Configuration API React ↔ Node.js ↔ MongoDB

✅ Stack Vérifiée et Fonctionnelle

Composants Installés

- **MongoDB**: Port 27017, base de données `porsche`
- **Node.js API**: Port 3000, Express avec CORS activé
- **React Frontend**: Vite + React 19

🔧 Configuration Réalisée

1. Backend (Node.js)

Le serveur est configuré dans `/Node/server.js`:

- ✅ MongoDB connecté via Mongoose
- ✅ CORS activé pour toutes origines
- ✅ Rate limiting configuré
- ✅ Helmet pour la sécurité
- ✅ Endpoints REST disponibles

2. Frontend (React)

Fichiers créés:

`/src/config/api.js`

- Configuration axios avec intercepteurs
- Gestion automatique du token JWT
- URL de base: `http://localhost:3000`

`/src/services/authService.js`

- Méthodes: login, register, logout
- Gestion du token dans localStorage
- Vérification d'authentification

`/src/services/porscheService.js`

- Méthodes pour récupérer:
  - Modèles Porsche
  - Voitures
  - Accessoires
  - Couleurs (intérieur/extérieur)
  - Packages
  - Tailles de jante

- Sièges

`/src/components/ApiTest.jsx`

- Composant de test de connexion
- Vérification du statut API
- Test de chargement de données

`.env`

```
VITE_API_URL=http://localhost:3000
```

🚀 Démarrage

Terminal 1 - Backend (Node.js)

```
cd Node
npm start
```

Le serveur démarre sur http://localhost:3000

Terminal 2 - Frontend (React)

```
cd React
npm run dev
```

L'application React démarre sur http://localhost:5173

📡 Utilisation des Services

Exemple: Récupérer les modèles Porsche

```
import porscheService from "./services/porscheService";

// Dans un composant React
const loadModels = async () => {
  try {
    const models = await porscheService.getAllModels();
    console.log(models);
  } catch (error) {
    console.error("Erreur:", error);
```

```
  }
};
```

Exemple: Authentification

```
import authService from "./services/authService";

// Login
const handleLogin = async (email, password) => {
  try {
    const data = await authService.login(email, password);
    console.log("Connecté:", data.user);
  } catch (error) {
    console.error("Erreur de connexion:", error);
  }
};

// Vérifier si connecté
if (authService.isAuthenticated()) {
  const user = authService.getCurrentUser();
  console.log("Utilisateur actuel:", user);
}
```

🔍 Endpoints API Disponibles

Utilisateurs

- `POST /user/register` - Inscription
- `POST /user/login` - Connexion
- `GET /user` - Liste des utilisateurs (admin)

Modèles & Voitures

- `GET /model_porsche` - Tous les modèles
- `GET /model_porsche/:id` - Modèle par ID
- `GET /voiture` - Toutes les voitures
- `GET /voiture/:id` - Voiture par ID

Accessoires

- `GET /accesoire` - Tous les accessoires
- `GET /photo_accesoire` - Photos d'accessoires
- `GET /couleur_accesoire` - Couleurs accessoires

Configuration

- `GET /couleur_interieur` - Couleurs intérieur

- `GET /couleur_exterieur` - Couleurs extérieur
- `GET /taille_jante` - Tailles de jante
- `GET /siege` - Types de sièges
- `GET /package` - Packages disponibles

Commandes

- `POST /commande` - Créer une commande
- `GET /commande` - Liste des commandes
- `GET /reservation` - Réservations

Paiement

- `POST /api/payment/create-payment-intent` - Stripe

🛡️ Sécurité

Fonctionnalités Activées

- ✅ CORS configuré
- ✅ Helmet (headers sécurisés)
- ✅ Rate limiting (protection DDoS)
- ✅ JWT pour authentification
- ✅ Validation des données

Limiteurs de taux

- **Global**: 100 requêtes/15 min
- **Login**: 10 tentatives/15 min
- **Register**: 5 inscriptions/heure
- **Upload**: 50 uploads/heure
- **Payment**: 20 tentatives/heure

🧪 Tests

Test Manuel de l'API

```
# Test de la route racine
curl http://localhost:3000/

# Test endpoint modèles
curl http://localhost:3000/model_porsche

# Test avec token
curl -H "Authorization: Bearer YOUR_TOKEN" http://localhost:3000/user
```

Test avec le Composant React

1. Ouvrir http://localhost:5173
2. Cliquer sur "🔄 Retester" pour vérifier la connexion
3. Cliquer sur "📥 Charger" pour tester les données

📦 Variables d'Environnement

Backend (.env dans /Node)

```
PORT=3000
DB_URI=mongodb://localhost:27017/porsche
SECRET_KEY=your_secret_key
JWT_EXPIRE=24h
FRONTEND_URL=http://localhost:5173/
STRIPE_PUBLISHABLE_KEY=pk_test_...
STRIPE_SECRET_KEY=sk_test_...
```

Frontend (.env dans /React)

```
VITE_API_URL=http://localhost:3000
```

❌ Résolution de Problèmes

MongoDB ne démarre pas

```
brew services start mongodb-community
mongosh --eval 'db.runCommand({ping:1})'
```

CORS Errors

Vérifier que le serveur Node.js utilise `cors()` et que FRONTEND_URL est correct.

Token invalide

Le token JWT expire après 24h. Reconnecter l'utilisateur.

Port déjà utilisé

```
# Trouver le processus
lsof -i :3000
```

```
# Tuer le processus
kill -9 PID
```

📝 Notes Importantes

1. **Toujours démarrer MongoDB avant Node.js**
2. **Le token JWT est stocké dans localStorage**
3. **CORS est ouvert à toutes origines (à restreindre en prod)**
4. **Les uploads sont limités à 50/heure**
5. **Utiliser HTTPS en production**

🎯 Prochaines Étapes

- [ ] Créer les pages de l'application
- [ ] Implémenter le panier d'achat
- [ ] Ajouter le système de paiement Stripe
- [ ] Créer les pages admin
- [ ] Optimiser les images
- [ ] Ajouter des tests unitaires
- [ ] Déployer en production

✅ **Status Actuel**: Tout est connecté et fonctionnel !