

**UNIVERSIDAD AUTÓNOMA DEL
ESTADO DE MÉXICO**
Centro Universitario UAEMex
Atlacomulco

**REPORTE DE PROYECTO FINAL DE
SEMESTRE**

**Asignatura: Paradigmas de la
Programación I**

Docente: Julio Alberto de la Teja López

Equipo:

Fernando Azuara Ibarra

Omar Andreí García López

Oswaldo Plata Navarrete

Jenifer Ramírez Ibarra

Jiovani Asael Tapia López

Ingeniería en Computación ICO 28

Fecha: 26 de Noviembre del 2024

Índice de Contenido

| | |
|--|----|
| Índice de Contenido | 2 |
| 1. Resumen | 3 |
| 2. Introducción | 4 |
| 3. Marco Teórico | 4 |
| 3.1 Python-Flask | 4 |
| 3.2 Uso del Modelo MVC | 5 |
| 3.3 Uso de MySQL usando XAMPP | 6 |
| 3.4 Framework Bootstrap | 7 |
| 4. Desarrollo | 8 |
| 4.1 Estructura del proyecto | 8 |
| 4.2 Diagramas UML | 9 |
| 4.2.1 Diagrama casos de uso | 9 |
| 4.2.2. Diagrama Sub-Escenarios | 10 |
| 4.2.3. Diagrama de Secuencias | 11 |
| 4.2.4. Diagrama de Clases | 12 |
| 4.2.5. Diagrama de Colaboración | 13 |
| 4.2.6. Diagrama de Estado | 13 |
| 4.3 Creación de base de datos | 14 |
| 4.4 Conexión con base de datos | 16 |
| 4.5 Estructura Front-End | 17 |
| 4.5.1 Creación de página de administrador | 18 |
| 4.5.2 Subpáginas de administrador | 19 |
| 4.5.3 Creación de página de inicio de sesión | 26 |
| 4.5.4 Creación de página de cliente | 30 |
| 4.6 Estructura Back-End | 36 |
| 4.6.1 Funciones para página de administrador | 37 |
| 4.6.2 Funciones para subpáginas de administrador | 38 |
| 4.6.3 Funciones para página de inicio de sesión | 40 |
| 4.6.4 Funciones para página de cliente | 41 |
| 4.7 Mejoras de examen ordinario | 43 |
| 4.8 . Errores Comunes | 55 |
| 5. Resultados | 55 |
| 5.1 Funcionalidades Implementadas | 56 |
| 5.2 Resultados del Sistema | 60 |
| 7. Link de PythonAnywhere | 62 |
| 8. Link del video de demostración: | 62 |
| 9. Conclusiones | 62 |
| 10. Bibliografía | 63 |

1. Resumen

El presente proyecto tiene como objetivo desarrollar un sistema web para la administración de un gimnasio, utilizando tecnologías modernas que permitan gestionar eficientemente los datos relacionados con clases, clientes, membresías, pagos y personal. Este sistema busca optimizar los procesos administrativos y mejorar la experiencia de los usuarios, tanto del personal administrativo como de los clientes.

El desarrollo del proyecto se llevó a cabo utilizando Python Flask como framework backend, implementando el patrón de diseño Modelo-Vista-Controlador (MVC) para garantizar una estructura organizada y modular del código. Además, se empleó MySQL como sistema de gestión de bases de datos, facilitando el almacenamiento, consulta y manipulación de la información. El diseño de la interfaz se desarrolló con Bootstrap, asegurando una experiencia de usuario moderna, responsive y visualmente atractiva.

Las principales funcionalidades del sistema incluyen:

- ✓ Inicio de sesión para administrador y cliente.
- ✓ Sistema interactivo para el cliente.
- ✓ Gestión de clientes: Registro, actualización y consulta de datos personales.
- ✓ Administración de clases: Creación y edición de horarios, instructores y capacidades.
- ✓ Control de membresías: Definición de tipos, costos y duraciones.
- ✓ Gestión de pagos: Registro y verificación de transacciones realizadas por los clientes.
- ✓ Administración de personal: Gestión de datos del equipo de trabajo, incluyendo roles y turnos.

El proyecto se diseñó pensando en la escalabilidad y facilidad de uso, permitiendo adaptarse a las necesidades específicas de cualquier gimnasio. Asimismo, se empleó XAMPP para configurar el entorno local, facilitando el desarrollo y pruebas del sistema.

En términos visuales, se incluyó un carrusel interactivo en la página principal, junto con formularios de registro e inicio de sesión para usuarios y administradores, utilizando ventanas modales que optimizan la interacción. El sistema también integra mensajes flash para notificar a los usuarios sobre acciones realizadas.

2. Introducción

En la actualidad, la gestión eficiente de un gimnasio representa un desafío significativo debido a la cantidad de información que se debe manejar, como el control de clases, registros de clientes, membresías, pagos y la administración del personal. La automatización de estos procesos mediante el uso de sistemas tecnológicos no solo optimiza las tareas administrativas, sino que también mejora la experiencia de los usuarios al brindar un acceso más rápido y sencillo a los servicios ofrecidos.

Este proyecto tiene como propósito desarrollar un sistema web para la administración integral de un gimnasio, utilizando tecnologías modernas como Python Flask para el backend, Bootstrap para el diseño de la interfaz y MySQL para la gestión de la base de datos. El sistema está estructurado bajo el patrón de diseño Modelo-Vista-Controlador (MVC), lo que garantiza una organización modular y facilita su mantenimiento y escalabilidad.

El sistema desarrollado permite a los administradores gestionar datos de clientes, clases, membresías, pagos y personal de manera eficiente. Además, incluye funcionalidades intuitivas para el registro de nuevos clientes, el inicio de sesión de usuarios y administradores, y un carrusel visual en la página principal para mejorar la interacción con los usuarios.

Con este proyecto, se busca cubrir la necesidad de los gimnasios de contar con herramientas tecnológicas accesibles, efectivas y adaptadas a sus requerimientos. Esto no solo simplifica la administración, sino que también promueve una relación más cercana con los clientes al ofrecerles una plataforma moderna y práctica.

En este documento se detalla el proceso de desarrollo del sistema, describiendo las herramientas y metodologías empleadas, así como los resultados obtenidos. A través de este trabajo, se demuestra cómo la integración de tecnología en la gestión de gimnasios puede transformar procesos tradicionales en soluciones innovadoras y eficaces.

3. Marco Teórico

3.1 Python-Flask

Flask es un microframework de desarrollo web para Python. Fue diseñado para ser ligero y modular, permitiendo a los desarrolladores crear aplicaciones web escalables y funcionales. Flask sigue una filosofía "minimalista", lo que significa que incluye las herramientas necesarias

para empezar rápidamente, pero es lo suficientemente flexible como para extenderse según las necesidades del proyecto.

Características principales de Flask:

- Ligereza y modularidad: Flask no incluye herramientas que el desarrollador no necesite, reduciendo la complejidad innecesaria.
- Motor de plantillas Jinja2: Facilita la creación de páginas web dinámicas mediante la separación de la lógica y el diseño.
- Extensibilidad: Se pueden integrar librerías y extensiones fácilmente, como soporte para bases de datos o autenticación de usuarios.
- Desarrollo rápido: Con una curva de aprendizaje baja, permite a los desarrolladores construir aplicaciones de manera eficiente.
- Soporte para pruebas: Incluye un servidor de desarrollo y un sistema de pruebas integrado para garantizar la calidad del código.

En este proyecto, Flask permitió la implementación de una arquitectura robusta y modular, facilitando la conexión entre las rutas, las vistas HTML, y la base de datos MySQL.

3.2 Uso del Modelo MVC

El patrón de diseño Modelo-Vista-Controlador (MVC) es una arquitectura utilizada para separar las responsabilidades en el desarrollo de aplicaciones. Este modelo es especialmente útil en aplicaciones web, ya que organiza el código para mejorar la mantenibilidad y escalabilidad del sistema.

Componentes del MVC:

1. **Modelo:**

Representa los datos y la lógica del negocio. En este proyecto, el modelo incluye las tablas de la base de datos y su interacción con la aplicación a través de consultas SQL.

2. **Vista:**

Es la interfaz de usuario, encargada de mostrar los datos al usuario y recibir sus interacciones. En este caso, se usaron plantillas HTML junto con Bootstrap para un diseño responsive.

3. Controlador:

Gestiona la lógica de la aplicación, actuando como intermediario entre el modelo y la vista. En Flask, los controladores se implementan en el archivo app.py, donde se definen las rutas y funciones que procesan las peticiones del usuario.

El uso del patrón MVC en este proyecto permitió una estructura clara y organizada del código, facilitando la colaboración y la futura escalabilidad de la aplicación.

3.3 Uso de MySQL usando XAMPP

MySQL es un sistema de gestión de bases de datos relacional (RDBMS) que se utiliza ampliamente en aplicaciones web debido a su velocidad, confiabilidad y flexibilidad. Este sistema permite almacenar y administrar grandes volúmenes de datos mediante el uso de tablas relacionadas.

XAMPP, por su parte, es un paquete de software que incluye Apache, MySQL, PHP, y Perl. Es una herramienta esencial para configurar un entorno de desarrollo local, proporcionando un servidor MySQL para gestionar bases de datos y probar aplicaciones web.

Ventajas de usar MySQL con XAMPP:

- Fácil instalación: XAMPP configura automáticamente MySQL junto con el servidor Apache, simplificando el proceso de preparación del entorno.
- Interfaz gráfica: Herramientas como *phpMyAdmin* permiten gestionar bases de datos MySQL mediante una interfaz gráfica intuitiva.
- Compatibilidad: MySQL funciona perfectamente con frameworks como Flask para manejar consultas y operaciones CRUD (Crear, Leer, Actualizar y Eliminar).
- Pruebas locales: Con XAMPP, el servidor y la base de datos están disponibles de forma local, lo que elimina la necesidad de estar conectado a un servidor remoto durante el desarrollo inicial.

En este proyecto, MySQL se utilizó para almacenar los datos relacionados con el gimnasio, como clientes, clases, membresías y pagos. La conexión entre Flask y MySQL se gestionó mediante un archivo dedicado (database.py), que empleó la librería *MySQL-Connector-Python* para ejecutar consultas y operaciones en la base de datos.

3.4 Framework Bootstrap

Bootstrap es un framework de código abierto para el diseño y desarrollo web, creado por los desarrolladores de Twitter. Se utiliza ampliamente para construir sitios y aplicaciones web responsivos debido a su simplicidad y conjunto de herramientas predefinidas. Este framework combina CSS, JavaScript y HTML para proporcionar componentes reutilizables que ahorran tiempo y garantizan consistencia en los diseños.

Características principales de Bootstrap:

1. Diseño responsivo:

Utiliza un sistema de rejilla flexible que permite crear diseños adaptables a diferentes tamaños de pantalla, desde dispositivos móviles hasta escritorios.
 2. Componentes predefinidos:

Incluye elementos como botones, tarjetas, formularios, modales, menús de navegación, alertas y más, que pueden personalizarse fácilmente.
 3. Compatibilidad multiplataforma:

Es compatible con todos los navegadores modernos, asegurando una experiencia de usuario consistente.
 4. Extensiones con JavaScript:

Proporciona funcionalidades avanzadas, como carruseles, tooltips y ventanas modales, usando JavaScript y su biblioteca integrada.
 5. Facilidad de personalización:

Los temas y estilos de Bootstrap pueden personalizarse para adaptarse a la identidad visual de cualquier proyecto.

Aplicación en el proyecto:

En este proyecto, Bootstrap desempeñó un papel clave en la creación de una interfaz de usuario atractiva y funcional. Algunos ejemplos del uso de Bootstrap incluyen:

- Sistema de rejilla:
Se utilizó para organizar elementos en filas y columnas, garantizando un diseño limpio y alineado en todas las páginas del sitio.

- Botones y formularios:
Se personalizaron botones y campos de entrada para los formularios de registro e inicio de sesión, mejorando la experiencia del usuario.
- Carrusel:
Un carrusel dinámico permitió mostrar imágenes animadas relacionadas con el gimnasio, haciendo la página más interactiva.
- Ventanas modales:
Estas se implementaron para los formularios de registro e inicio de sesión, permitiendo que los usuarios interactúen sin abandonar la página principal.

Ventajas de usar Bootstrap en este proyecto:

- Rapidez en el diseño:
Bootstrap simplificó la creación de estilos y componentes, reduciendo el tiempo necesario para desarrollar la interfaz.
- Consistencia visual:
Todos los elementos del sitio comparten un estilo uniforme, mejorando la estética general.
- Responsividad integrada:
El diseño se adaptó automáticamente a diferentes dispositivos, garantizando accesibilidad para todos los usuarios.

El uso de Bootstrap, junto con Python Flask y MySQL, permitió construir un sistema web moderno, eficiente y visualmente atractivo.

4. Desarrollo

4.1 Estructura del proyecto

Carpeta src

- static/: Contiene los recursos estáticos.
- blog/: Imágenes utilizadas en indexclientes.html.
- clientes/: Almacena imágenes subidas al registrar clientes.
- uploads/: Contiene imágenes relacionadas con tablas como "clases" y "membresías".
- styles.css: Archivo de estilos CSS para las plantillas HTML.
- Otros recursos: Gifs e imágenes generales.

templates/: Almacena las plantillas HTML.

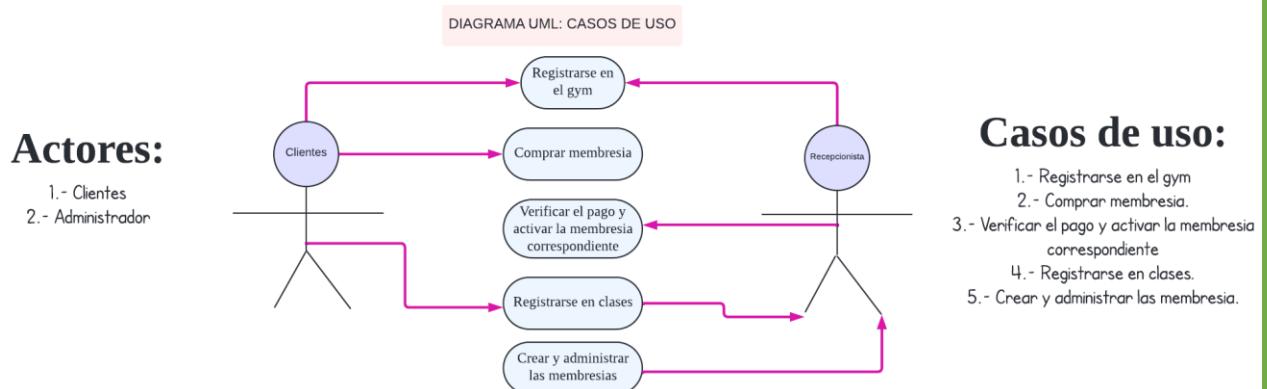
- **clases.html**: CRUD para administrar clases.
- **clientes.html**: CRUD para clientes.
- **indexadmin.html**: Página inicial para administradores con enlaces a los CRUD.
- **indexclientes.html**: Página de inicio para clientes tras iniciar sesión.
- **indexlogin.html**: Página principal para iniciar sesión o registrarse.
- **membresias.html**: CRUD para membresías.
- **pagos.html**: CRUD para pagos.
- **personal.html**: CRUD para personal.

app.py: Archivo principal del backend, con la lógica de la aplicación.

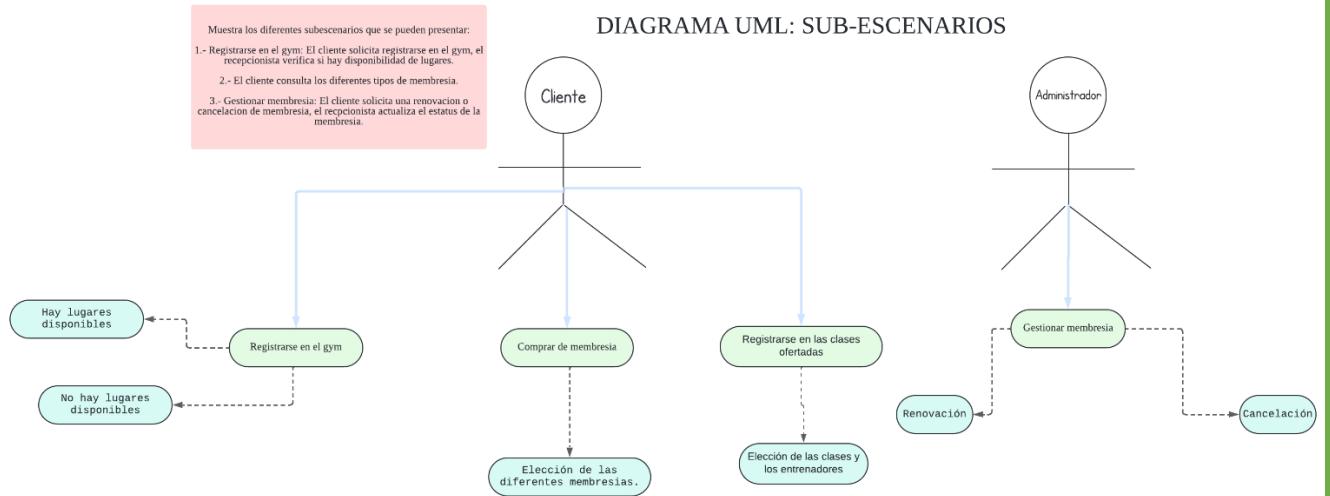
PasswordGen.py: Script para generar contraseñas y sentencias SQL para administrar usuarios.

4.2 Diagramas UML

4.2.1 Diagrama casos de uso

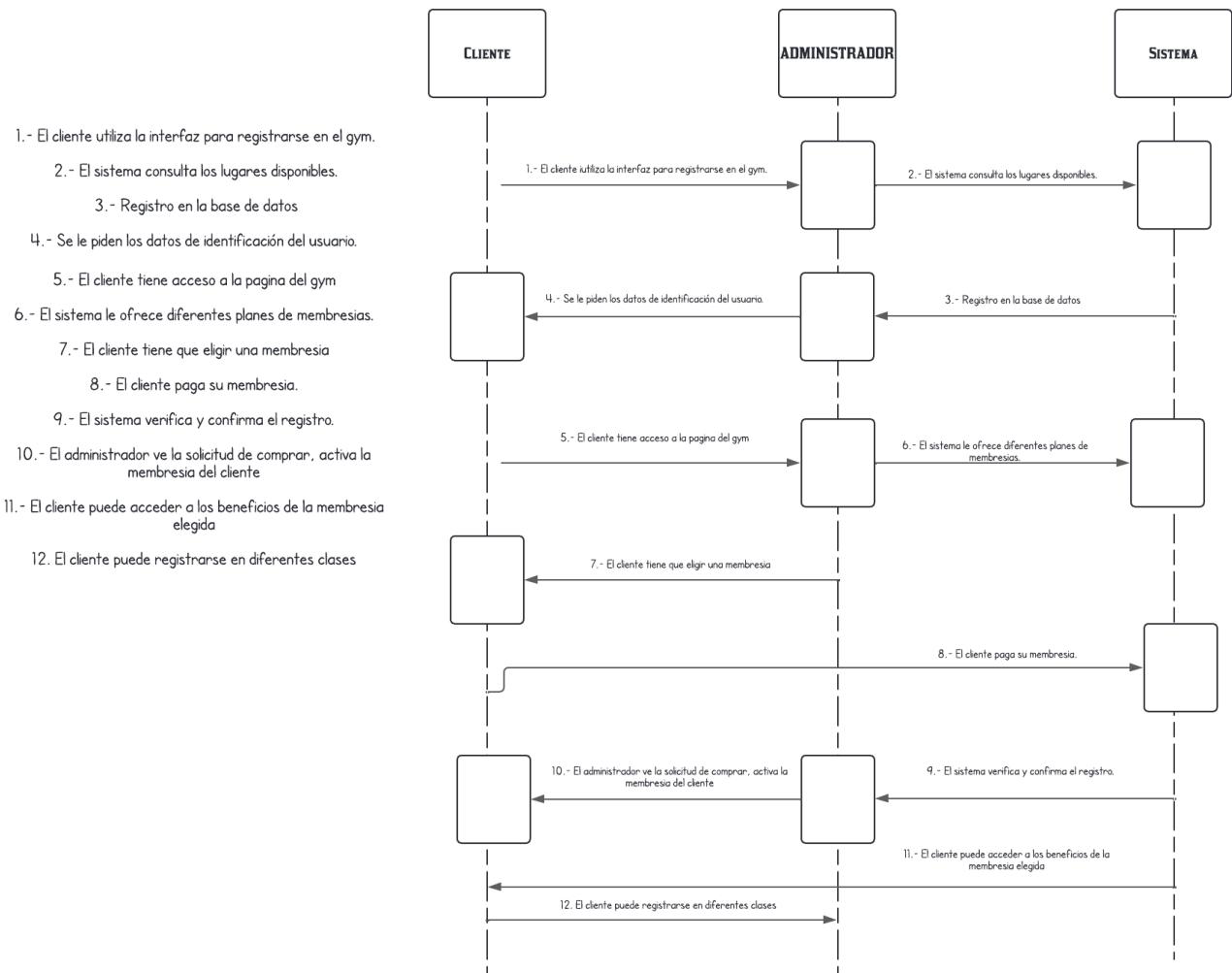


4.2.2. Diagrama Sub-Escenarios.



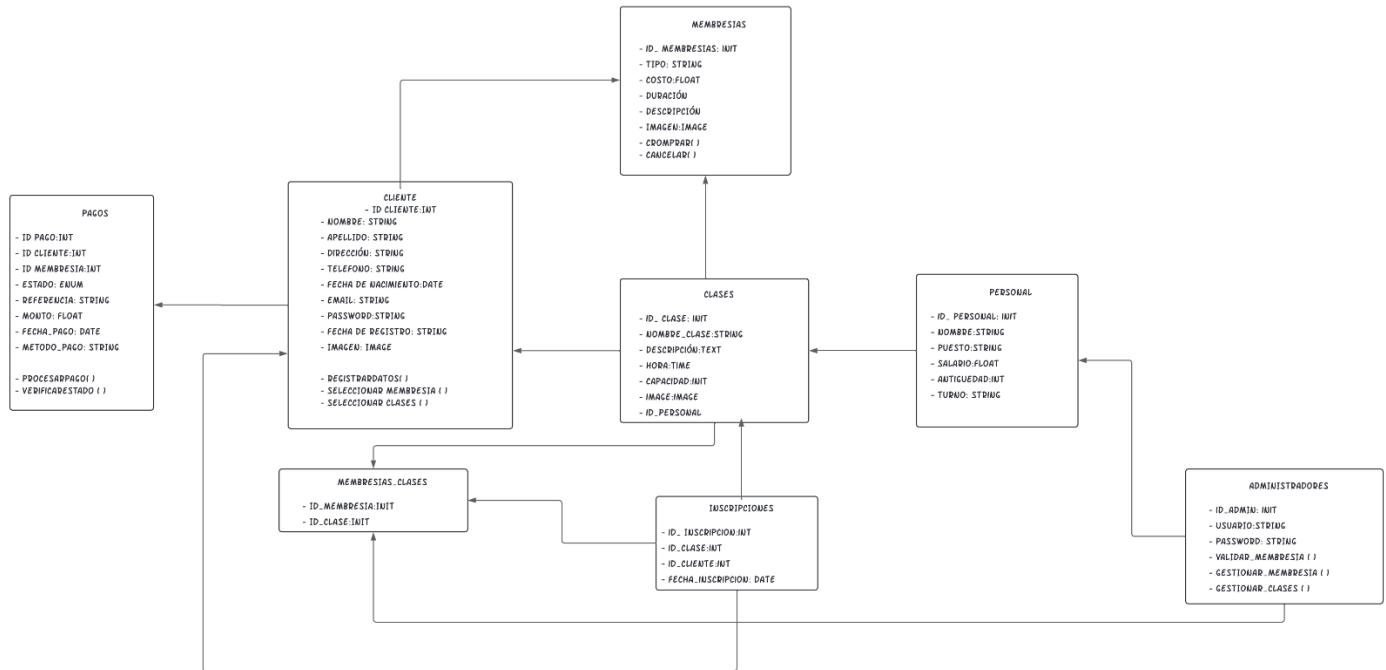
4.2.3. Diagrama de Secuencias

DIAGRAMA UML: SECUENCIAS



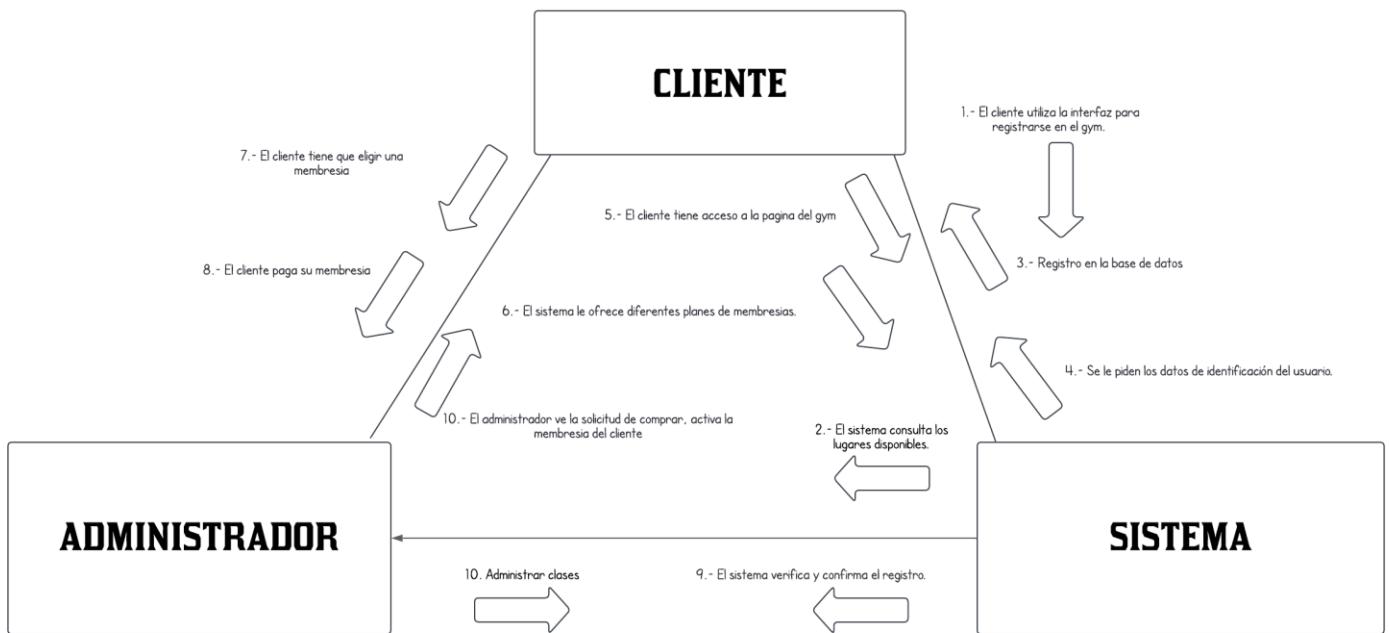
4.2.4. Diagrama de Clases

DIAGRAMA UML: DIAGRAMA DE CLASES



4.2.5. Diagrama de Colaboración

DIAGRAMA UML:COLABORACIÓN



4.2.6. Diagrama de Estado

DIAGRAMA UML: ESTADO



4.3 Creación de base de datos

Primero, se debe crear la base de datos donde se almacenarán las tablas.

```
CREATE DATABASE gimnasio CHARACTER SET utf8mb4  
COLLATE utf8mb4_general_ci;  
USE gimnasio;
```

- ✓ CREATE DATABASE gimnasio: Crea la base de datos con el nombre gimnasio.
- ✓ CHARACTER SET utf8mb4: Define el conjunto de caracteres para admitir texto multilingüe.
- ✓ USE gimnasio: Selecciona la base de datos creada para empezar a trabajar.

```

CREATE TABLE `administradores` (
  `ID_Admin` int(11) NOT NULL AUTO_INCREMENT,
  `User` varchar(50) NOT NULL,
  `Password` varchar(255) NOT NULL,
  PRIMARY KEY (`ID_Admin`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

CREATE TABLE `clases` (
  `ID_Clase` int(11) NOT NULL AUTO_INCREMENT,
  `Nombre_Clase` varchar(50) NOT NULL,
  `Descripcion` text DEFAULT NULL,
  `Instructor` varchar(50) DEFAULT NULL,
  `Hora` time DEFAULT NULL,
  `Capacidad` int(11) NOT NULL,
  `Imagen` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`ID_Clase`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

CREATE TABLE `clientes` (
  `ID_Cliente` int(11) NOT NULL AUTO_INCREMENT,
  `Nombre` varchar(50) NOT NULL,
  `Apellido` varchar(50) NOT NULL,
  `Fecha_Nacimiento` date NOT NULL,
  `Telefono` varchar(15) DEFAULT NULL,
  `Email` varchar(100) DEFAULT NULL,
  `password` varchar(255) NOT NULL,
  `Direccion` varchar(150) DEFAULT NULL,
  `Fecha_Registro` date NOT NULL DEFAULT curdate(),
  `Imagen` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`ID_Cliente`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

CREATE TABLE `membresias` (
  `ID_Membresia` int(11) NOT NULL AUTO_INCREMENT,
  `Tipo` varchar(50) NOT NULL,
  `Costo` decimal(10,2) NOT NULL,
  `Duracion` int(11) NOT NULL,
  `Descripcion` text DEFAULT NULL,
  `Imagen` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`ID_Membresia`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

CREATE TABLE `pagos` (
  `ID_Pago` int(11) NOT NULL AUTO_INCREMENT,
  `ID_Cliente` int(11) DEFAULT NULL,
  `ID_Membresia` int(11) DEFAULT NULL,
  `Fecha_Pago` date NOT NULL DEFAULT curdate(),
  `Monto` decimal(10,2) NOT NULL,
  `Metodo_Pago` varchar(50) DEFAULT NULL,
  PRIMARY KEY (`ID_Pago`),
  KEY `ID_Cliente`(`ID_Cliente`),
  KEY `ID_Membresia`(`ID_Membresia`),
  CONSTRAINT `pagos_ibfk_1` FOREIGN KEY (`ID_Cliente`) REFERENCES `clientes` (`ID_Cliente`),
  CONSTRAINT `pagos_ibfk_2` FOREIGN KEY (`ID_Membresia`) REFERENCES `membresias` (`ID_Membresia`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

CREATE TABLE `personal` (
  `ID_Personal` int(11) NOT NULL AUTO_INCREMENT,
  `Nombre` varchar(50) NOT NULL,
  `Puesto` varchar(50) NOT NULL,
  `Salario` decimal(10,2) NOT NULL,
  `Antiguedad` int(11) DEFAULT NULL,
  `Turno` varchar(20) DEFAULT NULL,
  PRIMARY KEY (`ID_Personal`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

```

- ✓ administradores: Guarda credenciales de acceso para los administradores.: Columnas: ID_Admin, User, Password.
- ✓ clases: Registra las clases ofrecidas en el gimnasio.: Columnas: ID_Clase, Nombre_Clase, Descripcion, Instructor, Hora, Capacidad, Imagen.
- ✓ clientes: Almacena datos personales y credenciales de los clientes: Columnas: ID_Cliente, Nombre, Apellido, Fecha_Nacimiento, Telefono, Email, password, Direccion, Fecha_Registro, Imagen.
- ✓ membresias: Define los tipos de membresías disponibles: Columnas: ID_Membresia, Tipo, Costo, Duracion, Descripcion, Imagen.
- ✓ pagos: Registra los pagos realizados por los clientes: Columnas: ID_Pago, ID_Cliente, ID_Membresia, Fecha_Pago, Monto, Metodo_Pago.
- ✓ personal: Contiene información sobre el personal que labora en el gimnasio: Columnas: ID_Personal, Nombre, Puesto, Salario, Antiguedad, Turno.

4.4Conexión con base de datos

En esta sección, nos aseguramos de establecer una conexión segura y eficiente con la base de datos MySQL para interactuar con los datos del gimnasio. Utilizamos la biblioteca mysql.connector para conectar el servidor de nuestra aplicación Flask con MySQL.

Importación de la Librería: Primero, aseguramos la importación de la librería mysql.connector necesaria para interactuar con MySQL.

```
pip install mysql-connector-python
```

Función get_db_connection(): La función get_db_connection() es responsable de establecer y devolver la conexión con la base de datos de manera eficiente y reutilizable. Esta función encapsula la lógica de conexión para que pueda ser llamada siempre que se necesite interactuar con la base de datos.

```
● ● ●

from mysql.connector import connect, Error

def get_db_connection():
    return connect(
        host="localhost", # Dirección del servidor MySQL, "localhost" en este caso
        user="root", # Nombre de usuario para acceder a MySQL (por defecto, "root")
        password="", # Contraseña para el usuario, en este caso vacía para la instalación local
        database="Gimnasio" # El nombre de la base de datos a la que nos conectamos
    )
```

Cierre de Conexión: Para evitar fugas de memoria y problemas de rendimiento, siempre nos aseguramos de cerrar la conexión con la base de datos después de realizar las operaciones necesarias. Este cierre se realiza con connection.close().

```
● ● ●

connection = get_db_connection()
if connection:
    # Realizamos alguna operación con la base de datos
    cursor = connection.cursor()
    cursor.execute("SELECT * FROM clientes")
    result = cursor.fetchall()
    print(result)

    # Cerramos la conexión
    connection.close()
```

4.5 Estructura Front-End

En esta sección, nos aseguramos de estructurar correctamente la parte visual de la aplicación utilizando HTML, CSS y JavaScript. Implementamos Bootstrap 5 para facilitar la creación de un diseño responsive y moderno. Las vistas HTML se organizan en carpetas dentro del directorio templates/, donde cada archivo corresponde a una funcionalidad específica (páginas

como index.html, clientes.html, etc.). Además, se utilizan archivos CSS personalizados dentro de la carpeta static/ para añadir estilos específicos a la interfaz. La interacción con el usuario es mejorada con JavaScript, especialmente para validar formularios y manejar dinámicamente las operaciones CRUD.

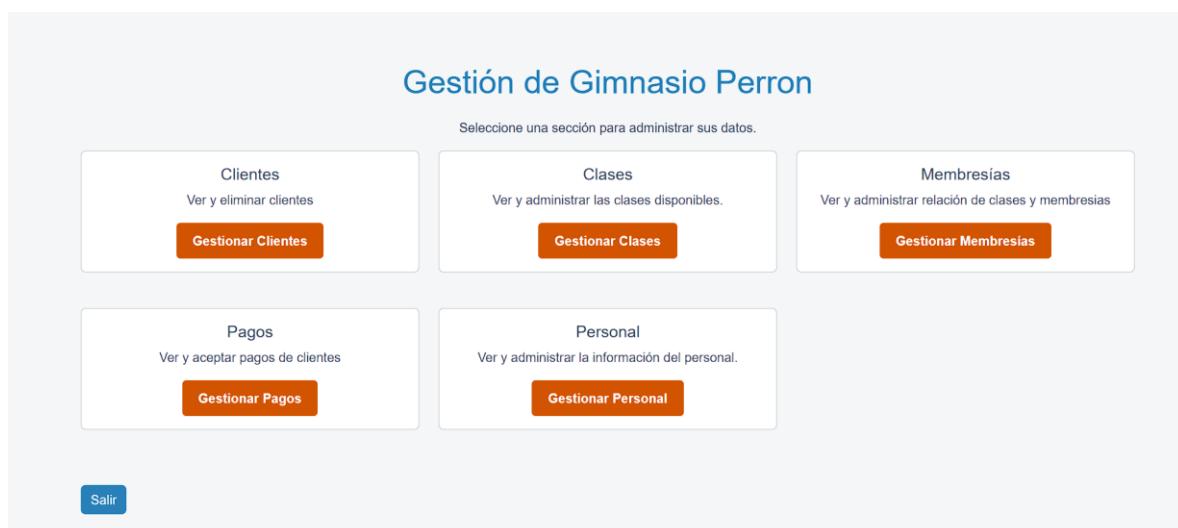
4.5.1 Creación de página de administrador

Indexadmin.html

Link del código: [https://github.com/igream/p2-flask-
SQL/blob/main/Interfaces%20Equipo/src/templates/indexadmin.html](https://github.com/igream/p2-flask-SQL/blob/main/Interfaces%20Equipo/src/templates/indexadmin.html)

La página de administrador es una interfaz que permite gestionar las principales secciones del gimnasio. Esta página está construida con HTML y utiliza Bootstrap 5 para un diseño limpio y responsive. En el cuerpo del documento, se incluyen varios enlaces dentro de tarjetas, cada una representando una sección que el administrador puede gestionar: *Clientes, Clases, Membresías, Pagos y Personal*. Cada tarjeta incluye un breve texto explicativo y un botón que redirige al usuario a las respectivas páginas de gestión de datos. La estructura está organizada en filas y columnas para facilitar la visualización y el acceso rápido a las diferentes funciones. Además, se proporciona un enlace de "Salir" que redirige a la página principal del sitio.

El resultado es este:



4.5.2 Subpáginas de administrador

Clases.html

Link del código:[https://github.com/igream/p2-flask-
SQL/blob/main/Interfaces%20Equipo/src/templates/clases.html](https://github.com/igream/p2-flask-SQL/blob/main/Interfaces%20Equipo/src/templates/clases.html)

La página de gestión de clases está diseñada para permitir al administrador ver, agregar, editar y eliminar clases dentro del gimnasio. Utiliza un diseño con Bootstrap 4 para la disposición visual, incluyendo una tabla que muestra la lista de clases con columnas para el nombre, descripción, instructor, hora, capacidad y una imagen asociada. En la misma página, se encuentra un formulario en el lateral derecho para agregar nuevas clases, que incluye campos para el nombre, descripción, instructor, hora, capacidad y una imagen opcional. Además, cada clase tiene un botón de "Editar" que abre un modal donde se pueden modificar los detalles de la clase. Las acciones de edición y eliminación están asociadas con formularios que interactúan con el backend para actualizar o eliminar los registros correspondientes en la base de datos. La estructura es eficiente y facilita la gestión de las clases con opciones claras de acción.

El resultado es este:

Gestión de Clases

Volver al Índice

Agregar Nueva Clase

Nombre: Descripción: Instructor: Hora: Capacidad:

Imagen: Ningún arc...leccinado Agregar Clase

| ID | Nombre | Descripción | Instructor | Hora | Capacidad | Imagen | Acciones |
|----|-------------------------|--|-------------|----------|-----------|---|---|
| 7 | Ejercicio Aerobico | El ejercicio aeróbico promueve niveles de colesterol saludables. Entre los aeróbicos de bajo impacto están el caminar y nadar. Correr, jugar tenis y bailar son aeróbicos de alto impacto. | Baki Hanma | 22:00:00 | 20 |  | <button>Editar</button> <button>Eliminar</button> |
| 8 | Entrenamiento de Fuerza | Es una rutina deportiva que busca fortalecer los músculos del cuerpo. Para ello, se utilizan distintos métodos de resistencia, como pesas, máquinas, bandas elásticas o el propio peso del cuerpo. | Sin asignar | 23:00:00 | 10 |  | <button>Editar</button> <button>Eliminar</button> |

Clientes.html

Link del código: [https://github.com/igream/p2-flask-
SQL/blob/main/Interfaces%20Equipo/src/templates/clientes.html](https://github.com/igream/p2-flask-SQL/blob/main/Interfaces%20Equipo/src/templates/clientes.html)

Esta página permite gestionar los clientes del gimnasio mediante un diseño limpio y estructurado con Bootstrap 5. Los elementos clave de la página son:

1. Formulario de Registro de Cliente: Ubicado en la parte superior, contiene campos para registrar un nuevo cliente, incluyendo nombre, apellido, fecha de nacimiento, teléfono, email y dirección. Los datos se envían a través de un formulario que se maneja en el backend con el método POST para agregar clientes a la base de datos.
2. Tabla de Clientes: En la parte inferior, se muestra una tabla con los clientes registrados, mostrando su ID, imagen (si está disponible), nombre, apellido, fecha de nacimiento, teléfono, email y dirección. Esta tabla también incluye botones de "Eliminar" y "Editar". Los botones permiten eliminar clientes o abrir un modal para editar la información de un cliente en particular.
3. Modal de Edición: Cada cliente tiene un modal asociado que se activa al hacer clic en "Editar". En este modal, el administrador puede modificar la información del cliente y guardar los cambios en la base de datos. El formulario de edición se prellena con los datos actuales del cliente.
4. Estilo Visual: La página utiliza una estructura responsive, con tarjetas, formularios y tablas bien organizados. La interfaz está pensada para ser intuitiva, con una estética profesional adecuada para la gestión de los clientes.
5. Enlaces de Navegación: El botón "Volver al Índice" permite regresar a la página principal de administración del gimnasio.

Resultado de la página:

Gestión de Clientes - Gimnasio

Nota: No puedes eliminar clientes si este tiene pagos generados

| | | |
|----------------------|----------------------|-----------------------------------|
| Nombre | Apellido | Fecha de Nacimiento dd/mm/aaaa |
| <input type="text"/> | <input type="text"/> | <input type="text"/> |
| Teléfono | Email | Dirección |
| <input type="text"/> | <input type="text"/> | <input type="text"/> |

Agregar Cliente

[Volver al Índice](#)

| ID | Imagen | Nombre | Apellido | Fecha Nacimiento | Teléfono | Email | Dirección | Acciones |
|----|---|---------------|---------------|------------------|------------|---------------------|-------------------------|-----------------|
| 17 |  | Fernando | Azuara Ibarra | 2005-03-06 | 7122411716 | ferazuiba@gmail.com | San Felipe del Progreso | Eliminar |
| 18 |  | Jiovani Asael | Tapia López | 2005-02-03 | 919238912 | Asael@gmail.com | Santiago Casandeje | Eliminar |

Membresías.html

Link del código: <https://github.com/igream/p2-flask-SQL/blob/main/Interfaces%20Equipo/src/templates/membresias.html>

La página de gestión de membresías permite a los administradores añadir, editar y eliminar tipos de membresías en el sistema. El diseño está organizado de forma clara y efectiva, con una estructura visual fácil de usar.

1. Lista de Membresías: Se presenta una tabla donde se listan todas las membresías disponibles. Cada fila muestra el ID, tipo, costo, duración, descripción, imagen asociada (si existe), y acciones para editar o eliminar la membresía.
 - ✓ Acciones: Los botones de "Editar" abren un modal con los datos pre-llenados para realizar modificaciones, mientras que el botón "Eliminar" permite eliminar la membresía seleccionada.
2. Modal de Edición:
 - ✓ Al hacer clic en "Editar" se abre un modal que permite modificar los detalles de la membresía. Este incluye campos para editar el tipo, costo, duración y descripción.
 - ✓ Los datos actuales de la membresía se cargan automáticamente en el formulario del modal para facilitar la edición.
3. Formulario para Añadir Membresías:

A la derecha de la tabla, se encuentra un formulario donde el administrador puede agregar una nueva membresía. Este formulario incluye campos para ingresar tipo, costo, duración y descripción, así como un campo para cargar una imagen que representará la membresía.

4. Diseño Responsive:

Se utiliza el framework Bootstrap 4 para asegurar que la página sea visualmente atractiva y se adapte a diferentes tamaños de pantalla. Los formularios y tablas son fáciles de navegar en dispositivos de escritorio y móviles.

5. Navegación:

Un botón de "Volver al Índice" al final de la página permite regresar a la página principal de administración.

El resultado visual de la página funcional es este:

Gestión de Membresías Y Clases Relacionadas

Nota: No se pueden eliminar membresías si están asociadas a clases o si hay clientes con pagos registrados. Para eliminarlas, primero debe asegurarse de que los clientes hayan completado su período de membresía y eliminar las clases asociadas.

[Volver al Índice](#)

Lista de Membresías

| Tipo | Costo | Duración | Descripción | Imagen | Acciones |
|--------|------------|----------|---|---|---|
| Carbón | \$400.00 | 31 días | Acceso a las |  | Editar Eliminar |
| Hierro | \$800.00 | 30 días | Carbon + Entrenamientos con proteína incluida |  | Editar Eliminar |
| Oros | \$12020.00 | 30 días | Hierro + Instructor personal |  | Editar Eliminar |

Clases asociadas:

- Entrenamiento de Atrofia Muscular [Eliminar](#)

Seleccionar Clase:

Ejercicio Aerobico

[Agregar Clase](#)

Clases asociadas:

Seleccionar Clase:

Ejercicio Aerobico

[Agregar Clase](#)

Agregar Nueva Membresía

Tipo de Membresía:

Costo:

Duración:

Descripción:

Imagen:

Seleccionar archivo Ningún archivo seleccionado

[Agregar Membresía](#)

Pagos.html

La página de gestión de pagos permite a los administradores gestionar los pagos realizados por los clientes. A través de esta interfaz, se puede agregar, actualizar y eliminar registros de pagos.

1. Lista de Pagos:

Se presenta una tabla con los pagos registrados, donde se muestra el ID de pago, ID de cliente, ID de membresía, fecha de pago, monto y método de pago.

Además, se incluyen botones para realizar acciones como actualizar o eliminar el pago.

2. Modal de Actualización de Pago:

Al hacer clic en el botón "Actualizar", se abre un modal donde el administrador puede editar los detalles del pago seleccionado, como el ID del cliente, el ID de la membresía, el monto y el método de pago.

El campo de método de pago tiene un select con las opciones de "Efectivo", "Tarjeta de Débito" y "Tarjeta de Crédito", y se preselecciona la opción correspondiente a la información del pago.

3. Formulario para Agregar Pago:

En el lado derecho de la página, se encuentra un formulario para agregar un nuevo pago. Este formulario incluye los siguientes campos:

- ✓ ID Cliente: Para vincular el pago al cliente correspondiente.
- ✓ ID Membresía: Para asociar el pago con la membresía que se pagó.
- ✓ Monto: El monto total pagado.
- ✓ Método de Pago: Un select con las opciones disponibles (Efectivo, Tarjeta de Débito y Tarjeta de Crédito).

4. Navegación:

Un botón al final del formulario permite regresar al Índice de administración.

5. Diseño Responsivo:

Se emplea Bootstrap 5 para garantizar que la página se ajuste bien a diferentes dispositivos y pantallas. Las tablas y formularios son fáciles de usar y se ven bien en móviles y escritorios.

Resultado visual:

| Gestión de Pagos | | | | | | | | | |
|---|------------|--------------|---------------|---------|-------------------|-----------|------------|---------------------|--|
| Volver al Índice | | | | | | | | | |
| El objetivo de esta sección es que el administrador valide el pago de la membresía del cliente, una vez el pago haya sido verificado, también podrá administrar el estado de la membresía | | | | | | | | | |
| Lista de Pagos | | | | | | | | | |
| ID Pago | ID Cliente | ID Membresia | Fecha de Pago | Monto | Método de Pago | Estado | Referencia | Estado de Membresía | Acciones |
| 14 | 18 | 1 | 2024-11-25 | 400.00 | Efectivo | Pagado | e70316f5c1 | Validado | Actualizar Eliminar |
| 24 | 17 | 8 | 2024-11-25 | 4000.00 | Tarjeta de Débito | Pendiente | 7232f49335 | Activo | Actualizar Eliminar |
| 25 | 19 | 1 | 2024-11-25 | 400.00 | Efectivo | Pendiente | 80d3a39e26 | Activo | Actualizar Eliminar |
| 26 | 19 | 2 | 2024-11-25 | 800.00 | Efectivo | Pendiente | f9ce2cb2bd | Inactivo | Actualizar Eliminar |
| 27 | 17 | 10 | 2024-11-25 | 8000.00 | Efectivo | Pendiente | 503fcd2eec | Inactivo | Actualizar Eliminar |
| 28 | 17 | 1 | 2024-11-25 | 400.00 | Efectivo | Pendiente | 2fc47fc8f3 | Inactivo | Actualizar Eliminar |

Personal.html

Link del código: <https://github.com/igream/p2-flask-SQL/blob/main/Interfaces%20Equipo/src/templates/personal.html>

La página de gestión de personal permite a los administradores administrar la información de los empleados del gimnasio, incluyendo la capacidad de agregar, editar y eliminar registros de personal. Esta página también está diseñada de manera que sea fácil de usar tanto en dispositivos móviles como de escritorio.

Lista de Personal:

La página muestra una tabla con la lista de empleados, donde se visualizan los siguientes datos de cada persona:

- ID: Identificador único del empleado.
- Nombre: Nombre completo del empleado.

- Puesto: El cargo o puesto que ocupa (Entrenador, Cajero, Conserje, Mantenimiento).
- Salario Mensual: El salario que recibe el empleado cada mes.
- Antigüedad: El número de meses que el empleado lleva trabajando en el gimnasio.
- Turno: El turno asignado al empleado (por ejemplo, mañana, tarde, noche).

Botón de Edición:

Cada fila de la tabla tiene un botón "Editar" que abre un modal para actualizar la información del empleado.

En el modal de edición, el administrador puede cambiar el nombre, el puesto, el salario, la antigüedad y el turno del empleado. El formulario pre-rellena los campos con la información actual del personal.

Formulario de Agregar Personal:

En el lado derecho de la página, se encuentra un formulario donde el administrador puede agregar un nuevo empleado. Los campos del formulario incluyen:

- Nombre: Campo para ingresar el nombre completo del nuevo empleado.
- Puesto: Un select para elegir el puesto que ocupará el nuevo empleado (Entrenador, Cajero, Conserje, Mantenimiento).
- Salario Mensual: Campo numérico para especificar el salario mensual del empleado.
- Antigüedad: Campo numérico para indicar cuántos meses lleva trabajando el empleado.
- Turno: Campo de texto para asignar el turno (mañana, tarde, noche).

Eliminación de Personal:

Al lado del botón de editar, hay un enlace "Eliminar" que permite borrar un registro de personal. Este enlace ejecuta una operación de eliminación en la base de datos.

Navegación:

Un botón al final de la página permite regresar al Índice de administración.

Diseño Responsivo:

Se utiliza Bootstrap 5 para garantizar que la página sea visualmente agradable y funcional en diferentes dispositivos y tamaños de pantalla, permitiendo una buena experiencia de usuario tanto en móviles como en computadoras de escritorio.

El resultado visual es:

The screenshot shows a web application titled "Gestión de Personal". At the top, there is a message: "El personal registrado como "Entrenador" será automáticamente agregado a las opciones de Instructores al agregar o editar una clase". Below this is a "Volver al Índice" button and the title "Lista de Personal". The main area displays a table of employee data with columns: ID, Nombre, Puesto, Salario Mensual, Antigüedad, Turno, and Acciones (with Editar and Eliminar buttons). The data is as follows:

| ID | Nombre | Puesto | Salario Mensual | Antigüedad | Turno | Acciones |
|----|----------------------|---------------|-----------------|------------|------------|---|
| 2 | Calamardo Tentaculos | Cajero | \$5000.00 | 12 meses | Matutino | <button>Editar</button> <button>Eliminar</button> |
| 3 | Levi Hackerman | Conserje | \$14123.00 | 15 meses | Vespertino | <button>Editar</button> <button>Eliminar</button> |
| 4 | Baki Hanma | Entrenador | \$10000.00 | 3 meses | Matutino | <button>Editar</button> <button>Eliminar</button> |
| 5 | Alma Marcela Gozo | Entrenador | \$15000.00 | 2 meses | Matutino | <button>Editar</button> <button>Eliminar</button> |
| 6 | Samuel del Luque | Entrenador | \$13000.00 | 24 meses | Vespertino | <button>Editar</button> <button>Eliminar</button> |
| 7 | Alan Brito | Entrenador | \$10000.00 | 12 meses | Vespertino | <button>Editar</button> <button>Eliminar</button> |
| 8 | Octavio Paz | Mantenimiento | \$5000.00 | 12 meses | Matutino | <button>Editar</button> <button>Eliminar</button> |

To the right, there is a form titled "Agregar Nuevo Personal" with fields for Nombre, Puesto (dropdown with "Entrenador" selected), Salario Mensual, Antigüedad en meses, Turno, and a large orange "Agregar Personal" button.

4.5.3 Creación de página de inicio de sesión

Indexlogin.html

Link del código: <https://github.com/igream/p2-flask-SQL/blob/main/Interfaces%20Equipo/src/templates/indexlogin.html>

Este código HTML corresponde a una página web para un Gimnasio, que incluye las siguientes funcionalidades:

Pantalla de inicio:

Un encabezado que da la bienvenida al gimnasio.

Botones para "Registrarse" e "Iniciar sesión", que abren modales con formularios correspondientes.

Un carrusel de imágenes (GIFs) con tres elementos visuales de temática de gimnasio.

Modales:

Modal de Registro: Formulario para que los usuarios nuevos puedan registrarse proporcionando información como nombre, apellido, fecha de nacimiento, teléfono, correo electrónico, contraseña y dirección, además de subir una imagen de perfil.

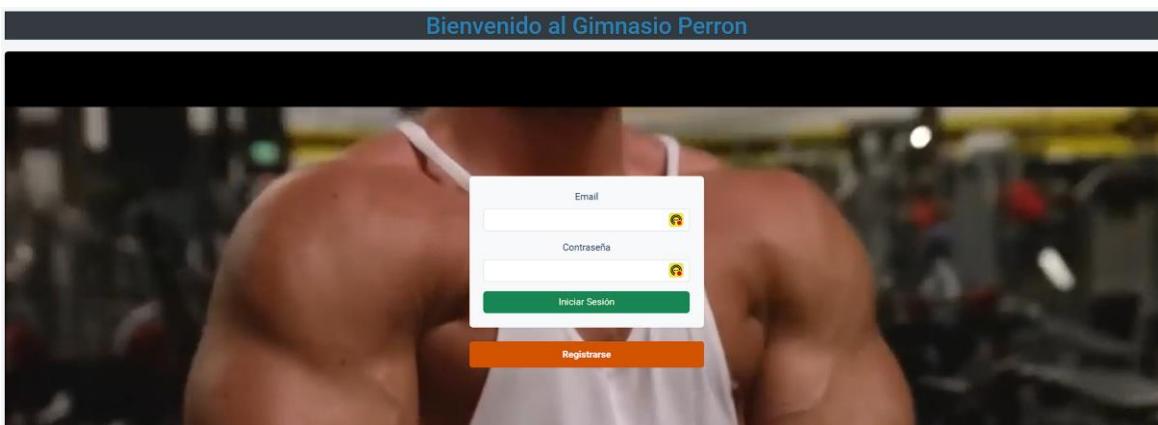
Modal de Login: Formulario para que los usuarios inicien sesión con su correo y contraseña.

Modal de Login para Administradores: Formulario para que los administradores inicien sesión utilizando un nombre de usuario y contraseña.

Mensajes Flash: Se usan para mostrar alertas con mensajes informativos o de error (ej., cuando el registro o inicio de sesión es exitoso o falla).

Pie de página (footer): Incluye derechos de autor y enlaces a redes sociales como Facebook, Twitter, Instagram y LinkedIn, así como enlaces a las secciones de "Sobre nosotros" y "Contacto". También ofrece un botón para que los administradores inicien sesión.

El resultado visual de la página es este:



Registro de Cliente

X

Nombre

Apellido

Imagen de perfil

Seleccionar archivo

Ningún archivo seleccionado

Fecha de Nacimiento

dd/mm/aaaa



Teléfono

Email

Contraseña

Dirección

Registrarse

Email

Contraseña

Iniciar Sesión

PARA EL INICIO DE SESION COMO ADMINISTRADOR SE PUEDEN UTILIZAR LAS SIGUIENTES CREDENCIALES:

- “admin”, “passadmin”
- “admin1”, “contrasena1”
- “ProfeTeja”, “Teja123”

Inicio de Sesión - Administrador

X

Usuario

Contraseña

Iniciar Sesión

4.5.4 Creación de página de cliente

Indexclientes.html

Link del código: [https://github.com/igream/p2-flask-
SQL/blob/main/Interfaces%20Equipo/src/templates/indexclientes.html](https://github.com/igream/p2-flask-SQL/blob/main/Interfaces%20Equipo/src/templates/indexclientes.html)

Este código HTML corresponde a la página principal de un cliente en un sitio web de un gimnasio. La página está diseñada para ser visualmente atractiva y funcional utilizando Bootstrap 5. A continuación, un resumen de las secciones clave:

1. Encabezado (Header): Muestra una imagen de perfil y un mensaje de bienvenida con el nombre del usuario, junto con un botón para cerrar sesión.
2. Sección Principal (Main):
 - ✓ Banner: Presenta una imagen destacada del gimnasio.
 - ✓ Acciones rápidas: Tres botones para interactuar con funcionalidades clave como ver clases, comprar membresía y consultar historial de pagos, que abren modales con la información correspondiente.
3. Modales:
 - ✓ Comprar Membresía: Muestra una lista de tipos de membresías con detalles y una opción para pagar.
 - ✓ Confirmación de Pago: Permite al cliente confirmar su pago, eligiendo el método de pago.
 - ✓ Historial de Pagos: Muestra una tabla con los pagos anteriores del cliente.
 - ✓ Ver Clases: Presenta una lista de clases disponibles con detalles y capacidad.
4. Blog del Gimnasio: Artículos informativos relacionados con entrenamiento, nutrición y motivación.
5. Membresía Activa: Muestra detalles de la membresía activa del cliente (si tiene alguna).
6. Pie de página (Footer): Información de contacto y derechos reservados del gimnasio.

El código utiliza Flask para integrar las variables dinámicas en la página, como el nombre de usuario y los datos de las clases, membresías y pagos, los cuales se cargan mediante Ninja.

Los resultados visuales son los siguientes:



[Ver Clases](#)[Comprar Membresía](#)[Historial de Pagos](#)

Blog del Gimnasio

Cómo estructurar un entrenamiento de fuerza

Si deseas ganar fuerza y músculo, es esencial que sepas cómo estructurar un entrenamiento adecuado. A continuación, te presentamos una estructura básica para tu rutina de fuerza:

- **Ejercicios compuestos:** Los ejercicios como sentadillas, press de banca y peso muerto deben ser la base de tu rutina.
- **Repeticiones y series:** Realiza entre 3-5 series de 6-12 repeticiones con un peso que te desafie.
- **Descanso:** Asegúrate de descansar entre 1 y 2 minutos entre cada serie para permitir la recuperación muscular.
- **Progresión:** Aumenta gradualmente el peso o la dificultad de los ejercicios cada semana para seguir desafiando a tu cuerpo.

Recuerda que la clave para mejorar es la consistencia. Si sigues este enfoque, notarás cómo tu fuerza y masa muscular aumentan con el tiempo.

Comidas pre-entrenamiento para mejorar tu rendimiento

Tu rendimiento en el gimnasio está estrechamente relacionado con lo que comes antes de entrenar. Aquí te damos algunas sugerencias de alimentos que te ayudarán a maximizar tu energía:

- **Carbohidratos complejos:** Alimentos como avena, batatas o arroz integral proporcionan energía sostenida durante el entrenamiento.
- **Proteínas:** El pollo, pavo, o proteína en polvo son ideales para ayudar a la reparación muscular después del entrenamiento.
- **Gorduras saludables:** Los aguacates y frutos secos proporcionan energía duradera sin causar picos de insulina.

Consuma estas comidas 30-60 minutos antes de entrenar para asegurarte de tener suficiente energía para un buen rendimiento en el gimnasio.

Cómo mantenerse motivado en tu rutina

La motivación puede fluctuar, pero existen estrategias que te pueden ayudar a mantener el enfoque. Aquí algunos consejos:

- **Establece metas pequeñas:** Los objetivos alcanzables te mantienen motivado y te dan una sensación de logro constante.
- **Encuentra un compañero de entrenamiento:** Tener alguien con quien entrenar te ayudará a mantener el compromiso.
- **Varia tus rutinas:** Cambiar tu rutina de ejercicios regularmente mantendrá las cosas interesantes y desafiantes.
- **Celebra tus logros:** No importa lo pequeños que sean, celebra tus avances para mantenerte motivado.

Recuerda que cada día es una oportunidad para mejorar. Mantén tu mentalidad positiva y verás los resultados.

Mi Membresía

No tienes ninguna membresía activa.

© 2024 Gimnasio Perron

Derechos reservados para WitsTeam

Fernando Azuara Ibarra, Omar Andrei García López, Jenifer Ramírez Ibarra, Oswaldo Plata Navarrete, Jiovani Asael Tapia López

Contacto: GIMNASIOPERRON@uaemex.mx

Clases Disponibles

X

| Nombre | Descripción | Instructor | Hora | Capacidad | Imagen |
|-----------------------------------|---|-------------------------|----------|-----------|---|
| Ejercicio Aerobico | El ejercicio aeróbico fortalece el corazón y promueve niveles de colesterol saludables. Entre los aeróbicos de bajo impacto están el caminar y nadar. Correr, jugar tenis y bailar son aeróbicos de alto impacto. | Alma Marcela Gozo | 22:00:00 | 20 |  |
| Entrenamiento de Fuerza | Es una rutina deportiva que busca fortalecer los músculos del cuerpo. Para ello, se utilizan distintos métodos de resistencia, como pesas, máquinas, bandas elásticas o el propio peso del cuerpo. | Alan Brito | 23:00:00 | 10 |  |
| Entrenamiento de Atrofia Muscular | El entrenamiento físico puede ayudar a tratar la atrofia muscular, que es la disminución de la masa muscular y el desgaste de los tejidos musculares. | Samuel del Luque | 1:00:00 | 5 |  |
| Entrenamiento de Espalda | La mejor manera de entrenar los músculos de la espalda es a través de levantamientos compuestos (multiarticulares) que implican levantar pesos pesados. Estos pueden ser el levantamiento de peso muerto, las cargadas, las arrancadas y los tirones. | Baki Hanma | 17:40:00 | 10 |  |

Comprar Membresía

X

| Tipo | Descripción | Duración | Costo | Imagen | Acción |
|----------------|---|----------|------------|---|-------------------------|
| Carbón | Acceso a lasas | 31 días | \$400.00 |  | Cotizar |
| Hierro | Carbon + Entrenamientos con proteina incluida | 30 días | \$800.00 |  | Cotizar |
| Oros | Hierro + Instructor personal | 30 días | \$12020.00 |  | Cotizar |
| Diamante | Oro + Nutriologo Incluido | 30 días | \$1600.00 |  | Cotizar |
| Super Carbón | Semestre de beneficios carbón | 180 días | \$2000.00 |  | Cotizar |
| Super Hierro | Semestre de beneficios Hierro | 180 días | \$4000.00 |  | Cotizar |
| Super Oro | Semestre de beneficios oro | 180 días | \$6000.00 |  | Cotizar |
| Super Diamante | Semestre de beneficios diamante | 180 días | \$8000.00 |  | Cotizar |

Generar Pago

X

¿Estás seguro de que deseas pagar la membresía **Super Diamante** por **\$8000.00**?

Método de Pago

Efectivo

Generar Referencia de pago

Referencia de pago.

Recibo de Pago

ID Cliente: 18

ID Membresía: 10

Método de Pago: Efectivo

Monto: \$8000.00

Referencia: 1eb32d37bf

Estado: Pendiente

Transferencia Bancaria: 8901 3801 0238 0012

Deposito en efectivo: 09128390103020012023

Gracias por su pago. Este es un recibo provisional.

En esta sección el estatus de la membresía se queda en estado pendiente, hasta que un administrador “valide” el pago para después proceder a activarla y el usuario pueda disfrutar de todos los beneficios.

| | | | | | | | | | |
|----|----|----|------------|---------|----------|-----------|------------|----------|----------------------------|
| 29 | 18 | 10 | 2024-11-26 | 8000.00 | Efectivo | Pendiente | 1eb32d37bf | Inactivo | Actualizar |
| | | | | | | | | | Eliminar |

Historial de Pagos



| Referencia | Membresía | Fecha de Pago | Monto | Metodo de Pago | Estado de Pago | Estado de Membresía |
|------------|----------------|---------------|-----------|----------------|----------------|---------------------|
| e70316f5c1 | Carbón | 2024-11-25 | \$400.00 | Efectivo | Pagado | Validado |
| 1eb32d37bf | Super Diamante | 2024-11-26 | \$8000.00 | Efectivo | Pendiente | Activo |

[Cerrar](#)

Mi Membresía

Super Diamante
Vigencia: -
Costo: \$8000.00



4.6 Estructura Back-End

Para este caso nos referimos a la organización y diseño de la parte del servidor de una aplicación web. En el contexto de un proyecto, el back-end maneja la lógica de negocio, la

interacción con bases de datos, la autenticación de usuarios y las respuestas a las solicitudes del cliente. La estructura de back-end debe estar bien organizada para asegurar que la aplicación funcione de manera eficiente y escalable.

En este sentido, se utilizan tecnologías como Flask para manejar las rutas y solicitudes HTTP, MySQL para la gestión de la base de datos, y el patrón de diseño MVC para separar las responsabilidades del modelo de datos, las vistas y el controlador. La arquitectura también debe ser segura, manejando correctamente la validación de datos y el control de acceso de los usuarios (administradores y clientes).

4.6.1 Funciones para página de administrador

Creación del archivo de las rutas “app.py”

Link del código: <https://github.com/igream/p2-flask-SQL/blob/main/Interfaces%20Equipo/app.py>

```
@app.route('/index_admin')
```

Esta ruta maneja la vista principal del administrador. Primero verifica si hay una sesión activa para el administrador mediante la clave 'admin_id'. Si es así, renderiza la plantilla indexadmin.html. Si no, muestra un mensaje de advertencia y redirige al inicio de sesión. Esto garantiza que solo los administradores autenticados puedan acceder a la página de administración.

```
@app.route('/logout')
```

Esta ruta permite al usuario cerrar sesión. Cuando un administrador o cliente accede a ella, se limpia la información de la sesión (session.clear()), lo que termina su sesión activa. Luego, se muestra un mensaje de confirmación y se redirige al usuario al formulario de inicio de sesión (indexlogin).

```
@app.route('/admin/login', methods=['POST'])
```

Esta ruta maneja el inicio de sesión del administrador. Recibe el nombre de usuario y la contraseña desde un formulario, y luego verifica si las credenciales son correctas consultando la base de datos. Si el administrador existe y la contraseña es válida, se guarda su ID en la sesión y se redirige al panel de administración (index_admin). Si las credenciales son incorrectas, muestra un mensaje de error y redirige al formulario de inicio de sesión.

```
@app.route('/some_protected_route')
```

Esta es una ruta protegida que solo pueden acceder los usuarios autenticados. Si el usuario no está autenticado (es decir, no hay una clave 'user_id' en la sesión), se le redirige al formulario de inicio de sesión con un mensaje pidiendo que inicie sesión primero. Si el usuario está autenticado, se ejecuta la lógica asociada a la ruta protegida.

4.6.2 Funciones para subpáginas de administrador

Funciones para las subpáginas, son 3 por cada página como mínimo.

Visualizar Clientes (/clientes):

Obtiene todos los registros de clientes desde la base de datos y los muestra en la plantilla clientes.html.

Agregar Cliente (/clientes/add):

Recibe datos desde un formulario, los inserta en la tabla Clientes y redirige a la página de clientes. Si ocurre un error, muestra un mensaje de error.

Actualizar Pago (/pagos/update/<int:id>):

Permite actualizar los detalles de un pago en la base de datos. Recibe los datos desde un formulario y realiza la actualización en la tabla Pagos.

Actualizar Personal (/personal/update/<int:id>):

Actualiza la información de un miembro del personal. Recibe datos de un formulario y realiza la actualización en la tabla Personal.

Actualizar Membresía (/membresias/update/<int:id>):

Permite editar los detalles de una membresía. Se actualiza la tabla Membresias con los datos proporcionados.

Editar Clase (/clases/edit/<int:id>):

Permite editar los detalles de una clase, incluyendo su nombre, descripción, instructor, hora y capacidad. Si se sube una imagen, la guarda en el directorio correspondiente.

Editar Cliente (/clientes/edit/<int:id>):

Actualiza los detalles de un cliente específico, incluidos datos como el nombre, fecha de nacimiento, teléfono, email y dirección.

Pagar Membresía (/pagar_membresia):

Permite realizar el pago de una membresía. Se registra en la base de datos el pago realizado por el cliente, incluyendo la cantidad y el método de pago.

Eliminar Cliente (/clientes/delete/<int:id>):

Elimina un cliente de la base de datos después de verificar que existe.

Obtener Clases (/get_clases):

Obtiene todas las clases disponibles y las muestra en la página indexclientes.html.

Eliminar Clase (/clases/delete/<int:id>):

Elimina una clase de la base de datos.

Visualizar Membresías (/membresias):

Muestra todas las membresías disponibles en la base de datos.

Agregar Membresía (/membresias/add):

Permite agregar una nueva membresía con su tipo, costo, duración, descripción e imagen.

Eliminar Membresía (/membresias/delete/<int:id>):

Elimina una membresía de la base de datos.

Ver Pagos (/pagos):

Muestra todos los pagos registrados en la base de datos.

Agregar Pago (/pagos/add):

Permite agregar un nuevo pago a la base de datos para un cliente y una membresía específica.

Eliminar Pago (/pagos/delete/<int:id>):

Elimina un pago específico de la base de datos.

Visualizar Personal (/personal):

Muestra todos los registros del personal en la base de datos.

Agregar Personal (/personal/add):

Permite agregar un nuevo miembro del personal a la base de datos con su nombre, puesto, salario, antigüedad y turno.

Eliminar Personal (/personal/delete/<int:id>):

Elimina un miembro del personal de la base de datos.

Cada sección está asociada a una ruta que interactúa con la base de datos y realiza operaciones CRUD (Crear, Leer, Actualizar y Eliminar) sobre las tablas Clientes, Pagos, Personal, Clases y Membresías.

4.6.3 Funciones para página de inicio de sesión

Registro y inicio de sesión en funciones Flask

Función `register() :`

Esta función se encarga de registrar un nuevo cliente en la base de datos.

- Recibe datos del formulario de registro, como nombre, apellido, fecha de nacimiento, teléfono, email, contraseña, dirección, y una imagen opcional.
- La contraseña se encripta mediante `generate_password_hash` para garantizar la seguridad.
- Si el cliente sube una imagen, la función valida y guarda el archivo en el directorio `src/static/clientes`, luego almacena la ruta de la imagen en la base de datos.

- Despues de insertar los datos en la base de datos, la función confirma la transacción y redirige al usuario a la página de inicio de sesión con un mensaje de éxito o error.

Función

`login():`

Esta función permite a un cliente iniciar sesión en la aplicación.

- Recibe el email y la contraseña del formulario de inicio de sesión.
- Busca al cliente en la base de datos utilizando el email proporcionado.
- Si se encuentra el usuario y la contraseña coincide con la almacenada (utilizando `check_password_hash`), se guarda el ID_Cliente, el nombre y la imagen del cliente en la sesión de Flask.
- Si la autenticación es exitosa, el cliente es redirigido a su página principal. Si las credenciales son incorrectas, se muestra un mensaje de error.

Ambas funciones utilizan flash para mostrar mensajes de éxito o error al usuario y se conectan a la base de datos a través de la función `get_db_connection`.

4.6.4 Funciones para página de cliente

Renderizar plantilla y funciones adicionales con la base de datos.

Ruta /index_cliente:

- Propósito: Esta es la página principal de un cliente después de iniciar sesión.
- Funcionamiento:
 - Primero, se verifica si el usuario está autenticado al comprobar si el `user_id` está presente en la sesión. Si no está autenticado, se redirige al inicio de sesión.
 - Se recuperan los datos del usuario desde la sesión (como el nombre).
 - Se consultan las siguientes tablas de la base de datos:
 - Clases: Obtiene todas las clases disponibles en el gimnasio.
 - Membresías: Obtiene todas las opciones de membresía.
 - Pagos: Recupera todos los pagos realizados por el cliente, junto con detalles de la membresía asociada.
 - Pago más caro: Si el cliente tiene pagos, se determina el pago más caro, se calcula la fecha de inicio y fin del pago (basada en la duración de la membresía) y se agrega esta información al pago más caro.

- Finalmente, se renderiza la plantilla indexclientes.html, pasando los datos del cliente, las clases, membresías, pagos y el pago más caro para ser visualizados en la interfaz.

Ruta /historial_pagos:

- Propósito: Muestra el historial de pagos de un cliente.
- Funcionamiento:
- Al igual que en la ruta anterior, primero se verifica si el usuario está autenticado.
- Si está autenticado, se recuperan los pagos realizados por el cliente desde la base de datos. Esta información incluye el tipo de membresía, la fecha del pago, el monto y el método de pago.
- La información de los pagos se pasa a la plantilla indexclientes.html para ser mostrada al cliente.

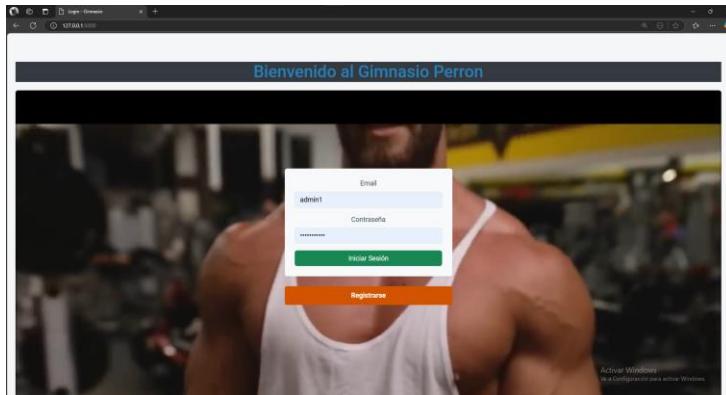
Ruta /get_clases:

- Propósito: Recupera todas las clases disponibles en el gimnasio.
- Funcionamiento:
- Realiza una consulta a la base de datos para obtener todas las clases disponibles.
- La lista de clases se pasa a la plantilla indexclientes.html para ser visualizada en la página.

4.7 Mejoras de examen ordinario

1. Implementación de nueva interfaz de inicio

Primera versión de ingreso a la página:



Nueva versión:

Sección inicial ([index.html](#))



Sección de inicio de sesión ([indexlogin.html](#))



Se crearon nuevos archivos html y rutas para vincular estas nuevas

```

# Página principal
@app.route('/')

def index():
    print("Accediendo a la ruta principal")
    return render_template('index.html')
@app.route('/indexlogin')
def indexlogin():
    print("Accediendo a la ruta principal")
    return render_template('indexlogin.html')
@app.route('/services')
def services():
    return render_template('info.html')

@app.route('/schedules')
def schedules():
    return render_template('membresia.html')

@app.route('/trainers')
def trainers():
    return render_template('sobre_nosotros.html')

@app.route('/pricing')
def pricing():
    return render_template('class.html')

@app.route('/location')
def location():
    return render_template('perron.html')

```

| Antes | Ahora | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|--|---|--|------------------|---------|--|--|---|-----------------|--|------------------|--------|---|-------------|--|------------------|--------|---|------------|--|------------------|--------|---|---------------|--|------------------|--------|---|------------|--|------------------|--------|---|-----------------|--|------------------|--------|---|--------------------|--|------------------|---------|---|-----------------|--|------------------|---------|---|-----------|--|------------------|--------|---|----------------|--|------------------|--------|---|-----------------|--|------------------|---------|---|------------|--|------------------|---------|---|-------------|--|------------------|--------|---|---------------|--|------------------|--------|---|---------------------|--|------------------|--------|
| <ul style="list-style-type: none"> ✓ templates <ul style="list-style-type: none"> ↳ adminlogin.html ↳ clases.html ↳ clientes.html ↳ indexadmin.html ↳ indexclientes.html ↳ indexlogin.html ↳ membresias.html ↳ pagos.html ↳ personal.html | <table border="1"> <thead> <tr> <th></th> <th></th> <th></th> <th></th> <th></th> </tr> </thead> <tbody> <tr> <td>↳</td><td>adminlogin.html</td><td></td><td>2024-12-03 22:34</td><td>2.6 KB</td></tr> <tr> <td>↳</td><td>clases.html</td><td></td><td>2024-11-26 06:13</td><td>9.9 KB</td></tr> <tr> <td>↳</td><td>class.html</td><td></td><td>2024-12-04 19:17</td><td>3.6 KB</td></tr> <tr> <td>↳</td><td>clientes.html</td><td></td><td>2024-11-26 06:13</td><td>5.8 KB</td></tr> <tr> <td>↳</td><td>index.html</td><td></td><td>2024-12-04 19:17</td><td>6.6 KB</td></tr> <tr> <td>↳</td><td>indexadmin.html</td><td></td><td>2024-12-04 19:18</td><td>4.3 KB</td></tr> <tr> <td>↳</td><td>indexclientes.html</td><td></td><td>2024-12-04 22:16</td><td>22.3 KB</td></tr> <tr> <td>↳</td><td>indexlogin.html</td><td></td><td>2024-12-04 19:18</td><td>10.0 KB</td></tr> <tr> <td>↳</td><td>info.html</td><td></td><td>2024-12-04 19:18</td><td>5.1 KB</td></tr> <tr> <td>↳</td><td>membresia.html</td><td></td><td>2024-12-04 19:18</td><td>5.8 KB</td></tr> <tr> <td>↳</td><td>membresias.html</td><td></td><td>2024-11-27 02:06</td><td>10.5 KB</td></tr> <tr> <td>↳</td><td>pagos.html</td><td></td><td>2024-12-03 22:43</td><td>11.4 KB</td></tr> <tr> <td>↳</td><td>perron.html</td><td></td><td>2024-12-04 19:19</td><td>4.4 KB</td></tr> <tr> <td>↳</td><td>personal.html</td><td></td><td>2024-11-26 06:14</td><td>9.7 KB</td></tr> <tr> <td>↳</td><td>sobre_nosotros.html</td><td></td><td>2024-12-04 19:19</td><td>4.1 KB</td></tr> </tbody> </table> | | | | | | ↳ | adminlogin.html | | 2024-12-03 22:34 | 2.6 KB | ↳ | clases.html | | 2024-11-26 06:13 | 9.9 KB | ↳ | class.html | | 2024-12-04 19:17 | 3.6 KB | ↳ | clientes.html | | 2024-11-26 06:13 | 5.8 KB | ↳ | index.html | | 2024-12-04 19:17 | 6.6 KB | ↳ | indexadmin.html | | 2024-12-04 19:18 | 4.3 KB | ↳ | indexclientes.html | | 2024-12-04 22:16 | 22.3 KB | ↳ | indexlogin.html | | 2024-12-04 19:18 | 10.0 KB | ↳ | info.html | | 2024-12-04 19:18 | 5.1 KB | ↳ | membresia.html | | 2024-12-04 19:18 | 5.8 KB | ↳ | membresias.html | | 2024-11-27 02:06 | 10.5 KB | ↳ | pagos.html | | 2024-12-03 22:43 | 11.4 KB | ↳ | perron.html | | 2024-12-04 19:19 | 4.4 KB | ↳ | personal.html | | 2024-11-26 06:14 | 9.7 KB | ↳ | sobre_nosotros.html | | 2024-12-04 19:19 | 4.1 KB |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ↳ | adminlogin.html | | 2024-12-03 22:34 | 2.6 KB | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ↳ | clases.html | | 2024-11-26 06:13 | 9.9 KB | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ↳ | class.html | | 2024-12-04 19:17 | 3.6 KB | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ↳ | clientes.html | | 2024-11-26 06:13 | 5.8 KB | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ↳ | index.html | | 2024-12-04 19:17 | 6.6 KB | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ↳ | indexadmin.html | | 2024-12-04 19:18 | 4.3 KB | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ↳ | indexclientes.html | | 2024-12-04 22:16 | 22.3 KB | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ↳ | indexlogin.html | | 2024-12-04 19:18 | 10.0 KB | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ↳ | info.html | | 2024-12-04 19:18 | 5.1 KB | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ↳ | membresia.html | | 2024-12-04 19:18 | 5.8 KB | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ↳ | membresias.html | | 2024-11-27 02:06 | 10.5 KB | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ↳ | pagos.html | | 2024-12-03 22:43 | 11.4 KB | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ↳ | perron.html | | 2024-12-04 19:19 | 4.4 KB | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ↳ | personal.html | | 2024-11-26 06:14 | 9.7 KB | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ↳ | sobre_nosotros.html | | 2024-12-04 19:19 | 4.1 KB | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Nuevas secciones de la página principal (sin inicial sesión):

¡Bienvenido a tu Gimnasio!
Descubre todos los beneficios que ofrecemos para tu bienestar.

¿Qué obtienes?

| | |
|--|--|
| | Apoyo en tu entrenamiento físico Más de 50 clases por semana en grupos pequeños. |
| | Diferentes clases entre semana Trabajamos en el horario que más te conviene. |
| | Gran variedad de equipo para hacer ejercicio* Nunca tendrás que esperar por máquinas. |
| | Membresías sin compromiso* Sómos flexibles para ajustarnos a tu vida. |

Beneficios de asistir al gimnasio

| | |
|--|--|
| | Mejorar la fuerza muscular |
| | Mejorar el sistema cardiovascular |
| | Mejorar la salud mental |
| | Dormir mejor |
| | Mantener los huesos y músculos fuertes |
| | Mejorar la autoestima |
| | Formar parte de una comunidad |
| | Evitar lesiones y accidentes |

Nota: Los músculos necesitan descansar, se recomienda un día de descanso entre entrenamientos.

¡Descubre Nuestras Membresías!

| | | |
|---|---|--|
| CARBONOS | HIERRO | OROS |
| Acceso a las instalaciones tiempo completo Duración: 40 días Costo: \$400.00 | Carbono + Entrenamientos con proteína incluida Duración: 30 días Costo: \$800.00 | Hierro + Instructor personal Duración: 30 días Costo: \$1,200.00 |
| DIAMANTE | SUPER CARBÓN | SUPER HIERRO |
| Diamante + Náutico incluido Duración: 30 días Costo: \$1,600.00 | Semáforo de beneficios carbono Duración: 180 días Costo: \$2,000.00 | Semáforo de beneficios Hierro Duración: 180 días Costo: \$4,000.00 |
| SUPER DIAMANTE | | |
| Semáforo de beneficios diamante Duración: 180 días Costo: \$8,000.00 | | |

Sobre Nosotros
Tu bienestar físico y mental es nuestra prioridad

¿Quiénes Somos?
En Gimnasio Perón, somos el gimnasio líder en México con una presencia sólida en múltiples estados y más de 15 clubes activos. Descubre nuestras modernas instalaciones y servicios exclusivos diseñados para transformar tu bienestar físico y mental.

Nuestro Compromiso

Nos especializamos en brindar un entorno de entrenamiento de primer nivel con instalaciones modernas en más de 5 estados de la República Mexicana, incluyendo Ciudad de México, Estado de México, Querétaro, Aguascalientes, y Baja California. Con más de 15 clubes, ofrecemos un espacio donde cada persona pueda alcanzar sus metas de bienestar.

Nuestros Servicios



Clases Ofertadas
Descubre nuestras clases y encuentra la que más se adapte a ti

Aerobic

El aerobic es un ejercicio colectivo ideal para todas las edades y géneros. Mejora tu salud física y mental con esta modalidad divertida y saludable.

Aerobox

Combina la aeróbica con el deporte. Quema calorías y libera estrés con movimientos inspirados en el kickboxing al ritmo de la música.



¿Por qué inscribirse a Gimnasio Perrón?

Fácil, te apoyamos en cualquier momento y en cualquier lugar. Nos dedicamos a crear un plan que realmente se adapte a tu vida. Nuestros entrenadores y comunidad están para apoyarte en todas las formas posibles hacia un estilo de vida más saludable.



ACOMPAÑAMIENTO

Cada cliente obtiene un plan de increíble apoyo. Nuestro amable staff está preparado para apoyarte en tu programa fitness, sin importar cuáles apoyos necesites. Además, puedes optar por programas de entrenamiento personalizado.



ENTRENAMIENTO

Juntas, podemos hacer que la saludable sea divertida. Ofrecemos clases y entrenamiento personal en el club y las herramientas adecuadas para mantenerte encantado cuando estés fuera del gimnasio.



SIEMPRE ABIERTO

Con acceso las 24 horas, los 7 días de la semana a más de 5,000 ubicaciones en todo el mundo, te estás uniendo a la comunidad saludable más grande del mundo.



2. Desvincular la página de administrador de la página principal.

Quitamos el botón de indexlogin.html

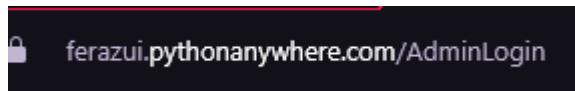


Esto se hizo creando un nuevo documento HTML, [adminlogin.html](#)

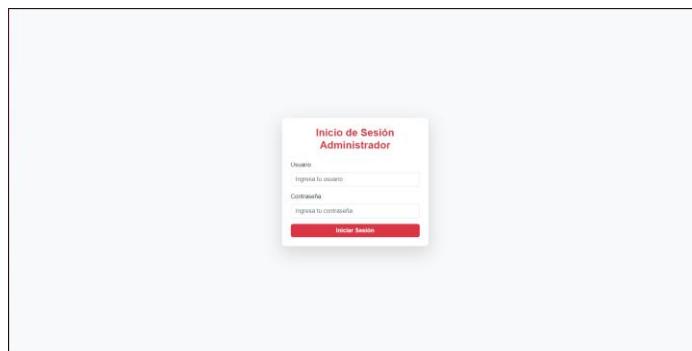
Se estableció una ruta personalizada para la interfaz de administrador:

```
47 #Pagina para inicio de sesión de administradores
48 @app.route('/AdminLogin')
49 def admin_login_template():
50     return render_template('adminlogin.html')
```

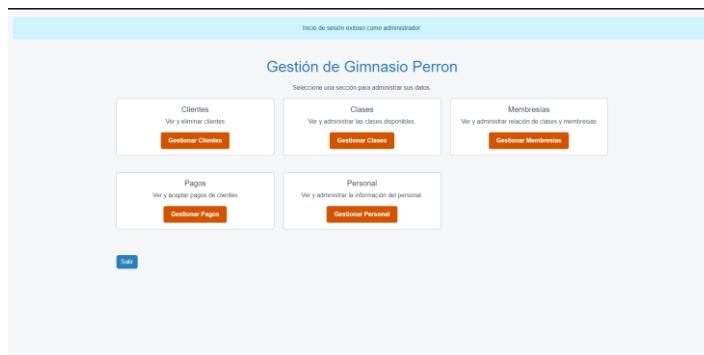
Dirección requerida:



Página cargada:



Esta nos dirige al panel principal del administrador



3. Mejorar la información mostrada en la tabla de pagos, agregar función de búsqueda y mostrar ingresos totales.

Nueva tabla de pagos:

| Gestión de Pagos | | | | | | | | | | |
|---|---------------------------|---------|-----------|------------|---------------|--------------------|----------------|------------|------------|--|
| Volver al Índice | | | | | | | | | | |
| El objetivo de esta sección es que el administrador valide el pago de la membresía del cliente, una vez el pago haya sido verificado, también poder administrar el estado de la membresía | | | | | | | | | | |
| Ingresos Totales: \$16400.00 | | | | | | | | | | |
| Lista de Pagos | | | | | | | | | | |
| ID | Pago | Cliente | Membresia | Precio | Fecha de Pago | Monto | Método de Pago | Estado | Referencia | Estado de Membresía |
| 14 | Jiovani Asael Tapia López | Carbón | \$400.00 | 2024-11-25 | 400.00 | Tarjeta de Credito | Pagado | e70316f5c1 | Validado | Actualizar Eliminar |
| 25 | Omar Aldrei García López | Carbón | \$400.00 | 2024-11-25 | 400.00 | Efectivo | Pendiente | 80d3a39e26 | Activo | Actualizar Eliminar |
| 28 | Fernando Azuara Ibarra | Carbón | \$400.00 | 2024-11-25 | 400.00 | Efectivo | Pendiente | 2fc47fc8f3 | Inactivo | Actualizar Eliminar |
| 26 | Omar Aldrei García López | Hierro | \$800.00 | 2024-11-25 | 800.00 | Efectivo | Pendiente | f9ce2cb2bd | Inactivo | Actualizar Eliminar |

Se realizaron cambios en la función que envía información a la plantilla, además, la consulta SQL cambio, para poder mostrar elementos de otra tabla se usó JOIN de la siguiente manera:

```
@app.route('/pagos', methods=['GET'])
def pagos():
    db = get_db_connection()
    cursor = db.cursor(dictionary=True)

    # Recuperar el término de búsqueda de la solicitud
    query_param = request.args.get('query', '').strip()

    # Base de la consulta SQL
    base_query = """
SELECT
    p.ID_Pago,
    CONCAT(c.Nombre, ' ', c.Apellido) AS Cliente,
    m.Tipo AS Membresia,
    m.Costo AS Precio,
    p.Fecha_Pago,
    p.Monto,
    p.Metodo_Pago,
    p.Estado,
    p.Referencia,
    p.Estado_Membresia,
    p.ID_Cliente,
    p.ID_Membresia
FROM Pagos p
JOIN Clientes c ON p.ID_Cliente = c.ID_Cliente
JOIN Membresias m ON p.ID_Membresia = m.ID_Membresia
    """
```

Así mismo, se agregó un algoritmo de búsqueda a la misma función:

```

# Filtrar si hay un término de búsqueda
if query_param:
    search_query = """
        WHERE c.Nombre LIKE %s
        OR c.Apellido LIKE %s
        OR m.Tipo LIKE %s
        OR p.Referencia LIKE %s
        """
    base_query += search_query
    search_value = f"%{query_param}%" 
    cursor.execute(base_query, (search_value, search_value, search_value, search_value))
else:
    cursor.execute(base_query)

pagos = cursor.fetchall()

```

También se usaron más sentencias SQL para sumar los pagos que estaban en Estado “Pagado”:

```

# Calcular ingresos totales
ingresos_query = """
SELECT SUM(m.Costo) AS IngresosTotales
FROM Pagos p
JOIN Membresias m ON p.ID_Membresia = m.ID_Membresia
WHERE p.Estado = 'Pagado'
"""

cursor.execute(ingresos_query)
ingresos_totales = cursor.fetchone()['IngresosTotales'] or 0 # Manejo de caso nulo

cursor.close()
db.close()

return render_template('pagos.html', pagos=pagos, ingresos_totales=ingresos_totales, query=query_param)

```

Se actualizó el archivo HTML [pagos.html](#), permitiendo usar la información definida en las funciones, esto mediante la iteración de Jinja.

Tabla:

| ID Pago | Cliente | Membresía | Precio | Fecha de Pago | Monto | Método de Pago | Estado | Referencia | Estado de Membresía | Acciones |
|--------------------|--------------------|----------------------|---------------------|-----------------------|------------------|------------------------|-------------------|-----------------------|-----------------------------|---------------------|
| {{ pago.ID_Pago }} | {{ pago.Cliente }} | {{ pago.Membresia }} | \${{ pago.Precio }} | {{ pago.Fecha_Pago }} | {{ pago.Monto }} | {{ pago.Metodo_Pago }} | {{ pago.Estado }} | {{ pago.Referencia }} | {{ pago.Estado_Membresia }} | Ver |

Ingresos:

```
<!-- Mostrar ingresos totales -->
<div class="alert alert-success text-center mt-4">
|   <h4>Ingresos Totales: ${{ ingresos_totales }}</h4>
</div>
<div>
```

Formulario de búsqueda:

```
<form action="{{ url_for('pagos') }}" method="GET" class="form-inline justify-content-center">
|   <input type="text" name="query" class="form-control mr-2" placeholder="Cliente/Referencia/>
|   <button type="submit" class="btn btn-primary">Buscar</button>
|   <a href="{{ url_for('pagos') }}" class="btn btn-secondary ml-2">Limpiar</a>
</form>
```

4. Permitir al cliente descargar su ticket de pago de manera ilimitada.

Se usó la función creada para poner la información en el ticket usando la extensión FPDF

```
def generar_recibo_pdf(id_cliente, id_membresia, monto, metodo_pago, referencia):
    # Crear el objeto PDF
    pdf = FPDF()
    pdf.add_page()
    pdf.set_font("Arial", size=12)

    # Título del recibo
    pdf.set_font("Arial", style="B", size=16)
    pdf.cell(200, 10, txt="Recibo de Pago", ln=True, align='C')
    pdf.ln(10) # Espacio entre líneas

    # Información del cliente y pago
    pdf.set_font("Arial", size=12)
    pdf.cell(200, 10, txt=f"ID Cliente: {id_cliente}", ln=True)
    pdf.cell(200, 10, txt=f"ID Membresía: {id_membresia}", ln=True)
    pdf.cell(200, 10, txt=f"Método de Pago: {metodo_pago}", ln=True)
    pdf.cell(200, 10, txt=f"Montón: ${monto:.2f}", ln=True)
    pdf.cell(200, 10, txt=f"Referencia: {referencia}", ln=True)
    pdf.cell(200, 10, txt="Estado: Pendiente", ln=True)
    pdf.cell(200, 10, txt="Transferencia Bancaria: 8901 3801 0238 0012", ln=True)
    pdf.cell(200, 10, txt="Depósito en efectivo: 09128390103020012023", ln=True)
    # Pie de página
    pdf.ln(20)
    pdf.set_font("Arial", size=10)
    pdf.cell(200, 10, txt="Gracias por su pago. Este es un recibo provisional.", ln=True, align='C')

    # Guardar PDF en un archivo temporal
    pdf_path = f"src/static/recibos/recibo_{id_cliente}_{referencia}.pdf"
    pdf.output(pdf_path)
    return pdf_path
```

Y se configuró el botón con las consultas necesarias para generar la referencia.

```
@app.route('/descargar_recibo<int:id_cliente><string:referencia>')
def descargar_recibo(id_cliente, referencia):
    # Obtener detalles del pago desde la base de datos
    db, cursor = get_cursor()
    try:
        cursor.execute("""
        SELECT p.ID_Membresia, p.Monto, p.Metodo_Pago, p.Referencia
        FROM Pagos p
        WHERE p.Referencia = %s AND p.ID_Cliente = %s
        """, (referencia, id_cliente))
        pago = cursor.fetchone()

        if not pago:
            flash("No se encontró el recibo solicitado.", "danger")
            return redirect(url_for('index_cliente'))

        # Generar el recibo PDF
        pdf_path = generar_recibo_pdf(
            id_cliente=id_cliente,
            id_membresia=pago['ID_Membresia'],
            monto=pago['Monto'],
            metodo_pago=pago['Metodo_Pago'],
            referencia=pago['Referencia']
        )
    finally:
        cursor.close()
        db.close()

    # Enviar el archivo PDF al cliente
    return send_file(pdf_path, as_attachment=True, download_name=f"recibo_{referencia}.pdf")
```

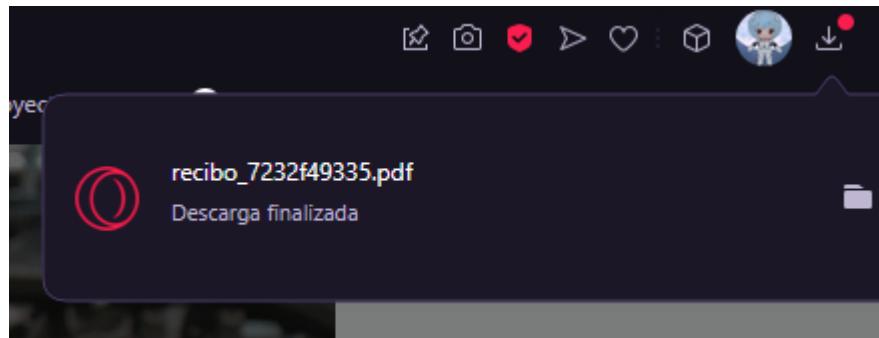
Se realizaron modificaciones mínimas en [indexclientes.html](#) y de esta manera poder mostrar el botón en la tabla.

```
<td>{{ pago.Referencia }}</td>
<td>{{ pago.Tipo }}</td>
<td>{{ pago.Fecha_Pago }}</td>
<td>${{ pago.Monto }}</td>
<td>{{ pago.Metodo_Pago }}</td>
<td>{{ pago.Estado }}</td>
<td>{{ pago.Estado_Membresia }}</td>
<td>
    <a href="{{ url_for('descargar_recibo', id_cliente=session['user_id'], referencia=pago.Referencia) }}" class="btn btn-link" title="Descargar Recibo">
        <i class="bi bi-download"></i>
    </a>
</td>
```

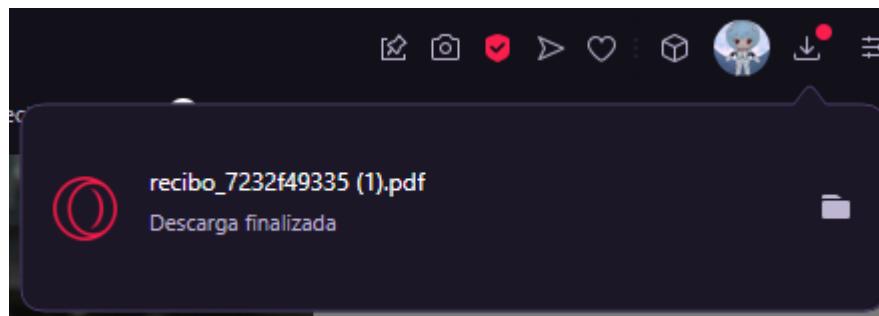
Y el resultado se muestra así:

| Historial de Pagos | | | | | | | |
|--------------------|--------------|---------------|-----------|-------------------|----------------|---------------------|---|
| Referencia | Membresia | Fecha de Pago | Monto | Método de Pago | Estado de Pago | Estado de Membresia | Descargar Recibo |
| 7232f49335 | Super Hierro | 2024-11-25 | \$4000.00 | Tarjeta de Debito | Pagado | Inactivo |  |

Prueba de función de descarga:



Descargando de nuevo:



5. Función de registro a clase con conteo real de clientes inscritos

Para hacer una relación entre la tabla de clientes y la tabla de clases, usando los siguientes comandos SQL:

```
CREATE TABLE `inscripciones` (
    `ID_Inscripcion` int(11) NOT NULL,
    `ID_Clase` int(11) NOT NULL,
    `ID_Cliente` int(11) NOT NULL,
    `Fecha_Incripcion` timestamp NOT NULL DEFAULT current_timestamp()
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

Una vez creada esa tabla, se configura la función del botón para registrarse, la cual se estructuró de esta manera:

```
@app.route('/registrarse_clase/<int:id_clase>', methods=['POST'])
def registrarse_clase(id_clase):
    if 'user_id' not in session: # Verifica si el usuario está autenticado
        flash("Por favor, inicia sesión primero.")
        return redirect(url_for('indexlogin'))

    user_id = session['user_id']

    db, cursor = get_cursor()
    try:
        # Verificar si el cliente ya está inscrito en la clase
        cursor.execute("SELECT COUNT(*) AS count FROM Inscripciones WHERE ID_Cliente = %s AND ID_Clase = %s", (user_id, id_clase))
        inscrito = cursor.fetchone()['count'] > 0

        if inscrito:
            flash("Ya estás registrado en esta clase.", "info")
            return redirect(url_for('index_cliente'))

        # Verificar capacidad de la clase
        cursor.execute("SELECT COUNT(*) FROM Inscripciones WHERE ID_Clase = %s" AS Inscritos FROM Clases WHERE ID_Clase = %s", (id_clase, id_clase))
        clase = cursor.fetchone()

        if clase and clase['Inscritos'] < clase['Capacidad']:
            # Registrar inscripción
            cursor.execute("INSERT INTO Inscripciones (ID_Clase, ID_Cliente) VALUES (%s, %s)", (id_clase, user_id))
            db.commit()
            flash("Te has registrado exitosamente en la clase.", "success")
        else:
            flash("Esta clase está llena. No es posible registrarte.", "danger")
    except Exception as e:
        db.rollback()
        flash("Ocurrió un error al intentar registrarte: " + str(e), "danger")
    finally:
        cursor.close()
        db.close()

    return redirect(url_for('index_cliente'))
```

Así mismo, el cliente también tiene la opción para salir de la clase, una vez este esté registrado.

```
@app.route('/salir_clase/<int:id_clase>', methods=['POST'])
def salir_clase(id_clase):
    if 'user_id' not in session: # Verifica si el usuario está autenticado
        flash("Por favor, inicia sesión primero.")
        return redirect(url_for('indexlogin'))

    user_id = session['user_id']

    db, cursor = get_cursor()
    try:
        # Eliminar inscripción de la tabla Inscripciones
        cursor.execute("DELETE FROM Inscripciones WHERE ID_Cliente = %s AND ID_Clase = %s", (user_id, id_clase))
        db.commit()
        flash("Has salido de la clase exitosamente.", "success")
    except Exception as e:
        db.rollback()
        flash("Ocurrió un error al intentar salir de la clase: " + str(e), "danger")
    finally:
        cursor.close()
        db.close()

    return redirect(url_for('index_cliente'))
```

También se actualizó el documento HTML [indexclientes.html](#), en la sección del modal que muestra las clases actuales de la membresía.

Contenido de la tabla:

```
<tbody>
  {% for clase in clases %}
    <tr>
      <td>{{ clase.Nombre_Clase }}</td>
      <td>
        <span
          data-bs-toggle="tooltip"
          title="{{ clase.Descripcion }}"
          style="max-width: 200px; display: inline-block; white-space: nowrap; overflow: hidden; text-overflow: ellipsis;">
          {{ clase.Descripcion }}
        
      </td>
      <td>{{ clase.Instructor }}</td>
      <td>{{ clase.Hora }}</td>
      <td>
        | {{ clase.Inscritos }}/{{ clase.Capacidad }} <!-- Mostrar inscritos/total -->
      </td>
      <td>
        {% if clase.Imagen %}
          | 
        {% else %}
          No hay imagen
        {% endif %}
      </td>
      <td>
        {% if clase.Inscrito %}
          <form method="POST" action="{{ url_for('salir_clase', id_clase=clase.ID_Clase) }}">
            | <button type="submit" class="btn btn-danger">Salir</button>
          </form>
        {% elif clase.Inscritos >= clase.Capacidad %}
          <button class="btn btn-secondary" disabled>Llena</button>
        {% else %}
          <form method="POST" action="{{ url_for('registrarse_clase', id_clase=clase.ID_Clase) }}">
            | <button type="submit" class="btn btn-primary">Registrarse</button>
          </form>
        {% endif %}
      </td>
    </tr>
  {% endfor %}
</tbody>
```

Podemos tener tres estados posibles:

Clases Disponibles

| Nombre | Descripción | Instructor | Hora | Capacidad | Imagen | Registro |
|--------------------|------------------------------|------------|----------|-----------|--------|------------------------------|
| Ejercicio Aerobico | El ejercicio aeróbazón y ... | Baki Hanma | 22:00:00 | 2/3 | | <button>Registrarse</button> |

Cerrar

Clases Disponibles

| Nombre | Descripción | Instructor | Hora | Capacidad | Imagen | Registro |
|--------------------|------------------------------|------------|----------|-----------|--------|------------------------|
| Ejercicio Aerobico | El ejercicio aeróbazón y ... | Baki Hanma | 22:00:00 | 3/3 | | <button>Salir</button> |

Cerrar

Clases Disponibles

| Nombre | Descripción | Instructor | Hora | Capacidad | Imagen | Registro |
|-----------------------------------|------------------------------|------------|----------|-----------|--------|------------------------------|
| Ejercicio Aerobico | El ejercicio aeróbazón y ... | Baki Hanma | 22:00:00 | 3/3 | | <button>Llena</button> |
| Entrenamiento de Atrofia Muscular | El entrenamiento físico p... | Alan Brito | 22:14:00 | 0/5 | | <button>Registrarse</button> |

Cerrar

4.8. Errores Comunes

Renderizar plantillas con información incompleta.

Es importante asegurarse de que todos los datos necesarios estén disponibles antes de renderizar las plantillas. Si se pasan variables incompletas o vacías al `render_template()`, esto puede causar que algunos elementos de la página no se muestren correctamente o incluso que se genere un error. Por ejemplo, si se espera que la plantilla reciba información sobre clases o pagos, pero no se han consultado correctamente, la página podría mostrar resultados incorrectos o vacíos.

Llamar mal a las funciones en los archivos HTML.

En los archivos HTML que utilizan Jinja2 para la renderización, es fundamental que las funciones y las variables pasadas desde el backend estén bien referenciadas. Llamar mal a las funciones o a las variables, por ejemplo, usando un nombre incorrecto o una sintaxis equivocada (`{{ funcion() }}` o `{{ variable }}`), generará errores o resultará en la visualización de datos vacíos o incorrectos.

Guardar mal las referencias de imágenes en los registros de las tablas de las bases de datos.

Al subir imágenes, es necesario asegurarse de que las rutas de los archivos se guardan correctamente en las tablas de la base de datos. Usualmente, las imágenes se almacenan en un directorio específico del servidor y las rutas relativas se almacenan en la base de datos. Si se guarda una ruta incorrecta o se olvida agregar la extensión o prefijo de la carpeta, esto causará que las imágenes no se puedan cargar o no se muestren correctamente en las páginas del cliente. Asegúrate también de verificar que las imágenes se suben correctamente y que las funciones para manejar archivos (como `secure_filename()` y `allowed_file()`) estén bien implementadas.

5. Resultados

Estos resultados reflejan el rendimiento y la funcionalidad del sistema en relación con las expectativas planteadas en las fases previas del desarrollo, incluyendo la interacción con la base

de datos, la administración de usuarios, y la presentación de información dinámica a través de la interfaz web.

5.1 Funcionalidades Implementadas

- Registro de Clientes:** El sistema permite a los nuevos clientes registrarse proporcionando su información personal, como nombre, apellido, correo electrónico, y datos adicionales como teléfono y dirección. También ofrece la opción de cargar una imagen, la cual se guarda en el servidor y se almacena como una ruta relativa en la base de datos. El registro se completa exitosamente y se confirma con un mensaje de éxito.

Visualización de función:

Registro de Cliente X

Nombre

Apellido

Imagen de perfil
 Seleccionar archivo yo.jpg

Fecha de Nacimiento
 □

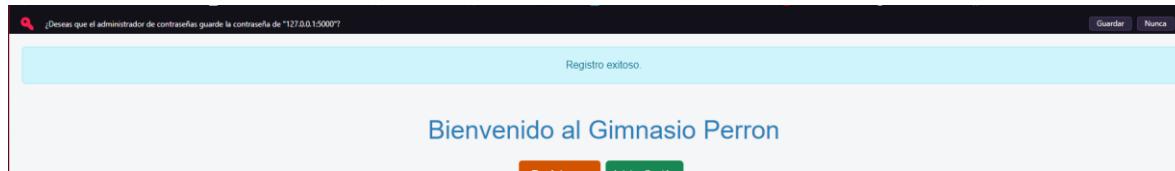
Teléfono

Email

Contraseña

Dirección

Registrarse



Agregado en registro 24

| | <input type="checkbox"/> | <input type="checkbox"/> Edit | <input type="checkbox"/> Copy | <input type="checkbox"/> Delete | ID_Cliente | Nombre | Apellido | Fecha_Nacimiento | Telefono | Email | password | Direccion | Fecha_Registro | Imagen |
|--|--------------------------|-------------------------------|-------------------------------|---------------------------------|------------|------------|-----------------|------------------|-------------|----------------------|---|-------------------------|----------------|----------------------|
| | <input type="checkbox"/> | <input type="checkbox"/> Edit | <input type="checkbox"/> Copy | <input type="checkbox"/> Delete | 17 | Fernando | Azuara Ibarra | 2005-03-06 | 7122411716 | ferazuliba@gmail.com | scrypt32768:8:1\$6DlBJ1VK3tmPPVp4\$lf6dba501e1d45843... | San Felipe del Progreso | 2024-11-24 | clientes/fer.jpg |
| | <input type="checkbox"/> | <input type="checkbox"/> Edit | <input type="checkbox"/> Copy | <input type="checkbox"/> Delete | 18 | Jiovani | Tapia López | 2005-02-03 | 919238912 | Asael@gmail.com | scrypt32768:8:1\$OiaOMGY3SQRzY-B43448ae89f7d6b21c... | Santiago Casandeje | 2024-11-24 | clientes/asael.jpg |
| | <input type="checkbox"/> | <input type="checkbox"/> Edit | <input type="checkbox"/> Copy | <input type="checkbox"/> Delete | 19 | Omar Aldei | García López | 2005-02-09 | 9812893123 | omar@gmail.com | scrypt32768:8:1\$oQtdj3utDThk6J5abaa9e2956b043... | Atiacomulco | 2024-11-24 | clientes/omar.jpg |
| | <input type="checkbox"/> | <input type="checkbox"/> Edit | <input type="checkbox"/> Copy | <input type="checkbox"/> Delete | 20 | Jenifer | Ramirez Ibarra | 2005-02-02 | 901203921 | jeni@gmail.com | scrypt32768:8:1\$V6rA3XbVeI21JIV4\$9d13aac492358dc... | Atiacomulco | 2024-11-24 | clientes/jeni.jpg |
| | <input type="checkbox"/> | <input type="checkbox"/> Edit | <input type="checkbox"/> Copy | <input type="checkbox"/> Delete | 21 | Oswaldo | Plata Navarrete | 2005-09-09 | 912893212 | oswaldo@gmail.com | scrypt32768:8:1\$V6hd5wKCfzavSWN571987c274559efc8... | Atiacomulco | 2024-11-24 | clientes/oswaldo.jpg |
| | <input type="checkbox"/> | <input type="checkbox"/> Edit | <input type="checkbox"/> Copy | <input type="checkbox"/> Delete | 22 | Ricardo | Anaya | 2024-11-06 | 97983217983 | ricardo@gmail.com | scrypt32768:8:1\$UxkaQlIxUnvQd4-9f5eD16cd8cf... | El Cielo | 2024-11-24 | clientes/ricardo.jpg |
| | <input type="checkbox"/> | <input type="checkbox"/> Edit | <input type="checkbox"/> Copy | <input type="checkbox"/> Delete | 23 | Rambo | Rambosio | 2002-01-01 | 17412312 | rambo@gmail.com | scrypt32768:8:1\$Qq3ylz3drWtUsd7\$c3bd0acd8f576a83... | Ucrania | 2024-11-24 | clientes/rambo.jpg |
| | <input type="checkbox"/> | <input type="checkbox"/> Edit | <input type="checkbox"/> Copy | <input type="checkbox"/> Delete | 24 | Gigachad | Apellido | 2024-10-30 | 1231 | giga@gmail.com | scrypt32768:8:1\$IXwrS8JaKDpoKB7R\$db1f171c2d7ecf0... | Cielo | 2024-11-25 | clientes/giga.jpg |

2. **Inicio de Sesión:** Los usuarios pueden iniciar sesión usando su correo electrónico y contraseña. La autenticación se maneja de forma segura utilizando el hash de contraseñas. Al iniciar sesión, se redirige a la página principal donde los clientes pueden ver clases, pagos y su historial de membresías.

Visualización de función:

Inicio de Sesión

Email
giga@gmail.com

Contraseña

Iniciar Sesión



3. Visualización de Clases y Membresías: Los usuarios pueden visualizar las clases disponibles y las opciones de membresía. La información se obtiene dinámicamente desde la base de datos y se presenta en la interfaz de usuario de manera clara y organizada.

Visualización de función:

| Nombre | Descripción | Instructor | Hora | Capacidad | Imagen |
|-----------------------------------|---|-------------------|----------|-------------|--------|
| Ejercicio Aeróbico | El ejercicio aeróbico fortalece el corazón y promueve niveles de colesterol saludables. Entre los aeróbicos de bajo impacto están el caminar y nadar. Correr, jugar tenis y bailar son aeróbicos de alto impacto. | Alma Marcela Gozo | 22:00:00 | 20 personas | |
| Entrenamiento de Fuerza | Es una rutina deportiva que busca fortalecer los músculos del cuerpo. Para ello, se utilizan distintos métodos de resistencia, como pesas, máquinas, bandas elásticas o el propio peso del cuerpo. | Alan Brito | 23:00:00 | 10 personas | |
| Entrenamiento de Atrofia Muscular | El entrenamiento físico puede ayudar a tratar la atrofia muscular, que es la disminución de la masa muscular y el desgaste de los tejidos musculares. | Samuel del Luque | 1:00:00 | 5 personas | |
| Entrenamiento de Espalda | La mejor manera de entrenar los músculos de la espalda es a través de levantamientos compuestos (multiarticulares) que implican levantar pesos pesados. Estos pueden ser el levantamiento de peso muerto, las cargadas, las arrancadas y los tirones. | Baki Hanma | 17:40:00 | 10 personas | |

| Tipo | Descripción | Duración | Costo | Imagen | Acción |
|--------------|---|----------|-----------|--------|-----------------------|
| Carbón | Acceso a las instalaciones tiempo completo. | 30 días | \$400.00 | | Pagar |
| Hierro | Carbon + Entrenamientos con proteína incluida | 30 días | \$800.00 | | Pagar |
| Oro | Hierro + Instructor personal | 30 días | \$1200.00 | | Pagar |
| Diamante | Oro + Nutriólogo Incluido | 30 días | \$1600.00 | | Pagar |
| Super Carbón | Semestre de beneficios carbón | 180 días | \$2000.00 | | Pagar |
| Super Hierro | Semestre de beneficios Hierro | 180 días | \$4000.00 | | Pagar |
| Super Oro | Semestre de beneficios oro | 180 días | \$6000.00 | | Pagar |

4. **Historial de Pagos:** Los clientes tienen acceso a un historial de sus pagos, mostrando detalles como el monto pagado, la fecha, y el tipo de membresía asociada. Esta funcionalidad permite a los usuarios tener un seguimiento completo de sus transacciones.

Visualización de función:

The screenshot shows a mobile application for a gym. At the top, there's a navigation bar with icons for home, classes, and profile. Below it, a banner for 'Blog del Gimnasio' features three images: a woman lifting weights, a man preparing food, and a person with arms raised at sunset. To the right of the banner is a modal window titled 'Historial de Pagos' containing a table of transaction details:

| ID Pago | Membresía | Fecha de Pago | Monto | Método de Pago |
|---------|-----------|---------------|-----------|----------------|
| 11 | Carbón | 2024-11-25 | \$400.00 | Efectivo |
| 12 | Hierro | 2024-11-25 | \$800.00 | Electrónico |
| 13 | Oro | 2024-11-25 | \$1200.00 | Electrónico |

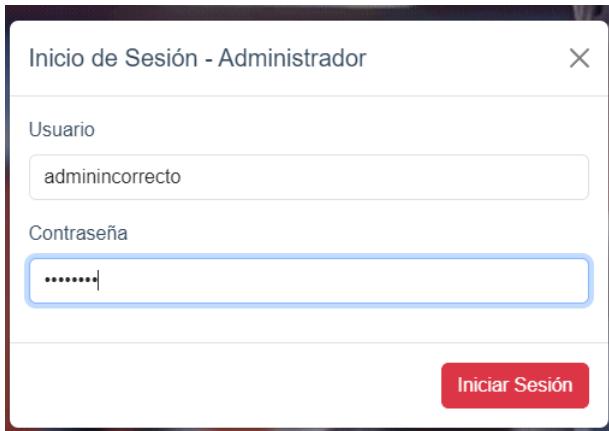
Below the table are three cards with training tips:

- Cómo estructurar un entrenamiento de fuerza**: A woman lifting weights. Text: "Si deseas ganar fuerza y músculo, es esencial que sepas cómo estructurar un entrenamiento adecuado. A continuación, te presentamos una estructura básica para tu rutina de fuerza."
 - Ejercicios compuestos: Los ejercicios como sentadillas, press de banca y peso muerto deben ser la base de tu rutina.
 - Repeticiones y series: Realiza entre 3-5 series de 8-12 repeticiones con un peso que te desafie.
- Comidas pre-entrenamiento para mejorar tu rendimiento**: A man preparing food. Text: "Tu rendimiento en el gimnasio está estrechamente relacionado con lo que comes antes de entrenar. Aquí te damos algunas sugerencias de alimentos que te ayudarán a maximizar tu energía."
 - Carbohidratos complejos: Alimentos como avena, batatas o arroz integral proporcionan energía sostenida durante el entrenamiento.
- Cómo mantenerte motivado en tu rutina**: A person with arms raised at sunset. Text: "La motivación puede fluctuar, pero existen estrategias que te pueden ayudar a mantener el enfoque. Aquí algunos consejos."
 - Establece metas pequeñas: Los objetivos alcanzables te mantienen motivado y te dan una sensación de logro constante.
 - Encuentra un compañero de entrenamiento: Tener alguien con quien entrenar te ayudará a mantener el compromiso.
 - Varia tus rutinas: Cambiar tu rutina de vez en cuando te desafiará.

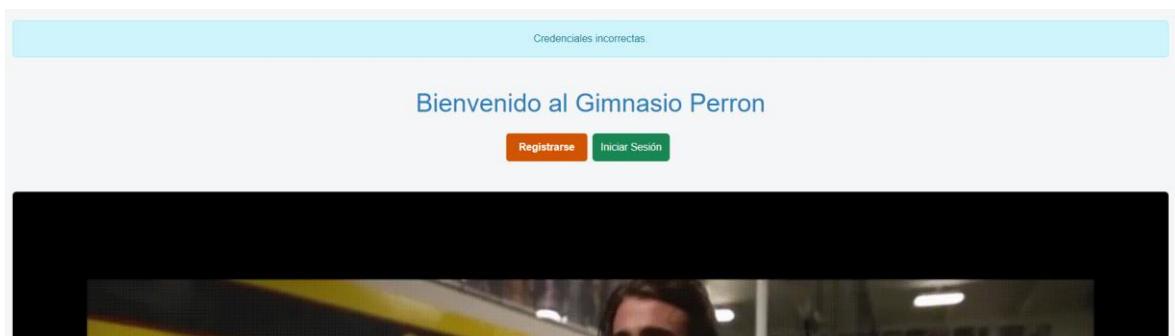
5. **Manejo de Errores:** Durante las pruebas, se identificaron y corrigieron varios errores comunes relacionados con la renderización de plantillas con datos incompletos, el manejo de funciones en HTML, y la correcta referencia de las imágenes en la base de datos. Estos problemas se resolvieron implementando verificaciones y control de errores adecuados.

Visualización de función:

Inicio de sesión con credenciales de administrador incorrectas



Mensaje que indica que las credenciales son incorrectas:



5.2 Resultados del Sistema

- Usabilidad:** El sistema proporciona una interfaz amigable y funcional para los usuarios, lo que facilita la interacción tanto para administradores como para clientes. La carga dinámica de información y las respuestas rápidas a las acciones del usuario (como iniciar sesión o visualizar clases) contribuyen a una experiencia de usuario fluida.
- Seguridad:** Las contraseñas de los usuarios se almacenan de manera segura utilizando el hashing, lo que garantiza que los datos sensibles estén protegidos. Además, la validación de los archivos cargados (como las imágenes de los usuarios) asegura que solo se acepten tipos de archivo permitidos, evitando problemas de seguridad.

Encriptación de contraseñas en base de datos:

password

m scrypt:32768:8:1\$6DtBJ1VK3tmPPVp4\$f6dba501e1d45843...

scrypt:32768:8:1\$OiaOMGY3SQRt2yB4\$3448ae89f7d6b21c...

scrypt:32768:8:1\$oQtodj3utDIThk6J\$abaaf9e2956b043f...

scrypt:32768:8:1\$V6rA3XbVei21JIV4\$9d13aaac492358dc...

1 scrypt:32768:8:1\$rV6hd5wKCfzavSWN\$71987c274559efc8...

scrypt:32768:8:1\$UlxakQIfIxaUnvQd\$4c9f5ef316cd8c9f...

scrypt:32768:8:1\$Qq3ylz3drWtUSd7I\$c3bd0acd8f576a83...

scrypt:32768:8:1\$IXwrS8JaKDPoKB7R\$8db1f171c2d76cf0...

3. **Desempeño:** Las consultas a la base de datos se realizan de manera eficiente y no se observa un retraso significativo en la carga de información. La implementación de un sistema de sesiones asegura que los datos del usuario sean persistentes durante su navegación, mejorando la fluidez del sistema.

Visualización de registro de clientes en tiempo real:

Gestión de Clientes - Gimnasio

| ID | Imagen | Nombre | Apellido | Fecha Nacimiento | Teléfono | Email | Dirección | Acciones |
|----|--------|---------------|---------------|------------------|------------|----------------------|-------------------------|---|
| 17 | | Fernando | Azuara Ibarra | 2005-03-06 | 7122411716 | ferazuliba@gmail.com | San Felipe del Progreso | <button>Eliminar</button> <button>Editar</button> |
| 18 | | Jiovani Asael | Tapia López | 2005-02-03 | 919238912 | Asael@gmail.com | Santiago Casandeje | <button>Eliminar</button> <button>Editar</button> |
| 19 | | Omar Aldrei | García López | 2005-02-09 | 9812893123 | omar@gmail.com | Atiacomulco | <button>Eliminar</button> <button>Editar</button> |

6. Link del repositorio del proyecto en GitHub. <https://github.com/igream/p2-flask-SQL>

7. Link de PythonAnywhere: <https://ferazui.pythonanywhere.com>
8. Link del video de demostración: <https://youtu.be/ydMRoXp8Vdc>

9. Conclusiones

Se obtuvieron los resultados esperados al desarrollar un sistema de gestión integral para un gimnasio, que cumplió con los objetivos establecidos al proporcionar una solución eficiente tanto para administradores como clientes. El sistema facilitó la gestión de la información de clientes, pagos, clases y membresías, optimizando la experiencia de usuario mediante una interfaz intuitiva y accesible. La implementación de funciones de autenticación y seguridad, como el cifrado de contraseñas y la validación de archivos, aseguró la protección de los datos sensibles, minimizando los riesgos asociados a la vulnerabilidad de la plataforma. Además, se resolvieron y previnieron errores comunes que podrían haber afectado el rendimiento del sistema, como la renderización de plantillas con información incompleta, el manejo incorrecto de funciones en archivos HTML y el almacenamiento inadecuado de referencias a imágenes en la base de datos. Estos problemas fueron identificados durante las pruebas del sistema y corregidos para mejorar la estabilidad y fiabilidad general. El uso de tecnologías como Flask, MySQL, y Bootstrap permitió que la aplicación fuera escalable, modular y fácil de mantener. A través de un enfoque ágil y detallado en el diseño y desarrollo del proyecto, se estableció una base sólida que no solo cumple con los requisitos iniciales, sino que también permite la incorporación de futuras mejoras, como la optimización de consultas, el refinamiento de la experiencia de usuario y la expansión de nuevas funcionalidades. En general, el proyecto demostró ser una solución efectiva y robusta, con un alto nivel de personalización, que aporta valor tanto a los usuarios del gimnasio como a los administradores del sistema.

10. Bibliografía

- *MVC - Glosario de MDN Web Docs: Definiciones de términos relacionados con la Web | MDN.* (2023, November 13). MDN Web Docs. <https://developer.mozilla.org/es/docs/Glossary/MVC>
- *Welcome to Flask — Flask Documentation (3.1.x).* (n.d.). <https://flask.palletsprojects.com/en/stable/>
- Contributors, M. O. J. T. a. B. (n.d.). *Introduction.* <https://getbootstrap.com/docs/5.0/getting-started/introduction/>
- *3.13.0 documentation.* (n.d.). <https://docs.python.org/3/>
- *MySQL :: MySQL Documentation.* (n.d.). <https://dev.mysql.com/doc/>
- *HTML5 - MDN Web Docs Glossary: Definitions of Web-related terms MDN.* (2024, July 25). MDN Web Docs. <https://developer.mozilla.org/en-US/docs/Glossary/HTML5>
- *Apache friends.* (n.d.). <https://www.apachefriends.org/docs/>
- *GitHub.com help documentation.* (n.d.). GitHub Docs. <https://docs.github.com/en>

