**Laboratorio 2.1: POO EN APLICACIONES WEB**

**Desarrollo de Casos Prácticos para una Zapatería**

El objetivo del presente documento es detallar los casos prácticos que deben ser desarrollados para la implementación de un sistema completo de gestión y operación de una zapatería. Estos casos prácticos deben implementar formularios reactivos en Angular, utilizando ReactiveFormsModule, FormControl, FormGroup, controles como checkbox, radio, y select, además de validaciones estándares y personalizadas.

### **1. Gestión de Inventario**

#### **Requerimientos:**

* Crear un formulario para registrar los productos de inventario.
* Cada zapato debe ser registrado con:
  + Nombre del producto.
  + Variantes: colores y tallas.
  + Marca.
  + Precio unitario.
  + Stock inicial.
* Validaciones:
  + Todos los campos son obligatorios.
  + Validación personalizada para que el stock inicial sea mayor o igual a 0.
* Mostrar en una tabla todos los productos registrados.

### **2. Control de Entrada y Salida de Productos**

#### **Requerimientos:**

* Crear un formulario para gestionar entradas y salidas de inventario.
* Los campos del formulario deben incluir:
  + Selección de producto.
  + Tipo de operación (radio): Entrada o Salida.
  + Cantidad a ingresar o retirar.
  + Fecha de la operación.
* Validaciones:
  + La cantidad debe ser un valor positivo.
  + En operaciones de salida, validar que la cantidad no exceda el stock disponible.
* Actualizar el stock de los productos según las operaciones realizadas.

### **3. Registro de Zapatos por Variantes, Colores y Tallas**

#### **Requerimientos:**

* Crear un formulario que permita registrar un zapato con:
  + Nombre.
  + Variantes de colores (checkbox para seleccionar varios colores).
  + Variantes de tallas (select para seleccionar una talla).
  + Precio.
* Validaciones:
  + El precio debe ser mayor a 0.
  + Al menos un color debe ser seleccionado.
  + Una talla debe ser seleccionada.

### **Tecnologías y Herramientas**

* **Angular**: Para la implementación de los formularios reactivos.
* **ReactiveFormsModule**: Para gestionar la lógica de los formularios.
* **Validaciones Estándares**: Uso de validadores como Validators.required, Validators.min, etc.
* **Validaciones Personalizadas**: Definición de validadores específicos para casos particulares.
* **CSS**: Para el diseño y estilo de la aplicación.

**Entrega Final:**

* Implementar los casos prácticos mencionados en una aplicación Angular.
* Asegurarse de que cada módulo sea funcional y pase las validaciones definidas.
* Documentar el funcionamiento de cada caso práctico.