Parcial Hotel. Emmanuel Martín.

El programa simula el manejo de registros de un hotel por un usuario para la atención a un huésped a hospedarse en el hotel.

El usuario puede tanto registrar huéspedes, habitaciones, reservaciones. A su vez puede modificar y eliminar datos de los mismos.

Puede también, seguir un registro de las cuentas de las reservaciones de los huéspedes con su registro en JSON y seguir un tracking de la actividad del usuario en el programa.

Adjunto imágenes del programa con explicaciones de funcionamiento. El orden de como ejecutar el programa es indistinto, pero si ciertos datos o funciones podrán realizarse en base a previos registros por el usuario.

Por ejemplo, sin huéspedes registrados, no se podrá generar reservaciones, ni modificar o eliminar estos datos.

1- INICIO DE PROGRAMA

| Tiotel Doll Jose | | | | | | | |
|------------------------|---------------------------------------|--|------------------------------|----------------------|------------------------------------|--------------------------------|-----------------------------------|
| | BIENVENII | DO A SISTEM | A DE RESE | RVA HOTE | L DON JOSI | E | |
| | | | | | | | |
| Generar Reservacion | Registrar Nuevo Huesped al sistema | Registrar Nueva habitacion al sistema | Ver Huespedes Registrados | Ver Reservaciones | Modificar Elimnar Reservaciones | Modificar Elimnar Huespedes | Modificar Elimnar Habitaciones |
| | | | | | | | |
| | | VER INFORM | | | | | |
| o hay nada pai | ro montror | | | | | | |
| , nay naua par | u mostiai | • | | | · · | | |
| | | | | | | | |
| | | | | | | | |
| | | VER TRACKI | | | | | |
| | | USUARI | 0 | | | | |
| | | | | | ~ | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |

En este FORM, el usuario puede elegir tanto como ver huéspedes o reservaciones en el sistema, reflejadas en el DataGrid.

Puede elegir generar una reservación, registrar huéspedes o habitaciones.

Puede modificar o eliminar reservaciones, huéspedes, habitaciones.

A su vez, puede ver los informes en JSON sobre las cuentas de las reservaciones de huéspedes, como el tracking de la actividad de usuario en el programa.

Acá usé tanto tema de Base de datos para las entidades huéspedes, habitaciones y reservaciones. En tema base de datos, utilizo Generic, Interface para intentar implementar el patrón de diseño de repositorio.

Uso delegado y evento para generar el tracking del usuario en el programa.

Método de extensión para ordenar los huéspedes por Apellido. También hay otros órdenes en la clase de extensión, pero se utiliza el mencionado

Use test para verificar validaciones y que se obtengan correctamente huéspedes por su dni desde la base de datos.

Generé excepciones a lo largo del programa, que son capturadas en catch por Exception y lanzan el mensaje respectivo.

NOMBRE BASE DE DATOS:

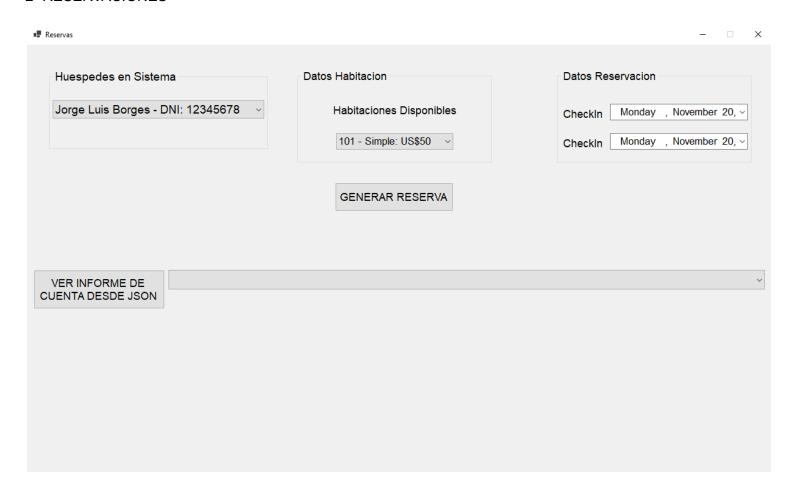
Hotel-Segundo-Parcial-Labo.

Dejé en el repo el backUp de la base datos.

Dejo el path del JSON generado.

string path = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "billings.json");

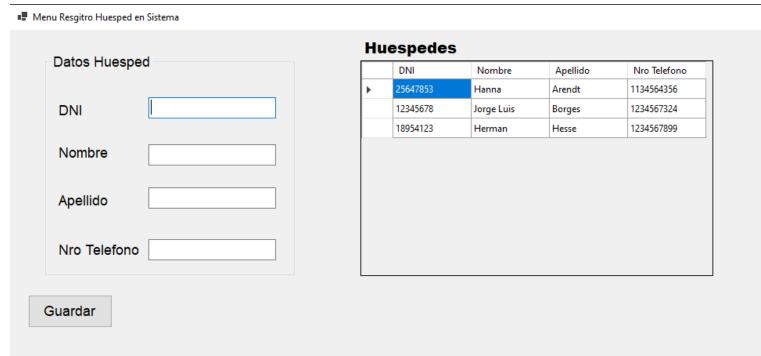
2- RESERVACIONES



En esta sección. Si es que hay usuarios registrados en el sistema y habitaciones disponibles, el usuario puede generar reservaciones, eligiendo la habitación y las fechas para ese huésped. A su vez verá reflejado los datos de la cuenta de las reservaciones hechas.

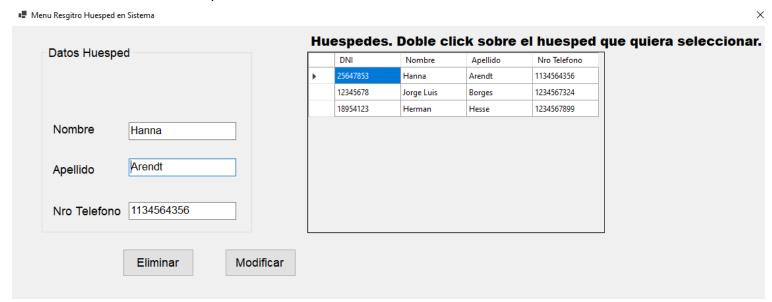
Acá usé Serialización y archivos para generar el informe JSON

3- Registro de huéspedes



En esta sección, podrá ver los huéspedes ya registrados en el sistema a través del datagrid. A su vez rellenando los campos correspondientes, podrá registrar un nuevo huésped al sistema, siempre y cuando este no exista en el sistema.

4- Modificación - borrado Huéspedes



En esta sección haciendo doble click sobre el huésped que quiera, podrá modificar sus datos.

Mientras que si el huésped está seleccionado, podrá eliminarlo. Esto generará una cadena de otros efectos, dependiendo de ese huésped. Por ejemplo.

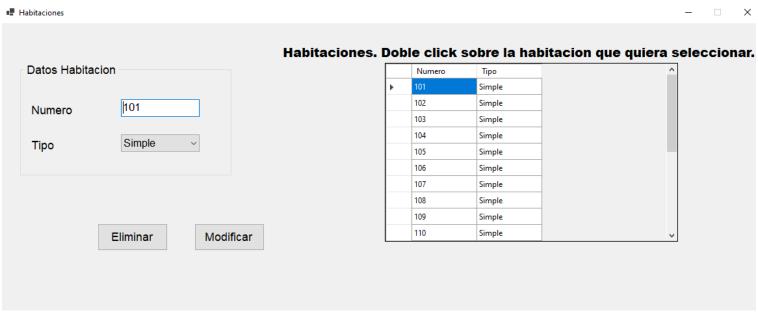
Si elimina un huésped con una reserva hecha, esta reserva automáticamente será eliminada del sistema y la habitación ocupada pasará a estar disponible.

5- HABITACIONES

| | | Habitaciones | | | | |
|------------|--------|--------------|----|--------|--------|--|
| os Habitad | cion | | 1 | Numero | Tipo | |
| | | • | 10 | 01 | Simple | |
| ımero | | | 10 | 02 | Simple | |
| | | | 10 | 03 | Simple | |
| | Simple | | 10 | 04 | Simple | |
| īpo | | | 10 | 05 | Simple | |
| | | | 10 | 06 | Simple | |
| | | | 10 | 07 | Simple | |
| | | | 10 | 08 | Simple | |
| | | | 10 | 09 | Simple | |
| dar | | | 11 | 10 | Simple | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

En esta sección verá las habitaciones en el sistema y a su vez podrá registrar nuevas al sistema.

6- Modificación - borrado habitaciones.



Podrá modificar el número o el tipo de habitación. Solo verá las habitaciones disponibles, es decir, que no tengan reservaciones hechas.

7- Modificación - borrado Reservaciones.

■ Menu Edicion Reservaciones Reservas Reservacion Nro CheckIn CheckOut Habitacion 12345678 20-Nov-23 20-Nov-23 101 CheckIn Monday , November : ~ Monday , November : ∨ CheckOut 101 Habitacion Guardar Eliminar

Igual que con las anteriores entidades, la reservación seleccionada podrá ser eliminada. En ese caso, la habitación correspondiente pasará a estar disponible nuevamente y el huésped no tendrá reservaciones a su nombre.

Doble click sobre la reservación para modificar sus datos.

Entities\SQLLogic\IDataBaseGenericRepository.cs

```
/// <summary>
/// CLASE ABSTRACTA DE LA CUAL HEREDAN TODOS LOS REPOSITORIOS DE LA BASE DE DATOS
/// EN ESTA CLASE SE DEFINEN LOS METODOS GENERICOS QUE SE USARAN EN TODOS LOS REPOSITORIOS
/// TEMA DE PARCIAL: INTERFACES, TIPO GENERICO Y METODOS ASINCRONOS (ASYNC Y AWAIT) MUTI-HILO
/// </summary>
// <typeparam name="T">
// <typeparam name="T"
// <ty
```

Entities\Handlers\GuestRepository.cs

```
Clase que se encarga de la logica de la base de datos de los huespedes
/// </summary>
public class GuestRepository : IDataBaseGenericRepository<Guest>
   private readonly ContextDb;
   public GuestRepository(ContextDb contextDb)
   {
       _contextDb = contextDb;
   }
   /// <summary>
   /// Agrega un huesped a la base de datos
    /// <param name="guest"></param>
   public async Task Add(Guest guest)
   {
       try
           string query = "INSERT INTO Huespedes (Dni, Nombre, Apellido, NumeroTelefono)" +
                          "values(@dni, @name, @lastName, @phoneNumber)";
           using (var command = await _contextDb.CreateCommand(query))
               command.Parameters.AddWithValue("dni", guest.Dni);
               command.Parameters.AddWithValue("name", guest.Name);
               command.Parameters.AddWithValue("LastName", guest.LastName);
               command.Parameters.AddWithValue("phoneNumber", guest.PhoneNumber);
               await _contextDb.ExecuteNonQuery(command);
```

Entities\Serialization\JSONSerialization.cs

```
3 references
public class JSONSerialization : JSONManagment
{
    /// <summary>
    /// Metodo para serializar una cuenta JSON
    /// </summary>
    /// <param name="billing"></param>
    /// <exception cref="NotSerializeJsonException"></exception>
    Oreferences
    public static void SerializeBillings(Billing billing)
    {
        try
        {
            string json = Serialize(billing);
            string path = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "billings.json");
            File.WriteAllText(path, json);
        }
        catch (Exception ex)
        {
            throw new NotSerializeJsonException("Error", ex.Message);
        }
}
```

Entities\SQLLogic\ContextDb.cs

```
public class ContextDb : ConnectionDataBase
    /// <summary>
    /// Abre la conexión con la base de datos y crea un comando
    /// <param name="query"></param>
   public async Task<SqlCommand> CreateCommand(string query)
       await this.Open();
       var command = new SqlCommand(query, this._connection);
       return command;
    /// </summary>
    /// <param name="c"></param>
   public async Task<DataTable> ExecuteReader(SqlCommand c)
       var reader = await c.ExecuteReaderAsync();
       var dataTable = new DataTable();
       dataTable.Load(reader);
       reader.Close();
       return dataTable;
```

UIHotel\BaseFormTrack.cs

```
public class BaseFormTrack : Form
{
    public delegate void UserActionTracker(string action);
    public event UserActionTracker ?OnUserTracker;
    /// <summary>
    /// METODO PARA DISPARAR EL EVENTO DE TRACKER

    /// </summary>
    /// <param name="action"></param>
    11 references
    protected void TriggerUserTracker(string action)
    {
            OnUserTracker?.Invoke(action);
        }
}
```

Entities\Models\GuestsExtensions.cs

```
public static class GuestsExtensions
{
    /// <summary>
    /// Ordena los huespedes por apellido
    /// </summary>
    /// <param name="guests"></param>
    /// <returns>
    /// <returns>
    /// <returns>
    /// <returns>
    /// creturns>
    /// <returns>
    /// <returns>
    // creturns>
    // creturns>
    // summary>
    // creturn guests;
}

/// <summary>
/// Ordena los huespedes por DNI
/// </summary>
/// ordena los huespedes por DNI
/// </summary>
/// ordena los huespedes por DNI
/// </summary>
/// creturns>
    // returns>
    oreferences
public static List<Guest> OrderGuestByDNI(this List<Guest> guests)

{
    var sortedGuests = new List<Guest>(guests);
    guests.Sort((x, y) => string.Compare(x.Dni.ToString(), y.Dni.ToString()));
    return guests;
}
```

```
/// <summary>
/// Evento que se ejecuta al hacer click en el boton de "Huespedes Registrados"
/// </summary>
/// <param name="sender"></param>
/// <param name="e"><</param>
/// <param name="e"><</param>
// <param name="e"><</param>
// <param name="e"><</param>
// <param name="e">
// 
// 
// 
// 
// 
// 
// 
// 
// 
// 
// 
// 
// 
// 
// 
// 
// 
// 
// 
// 
// 
// 
// 
// 
// 
// 
// 
// 
// 
// 
// 
// 
// 
// 
// 
// 
// 
// 
// 
// 
// 
// 
// 
// 
// 
// <p
```