Authorization, Authentification et CORS

Dans une api, pour se connécter on utilise un système de "jeton" (token). L'idée est la suivante :

1. John demande un jeton de connexion à l'api
2. L'api lui envoi un `JWT`
3. John conserve ce jeton
4. Lorsque john veut accéder à des données de l'api, il envoie son jeton dans le header HTTP : `Authorization`
5. L'api vérifie le jeton et lui donne accès à la resource

Éxemple de requête HTTP entre « John » et un « API Rest »

Création d'un « compte »

Tout d'abord, john doit avoir un compte sur l'application REST :

```
POST http://mon.api.com/users
Content-Type: application/json

{
  "email": "john@mail.com",
  "password: "john1234"
}
```

L'api créé le compte et retourne les données de john

```
HTTP/1.1 201 Created
Content-Type: application/json

{
  "_id": "fkdshfslkfshdfksdhlfk",
  "email": "john@mail.com",
  "password": "dkfhsdlfhslksdhfldskhdflkh"
}
```

Demander un jeton de connéxion

Pour cela john, envoi la requêtes suivante :

```
POST https://mon.api.com/token
Content-Type: application/json

{
  "email": "john@mail.com",
  "password": "john1234"
}
```

L'api vérifie les informations de john, si tout est correct, elle retourne son jeton :

```
HTTP/1.1 201 Created
Content-Type: application/json

{
  "token": "lkfhqsdlfjhslqsdlfqsdg_"-O_64o3864oOEHFLFSDJKHFSKDJ'Ç_GFKJGGKJG"
}
```

S'authentifier sur l'API Rest

Pour cela, john vas devoir envoyer son « jeton » lorsqu'il réalise un requête à des resources :

```
GET https://mon.api.com/announces?limit=20&user=rose
Content-Type: application/json
Authorization: Bearer <token>
```

L'api reçoit le jeton de john, il vérifie que ce jeton soit correct, si c'est la cas il répond :

```
HTTP/1.1 200 Ok
Content-Type: application/json

[
  // ... annonces de rose
]
```

Utiliser `fastify-jwt` afin de créer et valider des jetons

1. Installer

Pour cela, dans un terminal lancer la commande :

```
npm i @fastify/jwt
```

2. Enregistrer le plugin dans notre application

Dans le fichier `src/index.ts` (notre fichier principal), nous devons enregistrer ce tout nouveau plugin :

```
import fastify from 'fastify'
import fastifyJwt from '@fastify/jwt'

// Création de l'application fastify
const app = fastify()

// On enregistre le plugin nous permettant de gérer les jetons
app.register(fastifyJwt, {
  // On spécifie le secret utilisé pour générer et valider nos jetons
  secret: 'dkhfsdkfhslkfhsflkdh',
})
```

3. Générer un jeton

Pour générer un jeton de connexion `jwt` :

```
const token = app.jwt.sign({
  // Les données qui seront crypté dans notre jeton
  // (généralement il s'agot de l'identifiant de de certaines
  // informations de l'utilisateur. Attention à ne rien mettre de très sensible)
  _id: 'sdlkfhsdlkfhsdlkfh',
  email: 'john@mail.com',
})
```

4. Valider un jeton

Pour valider un jeton, rien de plus simple :

```
await request.jwtVerify()
```

> Il est aussi possible de récupérer le contenue du jwt après la connexion avec : `request.user`