Guide de déploiement sur Railway pour MPJFUMS

Ce document explique comment l'application MPJFUMS (site web de parfumerie gabonaise) a été déployée sur la plateforme Railway.

Prérequis

- Un compte sur Railway
- Le code source de l'application sur GitHub
- Un fichier `Procfile` dans le projet
- Un fichier `railway.json` pour la configuration Railway

Structure du projet pour Railway

Le projet a été organisé avec les fichiers de configuration suivants pour Railway :

Procfile

Le Procfile indique à Railway quelle commande exécuter pour démarrer l'application :

```
web: cd shop && gunicorn shop.wsgi:application --log-file -
```

railway.json

Ce fichier définit les paramètres de déploiement :

```
{
  "$schema": "https://railway.app/railway.schema.json",
  "build": {
    "builder": "NIXPACKS",
    "buildCommand": "python -m pip install -r requirements.txt"
  },
  "deploy": {
    "startCommand": "cd shop && python manage.py migrate && python manage.py
collectstatic --noinput && gunicorn shop.wsgi:application",
    "healthcheckPath": "/",
    "healthcheckTimeout": 300
  }
}
```

Optimisations pour Railway

Configuration de la base de données PostgreSQL

L'application utilise automatiquement PostgreSQL en production grâce à la détection des variables d'environnement de Railway :

```
# Dans settings.py
if os.environ.get('DATABASE_URL'):
    import dj_database_url
    DATABASES = {
        'default': dj_database_url.config(
            default=os.environ.get('DATABASE_URL'),
            conn_max_age=600,
        )
    }
```

Fichiers statiques avec WhiteNoise

Nous utilisons WhiteNoise pour servir les fichiers statiques, ce qui est crucial pour Railway qui n'a pas de système de stockage séparé :

```
# Dans settings.py
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'whitenoise.middleware.WhiteNoiseMiddleware',  # Middleware WhiteNoise
    # ...
]

STATICFILES_STORAGE = 'whitenoise.storage.CompressedManifestStaticFilesStorage'
```

Utilisation de variables d'environnement

Les variables sensibles comme SECRET_KEY devraient être configurées dans les variables d'environnement de Railway :

1. Aller dans le tableau de bord Railway
2. Sélectionner votre projet
3. Aller dans l'onglet "Variables"
4. Ajouter les variables nécessaires :
    - SECRET_KEY
    - DEBUG=False
    - ALLOWED_HOSTS=.railway.app,mpjfums-production.up.railway.app

Étapes de déploiement

1. **Connecter votre dépôt GitHub à Railway**
    - Créez un nouveau projet sur Railway
    - Choisissez "Deploy from GitHub repo"

    - Sélectionnez votre dépôt

2. **Configurer les variables d'environnement**
    - Ajoutez les variables d'environnement nécessaires

3. **Déploiement**
    - Railway détectera automatiquement votre application Django
    - Il utilisera le fichier railway.json pour la configuration

4. **Base de données**
    - Ajoutez un service PostgreSQL depuis l'interface Railway
    - Railway configurera automatiquement la variable DATABASE_URL

5. **Domaine personnalisé (optionnel)**
    - Dans l'onglet "Settings", configurez un domaine personnalisé pour votre application

Maintenance et mises à jour

Pour mettre à jour l'application, il suffit de pousser les modifications sur la branche principale de votre dépôt GitHub. Railway détectera automatiquement les changements et redéploiera l'application.

Pour surveiller les logs et les performances :

- Utilisez l'onglet "Metrics" pour voir l'utilisation des ressources
- Consultez les logs dans l'onglet "Logs"

Résolution des problèmes courants

1. **Erreur lors du collectstatic** :
    - Vérifiez que STATIC_ROOT est correctement configuré dans settings.py
    - Assurez-vous que WhiteNoise est correctement installé et configuré

2. **Erreurs de base de données** :
    - Vérifiez que les migrations sont à jour
    - Consultez les logs pour voir les erreurs spécifiques

3. **Application qui crash** :
    - Vérifiez les logs pour identifier la cause
    - Utilisez Sentry pour une meilleure surveillance des erreurs