EVIMERIA (anciennement JaelleShop)

Boutique e-commerce moderne créée avec Django, React et Cloudinary.

Prérequis

- Docker et Docker Compose
- Un compte Railway pour le déploiement
- Un compte Cloudinary pour la gestion des médias

Structure du Projet

```
evimeria/
├── backend/            # API Django
│   ├── jaelleshop/     # Configurations du projet
│   ├── products/       # App pour les produits
│   ├── users/          # App pour les utilisateurs
│   └── ...
├── frontend/           # Application React
├── Dockerfile          # Configuration Docker
├── docker-compose.yml  # Configuration Docker Compose
└── ...
```

Déploiement sur Railway

1. Préparer l'application

Tout d'abord, assurez-vous que votre application fonctionne correctement localement avec Docker :

```
# Copier le fichier d'exemple des variables d'environnement
cp env.sample .env

# Éditer les variables d'environnement avec vos propres valeurs
nano .env

# Construire et démarrer les conteneurs Docker
docker-compose up --build
```

2. Créer un nouveau projet sur Railway

1. Connectez-vous sur Railway
2. Créez un nouveau projet
3. Choisissez "Deploy from GitHub repo"
4. Sélectionnez votre dépôt GitHub

5. Configurez les variables d'environnement suivantes dans Railway :
    - `DEBUG=False`
    - `SECRET_KEY=<votre_secret_key>`
    - `ALLOWED_HOSTS=<votre_domaine_railway>.up.railway.app,<autres_domaines>`
    - `CLOUDINARY_CLOUD_NAME=<votre_cloudinary_cloud_name>`
    - `CLOUDINARY_API_KEY=<votre_cloudinary_api_key>`
    - `CLOUDINARY_API_SECRET=<votre_cloudinary_api_secret>`

3. Ajouter une base de données PostgreSQL

1. Dans votre projet Railway, allez dans "New"
2. Sélectionnez "Database" puis "PostgreSQL"
3. Une fois créée, Railway configurera automatiquement la variable `DATABASE_URL`

4. Configuration des variables d'environnement spéciales

Après avoir créé votre projet sur Railway, assurez-vous d'ajouter ces variables :

1. `PORT=8000` (Le port sur lequel l'application écoutera)
2. `NIXPACKS_BUILD_CMD=pip install -r backend/requirements.txt && cd frontend && npm install && npm run build`
3. `NIXPACKS_START_CMD=cd backend && python manage.py migrate && python manage.py collectstatic --noinput && gunicorn jaelleshop.wsgi:application --bind 0.0.0.0:$PORT`

5. Déploiement automatique

Une fois configuré, Railway déploiera automatiquement l'application à chaque push sur la branche principale.

Développement local avec Docker

```
# Démarrer l'application en mode développement
docker-compose up

# Reconstruire l'application après des modifications
docker-compose up --build

# Exécuter des commandes dans le conteneur
docker-compose exec web python backend/manage.py createsuperuser
```

Maintenance

Migration de la base de données

```
docker-compose exec web python backend/manage.py makemigrations
docker-compose exec web python backend/manage.py migrate
```

Création d'un utilisateur admin

```
docker-compose exec web python backend/manage.py createsuperuser
```

Sauvegarde et restauration de la base de données

```
# Sauvegarde
docker-compose exec db pg_dump -U postgres jaelleshop > backup.sql

# Restauration
cat backup.sql | docker-compose exec -T db psql -U postgres jaelleshop
```