Sistema Integral de Seguridad y Gestión Residencial

Lucas Jaramillo Areiza

Proyecto Final - Programación Aplicada

Universidad Distrital

10/02/2025

BOGOTA DC.

ING. NELLY ISABEL NAVARRO

Este proyecto tiene como objetivo el desarrollo de un sistema integral para la gestión y seguridad residencial, el cual incluye funcionalidades como el registro de visitantes, control de acceso, gestión de pagos y reservas de zonas comunes. El sistema está diseñado para mejorar la seguridad, eficiencia y comodidad en el manejo de los servicios dentro de un conjunto residencial.

Índice de Temas

1. Introducción

- **Objetivo del Sistema**: Breve descripción del sistema de gestión de acceso, pagos y reservas para un conjunto residencial.
- **Tecnologías Utilizadas**: Explicación sobre las tecnologías utilizadas (MySQL, ASP.NET, C#, etc.).
- **Estructura del Proyecto**: Descripción general de la estructura del proyecto.

2. Modelo de Datos

- **Tablas y Relaciones**:
    - **Usuarios**: Gestión de residentes y administradores (Campos y relaciones con otras tablas).
    - **Tarjetas**: Control de acceso mediante tarjetas RFID, su estado (activa/inactiva) y relación con usuarios.
    - **Accesos**: Registro de accesos (entradas y salidas) de los residentes o visitantes.
    - **Visitantes**: Registro y gestión de visitantes, con su estado (pendiente/aprobado).
    - **Reservas**: Gestión de reservas de zonas comunes (estado y fecha).
    - **Zonas Comunes**: Definición y gestión de las zonas comunes disponibles para reserva.
    - **Pagos**: Registro de pagos realizados por los residentes.
    - **Estado de Pagos**: Relación entre pagos y reservas, gestión del estado de pago (pendiente/pagado).

3. Operaciones Básicas del Sistema

- **Registrar un Visitante**:
    - Flujo de trabajo para registrar un visitante en el sistema.
    - Validación de residente y asignación de un estado inicial (pendiente).
- **Registrar Acceso**:
    - Verificación de tarjeta RFID y su estado.

    - Registro de un acceso exitoso o fallido en la tabla `accesos`.
- **Realizar una Reserva de Zona Común**:
    - Proceso de selección de zona común y asignación de fecha y hora.
    - Validación del estado de la zona (disponible/ocupada).
- **Registrar Pago**:
    - Ingreso de pagos y su relación con las reservas.
    - Actualización del estado del pago y su relación con los residentes.

4. Flujos de Trabajo

- **Flujo de Entrada y Salida de Residentes**:
    - Cómo se gestionan los accesos (entradas y salidas) de los residentes con tarjetas RFID.
- **Flujo de Reserva de Zonas Comunes**:
    - Cómo los residentes pueden hacer una reserva y cómo se valida la disponibilidad de las zonas comunes.
- **Flujo de Visitas**:
    - Cómo se registran los visitantes y cómo se valida la aprobación o rechazo de su visita.
- **Flujo de Pagos y Estado de Pagos**:
    - Proceso de registro de pagos, actualización del estado y vinculación con las reservas.

5. Validaciones y Seguridad

- **Validación de Datos de Entrada**:
    - Validación de los campos en los formularios (residencia, tarjeta, visitante, etc.).
- **Protección contra Inyección SQL**:
    - Aunque mencionaste que no se implementa protección contra inyección SQL, es importante mencionar la práctica recomendada.
- **Seguridad en el Acceso a la Información**:
    - Gestión de roles (residente, administrador) y permisos de acceso.

6. Interfaz de Usuario

- **Menú de Navegación**:
    - Funcionalidades disponibles para los usuarios (estado de pagos, historial de accesos, reservas, etc.).
- **Pantallas Principales**:
    - Descripción de las interfaces para los residentes y administradores.
- **Formulario de Registro de Visitantes**:
    - Diseño y funcionalidad del formulario de registro de visitantes.

7. Manejo de Errores y Excepciones

- **Errores Comunes en el Sistema**:
  - Ejemplos de errores posibles (residente no encontrado, tarjeta inactiva, zona ocupada, etc.).
- **Manejo de Excepciones en C#**:
  - Uso de bloques `try-catch` para manejar errores y excepciones durante el registro de accesos, pagos, reservas, etc.

8. Consultas y Reportes

- **Consultas de Accesos**:
  - Cómo obtener el historial de accesos de un residente o visitante.
- **Consultas de Pagos**:
  - Consultas para mostrar el estado de los pagos de un residente.
- **Consultas de Reservas**:
  - Cómo obtener información sobre las reservas realizadas por un residente.

9. Consideraciones Finales

- **Mejoras Futuras**:
  - Sugerencias sobre cómo mejorar el sistema (agregar más validaciones, integrar más funcionalidades, etc.).
- **Impacto del Sistema**:
  - Cómo este sistema facilita la gestión de un conjunto residencial (mejora de la eficiencia, reducción de conflictos, etc.).

10.Diagramas.

- Representación gráfica de los casos de uso
- Descripción de los actores y sus interacciones

1. Introducción

Objetivo del Sistema: El Sistema Integral de Seguridad y Gestión Residencial tiene como objetivo proporcionar una solución completa para la administración y seguridad de un conjunto residencial. Este sistema permite la gestión de acceso de residentes y visitantes mediante tarjetas RFID, el seguimiento de pagos mensuales, la reserva de zonas comunes, y la gestión de visitantes. A través de un portal web, los residentes pueden acceder a funcionalidades que optimizan el uso de las instalaciones y servicios del conjunto residencial, mejorando la seguridad y la experiencia del usuario. Además, permite a los administradores gestionar eficientemente los registros de acceso, los pagos y las reservas de manera automatizada.

Tecnologías Utilizadas: El desarrollo del sistema se ha llevado a cabo utilizando una combinación de tecnologías robustas y ampliamente utilizadas en el desarrollo web y la administración de bases de datos:

- **MySQL**: Es el sistema de gestión de bases de datos utilizado para almacenar la información relacionada con residentes, pagos, accesos, visitantes y reservas. Se utilizaron tablas específicas para cada módulo del sistema, lo que permite un almacenamiento organizado y eficiente de los datos.
- **ASP.NET Web Forms**: La interfaz web está construida utilizando ASP.NET, un framework de desarrollo web de Microsoft. Esta plataforma facilita la creación de aplicaciones web dinámicas y seguras, permitiendo la integración de múltiples funcionalidades con facilidad. En particular, se utiliza el modelo Web Forms, que permite la creación de páginas interactivas con controles de servidor y lógica de negocio.
- **C#**: El lenguaje de programación utilizado para el desarrollo de la lógica de negocio en el backend. C# proporciona una excelente integración con ASP.NET, lo que permite una gestión eficiente de las bases de datos, el procesamiento de formularios, la autenticación y la manipulación de los datos de los residentes y visitantes.
- **HTML, CSS y JavaScript**: Estas tecnologías son utilizadas para la creación de las interfaces de usuario. HTML es responsable de la estructura del contenido, CSS para la apariencia visual, y JavaScript para añadir interactividad y validar la información del formulario antes de enviarla.

Estructura del Proyecto: El proyecto está organizado en varias capas que permiten una fácil escalabilidad y mantenimiento. La estructura del proyecto se basa en una arquitectura en capas, que separa las distintas funcionalidades y responsabilidades:

- **Capa de Presentación (Frontend)**: Esta capa está formada por las páginas ASP.NET (.aspx) y los controles de servidor, junto con los archivos de estilo (CSS) y los scripts (JavaScript) que forman la interfaz de usuario. La capa de presentación permite la interacción entre el usuario y el sistema, ofreciendo formularios para el registro de visitantes, el control de pagos y la gestión de reservas.
- **Capa de Lógica de Negocio (Backend)**: La lógica de negocio es gestionada a través de clases de C# como `AccesoController` y `ConexionBD`. Estas clases manejan la interacción con la base de datos, realizando consultas, inserciones y actualizaciones según las necesidades del sistema. Por ejemplo, `AccesoController` maneja los accesos de los residentes y visitantes, mientras que `ConexionBD` gestiona la conexión a la base de datos MySQL.
- **Capa de Datos**: Esta capa está constituida por la base de datos MySQL, que almacena toda la información relacionada con el sistema. La base de datos contiene varias tablas, como `usuarios`, `tarjetas`, `accesos`, `pagos`, `reservas` y `visitantes`, que están interrelacionadas a través de claves

foráneas. La estructura de la base de datos está diseñada para garantizar la integridad de los datos y soportar todas las funcionalidades del sistema.

La combinación de estas tecnologías y la estructuración en capas permiten que el sistema sea flexible, escalable y fácil de mantener, adaptándose a futuras mejoras o ampliaciones de funcionalidades.

2. Modelo de Datos

El modelo de datos del Sistema Integral de Seguridad y Gestión Residencial está basado en una estructura relacional que organiza la información en varias tablas, cada una con un propósito específico. A continuación, se describe en detalle las tablas principales y sus relaciones.

Tablas y Relaciones:

Usuarios: Gestión de residentes y administradores

La tabla `usuarios` contiene los datos de los residentes y administradores del conjunto residencial. Es la tabla principal que interactúa con otras tablas, como las de `tarjetas`, `pagos`, `reservas`, y `visitantes`. Esta tabla almacena los datos personales básicos de cada usuario, como su nombre, apellido, cédula, correo y teléfono, además de su rol dentro del sistema (residente o administrador). La relación con otras tablas es a través del campo `ID_Usuario`, que actúa como clave primaria.

Estructura de la tabla `usuarios`:

Campo	Tipo de Datos	Descripción
ID_Usuario	INT (PK)	Identificador único de cada usuario (residente o administrador).
Nombre	VARCHAR(100)	Nombre del usuario.
Apellido	VARCHAR(100)	Apellido del usuario.
Cedula	VARCHAR(20)	Cédula de identidad del usuario.
Correo	VARCHAR(100)	Correo electrónico del usuario.
Telefono	VARCHAR(15)	Número de teléfono del usuario.
Rol	ENUM ('Administrador', 'Residente')	Rol del usuario, determina si es administrador o residente.
Contraseña	VARCHAR(255)	Contraseña del usuario para acceso al sistema.

Relaciones:

- **Tarjetas:** La tabla `usuarios` está relacionada con la tabla `tarjetas` mediante la clave `ID_Usuario`. Cada usuario tiene asociada una o varias tarjetas RFID para el acceso al conjunto residencial.
- **Pagos:** La tabla `usuarios` se relaciona con la tabla `pagos` a través del campo `ID_Residente`. Cada pago realizado por un residente se registra con su identificador de usuario.
- **Reservas:** La tabla `usuarios` también se relaciona con la tabla `reservas`, permitiendo a los residentes realizar reservas de las zonas comunes.
- **Visitantes:** A través del campo `ID_Residente`, los residentes pueden registrar a sus visitantes en la tabla `visitantes`.

Tarjetas: Control de acceso mediante tarjetas RFID

La tabla `tarjetas` gestiona la información relacionada con las tarjetas RFID de los residentes. Cada tarjeta tiene un identificador único (`ID_Tarjeta`) y está asociada a un residente mediante el campo `ID_Usuario`. El campo `Estado` de la tabla permite determinar si una tarjeta está activa o inactiva, controlando así si el residente tiene acceso al conjunto residencial. Si una tarjeta está activa, el residente puede utilizarla para ingresar o salir del lugar. En caso contrario, no podrá acceder.

Estructura de la tabla `tarjetas`:

Campo	Tipo de Datos	Descripción
ID_Tarjeta	INT (PK)	Identificador único de cada tarjeta RFID.
ID_Usuario	INT (FK)	Identificador del residente al que pertenece la tarjeta.
Estado	ENUM ('Activa', 'Inactiva')	Estado de la tarjeta, si está activa o inactiva.

Relaciones:

- **Usuarios:** La tabla `tarjetas` tiene una relación uno a muchos con la tabla `usuarios` a través del campo `ID_Usuario`. Esto indica que un residente puede tener varias tarjetas RFID asignadas, pero cada tarjeta solo está asociada a un único residente.
- **Accesos:** La tabla `tarjetas` se relaciona con la tabla `accesos` mediante la clave `ID_Tarjeta`. Cada vez que un residente accede al conjunto residencial, el sistema registra la entrada o salida en la tabla `accesos`.

Resumen de Relaciones:

- **Usuarios ↔ Tarjetas:** Un usuario puede tener varias tarjetas RFID asociadas, pero una tarjeta pertenece a un único usuario.
- **Usuarios ↔ Pagos:** Un usuario (residente) tiene registros de pagos asociados a su ID, lo que permite conocer el estado de sus pagos.
- **Usuarios ↔ Reservas:** Un residente puede hacer múltiples reservas de zonas comunes, cada una registrada en la tabla `reservas`.
- **Usuarios ↔ Visitantes:** Un residente puede registrar varios visitantes en la base de datos.

Este modelo de datos asegura que la información esté bien organizada y permite un acceso rápido y eficiente a los datos relacionados con el sistema de gestión del conjunto residencial. La correcta implementación de las relaciones entre las tablas garantiza la integridad de los datos y facilita el mantenimiento del sistema.

3. Operaciones Básicas del Sistema

A continuación se describen las operaciones básicas del sistema de gestión de acceso, pagos y reservas para el conjunto residencial. Estas operaciones son esenciales para asegurar el funcionamiento adecuado del sistema, garantizando que los residentes, visitantes y recursos se gestionen de manera eficiente.

Registrar un Visitante

Flujo de trabajo para registrar un visitante:

1. **Ingreso de los datos del visitante:**
    a. El sistema solicita al usuario (residente) ingresar los datos del visitante, como su nombre completo, documento de identificación, motivo de la visita y, en algunos casos, la fecha y hora de la visita.
    b. El residente también debe introducir su número de documento de identificación (cédula) para relacionar al visitante con su cuenta.
2. **Validación del residente:**
    a. El sistema verifica si el residente que está registrando al visitante existe en la base de datos a través de su número de documento (campo `ID_Residente`).
    b. Si el residente no está registrado, el sistema muestra un mensaje de error indicando que el residente no se encuentra en la base de datos.
3. **Asignación de estado inicial (Pendiente):**
    a. Una vez validado el residente, el sistema asigna automáticamente al visitante el estado `Pendiente`, ya que la aprobación final de la visita dependerá de la validación por parte de los administradores del conjunto residencial.

b. La información del visitante se guarda en la tabla `visitantes`.

Acciones posteriores:

- El estado del visitante se actualizará más adelante a `Aprobado` o `Rechazado` dependiendo de la verificación administrativa.

Registrar Acceso

Verificación de tarjeta RFID y su estado:

1. **Lectura de la tarjeta RFID:**
    a. Cuando un residente o visitante intenta acceder al conjunto residencial, el sistema lee la tarjeta RFID.
    b. El sistema verifica el estado de la tarjeta mediante la consulta en la tabla `tarjetas`, buscando si está activa o inactiva. Si la tarjeta está inactiva, el acceso es denegado.
2. **Registro del acceso:**
    a. Si la tarjeta está activa, el sistema registra la entrada o salida del residente/visitante en la tabla `accesos`.
    b. Los campos registrados incluyen el `ID_Tarjeta`, la fecha y hora del acceso, y el tipo de acceso (Entrada/Salida).
3. **Acceso exitoso o fallido:**
    a. Si el acceso se realiza correctamente, el sistema registra el evento en la tabla de accesos y muestra un mensaje de éxito.
    b. Si la tarjeta está inactiva o no es válida, el sistema deniega el acceso y muestra un mensaje de error.

Realizar una Reserva de Zona Común

Proceso de selección de zona común y asignación de fecha y hora:

1. **Selección de zona común:**
    a. El residente selecciona la zona común que desea reservar (por ejemplo, salón social, gimnasio, etc.). Esta selección se realiza mediante una interfaz de usuario en el sistema.
    b. El sistema muestra una lista de zonas comunes disponibles en la base de datos a través de la tabla `zonascomunes`.
2. **Asignación de fecha y hora:**
    a. El residente selecciona la fecha y hora en la que desea realizar la reserva. El sistema valida que la fecha y hora no coincidan con otras reservas ya existentes.

Validación del estado de la zona (Disponible/Ocupada):

1. **Verificación del estado de la zona:**
    a. El sistema valida que la zona común seleccionada esté disponible para la fecha y hora indicadas. Si la zona ya está ocupada, el sistema muestra un mensaje de error e invita al residente a elegir otra fecha o zona.
    b. Si la zona está disponible, el sistema reserva el espacio para el residente y cambia el estado de la zona a `Ocupada` en la tabla `zonascomunes`.
2. **Confirmación de la reserva:**
    a. La reserva se registra en la tabla `reservas` con el `ID_Residente`, el `ID_Zona`, la fecha y hora de la reserva y el estado `Confirmada`.

Registrar Pago

Ingreso de pagos y su relación con las reservas:

1. **Ingreso de pago:**
    a. El residente realiza el pago por el servicio de reservas u otros servicios relacionados con el conjunto residencial.
    b. El pago se ingresa en el sistema mediante un formulario de pago, donde se incluye el monto y la fecha del pago.
2. **Relación con las reservas:**
    a. El pago está relacionado con la reserva que realiza el residente. El sistema vincula el pago con la reserva correspondiente a través del `ID_Reserva` en la tabla `estado_pagos`.
    b. El monto del pago también se registra junto con la fecha y estado de la transacción.

Actualización del estado del pago y su relación con los residentes:

1. **Estado del pago:**
    a. El estado del pago se establece como `Pendiente` inicialmente. Una vez que el pago se confirma, el estado se actualiza a `Pagado`.
    b. El sistema asegura que el estado de pago se actualice de acuerdo con la transacción.
2. **Relación con los residentes:**
    a. Cada pago está asociado con un `ID_Residente`, lo que permite al sistema gestionar las deudas de cada residente y garantizar que se registren adecuadamente todas las transacciones financieras del residente.

Este conjunto de operaciones básicas garantiza que el sistema de gestión de acceso, pagos y reservas funcione correctamente y de manera eficiente, asegurando la correcta administración del conjunto residencial y el cumplimiento de los servicios para los residentes.

4. Flujos de Trabajo

A continuación, se describen los flujos de trabajo que permiten gestionar de manera eficiente las operaciones diarias dentro del sistema de gestión de acceso, pagos y reservas. Estos flujos garantizan que las tareas se realicen de forma ordenada, proporcionando una experiencia fluida tanto para los residentes como para los administradores del conjunto residencial.

Flujo de Entrada y Salida de Residentes

Cómo se gestionan los accesos (entradas y salidas) de los residentes con tarjetas RFID:

1. **Lectura de la tarjeta RFID:**
    a. El residente se aproxima al lector de tarjetas RFID instalado en la entrada del conjunto residencial.
    b. El sistema lee la tarjeta y verifica su estado en la base de datos.
2. **Verificación del estado de la tarjeta:**
    a. Si la tarjeta está activa, el sistema registra el acceso. Si la tarjeta está inactiva, el acceso es denegado y el residente recibe un mensaje de error que le informa de la inactivación de la tarjeta.
3. **Registro del acceso:**
    a. Si la tarjeta es válida, el sistema registra el acceso en la tabla `accesos`, guardando la fecha y hora del acceso y el tipo de acceso (entrada o salida).
4. **Acceso o denegación:**
    a. Si el acceso es una **entrada**, el sistema autoriza al residente y le permite ingresar.
    b. Si el acceso es una **salida**, el residente puede salir sin restricciones, siempre que la tarjeta esté activa y registrada.
5. **Estado de acceso:**
    a. El sistema muestra un mensaje de éxito o error dependiendo del estado de la tarjeta (activa o inactiva).

Flujo de Reserva de Zonas Comunes

Cómo los residentes pueden hacer una reserva y cómo se valida la disponibilidad de las zonas comunes:

1. **Selección de la zona común:**
    a. El residente ingresa al sistema y selecciona la zona común que desea reservar (por ejemplo, el salón social, gimnasio, etc.).
    b. El sistema presenta un listado de las zonas comunes disponibles para la fecha y hora solicitada.
2. **Validación de disponibilidad:**
    a. El sistema verifica la disponibilidad de la zona común. Si la zona está **disponible**, se procede a la reserva.
    b. Si la zona está **ocupada** en el horario solicitado, el residente recibe un mensaje que le invita a seleccionar otro horario o zona disponible.
3. **Confirmación de la reserva:**
    a. Una vez validada la disponibilidad, el residente puede confirmar la reserva. El sistema registra la reserva en la base de datos en la tabla `reservas`, asociando la zona, la fecha y hora, y el estado de la reserva como **Confirmada**.
4. **Cambio de estado de la zona:**
    a. El estado de la zona seleccionada en la tabla `zonascomunes` se actualiza a **Ocupada** para evitar que otros residentes reserven la misma zona en ese horario.

Flujo de Visitas

Cómo se registran los visitantes y cómo se valida la aprobación o rechazo de su visita:

1. **Registro del visitante:**
    a. El residente ingresa al sistema y registra los datos de su visitante: nombre completo, documento de identificación, motivo de la visita y, en algunos casos, fecha y hora de la visita.
    b. La información del visitante se guarda en la tabla `visitantes` con el estado **Pendiente**.
2. **Validación del visitante:**
    a. El sistema verifica que el visitante esté asociado a un residente válido en la base de datos, utilizando el `ID_Residente`.
3. **Aprobación o rechazo de la visita:**
    a. El administrador revisa la solicitud de visita. Si la visita es aprobada, el estado del visitante en la tabla `visitantes` se actualiza a **Aprobado**.

   b. Si la visita es rechazada, el estado del visitante se actualiza a **Rechazado** y el residente recibe la notificación correspondiente.
4. **Acceso del visitante:**
   a. Si el visitante es aprobado, se le permite ingresar al conjunto residencial (según la política del conjunto) con una tarjeta temporal o un acceso manual autorizado por el administrador.

Flujo de Pagos y Estado de Pagos

Proceso de registro de pagos, actualización del estado y vinculación con las reservas:

1. **Registro del pago:**
   a. El residente realiza el pago correspondiente a una reserva o servicio del conjunto residencial.
   b. El pago se registra en el sistema a través de un formulario, donde se ingresan el monto, la fecha del pago y el `ID_Residente` correspondiente.
2. **Relación con las reservas:**
   a. El sistema vincula el pago con una reserva específica mediante el `ID_Reserva`, lo que garantiza que el pago se asocie correctamente con el servicio o zona común reservada por el residente.
3. **Actualización del estado del pago:**
   a. Una vez ingresado el pago, el estado del pago se actualiza a **Pagado** en la tabla `pagos`.
   b. El estado de la reserva también se actualiza a **Confirmada**, si no se había hecho ya, en la tabla `reservas`.
4. **Confirmación del pago:**
   a. El sistema emite un recibo de pago o confirma la transacción mediante un mensaje al residente, notificándole que su pago ha sido registrado y su reserva está confirmada.
5. **Manejo de pagos pendientes:**
   a. Si el pago está en estado **Pendiente**, el sistema muestra una alerta al residente para que lo realice lo antes posible.

Estos flujos de trabajo proporcionan una visión clara de cómo los procesos se gestionan dentro del sistema, asegurando la eficiencia en la administración del conjunto residencial, y permitiendo una experiencia de usuario fluida para los residentes y administradores.

5. Validaciones y Seguridad

En este apartado se detallan las validaciones implementadas en el sistema, así como las prácticas recomendadas de seguridad para garantizar el correcto funcionamiento y la protección de los datos y accesos dentro del conjunto residencial.

Validación de Datos de Entrada

Es fundamental garantizar que los datos ingresados por los usuarios sean correctos y se ajusten a los formatos esperados. Las validaciones de entrada en los formularios permiten evitar datos erróneos que puedan generar inconsistencias o errores en la base de datos.

1. **Validación de campos de residentes:**
    a. **Nombre y Apellido:** Se valida que los campos de nombre y apellido no estén vacíos y que solo contengan caracteres alfabéticos. Se utiliza una expresión regular para asegurarse de que los valores no contengan números o caracteres especiales.
    b. **Correo Electrónico:** Se valida que el correo electrónico ingresado tenga el formato correcto utilizando una expresión regular. Además, se verifica que el correo no esté duplicado en la base de datos.
    c. **Cédula:** Se valida que la cédula esté en el formato adecuado (en este caso, solo números) y que no esté repetida.
2. **Validación de tarjetas RFID:**
    a. Al registrar una tarjeta RFID, se valida que el `ID_Tarjeta` sea único y esté asociado a un residente. Si se encuentra un ID repetido o inexistente, el sistema genera un mensaje de error.
3. **Validación de visitantes:**
    a. **Nombre y Documento:** Se valida que los campos de nombre y documento no estén vacíos y que el documento sea único.
    b. **Motivo de visita:** El campo "Motivo de visita" puede ser opcional, pero si se ingresa, se valida que no contenga caracteres especiales ni datos incorrectos.
4. **Validación de reservas:**
    a. Se valida que la fecha y hora de la reserva no estén en el pasado, garantizando que los residentes solo puedan realizar reservas para fechas futuras.
    b. Además, se valida que la zona común esté disponible en el horario solicitado antes de proceder con la reserva.
5. **Validación de pagos:**
    a. Se valida que el monto del pago sea mayor a cero y que esté correctamente asociado a una reserva específica.

Protección contra Inyección SQL

Aunque en este sistema no se implementa protección contra inyección SQL de manera explícita, es importante mencionar que existen prácticas recomendadas para mitigar este tipo de vulnerabilidades, las cuales deben ser aplicadas para garantizar la seguridad de la base de datos.

1. **Uso de consultas parametrizadas:** Las consultas parametrizadas son una de las mejores prácticas para prevenir inyecciones SQL. Aunque actualmente no se implementa de forma explícita, se recomienda que todas las consultas a la base de datos utilicen parámetros en lugar de concatenar valores directamente en la cadena SQL.

Ejemplo de consulta segura:

```
csharp
CopiarEditar
string query = "SELECT * FROM usuarios WHERE Correo = @correo";
MySqlCommand cmd = new MySqlCommand(query, conexion.conectar);
cmd.Parameters.AddWithValue("@correo", correo);
```

Este método asegura que los valores ingresados no sean tratados como código SQL, evitando posibles inyecciones maliciosas.

2. **Validación y Escapado de Entradas:** Las entradas de los usuarios deben ser validadas correctamente antes de ser utilizadas en consultas SQL. Además, cualquier entrada que pueda contener caracteres especiales debe ser escapada adecuadamente para prevenir su uso indebido en las consultas.
3. **Uso de procedimientos almacenados:** Otra opción para mejorar la seguridad es el uso de procedimientos almacenados en la base de datos. Al usar procedimientos almacenados, los parámetros se pasan de forma segura y se evita la manipulación directa de consultas SQL.

Seguridad en el Acceso a la Información

El sistema implementa un modelo de control de acceso basado en roles, lo que garantiza que solo los usuarios con los permisos adecuados puedan acceder a información sensible o realizar acciones críticas.

1. **Gestión de roles (residente, administrador):**

    a. **Administrador:** Los usuarios con el rol de administrador tienen acceso completo al sistema, incluyendo la gestión de residentes, tarjetas, pagos, reservas y la administración de visitantes. Los administradores pueden ver, editar y eliminar cualquier dato dentro del sistema.
    b. **Residente:** Los residentes tienen acceso limitado al sistema. Pueden gestionar sus propios datos, consultar el estado de pagos y accesos, hacer reservas y registrar visitantes. Sin embargo, no tienen acceso a la gestión de otros residentes ni a las funciones administrativas.
2. **Permisos de acceso:**
    a. **Página de administración:** Solo los usuarios con el rol de administrador pueden acceder a las páginas que permiten modificar o eliminar datos de otros residentes, cambiar el estado de pagos, gestionar visitantes y modificar configuraciones del sistema.
    b. **Página de residente:** Los residentes solo pueden acceder a su propia información personal, historial de accesos, estado de pagos, y realizar acciones relacionadas con su perfil, como la reserva de zonas comunes y el registro de visitantes.
3. **Autenticación y autorización:**
    a. El sistema requiere que los usuarios se autentiquen con su correo y contraseña antes de acceder a cualquier funcionalidad.
    b. La contraseña se almacena de manera segura utilizando técnicas de hash para evitar la exposición de contraseñas en texto claro.
    c. Una vez autenticados, se verifica el rol del usuario y se le redirige a la página correspondiente según su perfil.
4. **Protección contra accesos no autorizados:**
    a. Se implementan mecanismos para evitar que los usuarios puedan acceder a páginas o datos de otros residentes. Por ejemplo, si un residente intenta acceder a la página de administración, se redirige automáticamente a la página de inicio.
    b. Además, se revisan los permisos de los usuarios antes de realizar cualquier operación que pueda afectar los datos sensibles.

Conclusión de Validaciones y Seguridad:

La validación de datos y la seguridad en el acceso a la información son fundamentales para garantizar el buen funcionamiento del sistema y la protección de la privacidad y seguridad de los usuarios. Aunque el sistema actual no implementa algunas de las mejores prácticas de seguridad, es importante que se apliquen las recomendaciones mencionadas para reforzar la protección y prevenir vulnerabilidades.

6. Interfaz de Usuario

La interfaz de usuario (UI) es esencial para permitir una interacción eficiente con el sistema. Los usuarios (residentes y administradores) tienen accesos diferenciados dependiendo de su rol, y la navegación se realiza mediante botones, en lugar de un menú lateral (sidebar). A continuación, se detallan los componentes clave de la interfaz.

Menú de Navegación (Botones)

Los botones son la principal forma de navegación dentro del sistema, y se presentan de manera clara en cada pantalla para facilitar el acceso a las funcionalidades. Dependiendo del tipo de usuario (residente o administrador), los botones disponibles varían según las funcionalidades a las que tengan acceso.

Para el Residente:

1. **Estado de Pagos:** Botón para acceder a la pantalla donde el residente puede consultar el estado de sus pagos (pendientes o realizados).
2. **Historial de Accesos:** Accede a la sección donde el residente puede ver el registro de sus accesos (entradas y salidas) a las zonas comunes.
3. **Reserva de Zonas Comunes:** Permite hacer una nueva reserva de una zona común disponible, seleccionando la zona, la fecha y la hora.
4. **Registro de Visitantes:** A través de un botón, el residente puede registrar a un visitante proporcionando su nombre, documento de identificación y motivo de la visita.
5. **Estado de su Tarjeta RFID:** Al presionar el botón, el residente puede consultar el estado de su tarjeta RFID (activa o inactiva).
6. **Contacto con la Administración:** Botón que abre un formulario o ventana para contactar a la administración con cualquier duda o solicitud.

Para el Administrador:

1. **Gestión de Residentes:** El administrador puede agregar, editar o eliminar residentes, así como consultar los datos de cada uno.
2. **Gestión de Visitantes:** Los administradores tienen un botón para gestionar las solicitudes de los residentes, aprobar o rechazar las visitas, y ver detalles de los visitantes.
3. **Gestión de Reservas:** A través de este botón, el administrador puede ver todas las reservas, modificarlas o cancelarlas según sea necesario.

4. **Gestión de Pagos:** Botón para ver el estado de los pagos de los residentes, y modificar o actualizar el estado de los mismos.
5. **Gestión de Tarjetas RFID:** Permite al administrador activar o desactivar las tarjetas RFID según sea necesario.
6. **Visitas Pendientes:** El administrador puede revisar las solicitudes de visitas pendientes y aprobar o rechazar las mismas.

Pantallas Principales

Las pantallas principales varían según el tipo de usuario, pero ambas comparten la misma estructura de navegación mediante botones.

Pantalla Principal del Residente:

1. **Estado de Pagos:** Los residentes pueden consultar el estado de sus pagos mediante un botón que los redirige a una pantalla con el detalle de los pagos realizados y el estado actual de cada uno.
2. **Historial de Accesos:** Los residentes tienen acceso a su historial de accesos a las zonas comunes. El botón correspondiente los lleva a una lista con la fecha y hora de cada entrada y salida.
3. **Reserva de Zonas Comunes:** Este botón permite a los residentes seleccionar la zona común que desean reservar, y si está disponible, pueden proceder a asignar una fecha y hora para la reserva.
4. **Registro de Visitantes:** Los residentes pueden registrar visitantes a través de este botón, donde deben ingresar los detalles del visitante (nombre, documento, motivo de la visita).
5. **Estado de Tarjeta RFID:** El residente puede verificar el estado de su tarjeta RFID, ya sea activa o inactiva, mediante un botón en su pantalla.
6. **Contacto con Administración:** Un botón abre un formulario para contactar con la administración en caso de cualquier solicitud o problema.

Pantalla Principal del Administrador:

1. **Gestión de Residentes:** Este botón lleva al administrador a una pantalla donde puede agregar, editar o eliminar residentes y consultar sus datos personales.
2. **Gestión de Visitantes:** A través de este botón, el administrador puede ver las solicitudes de visita de los residentes y aprobar o rechazar las mismas.
3. **Gestión de Reservas:** El administrador puede acceder a esta opción para gestionar las reservas de las zonas comunes, ver las fechas y horas solicitadas, y realizar cambios o cancelaciones.

4. **Gestión de Pagos:** Con este botón, el administrador puede revisar el estado de los pagos de los residentes, actualizar los pagos y asociarlos con las reservas realizadas.
5. **Gestión de Tarjetas RFID:** Este botón permite al administrador gestionar el estado de las tarjetas RFID, activando o desactivando las mismas según sea necesario.
6. **Visitas Pendientes:** El administrador tiene un botón para ver las visitas pendientes de aprobación o rechazo, con detalles de cada solicitud.

Formulario de Registro de Visitantes

El formulario de **registro de visitantes** es esencial para controlar las entradas al conjunto residencial. A través de este formulario, los residentes pueden ingresar a los visitantes para su validación.

1. **Diseño del Formulario:**
    a. **Campos requeridos:**
        i. **Nombre del Visitante:** Ingreso del nombre completo del visitante.
        ii. **Documento de Identificación:** Número de documento de identificación del visitante (cédula, pasaporte, etc.).
        iii. **Motivo de la Visita:** Breve descripción del motivo de la visita (opcional).
        iv. **Fecha y Hora de Entrada:** Selección de la fecha y hora en que el visitante ingresará al conjunto residencial.
    b. **Botón de Enviar:** El residente envía la solicitud de visita mediante un botón. Esta solicitud será revisada por el administrador para su aprobación o rechazo.
2. **Funcionalidad del Formulario:**
    a. Al enviar el formulario, la solicitud de visita se marca inicialmente como **pendiente**.
    b. El administrador recibe la solicitud y tiene la opción de aprobar o rechazar la visita, con base en la disponibilidad y las políticas del conjunto residencial.
    c. Si la visita es aprobada, el residente recibe una notificación de confirmación.

Conclusión:

El sistema utiliza botones para facilitar la navegación entre las diferentes funcionalidades. Esto asegura que los usuarios puedan acceder a las opciones

necesarias de forma clara y rápida. Las pantallas están diseñadas para ser intuitivas y accesibles, con formularios simples y fáciles de llenar, como el de registro de visitantes.

7. Manejo de Errores y Excepciones

El manejo adecuado de errores y excepciones es crucial para asegurar la estabilidad del sistema y proporcionar una experiencia de usuario coherente. A continuación se describen los errores más comunes que pueden ocurrir en el sistema y cómo se manejan mediante excepciones en C#.

Errores Comunes en el Sistema

Durante el uso del sistema, varios tipos de errores pueden surgir, y es fundamental que el sistema los maneje correctamente para evitar que el usuario se vea afectado. Los errores comunes incluyen:

1. **Residente No Encontrado:**
    a. **Descripción:** Puede ocurrir cuando un residente intenta realizar una operación (como registrar un pago o acceder a zonas comunes) pero no existe en la base de datos.
    b. **Solución:** El sistema debe verificar si el residente existe antes de intentar realizar cualquier operación. Si no se encuentra, se debe notificar al usuario que el residente no existe.
2. **Tarjeta Inactiva:**
    a. **Descripción:** Cuando un residente intenta utilizar su tarjeta RFID para acceder, pero la tarjeta se encuentra en estado "Inactiva".
    b. **Solución:** Al intentar registrar un acceso, el sistema debe verificar el estado de la tarjeta. Si la tarjeta está inactiva, se debe mostrar un mensaje informando que el acceso no se puede realizar debido al estado de la tarjeta.
3. **Zona Ocupada:**
    a. **Descripción:** Ocurre cuando un residente intenta hacer una reserva en una zona común, pero la zona ya está reservada para la fecha y hora solicitada.
    b. **Solución:** El sistema debe verificar la disponibilidad de la zona antes de confirmar la reserva. Si la zona está ocupada, debe mostrarse un mensaje al residente indicando que no está disponible en el horario deseado.
4. **Pago Pendiente:**

a. **Descripción:** Cuando un residente intenta realizar una reserva o registrar un visitante, pero tiene un pago pendiente.
b. **Solución:** El sistema debe verificar el estado de los pagos antes de permitir realizar otras acciones. Si el pago está pendiente, se debe bloquear la acción y pedir al residente que realice el pago.

5. **Error en la Conexión a la Base de Datos:**
   a. **Descripción:** Un fallo en la conexión con la base de datos puede ocurrir si hay problemas de red o con el servidor.
   b. **Solución:** El sistema debe ser capaz de manejar este tipo de error mostrando un mensaje adecuado y tratando de reconectar o notificar al usuario sobre el problema.

Manejo de Excepciones en C#

El manejo adecuado de errores y excepciones en C# es esencial para evitar que el sistema se bloquee y para proporcionar información clara sobre lo que ha fallado. A continuación se describe cómo se manejan las excepciones utilizando bloques `try-catch` en el código.

Uso de Bloques try-catch

Los bloques `try-catch` permiten capturar y manejar excepciones de manera controlada. El flujo de trabajo básico consiste en colocar el código que podría generar una excepción dentro del bloque `try`. Si ocurre una excepción, se captura en el bloque `catch` y se maneja adecuadamente.

8. Consultas y Reportes

El sistema debe permitir a los administradores y residentes consultar y obtener reportes sobre diversos aspectos del sistema, como los accesos, los pagos y las reservas. A continuación se detallan los procesos y consultas necesarias para obtener la información relevante en cada uno de estos casos.

Consultas de Accesos

Las consultas de accesos permiten obtener el historial de entradas y salidas tanto de residentes como de visitantes, proporcionando una visión clara de los movimientos dentro del conjunto residencial.

Consulta para Obtener el Historial de Accesos de un Residente

Para consultar el historial de accesos de un residente, se debe realizar una consulta SQL que filtre los registros de la tabla `accesos` por el `ID_Tarjeta` asociado al residente, ya que cada acceso está relacionado con una tarjeta RFID. Aquí hay un ejemplo de cómo podría estructurarse la consulta:

```
sql
CopiarEditar
SELECT a.ID_Acceso, a.Fecha_Hora, a.Tipo, t.Estado AS Estado_Tarjeta
FROM accesos a
JOIN tarjetas t ON a.ID_Tarjeta = t.ID_Tarjeta
WHERE t.ID_Usuario = @idResidente
ORDER BY a.Fecha_Hora DESC;
```

- **Explicación:** Esta consulta obtiene el historial de accesos de un residente específico. Se hace un `JOIN` entre las tablas `accesos` y `tarjetas` para obtener el estado de la tarjeta (activa o inactiva) y se filtra por el `ID_Usuario` del residente. Los resultados se ordenan por la fecha y hora del acceso, mostrando las entradas y salidas.

Implementación en C# (ASP.NET)

```
csharp
CopiarEditar
public DataTable ObtenerHistorialAccesos(int idResidente)
{
    ConexionBD conexion = new ConexionBD();
    MySqlDataAdapter da = new MySqlDataAdapter();
    DataTable dt = new DataTable();
    try
    {
        conexion.AbrirConexion();
        string query = "SELECT a.ID_Acceso, a.Fecha_Hora, a.Tipo,
t.Estado AS Estado_Tarjeta " +
                       "FROM accesos a " +
                       "JOIN tarjetas t ON a.ID_Tarjeta =
t.ID_Tarjeta " +
                       "WHERE t.ID_Usuario = @idResidente " +
                       "ORDER BY a.Fecha_Hora DESC;";
        MySqlCommand cmd = new MySqlCommand(query,
conexion.conectar);
        cmd.Parameters.AddWithValue("@idResidente", idResidente);
```

```
        da.SelectCommand = cmd;
        da.Fill(dt);
    }
    catch (Exception ex)
    {
        Console.WriteLine("Error al obtener el historial de accesos:
" + ex.Message);
    }
    finally
    {
        conexion.CerrarConexion();
    }
    return dt;
}
```

- **Explicación:** Esta función en C# obtiene el historial de accesos de un residente al ejecutar la consulta SQL y devuelve un `DataTable` con los resultados. La función se encarga de abrir y cerrar la conexión a la base de datos y manejar excepciones.

9. Consideraciones Finales

A continuación, se presentan algunas sugerencias para futuras mejoras del sistema, así como una reflexión sobre su impacto en la gestión del conjunto residencial.

Mejoras Futuras

El sistema desarrollado cubre las funciones esenciales de gestión de accesos, pagos, reservas y visitas en un conjunto residencial. Sin embargo, siempre existen oportunidades para mejorar y expandir su funcionalidad. A continuación, se detallan algunas sugerencias para futuras mejoras:

1. **Integración con un Sistema de Notificaciones**:
    a. Implementar un sistema de notificaciones vía correo electrónico o mensajes dentro de la aplicación para alertar a los residentes y administradores sobre eventos importantes como pagos pendientes, reservas confirmadas o cambios en el estado de las zonas comunes.
2. **Mejoras en la Seguridad**:
    a. Aunque el sistema actual tiene una base sólida, se recomienda implementar protección contra inyección SQL, que aunque no está

activada en la versión actual, es una práctica crucial para prevenir vulnerabilidades.
    b. Además, se podrían agregar funcionalidades como autenticación de dos factores para los administradores, aumentando así la seguridad de la gestión del sistema.
3. **Ampliación de la Funcionalidad de las Reservas**:
    a. Se podría permitir a los residentes ver un calendario interactivo con la disponibilidad de las zonas comunes. Esto les permitiría hacer reservas de manera más visual y fácil.
    b. Además, sería útil poder incluir más detalles en las reservas, como la duración de la reserva y la cantidad de personas que podrán acceder a la zona común.
4. **Módulo de Reportes Avanzados**:
    a. Desarrollar un módulo de reportes que permita a los administradores generar informes detallados sobre accesos, pagos y reservas. Esto podría incluir filtros por fecha, estado de los pagos, y zonas comunes, proporcionando análisis más completos.
5. **Gestión de Incidencias y Solicitudes**:
    a. Introducir un módulo que permita a los residentes registrar incidencias o solicitudes (por ejemplo, reparaciones en las zonas comunes o mantenimiento en el conjunto residencial) y hacer seguimiento de su estado.
6. **Optimización del Rendimiento**:
    a. A medida que el sistema crece y se acumula más información en la base de datos, sería conveniente optimizar las consultas y la estructura de la base de datos para asegurar que el rendimiento se mantenga óptimo, especialmente cuando se manejan grandes volúmenes de datos.

Base de Datos: Estructura y Funcionamiento

La base de datos es el corazón del sistema de gestión de acceso, pagos y reservas, ya que almacena toda la información relevante sobre residentes, tarjetas RFID, pagos, reservas, accesos y visitas. El diseño adecuado de la base de datos permite que el sistema funcione de manera eficiente y que los datos estén bien organizados para facilitar consultas y operaciones.

Estructura de la Base de Datos

El sistema utiliza una base de datos **MySQL** llamada **`portal_molinos`**, que contiene varias tablas interrelacionadas:

- **`usuarios`**: Contiene la información de los residentes y administradores, como nombre, correo, teléfono y rol. Está relacionada con otras tablas como `tarjetas`, `accesos` y `pagos`.
- **`tarjetas`**: Almacena la información de las tarjetas RFID utilizadas por los residentes para el acceso al conjunto residencial. Cada tarjeta tiene un estado (activa/inactiva) y está asociada con un residente.
- **`accesos`**: Registra todos los accesos de los residentes, incluyendo el tipo de acceso (entrada o salida), la fecha y la hora. Cada acceso está vinculado a una tarjeta RFID.
- **`reservas`**: Contiene las reservas de zonas comunes realizadas por los residentes. Cada reserva está asociada con un residente y una zona común específica.
- **`pagos`**: Gestiona los pagos realizados por los residentes, indicando el monto, la fecha y el estado del pago (pendiente o pagado). Los pagos están asociados con los residentes y las reservas.
- **`visitantes`**: Registra los visitantes de los residentes, incluyendo el nombre, documento y motivo de la visita. También se registra el estado de la visita (pendiente o aprobada).
- **`zonascomunes`**: Almacena las zonas comunes del conjunto residencial, como salones, jardines, etc., junto con su capacidad y estado (disponible o ocupada).

Relaciones entre las Tablas

Las tablas están relacionadas mediante claves foráneas que garantizan la integridad referencial:

- **Usuarios a Tarjetas**: Un residente puede tener una tarjeta RFID asociada, lo que facilita el control de acceso.
- **Accesos a Tarjetas**: Cada registro de acceso está vinculado a una tarjeta RFID específica, lo que permite saber qué residente ha accedido y cuándo.
- **Reservas a Usuarios y Zonas Comunes**: Las reservas de zonas comunes están asociadas tanto al residente que realiza la reserva como a la zona común reservada.
- **Pagos a Usuarios y Reservas**: Los pagos están vinculados con los residentes y las reservas realizadas, garantizando que el pago se asocie correctamente con los servicios utilizados.
- **Visitantes a Usuarios**: Cada visitante está vinculado a un residente específico, lo que permite gestionar y controlar las visitas de manera efectiva.

Impacto en el Sistema

La correcta estructuración de la base de datos impacta directamente en el rendimiento y la eficiencia del sistema. Permite realizar consultas rápidas y precisas, facilita la actualización de datos (como cambios en el estado de un pago o en la disponibilidad de una zona común) y asegura la integridad de la información mediante relaciones entre las tablas. Sin una base de datos bien estructurada, el sistema podría experimentar fallos de rendimiento, dificultades en la actualización de datos y posibles incoherencias en la información almacenada.

Con esta adición, ahora se resalta la importancia de la base de datos y su impacto directo en el funcionamiento del sistema.

Hardware (Arduino)

Este código permite leer tarjetas RFID utilizando el módulo MFRC522 y una placa compatible con Arduino. A continuación, se detalla el funcionamiento del programa:

1. Inclusión de la Librería

#include <MFRC522.h>

Se incluye la librería **MFRC522**, necesaria para manejar el lector RFID.

2. Definición de Pines

#define RST_PIN 9 // Pin de reset para el MFRC522

#define SS_PIN 10 // Pin de selección de esclavo (SDA)

Se definen los pines de conexión entre el Arduino y el lector RFID:

- **RST_PIN**: Controla el reinicio del lector.
- **SS_PIN**: Pin de selección de esclavo para la comunicación SPI.

3. Creación de la Instancia del Lector

MFRC522 mfrc522(SS_PIN, RST_PIN);

Se crea una instancia del objeto **mfrc522** para interactuar con el lector RFID.

4. Configuración en el setup()

```
void setup() {
  Serial.begin(9600);     // Inicia la comunicación serial
  SPI.begin();            // Inicia el bus SPI
```

```
  mfrc522.PCD_Init();     // Inicializa el lector RFID
  Serial.println("Esperando por una tarjeta RFID...");
}
```

- Se inicia la comunicación serial a **9600 baudios**.
- Se inicia el bus **SPI**, necesario para la comunicación con el lector.
- Se inicializa el lector RFID mediante **mfrc522.PCD_Init()**.
- Se muestra un mensaje en el monitor serial indicando que el lector está listo.

5. Lógica en el loop()

```
void loop() {
  // Verifica si hay una tarjeta presente
  if (!mfrc522.PICC_IsNewCardPresent()) {
    return;
  }
```

Se verifica si hay una tarjeta RFID cerca del lector. Si no hay, el programa regresa al inicio del loop().

```
  // Lee la tarjeta RFID
  if (!mfrc522.PICC_ReadCardSerial()) {
    return;
  }
```

Si hay una tarjeta, se intenta leer su identificador. Si la lectura falla, se vuelve al inicio del loop().

```
  // Obtén el ID de la tarjeta
  String idTarjeta = "";
  for (byte i = 0; i < mfrc522.uid.size; i++) {
    idTarjeta += String(mfrc522.uid.uidByte[i], HEX);
  }
```

El ID de la tarjeta se almacena en una cadena de caracteres **idTarjeta**, convirtiendo cada byte del UID en una representación hexadecimal.

```
  // Imprime el ID de la tarjeta en el monitor serial
  Serial.print("ID de la tarjeta: ");
```

```
  Serial.println(idTarjeta);
```

Se muestra el ID de la tarjeta en el monitor serial.

```
  // Manda datos al proyecto
  Serial.print("Tarjeta: ");
  Serial.println(idTarjeta);
  delay(1000); // Espera 1 segundo antes de leer la siguiente tarjeta
}
```

Se imprime nuevamente el ID en el monitor serial y se espera **1 segundo** antes de volver a leer otra tarjeta, evitando lecturas repetitivas inmediatas.

Conclusión

Este código permite identificar tarjetas RFID y mostrar sus identificadores únicos en el monitor serial. Es una base para proyectos de control de acceso o identificación con RFID.

Diagrama de casos de uso

Actores del Sistema:

Administrador: Gestiona usuarios, tarjetas, accesos, pagos y zonas comunes.
Residente: Consulta pagos, accede al conjunto, reserva zonas comunes y registra visitantes.
Visitante: Solicita acceso al conjunto.

Casos de Uso y su Explicación:

1. Gestión de Usuarios
Registrar usuario: Un administrador puede registrar nuevos residentes.
Actualizar información: Los residentes pueden actualizar su información de contacto.
Eliminar usuario: Un administrador puede eliminar usuarios inactivos.

2. Control de Accesos
Registrar acceso: Se almacena la entrada o salida de un residente usando una tarjeta RFID.
Consultar historial de accesos: Los residentes pueden ver su historial de entradas y salidas.
Bloquear tarjeta: Un administrador puede inactivar una tarjeta RFID en caso de pérdida.

3. Gestión de Pagos
Registrar pago: Un residente realiza un pago que se almacena en la base de datos.
Consultar estado de pago: Un residente puede ver si tiene pagos pendientes o completados.
Actualizar estado de pago: Un administrador puede cambiar el estado de un pago cuando se verifica.

4. Reserva de Zonas Comunes
Solicitar reserva: Un residente solicita el uso de una zona común.

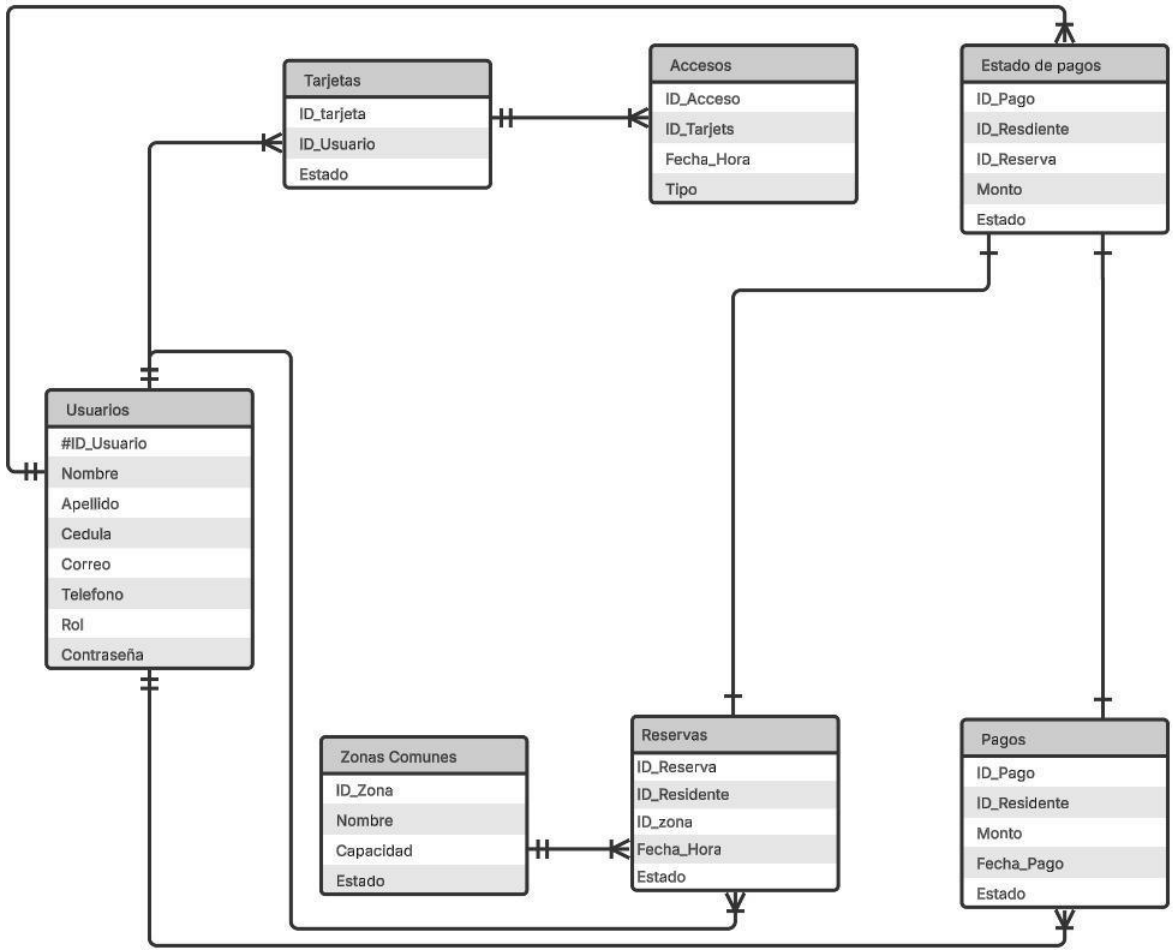
Cancelar reserva: Un residente puede cancelar su reserva antes de la fecha programada.
Administrar disponibilidad: Un administrador puede cambiar el estado de una zona común.

5. Gestión de Visitantes
Registrar visitante: Un residente puede registrar un visitante en el sistema.
Aprobar o rechazar visitante: Un administrador puede aprobar o rechazar la visita.
Consultar visitantes registrados: Un residente puede revisar la lista de visitantes que ha registrado.

Relación con la Base de Datos:

La tabla usuarios almacena los datos de residentes y administradores.
La tabla tarjetas gestiona las tarjetas RFID y su estado.
La tabla accesos almacena los registros de entrada y salida.
La tabla pagos y estado_pagos gestionan los pagos de residentes.
La tabla reservas controla las reservas de zonas comunes.
La tabla visitante almacena la información de los visitantes registrados.
La tabla zonascomunes define las zonas disponibles para reserva.
Este sistema permite la administración eficiente de accesos, pagos y reservas dentro de un conjunto residencial, asegurando seguridad y control para residentes y administradores.

Diagrama entidad relación:



Entidades Principales y sus Relaciones

1. **Usuarios** (usuarios)

    a. Representa a los residentes y administradores del sistema.
    b. Atributos: ID_Usuario, Nombre, Apellido, Cedula, Correo, Telefono, Rol, Contraseña.
    c. Relación:
        i. Un usuario puede tener **una o más tarjetas RFID** (relación con tarjetas).
        ii. Un usuario puede realizar **pagos** (relación con pagos).
        iii. Un usuario puede **registrar visitantes** (relación con visitantes).
        iv. Un usuario puede hacer **reservas** de zonas comunes (relación con reservas).

2. **Tarjetas RFID** (tarjetas)

    a. Representa las tarjetas asignadas a los residentes para el control de accesos.
    b. Atributos: ID_Tarjeta, ID_Usuario, Estado (Activa/Inactiva).
    c. Relación:
        i. Cada tarjeta pertenece a **un usuario** (usuarios).
        ii. Se utiliza para **registrar accesos** (relación con accesos).

3. **Accesos** (accesos)

    a. Registra los accesos de los residentes al conjunto residencial.
    b. Atributos: ID_Acceso, ID_Tarjeta, Fecha_Hora, Tipo (Entrada/Salida).

c. Relación:
    i. Cada acceso está vinculado a **una tarjeta** (tarjetas).

4. **Pagos** (pagos)

    a. Almacena los pagos realizados por los residentes.
    b. Atributos: ID_Pago, ID_Residente, Monto, Fecha_Pago, Estado (Pendiente/Pagado).
    c. Relación:
        i. Cada pago pertenece a **un usuario residente** (usuarios).

5. **Reservas de Zonas Comunes** (reservas)

    a. Guarda las reservas de los residentes para el uso de zonas comunes.
    b. Atributos: ID_Reserva, ID_Residente, ID_Zona, Fecha_Hora, Estado (Confirmada/Cancelada).
    c. Relación:
        i. Una reserva está asociada a **un residente** (usuarios).
        ii. Se asocia a **una zona común** (zonascomunes).

6. **Zonas Comunes** (zonascomunes)

    a. Representa las áreas comunes del conjunto residencial.
    b. Atributos: ID_Zona, Nombre, Capacidad, Estado (Disponible/Ocupada).
    c. Relación:
        i. Puede ser reservada por **varios residentes** (reservas).

7. **Visitantes** (visitantes)

    a. Registra a los visitantes del conjunto residencial.
    b. Atributos: ID_Visitante, Nombre, Documento, ID_Residente, Motivo_Visita, Estado (Pendiente/Aprobado).
    c. Relación:
        i. Cada visitante es asociado a **un residente** (usuarios).

Resumen de Relaciones Clave

Entidad	Relación	Entidad Relacionada
usuarios	1 a M	tarjetas
usuarios	1 a M	pagos
usuarios	1 a M	reservas
usuarios	1 a M	visitantes
tarjetas	1 a M	accesos
reservas	M a 1	zonascomunes
pagos	1 a 1	usuarios