Guide de Déploiement EVIMERIA sur Railway

📋 Prérequis

- Un compte Railway
- Un compte Cloudinary configuré
- Votre code poussé sur GitHub

🚀 Étapes de Déploiement

1. Créer un Nouveau Projet sur Railway

1. Connectez-vous à Railway
2. Cliquez sur "New Project"
3. Sélectionnez "Deploy from GitHub repo"
4. Choisissez votre repository EVIMERIA

2. Configuration de la Base de Données PostgreSQL

Railway va automatiquement détecter que vous avez besoin d'une base de données PostgreSQL et la créera.

Les variables suivantes seront automatiquement configurées par Railway :

- `DATABASE_URL`
- `PGDATABASE`
- `PGHOST`
- `PGPASSWORD`
- `PGPORT`
- `PGUSER`
- `POSTGRES_DB`
- `POSTGRES_PASSWORD`
- `POSTGRES_USER`

3. Variables d'Environnement à Configurer

Dans Railway, allez dans votre service backend et ajoutez ces variables dans l'onglet "Variables" :

Variables Django Essentielles

```
SECRET_KEY=django-insecure-c+h11ba43(notowv31(&=+)5^-h&$_2)9@
DEBUG=False
```

Variables Cloudinary

```
CLOUDINARY_CLOUD_NAME=dmcaguchx
CLOUDINARY_API_KEY=238869761337271
CLOUDINARY_API_SECRET=G1AQ85xIMHSFSLgPOXeNsGFnfJA
```

Variables Optionnelles (déjà configurées par défaut)

```
PORT=8000
PYTHONUNBUFFERED=1
NODE_ENV=production
```

4. Configuration du Domaine

Une fois déployé, Railway vous donnera une URL comme :
`https://votre-projet.up.railway.app`

Mettez à jour votre configuration CORS dans Railway en ajoutant :

```
RAILWAY_STATIC_URL=https://votre-projet.up.railway.app
```

5. Déploiement

1. Poussez votre code sur la branche `main` de votre repository
2. Railway détectera automatiquement les changements et commencera le déploiement
3. Le processus prendra environ 5-10 minutes

🔧 Structure des Fichiers de Configuration

Votre projet utilise plusieurs fichiers de configuration pour Railway :

- `railway.toml` - Configuration principale Railway
- `nixpacks.toml` - Configuration du buildpack
- `backend/Procfile` - Configuration de démarrage (backup)
- `backend/requirements.txt` - Dépendances Python

📊 Monitoring

Healthcheck

L'application inclut un endpoint de healthcheck à `/status/` qui permet à Railway de vérifier que l'application fonctionne correctement.

Logs

Vous pouvez voir les logs en temps réel dans Railway :

1. Allez dans votre projet
2. Cliquez sur votre service backend
3. Onglet "Logs"

🐛 Dépannage

Erreurs Communes

1. Erreur de Base de Données

```
django.db.utils.OperationalError: connection to server
```

Solution : Vérifiez que la variable `DATABASE_URL` est correctement configurée.

2. Erreur Cloudinary

```
cloudinary.exceptions.AuthorizationRequired
```

Solution : Vérifiez vos variables Cloudinary dans Railway.

3. Erreur de fichiers statiques

```
GET /static/... 404
```

Solution : Assurez-vous que le build frontend s'est bien déroulé.

Commands Utiles

Exécuter des migrations manuellement

Dans Railway CLI :

```
railway run python manage.py migrate
```

Créer un superuser

```
railway run python manage.py createsuperuser
```

Vérifier les variables d'environnement

```
railway variables
```

🔐 Sécurité en Production

L'application est configurée avec les mesures de sécurité suivantes en production :

- HTTPS forcé
- Cookies sécurisés
- CORS restreint à votre domaine
- Headers de sécurité configurés
- SSL requis pour la base de données

📈 Performance

Configuration Gunicorn

- 2 workers par défaut
- Timeout de 120 secondes
- Max requests : 1000 avec jitter
- Restart automatique en cas d'erreur

Variables d'environnement pour ajuster les performances

```
GUNICORN_WORKERS=2  # Nombre de workers
GUNICORN_TIMEOUT=120  # Timeout en secondes
```

🚀 Mise à Jour

Pour mettre à jour votre application :

1. Poussez vos changements sur la branche `main`
2. Railway redéploiera automatiquement
3. Les migrations se feront automatiquement

📞 Support

En cas de problème :

1. Vérifiez les logs dans Railway
2. Vérifiez que toutes les variables d'environnement sont configurées
3. Testez en local avec `DEBUG=False` pour reproduire l'environnement de production