**Universidad ORT Uruguay**

**Facultad de Ingeniería**

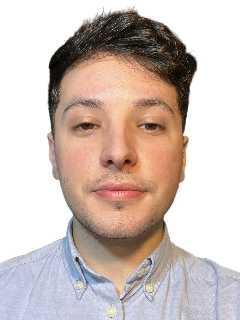
**Escuela de Tecnología**

**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 12/05/2025**

**Índice**

[**1. Descripción general del problema a resolver 3**](#_heading=h.68idtergn8sn)

[**2. Estructura del Sistema 4**](#_heading=h.t99a5zsvc5qs)

[2.1 Entidades Principales y atributos 4](#_heading=h.pf2ov5s2cgnc)

[**3. Funcionalidades Implementadas 4**](#_heading=h.ddxcg6th6gh3)

[3.1 Funcionalidad A - Listado de Pasajeros 4](#_heading=h.1jprpb3a5whi)

[3.2 Funcionalidad B - Listado de Vuelos por Código IATA 5](#_heading=h.38o9yujksdrj)

[3.3 Funcionalidad C - Alta de Cliente Ocasional 6](#_heading=h.yzpvdnvngypc)

[3.3.1 Evitar duplicados 7](#_heading=h.ncljfc6upu38)

[3.4 Funcionalidad D - Listado de Pasajes entre dos fechas 7](#_heading=h.rb8za91pdpmt)

[3.4.1 Inexistencia de pasajes: 8](#_heading=h.u23nuei6w5vm)

[**4. Validaciones Relevantes por Clase 9**](#_heading=h.ii89w52h7amo)

[4.1 Aeropuerto 9](#_heading=h.x0q32f7bqimg)

[4.1.1 Atributos: 9](#_heading=h.w6ur38j7wlbr)

[4.1.2 Validaciones: 9](#_heading=h.gdjc3yuh54rf)

[4.2 Avión 10](#_heading=h.ecfygiwr3tfr)

[4.2.1 Atributos: 10](#_heading=h.eas9vi2zuw1c)

[4.2.2 Validaciones: 10](#_heading=h.p61gtjhw1zdr)

[4.3 Ruta 11](#_heading=h.ghmtu56njrt1)

[4.3.1 Atributos: 11](#_heading=h.yr317lrrr717)

[4.3.2 Validaciones: 11](#_heading=h.ieqmhn8mit6f)

[4.4 Vuelo 11](#_heading=h.bkhuhd2p3di0)

[4.4.1 Atributos: 11](#_heading=h.ci2hziclvjsl)

[4.4.2 Validaciones: 12](#_heading=h.9a2931a6ssxf)

[4.5 Pasaje 12](#_heading=h.px5lz3gysyk)

[4.5.1 Atributos: 12](#_heading=h.4b6pp9h5dzll)

[4.5.2 Validaciones: 12](#_heading=h.bicwg7mry9ww)

[4.6 Usuario 13](#_heading=h.e2ukb9ldcah4)

[4.6.1 Atributos: 13](#_heading=h.mxoxjlk8zwow)

[4.6.2 Validaciones: 13](#_heading=h.7nvhzpj4iafe)

[4.7 Administrador 13](#_heading=h.9k9rhl2wev6a)

[4.7.1 Atributo: 13](#_heading=h.o0pjtst3x0ug)

[4.7.2 Validación: 13](#_heading=h.5jjk6tf2nhik)

[4.8 Clase Pasajero (abstracta de la que heredan Ocasional y Premium) 14](#_heading=h.wy86t93f4rbb)

[4.8.1 Atributos: 14](#_heading=h.daewrw7b2kbk)

[4.8.2 Validaciones: 14](#_heading=h.wrvknb11e474)

[4.9 Clase Ocasional (hereda de Pasajero) 14](#_heading=h.jnqd1n7n7ure)

[4.9.1 Atributos: 14](#_heading=h.daw2tmq40ub5)

[4.9.2 Lógica específica de esta clase: 15](#_heading=h.j326f23m0dm8)

[4.10 Clase Premium (hereda de Pasajero) 15](#_heading=h.i7cqbwwkcd4j)

[4.10.1 Atributo: 15](#_heading=h.cf91hwdzg0nx)

[**5. MÉTODOS AUXILIARES 15**](#_heading=h.r7orooafiyo9)

[5.1 SolicitarInt(string mensaje, int maximo, int minimo) 15](#_heading=h.x6fcfmvavtq1)

[5.2 SolicitarDecimal(string mensaje, decimal maximo, decimal minimo) 16](#_heading=h.v0m55mzgjj2i)

[5.3. SolicitarDateTime(string textoFecha) 16](#_heading=h.t9famkrh77mp)

[5.4 ExistePasajeroCon(string unDocIdentidad) 17](#_heading=h.9q3qlbyj69ha)

[5.5 Validar() (en todas las clases de dominio) 17](#_heading=h.218pjjbjxbzw)

[5.6 CalcularCostoPorAsiento() en Vuelo 18](#_heading=h.3em2bacnhk9h)

[**6. Precarga de Datos 18**](#_heading=h.d12kxga15pp9)

[6.1 Datos generados con prompt: 19](#_heading=h.uv7nfxagwckt)

# 

# Descripción general del problema a resolver

El presente sistema tiene como misión gestionar los distintos tópicos que componen una aerolínea. Esto es, aeropuertos, rutas, aviones, vuelos, pasajeros (tanto ocasionales como premium), administradores y pasajes. Con dicho sistema (y al menos para esta primera parte de Obligatorio P2), se busca una plataforma que permita dar de alta a usuarios ocasionales, realizar listados, controlar la coherencia de los datos ingresados y asegurar la integridad lógica del funcionamiento interno.

# 2. Estructura del Sistema

## 2.1 Entidades Principales y atributos

-**Aeropuerto**: Identificado por un código IATA, posee nombre de ciudad, costo operativo y tasas.

-**Avión**: Contiene datos como fabricante, modelo, alcance, cantidad de asientos, costo por kilómetro y tipo de aeronave.

-**Ruta**: Conecta dos aeropuertos distintos y establece la distancia entre ellos.

-**Vuelo**: Está vinculado a un avión y una ruta. Tiene una frecuencia (día de la semana) y un número identificador.

-**Pasaje**: Relaciona un pasajero con un vuelo en una fecha específica. Incluye tipo de equipaje y precio. Además se calcula mediante un método el costo por asiento.

-**Usuario**: Clase base para los distintos tipos de usuarios.

-**Administrador**: Tiene un nickname y credenciales de acceso.

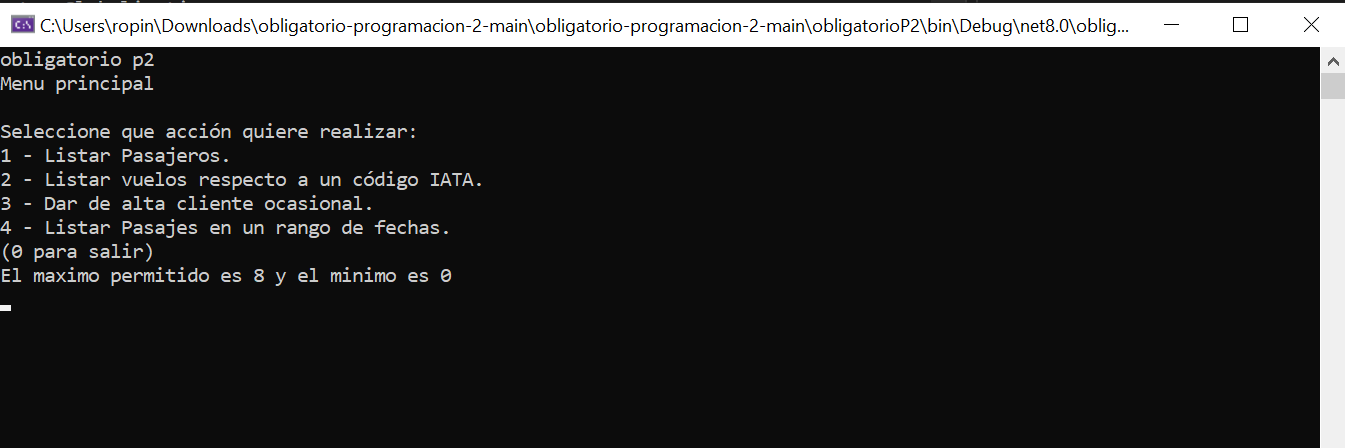
-**Pasajero**: Contiene datos personales, subdividido en:  
 -Premium.  
 -Ocasional (con atributo *elegible*).

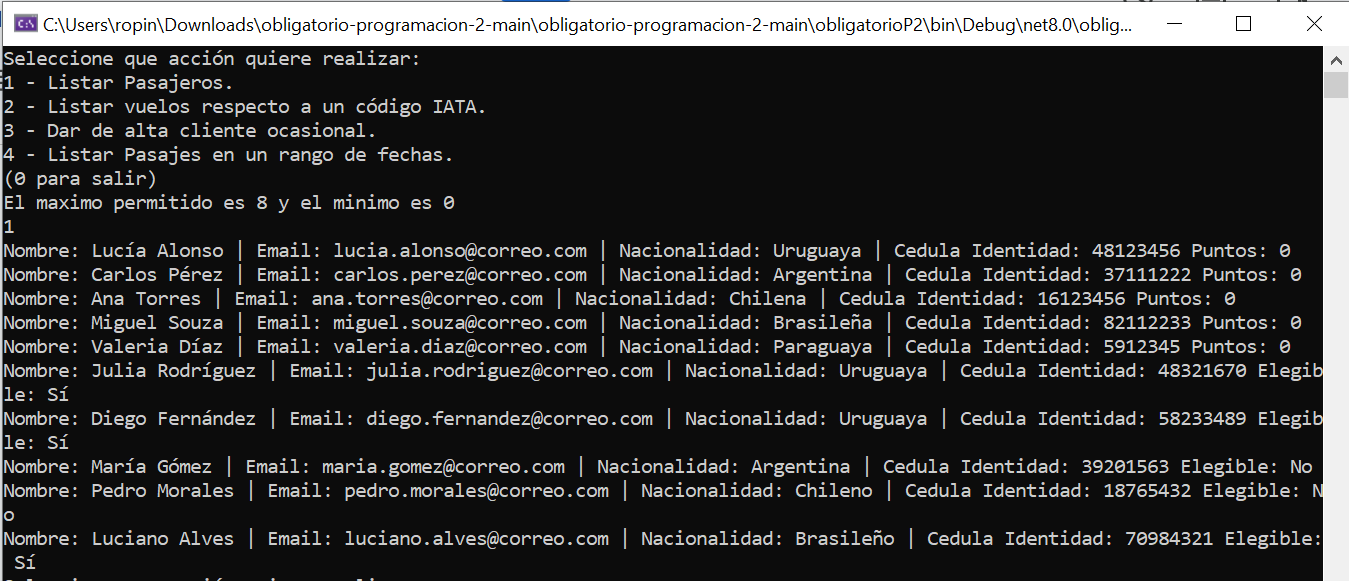
# 3. Funcionalidades Implementadas

## 3.1 Funcionalidad A - Listado de Pasajeros

Para realizar dicha funcionalidad, implementamos un método llamado ***ListarPasajeros****()* dentro de la clase ***Sistema***, que recorre la lista general de usuarios (***\_usuarios****)* y filtra únicamente aquellos que son de tipo ***Pasajero*** (ya sean ***Premium*** u ***Ocasional***). El listado se muestra por consola, utilizando **polimorfismo** a través de ***ToString****().*

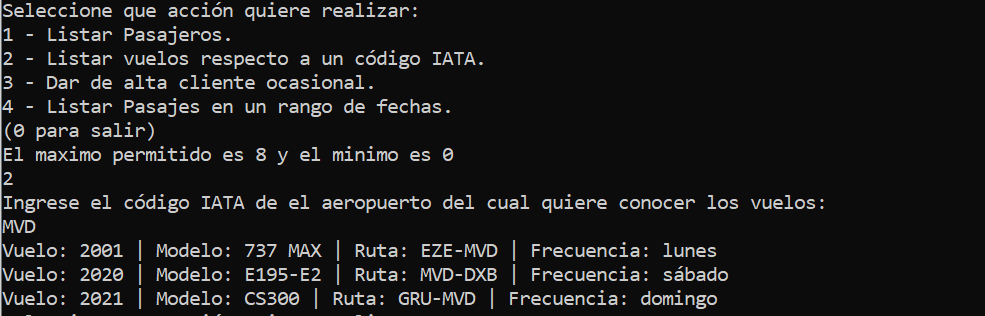
En dicha funcionalidad y puntualmente respecto a la información mostrada, como se visualiza en la imagen de abajo, debía mostrarse: nombre, email, nacionalidad, puntos (solo respecto de los pasajeros premium), y si es elegible o no (solo de los pasajeros

ocasionales).



## 3.2 Funcionalidad B - Listado de Vuelos por Código IATA

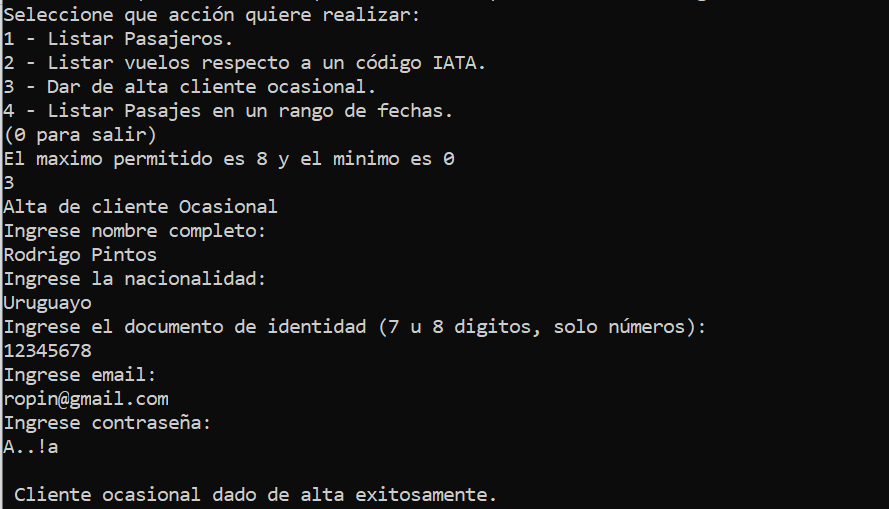
Permite ingresar un código IATA y listar todos los vuelos que lo incluyen, sea como aeropuerto de salida o de llegada. Se muestra el número de vuelo, modelo de avión, ruta (formato MVD-MIA) y su frecuencia.



## 3.3 Funcionalidad C - Alta de Cliente Ocasional

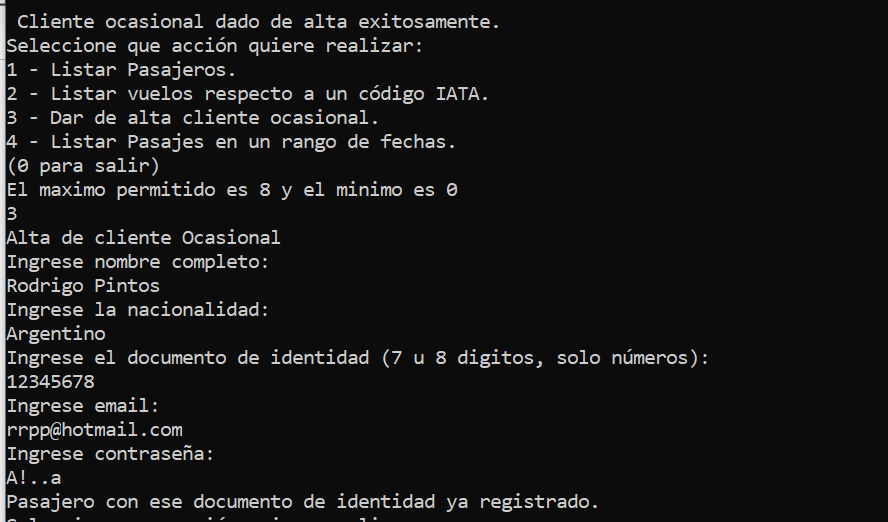
Para esta funcionalidad implementamos el método de ***DarDeAltaClienteOcasional****(...)* en ***Sistema***, donde se valida si ya existe un pasajero con el mismo documento mediante el método auxiliar ***ExistePasajeroCon****(...)*. De no existir, se agrega a la lista de usuarios.

Esto permite registrar un nuevo cliente ocasional con los siguientes datos: nombre completo, nacionalidad, cédula, contraseña y correo electrónico. Una vez ingresados los datos, se visualizará un mensaje de éxito, como se desprende de la captura se adjunta debajo.



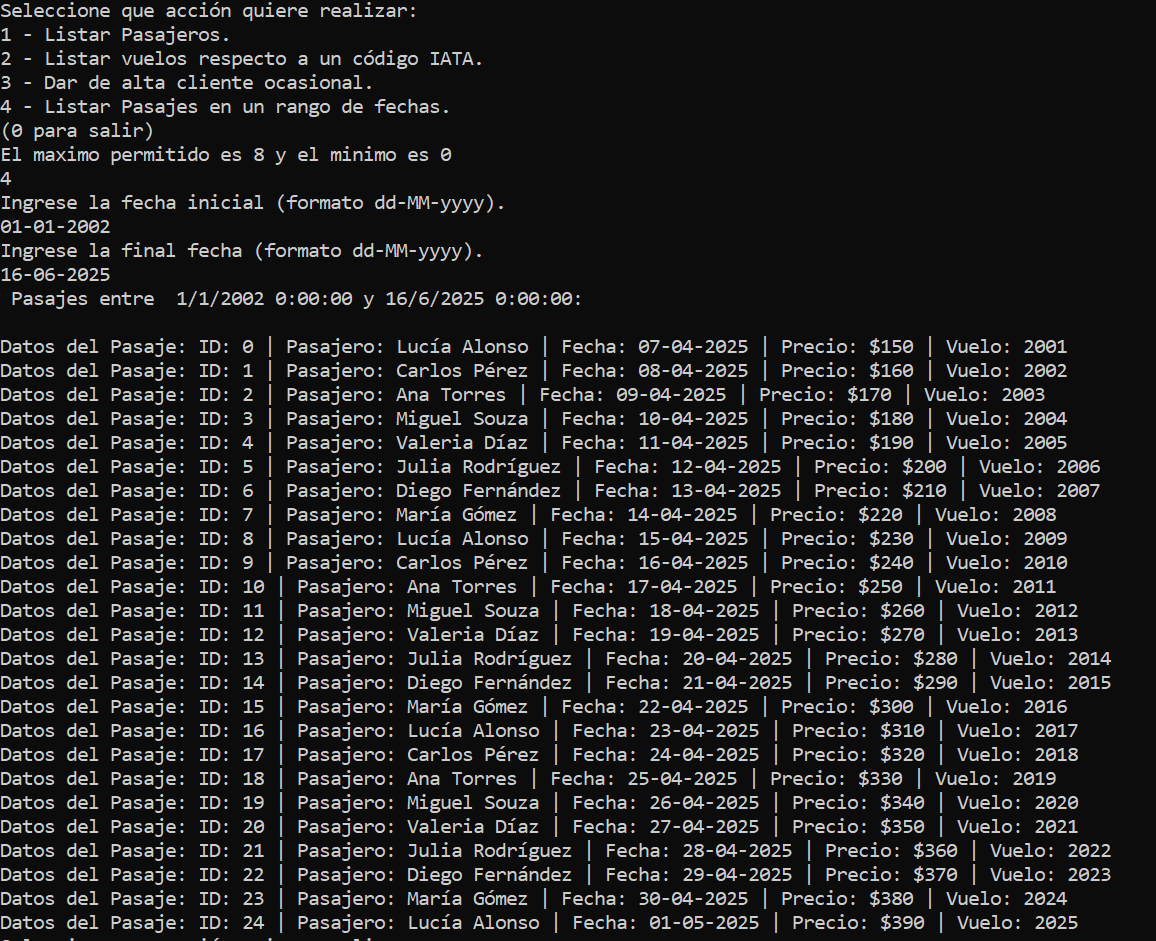
### 3.3.1 Evitar duplicados

Como mencionamos en el punto anterior, se realiza una validación para evitar duplicados mediante el método ***ExistePasajeroCon***. Para el mismo, tomamos como atributo determinante el documento de identidad. Es decir, si existe un cliente en nuestro sistema que ya posea ese documento de identidad, nos mostrará un mensaje de error, como se visualiza en la siguiente captura. (Se ingresó un pasajero con el mismo documento de identidad que usé para el alta del punto 3.3).



## 3.4 Funcionalidad D - Listado de Pasajes entre dos fechas

Para esta funcionalidad, implementamos un método llamado ***ObtenerPasajeEntre(DateTime, DateTime)****.* El mismo permite ingresar una fecha inicial y una final, y listar todos los pasajes expedidos entre esos días (inclusive). El sistema compara las fechas de los pasajes y muestra aquellos que están dentro del rango.



### 3.4.1 Inexistencia de pasajes:

Implementamos un método para que, en el caso de que no existan pasajes expedidos entre las dos fechas ingresadas, se visualice un mensaje de error como el que se muestra a continuación.

# 

# 4. Validaciones Relevantes por Clase

## 4.1 Aeropuerto

### 4.1.1 Atributos:

* ***\_IATACode***: Código IATA de tres letras ( "MVD").
* ***\_Ciudad:*** Ciudad del aeropuerto.
* ***\_costoOpp:*** Costo de operación.
* ***\_costoTasas:*** Costo de tasas aeroportuarias.

### 4.1.2 Validaciones:

* ***ValidarIATA():*** El código debe tener exactamente 3 letras. No se aceptan números, símbolos ni espacios.
* ***ValidarCiudad():*** Verifica que el nombre de la ciudad no sea nulo ni contenga solo espacios.
* ***ValidarCostoOpp():*** El costo debe ser mayor que 0.
* ***ValidarCostoTasas()***: Igual que el anterior, pero aplicado para las tasas.

## 4.2 Avión

### 4.2.1 Atributos:

* ***\_fabricante***: Marca (Boeing, etc.).
* ***\_modelo***: Modelo específico.
* ***\_alcance***: Máxima distancia que puede recorrer.
* ***\_cantAsientos***: Capacidad de pasajeros.
* ***\_costoXKm***: Costo operativo por kilómetro.
* ***\_tipoAeronave***: Tipo.

### 4.2.2 Validaciones:

* ***ValidarFabricante()***: Verifica que no sea nulo o vacío.
* ***ValidarModelo():*** Verifica que no sea nulo o vacío.
* ***ValidarAlcance()***: Debe ser un número mayor que 0.
* ***ValidarCantAsientos()***:Debe ser un número mayor que 0.
* ***ValidarCostXKm()***: Debe ser un número mayor que 0.

## 4.3 Ruta

### 4.3.1 Atributos:

* ***\_id***: Es autoincremental.
* ***\_aeropuertoSalida, \_aeropuertoLlegada***: Objetos Aeropuerto.
* ***\_distancia***: Son los kilómetros del trayecto.

### 4.3.2 Validaciones:

* ***ValidarDistancia()***: No puede ser 0 ni negativa.
* ***ValidarQueAeropuertoNoSeRepita()***: Valida que origen y destino no sean el mismo.

## 4.4 Vuelo

### 4.4.1 Atributos:

* ***\_numVuelo***: Es el número del vuelo..
* ***\_avion***: Objeto Avion.
* ***\_ruta***: Objeto Ruta.
* ***\_frecuencia***: Día de la semana.
* ***\_costoXAsiento***: Cálculo.

### 4.4.2 Validaciones:

* ***ValidarFrecuencia()***: Se valida que coincida con un día válido de la semana (acepta tildes y sin tildes).
* ***ValidarAvionPuedeCompletarRuta()***: El alcance del avión debe ser mayor o igual a la distancia de la ruta
* ***CalcularCostoPorAsiento():*** Usa datos del avión y la ruta para determinar el precio unitario por asiento. Para ello, se sigue la lógica se desprende de la letra del obligatorio para el cálculo de costo por asiento. (párrafo 6).

## 4.5 Pasaje

### 4.5.1 Atributos:

* ***\_id***: Identificador autogenerado.
* ***\_vuelo***: Referencia al objeto Vuelo.
* ***\_pasajero***: Referencia al objeto Pasajero.
* ***\_fecha***: Fecha del pasaje.
* ***\_equipaje***: Refiere al tipo de equipaje.
* ***\_precio***: Refiere al costo del pasaje.

### 4.5.2 Validaciones:

* ***ValidarFechaCorrespondeFrecuencia()***: Verifica que el día de la fecha coincida con la frecuencia del vuelo.

## 4.6 Usuario (abstracta de la que heredan Administrador y Pasajero)

## 

### 4.6.1 Atributos:

* ***\_email***
* ***\_password***

### 4.6.2 Validaciones:

* ***ValidarEmail():*** No vacío ni solo espacios.
* ***ValidarPassword():*** Máximo 64 caracteres. Debe contener al menos:  
  + 1 letra mayúscula
  + 1 letra minúscula
  + 1 número
  + 1 símbolo

## 4.7 Administrador

### 4.7.1 Atributo:

* ***\_nickname***

### 4.7.2 Validación:

* ***ValidarNickname()***: Con este método verificamos que el nickname no esté vacío ni contenga únicamente espacios.

## 4.8 Clase Pasajero (abstracta de la que heredan Ocasional y Premium)

### 4.8.1 Atributos:

* ***\_nacionalidad***
* ***\_docIdentidad***
* ***\_nombre***
* ***\_email (heredado de Usuario)***
* ***\_password (heredado de Usuario)***

### 4.8.2 Validaciones:

* ***ValidarNombre()***: No puede estar vacío ni tener solo espacios.
* ***ValidarNacionalidad()***: Idem de arriba.
* ***ValidarDocumentoDeIdentidad()***: Debe tener entre 7 y 8 dígitos, y contener solo números.
* Las validaciones de Usuario también se aplican (***email y password***).

## 4.9 Clase Ocasional (hereda de Pasajero)

### 4.9.1Atributos:

* ***\_elegible***: Determina si puede convertirse en premium, basado en un número aleatorio, tal y como solicita la letra al mencionar que la elegibilidad de los clientes ocasionales se determina de forma aleatoria al momento de ingresar al sistema.

### 4.9.2 Lógica específica de esta clase:

* ***CalcularElegibilidad()***: Genera un número aleatorio entre 0 y 100, y si es menor o igual a 70, el pasajero es considerado elegible.

## 4.10 Clase Premium (hereda de Pasajero)

### 4.10.1 Atributo:

* ***\_puntos:*** Representa los puntos acumulados del pasajero.

# 5. MÉTODOS AUXILIARES

Los llamamos así porque tienen el objetivo de facilitar la interacción con el usuario, validar datos ingresados y mejorar la reutilización del código como hemos venido dando en clase. Si bien estos métodos no forman parte directamente de las funcionalidades principales que fueron mencionadas ut-supra; son fundamentales para que se garantice un correcto funcionamiento del sistema y se ejecute de forma segura.

## 5.1 SolicitarInt(string mensaje, int maximo, int minimo)

**Ubicación: *Program.cs***

**Propósito:**Este método lo hemos venido utilizando bastante a lo largo del curso. En este caso, lo usamos para solicitar un número entero al usuario, asegurándonos de que esté dentro de un rango válido determinado por ***minimo y maximo***. Se utiliza principalmente en el menú principal que se muestra en consola para que el usuario seleccione una opción válida.

**Flujo de validación:**

* Muestra un mensaje al usuario.
* Intenta parsear el valor ingresado como entero.
* Si no es válido (no es un número o está fuera de rango), vuelve a solicitarlo.

## 5.2 SolicitarDecimal(string mensaje, decimal maximo, decimal minimo)

**Ubicación:** ***Program.cs***

**Propósito:**Este método solicita al usuario un número decimal y lo valida dentro de un rango ya definido. Aunque no es utilizado directamente en las funcionalidades previstas para esta primera parte del obligatorio, fue realizado para una futura parte 2 (por ejemplo, si se desea ingresar manualmente el precio de un pasaje o el costo de operación).

## 5.3. SolicitarDateTime(string textoFecha)

**Ubicación:** ***Program.cs***

**Propósito:**Nos permite ingresar fechas de manera segura, asegurando que se respete el formato ***dd-MM-yyyy***. El mismo se utiliza en el método **ListarPasajesSegunRangoDeFechas** del menú (listar pasajes entre dos fechas).

**Flujo:**

* Solicita una fecha en un formato correcto.
* Utiliza ***DateTime.ParseExact*** para convertir el texto ingresado a una fecha, el ***ParseExact*** es más rígido a la hora de evaluar un dato, tiene que ser el formato EXACTO.
* En caso de error, vuelve a solicitar hasta que se ingrese una fecha válida.

## 5.4 ExistePasajeroCon(string unDocIdentidad)

**Ubicación:** ***Sistema.cs***

**Propósito:**El mismo ya fue mencionado con anterioridad, y valida que no se intente registrar dos veces al mismo pasajero con el mismo número de documento. Se utiliza específicamente en el alta de pasajeros ocasionales (**funcionalidad C**) para evitar duplicados.

**Flujo:**

* Recorre la lista ***\_usuarios.***
* Si encuentra un ***Usuario*** que es también ***Pasajero*** y cuyo documento coincide con el ingresado, devuelve ***true*** y se lanza la excepción.
* Caso contrario, devuelve ***false***.

## 5.5 Validar() (en todas las clases de dominio)

**Propósito:** Este patrón está presente en la mayoría de las clases ***(Aeropuerto, Avion, Usuario, Pasajero, Vuelo, Ruta, etc.)***. Lo llamamos en el constructor de cada clase y permite centralizar todas las validaciones específicas en un solo lugar tal y como hemos dado en el curso.

**Ejemplos:**

* ***ValidarFrecuencia()*** en ***Vuelo***: Asegura que la frecuencia ingresada sea un día válido de la semana.
* ***ValidarDocumentoDeIdentidad()*** en ***Pasajero***: Controla el largo, solo números y formato correcto.
* ***ValidarPassword()*** en ***Usuario***: Verifica seguridad (mayúsculas, minúsculas, símbolos y longitud).

## 5.6 CalcularCostoPorAsiento() en Vuelo

**Propósito:** Calcula el costo de cada asiento en un vuelo en base a:

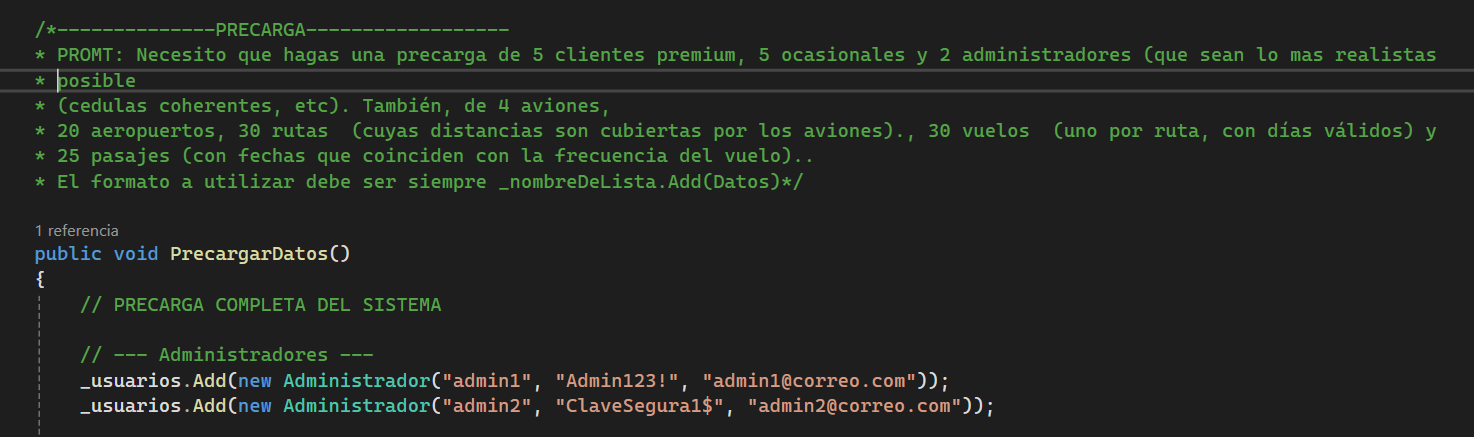
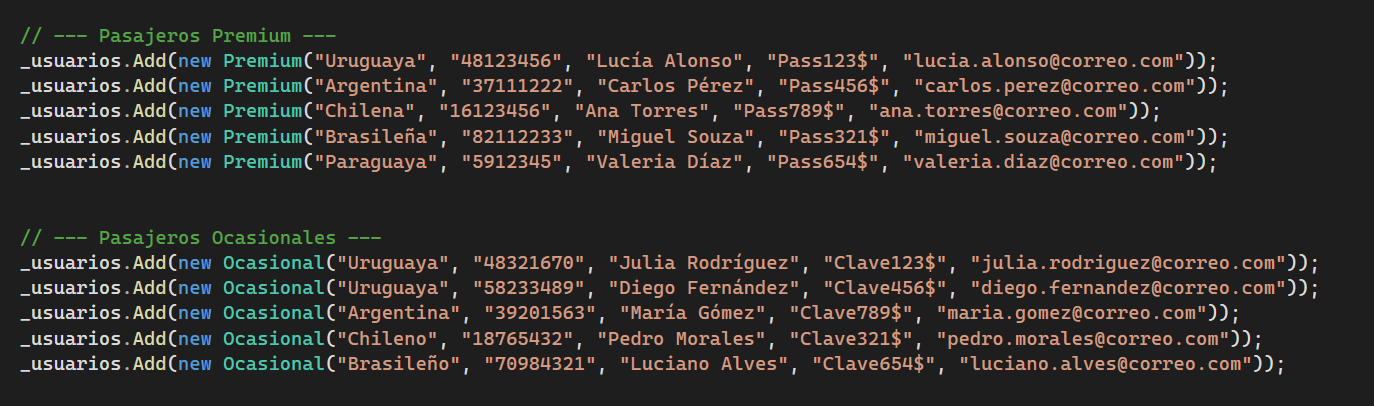
* Costo por kilómetro del avión.
* Distancia de la ruta.
* Costo de operación de ambos aeropuertos.
* Cantidad de asientos del avión.

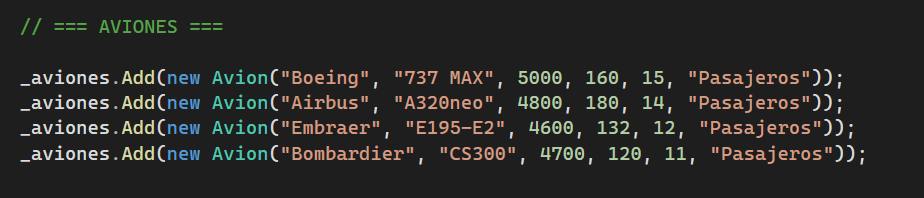
# 6. Precarga de Datos

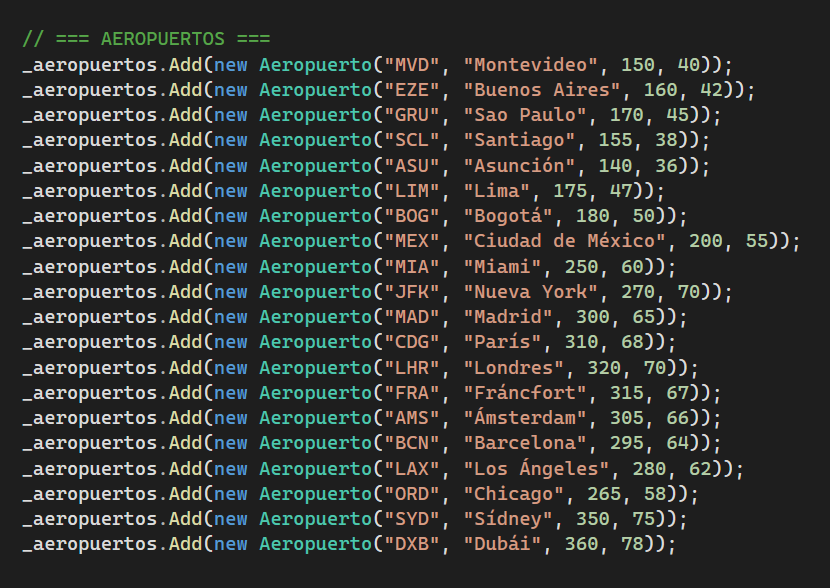
Para facilitar el testeo y uso del sistema, se realiza una precarga en el método ***PrecargarDatos()*** de la clase ***Sistema***. Esta incluye:

* 2 administradores
* 5 pasajeros premium
* 5 pasajeros ocasionales
* 4 aviones
* 20 aeropuertos
* 30 rutas (cuyas distancias son cubiertas por los aviones).
* 30 vuelos (uno por ruta, con días válidos).
* 25 pasajes (con fechas que coinciden con la frecuencia del vuelo).

## 6.1 Datos generados con prompt:

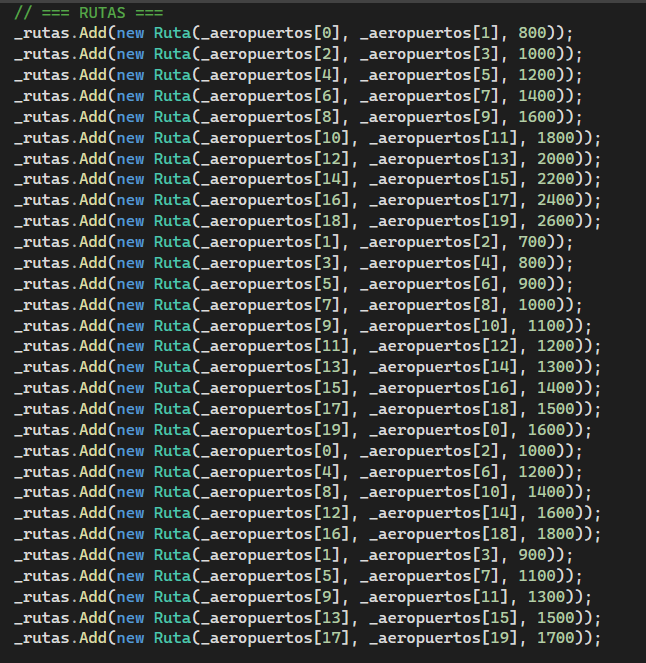
  












## 