TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE TIJUANA

SUBDIRECCIÓN ACADÉMICA
DEPARTAMENTO DE SISTEMAS Y COMPUTACIÓN

SEMESTRE:
Agosto - Diciembre 2025

MATERIA:
Patrones de diseño

TÍTULO ACTIVIDAD:
Examen unidad 4 y 5

UNIDAD A EVALUAR:
Unidad 5

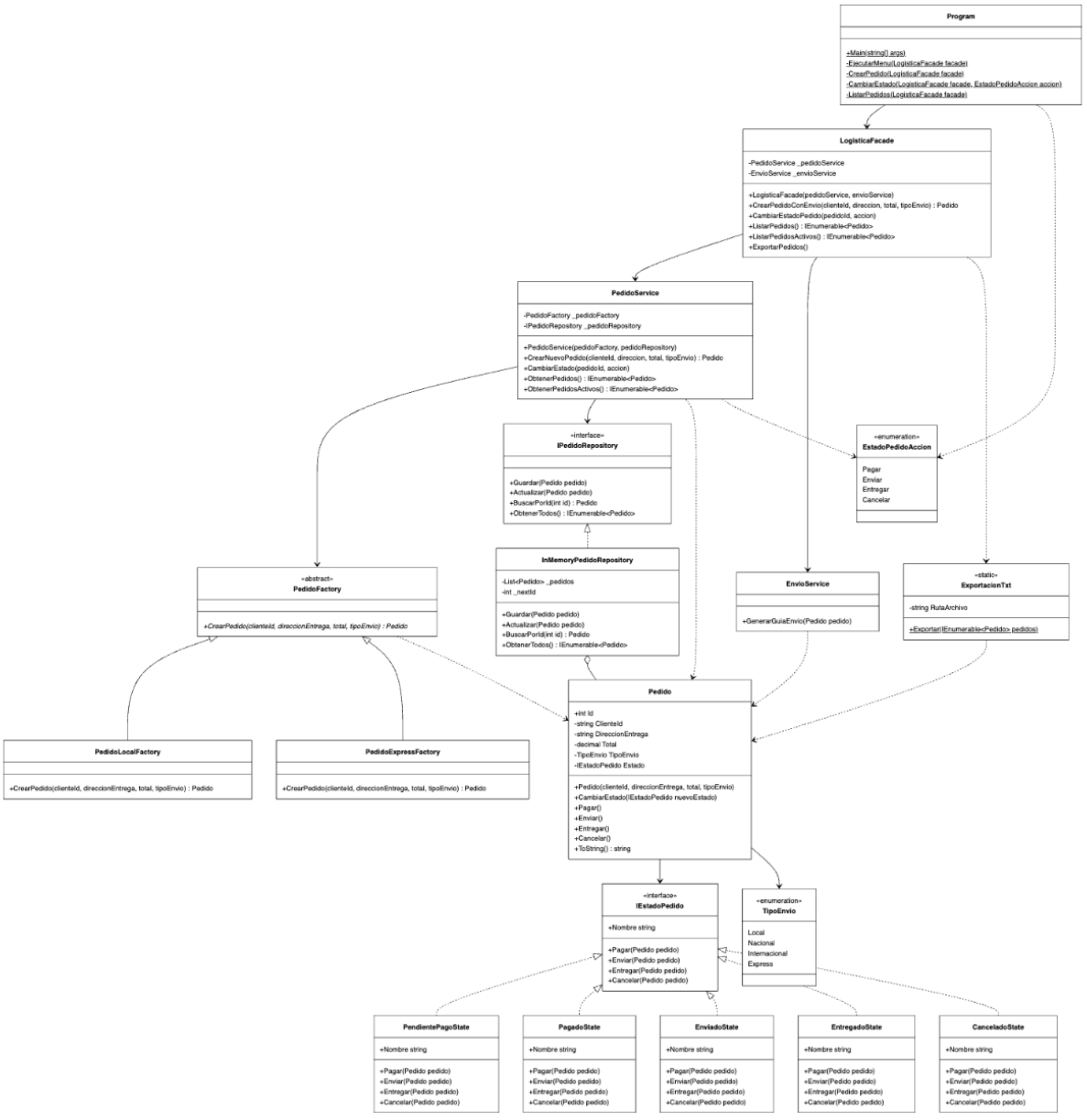
NOMBRE Y NÚMERO DE CONTROL DEL ALUMNO:

Vizuet Acosta Andre 21212372

Edgar Manuel Soriano Cabrera 21212860

NOMBRE DEL MAESTRO (A):
Maribel Guerrero Luis

UML


Program
+Main(string[] args)
-EjecutarMenu(LogisticaFacade facade)
-CrearPedido(LogisticaFacade facade)
-CambiarEstado(LogisticaFacade facade, EstadoPedidoAccion accion)
-ListarPedidos(LogisticaFacade facade)
LogisticaFacade
-PedidoService _pedidoService
-EnvioService _envioService
+LogisticaFacade(pedidoService, envioService)
+CrearPedidoConEnvio(clienteId, direccion, total, tipoEnvio) : Pedido
+CambiarEstadoPedido(pedidoId, accion)
+ListarPedidos() : IEnumerable<Pedido>
+ListarPedidosActivos() : IEnumerable<Pedido>
+ExportarPedidos()
PedidoService
-IPedidoFactory _pedidoFactory
-IPedidoRepository _pedidoRepository
+PedidoService(pedidoFactory, pedidoRepository)
+CrearNuevoPedido(clienteId, direccion, total, tipoEnvio) : Pedido
+CambiarEstado(pedidoId, accion)
+ObtenerPedidos() : IEnumerable<Pedido>
+ObtenerPedidosActivos() : IEnumerable<Pedido>
«interface»
IPedidoRepository
+Guardar(Pedido pedido)
+Actualizar(Pedido pedido)
+BuscarPorId(int id) : Pedido
+ObtenerTodos() : IEnumerable<Pedido>
«enumeration»
EstadoPedidoAccion
Pagar
Enviar
Entregar
Cancelar
«abstract»
PedidoFactory
+CrearPedido(clienteId, direccionEntrega, total, tipoEnvio) : Pedido
InMemoryPedidoRepository
-List<Pedido> _pedidos
-int _nextId
+Guardar(Pedido pedido)
+Actualizar(Pedido pedido)
+BuscarPorId(int id) : Pedido
+ObtenerTodos() : IEnumerable<Pedido>
EnvioService
+GenerarGuiaEnvio(Pedido pedido)
«static»
ExportacionTxt
-string RutaArchivo
+Exportar(IEnumerable<Pedido> pedidos)
Pedido
+int Id
-string ClienteId
-string DireccionEntrega
-decimal Total
-TipoEnvio TipoEnvio
-IEstadoPedido Estado
+Pedido(clienteId, direccionEntrega, total, tipoEnvio)
+CambiarEstado(IEstadoPedido nuevoEstado)
+Pagar()
+Enviar()
+Entregar()
+Cancelar()
+ToString() : string
PedidoLocalFactory
+CrearPedido(clienteId, direccionEntrega, total, tipoEnvio) : Pedido
PedidoExpressFactory
+CrearPedido(clienteId, direccionEntrega, total, tipoEnvio) : Pedido
«interface»
IEstadoPedido
+Nombre string
+Pagar(Pedido pedido)
+Enviar(Pedido pedido)
+Entregar(Pedido pedido)
+Cancelar(Pedido pedido)
«enumeration»
TipoEnvio
Local
Nacional
Internacional
Express
PendientePagoState
+Nombre string
+Pagar(Pedido pedido)
+Enviar(Pedido pedido)
+Entregar(Pedido pedido)
+Cancelar(Pedido pedido)
PagadoState
+Nombre string
+Pagar(Pedido pedido)
+Enviar(Pedido pedido)
+Entregar(Pedido pedido)
+Cancelar(Pedido pedido)
EnviadoState
+Nombre string
+Pagar(Pedido pedido)
+Enviar(Pedido pedido)
+Entregar(Pedido pedido)
+Cancelar(Pedido pedido)
EntregadoState
+Nombre string
+Pagar(Pedido pedido)
+Enviar(Pedido pedido)
+Entregar(Pedido pedido)
+Cancelar(Pedido pedido)
CanceladoState
+Nombre string
+Pagar(Pedido pedido)
+Enviar(Pedido pedido)
+Entregar(Pedido pedido)
+Cancelar(Pedido pedido)

Estados

```cs
using ExamenUnidad4_Patrones_VizuetAcostaAndre.Negocio.Dominio.Estados;

namespace ExamenUnidad4_Patrones_VizuetAcostaAndre.Negocio.Dominio
{
    public class Pedido
    {
        public int Id { get; set; }
        public string ClienteId { get; private set; }
        public string DireccionEntrega { get; private set; }
        public decimal Total { get; private set; }
        public TipoEnvio TipoEnvio { get; private set; }

        public IEstadoPedido Estado { get; private set; }

        public Pedido(string clienteId, string direccionEntrega, decimal total, TipoEnvio tipoEnvio)
        {
            ClienteId = clienteId;
            DireccionEntrega = direccionEntrega;
            Total = total;
            TipoEnvio = tipoEnvio;

            Estado = new PendientePagoState();
        }

        public void CambiarEstado(IEstadoPedido nuevoEstado)
        {
            Estado = nuevoEstado;
        }

        public void Pagar() => Estado.Pagar(this);
        public void Enviar() => Estado.Enviar(this);
        public void Entregar() => Estado.Entregar(this);
        public void Cancelar() => Estado.Cancelar(this);

        public override string ToString()
        {
            return $"[Id: {Id}] Cliente: {ClienteId} | Total: {Total:C} | Tipo envío: {TipoEnvio} | Estado: {Estado.Nombre}";
        }
    }
}
```

```cs
namespace ExamenUnidad4_Patrones_VizuetAcostaAndre.Negocio.Dominio.Estados
{
    public interface IEstadoPedido
    {
        string Nombre { get; }

        void Pagar(Pedido pedido);
        void Enviar(Pedido pedido);
        void Entregar(Pedido pedido);
        void Cancelar(Pedido pedido);
    }
}
```

```cs
using System;

namespace ExamenUnidad4_Patrones_VizuetAcostaAndre.Negocio.Dominio.Estados
{
    public class PendientePagoState : IEstadoPedido
    {
        public string Nombre => "Pendiente de pago";

        public void Pagar(Pedido pedido)
        {
            pedido.CambiarEstado(new PagadoState());
        }

        public void Enviar(Pedido pedido)
        {
            throw new InvalidOperationException("No se puede enviar un pedido que no ha sido pagado.");
        }

        public void Entregar(Pedido pedido)
        {
            throw new InvalidOperationException("No se puede entregar un pedido que no ha sido enviado.");
        }

        public void Cancelar(Pedido pedido)
        {
            pedido.CambiarEstado(new CanceladoState());
        }
    }
}
```

```cs
using System;

namespace ExamenUnidad4_Patrones_VizuetAcostaAndre.Negocio.Dominio.Estados
{
    public class PagadoState : IEstadoPedido
    {
        public string Nombre => "Pagado";

        public void Pagar(Pedido pedido)
        {
        }

        public void Enviar(Pedido pedido)
        {
            pedido.CambiarEstado(new EnviadoState());
        }

        public void Entregar(Pedido pedido)
        {
            throw new InvalidOperationException("No se puede entregar un pedido que no ha sido enviado.");
        }

        public void Cancelar(Pedido pedido)
        {
            pedido.CambiarEstado(new CanceladoState());
        }
    }
}
```

```cs
using System;

namespace ExamenUnidad4_Patrones_VizuetAcostaAndre.Negocio.Dominio.Estados
{
    public class EnviadoState : IEstadoPedido
    {
        public string Nombre => "Enviado";

        public void Pagar(Pedido pedido)
        {
        }

        public void Enviar(Pedido pedido)
        {
        }

        public void Entregar(Pedido pedido)
        {
            pedido.CambiarEstado(new EntregadoState());
        }

        public void Cancelar(Pedido pedido)
        {
            throw new InvalidOperationException("No se puede cancelar un pedido que ya fue enviado.");
        }
    }
}
```

```cs
using System;

namespace ExamenUnidad4_Patrones_VizuetAcostaAndre.Negocio.Dominio.Estados
{
    public class EntregadoState : IEstadoPedido
    {
        public string Nombre => "Entregado";

        public void Pagar(Pedido pedido)
        {
        }

        public void Enviar(Pedido pedido)
        {
        }

        public void Entregar(Pedido pedido)
        {
        }

        public void Cancelar(Pedido pedido)
        {
            throw new InvalidOperationException("No se puede cancelar un pedido que ya fue entregado.");
        }
    }
}
```

```cs
using System;

namespace ExamenUnidad4_Patrones_VizuetAcostaAndre.Negocio.Dominio.Estados
{
    public class CanceladoState : IEstadoPedido
    {
        public string Nombre => "Cancelado";

        public void Pagar(Pedido pedido)
        {
            throw new InvalidOperationException("No se puede pagar un pedido cancelado.");
        }

        public void Enviar(Pedido pedido)
        {
            throw new InvalidOperationException("No se puede enviar un pedido cancelado.");
        }

        public void Entregar(Pedido pedido)
        {
            throw new InvalidOperationException("No se puede entregar un pedido cancelado.");
        }

        public void Cancelar(Pedido pedido)
        {
        }
    }
}
```

Fabricas

```cs
using ExamenUnidad4_Patrones_VizuetAcostaAndre.Negocio.Dominio;

namespace ExamenUnidad4_Patrones_VizuetAcostaAndre.Negocio.Fabricas
{
    public abstract class PedidoFactory
    {
        public abstract Pedido CrearPedido(
            string clienteId,
            string direccionEntrega,
            decimal total,
            TipoEnvio tipoEnvio
        );
    }
}
```

```csharp
using ExamenUnidad4_Patrones_VizuetAcostaAndre.Negocio.Dominio;

namespace ExamenUnidad4_Patrones_VizuetAcostaAndre.Negocio.Fabricas
{
    public class PedidoExpressFactory : PedidoFactory
    {
        public override Pedido CrearPedido(string clienteId, string direccionEntrega, decimal total, TipoEnvio tipoEnvio)
        {
            decimal totalConRecargo = total * 1.10m;
            return new Pedido(clienteId, direccionEntrega, totalConRecargo, tipoEnvio);
        }
    }
}
```

```csharp
using ExamenUnidad4_Patrones_VizuetAcostaAndre.Negocio.Dominio;

namespace ExamenUnidad4_Patrones_VizuetAcostaAndre.Negocio.Fabricas
{
    public class PedidoLocalFactory : PedidoFactory
    {
        public override Pedido CrearPedido(string clienteId, string direccionEntrega, decimal total, TipoEnvio tipoEnvio)
        {
            return new Pedido(clienteId, direccionEntrega, total, tipoEnvio);
        }
    }
}
```

Servicios

```
using ExamenUnidad4_Patrones_VizuetAcostaAndre.Negocio.Dominio;

namespace ExamenUnidad4_Patrones_VizuetAcostaAndre.Negocio.Servicios
{
    public class LogisticaFacade
    {
        private readonly PedidoService _pedidoService;
        private readonly EnvioService _envioService;

        public LogisticaFacade(PedidoService pedidoService, EnvioService envioService)
        {
            _pedidoService = pedidoService;
            _envioService = envioService;
        }

        public Pedido CrearPedidoConEnvio(string clienteId, string direccion, decimal total, TipoEnvio tipoEnvio)
        {
            var pedido = _pedidoService.CrearNuevoPedido(clienteId, direccion, total, tipoEnvio);
            _envioService.GenerarGuiaEnvio(pedido);
            return pedido;
        }

        public void CambiarEstadoPedido(int pedidoId, EstadoPedidoAccion accion)
        {
            _pedidoService.CambiarEstado(pedidoId, accion);
        }

        public IEnumerable<Pedido> ListarPedidos()
        {
            return _pedidoService.ObtenerPedidos();
        }

        public IEnumerable<Pedido> ListarPedidosActivos()
        {
            return _pedidoService.ObtenerPedidosActivos();
        }

        public void ExportarPedidos()
        {
            var pedidos = _pedidoService.ObtenerPedidos();
            ExportacionTxt.Exportar(pedidos);
        }
```

```
namespace ExamenUnidad4_Patrones_VizuetAcostaAndre.Negocio.Servicios
{
    public class PedidoService
    {
        private readonly PedidoFactory _pedidoFactory;
        private readonly IPedidoRepository _pedidoRepository;

        public PedidoService(PedidoFactory pedidoFactory, IPedidoRepository pedidoRepository)
        {
            _pedidoFactory = pedidoFactory;
            _pedidoRepository = pedidoRepository;
        }

        public Pedido CrearNuevoPedido(string clienteId, string direccion, decimal total, TipoEnvio tipoEnvio)
        {
            var pedido = _pedidoFactory.CrearPedido(clienteId, direccion, total, tipoEnvio);
            _pedidoRepository.Guardar(pedido);
            return pedido;
        }

        public void CambiarEstado(int pedidoId, EstadoPedidoAccion accion)
        {
            var pedido = _pedidoRepository.BuscarPorId(pedidoId);

            if (pedido == null)
            {
                throw new InvalidOperationException("Pedido no encontrado.");
            }

            switch (accion)
            {
                case EstadoPedidoAccion.Pagar:
                    pedido.Pagar();
                    break;
                case EstadoPedidoAccion.Enviar:
                    pedido.Enviar();
                    break;
                case EstadoPedidoAccion.Entregar:
                    pedido.Entregar();
                    break;
                case EstadoPedidoAccion.Cancelar:
                    pedido.Cancelar();
```

```
            }

            _pedidoRepository.Actualizar(pedido);
        }

        public IEnumerable<Pedido> ObtenerPedidos()
        {
            return _pedidoRepository.ObtenerTodos();
        }

        public IEnumerable<Pedido> ObtenerPedidosActivos()
        {
            return _pedidoRepository.ObtenerTodos()
                .Where(p => p.Estado.Nombre != "Cancelado");
        }
    }
}
```

```
using System;
using ExamenUnidad4_Patrones_VizuetAcostaAndre.Negocio.Dominio;

namespace ExamenUnidad4_Patrones_VizuetAcostaAndre.Negocio.Servicios
{
    public class EnvioService
    {
        public void GenerarGuiaEnvio(Pedido pedido)
        {
            Console.WriteLine($"[ENVÍO] Se generó la guía para el pedido {pedido.Id} ({pedido.TipoEnvio}).");
        }
    }
}
```

```
using System;
using System.IO;
using System.Collections.Generic;
using ExamenUnidad4_Patrones_VizuetAcostaAndre.Negocio.Dominio;

namespace ExamenUnidad4_Patrones_VizuetAcostaAndre.Negocio.Servicios
{
    public static class ExportacionTxt
    {
        private static readonly string RutaArchivo = "pedidos_exportados.txt";

        public static void Exportar(IEnumerable<Pedido> pedidos)
        {
            using (StreamWriter writer = new StreamWriter(RutaArchivo, append: false))
            {
                writer.WriteLine("=== LISTADO DE PEDIDOS ===");
                writer.WriteLine($"Fecha de exportación: {DateTime.Now}");
                writer.WriteLine("-----------------------------------------------");

                foreach (var p in pedidos)
                {
                    writer.WriteLine($"ID: {p.Id}");
                    writer.WriteLine($"Cliente: {p.ClienteId}");
                    writer.WriteLine($"Dirección: {p.DireccionEntrega}");
                    writer.WriteLine($"Total: {p.Total:C}");
                    writer.WriteLine($"Tipo Envío: {p.TipoEnvio}");
                    writer.WriteLine($"Estado: {p.Estado.Nombre}");
                    writer.WriteLine("-----------------------------------------------");
                }
            }
        }
    }
}
```

Datos

```
using System.Collections.Generic;
using System.Linq;
using ExamenUnidad4_Patrones_VizuetAcostaAndre.Negocio.Dominio;
using ExamenUnidad4_Patrones_VizuetAcostaAndre.Negocio.Servicios;

namespace ExamenUnidad4_Patrones_VizuetAcostaAndre.Datos
{
    public class InMemoryPedidoRepository : IPedidoRepository
    {
        private readonly List<Pedido> _pedidos = new();
        private int _nextId = 1;

        public void Guardar(Pedido pedido)
        {
            pedido.Id = _nextId++;
            _pedidos.Add(pedido);
        }

        public void Actualizar(Pedido pedido)
        {
            var index = _pedidos.FindIndex(p => p.Id == pedido.Id);
            if (index >= 0)
            {
                _pedidos[index] = pedido;
            }
        }

        public Pedido? BuscarPorId(int id)
        {
            return _pedidos.FirstOrDefault(p => p.Id == id);
        }

        public IEnumerable<Pedido> ObtenerTodos()
        {
            return _pedidos.ToList();
        }
    }
}
```

Main

```
using System;
using ExamenUnidad4_Patrones_VizuetAcostaAndre.Datos;
using ExamenUnidad4_Patrones_VizuetAcostaAndre.Negocio.Dominio;
using ExamenUnidad4_Patrones_VizuetAcostaAndre.Negocio.Fabricas;
using ExamenUnidad4_Patrones_VizuetAcostaAndre.Negocio.Servicios;

namespace ExamenUnidad4_Patrones_VizuetAcostaAndre.Presentacion
{
    public class Program
    {
        public static void Main(string[] args)
        {
            var pedidoRepository = new InMemoryPedidoRepository();

            PedidoFactory pedidoFactory = new PedidoLocalFactory();

            var pedidoService = new PedidoService(pedidoFactory, pedidoRepository);
            var envioService = new EnvioService();

            var facade = new LogisticaFacade(pedidoService, envioService);

            EjecutarMenu(facade);
        }

        private static void EjecutarMenu(LogisticaFacade facade)
        {
            string opcion;
            do
            {
                Console.WriteLine("═══════════════════════════════════════");
                Console.WriteLine("  Gestión de Entregas de Comercio Electrónico");
                Console.WriteLine("═══════════════════════════════════════");
                Console.WriteLine("1. Crear nuevo pedido");
                Console.WriteLine("2. Pagar pedido");
                Console.WriteLine("3. Enviar pedido");
                Console.WriteLine("4. Entregar pedido");
                Console.WriteLine("5. Cancelar pedido");
                Console.WriteLine("6. Listar pedidos");
                Console.WriteLine("7. Exportar pedidos a TXT");
                Console.WriteLine("0. Salir");
                Console.Write("Seleccione una opción: ");
                opcion = Console.ReadLine() ?? "";
```

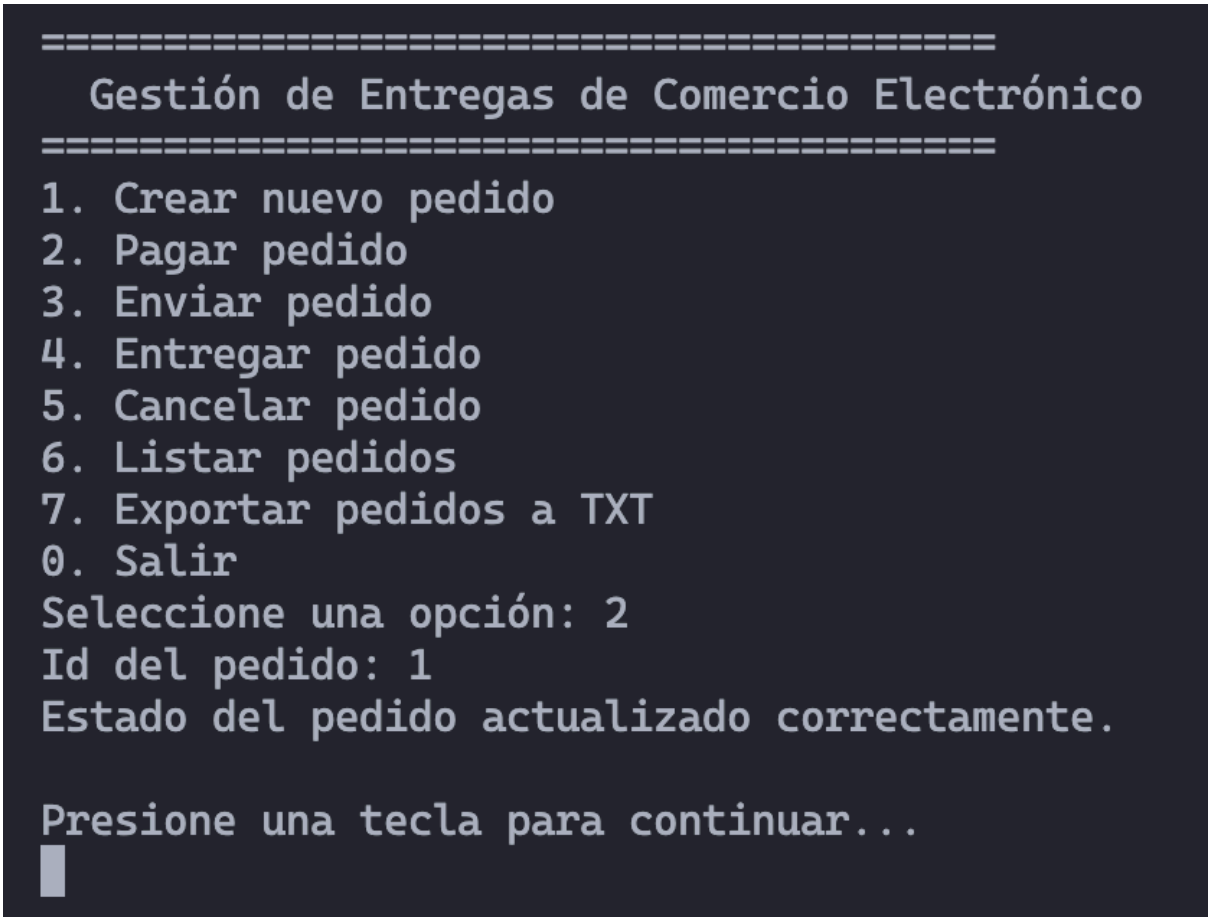
```
44                try
45                {
46                    switch (opcion)
47                    {
48                        case "1":
49                            CrearPedido(facade);
50                            break;
51                        case "2":
52                            CambiarEstado(facade, EstadoPedidoAccion.Pagar);
53                            break;
54                        case "3":
55                            CambiarEstado(facade, EstadoPedidoAccion.Enviar);
56                            break;
57                        case "4":
58                            CambiarEstado(facade, EstadoPedidoAccion.Entregar);
59                            break;
60                        case "5":
61                            CambiarEstado(facade, EstadoPedidoAccion.Cancelar);
62                            break;
63                        case "6":
64                            ListarPedidos(facade);
65                            break;
66                        case "7":
67                            facade.ExportarPedidos();
68                            Console.WriteLine("\nArchivo generado: pedidos_exportados.txt");
69                            break;
70                        case "0":
71                            Console.WriteLine("Saliendo ... ");
72                            break;
73                        default:
74                            Console.WriteLine("Opción no válida.");
75                            break;
76                    }
77                }
```

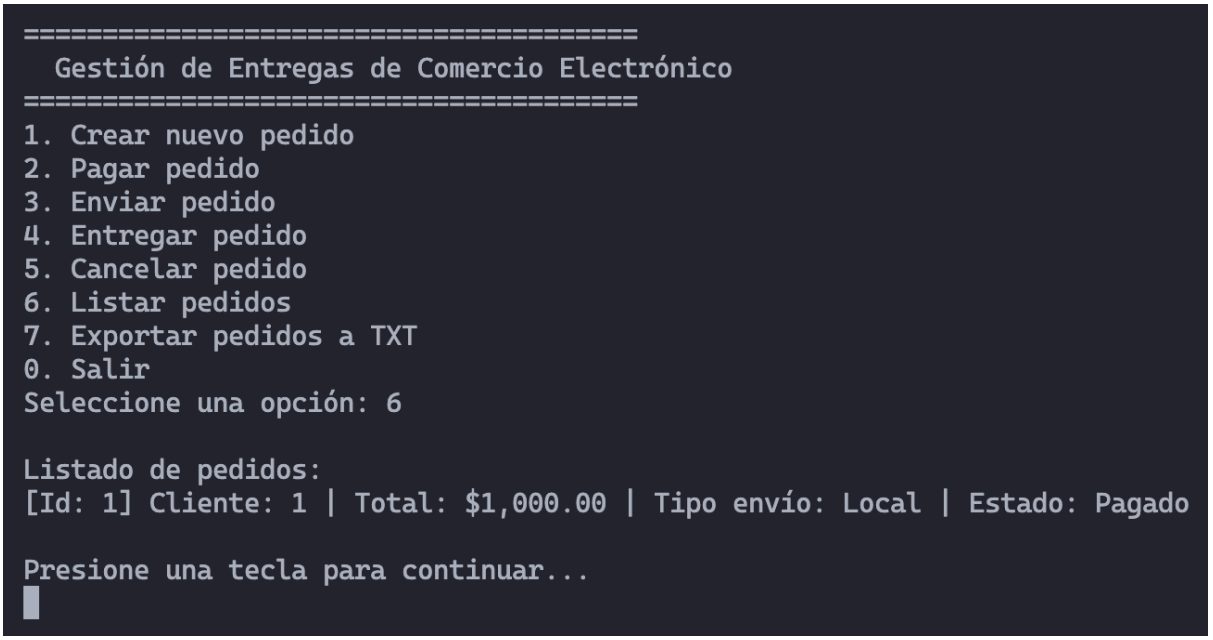
```
78                  catch (Exception ex)
79                  {
80                      Console.WriteLine($"[ERROR] {ex.Message}");
81                  }
82
83                  if (opcion ≠ "0")
84                  {
85                      Console.WriteLine("\nPresione una tecla para continuar ... ");
86                      Console.ReadKey();
87                      Console.Clear();
88                  }
89
90              } while (opcion ≠ "0");
91          }
92
93          private static void CrearPedido(LogisticaFacade facade)
94          {
95              Console.Write("Id del cliente: ");
96              var clienteId = Console.ReadLine() ?? "";
97
98              Console.Write("Dirección de entrega: ");
99              var direccion = Console.ReadLine() ?? "";
100
101             Console.Write("Total del pedido: ");
102             var totalTexto = Console.ReadLine();
103             decimal total = decimal.TryParse(totalTexto, out var t) ? t : 0m;
104
105             Console.WriteLine("Tipo de envío:");
106             Console.WriteLine("1. Local");
107             Console.WriteLine("2. Nacional");
108             Console.WriteLine("3. Internacional");
109             Console.WriteLine("4. Express");
110             Console.Write("Seleccione tipo de envío: ");
111             var tipoTexto = Console.ReadLine();
112
113             int tipoNumero = int.TryParse(tipoTexto, out var tempTipo) ? tempTipo : 1;
114             var tipoEnvio = (TipoEnvio)tipoNumero;
```

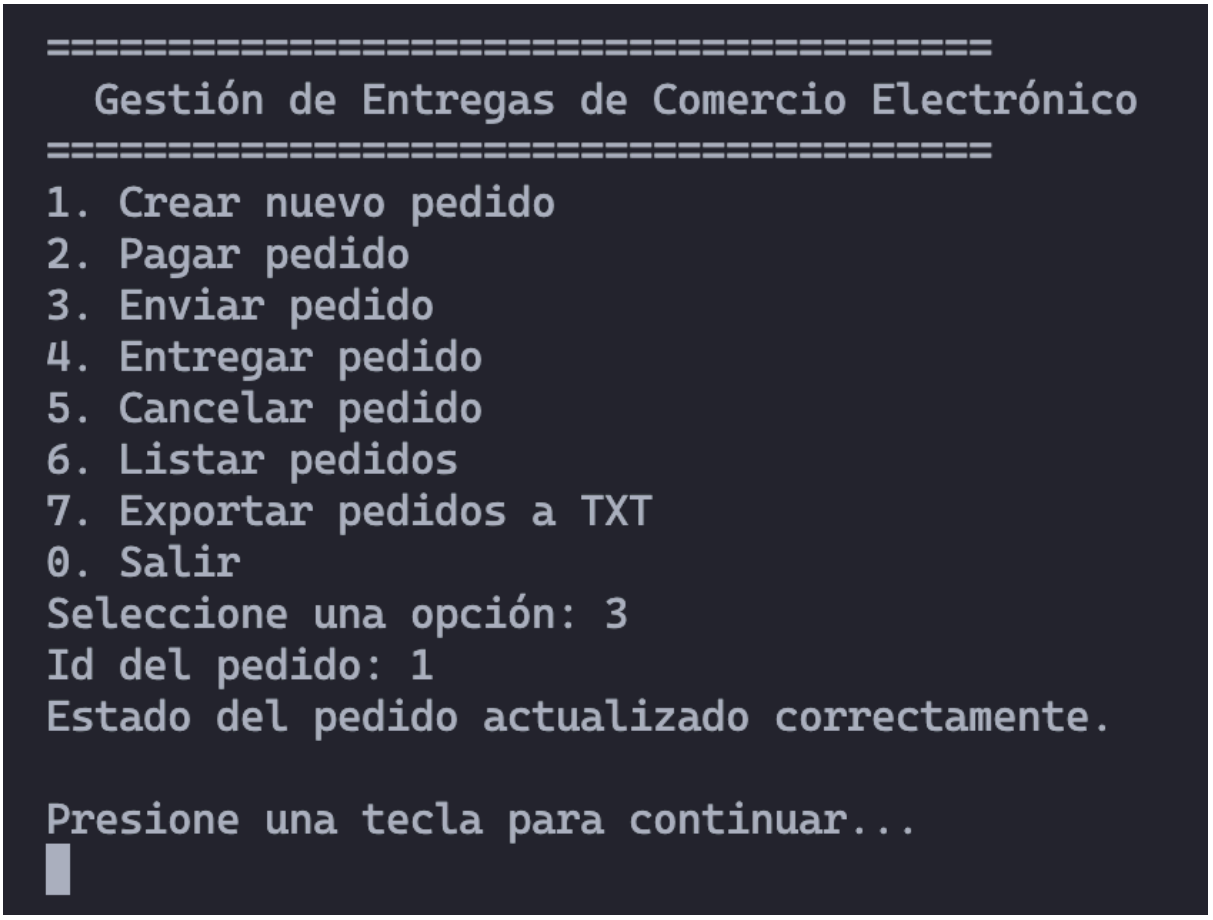
```
116             var pedido = facade.CrearPedidoConEnvio(clienteId, direccion, total, tipoEnvio);
117
118             Console.WriteLine("\nPedido creado correctamente:");
119             Console.WriteLine(pedido);
120         }
121
122         private static void CambiarEstado(LogisticaFacade facade, EstadoPedidoAccion accion)
123         {
124             Console.Write("Id del pedido: ");
125             var idTexto = Console.ReadLine();
126             int id = int.TryParse(idTexto, out var tempId) ? tempId : 0;
127
128             facade.CambiarEstadoPedido(id, accion);
129             Console.WriteLine("Estado del pedido actualizado correctamente.");
130         }
131
132         private static void ListarPedidos(LogisticaFacade facade)
133         {
134             var pedidos = facade.ListarPedidos();
135
136             Console.WriteLine("\nListado de pedidos:");
137             foreach (var p in pedidos)
138             {
139                 Console.WriteLine(p);
140             }
141         }
142     }
143 }
```

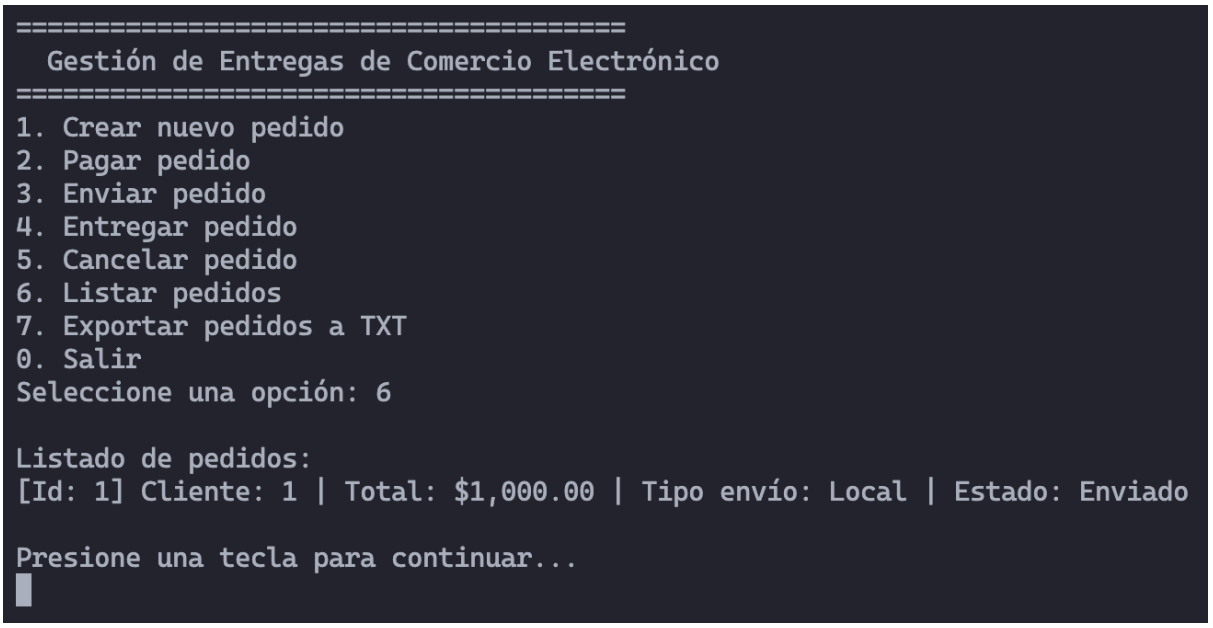
Ejecucion

```
ExamenUnidad4_Patrones_VizuetAcostaAndre/src on  main [?] via .NET v9.0.102 🎯 net9.0
❯ dotnet run
=====================================
  Gestión de Entregas de Comercio Electrónico
=====================================
1. Crear nuevo pedido
2. Pagar pedido
3. Enviar pedido
4. Entregar pedido
5. Cancelar pedido
6. Listar pedidos
7. Exportar pedidos a TXT
0. Salir
Seleccione una opción: 1
Id del cliente: 1
Dirección de entrega: Tijuana
Total del pedido: 1000
Tipo de envío:
1. Local
2. Nacional
3. Internacional
4. Express
Seleccione tipo de envío: 1
[ENVÍO] Se generó la guía para el pedido 1 (Local).

Pedido creado correctamente:
[Id: 1] Cliente: 1 | Total: $1,000.00 | Tipo envío: Local | Estado: Pendiente de pago

Presione una tecla para continuar...
```


==========
Gestión de Entregas de Comercio Electrónico
==========
1. Crear nuevo pedido
2. Pagar pedido
3. Enviar pedido
4. Entregar pedido
5. Cancelar pedido
6. Listar pedidos
7. Exportar pedidos a TXT
0. Salir
Seleccione una opción: 2
Id del pedido: 1
Estado del pedido actualizado correctamente.
Presione una tecla para continuar...


==========
Gestión de Entregas de Comercio Electrónico
==========
1. Crear nuevo pedido
2. Pagar pedido
3. Enviar pedido
4. Entregar pedido
5. Cancelar pedido
6. Listar pedidos
7. Exportar pedidos a TXT
0. Salir
Seleccione una opción: 6
Listado de pedidos:
[Id: 1] Cliente: 1 | Total: $1,000.00 | Tipo envío: Local | Estado: Pagado
Presione una tecla para continuar...


==========
Gestión de Entregas de Comercio Electrónico
==========
1. Crear nuevo pedido
2. Pagar pedido
3. Enviar pedido
4. Entregar pedido
5. Cancelar pedido
6. Listar pedidos
7. Exportar pedidos a TXT
0. Salir
Seleccione una opción: 3
Id del pedido: 1
Estado del pedido actualizado correctamente.
Presione una tecla para continuar...


==========
Gestión de Entregas de Comercio Electrónico
==========
1. Crear nuevo pedido
2. Pagar pedido
3. Enviar pedido
4. Entregar pedido
5. Cancelar pedido
6. Listar pedidos
7. Exportar pedidos a TXT
0. Salir
Seleccione una opción: 6
Listado de pedidos:
[Id: 1] Cliente: 1 | Total: $1,000.00 | Tipo envío: Local | Estado: Enviado
Presione una tecla para continuar...

```
=====================================
  Gestión de Entregas de Comercio Electrónico
=====================================
1. Crear nuevo pedido
2. Pagar pedido
3. Enviar pedido
4. Entregar pedido
5. Cancelar pedido
6. Listar pedidos
7. Exportar pedidos a TXT
0. Salir
Seleccione una opción: 5
Id del pedido: 1
[ERROR] No se puede cancelar un pedido que ya fue enviado.

Presione una tecla para continuar...
```

```
=====================================
  Gestión de Entregas de Comercio Electrónico
=====================================
1. Crear nuevo pedido
2. Pagar pedido
3. Enviar pedido
4. Entregar pedido
5. Cancelar pedido
6. Listar pedidos
7. Exportar pedidos a TXT
0. Salir
Seleccione una opción: 4
Id del pedido: 1
Estado del pedido actualizado correctamente.

Presione una tecla para continuar...
```

```
========================================
  Gestión de Entregas de Comercio Electrónico
========================================
1. Crear nuevo pedido
2. Pagar pedido
3. Enviar pedido
4. Entregar pedido
5. Cancelar pedido
6. Listar pedidos
7. Exportar pedidos a TXT
0. Salir
Seleccione una opción: 7

Archivo generado: pedidos_exportados.txt

Presione una tecla para continuar...
```

```
1  ═══ LISTADO DE PEDIDOS ═══
2  Fecha de exportación: 12/10/2025 11:58:28 AM
3  ──────────────────────────────────────────────
4  ID: 1
5  Cliente: 1
6  Dirección: Tijuana
7  Total: $1,000.00
8  Tipo Envío: Local
9  Estado: Entregado
10 ──────────────────────────────────────────────
11
```

Conclusion

En este proyecto, aplique 4 patrones, el primero fue factory method que lo utilizo para la creación de pedidos segun el tipo de envio, el segundo fue facade donde se oculta la creación de pedidos, el cambio de estados y se da una forma sencilla de usar, state para manejar el ciclo de vida del pedido y por ultimo la arquitectura por capas que facilita la mantenibilidad, escalabilidad y claridad del sistema.