



**PIIQUANTE**


# **PRESENTATION**

---

## **PROJET 6 PIIQUANTE**


OPENCLASSROOMS

# Composition du site Piquante

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

[SIGN UP](#) [LOGIN](#)


---

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

[SIGN UP](#) [LOGIN](#)

---


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

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


[LOGOUT](#)

---


THE SAUCES




SAUCE PIQUANTE  
BOMBE H  
Heat: 7/10



SAUCE PIQUANTE X  
MAISON MARTIN  
Heat: 7/10




PSYCHO JUICE  
Heat: 2/10




TABASCO -  
MCILHENNY  
Heat: 3/10

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

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

[LOGOUT](#)



---



**Tabasco - McIlhenny**  
by Tabasco

**Description**  
La pépite de Tabasco, une sauce unique et de qualité supérieure qui à l'origine était réservée à la famille McILHENNY

**Ingrédients principaux**  
Vinaigre de vin blanc, piment, sel.

 0  0

---

Heat  

1

Technologies et outils utilisées

- Visual Studio Code
  - MongoDB Atlas
  - Postman
- 
- Node (environnement d'exécution JavaScript open-source)
  - Bcrypt (algorithme de chiffrement)
  - Dotenv (module de Node.js)
  - Express (framework JavaScript pour Node.js)
  - Helmet (middleware pour Node.js)
  - jsonwebtoken (standard pour l'échange sécurisé de données)
  - Mongoose (module Node.js)
  - Mongoose-unique-validateur (plugin pour Mongoose)
  - multer (middleware pour Node.js)

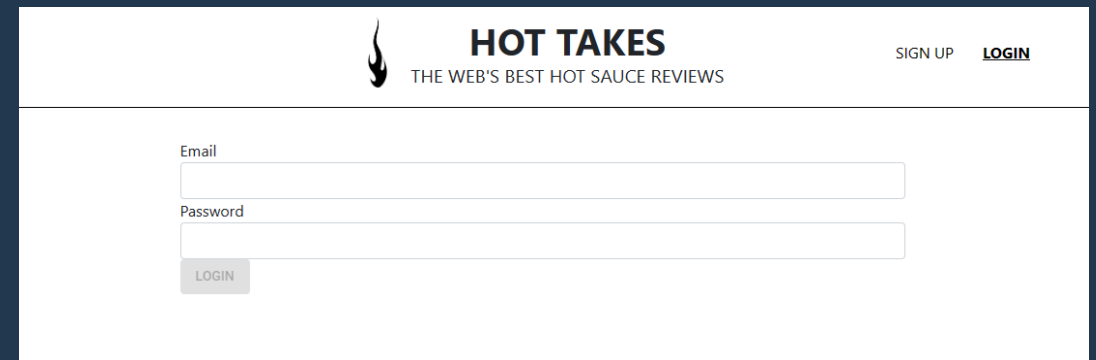
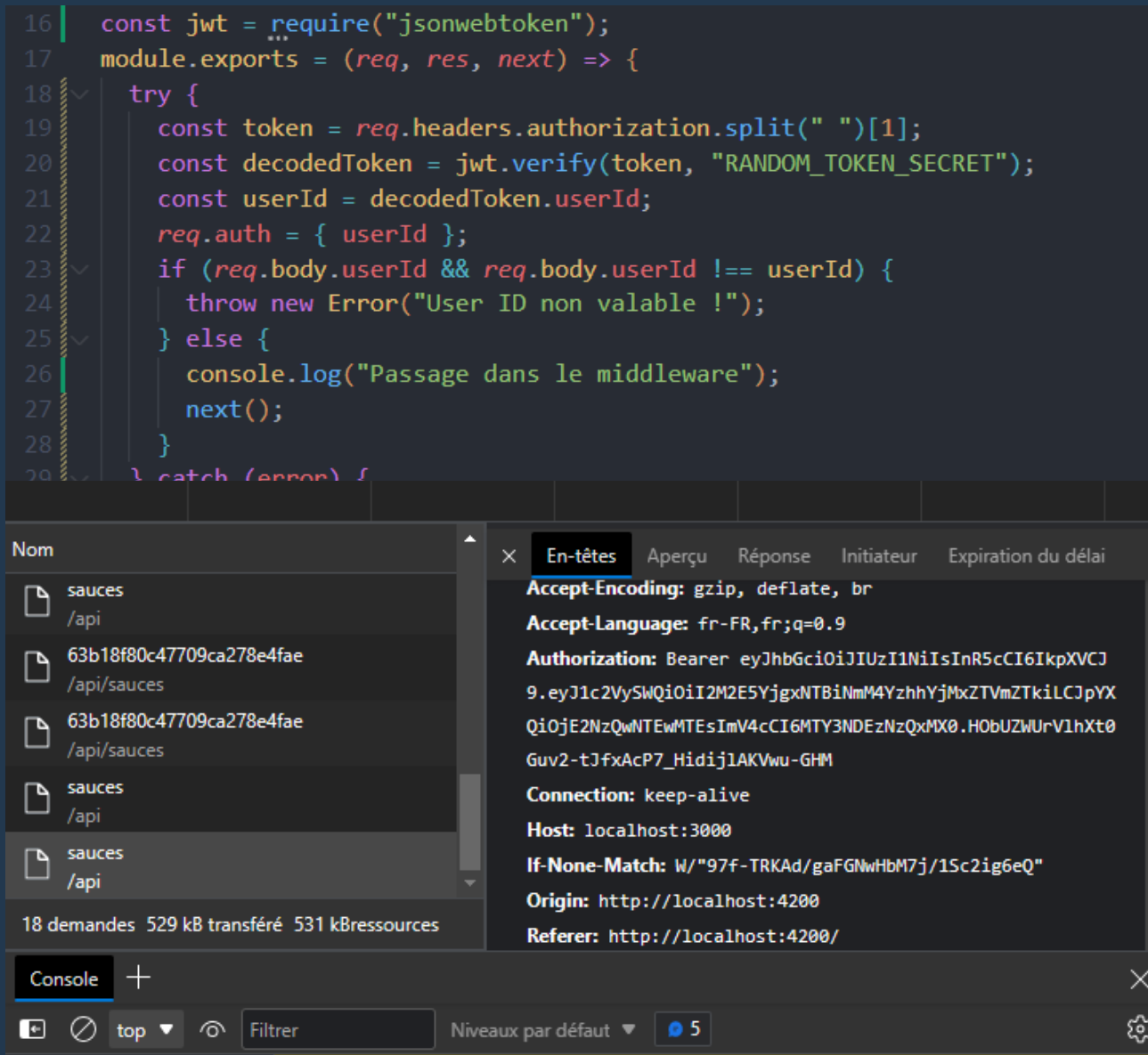
Explication et fonctionnement

# Le middleware d'authentification

Le middleware d'authentification va permettre de gérer et vérifier les informations d'identification d'un utilisateur.

Ils sera exécutés sur le serveur et inaccessibles pour l'utilisateur.

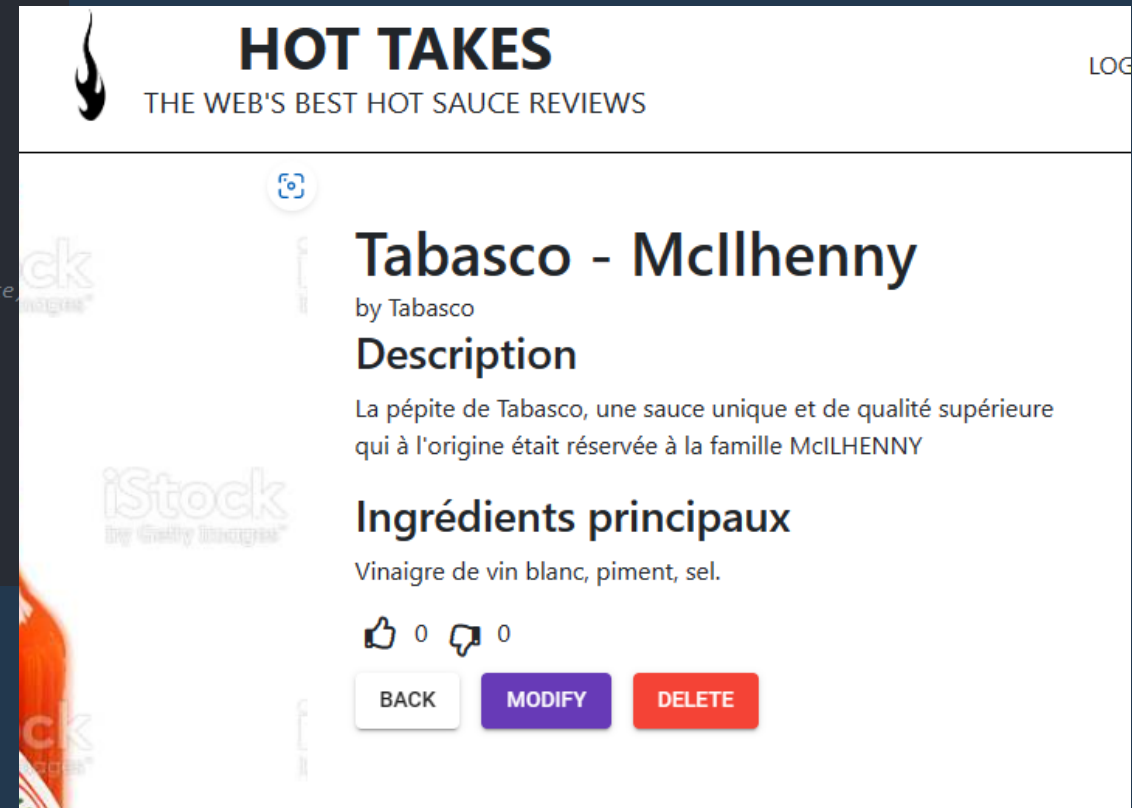
Jsonwebtoken est utilisé pour l'échange sécurisé de données en utilisant un jeton (Token) cryptographique.



# Les opérations CRUD

Les opérations CRUD (Create, Read, Update, Delete) sur les données stockées dans la base de données. Par exemple ici dans le dossier controllers nous avons une opérations CRUD, ce code va permettre de supprimer une sauce.

```
91 exports.deleteSauce = (req, res, next) => {
92   Sauce.findOne({ _id: req.params.id })
93     .then(sauce => {
94       if (sauce.userId !== req.auth.userId) {
95         res.status(401).json({ message: 'Not authorized' });
96       } else {
97         const filename = sauce.imageUrl.split('/images/')[1];
98         fs.unlink(`images/${filename}`, () => {
99           Sauce.deleteOne({ _id: req.params.id })
100             //Si suppression réussie, envoi d'un message (statut 200, indique la réussite d'une requête
101             .then(() => res.status(200).json({ message: 'Objet supprimé !' })))
102             //Sinon renvoi d'un message d'erreur 400 (Bad Request)
103             .catch(error => res.status(400).json({ error }));
104         });
105       }
106     })
107     .catch(error => res.status(500).json({ error }));
108 };
109
```



# La gestion des sessions

Les informations de session et les jetons d'authentification sont généralement stockés sur le serveur.

```
15  const jwt = require('jsonwebtoken');
16
17  //Rappel de la fonction de hachage de bcrypt dans le mot de passe et "salage" du mot de passe
18  //10 fois (plus la valeur est élevée, plus l'exécution de la fonction sera longue,
19  //et plus le hachage sera sécurisé.
20  //C'est une fonction asynchrone qui renvoie une Promise dans laquelle nous recevons le hash généré dans
21  //le bloc then, Création d'un utilisateur et enregistrement dans la base de données,
22  //en renvoyant une réponse de réussite en cas de succès,
23  //et des erreurs avec le code d'erreur en cas d'échec.
24  exports.signup = (req, res, next) => {
25    bcrypt.hash(req.body.password, 10)
26      .then(hash => {
27        const user = new User({
28          email: req.body.email,
29          password: hash
30        });
31        user.save()
32          .then(() => res.status(201).json({ message: 'Utilisateur créé !'}))
33          .catch(error => res.status(400).json({ error }));
34      })
35      .catch(error => res.status(500).json({ error }));
36  };
```



# La validation des données

La validation des données côté serveur est généralement exécutée sur le serveur avant de traiter les données.

```
2 //Importation de MONGOOSE
3 const mongoose = require('mongoose');
4
5 //Importation de mongoose-unique-validator
6 //plugin qui ajoute une validation de pré-enregistrement pour les champs uniques
7 //dans un schéma Mongoose, ce qui permet de ne valider q'un seul email par utilisateur
8 const uniqueValidator = require('mongoose-unique-validator');
```

The screenshot displays the MongoDB Atlas web interface. The top navigation bar includes the Atlas logo, a dropdown menu for 'Aykut's Org', and links for 'Gestionnaire d'accès' and 'Facturation'. The main navigation bar features 'Sélectionner un...', 'Services de données' (highlighted), 'Services d'application', and 'Graphiques'. The left sidebar contains sections for 'DÉPLOIEMENT', 'Base De Données', 'SERVICES', 'SÉCURITÉ', and 'Nouveautés D'Atlas'. The main content area is titled 'ClusterP6' and shows the following details:

- VERSION: 5.0.14
- RÉGION: AWS Paris (eu-west-3)
- NIVEAU DU CLUSTER: M0 Sandbox (Général)

The 'Aperçu' tab is selected, showing a 'BAC À SABLE' (Sandbox) environment. The 'RÉGION' is 'Paris (eu-west-3)'. The cluster is configured with three nodes: 'AC... partition\_00-00.f8...' (SECONDAIRE), 'AC... partition\_00-01.f8...' (SECONDAIRE), and 'AC... partition\_00-02.f8...' (PRIMAIRE). A central message states: 'Il s'agit d'un cluster de niveaux partagés. Si vous avez besoin d'une base de données adaptée aux applications de production hautes performances, effectuez une mise à niveau vers un cluster dédié.' A 'Mise à niveau' (Upgrade) button is visible. The 'Opérations' (Operations) section shows 'R:0 W:0' and a graph of operations over time. The 'Taille' (Size) section shows 'logique 48.1 Ko' and a graph of size over time. The 'Connexions' (Connections) section shows '4' connections and a graph of connections over time.

# Les traitements de fichiers

La gestion des téléchargements de fichiers, le stockage des fichiers sur le serveur et les traitements de fichiers.

```
1  // Importation de MULTER qui est un package de gestion de fichiers.
2  const multer = require('multer');
3
4  // Dictionnaires des types MIME des extension des fichiers
5  //qu'on peut trouver dans le frontend
6  const MIME_TYPES = {
7    'image/jpg': 'jpg',
8    'image/jpeg': 'jpg',
9    'image/png': 'png'
10 };
11
12
13 //Création d'une constante storage , à passer à multer comme configuration,
14 //qui contient la logique nécessaire pour indiquer à multer où enregistrer
15 //les fichiers entrants
16 //La méthode diskStorage() configure le chemin et le nom de fichier pour les fichiers entrants
17 const storage = multer.diskStorage({
18   // La fonction destination indique à multer d'enregistrer
19   //les fichiers dans le dossier images
20   destination: (req, file, callback) => {
21     callback(null, 'images')
22   },
```

# STRUCTURES

## Les contrôleurs

Responsables de la gestion des interactions entre l'utilisateur et l'application. Ils reçoivent les requêtes de l'utilisateur, les traitent et renvoient les réponses appropriées.

## Les routeurs

Responsables de l'acheminement des requêtes vers les contrôleurs appropriés. Ils examinent les informations de la requête, comme l'URL et les paramètres, pour déterminer quel contrôleur doit gérer la requête.

## Les modèles

Responsables de la gestion des données de l'application. Ils effectuent des opérations CRUD (Create, Read, Update, Delete) sur les données stockées dans la base de données.

# MÉTHODES POUR SÉCURISER LA BASE DE DONNÉES

## Expliquez vos méthodes pour sécuriser la base de données selon le RGPD et l'OWASP.

Il existe plusieurs méthodes pour sécuriser une base de données selon les exigences du RGPD et de l'OWASP. Voici quelques exemples :

- Chiffrement des données sensibles : cela permet de protéger les données contre les accès non autorisés et les fuites de données.
- Authentification forte : Il est important d'utiliser des méthodes d'authentification robustes pour s'assurer que seuls les utilisateurs autorisés ont accès à la base de données.
- Mise à jour et maintenance régulières : Les mises à jour de sécurité et les correctifs doivent être installés régulièrement pour protéger la base de données contre les vulnérabilités connues.
- Contrôle d'accès : Il est important de configurer les autorisations d'accès de manière à ce que seuls les utilisateurs ayant besoin d'accéder à certaines données puissent le faire.
- Sauvegarde régulière : Il est important de sauvegarder régulièrement les données pour pouvoir les restaurer en cas de perte ou de corruption.
- Surveillance et détection des intrusions : Il est important de surveiller les accès à la base de données et d'être en mesure de détecter les tentatives d'intrusion pour pouvoir y réagir rapidement.

# DÉFINITIONS

bcrypt est un algorithme de chiffrement de type "dérivé de mot de passe" utilisé pour crypter les mots de passe d'un utilisateur. Il utilise une fonction de hachage dérivée de mot de passe pour générer un hash unique pour chaque mot de passe, il utilise également un "coût" pour augmenter le temps nécessaire pour générer le hash, rendant les attaques par force brute plus difficiles. Il est considéré comme sécurisé car il utilise un sel aléatoire pour chaque mot de passe, il est également conçu pour être résistant aux attaques à accélération matérielle. Il est important de stocker les mots de passe des utilisateurs de manière sécurisée et de ne jamais stocker des mots de passe en clair.



dotenv est un module de Node.js qui permet de charger les variables d'environnement à partir d'un fichier `.env`. Il permet de séparer les données sensibles de votre code source et de configurer des environnements différents pour le développement et la production sans avoir à modifier le code source. Il charge les variables d'environnement de manière à les rendre accessibles via `process.env`. Il est important de noter que le fichier `.env` doit être exclu des outils de versionnement pour éviter de partager les informations sensibles avec d'autres membres de l'équipe de développement ou les utilisateurs.

Lorsque vous utilisez dotenv, vous pouvez créer un fichier `.env` dans le répertoire racine de votre projet. Ce fichier contient des variables d'environnement et leur valeur sous la forme suivante :

```
NOM_DE_VARIABLE=VALEUR
```

Dans votre code, vous pouvez charger ces variables d'environnement en utilisant la méthode `config()` de dotenv.

```
require('dotenv').config();
```

Les variables d'environnement chargées par dotenv sont ensuite disponibles via `process.env` :

```
console.log(process.env.NOM_DE_VARIABLE)
```

Helmet est un middleware pour Node.js qui permet de sécuriser les applications web en ajoutant des en-têtes HTTP de sécurité. Il ajoute des en-têtes pour protéger contre les attaques de type cross-site scripting (XSS), force les navigateurs à utiliser HTTPS, empêche les navigateurs de désactiver la vérification de type de contenu et empêche les navigateurs de charger l'application dans un cadre ou une iframe. Il facilite la configuration et l'ajout de ces en-têtes de sécurité pour les développeurs sans avoir à les gérer manuellement.

JSON Web Token (JWT) est un standard pour l'échange sécurisé de données en utilisant un jeton cryptographique sous forme de JSON. Il est utilisé pour l'authentification et l'autorisation dans les applications web et mobiles, les données telles que les informations d'identification de l'utilisateur et les claims sont stockées dans le jeton, qui est ensuite envoyé au client pour être utilisé pour les requêtes futures. Le serveur peut également décoder le JWT pour vérifier l'authenticité de la demande et donner accès aux ressources protégées.

Mongoose est un module Node.js qui facilite la mise en place de schémas de données pour les applications utilisant MongoDB. Il permet de définir des modèles pour les données, avec des fonctionnalités comme la validation des données, la gestion des relations entre les modèles et la génération automatique de méthodes CRUD pour les données.

Il fournit également des méthodes pour effectuer des opérations de base sur les données, telles que la recherche, la mise à jour et la suppression de documents, ainsi que des méthodes avancées telles que les agrégations et les index.

En utilisant Mongoose, vous pouvez facilement définir des modèles pour vos données et les utiliser pour effectuer des opérations CRUD sur les données stockées dans MongoDB, tout en profitant de la validation des données, de la gestion des relations entre les modèles et de la génération automatique de méthodes CRUD.

Mongoose-unique-validator est un plugin pour Mongoose qui permet de valider l'unicité de certaines propriétés de vos modèles Mongoose. Il ajoute une validation supplémentaire pour s'assurer qu'une propriété spécifique n'est pas déjà utilisée par un autre document dans la collection MongoDB. Cela est particulièrement utile pour les propriétés qui doivent être uniques dans la base de données, comme les adresses e-mail ou les noms d'utilisateur.

Il permet de vérifier l'unicité d'une propriété avant de sauvegarder un document dans la base de données, et émettra une erreur si une valeur dupliquée est trouvée. Il peut également vérifier l'unicité d'une propriété lors de la mise à jour d'un document existant en utilisant des options de mise à jour telles que "runValidators" ou "context: 'query'".

En utilisant Mongoose-unique-validator, vous pouvez facilement valider l'unicité des propriétés de vos modèles Mongoose, ce qui vous aide à garantir la qualité de vos données et à éviter les erreurs en cours de sauvegarde ou de mise à jour des données.

Multer est un middleware pour Node.js qui permet de gérer les fichiers envoyés à travers les formulaires HTML et les requêtes HTTP. Il permet de gérer les fichiers téléchargés par les utilisateurs, de les stocker sur le serveur et de les inclure dans les requêtes HTTP pour les traiter sur le serveur. Il prend en charge les fichiers multiples et les champs de formulaire multiples, et permet de définir des limites de taille de fichier et des filtres de type MIME pour les fichiers téléchargés.

Il est souvent utilisé avec des frameworks web tels que Express.js pour gérer les fichiers téléchargés par les utilisateurs dans les applications web. Il permet de gérer facilement les fichiers téléchargés par les utilisateurs, de les stocker sur le serveur et de les inclure dans les requêtes HTTP pour les traiter sur le serveur, Il permet de gérer les fichiers téléchargés de manière efficace.

CRUD est un acronyme pour les opérations de base pour gérer les données d'une base de données : Create, Read, Update et Delete.

- Create (Créer) : Il s'agit de l'opération qui permet d'ajouter de nouveaux enregistrements ou documents dans la base de données.
- Read (Lire) : Il s'agit de l'opération qui permet de lire ou de récupérer des données dans la base de données.
- Update (Mettre à jour) : Il s'agit de l'opération qui permet de mettre à jour ou de modifier les données existantes dans la base de données.
- Delete (Supprimer) : Il s'agit de l'opération qui permet de supprimer des enregistrements ou des documents de la base de données.

Ces opérations sont les plus utilisées dans les applications pour gérer les données d'une base de données, avec des outils comme SQL ou MongoDB, et sont généralement implémentées avec des systèmes de gestion de bases de données ou des ORM (Object-Relational Mapping) pour faciliter l'accès et la manipulation des données.

Un middleware est une couche logicielle qui se trouve entre la couche d'application et la couche de système ou de données. Il permet de gérer les tâches qui ne sont pas directement liées à la fonctionnalité principale de l'application. Les tâches gérées par les middlewares peuvent inclure:

- **Traitement des requêtes et des réponses** : Les middlewares peuvent être utilisés pour gérer les requêtes et les réponses HTTP, comme la gestion des en-têtes, des cookies, des paramètres de requête, etc.
- **Authentification et autorisation** : Les middlewares peuvent être utilisés pour gérer les processus d'authentification et d'autorisation, comme la vérification des informations d'identification de l'utilisateur pour accéder à certaines parties de l'application.
- **Validation des données** : Les middlewares peuvent être utilisés pour valider les données avant de les traiter.
- **Gestion des erreurs** : Les middlewares peuvent être utilisés pour gérer les erreurs qui se produisent dans l'application et pour fournir des réponses appropriées aux utilisateurs.
- **Traitement des fichiers** : Les middlewares peuvent être utilisés pour gérer les téléchargements et les téléchargements de fichiers.



FIN DE LA PRESENTATION