PROYECTO OBLIGATORIO DE PROGRAMACIÓN

"Gestión de Hoteles. Hotel Viña del Mar"

Profesor Carlos Rodriguez.

ÍNDICE

1)Declaración de autoría

Yo, Ignacio Negro, declaro que el presente trabajo titulado "**Sistema de Gestión de Reservas del Hotel Estrella del Mar**" ha sido realizado por mí de manera original como parte de las exigencias de la materia Programación 2.

El contenido del proyecto es de mi autoría. Afirmo que la lógica de la aplicación, el diseño de clases, y la implementación de las funcionalidades han sido desarrolladas por mí. Cualquier consulta realizada para la creación de este proyecto ha sido debidamente aplicada y adaptada a mis necesidades de aprendizaje.

Declaro que no he copiado ni plagiado ningún código de fuentes externas sin la debida atribución, y que he trabajado con integridad y en cumplimiento con las normas académicas de mi institución educativa.

Fecha: 06/11/2024
Nombre: Ignacio Negro
Firma: _____

2) **Abstract del Proyecto.**

Este proyecto consiste en un sistema de gestión de reservas hoteleras, con el principal objetivo defacilitar la administración de reservas, pagos y usuarios de manera eficiente y clara. La aplicación se enfoca en simplificar la estructura de datos y optimizar el mantenimiento del código, sin dejar de cumplir con los requisitos funcionales establecidos.

En términos de diseño, he optado por mantener una única categoría para los usuarios, quienes, una vez registrados y autenticados con su nombre de usuario y contraseña, pueden realizar reservas, gestionar pagos y llevar a cabo todas las operaciones necesarias. Esta decisión responde a la normativa legal uruguaya que exige registrar a cada huésped, pero en el sistema, un usuario logueado es quien registra y asocia las reservas a su nombre. Esto evita la creación de una clase específica para "huésped", simplificando la gestión del proyecto y reduciendo la complejidad innecesaria.

La fusión de funcionalidades para usuarios y huéspedes responde a la idea de que ambos roles comparten las mismas capacidades, como realizar reservas y efectuar pagos. Por lo tanto, implementar todos los métodos y propiedades en una sola clase garantiza una mayor consistencia en el modelo de datos y evita confusiones derivadas de gestionar

múltiples categorías. Esto también facilita el mantenimiento del proyecto, al ser un sistema pequeño que debe permanecer manejable.

Además, para cumplir con la exigencia de listar huéspedes, se implementa un método específico que filtra y muestra únicamente las reservas pagadas correspondientes a la fecha actual y con un else if, a aquellos que se hospedaron previamente.

De este modo, el sistema distingue a los huéspedes en el contexto hotelero sin complicar su arquitectura.

De todas formas cree la clase huésped, por si llega a ser necesario implementarlo a futuro pensando en el crecimiento del software, al igual que fue creada la clase Hotel.

En resumen, el proyecto no solo aborda los requisitos fundamentales, sino que también prioriza una estructura clara y unificada que facilita tanto el desarrollo como la futura gestión y escalabilidad.

3) **Cronograma de Trabajo**

Semana	Actividad	Horas
Semana 1	Planificación y diseño	15 horas
Semana 2	Creación de clases principales y definición de métodos	25 horas
Semana 3	Implementación de métodos y corrección de errores	25 horas
Semana 4	Pruebas, documentación y optimización de código	15 horas
Total		**80 horas**

El proyecto lo dividi en 4 semanas, la primera semana fue planificación y diseñar lo que iba a construir. Definir que clases iba a hacer, y la parte orgánica del proyecto. Investigue y mire muchos videos en youtube sobre sofware de gestión de reservas y lei mucho material antes de empezar a escribir el código.

Terminando la primer semana decidí como iba a hacer el menú y empecé a crear la clase menú con el diccinoario en program, para tener el “esqueleto” del proyecto armado.

La segunda semana empecé a nutrir y desarrollar las clases principales , como habitación, usuario y reserva**.**

Luego de terminar esas clases tuve que tomar la decisión de ver como afrontar la gestión de habitaciones y reservas. Al principio lo hice en clases separadas y luego opte por modular mejor las funciones y unificar gestión de habitaciones en gestión de reservas , cuya justificación esta desarrollada en otro punto.

La tercer semana fue de mucha investigación y creación de la clase gestión usuario y gestionar reservas, el método realizar reserva y optimizar código para no hacerlo tan largo e intentar que funcione de la manera mas compacta posible.

Fue una semana difícil de ensayo y error y mucha investigación.

Luego que a fin de la tercera semana quedo el código funcionando dedique y reserve los últimos días para realizar la documentación y probar todos los métodos y pedir feedback a mis compañeros.

4) Diccionario de clases

No incluye las clases huésped y hotel ya que en el model actual están sin uso.

CLASE GESTION DE USUARIOS

Esta clase se encarga de gestionar todas las operaciones relacionadas con los usuarios del sistema, como el registro, inicio de sesión, y recuperación de contraseñas. También maneja la lista de usuarios y la sesión actual, asegurando que las funcionalidades de autenticación y registro se implementen correctamente

Atributos:

Nombre	Tipo de Dato	Descripción
listaUsuarios	List<Usuario>	Lista que almacena todos los usuarios registrados.
usuarioActual	Usuario (estático)	Referencia al usuario autenticado en la sesión actual.

La inicio vacía a través de un new List, pero luego la lleno con la lista de precargas y doy lugar a crear instancias de usuarios a través del método registrar usuario.

Creo una variable estática usuarioActual para almacenar la referencia al usuario que esta identificado y autenticado a través del usuario y contraseña.

Métodos:

Nombre	Retorno	Parámetros	Descripción
GestionUsuario()	void	-	Constructor que inicializa la lista de usuarios con precarga.
RegistrarUsuario()	void	-	Solicita y valida datos del usuario, y lo registra en la lista.
IniciarSesion()	void	-	Permite iniciar sesión validando el email y la contraseña.
IngresarUsuario()	void	Usuario usuario	Añade un nuevo usuario a la listaUsuarios.
RecuperarContrasena()	void	-	Permite al usuario recuperar su contraseña mediante validación de email.

CLASE GESTION DE RESERVAS.

Decidí crear una clase para gestionar todas las reservas y las habitaciones unificándolas ya que están fuertemente relacionadas y sus operaciones de manipulaciones de datos dependen mutuamente en un sistema de gestión hotelera.

También optimizo recursos ya que facilita el manejo de operaciones comunes como verificar la disponibilidad de una habitación en relación a las reservas hechas, o actualizar el estado de reserva de una habitación luego de una cancelacion.

Mantengo de esta manera la alta cohesión ya que las responsabilidades de esta clase esta fuertemente relacionada a la gestión de reservas y habitaciones necesarias para dichas reservas.

Facilito de esta manera al unificar la gestión de habitación y reservas en una sola clase, el mantenimiento del software y la escalabilidad pensando a futuro.

Atributos:

Nombre	Tipo de Dato	Descripción
listaHabitaciones	List<Habitacion>	Lista que gestiona todas las habitaciones del sistema.
listaReservas	List<Reserva>	Lista que almacena todas las reservas registradas.
fechaLlegada	DateTime	Fecha de llegada para las reservas.
fechaSalida	DateTime	Fecha de salida para las reservas.

MÉTODOS:

Nombre	Retorno	Parámetros	Descripción
GestionReserva()	void	-	Constructor que inicializa listas de habitaciones y reservas con precarga.
ListarReservas()	void	-	Muestra una lista de las reservas actuales en consola.
ListarHabitaciones()	void	-	Muestra una lista de todas las habitaciones disponibles en consola.
ConsultarHabitacionesDisponibles()	void	-	Filtra y muestra las habitaciones disponibles.
BuscarHabitacionesDisponiblesParaReserva()	List<Habitacion>	DateTime, DateTime	Retorna habitaciones disponibles para las fechas indicadas.
RealizarReserva()	void	-	Permite al usuario realizar una reserva validando fechas y disponibilidad.
CancelarReserva()	void	-	Cancela una reserva solicitando el email del cliente y mostrando opciones.
ModificarReserva()	void	-	Permite modificar los detalles de una reserva existente.
ListarHuespedes()	void	-	Lista los huéspedes con reservas pagadas y activas en la fecha actual.

ObtenerHistorialReservasPorEmail()	List<Reserva>	string email	Retorna el historial de reservas asociadas a un email.
MostrarHistorialReservas()	void	-	Muestra el historial de reservas del usuario autenticado.
EjecutarPago()	void	-	Permite ejecutar el pago de una reserva y actualiza su estado.
GenerarComprobanteDePago()	void	Reserva reserva	Genera y muestra un comprobante de pago en formato de texto.
MostrarHabitacionesMasReservadas()	void	-	Muestra las habitaciones más reservadas con el conteo de reservas.

CLASE HABITACION

Clase molde para crear instancias de Habitación.

Atributos

Nombre	Tipo de Dato	Descripción
numeroHabitacion	int	Número de la habitación.
tipo	string	Tipo de habitación (sencilla, doble, suite).
cantidadPersonas	int	Número máximo de personas que la habitación puede alojar.
precio	decimal	Precio por noche de la habitación.
disponible	bool	Indica si la habitación está disponible para reservas.
fechasReservadas	List<DateTime>	Lista de fechas en las que la habitación ya está reservada.

Métodos

Nombre	Retorno	Parámetros	Descripción
Habitacion()	void	int, string, int, decimal	Constructor que inicializa los atributos de la habitación.

CLASE MENU

Clase para manejar el menú de la aplicación. Está encargada de manejar y mostrar las distintas opciones que el usuario puede seleccionar para interactuar con el sistema de gestión de reservas del hotel. Esta clase permite navegar por diferentes menús y submenús de forma organizada, facilitando al usuario el acceso a las funcionalidades del sistema, como la gestión de usuarios**,** reservas, cancelaciones, pagos y reportes estadísticos.

Nombre	Tipo de Dato	Descripción
sesionIniciada	bool	Indica si la sesión del usuario está iniciada. Es un atributo estático.

Indico si la sesión del usuario esta iniciada. Este atributo es estático y lo comparto con todas las instancias de la clase**.**

Métodos

Nombre	Retorno	Parámetros	Descripción
MenuIniciarSesion()	void	-	Muestra el menú de inicio de sesión con opciones.
MenuPrincipal()	void	-	Muestra el menú principal con las opciones principales del sistema.
GestionUsuarios()	void	-	Muestra el submenú de gestión de usuarios.
ReservasYCancelaciones()	void	-	Muestra el submenú de reservas y cancelaciones.
GestionDePagos()	void	-	Muestra el submenú de gestión de pagos.
EstadisticaYReportes()	void	-	Muestra el submenú de estadísticas y reportes del sistema.

CLASE PERSONA

La clase persona define la estructura de un individuo. incluyendo atributos personales como nombre, apellido, fecha de nacimiento, y otros datos de identificación. Esta clase sirve como base para la clase Usuario, que agrega funcionalidad adicional para la gestión de usuarios en el sistema.

Atributos:

Nombre	Tipo de Dato	Descripción
Nombre	string	Almacena el nombre de la persona.
Apellido	string	Almacena el apellido de la persona.
FechaNacimiento	DateTime	Almacena la fecha de nacimiento de la persona.
Email	string	Almacena el correo electrónico de la persona.
Pais	string	Almacena el país de residencia de la persona.
TipoDocumento	string	Almacena el tipo de documento (DNI, pasaporte, etc.).
NumeroDocumento	int	Almacena el número de documento de la persona.
Telefono	int	Almacena el número de teléfono de la persona.

Métodos

Nombre	Return	Parámetros	Descripción
Persona(string nombre, string apellido, DateTime fechaNacimiento, string email, string pais, string tipoDocumento, int numeroDocumento, int telefono)	void	nombre, apellido, fechaNacimiento, email, pais, tipoDocumento, numeroDocumento, telefono	Constructor que inicializa una nueva instancia de `Persona` con todos los atributos.

CLASE PRECARGA

Proporciona datos iniciales al sistema, incluyendo habitaciones, reservas y usuarios. Esto es útil para utilizar y manipular datos sin necesidad de ingresar manualmente información al sistema.

Métodos

Nombre	Return	Parámetros	Descripción
PrecargarHabitaciones()	List<Hab>	Ninguno	Crea y retorna una lista de habitaciones con sus detalles.
PrecargarReservas()	List<Reserva>	Ninguno	Crea y retorna una lista de reservas con información relevante.
PrecargarUsuarios()	List<Usuario>	Ninguno	Crea y retorna una lista de usuarios con datos personales y credenciales.

CLASE RESERVA

Esta clase gestiona la información de cada reserva, incluyendo detalles como las fechas de inicio y fin, la habitación reservada, y el cliente que la hizo. También controla el estado del pago y asigna un identificador único a cada reserva.

Atributos

Nombre	Tipo de Dato	Descripción
IDgenerador (static)	int	Contador estático para generar un ID único por reserva.
IDReserva	int	Identificador único de la reserva.
NumeroHabitacion	int	Número de la habitación reservada.
FechaInicio	DateTime	Fecha de inicio de la reserva.
FechaFin	DateTime	Fecha de finalización de la reserva.
EmailCliente	string	Correo electrónico del cliente.
FechaReserva	DateTime	Fecha en que se realizó la reserva.
UsuarioId	int	Identificador del usuario (no se usa en el constructor).
EstaPagada	bool	Indica si la reserva ha sido pagada.

MÉTODOS

Nombre	Return	Parámetros	Descripción
Reserva(int numeroHabitacion, DateTime fechaInicio, DateTime fechaFin, DateTime fechaReserva, string emailCliente)	void	numeroHabitacion, fechaInicio, fechaFin, fechaReserva, emailCliente	Constructor que inicializa una nueva reserva con todos los detalles.
Reserva(int numeroHabitacion, DateTime fechaInicio, DateTime fechaFin, string emailCliente)	void	numeroHabitacion, fechaInicio, fechaFin, emailCliente	Constructor que inicializa una reserva sin una fecha de reserva específica.

CLASE USUARIO

Esta clase es una extensión de la clase Persona, a la cual le agrego un ID UNICO y una contraseña.

Atributos

Nombre	Tipo de Dato	Descripción
UsuarioId	int	Identificador único del usuario.
NombreUsuario	string	Nombre de usuario para el inicio de sesión.
Contraseña	string	Contraseña del usuario (generalmente se almacena de forma segura).
NivelAcceso	string	Nivel de acceso del usuario (por ejemplo, administrador o cliente).

Métodos

Nombre	Return	Parámetros	Descripción
Usuario(int usuarioId, string nombreUsuario, string contraseña, string nivelAcceso, string nombre, string apellido, DateTime fechaNacimiento, string email, string pais, string tipoDocumento, int numeroDocumento, int telefono)	void	usuarioId, nombreUsuario, contraseña, nivelAcceso, nombre, apellido, fechaNacimiento, email, pais, tipoDocumento, numeroDocumento, telefono	Constructor que inicializa una nueva instancia de `Usuario` con todos los atributos, incluyendo los heredados de `Persona`.
CambiarContraseña(string nuevaContraseña)	void	nuevaContraseña	Permite al usuario cambiar su contraseña.
ValidarCredenciales(string nombreUsuario, string contraseña)	bool	nombreUsuario, contraseña	Valida si las credenciales proporcionadas coinciden con las almacenadas.

CLASE PROGRAM

La clase Program determina la lógica de la aplicación, controlando la navegación del usuario a través de un menú interactivo. Dependiendo de las selecciones del usuario, se invocan diferentes acciones que permiten gestionar usuarios, reservas, pagos y otras funciones del sistema hotelero. La implementación de un diccionario facilita la gestión de opciones del menú de manera eficiente y clara.

Decidí utilizar un diccionario porque me pareció una opción mas escalable y mantenible para agregar mas opciones en comparación con un switch ya que ofrece una forma mas modular y flexible de gestionar las opciones y permite que el código crezca y evolucione con mayor claridad a medida que se va complejizando.

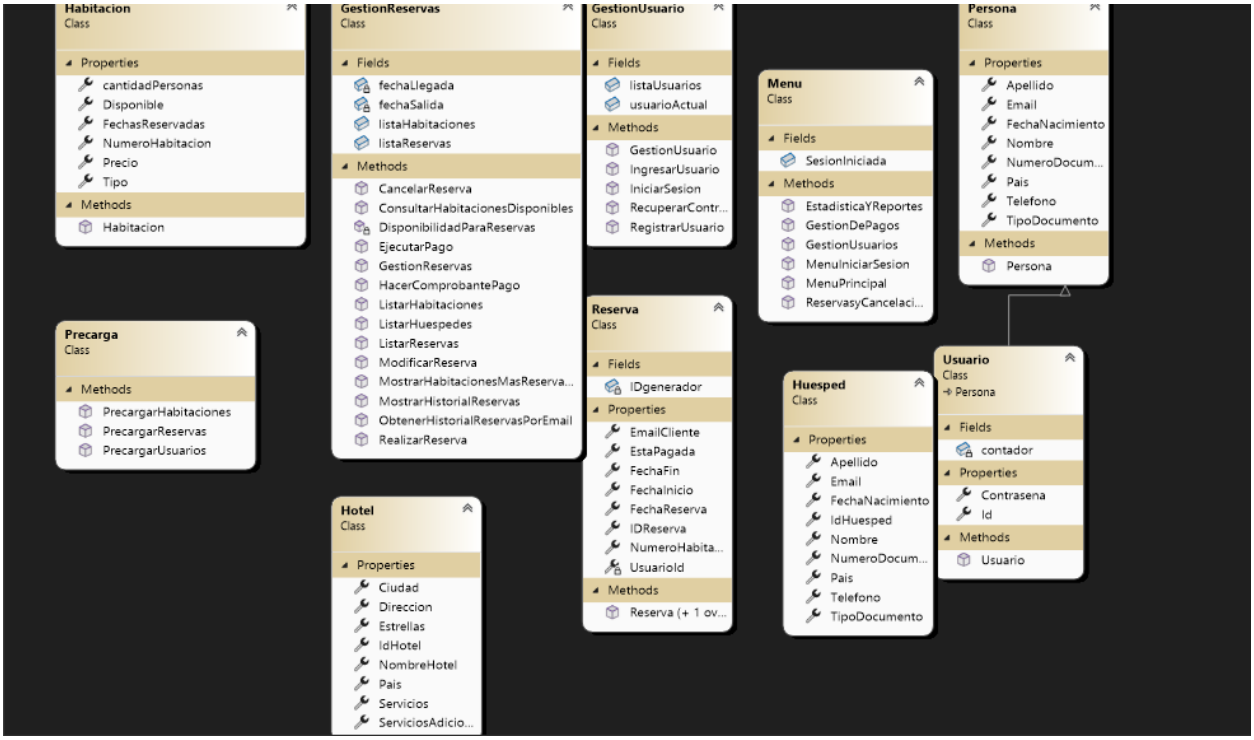
Atributos

Nombre	Tipo de Dato	Descripción
gestionUsuario	GestionUsuario	Instancia de la clase `GestionUsuario`, que maneja operaciones relacionadas con la gestión de usuarios.
gestionReserva	GestionReserva	Instancia de la clase `GestionReserva`, que gestiona las operaciones relacionadas con las reservas.
salir	bool	Variable booleana que controla el flujo del programa, indicando si el usuario desea salir del menú.
keyValuePairs	Dictionary<int, Action>	Diccionario que relaciona números de opción con acciones, donde cada número representa una opción del menú y cada acción es el método a ejecutar.

Métodos:

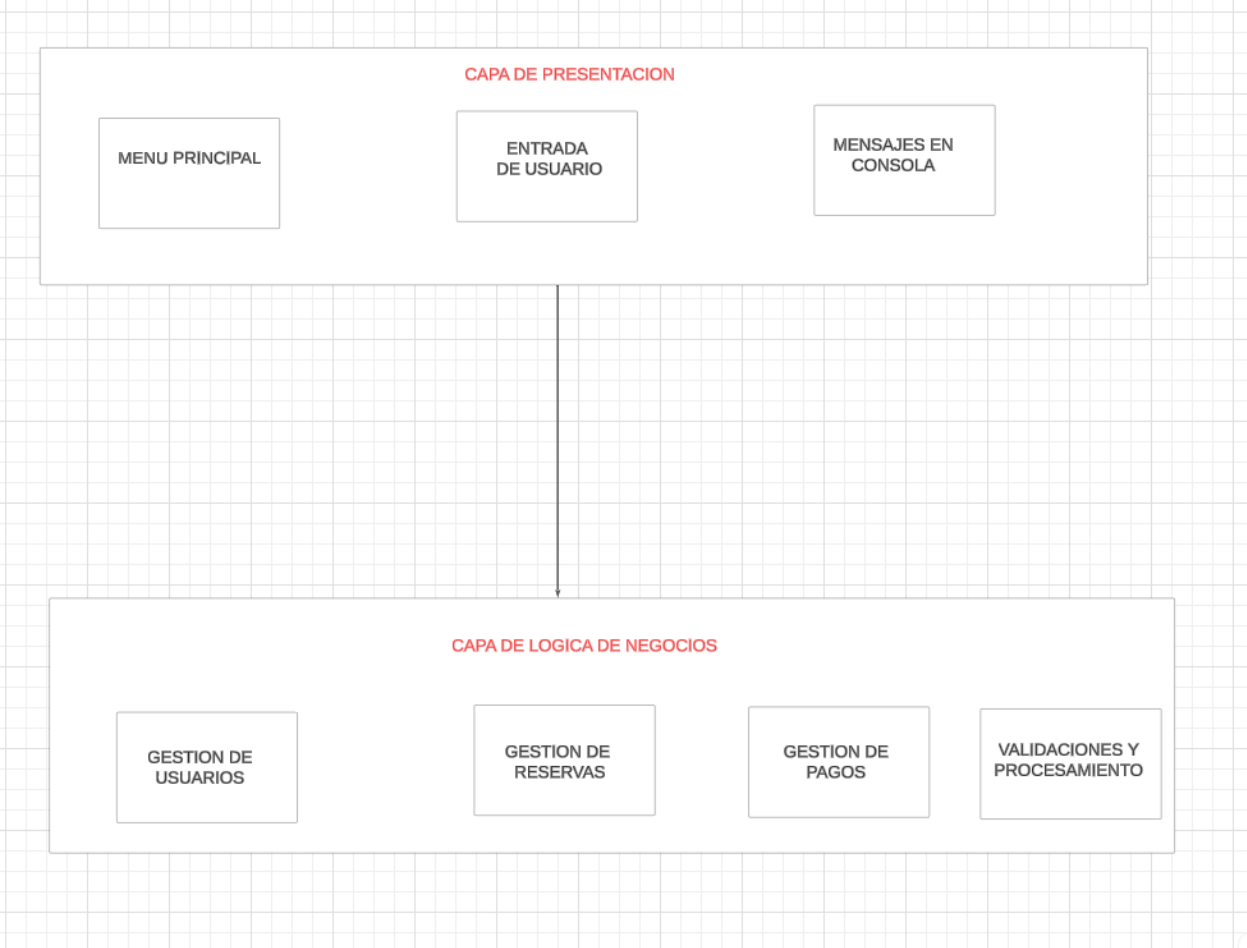
Nombre	Return	Parámetros	Descripción
Program()	void	Ninguno	Constructor que inicializa las instancias de `GestionUsuario` y `GestionReserva`, y configura el menú.
MenuIniciarSesion()	void	Ninguno	Muestra el menú de inicio de sesión en la consola, permitiendo al usuario iniciar sesión o salir.
MenuPrincipal()	void	Ninguno	Muestra el menú principal, con opciones para gestionar usuarios, reservas, pagos, y generar reportes.
GestionUsuarios()	void	Ninguno	Muestra el submenú de gestión de usuarios, permitiendo registrar nuevos usuarios o recuperar contraseñas.
ReservasYCancelaciones()	void	Ninguno	Muestra el submenú de reservas y cancelaciones, con opciones para realizar, modificar o cancelar reservas.
GestionDePagos()	void	Ninguno	Muestra el submenú de gestión de pagos, con opciones para ejecutar y gestionar pagos de reservas.
EstadisticaYReportes()	void	Ninguno	Muestra el submenú de estadísticas y reportes, como listar huéspedes y ver el historial de reservas.
EjecutarMenu()	void	Ninguno	Ejecuta el menú principal y gestiona la navegación en función de las opciones seleccionadas por el usuario.

5) Diagrama de la capa de dominio



Realizado en Visual Studio con la herramienta Class Diagram vista en clase(la clase derivada es usuario de persona y no al revés)

6) Diagrama de dos capas



Capa de presentación:

Esta es la capa encargada de la interacción del usuario , conlleva todo lo que el usuario puede ver e interactuar, menus formularios y mensajes de consola.

Capa de Lógica de Negocio:

Capa encargada de toda la lógica del negocio y la lógica principal del sistema. Es la capa encargada de procesar datos y ejecutar operaciones.

Con opciones de escalabilidad se puede dejar lugar para trabajar con BDD e implementar una capa de Acceso a Datos y tener un modelo de 3 capas.

7) Resultados de las pruebas (TESTING)

Se probaron los métodos individuales de las clases principales (GestionUsuario, GestionReserva, Usuario, Habitacion y Reserva).

RegistrarUsuario: Verificación de que se añada correctamente un nuevo usuario a la lista.

ConsultarHabitacionesDisponibles: Validación de que se muestren solo las habitaciones libres en un rango de fechas.

Todas las pruebas unitarias se ejecutaron con éxito, detectándose y corrigiéndose errores menores en la validación de datos y manejo de excepciones.

Se evaluaron las funcionalidades completas del sistema, asegurando que los casos de uso más relevantes cumplieran con los requisitos especificados.

Escenarios probados:

- Registro de usuario y recuperación de contraseña.
- Realización y cancelación de reservas.
- Procesamiento de pagos y visualización de estadísticas.

Las pruebas funcionales confirmaron que el funcionamiento del sistema es coherente y sin interrupciones, ,mostrando una experiencia de usuario adecuada.

Correcta comunicación entre la gestión de reservas y la actualización de disponibilidad de habitaciones.

Verificación de que las reservas realizadas se asocien correctamente a los usuarios.

La integración de las clases se realizó sin problemas significativos, y las interacciones entre los módulos fueron fluidas y consistentes.

Finalmente se llevó a cabo un test de usabilidad para evaluar la experiencia del usuario al interactuar con el sistema.

El menú de opciones fue calificado como intuitivo y fácil de navegar.

Algunas sugerencias de mejora incluyeron la adición de mensajes de confirmación más claros al realizar acciones críticas (como la cancelación de una reserva).

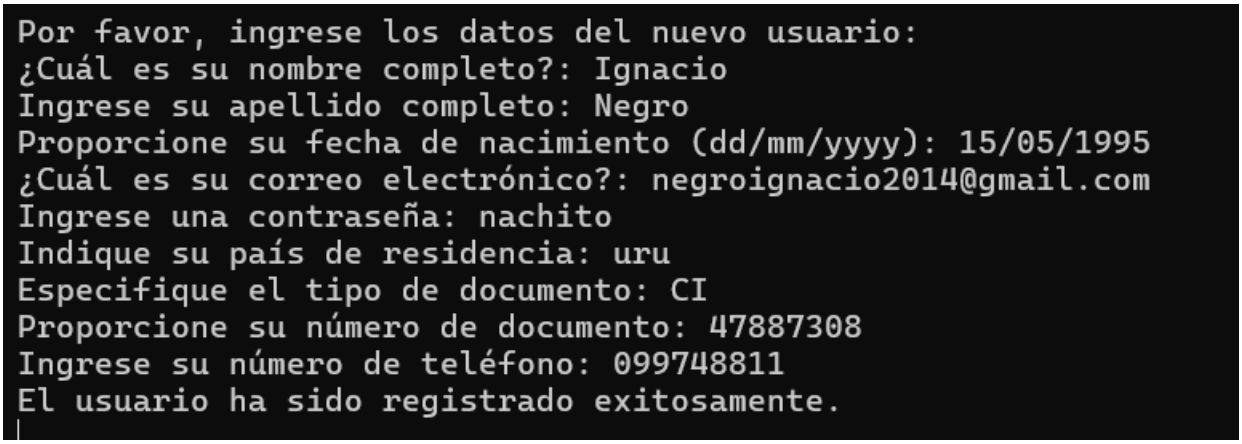
Se implementaron ajustes menores en la interfaz y en la forma de presentar mensajes al usuario.

Las pruebas abarcaron un 90% de las funcionalidades del sistema, dejando fuera solo escenarios extremadamente raros que no afectan el uso general.

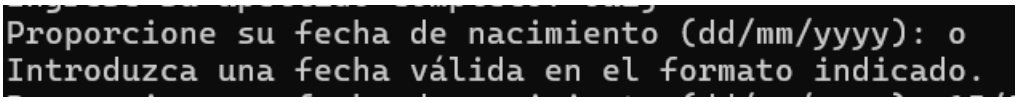
Se detectaron y solucionaron problemas de validación en la entrada de datos y en la lógica de negocios para casos límites.

Capturas de pantalla:

Ejecución de `RegistrarUsuario` La siguiente captura muestra la correcta adición de un nuevo usuario al sistema.



Validacion si ingreso un formato diferente a lo esperado (en este ejemplo con datetime)



Luego Intento iniciar sesión con el email y la contraseña, ya agregados como una instancia de Usuario en listaUsuarios.

```
INICIAR SESION
Ingrese su email:
negroignacio2014@gmail.com

Ingrese su contraseña:
nachito
```

Si coincide, se despliega el menú principal. Y ejecuta la parte del bloque que acepta todas las opciones para manejar correctamente el sistema.

```
***MENU PRINCIPAL***

Selecciona una opción:

1. Gestion de Usuarios
2. Reservas y Cancelaciones
3. Gestion de Pagos
4. Estadisticas y Reportes
0. Salir del programa
```

```
while (!salir)
{
    string? input = Console.ReadLine();

    if (int.TryParse(input, out int opcion))
    {

        if (!Menu.SesionIniciada)
        {
            if (opcion != 6 && opcion != 7 && opcion != 3)
            {
                Conso  (local variable) int opcion  válida en el menú de Inicio de Sesión.
                continue;
            }
        }

        if (keyValuePairs.ContainsKey(opcion))
        {
            keyValuePairs[opcion]();
            if (opcion == 7)
            {
                Menu.SesionIniciada = true;
            }
        }
        else
        {
            Console.WriteLine("Opción no válida.");
        }
    }
    else
```

Testing Gestión Reservas.

Luego de autentificarme ingreso la opción 12 en el Menu. Realizar Reserva.

Coloco las fechas deseadas y se desplegan las habitaciones disponibles en esa fecha.

```
Ingrese su fecha de llegada (dd/mm/yyyy): 05/12/2025
Ingrese su fecha de salida (dd/mm/yyyy): 10/12/2025
Seleccione el número de habitación de las disponibles:
Número: 101, Tipo: Simple, Precio: $ 75,50
Número: 102, Tipo: Doble, Precio: $ 120,00
Número: 103, Tipo: Triple, Precio: $ 150,75
Número: 104, Tipo: Suite, Precio: $ 300,00
Número: 105, Tipo: Doble, Precio: $ 115,00
Número: 201, Tipo: Simple, Precio: $ 70,00
Número: 202, Tipo: Suite Deluxe, Precio: $ 500,00
Número: 203, Tipo: Doble, Precio: $ 125,50
Número: 204, Tipo: Triple, Precio: $ 160,00
Número: 301, Tipo: Simple, Precio: $ 85,00
Número: 302, Tipo: Suite Junior, Precio: $ 280,00
Número: 303, Tipo: Doble, Precio: $ 135,00
Número: 304, Tipo: Suite Presidencial, Precio: $ 1.000,00
Número: 401, Tipo: Simple, Precio: $ 90,00
Número: 402, Tipo: Doble Superior, Precio: $ 150,00
Número: 403, Tipo: Triple Deluxe, Precio: $ 200,00
Número: 404, Tipo: Suite Familiar, Precio: $ 450,00
Número: 501, Tipo: Penthouse, Precio: $ 1.500,00
Número: 502, Tipo: Simple Económica, Precio: $ 60,00
Número: 503, Tipo: Doble Económica, Precio: $ 100,00
Ingrese el número de la habitación:
```

```
Reserva creada con los siguientes detalles:
ID Reserva: 5010
Número de Habitación: 503
Fecha de Llegada: 5/12/2025
Fecha de Salida: 10/12/2025
Fecha de Reserva: 1/1/0001
Email del Cliente: admin
```

Luego de tener la reserva confirmada, con el ID generado automáticamente número 5010, procedemos a crear el comprobante de pago.

Vamos a la opción 14 en el Menu.

```
GESTION DE PAGOS

14. Realizar Pago
5. Volver al menu Principal
14
Ingrese el número de reserva que desea pagar:
```

```
14
Ingrese el número de reserva que desea pagar: 5010
Ingrese su numero de tarjeta de crédito, o cuenta bancaria para confirmar el pagamento
65165
```

Simulamos un número de tarjeta de crédito o cuenta bancaria.

```
Comprobante de Pago
------------------------------
Número de Reserva: 5010
Fechas: Desde 5/12/2025 Hasta 10/12/2025
Email: admin
Estado: Pagada
------------------------------

El comprobante de pago se ha guardado en ComprobantePago_5010.txt
Ingrese una letra para continuar
```

El comprobante fue generado y el archivo txt generado se guardará en la carpeta del proyecto automáticamente..

Modificar Reserva (opción 12)

Veamos si ahora queremos modificar esa reserva(siempre verificar que fue hecha con el mismo usuario logueado, en este caso admin.)

```
Ingrese su email:
admin
Reservas encontradas:
ID: 5007, Habitación: 204, Cliente: admin, Desde: 20/2/2025 Hasta: 25/2/2025
ID: 5009, Habitación: 204, Cliente: admin, Desde: 25/1/2025 Hasta: 30/1/2025
ID: 5010, Habitación: 503, Cliente: admin, Desde: 5/12/2025 Hasta: 10/12/2025
Ingrese el ID de la reserva que desea modificar:
5010
```

Se despliegan las reservas hechas.

Ingresamos el numero de la reserva a modificar en este caso 5010.

```
Reservas encontradas:
ID: 5007, Habitación: 204, Cliente: admin, Desde: 20/2/2025 Hasta: 25/2/2025
ID: 5009, Habitación: 204, Cliente: admin, Desde: 25/1/2025 Hasta: 30/1/2025
ID: 5010, Habitación: 503, Cliente: admin, Desde: 5/12/2025 Hasta: 10/12/2025
Ingrese el ID de la reserva que desea modificar:
5010
Reserva encontrada: ID 5010, Habitación 503, Desde 5/12/2025 Hasta 10/12/2025
Ingrese el NUEVO número de habitación
502
Ingrese la nueva fecha de inicio dd/mm/yyyy
05/10/2024
Ingrese la nueva fecha de fin dd/mm/yyyy
05/20/2024
Reserva modificada exitosamente.
```

Ponemos los nuevos datos.

Chequemos en el Historial de reservas (opción 19) para verificar el status de la nueva reserva

```
ID Reserva: 5010, Número de Habitación: 502, Fecha de Inicio: 5/10/2024, Fecha de Fin: 10/12/2025, Email del Cliente: ad
min
Digite una tecla para continuar
```

Consultar habitaciones disponibles (opción 10)

```
Habitaciones disponibles:
Número: 105, Tipo: Doble, Capacidad: 2, Precio Diario: 115,00
Número: 202, Tipo: Suite Deluxe, Capacidad: 4, Precio Diario: 500,00
Número: 203, Tipo: Doble, Capacidad: 2, Precio Diario: 125,50
Número: 301, Tipo: Simple, Capacidad: 1, Precio Diario: 85,00
Número: 302, Tipo: Suite Junior, Capacidad: 2, Precio Diario: 280,00
Número: 303, Tipo: Doble, Capacidad: 2, Precio Diario: 135,00
Número: 304, Tipo: Suite Presidencial, Capacidad: 5, Precio Diario: 1000,00
Número: 401, Tipo: Simple, Capacidad: 1, Precio Diario: 90,00
Número: 402, Tipo: Doble Superior, Capacidad: 2, Precio Diario: 150,00
Número: 403, Tipo: Triple Deluxe, Capacidad: 3, Precio Diario: 200,00
Número: 404, Tipo: Suite Familiar, Capacidad: 4, Precio Diario: 450,00
Número: 501, Tipo: Penthouse, Capacidad: 6, Precio Diario: 1500,00
Número: 502, Tipo: Simple Económica, Capacidad: 1, Precio Diario: 60,00
Número: 503, Tipo: Doble Económica, Capacidad: 2, Precio Diario: 100,00
Presione una tecla para continuar.
```

Muestra todas las habitaciones que no están reservadas.

Para ver las habitaciones disponibles en una fecha determinada , ir al método realizar reserva e ingresar las fechas deseadas para consultar disponibldad. Simulando una página tipo booking.

Reservas duplicadas.

Observemos el caso del usuario admin, cargado en Precarga, lista de usuarios.

```
List<Reserva> listaReservas = new List<Reserva>
{
    new Reserva(101, new DateTime(2024, 10, 25), new DateTime(2024, 10, 28), "juan.perez@example.com"){ EstaPagada = true },
    new Reserva(102, new DateTime(2024, 11, 1), new DateTime(2024, 11, 5), "maria.gonzalez@example.com"){ EstaPagada = true },
    new Reserva(103, new DateTime(2024, 10, 30), new DateTime(2024, 11, 2), "carlos.lopez@example.com"){ EstaPagada = true },
    new Reserva(104, new DateTime(2024, 12, 15), new DateTime(2024, 12, 20), "ana.rodriguez@example.com"),
    new Reserva(104, new DateTime(2024, 12, 22), new DateTime(2024, 12, 28), "ana.rodriguez@example.com"),
    new Reserva(201, new DateTime(2025, 1, 5), new DateTime(2025, 1, 10), "pedro.martinez@example.com"),
    new Reserva(204, new DateTime(2025, 2, 5), new DateTime(2025, 2, 10), "pedro.martinez@example.com"),
    new Reserva(204, new DateTime(2025, 2, 20), new DateTime(2025, 2, 25), "admin"),
    new Reserva(204, new DateTime(2025, 2, 1), new DateTime(2025, 2, 5), "maria.gonzalez@example.com"),
    new Reserva(204, new DateTime(2025, 1, 25), new DateTime(2025, 1, 30), "admin")
};
```

Intentaremos crear una reserva para comprobar el código para no permitir reservas duplicadas, requisito del proyecto, ingresado dentro del Método realizar reserva. Con la siguiente lógica:

```
// Requisito de no tener reserva duplicada
string emailCliente = GestionUsuario.usuarioActual.Email;
if (listaReservas.Any(r => r.EmailCliente == emailCliente &&
(fechaLlegada < r.FechaFin && fechaSalida > r.FechaInicio) ||       // cuando coinciden en el medio
(fechaLlegada == r.FechaFin) ||                                     // el nuevo checkin es justo cuando termina una reserva
(fechaSalida == r.FechaInicio)                                      // el nuevo checkout es justo cuando empieza una reserva

{
    Console.WriteLine("Las fechas de la reserva se superponen con una reserva existente. Serás redirigido al Menu Principal.");
    return;
}
else
{
    Console.WriteLine("Reserva permitida.");
}
```

Comprobamos que el mensaje recibido es el siguiente

```
Ingrese su fecha de llegada (dd/mm/yyyy): 25/01/2025
Ingrese su fecha de salida (dd/mm/yyyy): 30/01/2025
Seleccione el número de habitación de las disponibles:
Número: 101, Tipo: Simple, Precio: $ 75,50
Número: 102, Tipo: Doble, Precio: $ 120,00
Número: 103, Tipo: Triple, Precio: $ 150,75
Número: 104, Tipo: Suite, Precio: $ 300,00
Número: 105, Tipo: Doble, Precio: $ 115,00
Número: 201, Tipo: Simple, Precio: $ 70,00
Número: 202, Tipo: Suite Deluxe, Precio: $ 500,00
Número: 203, Tipo: Doble, Precio: $ 125,50
Número: 301, Tipo: Simple, Precio: $ 85,00
Número: 302, Tipo: Suite Junior, Precio: $ 280,00
Número: 303, Tipo: Doble, Precio: $ 135,00
Número: 304, Tipo: Suite Presidencial, Precio: $ 1.000,00
Número: 401, Tipo: Simple, Precio: $ 90,00
Número: 402, Tipo: Doble Superior, Precio: $ 150,00
Número: 403, Tipo: Triple Deluxe, Precio: $ 200,00
Número: 404, Tipo: Suite Familiar, Precio: $ 450,00
Número: 501, Tipo: Penthouse, Precio: $ 1.500,00
Número: 502, Tipo: Simple Económica, Precio: $ 60,00
Número: 503, Tipo: Doble Económica, Precio: $ 100,00
Ingrese el número de la habitación: 101
Las fechas de la reserva se superponen con una reserva existente. Serás redirigido al Menu Principal.
```