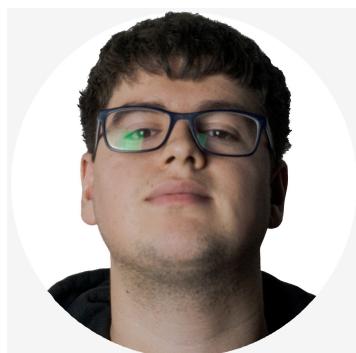


Universidad ORT Uruguay
Facultad de Ingeniería
Escuela de Tecnología

PRIMER OBLIGATORIO PROGRAMACIÓN 2 DOCUMENTO DE ANÁLISIS



Gonzalo Barreiro - 282899



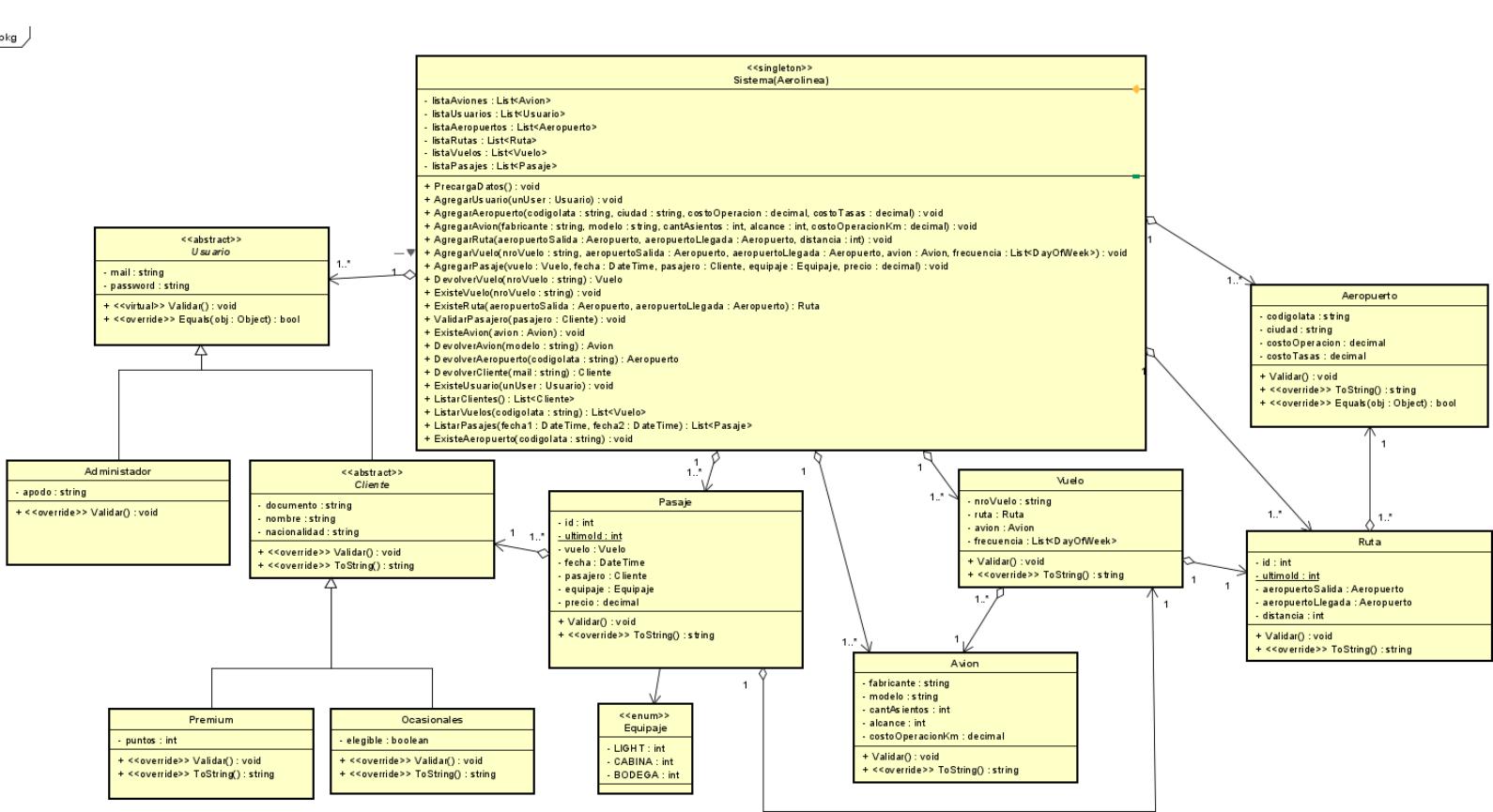
Lautaro Zapata - 294574

N2D
Docente: Alejandro Martinis
Analista en Tecnologías de la Información
Fecha de entrega del documento 12-05-2025

ÍNDICE

UML Completo del Dominio.....	3
TABLAS DE PRECARGAS.....	4
Precarga de usuarios:	
Usuario Administrador.....	4
Usuarios Premium.....	4
Usuarios Ocasionales.....	4
Precarga de Aeropuertos.....	5
Precarga de Aviones.....	6
Precarga de Rutas.....	6
Precarga de Vuelos.....	7
Precarga de Pasajes.....	9
Código Fuente del Dominio.....	10
Administrator.cs.....	10
Aeropuerto.cs.....	11
Avion.cs.....	12
Cliente.cs.....	14
Equipaje.cs.....	15
Ocasionales.cs.....	15
Pasaje.cs.....	16
Premium.cs.....	18
Ruta.cs.....	19
Sistema.cs.....	20
Usuario.cs.....	31
Vuelo.cs.....	32
Evidencia de las consultas de IA para la precarga.....	34
Para la precarga de Usuarios:.....	34
Para la precarga de Aeropuertos:.....	36
Para la precarga de Aviones:.....	38
Para la precarga de Rutas:.....	39
Para la precarga de Vuelos:.....	41
Para la precarga de Pasajes:.....	43

UML Completo del Dominio



TABLAS DE PRECARGAS

Precarga de usuarios:

Usuario Administrador

Apodo	Mail	Password
admin1	admin1@correo.com	Admin123
admin2	admin2@correo.com	Admin1234

Usuarios Premium

Puntos	Documento	Nombre	Mail	Password	Nacionalidad
1000	50506264	Lautaro	oratualzc@gmail.com	1234	Uruguayo
200	60458216	Valentina	valen.rios@example.com	pass123	Uruguaya
330	71234907	Diego	diegomez@example.com	abcd5678	Argentino
1200	83920148	Camila	camila.silva@example.com	qwerty789	Chilena
0	91563020	Felipe	felipe.travels@example.com	travel2025	Paraguayo

Usuarios Ocasionales

Documento	Nombre	Nacionalidad	Mail	Password
91563027	Felipe	Paraguayo	felipe.travelz@example.com	travel1025
82341074	Lucía	Uruguaya	lucia.mendez@example.com	lucia321
73429812	Marcos	Chileno	marcos.viaja@example.com	marcos123
61827341	Sofia	Argentina	sofia.ruta@example.com	sofiaviaje
70291561	Joaquín	Peruano	joaquin.aventura@example.com	joaquin456

Precarga de Aeropuertos

Código IATA	Ciudad	Costo Operación	Costo Tasas
MVD	Montevideo	1,500.00	500.00
EZE	Buenos Aires - Ezeiza	1,800.00	600.00
AEP	Buenos Aires - Aeroparque	1,700.00	550.00
PDP	Punta del Este	1,200.00	400.00
GRU	São Paulo	2,000.00	700.00
SCL	Santiago de Chile	1,900.00	650.00
LIM	Lima	2,100.00	750.00
ASU	Asunción	1,600.00	500.00
MIA	Miami	3,000.00	1,000.00
JFK	Nueva York	3,500.00	1,200.00
MAD	Madrid	3,200.00	1,100.00
BCN	Barcelona	3,100.00	1,050.00
LAX	Los Ángeles	3,600.00	1,250.00
DXB	Dubái	4,000.00	1,400.00
PEK	Pekín	4,200.00	1,500.00
COR	Córdoba	1,550.00	480.00
MDZ	Mendoza	1,580.00	500.00
CYD	Colonia del Sacramento	1,100.00	390.00
SJO	Salto	1,150.00	410.00
RIO	Río de Janeiro	1,950.00	670.00

Precarga de Aviones

Fabricante	Modelo	Cantidad Asientos	Alcance	Costo operación por km
Boeing	737 MAX	189	6570	5
Airbus	A320neo	180	6300	6
Embraer	E195-E2	132	4815	4
Bombardier	CS300	160	6112	7

Precarga de Rutas

Aeropuerto salida	Aeropuerto llegada	Distancia
MVD	EZE	200
MVD	GRU	1570
MVD	SCL	1370
EZE	LIM	3140
GRU	MIA	6540
SCL	MIA	6500
PDP	AEP	320
ASU	EZE	1080
MVD	LAX	6000
MVD	JFK	6000
JFK	LAX	3980
BCN	MAD	620
MIA	BCN	6000
DXB	PEK	5840

COR	MDZ	660
MDZ	SCL	480
CYD	MVD	170
SJO	PDP	550
PEK	LAX	6000
PEK	JFK	6000
DXB	MAD	5700
RIO	GRU	430
RIO	MVD	1820
COR	AEP	660
CYD	EZE	190
SJO	ASU	1080
PDP	RIO	1930
MVD	LIM	3120
MIA	JFK	1760
MAD	JFK	5760
LIM	SCL	2000

Precarga de Vuelos

Nro.Vuelo	Aeropuerto Salida	Aeropuerto Llegada	Avión	Frecuencia
AR101	MVD	EZE	E195-E2	Monday, Friday, Wednesday
AR102	MVD	GRU	E195-E2	Tuesday
AR103	MVD	SCL	E195-E2	Wednesday
AR104	EZE	LIM	A320neo	Thursday, Friday

AR105	GRU	MIA	737 MAX	Friday
AR106	SCL	MIA	737 MAX	Saturday
AR107	SCL	MIA	737 MAX	Saturday
AR108	SCL	MIA	737 MAX	Saturday
AR109	PDP	AEP	E195-E2	Sunday
AR110	ASU	EZE	E195-E2	Monday
AR111	JFK	LAX	737 MAX	Tuesday
AR190	BCN	MAD	E195-E2	Wednesday
AR140	DXB	PEK	737 MAX	Thursday
AR112	COR	MDZ	E195-E2	Friday
AR113	MDZ	SCL	E195-E2	Saturday
AR114	CYD	MVD	E195-E2	Sunday
AR115	SJO	PDP	E195-E2	Monday
AR116	DXB	MAD	737 MAX	Tuesday, Friday
AR117	RIO	GRU	E195-E2	Wednesday
AR118	RIO	MVD	E195-E2	Thursday
AR119	COR	AEP	E195-E2	Friday
AR120	CYD	EZE	E195-E2	Saturday
AR121	SJO	ASU	A320neo	Sunday
AR122	PDP	RIO	A320neo	Monday
AR123	MVD	LIM	A320neo	Tuesday, Friday
AR125	MAD	JFK	737 MAX	Thursday
AR124	MIA	JFK	E195-E2	Wednesday
AR126	LIM	SCL	A320neo	Friday
AR127	MIA	BCN	737 MAX	Saturday
AR199	MIA	BCN	CS300	Saturday

AR128	MVD	JFK	737 MAX	Sunday, Friday
AR129	PEK	LAX	737 MAX	Monday, Friday
AR130	PEK	JFK	737 MAX	Tuesday

Precarga de Pasajes

Nro. Vuelo	Fecha	Pasajero	Equipaje	Precio
AR101	2025, 5, 9	oratualzc@gmail.com	CABINA	250
AR102	2025, 5, 13	valen.rios@example.com	BODEGA	300
AR103	2025, 5, 14	diegomez@example.com	CABINA	270
AR104	2025, 5, 15	camila.silva@example.com	BODEGA	320
AR105	2025, 5, 16	felipe.travels@example.com	CABINA	310
AR106	2025, 5, 17	lucia.mendez@example.com	BODEGA	290
AR107	2025, 5, 17	marcos.viaja@example.com	CABINA	280
AR108	2025, 5, 17	sofia.ruta@example.com	CABINA	260
AR109	2025, 5, 11	joaquin.aventura@example.com	BODEGA	270
AR110	2025, 5, 12	oratualzc@gmail.com	CABINA	245
AR111	2025, 5, 13	valen.rios@example.com	BODEGA	330
AR112	2025, 5, 16	diegomez@example.com	CABINA	220
AR113	2025, 5, 17	camila.silva@example.com	BODEGA	210
AR114	2025, 5, 18	felipe.travels@example.com	CABINA	280
AR115	2025, 5, 19	lucia.mendez@example.com	BODEGA	270
AR116	2025, 5, 13	marcos.viaja@example.com	CABINA	350
AR117	2025, 5, 14	sofia.ruta@example.com	BODEGA	290
AR118	2025, 5, 15	joaquin.aventura@example.com	CABINA	260

AR119	2025, 5, 16	oratualzc@gmail.com	BODEGA	250
AR120	2025, 5, 17	valen.rios@example.com	CABINA	270
AR121	2025, 5, 18	diegomez@example.com	BODEGA	310
AR122	2025, 5, 19	camila.silva@example.com	CABINA	300
AR123	2025, 5, 13	felipe.travels@example.com	BODEGA	290
AR124	2025, 5, 14	lucia.mendez@example.com	CABINA	280
AR125	2025, 5, 15	joaquin.aventura@example.com	BODEGA	320

Código Fuente del Dominio

Administrador.cs

```

namespace Dominio
{
    public class Administrador : Usuario
    {
        string apodo;
        string mail;
        string password;

        public string Apodo { get => apodo; set => apodo = value; }
        public string Mail { get => mail; set => mail = value; }
        public string Password { get => password; set => password = value; }

        public Administrador()
        {
        }

        public Administrador(string apodo, string mail, string password) : base ( mail,
password)
        {
            Apodo = apodo;
        }
    }
}

```

```

        Mail = mail;
        Password = password;
    }

    public override void Validar()
    {
        base.Validar();
        if (string.IsNullOrEmpty(Apodo)) throw new Exception("El apodo no puede ser
vacio");
    }
}

```

[Aeropuerto.cs](#)

```

namespace Dominio
{
    public class Aeropuerto
    {
        string codigolata;
        string ciudad;
        decimal costoOperacion;
        decimal costoTasas;

        public string Codigolata { get => codigolata; set => codigolata = value; }
        public string Ciudad { get => ciudad; set => ciudad = value; }
        public decimal CostoOperacion { get => costoOperacion; set => costoOperacion =
value; }
        public decimal CostoTasas { get => costoTasas; set => costoTasas = value; }

        public Aeropuerto(string codigolata, string ciudad, decimal costoOperacion, decimal
costoTasas)
        {
            Codigolata = codigolata;
            Ciudad = ciudad;
            CostoOperacion = costoOperacion;
            CostoTasas = costoTasas;
        }

        public Aeropuerto()
        {
        }
    }
}

```

```

public void Validar()
{
    ValidarCodigolata(Codigolata);
    ValidarCiudad();
}

private void ValidarCodigolata(string codigolata)
{
    if (string.IsNullOrEmpty(codigolata)) throw new Exception("El código IATA no puede ser vacío");
    if (codigolata.Length != 3) throw new Exception("El código IATA debe ser de 3 letras");
    foreach (char letra in codigolata)
    {
        if (!char.IsLetter(letra))
            throw new Exception("El código IATA solo puede contener letras");
    }
}
private void ValidarCiudad()
{
    if (string.IsNullOrEmpty(Ciudad)) throw new Exception("La ciudad no puede ser vacía");
}
public override bool Equals(object? obj)
{
    return obj is Aeropuerto aeropuerto && this.ciudad == aeropuerto.Ciudad;
}

public override string ToString()
{
    return $"{Codigolata}";
}
}
}
}

```

[Avion.cs](#)

```

namespace Dominio
{
    public class Avion
    {
        string fabricante;
        string modelo;
        int cantAsientos;
    }
}

```

```
int alcance;
decimal costoOperacionKm;

public string Fabricante { get => fabricante; set => fabricante = value; }
public string Modelo { get => modelo; set => modelo = value; }
public int CantAsientos { get => cantAsientos; set => cantAsientos = value; }
public int Alcance { get => alcance; set => alcance = value; }
public decimal CostoOperacionKm { get => costoOperacionKm; set =>
costoOperacionKm = value; }

public Avion()
{
}

public Avion(string fabricante, string modelo, int cantAsientos, int alcance, decimal
costoOperacionKm)
{
    Fabricante = fabricante;
    Modelo = modelo;
    CantAsientos = cantAsientos;
    Alcance = alcance;
    CostoOperacionKm = costoOperacionKm;
}

public void Validar()
{
    ValidarFabricante();
    ValidarModelo();
    ValidarAsientos();
    ValidarAlcance();
    //CostoOperacionKm; Se realiza en la segunda entrega.
}

private void ValidarFabricante()
{
    if (string.IsNullOrEmpty(Fabricante)) throw new Exception("El fabricante no puede
ser vacio");
}

private void ValidarModelo()
{
    if (string.IsNullOrEmpty(Modelo)) throw new Exception("El modelo no puede ser
vacio");
}

private void ValidarAsientos()
{
```

```

        if (CantAsientos < 1 ) throw new Exception("La cantidad de asientos no puede ser
menor a 1");
    }
    private void ValidarAlcance()
    {
        if (Alcance < 1) throw new Exception("El alcance del avion no puede ser menor a
1KM");
    }

    public override string ToString()
    {
        return $"{Modelo}";
    }
}
}

```

[Cliente.cs](#)

```

namespace Dominio
{
    public abstract class Cliente : Usuario
    {
        string documento;
        string nombre;
        string nacionalidad;

        public string Documento { get => documento; set => documento = value; }
        public string Nombre { get => nombre; set => nombre = value; }
        public string Nacionalidad { get => nacionalidad; set => nacionalidad = value; }
        public Cliente()
        {
        }

        public Cliente(string documento, string nombre, string nacionalidad, string mail, string
password) : base(mail,password)
        {
            Documento = documento;
            Nombre = nombre;
            Nacionalidad = nacionalidad;
        }

        public override void Validar()
        {
            base.Validar();
            if (string.IsNullOrEmpty(Documento)) throw new Exception("El documento no puede
ser vacio");
        }
    }
}

```

```

        if(string.IsNullOrEmpty(Nombre)) throw new Exception("El nombre no puede ser
vacio");
        if (string.IsNullOrEmpty(Nacionalidad)) throw new Exception("La nacionalidad no
puede ser vacia");
        if (Documento.Length != 8) throw new Exception("El documento debe tener 8
numeros");

    }

    public override string ToString()
    {
        return $"{Nombre} , {Mail} , {Password} , {Documento} , {Nacionalidad}";
    }

}

```

[Equipaje.cs](#)

```

namespace Dominio
{
    public enum Equipaje
    {
        LIGHT =1,
        CABINA = 2,
        BODEGA =3
    }
}

```

[Opcionales.cs](#)

```

namespace Dominio
{
    public class Opcionales : Cliente
    {
        static Random random = new Random();
        bool elegible;

```

```

public bool Elegible { get => elegible; set => elegible = value; }

public Ocasionales()
{
}

public Ocasionales(string documento, string nombre, string nacionalidad, string mail,
string password) : base(documento, nombre, nacionalidad, mail, password)
{
    Elegible = random.Next(0, 2) == 1; // Agarra un numero entre 0 y 2 [0,1], si da 1 es
true si da 0 es false.
}

public override void Validar()
{
    base.Validar();
}

private string EsElegible()
{
    if (Elegible) return "Si";
    return "No";
}

public override string ToString()
{
    return $"{base.ToString()}, Es Elegible: {EsElegible()}, Tipo de cliente: Ocasional \n";
}

}

```

Pasaje.cs

```

namespace Dominio
{
    public class Pasaje
    {
        int id;
        static int ultimold;
        Vuelo vuelo;
        DateTime fecha;
        Cliente pasajero;
    }
}
```

```

Equipaje equipaje;
decimal precio;

public int Id { get => id; set => id = value; }
public static int Ultimold { get => ultimold; set => ultimold = value; }
public Vuelo Vuelo { get => vuelo; set => vuelo = value; }
public DateTime Fecha { get => fecha; set => fecha = value; }
public Cliente Pasajero { get => pasajero; set => pasajero = value; }
public Equipaje Equipaje { get => equipaje; set => equipaje = value; }
public decimal Precio { get => precio; set => precio = value; }

public Pasaje(Vuelo vuelo, DateTime fecha, Cliente pasajero, Equipaje equipaje,
decimal precio)
{
    Id = ++ultimold;
    Vuelo = vuelo;
    Fecha = fecha;
    Pasajero = pasajero;
    Equipaje = equipaje;
    Precio = precio;
}

public Pasaje()
{
    Id = ++ultimold;
}

public void Validar()
{
    ValidarFechaConFrecuencia();
    // Son metodos para la segunda entrega.
    //ValidarEquipaje();
    //ValidarPrecio();
}

private void ValidarFechaConFrecuencia()
{
    if (!Vuelo.Frecuencia.Contains(Fecha.DayOfWeek)) throw new Exception("La fecha
no coincide con el vuelo.");
}

public override string ToString()
{
    return $"Id: {Id} , Nombre: {Pasajero.Nombre} , Precio: {Precio} , Fecha: {Fecha} ,
NroVuelo {Vuelo.NroVuelo} \n ";
}

```

```
    }

}

}
```

[Premium.cs](#)

```
namespace Dominio
{
    public class Premium : Cliente
    {
        int puntos;

        public int Puntos { get => puntos; set => puntos = value; }

        public Premium()
        {
        }

        public Premium(int puntos, string documento, string nombre, string mail, string password, string nacionalidad ): base(documento, nombre, nacionalidad, mail, password )
        {
            Puntos = puntos;
        }

        public override void Validar()
        {
            base.Validar();
            if (Puntos < 0) throw new Exception("Los puntos no pueden ser menor a 0");
        }

        public override string ToString()
        {
            return $"{base.ToString()}, {Puntos} Tipo de cliente: Premium \n";
        }
    }
}
```

Ruta.cs

```
namespace Dominio
{
    public class Ruta
    {
        int id;
        static int ultimold;
        Aeropuerto aeropuertoSalida;
        Aeropuerto aeropuertoLlegada;
        int distancia;

        public int Id { get => id; set => id = value; }
        public static int Ultimold { get => ultimold; set => ultimold = value; }
        public Aeropuerto AeropuertoSalida { get => aeropuertoSalida; set => aeropuertoSalida
= value; }
        public Aeropuerto AeropuertoLlegada { get => aeropuertoLlegada; set =>
aeropuertoLlegada = value; }
        public int Distancia { get => distancia; set => distancia = value; }

        public Ruta(Aeropuerto aeropuertoSalida, Aeropuerto aeropuertoLlegada, int distancia)
        {
            Id = ++ultimold;
            AeropuertoSalida = aeropuertoSalida;
            AeropuertoLlegada = aeropuertoLlegada;
            Distancia = distancia;
        }

        public Ruta()
        {
            Id = ++ultimold;
        }

        public void Validar()
        {
            ValidarDistancia();
            ValidarAeropuertos();
        }

        private void ValidarDistancia()
        {
            if (Distancia < 1) throw new Exception("La distancia de la ruta debe ser mayor a
1KM");
        }
    }
}
```

```

private void ValidarAeropuertos()
{
    if (AeropuertoSalida.Equals(AeropuertoLlegada)) throw new Exception("El
aeropuerto de salida no puede ser igual al de llegada");
}

public override string ToString()
{
    return $"{AeropuertoSalida} - {AeropuertoLlegada}";
}

}

```

Sistema.cs

```

namespace Dominio
{
    public class Sistema
    {
        private static Sistema instancia;

        List<Avion> listaAviones = new List<Avion>();
        List<Usuario> listaUsuarios = new List<Usuario>();
        List<Aeropuerto> listaAeropuertos = new List<Aeropuerto>();
        List<Ruta> listaRutas = new List<Ruta>();
        List<Vuelo> listaVuelos = new List<Vuelo>();
        List<Pasaje> listaPasajes = new List<Pasaje>();

        public static Sistema Instancia
        {
            get
            {
                if (instancia == null) instancia = new Sistema();
                return instancia;
            }
        }
        private Sistema()
        {
            PrecargaDatos();
        }

        public void PrecargaDatos()
        {

```

// Puede que existan diferencias a los prompts de la documentación ya que la IA cometía errores en la generación de los datos. Por lo que corregimos manualmente algunas precargas.

```
#region PrecargaUsuarios
AregarUsuario(new Administrador("admin1", "admin1@correo.com", "Admin123"));
AregarUsuario(new Administrador("admin2", "admin2@correo.com", "Admin1234"));
AregarUsuario(new Premium(1000, "50506264", "Lautaro", "oratualzc@gmail.com",
"1234", "Uruguayo"));
AregarUsuario(new Premium(200, "60458216", "Valentina",
"valen.rios@example.com", "pass123", "Uruguaya"));
AregarUsuario(new Premium(330, "71234907", "Diego",
"diegomez@example.com", "abcd5678", "Argentino"));
AregarUsuario(new Premium(1200, "83920148", "Camila",
"camila.silva@example.com", "qwerty789", "Chilena"));
AregarUsuario(new Premium(0, "91563020", "Felipe",
"felipe.travels@example.com", "travel2025", "Paraguayo"));

AregarUsuario(new Ocasionales("91563027", "Felipe", "Paraguayo",
"felipe.travels@example.com", "travel2025"));
AregarUsuario(new Ocasionales("82341074", "Lucía", "Uruguaya",
"lucia.mendez@example.com", "lucia321"));
AregarUsuario(new Ocasionales("73429812", "Marcos", "Chileno",
"marcos.viaja@example.com", "marcos123"));
AregarUsuario(new Ocasionales("61827341", "Sofía", "Argentina",
"sofia.ruta@example.com", "sofiaviaje"));
AregarUsuario(new Ocasionales("70291561", "Joaquín", "Peruano",
"joaquin.aventura@example.com", "joaquin456"));
```

```
#endregion
```

```
#region PrecargaAeropuertos
AregarAeropuerto("MVD", "Montevideo", 1500.00m, 500.00m);
AregarAeropuerto("EZE", "Buenos Aires - Ezeiza", 1800.00m, 600.00m);
AregarAeropuerto("AEP", "Buenos Aires - Aeroparque", 1700.00m, 550.00m);
AregarAeropuerto("PDP", "Punta del Este", 1200.00m, 400.00m);
AregarAeropuerto("GRU", "São Paulo", 2000.00m, 700.00m);
AregarAeropuerto("SCL", "Santiago de Chile", 1900.00m, 650.00m);
AregarAeropuerto("LIM", "Lima", 2100.00m, 750.00m);
AregarAeropuerto("ASU", "Asunción", 1600.00m, 500.00m);
AregarAeropuerto("MIA", "Miami", 3000.00m, 1000.00m);
AregarAeropuerto("JFK", "Nueva York", 3500.00m, 1200.00m);
AregarAeropuerto("MAD", "Madrid", 3200.00m, 1100.00m);
AregarAeropuerto("BCN", "Barcelona", 3100.00m, 1050.00m);
AregarAeropuerto("LAX", "Los Ángeles", 3600.00m, 1250.00m);
AregarAeropuerto("DXB", "Dubái", 4000.00m, 1400.00m);
```

```
AgregarAeropuerto("PEK", "Pekín", 4200.00m, 1500.00m);
AgregarAeropuerto("COR", "Córdoba", 1550.00m, 480.00m);
AgregarAeropuerto("MDZ", "Mendoza", 1580.00m, 500.00m);
AgregarAeropuerto("CYD", "Colonia del Sacramento", 1100.00m, 390.00m);
AgregarAeropuerto("SJO", "Salto", 1150.00m, 410.00m);
AgregarAeropuerto("RIO", "Río de Janeiro", 1950.00m, 670.00m);
#endifregion

#region PrecargaAviones
AgregarAvion("Boeing", "737 MAX", 189, 6570, 5);
AgregarAvion("Airbus", "A320neo", 180, 6300, 6);
AgregarAvion("Embraer", "E195-E2", 132, 4815, 4);
AgregarAvion("Bombardier", "CS300", 160, 6112, 7);
#endifregion

#region PrecargaRutas
AgregarRuta(DevolverAeropuerto("MVD"), DevolverAeropuerto("EZE"), 200);
AgregarRuta(DevolverAeropuerto("MVD"), DevolverAeropuerto("GRU"), 1570);
AgregarRuta(DevolverAeropuerto("MVD"), DevolverAeropuerto("SCL"), 1370);
AgregarRuta(DevolverAeropuerto("EZE"), DevolverAeropuerto("LIM"), 3140);
AgregarRuta(DevolverAeropuerto("GRU"), DevolverAeropuerto("MIA"), 6540);
AgregarRuta(DevolverAeropuerto("SCL"), DevolverAeropuerto("MIA"), 6500);
AgregarRuta(DevolverAeropuerto("PDP"), DevolverAeropuerto("AEP"), 320);
AgregarRuta(DevolverAeropuerto("ASU"), DevolverAeropuerto("EZE"), 1080);
AgregarRuta(DevolverAeropuerto("MVD"), DevolverAeropuerto("LAX"), 6000);
AgregarRuta(DevolverAeropuerto("MVD"), DevolverAeropuerto("JFK"), 6000);
AgregarRuta(DevolverAeropuerto("JFK"), DevolverAeropuerto("LAX"), 3980);
AgregarRuta(DevolverAeropuerto("BCN"), DevolverAeropuerto("MAD"), 620);
AgregarRuta(DevolverAeropuerto("MIA"), DevolverAeropuerto("BCN"), 6000);
AgregarRuta(DevolverAeropuerto("DXB"), DevolverAeropuerto("PEK"), 5840);
AgregarRuta(DevolverAeropuerto("COR"), DevolverAeropuerto("MDZ"), 660);
AgregarRuta(DevolverAeropuerto("MDZ"), DevolverAeropuerto("SCL"), 480);
AgregarRuta(DevolverAeropuerto("CYD"), DevolverAeropuerto("MVD"), 170);
AgregarRuta(DevolverAeropuerto("SJO"), DevolverAeropuerto("PDP"), 550);
AgregarRuta(DevolverAeropuerto("PEK"), DevolverAeropuerto("LAX"), 6000);
AgregarRuta(DevolverAeropuerto("PEK"), DevolverAeropuerto("JFK"), 6000);
AgregarRuta(DevolverAeropuerto("DXB"), DevolverAeropuerto("MAD"), 5700);
AgregarRuta(DevolverAeropuerto("RIO"), DevolverAeropuerto("GRU"), 430);
AgregarRuta(DevolverAeropuerto("RIO"), DevolverAeropuerto("MVD"), 1820);
AgregarRuta(DevolverAeropuerto("COR"), DevolverAeropuerto("AEP"), 660);
AgregarRuta(DevolverAeropuerto("CYD"), DevolverAeropuerto("EZE"), 190);
AgregarRuta(DevolverAeropuerto("SJO"), DevolverAeropuerto("ASU"), 1080);
AgregarRuta(DevolverAeropuerto("PDP"), DevolverAeropuerto("RIO"), 1930);
AgregarRuta(DevolverAeropuerto("MVD"), DevolverAeropuerto("LIM"), 3120);
AgregarRuta(DevolverAeropuerto("MIA"), DevolverAeropuerto("JFK"), 1760);
AgregarRuta(DevolverAeropuerto("MAD"), DevolverAeropuerto("JFK"), 5760);
AgregarRuta(DevolverAeropuerto("LIM"), DevolverAeropuerto("SCL"), 2000);

#endifregion
```

```
#region PrecargaVuelos
    AgregarVuelo("AR101", DevolverAeropuerto("MVD"), DevolverAeropuerto("EZE"),
DevolverAvion("E195-E2"), new List<DayOfWeek> { DayOfWeek.Monday,
DayOfWeek.Friday, DayOfWeek.Wednesday });
    AgregarVuelo("AR102", DevolverAeropuerto("MVD"), DevolverAeropuerto("GRU"),
DevolverAvion("E195-E2"), new List<DayOfWeek> { DayOfWeek.Tuesday });
    AgregarVuelo("AR103", DevolverAeropuerto("MVD"), DevolverAeropuerto("SCL"),
DevolverAvion("E195-E2"), new List<DayOfWeek> { DayOfWeek.Wednesday });
    AgregarVuelo("AR104", DevolverAeropuerto("EZE"), DevolverAeropuerto("LIM"),
DevolverAvion("A320neo"), new List<DayOfWeek> { DayOfWeek.Thursday,
DayOfWeek.Friday });
    AgregarVuelo("AR105", DevolverAeropuerto("GRU"), DevolverAeropuerto("MIA"),
DevolverAvion("737 MAX"), new List<DayOfWeek> { DayOfWeek.Friday });
    AgregarVuelo("AR106", DevolverAeropuerto("SCL"), DevolverAeropuerto("MIA"),
DevolverAvion("737 MAX"), new List<DayOfWeek> { DayOfWeek.Saturday });
    AgregarVuelo("AR107", DevolverAeropuerto("SCL"), DevolverAeropuerto("MIA"),
DevolverAvion("737 MAX"), new List<DayOfWeek> { DayOfWeek.Saturday });
    AgregarVuelo("AR108", DevolverAeropuerto("SCL"), DevolverAeropuerto("MIA"),
DevolverAvion("737 MAX"), new List<DayOfWeek> { DayOfWeek.Saturday });
    AgregarVuelo("AR109", DevolverAeropuerto("PDP"), DevolverAeropuerto("AEP"),
DevolverAvion("E195-E2"), new List<DayOfWeek> { DayOfWeek.Sunday });
    AgregarVuelo("AR110", DevolverAeropuerto("ASU"), DevolverAeropuerto("EZE"),
DevolverAvion("E195-E2"), new List<DayOfWeek> { DayOfWeek.Monday });
    AgregarVuelo("AR111", DevolverAeropuerto("JFK"), DevolverAeropuerto("LAX"),
DevolverAvion("737 MAX"), new List<DayOfWeek> { DayOfWeek.Tuesday });
    AgregarVuelo("AR190", DevolverAeropuerto("BCN"), DevolverAeropuerto("MAD"),
DevolverAvion("E195-E2"), new List<DayOfWeek> { DayOfWeek.Wednesday });
    AgregarVuelo("AR140", DevolverAeropuerto("DXB"), DevolverAeropuerto("PEK"),
DevolverAvion("737 MAX"), new List<DayOfWeek> { DayOfWeek.Thursday });
    AgregarVuelo("AR112", DevolverAeropuerto("COR"), DevolverAeropuerto("MDZ"),
DevolverAvion("E195-E2"), new List<DayOfWeek> { DayOfWeek.Friday });
    AgregarVuelo("AR113", DevolverAeropuerto("MDZ"), DevolverAeropuerto("SCL"),
DevolverAvion("E195-E2"), new List<DayOfWeek> { DayOfWeek.Saturday });
    AgregarVuelo("AR114", DevolverAeropuerto("CYD"), DevolverAeropuerto("MVD"),
DevolverAvion("E195-E2"), new List<DayOfWeek> { DayOfWeek.Sunday });
    AgregarVuelo("AR115", DevolverAeropuerto("SJO"), DevolverAeropuerto("PDP"),
DevolverAvion("E195-E2"), new List<DayOfWeek> { DayOfWeek.Monday });
    AgregarVuelo("AR116", DevolverAeropuerto("DXB"), DevolverAeropuerto("MAD"),
DevolverAvion("737 MAX"), new List<DayOfWeek> { DayOfWeek.Tuesday,
DayOfWeek.Friday });
    AgregarVuelo("AR117", DevolverAeropuerto("RIO"), DevolverAeropuerto("GRU"),
DevolverAvion("E195-E2"), new List<DayOfWeek> { DayOfWeek.Wednesday });
    AgregarVuelo("AR118", DevolverAeropuerto("RIO"), DevolverAeropuerto("MVD"),
DevolverAvion("E195-E2"), new List<DayOfWeek> { DayOfWeek.Thursday });
    AgregarVuelo("AR119", DevolverAeropuerto("COR"), DevolverAeropuerto("AEP"),
DevolverAvion("E195-E2"), new List<DayOfWeek> { DayOfWeek.Friday });
    AgregarVuelo("AR120", DevolverAeropuerto("CYD"), DevolverAeropuerto("EZE"),
DevolverAvion("E195-E2"), new List<DayOfWeek> { DayOfWeek.Saturday });
```

```
        AgregarVuelo("AR121", DevolverAeropuerto("SJO"), DevolverAeropuerto("ASU"),
DevolverAvion("A320neo"), new List<DayOfWeek> { DayOfWeek.Sunday });
        AgregarVuelo("AR122", DevolverAeropuerto("PDP"), DevolverAeropuerto("RIO"),
DevolverAvion("A320neo"), new List<DayOfWeek> { DayOfWeek.Monday });
        AgregarVuelo("AR123", DevolverAeropuerto("MVD"), DevolverAeropuerto("LIM"),
DevolverAvion("A320neo"), new List<DayOfWeek> { DayOfWeek.Tuesday,
DayOfWeek.Friday });
        AgregarVuelo("AR125", DevolverAeropuerto("MAD"), DevolverAeropuerto("JFK"),
DevolverAvion("737 MAX"), new List<DayOfWeek> { DayOfWeek.Thursday });
        AgregarVuelo("AR124", DevolverAeropuerto("MIA"), DevolverAeropuerto("JFK"),
DevolverAvion("E195-E2"), new List<DayOfWeek> { DayOfWeek.Wednesday });
        AgregarVuelo("AR126", DevolverAeropuerto("LIM"), DevolverAeropuerto("SCL"),
DevolverAvion("A320neo"), new List<DayOfWeek> { DayOfWeek.Friday });
        AgregarVuelo("AR127", DevolverAeropuerto("MIA"), DevolverAeropuerto("BCN"),
DevolverAvion("737 MAX"), new List<DayOfWeek> { DayOfWeek.Saturday });
        AgregarVuelo("AR199", DevolverAeropuerto("MIA"), DevolverAeropuerto("BCN"),
DevolverAvion("CS300"), new List<DayOfWeek> { DayOfWeek.Saturday });
        AgregarVuelo("AR128", DevolverAeropuerto("MVD"), DevolverAeropuerto("JFK"),
DevolverAvion("737 MAX"), new List<DayOfWeek> { DayOfWeek.Sunday,
DayOfWeek.Friday });
        AgregarVuelo("AR129", DevolverAeropuerto("PEK"), DevolverAeropuerto("LAX"),
DevolverAvion("737 MAX"), new List<DayOfWeek> { DayOfWeek.Monday,
DayOfWeek.Friday });
        AgregarVuelo("AR130", DevolverAeropuerto("PEK"), DevolverAeropuerto("JFK"),
DevolverAvion("737 MAX"), new List<DayOfWeek> { DayOfWeek.Tuesday });

#endregion

#region PrecargaPasajes
        AgregarPasaje(DevolverVuelo("AR101"), new DateTime(2025, 5, 9),
DevolverCliente("oratualzc@gmail.com"), Equipaje.CABINA, 250);
        AgregarPasaje(DevolverVuelo("AR102"), new DateTime(2025, 5, 13),
DevolverCliente("valen.rios@example.com"), Equipaje.BODEGA, 300);
        AgregarPasaje(DevolverVuelo("AR103"), new DateTime(2025, 5, 14),
DevolverCliente("diegomez@example.com"), Equipaje.CABINA, 270);
        AgregarPasaje(DevolverVuelo("AR104"), new DateTime(2025, 5, 15),
DevolverCliente("camila.silva@example.com"), Equipaje.BODEGA, 320);
        AgregarPasaje(DevolverVuelo("AR105"), new DateTime(2025, 5, 16),
DevolverCliente("felipe.travels@example.com"), Equipaje.CABINA, 310);
        AgregarPasaje(DevolverVuelo("AR106"), new DateTime(2025, 5, 17),
DevolverCliente("lucia.mendez@example.com"), Equipaje.BODEGA, 290);
        AgregarPasaje(DevolverVuelo("AR107"), new DateTime(2025, 5, 17),
DevolverCliente("marcos.viaja@example.com"), Equipaje.CABINA, 280);
        AgregarPasaje(DevolverVuelo("AR108"), new DateTime(2025, 5, 17),
DevolverCliente("sofia.ruta@example.com"), Equipaje.CABINA, 260);
        AgregarPasaje(DevolverVuelo("AR109"), new DateTime(2025, 5, 11),
DevolverCliente("joaquin.aventura@example.com"), Equipaje.BODEGA, 270);
        AgregarPasaje(DevolverVuelo("AR110"), new DateTime(2025, 5, 12),
DevolverCliente("oratualzc@gmail.com"), Equipaje.CABINA, 245);
```

```

        AgregarPasaje(DevolverVuelo("AR111"), new DateTime(2025, 5, 13),
DevolverCliente("valen.rios@example.com"), Equipaje.BODEGA, 330);
        AgregarPasaje(DevolverVuelo("AR112"), new DateTime(2025, 5, 16),
DevolverCliente("diegomez@example.com"), Equipaje.CABINA, 220);
        AgregarPasaje(DevolverVuelo("AR113"), new DateTime(2025, 5, 17),
DevolverCliente("camila.silva@example.com"), Equipaje.BODEGA, 210);
        AgregarPasaje(DevolverVuelo("AR114"), new DateTime(2025, 5, 18),
DevolverCliente("felipe.travels@example.com"), Equipaje.CABINA, 280);
        AgregarPasaje(DevolverVuelo("AR115"), new DateTime(2025, 5, 19),
DevolverCliente("lucia.mendez@example.com"), Equipaje.BODEGA, 270);
        AgregarPasaje(DevolverVuelo("AR116"), new DateTime(2025, 5, 13),
DevolverCliente("marcos.viaja@example.com"), Equipaje.CABINA, 350);
        AgregarPasaje(DevolverVuelo("AR117"), new DateTime(2025, 5, 14),
DevolverCliente("sofia.ruta@example.com"), Equipaje.BODEGA, 290);
        AgregarPasaje(DevolverVuelo("AR118"), new DateTime(2025, 5, 15),
DevolverCliente("joaquin.aventura@example.com"), Equipaje.CABINA, 260);
        AgregarPasaje(DevolverVuelo("AR119"), new DateTime(2025, 5, 16),
DevolverCliente("oratualzc@gmail.com"), Equipaje.BODEGA, 250);
        AgregarPasaje(DevolverVuelo("AR120"), new DateTime(2025, 5, 17),
DevolverCliente("valen.rios@example.com"), Equipaje.CABINA, 270);
        AgregarPasaje(DevolverVuelo("AR121"), new DateTime(2025, 5, 18),
DevolverCliente("diegomez@example.com"), Equipaje.BODEGA, 310);
        AgregarPasaje(DevolverVuelo("AR122"), new DateTime(2025, 5, 19),
DevolverCliente("camila.silva@example.com"), Equipaje.CABINA, 300);
        AgregarPasaje(DevolverVuelo("AR123"), new DateTime(2025, 5, 13),
DevolverCliente("felipe.travels@example.com"), Equipaje.BODEGA, 290);
        AgregarPasaje(DevolverVuelo("AR124"), new DateTime(2025, 5, 14),
DevolverCliente("lucia.mendez@example.com"), Equipaje.CABINA, 280);
        AgregarPasaje(DevolverVuelo("AR125"), new DateTime(2025, 5, 15),
DevolverCliente("joaquin.aventura@example.com"), Equipaje.BODEGA, 320);

```

```

#endregion
}
// METODOS

#region Vuelo
public void AgregarVuelo(string nroVuelo, Aeropuerto aeropuertoSalida, Aeropuerto
aeropuertoLlegada, Avion avion, List<DayOfWeek> frecuencia)
{
    try
    {
        ExisteVuelo(nroVuelo);
        Vuelo vuelo = new Vuelo(nroVuelo, ExisteRuta(aeropuertoSalida,
aeropuertoLlegada), avion, frecuencia);
        vuelo.Validar();
        listaVuelos.Add(vuelo);
    }
}

```

```

        catch (Exception ex)
        {
            throw ex;
        }
    }

public Vuelo DevolverVuelo(string nroVuelo)
{
    foreach(Vuelo unVuelo in listaVuelos)
    {
        if(unVuelo.NroVuelo == nroVuelo) return unVuelo;
    }
    throw new Exception("El vuelo no existe");
}

public void ExisteVuelo(string nroVuelo)
{
    foreach (Vuelo unVuelo in listaVuelos)
    {
        if (unVuelo.NroVuelo.ToUpper() == nroVuelo.ToUpper()) throw new Exception("El
vuelo ya existe");
    }
}

public Ruta ExisteRuta(Aeropuerto aeropuertoSalida, Aeropuerto aeropuertoLlegada)
{
    // recorremos lista de rutas y si existe la ruta la devuelve.
    foreach(Ruta unaRuta in listaRutas)
    {
        if (unaRuta.AeropuertoLlegada == aeropuertoLlegada &&
unaRuta.AeropuertoSalida == aeropuertoSalida) return unaRuta;
    }
    throw new Exception("No existe una ruta con los aeropuertos especificados.");
}
#endifregion

#region Pasaje
public void AgregarPasaje(Vuelo vuelo, DateTime fecha, Cliente pasajero, Equipaje
equipaje, decimal precio)
{
    try
    {
        Pasaje pasaje = new Pasaje(vuelo, fecha, pasajero, equipaje, precio);
        pasaje.Validar();
        ValidarPasajero(pasajero);
        // CalcularPrecioPasaje() iría en este momento antes de la emision del pasaje. El
precio lo establecimos en la precarga para omitir el calculo en esta primera entrega.
    }
}

```

```

        listaPasajes.Add(pasaje);

    }

    catch (Exception ex)
    {
        throw ex;
    }

}

public void ValidarPasajero(Cliente pasajero)
{
    if (!listaUsuarios.Contains(pasajero)) throw new Exception("No existe el usuario");
}

#endregion

#region Avion
public void AgregarAvion(string fabricante, string modelo, int cantAsientos, int alcance,
decimal costoOperacionKm)
{
    try
    {
        Avion avion = new Avion(fabricante, modelo, cantAsientos, alcance,
costoOperacionKm);
        ExisteAvion(avion);
        avion.Validar();
        listaAviones.Add(avion);

    }
    catch (Exception ex)
    {
        throw ex;
    }
}

public void ExisteAvion(Avion avion)
{
    if (listaAviones.Contains(avion)) throw new Exception("El avion ya existe.");
}

public Avion DevolverAvion(string modelo)
{
    foreach(Avion unAvion in listaAviones)
    {
        if (unAvion.Modelo == modelo) return unAvion;
    }
    throw new Exception("No se encontro el avion");
}

```

```

#endregion

#region Ruta
public void AgregarRuta(Aeropuerto aeropuertoSalida, Aeropuerto aeropuertoLlegada,
int distancia)
{
    try
    {
        Ruta ruta = new Ruta(aeropuertoSalida,aeropuertoLlegada,distancia);
        ruta.Validar();
        listaRutas.Add(ruta);

    }
    catch (Exception ex)
    {
        throw ex;
    }
}

#endregion

#region Aeropuerto
public void AgregarAeropuerto(string codigolata, string ciudad, decimal
costoOperacion, decimal costoTasas)
{
    try
    {
        Aeropuerto aeropuerto = new Aeropuerto(codigolata, ciudad, costoOperacion,
costoTasas);
        aeropuerto.Validar();
        listaAeropuertos.Add(aeropuerto);

    }
    catch (Exception ex)
    {
        throw ex;
    }
}

public Aeropuerto DevolverAeropuerto(string codigolata)
{
    foreach (Aeropuerto unAeropuerto in listaAeropuertos)
    {
        if (unAeropuerto.Codigolata == codigolata) return unAeropuerto;
    }
    throw new Exception("No esta registrado el Aeropuerto");
}

```

```

        }
    public void ExisteAeropuerto(string codigolata)
    {
        if (codigolata.Length != 3) throw new Exception("El codigo lata no puede ser mayor a
3 caracteres");
        bool existe = false;
        foreach (Aeropuerto unAeropuerto in listaAeropuertos)
        {
            if (unAeropuerto.Codigolata == codigolata) existe = true;
        }
        if(!existe) throw new Exception("No esta registrado el Aeropuerto");
    }

}

#endregion

#region Usuario
public Cliente DevolverCliente(string mail)
{
    List<Cliente> aux = ListarClientes();
    foreach (Cliente unCliente in aux)
    {
        if (unCliente.Mail == mail)
        {
            return unCliente;
        }
    }
    throw new Exception("El cliente no existe");
}

public void ExisteUsuario(Usuario unUser)
{
    if (listaUsuarios.Contains(unUser)) throw new Exception("El usuario ya existe");
}

public void AgregarUsuario(Usuario unUser)
{
    try
    {
        unUser.Validar();
        ExisteUsuario(unUser);
        listaUsuarios.Add(unUser);
    }
    catch (Exception ex)
    {
        throw ex;
    }
}

```

```

        }

    }

#endregion

#region Listados
public List<Cliente> ListarClientes()
{
    List<Cliente> listaClientes = new List<Cliente>();
    foreach(Usuario unUser in listaUsuarios)
    {
        if (unUser is Cliente) listaClientes.Add((Cliente)unUser);
    }
    return listaClientes;
}

public List<Vuelo> ListarVuelos(string codigolata)
{
    List<Vuelo> aux = new List<Vuelo>();

    ExisteAeropuerto(codigolata);

    foreach (Vuelo unVuelo in listaVuelos)
    {
        if (unVuelo.Ruta.AeropuertoSalida.Codigolata == codigolata ||
unVuelo.Ruta.AeropuertoLlegada.Codigolata == codigolata)
        {
            aux.Add(unVuelo);
        }
    }
    return aux;
}

public List<Pasaje> ListarPasajes(DateTime fecha1, DateTime fecha2)
{
    if(fecha1 > fecha2 && fecha2 < fecha1) throw new Exception("La fecha 1 no puede
ser mayor a la fecha 2");

    List<Pasaje> aux = new List<Pasaje>();
    foreach (Pasaje unPasaje in listaPasajes)
    {
        if (unPasaje.Fecha >= fecha1 && unPasaje.Fecha <= fecha2) {
            aux.Add(unPasaje);
            unPasaje.Validar();
        }
    }
}

```

```

        else
    {
        throw new Exception("No existen pasajes en esas fechas");
    }
}
return aux;
}
#endregion
}
}

```

[Usuario.cs](#)

```

namespace Dominio
{
    public abstract class Usuario
    {

        string mail;
        string password;

        public string Mail { get => mail; set => mail = value; }
        public string Password { get => password; set => password = value; }
        public Usuario()
        {

        }

        public Usuario(string mail, string password)
        {
            Mail = mail;
            Password = password;
        }

        public virtual void Validar() // Metodo polimorfico.
        {
            if (string.IsNullOrEmpty(Mail)) throw new Exception("El mail es obligatorio");
            if (string.IsNullOrEmpty(Password)) throw new Exception("La contraseña es obligatoria");
        }

        public override bool Equals(object? obj)
        {
            return obj is Usuario unUser && unUser.Mail == Mail;
        }

    }
}

```

```
}
```

Vuelo.cs

```
namespace Dominio
{
    public class Vuelo
    {
        string nroVuelo;
        Ruta ruta;
        Avion avion;
        List<DayOfWeek> frecuencia = new List<DayOfWeek>();

        public string NroVuelo { get => nroVuelo; set => nroVuelo = value; }
        public Ruta Ruta { get => ruta; set => ruta = value; }
        public Avion Avion { get => avion; set => avion = value; }
        public List<DayOfWeek> Frecuencia { get => frecuencia; set => frecuencia = value; }

        public Vuelo()
        {

        }

        public Vuelo(string nroVuelo, Ruta ruta, Avion avion, List<DayOfWeek> frecuencia)
        {
            NroVuelo = nroVuelo;
            Ruta = ruta;
            Avion = avion;
            Frecuencia = frecuencia;
        }

        public void Validar()
        {
            ValidarNroVuelo();
            ValidarAlcance();
            TieneFrecuencia();
        }

        private void ValidarNroVuelo()
        {
            if (nroVuelo.Length > 6) throw new Exception("El numero de vuelo no puede ser mayor a 6 caracteres.");
            if (string.IsNullOrEmpty(nroVuelo)) throw new Exception("El numero de vuelo no puede ser vacio");
            int cantLetras = 0;
```

```

int cantNumeros = 0;
foreach (char c in nroVuelo)
{
    if (char.IsLetter(c))
        cantLetras++;
    else if (char.IsNumber(c))
        cantNumeros++;
    else
        throw new Exception("El número de vuelo solo puede contener letras y
números");
}
if (cantLetras != 2) throw new Exception("El numero de vuelo tiene que tener 2
letras");
if (cantNumeros < 1 || cantNumeros > 4) throw new Exception("La cantidad de
numeros del vuelo tiene que ser entre 1 y 4");
}

private void ValidarAlcance()
{
    if (avion.Alcance < ruta.Distancia) throw new Exception("El avion no tiene el
suficiente alcance para cubrir la distancia a la ruta");
}

private void TieneFrecuencia()
{
    if (frecuencia.Count == 0) throw new Exception("La frecuencia del vuelo debe ser
como minimo 1 vez a la semana.");
}

private string MostrarFrecuencia()
{
    string dias = "";
    foreach (DayOfWeek dia in Frecuencia)
    {
        dias += $"{dia}, ";
    }
    return dias;
}

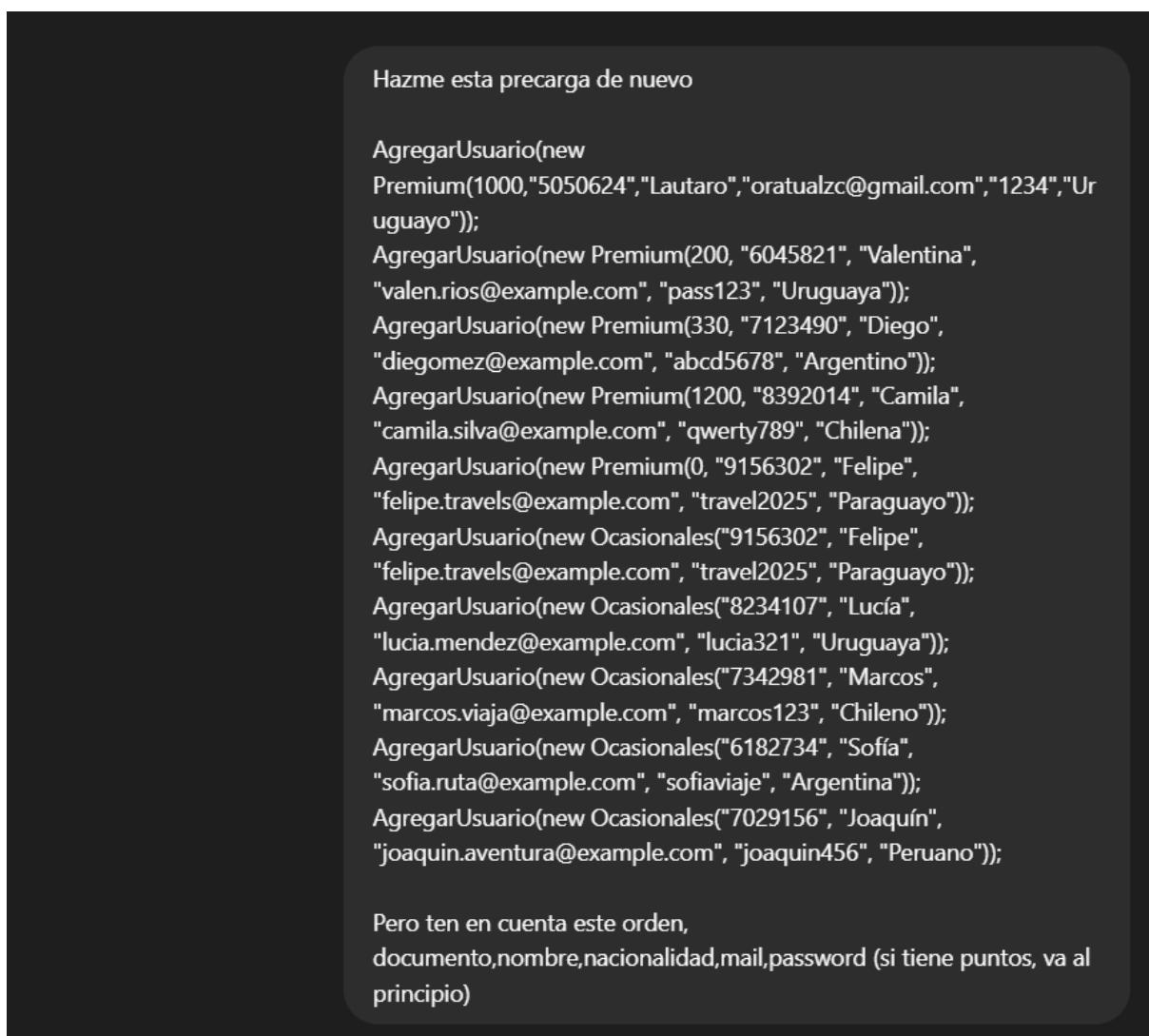
public override string ToString()
{
    return $"{NroVuelo} , {Avion}, {Ruta}, {MostrarFrecuencia()} \n";
}

}
}

```

Evidencia de las consultas de IA para la precarga

Para la precarga de Usuarios:



Claro, aquí tenés la precarga corregida con el orden solicitado: **documento, nombre, nacionalidad, mail, password**, y cuidando que si el mail tiene puntos, van al principio de la línea.

csharp

Copiar Editar

```
AgregarUsuario(new Premium(1000, "5050624", "Lautaro", "oratualzc@gmail.com", "1234", "Uruguayo")
AgregarUsuario(new Premium(200, "6045821", "Valentina", "valen.rios@example.com", "pass123", "Uruguayo")
AgregarUsuario(new Premium(330, "7123490", "Diego", "diegomez@example.com", "abcd5678", "Argentino")
AgregarUsuario(new Premium(1200, "8392014", "Camila", "camila.silva@example.com", "qwerty789", "Chileno")
AgregarUsuario(new Premium(0, "9156302", "Felipe", "felipe.travels@example.com", "travel2025", "Paraguayo")

AgregarUsuario(new Ocasionales("9156302", "Felipe", "Paraguayo", "felipe.travels@example.com", "travel2025", "Paraguayo")
AgregarUsuario(new Ocasionales("8234107", "Lucía", "Uruguaya", "lucia.mendez@example.com", "lucia123", "Uruguaya")
AgregarUsuario(new Ocasionales("7342981", "Marcos", "Chileno", "marcos.viaja@example.com", "marcos123", "Chileno")
AgregarUsuario(new Ocasionales("6182734", "Sofía", "Argentina", "sofia.ruta@example.com", "sofiav123", "Argentina")
AgregarUsuario(new Ocasionales("7029156", "Joaquín", "Peruano", "joaquin.aventura@example.com", "peru123", "Peruano")
```

¿Querés que también corrija o regenere otra parte de tu precarga?

Para la precarga de Aeropuertos:

Primero necesito crear los aeropuertos y utilizar estos para crear las rutas, te paso mi código de agregar aeropuerto para que hagas la precarga estática de estos. Necesito 20 aeropuertos

```
public void AgregarAeropuerto(string codigolata, string ciudad,  
decimal costoOperacion, decimal costoTasas)  
{  
    try  
    {  
        Aeropuerto aeropuerto = new Aeropuerto(codigolata, ciudad,  
costoOperacion, costoTasas);  
        aeropuerto.Validar();  
        listaAeropuertos.Add(aeropuerto);  
  
    }  
    catch (Exception ex)  
    {  
        throw ex;  
    }  
}
```

Perfecto, a continuación te paso la **precarga de 20 aeropuertos** usando directamente el método

`AgregarAeropuerto(...)` sin necesidad de usar variables intermedias. Esta es una precarga estática pura:

csharp

Copiar

Editar

```
public void PrecargarAeropuertos()
{
    AgregarAeropuerto("MVD", "Montevideo", 1500.00m, 500.00m);
    AgregarAeropuerto("EZE", "Buenos Aires - Ezeiza", 1800.00m, 600.00m);
    AgregarAeropuerto("AEP", "Buenos Aires - Aeroparque", 1700.00m, 550.00m);
    AgregarAeropuerto("PDP", "Punta del Este", 1200.00m, 400.00m);
    AgregarAeropuerto("GRU", "São Paulo", 2000.00m, 700.00m);
    AgregarAeropuerto("SCL", "Santiago de Chile", 1900.00m, 650.00m);
    AgregarAeropuerto("LIM", "Lima", 2100.00m, 750.00m);
    AgregarAeropuerto("ASU", "Asunción", 1600.00m, 500.00m);
    AgregarAeropuerto("MIA", "Miami", 3000.00m, 1000.00m);
    AgregarAeropuerto("JFK", "Nueva York", 3500.00m, 1200.00m);
    AgregarAeropuerto("MAD", "Madrid", 3200.00m, 1100.00m);
    AgregarAeropuerto("BCN", "Barcelona", 3100.00m, 1050.00m);
    AgregarAeropuerto("LAX", "Los Ángeles", 3600.00m, 1250.00m);
    AgregarAeropuerto("DXB", "Dubái", 4000.00m, 1400.00m);
    AgregarAeropuerto("PEK", "Pekín", 4200.00m, 1500.00m);
    AgregarAeropuerto("COR", "Córdoba", 1550.00m, 480.00m);
    AgregarAeropuerto("MDZ", "Mendoza", 1580.00m, 500.00m);
    AgregarAeropuerto("CYD", "Colonia del Sacramento", 1100.00m, 390.00m);
    AgregarAeropuerto("SJ0", "Salto", 1150.00m, 410.00m);
    AgregarAeropuerto("RIO", "Río de Janeiro", 1950.00m, 670.00m);
}
```

Para la precarga de Aviones:

Necesito hacer una precarga de datos en c# para mi lista de aviones llamada "listaAviones"

Este es mi metodo de AgregarAvion

```
public void AgregarAvion(string fabricante, string modelo, int cantAsientos, int alcance, decimal costoOperacionKm)
{
    try
    {
        Avion avion = new Avion(fabricante, modelo, cantAsientos,
        alcance, costoOperacionKm);
        ExisteAvion(avion);
        avion.Validar();
        listaAviones.Add(avion);

    }
    catch (Exception ex)
    {
        throw ex;
    }
}
```

Hazme una precarga de 4 aviones con todos los datos solicitados

Claro, aquí tienes una precarga de 4 aviones usando tu método `AgregarAvion`. Puedes colocar este código dentro de un método de inicialización (por ejemplo, `PrecargarAviones()`), o dentro del constructor si es apropiado:

csharp Copiar Editar

```
public void PrecargarAviones()
{
    AgregarAvion("Boeing", "737 MAX", 189, 6570, 5.6m);
    AgregarAvion("Airbus", "A320neo", 180, 6300, 5.2m);
    AgregarAvion("Embraer", "E195-E2", 132, 4815, 4.1m);
    AgregarAvion("Bombardier", "CS300", 160, 6700, 4.8m);
}
```

Para la precarga de Rutas:

Necesito hacer otra precarga de Rutas, estes el metodo para agregar ruta:

```
public void AgregarRuta(Aeropuerto aeropuertoSalida, Aeropuerto aeropuertoLlegada, int distancia)
{
    try
    {
        Ruta ruta = new
Ruta(aeropuertoSalida,aeropuertoLlegada,distancia);
        ruta.Validar();
        listaRutas.Add(ruta);

    }
    catch (Exception ex)
    {
        throw ex;
    }

}
```

Necesito la precarga de 30 rutas con esos datos que se especifican

csharp

Copiar

Editar

```
public void PrecargarRutas()
{
    AgregarRuta(DevolverAeropuerto("MVD"), DevolverAeropuerto("EZE"), 200);
    AgregarRuta(DevolverAeropuerto("MVD"), DevolverAeropuerto("GRU"), 1570);
    AgregarRuta(DevolverAeropuerto("MVD"), DevolverAeropuerto("SCL"), 1370);
    AgregarRuta(DevolverAeropuerto("EZE"), DevolverAeropuerto("LIM"), 3140);
    AgregarRuta(DevolverAeropuerto("GRU"), DevolverAeropuerto("MIA"), 6540);
    AgregarRuta(DevolverAeropuerto("SCL"), DevolverAeropuerto("MIA"), 6620);
    AgregarRuta(DevolverAeropuerto("PDP"), DevolverAeropuerto("AEP"), 320);
    AgregarRuta(DevolverAeropuerto("ASU"), DevolverAeropuerto("EZE"), 1080);
    AgregarRuta(DevolverAeropuerto("MVD"), DevolverAeropuerto("LAX"), 9900);
    AgregarRuta(DevolverAeropuerto("MVD"), DevolverAeropuerto("JFK"), 8600);
    AgregarRuta(DevolverAeropuerto("JFK"), DevolverAeropuerto("LAX"), 3980);
    AgregarRuta(DevolverAeropuerto("MAD"), DevolverAeropuerto("BCN"), 620);
    AgregarRuta(DevolverAeropuerto("MIA"), DevolverAeropuerto("BCN"), 7450);
    AgregarRuta(DevolverAeropuerto("DXB"), DevolverAeropuerto("PEK"), 5840);
    AgregarRuta(DevolverAeropuerto("COR"), DevolverAeropuerto("MDZ"), 660);
    AgregarRuta(DevolverAeropuerto("MDZ"), DevolverAeropuerto("SCL"), 480);
    AgregarRuta(DevolverAeropuerto("CYD"), DevolverAeropuerto("MVD"), 170);
    AgregarRuta(DevolverAeropuerto("SJO"), DevolverAeropuerto("PDP"), 550);
    AgregarRuta(DevolverAeropuerto("PEK"), DevolverAeropuerto("LAX"), 10000);
    AgregarRuta(DevolverAeropuerto("PEK"), DevolverAeropuerto("JFK"), 11000);
    AgregarRuta(DevolverAeropuerto("DXB"), DevolverAeropuerto("MAD"), 5700);
    AgregarRuta(DevolverAeropuerto("RIO"), DevolverAeropuerto("GRU"), 430);
    AgregarRuta(DevolverAeropuerto("RIO"), DevolverAeropuerto("MVD"), 1820);
    AgregarRuta(DevolverAeropuerto("COR"), DevolverAeropuerto("AEP"), 660);
    AgregarRuta(DevolverAeropuerto("CYD"), DevolverAeropuerto("EZE"), 190);
    AgregarRuta(DevolverAeropuerto("SJO"), DevolverAeropuerto("ASU"), 1080);
    AgregarRuta(DevolverAeropuerto("PDP"), DevolverAeropuerto("RIO"), 1930);
    AgregarRuta(DevolverAeropuerto("MVD"), DevolverAeropuerto("LIM"), 3120);
    AgregarRuta(DevolverAeropuerto("MIA"), DevolverAeropuerto("JFK"), 1760);
    AgregarRuta(DevolverAeropuerto("MAD"), DevolverAeropuerto("JFK"), 5760);
}
```

Para la precarga de Vuelos:

Necesito hacer una precarga de vuelos estatica, con el siguiente formato:

Este es el metodo

```
public void AgregarVuelo(string nroVuelo, Aeropuerto  
aeropuertoSalida, Aeropuerto aeropuertoLlegada, Avion avion,  
List<DayOfWeek> frecuencia)  
{  
    try  
    {  
        ExisteVuelo(nroVuelo);  
        ExisteAvion(avion);  
        Vuelo vuelo = new Vuelo(nroVuelo, ExisteRuta(aeropuertoSalida,  
aeropuertoLlegada), avion, frecuencia);  
        vuelo.Validar();  
        listaVuelos.Add(vuelo);  
    }  
    catch (Exception ex)  
    {  
        throw ex;  
    }  
}
```

Necesito hacer 30 vuelos



Gracias por la información. Aquí tienes una **precarga estática de 30 vuelos** utilizando el método

`AgregarVuelo`, asumiendo que ya tienes cargados:

- 20 aeropuertos (a través de `DevolverAeropuerto("XXX")` por código IATA),
- al menos 4 aviones (`listaAviones[0]` a `listaAviones[3]`),
- y que la función `ExisteRuta` valida correctamente la ruta entre aeropuerto de salida y llegada.

Perfecto, gracias por compartir los códigos de IATA pre cargados. A continuación, te dejo 30 vuelos estáticos usando exclusivamente los aeropuertos que ya tenés registrados.

csharp

Copiar Editar

```
AgregarVuelo("AR101", DevolverAeropuerto("MVD"), DevolverAeropuerto("EZE"), listaAviones[0], new  
AgregarVuelo("AR102", DevolverAeropuerto("EZE"), DevolverAeropuerto("MVD"), listaAviones[0], new  
  
AgregarVuelo("LA201", DevolverAeropuerto("GRU"), DevolverAeropuerto("SCL"), listaAviones[1], new  
AgregarVuelo("LA202", DevolverAeropuerto("SCL"), DevolverAeropuerto("GRU"), listaAviones[1], new  
  
AgregarVuelo("UY301", DevolverAeropuerto("PDP"), DevolverAeropuerto("MVD"), listaAviones[2], new  
AgregarVuelo("UY302", DevolverAeropuerto("MVD"), DevolverAeropuerto("PDP"), listaAviones[2], new  
  
AgregarVuelo("LAT401", DevolverAeropuerto("LIM"), DevolverAeropuerto("SCL"), listaAviones[3], new  
AgregarVuelo("LAT402", DevolverAeropuerto("SCL"), DevolverAeropuerto("LIM"), listaAviones[3], new  
  
AgregarVuelo("JJ501", DevolverAeropuerto("ASU"), DevolverAeropuerto("GRU"), listaAviones[0], new  
AgregarVuelo("JJ502", DevolverAeropuerto("GRU"), DevolverAeropuerto("ASU"), listaAviones[0], new  
  
AgregarVuelo("AA601", DevolverAeropuerto("MIA"), DevolverAeropuerto("JFK"), listaAviones[1], new  
AgregarVuelo("AA602", DevolverAeropuerto("JFK"), DevolverAeropuerto("MIA"), listaAviones[1], new  
  
AgregarVuelo("IB701", DevolverAeropuerto("MAD"), DevolverAeropuerto("BCN"), listaAviones[2], new  
AgregarVuelo("IB702", DevolverAeropuerto("BCN"), DevolverAeropuerto("MAD"), listaAviones[2], new  
  
AgregarVuelo("UX801", DevolverAeropuerto("LAX"), DevolverAeropuerto("MIA"), listaAviones[3], new  
AgregarVuelo("UX802", DevolverAeropuerto("MIA"), DevolverAeropuerto("LAX"), listaAviones[3], new  
  
AgregarVuelo("EK901", DevolverAeropuerto("DXB"), DevolverAeropuerto("MAD"), listaAviones[0], new  
AgregarVuelo("EK902", DevolverAeropuerto("MAD"), DevolverAeropuerto("DXB"), listaAviones[0], new  
  
AgregarVuelo("PEK101", DevolverAeropuerto("PEK"), DevolverAeropuerto("DXB"), listaAviones[1], new  
AgregarVuelo("PEK102", DevolverAeropuerto("DXB"), DevolverAeropuerto("PEK"), listaAviones[1], new  
  
AgregarVuelo("UY901", DevolverAeropuerto("COR"), DevolverAeropuerto("MDZ"), listaAviones[2], new  
AgregarVuelo("UY902", DevolverAeropuerto("MDZ"), DevolverAeropuerto("COR"), listaAviones[2], new  
  
AgregarVuelo("UY911", DevolverAeropuerto("CYD"), DevolverAeropuerto("SJO"), listaAviones[3], new  
AgregarVuelo("UY912", DevolverAeropuerto("SJO"), DevolverAeropuerto("CYD"), listaAviones[3], new  
  
AgregarVuelo("AR111", DevolverAeropuerto("AEP"), DevolverAeropuerto("EZE"), listaAviones[0], new  
AgregarVuelo("AR112", DevolverAeropuerto("EZE"), DevolverAeropuerto("AEP"), listaAviones[0], new
```

Para la precarga de Pasajes:

Necesito hacer la precarga estatica de 25 Pasajes

Ten en cuenta este formato

AgregarPasajes(vuelo,fecha,pasajero,equipaje,precio)

Equipaje es un enum

```
public enum Equipaje
{
    LIGHT =1,
    CABINA = 2,
    BODEGA =3
}
public void AgregarPasaje(Vuelo vuelo, DateTime fecha, Cliente
pasajero, Equipaje equipaje, decimal precio)
{
    try
    {
        Pasaje pasaje = new Pasaje(vuelo, fecha, pasajero, equipaje,
precio);
        pasaje.Validar();
        ValidarPasajero(pasajero);
        // CalcularPrecioPasaje() iría en este momento antes de la
emision del pasaje.
        listaPasajes.Add(pasaje);

    }
    catch (Exception ex)
    {
        throw ex;
    }
}
```

Aquí tienes 25 llamadas a `AgregarPasaje(...)` siguiendo el formato que pediste:

```
csharp

new DateTime(2025, 5, 9), DevolverUsuario("oratualzc@gmail.com"), Equipaje.CABINA, 250);
new DateTime(2025, 5, 13), DevolverUsuario("valen.rios@example.com"), Equipaje.BODEGA, 300);
new DateTime(2025, 5, 14), DevolverUsuario("diegomez@example.com"), Equipaje.CABINA, 270);
new DateTime(2025, 5, 15), DevolverUsuario("camila.silva@example.com"), Equipaje.BODEGA, 320);
new DateTime(2025, 5, 16), DevolverUsuario("felipe.travels@example.com"), Equipaje.CABINA, 310);
new DateTime(2025, 5, 17), DevolverUsuario("lucia.mendez@example.com"), Equipaje.BODEGA, 290);
new DateTime(2025, 5, 17), DevolverUsuario("marcos.viaja@example.com"), Equipaje.CABINA, 280);
new DateTime(2025, 5, 17), DevolverUsuario("sofia.ruta@example.com"), Equipaje.CABINA, 260);
new DateTime(2025, 5, 11), DevolverUsuario("joaquin.aventura@example.com"), Equipaje.BODEGA, 270);
new DateTime(2025, 5, 12), DevolverUsuario("oratualzc@gmail.com"), Equipaje.CABINA, 245);
new DateTime(2025, 5, 13), DevolverUsuario("valen.rios@example.com"), Equipaje.BODEGA, 330);
new DateTime(2025, 5, 16), DevolverUsuario("diegomez@example.com"), Equipaje.CABINA, 220);
new DateTime(2025, 5, 17), DevolverUsuario("camila.silva@example.com"), Equipaje.BODEGA, 210);
new DateTime(2025, 5, 18), DevolverUsuario("felipe.travels@example.com"), Equipaje.CABINA, 280);
new DateTime(2025, 5, 19), DevolverUsuario("lucia.mendez@example.com"), Equipaje.BODEGA, 270);
new DateTime(2025, 5, 13), DevolverUsuario("marcos.viaja@example.com"), Equipaje.CABINA, 350);
new DateTime(2025, 5, 14), DevolverUsuario("sofia.ruta@example.com"), Equipaje.BODEGA, 290);
new DateTime(2025, 5, 15), DevolverUsuario("joaquin.aventura@example.com"), Equipaje.CABINA, 260);
new DateTime(2025, 5, 16), DevolverUsuario("oratualzc@gmail.com"), Equipaje.BODEGA, 250);
new DateTime(2025, 5, 17), DevolverUsuario("valen.rios@example.com"), Equipaje.CABINA, 270);
new DateTime(2025, 5, 18), DevolverUsuario("diegomez@example.com"), Equipaje.BODEGA, 310);
new DateTime(2025, 5, 19), DevolverUsuario("camila.silva@example.com"), Equipaje.CABINA, 300);
new DateTime(2025, 5, 13), DevolverUsuario("felipe.travels@example.com"), Equipaje.BODEGA, 290);
new DateTime(2025, 5, 14), DevolverUsuario("lucia.mendez@example.com"), Equipaje.CABINA, 280);
new DateTime(2025, 5, 15), DevolverUsuario("joaquin.aventura@example.com"), Equipaje.BODEGA, 320);
```