Entrega de Proyecto Final

Requerimientos del curso de Front-End JS

Tendrás dos opciones para la realización de tu proyecto de "Back-End en Java":

- 1 Hacer uso del proyecto final del curso de **Front-End JS**.
- 2 Hacer uso de un template que te proporcionaremos, el cual cumple con los requisitos pero carece de estilos y personalización.



En caso de que desees continuar con el proyecto trabajando en el curso de Front-End JS, es necesario que te asegures haberlo completado correctamente. A continuación te presentaremos los requerimientos y funcionalidades esperadas para dicho proyecto:

Gestión de Productos

- La API deberá ofrecer endpoints que permitan:
- Listar productos disponibles.
- Obtener detalles individuales de un producto.
- Agregar nuevos productos al catálogo.
- Actualizar información de productos existentes.
- Eliminar productos.

Cada producto deberá tener atributos como:

- ID
- Nombre
- Descripción
- Precio
- Categoría
- Imagen (URL)
- Stock

Búsqueda y actualización de productos

- El sistema permitirá buscar un producto por su nombre o ID.
- Si se encuentra el producto, se mostrará su información completa.
- Opcionalmente, se podrá
 actualizar alguno de sus datos
 (precio o stock), validando que los
 valores sean coherentes (por
 ejemplo, que el stock no sea
 negativo).

Eliminación de productos

- El sistema debe permitir
 eliminar un producto de la lista, identificándolo por su ID o posición en la colección.
- Antes de eliminar, el sistema podría pedir confirmación (opcionales).

Creación de pedidos

- Además de manejar productos, se sugiere agregar la clase
 Pedido (o Orden) que contenga:
 - Una lista de productos asociados.
 - Cantidad deseada de cada producto (por ejemplo, usando un objeto intermedio LineaPedido o similar).

- El sistema debe permitir crear un pedido nuevo:
 - a. Solicitar al usuario qué productos desea y en qué cantidad (validar que haya suficiente stock).
 - b. Calcular el costo total (sumando precio * cantidad de cada producto).
 - c. **Disminuir el stock** de cada producto si el pedido se confirma.



Obligatorio | Entregable

Menú principal interactivo

El HTML tendrá una opción que permitirá

- Crear y registrar nuevos pedidos.
- Historial de pedidos por usuario.
- Gestión del estado de pedidos (pendiente, confirmado, enviado, entregado, cancelado).
- Actualización automática de stock al confirmar pedidoAlertas o reportes cuando el stock alcance niveles mínimos.
- Crear un pedido
- Listar pedidos



Requisitos que debe poseer el HTML:

- 1) Gestionar Productos
- 2) Gestionar Categorías
- 3) Ver Carrito de Compras
- 4) Realizar Pedido
- 5) Consultar Historial de Pedidos
- 6) Administración (usuarios y stock)
- 7) Salir



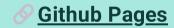
Ejemplo de la section Gestion de productos:

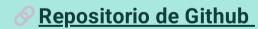
==== -----Gestion de Productos

- a) Agregar Producto
- b) Listar Productos
- c) Buscar Producto por ID
- d) Actualizar Producto
- e) Eliminar Producto
- f) Volver al menú principal

2. Template de Proyecto

En caso de elegir el template; es esencial elegir una temática adecuada que se alinee con los productos o servicios que que queres ofrecer. Una vez seleccionada la temática, realizaremos las personalizaciones y modificaciones necesarias en el template para adaptarlo a nuestras necesidades específicas.





A continuación, se presentan los elementos que deberás definir:

- 1. Nombre del Proyecto: Elegir un nombre que sea memorable y represente la esencia de la marca.
- 2. Imágenes a Utilizar: Seleccionar imágenes que resalten los productos y reflejen la temática elegida.
- 3. **Textos**: Redactar textos claros y persuasivos para las descripciones de productos, secciones del sitio y demás contenido relevante.
- 4. Paleta de Colores: Definir una paleta de colores que complemente la temática seleccionada y que ayuda a crear una experiencia visual atractiva.

Al finalizar estas definiciones, procederemos a implementar los temas que abordamos en la cursada, asegurándonos de que el resultado final sea un sitio de e-commerce atractivo y funcional.

Requerimientos de Entrega final

Objetivo general

Desarrollar una API RESTful completa en Java utilizando Spring Boot y MySQL para gestionar un sistema de E-commerce, integrándose con una aplicación frontend. La aplicación deberá aplicar correctamente conceptos avanzados de programación en Java, arquitectura REST, bases de datos relacionales, validaciones, excepciones y organización modular.



Obligatorio | Entregable

Requerimientos técnicos

Tipos de datos y variables

Colecciones (Arrays, Listas)

Emplear **variables** de tipo int (para cantidades e IDs), double (para precios), String (para nombres/descripciones), y boolean si fuera necesario.

Asegurate de **usar operadores aritméticos, lógicos y relacionales** en las funciones de cálculo y validación.

Para manejar los productos, se sugiere un ArrayList<Producto>.

Para manejar los productos dentro de un pedido, podría usarse otra lista, por ejemplo

ArrayList<LineaPedido>.

O bien, un Map<Integer, Integer> si querés asociar ID de producto con cantidad solicitada (detalles a tu elección).



Obligatorio | Entregable

POO y Colaboración de Clases

Clase Producto: con atributos id, nombre, precio, stock, getters y setters.

Clase Pedido (u Orden): con atributos id, lista de productos/lineas, metodos para calcular total, etc.

Herencia/Polimorfismo (opcional para extender)

Si deseás, podés crear subclases de Producto (por ejemplo, Bebida, Comida) con atributos específicos (fecha de vencimiento, volumen en litros, etc.).

Clase Principal (Main): orquesta el menú, invoca métodos de servicios (por ejemplo, un ProductoService que encapsule la lógica de agregar/buscar/eliminar).

Mostrar cómo el polimorfismo ayuda a tratar distintos productos de forma genérica.



Obligatorio | Entregable

Excepciones

Manejar errores con **try/catch**. Por ejemplo, al convertir datos ingresados por la usuaria o usuario, podrías

atrapar NumberFormatException si ingresa texto en lugar de un número.

Paquetes/módulos (organización de código)

Dividir las clases en paquetes lógicos:

• com.techlab.productos (para Producto, Bebida, etc.)

Ejemplo de flujo de uso (escenario)

intenta crear un pedido con cantidad mayor al stock disponible.

 El usuario entra a la página web del e-commerce y se encuentra con una barra de navegación con las siguientes opciones:

[Inicio] [Productos] [Categorías] [Carrito] [Pedidos]

1) El usuario selecciona la opción "Productos":

La página envía una petición HTTP GET a la API REST:

GET /api/productos

• La API responde con una lista en formato JSON que contiene todos los productos disponibles.

com.techlab.excepciones (para excepciones
personalizadas)

- El frontend renderiza esta lista mostrando ID, nombre, precio, categoría y stock.
- 2) El usuario hace clic en el botón "Agregar Producto":

Se abre un formulario HTML con campos para ingresar:

- Nombre del producto
- Descripción
- Precio
- Categoría (selección desplegable)
- URL de Imagen
- Cantidad en Stock

👺 Ejemplo de flujo de uso (escenario)

 Al completar el formulario y presionar "Guardar", el frontend hace una petición HTTP POST:

POST /api/productos

Con el cuerpo del mensaje JSON:

```
"nombre": "Café Premium",
  "descripcion": "Café colombiano tostado",
  "precio": 12.50,
  "categoriaId": 3,
  "imagenUrl": "http://ejemplo.com/cafe.jpg",
  "stock": 100
}
```

- La API valida y guarda el producto en la base de datos.
- El producto ahora se refleja automáticamente en la lista mostrada en el frontend.
- 3) El usuario selecciona la opción "Carrito":
 La interfaz web muestra los productos actualmente en el carrito del usuario. Al presionar "Realizar pedido", se ejecuta una petición HTTP POST:

POST /api/pedidos

Con cuerpo JSON similar a:



```
{
   "usuarioId": 5,
   "itemsPedido": [
        { "productoId": 10, "cantidad": 2 },
        { "productoId": 15, "cantidad": 1 }
   ]
}
```

- La API REST valida que haya suficiente stock para cada producto:
- Si no hay suficiente stock, responde con un código HTTP 400 Bad Request junto a un mensaje claro indicando el problema (por ejemplo, StockInsuficienteException).

- Si hay suficiente stock, la API confirma el pedido, descuenta el stock de los productos involucrados y genera un registro del pedido en estado "pendiente".
- 4) El usuario selecciona la opción "Pedidos":
 Se realiza una petición HTTP GET a la API REST:

GET /api/usuarios/5/pedidos

 La API devuelve un historial de pedidos realizados por el usuario actual y muestra los detalles como número del pedido, fecha, estado actual del pedido, costo total, y productos involucrados.

5) Finalmente, al seleccionar "Salir":

• El frontend gestiona la finalización de la sesión del usuario.