourner une valeur avec Promise, l'argument resolve s'il s'agit de la réponse correcte à envoyer ou error s'il s'agit d'une réponse incorrecte, nous avons également pu aborder Promise lors du TD avec l'exercice 2.

Étape 3 – une brique après l'autre

Pour l'étape 3, j'ai réalisé la méthode permettant de créer un bloc dans le fichier JSON.

Pour pouvoir réaliser cette étape, j'ai récupéré l'ensemble des valeurs de l'argument de la fonction sous forme de JSON et placer l'ensemble des valeurs dans une variable. Ensuite, j'ai stocké, l'ensemble des valeurs déjà présente dans le fichier JSON dans une variable grâce à la fonction réalisé précédemment, findBlocks. Pour la valeur de l'id, j'ai pu la générer grâce à la fonction uuidv4() du module uuid et j'ai utilisé getDate() pour obtenir la date.

Ensuite, j'inscris les blocs existants dans le fichier JSON ainsi que le nouveau bloc dans le fichier JSON grâce à la fonction writeFile proposé par le module promises. J'ai également ajouté les blocs existant dans le fichier JSON, car, si cela n'est pas réalisé alors cela supprimera l'ensemble des valeurs existante dans le fichier et ne gardera que la nouvelle valeur. J'ai retourné les donner à ajouter avec un Promise.

Ensuite, j'ai pu tester cette fonction avec l'application Postman qui m'a retourné en réponse les valeurs que j'ai ajoutées sur le fichier.

À travers la réalisation de cette étape, j'ai pu prendre connaissance de la librairie uuidv4() permettant de générer un id ainsi que la fonction writeFile de la librairie promises qui a pour but d'écrire des données à l'intérieur d'un fichier. J'ai également pu retravailler avec le spread operator qui ma permit dans cette étape, de pouvoir réécrire les informations déjà existantes sur le fichier puis d'ajouter les nouvelles informations.

Étape 4 – « Vers l'infini et au-delà ! »

Pour l'étape 4, j'ai réalisé la fonction findLastBlock permettant de retourner le dernier bloc d'un fichier JSON, si le fichier est vide alors il faut retourner null et j'ai ajouté ce dernier bloc sous forme de hachage dans les données à ajouter dans la fonction createBlock.

À travers cette étape, j'ai pu apprendre l'utilisation d'une nouvelle librairie crypto, plus particulièrement de sa fonction createHash qui m'a permis de transformer le dernier bloc contenu dans le fichier JSON sous forme de hachage.

Pour aller plus loin…

J'ai également réalise la fonction findBlockByID permettant de retourner un bloc en particulier à partir de son id.

Pour réaliser cette fonction, j'ai mis en place un nouveau case du switch sur le serveur avec la méthode GET et le chemin suivant http://localhost:3000/findBlockByID. Le serveur appelle la fonction findBlockByID présent dans le fichier blockchain.js, qui permet de faire appel à la fonction findBlock avec comme argument le body de la requête HTTP sous format JSON, cette méthode findBlock est présent dans le fichier blockchainStorage.js, cette fonction récupère l'ensemble des données présentes dans le fichier JSON puis de comparer l'id de chaque bloc du fichier JSON avec l'id fourni en paramètre de la fonction, si l'id correspond à un bloc du fichier JSON alors, on renvoie le bloc correspondant sinon envoie un retour JSON permettant de spécifier que l'id fourni est inconnu.

À travers cette étape, j'ai pu travailler avec le fichier serveur.js ainsi que blockchain.js contrairement aux autres étapes et cela m'a permis de mieux comprendre l'architecture du code fourni.