Documentación proyecto

Autor: Juan Pablo Castaño Castaño (LordCasta en github)

Materia: Herramientas de programación II

Fecha: 29/04/2025

Universidad: I. U. Pascual Bravo

**Funcionamiento Básico de PedidosApp**

**PedidosApp** es una aplicación de escritorio desarrollada en C# Windows Forms que permite gestionar pedidos en una tienda virtual, aplicando principios de la programación orientada a objetos (POO), los principios **SOLID** y patrones de diseño como **Strategy**, **Factory Method** y **Singleton**.

**Estructura general**

1. **Formulario principal**:  
   Permite al usuario registrar un nuevo pedido indicando:
   * Nombre del cliente
   * Tipo de producto (tecnología, accesorio, componente)
   * Urgencia del pedido
   * Peso y distancia en kilómetros
2. **Asignación de estrategia de entrega** (Patrón Strategy):  
   El sistema determina automáticamente el método de entrega más adecuado de acuerdo con reglas de negocio, como:
   * Tecnología urgente → **Dron**
   * Accesorio → **Motocicleta**
   * Componente o peso > 10kg → **Camión**
   * Extensión: Accesorio, no urgente, peso < 2kg → **Bicicleta (ecológica)**
3. **Fábrica de entregas** (Patrón Factory Method):  
   Centraliza la lógica de decisión en una clase EntregaFactory, que crea el objeto de entrega apropiado en tiempo de ejecución según las condiciones del pedido.
4. **Registro de pedidos** (Patrón Singleton):  
   Todos los pedidos realizados se almacenan en una lista única accesible desde toda la aplicación mediante la clase RegistroPedidos.
5. **Formulario de historial**:  
   Muestra todos los pedidos registrados en un DataGridView. Este formulario incluye un ComboBox que permite **filtrar** los pedidos por tipo de entrega (ej.: mostrar solo los pedidos entregados por dron o bicicleta).

**Patrones de diseño aplicados**

* **Interfaces (IMetodoEntrega)**: Desacopla la clase Pedido de las clases concretas de entrega, facilitando la extensibilidad.
* **Polimorfismo**: Todas las clases de entrega (EntregaDron, EntregaMoto, EntregaCamion, EntregaBicicleta) implementan la misma interfaz y pueden ser tratadas como IMetodoEntrega.
* **Principios SOLID**:
  + **S**: Cada clase tiene una sola responsabilidad.
  + **O**: Puedes agregar nuevas entregas sin modificar código existente.
  + **L**: Sustitución segura de clases derivadas por la interfaz.
  + **I**: La interfaz es específica y enfocada.
  + **D**: Se depende de abstracciones, no de clases concretas.

**Ejemplo de funcionamiento**

1. El usuario selecciona:  
   Cliente: "Ana", Producto: "accesorio", Urgente: **No**, Peso: 1.5kg, Distancia: 3km
2. El sistema detecta que:
   * El producto es un accesorio
   * No es urgente
   * Pesa menos de 2kg  
     👉 Se asigna **Entrega en bicicleta**
3. Se calcula el costo: 3 \* 3km = $9.00
4. El pedido se guarda en el historial mediante RegistroPedidos.
5. El historial puede visualizarse y filtrarse por tipo de entrega.