TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE TIJUANA

SUBDIRECCIÓN ACADÉMICA

DEPARTAMENTO DE SISTEMAS Y COMPUTACIÓN

SEMESTRE

AGOSTO - DICIEMBRE 2025

NOMBRE DE CARRERA

INGENIERÍA EN SISTEMAS COMPUTACIONALES

NOMBRE DE LA MATERIA

PATRONES DE DISEÑO

TÍTULO DE TRABAJO

Examen unidad 3

UNIDAD A EVALUAR

UNIDAD 3

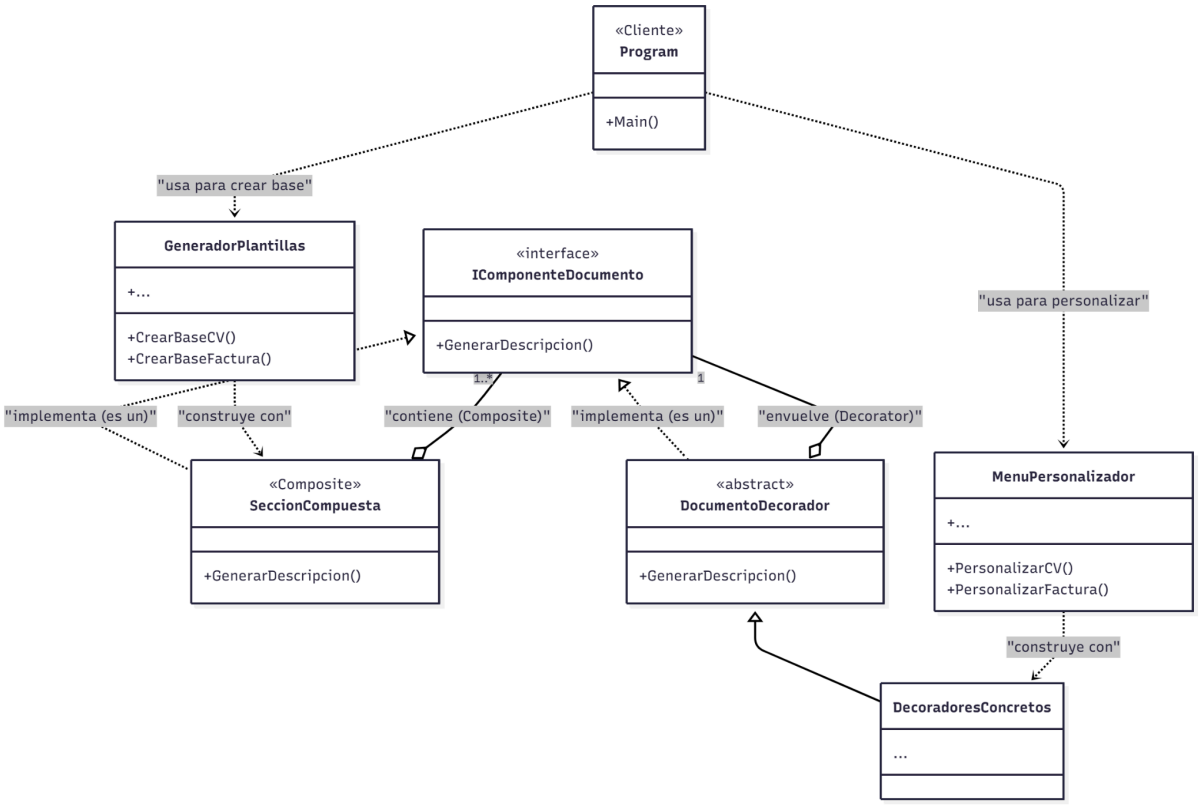
NOMBRE DEL ESTUDIANTE

HUERTA RIVAS OLAF ALEXANDRO - 21211966

MAESTRO

MARIBEL GUERRERO LUIS

Diagrama UML



Código

LogicaAplicacion.cs

```
public class GeneradorPlantillas
{
    public IComponenteDocumento CrearBaseCV()
    {
        var cv = new SeccionCompuesta("CV");
        cv.Agregar(new ElementoSimple("Datos Personales"));
        cv.Agregar(new SeccionCompuesta("Experiencia"));
        return cv;
    }
    public IComponenteDocumento CrearBaseFactura()
    {
        var factura = new SeccionCompuesta("Factura");
        factura.Agregar(new SeccionCompuesta("Tabla de Conceptos"));
        return factura;
    }
    public IComponenteDocumento CrearBaseFolleto()
    {
        var folleto = new SeccionCompuesta("Folleto");
        folleto.Agregar(new ElementoSimple("Cara Frontal"));
```

```
        return folleto;
    }
    public IComponenteDocumento CrearBaseCertificado()
    {
        var certificado = new SeccionCompuesta("Certificado");
        certificado.Agregar(new ElementoSimple("Nombre del Receptor"));
        return certificado;
    }
}
public class MenuPersonalizador
{
    public IComponenteDocumento PersonalizarCV(IComponenteDocumento doc)
    {
        bool personalizando = true;
        var opcionesElegidas = new List<string>();

        while (personalizando)
        {
            Console.Clear();
            Console.WriteLine("Personalizando CV");
            Console.WriteLine("-------------------------------");
            Console.WriteLine("Opciones de personalización:");
            Console.WriteLine($"1. Añadir Título {(opcionesElegidas.Contains("1") ?
"(Elegido)" : "")}");
            Console.WriteLine($"2. Añadir Foto {(opcionesElegidas.Contains("2") ?
"(Elegido)" : "")}");
            Console.WriteLine($"3. Añadir Resumen {(opcionesElegidas.Contains("3") ?
"(Elegido)" : "")}");
            Console.WriteLine($"4. Añadir Pie de Página
{(opcionesElegidas.Contains("4") ? "(Elegido)" : "")}");
            Console.WriteLine("5. Finalizar");
            string opcion = Console.ReadLine();

            if (opcionesElegidas.Contains(opcion))
            {
                Console.WriteLine("Opción ya seleccionada. Presiona Enter.");
                Console.ReadKey(true);
            }
            else
            {
                switch (opcion)
                {
                    case "1": doc = new TituloDecorator(doc); opcionesElegidas.Add("1");
```

```
                    break;

                case "2": doc = new FotoDecorator(doc); opcionesElegidas.Add("2");
                    break;

                case "3": doc = new ResumenDecorator(doc);
opcionesElegidas.Add("3");
                    break;

                case "4": doc = new PieDePaginaDecorator(doc);
opcionesElegidas.Add("4");
                    break;

                case "5": personalizando = false;
                    break;

                default: Console.WriteLine("Opción no válida.");
Console.ReadKey(true);
                    break;
            }
        }
    }
    return doc;
  }
  public IComponenteDocumento PersonalizarFactura(IComponenteDocumento
doc)
  {
    bool personalizando = true;
    var opcionesElegidas = new List<string>();

    while (personalizando)
    {
      Console.Clear();
      Console.WriteLine("Personalizando Factura");
      Console.WriteLine("------------------------------");
      Console.WriteLine("Opciones de personalización:");
      Console.WriteLine($"1. Añadir Datos Fiscales
{(opcionesElegidas.Contains("1") ? "(Elegido)" : "")}");
      Console.WriteLine($"2. Añadir Información de Impuestos
{(opcionesElegidas.Contains("2") ? "(Elegido)" : "")}");
      Console.WriteLine($"3. Añadir Sello Digital {(opcionesElegidas.Contains("3")
? "(Elegido)" : "")}");
```

```
            Console.WriteLine($"4. Añadir Pie de Página
{(opcionesElegidas.Contains("4") ? "(Elegido)" : "")}");
            Console.WriteLine("5. Finalizar");
            string opcion = Console.ReadLine();

            if (opcionesElegidas.Contains(opcion))
            {
                Console.WriteLine("Opción ya seleccionada. Presiona Enter.");
                Console.ReadKey(true);
            }
            else
            {
                switch (opcion)
                {
                    case "1": doc = new DatosFiscalesDecorator(doc);
opcionesElegidas.Add("1"); break;
                    case "2": doc = new DesgloseImpuestosDecorator(doc);
opcionesElegidas.Add("2"); break;
                    case "3": doc = new SelloDigitalDecorator(doc);
opcionesElegidas.Add("3"); break;
                    case "4": doc = new PieDePaginaDecorator(doc);
opcionesElegidas.Add("4"); break;
                    case "5": personalizando = false; break;
                    default: Console.WriteLine("Opción no válida.");
Console.ReadKey(true); break;
                }
            }
        }
        return doc;
    }

    public IComponenteDocumento PersonalizarFolleto(IComponenteDocumento
doc)
    {
        bool personalizando = true;
        var opcionesElegidas = new List<string>();

        while (personalizando)
        {
            Console.Clear();
            Console.WriteLine("Personalizando Folleto");
            Console.WriteLine("------------------------------");
            Console.WriteLine("Opciones de personalización:");
```

```
        Console.WriteLine($"1. Añadir Título {(opcionesElegidas.Contains("1") ?
"(Elegido)" : "")}");
        Console.WriteLine($"2. Añadir Gráficos de fondo
{(opcionesElegidas.Contains("2") ? "(Elegido)" : "")}");
        Console.WriteLine($"3. Añadir Diseño a Doble Cara
{(opcionesElegidas.Contains("3") ? "(Elegido)" : "")}");
        Console.WriteLine($"4. Añadir Pie de Página
{(opcionesElegidas.Contains("4") ? "(Elegido)" : "")}");
        Console.WriteLine("5. Finalizar");
        string opcion = Console.ReadLine();

        if (opcionesElegidas.Contains(opcion))
        {
            Console.WriteLine("Opción ya seleccionada. Presiona Enter.");
            Console.ReadKey(true);
        }
        else
        {
            switch (opcion)
            {
                case "1": doc = new TituloDecorator(doc); opcionesElegidas.Add("1");
break;
                case "2": doc = new GraficosDecorator(doc);
opcionesElegidas.Add("2"); break;
                case "3": doc = new DobleCaraDecorator(doc);
opcionesElegidas.Add("3"); break;
                case "4": doc = new PieDePaginaDecorator(doc);
opcionesElegidas.Add("4"); break;
                case "5": personalizando = false; break;
                default: Console.WriteLine("Opción no válida.");
Console.ReadKey(true); break;
            }
        }
    }
    return doc;
  }


  public IComponenteDocumento PersonalizarCertificado(IComponenteDocumento
doc)
  {
    bool personalizando = true;
    var opcionesElegidas = new List<string>();
```

```
        while (personalizando)
        {
            Console.Clear();
            Console.WriteLine("Personalizando Certificado");
            Console.WriteLine("-------------------------------");
            Console.WriteLine("Opciones de personalización:");
            Console.WriteLine($"1. Añadir Título {(opcionesElegidas.Contains("1") ?
"(Elegido)" : "")}");
            Console.WriteLine($"2. Añadir Borde Elegante
{(opcionesElegidas.Contains("2") ? "(Elegido)" : "")}");
            Console.WriteLine($"3. Añadir Espacio para Firma
{(opcionesElegidas.Contains("3") ? "(Elegido)" : "")}");
            Console.WriteLine($"4. Añadir Pie de Página
{(opcionesElegidas.Contains("4") ? "(Elegido)" : "")}");
            Console.WriteLine("5. Finalizar");
            string opcion = Console.ReadLine();

            if (opcionesElegidas.Contains(opcion))
            {
                Console.WriteLine("Opción ya seleccionada. Presiona Enter.");
                Console.ReadKey(true);
            }
            else
            {
                switch (opcion)
                {
                    case "1": doc = new TituloDecorator(doc); opcionesElegidas.Add("1");
break;
                    case "2": doc = new BordeEleganteDecorator(doc);
opcionesElegidas.Add("2"); break;
                    case "3": doc = new FirmaAutorizadaDecorator(doc);
opcionesElegidas.Add("3"); break;
                    case "4": doc = new PieDePaginaDecorator(doc);
opcionesElegidas.Add("4"); break;
                    case "5": personalizando = false; break;
                    default: Console.WriteLine("Opción no válida.");
Console.ReadKey(true); break;
                }
            }
        }
        return doc;
    }
```

```
}
```

PatronComposite.cs

```
namespace PlantillaDecoCompo
{
    public interface IComponenteDocumento
    {

        string GenerarDescripcion();
    }

    public class ElementoSimple : IComponenteDocumento
    {
        public ElementoSimple(string nombre) { }

        public string GenerarDescripcion() => string.Empty;
    }

    public class SeccionCompuesta : IComponenteDocumento
    {
        private List<IComponenteDocumento> _hijos = new
List<IComponenteDocumento>();

        public SeccionCompuesta(string titulo) { }

        public void Agregar(IComponenteDocumento componente)
        {
            _hijos.Add(componente);
        }


        public string GenerarDescripcion()
        {
            string desc = string.Empty;
            foreach (var hijo in _hijos)
            {
                desc += hijo.GenerarDescripcion();
            }
            return desc;
        }
    }
}
```

PatronDecorador

```
namespace PlantillaDecoCompo
{
   public abstract class DocumentoDecorador : IComponenteDocumento
   {
      protected IComponenteDocumento _componenteEnvuelto;

      public DocumentoDecorador(IComponenteDocumento componente)
      {
         _componenteEnvuelto = componente;
      }


      public virtual string GenerarDescripcion() =>
_componenteEnvuelto.GenerarDescripcion();
   }

   public class TituloDecorator : DocumentoDecorador
   {
      public TituloDecorator(IComponenteDocumento c) : base(c) { }
      public override string GenerarDescripcion() =>
         _componenteEnvuelto.GenerarDescripcion() + "- Título Principal\n";
   }

   public class PieDePaginaDecorator : DocumentoDecorador
   {
      public PieDePaginaDecorator(IComponenteDocumento c) : base(c) { }
      public override string GenerarDescripcion() =>
         _componenteEnvuelto.GenerarDescripcion() + "- Pie de Página (info
contacto)\n";
   }

   public class FotoDecorator : DocumentoDecorador
   {
      public FotoDecorator(IComponenteDocumento c) : base(c) { }
      public override string GenerarDescripcion() =>
         _componenteEnvuelto.GenerarDescripcion() + "- Espacio para Foto de
Perfil\n";
   }
   public class ResumenDecorator : DocumentoDecorador
   {
      public ResumenDecorator(IComponenteDocumento c) : base(c) { }
      public override string GenerarDescripcion() =>
```

```
        _componenteEnvuelto.GenerarDescripcion() + "- Sección de Resumen
Profesional\n";
    }

    public class DatosFiscalesDecorator : DocumentoDecorador
    {
        public DatosFiscalesDecorator(IComponenteDocumento c) : base(c) { }
        public override string GenerarDescripcion() =>
            _componenteEnvuelto.GenerarDescripcion() + "- Encabezado\n";
    }
    public class DesgloseImpuestosDecorator : DocumentoDecorador
    {
        public DesgloseImpuestosDecorator(IComponenteDocumento c) : base(c) { }
        public override string GenerarDescripcion() =>
            _componenteEnvuelto.GenerarDescripcion() + "- Información de
Impuestos\n";
    }

    public class SelloDigitalDecorator : DocumentoDecorador
    {
        public SelloDigitalDecorator(IComponenteDocumento c) : base(c) { }
        public override string GenerarDescripcion() =>
            _componenteEnvuelto.GenerarDescripcion() + "- Sello Digital\n";
    }

    public class GraficosDecorator : DocumentoDecorador
    {
        public GraficosDecorator(IComponenteDocumento c) : base(c) { }
        public override string GenerarDescripcion() =>
            _componenteEnvuelto.GenerarDescripcion() + "- Gráficos e imágenes de
fondo\n";
    }
    public class DobleCaraDecorator : DocumentoDecorador
    {
        public DobleCaraDecorator(IComponenteDocumento c) : base(c) { }
        public override string GenerarDescripcion() =>
            _componenteEnvuelto.GenerarDescripcion() + "- Diseño a Doble Cara\n";
    }

    public class BordeEleganteDecorator : DocumentoDecorador
    {
        public BordeEleganteDecorator(IComponenteDocumento c) : base(c) { }
        public override string GenerarDescripcion() =>
```

```
        _componenteEnvuelto.GenerarDescripcion() + "- Borde elegante y Sello de
Agua\n";
    }
    public class FirmaAutorizadaDecorator : DocumentoDecorador
    {
        public FirmaAutorizadaDecorator(IComponenteDocumento c) : base(c) { }
        public override string GenerarDescripcion() =>
            _componenteEnvuelto.GenerarDescripcion() + "- Espacio para Firma
Autorizada\n";
    }
}
```

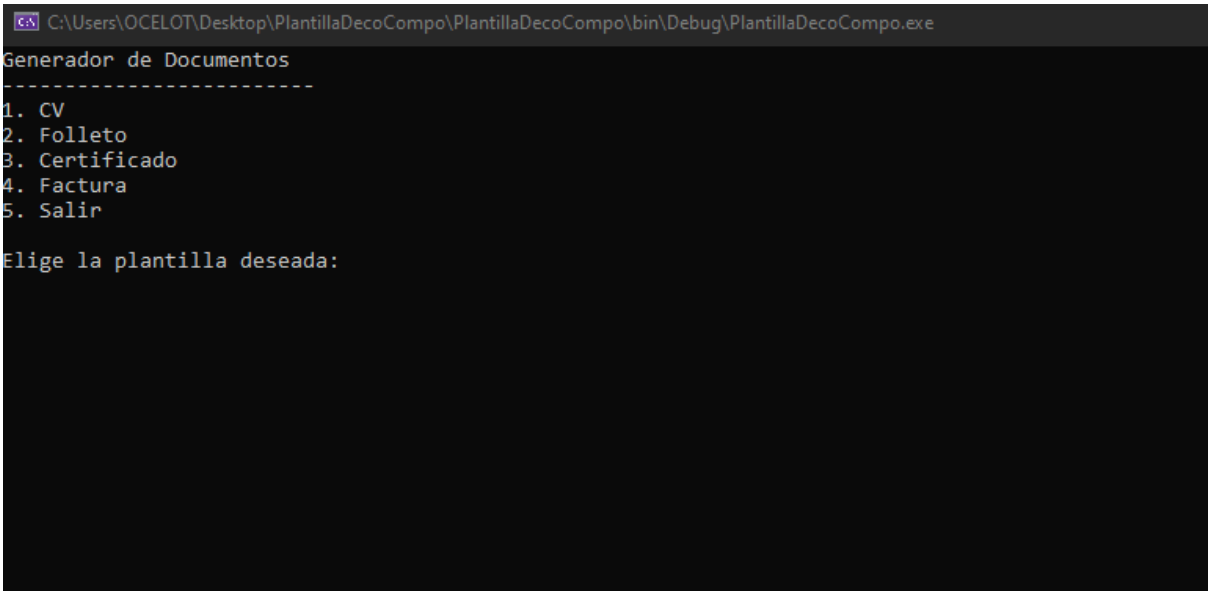
Program.cs

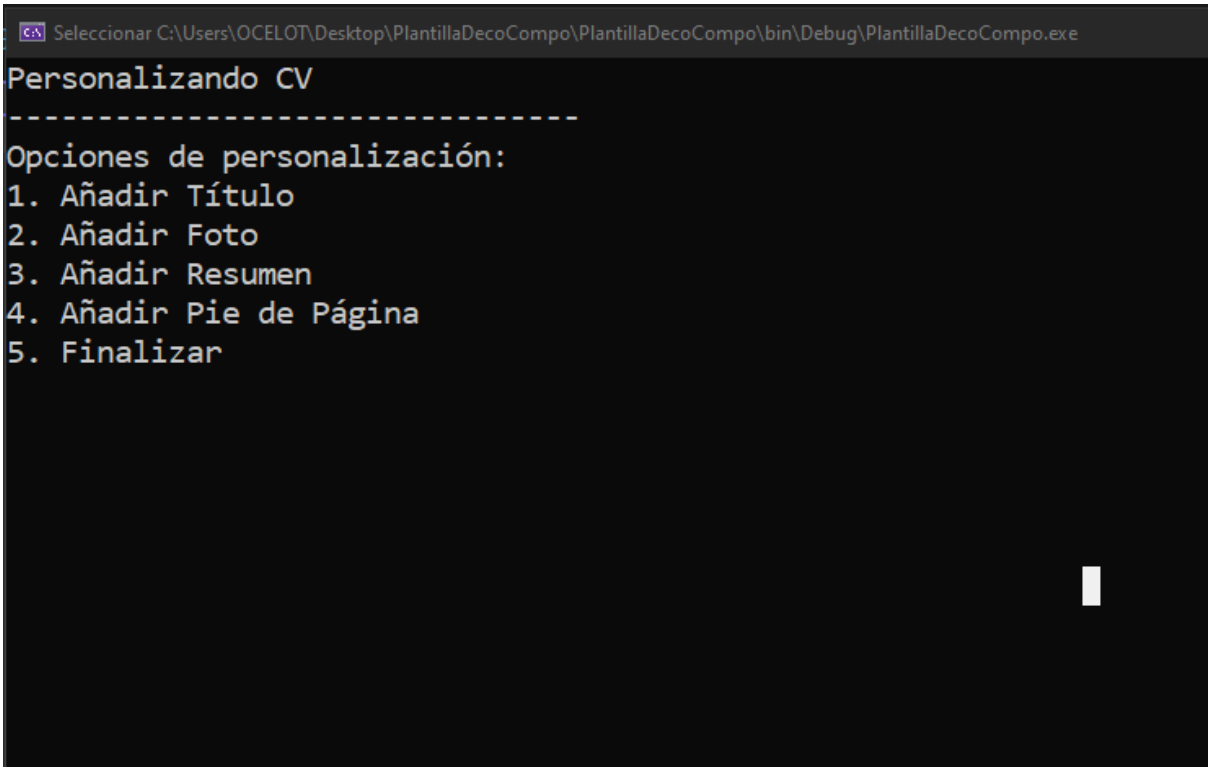
```
namespace PlantillaDecoCompo
{
    class Program
    {
        static void Main(string[] args)
        {
            while (true)
            {

                var generador = new GeneradorPlantillas();
                var menu = new MenuPersonalizador();

                IComponenteDocumento documentoBase = null;
                string nombrePlantilla = "";

                bool plantillaValida = false;
                while (!plantillaValida)
                {
                    Console.Clear();
                    Console.WriteLine("Generador de Documentos");
                    Console.WriteLine("------------------------");
                    Console.WriteLine("1. CV");
                    Console.WriteLine("2. Folleto");
                    Console.WriteLine("3. Certificado");
                    Console.WriteLine("4. Factura");
                    Console.WriteLine("5. Salir");
                    Console.Write("\nElige la plantilla deseada: ");
                    string tipo = Console.ReadLine();

                    switch (tipo)
```

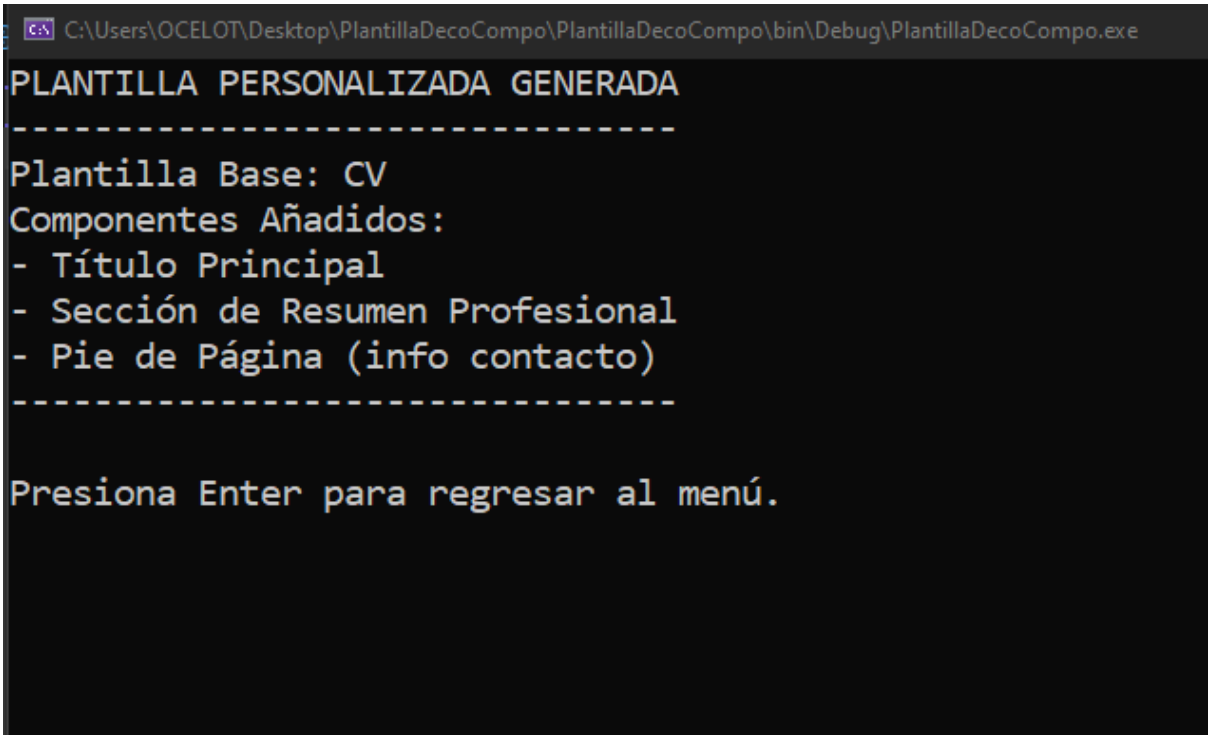
```
    {
        case "1":
            documentoBase = generador.CrearBaseCV();
            documentoBase = menu.PersonalizarCV(documentoBase);
            nombrePlantilla = "CV";
            plantillaValida = true;
            break;
        case "2":
            documentoBase = generador.CrearBaseFolleto();
            documentoBase = menu.PersonalizarFolleto(documentoBase);
            nombrePlantilla = "Folleto";
            plantillaValida = true;
            break;
        case "3":
            documentoBase = generador.CrearBaseCertificado();
            documentoBase = menu.PersonalizarCertificado(documentoBase);
            nombrePlantilla = "Certificado";
            plantillaValida = true;
            break;
        case "4":
            documentoBase = generador.CrearBaseFactura();
            documentoBase = menu.PersonalizarFactura(documentoBase);
            nombrePlantilla = "Factura";
            plantillaValida = true;
            break;
        case "5":
            return;
        default:
            Console.WriteLine("Opción no válida."); Console.ReadKey(true);
            break;
    }
}

Console.Clear();
Console.WriteLine("PLANTILLA PERSONALIZADA GENERADA");
Console.WriteLine("-------------------------------");
Console.WriteLine($"Plantilla Base: {nombrePlantilla}");
Console.WriteLine("Componentes Añadidos:");

string componentesAñadidos = documentoBase.GenerarDescripcion();

if (string.IsNullOrEmpty(componentesAñadidos))
{
```

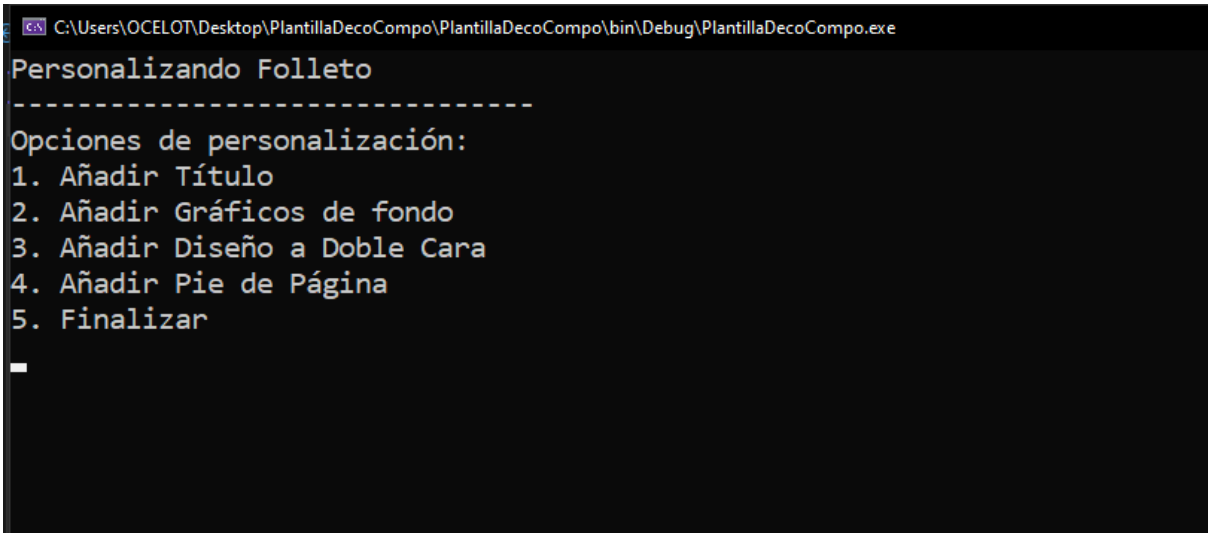
```
                Console.WriteLine("(Ninguno)");
            }
            else
            {
                Console.Write(componentesAñadidos);
            }

            Console.WriteLine("------------------------------");


            Console.WriteLine("\nPresiona Enter para regresar al menú.");
            Console.ReadLine();
        }
    }
  }
}
```

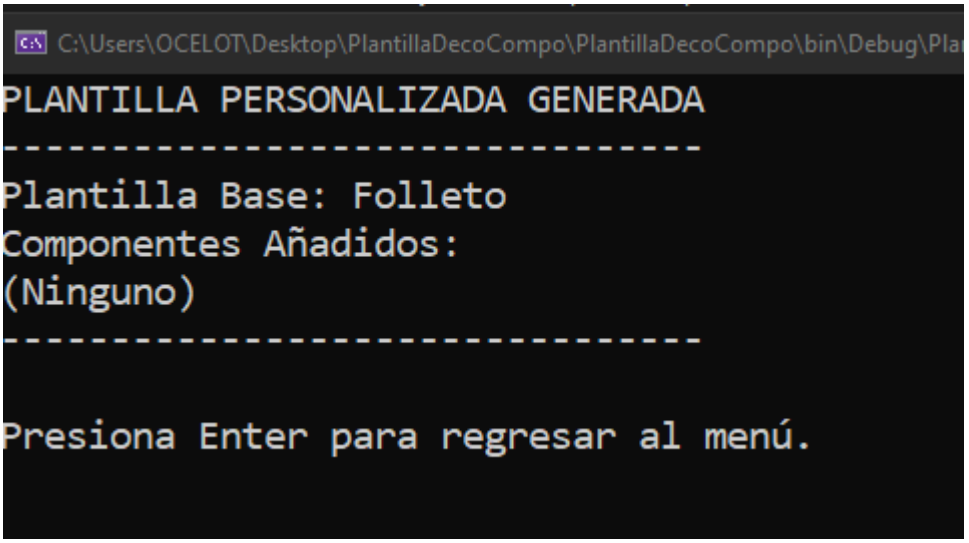
Ejecución

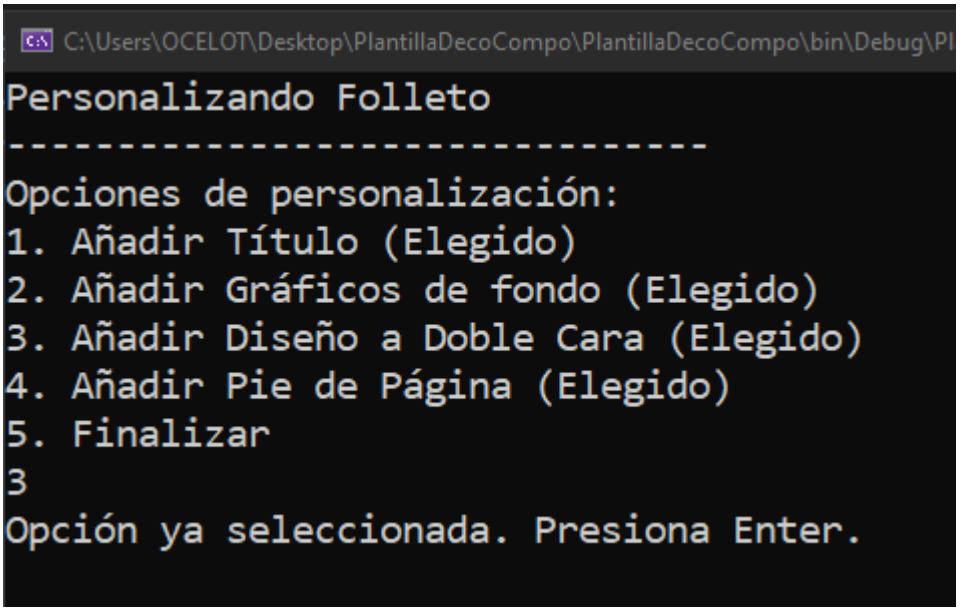


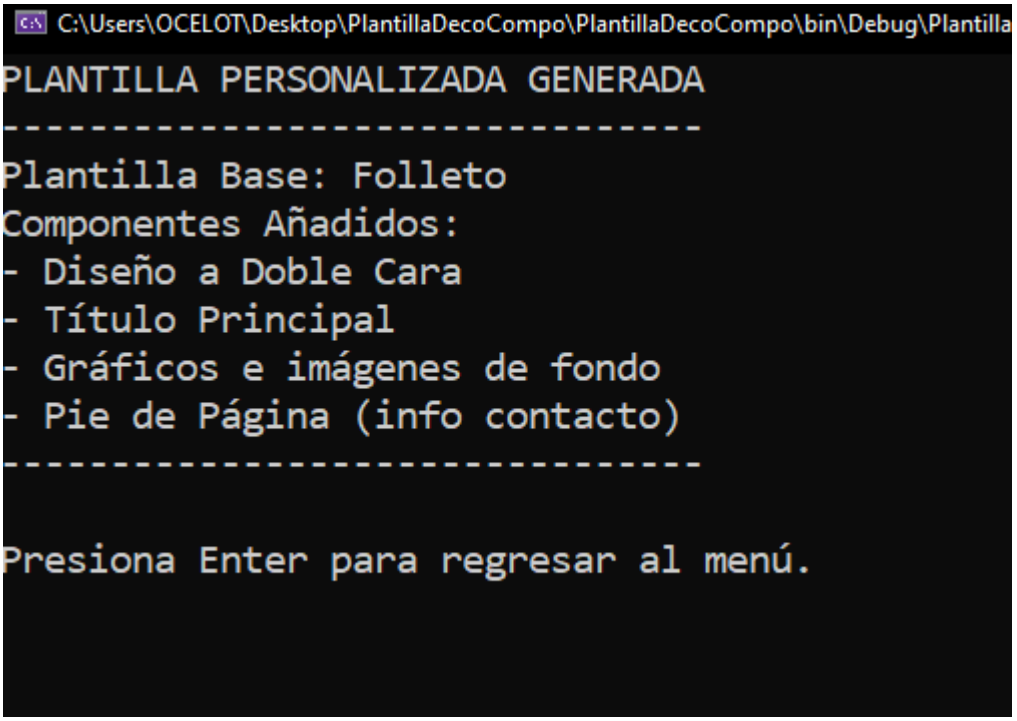

Seleccionar C:\Users\OCELOT\Desktop\PlantillaDecoCompo\PlantillaDecoCompo\bin\Debug\PlantillaDecoCompo.exe
Personalizando CV
Opciones de personalización:
1. Añadir Título
2. Añadir Foto
3. Añadir Resumen
4. Añadir Pie de Página
5. Finalizar


C:\Users\OCELOT\Desktop\PlantillaDecoCompo\PlantillaDecoCompo\bin\Debug\PlantillaDecoCompo.exe
PLANTILLA PERSONALIZADA GENERADA
Plantilla Base: CV
Componentes Añadidos:
- Título Principal
- Sección de Resumen Profesional
- Pie de Página (info contacto)
Presiona Enter para regresar al menú.


C:\Users\OCELOT\Desktop\PlantillaDecoCompo\PlantillaDecoCompo\bin\Debug\PlantillaDecoCompo.exe
Personalizando Folleto
Opciones de personalización:
1. Añadir Título
2. Añadir Gráficos de fondo
3. Añadir Diseño a Doble Cara
4. Añadir Pie de Página
5. Finalizar


C:\Users\OCELOT\Desktop\PlantillaDecoCompo\PlantillaDecoCompo\bin\Debug\Pla
PLANTILLA PERSONALIZADA GENERADA
Plantilla Base: Folleto
Componentes Añadidos:
(Ninguno)
Presiona Enter para regresar al menú.


C:\Users\OCELOT\Desktop\PlantillaDecoCompo\PlantillaDecoCompo\bin\Debug\Pl
Personalizando Folleto
Opciones de personalización:
1. Añadir Título (Elegido)
2. Añadir Gráficos de fondo (Elegido)
3. Añadir Diseño a Doble Cara (Elegido)
4. Añadir Pie de Página (Elegido)
5. Finalizar
3
Opción ya seleccionada. Presiona Enter.



Conclusión

En conclusión, la combinación de ambos patrones en este proyecto fue muy útil, ya que gracias al composite, se pudo tener una base sólida para todas las plantillas, y el patrón decorador permitió ir añadiendo las modificaciones requeridas para cada plantilla de una manera sencilla.