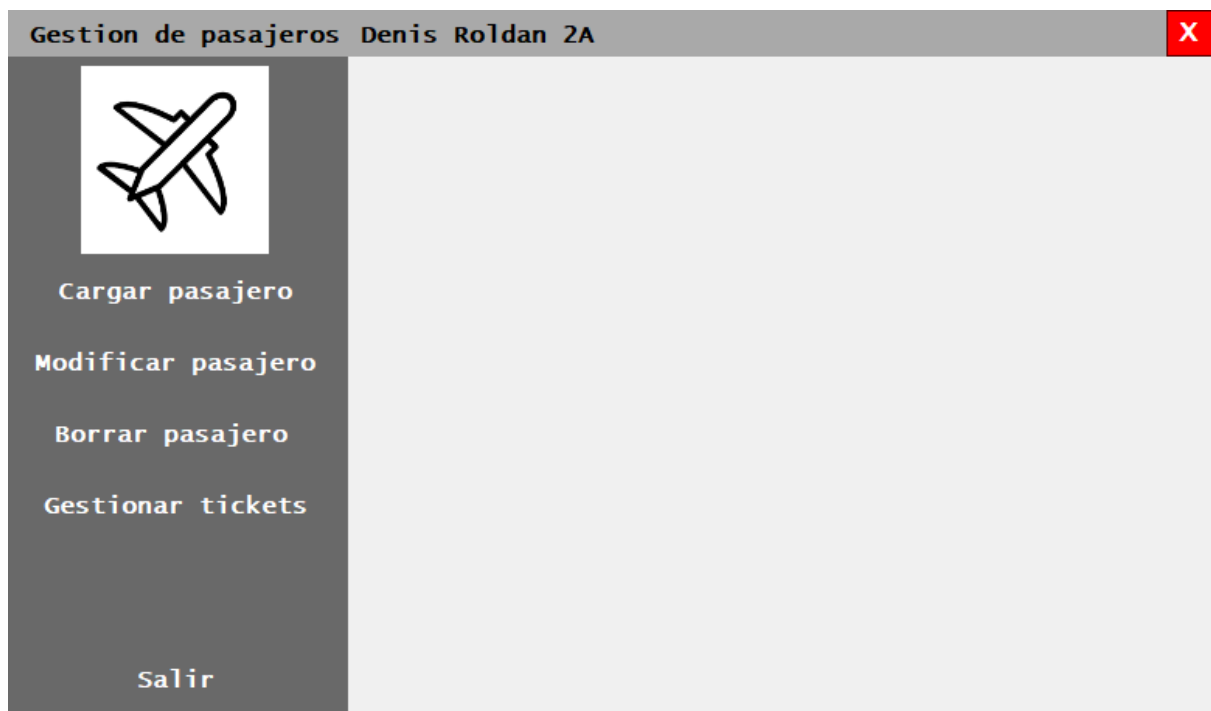


TRABAJO PRÁCTICO N° 4:

El gestor de pasajeros simula ser una aplicación para los empleados de una aerolínea la cual gestiona la carga , baja y modificación de los pasajeros. Los que estén a cargo del manejo de la app podrán cargar pasajeros en los 4 vuelos que tiene a disposición la empresa y podrá modificar los datos del pasajero en caso de que haya alguna equivocación o eliminarla en caso de que cancele su pasaje.

1- Ventana de inicio:



La ventana de inicio sirve de bienvenida a la aplicación , en donde se podrá hacer uso de las 4 opciones que tiene incorporadas. Internamente funciona con carga de archivos donde se cargaran los datos que hayan sido guardados previamente.

```
public Inicio()
{
    InitializeComponent();

    try
    {
        List<Avion> aviones;
        if (!new Json<List<Avion>>().Read(@"\Vuelos.json", out aviones))
        {
            MessageBox.Show("Hubo un error en la carga de datos");
        }
        else
        {
            Vuelos.vuelos = aviones;
        }
    }
    catch (Exception ex)
    {
        new Text().Save("logError.txt", LogErrors.LogError(ex, "form Inicio"));
        MessageBox.Show("Error fatal , por favor comunicarse con el área de sistemas", "ATENCIÓN!", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

2- Ventana de carga:

Al hacer click en “Cargar pasajero” se abre un nuevo formulario con la información de los aviones: Su destino y los asientos disponibles de cada avión:



Cargar pasajero

Modificar pasajero

Borrar pasajero

Gestionar tickets

Salir

Vuelo: 1
Destino: Cordoba
Asientos: 1

Reservar lugar

Vuelo: 2
Destino: Aeroparque
Asientos: 3

Reservar lugar

Vuelo: 3
Destino: Mendoza
Asientos: 1

Reservar lugar

Vuelo: 4
Destino: Aeroparque
Asientos: Completo

Reservar lugar

Al hacer click en uno de los botones de “Reservar lugar” se accede a la parte baja de la ventana donde se mostrará mediante botones los lugares disponibles. De no tener lugares disponibles o particularmente el número de asiento buscado esta ocupado el botón queda deshabilitado.



Cargar pasajero

Modificar pasajero

Borrar pasajero

Gestionar tickets

Salir

Vuelo: 1
Destino: Cordoba
Asientos: 1

Reservar lugar

Vuelo: 2
Destino: Aeroparque
Asientos: 3

Reservar lugar

Vuelo: 3
Destino: Mendoza
Asientos: 1

Reservar lugar

Vuelo: 4
Destino: Aeroparque
Asientos: Completo

Reservar lugar

Asiento 1


Asiento 2

Asiento 3

Asiento 4

Atrás

Luego una vez clickeado el asiento disponible se accede a la última instancia del formulario donde se cargan los datos del pasajero y luego de cargarlo , un mensaje anuncia la carga del mismo.



Cargar pasajero

Modificar pasajero

Borrar pasajero

Gestionar tickets

Salir

Vuelo: 1
Destino: Cordoba
Asientos: 1

Vuelo: 2
Destino: Aeroparque
Asientos: 3

Vuelo: 3
Destino: Mendoza
Asientos: 1

Vuelo: 4
Destino: Aeroparque
Asientos: Completo

Asiento 3

Asiento 4

Atrás

Reservar lugar

Reservar lugar

Prueba

Pdf

78945612

Cargar pasajero

Carga correcta !


i

Pasajero Prueba cargado correctamente !
El numero de vuelo es 1

OK

3- Ventana modificar pasajero:

En esta ventana al igual que en la ventana de carga primero se muestran todos los vuelos con su información respectiva. Luego una vez que elegimos en qué avión queremos hacer la modificación mediante el botón que se encuentra abajo del listBox , se accede al datagrid que muestra los pasajeros a bordo. Para poder modificar los pasajeros se tiene que hacer doble click en cualquier lugar de la fila, este evento permite acceder a la última instancia de la ventana que es donde se modifican los datos del pasajero.



Cargar pasajero

Modificar pasajero

Borrar pasajero

Gestionar tickets

Salir

Vuelo: 1
Destino: Cordoba
Asientos: Completo

Vuelo: 2
Destino: Aeroparque
Asientos: 3

Vuelo: 3
Destino: Mendoza
Asientos: 1

Vuelo: 4
Destino: Aeroparque
Asientos: Completo

Modificar vuelo

Modificar vuelo

Modificar vuelo

Modificar vuelo

Nombre	Apellido	Dni	IdAsiento
Prueba	Pdf	78945612	4
Takeshi	Kovacs	12345678	3
Milagros	Patane	38860480	2
Denis	Roldan	38797877	1

Atrás

38797877


Denis

Roldan

Aceptar cambios

4- Ventana modificar pasajero:

Misma mecánica que modifica al pasajero. En esta ventana se debe hacer un solo click sobre la persona a eliminar y luego clicar el botón Eliminar pasajero, saldrá un mensaje preguntando si se quiere hacer y dependiendo la respuesta se elimina o no.



Cargar pasajero

Modificar pasajero

Borrar pasajero

Gestionar tickets

Salir

Vuelo: 1
Destino: Cordoba
Asientos: Completo

Vuelo: 2
Destino: Aeroparque
Asientos: 3

Vuelo: 3
Destino: Mendoza
Asientos: 1

Vuelo: 4
Destino: Aeroparque
Asientos: Completo

Eliminar un pasajero


Eliminar un pasajero

			IdAsiento
			4
			3
			2
Denis	Roldan	38797877	1

Eliminar pasajero

Atrás

ATENCIÓN!



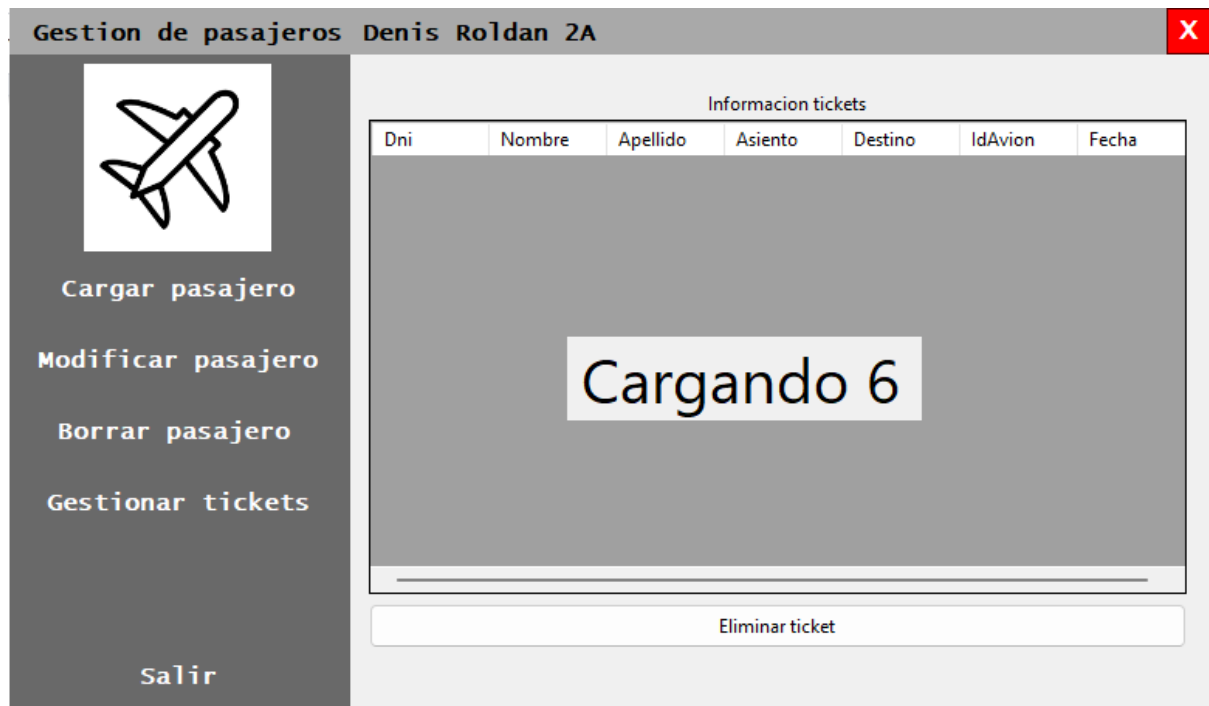
Esta seguro que desea eliminar del vuelo a:
Pasajero: Prueba Pdf
DNI: 78945612
N° Asiento: 4 ?

Yes

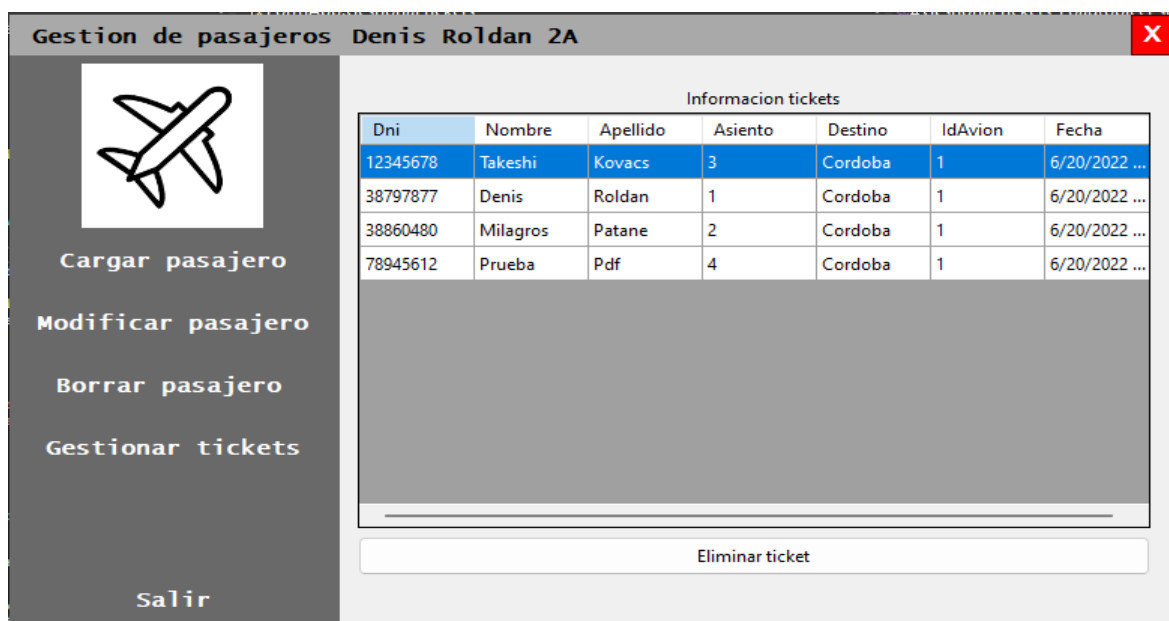
No

5- Ventana gestión de tickets:

La forma que tiene el empleado de ver a todos los pasajeros es mediante la gestión de tickets. Esta información se almacena en una base de datos por lo que al acceder a esta ventana hay una pequeña espera hasta que se carguen todos los tickets. NOTA: los tickets tienen fecha por si hay pasajeros repetidos en el mismo vuelo el empleado puede eliminar los tickets con fecha más antigua.

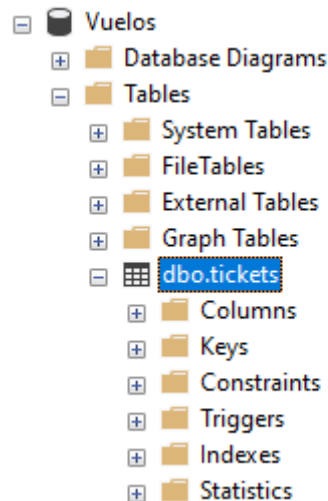


Luego con la misma mecánica de la ventana borrar , con un click se selecciona el ticket del pasajero que se desea eliminar y se clickea "Eliminar ticket"



6 SQL - base de datos:

La base de datos se llama Vuelos y cuenta con la tabla tickets para alojar la información de los pasajeros:



En la raíz del proyecto se encuentra la carpeta “DB” con el script para generar la base y la tabla.

En TicketDAO se encuentra el código necesario para guardar , leer y borrar filas de la tabla:

```
2 references
public void Guardar(Pasajero pasajero, Avion avion)
{
    string destino = "";
    try
    {
        connection.Open();
        string query = "INSERT INTO tickets (dni,nombre,apellido,asiento,destino,avion,fecha) VALUES (@dni,@nombre,@apellido,@asiento,@destino,@avion,@fecha)";

        switch (avion.Destino)
        {
            case Avion.EDestino.Cordoba:
                destino = "Cordoba";
                break;
            case Avion.EDestino.Aeroparque:
                destino = "Aeroparque";
                break;
            case Avion.EDestino.Mendoza:
                destino = "Mendoza";
                break;
        }

        command.CommandText = query;
        command.Parameters.AddWithValue("dni", pasajero.Dni);
        command.Parameters.AddWithValue("nombre", pasajero.Nombre);
        command.Parameters.AddWithValue("apellido", pasajero.Apellido);
        command.Parameters.AddWithValue("asiento", pasajero.IdAsiento);
        command.Parameters.AddWithValue("destino", destino);
        command.Parameters.AddWithValue("avion", avion.Id);
        command.Parameters.AddWithValue("fecha", DateTime.Now);

        command.ExecuteNonQuery();
    }
    catch (Exception ex)
    {
        new Text().Save("logError.txt", LogErrors.LogError(ex, "GeneracionDeTickets"));
    }
    finally
    {
        if (connection is not null && connection.State == System.Data.ConnectionState.Open)
        {
            connection.Close();
        }
    }
}
```

7 - Hilos y eventos:

Hilos se utilizó para que , al momento de leer los datos de la db y traerlos en el datagrid de la gestión de tickets , se simula una espera de 8 segundos. Aquí también se utilizó un evento que muestra la cuenta regresiva de los datos a traer.

```
24 private async void GestionarTickets_Load(object sender, EventArgs e)
25 {
26     try
27     {
28         Evento evento = new Evento();
29
30         label2.Show();
31         evento.tiempoFinalizado += NotificarCargaDeTickets;
32         evento.tiempoRestante += MostrarTiempoRestante;
33         evento.MensajeTiempoFinalizado(8);
34         dgv_Tickets.DataSource = await CargarDataGrid();
35         label2.Hide();
36     }
37     catch (Exception ex)
38     {
39         new Text().Save("LogError.txt", LogErrors.LogError(ex, "GestionTickets"));
40         MessageBox.Show("Hay errores en la carga del formulario", "ATENCIÓN!", MessageBoxButtons.OK, MessageBoxIcon.Error);
41     }
42 }
43
44
45 2 references
46 private async Task<List<Tickets>> CargarDataGrid()
47 {
48     TicketDAO ticketDAO = new TicketDAO();
49     List<Tickets> tickets = await Task.Run(() =>
50     {
51         Thread.Sleep(8000);
52         return ticketDAO.Leer();
53     });
54 }
55
56 return tickets;
57
58 }
```

7 - Delegado:

El delegado se utilizó para ordenar la información del data grid , ordenando según el criterio especificado las listas de pasajeros.

```
private void MostrarListaDePasajeros(int idVuelo)
{
    List<Pasajero> pasajeros = new List<Pasajero>();
    Avion avion = Vuelos.ObtenerAvion(idVuelo);

    ObtenerPasajerosDataGrid(pasajeros, avion);

    pasajeros.Sort(Pasajero.Comparar);
    btn_Atras.Show();
    dgv_Pasajeros.Show();
    dgv_Pasajeros.DataSource = pasajeros;
}
```