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 de software

TÍTULO ACTIVIDAD:
Examen Unidad 2

UNIDAD A EVALUAR:
2da unidad

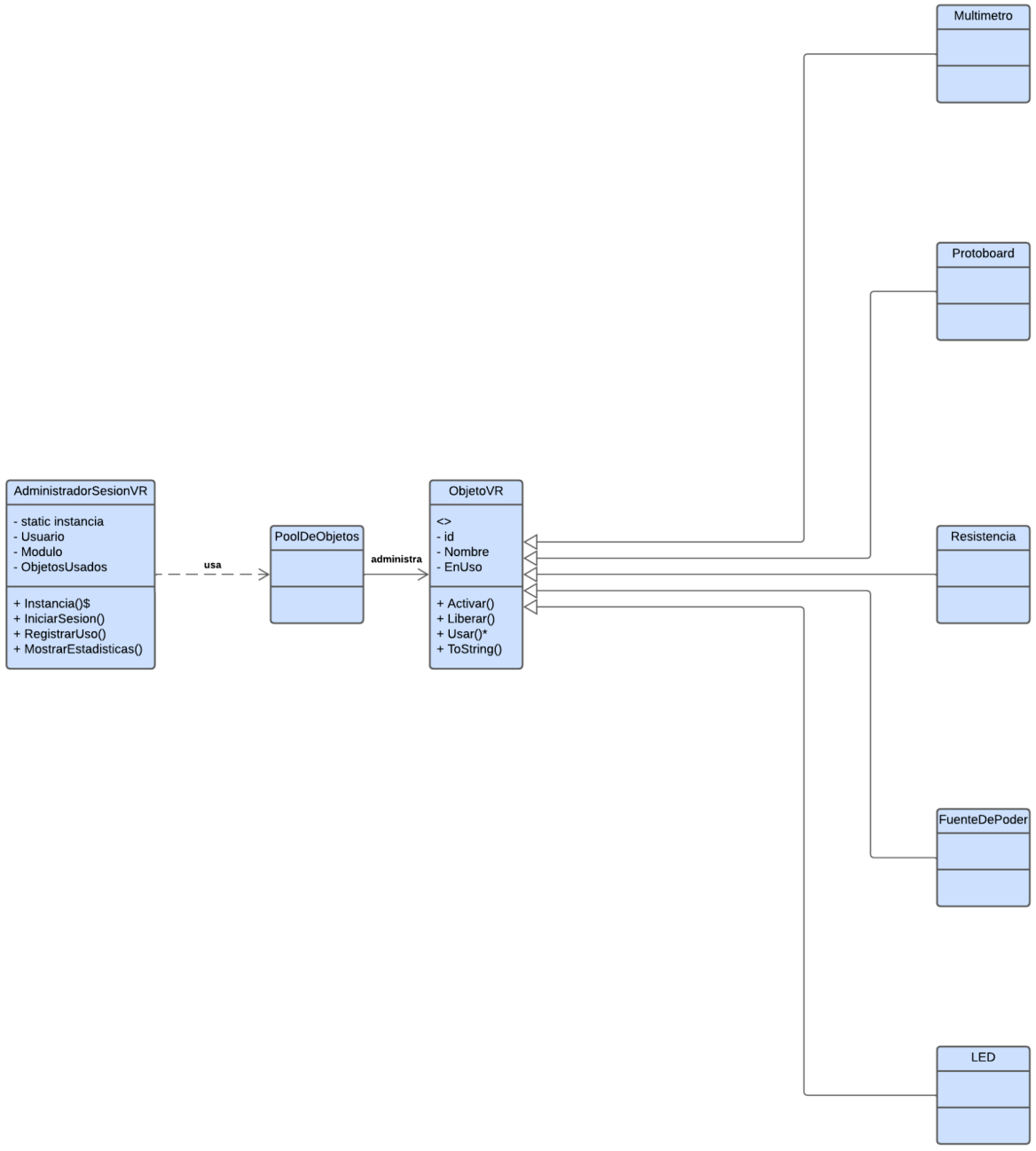
Fecha:
17 de octubre del 2025

NOMBRE Y NÚMERO DE CONTROL DEL ALUMNO:
Maya Lopez Diego Enrique

NOMBRE DEL MAESTRO (A):
Maribel Guerrero Luis

Índice

Diagrama UML



Código

AdministradorSesionVR

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

namespace CentroEducativoVR
{
    public class AdministradorSesionVR
    {
        private static AdministradorSesionVR _instancia;

        public string Usuario { get; private set; }
        public string Modulo { get; private set; }
        public int ObjetosUsados { get; private set; }

        private AdministradorSesionVR() { }

        public static AdministradorSesionVR Instancia
        {
            get
            {
                if (_instancia == null)
                    _instancia = new AdministradorSesionVR();
                return _instancia;
            }
        }

        public void IniciarSesion(string usuario, string modulo)
        {
            Usuario = usuario;
            Modulo = modulo;
            ObjetosUsados = 0;
            Console.WriteLine($"[SesionVR] Sesión iniciada: {usuario}
en {modulo}");
        }

        public void RegistrarUso()
        {
            ObjetosUsados++;
        }
```

```
        public void MostrarEstadisticas()
        {
            Console.WriteLine($"\n=== ESTADÍSTICAS ===");
            Console.WriteLine($"Usuario: {Usuario}");
            Console.WriteLine($"Módulo: {Modulo}");
            Console.WriteLine($"Objetos usados: {ObjetosUsados}");
            Console.WriteLine($"====================\n");
        }
    }
}
```

FuenteDePoder

```
using System;

namespace CentroEducativoVR
{
    public class FuenteDePoder : ObjetoVR
    {
        public FuenteDePoder() : base("Fuente de Poder 0-30V") { }

        public override void Usar()
        {
            Console.WriteLine("[Fuente] Suministrando 5V DC al
protoboard");
        }
    }
}
```

LED

```
using System;

namespace CentroEducativoVR
{
    public class LED : ObjetoVR
    {
        public LED() : base("LED Rojo 5mm") { }

        public override void Usar()
        {
            Console.WriteLine("[LED] Emitiendo luz roja - Circuito
funcionando correctamente");
        }
    }
}
```

Multímetro

```
using System;

namespace CentroEducativoVR
{
    public class Multimetro : ObjetoVR
    {
        public Multimetro() : base("Multímetro Fluke 117") { }

        public override void Usar()
        {
            Console.WriteLine("[Multímetro] Midiendo voltaje del
circuito: 5.2V DC");
        }
    }
}
```

ObjetoVR

```
using System;
namespace CentroEducativoVR
{
    public abstract class ObjetoVR
    {
        public int id { get; set; }
        public string Nombre { get; set; }
        public bool EnUso { get; set; }

        protected ObjetoVR(string nombre)
        {

            Nombre = nombre;
            EnUso = false;
        }

        public abstract void Usar();

        public void Activar()
        {
            EnUso = true;
            Console.WriteLine($"[VRObjeto] {Nombre} (ID: {id})
activado.");
        }

        public void Liberar()
        {
            EnUso = false;
            Console.WriteLine($"[VRObjeto] {Nombre} (ID: {id}) liberado
y listo para usar.");
        }

        public override string ToString()
        {
            return $"{Nombre} [ID: {id}, En uso: {EnUso}]";
        }
    }
}
```

PoolDeObjetos

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

namespace CentroEducativoVR
{
    public class PoolDeObjetos<T> where T : ObjetoVR
    {
        private List<T> _objetos = new List<T>();
        private Func<T> _crear;
        private int _siguienteId = 1;

        public PoolDeObjetos(Func<T> crear)
        {
            _crear = crear;
        }

        public T Obtener()
        {
            // Buscar objeto libre
            foreach (var obj in _objetos)
            {
                if (!obj.EnUso)
                {
                    Console.WriteLine("[Pool] Reutilizando objeto");
                    obj.Activar();
                    return obj;
                }
            }

            // Crear nuevo objeto
            Console.WriteLine("[Pool] Creando nuevo objeto");
            var nuevo = _crear();
            nuevo.id = _siguienteId++;
            nuevo.Activar();
            _objetos.Add(nuevo);
            return nuevo;
        }

        public void Liberar(T objeto)
        {
            objeto.Liberar();
        }
```

```
    }
}
```

Program

```
using System;

namespace CentroEducativoVR
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("=== LABORATORIO VIRTUAL DE ELECTRÓNICA
===\n");

            // Patrón Singleton
            var sesion = AdministradorSesionVR.Instancia;
            sesion.IniciarSesion("Diego Maya", "Módulo de Circuitos
Básicos");

            // Patrón Object Pool - Componentes electrónicos
            var poolMultimetros = new PoolDeObjetos<Multimetro>(() =>
new Multimetro());
            var poolProtoboards = new PoolDeObjetos<Protoboard>(() =>
new Protoboard());
            var poolResistencias = new PoolDeObjetos<Resistencia>(() =>
new Resistencia());
            var poolLEDs = new PoolDeObjetos<LED>(() => new LED());
            var poolFuentes = new PoolDeObjetos<FuenteDePoder>(() =>
new FuenteDePoder());

            // Práctica 1: Circuito LED simple
            Console.WriteLine("\n--- PRÁCTICA 1: CIRCUITO LED SIMPLE
---");
            var protoboard1 = poolProtoboards.Obtener();
            protoboard1.Usar();

            var fuente1 = poolFuentes.Obtener();
            fuente1.Usar();

            var resistencia1 = poolResistencias.Obtener();
            resistencia1.Usar();
```

```
            var led1 = poolLEDs.Obtener();
            led1.Usar();

            var multimetro1 = poolMultimetros.Obtener();
            multimetro1.Usar();

            // Liberar algunos componentes
            Console.WriteLine("\n--- FINALIZANDO PRÁCTICA 1 ---");
            poolResistencias.Liberar(resistencia1);
            poolLEDs.Liberar(led1);

            // Práctica 2: Circuito en paralelo
            Console.WriteLine("\n--- PRÁCTICA 2: CIRCUITO EN PARALELO
---");
            var resistencia2 = poolResistencias.Obtener(); // Reutiliza
            resistencia2.Usar();

            var led2 = poolLEDs.Obtener(); // Reutiliza
            led2.Usar();

            var led3 = poolLEDs.Obtener(); // Crea nuevo
            led3.Usar();

            // Cambiar módulo
            Console.WriteLine("\n--- CAMBIANDO A MÓDULO AVANZADO ---");
            sesion.IniciarSesion("Diego Maya", "Módulo de Circuitos
Integrados");

            // Usar más componentes
            var multimetro2 = poolMultimetros.Obtener();
            multimetro2.Usar();

            // Mostrar estadísticas finales
            sesion.MostrarEstadisticas();

            Console.WriteLine("\n=== FIN DEL LABORATORIO VIRTUAL ===");
            Console.WriteLine("Presiona cualquier tecla para
salir...");
            Console.ReadKey();
        }
    }
}
```

Protoboard

```
using System;

namespace CentroEducativoVR
{
    public class Protoboard : ObjetoVR
    {
        public Protoboard() : base("Protoboard 830 puntos") { }

        public override void Usar()
        {
            Console.WriteLine("[Protoboard] Conectando componentes en
la placa de pruebas");
        }
    }
}
```

Resistencia

```
using System;

namespace CentroEducativoVR
{
    public class Resistencia : ObjetoVR
    {
        public Resistencia() : base("Resistencia 220Ω") { }

        public override void Usar()
        {
            Console.WriteLine("[Resistencia] Limitando corriente en el
circuito LED");
        }
    }
}
```

Capturas del programa

```
PS J:\Patrones de diseño\ExamenUnidad2_Patrones_MayaLopezDiegoEnrique\src\Examen> dotnet run
=== LABORATORIO VIRTUAL DE ELECTRÓNICA ===

[SesionVR] Sesión iniciada: Diego Maya en Módulo de Circuitos Básicos

--- PRÁCTICA 1: CIRCUITO LED SIMPLE ---
[Pool] Creando nuevo objeto
[VRObjeto] Protoboard 830 puntos (ID: 1) activado.
[Protoboard] Conectando componentes en la placa de pruebas
[Pool] Creando nuevo objeto
[VRObjeto] Fuente de Poder 0-30V (ID: 1) activado.
[Fuente] Suministrando 5V DC al protoboard
[Pool] Creando nuevo objeto
[VRObjeto] Resistencia 220Ω (ID: 1) activado.
[Resistencia] Limitando corriente en el circuito LED
[Pool] Creando nuevo objeto
[VRObjeto] LED Rojo 5mm (ID: 1) activado.
[LED] Emitiendo luz roja - Circuito funcionando correctamente
[Pool] Creando nuevo objeto
[VRObjeto] Multímetro Fluke 117 (ID: 1) activado.
[Multímetro] Midiendo voltaje del circuito: 5.2V DC

--- FINALIZANDO PRÁCTICA 1 ---
[VRObjeto] Resistencia 220Ω (ID: 1) liberado y listo para usar.
[VRObjeto] LED Rojo 5mm (ID: 1) liberado y listo para usar.

--- PRÁCTICA 2: CIRCUITO EN PARALELO ---
[Pool] Reutilizando objeto
[VRObjeto] Resistencia 220Ω (ID: 1) activado.
[Resistencia] Limitando corriente en el circuito LED
[Pool] Reutilizando objeto
[VRObjeto] LED Rojo 5mm (ID: 1) activado.
[LED] Emitiendo luz roja - Circuito funcionando correctamente
[Pool] Creando nuevo objeto
[VRObjeto] LED Rojo 5mm (ID: 2) activado.
[LED] Emitiendo luz roja - Circuito funcionando correctamente

--- CAMBIANDO A MÓDULO AVANZADO ---
[SesionVR] Sesión iniciada: Diego Maya en Módulo de Circuitos Integrados
[Pool] Creando nuevo objeto
[VRObjeto] Multímetro Fluke 117 (ID: 2) activado.
[Multímetro] Midiendo voltaje del circuito: 5.2V DC

=== ESTADÍSTICAS ===
Usuario: Diego Maya
Módulo: Módulo de Circuitos Integrados
Objetos usados: 0
==================


=== FIN DEL LABORATORIO VIRTUAL ===
Presiona cualquier tecla para salir...
PS J:\Patrones de diseño\ExamenUnidad2_Patrones_MayaLopezDiegoEnrique\src\Examen>
```

Conclusión

La integración del patrón Singleton junto con Object Pool representa una práctica sólida en el diseño de sistemas que requieren control centralizado y uso eficiente de recursos. El Singleton asegura que toda la administración del entorno ocurra desde una única instancia, evitando conflictos y manteniendo coherencia en la sesión. Paralelamente, el Object Pool optimiza el rendimiento al reutilizar objetos en lugar de crear instancias nuevas de forma innecesaria.

Esta fusión no solo mejora la gestión de memoria y procesamiento, sino que también promueve un código más ordenado, escalable y alineado con principios de reutilización y eficiencia.