Documentation Technique - Microservice Order-Service

1. Présentation

Le microservice Order-Service est responsable de la gestion des commandes clients. Il permet de :
- Créer une commande
- Modifier une commande
- Supprimer une commande
- Récupérer une ou toutes les commandes enregistrées dans la base de données

Ce microservice est développé en Java avec Spring Boot 3, JPA, Hibernate Validator, PostgreSQL et expose une API REST consommée depuis Postman ou un frontend Angular.

2. Technologies utilisées

- Java 17
- Spring Boot 3.4.5
- Spring Data JPA
- Hibernate Validator
- PostgreSQL 17
- Maven
- Postman (tests)

3. Structure des entités

Les entités principales sont :
- Order : représente une commande client (clientName, orderDate, totalPrice, status, items)
- OrderItem : représente un produit dans une commande (productId, productName, quantity, unitPrice)

4. Endpoints disponibles

Base URL : http://localhost:8083/api/orders

- GET /api/orders
  → Liste toutes les commandes

- GET /api/orders/{id}
  → Récupère une commande par son ID

- POST /api/orders

→ Crée une commande
→ Exemple JSON :

```
{
  "clientName": "Kevin",
  "orderDate": "2025-05-19",
  "status": "EN_COURS",
  "items": [
    {
      "productId": 1,
      "productName": "Ordinateur portable",
      "quantity": 2,
      "unitPrice": 999.99
    }
  ]
}
```

- PUT /api/orders/{id}
  → Met à jour une commande existante

- DELETE /api/orders/{id}
  → Supprime une commande par ID

5. Tests Postman

Tous les endpoints ont été testés avec Postman.
- Pensez à utiliser l'en-tête suivant pour les requêtes POST/PUT :
  Content-Type: application/json
- Les tests ont été validés avec des statuts HTTP 200, 201, 204 et 400 en cas d'erreur de validation.

6. Gestion des erreurs

Le microservice retourne les erreurs au format JSON avec :
- message : description de l'erreur
- status : code HTTP
- timestamp : horodatage de l'erreur

Exemple :

```
{
  "message": "Le nom du client est obligatoire",
  "status": 400,
```

  "timestamp": "2025-05-20T00:00:00.000Z"
}

7. Mapper & DTO

Afin de ne pas exposer directement les entités JPA, des DTO sont utilisés :
- OrderDTO
- OrderItemDTO

Le mapper (OrderMapper) assure la conversion entre Entity et DTO. Cela évite les erreurs de lazy loading lors des requêtes GET.