INSTITUTO TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE TIJUANA

MATERIA

Patrones de diseño

NOMBRE DEL TRABAJO

Examen Unidad 4 y 5

UNIDAD 4 y 5

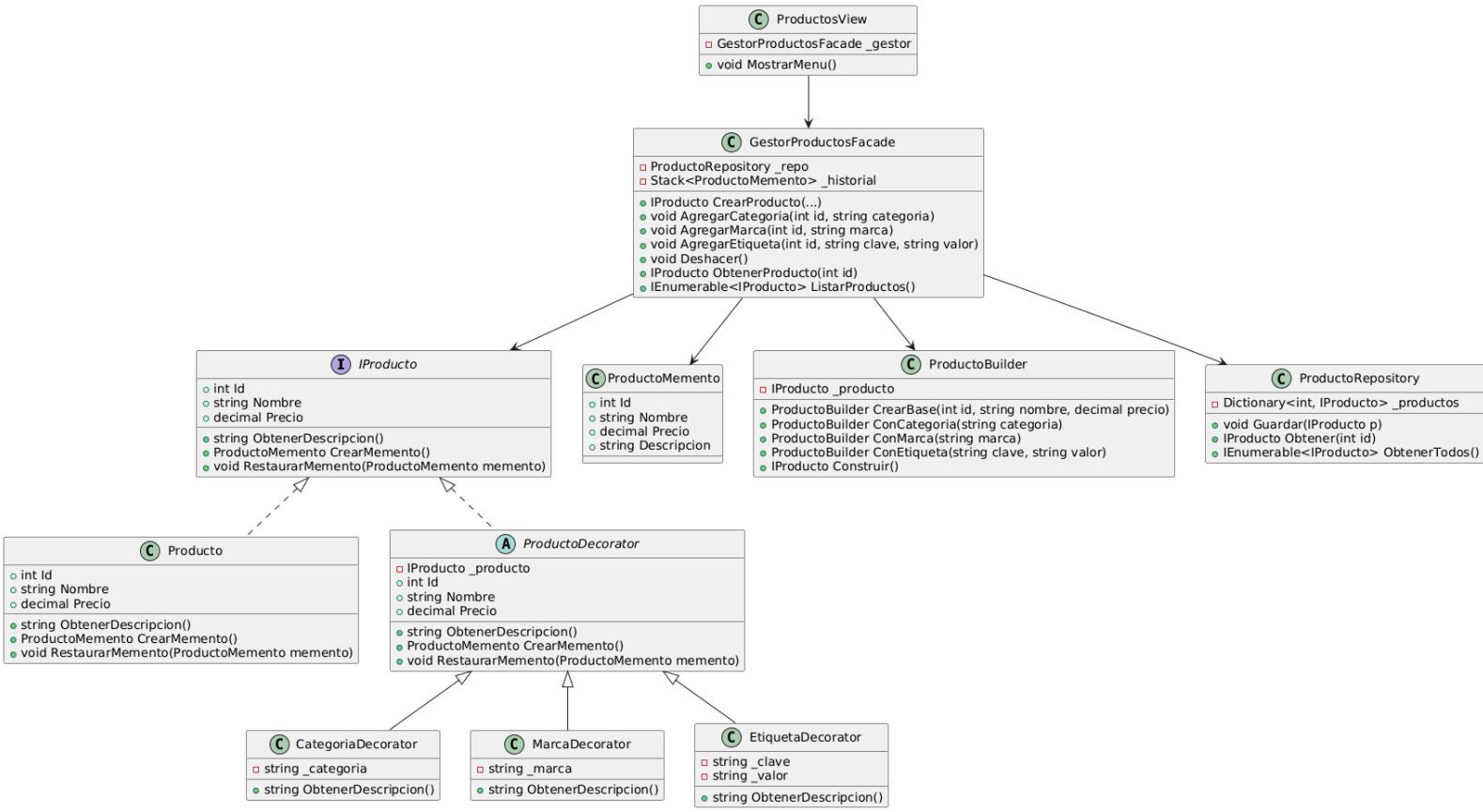
ALUMNO

Sandoval Ortiz Andy - 21210547

MAESTRO

Maribel Guerrero Luis

1. Diagrama UML



2. Código

```
0 referencias
class Program
{
    0 referencias
    static void Main()
    {
        var view = new ProductosView();
        view.MostrarMenu();
    }
}
```

```
using System;
using System.Collections.Generic;

1 referencia
public class ProductosView
{
    private readonly GestorProductosFacade _gestor = new();

    1 referencia
    public void MostrarMenu()
    {
        int opcion;
        do
        {
            Console.Clear();
            Console.WriteLine("====== GESTOR DE PRODUCTOS ======");
            Console.WriteLine("1. Crear producto");
            Console.WriteLine("2. Agregar categoría");
            Console.WriteLine("3. Agregar marca");
            Console.WriteLine("4. Agregar etiqueta");
            Console.WriteLine("5. Ver producto");
            Console.WriteLine("6. Listar todos los productos");
            Console.WriteLine("7. Deshacer último cambio");
            Console.WriteLine("0. Salir");
            Console.Write("Seleccione opción: ");

            if (!int.TryParse(Console.ReadLine(), out opcion)) opcion = -1;
            Console.Clear();

            switch (opcion)
            {
                case 1: CrearProducto(); break;
                case 2: AgregarCategoria(); break;
                case 3: AgregarMarca(); break;
                case 4: AgregarEtiqueta(); break;
                case 5: VerProducto(); break;
                case 6: ListarProductos(); break;
                case 7: _gestor.Deshacer(); break;
                case 0: Console.WriteLine("Saliendo..."); break;
                default: Console.WriteLine("Opción inválida"); break;
            }
```

```
            if (opcion != 0)
            {
                Console.WriteLine("\nPresione una tecla...");
                Console.ReadKey();
            }

        } while (opcion != 0);
    }

    1 referencia
    private void CrearProducto()
    {
        Console.Write("ID: "); int id = int.Parse(Console.ReadLine());
        Console.Write("Nombre: "); string nombre = Console.ReadLine();
        Console.Write("Precio: "); decimal precio = decimal.Parse(Console.ReadLine());
        Console.Write("Categoría (opcional): "); string cat = Console.ReadLine();
        Console.Write("Marca (opcional): "); string marca = Console.ReadLine();

        var producto = _gestor.CrearProducto(id, nombre, precio,
                                             string.IsNullOrEmpty(cat) ? null : cat,
                                             string.IsNullOrEmpty(marca) ? null : marca);
        Console.WriteLine("Producto creado:\n" + producto.ObtenerDescripcion());
    }

    1 referencia
    private void AgregarCategoria()
    {
        Console.Write("ID: "); int id = int.Parse(Console.ReadLine());
        Console.Write("Categoría: "); string cat = Console.ReadLine();
        _gestor.AgregarCategoria(id, cat);
    }

    1 referencia
    private void AgregarMarca()
    {
        Console.Write("ID: "); int id = int.Parse(Console.ReadLine());
        Console.Write("Marca: "); string marca = Console.ReadLine();
        _gestor.AgregarMarca(id, marca);
    }

    1 referencia
    private void AgregarEtiqueta()
    {
        Console.Write("ID: "); int id = int.Parse(Console.ReadLine());
        Console.Write("Clave: "); string clave = Console.ReadLine();
        Console.Write("Valor: "); string valor = Console.ReadLine();
        _gestor.AgregarEtiqueta(id, clave, valor);
    }
```

```
        1 referencia
        private void VerProducto()
        {
            Console.Write("ID: "); int id = int.Parse(Console.ReadLine());
            var p = _gestor.ObtenerProducto(id);
            if (p != null) Console.WriteLine(p.ObtenerDescripcion());
            else Console.WriteLine("Producto no encontrado.");
        }

        1 referencia
        private void ListarProductos()
        {
            foreach (var p in _gestor.ListarProductos())
                Console.WriteLine(p.ObtenerDescripcion());
        }
    }
```

```
using System.Collections.Generic;

2 referencias
public class ProductoRepository
{
    private readonly Dictionary<int, IProducto> _productos = new();

    2 referencias
    public void Guardar(IProducto p) => _productos[p.Id] = p;
    3 referencias
    public IProducto Obtener(int id) => _productos.ContainsKey(id) ? _productos[id] : null;
    1 referencia
    public IEnumerable<IProducto> ObtenerTodos() => _productos.Values;
}
```

```
9 referencias
public class ProductoMemento
{
    2 referencias
    public int Id { get; }
    1 referencia
    public string Nombre { get; }
    1 referencia
    public decimal Precio { get; }
    1 referencia
    public string Descripcion { get; }

    1 referencia
    public ProductoMemento(int id, string nombre, decimal precio, string descripcion)
    {
        Id = id;
        Nombre = nombre;
        Precio = precio;
        Descripcion = descripcion;
    }
}
```

```
7 referencias
public abstract class ProductoDecorator : IProducto
{
    protected IProducto _producto;

    3 referencias
    protected ProductoDecorator(IProducto producto)
    {
        _producto = producto;
    }

    3 referencias
    public virtual int Id => _producto.Id;
    2 referencias
    public virtual string Nombre => _producto.Nombre;
    2 referencias
    public virtual decimal Precio => _producto.Precio;
    11 referencias
    public virtual string ObtenerDescripcion() => _producto.ObtenerDescripcion();
    3 referencias
    public virtual ProductoMemento CrearMemento() => _producto.CrearMemento();
    3 referencias
    public virtual void RestaurarMemento(ProductoMemento memento) => _producto.RestaurarMemento(memento);
}
```

```
5 referencias
public class ProductoBuilder
{
    private IProducto _producto;

    1 referencia
    public ProductoBuilder CrearBase(int id, string nombre, decimal precio)
    {
        _producto = new Producto(id, nombre, precio);
        return this;
    }

    1 referencia
    public ProductoBuilder ConCategoria(string categoria)
    {
        _producto = new CategoriaDecorator(_producto, categoria);
        return this;
    }

    1 referencia
    public ProductoBuilder ConMarca(string marca)
    {
        _producto = new MarcaDecorator(_producto, marca);
        return this;
    }

    1 referencia
    public ProductoBuilder ConEtiqueta(string clave, string valor)
    {
        _producto = new EtiquetaDecorator(_producto, clave, valor);
        return this;
    }

    1 referencia
    public IProducto Construir() => _producto;
}
```

```
2 referencias
public class Producto : IProducto
{
    5 referencias
    public int Id { get; }
    5 referencias
    public string Nombre { get; }
    5 referencias
    public decimal Precio { get; }

    1 referencia
    public Producto(int id, string nombre, decimal precio)
    {
        Id = id;
        Nombre = nombre;
        Precio = precio;
    }

    6 referencias
    public virtual string ObtenerDescripcion() => $"{Nombre} - {Precio:C}";

    3 referencias
    public virtual ProductoMemento CrearMemento() =>
        new ProductoMemento(Id, Nombre, Precio, ObtenerDescripcion());

    3 referencias
    public virtual void RestaurarMemento(ProductoMemento memento) { }
}
```

```
3 referencias
public class MarcaDecorator : ProductoDecorator
{
    private readonly string _marca;
    2 referencias
    public MarcaDecorator(IProducto producto, string marca) : base(producto) => _marca = marca;

    9 referencias
    public override string ObtenerDescripcion() => base.ObtenerDescripcion() + $" | Marca: {_marca}";
}
```

```
1 referencia
public interface IProducto
{
    6 referencias
    int Id { get; }
    6 referencias
    string Nombre { get; }
    6 referencias
    decimal Precio { get; }
    13 referencias
    string ObtenerDescripcion();

    4 referencias
    ProductoMemento CrearMemento();
    4 referencias
    void RestaurarMemento(ProductoMemento memento);
}
```

```
using System;
using System.Collections.Generic;

2 referencias
public class GestorProductosFacade
{
    private readonly ProductoRepository _repo = new();
    private readonly Stack<ProductoMemento> _historial = new();

    1 referencia
    public IProducto CrearProducto(int id, string nombre, decimal precio,
                                   string categoria = null, string marca = null,
                                   Dictionary<string, string> etiquetas = null)
    {
        var builder = new ProductoBuilder().CrearBase(id, nombre, precio);

        if (!string.IsNullOrEmpty(categoria)) builder.ConCategoria(categoria);
        if (!string.IsNullOrEmpty(marca)) builder.ConMarca(marca);
        if (etiquetas != null)
            foreach (var kv in etiquetas)
                builder.ConEtiqueta(kv.Key, kv.Value);

        var producto = builder.Construir();
        _repo.Guardar(producto);
        return producto;
    }

    1 referencia
    public void AgregarCategoria(int id, string categoria) => Decorar(id, p => new CategoriaDecorator(p, categoria));
    1 referencia
    public void AgregarMarca(int id, string marca) => Decorar(id, p => new MarcaDecorator(p, marca));
    1 referencia
    public void AgregarEtiqueta(int id, string clave, string valor) => Decorar(id, p => new EtiquetaDecorator(p, clave, valor));
```

```
    3 referencias
    private void Decorar(int id, Func<IProducto, IProducto> decorador)
    {
        var p = _repo.Obtener(id);
        if (p != null)
        {
            _historial.Push(p.CrearMemento());
            p = decorador(p);
            _repo.Guardar(p);
        }
    }

    1 referencia
    public void Deshacer()
    {
        if (_historial.Count == 0) return;
        var memento = _historial.Pop();
        var p = _repo.Obtener(memento.Id);
        p?.RestaurarMemento(memento);
    }

    1 referencia
    public IProducto ObtenerProducto(int id) => _repo.Obtener(id);
    1 referencia
    public IEnumerable<IProducto> ListarProductos() => _repo.ObtenerTodos();
}
```

```
3 referencias
public class EtiquetaDecorator : ProductoDecorator
{
    private readonly string _clave;
    private readonly string _valor;
    2 referencias
    public EtiquetaDecorator(IProducto producto, string clave, string valor) : base(producto)
    {
        _clave = clave;
        _valor = valor;
    }

    9 referencias
    public override string ObtenerDescripcion() => base.ObtenerDescripcion() + $" | {_clave}: {_valor}";
}
```

```
3 referencias
public class CategoriaDecorator : ProductoDecorator
{
    private readonly string _categoria;
    2 referencias
    public CategoriaDecorator(IProducto producto, string categoria) : base(producto) => _categoria = categoria;

    9 referencias
    public override string ObtenerDescripcion() => base.ObtenerDescripcion() + $" | Categoría: {_categoria}";
}
```

3. Código Funcional