Práctica 1 – Arquitectura hexagonal

Convocatoria ordinaria

Se desea implementar una aplicación de comercio electrónico. La aplicación proporciona dos casos de uso diferentes:

- CRUD de productos que los clientes podrán comprar
- Gestión del carrito de la compra:
  - Añadir productos al carrito
  - Eliminar productos del carrito
  - Eliminar el carrito
  - Finalizar el carrito. Cuando se finaliza el carrito, la aplicación debe validar el carrito, porque puede haber productos que ya no están disponibles. Para ello, la aplicación utiliza un servicio externo que se simula como un servicio Spring que simplemente devuelve aleatoriamente true o false dependiendo de si el carrito ha sido validado o no. En caso positivo, la operación de finalización del carrito termina con éxito, en caso contrario se notifica que hay productos no disponibles.

Estos casos de uso se proporcionan a través de una API REST con los siguientes endpoints:

Productos:

```
GET /api/products - Muestra los productos

POST /api/products - Añade un producto

GET /api/products/:id - Muestra un producto en específico

DELETE /api/products/:id - Borra un producto en específico
```

ShoppingCart:

```
POST /api/shoppingcarts - Crea un carrito de compra

PATCH /api/shoppingcarts/:id - Modifica el carrito de compra para pasar el
estado a completo (finalizar el carrito)

GET /api/shoppingcarts/:id - Obtiene un carrito de compra específico

DELETE /api/shoppingcarts/:id - Borra un carrito de compra específico

POST /api/shoppingcarts/:cart_id/product/:prod_id/quantity/:prod_quantity -
Añade un producto al carrito de compra, en la cantidad indicada
por :prod_quantity. Si ya existiera lo modifica con la nueva cantidad.

DELETE /api/shoppingcarts/:cart_id/product/:prod_id - Borra un producto
específico de un carrito de compra
```

Uno de los requisitos a la hora de implementar esta aplicación es que se debe seguir la **arquitectura hexagonal**. Para ello, se deben separar las clases en paquetes que representan los diferentes componentes de la aplicación. El dominio debe quedar totalmente separado de cualquier tecnología.

Se pide:

- Una implementación basada en Spring, usando H2 para la persistencia (4 pts)
- Pruebas unitarias de (1 pt):
  - Crear y borrar producto
  - Crear carrito y añadir producto al carrito
- Una implementación basada en Node con Express, usando un motor de persistencia (relacional o no relacional) y el correspondiente ORM (sequalize, mongoose…) (4 pts)
- Pruebas unitarias de (1 pt):
  - Crear y borrar producto
  - Crear carrito y añadir producto al carrito