🍽️ API RESTful pour la gestion d'un système de livraison de repas

Ce projet est une API RESTful construite avec **Node.js**, **Express** et **MySQL**. Il gère les entités suivantes : administrateurs, clients, restaurants, livreurs, commandes, plats, livraisons, et catégories.

🚀 Fonctionnalités

- CRUD complet pour chaque entité (Administrateur, Client, Commande, etc.)
- Architecture MVC bien structurée
- Utilisation de routes modulaire avec Express
- Connexion sécurisée à une base de données MySQL
- Middleware pour la gestion des erreurs
- Chargement de configuration via `.env`

📦 Prérequis

- Node.js (v14+ recommandé)
- MySQL Server
- Postman ou tout autre client API
- Un éditeur de code (VS Code conseillé)

📁 Structure du projet

```
.
├── config/
│   └── database.js
├── controllers/
│   ├── administrateurController.js
│   └── ...
├── models/
│   ├── administrateurModel.js
│   └── ...
├── routes/
│   ├── administrateurRoutes.js
│   └── ...
├── .env
├── package.json
└── server.js
```

🛠️ Installation et configuration

1. Cloner le projet

```
git clone https://github.com/ton-utilisateur/ton-repo.git
cd ton-repo
```

2. Installer les dépendances

```
npm install
```

3. Créer un fichier `.env`

Dans la racine du projet :

```
DB_HOST=localhost
DB_USER=root
DB_PASSWORD=ton_mot_de_passe
DB_NAME=nom_de_la_base
PORT=3000
```

4. Démarrer le serveur

```
npm start
```

Endpoints de l'API

Base URL : `http://localhost:3000/api/v1`

Ressource	GET	POST	PUT	DELETE
/administrateurs	Tous	Créer		
/administrateurs/:id	Un			
/clients				
/commandes				
/livreurs				
/livraisons				
/plats				
/restaurants				
/categories				

Sécurité

- Pas encore d'authentification (peut être ajoutée avec JWT)
- Les requêtes SQL sont sécurisées grâce aux paramètres préparés

---

Exemple d'appel avec Postman

POST /api/v1/administrateurs

```
{
  "nom": "Doe",
```

```
  "prenom": "John",
  "email": "john.doe@example.com"
}
```

Gestion des erreurs

Une gestion d'erreur simple est implémentée dans `server.js` :

```
app.use((err, req, res, next) => {
  console.error('Erreur non gérée:', err.stack);
  res.status(500).json({ message: 'Une erreur interne est survenue' });
});
```

Suggestions d'améliorations futures

- Authentification JWT
- Swagger pour documentation
- Validation des entrées (Joi / express-validator)
- Tests unitaires avec Jest

Auteur

Junior Okawe – Projet réalisé dans un contexte de formation pratique sur les APIs REST.