Practica 3. Ejercicio Springboot

Crearemos una aplicación siguiendo el patrón Controlador-Servicio-Repositorio.

En la capa del controlador debemos de gestionar toda la lógica de la interfaz REST de la aplicación.

En la capa de servicio debemos de aplicar toda la lógica de nuestros casos de uso.

En la capa del repositorio debemos de utilizarlo para llamar al almacenamiento y consulta de nuestros datos. En nuestro caso, al no tener una base de datos, guardaremos los datos “a mano”.

Para ello necesitaremos crear la clase **Producto.java**.

La clase tendrá los atributos id, nombre, precio y cantidad. Debe de tener tres constructores, uno con todos los parámetros, otro que admita precio como número entero y otro que inicialice el id como nulo.

Los getters y setters los utilizaremos con la anotación de Lombok.

En el **ControladorProducto.java** debe de tener un constructor en el cual se inicialice el servicio de producto. Debe de tener los siguientes métodos y endpoints:

* cargarProductos() - /cargarProductos: Cargará por defecto los productos que definamos.
* getProductos() - /productos: Obtendrá todos los productos actuales.
* getProductoPorId() - /producto/id: Obtendrá un producto por id.
* getProductoPorNombre() - /producto/nombre: Obtendrá un producto por nombre.
* altaProducto() - /producto: Añadirá un producto con los datos que le enviemos en ProductoRequest. El id se generará de forma automática y el nombre no puede coincidir con un producto ya existente.
* modificarProducto() - /producto/id: Modificará un producto con los datos que le enviemos en ProductoRequest.
* eliminarProducto() - /producto/id: Eliminará un producto por id.
* eliminarProductos() - /productos: Eliminará todos los productos actuales.

**ProductRequest.java** y **ProductRequestMapper.java** nos ayudarán a recibir las peticiones y poder transformarlas al objeto Producto.java.

* ProductRequest.java tendrá como parámetros el nombre, precio y cantidad del Producto. Tendrá dos constructores, uno vacío y otro con todos los parámetros.
* ProductRequestMapper.java solo tendrá un método llamado toProducto() el cual recibirá un ProductRequest.java y nos devolverá un Producto.java

Practica 3. Opcional

Crearemos la clase **Usuario.java**

La clase tendrá los atributos id, username, password, carrito y productosComprados. Carrito y productos comprados serán dos listas de la clase Producto.

Debe tener dos constructores, uno con todos los parámetros y otro que solo tenga username y password.

Los getters y setters los utilizaremos con la anotación de Lombok.

En el **ControladorUsuario.java** debe de tener un constructor en el cual se inicialice el servicio del usuario. Debe de tener los siguientes métodos y endpoints:

* cargarUsuarios() - /cargarUsuarios: Cargará por defecto los usuarios que definamos.
* login() - /login: Recibé un UsuarioRequest y nos devolverá un usuario el cual coincida tanto el nombre como la contraseña.
* getUsuario() - /usuario/id: Obtendremos un usuario por id.
* altaUsuario(): - /usuario: Añadirá un usuario con los datos que le enviemos en UsuarioRequest. El id se generará de forma automática y el nombre no puede coincidir con uno ya existente.
* addProductoCarrito() - /usuario/id/carrito: Añadiremos un producto al carrito de un usuario pasándole como parámetros el id del producto y la cantidad. De no pasarle cantidad esta siempre será 1.
* comprar() - /usuario/id/comprar: Añadiremos todos los productos del carrito a los productos comprados del usuario, vaciando el carrito y eliminando la cantidad de productos de la lista de productos disponibles. Si uno de los productos no estuviera disponible la operación se cancela y devolveremos un mensaje de error.

**UsuarioRequest.java** y **UsuarioRequestMapper.java** nos ayudarán a recibir las peticiones y poder transformarlas al objeto Producto.java.

* UsuarioRequest.java tendrá como parámetros el username y password del Usuario. Tendrá dos constructores, uno vacío y otro con todos los parámetros.
* UsuarioRequestMapper.java solo tendrá un método llamado toUsuario() el cual recibirá un UsuarioRequest.java y nos devolverá un Usuario.java