E-commerce Monitoring System

Un système de paiement en temps réel combiné avec un module de monitoring d'infrastructure et de gestion des logs. Ce projet est conçu pour être modulaire, scalable et proche d'un environnement de production.

Technologies Utilisées

- **Backend**: FastAPI (Python)
- **Frontend**: Angular
- **Base de données**: MongoDB
- **Cache**: Redis
- **Monitoring**: Grafana & Prometheus
- **Déploiement**: Docker & Docker Compose
- **Paiement**: Stripe/PayPal (intégration optionnelle)

Fonctionnalités

Module de Paiement en Temps Réel

- Intégration avec Stripe/PayPal pour traiter les paiements.
- Stockage des paiements et utilisateurs dans MongoDB.
- Mise en cache des transactions récentes avec Redis.
- Gestion des statuts de paiement (succès, échec, en attente).
- WebSockets pour mise à jour en temps réel.
- Interface Angular avec historique des paiements et graphiques.

Module de Gestion des Logs et Monitoring

- Collecte des logs des paiements et erreurs API.
- Stockage des logs dans MongoDB.
- Cache des logs critiques récents avec Redis.
- WebSockets pour affichage en temps réel des logs.
- Tableau de bord Angular avec filtres et graphiques (ngx-charts).
- Intégration avec Grafana & Prometheus pour visualisation avancée.

Structure du Projet

```
ecommerce_monitoring/
├── backend/           # API FastAPI (paiements & logs)
│   ├── app/           # Code source de l'application
│   ├── Dockerfile     # Configuration Docker pour le backend
│   └── requirements.txt # Dépendances Python
├── frontend/          # Dashboard Angular (gestion des paiements & logs)
│   ├── src/           # Code source Angular
│   └── Dockerfile     # Configuration Docker pour le frontend
│   └── docker-compose.yml # Configuration des services Docker
```

```
└── Indication.md      # Indication sur le projet
└── README.md          # Documentation du projet
```

Installation

Prérequis

- Docker et Docker Compose installés sur votre machine.
- Python 3.10 (pour le développement local du backend).
- Node.js (pour le développement local du frontend).

Étapes d'Installation

1. **Cloner le dépôt** :

```
git clone https://github.com/votre-utilisateur/ecommerce-
monitoring.git
cd ecommerce-monitoring

 Configurer les variables d'environnement :

     Créez un fichier .env dans le dossier backend/ avec les variables
suivantes :
     env
     Copy

     MONGO_URI=mongodb://mongodb:27017
     REDIS_HOST=redis
     REDIS_PORT=6379

 Démarrer les services avec Docker Compose :
 bash
 Copy

 docker-compose up --build

 Accéder aux services :

     Backend API : http://localhost:8000

     Frontend Dashboard : http://localhost:4200

     MongoDB : mongodb://localhost:27017

     Redis : redis://localhost:6379
```

Utilisation

API Endpoints

Paiements

```
POST /payments : Créer un nouveau paiement.

GET /payments/{id} : Récupérer les détails d'un paiement.

GET /payments/recent : Obtenir les derniers paiements (cache Redis).
```

Logs

```
POST /logs : Enregistrer un log.

GET /logs/recent : Récupérer les logs récents.

GET /logs/stats : Obtenir les statistiques des logs.
```

Interface Utilisateur

```
Dashboard Angular : Accédez à l'interface utilisateur pour visualiser les
paiements et les logs en temps réel.
```

Contribution

Les contributions sont les bienvenues ! Voici comment vous pouvez contribuer :

```
Forker le projet.

Créer une branche pour votre fonctionnalité (git checkout -b
feature/AmazingFeature).

Commiter vos changements (git commit -m 'Add some AmazingFeature').

Pusher la branche (git push origin feature/AmazingFeature).

Ouvrir une Pull Request.
```

Prêt à révolutionner le e-commerce avec un système de paiement performant et monitoré ! 🚀 Copy

Explication des sections :

1. **Titre et Badge** : Le titre du projet et un badge de licence pour indiquer que le projet est open-source.
2. **Technologies Utilisées** : Une liste des technologies principales utilisées dans le projet.
3. **Fonctionnalités** : Une description des fonctionnalités principales des modules de paiement et de logs.

4. **Structure du Projet** : Une vue d'ensemble de la structure des fichiers et dossiers.
5. **Installation** : Des instructions détaillées pour installer et configurer le projet.
6. **Utilisation** : Des informations sur les endpoints de l'API et l'interface utilisateur.
7. **Contribution** : Des instructions pour contribuer au projet.

Ce fichier `README.md` est conçu pour être clair et complet, permettant aux nouveaux utilisateurs de comprendre rapidement le projet et de le mettre en place.