Progetto Spring Ecommerce

2 entity non collegate

Architettura completa

DAO mappa

Livello 2 del REST

Descrizione

- Si vuole realizzare un Web Service per la gestione del ciclo di vita degli ordini di un sito di ecommerce.
- Gli endpoint forniti devono **attenersi al livello 2 del REST**

- Le entità sono prodotti, prodotti ordinati e ordini.

- `Product` è definito da id (univoco), name, price e stock.
- `Order` è definito da id (univoco), orderItemList (lista dei prodotti ordinati), totalAmount (totale dell'ordine), status (CREATED, CONFIRMED, SHIPPED, DELIVERED, CANCELLED) e createdAt (data creazione)
- `OrderItem` è definito da productId (univoco), quantity e unitPrice.

Regole di business

- **Creazione ordine:**
  - Verifica che i prodotti esistano
  - Verifica disponibilità stock
  - Calcola totale
  - NON scala stock finché non viene confermato

- **Conferma ordine:**
  - Solo se status = CREATED
  - Scala lo stock
  - Se stock insufficiente → 409 Conflict
  - Cambia stato in CONFIRMED

Regole di business

- **Spedizione ordine:**
  - Solo se status = CONFIRMED
  - Cambia stato in SHIPPED

- **Consegna ordine:**
  - Solo se status = SHIPPED

- **Cancellazione ordine:**
  - Se CREATED → ok
  - Se CONFIRMED → ripristina stock
  - Se SHIPPED → 409 Conflict

Funzionalità sugli ordini

- Crea ordine
  - Status 201 created
- Cerca ordine per id
  - Status 200 ok / 404 Not Found
- Cerca tutti gli ordini CREATED
  - Status 200 ok
- Cerca tutti gli ordini CONFIRMED
  - Status 200 ok
- Cerca tutti gli ordini SHIPPED
  - Status 200 ok

Funzionalità sulle transizioni di stato degli ordini

Fissato un ordine per id:

- Passa un ordine in stato CONFIRMED
- Passa un ordine in stato SHIPPED
- Passa un ordine in stato DELIVERED
- Passa un ordine in stato CANCELED

Data Transfer Object (DTO)

`OrderCreateRequestDTO`:
- List<ItemRequest>
- productId
- quantity

`OrderResponseDTO`:
- id
- items (nome prodotto + quantità + subtotal)
- totalAmount
- status
- createdAt

`OrderStatusUpdateDTO`:
- status

Architettura

- Simulare la persistenza dei dati con l'uso di una mappa per ciascuna categoria: una per prodotti ed una per ordini
- Strutturare il progetto con i seguenti package e classi:
  - controller
  - service
  - dao
  - entity
  - dto
  - exception
- Usare la dependency injection per gestire le dipendenze fra strati

Gestione degli errori

Realizzare una gestione degli errori con metodi nello strato controller che gestiscano le seguenti anomalie e tornino i rispettivi HTTP status:

- `OrderNotFoundException` `404`
- `InvalidOrderStateException` `409`
- `ProductNotFoundException` `404`
- `InsufficientStockException` `409`

Uno di quelli previsti

Formato errore JSON:

```
{
  "timestamp": "...",
  "status": …,
  "error": "Conflict",
  "message": "...",
}
```