1. Système de Gestion des Stocks Distribué pour une Chaîne de Magasins

Problématique : Une entreprise possédant plusieurs magasins physiques a du mal à synchroniser les stocks entre ses différents entrepôts et points de vente.

Solution avec une architecture distribuée

- **RMI** : Pour permettre aux magasins de consulter et mettre à jour les stocks en temps réel.
- **Microservices** : Gestion des articles, des commandes et des notifications en cas de rupture de stock.

Étapes du projet

1. **Analyse des besoins** : Identification des entités (produits, magasins, commandes, etc.).
2. **Conception de l'architecture** :
    - Microservices pour la gestion des stocks, des commandes et des utilisateurs.
    - RMI pour la synchronisation en temps réel entre les magasins.
3. **Développement des services** :
    - Service de gestion des produits.
    - Service de gestion des commandes.
    - Service de notification (alerte en cas de stock faible).
4. **Déploiement sur des serveurs distribués** : Cloud ou plusieurs machines locales.
5. **Tests de charge et de scalabilité**.
6. **Sécurisation des échanges de données entre les services**.

2. Plateforme de Réservation de Rendez-vous pour une Entreprise Multisite

Problématique : Une entreprise de services (ex. clinique, salon de coiffure) souhaite offrir à ses clients un système de prise de rendez-vous en ligne, tout en permettant aux employés de gérer leur emploi du temps.

Solution avec une architecture distribuée

- **Microservices** : Gestion des utilisateurs, des rendez-vous et des notifications.
- **RMI** : Synchronisation des calendriers entre les différents sites en temps réel.

Étapes du projet

1. **Définition des besoins** : Qui peut prendre rendez-vous ? Qui gère les disponibilités ?
2. **Conception** :
    - Base de données distribuée pour stocker les rendez-vous.
    - Services de gestion des clients et des disponibilités.
3. **Implémentation des microservices** :
    - Service d'authentification des clients et employés.
    - Service de gestion des créneaux et réservations.
    - Service de notifications (email/SMS).
4. **Mise en place d'un système de cache pour améliorer les performances**.
5. **Déploiement et test du système**.

6. **Sécurisation des données et authentification sécurisée (JWT, OAuth2, etc.)**.

3. Système de Gestion de Paie Distribué pour une Multinationale

Problématique : Une entreprise internationale a des employés dans plusieurs pays et doit gérer la paie en fonction des lois locales.

Solution avec une architecture distribuée

- **Microservices** : Calcul des salaires, gestion des impôts et paiements bancaires.
- **RMI** : Communication entre les branches de différents pays.

Étapes du projet

1. **Analyse des besoins légaux par pays**.
2. **Conception de l'architecture distribuée** :
    - Microservices indépendants pour chaque pays.
    - Un service central qui agrège les données de paie.
3. **Développement des services** :
    - Service de calcul des salaires.
    - Service de gestion des impôts et cotisations.
    - Service d'intégration avec les banques.
4. **Sécurisation des transactions** (chiffrement des données, conformité RGPD).
5. **Déploiement dans le cloud et tests de charge**.
6. **Formation des équipes et documentation**.

4. Système de Suivi de Flotte pour une Entreprise Logistique

Problématique : Une entreprise de transport veut suivre en temps réel la position de ses véhicules et optimiser les itinéraires.

Solution avec une architecture distribuée

- **Microservices** : Gestion des véhicules, des trajets et des incidents.
- **RMI** : Communication entre les véhicules et les centres de gestion.

Étapes du projet

1. **Définition des besoins métier** (suivi GPS, gestion des itinéraires, alertes).
2. **Conception** :
    - Base de données distribuée pour stocker les trajets.
    - Services de gestion des véhicules et des chauffeurs.
3. **Développement des services** :
    - Service de suivi GPS (via API tierces comme Google Maps).
    - Service d'optimisation des trajets (algorithmes de routage).
    - Service de gestion des incidents.
4. **Implémentation d'un tableau de bord en temps réel**.
5. **Tests de montée en charge** (simulation avec des milliers de véhicules).
6. **Déploiement sur des serveurs répartis et monitoring**.

5. Système de Gestion des Documents en Entreprise

Problématique : Une entreprise souhaite centraliser la gestion et le partage sécurisé de ses documents entre plusieurs équipes.

Solution avec une architecture distribuée

- **Microservices** : Stockage, gestion des droits et versioning des fichiers.
- **RMI** : Accès rapide aux documents depuis plusieurs bureaux.

Étapes du projet

1. **Analyse des besoins** (qui peut voir/modifier quels documents ?).
2. **Conception de l'architecture** :
    - Stockage réparti sur plusieurs serveurs.
    - Service d'authentification et gestion des accès.
3. **Développement des services** :
    - Service de gestion des utilisateurs et permissions.
    - Service de stockage et récupération des fichiers.
    - Service de versioning des documents.
4. **Mise en place d'un moteur de recherche pour les documents**.
5. **Sécurisation (cryptage des fichiers, journalisation des accès)**.
6. **Déploiement et test d'utilisation**.

6. Système de Gestion d'Incidents IT pour une Entreprise

Problématique : Une entreprise souhaite un système pour suivre les tickets d'incidents IT et faciliter la communication entre les équipes techniques.

Solution avec une architecture distribuée

- **Microservices** : Gestion des tickets, des utilisateurs et des notifications.
- **RMI** : Partage en temps réel des incidents entre plusieurs équipes.

Étapes du projet

1. **Définition des processus de gestion des incidents**.
2. **Conception** :
    - Microservices pour les tickets, les notifications et le reporting.
    - Base de données distribuée pour les logs.
3. **Développement des services** :
    - Service de création et suivi des tickets.
    - Service d'affectation automatique aux équipes IT.
    - Service d'alertes en cas d'incident critique.
4. **Implémentation d'un tableau de bord** avec graphiques et KPIs.
5. **Tests et simulations d'incidents**.
6. **Déploiement avec scalabilité (serveurs cloud, Kubernetes, etc.)**.