

# Práctica 1

## **Nombre proyecto**

Turbina Tour And Resort

## **Asignatura**

Programación Orientada a Objetos

## **Grupo**

01

## **Equipo**

05

## **Integrantes**

Jhoneyker Delgado Urbina  
Emmanuel Valencia Lopera  
Simón Guarín Cortés  
Jesus Camilo Miranda Aguirre  
Andrés Jacobo Leal Aguirre

## **Docente**

Jaime Alberto Guzmán Luna

*Universidad Nacional de Colombia- Sede Medellín*

## **Fecha**

Enero 2025

## **1. Descripción general de la solución.**

El objetivo principal de *Turbina Tour and Resort* es implementar un sistema integral de reservas de vuelos con o sin alojamiento, diseñado para optimizar las operaciones de consulta, compra y gestión de vuelos y alojamientos, brindando una experiencia eficiente y accesible para los administradores. Este sistema busca resolver las dificultades asociadas con la venta de tiquetes, la gestión de aerolíneas y el manejo de alojamientos en una zona aeroportuaria. A través de una interfaz intuitiva y orientada al administrador, se ofrecerán herramientas prácticas que simplifican y agilizan la administración de estas tareas. El programa está diseñado para satisfacer las necesidades específicas establecidas en los requisitos funcionales, garantizando una solución efectiva y acorde a las exigencias del entorno aeroportuario.

### **Requisitos Funcionales**

#### **Búsqueda de Vuelos (Esencial)**

1. Permitir buscar vuelos según los siguientes criterios:
  - Por destino.
  - Por destino y fecha.
  - Por ID de vuelo.

#### **Gestión de Aerolíneas (Esencial)**

2. Permitir la creación de aerolíneas especificando:
  - Nombre de la aerolínea.
3. Permitir la eliminación de aerolíneas especificando:
  - Nombre de la aerolínea.

#### **Gestión de Vuelos (Esencial)**

4. Permitir la creación de vuelos asociados a una aerolínea con los siguientes detalles:
  - Origen.
  - Destino.
  - Distancia.
  - Fecha de salida.
  - Hora de salida.
5. Permitir la cancelación de vuelos asociados a una aerolínea con los siguientes detalles:
  - Nombre de la aerolínea
  - ID del vuelo

#### **Gestión de Alojamientos (Esencial)**

6. Permitir la creación de alojamientos especificando:
  - Nombre propio del alojamiento.
  - Ubicación.
  - Precio por día.
  - Número de estrellas.
7. Permitir la eliminación de alojamientos especificando:
  - Nombre del alojamiento

### **Compra de Tiquetes (Esencial)**

8. Permitir comprar un tiquete a través de los siguientes pasos:
  - Buscar vuelos por destino o por destino y fecha.
  - Seleccionar la aerolínea y el ID del vuelo.
  - Agregar la información del pasajero.
    - Nombre del pasajero.
    - Edad.
    - Número de pasaporte.
    - Correo electrónico.
    - Clase de la silla (económica o ejecutiva).
    - Ubicación de la silla.
9. Agregar Alojamientos al Tiquete:
  - Una vez comprado el tiquete, permitir añadir una reserva de alojamiento, especificando:
    - Destino del vuelo.
    - Tiempo de estadía.

### **Modificación de Tiquetes (Esencial)**

10. Permitir realizar modificaciones a un tiquete ya comprado:
  - Cambiar la silla seleccionada.
    - ID del tiquete.
    - Ubicación de la silla.
    - Clase de la silla (económica o ejecutiva).
  - Cambiar el alojamiento (si el tiquete ya tiene uno asociado).
    - ID del tiquete.
    - Nombre del alojamiento .
    - Tiempo de estadía.

### **Requisitos no funcionales**

1. Eficiente y accesible.
  - Fácil de usar: por la guía que se le hace al usuario a través de los numerales para cumplir su necesidad dentro del programa.
  - Rápido y óptimo :No sabemos con exactitud cuán rápido es el sistema pero al ser un programa en un ámbito de práctica el tiempo de respuesta es óptimo y eficiente ya que en sus funcionalidades no requiere de procesos largos.
2. Confiable
  - Proteger datos : El encapsulamiento que se le ha realizado al programa hace que los datos no sean accedidos y modificados desde otra funcionalidad u objeto del programa.
  - Reducir errores : El programa siempre sigue una secuencia de pasos para que sea fácil de entender para quien utiliza la aplicación por lo que se reducen los errores de uso de la aplicación y siempre que hay un error se le expresará al usuario de manera adecuada al usuario que esté utilizando el programa de TURBINA TOUR AND RESORT.

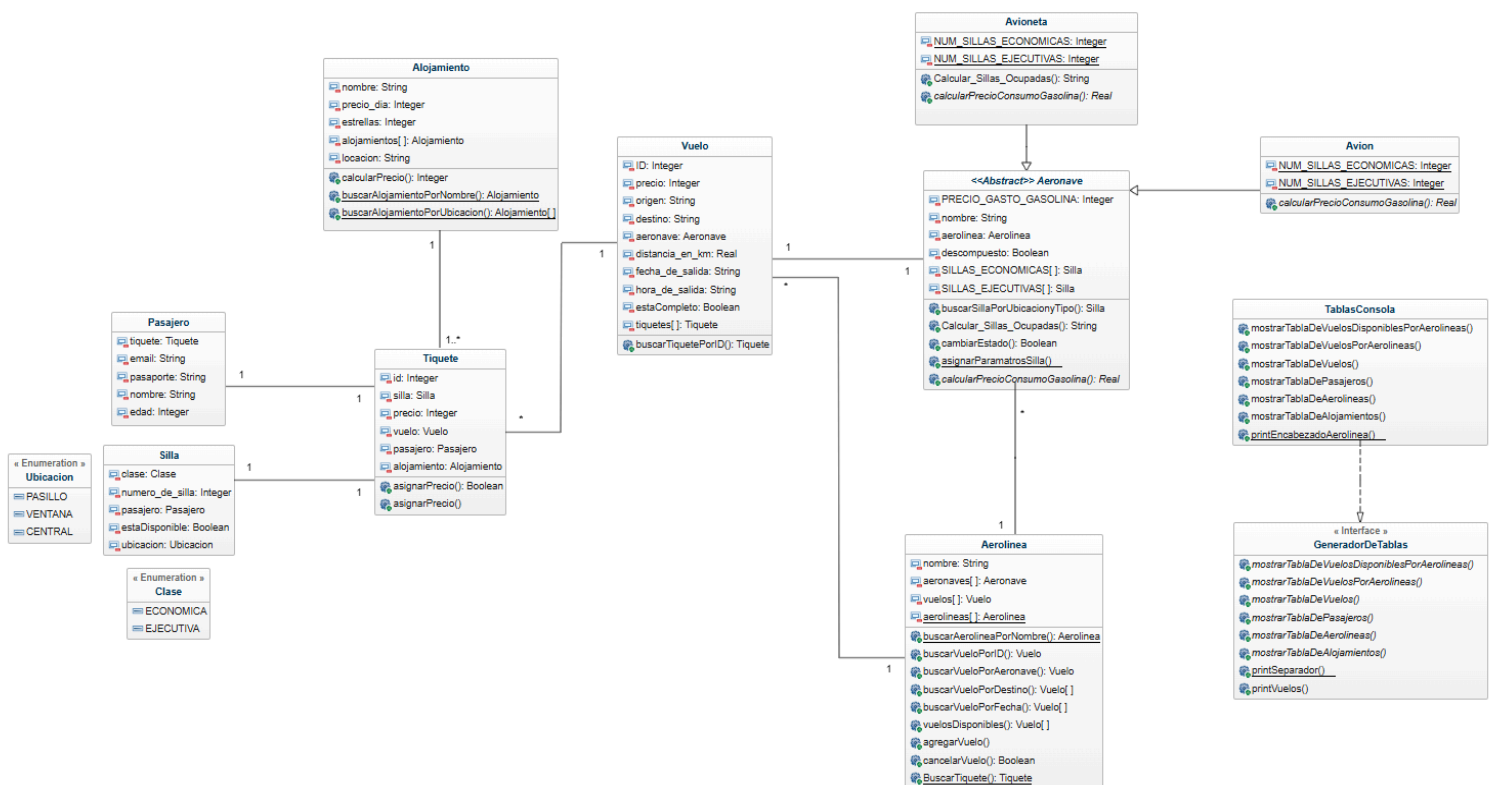
### 3. Usabilidad

- Fácil uso para el usuario final: Todo está especificado en la interfaz para que sea fácil al usuario que necesidad desea cumplir con la aplicación. La especificación está dada por medio de numerales, descripción de cada funcionalidad y precisión en los requerimientos que debe dar el usuario.

### 4. Mantenibilidad

- Adaptable a actualizaciones o correcciones. El programa fue diseñado para poder hacer más funcionalidades e implementaciones según las necesidades y requerimientos de este sector del transporte aéreo. El código cumple con las características de la POO por lo que la reutilización de código se hace esencial.
- Documentación técnica adecuada: Está propia memoria escrita que se está presentando hace parte de la documentación técnica ,que especifica cómo funciona el programa, que soluciona y como se realizan las implementaciones

## 2. Descripción del diseño estático del sistema en la especificación UML.



Link

del

Diagrama:

[https://app.genmymodel.com/api/projects/\\_CatGkM1CEe-0LfWRuUx-PQ/diagrams/\\_CatGk81CEe-0LfWRuUx-PQ/svg](https://app.genmymodel.com/api/projects/_CatGkM1CEe-0LfWRuUx-PQ/diagrams/_CatGk81CEe-0LfWRuUx-PQ/svg)

### **Clase Aerolínea**

Esta clase tiene el rol de almacenar las aerolíneas con sus respectivos vuelos y aeronaves y la información de cada uno de estos, además se agregan nuevos vuelos a las aerolíneas o se cancelan y se tiene la posibilidad de hacer una búsqueda de los vuelos (dependiendo de la Aerolínea) y búsqueda según el nombre de la Aerolínea.

### **Clase Pasajero**

Esta clase tiene el rol de asignar un Pasajero a un Tiquete junto con los datos de este, tales como: el nombre, edad, tiquete, email, pasaporte.

### **Clase Tiquete**

Esta clase tiene el rol de crear un tiquete para un respectivo Pasajero y contener un alojamiento (En caso de que el pasajero lo requiera). También tiene el trabajo de asignar valor a cada tiquete.

### **Clase Vuelo**

Esta clase tiene el rol de crear los vuelos junto con una lista de tiquetes asociada a cada vuelo, además de también asignársele a cada uno de estos una aeronave. También, se puede hacer la búsqueda de un tiquete según el ID en la lista de tiquetes respectiva del vuelo.

### **Clase Alojamiento**

Esta clase tiene el rol de ser el molde para crear alojamientos, además se almacenan los alojamientos. Aquí se calcula el precio por cada alojamiento y también se buscan alojamientos existentes según la ubicación o por nombre.

### **Clase Aeronave**

Esta clase es abstracta, tiene el rol de definir los datos principales de cualquier tipo de aeronave. Además se realizan procesos para saber si el vuelo que realiza esta aeronave ya está completo, también se calculan que cantidad de sillas hay ocupadas y se actualiza la ocupación de los puestos.

### **Clase Avión**

Esta clase tiene el rol de ser molde para crear aeronaves de tipo avión, se tienen establecidas unas capacidades de sillas ejecutivas y económicas. Al heredar de la clase Aeronave aplica todo lo mencionado anteriormente. Se calcula el precio según el gasto de gasolina para que en otro proceso fuera de la clase se sume al precio del tiquete.

### **Clase Avioneta**

Esta clase tiene el rol de ser molde para crear aeronaves de tipo avioneta, se tienen establecidas unas capacidades de sillas ejecutivas y económicas. Al heredar de la clase Aeronave aplica todo lo mencionado anteriormente. Se calcula el precio según el gasto de gasolina para que en otro proceso fuera de la clase se sume al precio del tiquete y además se calcula de otra manera la cantidad de sillas ocupadas.

### **Clase Enum "Clase"**

Esta clase tiene el rol de almacenar los tipos de clase que tiene un aeronave y en los cuales pueden escoger los pasajeros. En este caso están los enumerados: Ejecutiva y económica.

### **Clase Silla**

La clase Silla modela las características de una silla dentro de una aeronave. Almacena información como el número de silla, su clase (económica o ejecutiva), el pasajero asignado, su disponibilidad y su ubicación.

### **Clase Enum “Ubicación”**

Esta clase tiene el rol de almacenar los tipos de ubicación que tiene una silla de una aeronave y en los cuales pueden escoger los pasajeros. En este caso están los enumerados: Pasillo, ventana y central.

### **Interfaz GeneradorDeTablas**

Esta interfaz contiene métodos solo definidos para mostrar tablas al usuario al momento de realizar consultar o pedir algún requerimiento en la aplicación.

### **Clase TablasConsola**

Esta clase implementa los métodos descritos anteriormente y se estructura como deben mostrarse las tablas según los requerimientos del usuario que usa la aplicación.

### **Relaciones**

#### **Aerolínea — Vuelo**

Una aerolínea tiene 0 o muchos vuelos.

#### **Aerolínea — *Aeronave***

Una aerolínea tiene 0 o muchas aeronaves.

#### **Vuelo — Tiquete**

Un vuelo tiene muchos tiquetes.

#### **Vuelo — *Aeronave***

Un vuelo tiene una sola aeronave.

#### **Tiquete — Alojamiento**

Uno o muchos tiquetes tienen un solo alojamiento.

#### **Tiquete — Pasajero**

Un tiquete se relaciona con solo un pasajero.

#### **Tiquete — Silla**

Un tiquete se relaciona con una sola silla.

#### **Avion—> *Aeronave***

Avión hereda de Aeronave.

#### **Avioneta —> *Aeronave***

Avioneta hereda de Aeronave.

### 3. Descripción de la implementación de características de programación orientada a objetos en el proyecto.

#### Clases Abstractas

```
public abstract class Aeronave implements Serializable{  
    private static final long serialVersionUID = 1L;  
    protected final int PRECIO_GASTO_GASOLINA = 10000;  
    private String nombre;  
    private Aerolinea aerolinea;  
    private boolean descompuesto;  
    private Silla[] SILLAS_ECONOMICAS;  
    private Silla[] SILLAS_EJECUTIVAS;
```

Package gestorAplicación.hangar, clase Aeronave.

Esta clase abstracta nos permite la definición de una plantilla para las subclases, permitiendo la reutilización del código y flexibilidad de definir el método según el contexto de la subclase, en este caso la clase abstracta es Aeronave la cual es heredada por las subclases Avion y Avioneta permitiendo la reutilización (sobrescritura) del código con sus atributos y métodos comunes para las subclases.

#### Métodos Abstractos

```
public abstract double calcularPrecioConsumoGasolina(double distancia_en_km);
```

Package gestorAplicación.hangar, clase Aeronave , Línea 166

Este método abstracto que pertenece a la clase abstracta Aeronave indica que debe de ser utilizado por las subclases y como resultado del método retorna un dato de tipo double, para el código este método se definirá en las subclases para calcular el costo según la distancia del vuelo y su autonomía en COP, para posteriormente utilizarlo para darle el precio total al vuelo; así permitiendo reutilización del código y eficiencia, para un mayor entendimiento y velocidad.

#### Interfaces

```
public interface GeneradorDeTablas{  
    public abstract void mostrarTablaDeVuelosDisponiblesPorAerolineas(ArrayList<Aerolinea> aerolineas);  
    public abstract void mostrarTablaDeVuelosPorAerolineas(ArrayList<Aerolinea> aerolineas);  
    public abstract void mostrarTablaDeVuelos(Aerolinea aerolineas, ArrayList<Vuelo> vuelos);  
    public abstract void mostrarTablaDePasajeros(ArrayList<Tiquete> tiquetes);  
    public abstract void mostrarTablaDeAerolineas(ArrayList<Aerolinea> aerolineas);  
    public abstract void mostrarTablaDeAlojamientos(ArrayList<Alojamiento> alojamientos);
```

Package uiMain, interfaz generador de tablas.

Esta interfaz nos facilita la colección de métodos abstractos (sin definirlos) que una clase puede implementar, declarando así una estructura estándar para mostrar datos en forma de tablas donde cada clase que implemente la interfaz implementa los datos específicos para utilizarlo a su contexto, permitiendo una mejor flexibilidad en la

implementación de código, además de una mayor consistencia al momento de su reutilización.

## Métodos por default

```
public default void printVuelos(ArrayList<Vuelo> vuelos)
{
    for (int j = 0; j < vuelos.size(); j++)
    {
        System.out.format(format:"%5s %12s %13s %13s %15s %11s %21s", vuelos.get(j).getID(), vuelos.get(j).getPrecio(), vuelos.get(j).getOrigen(), vuelos.get(j).getDestino(), vuelos.get(j).getFechaSalida(), vuelos.get(j).getAeronave());
    }
}
```

### Package uiMain ,clase generador de tablas línea 27

Este método nos permite la impresión de los vuelos en una estructura definida de manera tabular, donde recibe el tamaño de la lista de vuelos (obteniendo así la cantidad de vuelos) y por cada vuelo imprimirá ID, precio, origen, destino, fecha de salida, hora de salida y aeronave. Y todo esto estando en una interfaz (en una interfaz sólo se pueden declarar no definir métodos), esto lo logramos gracias a la palabra default, impidiendo así la rotura del código, y facilitando la reutilización de código.

## Método estático

```
static void printSeparador()
{
    System.out.println(x:"-----");
}
```

### Package uiMain ,clase generador de tablas línea 19.

Este método no pertenece a instancias individuales sino a la clase en sí misma, tiene como propósito proporcionar una funcionalidad común para imprimir un separador visual en las tablas, que pueda ser utilizado sin necesidad de instanciar ninguna clase que implemente esta interfaz (es decir por todas las instancias que su clase implemente en el generador de tablas estarán dirigidas a este método y cada vez que se llame a este método por medio del nombre de la clase imprimirá el separador), permitiendo así una mayor legibilidad y reutilización del código.

## Herencia

```
public class Avion extends Aeronave {
    private final static int NUM_SILLAS_ECONOMICAS = 24;
    private final static int NUM_SILLAS_EJECUTIVAS = 12;
    private static final long serialVersionUID = 5930658813769177257L;
```

### Package gestorAplicación.hangar,clase Avión.

La herencia es aplicada en la clase Avion con la super clase Aeronave de la cual hereda el nombre de la aeronave, la aerolínea en la cual esta registrada, si la aeronave está descompuesta, y los atributos de sillitas economicas y ejecutivas, que más tardes serán sobrescritos para el contexto de la clase, además de definir el método abstracto calcularPrecioConsumoGasolina declarado en Aeronave (Aeronave es una clase abstracta), dando orden a las instrucciones y reutilizando código.



## Ligadura dinámica

```
public String toString() {  
    return this.nombre;  
}
```

Package gestorAplicación.hangar, clase Aeronave línea 38.

El primer método pertenece a la clase padre aerolínea y tiene como objetivo proporcionar una representación textual básica del objeto. En este caso, retorna el nombre de la aerolínea que está almacenado en el atributo nombre.

```
@Override  
public String toString() {  
    return this.getNombre() + "_A";  
}
```

Package gestorAplicación.hangar, clase Avion línea 21.

En el segundo método estamos sobrescribiendo el método heredado de la clase padre (Aerolínea), que a su vez lo obtuvo de la clase (Object). El método sobrescrito retorna el nombre del objeto (almacenado en el atributo nombre), concatenado con un sufijo (\_A). Para acceder al atributo nombre, utilizamos el método get nombre, que es el **getter** correspondiente. Esto asegura un encapsulamiento adecuado y un acceso controlado al atributo.

```
public String Calcular_Sillas_Ocupadas() {  
    int cont = 0;  
    for (Silla i : this.getSILLASECONOMICAS()) {  
        if (i.isEstado()) {  
            cont += 1;  
        }  
    }  
    for (Silla j : this.getSILLASEJECUTIVAS()) {  
        if (j.isEstado()) {  
            cont += 1;  
        }  
    }  
    return "Esta es la cantidad de silla ocupadas:" + cont;  
}
```

Package gestorAplicación.hangar clase Aeronave línea 65.

Este método, que no recibe parámetros y es de tipo String, se encarga de recorrer los arreglos que contienen las sillas ejecutivas y sillas económicas de la aeronave que lo invoque. Su objetivo principal es verificar la cantidad de sillas ocupadas en ambos arreglos y retornar dicha cantidad como parte de un mensaje en formato de texto.

```

public String Calcular_Sillas_Ocupadas() {
    int cont = 0;
    for (Silla i : this.getSILLASECONOMICAS()) {
        if (i.isEstado()) {
            cont += 1;
        }
    }
    for (Silla j : this.getSILLASEJECUTIVAS()) {
        if (j.isEstado()) {
            cont += 1;
        }
    }
    return "Sillas ocupadas en la avioneta"+cont;
}

```

Package gestorAplicación.hangar clase Avioneta línea 32.

En el segundo método estamos sobrescribiendo el método heredado de la clase padre (Aeronave). El método sobrescrito recorrerá los arreglos de sillas ejecutivas y económicas de la avioneta, luego verifica la cantidad que están ocupadas. Posteriormente, retorna un mensaje con la cantidad total de sillas ocupadas.

### **Ligadura estática**

```

Aeronave avion = new Avion(nombreAvion, Aerolinea.buscarAerolineaPorNombre(nombreAerolinea));
avion.asignarParamatrosSilla(avion, 1);

```

Package uiMain clase Admin línea 482.

La Ligadura estática permite que el método estático asignarParamatrosSillas se resuelva durante la compilación, este método recibe como parámetros (una instancia de tipo Aeronave y un parámetro entero). La ligadura estática se da en este caso, por la llamada al método que se realiza desde una instancia de la subclase (Avion) que está referenciada por un apuntador de tipo Aeronave. Método el cual está definido como estático asignarParamatrosSillas; y no tendría que buscar en la subclase una versión sobrescrita debido al tipo del apuntador (Aeronave). Y esto ayuda a reducir el espacio de memoria y da una mayor eficiencia en la interacción.

### **Atributos de clase**

```

private static ArrayList<Aerolinea> aerolíneas = new ArrayList<Aerolinea>();

```

Package gestorAplicación.adminVuelos clase Aerolinea línea 18.

Esto indica que cada instancia de la clase Aerolínea tendrá el atributo común aerolíneas que para este caso es un lista, que sin importar de la instancia que la accese retornara los mismos valores de la lista; según la lógica del programa cada que se cree

una instancia de esta clase se añadirá a la lista común dando un mejor orden y un mejor manejo de los atributos.

```
private static ArrayList<Alojamiento> alojamientos = new ArrayList<Alojamiento>();
```

Package gestorAplicación.alojamiento clase Alojamiento línea 17.

Esto indica que cada instancia de la clase Alojamiento tendrá el atributo común alojamientos que para este caso es una lista, que sin importar de la instancia que la accese retornara los mismos valores de la lista; según la lógica del programa cada que se cree una instancia de esta clase se añadirá a la lista común dando un mejor orden y un mejor manejo de los atributos.

## Métodos de clase

```
public static Aerolinea buscarAerolineaPorNombre(String nombre2)
{
    Aerolinea retorno = null;
    for (int i = 0; i < Aerolinea.getAerolineas().size(); i++)
    {
        if( Aerolinea.getAerolineas().get(i).getNombre().equalsIgnoreCase(nombre2))
        {
            retorno= Aerolinea.getAerolineas().get(i);
        }
    }
    return retorno;
}
```

Package gestorAplicación.adminVuelos clase Aerolinea línea 39.

Este método al ser estático puede ser llamado sin crear necesariamente una instancia, éste recibe como parámetro un String el cual corresponde a la aerolínea que se quiere buscar facilitando la reutilización, además de dar un encapsulamiento de la lógica y claridad al código.

## Uso de constantes

```
protected final int PRECIO_GASTO_GASOLINA = 10000;
```

Package gestorAplicacion.hangar clase Aeronave línea 13.

El uso de constantes indica que este será un atributo que no podrá ser cambiado más adelante en el programa; el valor definido para este caso es un entero (10000) y su nivel de accesibilidad será protected lo cual indica que es accesible por la misma clase, subclases y cualquier clase que pertenezca al mismo paquete, manteniendo así una mejor legibilidad y coherencia evitando errores en el código.

## Encapsulamiento

```
public String nombre;
```

Package gestor Aplicación.adminVuelos clase Pasajero línea 11.

Define un atributo de tipo String llamado nombre , permite asignar un nombre al pasajero. Su visibilidad en este caso es pública lo que significa que es accesible directamente desde cualquier parte del programa.

```
protected final int PRECIO_GASTO_GASOLINA = 10000;
```

Package gestorAplicacion.hangar clase Aeronave línea 13

Define un atributo de tipo entero llamado precio gasto gasolina. En este caso a la variable se le está asignando un valor el cual es fijo. Su visibilidad en este caso es protected lo que significa que es accesible por la misma clase, subclases y cualquier clase que pertenezca al mismo paquete.

```
private String nombre;
```

Package gestorAplicacion.adminVuelos clase Aerolínea línea 15.

Para este caso se tiene un atributo nombre que almacenará datos de tipo String (cadena de texto), con un nivel de accesibilidad privado lo que indica que solo se puede acceder desde la misma clase y se trata de acceder directamente desde fuera de la clase misma en este caso Aerolínea sacará error.

## Sobrecarga de métodos

```
public boolean asignarPrecio() {
    boolean hayDescuento = false;
    int precio_total = vuelo.getPrecio() + this.getSilla().getClase().getPrecio() + (int)(this.getVuelo().getAeronave().calcula
    if (pasajero.getEdad() < 5) {
        hayDescuento = true;
        this.precio = (int) (precio_total - (precio_total * 0.25));
    } else if (pasajero.getEdad() > 5 && pasajero.getEdad() ≤ 10) {
        this.precio = (int) (precio_total - (precio_total * 0.15));
        hayDescuento = true;
    } else {
        this.precio = precio_total;
    }
    return hayDescuento;
}

public void asignarPrecio(int num_dias) {
    int precio_total = vuelo.getPrecio() + alojamiento.calcularPrecio(num_dias) + this.getSilla().getClase().getPrecio() + (int)
    if (pasajero.getEdad() < 5) {
        this.precio = (int) (precio_total - (precio_total * 0.25));
    } else if (pasajero.getEdad() > 5 && pasajero.getEdad() ≤ 10) {
        this.precio = (int) (precio_total - (precio_total * 0.15));
    } else {
        this.precio = precio_total;
    }
}
```

Package gestorAplicacion.adminVuelos clase Tiquete líneas [48-61] y [66-75].

El primer método no recibe parámetros y tiene un retorno de tipo boolean . Calcula el precio del ticket considerando el costo base del vuelo, el costo de la silla, el consumo de gasolina del avión y aplica descuentos dependiendo de la edad del pasajero. El segundo método no retorna nada y recibe un valor de tipo entero. Calcula el precio del ticket de manera similar al método anterior, pero incluyendo el costo de un alojamiento. Este costo depende del número de días que desee pasar la persona.. La sobrecarga de los métodos asignar precio en este caso sirve para proporcionar dos formas diferentes de calcular y asignar el precio de un ticket, dependiendo de los datos y condiciones disponibles en el momento.

### Sobrecarga de constructores

```
public Tiquete(int id, int precio, Vuelo vuelo) {
    this.id = id;
    this.precio = precio;
    this.vuelo = vuelo;
    vuelo.getTiquetes().add(this);
}

public Tiquete(int id, int precio, Vuelo vuelo, Silla silla, Pasajero pasajero, Alojamiento alojamiento) {
    this.id = id;
    this.precio = precio;
    this.vuelo = vuelo;
    this.silla = silla;
    this.pasajero = pasajero;
    this.alojamiento = alojamiento;
    vuelo.getTiquetes().add(this);
}
```

Package gestorAplicacion.adminvuelos clase Tiquete líneas [25-30] y [ 32-40].

El primer constructor recibe como parámetros un id, un precio y un vuelo además de agregar el ticket a la lista de tickets del vuelo; el segundo constructor además de la información recibida en el primer constructor también recibe como parámetros una silla asignada en el vuelo, el pasajero asociado y su respectivo alojamiento. La sobrecarga de constructores en este caso sirve para proporcionar diferentes formas de instanciar la clase ticket, dependiendo de cuánta información o detalles están disponibles o necesarios en el momento de crear un objeto de tipo ticket.

```
public Aerolinea(String nombre, ArrayList<Vuelo> vuelo){
    this(nombre);
    setVuelos(vuelo);
}

public Aerolinea(String nombre) {
    this.nombre = nombre;
    aerolineas.add(this);
}
```

Package gestorAplicacion.adminvuelos clase Aerolinea líneas [21-25] y [26-29].

El primer constructor recibe como parámetros un nombre y un arreglo de tipo vuelo que estará asociado a la aerolínea, también hace referencia al segundo constructor de la clase que recibe como parámetro un nombre para asignarlo. Finalmente se llama al método que se encarga de asignar la lista de vuelos recibida como argumento al

objeto actual; el segundo constructor recibe como parámetro el nombre que se desea asignar y agrega el objeto de la aerolínea creada a una lista de aerolíneas. La sobrecarga de constructores en este caso sirve para proporcionar diferentes formas de instanciar la clase aerolínea, dependiendo de cuánta información o detalles están disponibles al momento de crear un objeto de esta clase..

### Manejo de referencia this

```
public Aerolinea(String nombre) {  
    this.nombre = nombre;  
    aerolineas.add(this);  
}
```

Package gestorAplicacion.adminvuelos clase Aerolínea línea 28.

Primero utilizamos this para evitar cualquier ambigüedad entre los nombres del parámetro y el atributo de clase. De esta manera, Java sabe que te refieres al atributo perteneciente al objeto actual y no a la variable local o parámetro recibido en el método o constructor. Esto asegura que el valor recibido en el parámetro sea asignado correctamente al atributo de la instancia. Finalmente lo utilizamos para hacer referencia al objeto creado y añadirlo a la lista de aerolíneas.

```
private Pasajero(Tiquete tiquete){  
    tiquete.setPasajero(this);  
}
```

Package gestor Aplicación.adminVuelos clase Pasajero línea 26.

En el constructor privado de la clase pasajero utilizamos this para referirnos al objeto actual. En este caso el propósito de this es pasar la instancia actual del objeto pasajero como argumento al método set pasajero del objeto tiquete.

### Manejo de referencia this()

```
public Aerolinea(String nombre, ArrayList<Vuelo> vuelo){  
    this(nombre);
```

Package gestorAplicacion.adminvuelos clase Aerolínea línea 22.

Hace referencia a el constructor de la clase Aerolínea que recibe como parámetro un nombre para luego asignarlo y así continuar con las instrucciones dentro del constructor, esto nos permite la reutilización del código dando una mejor eficiencia y entendimiento al momento de la lectura del código

```
public Pasajero(String pasaporte, String nombre, Tiquete tiquete, int edad, String email) {  
    this(tiquete);
```

Package gestorAplicacion.adminvuelos clase Pasajero linea 18.

Hace referencia a el constructor de la clase Pasajero que recibe como parámetro un tiquete para ser la primera parte en ejecución y así continuar con las instrucciones dentro del constructor, esto nos permite la reutilización del código dando una mejor eficiencia y entendimiento al momento de la lectura del código

### Caso de enumeración

```
public enum Clase {  
    ECONOMICA(precio:10000),EJECUTIVA(precio:70000);  
    private int precio;  
    private Clase(int precio) {  
        this.precio = precio;  
    }  
    public int getPrecio() {  
        return this.precio;  
    }  
    public void setPrecio(int precio) {  
        this.precio =precio;  
    }  
}
```

Package gestorAplicacion.hangar clase Clase

Este enum permite predefinir el tipo de sillas de la aeronave y asocia un valor para cada tipo de silla y es de gran importancia porque la información contenida en este no podrá ser modificada durante la ejecución además de facilitar la obtención del precio de las sillas.

## 4. Descripción de cada una de las 5 funcionalidades implementadas.

### Funcionalidad #1 Ver todos los vuelos disponibles por aerolínea

#### Descripción

Esta funcionalidad le permitirá al usuario ver todos los vuelos disponibles operados por cada una de las diferentes aerolíneas.

Se realizará una búsqueda global, sin especificación de fecha ni horario de todos los vuelos disponibles. El sistema validará los vuelos por medio de una secuencia de pasos excluyendo aquellos que se encuentren llenos para no almacenarlos en la lista de vuelos disponibles. Finalmente el sistema mostrará la información principal de cada uno de ellos organizada en una tabla, con información como:

ID | Precio | Origen | Destino | Fecha | Hora de salida | Nombre de la aeronave.

Cada una de esas columnas haciendo referencia a:

- **ID del vuelo:** Identificador único que permite distinguir cada vuelo.
- **Precio:** Costo del boleto correspondiente al vuelo.
- **Origen:** Ciudad desde donde parte el vuelo.
- **Destino:** Ciudad al cual llega el vuelo.
- **Fecha:** Día programado para el vuelo.
- **Hora de salida:** Hora exacta de despegue del vuelo.
- **Nombre de la aeronave:** Identificación del avión asignado.

Esta funcionalidad está diseñada para facilitar la consulta de opciones de vuelo disponibles, proporcionando una visión clara y organizada al usuario.



## Diagrama de interacción

1. Obtener la lista de aerolíneas disponibles utilizando:  
``ArrayList<Aerolinea> aerolineasDisponibles = Aerolinea.getAerolineas();``

2. Iterar sobre cada aerolínea y obtener su lista de vuelos:  
`java  
for (Aerolinea i : aerolineasDisponibles) {  
 List<Vuelo> vuelos = i.getVuelos();  
}`

3. Preparar los datos para enviarlos a la siguiente interacción. Retorno: aerolineasDisponibles con los vuelos asociados.



1. Recibir la lista de aerolíneas y sus vuelos desde la primera interacción.

2. Para cada vuelo, verificar y actualizar su estado basándose en la aeronave:  
`java  
for (Vuelo j : aerolinea.getVuelos()) {  
 j.setEstaCompleto(j.getAeronave().cambiarEstado());  
}`

3. Retornar los datos con los estados de los vuelos actualizados. Retorno: aerolineasDisponibles con los vuelos actualizados.



1. Recibir la lista de aerolíneas y vuelos actualizados desde la segunda interacción.

2. Usar el generador de tablas para presentar los vuelos disponibles:  
`java  
generadorDeTablas.mostrarTablaDeVuelosDisponiblesPorAerolineas(aerolineasDisponibles);`

3. Mostrar la tabla de vuelos al usuario. Resultado final: Tabla con vuelos disponibles clasificados por aerolínea.

## **Funcionalidad #2 Comprar tiquetes para un vuelo por destino y fecha.**

### **Descripción**

En esta funcionalidad por medio de la interfaz, se le brindará al usuario la opción de comprar un tiquete, este podrá elegir si buscar un tiquete basado en el destino al cual desea viajar o proporcionando el destino y la fecha en la que quiere realizar su viaje. Según la información ingresada por el usuario, se mostrará una lista de los viajes que cumplen con sus criterios de búsqueda junto con la información relevante de cada uno, este deberá ingresar el nombre de la aerolínea por la cual desea viajar seguido del ID del vuelo de su elección, también deberá especificar el tipo de silla que desea (económica o ejecutiva ) y según el tipo la ubicación, si es ejecutiva venta o pasillo o de ser económica ventana, central o pasillo. Luego se le pedirá ingresar sus datos personales para proceder con la creación del tiquete de vuelo.

Cada tiquete se imprimirá con los siguientes detalles :

- **Número del tiquete:** Único para cada pasaje.
- **Nombre del pasajero:** Corresponde al primer nombre y primer apellido del pasajero.
- **Fecha:** Corresponde a la fecha en que se realizará el vuelo.
- **Número del vuelo :** Este es el ID del vuelo el cual es único de cada uno.
- **Número de la silla y la ubicación :** Hace referencia al número de asiento que se le asignó para dicho vuelo junto con la ubicación.
- **Origen :** Lugar donde saldrá el vuelo.
- **Destino :** Lugar al que viajará.
- **Precio total:** Valor final que tendrá el pasaje teniendo en cuenta la clase del asiento y la distancia del vuelo.

## Diagrama de interacción

1. Obtener todas las aerolíneas disponibles:  
``Aerolinea.getAerolineas()``

2. Para cada vuelo, actualizar su estado verificando el estado de la aeronave:

```
java
for (Vuelo j : i.getVuelos()) {

j.setEstaCompleto(j.getAeronave().cambiarEstado());
}
```

3. Preparar la información de vuelos actualizada para la interacción con el usuario. Retorno: Lista de vuelos con sus estados actualizados.

1. Solicitar al usuario el tipo de búsqueda de vuelos:

- 1: Buscar por destino.
- 2: Buscar por destino y fecha.
- 3: Regresar.

2. Ejecutar la búsqueda según la opción elegida:

- Por destino: ``consultarVuelosPorDestino(destino)``
- Por destino y fecha: ``consultarVuelosPorDestinoYFecha(destino, fecha)``

3. Validar que el usuario elija una aerolínea válida:

```
java
Aerolinea aerolinea =
Aerolinea.buscarAerolineaPorNombre(nombre_aerolinea);
```

4. Validar que el usuario seleccione un vuelo válido por ID. Retorno: Vuelo seleccionado por el usuario.

1. Generar un ID aleatorio para el ticket y verificar que no esté duplicado:

```
java
double ID_ticket = 100 + Math.random() * 900;
```

2. Solicitar al usuario el tipo de silla y asignarla al vuelo:

```
java
Silla silla = elegirSilla(vuelo);
```

3. Solicitar los datos personales del pasajero: Nombre, edad, pasaporte, correo electrónico.

4. Crear el objeto Pasajero y asignarlo al ticket generado.

5. Asignar el precio al ticket, imprimir y mostrar un resumen de la compra al usuario. Retorno: Resumen del ticket generado con los datos del pasajero.

### **Funcionalidad #3 Agregar alojamiento en el destino del vuelo comprado**

#### **Descripción**

Esta funcionalidad permitirá agregar un alojamiento a un ticket de vuelo ya comprado y registrado a nombre de un usuario, para ello se le preguntará el id del ticket que se generó al momento de comprar el vuelo, según esta información se le mostrará al usuario los alojamientos disponibles para el lugar del destino marcado en el el ticket, se le solicitará ingresar el alojamiento que desea incluir en su ticket y los días que se quedara en el alojamiento. Finalmente se actualizará el precio del ticket , es decir al valor generado inicialmente se le sumará el valor de la estadía para dicho alojamiento.

El ticket con su alojamiento agregado se imprimirán en un tabla , especificando la información ahí consignada de la siguiente manera :

-----  
**Ticket No:** Número del ticket único por pasajero

-----  
**Nombre Pasajero:** Corresponde al primer nombre y primer apellido del pasajero.

**Fecha:** Corresponde a la fecha en que se realizará el vuelo.

**Vuelo:** Este es el ID del vuelo el cual es único de cada uno.

**Silla:** Hace referencia al número de asiento que se le asignó para dicho vuelo.

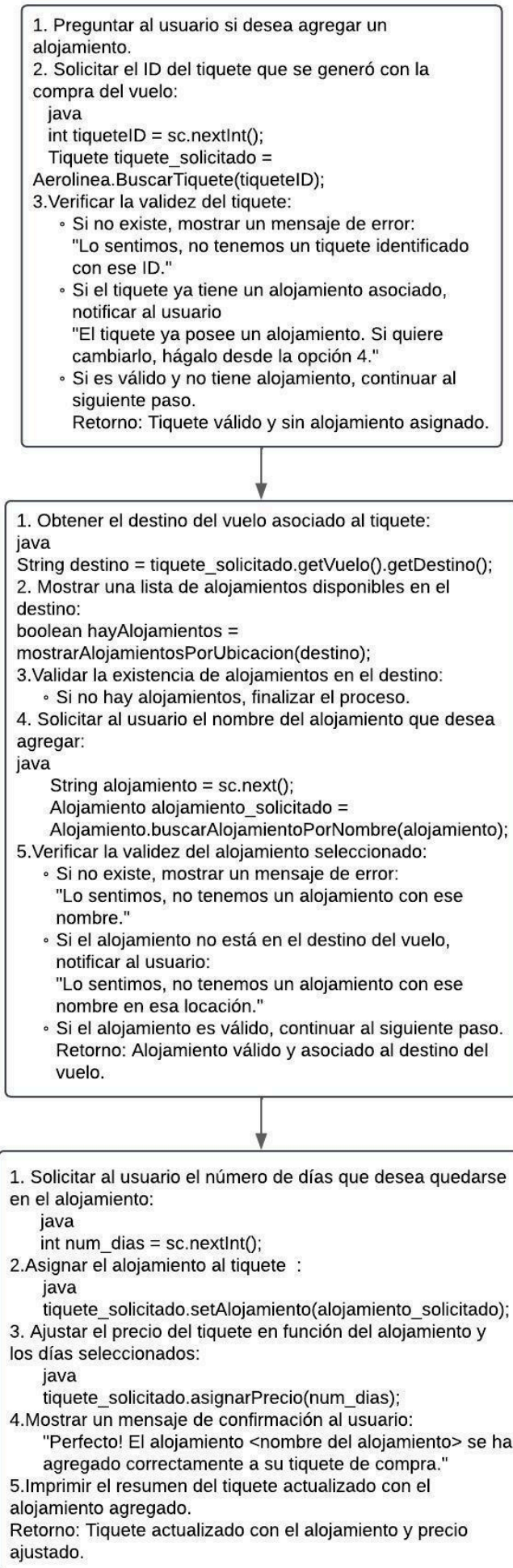
**Origen:** Lugar donde saldrá el vuelo.

**Destino:** Lugar al que viajará.

**Alojamiento:** Este es el que eligió según su destino.

**Precio Total:** Precio del vuelo más alojamiento.  
-----

## Diagrama de interacción



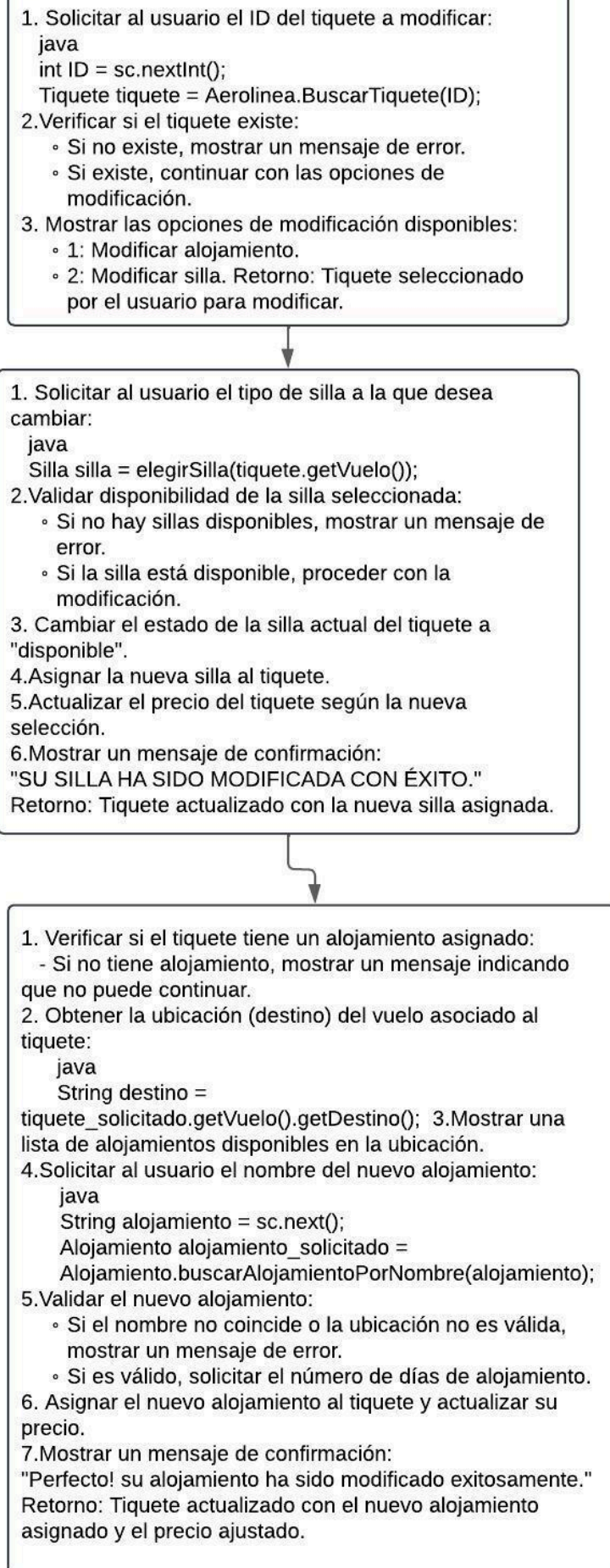
## **Funcionalidad #4. Modificar tiquete comprado**

### **Descripción**

Esta funcionalidad permitirá modificar el alojamiento y la silla de un tiquete previamente comprado (Una modificación a la vez), en el caso de querer modificar la silla se le preguntará a qué tipo de silla desea cambiar (económica o ejecutiva), luego deberá ingresar la ubicación de asiento en que desea ir en caso de no haber del tipo seleccionado se le informará antes de continuar. La información actualizada será agregada en el tiquete para así modificarlo y cambiar su precio.

Si desea modificar el alojamiento se verificará que tenga uno asociado al tiquete, en dado caso que el tiquete no presente un alojamiento asociado se le informará al usuario que podrá agregarle uno en la opción 3 del menú principal (funcionalidad 3). Si el tiquete tiene un alojamiento asociado se podrá cambiar el alojamiento por uno diferente, para ello deberá ingresar el nombre del alojamiento según el destino además de la cantidad de días que desea hospedarse. Para finalizar se realizará un cálculo para ajustar el valor del tiquete según el nuevo alojamiento y la cantidad de días. Este cálculo determinará si se le debe asignar un cargo adicional o restarle al precio total del tiquete.

## Diagrama de interacción





## Funcionalidad #5 Ver opciones de administrador

### Descripción

Esta funcionalidad le mostrará por pantalla al usuario en este caso al administrador un menú de opciones, que tienen como finalidad permitir una mejor gestión del sistema, entre estas opciones podemos destacar:

**Listar pasajeros:** Para esta opción se tendrá un manejo de datos de todos los pasajeros, según el vuelo y aerolínea al que están asociados para mostrarlos a través de la interfaz .

**Agregar Vuelo:** En esta opción del menú se le brindará la opción al administrador de agregar un vuelo. Conforme a las aerolíneas ya existentes, se podrá seleccionar una de ellas para agregarle un vuelo con la información pertinente. (ID, precio, origen, destino, distancia, fecha de salida, hora de salida, tipo de aeronave y nombre del avión).

**Cancelar un vuelo:** Esta opción permitirá la cancelación de un vuelo. Según las aerolíneas y vuelos creados, para ello debe ingresar la aerolínea a la que pertenece dicho vuelo y el ID del vuelo que desea eliminar.

**Retirar un avión:** Por medio de esta opción el usuario podrá retirar una aeronave. Se le solicitará que ingrese el nombre de la aeronave que desea retirar (previamente se le informo todas las aeronaves existentes por aerolíneas). Luego se realizará la búsqueda de esta y se le cambiara el estado del avión a está “descompuesto” y se eliminarán todos los vuelos asociados a dicha aeronave.

**Agregar alojamiento:** Ingresado el nombre del alojamiento, la locación, precio por día, número de estrellas podría agregar un alojamiento para que luego pueda ser asociado a cualquier tiquete.

**Eliminar alojamiento:** Se le mostrará los alojamientos disponibles y se le pedirá el nombre del alojamiento que desea retirar, lo buscará en la lista de alojamientos y será eliminado por completo.

**Agregar aerolínea:** Se le solicitará un nombre para la aerolínea que desea crear, al hacerlo se añadirá inmediatamente, además se le preguntará si desea agregar vuelos a alguna aerolínea de las previamente ya creadas y podrá agregar solo uno.

**Eliminar aerolínea:** Se deberá ingresar el nombre de la aerolínea a retirar, y la buscará entre las ya existentes y será eliminada junto con ella, los vuelos que hagan parte de esta.

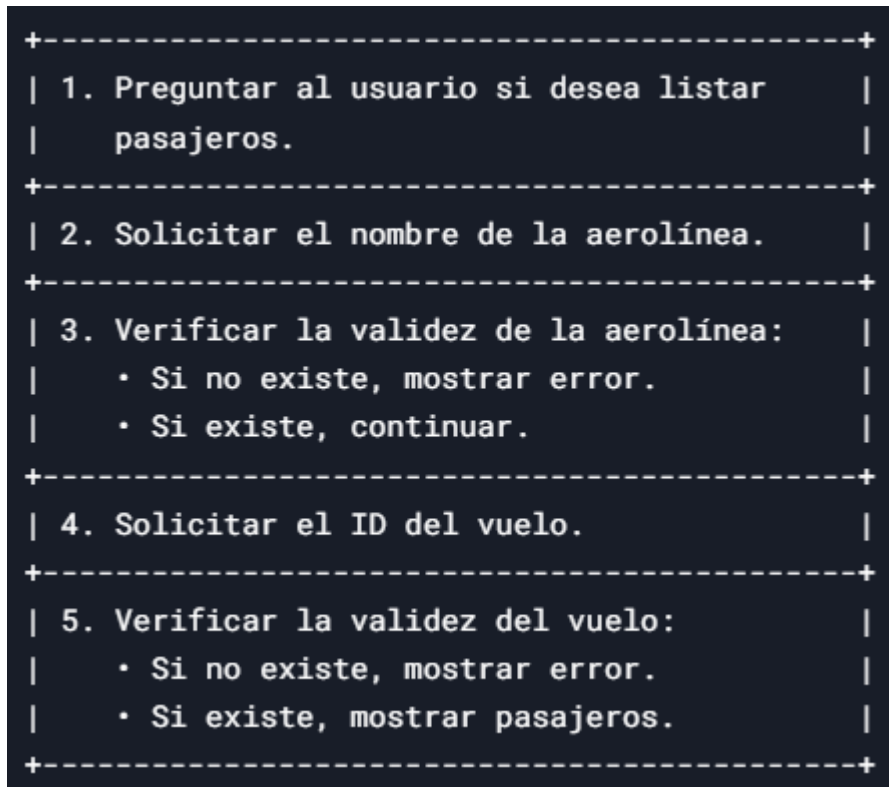
**Agregar aerolínea con vuelos:** Brindará la opción de agregar una aerolínea con vuelos (La cantidad que desee) para ello se le preguntará el nombre que le desea asignar a la aerolínea. El sistema verificará si la aerolínea existe, si la aerolínea existe le pedirá que ingrese otro nombre, en caso de que no la aerolínea deberá tener al menos un vuelo así que se le pedirá que ingrese un id (número de tres cifras). Posterior a ello deberá asignar un precio, establecer destino y origen en que operará el vuelo. También deberá agregar una distancia, fecha y hora de salida, además de especificar si la aerolínea será de tipo avión o avioneta, luego pedirá que asigne un nombre para este objeto. Finalmente el sistema mostrará un mensaje haciendo referencia a que la aerolínea ha sido registrada correctamente, luego le preguntará si desea agregar otro vuelo para dicha aerolínea y así hasta que no sea necesario agregar otro vuelo.



**Salir del administrador:** En esta opción de administrador, se imprimirá: "Gracias por usar nuestras opciones de administrador! y volverá al menú principal.

### Diagrama de interacción

#### Listar Pasajeros



## Agregar Vuelo

```
+-----+
| 1. Preguntar al usuario si desea agregar un vuelo. |
+-----+
| 2. Solicitar el nombre de la aerolínea. |
+-----+
| 3. Verificar la validez de la aerolínea: |
|   • Si no existe, mostrar error. |
|   • Si existe, continuar. |
+-----+
| 4. Solicitar datos del vuelo: |
|   - ID, precio, origen, destino, distancia, |
|   fecha y hora de salida. |
+-----+
| 5. Seleccionar tipo de aeronave (avión o |
|   avioneta). |
+-----+
| 6. Registrar vuelo y mostrar confirmación. |
+-----+
```

## Cancelar un vuelo

```
+-----+
| 1. Preguntar al usuario si desea cancelar un vuelo. |
+-----+
| 2. Solicitar el nombre de la aerolínea. |
+-----+
| 3. Verificar la validez de la aerolínea: |
|   • Si no existe, mostrar error. |
|   • Si existe, continuar. |
+-----+
| 4. Solicitar el ID del vuelo a cancelar. |
+-----+
| 5. Verificar la validez del vuelo: |
|   • Si no existe, mostrar error. |
|   • Si existe, cancelar vuelo. |
+-----+
```

### Retirar Avión

```
+-----+
| 1. Preguntar al usuario si desea retirar |
|      un avión.                          |
+-----+
| 2. Solicitar el nombre de la aeronave.   |
+-----+
| 3. Verificar la validez de la aeronave:  |
|      • Si no existe, mostrar error.      |
|      • Si existe, retirar aeronave y cancelar |
|          vuelo asociado.                 |
+-----+
```

### Agregar Alojamiento

```
+-----+
| 1. Preguntar al usuario si desea agregar |
|      un alojamiento.                    |
+-----+
| 2. Solicitar datos del alojamiento:      |
|      - Nombre, ubicación, precio por día, |
|          número de estrellas.            |
+-----+
| 3. Crear objeto Alojamiento.             |
+-----+
| 4. Mostrar confirmación de agregado.     |
+-----+
```

## Eliminar Alojamiento

```
+-----+
| 1. Preguntar al usuario si desea eliminar |
|      un alojamiento.                      |
+-----+
| 2. Mostrar todos los alojamientos disponibles. |
+-----+
| 3. Solicitar nombre del alojamiento.          |
+-----+
| 4. Verificar la validez del alojamiento:      |
|      • Si no existe, mostrar error.          |
|      • Si existe, eliminarlo.               |
+-----+
```

## Agregar Aerolínea

```
+-----+
| 1. Preguntar al usuario si desea agregar   |
|      una aerolínea.                        |
+-----+
| 2. Solicitar nombre de la aerolínea.        |
+-----+
| 3. Verificar si la aerolínea ya existe.     |
|      • Si existe, mostrar error.           |
|      • Si no existe, crear aerolínea.       |
+-----+
| 4. Preguntar si desea agregar vuelos.       |
|      • Si sí, llamar a agregarNuevoVuelo(). |
|      • Si no, finalizar.                   |
+-----+
```

## Eliminar Aerolínea

```
+-----+
| 1. Preguntar al usuario si desea agregar |
|      una aerolínea.                      |
+-----+
| 2. Solicitar nombre de la aerolínea.     |
+-----+
| 3. Verificar si la aerolínea ya existe.  |
|      • Si existe, mostrar error.         |
|      • Si no existe, crear aerolínea.    |
+-----+
| 4. Preguntar si desea agregar vuelos.    |
|      • Si sí, llamar a agregarNuevoVuelo(). |
|      • Si no, finalizar.                 |
+-----+
```

## Agregar Aerolínea con Vuelos

```
+-----+
| 1. Preguntar al usuario si desea agregar |
|      una aerolínea con vuelos.           |
+-----+
| 2. Solicitar nombre de la aerolínea.     |
+-----+
| 3. Verificar si la aerolínea ya existe.  |
|      • Si existe, mostrar error.         |
|      • Si no existe, continuar.         |
+-----+
| 4. Solicitar datos del vuelo:            |
|      - ID, precio, origen, destino, distancia, |
|      fecha y hora de salida.             |
+-----+
| 5. Seleccionar tipo de aeronave (avión o  |
|      avioneta).                          |
+-----+
| 6. Preguntar si desea agregar más vuelos. |
|      • Si sí, repetir desde el paso 4.   |
|      • Si no, finalizar.                 |
+-----+
```

## Salir del administrador

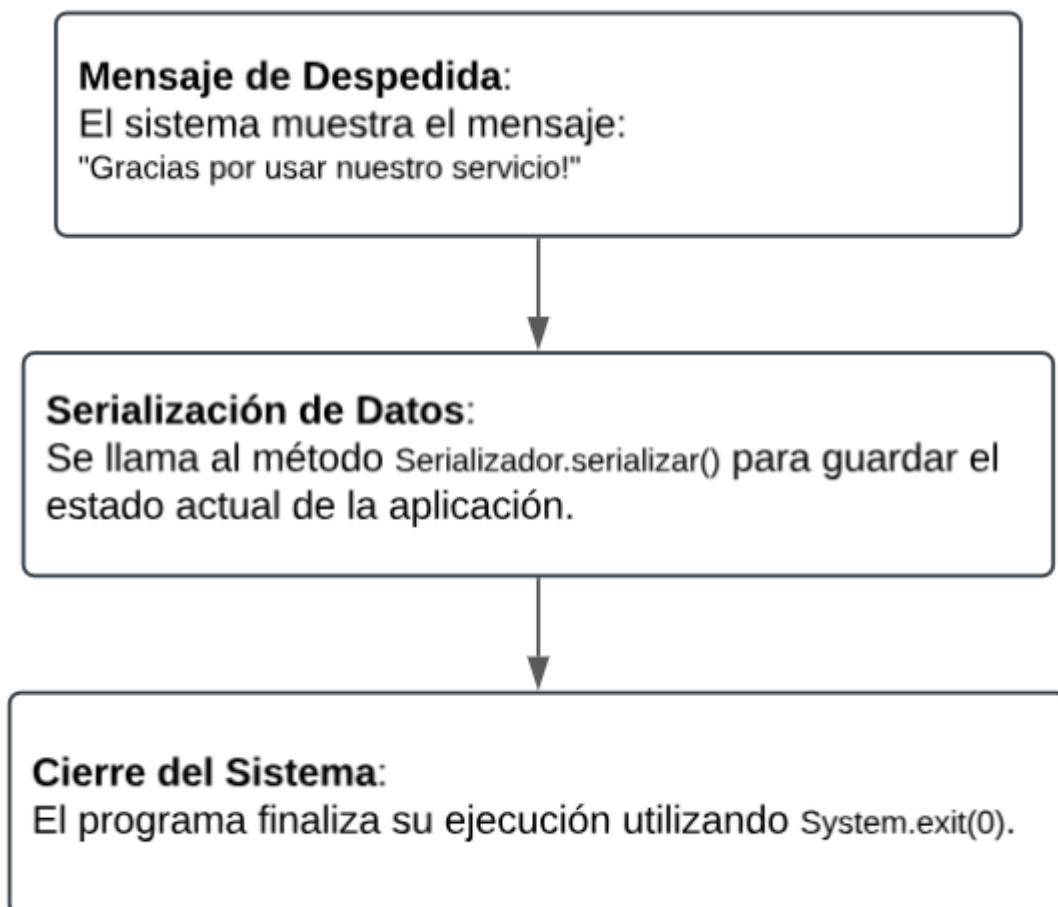
```
+-----+  
| 1. Mostrar mensaje de despedida. |  
+-----+
```

## Funcionalidad #6 Terminar

### Descripción

Esta funcionalidad es simple y directa, nos permitirá el cierre completo del programa pero lo más importante es que invoca al serializador lo cual nos permite guardar toda la información realizada durante la ejecución del programa en un archivo .txt en la ruta destinada en el archivo src/baseDatos/temp.

### Diagrama de interacción



## 5. Manual de usuario.

El menú principal de la aplicación es de la siguiente manera:

```
----- SISTEMA DE RESERVAS DE VUELO -----  
Que operacion desea realizar?  
1. Ver todos los vuelos disponibles por Aerolinea  
2. Comprar ticket para un vuelo por destino y fecha  
3. Agregar alojamiento en el destino del vuelo  
4. Modificar ticket comprado  
5. Ver opciones de administrador  
6. Terminar  
Por favor escoja una opcion:  
█
```

Usted podrá elegir la opción que esté interesado en ejecutar escribiendo el número que la acompaña seguido de la tecla “enter”, si la aplicación le solicita en algún momento ingresar texto, no afecta al funcionamiento del programa si usted escribe en mayúscula o en minúscula por lo que no deberá preocuparse por esto. Le pedimos que si desea ingresar algún texto compuesto de dos o más palabras, separe las palabras con un guión bajo “\_” para el buen funcionamiento del programa.

En caso de que haya elegido la opción 1 (Ver todos los vuelos disponibles por Aerolínea) usted podrá ver la lista de los vuelos disponibles de cada una de las aerolíneas con los siguientes datos: ID, PRECIO, ORIGEN, DESTINO, FECHA, HORA DE SALIDA y AERONAVE. Seguido a esto usted volverá al menú principal.

En caso de que haya elegido la opción 2 (Comprar ticket para un vuelo por destino y fecha) usted verá la siguiente interfaz:

```
Quieres buscar un vuelo por:  
1. Destino  
2. Destino y fecha  
3. Regresar  
█
```

Si desea buscar su vuelo por destino debe presionar la tecla “1” seguida de la tecla “enter”. Luego, el programa le pedirá escribir el destino de su interés seguido de la tecla “enter”. En el caso de que no se encuentren vuelos con ese destino, el programa mostrará “Lo sentimos, no tenemos vuelos disponibles para ese destino”.

Si encuentra uno o más vuelos con el destino de su interés, se los mostrará en pantalla organizados por la aerolínea que lo maneja. Luego, se le pedirá que ingrese el nombre

de la aerolínea por la que desea viajar, el id del vuelo, el tipo de silla que puede elegir entre estos dos:

```
A que tipo de silla desea cambiar?  
1: Ejecutiva  
2: Economica
```

Si desea una silla ejecutiva presione la tecla “1” o la tecla “2” si desea una económica y presione “enter” después de cualquiera de los casos. También la ubicación de esta, todos estos datos seguidos cada uno de la tecla “enter”. Cuando usted haya ingresado todos estos datos se pasará a pedirle sus datos personales tales como nombre, edad y correo electrónico, estos también seguidos cada uno de la tecla “enter”. Al finalizar, se mostrará en pantalla que su compra ha sido exitosa seguido de su tiquete con toda la información de su viaje. Luego usted volverá al menú principal.

Si desea buscar su vuelo por destino y fecha deberá presionar la tecla “2” seguida de la tecla “enter”, el programa le pedirá ingresar el destino de su interés, seguido de la fecha en la que desea viajar y luego de ingresar cada uno, presionar la tecla “enter”. Si el programa no encuentra ningún vuelo programado con la información que usted ingresó, se le mostrará “Lo sentimos, no tenemos vuelos disponibles para ese destino y fecha específicos”. En el caso de que el programa encuentre uno o más vuelos con sus especificaciones, se le mostrará una lista de los vuelos disponibles separados por cada una de las aerolíneas y se le pedirá que ingrese la aerolínea con la que desea viajar seguido de datos como el id del vuelo, el tipo de silla que puede elegir entre estos dos:

```
A que tipo de silla desea cambiar?  
1: Ejecutiva  
2: Economica
```

Si desea una silla ejecutiva presione la tecla “1” o la tecla “2” si desea una económica y presione “enter” después de cualquiera de los casos. También se le pedirá la ubicación de la silla, su nombre, edad y correo electrónico, cada uno seguido de la tecla “enter”. Luego se le mostrará un mensaje indicando que su compra ha sido exitosa seguido de su tiquete de vuelo con toda la información de su viaje. Luego usted volverá al menú principal.

Si desea regresar al menú principal deberá presionar la tecla “3” seguida de la tecla “enter”.



En caso de haber elegido la opción 3 (Agregar alojamiento en el destino del vuelo), debe tener presente que para poder utilizar esta opción usted deberá tener un ticket previamente. El programa le pedirá el id de su ticket y se le mostrará una lista con los hoteles disponibles en el destino de su ticket. Usted deberá escribir el nombre del hotel en el cual le interesaría hospedarse, luego se solicitará ingresar el número de días que desea hospedarse en este alojamiento y se le mostrará un mensaje diciendo que su alojamiento ha sido agregado a su ticket seguido de su ticket de viaje actualizado con la información del hospedaje y su nuevo precio. Luego usted volverá al menú principal.

En caso de haber elegido la opción 4 (Modificar ticket comprado), debe tener presente que para hacer uso de esta opción usted deberá tener un ticket previamente. Se le pedirá ingresar el id del ticket que desea modificar, y luego se le mostrará en pantalla lo siguiente:

```
Que aspectos de su ticket desea modificar?  
1: Modificar alojamiento  
2: Modificar Silla  
|
```

Si usted desea modificar el alojamiento debe tener en cuenta que antes debe tener un alojamiento ya enlazado a su ticket, si es así, presione la tecla “1” seguido de la tecla “enter”. Se le mostrará una lista de los hoteles disponibles en su destino y se le pedirá escribir el nombre del hotel en el cual desea hospedarse seguido del número de días que se va a hospedar. Luego se mostrará en pantalla un mensaje diciendo que su alojamiento ha sido modificado exitosamente y podrá ver su ticket de viaje actualizado con la nueva información del alojamiento y su nuevo precio. Luego usted volverá al menú principal.

Si usted desea modificar la silla de su ticket, presione la tecla “2” seguido de la tecla “enter” y se le mostrarán las siguientes opciones.

```
A que tipo de silla desea cambiar?  
1: Ejecutiva  
2: Economica  
|
```

Si desea cambiar a una silla ejecutiva, presione la tecla “1” seguido de la tecla “enter” y se le mostrarán las siguientes opciones:

```
Cual de las siguientes ubicaciones prefiere?  
1: Pasillo  
2: Ventana
```

deberá escribir el número de la opción que le interesa y luego presionar la tecla “enter”, se le mostrará un mensaje diciendo que su asiento ha sido modificado exitosamente seguido de su ticket con la información actualizada. Luego usted volverá al menú principal.

Si desea cambiar a silla económica, presione la tecla “2” seguido de la tecla “enter” y se le mostrarán las siguientes opciones:

```
Cual de las siguientes ubicaciones prefiere?  
1: Pasillo  
2: Ventana  
3: Central
```

deberá escribir el número de la opción que le interesa y luego presionar la tecla “enter”, se le mostrará un mensaje diciendo que su asiento ha sido modificado exitosamente seguido de su ticket con la información actualizada. Luego usted volverá al menú principal.

En caso de haber elegido la opción 5(Ver opciones de administrador) a usted se le mostrará el siguiente menú:

```
Que opcion desea realizar como administrador?  
1. Listar Pasajeros.  
2. Agregar Vuelo.  
3. Cancelar vuelo.  
4. Retirar un avion.  
5. Agregar Alojamiento.  
6. Eliminar Alojamiento.  
7. Agregar Aerolinea  
8. Eliminar Aerolinea.  
9. Agregar una aerolinea con vuelos  
10. Salir del administrador.  
Por favor escoja una opcion:
```

Si usted elige la opción 1 (Listar pasajeros), se le mostrará una lista con todos los vuelos disponibles de cada aerolínea y su información principal usted deberá escribir el número del ID del vuelo de su interés seguido de la tecla “enter”. Si el vuelo no tiene ningún pasajero se mostrará el mensaje “El vuelo aún no tiene pasajeros asociados”, en el caso de que haya uno o más pasajeros en este vuelo se le mostrará una lista con los pasajeros y la información de cada uno. Luego volverá al menú de administrador.

Si usted elige la opción 2 (Agregar vuelo), se le mostrará un listado de las aerolíneas existentes y se le pedirá escribir el nombre de la aerolínea a la cual quiere agregarle un vuelo seguido de la tecla “enter”, luego se le pedirán los siguientes datos: Id del vuelo, precio base, origen, destino, distancia en kilometros, fecha de salida, hora de salida, tipo de aeronave y nombre del avión. Usted deberá ingresar cada uno de los datos que se le solicitan seguido de la tecla “enter”, cuando haya ingresado todos los datos se mostrará en pantalla “Su vuelo se ha registrado correctamente” y volverá al menú de administrador.

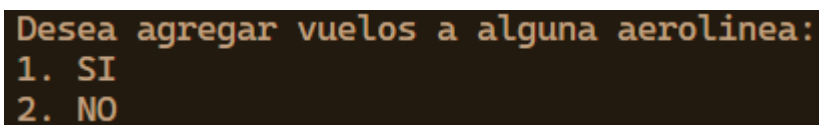
Si usted elige la opción 3 (Cancelar vuelo), se le mostrará una lista de todos los vuelos disponibles de cada aerolínea y se le pedirá que ingrese el nombre de la aerolínea y el id del vuelo que desea ingresar seguidos cada uno de la tecla “enter”, si no se encuentra ningún vuelo con ese id se mostrará el mensaje “No tenemos un vuelo identificado con ese ID”, si se encuentra un vuelo con este id se eliminará el vuelo y se mostrará “El vuelo se ha eliminado correctamente”. Luego volverá al menú de administrador.

Si usted elige la opción 4 (Retirar un avión), se le pedirá escribir el nombre de la aeronave que desea retirar seguido de la tecla “enter”, si no se encuentra ninguna aerolínea con ese nombre se mostrará “Lo sentimos, no encontramos una aeronave asociada al nombre que ingresó”, por el contrario si se encuentra una aerolínea con este nombre, se retirará esta aeronave y sus vuelos asociados y se mostrará “Se ha retirado la aeronave descompuesta y el vuelo asociado a este”. Luego volverá al menú de administrador.

Si usted elige la opción 5 (Agregar un alojamiento), se le pedirá escribir los siguientes datos cada uno seguido de la tecla “enter”: nombre del alojamiento, locación, precio por día y número de estrellas. Si todos los datos están correctos se mostrará el mensaje “Perfecto! El alojamiento hotel se ha agregado a nuestra lista”. Luego volverá al menú de administrador.

Si usted elige la opción 6 (Eliminar alojamiento), se le mostrará una lista con cada uno de nuestros alojamientos asociados con sus datos relevantes. Se le pedirá ingresar el nombre del hotel que desea eliminar, si el nombre coincide con algún hotel se mostrará "El alojamiento hotel se ha eliminado correctamente". Luego volverá al menú de administrador.

Si usted elige la opción 7 (Agregar aerolínea), se le pedirá que escriba el nombre que va a llevar la aerolínea que usted desea agregar, si el nombre es el mismo a alguna aerolínea que haya registrada se mostrará "Ya existe una aerolínea con ese nombre", no se agregará la aerolínea y usted volverá al menú de administrador. En el caso de que no haya ninguna aerolínea con ese nombre, se mostrarán las siguientes opciones:



```
Desea agregar vuelos a alguna aerolínea:  
1. SI  
2. NO
```

Si quiere agregar vuelos a alguna aerolínea presione la tecla "1" seguida de la tecla "enter". El programa ejecutará el proceso de agregar un vuelo explicado previamente como opción 2 del menú de administrador, si tiene dudas puede volver a revisarlo. En caso de que usted no quiera agregar vuelos, presione la tecla "2" seguida de la tecla "enter". Luego usted volverá al menú de administrador.

Si usted elige la opción 8 (Eliminar aerolínea), se le mostrará una lista con las aerolíneas registradas y se le pedirá ingresar el nombre de la aerolínea que desea eliminar seguido de la tecla "enter". Si no se encuentra ninguna aerolínea con el nombre que usted ingresó, se mostrará "No tenemos una aerolínea con este nombre". En el caso de que si se encuentre alguna aerolínea con ese nombre, se eliminará la aerolínea junto con los vuelos de está y se mostrará "Se ha eliminado correctamente la aerolínea {nombre de la aerolínea}". Luego usted volverá al menú principal.

Si usted elige la opción 9 (Agregar una aerolínea con vuelos), se le pedirá escribir el nombre de la aerolínea a agregar seguido de la tecla "enter". Si ya existe una aerolínea con ese nombre, se mostrará "La aerolínea ya existe por favor ingrese otro nombre" y usted deberá ingresar un nombre válido. Si el nombre de la aerolínea que usted ingresó no está registrado, se mostrará "La aerolínea debe tener por lo menos un vuelo" y luego se le pedirán los siguientes datos: Id del vuelo, precio base, origen, destino, distancia en kilometros, fecha de salida, hora de salida, tipo de aeronave y nombre del avión. Usted deberá ingresar cada uno de los datos que se le solicitan seguido de la tecla "enter", cuando haya ingresado todos los datos se mostrará en pantalla "Su vuelo se ha registrado correctamente". Luego se le preguntará si desea agregar otro vuelo a esa aerolínea, si desea agregar otro vuelo deberá presionar la tecla

“1” seguida de la tecla “enter” y deberá hacer el proceso de agregar un vuelo de nuevo. En caso de no querer agregar otro vuelo, presione la tecla “2” seguida de la tecla “enter” y volverá al menú de administrador.

Si usted elige la opción 10 (Salir del administrador), se mostrará “Gracias por usar nuestras opciones de administrador!” y usted volverá al menú principal.

Si en el menú principal elige la opción 6 (Terminar) el programa mostrará “Gracias por usar nuestro servicio!”, se guardarán todos los datos registrados y el programa se cerrará.