

# **TECNOLÓGICO NACIONAL DE MÉXICO**

## **INSTITUTO TECNOLÓGICO DE TIJUANA**

**SUBDIRECCIÓN ACADÉMICA**  
**DEPARTAMENTO DE SISTEMAS Y COMPUTACIÓN**

**SEMESTRE:**  
Agosto - Diciembre 2025

**CARRERA:**  
Ingeniería Informática

**MATERIA:**  
Patrones de diseño de software

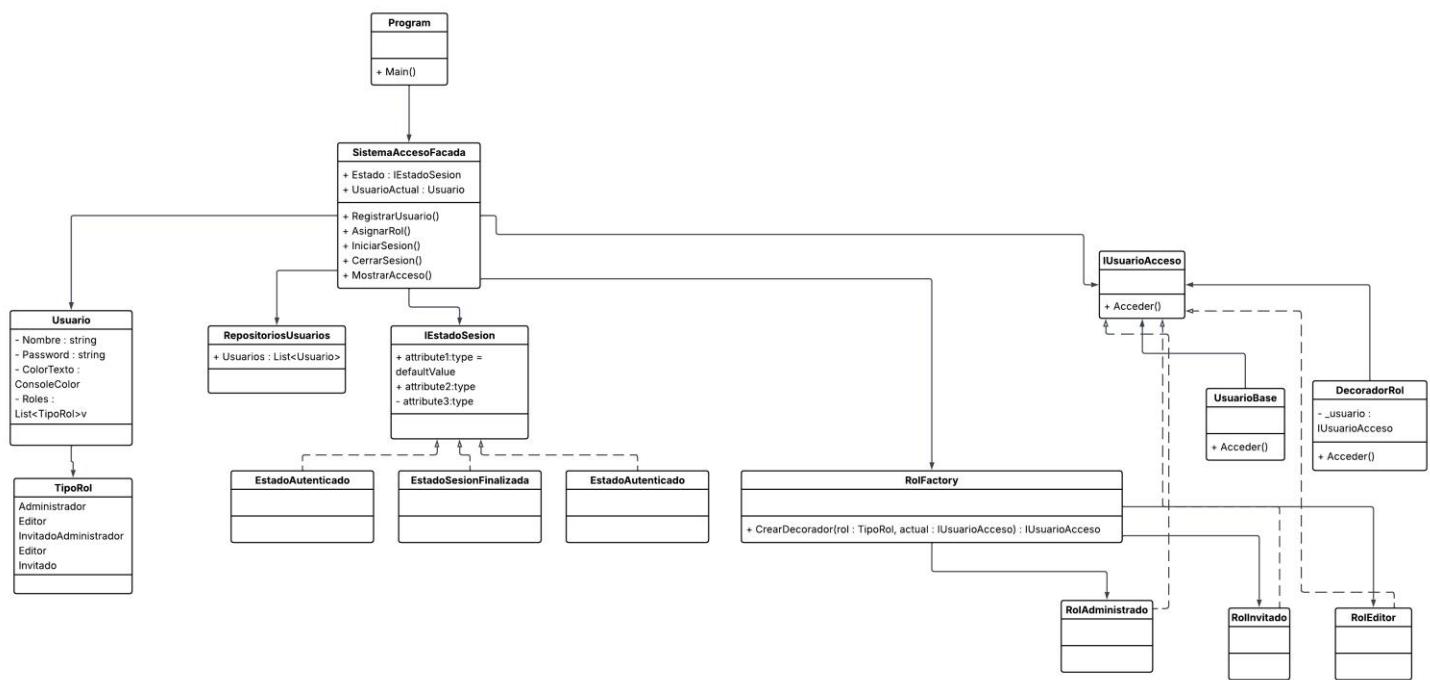
**TÍTULO ACTIVIDAD:**  
Examen unidad 4 y 5

**UNIDAD A EVALUAR:**  
Unidad 4 y 5

**NOMBRE Y NÚMERO DE CONTROL DEL ALUMNO:**  
Molina Hernández Kevin Alexander (21212351)

**NOMBRE DEL MAESTRO (A):**  
MARIBEL GUERRERO LUIS

## Diagrama UML



## Capturas de pantalla del código

- RepositorioUsuarios

```
namespace Examen_Unidad4y5
{
    3 referencias
    public static class RepositorioUsuarios
    {
        public static List<Usuario> Usuarios = new List<Usuario>();
    }
}
```

- TipoRol

```
namespace Examen_Unidad4y5.Modelos
{
    8 referencias
    public enum TipoRol
    {
        Administrador,
        Editor,
        Invitado
    }
}
```

- Usuario

```
namespace Examen_Unidad4y5.Modelos
{
    4 referencias
    public class Usuario
    {
        4 referencias
        public string Nombre { get; set; }
        2 referencias
        public string Password { get; set; }
        2 referencias
        public ConsoleColor ColorTexto { get; set; }
        2 referencias
        public List<TipoRol> Roles { get; set; } = new List<TipoRol>();
    }
}
```

## Patrón Fachada

- SistemaAccesoFacade

```
> namespace Examen_Unidad4y5.Fachada
{
    6 referencias
    public class SistemaAccesoFacade
    {
        3 referencias
        public IEstadoSesion Estado { get; set; } = new EstadoNoAutenticado();
        5 referencias
        public Usuario UsuarioActual { get; private set; }

        1 referencia
        public void RegistrarUsuario(string nombre, string password, ConsoleColor color)
        {
            RepositorioUsuarios.Usuarios.Add(new Usuario
            {
                Nombre = nombre,
                Password = password,
                ColorTexto = color
            });

            Console.WriteLine("Usuario registrado.");
        }

        1 referencia
        public void AsignarRol(string nombre, TipoRol rol)
        {
            var u = RepositorioUsuarios.Usuarios.FirstOrDefault(x => x.Nombre == nombre);
            if (u != null)
            {
                u.Roles.Add(rol);
                Console.WriteLine($"Rol {rol} asignado a {nombre}");
            }
        }

        1 referencia
        public bool IniciarSesion(string nombre, string password)
        {
            var u = RepositorioUsuarios.Usuarios.FirstOrDefault(x => x.Nombre == nombre && x.Password == password);

            if (u == null)
            {
                Console.WriteLine("Credenciales incorrectas.");
                return false;
            }
        }
}
```

```

1 referencia
public bool IniciarSesion(string nombre, string password)
{
    var u = RepositorioUsuarios.Usuarios.FirstOrDefault(x => x.Nombre == nombre && x.Password == password);

    if (u == null)
    {
        Console.WriteLine("Credenciales incorrectas.");
        return false;
    }

    UsuarioActual = u;
    Estado = new EstadoAutenticado();
    Console.WriteLine($"Bienvenido {u.Nombre}");
    return true;
}

1 referencia
public void CerrarSesion()
{
    Estado = new EstadoSesionFinalizada();
    UsuarioActual = null;
}

public void MostrarAcceso()
{
    if (UsuarioActual == null)
    {
        Estado.Manejar(this);
        return;
    }

    Console.ForegroundColor = UsuarioActual.ColorTexto;

    IUsuarioAcceso usuarioDecorado = new UsuarioBase();

    foreach (var rol in UsuarioActual.Roles)
        usuarioDecorado = RolFactory.CrearDecorador(rol, usuarioDecorado);

    usuarioDecorado.Acceder();

    Console.ResetColor();
}
}

```

## Patrón Decorar

- DecoradorRol

```

namespace Examen_Unidad4y5.Patrones.Decorator
{
    7 referencias
    public abstract class DecoradorRol : IUsuarioAcceso
    {
        protected IUsuarioAcceso _usuario;

        3 referencias
        protected DecoradorRol(IUsuarioAcceso usuario)
        {
            _usuario = usuario;
        }

        9 referencias
        public virtual void Acceder()
        {
            _usuario.Acceder();
        }
    }
}

```

- IusuarioAcceso

```
namespace Examen_Unidad4y5.Patrones.Decorator
{
    10 referencias
    public interface IUsuarioAcceso
    {
        10 referencias
        void Acceder();
    }
}
```

- RolAdministrador

```
namespace Examen_Unidad4y5.Patrones.Decorator
{
    2 referencias
    public class RolAdministrador : DecoradorRol
    {
        1 referencia
        public RolAdministrador(IUsuarioAcceso usuario) : base(usuario) { }

        7 referencias
        public override void Acceder()
        {
            base.Acceder();
            Console.WriteLine("→ Permiso: gestionar usuarios y roles.");
        }
    }
}
```

- RolEditor

```
namespace Examen_Unidad4y5.Patrones.Decorator
{
    public class RolEditor : DecoradorRol
    {
        1 referencia
        public RolEditor(IUsuarioAcceso usuario) : base(usuario) { }

        7 referencias
        public override void Acceder()
        {
            base.Acceder();
            Console.WriteLine("→ Permiso: editar contenido.");
        }
    }
}
```

- RolInvitado

```
namespace Examen_Unidad4y5.Patrones.Decorator
{
    2 referencias
    public class RolInvitado : DecoradorRol
    {
        1 referencia
        public RolInvitado(IUsuarioAcceso usuario) : base(usuario) { }

        7 referencias
        public override void Acceder()
        {
            base.Acceder();
            Console.WriteLine("→ Permiso: acceso de invitado.");
        }
    }
}
```

- UsuarioBase

```
namespace Examen_Unidad4y5.Patrones.Decorator
{
    1 referencia
    public class UsuarioBase : IUsuarioAcceso
    {
        3 referencias
        public void Acceder()
        {
            Console.WriteLine("Acceso básico al sistema.");
        }
    }
}
```

## Patrón Factory Method

- RolFactory

```
namespace Examen_Unidad4y5.Patrones.Factory
{
    1 referencia
    public static class RolFactory
    {
        1 referencia
        public static IUsuarioAcceso CrearDecorador(TipoRol rol, IUsuarioAcceso actual)
        {
            switch (rol)
            {
                case TipoRol.Administrador:
                    return new RolAdministrador(actual);

                case TipoRol.Editor:
                    return new RolEditor(actual);

                case TipoRol.Invitado:
                    return new RolInvitado(actual);

                default:
                    return actual;
            }
        }
    }
}
```

## Patrón State

- EstadoAutenticado

```
namespace Examen_Unidad4y5.Patrones.State
{
    1 referencia
    public class EstadoAutenticado : IEstadoSesion
    {
        2 referencias
        public void Manejar(SistemaAccesoFacade sistema)
        {
            Console.WriteLine("Usuario autenticado. Puedes acceder al sistema.");
        }
    }
}
```

- EstadoNoAutenticado

```
namespace Examen_Unidad4y5.Patrones.State
{
    1 referencia
    public class EstadoNoAutenticado : IEstadoSesion
    {
        2 referencias
        public void Manejar(SistemaAccesoFacade sistema)
        {
            Console.WriteLine("No has iniciado sesión.");
        }
    }
}
```

- EstadoSesionFinalizada

```
namespace Examen_Unidad4y5.Patrones.State
{
    public class EstadoSesionFinalizada : IEstadoSesion
    {

        public void Manejar(SistemaAccesoFacade sistema)
        {
            Console.WriteLine("Sesión finalizada.");
        }
    }
}
```

- IEstadoSesion

```
namespace Examen_Unidad4y5.Patrones.State
{
    public interface IEstadoSesion
    {
        void Manejar(SistemaAccesoFacade sistema);
    }
}
```

- Program

```
0 referencias
internal class Program
{
    static SistemaAccesoFacade sistema = new SistemaAccesoFacade();
0 referencias
    static void Main(string[] args)
    {
        int opcion;

        do
        {
            Console.WriteLine("\n--- CONTROL DE ACCESO ---");
            Console.WriteLine("1. Registrar usuario");
            Console.WriteLine("2. Asignar rol");
            Console.WriteLine("3. Iniciar sesión");
            Console.WriteLine("4. Mostrar permisos");
            Console.WriteLine("5. Cerrar sesión");
            Console.WriteLine("0. Salir");
            Console.Write("Seleccione: ");

            opcion = int.Parse(Console.ReadLine());

            switch (opcion)
            {
                case 1: RegistrarUsuario(); break;
                case 2: AsignarRol(); break;
                case 3: IniciarSesion(); break;
                case 4: sistema.MostrarAcceso(); break;
                case 5: sistema.CerrarSesion(); break;
            }
        } while (opcion != 0);
    }

1 referencia
    static void RegistrarUsuario()
    {
        Console.Write("Nombre: ");
        string n = Console.ReadLine();

        Console.Write("Password: ");
        string p = Console.ReadLine();

        Console.WriteLine("Elige color (1=Rojo, 2=Verde, 3=Azul): ");
        var color = int.Parse(Console.ReadLine());
    }
}
```

```
sistema.RegistrarUsuario(n, p,
    color == 1 ? ConsoleColor.Red :
    color == 2 ? ConsoleColor.Green :
    ConsoleColor.Blue);
}

1 referencia
static void AsignarRol()
{
    Console.WriteLine("Nombre usuario: ");
    string n = Console.ReadLine();

    Console.WriteLine("Rol (1=Admin, 2=Editor, 3=Invitado): ");
    int r = int.Parse(Console.ReadLine());

    sistema.AsignarRol(n, (TipoRol)(r - 1));
}

1 referencia
static void IniciarSesion()
{
    Console.WriteLine("Nombre: ");
    string n = Console.ReadLine();

    Console.WriteLine("Password: ");
    string p = Console.ReadLine();

    sistema.IniciarSesion(n, p);
}
}
```

## Ejecución de la consola

- Menú de selección

```
C:\Users\kevin\source\repos\Examen  
--- CONTROL DE ACCESO ---  
1. Registrar usuario  
2. Asignar rol  
3. Iniciar sesión  
4. Mostrar permisos  
5. Cerrar sesión  
0. Salir  
Seleccione: 
```

- Registro de usuario

```
C:\Users\kevin\source\repos\Examen_Unidad4y5\Examen  
--- CONTROL DE ACCESO ---  
1. Registrar usuario  
2. Asignar rol  
3. Iniciar sesión  
4. Mostrar permisos  
5. Cerrar sesión  
0. Salir  
Seleccione: 1  
Nombre: Kevin  
Password: 1234  
Elige color (1=Rojo, 2=Verde, 3=Azul):  
1  
Usuario registrado.
```

- Asignar rol

```
--- CONTROL DE ACCESO ---  
1. Registrar usuario  
2. Asignar rol  
3. Iniciar sesión  
4. Mostrar permisos  
5. Cerrar sesión  
0. Salir  
Seleccione: 2  
Nombre usuario: Kevin  
Rol (1=Admin, 2=Editor, 3=Invitado):  
1
```

- Iniciar sesión

```
--- CONTROL DE ACCESO ---
1. Registrar usuario
2. Asignar rol
3. Iniciar sesión
4. Mostrar permisos
5. Cerrar sesión
0. Salir
Seleccione: 3
Nombre: Kevin
Password: 1234
Bienvenido Kevin
```

- Mostrar permisos

```
--- CONTROL DE ACCESO ---
1. Registrar usuario
2. Asignar rol
3. Iniciar sesión
4. Mostrar permisos
5. Cerrar sesión
0. Salir
Seleccione: 4
Acceso básico al sistema.
→ Permiso: gestionar usuarios y roles.
```

- Cerrar sesión

```
--- CONTROL DE ACCESO ---
1. Registrar usuario
2. Asignar rol
3. Iniciar sesión
4. Mostrar permisos
5. Cerrar sesión
0. Salir
Seleccione: 5
```