🚗 Plateforme E-Commerce Porsche - API Backend

API REST complète et sécurisée pour une plateforme e-commerce dédiée aux véhicules Porsche. Gestion des voitures neuves, d'occasion, des accessoires et des commandes avec paiement Stripe intégré.

🎯 Démarrage Rapide

Installation en 3 Étapes

```
# 1. Installer les dépendances
npm install

# 2. Configurer l'environnement
cp .env.example .env
# Éditer .env avec vos paramètres

# 3. Démarrer le serveur
npm start
```

Tester avec Postman (2 minutes)

```
# 1. Ouvrir Postman
# 2. Import → Glisser ces fichiers :
#    - Porsche_API.postman_collection.json
#    - Porsche_API_Dev.postman_environment.json
# 3. Sélectionner environnement "Porsche API - Development"
# 4. Tester : 📁 00 - Authentification → Inscription → Send
# ✅ Prêt !
```

📚 Documentation

Ce projet contient une documentation complète organisée en plusieurs fichiers :

📖 Documentation Principale

Fichier	Description	Lien
📋 **DOCUMENTATION_INDEX.md**	**Point d'entrée - Liste tous les docs**	Voir →
📘 **README_API.md**	Documentation API complète	Voir →
🚀 **POSTMAN_GUIDE.md**	Guide Postman pas à pas	Voir →
🎨 **POSTMAN_VISUAL_GUIDE.md**	Guide visuel rapide	Voir →
📦 **POSTMAN_FILES.md**	Description fichiers Postman	Voir →

🎯 Par où commencer ?

👉 **Nouveau sur le projet ?** Consultez DOCUMENTATION_INDEX.md

---

✨ Fonctionnalités Principales

🔐 Authentification & Sécurité

- JWT (JSON Web Tokens)
- Bcrypt pour les mots de passe
- Rate limiting
- Helmet pour sécurité HTTP
- Rôles utilisateur/admin

👨‍💼 Espace Admin

- ✅ CRUD complet des voitures (neuves/occasion)
- ✅ CRUD des modèles Porsche
- ✅ CRUD des accessoires
- ✅ Gestion des couleurs (extérieur/intérieur)
- ✅ Gestion des tailles de jante
- ✅ Upload de photos multiples
- ✅ Supervision des réservations et commandes

👤 Espace Utilisateur

- ✅ Création et gestion de compte
- ✅ Ma Porsche actuelle (CRUD)
- ✅ Proposition de vente (conseil)
- ✅ Réservation de voitures d'occasion
- ✅ Commande de voitures neuves (acompte)
- ✅ Achat d'accessoires
- ✅ Gestion du panier
- ✅ Annulation (avant paiement)

💳 Paiement

- Intégration Stripe
- Paiements sécurisés
- Webhooks automatiques
- Gestion des sessions

🛠️ Technologies

- **Backend** : Node.js 18+, Express.js 5.1
- **Base de données** : MongoDB 8.19 avec Mongoose
- **Authentification** : JWT, bcrypt
- **Validation** : express-validator, Joi
- **Upload** : Multer
- **Paiement** : Stripe
- **Sécurité** : Helmet, express-rate-limit, mongo-sanitize
- **Testing** : Postman

📋 Structure du Projet

```
plateforme_porsche/Node/
├── 📁 controllers/          # Logique métier
├── 📁 models/               # Schémas MongoDB
├── 📁 routes/               # Définition des routes
├── 📁 middlewares/          # Auth, validation, upload
├── 📁 validations/          # Schémas de validation
├── 📁 db/                   # Configuration DB
├── 📁 uploads/              # Fichiers uploadés
├── 📄 server.js             # Point d'entrée
├── 📄 package.json          # Dépendances
├── 📄 .env                  # Configuration (à créer)
│
├── 📘 README.md             # Ce fichier
├── 📋 DOCUMENTATION_INDEX.md  # Index de la doc
├── 📖 README_API.md         # Doc API complète
├── 🚀 POSTMAN_GUIDE.md      # Guide Postman
├── 🎨 POSTMAN_VISUAL_GUIDE.md  # Guide visuel
├── 📦 POSTMAN_FILES.md      # Description Postman
│
├── 📦 Porsche_API.postman_collection.json
└── 📦 Porsche_API_Dev.postman_environment.json
```

⚙️ Configuration

Créez un fichier `.env` à la racine :

```
# Serveur
PORT=3000
NODE_ENV=development

# MongoDB
MONGODB_URI=mongodb://localhost:27017/porsche_ecommerce

# JWT
JWT_SECRET=votre_secret_jwt_tres_securise

# Stripe
STRIPE_SECRET_KEY=sk_test_votre_cle
STRIPE_PUBLISHABLE_KEY=pk_test_votre_cle
STRIPE_WEBHOOK_SECRET=whsec_votre_webhook

# Frontend
FRONTEND_URL=http://localhost:3001
```

🚀 Scripts Disponibles

```
# Démarrer en production
npm start

# Démarrer en mode développement
npm run dev

# Créer un compte admin
npm run create:admin

# Tests
npm test
npm run test:complete
npm run test:admin
npm run test:user

# Nettoyer la DB
npm run clean
```

🧪 Tester l'API

Avec Postman (Recommandé)

1. **Importer la collection** : `Porsche_API.postman_collection.json`
2. **Importer l'environnement** : `Porsche_API_Dev.postman_environment.json`
3. **Sélectionner l'environnement** (en haut à droite)

4. **Tester** !

📘 **Guide complet** : POSTMAN_GUIDE.md

Avec cURL

```
# Inscription
curl -X POST http://localhost:3000/user/register \
  -H "Content-Type: application/json" \
  -d '{
    "nom": "Dupont",
    "prenom": "Jean",
    "email": "jean.dupont@example.com",
    "password": "MotDePasse123!",
    "telephone": "+33612345678",
    "adresse": "123 Rue de Paris"
  }'

# Connexion
curl -X POST http://localhost:3000/user/login \
  -H "Content-Type: application/json" \
  -d '{
    "email": "jean.dupont@example.com",
    "password": "MotDePasse123!"
  }'
```

📖 Documentation API Détaillée

Consultez README_API.md pour :

- 📍 Tous les endpoints (60+)
- 🔐 Authentification détaillée
- 📊 Modèles de données
- 🎯 Workflows complets
- 🚨 Codes d'erreur
- 🔒 Sécurité

🔐 Sécurité

- ✅ **JWT** : Tokens sécurisés avec expiration
- ✅ **Bcrypt** : Hash des mots de passe (10 rounds)
- ✅ **Rate Limiting** : Protection anti-DDoS
- ✅ **Helmet** : Headers HTTP sécurisés
- ✅ **CORS** : Contrôle des origines
- ✅ **Validation** : Joi + express-validator
- ✅ **Sanitization** : Protection injection MongoDB

🎯 Exemples de Workflows

Workflow Admin

```
1. Connexion Admin
2. Créer couleur extérieure
3. Créer couleur intérieure
4. Créer taille de jante
5. Créer modèle Porsche
6. Associer les options au modèle
7. Créer voiture (neuve/occasion)
8. Ajouter photos
```

Workflow Utilisateur

```
1. Inscription / Connexion
2. Créer ma Porsche actuelle
3. Configurer (couleurs, jantes)
4. Proposer en vente (optionnel)
5. Réserver une voiture d'occasion
6. Commander (voiture neuve ou accessoires)
7. Payer via Stripe
```

📊 Base de Données

Collections MongoDB

- **users** : Utilisateurs et admins
- **model_porsche** : Modèles Porsche (admin)
- **model_porsche_actuel** : Porsches utilisateurs
- **voiture** : Voitures neuves/occasion
- **accesoire** : Accessoires
- **reservation** : Réservations
- **commande** : Commandes
- **ligneCommande** : Lignes de commande
- **couleur_exterieur** : Couleurs extérieures
- **couleur_interieur** : Couleurs intérieures
- **couleur_accesoire** : Couleurs accessoires
- **taille_jante** : Tailles de jante
- **photo_*** : Photos (divers types)

🌐 Endpoints Principaux

```
BASE_URL: http://localhost:3000

Auth:
  POST   /user/register          # Inscription
  POST   /user/login             # Connexion

Admin:
  POST   /couleur_exterieur/new  # Créer couleur
  POST   /model_porsche/new      # Créer modèle
  POST   /voiture/new            # Créer voiture
  POST   /accesoire/new          # Créer accessoire

User:
  POST   /model_porsche_actuel/new   # Ma Porsche
  POST   /reservation/new             # Réserver
  POST   /commande/new                # Commander
  POST   /api/payment/checkout/:id    # Payer
```

📖 **Documentation complète** : README_API.md

🆘 Support & Aide

Documentation

- 📋 Index Documentation
- 📘 API Complète
- 🚀 Guide Postman

Contact

- **Email** : support@porsche-api.com
- **Issues** : GitHub Issues

🤝 Contribution

Les contributions sont les bienvenues ! Merci de :

1. Fork le projet
2. Créer une branche (`git checkout -b feature/AmazingFeature`)
3. Commit vos changements (`git commit -m 'Add AmazingFeature'`)
4. Push vers la branche (`git push origin feature/AmazingFeature`)
5. Ouvrir une Pull Request

📄 Licence

ISC License - Voir le fichier `LICENSE` pour plus de détails.

🙏 Remerciements

- **Express.js** pour le framework
- **MongoDB** pour la base de données
- **Stripe** pour les paiements
- **Postman** pour les tests
- La communauté open source

🗺️ Roadmap

- [ ] Documentation Swagger/OpenAPI
- [ ] Tests unitaires (Jest)
- [ ] CI/CD GitHub Actions
- [ ] Docker containerization
- [ ] Logs avancés (Winston)
- [ ] Cache Redis
- [ ] GraphQL API

Développé avec ❤️ pour les passionnés de Porsche

🚗💨 Bon développement !