Mini Shop API
Technical Design Specification

Kelvin

January 13, 2026

Contents

1 Overview

Mini Shop là mt h thng backend n gin mô phng h thng bán hàng thc t, gm ba domain chính:

- User
- Product
- Order

Lung nghip v:

User → Xem Product → To Order → Thêm Product → Thanh toán → Hoàn tt

Mc tiêu ca h thng là giúp Junior Backend Developer luyn tp:

- REST API
- JPA Relationship
- Business Logic
- Transaction và Validation

2 Domain Model

2.1 User

Field	Type	Constraints
id	Long	Primary Key
name	String	Not null, 3–50 characters
email	String	Unique, not null
password	String	Not null, min 6
createdAt	LocalDateTime	Auto generated

2.2 Product

Field	Type	Constraints
id	Long	Primary Key
name	String	Not null
price	BigDecimal	> 0
stock	Integer	≥ 0
createdAt	LocalDateTime	Auto generated

2.3 Order

Field	Type	Constraints
id	Long	Primary Key
user	User	Many-to-One
status	Enum	CREATED, PAID, CANCELLED
totalPrice	BigDecimal	≥ 0
createdAt	LocalDateTime	Auto generated

2.4 OrderItem

Field	Type	Constraints
id	Long	Primary Key
order	Order	Many-to-One
product	Product	Many-to-One
quantity	Integer	> 0
price	BigDecimal	Price snapshot at purchase time

3 API Specification

Base URL: `/api`

3.1 User APIs

3.1.1 Create User

POST `/api/users`

```
{
  "name": "Kelvin",
  "email": "kelvin@gmail.com",
  "password": "123456"
}
```

Rules:

- Email must be unique
- Password length ≥ 6

3.1.2 Get User

GET `/api/users/{id}`

3.1.3 List Users

GET `/api/users`

3.2 Product APIs

3.2.1 Create Product

POST `/api/products`

```
{
  "name": "Iphone 15",
  "price": 25000000,
  "stock": 10
}
```

Rules:

- price > 0
- stock ≥ 0

3.2.2 List Products

GET `/api/products`

3.2.3 Get Product

GET `/api/products/{id}`

3.2.4 Update Product

PUT `/api/products/{id}`

3.2.5 Delete Product

DELETE `/api/products/{id}`
Product cannot be deleted if it appears in any PAID order.

3.3 Order APIs

3.3.1 Create Order

POST `/api/orders`

```
{
  "userId": 1
}
```

Result:

- status = CREATED
- totalPrice = 0

3.3.2 Add Product to Order

POST `/api/orders/{orderId}/items`

```
{
  "productId": 3,
  "quantity": 2
}
```

Rules:

- Order must be CREATED
- Product stock $\geq$ quantity
- price = product price at time of adding
- totalPrice must be recalculated

3.3.3 Get Order Detail

GET `/api/orders/{orderId}`

```
{
  "id": 10,
  "status": "CREATED",
  "totalPrice": 50000000,
  "items": [
    {
      "productName": "Iphone 15",
      "price": 25000000,
      "quantity": 2
    }
  ]
}
```

3.3.4 Get Orders of User

GET `/api/users/{userId}/orders`

3.3.5 Pay Order

POST `/api/orders/{orderId}/pay`

Rules:

- Order must be CREATED
- Reduce product stock
- Status becomes PAID

3.3.6 Cancel Order

POST `/api/orders/{orderId}/cancel`
Only CREATED orders can be cancelled.

4 Business Rules

- Product stock cannot be negative
- Cannot add items to PAID or CANCELLED orders
- totalPrice must equal sum of OrderItems
- PAID orders cannot be modified
- Products used in PAID orders cannot be deleted

5 Architecture Rules

```
Controller
   ↓
Service (Business Logic)
   ↓
Repository (JPA)
   ↓
Database
```

Rules:

- Controllers must not contain business logic
- All validation and processing must be in Service layer