OpenClassrooms - Développeur Web - P6
Construire une API sécurisée

Contexte

- Mon client souhaite développer une application web de critique sur les sauces piquantes.
- Mission : Concevoir une API pour cette Application Web

Mission

À partir du front-end mis à ma disposition :

- Réaliser une API respectant les spécifications fournies
- Système d'inscription et de connexion
- Les utilisateurs doivent pouvoir publier des sauces, ajouter des images stockées sur leur ordinateur, éditer leurs posts, ainsi que les supprimer
- Fonction permettant le liker / disliker les sauces
- Exigence en matière de sécurité à respecter

Présentation

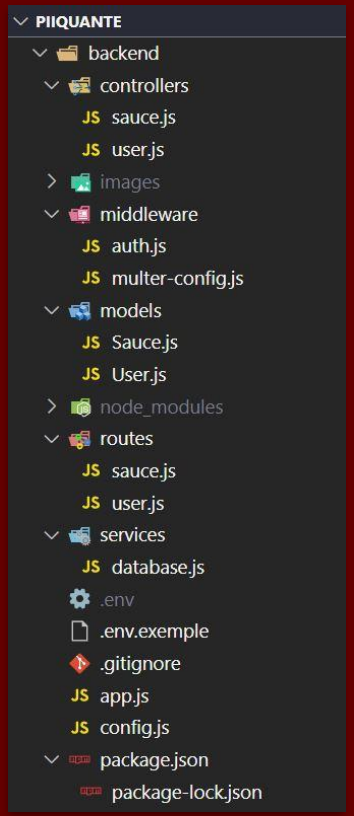
1 - Démonstration des fonctionnalités
2 - Présentation détaillée du code
3 - Bilan & Axes d'amélioration en termes de sécurité

1 - Test du site

2 - Explication du code

Mise en place du projet

Structure du backend




```
"dependencies": {
  latest
  "bcrypt": "^5.0.1",                          Hashage password
  latest
  "dotenv": "^16.0.2",                         Données sensibles
  latest
  "express": "^4.18.1",                        Framework
  latest
  "helmet": "^6.0.0",                          Pack Sécurité (XSS...)
  latest
  "jsonwebtoken": "^8.5.1",                    Authentification
  latest
  "mongoose": "^6.6.0",                        Librairie MongoDB
  latest
  "mongoose-unique-validator": "^3.1.0",
  latest prerelease ^1.4.5-lts.1
  "multer": "^1.4.5-lts.1",                    Upload images
  latest
  "nodemon": "^2.0.19"                         Actualiser changement
}
```

package.json : Librairies utilisées

```
[nodemon] restarting due to changes...
```

console.log

Application Express

Application | Imports, Express, Port

app. JS

```
//! IMPORT
//* IMPORT LIBRAIRIE
require('dotenv').config() // DOTENV (SECURITE)
const express = require("express") // EXPRESS
const helmet = require('helmet') // HELMET (SECURITE)
const path = require('path') // ACCES PATH SERVEUR (ROUTE /IMAGES POUR MULTER)

//* IMPORT CODE
require('./services/database') // DATABASE (CONNEXION)
const { port, errorHandler } = require('./config') // CONFIG : PORT, ERREURS
const userRoute = require('./routes/user') // ROUTES USER
const sauceRoute = require('./routes/sauce') // ROUTES SAUCE

//*-------------------------------------------------------------------------

//! APP + CONFIG
//* EXPRESS
const app = express() // APPLICATION EXPRESS

//* CONFIG : PORT, ERREURS
app.on('error', errorHandler)
app.on('listening', () => {
  const address = app.address();
  const bind = typeof address === 'string' ? 'pipe ' + address : 'port ' + port
  console.log('Listening on ' + bind);
})
```

Imports Librairie

Imports Code

Déclaration Express

Gestion des erreurs
+ Normalisation du port pour stabilité

config. JS

```
//! GESTION DU PORT :
//* "normalizePort" => RENVOIE UN PORT VALIDE,
//* QU'IL SOIT FOURNI SOUR LA FORME D'UN NUMERO OU D'UNE CHAINE
normalizePort = val => {
  const port = parseInt(val, 10)
  if (isNaN(port)) { return val }
  if (port >= 0) { return port }
  return false
}

//! EXPORT DU PORT QUI EST PAR DEFAULT SUR 3000
exports.port = normalizePort(process.env.PORT || '3000')

//*-------------------------------------------------------------------------

//! GESTION DES ERREURS :
//* "errorHandler" => RECHERCHE LES DIFFERENTES ERREURS ET LES GERES DE MANIERE APPROPRIEE
exports.errorHandler = error => {
  if (error.syscall !== 'listen') { throw error }
  const address = server.address()
  const bind = typeof address === 'string' ? 'pipe ' + address : 'port: ' + port
  switch (error.code) {
    case 'EACCES':
      console.error(bind + ' requires elevated privileges.')
      process.exit(1)
      break
    case 'EADDRINUSE':
      console.error(bind + ' is already in use.')
      process.exit(1);
      break
    default: throw error
  }
}
```

Renvoie un port valide

Gestion des erreurs

Gain de stabilité = Plus facile à déboguer

Application | Database

app.JS

```
//* IMPORT LIBRAIRIE
require('dotenv').config() // DOTENV (SECURITE)
//* IMPORT CODE
require('./services/database') // DATABASE (CONNEXION)
```

Dotenv

Database

.env

```
# IDENTIFIANT BASE DE DONNEES
MONGODB_LOGIN=valentin
MONGODB_PASSWORD=hLJqdpRId14wRBLz
```

Variables d'environnement connexion Database

database.JS

```
//* IMPORT LIBRAIRIE MONGOOSE
const mongoose = require('mongoose')

//* VARIABLES D'ENVIRONEMENTS
const password = process.env.MONGODB_PASSWORD
const login = process.env.MONGODB_LOGIN

//* CONNEXTION A LA BASE DE DONNEE MONGODB AVEC L'ADRESSE
const uri = `mongodb+srv://${login}:${password}@
sauces.erngyxz.mongodb.net/?retryWrites=true&w=majority`

//* RESUTAT CONNEXION A MONGODB
mongoose
  .connect(uri)
  .then(() => console.log('Connexion à MongoDB réussie !'))
  .catch((err) => console.error('Connexion à MongoDB échouée !' + err))

//* EXPORT
module.exports = { mongoose }
```

Mongoose

Variables environnement

Inclusion variables

Connexion

Application | Middleware et lancement

app.JS

```
//! MIDDLEWARE
//* HELMET => PROTEGE L'APPLICATION DE CERTAINES VULNERABILITES EN CONFIGURANT DE MANIERE APPROPRIEES DES HEADERS HTTP
app.use(helmet({ crossOriginResourcePolicy: { policy: "same-site" } }))

//* PARAMETRAGE DES HEADERS HTTP
app.use((req, res, next) => {
  // ACCEDER A NOTRE API DEPUIS N'IMPORTE QUELLE ORIGINE
  res.setHeader('Access-Control-Allow-Origin', '*')
  // AJOUTER LES HEADERS MENTIONNEES AUX REQUETES ENVOYEES VERS NOTRE API
  res.setHeader('Access-Control-Allow-Headers', 'Origin, X-Requested-With, Content, Accept, Content-Type, Authorization')
  // ENVOYER DES REQUETES AVEC LES METHODES MENTIONNEES
  res.setHeader('Access-Control-Allow-Methods', 'GET, POST, PUT, DELETE, PATCH, OPTIONS')
  next();
})

//* PARSER =>  ANALYSE LE CORPS D'UNE REQUETE HTTP, ASSEMBLE LES DONNEES, CREE UN OBJET BODY EXPLOITABLE
app.use(express.json())

//! ROUTES
app.use(userRoute)
app.use(sauceRoute)

//! CHEMIN IMAGE
app.use('/images', express.static(path.join(__dirname, '/images')))

//! LANCEMENT SUR LE PORT
app.listen(port, () => console.log("Listening on port : " + port))
```

Helmet : Package sécurité

Configuration headers

Origines autorisées

Ajout headers aux requêtes

Méthodes autorisées

Parser

Routes

Route images statique

Lie application au port
Écoute les connexion

console.log

```
[nodemon] starting `node start app.js`
Listening on port : 3000
```

user

User | Model + Router

models/user.JS

```
//* IMPORT MONGOOSE
const mongoose = require('mongoose')

//* IMPORT DE LA VALIDATION UNIQUE DE MONGOOSE
const uniqueValidator = require('mongoose-unique-validator')

//* CREER SCHEMA UTILISATEUR
const userSchema = new mongoose.Schema ({
  email: { type: String, required: true, unique: true },
  password: { type: String, required: true }
})

//* VERIFICATION DU SCHEMA UTILISATEUR UNIQUE AVEC LE VALIDATEUR MONGOOSE
userSchema.plugin(uniqueValidator)

//* EXPORTATION DU MODELE UTILISATEUR
module.exports = mongoose.model('User', userSchema)
```

Schema User

Ajout plugin

routes/user.JS

```
//* IMPORT EXPRESS
const express = require('express')

//* IMPORT LE CONTROLE DES UTILISATEURS
const { signup, login } = require("../controllers/user")

//* DECLARE ROUTER D'EXPRESS
const router = express.Router()

//* CONTROLE LA CREATION D'UN NOUVEL UTILISATEUR
router.post('/api/auth/signup', signup)

//* CONTROLE L'EXISTANCE D'UN UTILISATEUR DANS LA BASE DE DONNEE
router.post('/api/auth/login', login)

//* EXPORTATION DES ROUTES
module.exports = router
```

Imports : middleware, controller

Router

Routes

User controller | Imports + Inscription

controllers/user. JS

```
//* IMPORT BCRYPT                          Password
const bcrypt = require('bcrypt')

//* IMPORT JSONWEBTOKEN                    Token
const jwt = require('jsonwebtoken')

//* IMPORT DES MODELES UTILISATEURS        Model
const User = require('../models/User')

//* EXPORT DONNEES
module.exports = { signup, login }
```

Database : Objet User

```
_id: ObjectId("630c7eae14d20fae25d13441")
email: "user1@test.com"                    Password protégé
password: "$2b$10$XqouryhR0l0xOk18sERTzeDMOy6ge5w.6vpwL5tOBraInns1bHJOS"
```

controllers/user. JS

```
//* CREER UN NOUVEL UTILISATEUR
function signup (req, res, next) {
  //* HASHER LE PASSWORD 10 FOIS AVEC BYCRYPT
  bcrypt.hash(req.body.password, 10)          Hashage password
    //* TRANSMETTRE LE MAIL ET MDP A UN OBJET UTILISATEUR
    .then(hash => {
      //* CREER UN NOUVEL UTILISATEUR
      const user = new User({
      //* EMAIL DE LA REQUÊTE
      email: req.body.email,
      //* MDP HASHER                          Objet User
      password: hash
      })
      //* SAUVEGARDE DANS LA DATABASE AVEC LA METHODE ".save"
      user.save() Enregistrement
        .then(() => res.status(201).json({ message: 'Utilisateur créé !' }))
        .catch(error => res.status(500).json({ error }))
    })
    .catch(error => res.status(501).json({ error }))
}
```

User controller | Connexion

controllers/user.JS

```
//* CONNECTER UN UTILISATEUR EXISTANT
function login (req, res, next) {
  //* CHERCHER EMAIL DE L'UTILISATERUR DANS LA DATABASE AVEC LA METHODE ".findOne"
  User.findOne({ email: req.body.email })
    .then(user => {
      if (user === null) {
        //* SI AUCUNE CORRESPONDANCE, ECHEC DE CONNEXION ET ON RENVOI UN MSG
        res.status(401).json({ message: 'L'adresse e-mail que vous avez saisie n'est associée à aucun compte' })
      } else {
        //* SINON ON COMPARE LE MDP DONNEES AVEC CELUI DE LA DATABASE AVEC LA METHODE ".compare" DE BYCRYPT
        bcrypt.compare(req.body.password, user.password)
          .then(valid => {
            if (!valid) {
              //* SI IL EST INVALIDE, ECHEC DE LA CONNEXION (401)
              res.status(401).json({ message: 'Paire identifiant/mot de passe incorrecte' })
            } else {
              //* SINON IL EST VALIDE, POURSUIT LA CONNEXION (201)
              res.status(200).json({
                userId: user._id,
                //* ENVOI UN TOKEN D'AUTHENTIFICATION AVEC LA METHODE ".sign" DE JASONWEBTOKEN
                token: jwt.sign(
                  //* ARGUMENTS : userId, token dans ".env", durée de validité
                  { userId: user._id },
                  process.env.JWT_TOKEN,
                  { expiresIn: '24h' }
                )
              })
            }
          })
          .catch(error => res.status(500).json({ error }))
      }
    })
    .catch(error => res.status(501).json({ error }))
}
```

Correspondance Email : requête / database

Correspondance Password : requête / database

Token web d'authentification

.env

```
# PASSWORD JWT
JWT_TOKEN=45af32ec-0cd5-40ac-9227-60f559f337e2
```

sauce

Sauce | Model + Router

models/Sauce.JS

```
//* IMPORT MONGOOSE
const mongoose = require('mongoose')

//* CREER SCHEMA SAUCE
const sauceSchema = mongoose.Schema({
  userId: { type: String, required: true},
  name: { type: String, required: true},
  manufacturer: { type: String, required: true},
  description: { type: String, required: true},
  mainPepper: { type: String, required: true},
  imageUrl: { type: String, required: true},
  heat: { type: Number, required: true},
  likes: { type: Number, required: true},
  dislikes: { type: Number, required: true},
  usersLiked: { type: [String], required: true},
  usersDisliked: { type: [String], required: true},
})

//* EXPORTATION DU MODELE DE SCHEMA
module.exports = mongoose.model('Sauce', sauceSchema)
```

Mongoose

Schema “Sauce”

Export Model

routes/sauce.JS

```
//* IMPORTS
const express = require('express') // EXPRESS
const auth = require('../middleware/auth') // AUTHENTIFICATION (SECURITE)
const multer = require('../middleware/multer-config') // MULTER (IMAGES)
const { getAllSauces, createSauce, getOneSauce, modifySauce, deleteSauce,
likeDislikeSauce } = require("../controllers/sauce") // CONTROLEUR SAUCE

//* DECLARE ROUTER D'EXPRESS
const router = express.Router()

//* POST UNE CREATION DE SAUCE D'UN CLIENT
router.post('/api/sauces', auth, multer, createSauce)
//* PUBLI LES DONNEES DE CHAQUE SAUCES SUR LA PAGE ALL SAUCES
router.get('/api/sauces', auth, getAllSauces)
//* PUBLI LES DONNEES D'UNE SAUCE SUR SA PAGE
router.get('/api/sauces/:id', auth, getOneSauce)
//* MODIFIE UNE SAUCE DU CLIENT
router.put('/api/sauces/:id', auth, multer, modifySauce)
//* SUPPRIME UNE SAUCE DU CLIENT
router.delete('/api/sauces/:id', auth, deleteSauce)
//* LIKER OU DISLIKER UNE SAUCE CLIENT
router.post('/api/sauces/:id/like', auth, likeDislikeSauce)

//* EXPORTATION DES ROUTES
module.exports = router
```

Imports : middleware, controller

Router

Routes

Export Router

Sauce Middleware 1/2 | Authentification

middleware/auth.JS

```
//* IMPORT JSONWEBTOKEN
const jwt = require('jsonwebtoken')

module.exports = (req, res, next) => {
  try {
    //* RECUPERER HEADER "Authorization" ET GARDER SEULEMENT
    //* LE TOKEN GRACE A LA METHODE ".split"
    const token = req.headers.authorization.split(' ')[1]
    //* DECODER LE TOKEN AVEC LA METHODE "verify" AVEC EN ARGUMENTS
    //* (le token, clé secrète enregistré dans .env)
    const decodedToken = jwt.verify(token, process.env.JWT_TOKEN)
    //* RECUPERER "userId" DU TOKEN
    const userId = decodedToken.userId
    //* AJOUTER "userId" POUR QUE NOS ROUTES PUISSENT L'UTILISER
    req.auth = {
      userId: userId
    }
    next()
  }
  catch(error) {res.status(401).json({ error })}
}
//* ENSUITE => AJOUTER "auth" AUX ROUTES Sauce AVANT CHAQUE CONTROLLER
```

Import json-web-token

Récupérer Token (login) dans header

Décoder Token avec clé

Récupérer userId du Token

Correspondance user Id

Passer au middleware suivant

.env

```
# PASSWORD JWT
JWT_TOKEN=45af32ec-0cd5-40ac-9227-60f559f337e2
```

JWT dans Headers

Request Headers View source

Accept: application/json, text/plain, */*

Accept-Encoding: gzip, deflate, br

Accept-Language: en-GB,en;q=0.9,fr-FR;q=0.8,fr;q=0.7

Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOiI2MzE0Y2QzMTg3ZjlhODllMmNmNjJkODYiLCJpYXQiOjE2NjI3NTI0MzMsImV4cCI6MTY2MjgzODgzM30.mpkDWsYrV_Bv7OQiNYG3j47MW3aB9Ry7yzwOBvucaCA

routes/sauce.JS

```
//* POST UNE CREATION DE SAUCE D'UN CLIENT
router.post('/api/sauces', auth, multer, createSauce)
//* PUBLI LES DONNEES DE CHAQUE SAUCES SUR LA PAGE ALL SAUCES
router.get('/api/sauces', auth, getAllSauces)
//* PUBLI LES DONNEES D'UNE SAUCE SUR SA PAGE
router.get('/api/sauces/:id', auth, getOneSauce)
//* MODIFIE UNE SAUCE DU CLIENT
router.put('/api/sauces/:id', auth, multer, modifySauce)
//* SUPPRIME UNE SAUCE DU CLIENT
router.delete('/api/sauces/:id', auth, deleteSauce)
//* LIKER OU DISLIKER UNE SAUCE CLIENT
router.post('/api/sauces/:id/like', auth, likeDislikeSauce)
```

Sauce Middleware 2/2 | Multer

middleware/multer.JS

```
//* IMPORT MULTER
const multer = require('multer')

//* TYPES D'IMAGES ACCEPTE
const MIME_TYPES = {
  'image/jpg': 'jpg',
  'image/jpeg': 'jpeg',
  'image/png': 'png'
}

//* STOCKAGE DE L'IMAGE
//* UTILISER LA METHODE ".diskStorage" QUI CONFIGURE LE DOSSIER DE RECEPTION ET LE NOM DU FICHIER
const storage = multer.diskStorage ({
  destination: (req, file, callback) => {
    //* CALLBACK (null = PAS D'ERREUR, "DOSSIER DE RECEPTION")
    callback(null, 'images')
  },
  filename: (req, file, callback) => {
    //* REMPLACER LES ESPACES PAR DES UNDERSCORES DANS LE NOM DU FICHIER D'ORIGINE
    const name = file.originalname.split(' ').join('_')
    //* AJOUTER UNE EXTENSION A PARTIR DE "mimetype"
    const extension = MIME_TYPES[file.mimetype]
    //* CALLBACK (null = PAS D'ERREUR, "name" + "date à la miliseconde" + '.' + "extension")
    callback(null, name + Date.now() + '.' + extension)
  }
})

//* EXPORT DE LA CONFIGURATION DE MULTER
//* (AJOUTER AUX ROUTES SAUCE POUR ENRENGISTRER IMAGES AU SYSTEME DE FICHHIER DU SERVEUR)
module.exports = multer({ storage }).single('image')
//* APPEL MULTER ({ notre objet storage }) AVEC LA METHODE ".single('image')" = IMAGE UNIQUE
```

Import Multer

Pour ajouter extension pendant écriture nom fichier

Fonction diskStorage

Argument 1 : Destination

Argument 2 : Filename

Extension

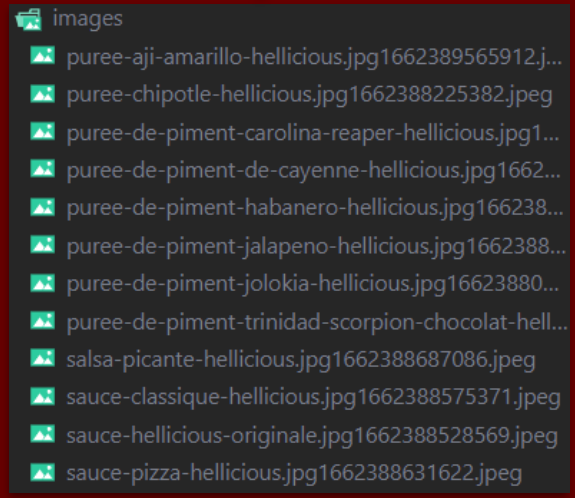
Ajout milliseconde pour nom unique

Export image

routes/sauce.JS

```
//* POST UNE CREATION DE SAUCE D'UN CLIENT
router.post('/api/sauces', auth, multer, createSauce)
//* MODIFIE UNE SAUCE DU CLIENT
router.put('/api/sauces/:id', auth, multer, modifySauce)
```

dossier images



Sauce Controller | Afficher

controllers/sauce.JS Toutes les sauces

```
//* AFFICHER TOUTES LES SAUCES DE LA DATABASE AVEC LA METHODE ".find"
function getAllSauces(req, res, next) {
  Sauce.find({})
    .then(sauces => res.send(sauces))
    .catch(error => res.status(400).json({ error }))
}
```

Model Sauce

controllers/sauce.JS Une seule sauce

```
//* AFFICHER UNE SAUCE DE LA DATABASE SELECTIONNEE AVEC LA METHODE ".findOne"
function getOneSauce(req, res, next) {
  Sauce.findOne({ _id: req.params.id })
    .then(sauce => res.send(sauce))
    .catch(error => res.status(400).json({ error }))
}
```

Model Sauce

Sauce Controller | Créer

controllers/sauce.JS

```
//* CREER UNE SAUCE DANS LA DATABASE
function createSauce(req, res, next) {
  //* PARSER L'OBJET DE LA REQUETE
  const sauceObject = JSON.parse(req.body.sauce)
  //* SUPPRIMER LE CHAMP "userId" DE LA REQUETE CLIENT
  delete sauceObject.userId
  //* CREER UN NOUVEL OBJET AVEC LE MODELE DE SAUCE
  const sauce = new Sauce ({
    //* ... = TOUS LES CHAMPS DE "sauceObject"
    ...sauceObject,
    //* RECUPERE "userId" DEPUIS LE TOKEN D'AUTHENTIFICATION
    userId: req.auth.userId,
    //* CREER L'URL DE L'IMAGE
    imageUrl: `${req.protocol}://${req.get('host')}/images/${req.file.filename}`,
    //* DECLARER O POUR LES LIKE ET DISLIKE
    likes: 0,
    dislikes: 0,
    //* CREER UN TABLEAU VIDE POUR LES UTILISATEURS QUI VONT LIKE ET DISLIKE
    usersLiked: [],
    usersDisliked: []
  })
  //* ENRENGISTRER DANS LA DATABASE
  sauce.save()
    .then(() => res.status(201).json({ message: "Sauce enrengistrée" }))
    .catch((error) => res.status(400).json( error ))
}
```

Parser pour manipuler objet

Supprimer userId requête

Nouvel objet à partir du Model

Récupérer userId depuis middleware

Générer url image

Valeur 0

Tableaux vides

Sauvegarde database

Database : objet Sauce


```
_id: ObjectId("6316065069386846e57d1e1a")
userId: "630c7eae14d20fae25d13441"
name: "CAROLINA REAPER"
manufacturer: "Hellicious"
              "À PROPOS DE PURÉE DE PIMENT CAROLINA REAPER HELLICIOUS
description:
              Imaginez les v..."
mainPepper: "Piment Carolina Reaper frais (origine Hollande), eau, vinaigre."
imageUrl: "http://localhost:3000/images/puree-de-piment-carolina-reaper-helliciou..."
heat: 10
likes: 3
dislikes: 0
› usersLiked: Array
› usersDisliked: Array
```

Sauce Controller | Modifier + Supprimer

controllers/sauce. JS

```
//* MODIFIER UNE SAUCE AVEC LA METHODE ".updateOne"
function modifySauce(req, res, next) {
  //* VERIFIER S'IL Y A UN OBJET DANS NOTRE REQUETE "req.file"
  const sauceObject = req.file ? {
    //* PARSE L'OBJET DE LA REQUETE
    ...JSON.parse(req.body.sauce),
    //* CREER L'URL DE L'IMAGE
    imageUrl: `${req.protocol}://${req.get('host')}/images/${req.file.filename}`
  //* ENSUITE RECUPERER LES DONNEES A MODIFIER "...req.body"
  } : {...req.body}
  //* SUPPRIMER LE CHAMP "userId" DE LA REQUETE
  delete sauceObject.userId
  //* CHERCHER L'OBJET DANS LA DATABASE
  Sauce.findOne({ _id: req.params.id })
    //* VERIFIER QUE L'UTILISATEUR EST LE PROPRIETAIRE DE L'OBJET A MODIFIER
    .then((sauce) => {
      //* SI "userId" DE LA DATABASE EST != DE "userId" DE LA REQUETE
      if (sauce.userId != req.auth.userId) {
        //* ANNULER LA REQUETE ET RENVOI UN MSG "Non-autorisé"
        res.status(401).json({ message: 'Non-autorisé' })
      //* SI IL EST LE PROPRIETAIRE
      } else {
        //* SI L'UTILISATEUR CHANGE L'IMAGE, SUPPRIMER L'ANCIENNE
        if (req.file) {
          const filename = sauce.imageUrl.split("/images/")[1]
          fs.unlink(`images/${filename}`, () => {})
        }
        //* ECRASER LES ANCIENNES DONNEES PAR LES NOUVELLES => "sauceObject"
        Sauce.updateOne({ _id: req.params.id}, {...sauceObject, _id: req.params.id})
          .then(() => res.status(200).json({message : ' Sauce modifiée! '}))
          .catch(error => res.status(401).json({ error }))
      }
    })
    .catch((error) => res.status(400).json({ error }))
}
```

Si modification image, parser + générer imageUrl

Trouver id objet dans database

Vérifier utilisateur

Si nouvelle image supprimer ancienne

Mise à jour objet en database

controllers/sauce. JS

```
//* IMPORT DE FS (SUPPRIME LES IMAGES)
const fs = require('fs')
```

Import *fs* : File System (module Node)

```
//* SUPPRIMER UNE SAUCE AVEC LA METHODE ".deleteOne"
function deleteSauce(req, res, next) {
  //* CHERCHER L'OBJET DANS LA DATABASE
  Sauce.findOne({ _id: req.params.id })
  //* VERIFIER QUE L'UTILISATEUR EST LE PROPRIETAIRE DE L'OBJET A SUPPRIMER
    .then (sauce => {
      //* SI "userId" DE LA DATABASE EST != DE "userId" DE LA REQUETE
      if (sauce.userId != req.auth.userId) {
        //* ANNULER LA REQUETE ET RENVOI UN MSG "Non-autorisé"
        res.status(401).json({ message: 'Non-autorisé' })
      } else {
        //* SINON CHERCHER LE NOM DE L'IMAGE A SUPPRIMER AVEC "split"
        const filename = sauce.imageUrl.split('images/')[1]
        //* UTILISER FS POUR SUPPRIMER L'IMAGE
        fs.unlink(`images/${filename}`, () => {
          //* CALLBACK POUR SUPPRIMER LA SAUCE DE LA DATABASE
          Sauce.deleteOne({ _id: req.params.id })
            .then(() => res.status(200).json({ message: 'Sauce supprimée !'}))
            .catch(error => res.status(400).json({ error }));
        })
      }
    })
    .catch(error => {res.status(500).json({ error })})
}
```

Trouver id objet dans database

Vérifier utilisateur

Supprimer image stockée

Supprimer objet database

Sauce Controller | Like / Dislike

controllers/sauce.JS

```
//* LIKE OU DISLIKE UNE SAUCE
function likeDislikeSauce(req, res, next) {

  const like = req.body.like
  const userId = req.body.userId
  const sauceId = req.params.id

  //* +1 like
  if (like === 1) {
    //* MODIFIER LA SAUCE DE LA REQUETE
    Sauce.updateOne({ _id: sauceId }, {
      //* AJOUTER "userId" DANS LE TABLEAU DES UTILISATEURS QUI ONT LIKE
      $push: { usersLiked: userId },
      //* AJOUTER "+1" SUR LE NOMBRE TOTAL DE LIKE
      $inc: { likes: +1 }
    })
      .then(() => res.status(200).json({ message: '1 like ajouté !' }))
      .catch((error) => res.status(400).json({ error }))
  }

  //* +1 dislike
  if (like === -1) {
    //* MODIFIER LA SAUCE DE LA REQUETE
    Sauce.updateOne({ _id: sauceId }, {
      //* AJOUTER "userId" DANS LE TABLEAU DES UTILISATEURS QUI ONT DISLIKE
      $push: { usersDisliked: userId },
      //* AJOUTER "+1" SUR LE NOMBRE TOTAL DE DISLIKE
      $inc: { dislikes: +1 }
    })
      .then(() => { res.status(200).json({ message: '1 dislike ajouté !' })})
      .catch((error) => res.status(400).json({ error }))
  }
```

Déclarer variables utiles

Dans l'objet Sauce concerné :
"Pousser" userId dans tableau
"usersLiked OU usersDisliked"
Incrémenter valeur +1 dans Likes OU Dislike

controllers/sauce.JS

```
//* 0 like OU  0 dislike
if (like === 0) {
  //* CHERCHER LA SAUCE A MODIFIER
  Sauce.findOne({ _id: sauceId })
    .then((sauce) => {
      //* SI L'UTILISATEUR A LIKE UNE SAUCE =>
      if (sauce.usersLiked.includes(userId)) {
        //* MODIFIER LA SAUCE DE LA REQUETE
        Sauce.updateOne({ _id: sauceId }, {
          //* RETIRER "userId" DU TABLEAU DES UTILISATEURS QUI ONT LIKE
          $pull: { usersLiked: userId },
          //* RETIRER "+1" SUR LE NOMBRE TOTAL DE LIKE
          $inc: { likes: -1 }
        })
          .then(() => res.status(200).json({ message: '-1 like' }))
          .catch((error) => res.status(400).json({ error }))
      }
      //* SI L'UTILISATEUR A DISLIKE UNE SAUCE =>
      if (sauce.usersDisliked.includes(userId)) {
        //* MODIFIER LA SAUCE DE LA REQUETE
        Sauce.updateOne({ _id: sauceId }, {
          //* RETIRER "userId" DU TABLEAU DES UTILISATEURS QUI ONT DISLIKE
          $pull: { usersDisliked: userId },
          //* RETIRER "+1" SUR LE NOMBRE TOTAL DE DISLIKE
          $inc: { dislikes: -1 }
        })
          .then(() => res.status(200).json({ message: '-1 dislike' }))
          .catch((error) => res.status(400).json({ error }))
      }
    })
    .catch((error) => res.status(404).json({ error }))
}
```

Dans l'objet Sauce concerné :
"Retirer" userId dans tableau
"usersLiked OU usersDisliked"
Incrémenter valeur -1 dans Likes OU Dislike

3 - Bilan & Axe d'amélioration

Cahier des charges

Inscription : Email unique ; Hachage mot de passe

Connexion : Renvoie un Token web signé (id), durée limitée

Routes Sauce : Authentification renforcée sur toutes les routes Sauce par vérification du Token web signé

Database : Mongoose renvoie les erreurs ; Identifiants en variable d'environnement (.env n'est pas envoyé à GitHub)

Dépendances : Versions les plus récentes incluant les derniers correctifs de sécurité

Images : Contenu dossier image n'est pas envoyé à GitHub

Ajouts : Sécurité headers HTTP via Helmet + dotenv

Axe d'amélioration

Prendre en considération les recommandations de l'OWASP

Injection : Protection des entrées, Fuzzing (test) -> Protection contre injection SQL, attaques XSS

Piratage de session : Limiter nombre de requête de connexion (Force brute)

Données en transit : HTTPS pour l'ensemble du site

Contrôles d'accès : Si l'application évolue, toujours s'assurer que toutes les pages sont verrouillées par contrôle d'accès

Tester la sécurité : Faire réaliser des tests de sécurité par un prestataire spécialisé en sécurité (tests d'intrusion, menaces potentielles…)

Veille sécurité : S'informer sur les nouvelles vulnérabilités

Questions ?