# Documentation de l'Architecture Monolithique Distribuée de la plateforme de management en ligne des chantiers pétroliers

## 1. Introduction

L'architecture monolithique distribuée combine les avantages d'une architecture monolithique traditionnelle et des principes de scalabilité d'une architecture microservices. Elle permet de maintenir une base de code unique tout en offrant une flexibilité pour la distribution et la gestion des composants.

### 1.1 Avantages de cette architecture

- **Simplicité de développement et de déploiement** : Contrairement aux microservices, elle évite la complexité de la communication interservices excessive.
- **Meilleure scalabilité** : Les modules peuvent être scalés indépendamment si nécessaire.
- **Sécurité renforcée** : Moins de points d'entrée pour les attaques comparé à une architecture microservices complète.
- **Facilité de test** : Tous les composants peuvent être testés de manière unifiée.

---

## 2. Modules et Technologies Utilisées

### 2.1 Frontend : React.js

- **Framework utilisé** : React.js
- **Objectif** : Gérer l'interface utilisateur (UI), l'authentification, et l'expérience utilisateur.
- **Optimisations** : Utilisation de Redux pour la gestion d'état, React Query pour la gestion des requêtes, et TailwindCSS pour un rendu rapide et optimisé.

### 2.2 Backend API : Node.js + NestJS

- **Frameworks utilisés** : Node.js avec NestJS
- **Objectif** : Servir de passerelle principale pour les requêtes front-end et gérer la logique métier.
- **Optimisations** : Utilisation de TypeScript, validation avec Zod, et gestion des erreurs centralisée.

### 2.3 Base de données : MongoDB & PostgreSQL

- **MongoDB** : Stockage des documents, logs et événements en temps réel.
- **PostgreSQL** : Gestion des données structurées (utilisateurs, projets, transactions financières).
- **Redis** : Cache pour améliorer les performances.

## 2.4 Services indépendants

### 1 Service Utilisateur

- Inscription et connexion sécurisée des utilisateurs
- Gestion des rôles et permissions (administrateur, employé, superviseur...)
- Mise à jour des informations personnelles et changement de mot de passe
- Suivi des connexions et historique d'activité des utilisateurs

### 2 Service Projet

- Création, modification et suppression de projets
- Gestion des tâches avec assignation et suivi de l'avancement
- Collaboration en temps réel entre les équipes (commentaires, mise à jour instantanée)
- Notifications et rappels des échéances de projet
- Historique des modifications et traçabilité des actions

### 3 Service Budget

- Suivi des dépenses et gestion des budgets alloués aux projets
- Alertes en cas de dépassement de budget
- Génération de rapports financiers détaillés
- Validation et approbation des demandes de budget par les responsables
- Suivi des paiements et facturation

### 4 Service de Stockage

- Téléversement et gestion des documents et fichiers liés aux projets
- Organisation des fichiers par catégorie et projet
- Historique des versions et récupération des fichiers supprimés
- Accès contrôlé et sécurisé aux documents partagés

### 5 Service de Messagerie

- Envoi de notifications instantanées aux utilisateurs
- Suivi des messages et accusés de réception
- Alertes en temps réel sur les tâches, budgets et documents mis à jour
- Canaux de discussion internes pour faciliter la communication entre équipes

## 2.5 Infrastructure et Conteneurisation

- **Docker & Kubernetes** : Conteneurisation et orchestration pour une meilleure scalabilité.
- **CI/CD avec Jenkins** : Automatisation du déploiement et des tests (Jest, Cypress).

- **AWS (EC2, S3, RDS)** : Hébergement et stockage sécurisé.

---

## 3. Construction d'une Application Web Performante et Robuste

### 3.1 Sécurité

- **Authentification et Autorisation** : JWT + OAuth2 pour les accès sécurisés.
- **Protection contre les attaques** : Sécurisation des API avec Helmet, rate-limiting et validation des entrées.

### 3.2 Performance

- **Optimisation des requêtes** : Indexation dans PostgreSQL, partitionnement des données.
- **Mise en cache** : Utilisation de Redis pour réduire les appels aux bases de données.

### 3.3 Évolutivité et Maintenance

- **Orchestration des conteneurs** : Kubernetes pour gérer la montée en charge.
- **Logging et monitoring** : ELK stack (Elasticsearch, Logstash, Kibana) pour surveiller l'activité.
- **Tests automatisés** : Unitaires avec Jest, intégration avec Cypress, et pipeline CI/CD.

---

Cette architecture permet de construire une application robuste et scalable, en exploitant les meilleures technologies modernes.