**PATRONES DE DISEÑO**

**PRESENTADO POR**

**LUISA D. MORALES HURTADO**

**ANA SOFIA PALECHOR GOMEZ**

**ANGIE YULIANA NARVAEZ ROSERO**

**DOCENTE**

**MARIA ISABEL BASTIDAS**

Imagen que contiene firmar, dibujo, calle, señal

El contenido generado por IA puede ser incorrecto.

**INSTITUCIÓN UNIVERSITARIA COLEGIO MAYOR DEL CAUCA**

**INGENIERÍA INFORMÁTICA**

**POPAYÁN-CAUCA**

**2025**

PATRONES DE DISEÑO

En el desarrollo de la plataforma se aplicaron diferentes **patrones de diseño** que permiten organizar mejor el código, facilitar la extensibilidad del sistema y reducir el acoplamiento entre componentes. A continuación, se describen los principales patrones utilizados:

**1. Patrón Creacional: Factory Method**

El Factory Method se utilizó en la creación de usuarios.

Se definió una clase general Usuario, de la cual heredan Administrador, Cliente y Emprendedor. Cada tipo de usuario posee responsabilidades y comportamientos diferentes dentro de la plataforma.

El Factory Method permite que, al momento de ingresar un usuario, no sea necesario especificar en el código qué tipo concreto se va a crear. El sistema es quien se encarga de instanciar la subclase correcta según el caso.

Ventajas en el proyecto:

* Desacopla la lógica de creación de objetos del resto del sistema.
* Permite que, si en el futuro se agrega un nuevo tipo de usuario (por ejemplo, un Repartidor), no se modifique el código base, sino que solo se cree la nueva subclase.
* Mejora la mantenibilidad y escalabilidad del sistema.

**2. Patrón Estructural: Composite**

El Composite se aplicó en la relación entre Pedido y Producto.

Un Pedido puede contener varios productos, pero cada producto también puede tratarse de manera individual. En este patrón:

* Producto actúa como la hoja (unidad simple).
* Pedido actúa como el compuesto que agrupa varios productos.

Gracias al patrón Composite, se pueden recorrer los pedidos, calcular totales o aplicar operaciones sin importar si se trata de un solo producto o de un conjunto.

Ventajas en el proyecto:

* Permite tratar objetos individuales y colecciones de objetos de manera uniforme.
* Simplifica operaciones como el cálculo del costo total del carrito.
* Facilita la extensión del sistema, ya que se pueden agregar nuevos tipos de productos sin alterar el funcionamiento general.

**3. Patrón Comportamental: State**

El State se aplicó en las entidades que cambian de estado durante su ciclo de vida:

* Una Cuenta puede estar en estado *Pendiente*, *Aprobada* o *Rechazada*.
* Un Producto puede estar en estado *Validado*, *Rechazado* o *En espera*.
* Un Pedido puede estar en estado *Pendiente*, *Entregado* o *Cancelado*.

En lugar de llenar el código con condicionales (if o switch) para verificar en qué estado se encuentra un objeto, se encapsuló la lógica en clases específicas de estado.

Ventajas en el proyecto:

* Se evita la proliferación de estructuras condicionales difíciles de mantener.
* Cada estado define su propio comportamiento, lo cual facilita modificaciones o la incorporación de nuevos estados.
* El sistema es más claro, flexible y extensible.