# 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 4 y 5

**UNIDAD A EVALUAR**

UNIDAD 4 y 5

**NOMBRE DEL ESTUDIANTE**

HUERTA RIVAS OLAF ALEXANDRO - 21211966

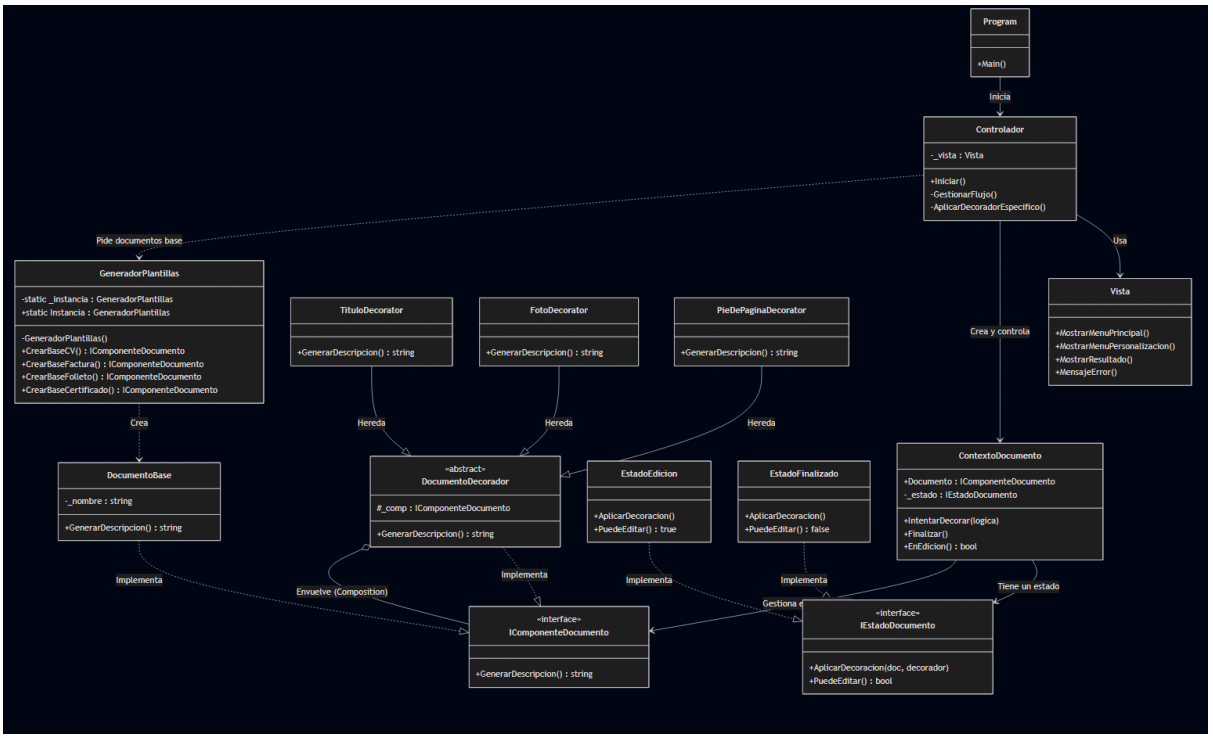
**MAESTRO**

MARIBEL GUERRERO LUIS

**FECHA DE ENTREGA**

09 DE DICIEMBRE DE 2025

## Diagrama UML




## Código

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


namespace ProyectoFinalPatrones
{
    public class GeneradorPlantillas
    {
        private static GeneradorPlantillas _instancia;

        private GeneradorPlantillas() { }

        public static GeneradorPlantillas Instancia
```

```
        {
            get
            {
                if (_instancia == null)
                    _instancia = new GeneradorPlantillas();
                return _instancia;
            }
        }
        public IComponenteDocumento CrearBaseCV() => new DocumentoBase("CV (Curriculum Vitae)");
        public IComponenteDocumento CrearBaseFactura() => new DocumentoBase("Factura Fiscal");
        public IComponenteDocumento CrearBaseFolleto() => new DocumentoBase("Folleto Informativo");
        public IComponenteDocumento CrearBaseCertificado() => new DocumentoBase("Certificado de Reconocimiento");
    }

    public interface IComponenteDocumento
    {
        string GenerarDescripcion();
    }
    public class DocumentoBase : IComponenteDocumento
    {
        private string _nombre;
        public DocumentoBase(string nombre) { _nombre = nombre; }
```

```
        // Devuelve el título base del documento
        public string GenerarDescripcion() => $"--- FORMATO BASE: {_nombre} ---\n";
    }
    public abstract class DocumentoDecorador : IComponenteDocumento
    {
        protected IComponenteDocumento _comp;
        public DocumentoDecorador(IComponenteDocumento c) { _comp = c; }
        public virtual string GenerarDescripcion() => _comp.GenerarDescripcion();
    }

    public class TituloDecorator : DocumentoDecorador {
        public TituloDecorator(IComponenteDocumento c) : base(c) { }
        public override string GenerarDescripcion() => _comp.GenerarDescripcion() +
"- Título Principal\n";
    }
    public class PieDePaginaDecorator : DocumentoDecorador {
        public PieDePaginaDecorator(IComponenteDocumento c) : base(c) { }
        public override string GenerarDescripcion() => _comp.GenerarDescripcion() +
"- Pie de Página (info contacto)\n";
    }
    public class FotoDecorator : DocumentoDecorador {
        public FotoDecorator(IComponenteDocumento c) : base(c) { }
        public override string GenerarDescripcion() => _comp.GenerarDescripcion() +
"- Espacio para Foto de Perfil\n";
    }
    public class ResumenDecorator : DocumentoDecorador {
```

```
        public ResumenDecorator(IComponenteDocumento c) : base(c) { }

        public override string GenerarDescripcion() => _comp.GenerarDescripcion() +
"- Sección de Resumen Profesional\n";

    }

    public class DatosFiscalesDecorator : DocumentoDecorador {

        public DatosFiscalesDecorator(IComponenteDocumento c) : base(c) { }

        public override string GenerarDescripcion() => _comp.GenerarDescripcion() +
"- Encabezado (Datos Fiscales y Logo)\n";

    }

    public class DesgloseImpuestosDecorator : DocumentoDecorador {

        public DesgloseImpuestosDecorator(IComponenteDocumento c) : base(c) { }

        public override string GenerarDescripcion() => _comp.GenerarDescripcion() +
"- Desglose de Impuestos (IVA, Total)\n";

    }

    public class SelloDigitalDecorator : DocumentoDecorador {

        public SelloDigitalDecorator(IComponenteDocumento c) : base(c) { }

        public override string GenerarDescripcion() => _comp.GenerarDescripcion() +
"- Sello Digital (CFDI)\n";

    }

    public class GraficosDecorator : DocumentoDecorador {

        public GraficosDecorator(IComponenteDocumento c) : base(c) { }

        public override string GenerarDescripcion() => _comp.GenerarDescripcion() +
"- Gráficos e Imágenes de fondo\n";

    }

    public class DobleCaraDecorator : DocumentoDecorador {

        public DobleCaraDecorator(IComponenteDocumento c) : base(c) { }

        public override string GenerarDescripcion() => _comp.GenerarDescripcion() +
"- Diseño a Doble Cara (Tríptico)\n";
```

```
    }

    public class BordeEleganteDecorator : DocumentoDecorador {

        public BordeEleganteDecorator(IComponenteDocumento c) : base(c) { }

        public override string GenerarDescripcion() => _comp.GenerarDescripcion() +
"- Borde Elegante y Sello de Agua\n";

    }

    public class FirmaAutorizadaDecorator : DocumentoDecorador {

        public FirmaAutorizadaDecorator(IComponenteDocumento c) : base(c) { }

        public override string GenerarDescripcion() => _comp.GenerarDescripcion() +
"- Espacio para Firma Autorizada\n";

    }

    public interface IEstadoDocumento

    {

        IComponenteDocumento AplicarDecoracion(IComponenteDocumento doc,
Func<IComponenteDocumento, IComponenteDocumento> decorador);

        bool PuedeEditar();

    }



    public class EstadoEdicion : IEstadoDocumento

    {

        public bool PuedeEditar() => true;

        public IComponenteDocumento AplicarDecoracion(IComponenteDocumento
doc, Func<IComponenteDocumento, IComponenteDocumento> decorador)

        {

            return decorador(doc);

        }

    }
```

```
    public class EstadoFinalizado : IEstadoDocumento
    {
        public bool PuedeEditar() => false;
        public IComponenteDocumento AplicarDecoracion(IComponenteDocumento
doc, Func<IComponenteDocumento, IComponenteDocumento> decorador)
        {
            Console.WriteLine("BLOQUEADO: El documento está finalizado. No se
pueden hacer cambios.");
            return doc;
        }
    }

    public class ContextoDocumento
    {
        public IComponenteDocumento Documento { get; private set; }
        private IEstadoDocumento _estado;

        public ContextoDocumento(IComponenteDocumento docInicial)
        {
            Documento = docInicial;
            _estado = new EstadoEdicion();
        }

        public void IntentarDecorar(Func<IComponenteDocumento,
IComponenteDocumento> logicaDecorador)
```

```
    {
        Documento = _estado.AplicarDecoracion(Documento, logicaDecorador);
    }

    public void Finalizar()
    {
        _estado = new EstadoFinalizado();
    }

    public bool EnEdicion() => _estado.PuedeEditar();
}
public class Vista
{
    public void MostrarMenuPrincipal()
    {
        Console.Clear();
        Console.WriteLine("=== SISTEMA DE PLANTILLAS ===");
        Console.WriteLine("=== (MVC + State + Singleton + Decorator) ===");
        Console.WriteLine("1. Crear CV");
        Console.WriteLine("2. Crear Folleto");
        Console.WriteLine("3. Crear Certificado");
        Console.WriteLine("4. Crear Factura");
        Console.WriteLine("5. Salir");
        Console.Write("Seleccione una opción: ");
```

```
    }

    public void MostrarMenuPersonalizacion(string titulo, List<string> opciones,
List<string> elegidas)
    {
        Console.Clear();
        Console.WriteLine($"--- Personalizando {titulo} ---");
        for (int i = 0; i < opciones.Count; i++)
        {
            string estado = elegidas.Contains((i + 1).ToString()) ? "(Elegido)" : "";
            Console.WriteLine($"{i + 1}. {opciones[i]} {estado}");
        }
        Console.WriteLine($"{opciones.Count + 1}. Finalizar Documento");
        Console.Write("Seleccione opción: ");
    }

    public void MostrarResultado(string nombre, string contenido)
    {
        Console.Clear();
        Console.WriteLine("=== DOCUMENTO GENERADO ===");
        Console.WriteLine($"Tipo: {nombre}");
        Console.WriteLine("--------------------------");
        if (string.IsNullOrEmpty(contenido)) Console.WriteLine("(Sin componentes
extra)");
        else Console.Write(contenido);
```

```
        Console.WriteLine("-------------------------");
        Console.WriteLine("Presione Enter para volver al menú...");
        Console.ReadLine();
    }

    public void MensajeError(string msg)
    {
        Console.WriteLine(msg);
        Console.ReadKey();
    }
}
public class Controlador
{
    private Vista _vista;

    public Controlador()
    {
        _vista = new Vista();
    }

    public void Iniciar()
    {
        bool salir = false;
        while (!salir)
```

```
        {
            _vista.MostrarMenuPrincipal();
            string op = Console.ReadLine();

            switch (op)
            {
                case "1":
                    GestionarFlujo("CV", GeneradorPlantillas.Instancia.CrearBaseCV(),
                        new List<string> { "Añadir Título", "Añadir Foto", "Añadir
Resumen", "Añadir Pie de Página" });
                    break;
                case "2":
                    GestionarFlujo("Folleto",
GeneradorPlantillas.Instancia.CrearBaseFolleto(),
                        new List<string> { "Añadir Título", "Añadir Gráficos Fondo", "Añadir
Doble Cara", "Añadir Pie de Página" });
                    break;
                case "3":
                    GestionarFlujo("Certificado",
GeneradorPlantillas.Instancia.CrearBaseCertificado(),
                        new List<string> { "Añadir Título", "Añadir Borde Elegante", "Añadir
Espacio Firma", "Añadir Pie de Página" });
                    break;
                case "4":
                    GestionarFlujo("Factura",
GeneradorPlantillas.Instancia.CrearBaseFactura(),
                        new List<string> { "Añadir Datos Fiscales", "Añadir Desglose
Impuestos", "Añadir Sello Digital", "Añadir Pie de Página" });
```

```
                    break;
                case "5":
                    salir = true;
                    break;
                default:
                    _vista.MensajeError("Opción no válida.");
                    break;
            }
        }
    }

    private void GestionarFlujo(string tipo, IComponenteDocumento docBase,
List<string> textosOpciones)
    {
        ContextoDocumento contexto = new ContextoDocumento(docBase);
        List<string> elegidas = new List<string>();

        while (contexto.EnEdicion())
        {
            _vista.MostrarMenuPersonalizacion(tipo, textosOpciones, elegidas);
            string entrada = Console.ReadLine();
            int opcionNum;

            if (int.TryParse(entrada, out opcionNum) && opcionNum ==
textosOpciones.Count + 1)
```

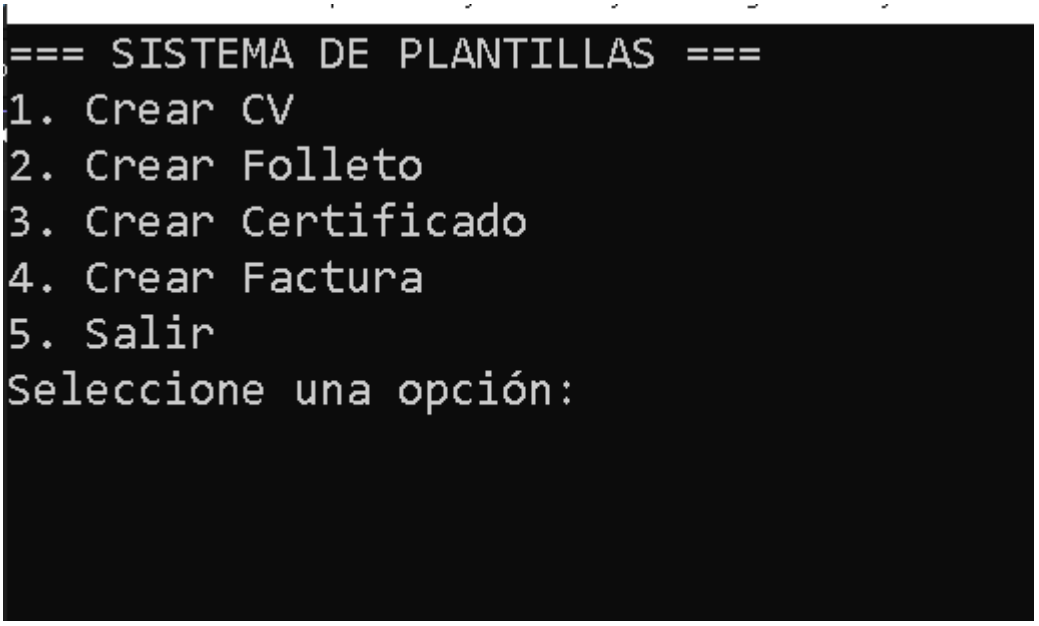
```
            {
                contexto.Finalizar();
            }
            else if (elegidas.Contains(entrada))
            {
                _vista.MensajeError("Opción ya seleccionada. Presiona Enter.");
            }
            else
            {
                bool aplicado = AplicarDecoradorEspecifico(contexto, tipo, entrada);

                if (aplicado)
                    elegidas.Add(entrada);
                else
                    _vista.MensajeError("Opción no válida.");
            }
        }
        _vista.MostrarResultado(tipo, contexto.Documento.GenerarDescripcion());
    }

    private bool AplicarDecoradorEspecifico(ContextoDocumento ctx, string
tipoDoc, string opcion)
    {
        switch (tipoDoc)
        {
```

```
            case "CV":
                switch (opcion)
                {
                    case "1": ctx.IntentarDecorar(d => new TituloDecorator(d)); return
true;
                    case "2": ctx.IntentarDecorar(d => new FotoDecorator(d)); return true;
                    case "3": ctx.IntentarDecorar(d => new ResumenDecorator(d)); return
true;
                    case "4": ctx.IntentarDecorar(d => new PieDePaginaDecorator(d));
return true;
                }
                break;
            case "Folleto":
                switch (opcion)
                {
                    case "1": ctx.IntentarDecorar(d => new TituloDecorator(d)); return
true;
                    case "2": ctx.IntentarDecorar(d => new GraficosDecorator(d)); return
true;
                    case "3": ctx.IntentarDecorar(d => new DobleCaraDecorator(d));
return true;
                    case "4": ctx.IntentarDecorar(d => new PieDePaginaDecorator(d));
return true;
                }
                break;
            case "Certificado":
                switch (opcion)
                {
```

```
                    case "1": ctx.IntentarDecorar(d => new TituloDecorator(d)); return
true;

                    case "2": ctx.IntentarDecorar(d => new BordeEleganteDecorator(d));
return true;

                    case "3": ctx.IntentarDecorar(d => new
FirmaAutorizadaDecorator(d)); return true;

                    case "4": ctx.IntentarDecorar(d => new PieDePaginaDecorator(d));
return true;

                }

                break;

            case "Factura":

                switch (opcion)

                {

                    case "1": ctx.IntentarDecorar(d => new DatosFiscalesDecorator(d));
return true;

                    case "2": ctx.IntentarDecorar(d => new
DesgloseImpuestosDecorator(d)); return true;

                    case "3": ctx.IntentarDecorar(d => new SelloDigitalDecorator(d));
return true;

                    case "4": ctx.IntentarDecorar(d => new PieDePaginaDecorator(d));
return true;

                }

                break;

        }

        return false;

    }

}


class Program
```

```
    {
        static void Main(string[] args)
        {
            new Controlador().Iniciar();
        }
    }
}
```

**Funcionamiento**

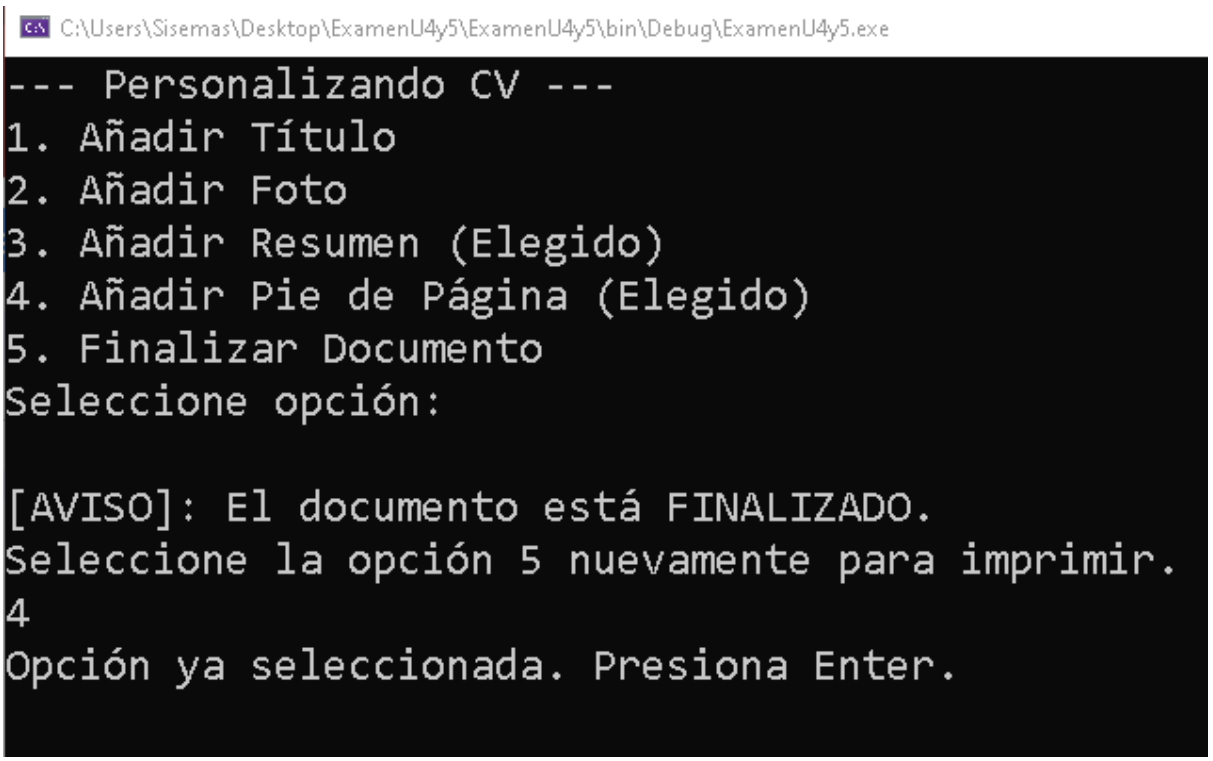


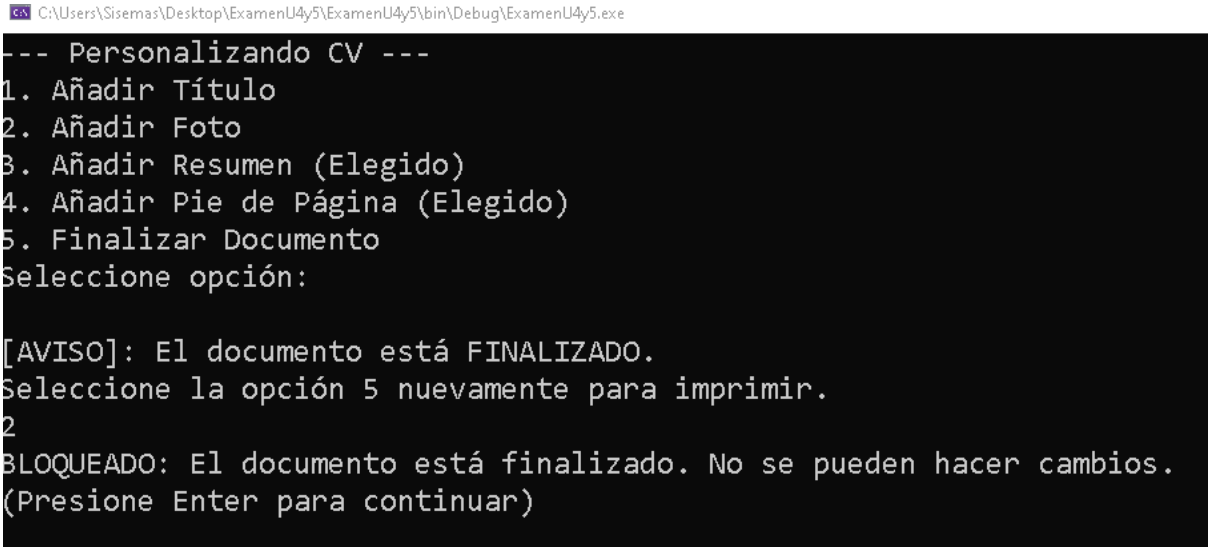
C:\Users\Sisemas\Desktop\ExamenU4y5\ExamenU4y5\bin\Debug\ExamenU4y5.exe

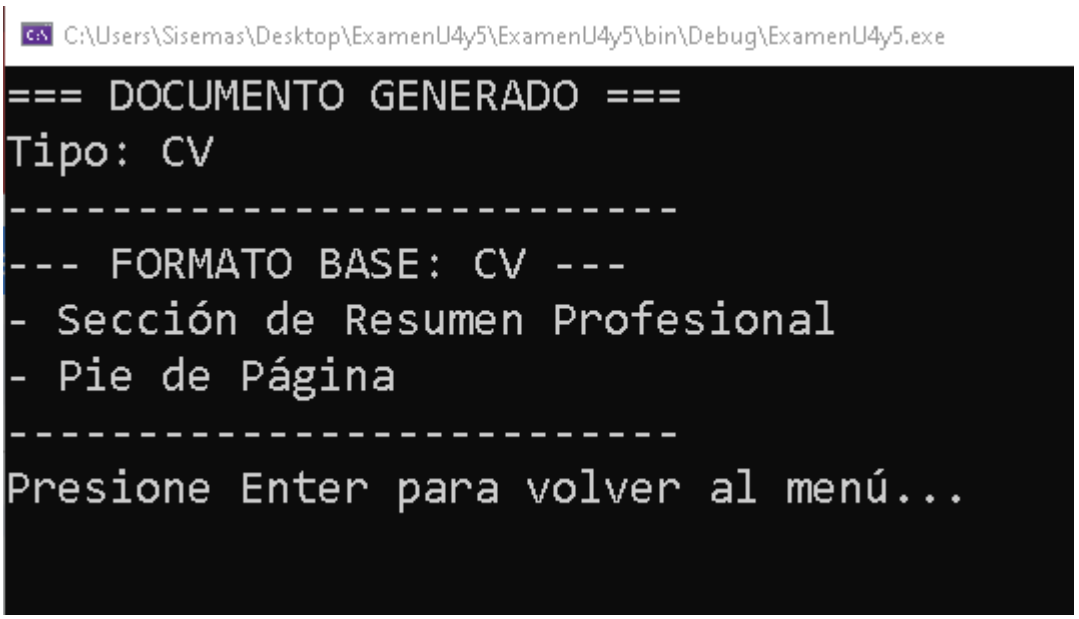
```
--- Personalizando CV ---
1. Añadir Título
2. Añadir Foto
3. Añadir Resumen
4. Añadir Pie de Página
5. Finalizar Documento
Seleccione opción:
```

C:\Users\Sisemas\Desktop\ExamenU4y5\ExamenU4y5\bin\Debug\ExamenU4y5.exe

```
--- Personalizando CV ---
1. Añadir Título
2. Añadir Foto
3. Añadir Resumen (Elegido)
4. Añadir Pie de Página (Elegido)
5. Finalizar Documento
Seleccione opción:

[AVISO]: El documento está FINALIZADO.
Seleccione la opción 5 nuevamente para imprimir.
```


C:\Users\Sisemas\Desktop\ExamenU4y5\ExamenU4y5\bin\Debug\ExamenU4y5.exe
--- Personalizando CV ---
1. Añadir Título
2. Añadir Foto
3. Añadir Resumen (Elegido)
4. Añadir Pie de Página (Elegido)
5. Finalizar Documento
Seleccione opción:
[AVISO]: El documento está FINALIZADO.
Seleccione la opción 5 nuevamente para imprimir.
4
Opción ya seleccionada. Presiona Enter.


C:\Users\Sisemas\Desktop\ExamenU4y5\ExamenU4y5\bin\Debug\ExamenU4y5.exe
--- Personalizando CV ---
1. Añadir Título
2. Añadir Foto
3. Añadir Resumen (Elegido)
4. Añadir Pie de Página (Elegido)
5. Finalizar Documento
Seleccione opción:
[AVISO]: El documento está FINALIZADO.
Seleccione la opción 5 nuevamente para imprimir.
2
BLOQUEADO: El documento está finalizado. No se pueden hacer cambios.
(Presione Enter para continuar)




**Conclusión**

En conclusión, estos patrones son muy útiles para el manejo tanto de una instancia segura, en el caso del singleton, y un manejo eficiente y barato de recursos; en el caso del decorador, se ajusta perfectamente a lo solicitado por el proyecto, lo cual es agregar componentes a cada plantilla; el patrón estado funciona como un módulo de seguridad que garantiza que una vez finalizada la edición, esta no se puede seguir modificando, y finalmente el patrón estructural MVC nos permite tener bien estructurado y dividido lo visual y lo lógico en nuestro programa.