

Universidad ORT Uruguay

Facultad de Ingeniería

Escuela de Tecnología

OBLIGATORIO PROGRAMACIÓN 2

DOCUMENTO DE ANÁLISIS



Emmanuel Moyano - 321266



Facundo Malacrida – 296868

M2E

Docente: Lucas López

Analista en Tecnologías de la Información

16-10-2025

<u>DOCUMENTO DE ANÁLISIS</u>	1
<u>Diagrama de Clases</u>	2
<u>Diagrama de Casos de Uso</u>	2
<u>Azure Deploy</u>	2
<u>Precarga de Datos</u>	3
<u>Equipos</u>	3
<u>Tipos de Gastos</u>	4
<u>Usuarios</u>	5
<u>Pagos Únicos</u>	6
<u>Pagos Recurrentes</u>	8
<u>Código Fuente</u>	8
<u>En Sistema</u>	8
<u>En MetodoDePago</u>	16
<u>En Equipo</u>	17
<u>En Usuario</u>	18
<u>En TipoDeGasto</u>	21
<u>En Pago</u>	22
<u>En Recurrente</u>	24
<u>En Unico</u>	26
<u>En Program</u>	26
<u>Evidencias IAG</u>	31
<u>Evidencia de testing:</u>	35

Diagrama de Clases

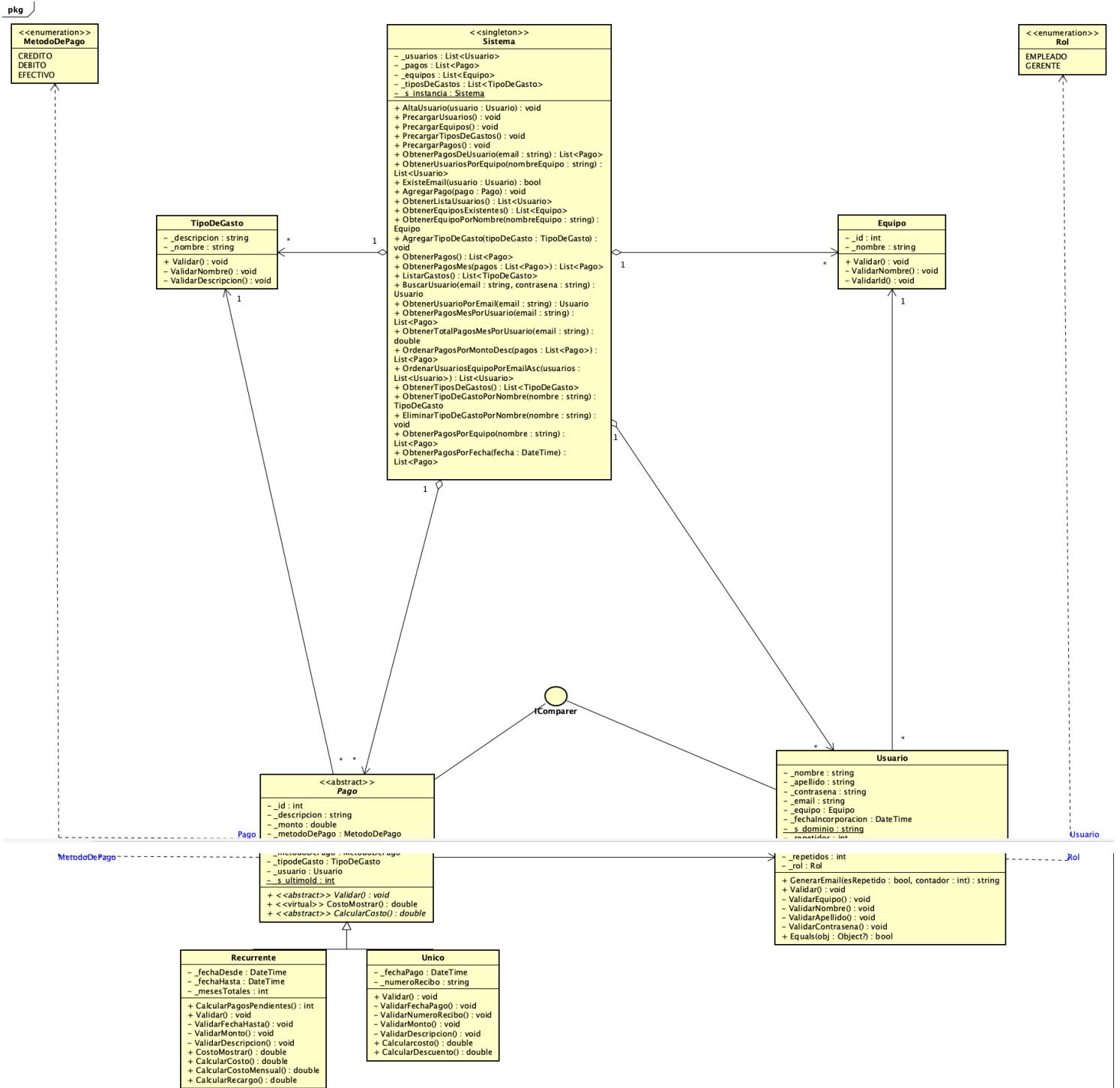
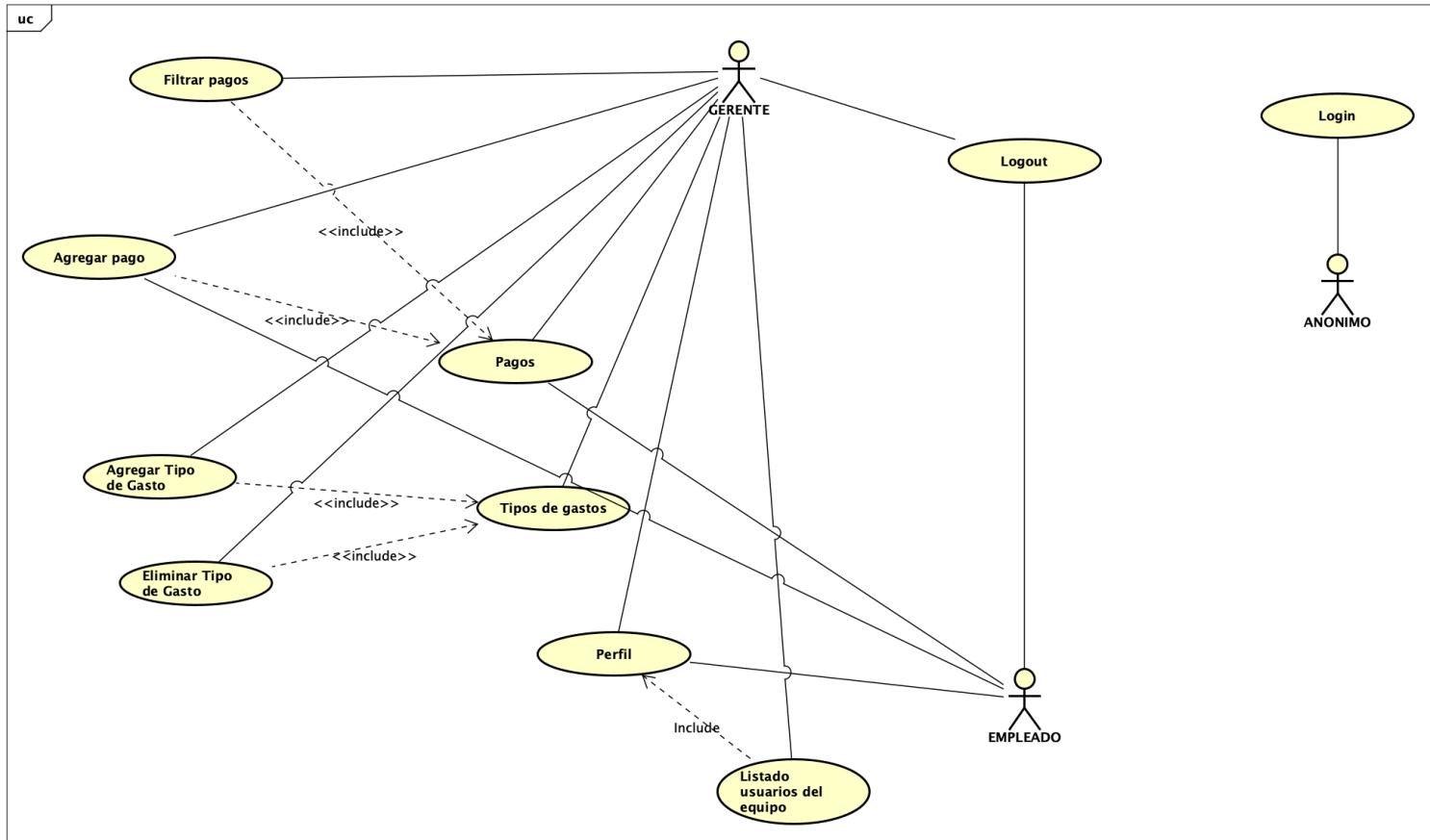


Diagrama de Casos de Uso



Azure Deploy

[Link al deploy del proyecto](#)

Precarga de Datos

Equipos

Equipos		▼	▼
id	▼	nombre	▼
1	"Desarrollo"		
2	"Marketing"		
3	"Finanzas"		
4	"Recursos Humanos"		

Tipos de Gastos

Tipos de Gastos	▼	Calendario	
nombre	▼	descripcion	▼
"Alquiler"		"Pago mensual del local u oficina"	
"Servicios"		"Luz, agua, internet, etc."	
"Marketing"		"Publicidad y redes sociales"	
"Salarios"		"Pago de sueldos a empleados"	
"Transporte"		"Gastos de movilidad"	
"Comida"		"Snacks y almuerzos de equipo"	
"Software"		"Licencias y herramientas digitales"	
"Mantenimiento"		"Reparaciones o limpieza"	
"Eventos"		"Capacitaciones y eventos corporativos"	
"Otros"		"Gastos varios no categorizados"	

Usuarios

Usuarios				
nombre	apellido	contrasena	Equipo	fechalIncorporacion
"Ana"	"Lopez"	"pass123n"	_equipos[0]	...AddDays(-400)
"Bruno"	"Perez"	"clave789"	_equipos[0]	...AddDays(-300)
"Camila"	"Ramos"	"abc12341a"	_equipos[1]	...AddDays(-200)
"Diego"	"Suarez"	"xyz789ba0"	_equipos[1]	...AddDays(-150)
"Elena"	"Martinez"	"elena212kj"	_equipos[2]	...AddDays(-120)
"Federico"	"Gomez"	"fede456hajw"	_equipos[2]	...AddDays(-100)
"Gabriela"	"Diaz"	"gaby789n"	_equipos[3]	...AddDays(-90)
"Hector"	"Torres"	"htorres1"	_equipos[0]	...AddDays(-80)
"Isabel"	"Vega"	"isavega7"	_equipos[0]	...AddDays(-75)
"Javier"	"Rosa"	"javiro521"	_equipos[1]	...AddDays(-60)
"Karen"	"Nuñez"	"karenn999"	_equipos[1]	...AddDays(-50)
"Leonardo"	"Paz"	"leopaz666"	_equipos[2]	...AddDays(-40)
"Maria"	"Quintero"	"maria1010"	_equipos[2]	...AddDays(-30)
"Nicolas"	"Rivero"	"nrivero8"	_equipos[3]	...AddDays(-25)
"Olga"	"Fernandez"	"olga12345"	_equipos[0]	...AddDays(-20)
"Pablo"	"Lopez"	"pablo2211"	_equipos[1]	...AddDays(-15)
"Rocio"	"Santos"	"rocio33n1"	_equipos[1]	...AddDays(-10)
"Sergio"	"Herrera"	"sergioa7"	_equipos[2]	...AddDays(-8)
"Tamara"	"Rey"	"tammy44455"	_equipos[3]	...AddDays(-5)
"Ulises"	"Roldan"	"ulises101"	_equipos[3]	...AddDays(-3)
"Valeria"	"Mendez"	"valm123jas"	_equipos[2]	...AddDays(-2)
"Walter"	"Iglesias"	"walter9walter"	_equipos[0]	DateTime.Now

Pagos Únicos

Pagos Únicos													
descripcion	▼	monto	▼	metodoDePago	▼	tipoDeGasto	▼	usuario	▼	fechaPago	▼	numeroRecibo	▼
"Alquiler Octubre"		1200		CREDITO		g[0]		u[0]		...AddDays(-30)		"REC001"	
"Factura de Luz"		300		DEBITO		g[1]		u[2]		...AddDays(-25)		"REC002"	
"Campaña Facebook"		500		CREDITO		g[2]		u[3]		...AddDays(-20)		"REC003"	
"Pago de Sueldo Elena"		2200		EFFECTIVO		g[3]		u[4]		...AddDays(-15)		"REC004"	
"Taxi reunión cliente"		80		DEBITO		g[4]		u[5]		...AddDays(-10)		"REC005"	
"Comida equipo"		150		EFFECTIVO		g[5]		u[6]		...AddDays(-5)		"REC006"	
"Licencia Photoshop"		400		CREDITO		g[6]		u[7]		...AddDays(-45)		"REC007"	
"Reparación PC"		250		DEBITO		g[7]		u[8]		...AddDays(-35)		"REC008"	
"Evento Team Building"		1000		CREDITO		g[8]		u[9]		...AddDays(-60)		"REC009"	
"Papel y útiles"		90		EFFECTIVO		g[9]		u[10]		...AddDays(-7)		"REC010"	
"Mantenimiento aire"		350		DEBITO		g[7]		u[11]		...AddDays(-40)		"REC011"	
"Publicidad Instagram"		270		CREDITO		g[2]		u[12]		...AddDays(-12)		"REC012"	
"Snacks oficina"		50		EFFECTIVO		g[5]		u[13]		...AddDays(-3)		"REC013"	
"Taxi aeropuerto"		60		DEBITO		g[4]		u[14]		...AddDays(-8)		"REC014"	
"Hosting web"		200		CREDITO		g[6]		u[15]		...AddDays(-6)		"REC015"	
"Desinfección oficina"		180		EFFECTIVO		g[7]		u[16]		...AddDays(-4)		"REC016"	
"Cafetera nueva"		120		DEBITO		g[9]		u[17]		...AddDays(-2)		"REC017"	

Pagos Recurrentes

Pagos Recurrentes										
descripcion	monito	metodoDePago	tipoDeGasto	usuario	desde			hasta		
"Internet mensual"	60	DEBITO	g[1]	u[0]	...AddMonths(-6)			...AddMonths(-1)		
"Servicio limpieza"	100	CREDITO	g[7]	u[1]	...AddMonths(-5)			...AddMonths(-1)		
"Netflix oficina"	15	CREDITO	g[6]	u[2]	...AddMonths(-4)			...AddMonths(-1)		
"Seguro empresa"	300	DEBITO	g[9]	u[3]	...AddMonths(-12)			...AddMonths(-2)		
"Plan telefonía"	80	EFECTIVO	g[1]	u[4]	...AddMonths(-6)			...AddMonths(-1)		
"Alquiler mensual"	1200	CREDITO	g[0]	u[5]	...AddMonths(-2)			...AddMonths(4)		
"Electricidad"	250	DEBITO	g[1]	u[6]	...AddMonths(-1)			...AddMonths(5)		
"Spotify"	10	CREDITO	g[6]	u[7]	...AddMonths(-3)			...AddMonths(3)		
"Publicidad Google"	400	CREDITO	g[2]	u[8]	...AddMonths(-2)			...AddMonths(6)		
"Plan comida"	180	EFECTIVO	g[5]	u[9]	...AddMonths(-1)			...AddMonths(5)		
"Hosting servidor"	220	CREDITO	g[6]	u[10]	...AddMonths(-2)			...AddMonths(6)		
"Mantenimiento impresora"	90	DEBITO	g[7]	u[11]	...AddMonths(-1)			...AddMonths(5)		
"Capacitación equipo"	350	CREDITO	g[8]	u[12]	...AddMonths(-1)			...AddMonths(3)		
"Suscripción diseño"	45	CREDITO	g[6]	u[13]	...AddMonths(-2)			...AddMonths(4)		
"Transporte interno"	100	EFECTIVO	g[4]	u[14]	...AddMonths(-1)			...AddMonths(3)		
"Snacks oficina"	60	DEBITO	g[5]	u[15]	...AddMonths(-2)			...AddMonths(5)		
"Publicidad mensual"	300	CREDITO	g[2]	u[16]	...AddMonths(-1)			...AddMonths(6)		
"Limpieza vidrios"	80	EFECTIVO	g[7]	u[17]	...AddMonths(-1)			...AddMonths(4)		
"Software contable"	250	CREDITO	g[6]	u[18]	...AddMonths(-2)			...AddMonths(5)		
"Asesoría legal"	500	DEBITO	g[9]	u[19]	...AddMonths(-3)			...AddMonths(3)		
"Mantenimiento ascensor"	120	CREDITO	g[7]	u[20]	...AddMonths(-1)			...AddMonths(5)		
"Plan de salud"	350	DEBITO	g[9]	u[21]	...AddMonths(-4)			...AddMonths(2)		
"Correo corporativo"	30	CREDITO	g[6]	u[0]	...AddMonths(-5)			...AddMonths(5)		
"Publicidad local"	150	EFECTIVO	g[2]	u[1]	...AddMonths(-2)			...AddMonths(3)		
"Combustible mensual"	200	DEBITO	g[4]	u[2]	...AddMonths(-3)			...AddMonths(2)		

Código Fuente

LogicaNegocio

En Sistema

```
using LogicaNegocio.Ordenamiento;
```

```
namespace LogicaNegocio;

public class Sistema
{
    private List<Usuario> _usuarios;
    private List<Pago> _pagos;
    private List<Equipo> _equipos;
    private List<TipoDeGasto> _tiposDeGastos;
    private static Sistema _s_instancia;

    public List<Usuario> Usuarios
    {
        get { return new List<Usuario>(this._usuarios); }
    }

    public List<Equipo> Equipos
    {
        get { return new List<Equipo>(this._equipos); }
    }

    public List<Pago> Pagos
    {
        get { return new List<Pago>(this._pagos); }
    }

    public static Sistema Instancia
    {
        get
        {
            if (_s_instancia == null)
            {
                _s_instancia = new Sistema();
            }
            return _s_instancia;
        }
    }

    private Sistema()
    {
        _usuarios = new List<Usuario>();
        _equipos = new List<Equipo>();
        _pagos = new List<Pago>();
        _tiposDeGastos = new List<TipoDeGasto>();
        this.PrecargarEquipos();
    }
}
```

```
        this.PrecargarUsuarios();
        this.PrecargarTiposDeGastos();
        this.PrecargarPagos();
    }

    public void PrecargarUsuarios()
    {
        this.AltaUsuario(new Usuario("Ana", "Lopez", "seguro12", this._equipos[0],
DateTime.Now.AddDays(-400), Rol.EMPLEADO));
        this.AltaUsuario(new Usuario("Bruno", "Perez", "clave789", this._equipos[0],
DateTime.Now.AddDays(-300), Rol.EMPLEADO));
        this.AltaUsuario(new Usuario("Camila", "Ramos", "seguro12", this._equipos[1],
DateTime.Now.AddDays(-200), Rol.GERENTE));
        this.AltaUsuario(new Usuario("Diego", "Suarez", "xyz789ba0", this._equipos[1],
DateTime.Now.AddDays(-150), Rol.EMPLEADO));
        this.AltaUsuario(new Usuario("Elena", "Martinez", "elena212kj", this._equipos[2],
DateTime.Now.AddDays(-120), Rol.GERENTE));
        this.AltaUsuario(new Usuario("Federico", "Gomez", "fede456hajw", this._equipos[2],
DateTime.Now.AddDays(-100), Rol.EMPLEADO));
        this.AltaUsuario(new Usuario("Gabriela", "Diaz", "gaby789n", this._equipos[3],
DateTime.Now.AddDays(-90), Rol.GERENTE));
        this.AltaUsuario(new Usuario("Hector", "Torres", "htorres1", this._equipos[0],
DateTime.Now.AddDays(-80), Rol.EMPLEADO));
        this.AltaUsuario(new Usuario("Isabel", "Vega", "isavega7", this._equipos[0],
DateTime.Now.AddDays(-75), Rol.EMPLEADO));
        this.AltaUsuario(new Usuario("Javier", "Rosa", "javiro521", this._equipos[1],
DateTime.Now.AddDays(-60), Rol.EMPLEADO));
        this.AltaUsuario(new Usuario("Karen", "Nuñez", "karenn999", this._equipos[1],
DateTime.Now.AddDays(-50), Rol.EMPLEADO));
        this.AltaUsuario(new Usuario("Leonardo", "Paz", "leopaz666", this._equipos[2],
DateTime.Now.AddDays(-40), Rol.EMPLEADO));
        this.AltaUsuario(new Usuario("Maria", "Quintero", "marial010", this._equipos[2],
DateTime.Now.AddDays(-30), Rol.EMPLEADO));
        this.AltaUsuario(new Usuario("Nicolas", "Rivero", "nrivero8", this._equipos[3],
DateTime.Now.AddDays(-25), Rol.EMPLEADO));
        this.AltaUsuario(new Usuario("Olga", "Fernandez", "olga12345", this._equipos[0],
DateTime.Now.AddDays(-20), Rol.EMPLEADO));
        this.AltaUsuario(new Usuario("Pablo", "Lopez", "pablo2211", this._equipos[1],
DateTime.Now.AddDays(-15), Rol.EMPLEADO));
        this.AltaUsuario(new Usuario("Rocio", "Santos", "rocio33n1", this._equipos[1],
DateTime.Now.AddDays(-10), Rol.EMPLEADO));
        this.AltaUsuario(new Usuario("Sergio", "Herrera", "sergioa7", this._equipos[2],
DateTime.Now.AddDays(-8), Rol.EMPLEADO));
        this.AltaUsuario(new Usuario("Tamara", "Rey", "tammy44455", this._equipos[3],
DateTime.Now.AddDays(-5), Rol.EMPLEADO));
    }
```

```

        this.AltaUsuario(new Usuario("Ulises", "Roldan", "ulises101", this._equipos[3],
DateTime.Now.AddDays(-3), Rol.EMPLEADO));
        this.AltaUsuario(new Usuario("Valeria", "Mendez", "valm123jas", this._equipos[2],
DateTime.Now.AddDays(-2), Rol.EMPLEADO));
        this.AltaUsuario(new Usuario("Walter", "Iglesias", "walter9walter",
this._equipos[0], DateTime.Now, Rol.EMPLEADO));

    }

    public void PrecargarEquipos()
    {

        this._equipos.Add(new Equipo(1, "Desarrollo"));
        this._equipos.Add(new Equipo(2, "Marketing"));
        this._equipos.Add(new Equipo(3, "Finanzas"));
        this._equipos.Add(new Equipo(4, "Recursos Humanos"));

    }

    public void PrecargarPagos()
    {

        List<Usuario> u = this._usuarios;
        List<TipoDeGasto> g = this._tiposDeGastos;

        // ---- 17 Pagos Únicos ----
        this._pagos.Add(new Unico("Alquiler Octubre", 1200, MetodoDePago.CREDITO, g[0],
u[0], DateTime.Now.AddDays(-30), "REC001"));
        this._pagos.Add(new Unico("Factura de Luz", 300, MetodoDePago.DEBITO, g[1], u[2],
DateTime.Now.AddDays(-13), "REC002"));
        this._pagos.Add(new Unico("Campaña Facebook", 500, MetodoDePago.CREDITO, g[2],
u[3], DateTime.Now.AddDays(-20), "REC003"));
        this._pagos.Add(new Unico("Pago de Sueldo Elena", 2200, MetodoDePago.EFECTIVO,
g[3], u[4], DateTime.Now.AddDays(-15), "REC004"));
        this._pagos.Add(new Unico("Taxi reunión cliente", 80, MetodoDePago.DEBITO, g[4],
u[2], DateTime.Now.AddDays(-10), "REC005"));
        this._pagos.Add(new Unico("Comida equipo", 150, MetodoDePago.EFECTIVO, g[5], u[6],
DateTime.Now.AddDays(-5), "REC006"));
        this._pagos.Add(new Unico("Licencia Photoshop", 400, MetodoDePago.CREDITO, g[6],
u[7], DateTime.Now.AddDays(-45), "REC007"));
        this._pagos.Add(new Unico("Reparación PC", 250, MetodoDePago.DEBITO, g[7], u[8],
DateTime.Now.AddDays(-35), "REC008"));
        this._pagos.Add(new Unico("Evento Team Building", 1000, MetodoDePago.CREDITO, g[8],
u[9], DateTime.Now.AddDays(-60), "REC009"));
        this._pagos.Add(new Unico("Papel y útiles", 90, MetodoDePago.EFECTIVO, g[9], u[10],
DateTime.Now.AddDays(-7), "REC010"));
        this._pagos.Add(new Unico("Mantenimiento aire", 350, MetodoDePago.DEBITO, g[7],
u[11], DateTime.Now.AddDays(-40), "REC011"));

    }
}

```

```
        this._pagos.Add(new Unico("Publicidad Instagram", 270, MetodoDePago.CREDITO, g[2],  
u[12], DateTime.Now.AddDays(-12), "REC012"));  
        this._pagos.Add(new Unico("Snacks oficina", 50, MetodoDePago.EFECTIVO, g[5], u[13],  
DateTime.Now.AddDays(-3), "REC013"));  
        this._pagos.Add(new Unico("Taxi aeropuerto", 60, MetodoDePago.DEBITO, g[4], u[14],  
DateTime.Now.AddDays(-8), "REC014"));  
        this._pagos.Add(new Unico("Hosting web", 200, MetodoDePago.CREDITO, g[6], u[15],  
DateTime.Now.AddDays(-6), "REC015"));  
        this._pagos.Add(new Unico("Desinfección oficina", 180, MetodoDePago.EFECTIVO, g[7],  
u[16], DateTime.Now.AddDays(-4), "REC016"));  
        this._pagos.Add(new Unico("Cafetera nueva", 120, MetodoDePago.DEBITO, g[9], u[17],  
DateTime.Now.AddDays(-2), "REC017"));  
  
        // ---- 25 Pagos Recurrentes ----  
        // 5 totalmente pagados (fechaHasta ya vencida)  
        this._pagos.Add(new Recurrente("Internet mensual", 60, MetodoDePago.DEBITO, g[1],  
u[0], DateTime.Now.AddMonths(-6), DateTime.Now.AddMonths(-1)));  
        this._pagos.Add(new Recurrente("Servicio limpieza", 100, MetodoDePago.CREDITO,  
g[7], u[1], DateTime.Now.AddMonths(-5), DateTime.Now.AddMonths(-1)));  
        this._pagos.Add(new Recurrente("Netflix oficina", 15, MetodoDePago.CREDITO, g[6],  
u[2], DateTime.Now.AddMonths(-4), DateTime.Now.AddMonths(-1)));  
        this._pagos.Add(new Recurrente("Seguro empresa", 300, MetodoDePago.DEBITO, g[9],  
u[3], DateTime.Now.AddMonths(-12), DateTime.Now.AddMonths(-2)));  
        this._pagos.Add(new Recurrente("Plan telefonía", 80, MetodoDePago.EFECTIVO, g[1],  
u[4], DateTime.Now.AddMonths(-6), DateTime.Now.AddMonths(-1)));  
  
        // 20 activos (aún no vencen)  
        this._pagos.Add(new Recurrente("Alquiler mensual", 1200, MetodoDePago.CREDITO,  
g[0], u[5], DateTime.Now.AddMonths(-2), DateTime.Now.AddMonths(4)));  
        this._pagos.Add(new Recurrente("Electricidad", 250, MetodoDePago.DEBITO, g[1],  
u[6], DateTime.Now.AddMonths(-1), DateTime.Now.AddMonths(5)));  
        this._pagos.Add(new Recurrente("Spotify", 10, MetodoDePago.CREDITO, g[6], u[7],  
DateTime.Now.AddMonths(-3), DateTime.Now.AddMonths(3)));  
        this._pagos.Add(new Recurrente("Publicidad Google", 400, MetodoDePago.CREDITO,  
g[2], u[8], DateTime.Now.AddMonths(-2), DateTime.Now.AddMonths(6)));  
        this._pagos.Add(new Recurrente("Plan comida", 180, MetodoDePago.EFECTIVO, g[5],  
u[9], DateTime.Now.AddMonths(-1), DateTime.Now.AddMonths(5)));  
        this._pagos.Add(new Recurrente("Hosting servidor", 220, MetodoDePago.CREDITO, g[6],  
u[10], DateTime.Now.AddMonths(-2), DateTime.Now.AddMonths(6)));  
        this._pagos.Add(new Recurrente("Mantenimiento impresora", 90, MetodoDePago.DEBITO,  
g[7], u[11], DateTime.Now.AddMonths(-1), DateTime.Now.AddMonths(5)));  
        this._pagos.Add(new Recurrente("Capacitación equipo", 350, MetodoDePago.CREDITO,  
g[8], u[12], DateTime.Now.AddMonths(-1), DateTime.Now.AddMonths(3)));  
        this._pagos.Add(new Recurrente("Suscripción diseño", 45, MetodoDePago.CREDITO,  
g[6], u[13], DateTime.Now.AddMonths(-2), DateTime.Now.AddMonths(4)));
```

```

        this._pagos.Add(new Recurrente("Transporte interno", 100, MetodoDePago.EFECTIVO,
g[4], u[14], DateTime.Now.AddMonths(-1), DateTime.Now.AddMonths(3)));
        this._pagos.Add(new Recurrente("Snacks oficina", 60, MetodoDePago.DEBITO, g[5],
u[15], DateTime.Now.AddMonths(-2), DateTime.Now.AddMonths(5)));
        this._pagos.Add(new Recurrente("Publicidad mensual", 300, MetodoDePago.CREDITO,
g[2], u[16], DateTime.Now.AddMonths(-1), DateTime.Now.AddMonths(6)));
        this._pagos.Add(new Recurrente("Limpieza vidrios", 80, MetodoDePago.EFECTIVO, g[7],
u[17], DateTime.Now.AddMonths(-1), DateTime.Now.AddMonths(4)));
        this._pagos.Add(new Recurrente("Software contable", 250, MetodoDePago.CREDITO,
g[6], u[18], DateTime.Now.AddMonths(-2), DateTime.Now.AddMonths(5)));
        this._pagos.Add(new Recurrente("Asesoria legal", 500, MetodoDePago.DEBITO, g[9],
u[19], DateTime.Now.AddMonths(-3), DateTime.Now.AddMonths(3)));
        this._pagos.Add(new Recurrente("Mantenimiento ascensor", 120, MetodoDePago.CREDITO,
g[7], u[20], DateTime.Now.AddMonths(-1), DateTime.Now.AddMonths(5)));
        this._pagos.Add(new Recurrente("Plan de salud", 350, MetodoDePago.DEBITO, g[9],
u[21], DateTime.Now.AddMonths(-4), DateTime.Now.AddMonths(2)));
        this._pagos.Add(new Recurrente("Correo corporativo", 30, MetodoDePago.CREDITO,
g[6], u[0], DateTime.Now.AddMonths(-5), DateTime.Now.AddMonths(5)));
        this._pagos.Add(new Recurrente("Publicidad local", 150, MetodoDePago.EFECTIVO,
g[2], u[1], DateTime.Now.AddMonths(-2), DateTime.Now.AddMonths(3)));
        this._pagos.Add(new Recurrente("Combustible mensual", 200, MetodoDePago.DEBITO,
g[4], u[2], DateTime.Now.AddMonths(-3), DateTime.Now.AddMonths(2)));
    }
}

```

```

public void PrecargarTiposDeGastos()
{
    this._tiposDeGastos.Add(new TipoDeGasto("Alquiler", "Pago mensual del local u
oficina"));
    this._tiposDeGastos.Add(new TipoDeGasto("Servicios", "Luz, agua, internet, etc."));
    this._tiposDeGastos.Add(new TipoDeGasto("Marketing", "Publicidad y redes
sociales"));
    this._tiposDeGastos.Add(new TipoDeGasto("Salarios", "Pago de sueldos a
empleados"));
    this._tiposDeGastos.Add(new TipoDeGasto("Transporte", "Gastos de movilidad"));
    this._tiposDeGastos.Add(new TipoDeGasto("Comida", "Snacks y almuerzos de equipo"));
    this._tiposDeGastos.Add(new TipoDeGasto("Software", "Licencias y herramientas
digitales"));
    this._tiposDeGastos.Add(new TipoDeGasto("Mantenimiento", "Reparaciones o
limpieza"));
    this._tiposDeGastos.Add(new TipoDeGasto("Eventos", "Capacitaciones y eventos
corporativos"));
    this._tiposDeGastos.Add(new TipoDeGasto("Otros", "Gastos varios no
categorizados"));
}
public bool existeEmail(Usuario usuario)
{
    return this._usuarios.Any(u => u.Email == usuario.Email);
}

```

```
{  
    foreach (Usuario unUsuario in this._usuarios)  
    {  
        if (unUsuario.Email == usuario.Email)  
        {  
            return true;  
        }  
    }  
  
    return false;  
}  
  
public void AltaUsuario(Usuario usuario)  
{  
    usuario.Validar();  
    int contador = 1;  
    string email = usuario.GenerarEmail(false, contador);  
    usuario.Email = email;  
  
    //Mientras se encuentre una coincidencia, regeneramos el email  
    while (existeEmail(usuario))  
    {  
        email = usuario.GenerarEmail(true, contador);  
        usuario.Email = email;  
        contador++;  
    }  
  
    this._usuarios.Add(usuario);  
}  
  
public void AgregarPago(Pago pago)  
{  
    pago.Validar();  
    if (pago != null)  
    {  
        this._pagos.Add(pago);  
    }  
}  
  
public void AgregarTipoDeGasto(TipoDeGasto tipoDeGasto)  
{  
    tipoDeGasto.Validar();  
    if (tipoDeGasto != null)  
    {  
        this._tiposDeGastos.Add(tipoDeGasto);  
    }  
}
```

```
public List<Pago> ObtenerPagos()
{
    List<Pago> aRetornar = new List<Pago>();

    foreach (Pago pago in this._pagos)
    {
        aRetornar.Add(pago);
    }

    return aRetornar;
}

public List<Pago> ObtenerPagosDeUsuario(string email)
{
    List<Pago> aRetornar = new List<Pago>();

    foreach (Pago pago in this._pagos)
    {
        if (pago.Usuario.Email == email)
        {
            aRetornar.Add(pago);
        }
    }

    return aRetornar;
}

public List<Usuario> ObtenerListaUsuarios()
{
    List<Usuario> nuevaLista = new List<Usuario>();
    foreach (Usuario usuario in this._usuarios)
    {
        nuevaLista.Add(usuario);
    }

    return nuevaLista;
}

//Devuelve una lista nueva que contiene todos los equipos almacenados
public List<Equipo> ObtenerEquiposExistentes()
{
    List<Equipo> aRetornar = new List<Equipo>();
    foreach (Equipo equipo in this._equipos)
    {
        aRetornar.Add(equipo);
    }

    return aRetornar;
}
```

```

}

//Devuelve una nueva lista de usuarios que contiene los pertenecientes a determinado
equipo
public List<Usuario> ObtenerUsuariosPorEquipo(string nombreEquipo)
{
    List<Usuario> nuevaLista = new List<Usuario>();
    foreach (Usuario usuario in this._usuarios)
    {
        if (usuario.Equipo.Nombre.ToLower() == nombreEquipo.ToLower())
        {
            nuevaLista.Add(usuario);
        }
    }
    return nuevaLista;
}

//Se encarga de buscar y devolver el equipo cuyo nombre coincida con el recibido por
parametro
public Equipo ObtenerEquipoPorNombre(string nombreEquipo)
{
    foreach (Equipo equipo in this._equipos)
    {
        if (equipo.Nombre.ToLower() == nombreEquipo.ToLower())
        {
            return equipo;
        }
    }
    throw new Exception("No se encontro el equipo");
}

public List<Pago> ObtenerPagosMes(List<Pago> pagos)
{
    List<Pago> aRetornar = new List<Pago>();
    DateTime hoy = DateTime.Now;
    int añoActual = hoy.Year;
    int mesActual = hoy.Month;

    // Determinamos el primer y ultimo dia del mes actual
    DateTime inicioMes = new DateTime(añoActual, mesActual, 1);
    DateTime finMes = inicioMes.AddMonths(1).AddDays(-1);

    foreach (Pago pago in pagos)
    {
        if (pago is Recurrente recurrente)
        {
            if (recurrente.FechaDesde <= finMes &&

```

```

        (recurrente.FechaHasta >= inicioMes || recurrente.FechaHasta == null))
    {
        aRetornar.Add(pago);
    }
}

else if(pago is Unico unico)
{
    if (unico.FechaPago >= inicioMes && unico.FechaPago <= finMes)
    {
        aRetornar.Add(pago);
    }
}

return aRetornar;
}

public List<TipoDeGasto> ListarGastos()
{
    List<TipoDeGasto> aRetornar = new List<TipoDeGasto>();
    List<Pago> pagosMes = ObtenerPagosMes(this._pagos);

    foreach (Pago pago in pagosMes)
    {
        aRetornar.Add(pago.TipoDeGasto);
    }

    return aRetornar;
}

public Usuario BuscarUsuario(string email, string contrasena)
{
    Usuario usuario = null;
    foreach (Usuario u in this._usuarios)
    {
        if (u.Email == email && u.Contrasena == contrasena)
        {
            return u;
        }
    }
    return usuario;
}

public List<Pago> ObtenerPagosMesPorUsuario(string email)
{
    List<Pago> pagosMensuales = new List<Pago>(ObtenerPagosMes(this._pagos));
    pagosMensuales.RemoveAll(pago => !pago.Usuario.Email == email);
}

```

```

        List<Pago> aRetornar = new List<Pago>();
        foreach (Pago pago in pagosMensuales)
        {
            if (pago.Usuario.Email == email)
            {
                aRetornar.Add(pago);
            }
        }
        return aRetornar;
    }

    public double ObtenerTotalPagosMesPorUsuario(string email)
    {
        double total = 0;
        List<Pago> pagosMes = new List<Pago>(ObtenerPagosMesPorUsuario(email));
        foreach (Pago pago in pagosMes)
        {
            if (pago is Recurrente recurrente)
            {
                total += recurrente.CalcularCostoMensual();
            }else if (pago is Unico unico)
            {
                total += unico.CalcularCosto();
            }
        }
        return total;
    }

    public List<Pago> OrdenarPagosPorMontoDesc(List<Pago> pagos)
    {
        pagos.Sort(new PagoOrdenPorMonto());
        return pagos;
    }

    public List<Usuario> OrdenarUsuariosEquipoPorEmailAsc(List<Usuario> usuarios)
    {
        usuarios.Sort(new EquipoUsuariosPorEmailAsc());
        return usuarios;
    }

    public List<TipoDeGasto> ObtenerTiposDeGastos()
    {
        List<TipoDeGasto> aRetornar = new List<TipoDeGasto>();
        foreach (TipoDeGasto tipoDeGasto in this._tiposDeGastos)

```

```

    {
        aRetornar.Add(tipoDeGasto);
    }
    return aRetornar;
}

public TipoDeGasto ObtenerTipoDeGastoPorNombre(string nombre)
{
    foreach (TipoDeGasto tipo in this._tiposDeGastos)
    {
        if (tipo.Nombre.ToLower() == nombre.ToLower())
        {
            return tipo;
        }
    }

    return null;
}

public void EliminarTipoDeGastoPorNombre(string nombre)
{
    for (int i = 0; i < _tiposDeGastos.Count; i++)
    {
        if (_tiposDeGastos[i].Nombre == nombre)
        {
            _tiposDeGastos.RemoveAt(i);
            break;
        }
    }
}

public List<Pago> ObtenerPagosPorEquipo(string nombre)
{
    List<Pago> aRetornar = new List<Pago>();

    foreach (Pago pago in this._pagos)
    {
        Usuario usuario = pago.Usuario;
        if (usuario.Equipo.Nombre == nombre)
        {
            aRetornar.Add(pago);
        }
    }

    return aRetornar;
}

```

```
    }

    public Usuario ObtenerUsuarioPorEmail(string email)
    {
        Usuario aRetornar = null;

        foreach (Usuario usuario in this._usuarios)
        {
            if (usuario.Email == email)
            {
                aRetornar = usuario;
                break;
            }
        }

        return aRetornar;
    }

    public List<Pago> ObtenerPagosPorFecha(DateTime fecha)
    {
        List<Pago> aRetornar = new List<Pago>();

        foreach (Pago pago in this._pagos)
        {
            if (pago is Recurrente recurrente)
            {
                if (recurrente.FechaDesde.Month == fecha.Month &&
recurrente.FechaDesde.Year == fecha.Year)
                {
                    if (recurrente.FechaDesde.Month >= DateTime.Now.Month &&
recurrente.FechaHasta?.Month <= DateTime.Now.Month)
                    {
                        aRetornar.Add(pago);
                    }
                }
            }
            else if(pago is Unico unico)
            {
                if (unico.FechaPago.Month == fecha.Month && unico.FechaPago.Year ==
fecha.Year)
                {
                    aRetornar.Add(pago);
                }
            }
        }
    }
}
```

```
        return aRetornar;
```

```
}
```

```
}
```

En MetodoDePago

```
namespace LogicaNegocio;  
  
public enum MetodoDePago  
{  
    CREDITO = 1,  
    DEBITO = 2,  
    EFECTIVO = 3  
}
```

En Equipo

```
namespace LogicaNegocio;  
  
public class Equipo  
{  
    private int _id;  
    private string _nombre;  
  
    public string Nombre  
    {  
        get { return this._nombre; }  
    }  
    public Equipo(int id, string nombre)  
    {  
        this._id = id;  
        this._nombre = nombre;  
    }  
  
    public void Validar()  
    {  
        this.ValidarNombre();  
        this.ValidarId();  
    }  
}
```

```

    }

    private void ValidarNombre()
    {
        if (string.IsNullOrEmpty(this._nombre))
        {
            throw new Exception("El nombre del equipo no puede estar vacío");
        }
    }

    private void ValidarId()
    {
        if (this._id == null)
        {
            throw new Exception("Id inválido");
        }
    }
}

```

En Usuario

```

namespace LogicaNegocio;

public class Usuario
{
    private string _nombre;
    private string _apellido;
    private string _contrasena;
    private string _email;
    private Equipo _equipo;
    private DateTime _fechaIncorporacion;
    private static string _s_dominio = "@laEmpresa.com";
    private int _repetidos = 1;
    private Rol _rol;

    public string Nombre
    {
        get { return this._nombre; }
        set { this._nombre = value; }
    }

    public string Apellido
    {

```

```
        get { return this._apellido; }
        set { this._apellido = value; }
    }

    public string Contrasena
    {
        get { return this._contrasena; }
        set { this._contrasena = value; }
    }

    public string Email
    {
        get { return this._email; }
        set { this._email = value; }
    }

    public Equipo Equipo
    {
        get { return this._equipo; }
        set { this._equipo = value; }
    }

    public DateTime FechaIncorporacion
    {
        get { return this._fechaIncorporacion; }
        set { this._fechaIncorporacion = value; }
    }

    public int Repetidos
    {
        get { return this._repetidos; }
        set { this._repetidos = value; }
    }

    public Rol Rol
    {
        get { return this._rol; }
        set { this._rol = value; }
    }

    public Usuario(string nombre, string apellido, string contrasena, Equipo equipo,
                  DateTime fechaIncorporacion, Rol rol)
    {
        this._nombre = nombre;
        this._apellido = apellido;
        this._contrasena = contrasena;
        this._equipo = equipo;
        this._fechaIncorporacion = fechaIncorporacion;
        this._rol = rol;
    }
}
```

```
public void Validar()
{
    this.ValidarContrasena();
    this.ValidarEquipo();
    this.ValidarNombre();
    this.ValidarApellido();
}

private void ValidarEquipo()
{
    if (this._equipo == null)
    {
        throw new Exception("El usuario debe pertenecer a un equipo");
    }
}

private void ValidarNombre()
{
    if (string.IsNullOrWhiteSpace(this._nombre))
    {
        throw new Exception("Debe ingresar un nombre valido.");
    }
}

private void ValidarApellido()
{
    if (string.IsNullOrWhiteSpace(this._apellido))
    {
        throw new Exception("Debe ingresar un apellido valido.");
    }
}

private void ValidarContrasena()
{
    if (this.Contrasena.Length < 8)
    {
        throw new Exception("La contraseña debe tener al menos 8 caracteres");
    }
}

//Recibe 2 parametros. Dependiendo del valor de esRepetido, se genera un email con
//contador o sin el.
public string GenerarEmail(bool esRepetido, int contador)
{
    string nuevoNombre;
    string nuevoApellido;
    string emailARetornar = "";
    if (this._nombre.Length < 3)
    {
        nuevoNombre = this._nombre;
    }
}
```

```

        else
        {
            nuevoNombre = this._nombre.Substring(0, 3);
        }

        if (this._apellido.Length < 3)
        {
            nuevoApellido = this._apellido;
        }
        else
        {
            nuevoApellido = this._apellido.Substring(0, 3);
        }

        if (!esRepetido)
        {
            emailARetornar = $"{nuevoNombre}{nuevoApellido}{Usuario._s_dominio}";
        }
        else
        {
            emailARetornar = $"{nuevoNombre}{nuevoApellido}{contador}{Usuario._s_dominio}";
        }
    }

    return emailARetornar;
}

public override bool Equals(object? obj)
{
    if (obj == null || !(obj is Usuario))
    {
        return false;
    }

    Usuario user = obj as Usuario;
    return this.Email == user.Email;
}
}

```

En TipoDeGasto

```

namespace LogicaNegocio;

public class TipoDeGasto
{
    private string _nombre;

```

```

private string _descripcion;

public string Nombre
{
    get { return _nombre; }
}

public string Descripcion
{
    get { return _descripcion; }
    set { this._descripcion = value; }
}

public TipoDeGasto(string nombre, string descripcion)
{
    this._nombre = nombre;
    this._descripcion = descripcion;
}

public void Validar()
{
    this.ValidarNombre();
    this.ValidarDescripcion();
}

private void ValidarNombre()
{
    if (string.IsNullOrEmpty(this._nombre))
    {
        throw new Exception("El nombre no puede estar vacío");
    }
}

private void ValidarDescripcion()
{
    if (string.IsNullOrEmpty(this._descripcion))
    {
        throw new Exception("La descripción no puede estar vacía");
    }
}

```

En Pago

```
namespace LogicaNegocio;
```

```
public abstract class Pago
{
    private int _id;
    private string _descripcion;
    private double _monto;
    private MetodoDePago _metodoDePago;
    private TipoDeGasto _tipoDeGasto;
    private Usuario _usuario;
    private static int _s_ultimoId = 0;

    public int Id
    {
        get { return this._id; }
        set { this._id = value; }
    }

    public string Descripcion
    {
        get { return this._descripcion; }
        set { this._descripcion = value; }
    }

    public double Monto
    {
        get { return this._monto; }
        set { this._monto = value; }
    }

    public MetodoDePago MetodoDePago
    {
        get { return this._metodoDePago; }
        set { this._metodoDePago = value; }
    }

    public TipoDeGasto TipoDeGasto
    {
        get { return this._tipoDeGasto; }
        set { this._tipoDeGasto = value; }
    }

    public Usuario Usuario
    {
        get { return this._usuario; }
        set { this._usuario = value; }
    }

    public Pago(string descripcion, double monto, MetodoDePago metodoDePago, TipoDeGasto tipoDeGasto, Usuario usuario)
    {
        this._id = Pago._s_ultimoId++;
        this._descripcion = descripcion;
        this._monto = monto;
        this._metodoDePago = metodoDePago;
        this._tipoDeGasto = tipoDeGasto;
        this._usuario = usuario;
    }
}
```

```

public Pago()
{
    this._id = Pago._s_ultimoId++;
}

public abstract void Validar();

public virtual double CostoMostrar
{
    get
    {
        return Math.Round(CalcularCosto(),2);
    }
}
public abstract double CalcularCosto();
}

```

En Recurrente

```

namespace LogicaNegocio;

public class Recurrente : Pago
{
    private DateTime _fechaDesde;
    private DateTime? _fechaHasta;
    private int _mesesTotales;

    public DateTime FechaDesde
    {
        get { return this._fechaDesde; }
        set { this._fechaDesde = value; }
    }

    public DateTime? FechaHasta
    {
        get { return this._fechaHasta; }
        set { this._fechaHasta = value; }
    }

    public int MesesTotales
    {
        get { return this._mesesTotales; }
    }

    public Recurrente(string descripcion, double monto, MetodoDePago metodo, TipoDeGasto tipo, Usuario usuario, DateTime desde, DateTime? hasta)
        : base(descripcion, monto, metodo, tipo, usuario)
    {
    }
}

```

```
{  
    this._fechaDesde = desde;  
    this._fechaHasta = hasta;  
  
    if (_fechaHasta.HasValue)  
    {  
        //Value asegura que atributo FechaHasta no venga null, se almacena un entero  
        con la cantidad de meses totales por pagar cuota.  
        this._mesesTotales = ((FechaHasta.Value.Year - FechaDesde.Year) * 12) +  
                            (FechaHasta.Value.Month - FechaDesde.Month) + 1;  
    }  
    else  
    {  
        this._mesesTotales = 0;  
    }  
}  
  
public Recurrente(DateTime? hasta)  
{  
    this._fechaDesde = DateTime.Now;  
    this._fechaHasta = hasta;  
}  
  
public override void Validar()  
{  
    this.ValidarFechaHasta();  
    this.ValidarMonto();  
    this.ValidarDescripcion();  
}  
  
private void ValidarFechaHasta()  
{  
    if (this._fechaHasta < this._fechaDesde)  
    {  
        throw new Exception("Fecha inválida.");  
    }  
}  
  
private void ValidarMonto()  
{  
    if (this.Monto < 1)  
    {  
        throw new Exception("Monto inválido.");  
    }  
}
```

```
private void ValidarDescripcion()
{
    if (string.IsNullOrEmpty(this.Descripcion))
    {
        throw new Exception("Descripción inválida.");
    }
}

public int CalcularPagosPendientes()
{
    //0 pagos pendientes.
    if (this._fechaHasta.HasValue && DateTime.Now > this._fechaHasta)
    {
        return 0;
    }

    if (this._mesesTotales != 0)
    {
        int mesesPasados = ((DateTime.Now.Year - FechaDesde.Year) * 12) +
DateTime.Now.Month - FechaDesde.Month;
        return this._mesesTotales - mesesPasados;
    }

    return -1; // para hacer referencia que es indefinido.
}

public override double CostoMostrar => Math.Round(CalcularCostoMensual(),2);

public override double CalcularCosto()
{
    double recargo = this.Monto * CalcularRecargo();
    double totalConRecargo = this.Monto + recargo;

    if (this._mesesTotales == 0)
    {
        return totalConRecargo;
    }

    return (totalConRecargo) * this._mesesTotales;
}

public double CalcularCostoMensual()
{
    double recargo = this.Monto * CalcularRecargo();
    double totalConRecargo = this.Monto + recargo;
```

```

        if (this._mesesTotales == 0)
        {
            return totalConRecargo;
        }

        return totalConRecargo / this._mesesTotales;
    }

    public double CalcularRecargo()
    {
        double aRetornar = 0.03;
        if (this._fechaHasta != null)
        {
            if (this._mesesTotales > 10)
            {
                aRetornar = 0.10;
            }
            else if (this._mesesTotales >= 6 && this._mesesTotales <= 9)
            {
                aRetornar = 0.05;
            }
        }

        return aRetornar;
    }
}

```

En Unico

```

namespace LogicaNegocio;

public class Unico : Pago
{
    private DateTime _fechaPago;
    private string _numeroRecibo;

    public DateTime FechaPago
    {
        get { return _fechaPago; }
    }

    public Unico(string descripcion, double monto, MetodoDePago metodo, TipoDeGasto tipo,
    Usuario usuario, DateTime fechaPago, string numeroRecibo)
        : base(descripcion, monto, metodo, tipo, usuario)
    {
        this._fechaPago = fechaPago;
    }
}

```

```
        this._numeroRecibo = numeroRecibo;
    }

    public override void Validar()
    {
        this.ValidarFechaPago();
        this.ValidarNumeroRecibo();
        this.ValidarMonto();
        this.ValidarDescripcion();
    }

    private void ValidarFechaPago()
    {
        if (this._fechaPago.Day < DateTime.Now.Day)
        {
            throw new Exception("Fecha de pago inválida.");
        }
    }

    private void ValidarNumeroRecibo()
    {
        if (string.IsNullOrEmpty(this._numeroRecibo))
        {
            throw new Exception("Número de recibo inválido.");
        }
    }

    private void ValidarMonto()
    {
        if (this.Monto < 1)
        {
            throw new Exception("Monto inválido.");
        }
    }

    private void ValidarDescripcion()
    {
        if (string.IsNullOrEmpty(this.Descripcion))
        {
            throw new Exception("Descripción inválida.");
        }
    }

    public override double CalcularCosto()
    {
```

```

        double descuento = this.Monto * (1 - CalcularDescuento());
        return this.Monto - descuento;
    }

    public double CalcularDescuento()
    {
        if (this.MetodoDePago == MetodoDePago.EFECTIVO)
        {
            return 0.2;
        }
        return 0.1;
    }
}

```

En WebApp

Controllers

PagoControllers

```

using Microsoft.AspNetCore.Mvc;
using LogicaNegocio;
using WebApp.Filters;

namespace WebApp.Controllers;

[LoginFilter]
public class PagoController : Controller
{
    private Sistema sistema = Sistema.Instancia;

    public IActionResult Index()
    {
        string email = HttpContext.Session.GetString("Email");
        Usuario usuarioActual = sistema.ObtenerUsuarioPorEmail(email);

        if (HttpContext.Session.GetString("Rol") == Rol.EMPLEADO.ToString())
        {
            List<Pago> pagosMes = sistema.ObtenerPagosMesPorUsuario(email);
            pagosMes = sistema.OrdenarPagosPorMontoDesc(pagosMes);
            return View(pagosMes);
        }

        if (usuarioActual != null)
        {
            List<Pago> pagosEquipo = new
List<Pago>(sistema.ObtenerPagosPorEquipo(usuarioActual.Equipo.Nombre));

```

```

    pagosEquipo =
sistema.OrdenarPagosPorMontoDesc(sistema.ObtenerPagosMes( (pagosEquipo) ) );
        return View(pagosEquipo);
    }

    return View();
}

public IActionResult AgregarPago(string mensaje)
{
    if (mensaje != null && !string.IsNullOrEmpty(mensaje))
    {
        ViewBag.Mensaje = mensaje;
    }

    string email = HttpContext.Session.GetString("Email");
    Usuario usuario = sistema.ObtenerUsuarioPorEmail(email);
    ViewBag.Usuario = usuario;
    List<TipoDeGasto> tiposDeGastos = sistema.ObtenerTiposDeGastos();
    ViewBag.TiposDeGastos = tiposDeGastos;
    return View();
}

[HttpPost]
public IActionResult AgregarPago
(
    string Descripcion,
    double Monto,
    MetodoDePago MetodoDePago,
    string TipoPagoSeleccionado,
    string TipoDeGasto,
    DateTime FechaDesde,
    DateTime? FechaHasta,
    DateTime FechaPago,
    string NumeroRecibo)
{
    try
    {
        Usuario usuario =
sistema.ObtenerUsuarioPorEmail(HttpContext.Session.GetString("Email"));
        //Obtenemos el tipo de gasto por nombre
        TipoDeGasto encontrado = sistema.ObtenerTipoDeGastoPorNombre(TipoDeGasto);
        TipoDeGasto tipoDeGasto = new TipoDeGasto(encontrado.Nombre,
encontrado.Descripcion);

        Pago nuevoPago;

        if (TipoPagoSeleccionado == "RECURRENTE")
        {
            nuevoPago = new Recurrente(
                Descripcion,
                Monto,
                MetodoDePago,
                tipoDeGasto,
                usuario,
                FechaDesde,
                FechaHasta
            );
        }
    }
}

```

```

    } else
    {
        nuevoPago = new Unico(
            Descripcion,
            Monto,
            MetodoDePago,
            tipoDeGasto,
            usuario,
            FechaPago,
            NumeroRecibo
        );
    }

    sistema.AgregarPago(nuevoPago);
    TempData["Mensaje"] = "Pago cargado con éxito!";
    return RedirectToAction("Index");
}
catch (Exception e)
{
    return RedirectToAction("AgregarPago", new { mensaje = e.Message });
}
}

[GerenteFilter]
public IActionResult FiltrarPorFecha(DateTime Fecha)
{
    string email = HttpContext.Session.GetString("Email");
    Usuario usuarioActual = sistema.ObtenerUsuarioPorEmail(email);

    if (usuarioActual != null)
    {
        List<Pago> pagosFiltrados = new
List<Pago>(sistema.ObtenerPagosPorEquipo(usuarioActual.Equipo.Nombre));
        pagosFiltrados = sistema.ObtenerPagosPorFecha(Fecha);
        return View("Index", pagosFiltrados);
    }

    return View();
}
}

```

TipoDeGastoController

```

using LogicaNegocio;
using Microsoft.AspNetCore.Mvc;
using WebApp.Filters;

namespace WebApp.Controllers

[LoginFilter]
public class TipoDeGastoController : Controller
{
    private Sistema sistema = Sistema.Instancia;

```

```

public IActionResult Index()
{
    List<TipoDeGasto> tiposDeGastos = sistema.ObtenerTiposDeGastos();

    return View(tiposDeGastos);
}

[GerenteFilter]
public IActionResult AgregarTipoDeGasto()
{
    return View();
}

[HttpPost]
public IActionResult AgregarTipoDeGasto(string Nombre, string Descripcion)
{
    try
    {
        // Validar que no exista
        if (sistema.ObtenerTipoDeGastoPorNombre(Nombre) != null)
        {
            ViewBag.Mensaje = "Tipo de gasto ya existente";
            return View();
        }
        else
        {
            TipoDeGasto nuevo = new TipoDeGasto(Nombre, Descripcion);
            sistema.AgregarTipoDeGasto(nuevo);
            TempData["Mensaje"] = "Tipo de Gasto registrado con éxito";
        }
    }
    catch (Exception e)
    {
        throw new Exception(e.Message);
    }
    return RedirectToAction("Index");
}

[GerenteFilter]
public IActionResult EliminarTipoDeGasto(string Nombre)
{
    TipoDeGasto gasto = sistema.ObtenerTipoDeGastoPorNombre(Nombre);
    if (gasto != null)
    {
        return View(gasto);
    }

    return RedirectToAction("Index");
}

[HttpPost]
public IActionResult EliminarTipoDeGasto(string Nombre, string Descripcion)
{
    try
    {
        List<Pago> pagosARecorrer = new List<Pago>(sistema.ObtenerPagos());
        TipoDeGasto gasto = sistema.ObtenerTipoDeGastoPorNombre(Nombre);
    }
}

```

```

        foreach (Pago pago in pagosARcorrer)
        {
            if (pago.TipoDeGasto.Nombre == Nombre && pago.TipoDeGasto.Descripcion == Descripcion)
            {
                ViewBag.Mensaje = "No es posible eliminar este Tipo de Gasto.";
                return View(gasto);
            }
        }

        sistema.EliminarTipoDeGastoPorNombre(Nombre);
        TempData["Mensaje"] = "Tipo de Gasto eliminado correctamente.";
    }
    catch (Exception e)
    {
        throw new Exception(e.Message);
    }
    return RedirectToAction("Index");
}
}

```

UsuarioController

```

using Microsoft.AspNetCore.Mvc;
using LogicaNegocio;
using WebApp.Filters;

namespace WebApp.Controllers;

public class UsuarioController : Controller
{
    private Sistema sistema = Sistema.Instancia;

    public IActionResult Login()
    {
        return View();
    }

    [HttpPost]
    public IActionResult Login(string Email, string Contrasena)
    {
        try
        {
            Usuario usuario = sistema.BuscarUsuario(Email, Contrasena);
            if (usuario != null)
            {
                HttpContext.Session.SetString("Email", usuario.Email);
                HttpContext.Session.SetString("Rol", usuario.Rol.ToString());
                TempData["Mensaje"] = "Logueado con éxito";
                return RedirectToAction("Perfil");
            }
            else
            {
                ViewBag.Mensaje = "Datos incorrectos";
            }
        }
        catch (Exception ex)
    }
}

```

```

    {
        ViewBag.Mensaje = ex.Message;
    }

    return View();
}

[LoginFilter]
public IActionResult Logout()
{
    HttpContext.Session.Clear();
    return RedirectToAction("Login");
}

[LoginFilter]
public IActionResult Perfil()
{
    string email = HttpContext.Session.GetString("Email");

    if (email == null) return RedirectToAction("Login");

    Usuario usuario = sistema.ObtenerUsuarioPorEmail(email);

    List<Usuario> usuariosEquipo = sistema.ObtenerUsuariosPorEquipo(usuario.Equipo.Nombre);
    usuariosEquipo = sistema.OrdenarUsuariosEquipoPorEmailAsc(usuariosEquipo);
    ViewBag.Usuarios = usuariosEquipo;

    double totalGastadoMes = sistema.ObtenerTotalPagosMesPorUsuario(usuario.Email);
    ViewBag.TotalGastadoMes = totalGastadoMes;
    return View(usuario);
}
}

```

Filters

GerenteFilter

```

using LogicaNegocio;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Filters;

namespace WebApp.Filters;

public class GerenteFilter : ActionFilterAttribute
{
    public override void OnActionExecuting(ActionExecutingContext context)
    {
        string rol = context.HttpContext.Session.GetString("Rol");
        if (rol != Rol.GERENTE.ToString())
        {
            context.Result = new RedirectToActionResult("Perfil", "Usuario", new { error =
"Permisos Insuficientes" });
        }
        base.OnActionExecuting(context);
    }
}

```

```
}
```

LoginFilter

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Filters;

namespace WebApp.Filters;

public class LoginFilter : ActionFilterAttribute
{
    public override void OnActionExecuting(ActionExecutingContext context)
    {
        string? email = context.HttpContext.Session.GetString("Email");
        if (email == null)
        {
            context.Result = new RedirectToActionResult("Login", "Usuario", new { error = "Debe iniciar sesión" });
        }
        base.OnActionExecuting(context);
    }
}
```

Views

Pago

AgregarPago

```
@using LogicaNegocio
@model LogicaNegocio.Pago



@if (ViewBag.Mensaje != null)
{
    <span class="alert alert-danger text-center">@ViewBag.Mensaje</span>
}

<h2>Aregar Pago</h2>

<form method="post" class="d-flex flex-column gap-2 mb-4 w-100 p-4">
    <div>
        <label class="form-label">Tipo de pago</label>
        <select name="TipoPagoSeleccionado" class="form-control">
            <option value="RECURRENTE">RECURRENTE</option>
            <option value="UNICO">UNICO</option>
        </select>
    </div>
    <div class="mb-3">
        <label for="Descripcion" class="form-label">Descripcion</label>
        <input type="text" class="form-control" name="Descripcion" placeholder="Ej: Wi-Fi para las oficinas" required>
    </div>


```

```

        </div>
        <div class="mb-3">
            <label for="Monto" class="form-label">Monto</label>
            <input type="number" class="form-control" name="Monto" min="1" placeholder="Ej: 2200" required>
        </div>
        <div class="mb-3">
            <label class="form-label" for="MetodoDePago"> Método de pago</label>
            <select class="form-control" name="MetodoDePago">
                <option value="@MetodoDePago.CREDITO">CREDITO</option>
                <option value="@MetodoDePago.DEBITO">DEBITO</option>
                <option value="@MetodoDePago.EFECTIVO">EFECTIVO</option>
            </select>
        </div>
        <div class="mb-3">
            <label for="TipoDeGasto" class="form-label">Tipo de Gasto</label>
            <select name="TipoDeGasto" class="form-control">
                @foreach (TipoDeGasto tGasto in ViewBag.TiposDeGastos)
                {
                    <option value="@tGasto.Nombre">@tGasto.Nombre - 
<span>@tGasto.Descripcion</span></option>
                }
            </select>
        </div>
        <div>
            <label for="Usuario" class="form-label">Usuario</label>
            <input type="text" class="form-control" name="Usuario" value="@ViewBag.Usuario" readonly>
        </div>

        <div id="campos-recurrente">
            <div class="mb-3">
                <label for="FechaDesde" class="form-label">Fecha Desde</label>
                <input type="date" class="form-control" name="FechaDesde" value="@DateTime.Now.ToString("yyyy-MM-dd")" readonly>
            </div>

            <div class="mb-3">
                <label for="FechaHasta" class="form-label">Fecha Hasta</label>
                <input type="date" class="form-control" name="FechaHasta">
            </div>
        </div>

        <div id="campos-unico" style="display:none;">
            <div class="mb-3">
                <label for="FechaPago" class="form-label">Fecha de Pago</label>
                <input type="date" class="form-control" name="FechaPago" value="@DateTime.Now.ToString("yyyy-MM-dd")" readonly>
            </div>

            <div class="mb-3">
                <label for="NumeroRecibo" class="form-label">Número de Recibo</label>
                <input type="text" class="form-control" name="NumeroRecibo">
            </div>
        </div>

        <script>
document.querySelector("select[name='TipoPagoSeleccionado']")

```

```

    .addEventListener("change", function() {
        const camposUnico = document.getElementById("campos-unico");
        const camposRecurrente = document.getElementById("campos-recurrente");

        const inputNumeroRecibo = camposUnico.querySelector("input[name='NumeroRecibo']");

        if (this.value === "UNICO") {
            camposUnico.style.display = "block";
            inputNumeroRecibo.setAttribute("required", "required");
            camposRecurrente.style.display = "none";
        } else {
            camposUnico.style.display = "none";
            inputNumeroRecibo.removeAttribute("required");
            camposRecurrente.style.display = "block";
        }
    });
</script>

<button type="submit" class="btn btn-primary">Agregar</button>
</form>
</div>

```

Index

```

@using LogicaNegocio
@model IEnumerable<Pago>

<div class="d-flex flex-column gap-2 p-2">
    @{
        <h1>
            @(Context.Session.GetString("Rol") == Rol.GERENTE.ToString() ? "Pagos" : "Mis pagos
mensuales")
        </h1>
    }
    @if (TempData["Mensaje"] != null)
    {
        <span class="alert alert-info text-center">@TempData["Mensaje"]</span>
    }
    @if (Model != null && Model.Count() > 0)
    {
        <table class="table table-dark table-hover">
            <thead>
                @if (Context.Session.GetString("Rol") == Rol.GERENTE.ToString())
                {
                    <tr>
                        <td colspan="4">
                            <form action="/Pago/FiltrarPorFecha" class="d-flex justify-content-end
align-items-center gap-2">
                                <div class="form-label m-0">
                                    <input type="date" name="Fecha" class="form-control " required/>
                                </div>
                                <input type="submit" value="Filtrar" class="btn btn-primary"/>
                            </form>
                        </td>
                    </tr>
                }
            </thead>
            <tbody>
                @foreach (var pago in Model)
                {
                    <tr>
                        <td>@pago.Id</td>
                        <td>@pago.Fecha</td>
                        <td>@pago.Monto</td>
                        <td>@pago.Status</td>
                    </tr>
                }
            </tbody>
        </table>
    }
</div>

```

```

        }
    <tr>
        <th>Id</th>
        <th>Descripcion</th>
        <th>Usuario</th>
        <th>Costo</th>
    </tr>
</thead>
<tbody>
@foreach (Pago pago in Model)
{
    <tr>
        <td>@pago.Id</td>
        <td>@pago.Descripcion</td>
        <td>@pago.Usuario.Email</td>
        <td>$@pago.CostoMostrar</td>
    </tr>
}
</tbody>
</table>
<a href="~/Pago/AgregarPago"><input type="button" value="Cargar Pago" class="btn btn-primary"/> </a>
}
else
{
    <span class="alert alert-info text-center mt-2">No existen pagos para la fecha seleccionada...</span>
}
</div>

```

Shared

Layout

```

@using LogicaNegocio
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8"/>
    <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
    <title>@ ViewData["Title"] - Gestor Pagos V1</title>
    <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css"/>
    <link rel="stylesheet" href="~/css/site.css" asp-append-version="true"/>
    <link rel="stylesheet" href="~/WebApp.styles.css" asp-append-version="true"/>
</head>
<body>
<header>
    <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-white border-bottom box-shadow mb-3">
        <div class="container-fluid">
            @if (Context.Session.GetString("Email") == null)
            {
                <a class="navbar-brand" asp-area="" asp-controller="Usuario" asp-action="Login">Gestor Pagos V1</a>
            }
            else

```

```

    {
        <a class="navbar-brand" asp-area="" asp-controller="Usuario"
asp-action="Perfil">Gestor Pagos V1</a>
    }
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
data-bs-target=".navbar-collapse" aria-controls="navbarSupportedContent"
        aria-expanded="false" aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
    </button>
    <div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">
        <ul class="navbar-nav flex-grow-1">
            @if (Context.Session.GetString("Email") == null)
            {
                <li class="nav-item">
                    <a class="nav-link text-dark" asp-area="" asp-controller="Usuario"
asp-action="Login">Login</a>
                </li>
            }
            else if (Context.Session.GetString("Email") != null &&
Context.Session.GetString("Rol") == Rol.GERENTE.ToString())
            {
                <li class="nav-item">
                    <a class="nav-link text-dark" asp-area="" asp-controller="Usuario"
asp-action="Perfil">Perfil</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link text-dark" asp-area="" asp-controller="Pago"
asp-action="Index">Pagos</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link text-dark" asp-area="" asp-controller="TipoDeGasto"
asp-action="Index">Tipo de Gastos</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link text-dark" asp-area="" asp-controller="Usuario"
asp-action="Logout">Logout</a>
                </li>
            }
            else
            {
                <li class="nav-item">
                    <a class="nav-link text-dark" asp-area="" asp-controller="Usuario"
asp-action="Perfil">Perfil</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link text-dark" asp-area="" asp-controller="Pago"
asp-action="Index">Pagos mensuales</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link text-dark" asp-area="" asp-controller="Usuario"
asp-action="Logout">Logout</a>
                </li>
            }
        </ul>
    </div>
</div>
</nav>
</header>

```

```

<div class="container">
    <main role="main" class="pb-3">
        @RenderBody()
    </main>
</div>

<footer class="border-top footer text-muted">
    <div class="container text-center">
        &copy; 2025 - Gestor Pagos V1 - <a href="https://github.com/facundoms284/gestion-de-pagos" target="_blank" class="text-decoration-none">GitHub</a>
    </div>
</footer>
<script src="~/lib/jquery/dist/jquery.min.js"></script>
<script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
<script src="~/js/site.js" asp-append-version="true"></script>
@await RenderSectionAsync("Scripts", required: false)
</body>
</html>

```

TipoDeGasto

AgregarTipoDeGasto

```

@using LogicaNegocio
@model LogicaNegocio.TipoDeGasto

<div class="d-flex flex-column gap-2 w-60 justify-content-center align-items-center border p-4">
    <h2>Agregar Gasto</h2>

    <form method="post" class="d-flex flex-column gap-2 mb-4 p-4 w-100">
        <div>
            <label for="Nombre" class="form-label">Nombre</label>
            <input type="text" class="form-control" name="Nombre" required>
        </div>
        <div class="mb-3">
            <label for="Descripcion" class="form-label">Descripcion</label>
            <input type="text" class="form-control" name="Descripcion" required>
        </div>

        <button type="submit" class="btn btn-primary">Agregar</button>
    </form>
    @if (ViewBag.Mensaje != null)
    {
        <span class="alert alert-danger text-center">@ViewBag.Mensaje</span>
    }
</div>

```

EliminarTipoDeGasto

```

@using LogicaNegocio
@model LogicaNegocio.TipoDeGasto

<div class="d-flex flex-column gap-2 w-60 justify-content-center align-items-center border p-4">

```

```

<h2>Eliminar Gasto</h2>

<form method="post" class="d-flex flex-column gap-2 p-4 w-100">
    <div>
        <label for="Nombre" class="form-label">Nombre</label>
        <input type="text" class="form-control" name="Nombre" value="@Model.Nombre"
readonly>
    </div>
    <div>
        <label for="Descripcion" class="form-label">Descripcion</label>
        <input type="text" class="form-control" name="Descripcion"
value="@Model.Descripcion" readonly>
    </div>
    <button type="submit" class="btn btn-danger">Eliminar</button>
</form>

@if (ViewBag.Mensaje != null)
{
    <span class="alert alert-danger text-center">@ViewBag.Mensaje</span>
}
</div>

```

Index

```

@using LogicaNegocio;
@model IEnumerable<TipoDeGasto>
<div class="d-flex flex-column gap-2">
    @if (TempData["Mensaje"] != null)
    {
        <span class="alert alert-info m-2 text-center">
            @ TempData["Mensaje"]
        </span>
    }
    <h1>Tipos de gastos</h1>

    <table class="table table-dark table-hover">
        <thead>
            <tr>
                <th>Nombre</th>
                <th>Descripción</th>
                <th></th>
            </tr>
        </thead>
        <tbody>
            @foreach (TipoDeGasto tipoDeGasto in Model)
            {
                <tr>
                    <td>@tipoDeGasto.Nombre</td>
                    <td>@tipoDeGasto.Descripcion</td>
                    <td class="mt-4">
                        <a asp-controller="TipoDeGasto"
                            asp-action="EliminarTipoDeGasto"
                            asp-route-Nombre="@tipoDeGasto.Nombre">
                            <input type="button" value="Eliminar" class="btn btn-danger"/>
                        </a>
                    </td>
                </tr>
            }
        </tbody>
    </table>
</div>

```

```

        }
    </tbody>
</table>

<a href=~~/TipoDeGasto/AgregarTipoDeGasto><input type="button" value="Cargar Gasto" class="btn btn-primary"/> </a>
</div>

```

Usuario

Login

```

@{
    ViewData["Title"] = "Login";
}

<div class="d-flex justify-content-center align-items-center">
    <div class="w-100 p-4 border" style="max-width: 50%;">
        <h1 class="text-center">Login</h1>
        @if (!string.IsNullOrEmpty(ViewBag.Mensaje))
        {
            <div class="alert alert-danger text-center">
                @ViewBag.Mensaje
            </div>
        }

        <form method="post">
            <div class="mb-3">
                <label>Email</label>
                <input type="text" class="form-control" name="Email" required>
            </div>
            <div class="mb-3">
                <label>Contraseña</label>
                <input type="password" class="form-control" name="Contrasena" required>
            </div>
            <button type="submit" class="btn btn-primary">Login</button>
        </form>
    </div>
</div>

```

Perfil

```

@using LogicaNegocio
@model LogicaNegocio.Usuario

@{
    ViewData["Title"] = "Perfil";
}

<div class="d-flex flex-column gap-2">

    @if (TempData["Mensaje"] != null)
    {
        <span class="alert alert-info text-center">
            @ TempData["Mensaje"]
    
```

```

        </span>
    }

<h1>Perfil</h1>



| Rol        | Nombre        | Apellido        | Email        | Fecha Incorporación       | Equipo               | Total gastado este mes     |
|------------|---------------|-----------------|--------------|---------------------------|----------------------|----------------------------|
| @Model.Rol | @Model.Nombre | @Model.Apellido | @Model.Email | @Model.FechaIncorporacion | @Model.Equipo.Nombre | \$@ViewBag.TotalGastadoMes |



@if (Context.Session.GetString("Rol") == Rol.GERENTE.ToString())
{
    <h2>Usuarios del equipo</h2>
    List<Usuario> usuariosEquipo = ViewBag.Usuarios;

    @if (usuariosEquipo != null && usuariosEquipo.Count > 0)
    {
        <table class="table table-dark table-hover">
            <thead>
                <tr>
                    <th>Email</th>
                    <th>Nombre Completo</th>
                    <th>Fecha Incorporación</th>
                </tr>
            </thead>
            <tbody>
                @foreach (Usuario usuario in usuariosEquipo)
                {
                    <tr>
                        <td>@usuario.Email</td>
                        <td>@usuario.Nombre @usuario.Apellido</td>
                        <td>@usuario.FechaIncorporacion.ToString("dd/MM/yyyy")</td>
                    </tr>
                }
            </tbody>
        </table>
    }
    else
    {
}
}

```

```
        <p>No hay otros usuarios en este equipo registrados.</p>
    }
</div>
```

Evidencias IAG

En base a los siguientes constructores en C#, creamos una precarga de:
Usuarios: 22
Grupos: 4
Gastos: 10
Pagos: 42 pagos totales, 25 recurrentes (5 de ellos deben estar pagados totalmente)

En cada uno de ellos, te dejo un ejemplo de como quiero que quede hecho.

```
public Equipo(int id, string nombre)
{
    this._id = id;
    this._nombre = nombre;
}
this._equipos.Add(new Equipo(1, "Desarrollo"));

public Pago(string descripcion, double monto, MetodoDePago
metodoDePago, TipoDeGasto tipoDeGasto, Usuario usuario)
{
    this._id = Pago._s_ultimoid;
    this._descripcion = descripcion;
    this._monto = monto;
    this._metodoDePago = metodoDePago;
    this._tipoDeGasto = tipoDeGasto;
    this._usuario = usuario;
}
this._pagos.Add(new Unico("Alquiler Octubre", 1200,
MetodoDePago.CREDITO, g[0], u[0], DateTime.Now.AddDays(-30),
"REC001"));

public enum MetodoDePago
{
    CREDITO = 1,
    DEBITO = 2,
    EFECTIVO = 3
} public Recurrente(string descripcion, double monto, MetodoDePago
metodo, TipoDeGasto tipo, Usuario usuario, DateTime desde, DateTime
hasta)
: base(descripcion, monto, metodo, tipo, usuario)
{
    this._fechaDesde = desde;
    this._fechaHasta = hasta;
} public Unico(string descripcion, double monto, MetodoDePago
metodo, TipoDeGasto tipo, Usuario usuario, DateTime fechaPago,
string numeroRecibo)
: base(descripcion, monto, metodo, tipo, usuario)
{
    this._fechaPago = fechaPago;
```

```

        this._fechaPago = fechaPago;
        this._numeroRecibo = numeroRecibo;
    } public TipoDeGasto(string nombre, string descripcion)
    {
        this._nombre = nombre;
        this._descripcion = descripcion;
    } public Usuario(string nombre, string apellido, string contrasena,
Equipo equipo,
        DateTime fechalIncorporacion)
    {
        this._nombre = nombre;
        this._apellido = apellido;
        this._contrasena = contrasena;
        this._equipo = equipo;
        this._fechalIncorporacion = fechalIncorporacion;
    }

AltaUsuario(new Usuario("Ana", "Lopez", "pass123", this._equipos[0],
DateTime.Now));

```

La contraseña de usuario debe tener al menos 6 caracteres.

Perfecto ✓

Vamos a generar una **precarga completa y coherente** con las cantidades y tipos que pedís, siguiendo exactamente el formato de tus ejemplos.

Tendremos:

- **4 grupos (Equipo)**
- **22 usuarios**
- **10 tipos de gasto**
- **42 pagos totales**
 - 25 recurrentes
 - 17 únicos
 - De los recurrentes, 5 estarán **totalmente pagados** (su `fechaHasta` ya venció).

 **Precarga sugerida** (lista lista para pegar en tu método `PrecargarDatos()` o similar):

csharp

 Copiar código

```

// =====
// EQUIPOS (Grupos)
// =====
this._equipos.Add(new Equipo(1, "Desarrollo"));
this._equipos.Add(new Equipo(2, "Marketing"));
this._equipos.Add(new Equipo(3, "Finanzas"));
this._equipos.Add(new Equipo(4, "Recursos Humanos"));

```

```

// =====
// USUARIOS (22)
// =====

this.AltaUsuario(new Usuario("Ana", "Lopez", "pass123", this._equipos[0], DateTime.Now.AddDays(-46));
this.AltaUsuario(new Usuario("Bruno", "Perez", "clave789", this._equipos[0], DateTime.Now.AddDays(0));
this.AltaUsuario(new Usuario("Camila", "Ramos", "abc1234", this._equipos[1], DateTime.Now.AddDays(0));
this.AltaUsuario(new Usuario("Diego", "Suarez", "xyz7890", this._equipos[1], DateTime.Now.AddDays(0));
this.AltaUsuario(new Usuario("Elena", "Martinez", "elena22", this._equipos[2], DateTime.Now.AddDays(0));
this.AltaUsuario(new Usuario("Federico", "Gomez", "fede456", this._equipos[2], DateTime.Now.AddDays(0));
this.AltaUsuario(new Usuario("Gabriela", "Diaz", "gaby789", this._equipos[3], DateTime.Now.AddDays(0));
this.AltaUsuario(new Usuario("Hector", "Torres", "htorres1", this._equipos[0], DateTime.Now.AddDays(0));
this.AltaUsuario(new Usuario("Isabel", "Vega", "isavega7", this._equipos[0], DateTime.Now.AddDays(0));
this.AltaUsuario(new Usuario("Javier", "Rosa", "javiro5", this._equipos[1], DateTime.Now.AddDays(0));
this.AltaUsuario(new Usuario("Karen", "Nuñez", "karenn9", this._equipos[1], DateTime.Now.AddDays(0));
this.AltaUsuario(new Usuario("Leonardo", "Paz", "leopaz6", this._equipos[2], DateTime.Now.AddDays(0));
this.AltaUsuario(new Usuario("Maria", "Quintero", "maria10", this._equipos[2], DateTime.Now.AddDays(0));
this.AltaUsuario(new Usuario("Nicolas", "Rivero", "nrivero8", this._equipos[3], DateTime.Now.AddDays(0));
this.AltaUsuario(new Usuario("Olga", "Fernandez", "olga123", this._equipos[0], DateTime.Now.AddDays(0));
this.AltaUsuario(new Usuario("Pablo", "Lopez", "pablo22", this._equipos[1], DateTime.Now.AddDays(0));
this.AltaUsuario(new Usuario("Rocio", "Santos", "rocio33", this._equipos[1], DateTime.Now.AddDays(0));
this.AltaUsuario(new Usuario("Sergio", "Herrera", "sergio7", this._equipos[2], DateTime.Now.AddDays(0));
this.AltaUsuario(new Usuario("Tamara", "Rey", "tammy44", this._equipos[3], DateTime.Now.AddDays(-5));
this.AltaUsuario(new Usuario("Ulises", "Roldan", "ulises1", this._equipos[3], DateTime.Now.AddDays(0));
this.AltaUsuario(new Usuario("Valeria", "Mendez", "valm123", this._equipos[2], DateTime.Now.AddDays(0));
this.AltaUsuario(new Usuario("Walter", "Iglesias", "walter9", this._equipos[0], DateTime.Now));

// =====
// TIPOS DE GASTO (10)
// =====

this._tiposDeGasto.Add(new TipoDeGasto("Alquiler", "Pago mensual del local u oficina"));
this._tiposDeGasto.Add(new TipoDeGasto("Servicios", "Luz, agua, internet, etc."));
this._tiposDeGasto.Add(new TipoDeGasto("Marketing", "Publicidad y redes sociales"));
this._tiposDeGasto.Add(new TipoDeGasto("Salarios", "Pago de sueldos a empleados"));
this._tiposDeGasto.Add(new TipoDeGasto("Transporte", "Gastos de movilidad"));
this._tiposDeGasto.Add(new TipoDeGasto("Comida", "Snacks y almuerzos de equipo"));
this._tiposDeGasto.Add(new TipoDeGasto("Software", "Licencias y herramientas digitales"));
this._tiposDeGasto.Add(new TipoDeGasto("Mantenimiento", "Reparaciones o limpieza"));
this._tiposDeGasto.Add(new TipoDeGasto("Eventos", "Capacitaciones y eventos corporativos"));
this._tiposDeGasto.Add(new TipoDeGasto("Otros", "Gastos varios no categorizados"));

// =====
// PAGOS (42 totales)
// =====

List<Usuario> u = this._usuarios;
List<TipoDeGasto> g = this._tiposDeGasto;

```

```

// ---- 17 Pagos Únicos ----
this._pagos.Add(new Unico("Alquiler Octubre", 1200, MetodoDePago.CREDITO, g[0], u[0], DateTime.Now
this._pagos.Add(new Unico("Factura de Luz", 300, MetodoDePago.DEBITO, g[1], u[2], DateTime.Now.Add
this._pagos.Add(new Unico("Campaña Facebook", 500, MetodoDePago.CREDITO, g[2], u[3], DateTime.Now.
this._pagos.Add(new Unico("Pago de Sueldo Elena", 2200, MetodoDePago.EFECTIVO, g[3], u[4], DateTim
this._pagos.Add(new Unico("Taxi reunión cliente", 80, MetodoDePago.DEBITO, g[4], u[5], DateTime.Ne
this._pagos.Add(new Unico("Comida equipo", 150, MetodoDePago.EFECTIVO, g[5], u[6], DateTime.Now.Ad
this._pagos.Add(new Unico("Licencia Photoshop", 400, MetodoDePago.CREDITO, g[6], u[7], DateTime.Ne
this._pagos.Add(new Unico("Reparación PC", 250, MetodoDePago.DEBITO, g[7], u[8], DateTime.Now.AddD
this._pagos.Add(new Unico("Evento Team Building", 1000, MetodoDePago.CREDITO, g[8], u[9], DateTime
this._pagos.Add(new Unico("Papel y útiles", 90, MetodoDePago.EFECTIVO, g[9], u[10], DateTime.Now.A
this._pagos.Add(new Unico("Mantenimiento aire", 350, MetodoDePago.DEBITO, g[10], u[11], DateTime.Ne
this._pagos.Add(new Unico("Publicidad Instagram", 270, MetodoDePago.CREDITO, g[12], u[12], DateTime
this._pagos.Add(new Unico("Snacks oficina", 50, MetodoDePago.EFECTIVO, g[13], u[13], DateTime.Now.A
this._pagos.Add(new Unico("Taxi aeropuerto", 60, MetodoDePago.DEBITO, g[14], u[14], DateTime.Now.Ad
this._pagos.Add(new Unico("Hosting web", 200, MetodoDePago.CREDITO, g[15], u[15], DateTime.Now.AddD
this._pagos.Add(new Unico("Desinfección oficina", 180, MetodoDePago.EFECTIVO, g[16], u[16], DateTim
this._pagos.Add(new Unico("Cafetera nueva", 120, MetodoDePago.DEBITO, g[17], u[17], DateTime.Now.Ad

// ---- 25 Pagos Recurrentes ----
// 5 totalmente pagados (fechaHasta ya vencida)
this._pagos.Add(new Recurrente("Internet mensual", 60, MetodoDePago.DEBITO, g[1], u[0], DateTime.N
this._pagos.Add(new Recurrente("Servicio limpieza", 100, MetodoDePago.CREDITO, g[2], u[1], DateTim
this._pagos.Add(new Recurrente("Netflix oficina", 15, MetodoDePago.CREDITO, g[3], u[2], DateTime.N
this._pagos.Add(new Recurrente("Seguro empresa", 300, MetodoDePago.DEBITO, g[4], u[3], DateTime.Ne
this._pagos.Add(new Recurrente("Plan telefonía", 80, MetodoDePago.EFECTIVO, g[5], u[4], DateTime.N

// 20 activos (aún no vencen)
this._pagos.Add(new Recurrente("Alquiler mensual", 1200, MetodoDePago.CREDITO, g[6], u[5], DateTim
this._pagos.Add(new Recurrente("Electricidad", 250, MetodoDePago.DEBITO, g[7], u[6], DateTime.Now.
this._pagos.Add(new Recurrente("Spotify", 10, MetodoDePago.CREDITO, g[8], u[7], DateTime.Now.AddMc
this._pagos.Add(new Recurrente("Publicidad Google", 400, MetodoDePago.CREDITO, g[9], u[8], DateTim
this._pagos.Add(new Recurrente("Plan comida", 180, MetodoDePago.EFECTIVO, g[10], u[9], DateTime.Now
this._pagos.Add(new Recurrente("Hosting servidor", 220, MetodoDePago.CREDITO, g[11], u[10], DateTim
this._pagos.Add(new Recurrente("Mantenimiento impresora", 90, MetodoDePago.DEBITO, g[12], u[11], Da
this._pagos.Add(new Recurrente("Capacitación equipo", 350, MetodoDePago.CREDITO, g[13], u[12], Date
this._pagos.Add(new Recurrente("Suscripción diseño", 45, MetodoDePago.CREDITO, g[14], u[13], DateTi
this._pagos.Add(new Recurrente("Transporte interno", 100, MetodoDePago.EFECTIVO, g[15], u[14], Date
this._pagos.Add(new Recurrente("Snacks oficina", 60, MetodoDePago.DEBITO, g[16], u[15], DateTime.Ne
this._pagos.Add(new Recurrente("Publicidad mensual", 300, MetodoDePago.CREDITO, g[17], u[16], DateT
this._pagos.Add(new Recurrente("Limpieza vidrios", 80, MetodoDePago.EFECTIVO, g[18], u[17], DateTi
this._pagos.Add(new Recurrente("Software contable", 250, MetodoDePago.CREDITO, g[19], u[18], DateTi
this._pagos.Add(new Recurrente("Asesoría legal", 500, MetodoDePago.DEBITO, g[20], u[19], DateTime.N
this._pagos.Add(new Recurrente("Mantenimiento ascensor", 120, MetodoDePago.CREDITO, g[21], u[20], D
this._pagos.Add(new Recurrente("Plan de salud", 350, MetodoDePago.DEBITO, g[22], u[21], DateTime.Ne
this._pagos.Add(new Recurrente("Correo corporativo", 30, MetodoDePago.CREDITO, g[23], u[20], DateTim
this._pagos.Add(new Recurrente("Publicidad local", 150, MetodoDePago.EFECTIVO, g[24], u[21], DateTim
this._pagos.Add(new Recurrente("Combustible mercantil", 200, MetodoDePago.DEBITO, g[25], u[24], DateTi

```

Aclaración:

Se modificó manualmente el atributo contraseña de cada Usuario para que tenga un mínimo de 8 caracteres.

Evidencias de Testing:

Funcionalidad	Origen	Intento	Resultado
N1	Login	Usuario intenta registrarse con datos inválidos/faltantes.	Se rechaza conexión. "Datos incorrectos".
N2	Login	Usuario intenta registrarse con datos correctos.	Se redirecciona a Perfil.
N3	Login	Usuario intenta redirigir mediante URL a cualquier vista del proyecto.	Se rechaza la conexión. "Permisos insuficientes".
N4	Dentro del sistema	Usuario Empleado intenta redirigir mediante URL a vistas no accesibles a su rol.	Se rechaza la petición. "Permisos insuficientes".
N5	Pago	Usuario rol Gerente intenta filtrar pagos sin haber introducido fecha.	Se pide que ingrese una fecha válida.
N6	Agregar Pago	Usuario rol Gerente/Empleado intenta añadir pagos con datos correctos.	El sistema crea el nuevo pago. "Pago cargado con éxito!"
N7	Agregar Pago	Usuario rol Gerente/Empleado intenta añadir pagos con datos faltantes.	Se pide completar los campos obligatorios.
N8	Agregar Pago	Usuario rol Gerente/Empleado intenta añadir un pago con monto negativo.	El sistema rechaza la creación del pago. "El valor debe ser mayor de o igual a 1".
N9	Agregar Pago	Usuario rol Gerente/Empleado intenta cargar un tipo de pago recurrente sin completar el campo Fecha Hasta.	El sistema permite la creación del pago.
N10	Agregar Pago	Usuario rol Gerente/Empleado intenta cargar un tipo de pago con los campos Fecha Hasta menor a Fecha Desde.	El sistema rechaza la creación del pago. "Fecha inválida".
N11	Agregar Gasto	Usuario rol Gerente intenta añadir un tipo de gasto con nombre igual a un gasto ya existente.	El sistema rechaza la creación del tipo de gasto. "Tipo de gasto ya existente".
N12	Eliminar Gasto	Usuario rol Gerente intenta eliminar un tipo de gasto ya existente en algún pago.	El sistema rechaza la eliminación. "No es posible eliminar este Tipo de Gasto".
N13	Eliminar Gasto	Usuario rol Gerente intenta eliminar un tipo de gasto que no ha sido usado en un pago.	El sistema elimina el tipo de gasto.
N14	Dentro del sistema, usuario ya logueado.	Usuario intenta desloguearse.	El sistema desloguea al usuario.
N15	Dentro del sistema, usuario ya logueado.	Usuario selecciona "Gestor Pagos V1".	El sistema redirecciona a Perfil.

