Cahier des Charges : Système de Paiement et Monitoring des Logs

1. Introduction

1.1 Contexte

Ce projet vise à développer un système de paiement en temps réel combiné avec un module de monitoring d'infrastructure et de gestion des logs. Il repose sur des technologies modernes telles que **FastAPI**, **Angular**, **MongoDB** et **Redis**, avec une approche orientée DevOps pour assurer la scalabilité et la sécurité.

1.2 Objectifs

- Implémenter un système de paiement en ligne sécurisé et performant.
- Gérer les transactions et leur historique en **temps réel**.
- Collecter, stocker et analyser les logs des paiements et des erreurs serveur.
- Fournir un tableau de bord interactif pour la visualisation des logs et des paiements.
- Assurer un monitoring avancé avec Grafana & Prometheus.
- Intégrer une architecture **modulaire, scalable et DevOps-friendly**.

2. Architecture du Projet

2.1 Technologies Utilisées

Technologie	Rôle
FastAPI	Backend pour la gestion des paiements et des logs
Angular	Frontend pour l'interface utilisateur et le monitoring
MongoDB	Stockage des transactions et des logs
Redis	Cache des transactions et logs récents
WebSockets	Communication en temps réel
Grafana & Prometheus	Monitoring et visualisation des métriques
Docker & CI/CD	Déploiement et automatisation
Stripe/PayPal	Intégration du système de paiement

2.2 Structure du Projet

```
ecommerce-monitoring/
|── backend/        # API FastAPI (paiements & logs)
|── frontend/       # Dashboard Angular (gestion des paiements & logs)
|── infrastructure/ # Docker, configuration Redis, MongoDB, monitoring
```

3. Détails des Modules

3.1 Module Paiement en Temps Réel

Fonctionnalités :

✅ 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.

Routes API :

Route	Méthode	Description
`/payments`	**POST**	Créer un paiement
`/payments/{id}`	**GET**	Récupérer les détails d'un paiement
`/payments/recent`	**GET**	Obtenir les derniers paiements (cache Redis)

3.2 Module de Gestion des Logs et Monitoring

Fonctionnalités :

✅ 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.

Routes API :

Route	Méthode	Description
`/logs`	**POST**	Enregistrer un log
`/logs/recent`	**GET**	Récupérer les logs récents
`/logs/stats`	**GET**	Obtenir les statistiques des logs

4. Planification du MVP (Lundi → Samedi)

📅 Lundi – Initialisation & Setup

✅ Créer l'architecture du projet.
✅ Configurer **MongoDB et Redis**.
✅ Dockeriser l'application.
✅ Déployer un **Hello World API FastAPI** et **Hello World Angular**.

📅 Mardi – Implémentation des Paiements

✅ Modéliser et implémenter **Payment** dans MongoDB.
✅ Ajouter une API **POST /payments**.
✅ Intégration **Stripe/PayPal**.
✅ Stockage des paiements en cache Redis.
✅ Interface Angular pour la gestion des paiements.

📅 Mercredi – Implémentation du Système de Logs

✅ Ajouter le modèle **Log** dans MongoDB.
✅ Créer API **POST /logs**.
✅ Stockage des logs critiques en cache Redis.
✅ WebSocket pour affichage des logs en live.
✅ Interface Angular pour la visualisation des logs.

📅 Jeudi – WebSockets & Monitoring

✅ Affichage des paiements en **temps réel** (WebSockets).
✅ Affichage des logs en temps réel.
✅ Graphes interactifs avec **ngx-charts**.
✅ Test du flux complet **Paiement → Log → Dashboard Angular**.

📅 Vendredi – Optimisation & Sécurisation

✅ Intégration de **JWT/OAuth2**.
✅ Monitoring avec **Grafana & Prometheus**.
✅ Alertes en cas d'échec de paiement.
✅ Optimisation Redis (TTL et Index MongoDB).

📅 Samedi – Tests & Déploiement

✅ Tests unitaires avec `pytest` et `Jest`.
✅ Mise en place de CI/CD avec **GitHub Actions**.
✅ Déploiement sur **AWS/DigitalOcean/VPS**.
✅ 🎉 **Projet terminé et prêt pour les entretiens !**

---

5. Conclusion

Ce projet combine **une solution de paiement en temps réel** avec **un système de monitoring avancé**, mettant en avant des compétences **Full Stack et DevOps**. Il servira de **référence professionnelle** pour démontrer une maîtrise des systèmes distribués, de la scalabilité et de l'observabilité d'une infrastructure cloud.

🚀 **Prêt à coder et à révolutionner le e-commerce avec un système de paiement performant et monitoré ?** 😎