TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE TIJUANA

SUBDIRECCIÓN ACADÉMICA
DEPARTAMENTO DE SISTEMAS Y COMPUTACIÓN

SEMESTRE:
Agosto-Diciembre 2025

CARRERA:
INGENIERÍA EN SISTEMAS COMPUTACIONALES

MATERIA:
Patrones de diseño

TÍTULO ACTIVIDAD :
Examen unidad 4 y 5

UNIDAD 4 Y 5

NOMBRE Y NÚMERO DE CONTROL:
Ochoa Moran Victor Alejandro - 22210329

NOMBRE DEL MAESTRO (A):

MARIBEL GUERRERO LUIS

Código

Builder

```
using System;
using EXAMEN_4_Y_5_Unidad.Models;
using EXAMEN_4_Y_5_Unidad.Models.Flyweight;

namespace EXAMEN_4_Y_5_Unidad.Models.Builder
{
    public class EventoBuilder : IEventoBuilder
    {
        private DateTime _fecha;
        private string _descripcion = "";
        private TipoEvento _tipo = null!;

        public void SetFecha(DateTime fecha) => _fecha = fecha;
        public void SetDescripcion(string descripcion) => _descripcion
= descripcion;
        public void SetTipo(TipoEvento tipo) => _tipo = tipo;

        public Evento Build()
        {
            return new Evento(_fecha, _descripcion, _tipo);
        }
    }
}
```

```
using System;
using EXAMEN_4_Y_5_Unidad.Models;
using EXAMEN_4_Y_5_Unidad.Models.Flyweight;

namespace EXAMEN_4_Y_5_Unidad.Models.Builder
{
    public interface IEventoBuilder
    {
        void SetFecha(DateTime fecha);
        void SetDescripcion(string descripcion);
        void SetTipo(TipoEvento tipo);
        Evento Build();
    }
}
```

Memento

```
using System.Collections.Generic;

namespace EXAMEN_4_Y_5_Unidad.Models.Memento
{
    public class AgendaCaretaker
    {
        private readonly Stack<AgendaMemento> _historial = new
Stack<AgendaMemento>();

        public void Guardar(AgendaMemento m) => _historial.Push(m);

        public AgendaMemento? Restaurar()
        {
            return _historial.Count > 0 ? _historial.Pop() : null;
        }
    }
}
```

```
using System.Collections.Generic;
using EXAMEN_4_Y_5_Unidad.Models;

namespace EXAMEN_4_Y_5_Unidad.Models.Memento
{
    public class AgendaMemento
    {
        public List<Evento> EstadoEventos { get; }

        public AgendaMemento(List<Evento> estado)
        {
            EstadoEventos = new List<Evento>(estado);
        }
    }
}
```

Peso ligero

```
using System.Collections.Generic;

namespace EXAMEN_4_Y_5_Unidad.Models.Flyweight
{
    public static class FabricaTiposEvento
    {
        private static Dictionary<string, TipoEvento> tipos = new
Dictionary<string, TipoEvento>();

        public static TipoEvento ObtenerTipo(string nombre)
        {
            if (!tipos.ContainsKey(nombre))
            {
                tipos[nombre] = CrearTipo(nombre);
            }

            return tipos[nombre];
        }

        private static TipoEvento CrearTipo(string nombre)
        {
            return nombre switch
            {
                "Reunión"      => new TipoEvento("Reunión", ""),
                "Pago"         => new TipoEvento("Pago", ""),
                "Cita"         => new TipoEvento("Cita", ""),
                "Recordatorio" => new TipoEvento("Recordatorio", ""),
                _              => new TipoEvento("General", ""),
            };
        }
    }
}
```

```
namespace EXAMEN_4_Y_5_Unidad.Models.Flyweight
{
    public class TipoEvento
    {
        public string Nombre { get; }
        public string Color { get; }

        public TipoEvento(string nombre, string color)
        {
            Nombre = nombre;
            Color = color;
        }
    }
}
```

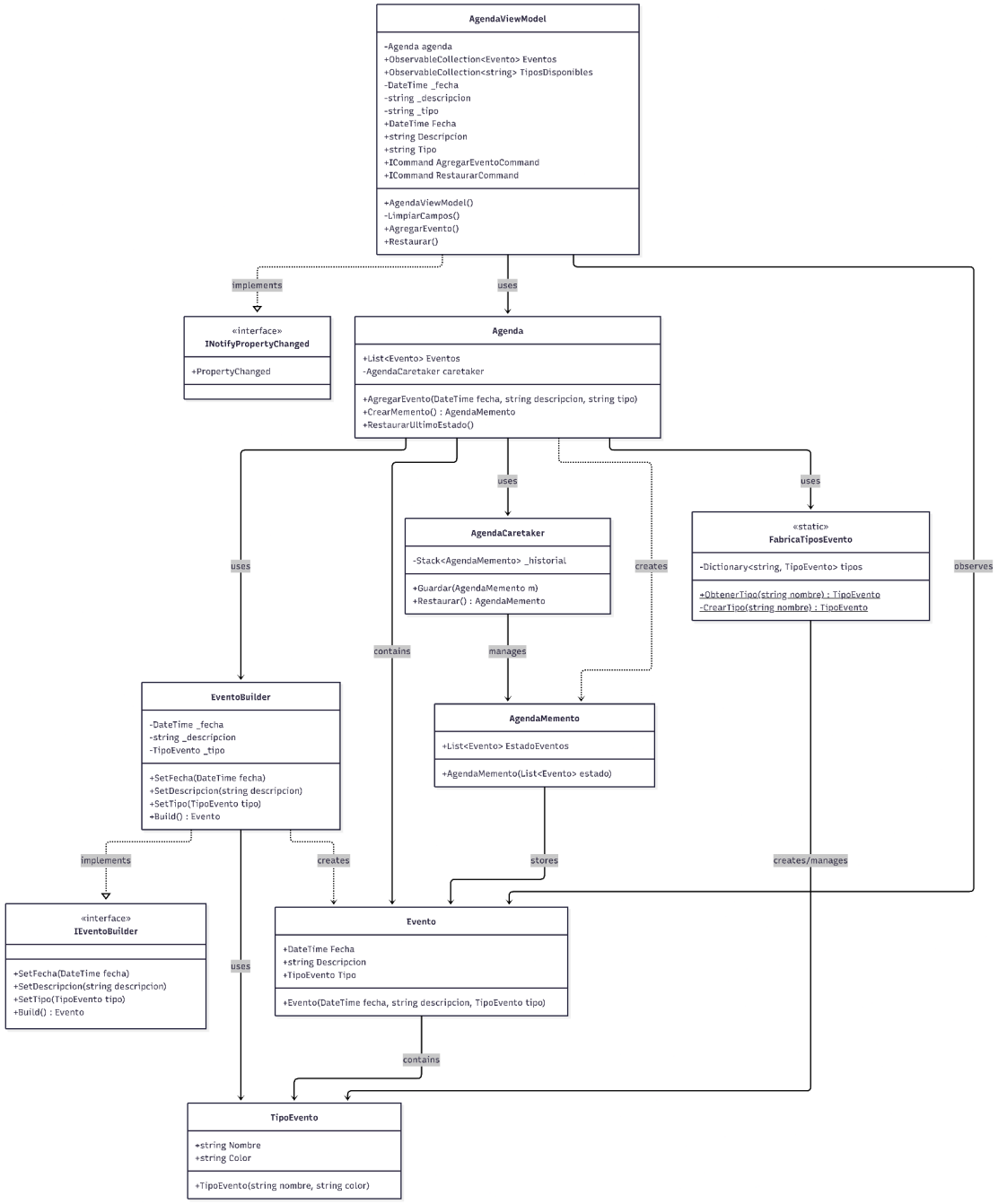
MVVM

```
using System;
using System.Collections.ObjectModel;
using System.ComponentModel;
using System.Runtime.CompilerServices;
using EXAMEN_4_Y_5_Unidad.Models;
using System.Windows.Input;
using Microsoft.Maui.Controls;

namespace EXAMEN_4_Y_5_Unidad.ViewModels
{
    public class AgendaViewModel : INotifyPropertyChanged
    {
        private Agenda agenda = new Agenda();


        public ObservableCollection<Evento> Eventos { get; set; }
            = new ObservableCollection<Evento>();


        public ObservableCollection<string> TiposDisponibles { get;
set; }
            = new ObservableCollection<string>
            {
                "Reunión",
                "Pago",
                "Cita",
```

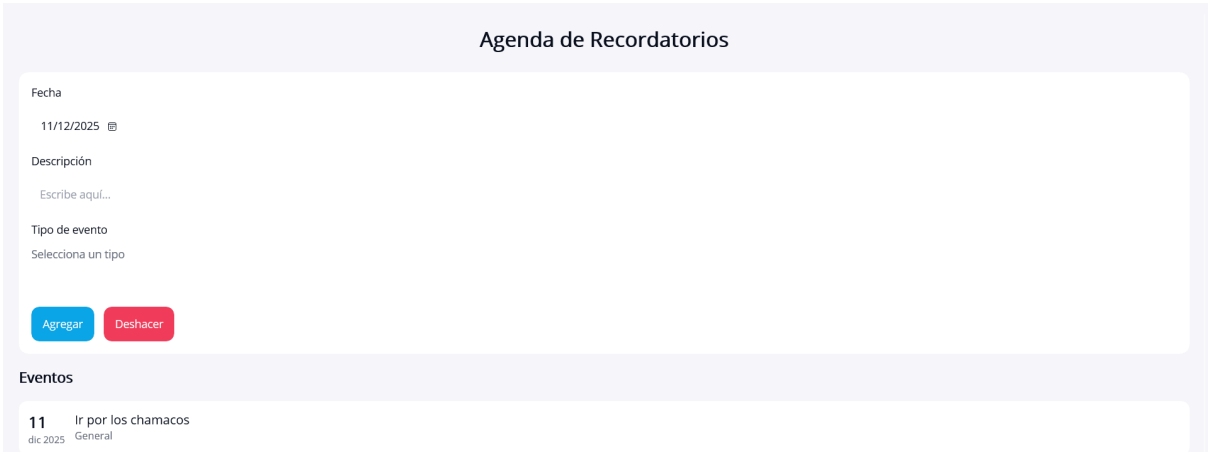
```
            "Recordatorio"
        };


    private DateTime _fecha = DateTime.Now;
    public DateTime Fecha
    {
        get => _fecha;
        set { _fecha = value; OnPropertyChanged(); }
    }

    private string _descripcion = "";
    public string Descripcion
    {
        get => _descripcion;
        set { _descripcion = value; OnPropertyChanged(); }
    }

    private string _tipo = "";
    public string Tipo
    {
        get => _tipo;
        set { _tipo = value; OnPropertyChanged(); }
    }


    public ICommand AgregarEventoCommand { get; }
    public ICommand RestaurarCommand { get; }


    public AgendaViewModel()
    {
        AgregarEventoCommand = new Command(AgregarEvento);
        RestaurarCommand = new Command(Restaurar);
    }



    private void LimpiarCampos()
    {
        Fecha = DateTime.Now;
        Descripcion = "";
        Tipo = "";
```

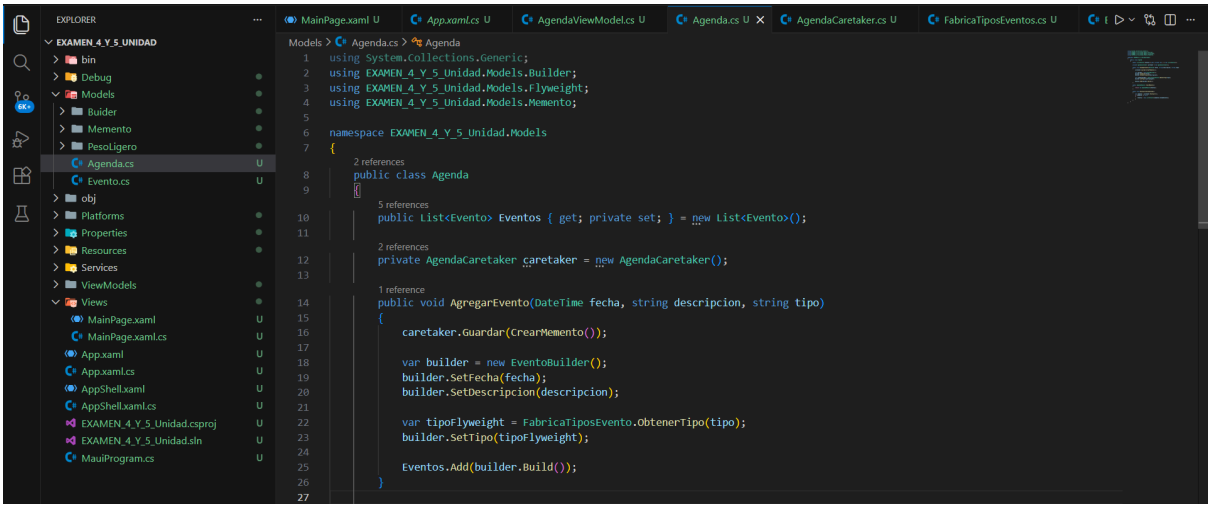
```
        }

        public void AgregarEvento()
        {
            agenda.AgregarEvento(Fecha, Descripcion, Tipo);

            Eventos.Clear();
            foreach (var e in agenda.Eventos)
                Eventos.Add(e);

            LimpiarCampos();
        }



        public void Restaurar()
        {
            agenda.RestaurarUltimoEstado();

            Eventos.Clear();
            foreach (var e in agenda.Eventos)
                Eventos.Add(e);
        }



        public event PropertyChangedEventHandler? PropertyChanged;
        protected void OnPropertyChanged([CallerMemberName] string name
= "")
            => PropertyChanged?.Invoke(this, new
PropertyChangedEventArgs(name));
    }
}
```

UML


AgendaViewModel
-Agenda agenda
+ObservableCollection<Evento> Eventos
+ObservableCollection<string> TiposDisponibles
-DateTime _fecha
-string _descripcion
-string _tipo
+DateTime Fecha
+string Descripcion
+string Tipo
+ICommand AgregarEventoCommand
+ICommand RestaurarCommand
+AgendaViewModel()
-LimpiarCampos()
+AgregarEvento()
+Restaurar()
implements
uses
«interface»
INotifyPropertyChanged
+PropertyChanged
Agenda
+List<Evento> Eventos
-AgendaCaretaker caretaker
+AgregarEvento(DateTime fecha, string descripcion, string tipo)
+CrearMemento() : AgendaMemento
+RestaurarUltimoEstado()
uses
uses
uses
AgendaCaretaker
-Stack<AgendaMemento> _historial
+Guardar(AgendaMemento m)
+Restaurar() : AgendaMemento
«static»
FabricaTiposEvento
-Dictionary<string, TipoEvento> tipos
+ObtenerTipo(string nombre) : TipoEvento
-CrearTipo(string nombre) : TipoEvento
creates
observes
contains
manages
EventoBuilder
-DateTime _fecha
-string _descripcion
-TipoEvento _tipo
+SetFecha(DateTime fecha)
+SetDescripcion(string descripcion)
+SetTipo(TipoEvento tipo)
+Build() : Evento
AgendaMemento
+List<Evento> EstadoEventos
+AgendaMemento(List<Evento> estado)
implements
creates
stores
creates/manages
«interface»
IEventoBuilder
+SetFecha(DateTime fecha)
+SetDescripcion(string descripcion)
+SetTipo(TipoEvento tipo)
+Build() : Evento
uses
Evento
+DateTime Fecha
+string Descripcion
+TipoEvento Tipo
+Evento(DateTime fecha, string descripcion, TipoEvento tipo)
contains
TipoEvento
+string Nombre
+string Color
+TipoEvento(string nombre, string color)

Capturas


Agenda de Recordatorios
Fecha
11/12/2025
Descripción
Escribe aquí...
Tipo de evento
Selecciona un tipo
Agregar
Deshacer
Eventos
11
dic 2025
Ir por los chamacos
General


EXPLORER
EXAMEN_4_Y_5_UNIDAD
bin
Debug
Models
Buider
Memento
PesoLigero
Agenda.cs
Evento.cs
obj
Platforms
Properties
Resources
Services
ViewModels
Views
MainPage.xaml
MainPage.xaml.cs
App.xaml
App.xaml.cs
AppShell.xaml
AppShell.xaml.cs
EXAMEN_4_Y_5_Unidad.csproj
EXAMEN_4_Y_5_Unidad.sln
MauiProgram.cs
MainPage.xaml U
App.xaml.cs U
AgendaViewModel.cs U
Agenda.cs U
AgendaCaretaker.cs U
FabricaTiposEventos.cs U
Models > Agenda.cs > Agenda
using System.Collections.Generic;
using EXAMEN_4_Y_5_Unidad.Models.Builder;
using EXAMEN_4_Y_5_Unidad.Models.Flyweight;
using EXAMEN_4_Y_5_Unidad.Models.Memento;
namespace EXAMEN_4_Y_5_Unidad.Models
{
2 references
public class Agenda
{
5 references
public List<Evento> Eventos { get; private set; } = new List<Evento>();
2 references
private AgendaCaretaker caretaker = new AgendaCaretaker();
1 reference
public void AgregarEvento(DateTime fecha, string descripcion, string tipo)
{
caretaker.Guardar(CrearMemento());
var builder = new EventoBuilder();
builder.SetFecha(fecha);
builder.SetDescripcion(descripcion);
var tipoFlyweight = FabricaTiposEvento.ObtenerTipo(tipo);
builder.SetTipo(tipoFlyweight);
Eventos.Add(builder.Build());
}

Conclusión

En el desarrollo de esta aplicación implemente los patrones Builder, Flyweight, Memento y MVVM, ya que cada uno aporta soluciones específicas a problemas comunes en la gestión de objetos, optimización de recursos, control de estados y separación de responsabilidades. El patrón Builder lo escogí para estandarizar la creación de eventos, evitando constructores largos y asegurando que cada evento se construya de forma facil y flexible. Esto facilita que, si en un futuro un evento requiere más atributos, el proceso de construcción siga siendo ordenado y fácil de mantener. El patrón Flyweight se implementó para optimizar memoria al reutilizar objetos que se repiten, como los tipos de eventos.

El patrón Memento lo agregue para ofrecer una funcionalidad esencial en aplicaciones donde existe un historial de cambios: deshacer acciones. Al guardar estados previos de la agenda y el patrón MVVM se utilizó como la arquitectura principal debido a que separa la interfaz gráfica, la lógica de interacción y las reglas del sistema .

Estos patrones ayudaron a tener más claridad de cómo funcionan y actúan cuando el programa está en funcionamiento.