

Universidad ORT Uruguay
Facultad de Ingeniería
Escuela de Tecnología

OBLIGATORIO PROGRAMACIÓN 2

DOCUMENTO DE ANÁLISIS



IGNACIO MARICHAL DEL SUR - 353739



RODRIGO PINTOS – 346421

M2B

Docente: LUCAS LÓPEZ

ANALISTA TECNOLOGÍAS DE LA INFORMACIÓN

Fecha de entrega del documento 26/06/2025

Link Azure: <https://obligatoriop2rpim.azurewebsites.net/>

Link GitHub: <https://github.com/MaxdemonIM/obligatorio-programacion-2>

Índice

1. Introducción general del problema a resolver y Diagrama UML Dominio actualizado	2
2. Funcionalidades Implementadas	3
2.1 Acceso y Registro	3
2.2 Funcionalidades para Usuario Cliente (Ocasional o Premium)	5
2.3 Funcionalidades para Administrador	7
3. Validaciones Relevantes Agregadas o Modificadas	8
4. Controladores y Filtros	9
5. Métodos Auxiliares Nuevos	10
6. Precarga de Datos:	10
6.1 Tabla de Precarga de Datos:	11
6.1.1 Administradores:	11
6.1.2 Pasajeros premium:	12
6.1.3 Pasajeros ocasionales:	12
6.1.4 Aviones:	13
6.1.5 Aeropuertos:	14
6.1.6 Rutas:	15
6.1.7 Vuelos:	17
6.1.8 Pasajes:	19
7. Diagrama Casos de Uso:	20
8. Prompts consultas CHAT GPT:	21
9. Tabla de testing	21
10.Código fuente comentado de toda la aplicación:	23

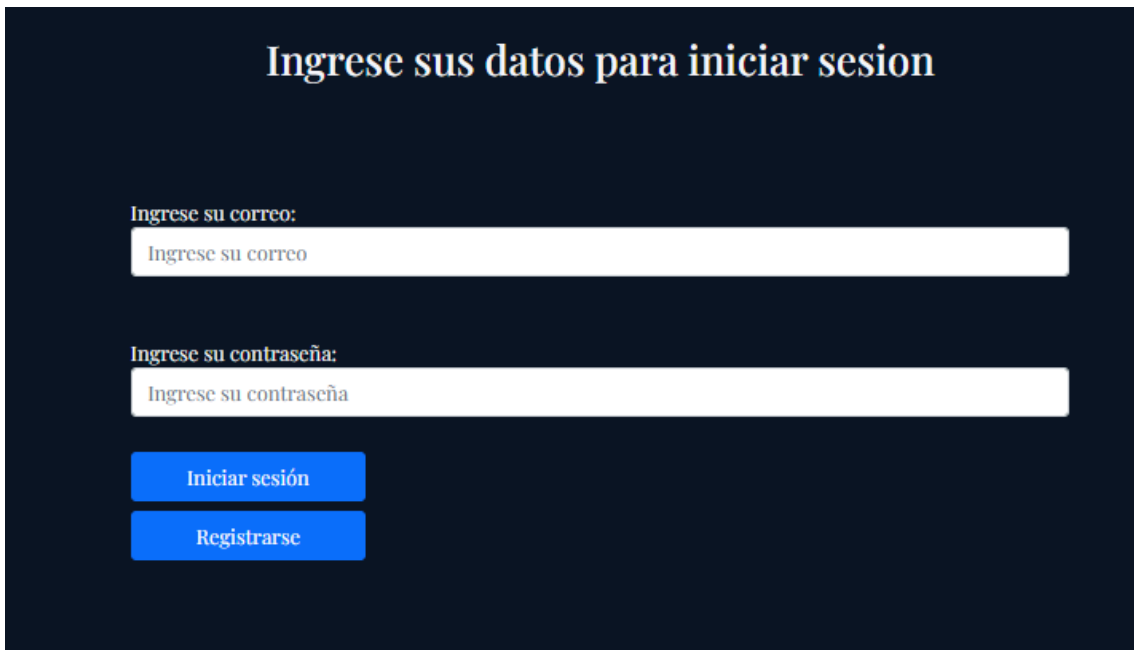
Esta segunda parte del sistema de gestión de aerolínea extiende la funcionalidad previamente desarrollada en la versión de consola, implementando una aplicación web en ASP.NET Core MVC 8.0. El objetivo principal es habilitar el acceso autenticado a las funcionalidades del sistema, discriminando qué operaciones puede realizar cada tipo de usuario (anónimo, pasajero o administrador). Asimismo, se conserva la lógica de negocio del dominio ya implementada y se expone a través de controladores, vistas y filtros personalizados, como hemos venido dando en el curso.



2. Funcionalidades Implementadas

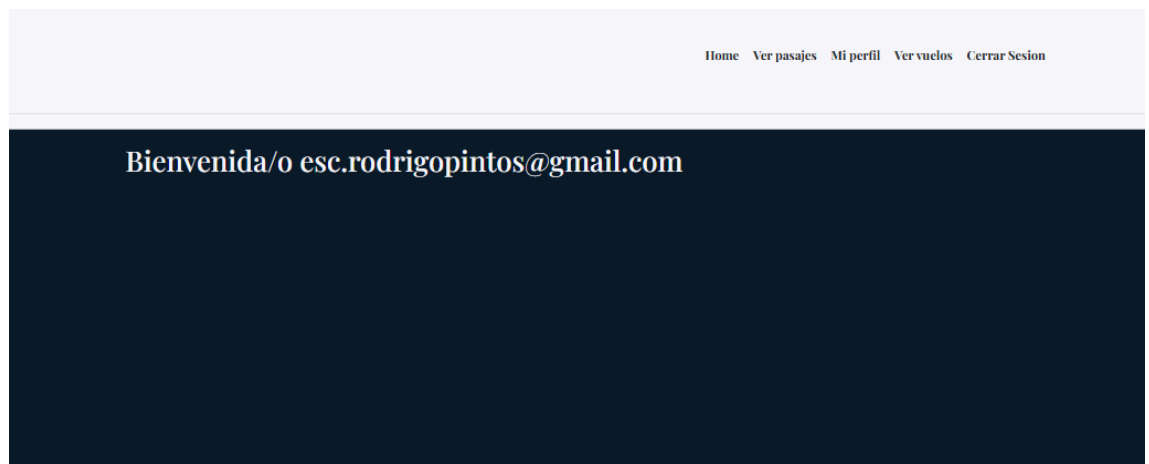
2.1 Acceso y Registro

- **Login:** a través del controlador **HomeController**, se valida el email y contraseña. Si las credenciales son válidas, se guarda la sesión con el email y el rol del usuario.



The image shows a login form with a dark blue background. At the top, the text "Ingrese sus datos para iniciar sesion" is displayed in a light blue font. Below this, there are two input fields. The first is labeled "Ingrese su correo:" and the second is labeled "Ingrese su contraseña:". Both fields have a light blue border and placeholder text. Below the input fields, there are two blue buttons: "Iniciar sesión" and "Registrarse".

- **Logout:** elimina la sesión del usuario.



The image shows a user profile interface. At the top, there is a light blue navigation bar with links: "Home", "Ver pasajes", "Mi perfil", "Ver vuelos", and "Cerrar Sesion". Below this, the main content area has a dark blue background. The text "Bienvenida/o esc.rodrigopintos@gmail.com" is displayed in a light blue font.

- **Registro de Ocasional:** un usuario anónimo puede registrarse como cliente ocasional desde **OcasionalController**, completando un formulario con validaciones.



The image shows a registration form titled "REGISTRARSE" on a dark blue background. The form contains the following fields and labels:

- Nacionalidad**: Ingrese nacionalidad..
- Documento de identidad**: Ingrese su documento de identidad..
- Nombre**: Ingrese su nombre..
- Ingrese su Contraseña**: Ingrese su contraseña..
- Ingrese su email**: Ingrese su email..

At the bottom of the form, there are two buttons: "Registrarse" (highlighted in blue) and "Volver" (outlined in blue).

2.2 Funcionalidades para Usuario Cliente (Ocasional o Premium)

- **Ver vuelos:** se listan todos los vuelos disponibles y se permite filtrar por código IATA de salida/llegada. Cada vuelo permite acceder al detalle.

Home
Ver pasajes
Mi perfil
Ver vuelos
Cerrar Sesión

LISTA DE VUELOS:

Elija un codigo de aeropuerto de salida
Elija un codigo de aeropuerto de llegada
Filtrar

NUMERO DE VUELO	ROUTA	FRECUENCIA			
1	EZE - MVD	Tuesday	Ver detalle	dd/mm/aaaa	Comprar pasaje
2	LIM - SCL	Wednesday	Ver detalle	dd/mm/aaaa	Comprar pasaje
3	GIG - BOG	Saturday	Ver detalle	dd/mm/aaaa	Comprar pasaje
4	UIO - GRU	Monday	Ver detalle	dd/mm/aaaa	Comprar pasaje
5	PTY - CCS	Tuesday	Ver detalle	dd/mm/aaaa	Comprar pasaje
6	LPB - ASU	Wednesday	Ver detalle	dd/mm/aaaa	Comprar pasaje
7	MIA - MEX	Thursday	Ver detalle	dd/mm/aaaa	Comprar pasaje
8	FCO - MAD	Friday	Ver detalle	dd/mm/aaaa	Comprar pasaje
9	LIS - CDG	Saturday	Ver detalle	dd/mm/aaaa	Comprar pasaje
10	YYZ - LHR	Sunday	Ver detalle	dd/mm/aaaa	Comprar pasaje
11	SCL - MVD	Monday	Ver detalle	dd/mm/aaaa	Comprar pasaje
12	GIG - LIM	Tuesday	Ver detalle	dd/mm/aaaa	Comprar pasaje
13	PTY - GRU	Wednesday	Ver detalle	dd/mm/aaaa	Comprar pasaje
14	MIA - ASU	Thursday	Ver detalle	dd/mm/aaaa	Comprar pasaje
15	FCO - MEX	Friday	Ver detalle	dd/mm/aaaa	Comprar pasaje

- Detalle de vuelo:** se muestra la ruta, frecuencia, precio sin equipaje y un formulario para comprar pasaje.

Vuelo EZE - MVD
Precio : \$149,34
Frecuencia:
Tuesday

Volver

dd/mm/aaaa

LIGHT
CABINA
BODEGA

Comprar pasaje

6

- **Comprar pasaje:** validando la frecuencia del vuelo y el tipo de equipaje obligatorio. Se calcula el precio total según el tipo de pasajero y se guarda el pasaje.
- **Ver mis pasajes:** se listan los pasajes del usuario logueado, ordenados por precio descendente.

Home Ver pasajes Mi perfil Ver vuelos Cerrar Sesión					
LISTA DE PASAJES:					
ID	RUTA	PASAJERO	FECHA	TIPO EQUIPAJE	PRECIO
25	EZE - MVD	Rodrigo	17/6/2025	CABINA	\$363,29

- **Ver mi perfil:** se muestra la información del pasajero logueado, y si es Premium también los puntos acumulados. (en el caso es ocasional, por lo que los puntos no se ven).

Home Ver pasajes Mi perfil Ver vuelos Cerrar Sesión	
Perfil de: Rodrigo	
Nombre	Rodrigo
Email	esc.rodrigopintos@gmail.com
Nacionalidad	Hola
Documento de identidad	526987ABC

2.3 Funcionalidades para Administrador

- **Ver todos los pasajes:** listado general ordenado por fecha de emisión (ascendente).

LISTA DE PASAJES:

ID	ruta	PASAJERO	FECHA	TIPO EQUIPAJE	PRECIO
25	EZE - MVD	Rodrigo	17/6/2025	CABINA	\$363,39
0	EZE - MVD	Lucía González	5/5/2026	CABINA	\$351,34
1	LIM - SCL	Carlos Pérez	6/5/2026	LIGHT	\$321,95
2	GIG - BOG	Ana Torres	9/5/2026	BODEGA	\$384,21
3	UIO - GRU	Mateo Fernández	11/5/2026	CABINA	\$368,01
4	PTY - CCS	Sofía Ramírez	12/5/2026	LIGHT	\$369,31
5	LPB - ASU	Tomás López	13/5/2026	BODEGA	\$405,98
6	MIA - MEX	Camila García	14/5/2026	CABINA	\$472,28
7	FCO - MAD	Joaquín Sánchez	15/5/2026	LIGHT	\$496,96
8	LIS - CDG	Valentina Rodríguez	16/5/2026	BODEGA	\$465,90
9	YYZ - LHR	Lucía González	17/5/2026	CABINA	\$430,36
10	SCL - MVD	Carlos Pérez	18/5/2026	LIGHT	\$327,38
11	GIG - LIM	Ana Torres	19/5/2026	BODEGA	\$383,27
12	PTY - GRU	Mateo Fernández	20/5/2026	CABINA	\$416,28
13	MIA - ASU	Sofía Ramírez	21/5/2026	LIGHT	\$445,08
14	FCO - MEX	Tomás López	22/5/2026	BODEGA	\$555,53
15	BOG - EZE	Camila García	23/5/2026	CABINA	\$518,03
16	ASU - UIO	Joaquín Sánchez	24/5/2026	LIGHT	\$427,65
17	LIS - MAD	Valentina Rodríguez	25/5/2026	BODEGA	\$526,71

- **Ver y editar clientes:** se listan todos los pasajeros ordenados por documento. Se permite:
 - Editar puntos de un pasajero Premium.
 - Modificar la elegibilidad de un pasajero Ocasional.

LISTA DE PASAJEROS:

NOMBRE	EMAIL	NACIONALIDAD	DOCUMENTO	PUNTOS / ELEGIBLE
Lucía González	lucia.gonzalez@mail.com	Uruguaya	12345678	Tiene o puntos. o <input type="text"/> Actualizar
Carlos Pérez	carlos.perez@mail.com	Argentina	23456789	Tiene o puntos. o <input type="text"/> Actualizar
Ana Torres	ana.torres@mail.com	Chilena	34567890	Tiene o puntos. o <input type="text"/> Actualizar
Mateo Fernández	mateo.fernandez@mail.com	Brasilera	45678901	Tiene o puntos. o <input type="text"/> Actualizar
Sofía Ramírez	sofia.ramirez@mail.com	Colombiana	56789012	Tiene o puntos. o <input type="text"/> Actualizar
Tomás López	tomas.lopez@mail.com	Uruguaya	22334455	No es elegible No <input type="button" value="Actualizar"/>
Camila García	camila.garcia@mail.com	Paraguaya	33445566	No es elegible No <input type="button" value="Actualizar"/>

3. Validaciones Relevantes Agregadas o Modificadas

- **Login:** Se implementa en **Sistema.Login()**. Verifica si el email ingresado corresponde a un usuario registrado. Si existe, compara la contraseña. Si no coincide, lanza una excepción con mensaje personalizado. Esta validación garantiza que solo usuarios existentes accedan al sistema.
- **Password:** La validación se hace en **Usuario.ValidarPassword()**, usada tanto en registro como en login. Requiere una longitud mínima de 8 caracteres, y al menos una letra mayúscula, una minúscula, un número y un símbolo. Se usa cuando se registra un nuevo usuario y en la precarga.
- **Fecha de pasaje:** Validada en **Pasaje.ValidarFecha()** y **ValidarFechaCorrespondeFrecuencia()**. Se usa al crear un pasaje desde la vista de detalle del vuelo. Verifica que la fecha no sea anterior a la actual y que coincida con un día de operación del vuelo. Si no se cumple, lanza excepción y no permite completar la compra.
- **Tipo de equipaje:** En el controlador **PasajeController**, método **Add(...)**, se llama a **Sistema.EsTipoEquipajeValido(...)**. Este método lanza excepción si no se selecciona ningún valor de equipaje. Se aplica durante la compra de pasaje.

4. Controladores y Filtros

- **HomeController:** maneja el acceso al sistema, login, logout y redireccionamiento según rol.
- **OcasionalController:** gestiona el alta de un nuevo cliente ocasional desde una vista pública.
- **VueloController:** permite ver el listado de vuelos, aplicar filtros por aeropuertos y acceder al detalle del vuelo. Este controlador está protegido con el filtro **[SoloPasajero]**.
- **PasajeController:** permite la compra de pasajes desde el detalle de vuelo, y visualizar pasajes según si el usuario es cliente o administrador. Gestiona el ordenamiento por precio o por fecha. (también por usuario).

- **PasajeroController:** muestra el perfil del pasajero logueado y, en caso de ser administrador, permite ver la lista completa de pasajeros con opciones de edición.

Comparadores:

- **CompararPasajePorPrecio:** clase comparadora que implementa `IComparer<Pasaje>`. Permite ordenar los pasajes por precio total en orden descendente, utilizando el método `CalcularPrecioDelPasaje()`. Se aplica en `Sistema.OrdenarPasajesPorPrecio()` para mostrar los pasajes del cliente ordenados de mayor a menor costo.

Filtros:

- **Authentication:** aplicado globalmente a las acciones restringidas. Si no hay sesión activa (email vacío), redirige al login.
- **SoloAdmin:** limita el acceso a funcionalidades exclusivas del rol administrador, como editar puntos o elegibilidad.
- **SoloPasajero:** limita el acceso a funcionalidades de clientes como ver vuelos, perfil y pasajes.

5. Métodos Auxiliares Nuevos

- **ActualizarPuntosPremium(int puntos, string email):** Permite modificar los puntos de un pasajero premium desde la vista de lista de pasajeros. Se ejecuta desde `PasajeroController.Index()` mediante formulario.
- **ActualizarElegibilidadOcasional(string email, bool nuevoEstado):** Modifica el valor de elegibilidad de un pasajero ocasional. También se invoca desde `PasajeroController.Index()` cuando el administrador marca o desmarca la casilla correspondiente.

- **ObtenerVueloPorNumVuelo(int num):** Nos da un vuelo según su número único. Se utiliza al acceder al detalle del vuelo (**VueloController.Details()**) y al crear un nuevo pasaje (**PasajeController.Add()**).
- **OrdenarPasajes():** Organiza todos los pasajes del sistema por fecha de emisión en orden ascendente. Se aplica cuando un administrador visualiza la lista completa de pasajes.
- **OrdenarPasajesPorPrecio():** Ordena los pasajes comprados por un pasajero en orden descendente de precio. Lo usamos en **PasajeController.VerPasajesUsuario()**.
- **EsTipoEquipajeValido(string equipaje):** Valida que el tipo de equipaje ingresado sea válido y no vacío. Se invoca durante la validación de datos en **PasajeController.Add()**.
- **ObtenerListaPasajeDeUsuario(string email):** Devuelve la lista de pasajes asociados a un pasajero identificado por su email. Se utiliza en la vista de pasajes del cliente para mostrar solo sus compras.

6. Precarga de Datos:

Para facilitar el testeo y uso del sistema, se realiza una precarga en el método **PrecargarDatos()** de la clase Sistema.

Esta incluye:

- 2 administradores
- 5 pasajeros premium
- 5 pasajeros ocasionales
- 4 aviones
- 20 aeropuertos

- 30 rutas
- 30 vuelos
- 25 pasajes

6.1 Tabla de Precarga de Datos:

6.1.1 Administradores:

	Nombre	Contraseña	Email
1	admin1	Admin123@	admin1@empresa.com
2	admin2	Admin456@	admin2@empresa.com

6.1.2 Pasajeros premium:

Pasajeros Premium



	Nombre	Nacionalidad	Documento	Contraseña	Email
1	Lucía González	Uruguay	12345678	Password1.	lucia.gonzalez@mail.com
2	Carlos Pérez	Argentina	23456789	Password2@	carlos.perez@mail.com
3	Ana Torres	Chilena	34567890	Password3@	ana.torres@mail.com
4	Mateo Fernández	Brasilera	45678901	Password4@	mateo.fernandez@mail.com
5	Sofía Ramírez	Colombiana	56789012	Password5@	sofia.ramirez@mail.com

6.1.3 Pasajeros ocasionales:

	Nombre	Nacionalidad	Documento	Contraseña	Email
1	Tomás López	Uruguaya	22334455	Password6@	tomas.lopez@mail.com
2	Camila García	Paraguaya	33445566	Password7@	camila.garcia@mail.com
3	Joaquín Sánchez	Mexicana	44556677	Password8@	joaquin.sanchez@mail.com
4	Valentina Rodríguez	Venezolana	55667788	Password9@	valentina.rodriguez@mail.com
5	Martín Martínez	Boliviana	66778899	Password10@	martin.martinez@mail.com

6.1.4 Aviones:

Aviones

	Fabricante	Modelo	Alcance (km)	Capacidad	Velocidad (km/h)	Tipo
1	Boeing	737	5000	180	10.5	Narrow-body
2	Airbus	A320	4800	170	9.8	Narrow-body
3	Embraer	E190	4000	100	7.2	Regional
4	Bombardier	CRJ900	3700	90	6.9	Regional

6.1.5 Aeropuertos:

	Código IATA	Ciudad	Latitud	Longitud
1	MVD	Montevideo	250	100
2	EZE	Buenos Aires	255	102
3	SCL	Santiago	260	104
4	LIM	Lima	265	106
5	BOG	Bogotá	270	108
6	GIG	Río	275	110
7	GRU	São Paulo	280	112
8	UIO	Quito	285	114
9	CCS	Caracas	290	116
10	PTY	Panamá	295	118
11	ASU	Asunción	300	120
12	LPB	La Paz	305	122
13	MEX	Ciudad de México	310	124
14	MIA	Miami	315	126
15	MAD	Madrid	320	128
16	FCO	Roma	325	130
17	CDG	París	330	132
18	LIS	Lisboa	335	134
19	LHR	Londres	340	136
20	YYZ	Toronto	345	138

6.1.6 Rutas:

	Origen	Destino	Distancia (km)
1	Montevideo	Buenos Aires	2000
2	Santiago	Lima	1500
3	Bogotá	Río	1700
4	São Paulo	Quito	1400
5	Caracas	Panamá	1800
6	Asunción	La Paz	1900
7	Ciudad de México	Miami	2200
8	Madrid	Roma	2400
9	París	Lisboa	2300
10	Londres	Toronto	2100
11	Montevideo	Santiago	1300
12	Lima	Río	1600
13	São Paulo	Panamá	2500
14	Asunción	Miami	2700
15	Ciudad de México	Roma	2800
16	Buenos Aires	Bogotá	2900
17	Quito	Asunción	2600
18	Madrid	Lisboa	3100
19	Caracas	La Paz	3300

20	Río	Londres	3000
21	Santiago	São Paulo	2700
22	Lima	Quito	2800
23	Buenos Aires	Caracas	3000
24	Bogotá	Asunción	2700
25	Miami	Roma	3400
26	París	Toronto	3600
27	La Paz	Londres	3800
28	Panamá	Lisboa	3500
29	Ciudad de México	Madrid	3200
30	Montevideo	Toronto	3900

6.1.7 Vuelos:

	Número de Vuelo	Avión	Ruta	Frecuencia
1	1	Boeing 737	Montevideo - Buenos Aires	Tuesday
2	2	Airbus A320	Santiago - Lima	Wednesday
3	3	Embraer E190	Bogotá - Río	Saturday
4	4	Bombardier CRJ900	São Paulo - Quito	Monday
5	5	Boeing 737	Caracas - Panamá	Tuesday
6	6	Airbus A320	Asunción - La Paz	Wednesday
7	7	Embraer E190	Ciudad de México - Miami	Thursday
8	8	Bombardier CRJ900	Madrid - Roma	Friday
9	9	Boeing 737	París - Lisboa	Saturday
10	10	Airbus A320	Londres - Toronto	Sunday
11	11	Embraer E190	Montevideo - Santiago	Monday
12	12	Bombardier CRJ900	Lima - Río	Tuesday
13	13	Boeing 737	São Paulo - Panamá	Wednesday
14	14	Airbus A320	Asunción - Miami	Thursday
15	15	Embraer E190	Ciudad de México - Roma	Friday
16	16	Bombardier CRJ900	Buenos Aires - Bogotá	Saturday

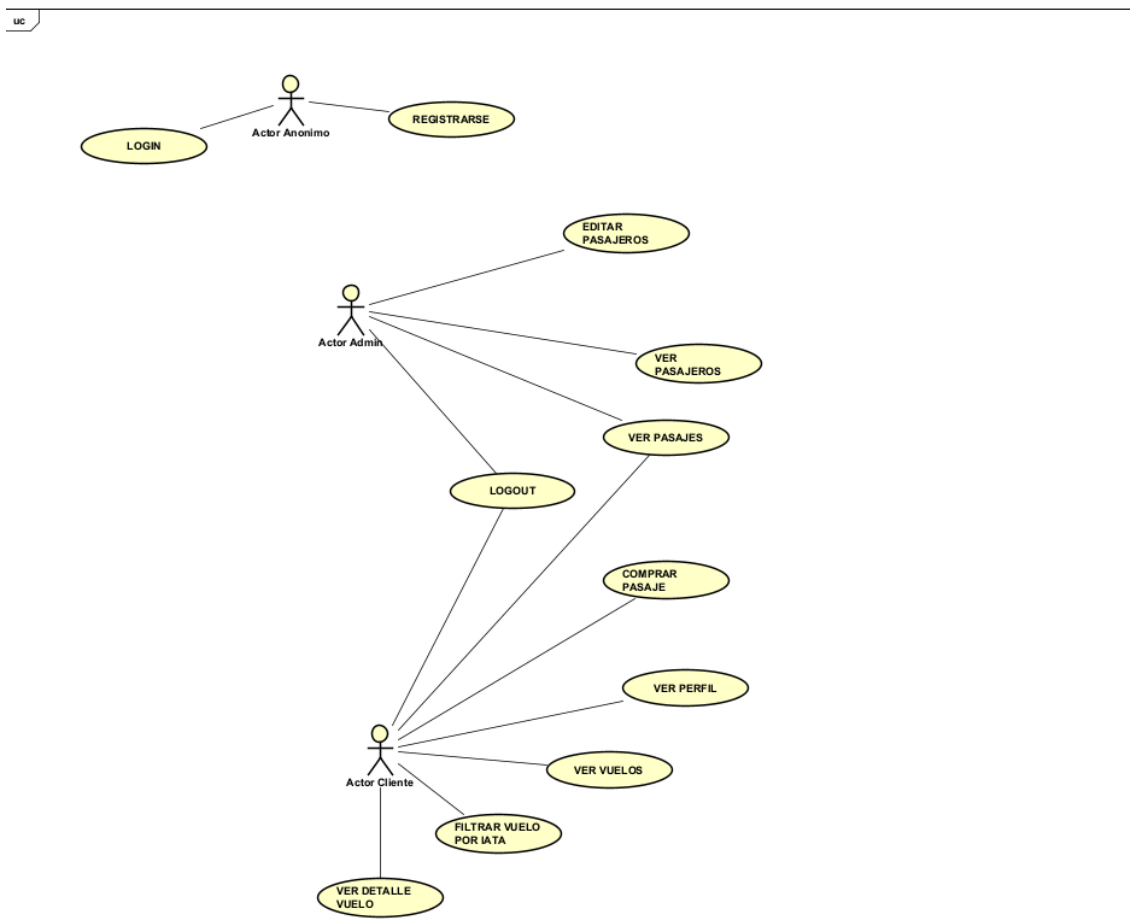
17	Joaquín Sánchez	17	2026-05-21	LIGHT
18	Valentina Rodríguez	18	2026-05-22	BODEGA
19	Lucía González	19	2026-05-23	CABINA
20	Carlos Pérez	20	2026-05-24	LIGHT
21	Ana Torres	21	2026-05-25	BODEGA
22	Mateo Fernández	22	2026-05-26	CABINA
23	Sofía Ramírez	23	2026-05-27	LIGHT
24	Tomás López	24	2026-05-28	BODEGA
25	Camila García	25	2026-05-29	CABINA

6.1.8 Pasajes:

	Pasajero	Vuelo	Fecha	Tipo de Equipaje
1	Lucía González	1	2026-05-05	CABINA
2	Carlos Pérez	2	2026-05-06	LIGHT
3	Ana Torres	3	2026-05-07	BODEGA
4	Mateo Fernández	4	2026-05-08	CABINA
5	Sofía Ramírez	5	2026-05-09	LIGHT
6	Tomás López	6	2026-05-10	BODEGA
7	Camila García	7	2026-05-11	CABINA
8	Joaquín Sánchez	8	2026-05-12	LIGHT
9	Valentina Rodríguez	9	2026-05-13	BODEGA
10	Lucía González	10	2026-05-14	CABINA
11	Carlos Pérez	11	2026-05-15	LIGHT
12	Ana Torres	12	2026-05-16	BODEGA
13	Mateo Fernández	13	2026-05-17	CABINA
14	Sofía Ramírez	14	2026-05-18	LIGHT
15	Tomás López	15	2026-05-19	BODEGA
16	Camila García	16	2026-05-20	CABINA
17	Joaquín Sánchez	17	2026-05-21	LIGHT
18	Valentina Rodríguez	18	2026-05-22	BODEGA

19	Lucía González	19	2026-05-23	CABINA
20	Carlos Pérez	20	2026-05-24	LIGHT
21	Ana Torres	21	2026-05-25	BODEGA
22	Mateo Fernández	22	2026-05-26	CABINA
23	Sofía Ramírez	23	2026-05-27	LIGHT
24	Tomás López	24	2026-05-28	BODEGA
25	Camila García	25	2026-05-29	CABINA

7. Diagrama Casos de Uso:



8. Prompts consultas CHAT GPT:

1- “Dada la precarga que use para la primer parte del obligatorio, necesito me actualices las fechas para que sean posteriores a las que figuran allí (fecha mayo, entre del obligatorio), para que coincidan con la frecuencia de los vuelos. (especialmente las fechas de los pasajes, ya que sino me va lanzar las excepciones de mi lógica).”

2- “Necesito que, teniendo en cuenta la precarga final que te acabo de adjuntar, me realices una tabla de cada una de ellas, clasificadas por objeto, y a su vez, individualizando en cada una de ellas, los datos individualizantes que poseen”.

9. Tabla de testing

ID	Escenario de test	Pasos	Datos utilizados	Resultado esperado	Resultado obtenido	Correcto / Incorrecto
1.1-O	Registro exitoso de usuario ocasional	Completar todos los campos del formulario respetando validaciones. Click en 'Registrarse'.	Nacionalidad: Uruguaya Documento: 12345678 Nombre: Ana Contraseña: Ana1234@ Email: ana@mail.com	Usuario registrado exitosamente. Redirige al home.	Usuario registrado exitosamente. Redirige al home.	Correcto
1.2-O	Registro con campos vacíos	Dejar uno o más campos vacíos y hacer click en	Nombre vacío, nacionalidad vacía, documento vacío, contraseña o mail vacíos	Mensaje de error específico de cada campo	Mensaje de error específico de cada campo	Correcto

		'Registrarse'				
1.2.2-O	Registro con mail repetido	Completar todos los campos del formulario con datos validos pero con mail previamente registrado	Nacionalidad: Uruguay Documento: 11111118 Nombre: Ana Lia Contraseña: Aaana1234@ Email: ana@mail.com	Mensaje de error: No puede registrarse ya que el email ingresado se encuentra en uso.	Mensaje de error: No puede registrarse ya que el email ingresado se encuentra en uso.	Correcto
1.3-O	Registro con contraseña inválida	Ingresar contraseña sin símbolo, número o mayúscula y click en 'Registrarse'.	Contraseña: anaana	Mensaje de error: 'La contraseña debe contener mayúscula, minúscula, número y símbolo'.	Mensaje de error mostrado.	Correcto
2.1-L	Login exitoso	Ingresar email y contraseña válidos y hacer click en 'Login'.	Email: ana@mail.com , Contraseña: Ana1234@	Redirección a vista de pasajero.	Redirección a vista de pasajero.	Correcto
2.2-L	Login fallido	Ingresar email o contraseña incorrectos.	Email: aaana@mail.com, Contraseña: Ana12345@	Mensaje de error: 'Nombre de usuario o contraseña Incorrecta.'	Mensaje de error mostrado.	Correcto

3.1-P	Ver pasajes con lista vacía	Login como pasajero. Click en 'Ver pasajes'.	Pasajero sin pasajes.	Mensaje: 'No hay pasajes para mostrar'.	Mensaje mostrado.	Correcto
3.2-P	Ver perfil del pasajero	Login como pasajero. Click en 'Mi perfil'.	Pasajero registrado.	Visualización de datos personales.	Datos mostrados correctamente.	Correcto
4.1-A	Ver lista de pasajeros y modificar puntos	Login como administrador. Click en 'Ver pasajeros'. Editar puntos.	Cantidad de puntos ingresados en el campo de texto.	Puntos actualizados en la base de datos.	Puntos actualizados.	Correcto

10.Código fuente comentado de toda la aplicación:

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.InteropServices;

using System.Security.Cryptography.X509Certificates;

using System.Text;

using System.Threading.Tasks;

```
using Dominio.Comparadores;
```

```
namespace Dominio
```

```
{
```

```
    public class Sistema
```

```
    {
```

```
        private static Sistema s_instancia;
```

```
        private List<Usuario> _usuarios;
```

```
        private List<Vuelo> _vuelos;
```

```
        private List<Ruta> _rutas;
```

```
        private List<Aeropuerto> _aeropuertos;
```

```
        private List<Pasaje> _pasajes;
```

```
        private List<Avion> _aviones;
```

```
        //GET PARA ACCEDER A LISTAS DESDE EL MVC
```

```
        public List<Usuario> Usuarios { get { return _usuarios; } }
```

```
        public List<Vuelo> Vuelos { get { return _vuelos; } }
```

```
        public List<Ruta> Rutas { get { return _rutas; } }
```

```
        public List<Aeropuerto> Aeropuertos { get { return _aeropuertos; } }
```



```
public List<Pasaje> Pasajes { get { return _pasajes; } }
```

```
public List<Avion> Avion { get { return _aviones; } }
```

```
private Sistema()
```

```
{
```

```
    this._usuarios = new List<Usuario>();
```

```
    this._vuelos = new List<Vuelo>();
```

```
    this._rutas = new List<Ruta>();
```

```
    this._aeropuertos = new List<Aeropuerto>();
```

```
    this._pasajes = new List<Pasaje>();
```

```
    this._aviones = new List<Avion>();
```

```
    this.PrecargarDatos();
```

```
}
```

```
//SINGLETON
```

```

public static Sistema Instancia { get { if (Sistema.s_instancia == null)

    {

        Sistema.s_instancia = new Sistema();

    }

    return s_instancia;

}

}

```

//METODOS PARA MANEJAR LISTAS FILTRADAS:

```

public List<Pasajero> ListarPasajeros()

{

    List<Pasajero> lista = new List<Pasajero>();

    foreach (Usuario unUsuario in _usuarios)

    {

        if (unUsuario is Pasajero unPasajero)

        {

            lista.Add(unPasajero);

        }

    }

}

```

```

        if (lista.Count == 0)
        {
            throw new Exception("No hay pasajeros en el sistema");
        }

        return lista;
    }

```

```

        public List<Vuelo> ListarVuelosPorAeropuerto (string IATAfiltro, string
IATAfiltro2)

```

```

    {
        List<Vuelo> vuelosFiltradosPorAeropuerto = new List<Vuelo>();

        //invierte el filtro vacio y el ingresado para manejar menos condiciones luego

        if (string.IsNullOrEmpty(IATAfiltro) &&
!string.IsNullOrEmpty(IATAfiltro2))
        {
            IATAfiltro = IATAfiltro2;

            IATAfiltro2 = "";

```

```

    }

    // sin filtros, se devuelve todos los vuelos

    if (string.IsNullOrEmpty(IATAfiltro) &&
string.IsNullOrEmpty(IATAfiltro2))

    {

        return _vuelos;

    }

    foreach (Vuelo unVuelo in _vuelos)

    {

        string iataSalida = unVuelo.Ruta.ObtenerIATAAeropuertoDeSalida();

        string iataLlegada = unVuelo.Ruta.ObtenerIATAAeropuertoDeLlegada();

        // Si se ingresó solo un filtro devuelve todos los que contengan ese aeropuerto,
        tanto en salida como en llegada

        if (!string.IsNullOrEmpty(IATAfiltro) &&
string.IsNullOrEmpty(IATAfiltro2))

        {

            if (iataSalida == IATAfiltro || iataLlegada == IATAfiltro)

            {

                vuelosFiltradosPorAeropuerto.Add(unVuelo);

```

```

    }

}

// Si se ingresaron dos filtros devuelve los que tengan esa ruta exacta, no
devuelve inversos

        else if (!string.IsNullOrEmpty(IATAfiltro) &&
!string.IsNullOrEmpty(IATAfiltro2))

    {

        if (iataSalida == IATAfiltro && iataLlegada == IATAfiltro2)

        {

            vuelosFiltradosPorAeropuerto.Add(unVuelo);

        }

    }

}

return vuelosFiltradosPorAeropuerto;

}

```

```

public Usuario ObtenerUsuarioPorMail(string mail)
{
    foreach (Usuario unUsuario in this.Usuarios)
    {
        if (unUsuario.Email == mail)
        {
            return unUsuario;
        }
    }
    return null;
}

```

```

public Vuelo ObtenerVueloPorNumVuelo(int numVuelo)
{
    foreach (Vuelo unVuelo in this._vuelos)
    {
        if (unVuelo.NumVuelo == numVuelo)
        {

```

```

        return unVuelo;

    }

}

throw new Exception($"No existe vuelo con el número: {numVuelo}");

}

```

```

public List<Pasaje> ObtenerListaPasajeDeUsuario(Pasajero logueado)
{
    List<Pasaje> listaFiltrada = new List<Pasaje>();

    foreach (Pasaje unPasaje in this._pasajes)
    {
        if (unPasaje.Pasajero.Equals(logueado))
        {
            listaFiltrada.Add(unPasaje);
        }
    }

    return listaFiltrada;
}

```

//Crea el usuario que va a iniciar sesion segun los parametros ingresados y lo busca en la lista de usuarios, si existe lo devuelve, si no tira excepción.

```
public Usuario Login(string email, string password)

{

    foreach (Usuario usuario in this._usuarios) {

        if (usuario.Email== email)

        {

            if(usuario.Password == password)

            {

                return usuario;

            } else

            {

                throw new Exception("Contraseña incorrecta");

            }

        }

    }

}
```



```

    }

    }

    throw new Exception("Nombre de usuario o contraseña Incorrecta.");

}

```

```

//Para editar

```

```

public void ActualizarPuntosPremium(int puntos, string email)
{

    foreach (Usuario usuario in this._usuarios)
    {
        if (usuario is Premium premium && premium.Email == email)
        {
            premium.ValidarPuntos(puntos);

            premium.Puntos = puntos;

            return;
        }
    }
}

```

```

    }

    throw new Exception ("No existe premium con el email:" + email);

}

public void ActualizarElegibilidadOcasional( string email, bool nuevoEstado)

{

    foreach (Usuario usuario in this._usuarios)

    {

        if (usuario is Ocasional ocasional && ocasional.Email == email)

        {

            ocasional.elegible = nuevoEstado;

            return;

        }

    }

    throw new Exception("No existe ocasional con el email:" + email);

}

```

//ver pasajes ordenados por FECHA desc(ya hacemos uso del COMPARE TO en pasajes)

```
public void OrdenarPasajes()
```

```
{
```

```
    this._pasajes.Sort();
```

```
}
```

//Ver pasajes PARA CLIENTE, ordenados por PRECIO (usamos la clase COMPARADORA CompararPasajePorPrecio creada).

```
public void OrdenarPasajesPorPrecio()
```

```
{
```

```
    this.Pasajes.Sort(new CompararPasajePorPrecio());
```

```
}
```

//METODOS AUXILIARES de validación:

```
public void ValidarExisteUsuario(Usuario nuevo)
```

```

{

    if (_usuarios.Contains(nuevo))

    {

        throw new Exception("No puede registrarse ya que el email ingresado se
encuentra en uso.");

    }

}

```

```

public void ValidarExisteAvion(Avion nuevo)

{

    if (_aviones.Contains(nuevo))

    {

        throw new Exception("El avion no se pudo registrar porque ya fue registrado
previamente.");

    }

}

```

```

public void ValidarExisteAeropuerto(Aeropuerto nuevo)

{

    if (_aeropuertos.Contains(nuevo))

    {

```

```
        throw new Exception("El aeropuerto no se pudo registrar porque ya fue  
registrado previamente.");
```

```
    }
```

```
}
```

```
public void ValidarExisteRuta(Ruta nuevo)
```

```
{
```

```
    if (_rutas.Contains(nuevo))
```

```
    {
```

```
        throw new Exception("La ruta no se pudo registrar porque ya fue registrada  
previamente.");
```

```
    }
```

```
}
```

```
public void ValidarExisteVuelo(Vuelo nuevo)
```

```
{
```

```
    if (_vuelos.Contains(nuevo))
```

```
    {
```

```
        throw new Exception("El vuelo no se pudo registrar porque ya fue registrado  
previamente.");
```

```
    }
```

```
}
```

```
public void ValidarExistePasaje(Pasaje nuevo)
```

```

{

    if (_pasajes.Contains(nuevo))

    {

        throw new Exception("El pasaje no se pudo registrar porque ya fue registrado
previamente.");

    }

}

```

```

                                public void ValidarListaDeVuelosFiltrados(List<Vuelo>
vuelosFiltradosPorAeropuerto)

{

    if (vuelosFiltradosPorAeropuerto.Count == 0)

    {

        throw new Exception("No hay vuelos para el codigo IATA ingresados");

    }

}

```

```

public void EsTipoEquipajeValido(TipoEquipaje? tipoEquipaje)

{

    if (tipoEquipaje == null)

    {

```

```
        throw new Exception("Debe seleccionar un tipo de equipaje para comprar el  
pasaje.");
```

```
    }
```

```
}
```

//METODOS AUXILIARES para dar de alta:

//Primero vamos a recibir el nuevo usuario. Se verifica que no haya sido dado de alta previamente, se valida que lleguen correctos los datos segun lo que definimos en la clase y

//por ultimo agregamos el usuario a la lista general de usuarios con el .add

```
public void DarDeAltaUsuario(Ocasional nuevoUsuario)
```

```
{
```

```
    this.ValidarExisteUsuario(nuevoUsuario);
```

```
    nuevoUsuario.Validar();
```

```
    _usuarios.Add(nuevoUsuario);
```

```
}
```

```
public void DarDeAltaUsuario(Administrador nuevoUsuario)
```

```

{

    this.ValidarExisteUsuario(nuevoUsuario);

    nuevoUsuario.Validar();

    _usuarios.Add(nuevoUsuario);

}

public void DarDeAltaUsuario(Premium nuevoUsuario)

{

    this.ValidarExisteUsuario(nuevoUsuario);

    nuevoUsuario.Validar();

    _usuarios.Add(nuevoUsuario);

}

public void AgregarAvion(Avion unAvion)

{

    this.ValidarExisteAvion(unAvion);

    unAvion.Validar();

    _aviones.Add(unAvion);

}

public void AgregarAeropuerto(Aeropuerto unAeropuerto)

```



```
{  
  
    this.ValidarExisteAeropuerto(unAeropuerto);  
  
    unAeropuerto.Validar();  
  
    _aeropuertos.Add(unAeropuerto);  
  
}
```

```
public void AgregarRuta(Ruta unRuta)
```

```
{  
  
    this.ValidarExisteRuta(unRuta);  
  
    unRuta.Validar();  
  
    _rutas.Add(unRuta);  
  
}
```

```
public void AgregarVuelo(Vuelo unVuelo)
```

```
{  
  
    this.ValidarExisteVuelo(unVuelo);  
  
    unVuelo.Validar();  
  
    _vuelos.Add(unVuelo);  
  
}
```

```
public void AgregarPasaje(Pasaje unPasaje)
```

```
{  
  
    this.ValidarExistePasaje(unPasaje);  
  
    unPasaje.Validar();  
  
}
```

```

        _pasajes.Add(unPasaje);
    }

public void PrecargarDatos()
{

    // ----- ADMINISTRADORES -----

        DarDeAltaUsuario(new Administrador("admin1", "Admin123@",
"admin1@empresa.com"));

        DarDeAltaUsuario(new Administrador("admin2", "Admin456@",
"admin2@empresa.com"));

    // ----- PASAJEROS PREMIUM -----

        DarDeAltaUsuario(new Premium("Uruguaya", "12345678", "Lucía González",
"Password1.", "lucia.gonzalez@mail.com"));

        DarDeAltaUsuario(new Premium("Argentina", "23456789", "Carlos Pérez",
"Password2@", "carlos.perez@mail.com"));

        DarDeAltaUsuario(new Premium("Chilena", "34567890", "Ana Torres",
"Password3@", "ana.torres@mail.com"));

        DarDeAltaUsuario(new Premium("Brasilera", "45678901", "Mateo Fernández",
"Password4@", "mateo.fernandez@mail.com"));

```

```
DarDeAltaUsuario(new Premium("Colombiana", "56789012", "Sofía Ramírez",  
"Password5@", "sofia.ramirez@mail.com"));
```

```
// ----- PASAJEROS OCASIONALES -----
```

```
DarDeAltaUsuario(new Ocasional("Uruguaya", "22334455", "Tomás López",  
"Password6@", "tomas.lopez@mail.com"));
```

```
DarDeAltaUsuario(new Ocasional("Paraguaya", "33445566", "Camila García",  
"Password7@", "camila.garcia@mail.com"));
```

```
DarDeAltaUsuario(new Ocasional("Mexicana", "44556677", "Joaquín  
Sánchez", "Password8@", "joaquin.sanchez@mail.com"));
```

```
DarDeAltaUsuario(new Ocasional("Venezolana", "55667788", "Valentina  
Rodríguez", "Password9@", "valentina.rodriguez@mail.com"));
```

```
DarDeAltaUsuario(new Ocasional("Boliviana", "66778899", "Martín Martínez",  
"Password10@", "martin.martinez@mail.com"));
```

```
// ----- AVIONES -----
```

```
AgregarAvion(new Avion("Boeing", "737", 5000, 180, 10.5m, "Narrow-body"));
```

```
AgregarAvion(new Avion("Airbus", "A320", 4800, 170, 9.8m,  
"Narrow-body"));
```

```
AgregarAvion(new Avion("Embraer", "E190", 4000, 100, 7.2m, "Regional"));
```

```
AgregarAvion(new Avion("Bombardier", "CRJ900", 3700, 90, 6.9m,  
"Regional"));
```

```
// ----- AEROPUERTOS -----
```

AgregarAeropuerto(new Aeropuerto("MVD", "Montevideo", 250, 100));

AgregarAeropuerto(new Aeropuerto("EZE", "Buenos Aires", 255, 102));

AgregarAeropuerto(new Aeropuerto("SCL", "Santiago", 260, 104));

AgregarAeropuerto(new Aeropuerto("LIM", "Lima", 265, 106));

AgregarAeropuerto(new Aeropuerto("BOG", "Bogotá", 270, 108));

AgregarAeropuerto(new Aeropuerto("GIG", "Río", 275, 110));

AgregarAeropuerto(new Aeropuerto("GRU", "São Paulo", 280, 112));

AgregarAeropuerto(new Aeropuerto("UIO", "Quito", 285, 114));

AgregarAeropuerto(new Aeropuerto("CCS", "Caracas", 290, 116));

AgregarAeropuerto(new Aeropuerto("PTY", "Panamá", 295, 118));

AgregarAeropuerto(new Aeropuerto("ASU", "Asunción", 300, 120));

AgregarAeropuerto(new Aeropuerto("LPB", "La Paz", 305, 122));

AgregarAeropuerto(new Aeropuerto("MEX", "Ciudad de México", 310, 124));

AgregarAeropuerto(new Aeropuerto("MIA", "Miami", 315, 126));

AgregarAeropuerto(new Aeropuerto("MAD", "Madrid", 320, 128));

AgregarAeropuerto(new Aeropuerto("FCO", "Roma", 325, 130));

AgregarAeropuerto(new Aeropuerto("CDG", "París", 330, 132));

AgregarAeropuerto(new Aeropuerto("LIS", "Lisboa", 335, 134));

AgregarAeropuerto(new Aeropuerto("LHR", "Londres", 340, 136));

AgregarAeropuerto(new Aeropuerto("YYZ", "Toronto", 345, 138));

```
// ----- RUTAS -----
```

```
AgregarRuta(new Ruta(_aeropuertos[0], _aeropuertos[1], 2000));  
AgregarRuta(new Ruta(_aeropuertos[2], _aeropuertos[3], 1500));  
AgregarRuta(new Ruta(_aeropuertos[4], _aeropuertos[5], 1700));  
AgregarRuta(new Ruta(_aeropuertos[6], _aeropuertos[7], 1400));  
AgregarRuta(new Ruta(_aeropuertos[8], _aeropuertos[9], 1800));  
AgregarRuta(new Ruta(_aeropuertos[10], _aeropuertos[11], 1900));  
AgregarRuta(new Ruta(_aeropuertos[12], _aeropuertos[13], 2200));  
AgregarRuta(new Ruta(_aeropuertos[14], _aeropuertos[15], 2400));  
AgregarRuta(new Ruta(_aeropuertos[16], _aeropuertos[17], 2300));  
AgregarRuta(new Ruta(_aeropuertos[18], _aeropuertos[19], 2100));  
AgregarRuta(new Ruta(_aeropuertos[0], _aeropuertos[2], 1300));  
AgregarRuta(new Ruta(_aeropuertos[3], _aeropuertos[5], 1600));  
AgregarRuta(new Ruta(_aeropuertos[6], _aeropuertos[9], 2500));  
AgregarRuta(new Ruta(_aeropuertos[10], _aeropuertos[13], 2700));  
AgregarRuta(new Ruta(_aeropuertos[12], _aeropuertos[15], 2800));  
AgregarRuta(new Ruta(_aeropuertos[1], _aeropuertos[4], 2900));  
AgregarRuta(new Ruta(_aeropuertos[7], _aeropuertos[10], 2600));  
AgregarRuta(new Ruta(_aeropuertos[14], _aeropuertos[17], 3100));  
AgregarRuta(new Ruta(_aeropuertos[8], _aeropuertos[11], 3300));
```

```

AgregarRuta(new Ruta(_aeropuertos[5], _aeropuertos[18], 3000));

AgregarRuta(new Ruta(_aeropuertos[2], _aeropuertos[6], 2700));

AgregarRuta(new Ruta(_aeropuertos[3], _aeropuertos[7], 2800));

AgregarRuta(new Ruta(_aeropuertos[1], _aeropuertos[8], 3000));

AgregarRuta(new Ruta(_aeropuertos[4], _aeropuertos[10], 2700));

AgregarRuta(new Ruta(_aeropuertos[13], _aeropuertos[15], 3400));

AgregarRuta(new Ruta(_aeropuertos[16], _aeropuertos[19], 3600));

AgregarRuta(new Ruta(_aeropuertos[11], _aeropuertos[18], 3800));

AgregarRuta(new Ruta(_aeropuertos[9], _aeropuertos[17], 3500));

AgregarRuta(new Ruta(_aeropuertos[12], _aeropuertos[14], 3200));

AgregarRuta(new Ruta(_aeropuertos[0], _aeropuertos[19], 3900));

```

```

AgregarVuelo(new Vuelo(1, _aviones[0], _rutas[0], new List<DayOfWeek> {
DayOfWeek.Tuesday })); // 2026-05-05

```

```

AgregarVuelo(new Vuelo(2, _aviones[1], _rutas[1], new List<DayOfWeek> {
DayOfWeek.Wednesday })); // 2026-05-06

```

```

AgregarVuelo(new Vuelo(3, _aviones[2], _rutas[2], new List<DayOfWeek> {
DayOfWeek.Saturday })); // 2026-05-09

```

```

AgregarVuelo(new Vuelo(4, _aviones[3], _rutas[3], new List<DayOfWeek> {
DayOfWeek.Monday })); // 2026-05-11

```

```
    AgregarVuelo(new Vuelo(5, _aviones[0], _rutas[4], new List<DayOfWeek> {  
DayOfWeek.Tuesday }));    // 2026-05-12
```

```
    AgregarVuelo(new Vuelo(6, _aviones[1], _rutas[5], new List<DayOfWeek> {  
DayOfWeek.Wednesday }));    // 2026-05-13
```

```
    AgregarVuelo(new Vuelo(7, _aviones[2], _rutas[6], new List<DayOfWeek> {  
DayOfWeek.Thursday }));    // 2026-05-14
```

```
    AgregarVuelo(new Vuelo(8, _aviones[3], _rutas[7], new List<DayOfWeek> {  
DayOfWeek.Friday }));    // 2026-05-15
```

```
    AgregarVuelo(new Vuelo(9, _aviones[0], _rutas[8], new List<DayOfWeek> {  
DayOfWeek.Saturday }));    // 2026-05-16
```

```
    AgregarVuelo(new Vuelo(10, _aviones[1], _rutas[9], new List<DayOfWeek> {  
DayOfWeek.Sunday }));    // 2026-05-17
```

```
    AgregarVuelo(new Vuelo(11, _aviones[2], _rutas[10], new List<DayOfWeek> {  
DayOfWeek.Monday }));    // 2026-05-18
```

```
    AgregarVuelo(new Vuelo(12, _aviones[3], _rutas[11], new List<DayOfWeek> {  
DayOfWeek.Tuesday }));    // 2026-05-19
```

```
    AgregarVuelo(new Vuelo(13, _aviones[0], _rutas[12], new List<DayOfWeek> {  
DayOfWeek.Wednesday }));    // 2026-05-20
```

```
    AgregarVuelo(new Vuelo(14, _aviones[1], _rutas[13], new List<DayOfWeek> {  
DayOfWeek.Thursday }));    // 2026-05-21
```

```
    AgregarVuelo(new Vuelo(15, _aviones[2], _rutas[14], new List<DayOfWeek> {  
DayOfWeek.Friday }));    // 2026-05-22
```

```
    AgregarVuelo(new Vuelo(16, _aviones[3], _rutas[15], new List<DayOfWeek> {  
DayOfWeek.Saturday }));    // 2026-05-23
```

```
AgregarVuelo(new Vuelo(17, _aviones[0], _rutas[16], new List<DayOfWeek> {  
DayOfWeek.Sunday })); // 2026-05-24
```

```
AgregarVuelo(new Vuelo(18, _aviones[1], _rutas[17], new List<DayOfWeek> {  
DayOfWeek.Monday })); // 2026-05-25
```

```
AgregarVuelo(new Vuelo(19, _aviones[2], _rutas[18], new List<DayOfWeek> {  
DayOfWeek.Tuesday })); // 2026-05-26
```

```
AgregarVuelo(new Vuelo(20, _aviones[3], _rutas[19], new List<DayOfWeek> {  
DayOfWeek.Wednesday })); // 2026-05-27
```

```
AgregarVuelo(new Vuelo(21, _aviones[0], _rutas[20], new List<DayOfWeek> {  
DayOfWeek.Thursday })); // 2026-05-28
```

```
AgregarVuelo(new Vuelo(22, _aviones[1], _rutas[21], new List<DayOfWeek> {  
DayOfWeek.Friday })); // 2026-05-29
```

```
AgregarVuelo(new Vuelo(23, _aviones[2], _rutas[22], new List<DayOfWeek> {  
DayOfWeek.Saturday })); // 2026-05-30
```

```
AgregarVuelo(new Vuelo(24, _aviones[3], _rutas[23], new List<DayOfWeek> {  
DayOfWeek.Sunday })); // 2026-05-31
```

```
AgregarVuelo(new Vuelo(25, _aviones[0], _rutas[24], new List<DayOfWeek> {  
DayOfWeek.Monday })); // 2026-06-01
```

```
// ----- PASAJES -----
```

```
AgregarPasaje(new Pasaje(_vuelos[0], (Pasajero)_usuarios[2], new  
DateTime(2026, 5, 5), TipoEquipaje.CABINA));
```

```
AgregarPasaje(new Pasaje(_vuelos[1], (Pasajero)_usuarios[3], new  
DateTime(2026, 5, 6), TipoEquipaje.LIGHT));
```


AgregarPasaje(new Pasaje(_vuelos[2], (Pasajero)_usuarios[4], new DateTime(2026, 5, 9), TipoEquipaje.BODEGA));

AgregarPasaje(new Pasaje(_vuelos[3], (Pasajero)_usuarios[5], new DateTime(2026, 5, 11), TipoEquipaje.CABINA));

AgregarPasaje(new Pasaje(_vuelos[4], (Pasajero)_usuarios[6], new DateTime(2026, 5, 12), TipoEquipaje.LIGHT));

AgregarPasaje(new Pasaje(_vuelos[5], (Pasajero)_usuarios[7], new DateTime(2026, 5, 13), TipoEquipaje.BODEGA));

AgregarPasaje(new Pasaje(_vuelos[6], (Pasajero)_usuarios[8], new DateTime(2026, 5, 14), TipoEquipaje.CABINA));

AgregarPasaje(new Pasaje(_vuelos[7], (Pasajero)_usuarios[9], new DateTime(2026, 5, 15), TipoEquipaje.LIGHT));

AgregarPasaje(new Pasaje(_vuelos[8], (Pasajero)_usuarios[10], new DateTime(2026, 5, 16), TipoEquipaje.BODEGA));

AgregarPasaje(new Pasaje(_vuelos[9], (Pasajero)_usuarios[2], new DateTime(2026, 5, 17), TipoEquipaje.CABINA));

AgregarPasaje(new Pasaje(_vuelos[10], (Pasajero)_usuarios[3], new DateTime(2026, 5, 18), TipoEquipaje.LIGHT));

AgregarPasaje(new Pasaje(_vuelos[11], (Pasajero)_usuarios[4], new DateTime(2026, 5, 19), TipoEquipaje.BODEGA));

AgregarPasaje(new Pasaje(_vuelos[12], (Pasajero)_usuarios[5], new DateTime(2026, 5, 20), TipoEquipaje.CABINA));

AgregarPasaje(new Pasaje(_vuelos[13], (Pasajero)_usuarios[6], new DateTime(2026, 5, 21), TipoEquipaje.LIGHT));

AgregarPasaje(new Pasaje(_vuelos[14], (Pasajero)_usuarios[7], new
DateTime(2026, 5, 22), TipoEquipaje.BODEGA));

AgregarPasaje(new Pasaje(_vuelos[15], (Pasajero)_usuarios[8], new
DateTime(2026, 5, 23), TipoEquipaje.CABINA));

AgregarPasaje(new Pasaje(_vuelos[16], (Pasajero)_usuarios[9], new
DateTime(2026, 5, 24), TipoEquipaje.LIGHT));

AgregarPasaje(new Pasaje(_vuelos[17], (Pasajero)_usuarios[10], new
DateTime(2026, 5, 25), TipoEquipaje.BODEGA));

AgregarPasaje(new Pasaje(_vuelos[18], (Pasajero)_usuarios[2], new
DateTime(2026, 5, 26), TipoEquipaje.CABINA));

AgregarPasaje(new Pasaje(_vuelos[19], (Pasajero)_usuarios[3], new
DateTime(2026, 5, 27), TipoEquipaje.LIGHT));

AgregarPasaje(new Pasaje(_vuelos[20], (Pasajero)_usuarios[4], new
DateTime(2026, 5, 28), TipoEquipaje.BODEGA));

AgregarPasaje(new Pasaje(_vuelos[21], (Pasajero)_usuarios[5], new
DateTime(2026, 5, 29), TipoEquipaje.CABINA));

AgregarPasaje(new Pasaje(_vuelos[22], (Pasajero)_usuarios[6], new
DateTime(2026, 5, 30), TipoEquipaje.LIGHT));

AgregarPasaje(new Pasaje(_vuelos[23], (Pasajero)_usuarios[7], new
DateTime(2026, 5, 31), TipoEquipaje.BODEGA));

AgregarPasaje(new Pasaje(_vuelos[24], (Pasajero)_usuarios[8], new
DateTime(2026, 6, 1), TipoEquipaje.CABINA));

}

}

```

    }

using Dominio.Interfaces;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;


namespace Dominio

{

    public class Administrador : Usuario, IValidable

    {

        private string _nickname;


        //public string Nickname { get { return _nickname; } }


        public Administrador(string nickname, string password, string email) :
base(password, email)

        {

            this._nickname = nickname;

```

```

        this.Validar();
    }

    public void Validar()
    {
        this.ValidarNickname();
    }

    public void ValidarNickname()
    {
        if (string.IsNullOrEmpty(this._nickname))
        {
            throw new Exception("El valor no puede estar vacío o solo contener
espacios.");
        }
    }
}

using Dominio.Interfaces;

```

```

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;


namespace Dominio
{

    public class Aeropuerto : IValidable
    {

        private string _IATACode;

        private string _ciudad;

        private decimal _costoOpp;

        private decimal _costoTasas;


        public string IATACode
        {
            get { return _IATACode; }
        }
    }
}

```

```
public decimal CostoOpp { get { return _costoOpp; } }
```

```
public decimal CostoTasas { get { return _costoTasas; } }
```

```
public Aeropuerto(string IATACode, string ciudad, decimal costoOpp, decimal  
costoTasas)
```

```
{
```

```
    this._IATACode = IATACode.ToUpper();//Forzamos que este en mayúscula  
desde un comienzo para no hacer validación.
```

```
    this._ciudad = ciudad;
```

```
    this._costoOpp = costoOpp;
```

```
    this._costoTasas = costoTasas;
```

```
    this.Validar();
```

```
}
```

```
public void Validar()
```

```
{
```

```
    this.ValidarCiudad();
```

```
    this.ValidarCostoOpp();
```

```
    this.ValidarCostoTasas();
```

```

        this.ValidarIATA();
    }

    public void ValidarIATA()
    {
        if (IATACode.Length != 3)
        {
            throw new Exception("El código IATA debe tener exactamente 3
caracteres.");
        }

        foreach (char character in IATACode)
        {
            if (!char.IsLetter(character))
            {
                throw new Exception("Solo ingresar letras en el código IATA, no números,
espacios en blanco ni simbolos.");
            }
        }
    }
}

```

```

public void ValidarCiudad()

{

    if (string.IsNullOrEmpty(this._ciudad))

    {

        throw new Exception("El nombre de la ciudad no puede estar vacía o solo
contener espacios.");

    }

}

public void ValidarCostoOpp()

{

    if (this._costoOpp <= 0)

    {

        throw new Exception("El costo de operación no puede tener un valor negativo
o igual a 0.");

    }

}

```



```

public void ValidarCostoTasas()

{

    if (this._costoTasas <= 0)

    {

        throw new Exception("Las tasas no pueden ser 0 o un número negativo");

    }

}

```

```

public override string ToString()

{

    return $" {_IATACode}";

}

```

```

public override bool Equals(object? obj)

{

    if (obj == null || !(obj is Aeropuerto)) return false;

    Aeropuerto otro = (Aeropuerto)obj;

    return this.IATACode.Equals(otro.IATACode);

}

```

```

    }

}

using Dominio.Interfaces;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Dominio
{
    public class Avion : IValidable
    {
        private string _fabricante;

        private string _modelo;

        private int _alcance;

        private int _cantAsientos;

        private decimal _costoXKm;

        private string _tipoAeronave;
    }
}

```

```
public int Alcance { get { return this._alcance; } } //Para validar que el avion  
pueda hacer el vuelo.
```

```
public decimal CostoXKm { get { return this._costoXKm;} }
```

```
public int CantAsientos { get { return this._cantAsientos; } }
```

```
public string Modelo { get { return this._modelo; } }
```

```
public Avion(string fabricante, string modelo, int alcance, int cantAsientos,  
decimal costoXKm, string tipoAeronave)
```

```
{  
  
    this._fabricante = fabricante;  
  
    this._modelo = modelo;  
  
    this._alcance = alcance;  
  
    this._cantAsientos = cantAsientos;  
  
    this._costoXKm = costoXKm;  
  
    this._tipoAeronave = tipoAeronave;  
  
    this.Validar();  
  
}
```

```

public void Validar()
{
    this.ValidarFabricante();

    this.ValidarModelo();

    this.ValidarAlcance();

    this.ValidarCantAsientos();

    this.ValidarCostXKm();
}

```

```

public void ValidarFabricante()
{
    if (string.IsNullOrEmpty(this._fabricante))
    {
        throw new Exception("El valor no puede estar vacío o solo contener
espacios.");
    }
}

```

```

public void ValidarModelo()
{

```

```

        if (string.IsNullOrEmpty(this._modelo))
        {
            throw new Exception("El valor no puede estar vacío o solo contener
espacios.");
        }

    }

    public void ValidarAlcance()
    {
        if(this._alcance <= 0)
        {
            throw new Exception("El avión no puede tener un alcance negativo o igual a
0.");
        }

    }

    public void ValidarCantAsientos()
    {
        if (this._cantAsientos <= 0)
        {

```

```
        throw new Exception("La cantidad de asientos no puede ser 0 o un número negativo.");
```

```
    }
```

```
}
```

```
public void ValidarCostoXKm()
```

```
{
```

```
    if (this._costoXKm <= 0)
```

```
    {
```

```
        throw new Exception("El costo por kilómetro no puede ser 0 o negativo.");
```

```
    }
```

```
}
```

```
public override bool Equals(object? obj)
```

```
{
```

```
    if (obj == null || !(obj is Avion)) return false;
```

```
    Avion otro = (Avion)obj;
```

```
        return this._fabricante.Equals(otro._fabricante) &&  
this._modelo.Equals(otro._modelo);
```

```
}
```

```

        public override string ToString()
        {
            return $"Modelo de Avion : {this._modelo} | Fabricante : {this._fabricante}";
        }
    }
}

using Dominio;

using Dominio.Interfaces;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Security.Cryptography;

using System.Text;

using System.Threading.Tasks;

namespace Dominio
{
    public class Ocasional : Pasajero, IValidable
    {

```

```

private bool _elegible;

public bool elegible { get { return _elegible; } set { _elegible = value; } }

    public Ocasional(string nacionalidad, string docIdentidad, string nombre, string
password, string email) : base(nacionalidad, docIdentidad, nombre, password, email)

    {

        this._elegible = CalcularElegibilidad();

    }

public Ocasional() { }

private bool CalcularElegibilidad()

{

    //si el random cae en el numero limite o antes es elegible, si es mayor no es
elegible

    int chance = 50;

    Random numeroRandom = new Random();

    int numero = numeroRandom.Next(0, 100);

    return numero <= chance;

}

```



```
//METODO POLIMORFICO QUE SOBREScribe METODO DE CLASE  
BASE (PASAJERO)
```

```
public override decimal CalcularPrecioEquipaje(TipoEquipaje tipoEquipaje)  
{  
    switch (tipoEquipaje)  
    {  
        case TipoEquipaje.CABINA:  
  
            return 1.10m; //porcentaje de equipaje CABINA;  
  
        case TipoEquipaje.BODEGA:  
  
            return 1.20m; //porcentaje de equipaje CABINA;  
  
        default:  
  
            return 1m;  
  
    }  
  
}
```

```

        public override string ToString()
        {
            return base.ToString() + $" Elegible: {(_elegible ? "Sí" : "No")}";
        }
    }
}

using Dominio.Interfaces;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Dominio
{
    public class Pasaje : IValidable , IComparable<Pasaje>
    {
        private static int s_ultimoId = 0;

        private int _id;

```

```
private Vuelo _vuelo;
```

```
private Pasajero _pasajero;
```

```
private DateTime _fecha;
```

```
private TipoEquipaje _tipoEquipaje;
```

```
private decimal _precio;
```

```
public static decimal PORCENTAJE_GANANCIA = 0.25m;
```

```
//GET para acceder desde MVC
```

```
public int Id { get { return _id; } }
```

```
public Vuelo Vuelo { get { return _vuelo; } }
```

```
public Pasajero Pasajero { get { return _pasajero; } }
```

```
public DateTime Fecha { get { return _fecha; } }
```

```
public TipoEquipaje TipoEquipaje { get { return _tipoEquipaje; } }
```

```
public decimal Precio { get { return _precio; } }
```

```
public Pasaje(Vuelo vuelo, Pasajero pasajero, DateTime fecha, TipoEquipaje  
tipoEquipaje)
```

```

{

    //aumenta el valor de id cada vez que se crea la instancia

    this._id = s_ultimoId++;

    this._vuelo = vuelo;

    this._pasajero = pasajero;

    this._fecha = fecha;

    this._tipoEquipaje = tipoEquipaje;

    this._precio = this.CalcularPrecioDelPasaje();

    this.Validar();

}

```

```

public Pasaje()

```

```

{

```

```

}

```

//el get para poder ver el valor del ultimo id asignado a la instancia anterior (NO NECESARIO, PERO PODEMOS USARLO PARA TESTING).

```

public static int UltimoId

```

```

{

    get { return s_ultimoId; }

}


public void Validar()

{

    this.ValidarFechaCorrespondeFrecuencia();

    this.ValidarFecha();

}


public void ValidarFecha()

{

    if(this._fecha < DateTime.Today)

    {

        throw new Exception("La fecha del pasaje no puede ser anterior a la fecha
actual.");

    }

}

```

```

public void ValidarFechaCorrespondeFrecuencia()
{
    if (!_vuelo.Frecuencia.Contains(_fecha.DayOfWeek))
    {
        throw new Exception("La fecha del pasaje no coincide con la frecuencia del
vuelo.");
    }
}

```

//Asigna en el atributo de precio, el valor obtenido mediante metodo
CalcularPrecioPasaje realizado en Pasajero para guardarlo.

```

public decimal CalcularPrecioDelPasaje()
{

    decimal costoPorAsiento = _vuelo.CalcularCostoPorAsiento();

    decimal precioBase = costoPorAsiento * PORCENTAJE_GANANCIA;

    decimal cargoPorEquipaje =
_pasajero.CalcularPrecioEquipaje(this._tipoEquipaje);

    decimal tasasPortuarias = _vuelo.ObtenerCostoTasasPortuarias();

```

```

        return costoPorAsiento * (PORCENTAJE_GANANCIA + cargoPorEquipaje) +
tasasPortuarias;

    }

```

```

public override bool Equals(object? obj)

{

    if (obj == null || !(obj is Pasaje)) return false;

    Pasaje otro = (Pasaje)obj;

    return this._fecha.Equals(otro._fecha) && this._pasajero.Equals(otro._pasajero);

}

```

```

public override string ToString()

{

    return $"Datos del Pasaje: ID: {_id} | Pasajero: {_pasajero.Nombre} | Fecha:
{_fecha:dd-MM-yyyy} | Precio: ${_precio.ToString("F2")} | Vuelo número:
{_vuelo.NumVuelo}";

}

```

```
//ESTO ORDENA AL ADMINISTRADOR LOS PASAJES EMITIDOS POR  
FECHA
```

```
public int CompareTo(Pasaje? other)  
  
{  
  
    return this._fecha.CompareTo(other._fecha);  
  
}  
  
}
```

```
using Dominio.Interfaces;
```

```
using System;
```

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

```
using System.Linq;
```

```
using System.Text;
```

```
using System.Threading.Tasks;
```

```
namespace Dominio
```

```
{  
  
    public abstract class Pasajero : Usuario , IValidable, IComparable <Pasajero>  
  
    {  
  
        private string _nacionalidad;  
  
        private string _docIdentidad;
```



```
private string _nombre;
```

```
//public, con get y set para usar el Model Binding en Registrar el Ocasional
```

```
public string Nacionalidad { get { return _nacionalidad; } set { this._nacionalidad  
= value; } }
```

```
public string DocIdentidad { get { return _docIdentidad; } set { this._docIdentidad  
= value; } }
```

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

```
public Pasajero(string nacionalidad, string docIdentidad, string nombre, string  
password, string email) : base(password, email)
```

```
{
```

```
    this._nacionalidad = nacionalidad;
```

```
    this._docIdentidad = docIdentidad;
```

```
    this._nombre = nombre;
```

```
    this.Validar();
```

```
}
```

```
public Pasajero() { }
```

```
public void Validar()
```

```
{
```

```
    base.Validar();
```

```
    this.ValidarNombre();
```

```
    this.ValidarNacionalidad();
```

```
    this.ValidarDocumentoDeIdentidad();
```

```
}
```

```
public void ValidarNombre()
```

```
{
```

```
    if (string.IsNullOrEmpty(this._nombre))
```

```
    {
```

```
        throw new Exception("El nombre no puede estar vacío ni contener espacios  
en blanco.");
```

```
    }
```

```
}
```

```
public void ValidarNacionalidad()
```

```
{
```

```

        if (string.IsNullOrEmpty(this._nacionalidad))
        {

            throw new Exception("La nacionalidad no puede estar vacía ni contener
espacios en blanco.");

        }
    }

```

```

public void ValidarDocumentoDeIdentidad()
{
    if (string.IsNullOrEmpty(this._docIdentidad))
    {

        throw new Exception("El documento de identidad no puede estar vacío ni
contener espacios en blanco.");

    }
}

```

// Metodo ABSTRACT POLIMORFICO para calculo pasaje. Lo hacemos acá ya que esta es la clase base, y se va aplicar override en las subclases.

// Le pasamos por parámetro tipo de equipaje que es lo que necesitamos para aplicar las distintas casuísticas de la letra.

```

public abstract decimal CalcularPrecioEquipaje(TipoEquipaje tipoEquipaje);


public override string ToString()
{
    return $"Nombre: {this._nombre} | Email: {this._email} | Nacionalidad:
{this._nacionalidad} | Cedula Identidad: {this._docIdentidad}";
}


public int CompareTo(Pasajero? other)
{
    return this._docIdentidad.CompareTo(other._docIdentidad);
}
}

using Dominio.Interfaces;

using System;

using System.Collections.Generic;

using System.Linq;

```

```

using System.Text;

using System.Threading.Tasks;

namespace Dominio
{
    public class Premium : Pasajero, IValidable
    {
        private int _puntos;

        public int Puntos { get { return _puntos; } set { _puntos = value; } }

        public Premium(string nacionalidad, string docIdentidad, string nombre, string
password, string email) : base(nacionalidad, docIdentidad, nombre, password, email)
        {
            this._puntos = 0;
        }

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

```

```

        this.ValidarPuntos(this._puntos);

    }

    public override string ToString()

    {

        return base.ToString() + $" Puntos: {_puntos}";

    }

```

//METODO POLIMORFICO QUE SOBREScribe METODO DE CLASE
BASE (PASAJERO)

```

    public override decimal CalcularPrecioEquipaje(TipoEquipaje tipoEquipaje)

    {

        switch (tipoEquipaje)

        {

            case TipoEquipaje.BODEGA:

                return 1.05m; //porcentaje de equipaje BODEGA;

            default:

                return 1m;

```

```
    }  
}  
  
public void ValidarPuntos(int puntos)  
{  
    if (puntos < 0)  
    {  
        throw new Exception("Los puntos asignados no pueden ser negativos.");  
    }  
  
}  
  
}  
  
}
```

```

using Dominio.Interfaces;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Dominio
{
    public class Ruta : IValidable
    {

        private static int s_ultimoId = -1;

        private int _id;

        private Aeropuerto _aeropuertoLlegada;

        private Aeropuerto _aeropuertoSalida;

        private decimal _distancia;
    }
}

```



```
    public decimal Distancia { get { return this._distancia; } } //Para validar que el
avion pueda hacer el vuelo.
```

```
public Aeropuerto AeropuertoLlegada { get { return this._aeropuertoLlegada; } }
```

```
public Aeropuerto AeropuertoSalida { get { return this._aeropuertoSalida; } }
```

```
public Ruta(Aeropuerto aeropuertoLlegada, Aeropuerto aeropuertoSalida, decimal
distancia)
```

```
{
```

```
    //aumenta el valor de id cada vez que se crea la instancia
```

```
    this._id = s_ultimoId++;
```

```
    this._aeropuertoLlegada = aeropuertoLlegada;
```

```
    this._aeropuertoSalida = aeropuertoSalida;
```

```
    this._distancia = distancia;
```

```
    this.Validar();
```

```
}
```

```
//el get para poder ver el valor del ultimo id asignado a la instancia anterior
```

```
public static int UltimoIdRuta
```

```
{
```

```

        get { return s_ultimoId; }

    }

    public void Validar()
    {
        this.ValidarDistancia();

        this.ValidarQueAeropuertoNoSeRepita();
    }

    public void ValidarDistancia()
    {
        if (this._distancia <= 0)
        {
            throw new Exception("La distancia de la ruta debe ser mayor que 0.\n");
        }
    }

    public void ValidarQueAeropuertoNoSeRepita()

```

```

{

    if (_aeropuertoLlegada.Equals(_aeropuertoSalida))

    {

        throw new Exception("El aeropuerto de salida no puede ser el mismo que el
de llegada\n");

    }

}

public string ObtenerIATAAeropuertoDeSalida()

{

    return this._aeropuertoSalida.IATACode;

}

public string ObtenerIATAAeropuertoDeLlegada()

{

    return this._aeropuertoLlegada.IATACode;

}

public override bool Equals(object? obj)

{

    if (obj == null || !(obj is Ruta)) return false;

    Ruta otro = (Ruta)obj;

```

```

        return this._aeropuertoSalida.Equals(otro._aeropuertoSalida) &&
this._aeropuertoLlegada.Equals(otro._aeropuertoLlegada);

    }

```

```

    public override string ToString()

    {

        return $"{ObtenerIATAAeropuertoDeSalida()} -
{ObtenerIATAAeropuertoDeLlegada()}";

    }

}

```

```

using System;

```

```

using System.Collections.Generic;

```

```

using System.Linq;

```

```

using System.Text;

```

```

using System.Threading.Tasks;

```

```

namespace Dominio

```

```

{

    public enum TipoEquipaje

    {

```

```

        LIGHT = 0,

        CABINA = 1,

        BODEGA = 2

    }

}

using Dominio.Interfaces;

namespace Dominio

{

    public class Usuario : IValidable

    {

        private string _password;

        protected string _email;

        //public, con get y set para usar el Model Binding en Registrar el Ocasional

        public string Password { get { return _password; } set { this._password = value; }

    }

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

```

```
public Usuario(string password, string email)
```

```
{
```

```
    this._password = password;
```

```
    this._email = email;
```

```
    this.Validar();
```

```
}
```

```
public Usuario() { }
```

```
public void Validar()
```

```
{
```

```
    this.ValidarPassword();
```

```
    this.ValidarEmail();
```

```
}
```

```
public void ValidarPassword()
```

```
{
```

```
    if (string.IsNullOrEmpty(this._password))
```

```

    {
        throw new Exception("La contraseña no puede estar vacía o solo contener
espacios.");
    }

    if(_password.Length < 8)
    {
        throw new Exception("La contraseña no puede ser menor a 8 caracteres.");
    }

    int i = 0;

    int cantidadMinima = 1;

    int contadorMayusculas = 0;

    int contadorNumeros = 0;

    int contadorMinusculas = 0;

    int contadorSimbolo = 0;

    foreach (char caracter in _password)
    {
        if (char.IsDigit(caracter))

```

```

    {

        contadorNumeros++;

    }

    else if (char.IsLower(caracter))

    {

        contadorMinusculas++;

    }

    else if (char.IsUpper(caracter))

    {

        contadorMayusculas++;

    }

    else

    {

        contadorSimbolo++;

    }

}

```

```

        if (contadorMayusculas < cantidadMinima || contadorMinusculas <
cantidadMinima || contadorSimbolo < cantidadMinima || contadorNumeros <
cantidadMinima)

    {

```



```
        throw new Exception("La contraseña debe contener al menos una mayúscula,  
una minúscula, un número y un símbolo.");
```

```
    }
```

```
}
```

```
public void ValidarEmail()
```

```
{
```

```
    if (string.IsNullOrEmpty(this._email))
```

```
    {
```

```
        throw new Exception("El email no puede estar vacío o solo contener  
espacios.");
```

```
    }
```

```
    bool tieneArroba = false;
```

```
    bool tienePunto = false;
```

```
    foreach (char c in this._email)
```

```
    {
```

```
        if (c == '@')
```

```
        {
```

```
            tieneArroba = true;
```

```

    }

    else if (tieneArroba && c == '.')

    {

        tienePunto = true;

    }

}

if (!tieneArroba || !tienePunto)

{

    throw new Exception("El correo debe contener '@' y un '.' después del '@'.");

}

}

public override bool Equals(object? obj)

{

    if (obj == null || !(obj is Usuario)) return false;

    Usuario otro = (Usuario)obj;

    return this._email.Equals(otro._email);

}

public override string ToString()

```

```

        {
            return $" {_email}";
        }

    }

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Globalization;

using Dominio.Interfaces;

namespace Dominio
{
    public class Vuelo : IValidable
    {

```

```
private int _numVuelo;
```

```
private Avion _avion; //SE PASA OBJETO
```

```
private Ruta _ruta;
```

```
private List<DayOfWeek> _frecuencia;
```

```
private decimal _costoXAsiento;
```

```
public int NumVuelo { get { return _numVuelo; } }
```

```
        public List<DayOfWeek> Frecuencia { get { return new  
List<DayOfWeek>(_frecuencia); } } //para pueda ser accesible desde pasaje que lo  
vamos usar.
```

```
public Ruta Ruta { get { return this._ruta; } } //para sean publicas para la PARTE B
```

```
        public Avion Avion { get { return this._avion; } } ////para sean publicas para la  
PARTE B
```

```
public decimal CostoXAsiento { get { return _costoXAsiento; } }
```

```
public Vuelo(int numVuelo, Avion avion, Ruta ruta, List<DayOfWeek> frecuencia)
```

```
{
```

```
    this._numVuelo = numVuelo;  
  
    this._avion = avion;  
  
    this._ruta = ruta;  
  
    this._frecuencia = frecuencia;  
  
    this.Validar();  
  
}
```

```
public void Validar()  
{  
  
    this.ValidarAvionPuedeCompletarRuta();  
  
    this.ValidarFrecuencia();  
  
}
```

```
public void ValidarAvionPuedeCompletarRuta()  
{  
  
    if (_avion.Alcance < _ruta.Distancia)  
  
    {  
  
        throw new Exception("El avión no puede recorrer la totalidad de la distancia  
de la ruta");  
  
    }  
  
}
```

```
public void ValidarFrecuencia()
```

```
{
```

```
    if (_frecuencia == null || _frecuencia.Count == 0)
```

```
        throw new Exception("La frecuencia del vuelo no puede estar vacía.");
```

```
}
```

```
public decimal CalcularCostoPorAsiento()
```

```
{
```

```
    decimal costoXKm = _avion.CostoXKm;
```

```
    decimal distanciaRuta = _ruta.Distancia;
```

```
        decimal costoOppAeropuertos = _ruta.AeropuertoSalida.CostoOpp +  
_ruta.AeropuertoLlegada.CostoOpp;
```

```
    int cantAsientos = _avion.CantAsientos;
```

```
    return ((costoXKm * distanciaRuta) + costoOppAeropuertos) / cantAsientos;
```

```
}
```

```
public string ObtenerFrecuenciaFormateada() //para poder mostrar la frecuencia  
como string en el programa, porque se guarda en formato DayOfWeek
```

```
{
```

```
    string resultado = "";
```

```

CultureInfo cultura = new CultureInfo("es-ES");

for (int i = 0; i < _frecuencia.Count; i++)
{
    string diaEnEspanol = cultura.DateTimeFormat.GetDayName(_frecuencia[i]);

    diaEnEspanol = char.ToUpper(diaEnEspanol[0]) +
diaEnEspanol.Substring(1);

    resultado += diaEnEspanol;

    if (i < _frecuencia.Count - 1)
    {
        resultado += " - ";
    }
}

return resultado;
}

```

//Metodo AUXILIAR para obtener el costo de las tasas portuarias de cada aeropuerto involucrado, sumarlas y obtener el total.

//Luego lo llamamos en cada una de las subclases, para sumar el valor en la operación, dependiendo si es ocasional o premium.

```
public decimal ObtenerCostoTasasPortuarias()
{
    decimal costoAeropuertoSalida = this._ruta.AeropuertoSalida.CostoTasas;
    decimal costoAeropuertoLlegada = this._ruta.AeropuertoLlegada.CostoTasas;

    decimal costoTasasTotal = costoAeropuertoLlegada + costoAeropuertoSalida;

    return costoTasasTotal;
}

public decimal CalcularCostoSinEquipajeParaVistaDetails()
{
    decimal costoXAsiento = CalcularCostoPorAsiento();
    return costoXAsiento + (costoXAsiento * Pasaje.PORCENTAJE_GANANCIA);
}

public override bool Equals(object? obj)
```



```

    {
        if (obj == null || !(obj is Vuelo)) return false;

        Vuelo otro = (Vuelo)obj;

        return this._numVuelo.Equals(otro._numVuelo);
    }

    public override string ToString()
    {
        return $"Vuelo # {this._numVuelo} | Modelo de avion: {this._avion} | Ruta:
{this._ruta} Frecuencia: {this.ObtenerFrecuenciaFormateada()}\n";
    }

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

```

```

namespace Dominio.Comparadores

{

    //NOS SIRVE PARA ORDENAR LOS PASAJES POR PRECIO DESCENDENTE
    PARA EL CLIENTE

    public class CompararPasajePorPrecio : IComparer<Pasaje>

    {

        public int Compare(Pasaje? este, Pasaje? otro)

        {

                                                                    return
este.CalcularPrecioDelPasaje().CompareTo(otro.CalcularPrecioDelPasaje()) * -1;

        }

    }

}

using System.Runtime.CompilerServices;

using Dominio;

using Microsoft.AspNetCore.Mvc;

namespace WebApp_Obligatorio_P2.Controllers

{

    public class HomeController : Controller

    {

```

```

private Sistema _sistema = Sistema.Instancia;

public IActionResult Index( string mensaje)

{

    if (string.IsNullOrEmpty(mensaje)) //para el error de mensaje ya logueado

    {

        mensaje = HttpContext.Session.GetString("mensaje");

    }

    ViewBag.Mensaje = mensaje;

    return View();

}

public IActionResult Login(string mensaje)

{

    ViewBag.Mensaje = mensaje;

    return View();

}

[HttpPost]

public IActionResult Login(string email, string password)

{

```

```

try

{

    Usuario logueado = _sistema.Login(email, password);


    HttpContext.Session.SetString("email", email);


    HttpContext.Session.SetString("rol", logueado.GetType().Name);


    return RedirectToAction("Index", "Home");


}

catch (Exception ex)

{

    return RedirectToAction("Login", "Home", new { mensaje = ex.Message, });

}

}

public IActionResult Logout()

{

    HttpContext.Session.SetString("email", "");

```

HttpContext.Session.Clear(); // Limpia toda la sesión para me deje ingresar de nuevo y no me lance la excepcion de ya estoy registrado o logueado.

return RedirectToAction("Login");

}

}

}

using Dominio;

using Microsoft.AspNetCore.Mvc;

using WebApp_Obligatorio_P2.Filters;

namespace WebApp_Obligatorio_P2.Controllers

{

public class OcasionalController : Controller

```

{

private Sistema _sistema = Sistema.Instancia;


public IActionResult RegistrarOcasional()

{

    if (!string.IsNullOrEmpty(HttpContext.Session.GetString("email")))

    {

        HttpContext.Session.SetString("mensaje", "Ya estás registrado/a y
logueado/a.");

        return RedirectToAction("Index", "Home");

    }

    return View();

}


[HttpPost]

public IActionResult RegistrarOcasional(Ocasional ocasional)

{

    try

    {

        _sistema.DarDeAltaUsuario(ocasional);

```

```

        HttpContext.Session.SetString("email", ocasional.Email);

        HttpContext.Session.SetString("password", ocasional.Password);

        HttpContext.Session.SetString("rol", ocasional.GetType().Name); // para
cuando se registre guarde el rol en la sesión


        return RedirectToAction("Index", "Home");

    }

    catch (Exception ex)

    {

        ViewBag.Error = ex.Message;

        return View();

    }

}

}

}

using Dominio;

```

```

using Microsoft.AspNetCore.Mvc;

using WebApp_Obligatorio_P2.Filters;

namespace WebApp_Obligatorio_P2.Controllers
{
    [Authentication]

    public class PasajeController : Controller
    {
        private Sistema _sistema = Sistema.Instancia;

        public IActionResult Index()
        {

            if (HttpContext.Session.GetString("rol") == "Administrador")
            {
                return RedirectToAction("VerTodosLosPasajes");
            }
            else
            {
                return RedirectToAction("VerPasajesUsuario");
            }
        }
    }
}

```



```
}
```

```
[SoloPasajero]
```

```
public IActionResult VerPasajesUsuario()
```

```
{
```

```
    string mailLogueado = HttpContext.Session.GetString("email");
```

```
    Pasajero pasajeroLogueado =  
(Pasajero)_sistema.ObtenerUsuarioPorMail(mailLogueado);
```

```
    _sistema.OrdenarPasajesPorPrecio();
```

```
    return View("Index",  
_sistema.ObtenerListaPasajeDeUsuario(pasajeroLogueado));
```

```
}
```

```
[SoloAdmin]
```

```
public IActionResult VerTodosLosPasajes()
```

```
{
```

```

        _sistema.OrdenarPasajes();

        return View("Index", _sistema.Pasajes);
    }

```

[HttpPost]

[SoloPasajero]

```

        public IActionResult Add(int numVuelo, DateTime fecha, TipoEquipaje?
tipoEquipaje)
        {
            try
            {
                Vuelo vuelo = _sistema.ObtenerVueloPorNumVuelo(numVuelo);

                string mailLogueado = HttpContext.Session.GetString("email");

                Pasajero logueado =
(Pasajero)_sistema.ObtenerUsuarioPorMail(mailLogueado);

                Pasaje nuevo = new Pasaje(vuelo, logueado, fecha, tipoEquipaje.Value);

                _sistema.EsTipoEquipajeValido(tipoEquipaje);

```

```
_sistema.AgregarPasaje(nuevo);
```

```
        return RedirectToAction("Index", "Vuelo", new { mensaje = $"Se compró el  
pasaje para la fecha " +
```

```
        $"{fecha.ToString("dd MMM yyyy")}"}");
```

```
    } catch (Exception ex)
```

```
    {
```

```
        return RedirectToAction("Index", "Vuelo", new { mensaje = ex.Message,});
```

```
    };
```

```
}
```

```
}
```

```
}
```

```
using System.Linq.Expressions;
```

```
using Dominio;
```

```

using Microsoft.AspNetCore.Mvc;

using WebApp_Obligatorio_P2.Filters;

namespace WebApp_Obligatorio_P2.Controllers
{
    [Authentication]

    public class PasajeroController : Controller
    {
        private Sistema _sistema = Sistema.Instancia;

        [SoloAdmin]

        public IActionResult Index(string mensaje)
        {

            ViewBag.Mensaje = mensaje;

            return View(_sistema.ListarPasajeros());
        }

        [SoloPasajero]

        public IActionResult VerPerfil()

```

```

{

    string mailLogueado = HttpContext.Session.GetString("email");

    Pasajero logueado =
(Pasajero)_sistema.ObtenerUsuarioPorMail(mailLogueado);

    return View(logueado);

}

[HttpPost]

[SoloAdmin]

public IActionResult Index(int puntos, string elegible, string pasajeroEmail)

{

    try

    {

        foreach (Usuario usuario in _sistema.Usuarios) {

            if (usuario.Email == pasajeroEmail && usuario is Premium)

            {

                _sistema.ActualizarPuntosPremium(puntos, pasajeroEmail);

                return RedirectToAction("Index", new { mensaje = "Puntos actualizados
correctamente" });
            }
        }
    }
}

```

```

    }

    else if(usuario.Email == pasajeroEmail && usuario is Ocasional)

    {

        bool esElegible = elegible == "true";

        _sistema.ActualizarElegibilidadOcasional(pasajeroEmail, esElegible);

        return RedirectToAction("Index", new { mensaje = "Elegibilidad
actualizada correctamente" });

    }

}

return View();

} catch (Exception ex)

{

    return RedirectToAction("Index", new { mensaje = ex.Message });

}

}

}

```

```

    }

    using Dominio;

    using Microsoft.AspNetCore.Mvc;

    using WebApp_Obligatorio_P2.Filters;


    namespace WebApp_Obligatorio_P2.Controllers
    {

        [Authentication]

        [SoloPasajero]

        public class VueloController : Controller
        {

            private Sistema _sistema = Sistema.Instancia;

            public IActionResult Index(string mensaje)
            {

                ViewBag.Mensaje = mensaje;

                ViewBag.Aeropuertos = _sistema.Aeropuertos;

                return View(_sistema.Vuelos);

            }

            [HttpPost]

```

```

public IActionResult Index(string IATAsalida, string IATAllegada)
{
    try
    {
        ViewBag.Aeropuertos = _sistema.Aeropuertos;

        List<Vuelo> vuelos = _sistema.ListarVuelosPorAeropuerto(IATAsalida,
IATAllegada);

        return View(vuelos);
    }
    catch (Exception ex)
    {

        return RedirectToAction("Index", "Vuelo", new { mensaje = ex.Message, });
    }
    ;
}

```



```

public IActionResult Details(int id)
{
    Vuelo vuelo = _sistema.ObtenerVueloPorNumVuelo(id);

    return View(vuelo);
}
}
}

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

namespace WebApp_Obligatorio_P2.Filters
{
    public class Authentication : ActionFilterAttribute
    {
        public override void OnActionExecuting(ActionExecutingContext context)
        {
            string logueado = context.HttpContext.Session.GetString("email");

            if (string.IsNullOrEmpty(logueado))

```

```
        context.Result = new RedirectToActionResult("index","Home", new {  
mensaje = "Debe iniciar sesión para acceder a esa función." });
```

```
        base.OnActionExecuting(context);
```

```
    }
```

```
}
```

```
}
```

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

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

```
namespace WebApp_Obligatorio_P2.Filters
```

```
{
```

```
    public class SoloAdmin : ActionFilterAttribute
```

```
    {
```

```
        public override void OnActionExecuting(ActionExecutingContext context)
```

```
        {
```

```
            string rol = context.HttpContext.Session.GetString("rol");
```

```
            if (rol != "Administrador")
```

```
            {
```

```

        context.Result = new RedirectToActionResult("Index", "Home", new {
mensaje = "Acceso no autorizado." });

    }

    base.OnActionExecuting(context);

}

}

}

using Microsoft.AspNetCore.Mvc.Filters;

using Microsoft.AspNetCore.Mvc;

namespace WebApp_Obligatorio_P2.Filters

{

    public class SoloPasajero : ActionFilterAttribute

    {

        public override void OnActionExecuting(ActionExecutingContext context)

        {

            string rol = context.HttpContext.Session.GetString("rol");

            if (rol != "Ocasional" && rol != "Premium")

            {

```

```
        context.Result = new RedirectToActionResult("Index", "Home", new {  
    mensaje = "Solo los pasajeros pueden acceder a esta función." });
```

```
    }
```

```
    base.OnActionExecuting(context);
```

```
    }
```

```
}
```

```
}
```

```
@using Dominio
```

```
@model Pasajero
```

```
@{
```

```
    ViewData["Title"] = "Pagina bienvenida ";
```

```
}
```

```
<h2>Bienvenida/o @Context.Session.GetString("email")</h2>
```

```
@if (!string.IsNullOrEmpty(ViewBag.Mensaje))
```

```
{
```

```
    <div class="alert alert-danger">
```

```
        @ViewBag.Mensaje
```

```

</div>

}

@{
    ViewData["Title"] = "Iniciar sesión ";
}

<div id="login" class="container p-4">

    <div class="align-content-between">

        <head>

            <title>@ViewData["Title"]="Iniciar sesión";</title>

        </head>

        <h2 class="text-center">Ingrese sus datos para iniciar sesion</h2>

        @if (!string.IsNullOrEmpty(ViewBag.Mensaje))
        {
            <div class="alert alert-danger">

```

```

        @ViewBag.Mensaje

    </div>

}

<div class=" d-flex justify-content-center align-content-center">

    <h3> @Context.Session.GetString("Email")</h3>

    @if (string.IsNullOrEmpty(Context.Session.GetString("email")))
    {

        <form class="d-flex row justify-content-center align-content-between py-2"
method="post">

            <div class="form-group col-12 my-4">

                <label for="">Ingrese su correo:</label>

                <input type="email" class="form-control" name="email"
id="exampleInputEmail1" aria-describedby="emailHelp" placeholder="Ingrese su
correo">

            </div>

```

```

<div class="form-group col-12 my-4">

    <label for="exampleInputPassword1">Ingrese su contraseña:</label>

    <input type="password" class="form-control" name="password"
id="exampleInputPassword1" placeholder="Ingrese su contraseña">

</div>

<div>

    <button type="submit" class="btn btn-primary col-3 my-1">Iniciar
sesión</button>

</div>

<div>

    <a href="~/Ocasional/RegistrarOcasional" class="btn btn-primary col-3
my-1">Registrarse</a>

</div>

</form>

}

</div>

</div>

</div>

@{

    ViewData["Title"] = "Registrarse ";

```

```
}
```

```
@using Dominio
```

```
@model Ocasional
```

```
<h2>REGISTRARSE</h2>
```

```
@if (!string.IsNullOrEmpty(ViewBag.Error)){
```

```
<div class="alert alert-danger">
```

```
@ViewBag.Error
```

```
</div>
```

```
}
```

```
<form method="post">
```

```
@*NACIONALIDAD*@
```

```
<div class="form-group">
```


<label for="txtNacionalidad">Nacionalidad</label>

<input type="text" class="form-control" name="nacionalidad"
id="txtNacionalidad" placeholder="Ingrese nacionalidad..">

</div>

@*DOCUMENTO DE IDENTIDAD*@

<div class="form-group">

<label for="txtDocIdentidad">Documento de identidad</label>

<input type="text" class="form-control" name="docIdentidad"
id="txtDocIdentidad" placeholder="Ingrese su documento de identidad..">

</div>

@*NOMBRE*@

<div class="form-group">

<label for="txtNombre">Nombre </label>

<input type="text" class="form-control" name="nombre" id="txtNombre"
placeholder="Ingrese su nombre..">

</div>

@*CONTRASEÑA*@

```
<div class="form-group">
```

```
<label for="txtContraseña">Ingrese su Contraseña</label>
```

```
<input type="password" class="form-control" name="password"
id="txtContraseña" placeholder="Ingrese su contraseña..">
```

```
</div>
```

@*EMAIL*@

```
<div class="form-group">
```

```
<label for="email">Ingrese su email</label>
```

```
<input type="email" class="form-control" id="email" name="email"
placeholder="Ingrese su email..">
```

```
</div>
```

```
<br/>
```

```
<div class="d-flex gap-3">
```

```
<button type="submit" class="btn btn-primary">Registrarse</button>
```

@*ANCLA PARA A FUTURO REDIRIJA A LA VISTA DE ANONIMO
(PREVIO A REGISTRARSE) *@

```
<a href="~/Home/Login" class="btn btn-outline-info"> Volver</a>
```

</div>

</form>

@{

 ViewData["Title"] = "Lista de pasajes ";

}

@using Dominio;

@model List<Pasaje>

<h2>LISTA DE PASAJES:</h2>

@if (Model.Count() == 0)

{

 <div class="alert alert-danger">

 <p>No hay pasajes para mostrar. </p>

```
</div>
```

```
}
```

```
else
```

```
{
```

```
<table class="table table-hover table-dark">
```

```
<thead>
```

```
<tr>
```

```
<th scope="col">ID</th>
```

```
<th scope="col">RUTA</th>
```

```
<th scope="col">PASAJERO</th>
```

```
<th scope="col">FECHA</th>
```

```
<th scope="col">TIPO EQUIPAJE</th>
```

```
<th scope="col">PRECIO</th>
```

```
</tr>
```

```
</thead>
```

```
<tbody>
```

```
@foreach (Pasaje pasaje in Model)
```

```
{
```

```
<tr>
```

```
<td>@pasaje.Id</td>
```

```
<td>@pasaje.Vuelo.Ruta</td>
```

```
<td>@pasaje.Pasajero.Nombre</td>
```

```
<td>@pasaje.Fecha.ToShortDateString()</td>
```

```
<td>@pasaje.TipoEquipaje</td>
```

```
<td>${@pasaje.Precio.ToString("F2")}</td>
```

```
</tr>
```

```
}
```

```
</tbody>
```

```
</table>
```

```
}
```

```
@{
```

```
    ViewData["Title"] = "Lista de pasajeros ";
```

```
}
```

```
@using Dominio;
```

```
@model List<Pasajero>
```

```
<h2>LISTA DE PASAJEROS:</h2>
```

```
@if (!string.IsNullOrEmpty(ViewBag.Mensaje))
```

```
{
```

```
<div class="alert alert-success">
```

```
    @ViewBag.Mensaje
```

```
</div>
```

```
}
```

```
@if (Model.Count() == 0)
```

```
{
```

```
<div class="alert alert-danger">
```

```
<p>No hay pasajeros para mostrar. </p>
```

```
</div>
```

```
}
```

```
else
```

```
{
```

```
<table class="table table-hover table-dark">
```

```
<thead>
```

```
<tr>
```

```
<th scope="col">NOMBRE</th>
```

```
<th scope="col">EMAIL</th>
```

```
<th scope="col">NACIONALIDAD</th>
```

```
<th scope="col">DOCUMENTO</th>
```

```
<th scope="col">PUNTOS / ELEGIBLE</th>
```

```
</tr>
```

```
</thead>
```

```
<tbody>
```

```
@foreach (Pasajero pasajero in Model)
```

```
{
```

```
<tr>
```

```
<td>@pasajero.Nombre</td>
```

```
<td>@pasajero.Email</td>
```

```
<td>@pasajero.Nacionalidad</td>
```

```
<td>@pasajero.DocIdentidad</td>
```

```
<td>
```

```
<form method="post" action ="/Pasajero/Index">
```

```
@if (pasajero is Premium premium)
```

```
{
```

```
<input type="hidden" name="pasajeroEmail"
```

```
value="@premium.Email" />
```

```
<p>Tiene @premium.Puntos puntos.</p>
```

```
<input type="number" name="puntos"
```

```
value="@premium.Puntos"/>
```

```
<input type="submit" value="Actualizar"/>
```

```
} else if (pasajero is Ocasional ocasional)
```



```

{
    <input type="hidden" name="pasajeroEmail"
value="@ocasional.Email" />

    <p>@(ocasional.elegible ? "Si" : "No") es elegible</p>

    <select name="elegible">

        <option value="true" selected="@(ocasional.elegible ? "
selected" : null)"> Si </option>

        <option value="false" selected="@(ocasional.elegible ? null :
"selected")"> No </option>

    </select>

    <input type="submit" value="Actualizar" />

}

</form>

</td>

</tr>

```

```

    }

</tbody>

</table>

}

@{
    ViewData["Title"] = "Mi perfil ";
}

```

```
@using Dominio
```

```
@model Pasajero
```

```
<h2>Perfil de: @Model.Nombre</h2>
```

```
<table class="table table-hover table-info">
```

```
    <tbody>
```

```
        <tr>
```

```
            <th>Nombre</th>
```

```

        <td>@Model.Nombre</td>

</tr>

<tr>

    <th>Email</th>

    <td>@Model.Email</td>

</tr>

<tr>

    <th>Nacionalidad</th>

    <td>@Model.Nacionalidad</td>

</tr>

<tr>

    <th>Documento de identidad</th>

    <td>@Model.DocIdentidad</td>

</tr>

@if (Model is Premium premium)
{
    <tr>

        <th>Puntos</th>

        <td>@premium.Puntos</td>

    </tr>

```

```

    }

</tbody>

</table>

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="utf-8" />

    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    <title>@ViewData["Title"]Altavia &trade;</title>

    <link rel="preconnect" href="https://fonts.googleapis.com">

    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>

    <link
href="https://fonts.googleapis.com/css2?family=Playfair+Display:ital,wght@0,400..900
;1,400..900&display=swap" rel="stylesheet">

    <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_Obligatorio_P2.styles.css"
asp-append-version="true" />

    <link rel="icon" type="image/x-icon" href="https://i.imgur.com/e0VxB0m.png">

```

```
</head>
```

```
<body>
```

```
    <header class="container-fluid justify-content-between">
```

```
        <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light
border-bottom box-shadow mb-3">
```

```
            <a asp-area="" asp-controller="Home" asp-action="Index" class="col-6"></a>
```

```
                <button class="navbar-toggler border-light" 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 text-white ">
```

```
        @if (!string.IsNullOrEmpty(Context.Session.GetString("email")))
```

```
        {
```

```
            string rol = Context.Session.GetString("rol");
```

```
<li class="nav-item">

    <a class="nav-link text-dark" asp-area="" asp-controller="Home"
asp-action="Index">Home</a>

</li>
```

```
@* SI ES PASAJERO *@
```

```
@if (rol == "Ocasional" || rol == "Premium")

{

    <li class="nav-item">

        <a class="nav-link text-dark" asp-area="" asp-controller="Pasaje"
asp-action="Index"> Ver pasajes</a>

    </li>
```

```
    <li class="nav-item">

        <a class="nav-link text-dark" asp-area="" asp-controller="Pasajero"
asp-action="VerPerfil"> Mi perfil</a>

    </li>
```

```
    <li class="nav-item">

        <a class="nav-link text-dark" asp-area="" asp-controller="Vuelo"
asp-action="Index"> Ver vuelos</a>

    </li>
```

```

    }

    @if (rol == "Administrador")
    {
        <li class="nav-item">

            <a class="nav-link text-dark" asp-area="" asp-controller="Pasajero"
asp-action="Index"> Ver pasajeros</a>

        </li>

        <li class="nav-item">

            <a class="nav-link text-dark" asp-area="" asp-controller="Pasaje"
asp-action="Index"> Ver pasajes</a>

        </li>

    }

    <li class="nav-item">

        <a class="nav-link text-dark" asp-area="" asp-controller="Home"
asp-action="Logout"> Cerrar Sesion</a>

    </li>

}

</ul>

```

```

        </div>

    </nav>

</header>

<div class="container">

    <main role="main" class="pb-3">

        @RenderBody()

    </main>

</div>


<footer class="border-top container-fluid footer text-muted">

    <div class="d-flex justify-content-between align-items-center">

        <div>

        </div>

        &copy;Altavia 2025 - Todos los derechos reservados - <a asp-area=""
asp-controller="Home" asp-action="Privacy">Privacy</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>

@{
    ViewData["Title"] = "Detalles del vuelo ";
}

@using Dominio

@model Vuelo

<div class="card text-dark" style="width: 28rem;">

    <div class="card-body">

        <h5 class="card-title">Vuelo @Model.Ruta</h5>

        <h6 class="card-subtitle mb-2 text-body-secondary">Precio :
        $@(Model.CalcularCostoSinEquipajeParaVistaDetails().ToString("F2"))</h6>

        <h6 class="mt-3">Frecuencia:</h6>

        <p>

```

```

@foreach (DayOfWeek dia in Model.Frecuencia)

{

    @dia <br />

}

</p>

<a href="~/Vuelo/Index" class="btn btn-primary">Volver</a>

<form action="~/Pasaje/Add" method="post">

    <input type="hidden" value="@Model.NumVuelo" name="numVuelo" />

    <input type="date" name="fecha" />

    <select name="tipoEquipaje">

        <option value="">-----</option>

        @foreach (string equipaje in Enum.GetNames(typeof(TipoEquipaje)))

        {

            <option value="@equipaje"> @equipaje</option>

            <br />

        }

    </select>

    <input type="submit" class="btn btn-success" value="Comprar pasaje" />

```

</form>

</div>

</div>

@{

ViewData["Title"] = "Lista de vuelos ";

}

@using Dominio;

@model List<Vuelo>

<div class="d-flex justify-content-center align-items-center">

<h2 class="m-4">LISTA DE VUELOS:</h2>

<div class="d-block">

```
@if (!string.IsNullOrEmpty(ViewBag.Mensaje))
```

```
{
```

```
<div class="alert alert-success">
```

```
    @ViewBag.Mensaje
```

```
</div>
```

```
}
```

```
@if (Model.Count() == 0)
```

```
{
```

```
<div class="alert alert-danger">
```

```
<p>No hay vuelos para mostrar. </p>
```

```
</div>
```

```
} else
```

```
{
```

```
<form method="post" class="my-2">
```

```
<select name="IATAsalida">
```

```
<option value="">Elija un codigo de aeropuerto de salida</option>
```

```
@foreach (Aeropuerto aeropuerto in ViewBag.Aeropuertos)
```

```
{
```

```

value="@aeroporto.IATACode">@aeroporto.IATACode</option>
</option>

}

</select>

<select name="IATAllegada">

    <option value="">Elija un codigo de aeropuerto de llegada</option>

    @foreach (Aeropuerto aeropuerto in ViewBag.Aeropuertos)

    {

        value="@aeroporto.IATACode">@aeroporto.IATACode</option>
        </option>

    }

</select>

<input class="px-1 mx-2" type="submit" value="Filtrar">

</form>

<table class="table table-hover table-dark">

```

```
<thead>
```

```
<tr>
```

```
<th scope="col">NUMERO DE VUELO</th>
```

```
<th scope="col">RUTA</th>
```

```
<th scope="col">FRECUENCIA</th>
```

```
<th scope="col"></th>
```

```
<th scope="col"></th>
```

```
</tr>
```

```
</thead>
```

```
<tbody>
```

```
@foreach (Vuelo vuelo in Model)
```

```
{
```

```
<tr>
```

```
<td>@vuelo.NumVuelo</td>
```

```
<td>@vuelo.Ruta</td>
```

```

<td>

    @foreach (DayOfWeek dia in vuelo.Frecuencia) //para mostrar la
frecuencia

    {

        @dia <br />

    }

</td>

    <td><a href="~/Vuelo/Details/@vuelo.NumVuelo" class="btn
btn-info">Ver detalle</a></td>

<td>

    <form action="~/Pasaje/Add" method="post">

        <input type="hidden" value="@vuelo.NumVuelo"
name="numVuelo" />

        <input type="date" name="fecha"/>

        <select name="tipoEquipaje">

            <option value="">-----</option>

            @foreach (string equipaje in
Enum.GetNames(typeof(TipoEquipaje)))

            {

                <option value="@equipaje"> @equipaje</option>

                <br />

```

```

        }

    </select>

    <input type="submit" class="btn btn-success" value="Comprar
pasaje" />

</form>

</td>

</tr>

}

</tbody>

</table>

}

</div>

</div>

```