**Sistema Web para La Reserva de Canchas Sintéticas**

Agosto 23,2025

Índice

[Introducción 5](#_Toc204682404)

[Contexto y Antecedentes 5](#_Toc204682405)

[Descripción de la Unidad de Estudio 5](#_Toc204682406)

[Diagnóstico de la Situación Actual 6](#_Toc204682407)

[Problema de Investigación 8](#_Toc204682408)

[Enunciado del Problema 8](#_Toc204682409)

[Descripción del problema 8](#_Toc204682410)

[Justificación 9](#_Toc204682411)

[Objetivos 9](#_Toc204682412)

[Objetivo General 9](#_Toc204682413)

[Objetivo Especifico 9](#_Toc204682414)

[Hipótesis 10](#_Toc204682415)

[Marco Teórico 11](#_Toc204682416)

[Sistemas de información 11](#_Toc204682417)

[Aplicación Web 11](#_Toc204682418)

[Análisis de Requerimiento 11](#_Toc204682419)

[Figma y Diseño de Interfaces 12](#_Toc204682420)

[Modelo de Capas 12](#_Toc204682421)

[Lenguaje de Programación y Herramientas 13](#_Toc204682422)

[Seguridad y Validación de Contraseñas con ASP.NET Core Identity 13](#_Toc204682423)

[Base de Datos Relacional (SQL Server) 13](#_Toc204682424)

[Diagramas UML 14](#_Toc204682425)

[Data Annotations 14](#_Toc204682426)

[Librerías Utilizadas 15](#_Toc204682427)

[Metodología 15](#_Toc204682428)

[Análisis de Requerimientos 17](#_Toc204682429)

[Requerimientos Funcionales 18](#_Toc204682430)

[REQUERIMIENTOS NO FUNCIONALES 20](#_Toc204682431)

[Diseño del Sistema 21](#_Toc204682432)

[Caso de Uso 21](#_Toc204682433)

[Especificación de Casos de Uso 22](#_Toc204682434)

[Diagramas de Clases 30](#_Toc204682435)

[Diagramas de Secuencia 31](#_Toc204682436)

[Interfaces 37](#_Toc204682437)

[Arquitectura 48](#_Toc204682438)

[Desarrollo 52](#_Toc204682439)

[Desarrollo e Implementación 54](#_Toc204682440)

[Desarrollo del Módulo 54](#_Toc204682441)

[Tecnologías Utilizadas 54](#_Toc204682442)

[Estructura del Proyecto 55](#_Toc204682443)

[Conclusiones 58](#_Toc204682444)

[Recomendaciones 60](#_Toc204682445)

[Bibliografía 62](#_Toc204682446)

# Introducción

## Contexto y Antecedentes

### Descripción de la Unidad de Estudio

El objetivo de este proyecto es la Empresa 'El Relámpago', una empresa cuyo número de RUC es 10105061256 y que se enfoca en alquilar sus propios campos de fútbol de Grass sintético para la práctica amateur del juego.

Este negocio está situado en el distrito de Miraflores, específicamente en la calle teniente Rodríguez No. 206 Miraflores - Arequipa, y proporciona espacios deportivos equipados para partidos amistosos, entretenimiento entre amigos y ejercicio cotidiano no solo a sus propios clientes sino también a otros que realicen actividad deportiva.

Un total de 3 campos frecuentemente ventilados, todos completamente sintéticos, se alquilarán por hora a individuos, grupos de amigos o equipos deportivos. El servicio incluye el uso de infraestructura deportiva, iluminación nocturna, acceso a vestuarios u otros servicios complementarios.

La empresa, gracias a su ubicación estratégica y servicio de calidad, se ha establecido como la opción preferida para los entusiastas del fútbol que requieren espacios cómodos y accesibles para sus actividades deportivas recreativas.

**Misión**

Proporcionar espacios deportivos de calidad para la práctica de fútbol amateur, y al mismo tiempo promover la recreación, el bienestar y la integración social a través de un servicio eficiente pero económico.

**Visión**

Ser la empresa líder en el alquiler de canchas de fútbol artificial en Arequipa, famosa por su excelencia en el servicio, innovación en infraestructura y dedicación al deporte y las personas.

### Diagnóstico de la Situación Actual

En la actualidad, El Relámpago maneja las reservas de sus campos de césped artificial de manera no adecuada. No hay automatización ni especialización involucrada. Por lo tanto, la empresa depende del trabajo manual y herramientas básicas para manejar la atención al cliente, lo que lleva a una gestión desorganizada, retrasos en las respuestas que insatisfacen a los clientes y, por lo tanto, el orden de eficiencia no es alto en absoluto.

La demanda de espacios deportivos está en constante aumento, pero la tecnología disponible no parece adecuada; por lo tanto, se hace necesario modernizar el proceso de reservas y también mejorar la experiencia del usuario desde ambos lados, tanto del cliente como del administrativo.

**Situación y Proceso Existente**

En la actualidad, las reservas de espacios deportivos se realizan principalmente a través de llamadas telefónicas o redes sociales, en canales de WhatsApp. Una vez que el cliente solicita un horario, la persona a cargo simplemente registra la información de la reserva en papel (nombre del cliente, fecha y hora de la reserva). Después de anotar la reserva en el papel es fotografiada y enviada a un grupo de WhatsApp donde todas las personas involucradas en la realización de reservas incluidos trabajadores y administradores, pueden verla las reservas realizadas lo cual es necesario una inspección manual de la imagen y verificación del estado del horario disponibles y ocupados de las canchas sintéticas.

Esto es engorroso y lleva a retrasos en el manejo de reservas, la replicación de información y la falta de sincronización contribuyen a más errores. Además, no hay un archivo centralizado o base de datos digital que pueda usarse para consultar el historial de reservas.

**Deficiencias**

* Ausencia de un sistema centralizado para el registro y control de reservas.
* Falta de visibilidad en tiempo real para ver la disponibilidad de las canchas sintéticas.
* Riegos de errores humanos por el uso de formatos manuales y la comunicación informar que pueden incluir con errores.
* Demoras en la atención en el cliente debido a la verificación manual de la información de reservas disponibles.
* Poca trazabilidad y control sobre las actividades realizadas por el personal encargado en la realización de reservas.

**Oportunidad de Mejora**

La implementación de un sistema web especializado para la gestión de reservas de canchas sintéticas, representa una oportunidad significativa, para optimizar los procesos actuales de la empresa El Relámpago Esta solución permitiría el registro y consulta en tiempo real de las reservas, mejoraría el servicio al cliente y reduciría los errores operativos. Además, traería un mayor control y trazabilidad sobre las operaciones, beneficiando tanto al equipo administrativo como a los clientes.

# Problema de Investigación

## Enunciado del Problema

La empresa El Relámpago, ubicada en el distrito de Miraflores de Arequipa, no gestiona ni procesa eficientemente las reservas de sus campos de fútbol sintéticos.

## Descripción del problema

El Relámpago, dedicada al alquiler de campos sintéticos para fútbol amateur, gestiona sus reservas de manera manual. Ha utilizado el medio de comunicación de WhatsApp para ver las reservas disponibles del día. Sin embargo, este enfoque no estándar ha resultado en múltiples deficiencias en el proceso de atención al cliente. Errores en la asignación de reservas, retrasos en la atención del cliente, reservas conflictivas, en resumen, falta de orden. Esto significa que desde la información guardad en hojas y en fotografías es real mente ineficiente para la gestión de reservas.

Los métodos informales de la empresa sobre cómo asigna el uso de los campos resultan en muchas ineficiencias. Al no haber un sistema especialmente diseñado, la empresa no puede monitorear eficientemente las reservas. Más importante aún, esta falta de un sistema ha causado problemas para verificar la disponibilidad en tiempo real y hace que actualizar la información a través de grupos de WhatsApp sea bastante complejo para los trabajadores que también deben tomar fotografías. No es sorprendente que esto aumente las posibilidades de errores humanos.

Estas limitaciones, tanto en la experiencia del cliente como en la operación de la empresa, muestran la necesidad de una solución tecnológica. Esto optimizará el proceso de reservas de El Relámpago, mejorando la organización interna.

## Justificación

El desarrollo de un sistema web para la gestión de reservas permitirá a la empresa El Relámpago modernizar su proceso de atención al cliente, automatizar los registros de horarios y reducir los errores asociados con el manejo manual. Además, proporcionará al personal una herramienta centralizada y accesible para gestionar los campos. Los administradores o trabajadores pueden ir directamente a realizar las reservas en tiempo real desde sus teléfonos o desde una computadora de escritorio.

Esta solución tecnológica también contribuirá a una mejor toma de decisiones al permitir la generación de informes sobre la demanda de reservas, la utilización de la infraestructura y el comportamiento del cliente. Se espera que la eficiencia operativa pueda mejorarse solo para aumentar la satisfacción del cliente y la competitividad empresarial en el mundo actual donde todos los servicios se están digitalizando.

# Objetivos

## Objetivo General

Crear un sistema web que optimice y agilice la gestión de reservas de campos sintéticos para una empresa llamada El Relámpago en el distrito de Miraflores.

## Objetivo Especifico

1. Analizar las deficiencias del proceso actual de reservas y determinar los requerimientos del sistema.
2. Diseñar una solución web que permita registrar, gestionar y consultar las reservas en tiempo real.
3. Implementar un sistema web funcional, accesible y adaptable a las necesidades del negocio.
4. Validar y probar el sistema mediante simulaciones controladas en un entorno local, utilizando escenarios representativos que permitan verificar su correcto funcionamiento antes de una posible implementación real.

## Hipótesis

Un sistema de reservas avanzado basado en la web mejorará significativamente la efectividad de la gestión de campos sintéticos en El Relámpago, ya sea reduciendo errores, tiempos de respuesta o mejorando la satisfacción del cliente.

# Marco Teórico

## Sistemas de información

Los sistemas de información son "herramientas" que poseen funciones de recopilación, almacenamiento, procesamiento y distribución de información, con el fin de apoyar y mejorar las operaciones, la gestión y el sistema de toma de decisiones dentro de una organización (Laudon y Laudon, 2020). Para El Relámpago, el sistema web con sus funciones de gestión de reservas de campos sintéticos reemplazará los procesos manuales tradicionales, propensos a errores y desorden, para no solo mejorar la productividad sino también la satisfacción del usuario.

## Aplicación Web

Una aplicación web es un tipo de software que se ejecuta desde un servidor y se accede a través de un navegador web, lo que significa que cualquier dispositivo conectado a internet puede ser utilizado sin necesidad de instalación local (Sommerville, 2016). Para este proyecto, se desarrollará una aplicación web. La aplicación web podrá ser utilizada por administradores y empleados para realizar reservas y verificar rápidamente si hay franjas horarias disponibles para nuevas reservas.

## Análisis de Requerimiento

El análisis de requisitos es la primera de las etapas del desarrollo de software, en la que se identifican en detalle las necesidades y expectativas que un cliente o usuario final obtendrá posteriormente (Pressman, 2014). Mediante técnicas como reuniones, observación y examen de procedimientos existentes para funciones de alto nivel, se definen tanto los requisitos funcionales (lo que el sistema debe hacer, como el registro de reservas) como los requisitos no funcionales (en aspectos como rendimiento, seguridad, escalabilidad). Es esencial que este proceso se complete adecuadamente si se esperan obtener resultados esperados de las soluciones de aplicación.

## Figma y Diseño de Interfaces

Para generar diseño visual y experiencia de usuario, se ha empleado Figma como una herramienta que permite realizar wireframes de diseños de interfaz de manera eficiente. A través de Figma, se crearán maquetas para las pantallas principales del sistema, como inicio de sesión, gestión de reservas o visualización de horarios. Estos prototipos servirán para validar con los usuarios la usabilidad y funcionalidad antes de que comience la implementación real. Además de facilitar la comunicación y entendimiento de cómo debería ser el sistema web de reservas.

## Modelo de Capas

Este sistema adoptará un diagrama en capas, estructurando el código en 4 capas cruciales:

* **Capa de Presentación**: Gestiona la interacción con el usuario a través de formularios web, menús y mensajes.
* **Capa de Client**: Actúa como un intermediario entre las dos partes, gestionando algunos servicios especiales que son necesarios para facilitar la comunicación real entre la capa de presentación y la capa de lógica de negocio.
* **Capa de Servicios**: Implementa las reglas y procesos del negocio, como la gestión de reservas, conflictos de reservas, disponibilidad de reservas entre otros.
* **Capa Respository**: Se encarga de las operaciones sobre la base de datos, incluyendo guardar, editar, eliminar y consultar reservas.

Esta arquitectura permite separar claramente las responsabilidades, facilitando el mantenimiento, la escalabilidad y la reutilización del código (Sommerville, 2016).

## Lenguaje de Programación y Herramientas

Para el desarrollo del sistema se emplearán:

* Lenguaje: C#
* Framework: ASP.NET Core
* IDE: Visual Studio 2022
* Base de Datos: SQL Server

Estas herramientas permiten crear aplicaciones web modernas, seguras y eficientes, optimizando el proceso de desarrollo y mantenimiento (Microsoft Docs, 2023).

## Seguridad y Validación de Contraseñas con ASP.NET Core Identity

ASP.NET Core Identity es un sistema seguro para la gestión de contraseñas que utiliza algoritmos de hashing criptográfico, como PBKDF2, junto con una sal aleatoria y múltiples iteraciones. Esto impide el almacenamiento de contraseñas en texto plano y protege contra ataques de fuerza bruta, fortaleciendo la seguridad desde el proceso de autenticación (Microsoft Docs, 2023).

## Base de Datos Relacional (SQL Server)

SQL Server es un sistema de gestión de bases de datos relacional que permite almacenar y consultar grandes volúmenes de información de manera estructurada. Será utilizado para registrar reservas, usuarios, pagos y reportes, asegurando la integridad y disponibilidad de los datos (Microsoft Docs, 2023).

## Diagramas UML

Se emplearán diagramas UML para representar gráficamente la estructura y comportamiento del sistema, facilitando su comprensión antes de la programación. Entre ellos se incluyen:

* **Diagrama de casos de uso**: Identifica las funcionalidades principales y la interacción entre los actores y el sistema
* **Diagrama de clases**: Representa la estructura estática del sistema, mostrando las clases, atributos y relaciones
* **Diagrama de secuencia**: Para mostrar el flujo de interacción entre los componentes.

Estos diagramas son fundamentales para la planificación y comunicación efectiva del diseño del software (Ambler, 2004).

## Data Annotations

Para la validación y definición de reglas de negocio en el modelo de datos, se emplean Data Annotations, que son atributos aplicados directamente sobre las propiedades de las clases en C#. Estas anotaciones permiten validar datos de entrada, definir restricciones, formatos, y mensajes de error de forma declarativa, facilitando la integridad y calidad de la información antes de su procesamiento o almacenamiento (Esposito, 2018).

## Librerías Utilizadas

Para mejorar la experiencia de usuario y la funcionalidad del sistema, se integran las siguientes librerías externas:

**Flatpickr**: Es una biblioteca ligera de selección de fechas y horas para interfaces web. Proporciona un calendario intuitivo y configurable que facilita la elección de fechas en los formularios de reserva (Flatpickr, 2023).

* **FullCalendar**: Librería JavaScript que permite mostrar y administrar calendarios interactivos con múltiples vistas (mes, semana, día). Es ideal para visualizar y gestionar la disponibilidad de horarios y reservas en el sistema (FullCalendar, 2023).
* **Web Compiler**: Herramienta que permite compilar archivos Sass (Syntactically Awesome Style Sheets) en CSS, facilitando la organización y mantenimiento de estilos con características avanzadas como variables y anidamiento (Web Compiler, 2023).

Estas librerías contribuyen a crear una interfaz más amigable, dinámica y mantenible.

# 

# Metodología

En este proyecto se desarrollará un sistema web para gestionar las reservas de canchas sintéticas de la empresa El Relámpago. Para identificar las necesidades reales y comprender el proceso actual, se realizó un análisis detallado utilizando las siguientes técnicas de recolección de información:

**Observación directa**:

Se examinó el procedimiento actual de reservas, desde el primer contacto del cliente, la toma de información para las reservas, él envió de fotografía al grupo de WhatsApp para indicar que la cancha en la hora solicitada por el cliente ya está ocupada.

**Entrevistas**:

Se mantuvieron conversaciones con el personal encargado de realizar las reservas, con el objetivo de conocer las dificultades, requerimientos y expectativas en el manejo de las reservas.

**Revisión de documentos**: Se analizaron hojas de reserva manuscritas y mensajes utilizados para confirmar las reservas, con el fin de entender la estructura y el flujo de la información.

Con base en los datos obtenidos, se organizaron y definieron los requerimientos del sistema, los cuales guiaron el diseño y desarrollo del mismo. Para la implementación se utilizarán las siguientes herramientas y tecnologías:

* **Lenguaje de programación**: C#, por su robustez y soporte para desarrollo empresarial.
* **Tecnología web**: ASP.NET Core Web, que permite crear aplicaciones web seguras, escalables y de alto rendimiento.
* **Base de datos:** SQL Server, para el almacenamiento eficiente de la información
* **Entorno de desarrollo**: Visual Studio 2022, que proporciona un conjunto completo de herramientas para codificación, depuración y despliegue.
* **Diseño de pantallas**: Figma, para crear prototipos que mejoran la experiencia de usuario.
* **Diagramas del sistema**: UML, para modelar y comunicar la estructura y funcionamiento.
* **Arquitectura**: Modelo en capas (presentación, lógica de negocio y acceso a datos) para modularidad y mantenibilidad.
* **Seguridad**: ASP.NET Core Identity para gestión segura de usuarios.
* **Validación**: Uso de Data Annotations para asegurar la integridad de los datos desde la capa de presentación y lógica de negocio.
* **Experiencia de usuario**: Integración de librerías como Flatpickr y FullCalendar para facilitar la selección de fechas y visualización de reservas.

Esta combinación tecnológica y metodológica asegura el desarrollo de un sistema eficiente, seguro y amigable para los usuarios, alineado con las necesidades reales del negocio.

## Análisis de Requerimientos

El sistema de reservas será una aplicación web responsiva que facilitará el proceso de reservas de canchas para la empresa El Relámpago, brindando a los usuarios una experiencia simple, intuitiva y accesible desde cualquier dispositivo, ya sea desde computadoras de escritorio y celulares.

### Requerimientos Funcionales

El sistema contará con dos tipos de usuarios:

**Administrador:**

* Tiene acceso total al sistema.
* Puede realizar tareas de gestión general y configuración.

**Trabajador:**

* Accede solo a los módulos permitidos según los permisos otorgados por el administrador.
* Realiza tareas operativas relacionadas a reservas, pagos y egresos.

**Roles y Funcionalidades**

**Funciones del Administrador**

* Crear, editar y eliminar usuarios del sistema.
* Gestionar los permisos de los usuarios del sistema (asignar, modificar y revocar).
* Gestionar las canchas, incluyendo el registro, edición y eliminación de canchas.
* Gestionar medios de pago (yape, plin, al contado, etc.)
* Visualizar los siguientes reportes:
* Ingresos diarios, semanales y mensuales.
* Ganancias generadas por cada cancha.

**Funciones del Trabajador y también del administrador con permisos básicos**

* Registrar, editar o eliminar reservas.
* Registrar reservas con pagos parciales o pagos completos.

**Funcionalidades del Sistema**

**1. Validación de Disponibilidad en Tiempo Real**

* Antes de confirmar una reserva, el sistema verifica automáticamente si el horario seleccionado está disponible.
* En caso de conflicto con otra reserva ya existente, se mostrará una advertencia y no se permitirá guardar la nueva reserva.

**2. Registro de Pagos Parciales y Cálculo de Saldos**

* Al registrar una reserva con pago parcial, el sistema calcula automáticamente el saldo pendiente.
* La reserva quedará con estado "Parcial" (color naranja).
* Una vez completado el pago, el estado cambiará a "Pagado" (color verde) de forma automática.

**3. Control de Acceso mediante Login**

* Solo usuarios registrados como trabajadores o administradores podrán ingresar al sistema.
* El acceso se realiza a través de una pantalla de inicio de sesión (login) con credenciales válidas (correo y contraseña).

### REQUERIMIENTOS NO FUNCIONALES

**Disponibilidad del sistema**

El sistema debe estar disponible al menos el 99.5% del tiempo, especialmente en horarios de atención al público (9:00 a.m. – 11:00 p.m.).

**Seguridad**

El acceso al panel de gestión está protegido mediante autenticación con credenciales seguras, gestionadas a través de **ASP.NET Core Identity**. Las contraseñas se almacenan utilizando algoritmos de **hashing criptográfico con sal aleatoria**, cumpliendo con estándares modernos de seguridad. Este enfoque evita el almacenamiento de contraseñas en texto plano y protege contra ataques como fuerza bruta y rainbow tables.

**Usabilidad**

El sistema debe permitir que un nuevo usuario pueda consultar la disponibilidad y hacer una reserva en menos de 3 minutos, sin capacitación previa.

**Rendimiento**

Las acciones principales del sistema (consultar disponibilidad, registrar reserva) deben tener un tiempo de respuesta inferior a 30 segundos.

**Compatibilidad**

El sistema debe ser compatible con los navegadores web más utilizados (Chrome, Firefox, Brave) y adaptarse correctamente a dispositivos móviles.

# Diseño del Sistema

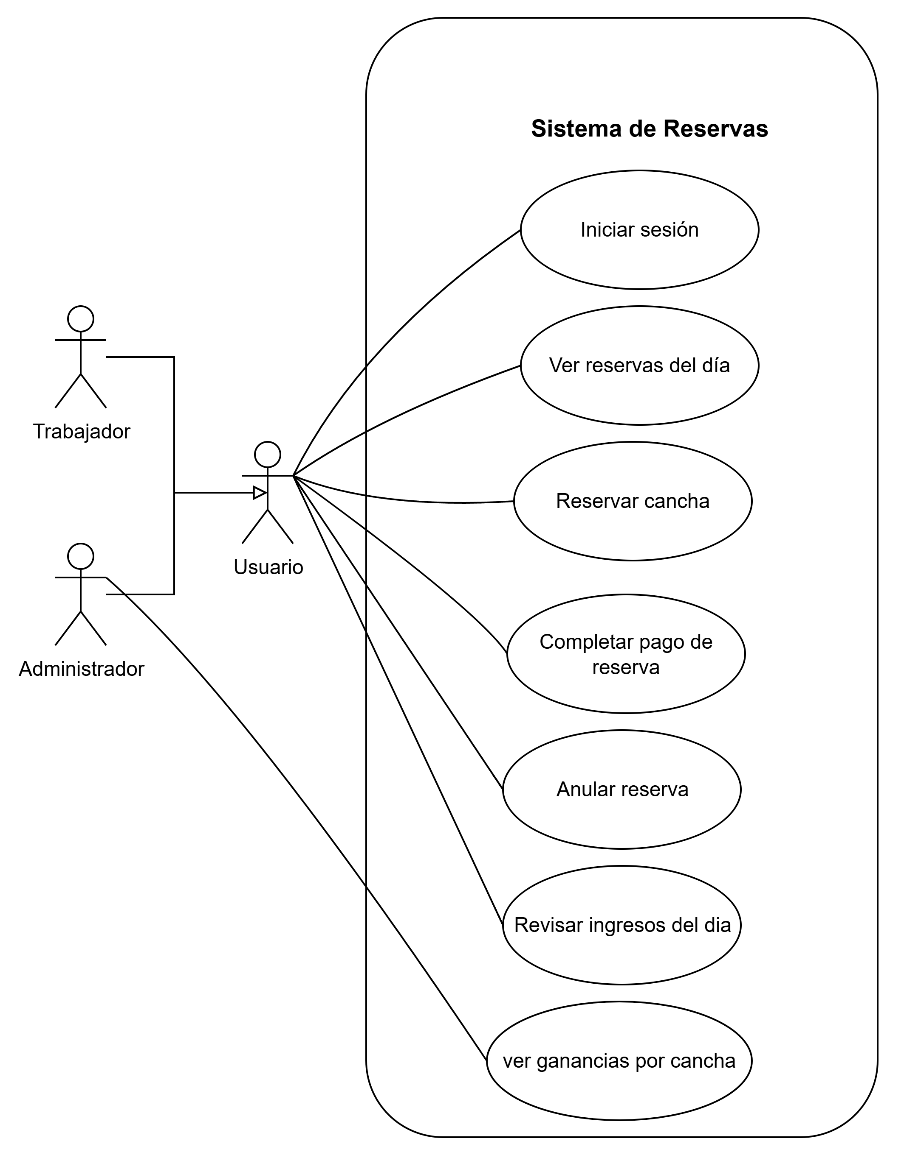
## Caso de Uso

La Figura 1 muestra los principales casos de uso del sistema de reservas, diferenciando entre los roles de administrador y trabajador.

El administrador tiene acceso total, mientras que el trabajador solo accede a funciones operativas según permisos asignados.

**Figura 1**

#### Diagrama de casos de uso del sistema de reservas de canchas



## Especificación de Casos de Uso

A continuación, se presentan las especificaciones detalladas de cada uno de los casos de uso identificados para el sistema de reservas de canchas de la empresa El Relámpago. Estas especificaciones describen de manera clara y estructurada la interacción entre los usuarios (administrador y trabajador) y el sistema, incluyendo los flujos principales de eventos, precondiciones, postcondiciones y posibles excepciones.

La documentación precisa de estos casos de uso es fundamental para guiar el diseño, implementación y validación del sistema, asegurando que cada funcionalidad requerida esté correctamente definida y alineada con los objetivos establecidos por el cliente.

**Tabla 1***Especificación del caso de uso: Iniciar sesión*

|  |  |
| --- | --- |
| **Elemento** | **Descripción** |
| **ID** | UC01 |
| **Nombre** | Iniciar sesión |
| **Breve Descripción** | Permite a un usuario autenticarse en el sistema ingresando su correo electrónico y contraseña. |
| **Actores principales** | Usuario |
| **Actores secundarios** | Ninguno |
| **Precondiciones** | 1. El usuario ya debe estar registrado en el sistema. 2. El usuario debe tener su cuenta activa (no desactivada por el administrador). |
| **Flujo Principal** | 1. El usuario accede a la pantalla de inicio de sesión. 2. El sistema solicita correo y contraseña. 3. El usuario ingresa sus credenciales. 4. El sistema verifica los datos. 5. El usuario accede al sistema. |
| **Postcondiciones** | El usuario queda autenticado y puede utilizar el sistema según los permisos asignados |
| **Flujo alternativo** | **Credenciales incorrectas**: El sistema muestra mensaje de Credenciales incorrectas.  **Usuario desactivado**: El sistema muestra mensaje 'Su cuenta ha sido desactivada. Contacte al administrador. |

**Tabla 2***Especificación del Caso de Uso – Ver Reservas del Día*

|  |  |
| --- | --- |
| **Elemento** | **Descripción** |
| **ID** | UC02 |
| **Nombre** | Ver Reservas del Día |
| **Breve Descripción** | Permite al usuario consultar todas las reservas registradas para el día actual, organizadas por cancha y horario, a través de una vista tipo calendario. |
| **Actores principales** | Usuario |
| **Actores secundarios** | Cliente |
| **Precondiciones** | 1. El usuario debe estar autenticado en el sistema. 2. Debe haber reservas registradas para el día actual (de lo contrario, la lista estará vacía). |
| **Flujo Principal** | 1. El usuario accede al módulo de reservas. 2. El sistema muestra por defecto la vista del día actual en formato calendario. 3. Para cada reserva, se muestra:  * Nombre del cliente * Hora de inicio y fin * Estado de pago (parcial o completo, con color indicador) * Monto pagado / saldo  1. El usuario puede desplazarse por fecha. 2. El sistema actualiza automáticamente si hay nuevos registros o cambios. |
| **Postcondiciones** | 1. El usuario puede visualizar todas las reservas activas del día. 2. Puede detectar reservas pendientes de cobro o vacíos de horario para nuevas reservas. |
| **Flujo alternativo** | Se permite navegar a otros días o crear una nueva reserva |

**Tabla 3***Especificación del Caso de Uso – Reservar Cancha*

|  |  |
| --- | --- |
| **Elemento** | **Descripción** |
| **ID** | UC03 |
| **Nombre** | Reservar Cancha |
| **Breve Descripción** | Permite al usuario registrar una reserva de cancha para un cliente en una fecha y hora específica. Incluye la selección del medio de pago, tipo de pago (parcial o completo), cálculo de saldo y estado visual en calendario. |
| **Actores principales** | Usuario |
| **Actores secundarios** | Cliente |
| **Precondiciones** | 1. El usuario debe estar autenticado. 2. Debe existir al menos una cancha registrada. 3. Deben existir medios de pago configurados. |
| **Flujo Principal** | 1. El usuario accede al módulo de reservas. 2. El sistema muestra la vista de calendario con las canchas y horarios disponibles. 3. El usuario presiona el botón Nueva reserva. 4. El sistema presenta un formulario para completar la reserva. 5. El usuario ingresa los datos requeridos (nombre del cliente, fecha, hora inicio y fin, medio de pago, tipo de pago (parcial o completo), cancha, teléfono, monto total, monto pagado si es parcial). 6. El sistema valida la disponibilidad de la cancha para la fecha y hora seleccionadas. 7. Si la cancha está disponible, el sistema registra la reserva y asigna el estado correspondiente (pagado / pendiente) con su color indicativo 8. El sistema actualiza automáticamente la vista de calendario con la nueva reserva registrada |
| **Postcondiciones** | 1. La reserva se registra y se muestra en el calendario con su estado visual. 2. Se almacenan todos los datos de la reserva. |
| **Flujo alternativo** | 1. Si la cancha está ocupada, el sistema muestra un mensaje de error. 2. El usuario puede proponer otro horario. 3. Si hay error posterior, se puede editar desde la lista de reservas. |

**Tabla 4***Especificación del Caso de Uso – Completar pago de reserva*

|  |  |
| --- | --- |
| **Elemento** | **Descripción** |
| **ID** | UC04 |
| **Nombre** | Completar pago de reserva |
| **Breve Descripción** | Este caso de uso permite al usuario completar el pago de una reserva previamente registrada con un pago parcial. Al registrar el pago restante, el sistema actualiza automáticamente el estado de la reserva a “Pagado” y modifica su representación visual en la grilla de calendario. |
| **Actores principales** | Usuario |
| **Actores secundarios** | Cliente |
| **Precondiciones** | 1. El usuario debe estar autenticado en el sistema. 2. Debe existir al menos una reserva registrada con estado Parcial. |
| **Flujo Principal** | 1. El usuario accede al módulo de reservas. 2. Busca la reserva por fecha. 3. Selecciona una reserva con estado “Parcial” (color naranja). 4. El sistema muestra la información actual de la reserva, incluyendo:  * Nombre del cliente * Fecha (prellenada con la fecha seleccionada) * Hora de inicio y fin * Medio de pago * Tipo de pago (parcial / completo) * Cancha * Monto total * Monto pagado hasta el momento * Saldo pendiente * Teléfono del cliente (opcional)  1. El usuario ingresa el monto del pago adicional o selecciona el tipo de pago completo. 2. El sistema valida que el nuevo pago no exceda el saldo pendiente. 3. Si el monto ingresado es válido:  * Se actualiza el monto pagado. * Se cambia el estado a “Pagado”. * Se actualiza el color de la reserva en la grilla a verde.  1. El sistema guarda el nuevo estado y el registro de pago en la base de datos. |
| **Postcondiciones** | * 1. El estado de la reserva cambia de “Parcial” a “Pagado”, si el monto pagado cubre el total. |
| **Flujo alternativo** | Si el usuario intenta ingresar un monto mayor al saldo pendiente:   * El sistema muestra un mensaje de error indicando que el monto excede el permitido. * No se permite registrar el pago hasta que se corrija el monto ingresado. |

**Tabla 5***Especificación del Caso de Uso – Anular reserva*

|  |  |
| --- | --- |
| **Elemento** | **Descripción** |
| **ID** | UC05 |
| **Nombre** | Anular reserva |
| **Breve Descripción** | Este caso de uso permite al usuario anular una reserva previamente registrada en el sistema. La anulación puede realizarse por errores en la reserva, cambios de decisión del cliente, o motivos operativos. Al anularse, la cancha queda nuevamente disponible para otros usuarios. El sistema registra la anulación y actualiza el estado de la reserva. |
| **Actores principales** | Usuario |
| **Actores secundarios** | Cliente |
| **Precondiciones** | 1. El usuario debe estar autenticado en el sistema. 2. Debe existir una reserva registrada en estado activo |
| **Flujo Principal** | 1. El usuario accede al módulo de reservas. 2. El usuario selecciona la reserva que desea anular. 3. El sistema muestra los detalles de la reserva seleccionada. 4. El usuario presiona el botón Anular. 5. El sistema solicita la confirmación de anulación. 6. El usuario confirma la acción. 7. El sistema actualiza el estado de la reserva a anulado. 8. La cancha queda disponible nuevamente en el calendario. 9. El sistema registra el usuario que realizó la anulación. |
| **Postcondiciones** | 1. La reserva cambia su estado a "Anulada". 2. La cancha queda libre para futuras reservas. 3. Se registra la trazabilidad de la anulación (fecha, usuario). |
| **Flujo alternativo** | El usuario cancela la acción de anulación:   * El sistema no realiza cambios y vuelve a la vista anterior. |

**Tabla 6***Especificación del Caso de Uso – Visualizar ingresos diarios*

|  |  |
| --- | --- |
| **Elemento** | **Descripción** |
| **ID** | UC06 |
| **Nombre** | Visualizar ingresos diarios |
| **Breve Descripción** | Permite al usuario consultar el total de ingresos generados por las reservas en una fecha determinada. Esta funcionalidad es clave para el control financiero diario del negocio. |
| **Actores principales** | Usuario |
| **Actores secundarios** | Ninguno |
| **Precondiciones** | 1. El usuario debe haber iniciado sesión en el sistema. 2. Deben existir reservas con pagos registrados en la fecha seleccionada para la consulta. |
| **Flujo Principal** | 1. El usuario accede al módulo de Ingresos. 2. El sistema muestra por defecto la vista correspondiente al día actual en formato calendario. 3. Para cada reserva del día, el sistema presenta:  * Nombre del cliente * Hora de inicio y fin * Monto pagado * Estado * Usuario que realizo la reserva  1. El usuario puede desplazarse a otras fechas para visualizar ingresos de días anteriores o posteriores. |
| **Postcondiciones** | El usuario obtiene una visión clara de los ingresos generados en la fecha seleccionada |
| **Flujo alternativo** | **No existen ingresos en la fecha seleccionada:**   * El sistema muestra un mensaje informativo: “No se encontraron registros.” * El usuario puede seleccionar otro rango de fechas. |

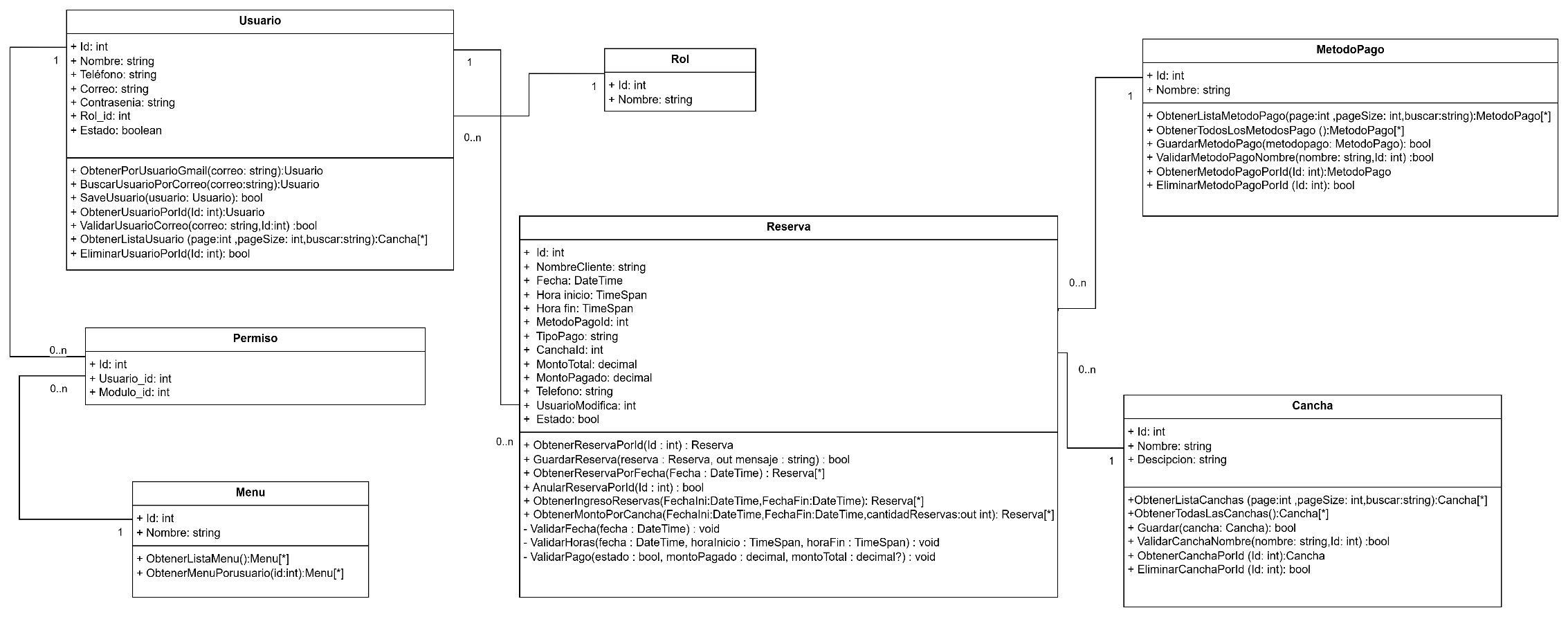
**Tabla 7***Especificación del Caso de Uso – Visualizar ingresos por canchas*

|  |  |
| --- | --- |
| **Elemento** | **Descripción** |
| **ID** | UC07 |
| **Nombre** | Visualizar ingresos por canchas |
| **Breve Descripción** | Permite al administrador consultar un reporte con el total de ingresos generados por cada cancha durante un período de tiempo determinado.  El sistema muestra la suma total recaudada por cancha.  Este reporte permite identificar qué canchas generan más ingresos en el sistema |
| **Actores principales** | Administrador |
| **Actores secundarios** | Ninguno |
| **Precondiciones** | 1. El Administrador debe estar autenticado en el sistema. 2. Deben existir reservas registradas con pagos dentro del período consultado. 3. El sistema debe tener al menos una cancha registrada. |
| **Flujo Principal** | 1. El Administrador ingresa al módulo de Ingresos. 2. Selecciona la opción canchas. 3. El sistema consulta las reservas agrupadas por cancha según rango de fechas. 4. El sistema muestra un resumen total de ingresos por todas las canchas. |
| **Postcondiciones** | El administrador visualiza la información de ingresos por canchas |
| **Flujo alternativo** | **No existen ingresos en la fecha seleccionada:**   * El sistema muestra un mensaje informativo: “No se encontraron registros.” * El usuario puede seleccionar otro rango de fechas. |

## Diagramas de Clases

**Figura 2**

#### Clases: Entidades



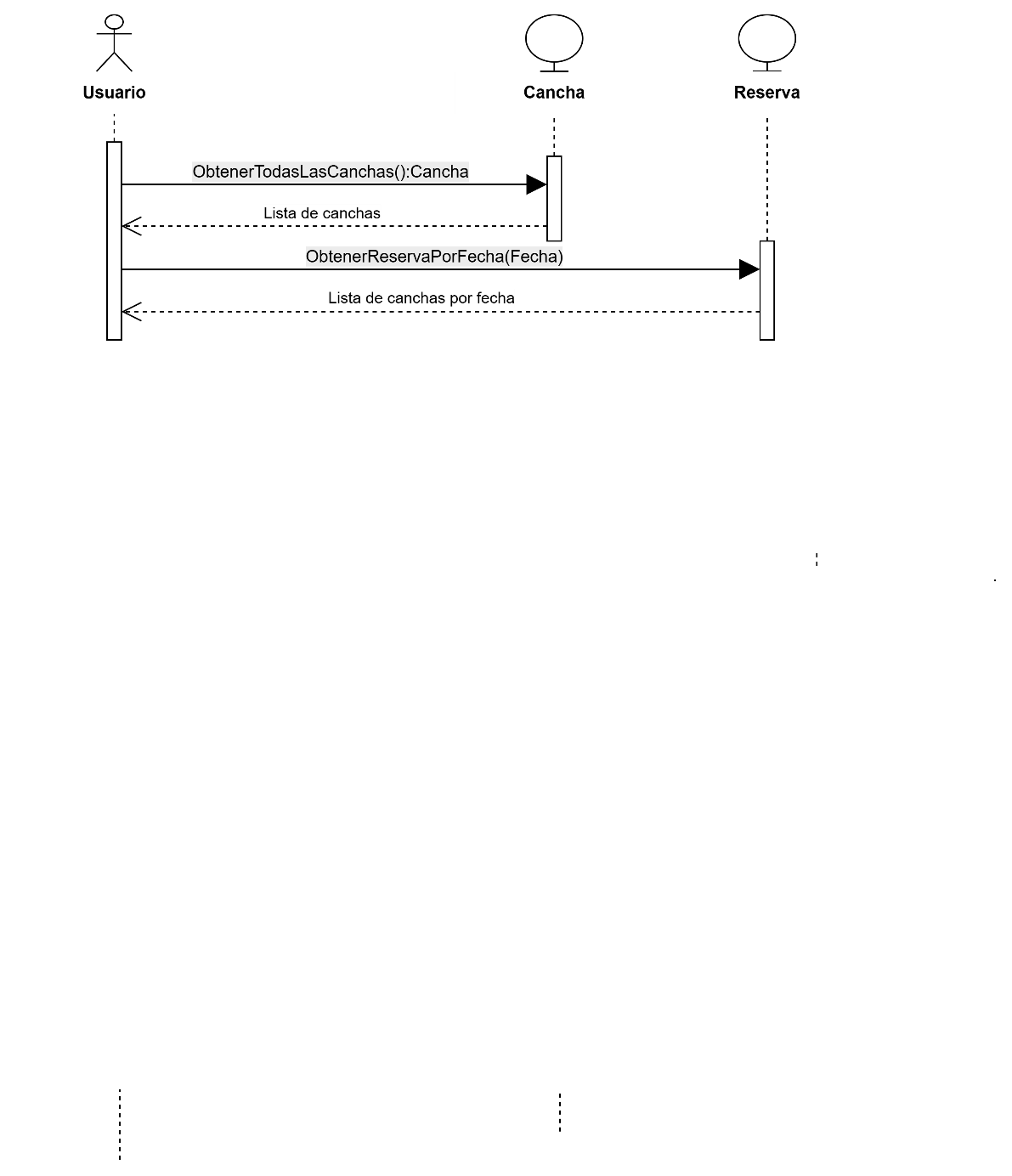
## Diagramas de Secuencia

Los diagramas de secuencia representan cómo interactúan los distintos objetos o actores del sistema a lo largo del tiempo para llevar a cabo un caso de uso específico. Estos diagramas permiten visualizar el flujo de mensajes entre los participantes del sistema y ayudan a comprender la lógica del comportamiento dinámico del software.

A continuación, se presentan los diagramas de secuencia correspondientes a los principales casos de uso del sistema de reservas de canchas deportivas.

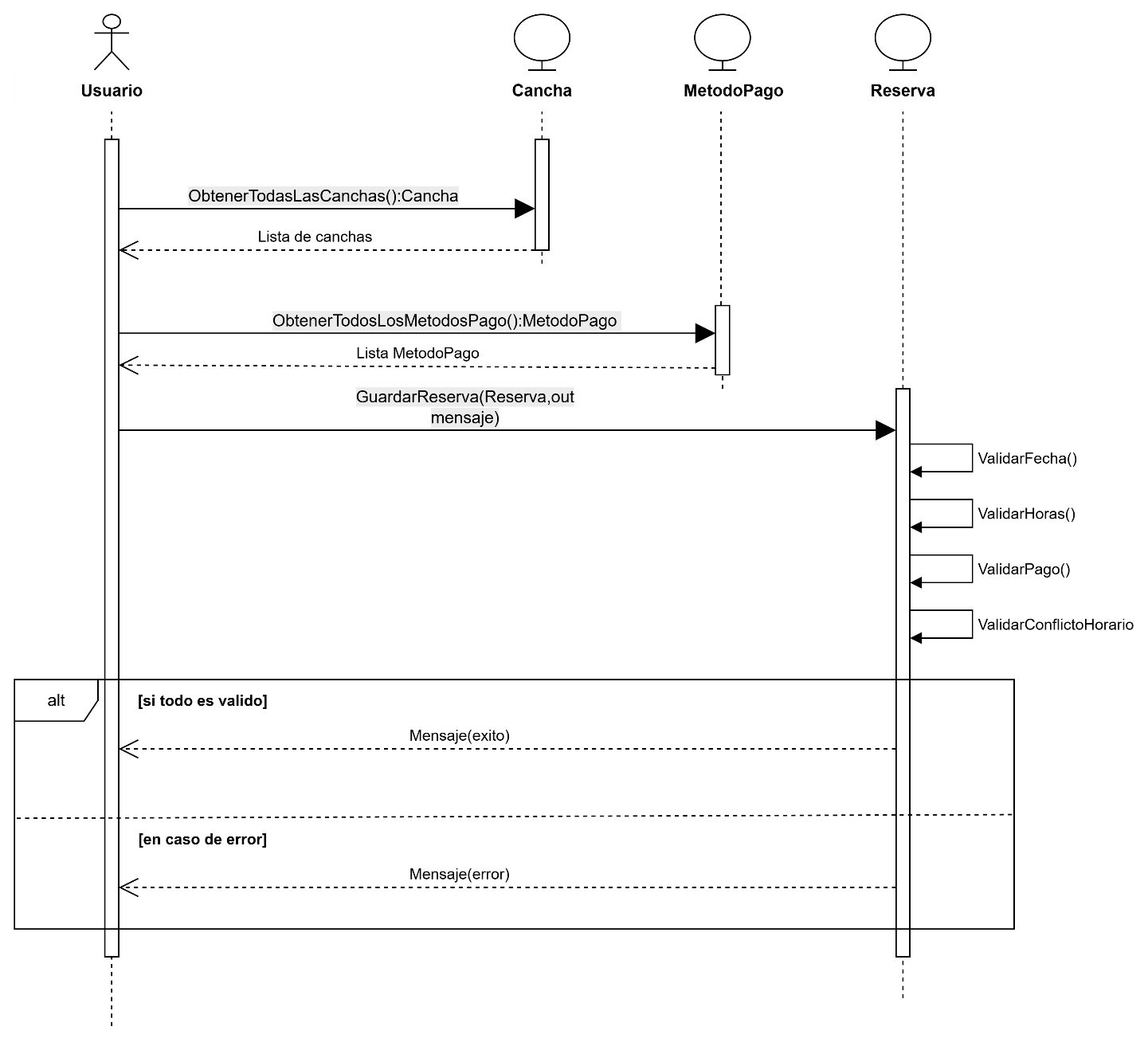
**Figura 3**

#### Diagrama de secuencia: Ver reservas del día

**

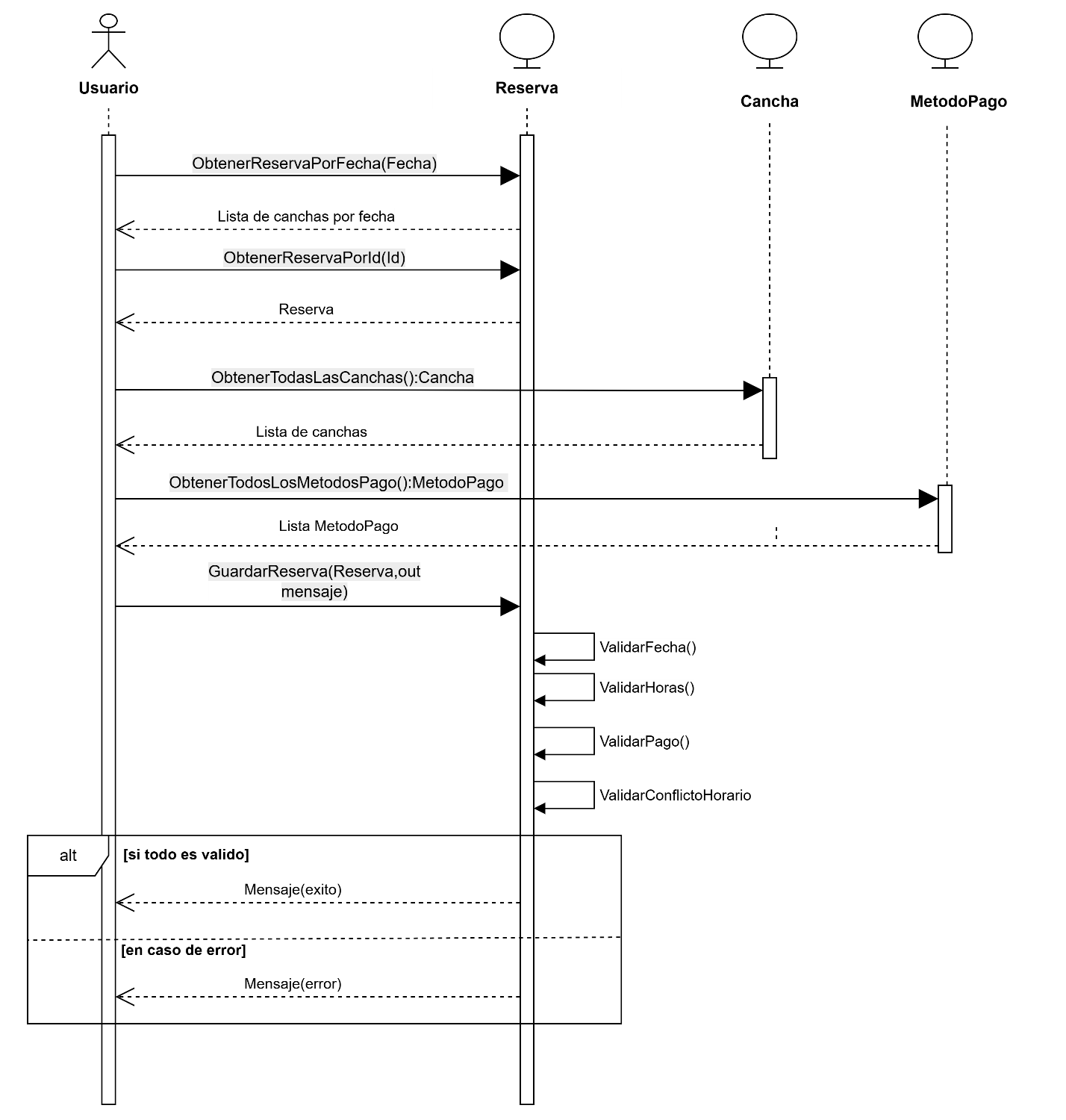
**Figura 4**

#### Diagrama de secuencia: Reservar cancha



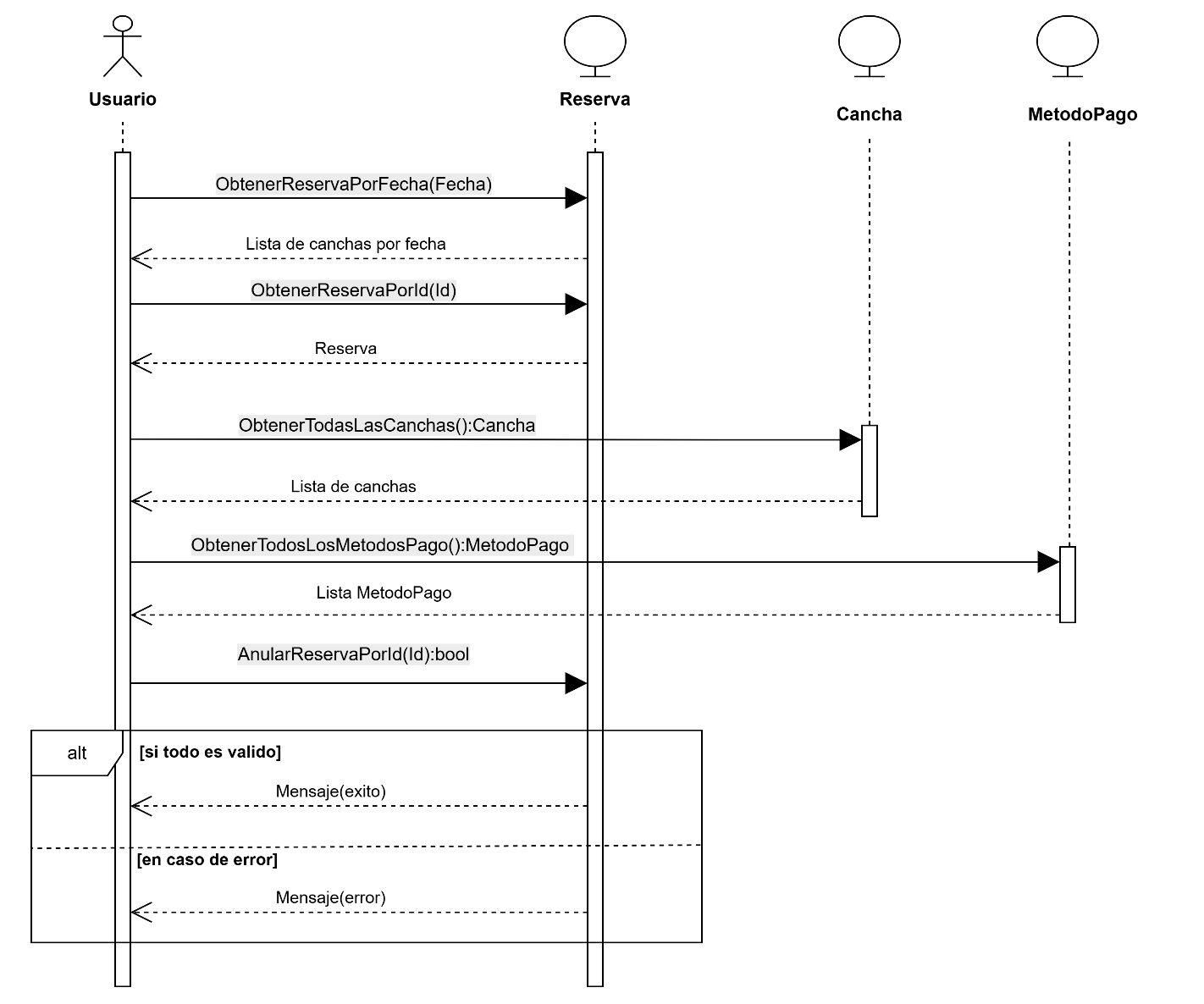
**Figura 5**

#### Diagrama de secuencia: Completar pago de reserva

**

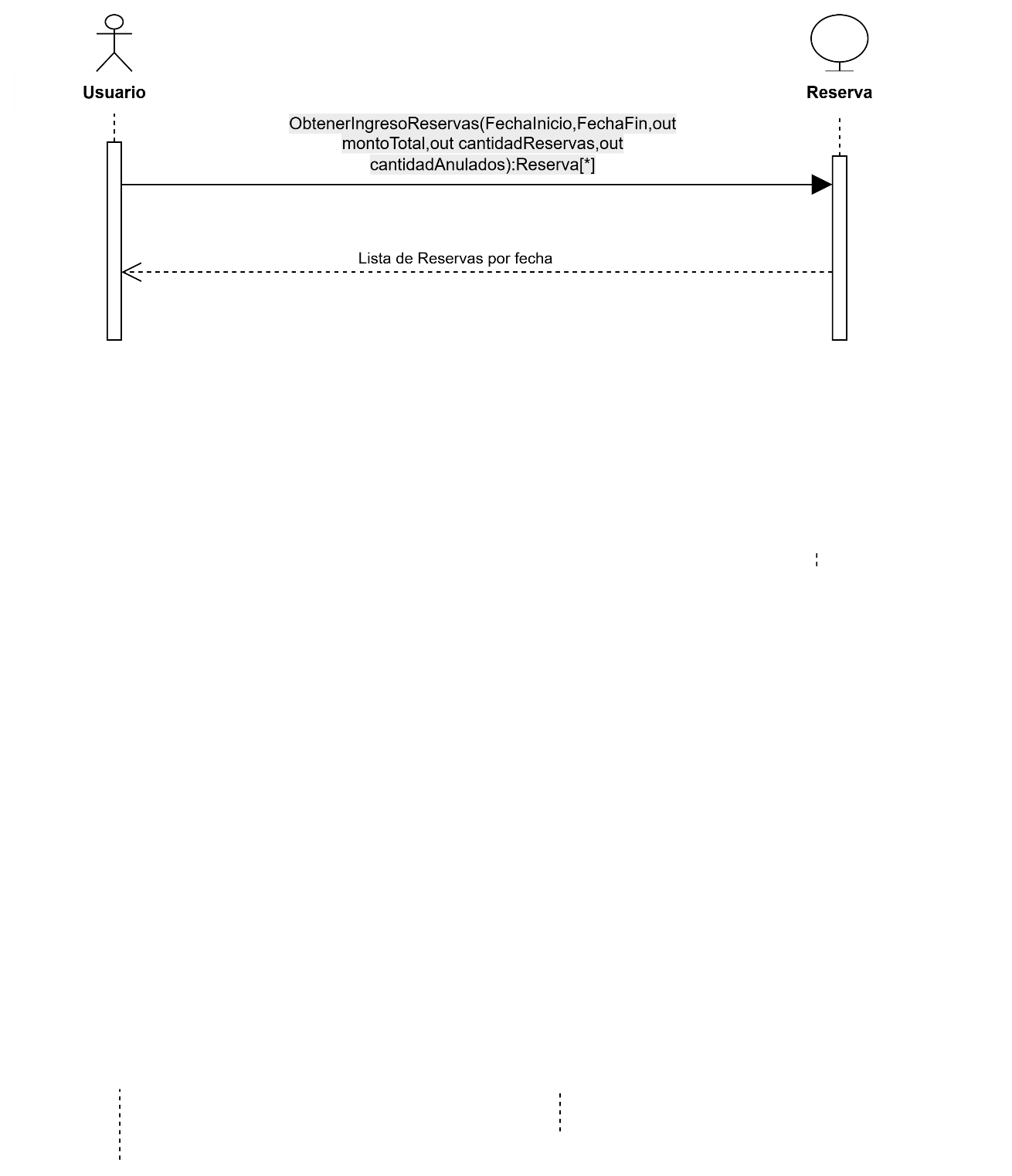
**Figura 6**

#### Diagrama de secuencia: Anular reserva

**

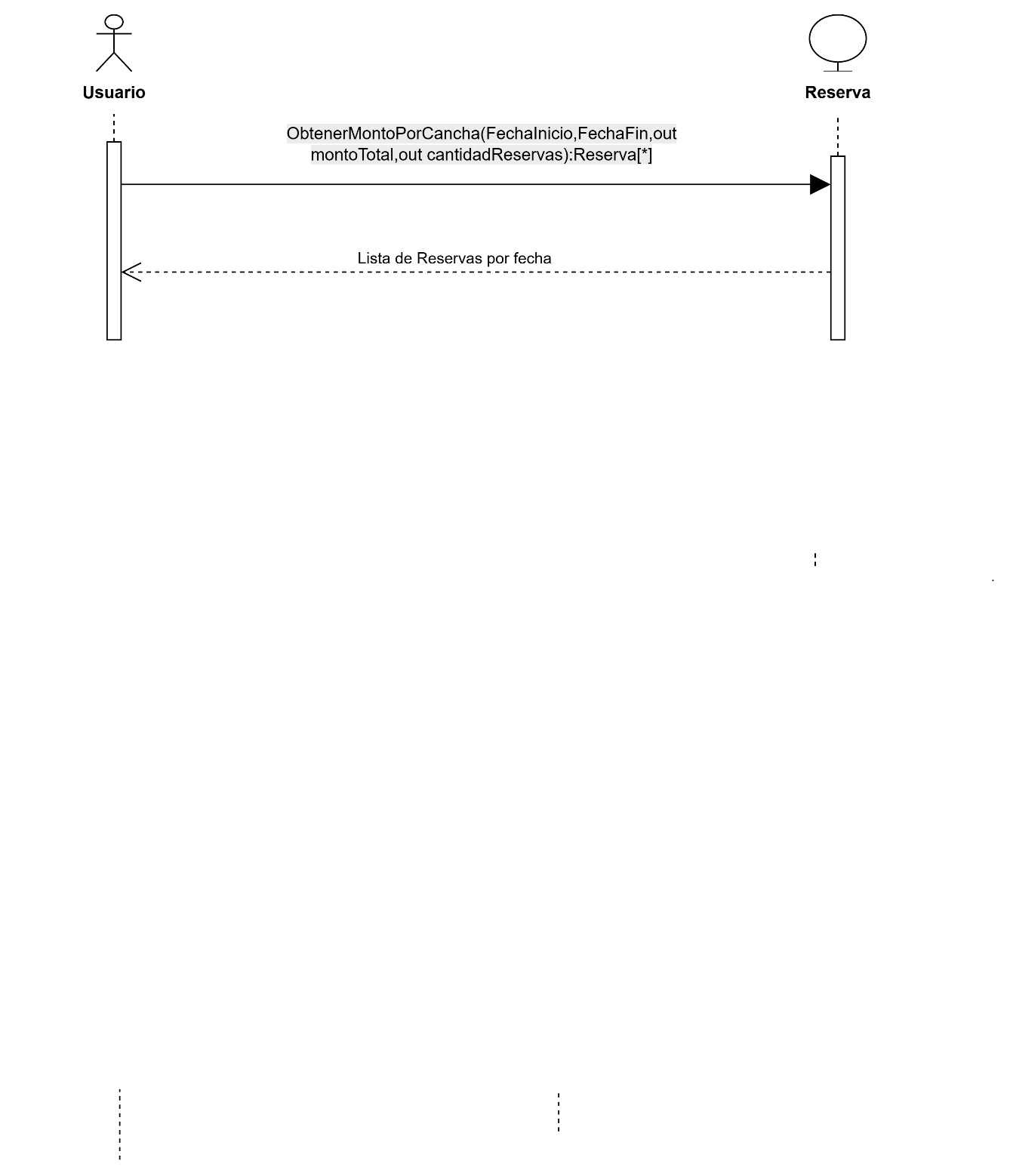
**Figura 7**

#### Diagrama de secuencia: Revisar Ingresos del día



**Figura 8**

#### Diagrama de secuencia: Ver ganancias por cancha

**

## Interfaces

El diseño de la interfaz de usuario es fundamental para asegurar que los usuarios puedan interactuar de manera eficiente y sencilla con el módulo de gestión de reservas de canchas. A continuación, se presentan los diseños que ilustran las principales pantallas del sistema, tales como módulo de gestión de usuarios, canchas, medios de pago, reservas.

Estos diseños permiten anticipar la distribución de los elementos visuales en cada pantalla y facilitan la validación de la usabilidad del sistema antes de su implementación final. Asimismo, aseguran una experiencia de usuario coherente e intuitiva, minimizando errores y facilitando la navegación por parte de los usuarios finales.

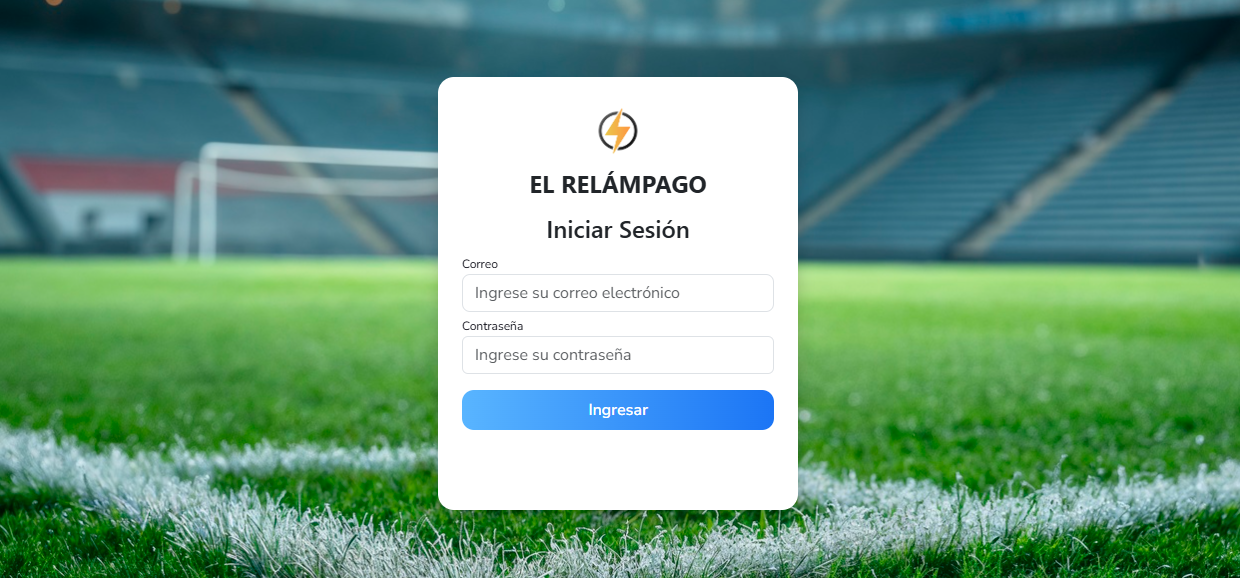
**Pantalla de Inicio de Sesión (Figura 9 y 10)**

La pantalla de inicio de sesión representa el primer punto de contacto entre el usuario y el sistema. En ella, el usuario debe ingresar sus credenciales (correo electrónico y contraseña) para autenticar su identidad y acceder a las funcionalidades disponibles.

La interfaz ha sido diseñada con un enfoque minimalista y claro, priorizando la usabilidad. Contiene campos de entrada para el correo y la contraseña, acompañados de un botón de acción para ingresar. Cabe destacar que el campo "correo" no requiere una cuenta de correo real (como Gmail o Outlook), sino que funciona como un identificador único de usuario en el sistema. Además, se ha considerado la adaptabilidad del diseño tanto para dispositivos de escritorio como para móviles, asegurando una experiencia uniforme en diferentes tamaños de pantalla se amigable para el usuario.

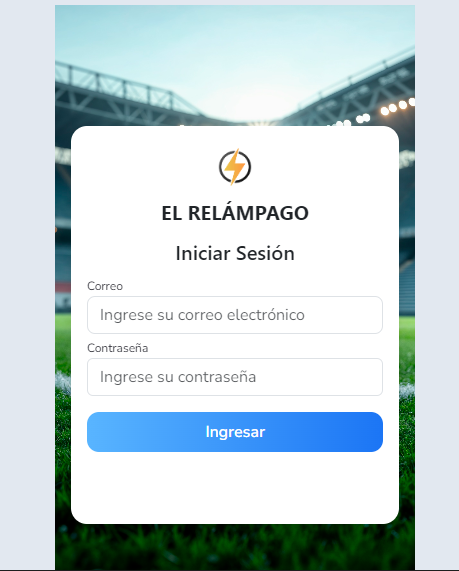
**Figura 9**

#### Mockup: Inicio de Sesión – vista escritorio

****

**Figura 9**

#### Mockup: Inicio de Sesión – vista móvil

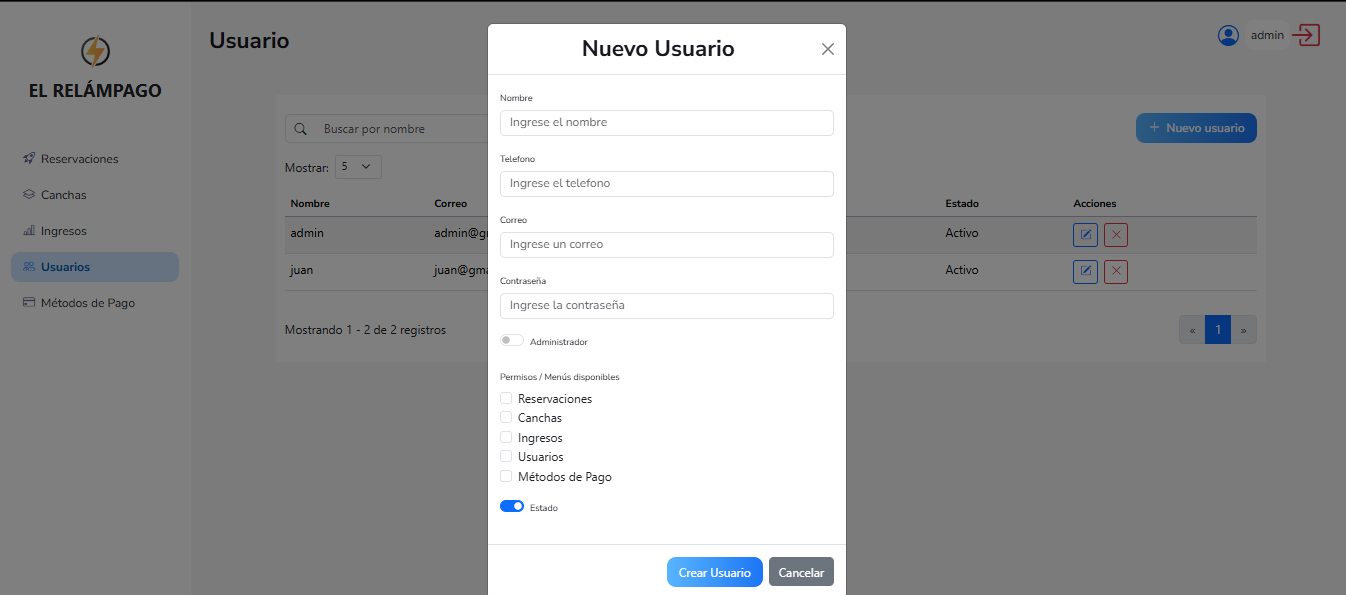
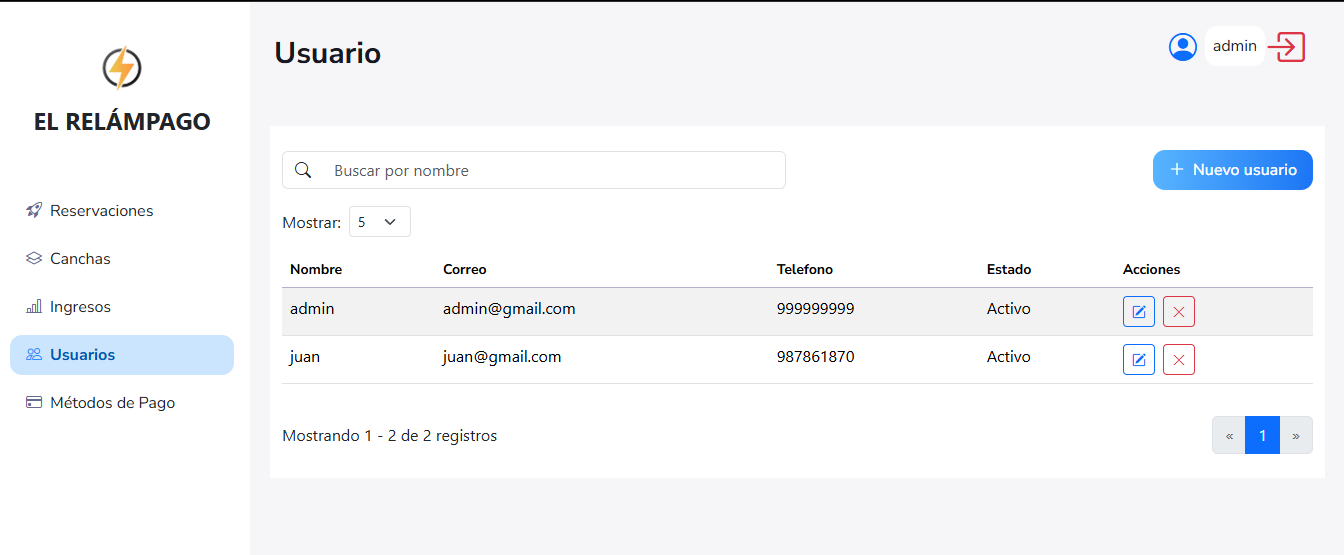


**Pantalla de Gestión de usuarios (Figura 11 y 12)**

Esta interfaz permite al usuario realizar operaciones de gestión sobre los usuarios del sistema, tales como registrar nuevos usuarios, modificar información existente del los usuarios que acceden al sistema o eliminar o desactivar. Asimismo, proporciona funcionalidades para la asignación de permisos a los módulos del sistema, vinculando a cada usuario con los módulos específicos a los que se le otorga acceso dentro del sistema.

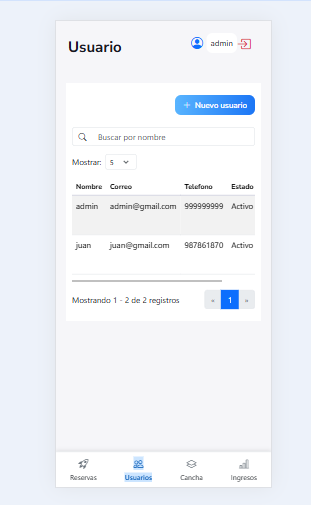
**Figura 11**

#### Mockup: Modulo Usuario – vista escritorio



**Figura 12**

#### Mockup: Modulo usuario – vista móvil

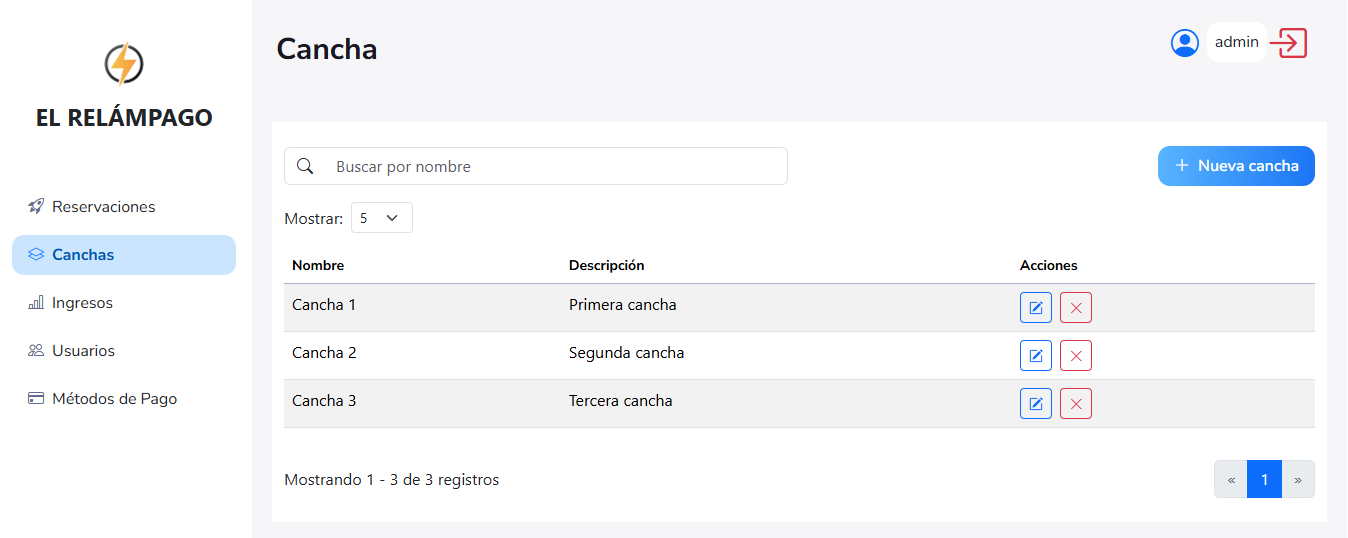


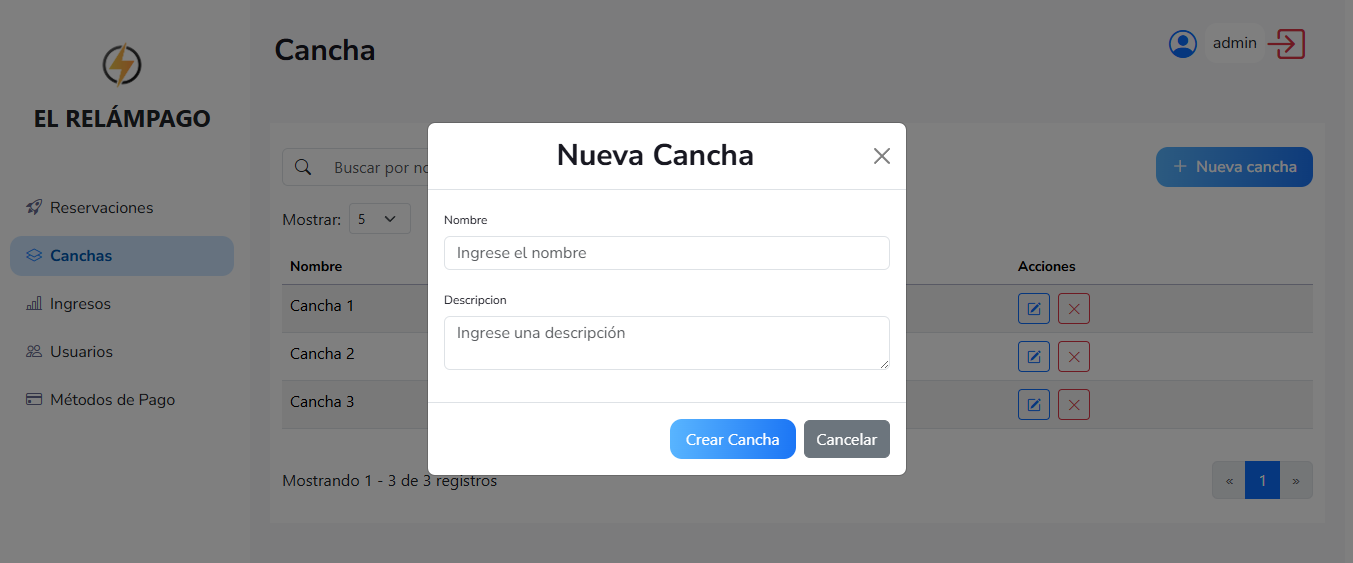
**Pantalla de Gestión de canchas (Figura 13 y 14)**

Esta interfaz permite al usuario gestionar las canchas como el proceso de registrar la canchas en el sistema para su posterior uso en el módulo de reservas, incluyendo la edición de datos existentes y la eliminación de registros según sea necesario.

**Figura 13**

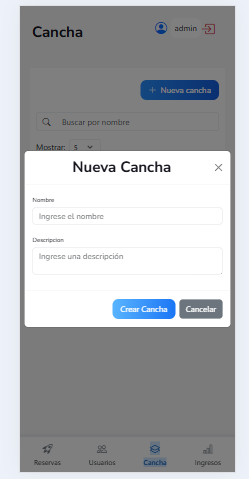
#### Mockup: Modulo Cancha – vista escritorio

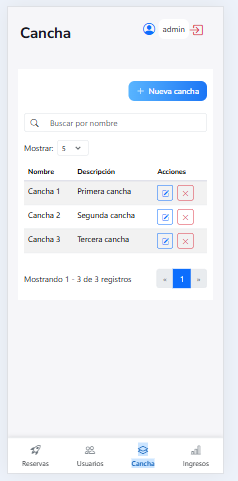




**Figura 14**

#### Mockup: Modulo Cancha – vista móvil



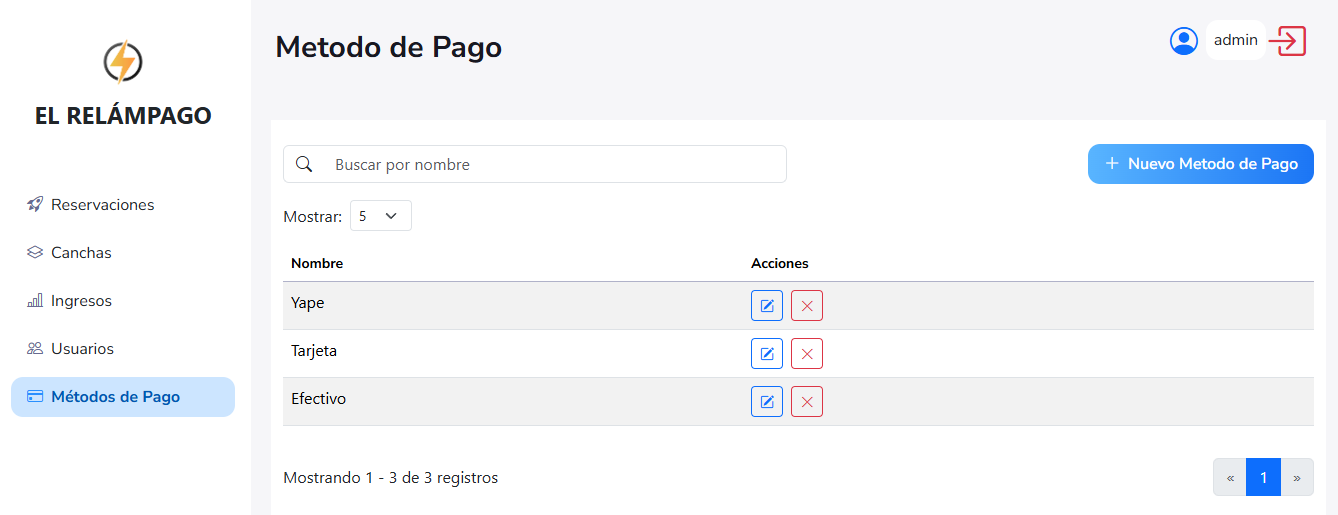


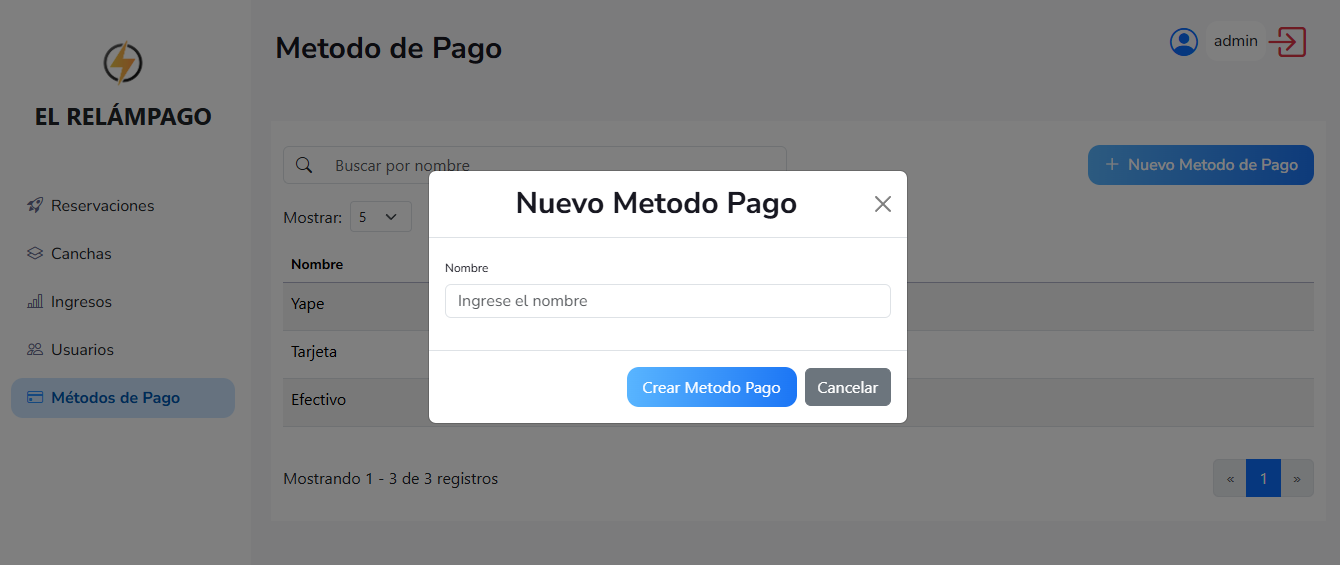
**Pantalla de Gestión de Medios de Pago (Figura 15 )**

Esta interfaz permite al usuario gestionar los distintos medios de pago registrados en el sistema. Incluye funcionalidades para registrar nuevos métodos de pago, modificar la información de los ya existentes y eliminar aquellos que no se requieran. El diseño asegura una gestión eficiente y coherente de estas entidades dentro del módulo correspondiente.

**Figura 15**

#### Mockup: Modulo Métodos de Pago – vista escritorio



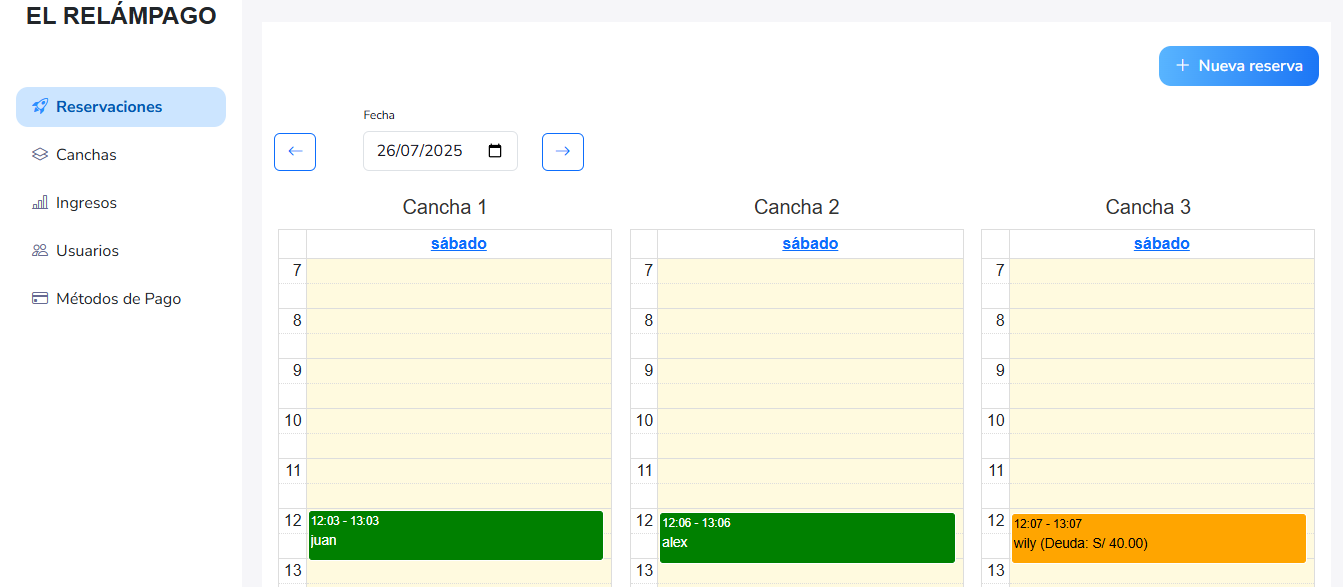


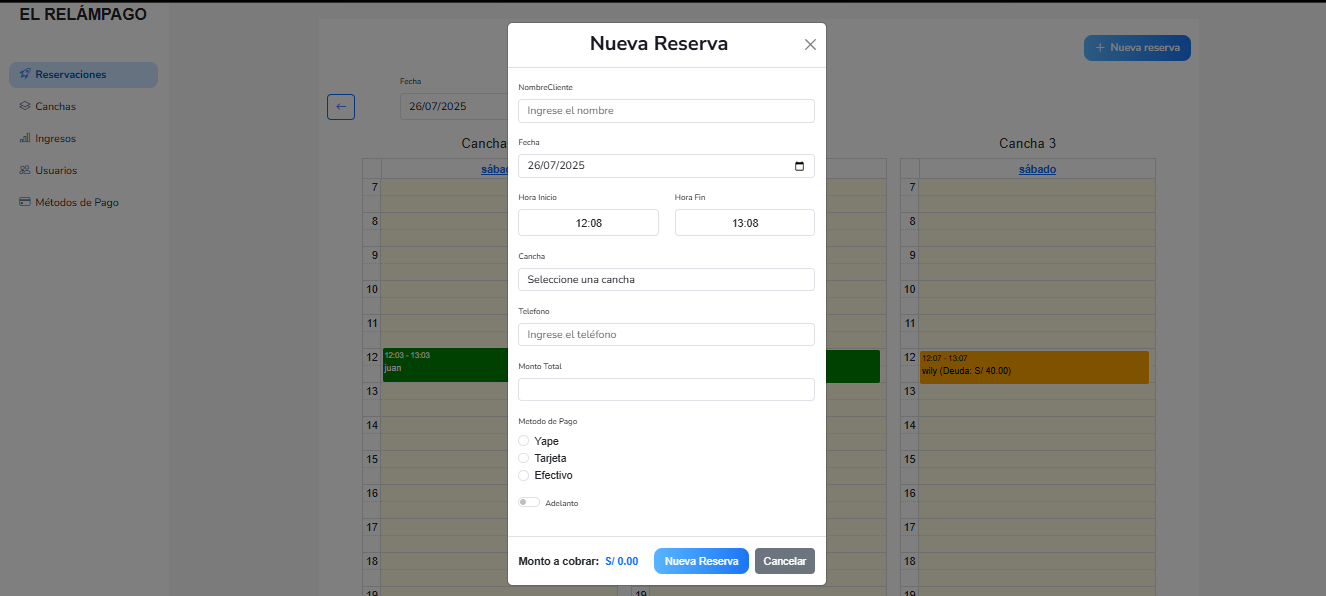
**Pantalla de Gestión de Reservas (Figura 16 y 17)**

Esta interfaz permite al usuario visualizar y administrar las reservas del sistema mediante una vista tipo calendario, segmentada por fechas, canchas y franjas horarias. Desde esta pantalla se pueden registrar nuevas reservas, así como anular o consultar las ya existentes de forma clara y ordenada.

**Figura 16**

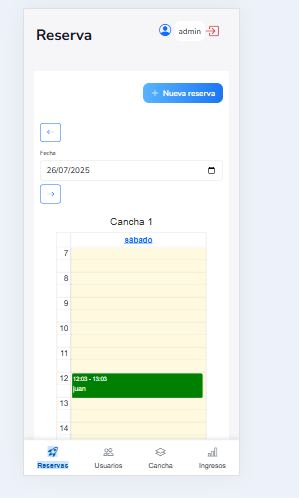
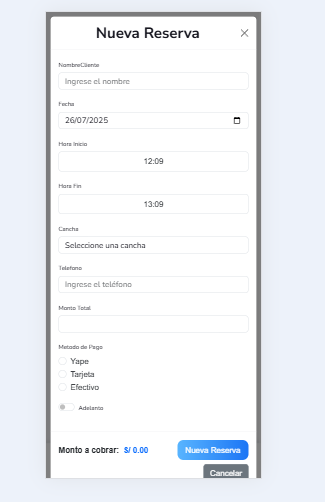
#### Mockup: Modulo Reserva – vista escritorio





**Figura 16**

#### Mockup: Modulo Reserva – vista móvil

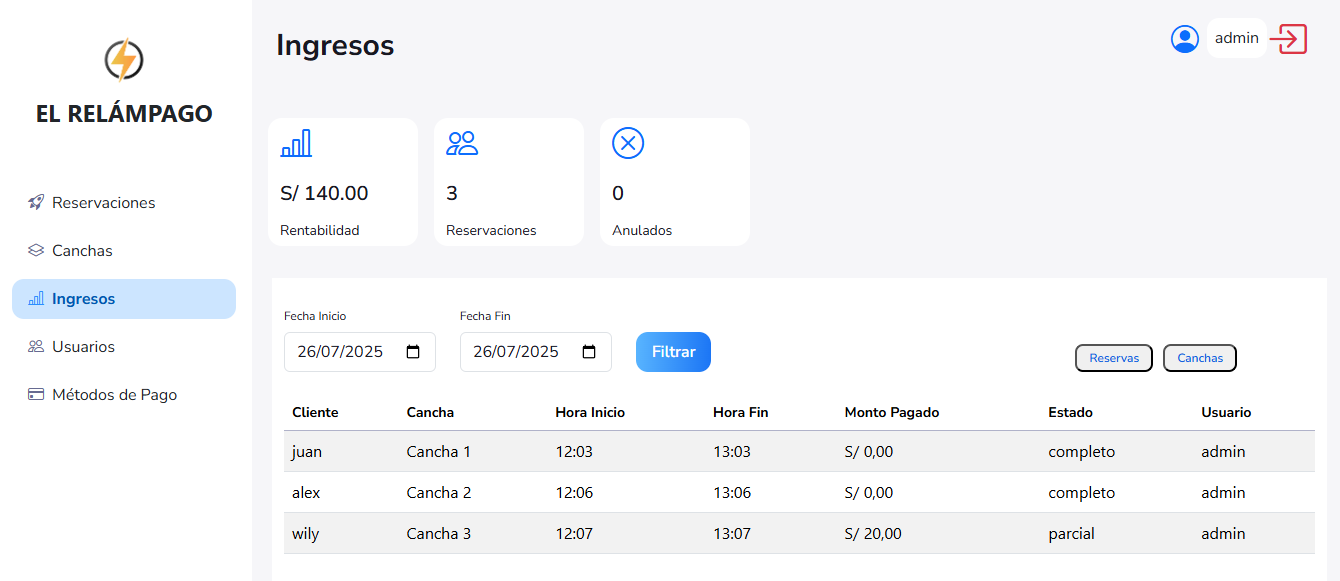


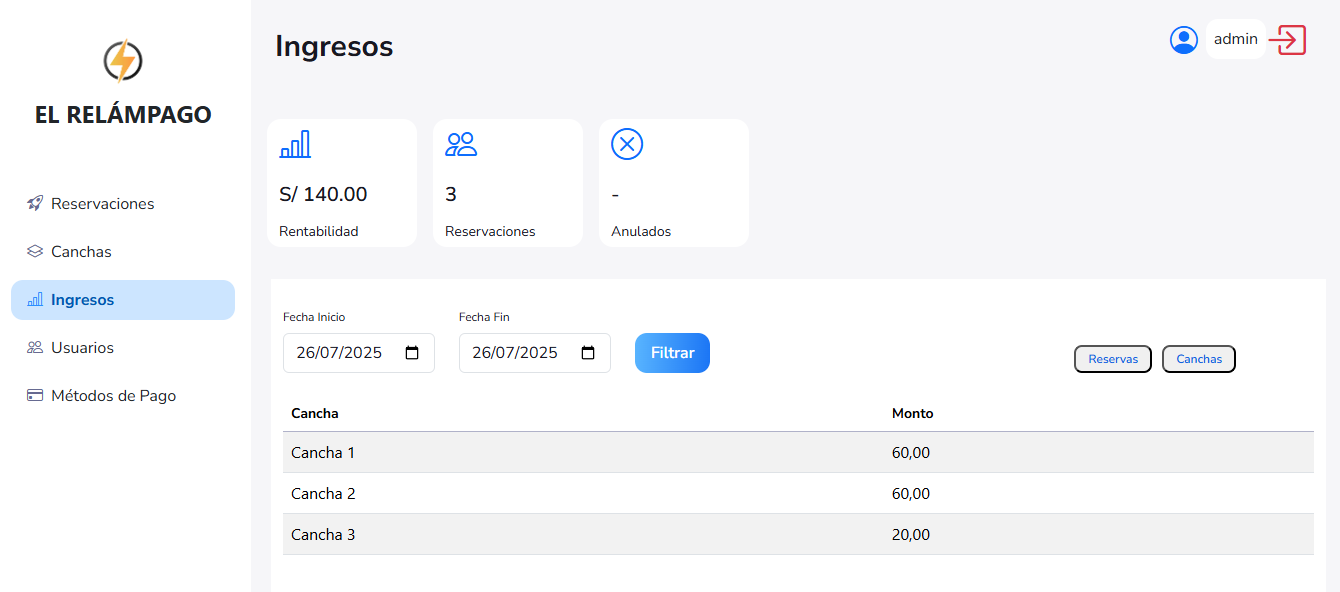
**Pantalla de Gestión de Ingresos (Figura 18 y 19)**

Esta interfaz permite al usuario visualizar los ingresos generados por reservas y, de forma específica, por cancha al seleccionar la opción correspondiente. Además, proporciona un resumen de la rentabilidad total dentro de un rango de fechas determinado, junto con el número de reservas efectuadas y anuladas en dicho periodo. También se presenta información detallada sobre los clientes que realizaron las reservas, así como del personal encargado de registrarlas.

**Figura 18**

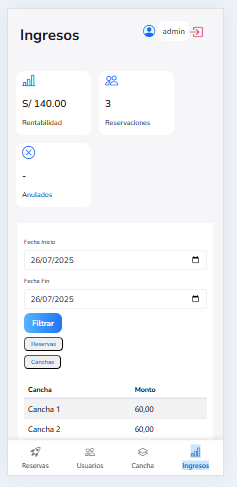
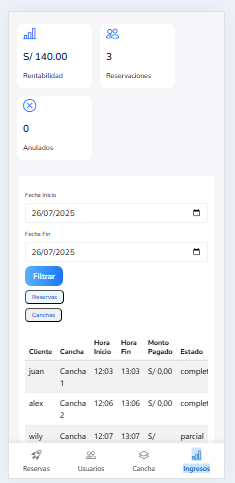
#### Mockup: Modulo Ingresos – vista escritorio





**Figura 19**

#### Mockup: Modulo Ingresos – vista móvil



## Arquitectura

La arquitectura adoptada en este sistema sigue un modelo multicapa, el cual permite una separación clara de responsabilidades, facilitando la escalabilidad, el mantenimiento y la evolución del sistema. Este modelo organiza el código en cuatro niveles bien definidos, cada uno con funciones específicas:

**Capa de Presentación**

Responsable de la interacción con el usuario. Se implementa mediante páginas web desarrolladas en ASP.NET Core con Razor Pages, utilizando elementos visuales intuitivos y modernos. Esta capa permite al usuario:

* Iniciar sesión.
* Gestionar reservas.
* Visualizar ingresos y reportes.
* Interactuar con interfaces como calendarios (FullCalendar) y selectores de fechas (Flatpickr).

También incluye validaciones de datos mediante Data Annotations para asegurar la integridad de la información antes de enviarla a las capas inferiores.

**Capa Client**

Esta capa intermedia actúa como puente de comunicación entre la interfaz de usuario y la lógica de negocio. Se encarga de:

* Exponer métodos específicos que encapsulan acciones como crear reservas, calcular ingresos o validar disponibilidad.
* Facilitar la reutilización del código.

Gracias a esta capa, se logra desacoplar la presentación del procesamiento interno del sistema.

**Capa de Servicios**

Contiene todas las reglas, validaciones y procesos clave del negocio. Aquí se implementan funcionalidades como:

* Gestión de estados de pago (Parcial, Pagado).
* Anulación de reservas.
* Generación de reportes Ingresos.

Esta capa maneja las entidades del sistema (Reserva, Usuario, Cancha, etc.) y define su comportamiento, asegurando que todas las operaciones cumplan con las reglas del negocio de la empresa *El Relámpago*.

**Capa de Repositorio**

Es responsable de la comunicación directa con la base de datos relacional SQL Server. En este sistema, se ha optado por una implementación mediante ADO.NET, utilizando procedimientos almacenados para realizar las operaciones de persistencia, lo cual garantiza un mayor control sobre el acceso a los datos y mejora el rendimiento en entornos con alta concurrencia.

Las funciones clave de esta capa incluyen:

* Ejecución de operaciones CRUD (Crear, Leer, Actualizar, Eliminar) mediante comandos SQL encapsulados en procedimientos almacenados.
* Establecimiento de conexiones y comandos SQL usando SqlConnection, SqlCommand, SqlDataReader y otros objetos del espacio de nombres System.Data.SqlClient.
* Separación clara entre la lógica del negocio y el acceso físico a los datos, permitiendo que los cambios en la base de datos no impacten directamente la lógica del sistema.
* Manejo controlado de errores y transacciones, asegurando la integridad de los datos durante las operaciones críticas.

Este enfoque proporciona mayor seguridad y rendimiento, al evitar inyecciones SQL y optimizar la ejecución de consultas a través de procedimientos ya compilados en el motor de base de datos.

# Desarrollo e Implementación

## Desarrollo del Módulo

El desarrollo del sistema web para la gestión de reservas de canchas sintéticas en la empresa *El Relámpago* se realizó utilizando ASP.NET Core Web con C#. El sistema está basado en una arquitectura en capas, lo que permite organizar el código en diferentes niveles de responsabilidad: presentación, lógica de negocio y acceso a datos. Este enfoque simplifica la gestión del sistema y mejora su mantenibilidad.

### Tecnologías Utilizadas

Se utilizó ASP.NET Core como plataforma para el desarrollo del sistema web, aprovechando sus capacidades para construir aplicaciones dinámicas y seguras. Las páginas se diseñaron utilizando Razor Pages, que facilita la interacción entre el código y las vistas, todo dentro del mismo proyecto. jQuery se integró para proporcionar interactividad dinámica en los formularios, como la validación de datos antes de enviarlos al servidor.

**Acceso a la Base de Datos**

La gestión de la base de datos se realizó con ADO.NET para ejecutar procedimientos almacenados directamente en SQL Server. Este enfoque permitió manejar las operaciones de CRUD (Crear, Leer, Actualizar y Eliminar) de manera eficiente, asegurando que la información sobre las reservas, pagos y canchas fuera gestionada correctamente sin complicaciones adicionales.

### Estructura del Proyecto

**Capa de Entidad**

Para completar la estructura del sistema, se agrega la Capa de Entidad, que es responsable de definir las clases y modelos de datos utilizados en el sistema. Estas entidades representan los objetos del dominio, como Reserva, Usuario, Cancha, entre otros, y se utilizan para transportar los datos entre las capas de la aplicación.

En este caso, las clases de la capa de entidad son clases simples con propiedades que se corresponden directamente con las tablas de la base de datos. Se utilizan para almacenar la información de las reservas, los usuarios y las canchas, y para facilitar el acceso y la manipulación de los datos en el sistema.

La Capa de Entidad está compuesta por las clases que definen las entidades principales del sistema. Cada entidad tiene propiedades que corresponden a los campos de la base de datos y métodos que gestionan el comportamiento de esos datos.

**Código 1 ;**

NET 8 Entidad

**namespace** **SR.Entities.BaseEntities.ReservaEntities**

{

**public** **class** **Reserva**

{

**public** **int** Id { **get**; **set**; }

**public** **string?** NombreCliente { **get**; **set**; }

**public** DateTime? Fecha { **get**; **set**; }

**public** TimeSpan? HoraInicio { **get**; **set**; }

**public** TimeSpan? HoraFin { **get**; **set**; }

**public** **int?** MetodoPagoId { **get**; **set**; }

**public** **string?**TipoPago { **get**; **set**; }

**public** **int?** CanchaId { **get**; **set**; }

**public** **decimal?** MontoTotal { **get**; **set**; }

**public** **decimal?** MontoPagado { **get**; **set**; }

**public** **string?** Telefono { **get**; **set**; }

**public** **string?** NombreCancha { **get**; **set**; }

**public** **string?** UsuarioNombre { **get**; **set**; }

**public** **string?** Estado { **get**; **set**; }

}

}

**namespace** **SR.Entities.ViewModels**

{

**public** **class** **ReservaViewModel**

{

**public** **int** Id { **get**; **set**; }

[Required(ErrorMessage = "El nombre del cliente es obligatorio.")]

[StringLength(100, ErrorMessage = "El nombre no puede tener más de 100 caracteres.")]

**public** **string?** NombreCliente { **get**; **set**; }

[Required(ErrorMessage = "La fecha es obligatoria.")]

[DataType(DataType.Date)]

**public** DateTime? Fecha { **get**; **set**; }

[Required(ErrorMessage = "La hora de inicio es obligatoria.")]

[DataType(DataType.Time)]

**public** TimeSpan? HoraInicio { **get**; **set**; }

[Required(ErrorMessage = "La hora de fin es obligatoria.")]

[DataType(DataType.Time)]

**public** TimeSpan? HoraFin { **get**; **set**; }

[Required(ErrorMessage = "Debe seleccionar un método de pago.")]

**public** **int?** MetodoPagoId { **get**; **set**; }

**public** **string?** TipoPago { **get**; **set**; }

[Required(ErrorMessage = "Debe seleccionar una cancha.")]

**public** **int?** CanchaId { **get**; **set**; }

[Required(ErrorMessage = "El monto total es obligatorio.")]

[Range(0.01, double.MaxValue, ErrorMessage = "El monto total debe ser mayor a cero.")]

**public** **decimal?** MontoTotal { **get**; **set**; }

**public** **decimal?** MontoPagado { **get**; **set**; }

[RegularExpression(@"^9\d{8}$", ErrorMessage = "El número de teléfono debe tener 9 dígitos y comenzar con 9.")]

**public** **string?** Telefono { **get**; **set**; }

**public** ObservableCollection<Cancha> Canchas { **get**; **set**; } = **new**();

**public** ObservableCollection<MetodoPago> MetodoPagos { **get**; **set**; } = **new**();

**public** Boolean Estado { **get**; **set**; }

}

}

**Capa de Presentación**

La Capa de Presentación se construyó usando Razor Pages, donde cada página maneja tanto la lógica de presentación como su modelo de datos asociado. Los formularios permiten a los usuarios registrar reservas, consultar la disponibilidad de las canchas y realizar pagos. Además, se implementaron validaciones utilizando **Data Annotations** para asegurar que los datos ingresados sean correctos antes de ser procesados.

**@using System.Collections.ObjectModel**

**@using SR.Entities.BaseEntities.CanchaEntities**

**@model ObservableCollection<Cancha>**

**@{**

**Layout = "~/Views/Shared/\_LayoutHome.cshtml";**

**ViewData["Titulo"] = "Reserva";**

**}**

**<div class="container py-4 table-background-white">**

**<div class="row align-items-center mb-3 g-4">**

**<div class="col-12 col-lg-6 order-2 order-lg-1">**

**<div class="input-group">**

**</div>**

**</div>**

**<div class="col-12 col-lg-auto order-1 order-lg-2 ms-lg-auto d-flex justify-content-end">**

**<button type="button" class="body-01-semibold btn-primary-01 w-70 w-lg-auto"**

**data-bs-toggle="modal"**

**data-bs-target="#reservaModal"**

**data-url="/Reserva/Create">**

**<i class="bi bi-plus-lg"></i> Nueva reserva**

**</button>**

**</div>**

**</div>**

**<div class="row pb-3">**

**<div class="col-md-1 mb-2 align-self-end">**

**<button id="prevDay" class="btn btn-outline-primary">**

**<i class="bi bi-arrow-left"></i>**

**</button>**

**</div>**

**<div class="col-md-2 mb-2">**

**<label class="form-label caption-regular">Fecha</label>**

**<input type="date" class="body-01-regular fecha-date form-control" id="datePicker">**

**</div>**

**<div class="col-md-2 mb-2 align-self-end">**

**<button id="nextDay" class="btn btn-outline-primary">**

**<i class="bi bi-arrow-right"></i>**

**</button>**

**</div>**

**</div>**

**<div class="calendar-container">**

**@foreach (var cancha in Model)**

**{**

**<div class="calendar-box">**

**<h3>@cancha.Nombre</h3>**

**<div id="@cancha.Id" class="calendar"></div>**

**</div>**

**}**

**</div>**

**</div>**

<div id="calendar"></div> <!-- Aquí se renderizará el calendario -->

<!-- Modal -->

<div class="modal fade" id="reservaModal" tabindex="-1" aria-hidden="true">

<div class="modal-dialog modal-dialog-centered modal-md">

<div class="modal-content" id="reservaModalContent">

<!-- Contenido dinámico -->

</div>

</div>

</div>

<script>

**let** calendars = {};

$(document).ready(function () {

**const** localToday = **new** Date();

**const** yyyy = localToday.getFullYear();

**const** mm = String(localToday.getMonth() + **1**).padStart(**2**, '0');

**const** dd = String(localToday.getDate()).padStart(**2**, '0');

**const** today = `${yyyy}-${mm}-${dd}`;

$('#datePicker').val(today);

inicializarCalendarios(today);

$('#datePicker').**on**('change', function () {

**const** date = **this**.**value**;

actualizarCalendarios(date);

cargarEventos(date);

});

$('#prevDay').click(function () {

**const** newDate = moverFecha(-**1**);

$('#datePicker').val(newDate);

actualizarCalendarios(newDate);

cargarEventos(newDate);

});

$('#nextDay').click(function () {

**const** newDate = moverFecha(**1**);

$('#datePicker').val(newDate);

actualizarCalendarios(newDate);

cargarEventos(newDate);

});

$('[data-bs-target="#reservaModal"]').**on**('click', function () {

**const** url = $(**this**).data('url');

$.**get**(url, function (html) {

$('#reservaModalContent').html(html);

inicializarFlatpickr();

inicializarToggleMontoPagado();

});

});

$(document).**on**('click', '.open-modal', function (e) {

e.preventDefault();

**const** url = $(**this**).attr('href');

$.**get**(url, function (html) {

$('#reservaModalContent').html(html);

**new** bootstrap.Modal($('#reservaModal')[**0**]).show();

inicializarFlatpickr();

inicializarToggleMontoPagado();

});

});

$(document).off('submit', '#reservaForm').**on**('submit', '#reservaForm', function (e) {

e.preventDefault();

**const** $form = $(**this**);

**const** url = $form.attr('action');

**const** data = $form.serialize();

$.post(url, data)

.done(function (response) {

**if** (**typeof** response === '**object**' && response.success) {

**const** modal = bootstrap.Modal.getInstance($('#reservaModal')[**0**]);

modal.hide();

Swal.fire({

icon: 'success',

title: '¡Éxito!',

text: 'La Reserva ha sido guardada correctamente.',

timer: **2000**,

showConfirmButton: **false**

});

**const** fechaSeleccionada = $('#datePicker').val();

cargarEventos(fechaSeleccionada);

} **else** **if** (**typeof** response === '**object**' && !response.success){

Swal.fire({

icon: 'error',

title: '¡Error!',

text: response.message,

timer: **3000**,

showConfirmButton: **false**

});

}**else**{

$('#reservaModalContent').html(response);

inicializarFlatpickr();

inicializarToggleMontoPagado();

}

})

.fail(function () {

Swal.fire({

icon: 'error',

title: 'Error del servidor',

text: 'No se pudo procesar la solicitud.',

timer: **3000**,

showConfirmButton: **false**

});

});

});

$(document).off('click', '#btnAnularReserva').**on**('click', '#btnAnularReserva', function () {

**const** idReserva = $(**this**).data('id');

Swal.fire({

title: '¿Estás seguro?',

text: "Esta acción anulará la reserva.",

icon: 'warning',

showCancelButton: **true**,

confirmButtonColor: '#d33',

cancelButtonColor: '#**6**c757d',

confirmButtonText: 'Sí, anular',

cancelButtonText: 'Cancelar'

}).then((result) => {

**if** (result.isConfirmed) {

$.post('/Reserva/Anular', { id: idReserva }, function (response) {

**if** (response.success) {

Swal.fire({

icon: 'success',

title: 'Reserva anulada',

text: response.message || 'La reserva fue anulada correctamente.',

timer: **2000**,

showConfirmButton: **false**

});

**const** modal = bootstrap.Modal.getInstance($('#reservaModal')[**0**]);

modal.hide();

**const** fechaSeleccionada = $('#datePicker').val();

cargarEventos(fechaSeleccionada);

} **else** {

Swal.fire({

icon: 'error',

title: 'Error',

text: response.message || 'No se pudo anular la reserva.'

});

}

});

}

});

});

});

function **moverFecha**(dias) {

**const** fecha = **new** Date($('#datePicker').val());

fecha.setDate(fecha.getDate() + dias);

**return** fecha.toISOString().split('T')[**0**];

}

function **inicializarCalendarios**(fecha) {

$('.calendar').each(function () {

**const** canchaId = $(**this**).attr('id').replace('cancha', '').toString();

**const** calendarEl = **this**;

**const** calendar = **new** FullCalendar.Calendar(calendarEl, {

initialView: 'timeGridDay',

initialDate: fecha,

slotMinTime: '**07**:**00**:**00**',

slotMaxTime: '**24**:**00**:**00**',

height: 'auto',

allDaySlot: **false**,

headerToolbar: **false**,

locale: 'es',

events: [],

eventClick: function (info) {

// Solo si es amarillo (deuda)

**if** (info.**event**.backgroundColor === 'orange' || info.**event**.backgroundColor === 'green') {

**const** idReserva = info.**event**.extendedProps.idReserva;

$.**get**(`/Reserva/Editar`, { id: idReserva }, function (html) {

$('#reservaModalContent').html(html);

**new** bootstrap.Modal($('#reservaModal')[**0**]).show();

inicializarFlatpickr();

inicializarToggleMontoPagado();

});

}

}

});

calendar.render();

calendars[canchaId] = calendar;

});

cargarEventos(fecha);

}

function **actualizarCalendarios**(fecha) {

Object.values(calendars).forEach(cal => cal.gotoDate(fecha));

}

function **cargarEventos**(fecha) {

$.**get**('/Reserva/ObtenerReservasPorFecha', { fecha })

.done(function (data) {

Object.values(calendars).forEach(cal => cal.removeAllEvents());

data.forEach(r => {

**const** canchaId = r.canchaId.toString();

**if** (calendars[canchaId]) {

calendars[canchaId].addEvent({

title: r.title,

start: r.start,

end: r.end,

backgroundColor: r.backgroundColor,

textColor: r.textColor,

borderColor: r.borderColor,

idReserva: r.idReserva

});

}

});

})

.fail(function (xhr, status, error) {

console.error("Error al cargar eventos:", status, error);

console.error("Respuesta del servidor:", xhr.responseText);

});

}

function **inicializarFlatpickr**() {

$('#reservaModalContent').find('.flatpickr-time').flatpickr({

enableTime: **true**,

noCalendar: **true**,

dateFormat: "H:i",

time\_24hr: **true**,

minuteIncrement: **15**,

disableMobile:**true**

});

}

function **inicializarToggleMontoPagado**() {

**const** estadoCheckbox = document.getElementById('estadoCheckbox');

**const** montoPagadoDiv = document.getElementById('montoPagadoDiv');

**const** montoPagadoInput = document.getElementById('MontoPagado');

**const** montoTotalInput = document.getElementById('MontoTotal');

**const** montoPorCobrarSpan = document.getElementById('montoPorCobrar');

**const** montoLabel = document.getElementById('montoLabel');

**if** (!estadoCheckbox || !montoPagadoDiv) **return**;

function **toggleMontoPagado**() {

**if** (estadoCheckbox.**checked**) {

montoPagadoDiv.style.display = 'block';

} **else** {

montoPagadoDiv.style.display = 'none';

montoPagadoInput.**value** = '';

}

calcularMontoPorCobrar();

}

function **calcularMontoPorCobrar**() {

**const** total = parseFloat(montoTotalInput.**value**) || **0**;

**const** pagado = estadoCheckbox.**checked** ? (parseFloat(montoPagadoInput.**value**) || **0**) : **0**;

**const** porCobrar = Math.max(total - pagado, **0**);

**if** (estadoCheckbox.**checked**) {

montoLabel.textContent = "Monto de deuda:";

} **else** {

montoLabel.textContent = "Monto a cobrar:";

}

montoPorCobrarSpan.textContent = `S/ ${porCobrar.toFixed(**2**)}`;

}

toggleMontoPagado();

estadoCheckbox.addEventListener('change', toggleMontoPagado);

montoPagadoInput.addEventListener('input', calcularMontoPorCobrar);

montoTotalInput.addEventListener('input', calcularMontoPorCobrar);

}

</script>

**Capa Client**

La Capa Client actúa como un intermediario entre la Capa de Presentación y la Capa de Servicios, facilitando la comunicación entre ambas y delegando las operaciones de negocio específicas a los servicios adecuados. Esta capa no contiene lógica de negocio compleja, sino que simplemente dirige las solicitudes de la Capa de Presentación a la Capa de Servicios, asegurando que las operaciones sean procesadas correctamente.

En el código proporcionado, la clase ReservaClient es un buen ejemplo de un intermediario que maneja las interacciones con la Capa de Lógica de Negocio (a través de IReservaService), recibiendo las solicitudes desde la Capa de Presentación y delegándolas al servicio correspondiente. La Capa de Servicios maneja la excepción de cualquier error que pueda ocurrir durante el proceso y garantiza que los datos fluyan correctamente.

**public** **class** **ReservaClient** : IReservaClient

{

**public** **readonly** IReservaService \_reservaService;

**public** **ReservaClient**(IReservaService reservaService)

{

\_reservaService = reservaService;

}

**public** **bool** **GuardarReserva**(Reserva reserva, **out** **string** mensaje)

{

**try**

{

**return** \_reservaService.GuardarReserva(reserva, **out** mensaje);

}

**catch** (Exception ex)

{

**throw** **new** **Exception**("Error: ", ex);

}

}

}

**namespace** **SR.Presentation.Controllers**

{

[Authorize]

**public** **class** **ReservaController** : Controller

{

**private** **readonly** ICanchaClient \_canchaClient;

**private** **readonly** IMetodoPagoClient \_metodoPagoClient;

**private** **readonly** IReservaClient \_reservaClient;

**public** **ReservaController**(IReservaClient reservaClient,ICanchaClient canchaClient, IMetodoPagoClient metodoPagoClient)

{

\_canchaClient = canchaClient;

\_metodoPagoClient = metodoPagoClient;

\_reservaClient=reservaClient;

}

**public** IActionResult **Index**()

{

**var** canchas = \_canchaClient.ObtenerTodasLasCanchas();

**return** **View**(canchas);

}

[HttpPost]

**public** IActionResult **Create**(ReservaViewModel reserva)

{

reserva.Canchas = \_canchaClient.ObtenerTodasLasCanchas();

reserva.MetodoPagos = \_metodoPagoClient.ObtenerTodosLosMetodosPago();

ValidarFecha(reserva.Fecha);

ValidarHoras(reserva.Fecha, reserva.HoraInicio, reserva.HoraFin);

ValidarPago(reserva.Estado, reserva.MontoPagado, reserva.MontoTotal);

**if** (ModelState.IsValid)

{

**string** mensaje="";

**var** claims = HttpContext.User;

**var** idClaim = claims.FindFirst(ClaimTypes.NameIdentifier)?.Value;

**var** userId = **int**.TryParse(idClaim, **out** **var** idParsed) ? idParsed : **0**;

Reserva model = **new** Reserva();

model.NombreCliente = reserva.NombreCliente;

model.Fecha = reserva.Fecha;

model.HoraInicio = reserva.HoraInicio;

model.HoraFin = reserva.HoraFin;

model.CanchaId = reserva.CanchaId;

model.Telefono = reserva.Telefono;

model.MontoTotal = reserva.MontoTotal;

model.MetodoPagoId = reserva.MetodoPagoId;

model.TipoPago = reserva.Estado == **true** ? "parcial" : "completo";

model.MontoPagado = reserva.MontoPagado;

model.UsuarioModifica = userId;

**bool** result = \_reservaClient.GuardarReserva(model,**out** mensaje);

**return** **Json**(**new** { success = result, message = mensaje });

}

ViewBag.Canchas = **new** SelectList(reserva.Canchas, "Id", "Nombre");

ViewBag.MetodoPagos = **new** SelectList(reserva.MetodoPagos, "Id", "Nombre");

**return** **PartialView**("\_ReservaForm", reserva);

}

}

**Capa de Servicios**

Es responsable de implementar las reglas y procesos del negocio, gestionando las operaciones relacionadas con las reservas de canchas, como la creación, anulación y consulta de reservas. Esta capa no interactúa directamente con la interfaz de usuario ni con la base de datos, sino que delega estas interacciones a la Capa Repositorio (en este caso, el IReservaRepositorio).

En tu implementación, la clase ReservaService en este ejemplo maneja la lógica de negocio para gestionar las reservas. A través de esta clase, se definen las acciones de negocio que deben cumplirse al manipular las reservas, como la validación de tipo de pago (como "completo", "parcial" o "anulado").

La Capa de Servicios realiza las operaciones necesarias para cumplir con las reglas de negocio antes de interactuar con la Capa de repositorio.

**public** **class** **ReservaService** : IReservaService

{

**public** **readonly** IReservaRepository \_reservaRepository;

**public** **ReservaService**(IReservaRepository reservaRepository) {

\_reservaRepository = reservaRepository;

}

**public** **bool** **GuardarReserva**(Reserva reserva, **out** **string** mensaje)

{

**try**

{

**if** (reserva.MontoTotal == reserva.MontoPagado && reserva.TipoPago== "parcial") {

reserva.TipoPago = "completo";

}

**if** (reserva.TipoPago == "completo" && reserva.MontoPagado!= **null**)

{

reserva.MontoPagado = **null**;

reserva.TipoPago = "completo";

}

**return** \_reservaRepository.GuardarReserva(reserva,**out** mensaje);

}

**catch** (Exception ex)

{

**throw** **new** **Exception**("Error: ", ex);

}

}

}

**Capa de Repositorio**

La Capa de Repositorio se encarga de la interacción directa con la base de datos. En esta capa se implementan las operaciones de acceso a los datos, como la inserción, actualización, eliminación y consulta de registros, utilizando procedimientos almacenados en SQL Server.

El repositorio ReservaRepository en este ejemplo es el encargado de realizar las operaciones relacionadas con las reservas en la base de datos. Esta clase delega las operaciones a los procedimientos almacenados definidos en la base de datos, simplificando el acceso a los datos y asegurando que el resto del sistema no tenga que lidiar directamente con las complejidades de la base de datos.

**public** **class** **ReservaRepository** : IReservaRepository

{

**private** **readonly** **string** \_connectionString;

**public** **ReservaRepository**(ConnectionService connectionService) {

\_connectionString = connectionService.ConnectionString;

}

**public** **bool** **GuardarReserva**(Reserva reserva,**out** **string** mensaje)

{

**try**

{

**using** **var** connection = **new** SqlConnection(\_connectionString);

**using** **var** command = **new** SqlCommand("SP\_RESERVA\_INSERT\_UPDATE", connection)

{

CommandType = CommandType.StoredProcedure

};

command.Parameters.Add("@Id", SqlDbType.Int, **4**).Value = reserva.Id;

command.Parameters.Add("@NombreCliente", SqlDbType.NVarChar,**50**).Value = reserva.NombreCliente;

command.Parameters.Add("@Fecha", SqlDbType.DateTime, **100**).Value = reserva.Fecha;

command.Parameters.Add("@HoraInicio", SqlDbType.Time).Value = reserva.HoraInicio;

command.Parameters.Add("@HoraFin", SqlDbType.Time).Value = reserva.HoraFin;

command.Parameters.Add("@MetodoPagoId", SqlDbType.Int,**4**).Value = reserva.MetodoPagoId;

command.Parameters.Add("@TipoPago", SqlDbType.NVarChar, **20**).Value = reserva.TipoPago;

command.Parameters.Add("@CanchaId", SqlDbType.Int).Value = reserva.CanchaId;

command.Parameters.Add("@MontoTotal", SqlDbType.Decimal).Value = reserva.MontoTotal ;

command.Parameters.Add("@MontoPagado", SqlDbType.Decimal).Value = reserva.MontoPagado;

command.Parameters.Add("@Telefono", SqlDbType.NVarChar, **10**).Value = reserva.Telefono;

command.Parameters.Add("@Usuario", SqlDbType.Int, **4**).Value = reserva.UsuarioModifica;

**var** resultadoParam = **new** SqlParameter("@Resultado", SqlDbType.Int) { Direction = ParameterDirection.Output };

command.Parameters.Add(resultadoParam);

connection.Open();

command.ExecuteNonQuery();

**int** resultado = resultadoParam.Value != DBNull.Value ? Convert.ToInt32(resultadoParam.Value) : -**99**;

**switch** (resultado)

{

**case** **1**:

mensaje = "Reserva creada con éxito."; **return** **true**;

**case** **2**:

mensaje = "Reserva actualizada con éxito."; **return** **true**;

**case** -**1**:

mensaje = "Conflicto: ya existe una reserva en ese horario."; **return** **false**;

**default**:

mensaje = "Error inesperado."; **return** **false**;

}

}

**catch** (Exception ex)

{

**throw** **new** **Exception**("Error: ", ex);

}

}

# Conclusiones

El desarrollo del sistema web de gestión de reservas para El Relámpago ha demostrado ser una solución efectiva para optimizar los procesos administrativos internos y mejorar la experiencia tanto de los clientes como del personal administrativo. A través de las pruebas realizadas hasta el momento, el sistema ha cumplido con los objetivos establecidos en las fases iniciales del proyecto. A continuación, se destacan los aspectos más relevantes que se han logrado hasta ahora:

1. **Análisis de las Deficiencias y Requerimientos del Sistema:** mediante un análisis exhaustivo del proceso de reservas actual, se identificaron varias deficiencias, como la falta de un sistema centralizado, errores humanos ocasionados por procesos manuales y la falta de visibilidad en tiempo real de la disponibilidad de las canchas. Con base en este análisis, se definieron los requerimientos necesarios para una solución más eficiente y organizada, orientada a automatizar la gestión de reservas y eliminar las duplicidades reservas, mejorando así la precisión y la rapidez en la atención a los clientes.
2. **La Eficiencia Operativa Mejorada**: El sistema ha demostrado un potencial significativo para mejorar la eficiencia operativa al automatizar tareas clave, como la verificación de la disponibilidad, la gestión de pagos por reservas y la asignación de horarios. En las pruebas realizadas, ha sido posible gestionar las reservas, modificar o anular reservas y hacer un seguimiento de manera más ágil y precisa. La eficiencia operativa se ha incrementado al eliminar tareas repetitivas y al utilizar un sistema que actualiza y registra la información de forma adecuada. No obstante, su impacto real en el entorno operativo completo deberá validarse al ser implementado en condiciones reales con los usuarios finales.
3. **Diseño de una Solución Web Eficaz:** El diseño de la solución web fue centrado en la modularidad y accesibilidad, permitiendo a los usuarios registrar, gestionar y consultar las reservas de las canchas en tiempo real. Esto ha logrado una mejora notable en la interacción del usuario con el sistema, y la arquitectura en capas (utilizando ASP.NET Core y C#) ha garantizado un sistema robusto y escalable. La solución se adaptó perfectamente a las necesidades del negocio, asegurando que los horarios no se solapen y que las reservas se gestionen correctamente.
4. **Implementación de un Sistema Funcional y Adaptable**: La implementación del sistema ha permitido gestionar las reservas en tiempo real, lo que ha optimizado la disponibilidad de las canchas. Este avance ha mejorado tanto la experiencia del cliente como la gestión interna, ya que los administradores pueden ahora verificar la disponibilidad de las canchas de manera eficiente y precisa. Además, la adaptabilidad del sistema a futuras necesidades del negocio ha sido fundamental, permitiendo futuras ampliaciones con características como la integración con pasarelas de pago u otras funcionalidades.
5. **Validación y Pruebas del Sistema**: Durante el proceso de desarrollo, el sistema ha pasado por pruebas en un entorno controlado que han garantizado que cumple con los requisitos funcionales. Estas pruebas abarcan aspectos clave como la verificación de la disponibilidad en tiempo real de las canchas, la gestión de cancelación de reservas o su anulación, las pruebas realizas han demostrado que el sistema es fiable, rápido y seguro en un ambiente de pruebas controlado, sin embargo, la validación final debe realizarse en el entorno de producción con los usuarios reales para confirmar su efectividad en condiciones operativas completas.

# Recomendaciones

: A partir de los avances obtenidos durante el desarrollo y las pruebas preliminares del sistema, se proponen las siguientes recomendaciones para optimizar su funcionamiento y asegurar una implementación efectiva cuando el sistema sea finalmente puesto en uso por los trabajadores. Estas recomendaciones están orientadas a mejorar la eficiencia operativa, la experiencia del usuario y la escalabilidad del sistema, de modo que esté completamente preparado para satisfacer las necesidades del negocio.

1. **Capacitación Continua para los Usuarios**: Aunque el sistema ha sido diseñado con un enfoque intuitivo, es fundamental ofrecer sesiones periódicas de capacitación para los usuarios del sistema. Esto asegurará que todos los usuarios comprendan completamente las funcionalidades y características del sistema, lo que contribuirá a mejorar el desempeño operativo, reducir la curva de aprendizaje y minimizar posibles errores humanos. Además, la capacitación constante permitirá adaptar al personal a futuras actualizaciones del sistema de manera más eficiente.
2. **Mantenimiento y Actualización Regular**: Es esencial establecer un plan de mantenimiento preventivo para monitorear el rendimiento del sistema y resolver cualquier posible problema de forma proactiva. Además, se recomienda realizar actualizaciones regulares para incorporar nuevas funcionalidades que sigan mejorando la eficiencia, como la integración de pagos en línea o facturación electrónica para los clientes.
3. **Mejoras en la Personalización para Clientes Frecuentes**: Aunque actualmente el sistema es operado únicamente por el personal administrativo, se recomienda considerar, en una fase futura del proyecto, el desarrollo de funcionalidades orientadas a la personalización de la experiencia para clientes frecuentes. Esto podría incluir la creación de perfiles individuales con datos relevantes como preferencias de horarios, historial de reservas, métodos de pago habituales y recomendaciones personalizadas, lo cual permitiría mejorar la atención y eficiencia del servicio. Asimismo, la implementación de un sistema de fidelización, basado en puntos o recompensas por uso recurrente, podría incentivar la continuidad de los clientes y aumentar la satisfacción general. Esta recomendación deberá evaluarse en función de la posible evolución del sistema hacia una interfaz de uso directo por parte del cliente final.
4. **Incorporación de un sistema de notificaciones automáticas**: Se recomienda, en futuras versiones del sistema, la implementación de un mecanismo de notificaciones automáticas mediante mensajería como WhatsApp. Esta funcionalidad permitiría enviar recordatorios de reservas, confirmaciones, notificaciones de cambios o cancelaciones, entre otros. La integración de este tipo de alertas contribuiría significativamente a optimizar la comunicación con los clientes, reducir ausencias por olvido y mejorar la puntualidad, generando así una experiencia más eficiente y confiable para los clientes.

Bibliografía**:**  
Laudon, K. C., & Laudon, J. P. (2020). Sistemas de Información Gerenciales (16ª ed.). Pearson.

Sommerville, I. (2016). Ingeniería de software (10.ª ed.). Pearson.

Pressman, R. S. (2014). Ingeniería de software: Un enfoque práctico (7.ª ed.). McGraw-Hill.

Sommerville, I. (2016). Ingeniería de software (10.ª ed.). Pearson.

Microsoft Docs. (2023). ASP.NET Core documentation. <https://learn.microsoft.com/en-us/aspnet/core/>

Microsoft Docs. (2023). ASP.NET Core Identity documentation. <https://learn.microsoft.com/en-us/aspnet/core/security/authentication/identity>

Microsoft Docs. (2023). SQL Server documentation. <https://learn.microsoft.com/en-us/sql/sql-server/>

Ambler, S. (2004). The Object Primer: Agile Model-Driven Development with UML 2.

 Esposito, D. (2018). Programming Microsoft ASP.NET MVC (5th ed.). Microsoft Press.

 Flatpickr. (2023). Flatpickr documentation. https://flatpickr.js.org/

 FullCalendar. (2023). FullCalendar documentation. https://fullcalendar.io/docs

 Web Compiler. (2023). Web Compiler tool. <https://github.com/madskristensen/WebCompiler>