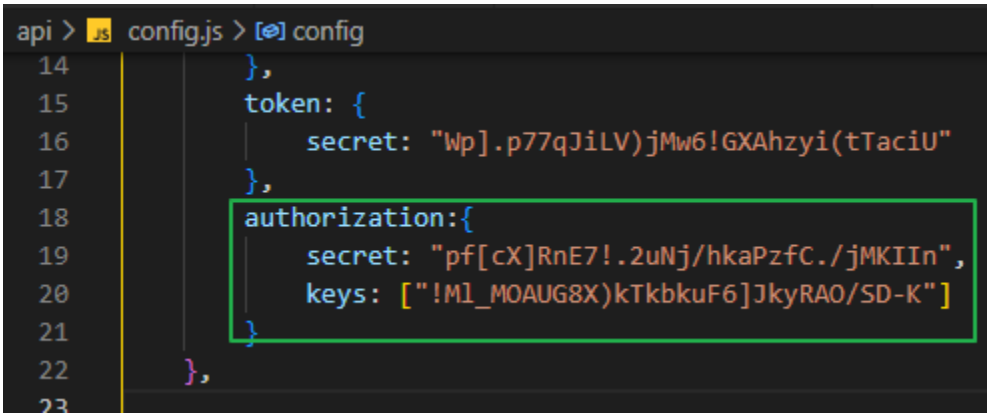
React, Node et MySql Login (Partie 8)

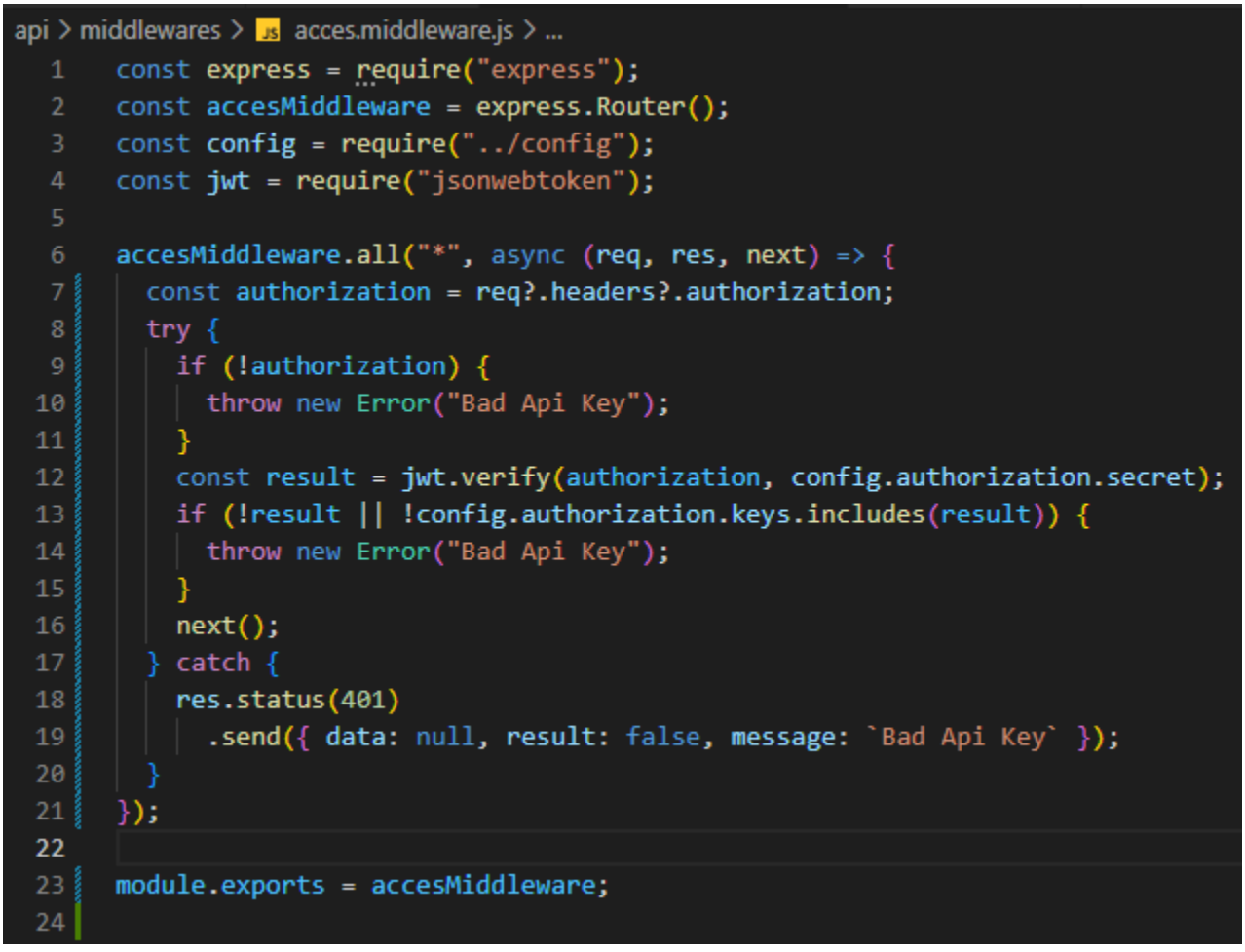
Création d'un middleware vérifiant la clé d'accès à l'API (côté Node)

Étape 17 : Création du middleware de contrôle d'accès à l'API
Nous commençons par ajouter la config nécessaire pour le middleware



Ln 19 : secret servira à décoder les token d'autorisation
Ln 20 : keys contient toutes les clé d'accès à l'API (une seule ici)

Puis nous créons le fichier acces.middleware.js dans un sous-répertoire middlewares de src



Ln 2 : Import du router d'express
Ln 6 : On capte toutes les routes et toutes les méthodes (GET, POST, …) - Attention au 3ème paramètre next, qui permet d'autoriser la poursuite du code (passage au middleware suivant)
…

Ln 7 : On récupère le header Authorization
Ln 9 à 11 : S'il n'y a pas de header Authorization, on déclenche une erreur
Ln 12 : Vérification du token d'autorisation
Ln 13 à 15 : Si le token ne correspond à aucune clé stockée en config, on déclenche une erreur
Ln 16 : Si tout est ok, on passe au middleware suivant
Ln 17 à 20 : En cas d'erreur déclenchée on renvoie un réponse HTTP avec le status code 401

Enfin, nous utilisons le middleware dans index.js

index.js > ...

```
9   const cookieParser = require("cookie-parser");
10  app.use(cookieParser());
11
12  const { query } = require("./api/services/database.service");
13  const bcrypt = require("bcrypt");
14  const config = require("./api/config");
15  const jwt = require("jsonwebtoken");
16
17  const accesMiddleware = require('./api/middlewares/acces.middleware')
18  app.use(accesMiddleware);
19  // app.all("*", async (req, res, next) => {
20  //   const authorization = req?.headers?.authorization;
21  //   try {
22  //     if (!authorization) {
23  //       throw new Error("Bad Api Key");
24  //     }
25  //     const result = jwt.verify(authorization, config.authorization.secret);
26  //     if (!result || !config.authorization.keys.includes(result)) {
27  //       throw new Error("Bad Api Key");
28  //     }
29  //     next();
30  //   } catch {
31  //     res.send({ data: null, result: false, message: `Bad Api Key` });
32  //   }
33  // });
34
```

Les lignes 17 et 18 remplacent les lignes 19 à 33

<u>Étape 18 :</u> Essayer de créer un routeur pour l'authentification (même principe que pour l'étape 17) afin d'extraire les routes GET/auth et POST/login d'index.js
Commencez par créer un fichier auth.router.js dans un sous-répertoire routers de src, puis utilisez le dans index.js

api > routers > auth.router.js > ...

```
1  const express = require("express");
2  const authRouter = express.Router();
3
4  //TODO GET/auth et POST/login
5
6  module.exports = authRouter;
```