IES THIAR
C/ Pilar Pérez Aguirre,13
03190 PILAR DE LA HORADADA

PRÁCTICA 4

Desarrollo de una aplicación web de Pizzería con Spring

Curso 2024 - 2025

UNIDAD 4. DESARROLLO DE COMPONENTES CON SPRING

ACCESO A DATOS – 2º DAM

I.E.S THIAR

Implementa en Java los siguientes ejercicios, utilizando como base la práctica anterior. Al finalizar, adjunta el proyecto comprimido en un zip en la tarea con esta nomenclatura: Apellido1-Nombre-Practica4.zip. Para la realización de esta práctica hay que seguir las buenas prácticas de programación:

- **Seguir las "naming conventions" de Java.**
- **Gestionar de forma eficiente los recursos externos.**
- **Control de posibles excepciones.**
- **Encapsulamiento.**
- **Principio de Responsabilidad Única.**
- **Baja complejidad ciclomática. Mantener simplicidad de las funciones.**
- **Se debe testear el servicio REST utilizando Postman y se debe adjuntar en la tarea los JSON de estas pruebas mediante la exportación.**

El esquema de la BD ya estará creado previamente con el nombre "**pizzeria-spring**", ofreciéndose el servicio en el puerto 3306 con el motor de MySQL. El esquema estará vacío, por lo que la responsabilidad de crear las tablas, insertar registros en ellas, leerlos, actualizarlos y borrarlos recaerá totalmente en la tecnología JPA.

Requisitos de la aplicación:

- La lógica de negocio y persistencia será la misma que en la Práctica 3.
- Se debe utilizar el **patrón MVC de Spring** para la arquitectura.
- Se debe implementar el **CRUD** usando **Spring JPA**.
- Se debe utilizar la **inyección de dependencias** de Spring.
- Se debe utilizar **Lombok** para la automatización del modelo.
- Se debe implementar servicios Restful usando **Spring Web**.
- El servicio REST debe devolver el paquete **HTTP Response** correspondiente según el contexto y siguiendo las **buenas prácticas RESTful**, incluyendo los casos de error posibles.

(2 puntos). Servicio REST de clientes. El endpoint será: /api/clientes y debe implementarse un servicio REST que contenga las operaciones básicas CRUD, además de la siguiente:

- Obtener todos. Debe devolver todos los clientes del sistema. Además, puede recibir un parámetro opcional "nombre", que permita filtrar los clientes que contengan el nombre, ignorando mayúscula y minúscula (usar findByXContainingIgnoreCase). Por ejemplo, "Juan" podría devolver 2 clientes: Juan Martínez Sánchez y Juan Sorolla Díaz.

- Guardar cliente. Debe comprobarse que no exista un cliente con el mismo email, en cuyo caso devolver el Response de error adecuado.

(4 puntos). Servicio REST de productos. El endpoint será /api/productos y debe implementarse un servicio REST que contenga las operaciones básicas CRUD, cumpliendo los siguientes requisitos:

- Hay que utilizar JsonTypeInfo y JsonSubTypes para conseguir que la jerarquía de producto se mapee en JSON correctamente, utilizando un atributo llamado "tipo" al hacer las peticiones REST donde se especifique el tipo de producto.
- Cuando se guarde un producto habrá que evitar ingredientes repetidos, para lo cual

habrá que implementar findByNombre en IngredienteRepository.

(3 puntos). Servicio REST de pedidos. El endpoint será /api/pedidos y debe implementarse un servicio REST que contenga las operaciones básicas CRUD, cumpliendo los siguientes requisitos:

- Se debe implementar un DTO CarritoRequest que contenga los campos necesarios para añadir al carrito. Este DTO se utilizará en el POST para recibir la información de la línea de pedido en el body.
- Se debe implementar un método en el Repository que busque los pedidos pendientes del cliente.
- En caso de que no exista un pedido pendiente se creará un nuevo pedido y en caso de que ya exista simplemente se añadirá una línea de pedido al pedido existente. En caso de que exista más de 1 pedido pendiente se debe lanzar una excepción.
- Debe implementarse el endpoints de método PUT para finalizar el pedido, pasando como parámetro un DTO FinalizarPedidoRequest (que contiene el cliente y el método de pago para realizar el pago).
- Debe implementarse el endpoint de método PUT para cancelar el pedido, pasando como parámetro el cliente.
- Debe implementarse el endpoint de método PUT para entregar el pedido, pasando como parámetro el id del pedido.

(1 punto) Vista de Clientes con Thymeleaf. Implementa un frontend con Thymeleaf que muestre la lista de los clientes actuales y el detalle de un cliente.

Ejemplo usando jsonType y jsonSubTypes para que el JSON gestione la herencia:

```
@JsonTypeInfo(use = JsonTypeInfo.Id.NAME, property =
"nombreColumnaTipo")
@JsonSubTypes({
        @JsonSubTypes.Type(value = Subclase.class, name =
"valorSubclase"),
        @JsonSubTypes.Type(value = Subclase2.class, name =
"valorSubclase2"
        })
```