EVIMERIA (anciennement JaelleShop)

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

Prérequis

- Docker et Docker Compose pour le développement local
- Un compte Railway pour le déploiement
- Un compte Cloudinary pour la gestion des médias
- Un compte DockerHub pour stocker les images

Structure du Projet

```
evimeria/
├── backend/            # API Django
│   ├── Dockerfile      # Dockerfile du backend
│   ├── jaelleshop/     # Configurations du projet
│   ├── products/       # App pour les produits
│   ├── users/          # App pour les utilisateurs
│   └── ...
├── frontend/           # Application React
│   ├── Dockerfile      # Dockerfile du frontend
│   ├── nginx.conf      # Configuration Nginx
│   └── ...
├── docker-compose.yml  # Configuration Docker Compose
├── .github/workflows/  # Configuration CI/CD GitHub Actions
└── ...
```

Déploiement sur Railway

1. Préparation du code

Assurez-vous que votre dépôt contient tous les fichiers suivants:

- `backend/Dockerfile`
- `frontend/Dockerfile`
- `frontend/nginx.conf`
- `docker-compose.yml`
- `.github/workflows/railway-deploy.yml`

2. Configuration des secrets GitHub

Allez dans les paramètres de votre dépôt GitHub et ajoutez les secrets suivants:

```
DOCKERHUB_USERNAME=votre_nom_utilisateur_dockerhub
DOCKERHUB_TOKEN=votre_token_dockerhub
RAILWAY_TOKEN=votre_token_railway
```

3. Configuration des variables d'environnement

Dans Railway, configurez les variables d'environnement suivantes:

```
# Base de données PostgreSQL
DATABASE_URL=postgresql://${PGUSER}:${POSTGRES_PASSWORD}@${PGHOST}:${PGPORT}/${
PGDATABASE}
POSTGRES_PASSWORD=votre_mot_de_passe
POSTGRES_USER=postgres
POSTGRES_DB=railway

# Configuration Django
DEBUG=False
SECRET_KEY=votre_secret_key
ALLOWED_HOSTS=*.up.railway.app,localhost,127.0.0.1

# Cloudinary
CLOUDINARY_CLOUD_NAME=dmcaguchx
CLOUDINARY_API_KEY=votre_api_key
CLOUDINARY_API_SECRET=votre_api_secret

# Configuration de build
NODE_OPTIONS=--max_old_space_size=465
PIP_NO_CACHE_DIR=true
PYTHONUNBUFFERED=1
NODE_ENV=production
NPM_CONFIG_PRODUCTION=false

# Port
PORT=8000
```

4. Déploiement automatique avec GitHub Actions

1. Poussez votre code sur la branche main de votre dépôt GitHub
2. GitHub Actions va automatiquement:
    - Construire vos images Docker pour le backend et le frontend
    - Pousser ces images sur DockerHub
    - Déployer les images sur Railway
3. Vous pouvez également déclencher manuellement le déploiement depuis l'onglet "Actions" de votre dépôt GitHub.

5. Déploiement manuel sur Railway

Si vous préférez déployer manuellement:

1. Connectez-vous à Railway et créez un nouveau projet
2. Choisissez "Deploy from GitHub repo"
3. Sélectionnez votre dépôt GitHub
4. Railway détectera automatiquement les fichiers Docker et déploiera votre application

Développement local avec Docker

```
# Créer un fichier .env à partir de .env.example
cp .env.example .env
# Modifier le fichier .env avec vos propres valeurs

# 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 backend
docker-compose exec backend python manage.py createsuperuser
```

Maintenance

Migration de la base de données

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

Création d'un utilisateur admin

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

Sauvegarde et restauration de la base de données

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

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

Configuration des variables d'environnement pour l'application

Pour que l'application fonctionne correctement, vous devez configurer les variables d'environnement suivantes dans Railway:

1. **Pour le backend (Django)**:
    - DATABASE_URL: L'URL de connexion à votre base de données PostgreSQL
    - SECRET_KEY: Clé secrète Django pour la sécurité
    - DEBUG: Toujours "False" en production
    - CLOUDINARY_CLOUD_NAME, CLOUDINARY_API_KEY, CLOUDINARY_API_SECRET: Identifiants Cloudinary
2. **Pour le frontend (React)**:
    - API_URL: L'URL de l'API backend (http://backend:8000 en local, votre URL Railway en production)

Vous pouvez copier les variables de la base de données depuis Railway, comme indiqué dans le fichier variablesenv.txt.