VanPlanner - Guide de Déploiement

Introduction

Ce guide explique comment publier le projet monorepo VanPlanner en production, de manière simple et progressive. Même si vous n'êtes pas technique, vous pouvez suivre les étapes "straightforward".

1. Structure du projet

VanPlanner/
+-- apps/
| +-- api/   -> Backend (NestJS + Prisma)
| +-- web/   -> Frontend (React + Vite)
+-- packages/   -> Packages partagés (si existants)
+-- prisma/   -> Schéma Prisma et seed
+-- node_modules/
+-- package.json   -> Configuration du monorepo
+-- .env   -> Variables d'environnement

Remarques :
- apps/api contient le serveur NestJS et Prisma.
- apps/web contient le frontend React + Vite.
- prisma/schema.prisma définit la structure de la base de données.

2. Préparation de l’environnement serveur

1þ ã Installer Node.js (v22.x ou supérieur) sur le serveur.
2þ ã Installer PostgreSQL et créer la base de données "vanplanner".
3þ ã Copier les fichiers du projet sur le serveur (via Git ou FTP/SFTP).
4þ ã Créer un fichier .env sur le serveur avec les variables :
   - DATABASE_URL="postgresql://USER:PASSWORD@HOST:PORT/vanplanner"
   - PORT=3000
   - JWT_SECRET=<clé super secrète>
   - ALLOWED_ORIGINS=http://mon-site.fr,https://mon-vanplanner.com
   - VITE_API_URL=http://adresse-du-backend:3000

3. Installation des dépendances

Le projet utilise pnpm et un monorepo. Depuis la racine du projet :
1þ ã Installer pnpm si ce n’est pas déjà fait :
npm install -g pnpm

2þ ã Installer toutes les dépendances pour le monorepo :
pnpm install

4. Génération du client Prisma et migration BD

1þ ã Générer le client Prisma :
cd apps/api
npx prisma generate

2þ ã Appliquer les migrations (création des tables) :
npx prisma migrate deploy

3þ ã Optionnel : remplir la table Van avec les modèles pré-définis (pour que savePlan fonctionne) :
npx ts-node prisma/seed.ts

5. Build et déploiement du frontend

1þ ã Aller dans le dossier frontend :
cd apps/web

2þ ã Installer les dépendances si nécessaire :
pnpm install

3þ ã Build du frontend pour la production :
pnpm run build

4þ ã Les fichiers statiques sont générés dans apps/web/dist
à copier sur le serveur web (Nginx, Apache, etc.)

6. Lancement du backend en production

1þ ã Aller dans apps/api :
cd apps/api

2þ ã Build NestJS pour la production :
pnpm run build

3þ ã Lancer le serveur :
node dist/main.js

4þ ã Vérifier que le serveur écoute bien sur le PORT défini.

7. Configuration CORS et sécurité

- Le fichier main.ts du backend utilise process.env.ALLOWED_ORIGINS pour autoriser les origines.
- Helmet est activé pour sécuriser les headers HTTP.
- Rate-limit pour limiter les abus (max 200 requêtes / 15min / IP).
- Valider que ALLOWED_ORIGINS contient votre domaine frontend en production.

8. Mise en place des services (optionnel)

- Utiliser PM2 ou systemd pour maintenir le backend actif après reboot :
  pm2 start dist/main.js --name vanplanner-api
  pm2 save
- Configurer un serveur web (Nginx ou Apache) pour servir le frontend et proxy les requêtes API vers le backend.

9. Vérifications finales

1þ ã Le frontend s’affiche correctement sur votre domaine.
2þ ã Les appels API vers /plans, /users, etc., fonctionnent.
3þ ã La table Van est bien peuplée (prisma/seed.ts).
4þ ã Les logs du backend ne montrent pas d’erreurs CORS ou 500.

10. Notes importantes

- Toujours sécuriser votre JWT_SECRET.
- ALLOWED_ORIGINS doit être exact pour votre frontend.
- En cas de modification du schema.prisma, exécuter :
  npx prisma generate
  npx prisma migrate dev

' Le guide est terminé. Votre VanPlanner devrait être opérationnel en production !