```
1   const express = require('express'); // Permet l'importation du framework Express
2   const mongoose = require('mongoose');
3   const path = require('path'); // Permet l'accès au chemin de notre système de
    fichiers
4
5   const saucesRoutes = require('./routes/sauces');
6   const userRoutes = require('./routes/user');
7
8   require('dotenv').config(); // Permet de masquer les informations de connexion à la
    base de données
9
10  mongoose.connect('mongodb+srv://' + process.env.DB_USER + ':' +
    process.env.DB_PASSWORD + '@cluster0.xurzo.mongodb.net/' + process.env.DB_NAME + '?
    retryWrites=true&w=majority',
11      {
12          useNewUrlParser: true,
13          useUnifiedTopology: true
14      })
15      .then(() => console.log('Connexion à MongoDB réussie !'))
16      .catch(() => console.log('Connexion à MongoDB échouée !'));
17
18  const app = express(); // Permet l'utilisation du framework Express
19
20  app.use((req, res, next) => { // Middleware permettant d'éviter les erreurs CORS
    (Cross Origin Resource Sharing)
21      res.setHeader('Access-Control-Allow-Origin', '*');
22      res.setHeader('Access-Control-Allow-Headers', 'Origin, X-Requested-With, Content,
    Accept, Content-Type, Authorization');
23      res.setHeader('Access-Control-Allow-Methods', 'GET, POST, PUT, DELETE, PATCH,
    OPTIONS');
24      next();
25  });
26
27  app.use(express.json()); // Permet de ne plus utiliser body-parser qui est inclut à
    présent dans Express
28
29  app.use('/images', express.static(path.join(__dirname, 'images'))); // Signale à
    Express qu'on utilise les images de manière statique
30
31  app.use('/api/sauces', saucesRoutes);
32  app.use('/api/auth', userRoutes);
33
34  module.exports = app;
```

```
{
  "name": "backend",
  "version": "1.0.0",
  "description": "",
  "main": "server.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "bcrypt": "^5.0.1",
    "dotenv": "^10.0.0",
    "express": "^4.17.1",
    "jsonwebtoken": "^8.5.1",
    "mongoose": "^6.0.13",
    "mongoose-unique-validator": "^3.0.0",
    "multer": "^1.4.3"
  }
}
```

```
1   /node_modules
2
3   /images
4
5   .env
```

```
1   // Création du serveur Node
2
3   const http = require('http');
4   const app = require('./app');
5
6   const normalizePort = val => {
7       const port = parseInt(val, 10);
8
9       if (isNaN(port)) {
10          return val;
11      }
12      if (port >= 0) {
13          return port;
14      }
15      return false;
16  };
17  const port = normalizePort(process.env.PORT || '3000'); // NormalizePort permet de
    renvoyer un port valide, qu'il soit fourni sous la forme d'un numéro ou d'une chaîne
18  app.set('port', port);
19
20  const errorHandler = error => { // Recherche les différentes erreurs pour les gérer
    de manière adéquate selon leur type
21      if (error.syscall !== 'listen') {
22          throw error;
23      }
24      const address = server.address();
25      const bind = typeof address === 'string' ? 'pipe ' + address : 'port: ' + port;
26      switch (error.code) {
27          case 'EACCES':
28              console.error(bind + ' requires elevated privileges.');
29              process.exit(1);
30              break;
31          case 'EADDRINUSE':
32              console.error(bind + ' is already in use.');
33              process.exit(1);
34              break;
35          default:
36              throw error;
37      }
38  };
39
40  const server = http.createServer(app);
41
42  server.on('error', errorHandler);
43  server.on('listening', () => { // Ecouteur d'événement qui consigne dans la console
    le port sur lequel le serveur s'exécute
44      const address = server.address();
45      const bind = typeof address === 'string' ? 'pipe ' + address : 'port ' + port;
46      console.log('Listening on ' + bind);
47  });
48
49  server.listen(port);
```

```
1   const express = require('express');
2   const router = express.Router();
3   const userCtrl = require('../controllers/user'); // Importation du controlleur
    contenant la logique liée à la création ou connexion d'un compte => logique de routing
    claire et explicite
4
5   router.post('/signup', userCtrl.signup);
6   router.post('/login', userCtrl.login);
7
8   module.exports = router;
```

```
1   const express = require('express');
2   const router = express.Router();
3
4   const saucesCtrl = require('../controllers/sauces'); // Importation des controlleurs
    et middlewares contenant la logique métier => logique de routing claire et explicite
5   const auth = require('../middleware/auth');
6   const multer = require('../middleware/multer-config');
7
8   router.post('/', auth, multer, saucesCtrl.createSauce); // Requêtes authentifiées
    même pour l'envoi d'images (également au niveau de la modification d'une sauce)
9   router.post('/:id/like', auth, saucesCtrl.likeOrDislike);
10  router.put('/:id', auth, multer, saucesCtrl.modifySauce);
11  router.delete('/:id', auth, saucesCtrl.deleteSauce);
12  router.get('/:id', auth, saucesCtrl.getOneSauce);
13  router.get('/', auth, saucesCtrl.getAllSauces);
14
15  module.exports = router;
```

```
1   const mongoose = require('mongoose');
2   const uniqueValidator = require('mongoose-unique-validator');
3
4   const userSchema = mongoose.Schema({ // Schéma mongoose strict permettant la création
    d'utilisateur
5       email: { type: String, required: true, unique: true },
6       password: { type: String, required: true }
7   });
8
9   userSchema.plugin(uniqueValidator); // Plugin permettant l'unicité d'une adresse mail
    pour la création d'un compte
10
11  module.exports = mongoose.model('User', userSchema);
```

```
1   const mongoose = require('mongoose');
2
3   const sauceSchema = mongoose.Schema({ // Schéma mongoose strict permettant la
    création d'une sauce
4       userId: { type: String, required: true },
5       name: { type: String, required: true },
6       manufacturer: { type: String, required: true },
7       description: { type: String, required: true },
8       mainPepper: { type: String, required: true },
9       imageUrl: { type: String, required: true },
10      heat: { type: Number, required: true },
11      likes: { type: Number, required: true },
12      dislikes: { type: Number, required: true },
13      usersLiked: { type: Array, required: true },
14      usersDisliked: { type: Array, required: true },
15  });
16
17  module.exports = mongoose.model('Sauce', sauceSchema);
```

```
1   const jwt = require('jsonwebtoken');
2
3   module.exports = (req, res, next) => {
4       try {
5           const token = req.headers.authorization.split(' ')[1]; // Récupération du
    token dans le header authorization
6           const decodedToken = jwt.verify(token, process.env.TOKEN); // On décode le
    token avec la fonction VERIFY
7           const userId = decodedToken.userId; // On récupère le userId présent dans le
    token
8           req.auth = { userId };
9           if (req.body.userId && req.body.userId !== userId) { // On vérifie le userId
    du corps de la requete avec celui du token
10              throw 'User ID non valable!';
11          } else {
12              //console.log(token) a utiliser pour verifier sur postman les requetes
    HTTP
13              next();
14          }
15      } catch (error) {
16          res.status(403).json({ error: error | 'Unauthorized request' });
17      }
18  };
```

```js
const multer = require('multer');

const MIME_TYPES = { // Dictionnaire mime types pour ceux disponibles depuis le frontend
    'image/jpg': 'jpg',
    'image/jpeg': 'jpg',
    'image/png': 'png'
};

const storage = multer.diskStorage({ // Objet de configuration pour multer. Nécessite 2 élements: destination et filename
    destination: (req, file, callback) => {
        callback(null, 'images')
    },
    filename: (req, file, callback) => { // Générer le nouveau nom
        const name = file.originalname.split(' ').join('_'); // Partie avant l'extension
        const extension = MIME_TYPES[file.mimetype]; // Extension
        callback(null, name + Date.now() + '.' + extension); // Nom global: partie avant l'extension + Timestamp créant un fichier unqiue + extension
    }
});

module.exports = multer({ storage: storage }).single('image'); // Export du middleware configuré avec méthode multer précisant l'unicité du fichier et le type image
```

```
1   const bcrypt = require('bcrypt');
2   const jwt = require('jsonwebtoken');
3
4   const User = require('../models/User');
5
6
7   exports.signup = (req, res, next) => {
8       bcrypt.hash(req.body.password, 10)
9           .then(hash => {
10              const user = new User({
11                  email: req.body.email,
12                  password: hash // Hash créé par BCRYPT permettant de ne pas stocker
    directement le mot de passe dans la base de données
13              });
14              user.save()
15                  .then(() => res.status(201).json({ message: 'Utilisateur créé!' }))
16                  .catch(error => res.status(400).json({ error }));
17          })
18          .catch(error => res.status(500).json({ error }));
19
20  };
21
22  exports.login = (req, res, next) => {
23      User.findOne({ email: req.body.email })
24          .then(user => {
25              if (!user) {
26                  return res.status(401).json({ error: 'Utilisateur non trouvé! ' });
27              }
28              bcrypt.compare(req.body.password, user.password)
29                  .then(valid => {
30                      if (!valid) {
31                          return res.status(401).json({ error: 'Mot de passe incorrect!
    ' });
32                      }
33                      res.status(200).json({
34                          userId: user._id,
35                          token: jwt.sign( // Token signé créé par JSONWEBTOKEN
    comprenant l'id de l'utilisateur avec une expiration au bout de 24h
36                              { userId: user._id },
37                              process.env.TOKEN,
38                              { expiresIn: '24h' }
39                          )
40                      });
41                  })
42                  .catch(error => res.status(500).json({ error }));
43          })
44          .catch(error => res.status(500).json({ error }));
45
46  };
```

```
1  const Sauce = require('../models/Sauce');
2  const fs = require('fs'); // Importation du package fs (file system) de Node. Permet
   d'accéder aux fonctions de modification du système de fichiers
3
4  exports.createSauce = (req, res, next) => {
5    const sauceObject = JSON.parse(req.body.sauce);
6    delete sauceObject._id;
7    const sauce = new Sauce({
8      ...sauceObject,
9      imageUrl: `${req.protocol}://${req.get('host')}/images/${req.file.filename}`, //
   Résolution complète de l'URL de l'image
10     // req.protocol => premier segment, req.get('host') => résolution de l'hôte,
   dossier images, req.file.filename => nom du fichier
11     likes: 0,
12     dislikes: 0,
13     usersLiked: [],
14     usersDisliked: []
15   });
16   sauce.save()
17     .then(() => res.status(201).json({ message: 'Sauce enregistrée' }))
18     .catch(error => res.status(400).json({ error }));
19 };
20
21 exports.modifySauce = (req, res, next) => {
22   const sauceObject = req.file ? // L'opérateur ternaire permet de traiter 2 cas.
   L'image a ou n'a pas été modifiée.
23     {
24       ...JSON.parse(req.body.sauce),
25       imageUrl: `${req.protocol}://${req.get('host')}/images/${req.file.filename}`
26     } : { ...req.body };
27   Sauce.updateOne({ _id: req.params.id }, { ...sauceObject, _id: req.params.id })
28     .then(() => res.status(200).json({ message: 'Sauce modifiée' }))
29     .catch(error => res.status(400).json({ error }));
30 };
31
32 exports.deleteSauce = (req, res, next) => {
33   Sauce.findOne({ _id: req.params.id })
34     .then(sauce => { // Mise à jour d'une faille de sécurité: on vérifie que la sauce
   a bien été créée par l'utilisateur faisant la requête
35       if (!sauce) {
36         return res.status(404).json({ error: new Error('Objet non trouvé!') });
37       }
38       if (sauce.userId !== req.auth.userId) {
39         return res.status(401).json({ error: new Error('Requête non autorisée!') });
40       }
41       const filename = sauce.imageUrl.split('/images/')[1]; // Récupération du 2ième
   élément du tableau créé par split
42       fs.unlink(`images/${filename}`, () => {
43         Sauce.deleteOne({ _id: req.params.id })
44           .then(() => res.status(200).json({ message: 'Sauce supprimée' }))
45           .catch(error => res.status(400).json({ error }));
46       });
47     })
48     .catch(error => res.status(500).json({ error }));
49 };
50
51 exports.getOneSauce = (req, res, next) => {
52   Sauce.findOne({ _id: req.params.id })
53     .then(
54       (sauce) => { res.status(200).json(sauce); })
55     .catch((error) => { res.status(404).json({ error: error }); });
56 };
57
58 exports.getAllSauces = (req, res, next) => {
59   Sauce.find()
60     .then(sauces => res.status(200).json(sauces))
```

```
61        .catch(error => res.status(400).json({ error }));
62    };
63
64    exports.likeOrDislike = (req, res, next) => {
65
66      let like = req.body.like
67
68      let userId = req.body.userId
69
70      let sauceId = req.params.id
71
72      if (like === 1) {
73        Sauce.updateOne({ _id: sauceId }, { $push: { usersLiked: userId }, $inc: { likes:
    +1 } })
74          .then(() => res.status(200).json({ message: 'LIKE!' }))
75          .catch(error => res.status(400).json({ error }))
76      }
77      if (like === -1) {
78        Sauce.updateOne({ _id: sauceId }, { $push: { usersDisliked: userId }, $inc:
    { dislikes: +1 } })
79          .then(() => res.status(200).json({ message: 'DISLIKE!' }))
80          .catch(error => res.status(400).json({ error }))
81      }
82      if (like === 0) {
83        Sauce.findOne({ _id: sauceId })
84          .then((sauce) => {
85            if (sauce.usersLiked.includes(userId)) {
86              Sauce.updateOne({ _id: sauceId }, { $pull: { usersLiked: userId }, $inc:
    { likes: -1 } })
87                .then(() => res.status(200).json({ message: 'LIKE supprimé!' }))
88                .catch(error => res.status(400).json({ error }))
89            }
90            if (sauce.usersDisliked.includes(userId)) {
91              Sauce.updateOne({ _id: sauceId }, { $pull: { usersDisliked: userId }, $inc:
    { dislikes: -1 } })
92                .then(() => res.status(200).json({ message: 'DISLIKE supprimé!' }))
93                .catch(error => res.status(400).json({ error }))
94            }
95          })
96          .catch(error => res.status(404).json({ error }))
97      }
98    }
```

# PIIQUANTE

## Construisez une API sécurisée pour une application d'avis gastronomiques

*Projet 6 de la formation Développeur web d'OpenClassRooms.*

Le projet consiste à développer le backend d'une application permettant d'ajouter des sauces afin de les partager avec d'autres utilisateurs. Il est également possible de liker ou disliker les sauces.

Le frontend est fourni et a été développé/compilé à l'aide d'Angular. Il nous est demandé de créer une API en utilisant Node, le framework Express et une base de données afin de stocker les utilisateurs et sauces créés.

## Compétences évaluées

#### La création de cette API permet de:

- Stocker des données de manière sécurisée
- Implémenter un modèle logique de données conformément à la réglementation
- Mettre en œuvre des opérations CRUD de manière sécurisée

## Exigences de sécurité

*Il est primordial de porter une attention particulière sur la sécurité de l'application.*

- Le mot de passe de l'utilisateur doit être haché.
- L'authentification doit être renforcée sur toutes les routes sauce requises.
- Les adresses électroniques dans la base de données sont uniques et un plugin Mongoose approprié est utilisé pour garantir leur unicité et signaler les erreurs.
- La sécurité de la base de données MongoDB (à partir d'un service tel que MongoDB Atlas) ne doit pas empêcher l'application de se lancer sur la machine d'un utilisateur.
- Un plugin Mongoose doit assurer la remontée des erreurs issues de la base de données.
- Les versions les plus récentes des logiciels sont utilisées avec des correctifs de sécurité actualisés.
- Le contenu du dossier images ne doit pas être téléchargé sur GitHub.

## Comment utiliser l'application?

#### Les installations et commandes suivantes sont nécessaires:

- Installer Node, Sass, Npm sur votre poste de travail
- Télécharger ou cloner le projet
- Aller dans le frontend (via un terminal) et faire `npm install`
- Faire `npm start` , le serveur se lance sur http://localhost:8081
- Aller dans le backend (via un terminal) et faire `npm install`
- Toujours dans le dossier backend, créer un fichier .env avec les données fournies par le développeur (DB_*USER*, *DB*_PASSWORD, DB_NAME, TOKEN) (ignoré avec gitignore pour des raisons de sécurité)
- Créer également un dossier /images à la racine (ignoré avec gitignore pour optimiser la taille du projet)
- Faire `node server`(ou nodemon server si celui-ci est installé). La connexion se fait sur le port 3000.

#### Sécurisation de l'application et des données?

- bcrypt (hash et salage du mot de passe utilisateur)
- jsonwebtoken (sécurisation de l'authentification pour les requêtes )
- mongoose (schémas stricts)
- mongoose-unique-validator (création d'un compte par adresse mail)
- dotenv (sécurisation des informations de connexion et d'accès à la base de données)
- dossier images et dotenv non téléchargés sur github (via gitignore)

```
58  - helmet et cookie-session auraient été judicieux et sont préconisés dans les
    recommandations de sécurité d'Express mais ils nécessitaient quelques ajustements
    côté frontend
59
60  ## Environnement de développement
61
62  ####
63
64  - Visual Studio Code
65  - Node / Sass / Npm
66  - Nodemon
67  - Express
68  - Mongo DB / Mongo Atlas / Mongoose
69
70  #### Pour en savoir plus sur la sécurité informatique (développement/production/
    utilisation):
71
72  - [Framework Express: meilleures pratiques en production](https://expressjs.com/fr/
    advanced/best-practice-security.html)
73  - [GitHub Advisory Database](https://github.com/advisories)
74  - [OWASP Top 10 Web Application Security Risks](https://owasp.org/www-project-top-
    ten/)
```