Mouro

1. Descripción

   Mouro es un gestor de base de datos, el cual se comunica con una base de datos ‘sqlite’ para almacenar las credenciales emitidas por el didi-server y los distintos issuers, permite el acceso a los usuarios de didi a sus credenciales y verificar el estado de los mismos.

2. Ejecución
   a. Instalar Node:
      Este módulo utiliza npm como gestor de paquetes, el cual viene integrado con Node: https://nodejs.org/en/
   b. Instalar y ejecutar cliente de swarm (opcional):
      Este módulo utiliza swarm para generar y almacenar los backups: https://swarm-guide.readthedocs.io/en/latest/node_operator.html#download-pre-compiled-swarm-binaries
   c. Instalar dependencias:
      Abrir una terminal en la carpeta principal del módulo (a la altura del archivo ‘package.json’) y ejecutar el comando **‘npm install’**
   d. Ejecutar mouro:
      A la misma altura, ejecutar el comando **‘ $ENV_VARS npm run start’**, donde $ENV_VARS son las distintas variables de entorno del módulo.
      Ej:

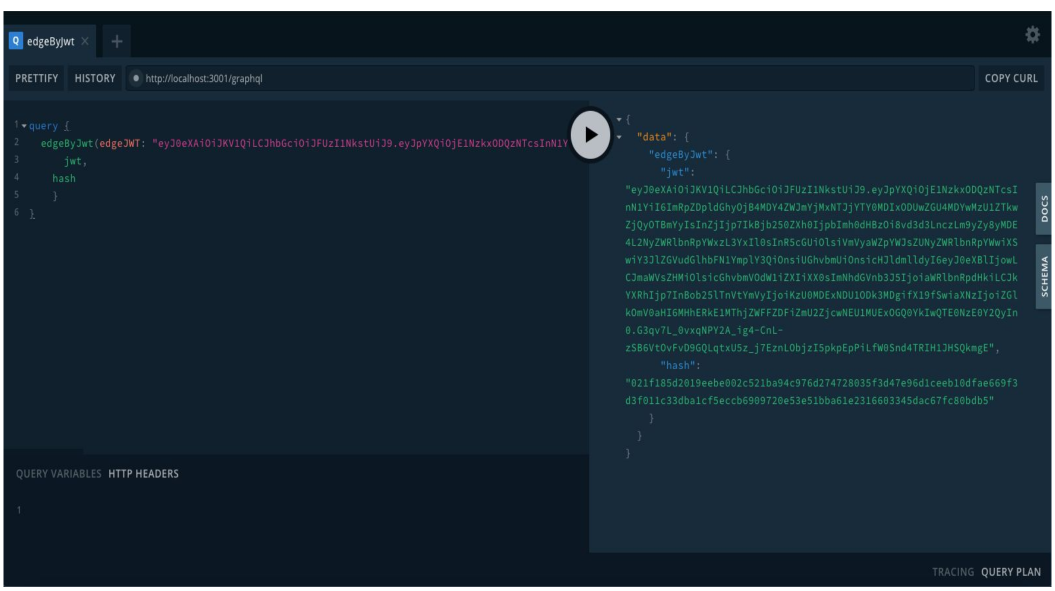
      ‘PORT=3001 SWARM_URL=*http://localhost:8500* npm run start’

e. Variables de entorno:

Nombre	Obligatorio	Descripción	notas	ej
DIDI_SERVER_DID	Si	*Did correspondiente al didi-server*	*La mayor parte de los métodos solo pueden ser llamados por el didi-server*	*0xDFA518ceaEd1bfe6f704E51A18d4bB0A14714cd2*
PORT	Si	*puerto en que corre*	-	*3001*
SWARM_URL	No	*dirección en la que está corriendo swarm*	*Si no está definido, no se generan backups*	*http://192.168.2.113:8500*
SQLITE_FILE	No	*el nombre del archivo*	*Si no está definido, se genera un archivo por usuario, si lo está, se guarda todo ahí*	*database*
BLOCK_CHAIN_URL	Si	*Url donde se encuentra corriendo la blockchain*	-	*https://mainnet.infura.io/v3/21d2ee9de6484541b66eb06942ceb3c7*
BLOCK_CHAIN_CONTRACT	Si	*Identificador del contrato de ethr-did-registry dentro de la blockchain*	-	*0xdca7ef03e98e0dc2b855be647c39abe984fcf21b*

3. Comandos

Una vez iniciado el programa, se mostrará la url con la que se pueden realizar querys sobre Mouro de forma directa:



Los comandos existentes son los siguientes:

Nota: Algunas de las llamadas requieren de la generación de un token vacío firmado por el DID del usuario que realiza la llamada, este se ingresa en la sección “HTTP HEADERS” y tiene el siguiente formato:

```
{  "Authorization": "Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJFUzI1NkstUiJ9.eyJpYXQiOjE1ODAyMjY0NzEsImV4c
CI6MTU4MDIyNjk3MS4xMTIsImlzcyI6ImRpZDpldGhyOjB4REZBNTE4Y2VhRWQxYm
ZlNmY3MDRFNTFBMThkNGJCMEExNDcxNGNkMiJ9.NdihoBy4uEMsCLaitRIETQ-fn
B2SGJLyxfApked_42T9lZHJkIcGRQmxweOaer-UuG3A4R7-LOQYN76MivlQCgA" }
```

me: Retorna la info del usuario a partir del token.
Ej:

```
query {
        me() {
                did
         }
}
```

findEdges: Retorna los certificados del dueño del token.
Ej:

```
query {
        findEdges(toDid: “did:ethr:....”) {
                jwt
        }
}
```

Hash: Retorna el hash a utilizarse para la obtención del último backup usando swarm.
Ej:

```
query {
        hash(did: “did:ethr:....”)
}
```

edgeByJwt: Obtiene un certificado a partir de su jwt.
Ej:

```
query {
        edgeByJwt(edgeJWT: “...”, did: “did:ethr:....”) {
                jwt,
                hash
        }
```

addEdge: Guarda un nuevo certificado.
Ej:

```
query {
        addEdge(edgeJWT: “...”, did: “did:ethr:....”) {
                jwt,
                hash
        }
}
```

4. Backup

Si se recibe la variable de entorno “SWARM_URL” se generará backups en la url especificada por ella en cada modificación de la base de datos, generandose un nuevo hash cada vez que se agrega o revoca un certificado. Estos hashes se almacenan tanto en didi-server como mouro y pueden recuperarse con la query desde mouro:

```
query {
        hash(did: “did:ethr:....”)
}
```

O sobre la base de datos de didi-server con la query:

```
db.getCollection('users').find({did: "did:ethr:..."}, {backupHash: 1})
```

Una vez obtenido ese hash, puede recuperarse el último backup utilizando el comando:

```
curl $SWARM_URL/bzz-raw:/$HASH/ >> $NOMBRE_DE_ARCHIBO
```

Donde **$SWARM_URL** es la url donde se encuentra corriendo swarm, **$HASH** es el hash obtenido en el paso anterior y **$NOMBRE_DE_ARCHIBO** es el nombre de archivo al que bajar el backup.