

# Sommaire

I. Contexte du projet.

II. Présentation du projet.

III. Code.



# PIQUANT

Novembre 2023

# I. Contexte du projet.

Piquante se dédie à la création de sauces épicées dont les recettes sont gardées secrètes. Pour tirer parti de son succès et générer davantage de buzz, l'entreprise souhaite créer une application web dans laquelle les utilisateurs peuvent ajouter leurs sauces préférées et liker ou disliker les sauces ajoutées par les autres.

## **Mission** :

Construisez une API sécurisée pour une application d'avis gastronomiques

- Création d'une API sécurisée en prenant en compte les exigences
- Phase 1 : Créer les routes nécessaires afin de faire fonctionner notre API.
- Phase 2 : Sécurisée notre API.
- Dernière phase : Vérifier que tout fonctionne et que la sécurité du site est fonctionnel.

Spécifications de l'API

	Point d'accès	Request body (le cas échéant)	Type de réponse attendue	Fonction
POST	/api/auth/signup	{ email: string, password: string }	{ message: string }	Hachage du mot de passe de l'utilisateur, ajout de l'utilisateur à la base de données.
POST	/api/auth/login	{ email: string, password: string }	{ userId: string, token: string }	Vérification des informations d'identification de l'utilisateur, renvoie l_id de l'utilisateur depuis la base de données et un token web JSON signé (contenant également l_id de l'utilisateur).
GET	/api/sauces	-	Array of sauces	Renvoie un tableau de toutes les sauces de la base de données.
GET	/api/sauces/:id	-	Single sauce	Renvoie la sauce avec l' _id fourni.
POST	/api/sauces	{ sauce: String, image: File }	{ message: String } <b>Verb</b>	Capture et enregistre l'image, analyse la sauce transformée en chaîne de caractères et l'enregistre dans la base de données en définissant correctement son imageUrl. Initialise les likes et dislikes de la sauce à 0 et les usersLiked et usersDisliked avec des tableaux vides. Remarquez que le corps de la demande initiale est vide ; lorsque multer est ajouté, il renvoie une chaîne pour le corps de la demande en fonction des données soumises avec le fichier.

PUT	/api/sauces/:id	EITHER Sauce as JSON OR { sauce: String, image: File }	{ message: String }	Met à jour la sauce avec l' _id fourni. Si une image est téléchargée, elle est capturée et l'imageUrl de la sauce est mise à jour. Si aucun fichier n'est fourni, les informations sur la sauce se trouvent directement dans le corps de la requête (req.body.name, req.body.heat, etc.). Si un fichier est fourni, la sauce transformée en chaîne de caractères se trouve dans req.body.sauce. Notez que le corps de la demande initiale est vide ; lorsque multer est ajouté, il renvoie une chaîne du corps de la demande basée sur les données soumises avec le fichier.
DELETE	/api/sauces/:id	-	{ message: String }	Supprime la sauce avec l' _id fourni.
POST	/api/sauces/:id/like	{ userId: String, like: Number }	{ message: String }	Définit le statut « Like » pour l' userId fourni. Si like = 1, l'utilisateur aime (= like) la sauce. Si like = 0, l'utilisateur annule son like ou son dislike. Si like = -1, l'utilisateur n'aime pas (= dislike) la sauce. L'ID de l'utilisateur doit être ajouté ou retiré du tableau approprié. Cela permet de garder une trace de leurs préférences et les empêche de liker ou de ne pas disliker la même sauce plusieurs fois : un utilisateur ne peut avoir qu'une seule valeur pour chaque sauce. Le nombre total de « Like » et de « Dislike » est mis à jour à chaque nouvelle notation.

❖ Consigne :  
❖

Le développement devra se faire uniquement en JavaScript.  
Utiliser les Spécifications de l'API.

Reprendre la Data ModelsSauce.

Respecter les exigences de sécurité.  
Interdiction de modifier ou d’ajouter du code HTML.

## II. Présentation du projet.

# Création de compte.



**HOT TAKES**  
THE WEB'S BEST HOT SAUCE REVIEWS

[SIGN UP](#)

[LOGIN](#)

Email

Password

[SIGN UP](#)

# Pourvoir se connecter.



**HOT TAKES**

THE WEB'S BEST HOT SAUCE REVIEWS

[SIGN UP](#)

[\*\*LOGIN\*\*](#)

Email

exemplepp@hotmail.fr

Password

.....

**LOGIN**



# Ajouter une ou des sauces

[ALL SAUCES](#) [ADD SAUCE](#)



**HOT TAKES**  
THE WEB'S BEST HOT SAUCE REVIEWS

[LOGOUT](#)

Name

Manufacturer

Description

ADD IMAGE

Main Pepper Ingredient

Heat

1

SUBMIT

# Voir tout les sauces créés

---

[ALL SAUCES](#)

[ADD SAUCE](#)



**HOT TAKES**

THE WEB'S BEST HOT SAUCE REVIEWS

[LOGOUT](#)

---

THE SAUCES



**CURRY**

Heat: 4/10

---

# Like - Dislike - Modifier - Supprimer

[ALL SAUCES](#) [ADD SAUCE](#)



**HOT TAKES**  
THE WEB'S BEST HOT SAUCE REVIEWS

[LOGOUT](#)



## Curry

by 4.50€

### Description

Douce et savoureuse

 1  0

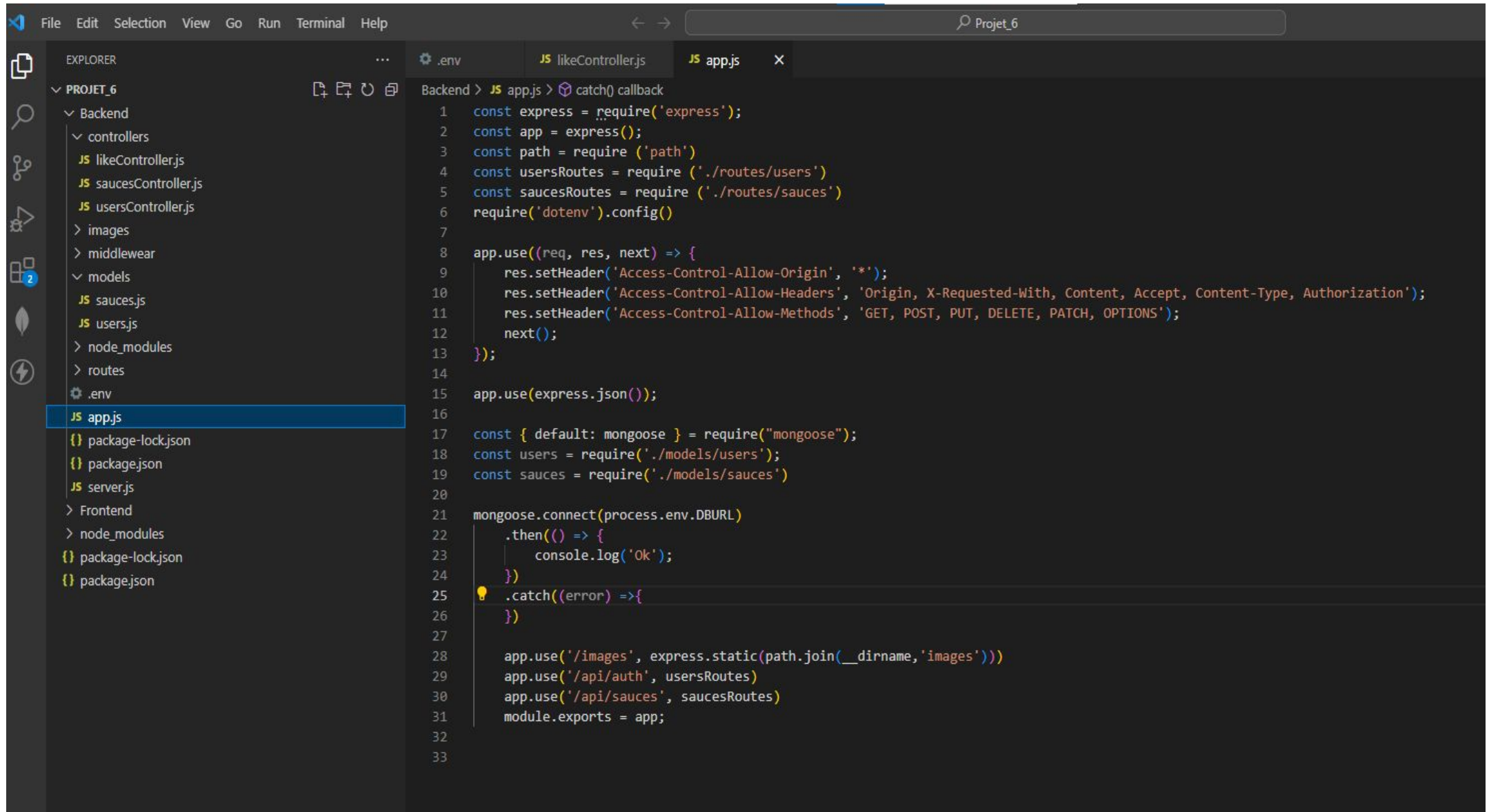
[BACK](#)

[MODIFY](#)

[DELETE](#)

# III. Code.

## Partie n°1: Javascript.



The image shows a Visual Studio Code editor window with a project named 'Projet\_6'. The Explorer sidebar on the left displays the project structure, with 'app.js' selected under the 'Backend' directory. The main editor area shows the content of 'app.js', which is a Node.js application using Express.js and Mongoose. The code includes imports for express, path, and mongoose, sets up middleware for CORS and JSON, connects to a MongoDB database, and defines routes for users and saucuses. A lightbulb icon indicates a suggestion or error at line 25.

```
Backend > JS app.js > catch() callback
1  const express = require('express');
2  const app = express();
3  const path = require('path')
4  const usersRoutes = require('./routes/users')
5  const saucusesRoutes = require('./routes/sauces')
6  require('dotenv').config()
7
8  app.use((req, res, next) => {
9    res.setHeader('Access-Control-Allow-Origin', '*');
10   res.setHeader('Access-Control-Allow-Headers', 'Origin, X-Requested-With, Content, Accept, Content-Type, Authorization');
11   res.setHeader('Access-Control-Allow-Methods', 'GET, POST, PUT, DELETE, PATCH, OPTIONS');
12   next();
13 });
14
15 app.use(express.json());
16
17 const { default: mongoose } = require("mongoose");
18 const users = require('./models/users');
19 const saucuses = require('./models/sauces')
20
21 mongoose.connect(process.env.DBURL)
22   .then(() => {
23     console.log('Ok');
24   })
25   .catch((error) =>{
26   })
27
28 app.use('/images', express.static(path.join(__dirname, 'images')))
29 app.use('/api/auth', usersRoutes)
30 app.use('/api/saucuses', saucusesRoutes)
31 module.exports = app;
32
33
```

Merci pour votre lecture et pour votre temps.