Liaison avec un serveur nodeJs

Prérequis

- JRE Java
- nodeJS
- npm

Architecture

Il n'existe pas de nœud Kalima Javascript, cependant il existe une librairie JS permettant de faire la liaison entre un programme JS (un serveur nodeJS par exemple) et un nœud java spécifique (KalimaNodeAdmin).

Dans ce tutorial nous allons donc voir comment développer un serveur nodeJS qui utilise la librairie JS Kalima, puis comment configurer et lancer la partie Java puis le serveur nodeJS.

Enfin le serveur nodejs sera testé via une requête curl. Vous pouvez vous référer à la documentation de l'api REST pour tester d'autres requêtes : https://doc.kalimadb.com/APIs/Rest/restAPI.htm.

NodeJS

Pour commencer, créez un nouveau dossier pour votre projet.

Librairie Kalima

Dans votre projet, il faut copier la librairie Kalima. Créer par exemple un dossier libs dans votre projet dans lequel vous copié le dossier kalima qui se trouve dans etc/lib/JS.

app.js

Créez ensuite un fichier js, qui sera le point d'entrée de votre serveur, dans notre cas app.js :

```
require("dotenv").config();

var express = require("express");
var bodyParser = require("body-parser");
fs = require("fs");
var kalimaApiRouter = require("./routes/kalimaApiRouter");
var app = express();

app.use(bodyParser.json());
app.use(express.json());
app.use(express.urlencoded({ extended: false }));

app.listen(process.env.PORT);
console.log(`app listening on port ${process.env.PORT}`);

app.use("/api", kalimaApiRouter);

module.exports = app;
```

Comment on peut le voir, on fait appel à kalimaApiRouter, que l'on va créer juste après.

De plus le code fait appel a un fichier d'environnement que nous configurerons plus tard.

kalimaApiRouter.js

Il faut ensuite ajouter le router (kalimaApiRouter.js) qui va faire appel à la librairie Kalima. Pour cela, vous pouvez créer un dossier routes dans votre projet, et y ajouter un nouveau fichier kalimaApiRouter.js :

```
var express = require("express");
var router = express.Router();
var kalimaApi = require("../libs/kalima/kalimaApi");
var kalimaEventsApi = require("../libs/kalima/kalimaEventsApi");

kalimaApi.init(process.env.FILES_PATH);
kalimaEventsApi.init(process.env.FILES_PATH);

router.get("/events", function (req, res) {
  kalimaEventsApi.addClient(req, res);
});

router.get("/*", function (req, res) {
  kalimaApi.get(req, res);
});

router.delete("/*", function (req, res) {
  kalimaApi.delete(req, res);
});

router.post("/*", function (req, res) {
  kalimaApi.post(req, res);
});

module.exports = router;
```

Notez que cet exemple reste le plus simple possible et n'offre aucune sécurisation d'accès. Libre à vous d'y ajouter par la suite une authentification par exemple.

Installation des dépendances

Notre code dépend de quelques librairies externes qu'il faut installer via npm :

```
npm install dotenv
npm install express
npm install btoa
```

Configurations

Java

Comme nous l'avons décrit plus haut, notre serveur nodeJS va communiquer avec un nœud java : KalimaNodeAdmin. Comme tout nœud Kalima Java, il sera lancé avec un paramètre : le chemin vers un fichier de config.

Créer donc un fichier de config, node.config par exemple :

```
SERVER_PORT=9100
FILES_PATH=/home/rcs/jit/KalimaNodeJS/
# CHANGE IT WITH THE SERIALID GRANTED AFTER ADMINISTRATIVE VALIDATION
SerialId=KalimaNodeAdmin
PRIVACHAIN=org.kalima.tuto
```

- SERVER_PORT ➔ Choisissez un port libre
- FILES_PATH ➔ Vous pouvez indiquer n'importe quel chemin. Ce dossier contiendra plusieurs fichiers nécessaires à l'application, ainsi que les logs du nœud java
- SerialId ➔ ID permettant l'autorisation de votre nœud sur la blockchain (si vous avez utilisé le formulaire d'inscription aux tutoriaux, vous disposez de 10 serialId reçus par mail)
- PRIVACHAIN ➔ Permet de choisir la blockchain sur laquelle le nœud sera connecté, dans le cadre des tutoriaux, laisser tel quel

Le jar KalimaNodeAdmin.jar se trouve dans etc/lib.

nodeJS

Notre serveur nodeJS va également lire un fichier de configuration. Créer donc un fichier .env :

```
PORT=9000
FILES_PATH=/home/rcs/jit/KalimaNodeJS/
```

- PORT ➔ Port d'écoute de votre serveur nodeJS
- FILES_PATH ➔ Doit impérativement être identique à celui de la partie JAVA

Exécution

Commencez par lancer la partie Java :

```
java -jar KalimaNodeAdmin.jar node.config
```

Dans une autre console, lancer votre serveur nodeJS :

```
node app.js
```

Dans une troisème console, testez le tout via une requête curl :

```
curl http://localhost:9000/api/cache/list
```

Vous devez obtenir en retour un JSON contenant la liste des adresses sur lesquelles votre nœud Java est autorisé.