Intitulé du projet : Gestion unifiée des commandes multi-canaux

Contexte

Une entreprise souhaite centraliser la gestion de ses commandes provenant de plusieurs sources :

- Site web e-commerce,
- Application mobile,
- Partenaires B2B.

Pour répondre à cette diversité, elle veut exposer ses fonctionnalités sous plusieurs formes d'API afin de permettre à différents types de clients ou systèmes d'y accéder. L'idée est d'exposer ses services via différentes interfaces (REST, SOAP, GraphQL, gRPC) pour s'adapter aux besoins variés de ses clients internes et externes.

Variantes possibles :

- Variante 1 : Gestion de stock et fournisseurs
- Variante 2 : Gestion de réservations (vols, hôtels, événements)
- Variante 3 : Catalogue de produits distribué

Objectifs pédagogiques

1. **Conception modulaire** et usage de l'**Inversion de Contrôle (IoC)** via Spring et les interfaces ;
2. **Exposition de services métier** via plusieurs technologies d'intégration :
    - **Spring Data REST** pour CRUD automatique ;
    - **Service SOAP** pour compatibilité avec des systèmes legacy ;
    - **API GraphQL** pour des requêtes flexibles ;
    - **Service gRPC** pour communication performante entre microservices ;
3. **Architecture en couches** : présentation, service métier, accès aux données ;

Spécifications fonctionnelles

- Gérer des clients, produits, et commandes.
- Chaque commande porte sur des produits différents avec des quantités précises.
- Un service doit permettre :
    - de créer une commande ;
    - de lister les commandes d'un client ;
    - de calculer le montant total ;
    - de mettre à jour l'état d'une commande.

Architecture logicielle proposée

- **Spring Boot** comme socle principal.
- **Spring Data JPA** + **Spring Data REST** pour la persistance et l'exposition CRUD.
- **Service SOAP** : implémentation d'une interface `OrderService` exposée via `@Endpoint`.

- **Service GraphQL** : un contrôleur GraphQL pour des requêtes personnalisées (ex. total d'un client).
- **Service gRPC** : microservice "NotificationService" qui reçoit un message lors de la création d'une commande.
- **Spring IoC / Dependency Injection** pour la gestion des composants (repositories, services métier, contrôleurs).

Organisation du projet

- `model/` → entités JPA (Client, Product, Order, OrderItem)
- `repository/` → interfaces Spring Data
- `service/` → logique métier (calculs, validation)
- `web/rest/` → endpoints REST
- `web/soap/` → service SOAP
- `web/graphql/` → endpoint GraphQL
- `grpc/` → service et client gRPC

Livrables attendus

1. Code source complet (projet Maven/Gradle).
2. Documentation d'architecture (diagrammes de composants et de séquence).
3. Capture d'écran / tests Postman ou scripts de test.
4. (Optionnel) README expliquant comment lancer et tester chaque type d'API.